UNIVERSITY OF WATERLOO
COMPUTER SCIENCE DEPARTMENT

Transforming from Flat Algebra to Nested Algebra

V. Deshpande
P. Å. Larson

Research Report
CS-89-19

May, 1989

Transforming from Flat Algebra to Nested Algebra

*V. Deshpande and P.-Å. Larson**
Department of Computer Science
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1
Report CS-89-19

Abstract

Consider a database which at the conceptual level consists of a set of 1NF (flat) relations but which is physically stored in denormalized (prejoined) form. Denormalization reduces the number of joins required to compute a query. To reduce redundancy, each prejoined relation can be converted into a NF^2 (nested) relation by a series of nest operations.

In this paper, we consider the problem of converting from flat algebra to nested algebra. We provide a series of legal transformations that can be applied to a query Q_e on flat relations, to obtain an equivalent query Q_r on the nested relations. Several fundamental theorems are proved, which allow the selection conditions of the 1NF case to be applied directly to the NF^2 relations. This is done through changing the nesting structure internally, without unnesting, and pushing the selection conditions down to the appropriate subrelations.

1 Introduction

The idea of storing a relation in prejoined form was introduced by [SS80] and [SS81]. If the query types and relative frequencies of queries are known, then precomputing joins may result in considerable savings at query-execution time. Prejoined relations are a special case of stored *derived* relations. [LY87] have already solved the problem of taking a project-select-join (PSJ) query Q posed on conceptual (flat) relations, and transforming it into an equivalent

*This research was supported by Cognos, Inc., Ottawa and by the Natural Sciences and Engineering Research Council of Canada under grant No. A-2460.
Authors' e-mail addresses: {vdeshpande, palarson}@waterloo.{edu,cdn,csnet}

query Q_e posed on derived relations. The conceptual database consists of a number of flat relations $A_1, A_2, \ldots, A_m$. The derived relations also consist of flat relations $E_1, E_2, \ldots, E_n$. Notice that there is not a one-to-one correspondence between the conceptual and derived relations. Having derived relations provides a more flexible approach where the choice of derived relations is guided by the (actual or anticipated) query load, resulting in fewer file accesses.

We now propose to store the derived relations as *nested* relations. [SS86] proposed nested relations as storage structures for flat relations. In [S86] it is shown that any conjunctive PSJ query on a conceptual 4NF schema can be efficiently optimized and transformed to a query on nested relations. They show that using nested relations internally does not result in more complexity to the join elimination process.

We assume that there is a one-to-one correspondence between each derived relation E_i and a stored nested relation R_i $(1 \leq i \leq n)$. The aim in nesting is twofold: reduce redundancy, and capture the semantic connections among the attributes [RK87]. Capturing the semantic connections is what reduces the joins at query-execution time. It is now required to transform the query further from Q_e to an equivalent query Q_r posed on nested relations. The algebra that Q_r is expressed in is the algebra for nested relations described in [DL87]. Naturally, for a given Q_e, the corresponding Q_r can have many equivalent expressions. We give an algorithm which generates one such Q_r. It does not necessarily find the optimal expression, but does eliminate inefficient expressions for Q_r.

There are many ways of nesting a flat relation. A normal form for nested relations, called *nested normal form*, is discussed in [RK87]. We will not discuss the criteria for obtaining the nested relations $R_1, \ldots, R_n$, except to say that it is reasonable to insist that a nested relation be in *Partitioned Normal Form* (PNF). A relation R is in PNF if the single-valued attributes of R form a key for the relation, and recursively, each relation-valued attribute of R is also in PNF. A formal definition is given in [RKS84]. Any relation that is in nested normal form is also in PNF. The theorems presented in this paper in Section 5 will hold for any nested relation, even if it is not in PNF.

In this paper, we describe a procedure for transforming a query Q_e posed on the flat derived relations into an equivalent query Q_r posed on the nested relations. Six theorems are presented which enable us to define legal steps in the transformation process. The procedure described always selects a legal transformation step which will minimize the number of tuples remaining at that step of query execution.

The paper is organized as follows. In section 2 we define a number of basic concepts and introduce some notation. Section 3 gives a brief description of the algebra for nested relations. Section 4 formalizes the problem. A start

is made in this section towards transforming the query. Section 5 contains a number of theorems which are used as the basis for formulating legal steps in the transformation procedure described in Section 6. Section 7 contains some concluding remarks.

2 Basic Concepts

Before looking at the algebra for nested relations and defining the problem of transforming the query, we define some basic concepts and introduce the needed notation.

In a nested relation, a relation-valued attribute is also called a *subrelation.* An attribute which is atomic-valued is called a *single-valued* attribute. We give the following recursive definition.

Definition 2.1 [Nested Relation] The scheme of a *nested relation* is of the form $R(A_1, A_2, \ldots, A_n)$, $n \geq 1$, where R is the name of the scheme and each A_i is either the name of a single-valued attribute (defined over some domain) or the scheme of a nested (sub)relation. □

Let R be the name of a (sub)relation. Then *attr*(R) denotes the set of (single-valued and relation-valued) attribute names of R, *sattr*(R) the set of single-valued attribute names of R, and *rattr*(R) the set of relation-valued attribute names of R. An *instance* r of a relation R consists of *tuples* over the scheme R. For example, in the EMPLOYEE relation given below,

EMPLOYEE(ENO,NAME,CHILDREN(NAME,DOB,SEX),
TRAINING(CNO,DATE))

the attributes are ENO, NAME, CHILDREN, and TRAINING. The single-valued attributes are ENO and NAME, and the relation-valued attributes are CHILDREN and TRAINING. If $Z \subseteq$ attr(R) and t is a tuple over R, we write $t[Z]$ to denote the projection of t onto the attributes in Z.

To define the structure of a relation we introduce the concept of the *format* of a relation, and define when two formats are equivalent.

Definition 2.2 [Format] Let X be a single-valued attribute, and $R(Z_1, \ldots, Z_n)$ a nested relation scheme where R is the name of the relation and $Z_1, \ldots, Z_n$ are attribute names. Then

$$\begin{aligned} \text{format}(X) &= \text{':'domain}(X) \\ \text{format}(R(Z_1, \ldots, Z_n)) &= \text{'('}Z_1\text{format}(Z_1), \ldots, Z_n\text{format}(Z_n)\text{')'} \end{aligned}$$

□

Definition 2.3 [Equivalence of Formats] Let R_1, R_2 be two nested relations. The formats of R_1 and R_2 are equivalent, written $format(R_1) = format(R_2)$, if

$$attr(R_1) = attr(R_2) \wedge \forall A_i \in attr(R_1), B_j \in attr(R_2),$$
$$(A_i = B_j \Rightarrow format(A_i) = format(B_j))$$

□

Note that according to the above definition, the ordering of attributes within a relation is unimportant in determining the equivalence of relation structures. Attributes are identified by their names, not their positions.

The structure of a nested relation R can also be represented by a tree, T_R, called the *scheme diagram*. This is a convenient way of graphically illustrating the nesting structure of the relation. T_R is a tree with R as the root node and the relation-valued attributes of R as subtrees. The nodes of the scheme diagram are labelled with the names of the corresponding (sub)relations. As a convention, the attributes of the (sub)relation are listed to the right of a node.

Pathnames are used for referring to subrelations which are nested in the relation structure. We give the following definition.

Definition 2.4 [Pathname] Let R be a nested relation. An expression of the form $Y_1.Y_2.\cdots.Y_k$ is a valid pathname in R if $Y_1 \in rattr(R)$ and $Y_i \in rattr(Y_{i-1})$, $i = 2, 3, \ldots, k$. □

A pathname uniquely identifies a subrelation within R. Note that the relation name R is not included in the pathname. The *complete name* of an attribute Z in a subrelation Y_j of a relation R, is then of the form $Y_1.Y_2.\cdots.Y_j.Z$ where $Y_1.Y_2.\cdots.Y_j$ is the pathname identifying Y_j. By using complete names we can always uniquely identify an attribute within a relation. For example, in the EMPLOYEE relation given above, CHILDREN.NAME refers to the attribute NAME of the subrelation CHILDREN, while NAME refers to the attribute NAME of relation EMPLOYEE.

Next we define the *scope* of a pathname. The scope consists of the set of attributes (names) which can be referred to when the pathname is specified. Intuitively, the scope consists of all the attributes which are 'seen' as one goes along the path starting at the root of the scheme diagram of R and going down to the subrelation identified by the pathname.

Definition 2.5 [Scope of a Pathname] Let $S = \{R_1, R_2, \ldots, R_m\}$ be a database scheme and $P = Y_1.Y_2.\cdots.Y_k$ a valid pathname in relation R_j.

Then the scope of P is defined as

$$scope(P) = S \cup attr(R_j) \cup attr(Y_1) \cup attr(Y_1.Y_2) \cup \ldots \cup attr(Y_1.Y_2.\cdots.Y_k) - \{R_j, Y_1, Y_1.Y_2, \ldots, Y_1.Y_2.\cdots.Y_k\}$$

The immediate scope of P refers to the set of attributes that are seen at the lowest level of the path, that is, $iscope(P) = attr(Y_1.\cdots.Y_k)$. □

For example, consider a database containing a single relation that has the scheme R(A,B,C(D,E), H(I,J(K,L))), and consider the path P=H.J. Then

$$\begin{aligned} \text{scope(H.J)} &= \{\text{R}\} \cup \text{attr(R)} \cup \text{attr(H)} \cup \text{attr(H.J)} - \{\text{R,H,H.J}\} \\ &= \{\text{R}\} \cup \{\text{A,B,C,H}\} \cup \{\text{H.I,H.J}\} \cup \{\text{H.J.K,H.J.L}\} - \{\text{R,H,H.J}\} \\ &= \{\text{A,B,C,H.I,H.J.K,H.J.L}\} \\ \text{iscope(H.J)} &= \text{attr(H.J)} \\ &= \{\text{H.J.K,H.J.L}\} \end{aligned}$$

At the instance level, each occurrence of J consists of a set of tuples (possibly empty) and everything above it along the path will have fixed values which can be referred to as needed. Note that it is not possible to (directly) refer to the attributes D and E, which are inside C, since they are not within the scope of P.

3 An Algebra for Nested Relations

In this section, we present briefly a portion of the algebra used for nested relations. A detailed presentation of the algebra is found in [DL87]. It is a recursive algebra which handles null values. It has a powerful operator called the *subrelation constructor*, used for doing operations on subrelations at any level. We give below only the definitions of selection, Cartesian product, nest, unnest, and the subrelation constructor.

3.1 Selection

The selection predicate is allowed to contain relational algebra expressions over the relation-valued attributes of the relation, and set comparisons are allowed. The syntax of the selection operator is $\sigma[F](R)$ where R is a relation expression and F is a selection predicate over attributes in $scope(R)$. Note that although the selection predicate may be testing the tuples in a subrelation of R, *entire* tuples of R are selected. Selection is a format preserving operator, that is, the format of the result relation is the same as the format of the operand relation, R.

The selection predicate F is a logical combination of atomic selection conditions. We allow the standard boolean operators: *AND* ($\wedge$), *OR* ($\vee$) and *NOT* ($\neg$).

Definition 3.1 [Atomic selection condition] An atomic selection condition has one of two forms: Z_1 *op* C or Z_1 *op* Z_2 where *op* $\in \{<, \leq, >, \geq, =, \neq, \subset, \subseteq, \supset, \supseteq\}$, C is either a single-valued or relation-valued constant, and Z_1, Z_2 are one of the following:

- the name of a single-valued or relation-valued attribute within the scope of the operand relation of the selection operation.
- a relational algebra expression operating on relation-valued attributes within the scope of the operand relation of the select statement.

All scalar comparisons must be made between comparable domains. In set comparisons, the two operands must have equivalent formats. □

The formal definition of the effect of the select operator at the instance level can now be given. Consider the selection statement

$$\sigma[F(Z)](R)$$

where $Z \subseteq$ *scope*(R) and F is a selection predicate over the attributes Z.

$$\begin{aligned} \sigma[F](r) &= \{s \mid \exists t \in r : s = t \wedge F(t[Z]) = true\} \\ \text{format}(\sigma[F(Z)](R)) &= \text{format}(R) \end{aligned}$$

When constructing selection conditions involving algebra expressions, the scoping rules of the operators must be strictly observed. Consider the relation R(A,B,C(D,E,F(G)),H(I,J)) and the selection

$$\sigma[\sigma[\text{A} > \text{D}](\text{C}) \neq \emptyset](\text{R})$$

Is the comparison A>D allowed? The answer is provided by the scoping rules of the selection operator: the selection predicate must be over attributes in the scope of the operand relation. The comparison is valid because C is in *scope*(R), and A and D are in *scope*(C). However, the scalar comparisons in

$$\sigma[\sigma[(\text{D} < \text{G}) \wedge (\text{B} = \text{I})](\text{C}) \neq \emptyset](\text{R})$$

are not allowed because G and I are not in *scope*(C).

3.2 Cartesian Product

Cartesian product is trivially extended to tuples of nested relations. The syntax of the Cartesian product operator is

$$R_1 \times R_2$$

For the Cartesian product to be defined, we must have $\text{attr}(R_1) \cap \text{attr}(R_2) = \emptyset$. Let $\text{attr}(R_1) = (A_1, \ldots, A_m)$, and $\text{attr}(R_2) = (B_1, \ldots, B_n)$. The effect on the instance level is defined as follows:

$$\begin{aligned} r_1 \times r_2 &= \{s \mid \exists t_1 \in r_1, \exists t_2 \in r_2 : s[\text{attr}(R_1)] = t_1 \\ &\qquad \wedge s[\text{attr}(R_2)] = t_2\} \\ \text{format}(R_1 \times R_2) &= (A_1\text{format}(A_1), \ldots, A_m\text{format}(A_m), \\ &\qquad B_1\text{format}(B_1), \ldots, B_n\text{format}(B_n)) \end{aligned}$$

3.3 Nest

The nest operator creates a new subrelation and thus changes the structure of a relation. It was first introduced by Jaeschke and Schek in [JS82]. The syntax is

$$\nu[Z' = (Z)](R)$$

where R is a relational algebra expression, $Z \subset \text{attr}(R)$ is the list of attributes over which to nest, and Z' is the new name given to the the subrelation composed of the nested attributes. Nesting is not allowed when the attributes $R - Z$ consist of only null values. Let $V = \{V_1, \ldots, V_m\}$, $Z = \{Z_1, \ldots, Z_n\}$, $m, n \geq 1$; let $\text{attr}(R) = \{V, Z\}$. It is required that $\text{sattr}(V) \neq \emptyset$, $\text{sattr}(Z) \neq \emptyset$, and $t[V]$ does not consist of only nulls.

$$\begin{aligned} \nu[Z' = (Z)](r) &= \{s \mid \exists t \in r : s[V] = t[V] \\ &\qquad \wedge s[Z'] = \{u[Z] \mid \exists u \in r : u[V] = s[V]\}\} \\ \text{format}(\nu[Z' = (Z)](R)) &= (V_1\text{format}(V_1), \ldots, V_m\text{format}(V_m), \\ &\qquad Z'(Z_1\text{format}(Z_1), \ldots, Z_n\text{format}(Z_n))) \end{aligned}$$

3.4 Unnest

Unnest is the inverse of nest. It was also introduced by Jaeschke and Schek in [JS82]. The syntax is

$$\mu[Y](R)$$

where $Y \in \text{attr}(R)$ is a relation-valued attribute. For this operator to be defined, we must have $\text{attr}(Y) \cap (\text{attr}(R)\text{-}\{Y\}) = \emptyset$. The subrelation specified by Y is unnested. Note that if the subrelation Y consists of the empty

set, it is unnested and filled with *dne* nulls. Let $\text{attr}(Y) = \{Y_1, \ldots, Y_n\}$, and $\text{attr}(R) = \{Z_1, \ldots, Z_m, Y\}$.

$$\begin{aligned}
\mu[Y](r) &= \{s \mid (\exists t \in r : t[Y] \neq \emptyset \ \wedge \ s[\text{attr}(R) - Y] = t[\text{attr}(R) - Y] \\
&\quad \wedge s[\text{attr}(Y)] \in t[Y]) \vee \\
&\quad (\exists t \in r : t[Y] = \emptyset \ \wedge \ s[\text{attr}(R) - Y] = t[\text{attr}(R) - Y] \\
&\quad \wedge s[Y_1] = \text{dne} \wedge \ldots \wedge s[Y_n] = \text{dne})\} \\
\text{format}(\mu[Y](R)) &= (Z_1\text{format}(Z_1), \ldots, Z_m\text{format}(Z_m), Y_1\text{format}(Y_1), \\
&\quad \ldots, Y_n\text{format}(Y_n))
\end{aligned}$$

3.5 The Subrelation Constructor

This new operator provides us with the capability of modifying the interior of a nested relation. As an introduction, consider the following query against a University database. The database has the following nested relations:
OFFERINGS(CNO,TERM,ENROLLMENT(SNO,GRADE))
STUDENT(SNO,NAME,FACULTY).

Example 3.1 For each course offered, list the student number, student name and grade of all students who received a grade of 85 or above. □

To answer this query we must modify the subrelation ENROLLMENT by selecting tuples where GRADE $\geq$ 85 and then joining with the STUDENT relation to obtain the student name. Using our new subrelation constructor this query can be expressed as follows:

$$\begin{aligned}
\text{R} \ := \ &\{(\text{CNO,TERM,SCHOLARS}); \ \text{SCHOLARS} := \sigma[\text{GRADE} \geq 85] \\
&(\text{ENROLLMENT}) \bowtie \pi[\text{SNO,NAME}](\text{STUDENT})\}(\text{OFFERINGS})
\end{aligned}$$

This expression is interpreted in the following way. For each tuple (at the root level) of OFFERINGS, construct a new tuple which consists of CNO, TERM and a new subrelation SCHOLARS. (The subrelation ENROLLMENT disappears.) SCHOLARS is constructed from the tuple's ENROLLMENT relation and the (external) relation STUDENT, as specified by the given algebra expression.

We define the syntax and effect of the subrelation constructor for the case when only one new subrelation is constructed. The syntax is

$$\{P(A_1, \ldots, A_k); A_{i_1} := E\}(R)$$

where P is a pathname in relation R, $A_1, \ldots, A_k$ are attribute names in *iscope*(P), and E is a relational algebra expression specifying how A_{i_1} is to

be computed. All attributes $A_1 \ldots A_k$ retain their old values except A_{i_1}. Note that A_{i_1} can be an entirely new attribute. The expression E can only operate on relations in *scope*(P). In looking at the effect of the subrelation constructor, there are two cases to consider:

1. P is empty. We consider the constructor

$$\{(Z, W); W := E(V_0, S)\}(R)$$

where $Z = (Z_1, \ldots, Z_n) \subseteq \text{attr}(R)$ are the attributes to be preserved; W is the new subrelation to be computed; $V_0 \subseteq \text{rattr}(R)$ are relation-valued attributes of R; and S is some relation (or set of relations) in the database. The effect of this operation is then

$$\{(Z, W); W := E(V_0, S)\}(r) =$$
$$\{u | \exists t \in r : u[Z] = t[Z] \wedge u[W] = E(t[V_0], s)\}$$
$$\text{format}(\{(Z, W);\ W := E(V_0, S)\}(R)) =$$
$$(Z_1\text{format}(Z_1), \ldots, Z_n\text{format}(Z_n), W\text{format}(E(V_0, S)))$$

Note that $t[V_0]$ represents a set of relation instances, and s a relation instance.

2. P is not empty, $P = Y_1.Y_2.\cdots.Y_k$. The constructing expression may now operate on any relation-valued attributes in scope(P). We consider the constructor

$$\{Y_1.Y_2.\cdots.Y_k(Z, W);\ W := E(V_0, V_1, \ldots, V_k, S)\}(R)$$

where $Z \subseteq \text{iscope}(P)$ are the attributes to be preserved; W is the new subrelation to be computed; $V_0 \subseteq \text{rattr}(R)$; $V_j \subseteq \text{rattr}(Y_1.\cdots.Y_j)$ $(1 \leq j \leq k)$; and S is some relation in the database. The effect of this operator is then

$$\{Y_1.Y_2.\cdots.Y_k(Z, W);\ W := E(V_0, \ldots, V_k, R_2)\}(r_1) =$$
$$\{u | \exists t \in r : u[\text{attr}(R) - Y_1] = t[\text{attr}(R) - Y_1]$$
$$\wedge s[Y_1] = \{Y_2.Y_3.\cdots.Y_k(Z, W);\ W := E(t[V_0], V_1, \ldots, V_k, S)\}(t[Y_1])\}$$
$$\text{format}(\{Y_1.Y_2.\cdots.Y_k(Z, W);\ W := E(V_0, \ldots, V_k, S)\}(R)) =$$
$$\text{format}(R_1) \text{ except that the term } Y_1\text{ format}(Y_1) \text{ is replaced by}$$
$$Y_1\text{format}(\{Y_2.Y_3.\cdots.Y_k(Z, W);\ W := E(V_0, \ldots, V_k, R)\}(Y_1))$$

The algebra expressions used within a subrelation constructor may, of course, in turn contain subrelation constructors. However, this is needed only rarely. The subrelation constructor is a simple, but very powerful operator.

4 Problem Definition

Consider now a project-select-join (PSJ) query Q_e written in the form given below. That is,

$$Q_e = \pi[A]\sigma[C](e_1 \times e_2 \times \cdots \times e_k)$$

where A is the projection list and C is the selection condition. Any query composed of PSJ-expressions can be put into this standard form [Y87]. We assume that C is given in conjunctive normal form, with conjuncts $C_1, \ldots, C_s$. Each conjunct C_i $(1 \leq i \leq n)$ consists of a disjunction of one or more atomic selection conditions. For example, consider the flat relation $R(A, B, C, D)$. A valid selection condition is the following: $\sigma[(A > 4) \wedge (B \leq C + 6) \wedge ((A = D) \vee (C < 25) \vee (B > 14))](r)$. Notice that the last conjunct is a disjunction of three atomic conditions.

We now examine the problem of transforming the query expression Q_e to an equivalent expression Q_r. In this paper, we limit Q_e to be a select-join expression

$$\sigma[C_1 \wedge \cdots \wedge C_s](e_1 \times \cdots \times e_k). \quad (1)$$

Each e_i $(1 \leq i \leq k)$ is stored as a nested relation r_i. We are given that the transformation from e_i to r_i involves only a sequence of nest operations. Let ν^i denote the sequence of nest operations that are performed. Then $r_i = \nu^i(e_i)$. The sequence of unnests which transforms any nested relation R into a flat relation will be denoted by μ^*. This definition is taken from [TF86], and is defined as follows:

Definition 4.1 $[\mu^*]$ Let R be any nested relation, and let $Y = \text{rattr}(R)$. While Y is not empty, repeat the following: Choose a subrelation $Z \in Y$ and perform $r' = \mu[Z](r)$. Now let r' be the new r, and define the new Y as $Y = \text{rattr}(R)$. □

It is clear that the sequence of unnest operations used to flatten out a nested relation is not necessarily unique. Using the fact that $r_i = \nu^i(e_i)$, we get

$$\mu^*(r_i) = \mu^*(\nu^i(e_i)) = e_i. \quad (2)$$

We now state without proof an important theorem from [TF86], which is used later to begin the query transformation.

Theorem 4.1 *Let R_i and R_j be nested relations. Then*

$$\mu^*(r_i \times r_j) = \mu^*(r_i) \times \mu^*(r_j).$$

□

Substituting equation 2 into equation 1, we get

$$\sigma[C_1 \wedge \cdots \wedge C_s](\mu^*(r_1) \times \cdots \times \mu^*(r_k)).$$

Applying Theorem 4.1 repeatedly, this becomes

$$\sigma[C_1 \wedge \cdots \wedge C_s]\mu^*(r_1 \times \cdots \times r_k). \tag{3}$$

It is easy to see that this is equivalent to

$$\sigma[C_1]\sigma[C_2]\ldots\sigma[C_s]\mu^*(r_1 \times \cdots \times r_k). \tag{4}$$

We have just proved the following theorem.

Theorem 4.2 *Let $E_1, \ldots, E_k$ be flat relations, and let $R_1, \ldots, R_k$ be the nested relations obtained by the operations $r_i = \nu^i(e_i)$, $(1 \leq i \leq k)$ where each ν^i is a sequence of nest operations. Let $C = C_1 \wedge \cdots \wedge C_s$ be a selection condition given in conjunctive normal form posed on the relation $E_1 \times \cdots \times E_k$. Then*

$$\sigma[C](e_1 \times \cdots \times e_k) \;=\; \sigma[C_1]\sigma[C_2]\ldots\sigma[C_s]\mu^*(r_1 \times \cdots \times r_k).$$

□

Let us pause here and consider a few examples. Table 1 shows three flat derived relations for an employee database which stores the children of each employee, and the training courses each employee has taken. The employee number is the key in all three, but has different names in order to make the transformations easier to follow. Table 2 shows how the derived relations are stored in a nested form according to the following nest operations:

$$\begin{aligned} r_1 &= e_1 \\ r_2 &= \nu[\text{CHILDREN} = (\text{CNAME,DOB,SEX})](e_2) \\ r_3 &= \nu[\text{TRAINING} = (\text{CNO,DATE})](e_3) \end{aligned}$$

Example 4.1 A query is posed on the derived relations asking for the male children of employee number 105.

$$\sigma[(\text{E\#=105}) \wedge (\text{SEX} = \text{M})](e_2).$$

Applying Theorem 4.2, an equivalent query is

$$\sigma[(\text{E\#=105}) \wedge (\text{SEX} = \text{M})]\mu[\text{CHILDREN}](r_2).$$

□

e_1

ENO	NAME
105	John
123	Anne
153	Bruce
205	Ian

e_2

E#	CNAME	DOB	SEX
105	Jane	80/05/10	F
105	Eric	82/10/05	M
123	Maria	79/10/10	F
205	Bob	70/10/16	M
205	Steve	75/01/15	M

e_3

EMP	CNO	DATE
105	314	79/10/10
105	606	81/05/05
105	714	82/06/20
123	315	81/06/13
123	423	82/07/11
153	314	79/10/10

Table 1: The derived relations for the employee database.

r_1

ENO	NAME
105	John
123	Anne
153	Bruce
205	Ian

r_2

E#	CHILDREN		
	CNAME	DOB	SEX
105	Jane	80/05/10	F
	Eric	82/10/05	M
123	Maria	75/11/12	F
205	Bob	70/10/16	M
	Steve	75/01/15	M

r_3

EMP	TRAINING	
	CNO	DATE
105	314	791/0/10
	606	81/05/05
	714	82/06/20
123	315	81/06/13
	423	82/07/11
153	314	79/10/10

Table 2: The derived relations stored as nested relations.

Example 4.2 A query is posed on the derived relations asking if any employee had a child born on the same date that a course took place.

$$\sigma[(\text{E\#=EMP}) \wedge (\text{DOB=DATE})](e_2 \times e_3).$$

Applying Theorem 4.2, an equivalent query is

$$\sigma[(\text{E\#=EMP}) \wedge (\text{DOB=DATE})]\mu[\text{CHILDREN}]\mu[\text{TRAINING}](r_2 \times r_3).$$

□

We now turn to the problem of further transforming the query. The idea is to push the selection conditions past the unnests. This means that the selection is now operating on a nested relation rather than on a flat one. It must do this in a valid manner, observing all the scoping rules. For example, the query expression

$$\sigma[(\text{E\#=EMP}) \wedge (\text{CHILDREN.DOB=TRAINING.DATE})](r_2 \times r_3)$$

makes no sense, since the scoping rules of the selection operator are violated. An equivalent query posed on a flattened structure does make sense, and is

given below.

$$\sigma[(\text{E\#=EMP}) \wedge (\text{DOB=DATE})]\mu^*(r_2 \times r_3).$$

This brings us to a very important point. It is not necessary to flatten out the whole structure before applying the selection condition. Whenever an atomic selection condition involves two or more attributes, all that is required is that the attributes *lie along the same path from the root of the structure.* One way that this can be accomplished is by unnesting in a valid order until the attributes come on the same path. It would be logical to unnest as few subrelations as possible, since unnesting always creates more tuples.

Example 4.3 Using the relations of Tables 1 and 2, the following four queries are equivalent.

$$\sigma[\text{E\#=EMP} \wedge \text{DOB=DATE}](e_2 \times e_3) \tag{5}$$

$$\sigma[\text{E\#=EMP} \wedge \text{DOB=DATE}]\mu[\text{TRAINING}]\mu[\text{CHILDREN}](r_2 \times r_3) \tag{6}$$

$$\mu[\text{TRAINING}']\{(\text{E\#,EMP,NAME,DOB,SEX,TRAINING}'); \text{TRAINING}' := \sigma[\text{E\#=EMP} \wedge \text{DOB=DATE}](\text{TRAINING})\}\mu[\text{CHILDREN}](r_2 \times r_3) \tag{7}$$

$$\mu[\text{CHILDREN}']\{(\text{E\#,EMP,CNO,DATE,CHILDREN}'); \text{CHILDREN}' := \sigma[\text{E\#=EMP} \wedge \text{DOB=DATE}](\text{CHILDREN})\}\mu[\text{TRAINING}](r_2 \times r_3) \tag{8}$$

□

Note that in the third expression of the above example, the subrelation CHILDREN is unnested. This brings all attributes in the selection predicate into the scope of the operand $\mu[\text{CHILDREN}](r_2 \times r_3)$. For the same reason, the subrelation TRAINING is unnested in the fourth expression. What we have done in the last two expressions is succeeded in pushing the select past an unnest, and selected out sub-tuples *lower down* in the structure.

The second way of getting all the attributes in an atomic condition to lie on the same path is by performing *Cartesian products* between subrelations in the structure. For example, the following query is equivalent to the four queries of Example 4.3.

$$\mu[\text{CT}]\{(\text{E\#,EMP,CT}); \text{CT} = \sigma[(\text{E\#=EMP}) \wedge (\text{DOB=DATE})](\text{CHILDREN} \times \text{TRAINING})\}(r_2 \times r_3).$$

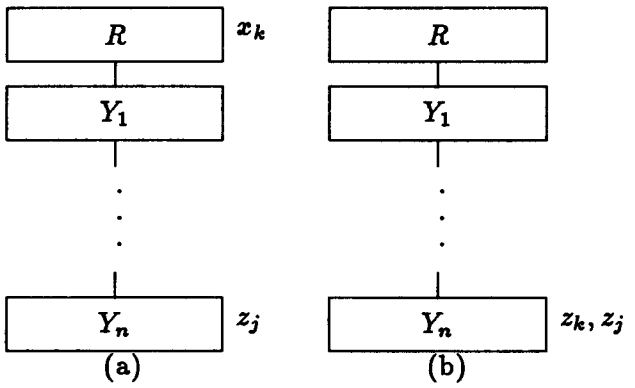

Figure 1: Scheme Tree Diagram for Theorems 5.1 and 5.2.

In this example, by performing a Cartesian product between subrelations CHILDREN and TRAINING, the attributes DOB and DATE have come on the same path (and in this case, are in the same subrelation) from the root. We can then apply the selection condition directly to the nested relation before doing any unnesting.

In general, it is possible to push the selection condition past all the unnests by altering the structure of the nested relation R. Furthermore, this altering of the structure consists of performing only Cartesian products between subrelations in R.

5 Transformation Theorems

In this section, we present theorems which allow us to push an atomic selection condition past all the unnests, possibly performing a change in the structure while doing so. This forms the basis of the query transformation procedure described in Section 6. The structure change considered consists of performing only Cartesian products, or only unnests. The whole idea is to bring the attributes involved in the selection condition along the same path from the root of the structure. In the first two theorems, no structure change is needed, since the attributes are already along the same path.

The first theorem considers a selection condition involving a comparison between a top level attribute and an attribute lower down in the structure.

Theorem 5.1 *(See figure 1a). Let R be a nested relation such that it contains the pathname $P = Y_1.Y_2.\cdots.Y_{n-1}.Y_n$. Without loss of generality, assume that all subrelation names in the scheme tree of R are unique. Let $x_k \in sattr(R), z_j \in sattr(Y_n)$. Let $X_k = attr(Y_k) - Y_{k+1}, k = 1, 2, \ldots, n-1$,*

and $X_0 = attr(R) - Y_1$. *Then*

$$\begin{aligned} & \sigma[x_k \ op \ z_j]\mu[Y_n]\mu[Y_{n-1}]\ldots\mu[Y_1](r) \\ = \ & \mu[Y_n']\mu[Y_{n-1}]\ldots\mu[Y_1]\{(Y_1.\cdots.Y_{n-1}(X_{n-1}, Y_n'); Y_n' := \sigma[x_k \ op \ z_j](Y_n)\}(r) \end{aligned}$$

Proof: Proof by induction:

We first show it for one level of nesting ($n = 1$). We need to show that

$$\sigma[x_k \text{ op } z_j]\mu[Y_1](r) \ = \ \mu[Y_1']\{(X_0, Y_1'); Y_1' := \sigma[x_k \ \theta \ z_j](Y_1)\}(r) \quad (9)$$

The left side of (9) gives

$$\begin{aligned} e_1 &= \mu[Y_1](r) \\ &= \{s | \exists t \in r : s[X_0] = t[X_0] \wedge s[attr(Y_1)] \in t[Y_1]\} \\ \sigma[x_k \text{ op } z_j](e_1) &= \{u | u \in e_1 \wedge (u[x_k] \text{ op } u[z_j]) = \text{true}\} \\ &= \{u | \exists t \in r : u[X_0] = t[X_0] \\ &\quad \wedge u[attr(Y_1)] \in \{v | v \in t[Y_1] \wedge (v[z_j] \text{ op } t[x_k])\}\} \end{aligned}$$

The right side of (9) gives

$$\begin{aligned} r_1 &= \{(X_0, Y_1'); Y_1' := \sigma[x_k \text{ op } z_j](Y_1)\}(r) \\ &= \{s | \exists t \in r : s[X_0] = t[X_0] \wedge s[Y_1'] = \sigma[x_k \text{ op } z_j](t[Y_1])\} \\ &\quad \text{by definition of the subrelation constructor} \\ &= \{s | \exists t \in r : s[X_0] = t[X_0] \wedge s[Y_1'] = \{v | v \in t[Y_1] \wedge (v[z_j] \text{ op } t[x_k])\}\} \\ \mu[Y_1'](r_1) &= \{u | \exists s \in r_1 : u[X_0] = s[X_0] \wedge u[attr(Y_1')] \in s[Y_1']\} \\ &= \{u | \exists t \in r : u[X_0] = t[X_0] \\ &\quad \wedge u[attr(Y_1)] \in \{v | v \in t[Y_1] \wedge (v[z_j] \text{ op } t[x_k])\}\} \\ &\quad \text{(note that attr}(Y_1') = \text{attr}(Y_1)) \end{aligned}$$

Now assume it is true for i levels of nesting ($n = i$). Let $x_k \in \text{sattr}(Y_1)$, $z_j \in \text{sattr}(Y_i)$. Assume

$$\begin{aligned} & \sigma[x_k \text{ op } z_j]\mu[Y_i]\mu[Y_{i-1}]\ldots\mu[Y_1](r) \\ = \ & \mu[Y_i']\mu[Y_{i-1}]\ldots\mu[Y_1]\{(Y_1.\cdots.Y_{i-1}(X_{i-1}, Y_i'); Y_i' := \sigma[x_k \text{ op } z_j](Y_i)\}(r) \end{aligned}$$

Now show for $n = i + 1$ levels of nesting, where $z_j \in \text{sattr}(Y_{i+1})$.

$$\begin{aligned} & \sigma[x_k \text{ op } z_j]\mu[Y_{i+1}]\mu[Y_i]\ldots\mu[Y_1](r) \\ = \ & \sigma[x_k \text{ op } z_j]\mu[Y_{i+1}]\ldots\mu[Y_2](\mu[Y_1](r)) \\ = \ & \mu[Y_{i+1}']\mu[Y_i]\ldots\mu[Y_2]\{(Y_2.\cdots.Y_i(X_i, Y_{i+1}'); Y_{i+1}' := \sigma[x_k \text{ op } z_j](Y_{i+1})\}(\mu[Y_1](r)) \\ & \text{by the inductive hypothesis} \\ = \ & \mu[Y_{i+1}']\mu[Y_i]\ldots\mu[Y_1]\{(Y_1.\cdots.Y_i(X_i, Y_{i+1}'); Y_{i+1}' := \sigma[x_k \text{ op } z_j](Y_{i+1})\}(r) \end{aligned}$$

□

In this theorem, we consider a selection condition comparing two attributes which are in the same subrelation.

Theorem 5.2 *(See figure 1b). Let R be a nested relation such that it contains the pathname $P = Y_1.Y_2.\cdots.Y_{n-1}.Y_n$. Without loss of generality, assume that all subrelation names in the scheme tree of R are unique. Let $z_k, z_j \in sattr(Y_n)$. Let $X_k = attr(Y_k) - Y_{k+1}, k = 1, 2, \ldots, n-1$, and $X_0 = attr(R) - Y_1$. Then*

$$\begin{aligned} & \sigma[z_k \ op \ z_j]\mu[Y_n]\mu[Y_{n-1}]\ldots\mu[Y_1](r) \\ = \ & \mu[Y_n']\mu[Y_{n-1}]\ldots\mu[Y_1]\{Y_1.\cdots.Y_{n-1}(X_{n-1}, Y_n'); Y_n' := \sigma[z_k \ op \ z_j](Y_n)\}(r) \end{aligned}$$

Proof: Proof by induction:

We first show it for one level of nesting. ($n = 1$). We need to show that

$$\sigma[z_k \text{ op } z_j]\mu[Y_1](r) \ = \ \mu[Y_1']\{(X_0, Y_1'); Y_1' := \sigma[z_k \text{ op } z_j](Y_1)\}(r) \quad (10)$$

The left side of (10) gives

$$\begin{aligned} e_1 &= \mu[Y_1](r) \\ &= \{s | \exists t \in r : s[X_0] = t[X_0] \wedge s[attr(Y_1)] \in t[Y_1]\} \\ \sigma[x_k \text{ op } z_j](e_1) &= \{u | u \in e_1 \wedge (u[z_k] \text{ op } u[z_j])\} \\ &= \{u | \exists t \in r : u[X_0] = t[X_0] \\ &\quad \wedge u[attr(Y_1)] \in \{v | v \in t[Y_1] \wedge (v[z_k] \text{ op } v[z_j])\}\} \end{aligned}$$

The right side of (10) gives

$$\begin{aligned} r_1 &= \{(X_0, Y_1'); Y_1' := \sigma[z_k \text{ op } z_j](Y_1)\}(r) \\ &= \{s | \exists t \in r : s[X_0] = t[X_0] \wedge s[Y_1'] = \sigma[z_k \text{ op } z_j](t[Y_1])\} \\ &\quad \text{by definition of the subrelation constructor} \\ &= \{s | \exists t \in r : s[X_0] = t[X_0] \wedge s[Y_1'] = \{v | v \in t[Y_1] \wedge (v[z_k] \text{ op } v[z_j])\}\} \\ \mu[Y_1'](r_1) &= \{u | \exists s \in r_1 : u[X_0] = s[X_0] \wedge u[attr(Y_1')] \in s[Y_1']\} \\ &= \{u | \exists t \in r_1 : u[X_0] = t[X_0] \\ &\quad \wedge u[attr(Y_1')] \in \{v | v \in t[Y_1'] \wedge (v[z_k] \text{ op } v[z_j])\}\} \end{aligned}$$

Now assume it is true for i levels of nesting. ($n = i$). Therefore, assume

$$\begin{aligned} & \sigma[z_k \text{ op } z_j]\mu[Y_i]\mu[Y_{i-1}]\ldots\mu[Y_1](r) \\ = \ & \mu[Y_i']\mu[Y_{i-1}]\ldots\mu[Y_1]\{(Y_1.\cdots.Y_{i-1}(X_{i-1}, Y_i'); Y_i' := \sigma[z_k \text{ op } z_j](Y_i)\}(r) \end{aligned}$$

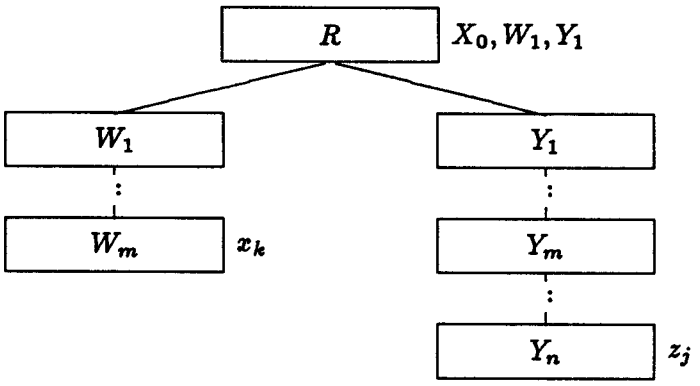

Figure 2: Scheme Tree Diagram for Theorems 5.3 and 5.4

Now show for $n = i + 1$ levels of nesting.

$$\begin{aligned}
& \sigma[z_k \text{ op } z_j]\mu[Y_{i+1}]\ldots\mu[Y_2]\mu[Y_1](r) \\
= \; & \sigma[z_k \text{ op } z_j]\mu[Y_{i+1}]\ldots\mu[Y_2](\mu[Y_1](r)) \\
= \; & \mu[Y'_{i+1}]\mu[Y_i]\ldots\mu[Y_2]\{Y_2.\cdots.Y_i(X_i, Y'_{i+1}); Y'_{i+1} := \sigma[z_k \text{ op } z_j](Y_i)\}(\mu[Y_1](r)) \\
& \text{by the inductive hypothesis} \\
= \; & \mu[Y'_{i+1}]\mu[Y_i]\ldots\mu[Y_1]\{Y_1.\cdots.Y_i(X_i, Y'_{i+1}); Y'_{i+1} := \sigma[z_k \text{ op } z_j](Y_i)\}(r)
\end{aligned}$$

□

This next theorem looks at a selection condition comparing between two attributes that lie along different paths from the root. One is the attribute of a top-level relation. The other attribute is at an arbitrary depth along a different path.

Theorem 5.3 *(See figure 2). Let R be a nested relation such that it contains a path $P_1 = W_1$, and a path $P_2 = Y_1.Y_2.\cdots.Y_{n-1}.Y_n$. Without loss of generality, assume that all subrelation names in the scheme tree of R are unique. Let $x_k \in sattr(W_1), z_j \in sattr(Y_n)$. Let $X_k = attr(Y_k) - Y_{k+1}, k = 1, 2, \ldots, n-1$, and $X_0 = attr(R) - \{W_1, Y_1\}$. Then*

$$\begin{aligned}
& \sigma[x_k \; op \; z_j]\mu[W_1]\mu[Y_n]\mu[Y_{n-1}]\ldots\mu[Y_1](r) \\
= \; & \mu[Y'_n]\mu[Y_{n-1}]\ldots\mu[Y_2]\mu[W_1Y_1]\{W_1Y_1.Y_2.\cdots.Y_{n-1}(X_{n-1}, Y'_n); Y'_n := \sigma[x_k \; op \; z_j](Y_n)\} \\
& \{(X_0, W_1Y_1); W_1Y_1 := W_1 \times Y_1\}(r)
\end{aligned}$$

Proof: Proof by induction on n:

We first show it for one level of nesting ($n = 1$). Let $x_k \in sattr(W_1)$, $z_j \in sattr(Y_1)$. We need to show that

$$\sigma[x_k \text{ op } z_j]\mu[W_1]\mu[Y_1](r) \;=\; \mu[W_1Y_1']\{(X_0, W_1Y_1'); W_1Y_1' := \sigma[x_k \text{ op } z_j](W_1Y_1)\}$$

$$\{(X_0, W_1Y_1); W_1Y_1 := W_1 \times Y_1\}(r)$$

This is equivalent to showing that

$$\begin{aligned} \sigma[x_k \text{ op } z_j]\mu[W_1]\mu[Y_1](r) \;=\; & \mu[W_1Y_1']\{(X_0, W_1Y_1'); \\ & W_1Y_1' := \sigma[x_k \text{ op } z_j](W_1 \times Y_1)\}(r) \end{aligned} \tag{11}$$

The left side of (12) gives

$$\begin{aligned} r_1 &= \mu[W_1]\mu[Y_1](r) \\ &= \{u | \exists t \in r : u[X_0] = t[X_0] \wedge u[attr(W_1)] \in t[W_1] \\ &\quad \wedge u[attr(Y_1)] \in t[Y_1]\} \\ e_1 &= \sigma[x_k \text{ op } z_j](r_1) \\ &= \{u | \exists u \in r_1 \wedge (u[x_k] \text{ op } u[z_j])\} \\ &= \{u | \exists t \in r : u[X_0] = t[X_0] \wedge u[attr(W_1)] \in t[W_1] \\ &\quad \wedge u[attr(Y_1)] \in t[Y_1] \wedge (u[x_k] \text{ op } u[z_j])\} \end{aligned}$$

The right side of (12) gives

$$\begin{aligned} r_2 &= \{(X_0, W_1Y_1'); W_1Y_1' := \sigma[x_k \text{ op } z_j](W_1 \times Y_1)\}(r) \\ &= \{s | \exists t \in r : s[X_0] = t[X_0] \wedge s[W_1Y_1'] = \sigma[x_k \text{ op } z_j](t[W_1] \times t[Y_1])\} \\ &\quad \text{by definition of the subrelation constructor} \\ &= \{s | \exists t \in r : s[X_0] = t[X_0] \wedge s[W_1Y_1'] = \{v |\ v[attr(W_1)] \in t[W_1] \\ &\quad \wedge v[attr(Y_1)] \in t[Y_1] \ \wedge (v[x_k] \text{ op } v[z_j])\}\} \\ e_2 &= \mu[W_1Y_1'](r_2) \\ &= \{u | \exists s \in r_2 : u[X_0] = s[X_0] \wedge u[attr(W_1Y_1')] \in s[W_1Y_1']\} \\ &= \{u | \exists t \in r : u[X_0] = t[X_0] \\ &\quad \wedge u[attr(W_1Y_1')] \in \{v | v[attr(W_1)] \in t[W_1] \wedge v[attr(Y_1)] \in t[Y_1] \\ &\quad \wedge (v[x_k] \text{ op } v[z_j])\}\} \\ &= \{u | \exists t \in r : u[X_0] = t[X_0] \wedge u[attr(W_1)] \in t[W_1] \wedge u[attr(Y_1)] \in t[Y_1] \\ &\quad \wedge (u[x_k] \text{ op } u[z_j])\} \end{aligned}$$

Now assume it is true for i levels of nesting ($n = i$). Let $x_k \in \text{sattr}(W_1)$, $z_j \in \text{sattr}(Y_i)$. Therefore, assume

$$\begin{aligned} & \sigma[x_k \text{ op } z_j]\mu[W_1]\mu[Y_i]\mu[Y_{i-1}] \ldots \mu[Y_1](r) \\ = \;& \mu[Y_i']\mu[Y_{i-1}] \ldots \mu[Y_2]\mu[W_1Y_1] \\ & \{W_1Y_1.\cdots.Y_{i-1}(X_{i-1}, Y_i'); Y_i' := \sigma[x_k \text{ op } z_j](Y_i)\} \\ & \{(X_0, W_1Y_1); W_1Y_1 := W_1 \times Y_1\}(r) \end{aligned}$$

Now show for $n = i + 1$ levels of nesting. Let $x_k \in \text{sattr}(W_1)$, $z_j \in \text{sattr}(Y_{i+1})$.

$$
\begin{aligned}
& \sigma[x_k \text{ op } z_j]\mu[W_1]\mu[Y_{i+1}]\mu[Y_i]\ldots\mu[Y_1](r) \\
= \; & \sigma[x_k \text{ op } z_j]\mu[Y_{i+1}]\mu[W_1]\mu[Y_i]\ldots\mu[Y_1](r) \\
& \text{by Theorem 2 in [TF86]} \\
= \; & \sigma[x_k \text{ op } z_j]\mu[Y_{i+1}](\mu[Y_i']\ldots\mu[Y_2]\mu[W_1Y_1] \\
& \{W_1Y_1Y_2.\cdots.Y_{i-1}(X_{i-1}, Y_i'); Y_i' = Y_i\} \\
& \{(X_0, W_1Y_1); W_1Y_1 = W_1 \times Y_1\}(r)) \\
& \text{by the inductive hypothesis} \\
= \; & \sigma[x_k \text{ op } z_j]\mu[Y_{i+1}](u), \\
& \text{where} \\
u \; = \; & \mu[Y_i']\ldots\mu[Y_2]\mu[W_1Y_1]\{(W_1Y_1Y_2.\cdots.Y_{i-1}(X_{i-1}, Y_i'); Y_i' := Y_i\} \\
& \{(X_0, W_1Y_1); W_1Y_1 := W_1 \times Y_1\}(r) \\
& \text{note that } x_k \in \text{sattr}(U), z_j \in \text{sattr}(Y_{i+1}), Y_{i+1} \in \text{rattr}(U) \\
& \text{Therefore, from Theorem 5.1} \\
& \sigma[x_k \text{ op } z_j]\mu[Y_{i+1}](u) \\
= \; & \mu[Y_{i+1}']\{(U_0, Y_{i+1}'); Y_{i+1}' := \sigma[x_k \text{ op } z_j](Y_{i+1})\}(u), \\
& \text{where } U_0 = \text{attr}(U) - Y_{i+1} \\
= \; & \mu[Y_{i+1}']\{(U_0, Y_{i+1}'); Y_{i+1}' := \sigma[x_k \text{ op } z_j](Y_{i+1})\}\mu[Y_i]\ldots\mu[Y_2]\mu[W_1Y_1] \\
& \{(X_0, W_1Y_1); W_1Y_1 := W_1 \times Y_1\}(r) \\
= \; & \mu[Y_{i+1}']\mu[Y_i]\ldots\mu[Y_2]\mu[W_1Y_1] \\
& \{W_1Y_1.Y_2.\cdots.Y_i(X_i, Y_{i+1}'); Y_{i+1}' := \sigma[x_k \text{ op } z_j](Y_{i+1})\} \\
& \{(X_0, W_1Y_1); W_1Y_1 := W_1 \times Y_1\}(r)
\end{aligned}
$$

□

This theorem looks at a selection condition comparing between two attributes that lie along different paths at variable depths in the structure.

Theorem 5.4 *(See figure 2). Let R be a nested relation such that it contains a path $P_1 = W_1.\cdots.W_m$, and a path $P_2 = Y_1.\cdots.Y_n$. Without loss of generality, assume $m \leq n$, and that all subrelation names in the scheme tree of R are unique. Let $x_k \in sattr(W_m), z_j \in sattr(Y_n)$. Let $V_l = attr(W_l) - W_{l+1}, l = 1, 2, \ldots, m-1$. Let $X_k = attr(Y_k) - Y_{k+1}, k = 1, 2, \ldots, n-1$, and $X_0 = attr(R) - \{W_1, Y_1\}$. Then*

$$\sigma[x_k \; op \; z_j]\mu[W_m]\ldots\mu[W_1]\mu[Y_n]\mu[Y_{n-1}]\ldots\mu[Y_1](r)$$

$$
\begin{aligned}
= \; & \mu[Y'_n]\mu[Y_{n-1}]\ldots\mu[Y_{m+1}]\mu[W_mY_m]\ldots\mu[W_1Y_1] \\
& \{W_1Y_1\ldots W_mY_m.Y_{m+1}.\cdots.Y_{n-1}(X_{n-1},Y'_n);\, Y'_n := \sigma[x_k \; op \; z_j](Y_n)\} \\
& \quad \{(V_{m-1},X_{m-1},W_mY_m);\, W_mY_m := W_m \times Y_m\} \\
& \quad \vdots \\
& \quad \{(X_0,W_1Y_1);\, W_1Y_1 := W_1 \times Y_1\}(r)
\end{aligned}
$$

Proof: Proof by induction on m:

We hold n fixed. Note that $m \leq n$. If $n = 1$, we have already shown the result for $m = 0$ (Theorem 5.1) and for $m = 1$ (Theorem 5.3). Therefore, assume $n > 1$.

When $m = 1$, the result follows from Theorem 5.3. Assume it is true for $m = i$ levels of nesting ($i < n$). Let $x_k \in \text{sattr}(W_i)$, $z_j \in \text{sattr}(Y_n)$. Therefore, assume

$$
\begin{aligned}
& \sigma[x_k \text{ op } z_j]\mu[W_i]\ldots\mu[W_1]\mu[Y_n]\mu[Y_{n-1}]\ldots\mu[Y_1](r) \\
= \; & \mu[Y'_n]\mu[Y_{n-1}]\ldots\mu[Y_{i+1}]\mu[W_iY_i]\ldots\mu[W_1Y_1] \\
& \{W_1Y_1\ldots W_iY_i.Y_{i+1}.\cdots.Y_{n-1}(X_{n-1},Y'_n);\, Y'_n := \sigma[x_k \text{ op } z_j](Y_n)\} \\
& \quad \{(V_{i-1},X_{i-1},W_iY_i);\, W_iY_i := W_i \times Y_i\} \\
& \quad \vdots \\
& \quad \{(X_0,W_1Y_1);\, W_1Y_1 := W_1 \times Y_1\}(r)
\end{aligned}
$$

Now show for $m = i + 1$ levels of nesting. Let $x_k \in \text{sattr}(W_{i+1})$, $z_j \in \text{sattr}(Y_n)$.

$$
\begin{aligned}
& \sigma[x_k \text{ op } z_j]\mu[W_{i+1}]\mu[W_i]\ldots\mu[W_1]\mu[Y_n]\mu[Y_{n-1}]\ldots\mu[Y_1](r) \\
= \; & \sigma[x_k \text{ op } z_j]\mu[W_{i+1}]\mu[Y'_n]\mu[Y_{n-1}]\ldots\mu[Y_{i+1}]\mu[W_iY_i]\ldots\mu[W_1Y_1] \\
& \{W_1Y_1\ldots W_iY_i.Y_{i+1}.\cdots.Y_{n-1}(X_{n-1},Y'_n);\, Y'_n := Y_n\} \\
& \quad \{(V_{i-1},X_{i-1},W_iY_i);\, W_iY_i := W_i \times Y_i\} \\
& \quad \vdots \\
& \quad \{(X_0,W_1Y_1);\, W_1Y_1 := W_1 \times Y_1\}(r) \\
& \text{by the inductive hypothesis} \\
= \; & \sigma[x_k \text{ op } z_j]\mu[W_{i+1}]\mu[Y_n]\ldots\mu[Y_{i+1}](u) \\
& \text{where} \\
u \; = \; & \mu[W_iY_i]\ldots\mu[W_1Y_1] \\
& \quad \{(V_{i-1},X_{i-1},W_iY_i);\, W_iY_i := W_i \times Y_i\} \\
& \quad \vdots \\
& \quad \{(X_0,W_1Y_1);\, W_1Y_1 := W_1 \times Y_1\}(r)
\end{aligned}
$$

note that $x_k \in \text{sattr}(W_{i+1}), z_j \in \text{sattr}(Y_n)$, and $W_{i+1}, Y_{i+1} \in \text{rattr}(u)$.
It follows from Theorem 5.3 that

$$
\begin{aligned}
& \sigma[x_k \text{ op } z_j]\mu[W_{i+1}]\mu[Y_n]\ldots\mu[Y_{i+1}](u) \\
= \; & \mu[Y_n']\mu[Y_{n-1}]\ldots\mu[Y_{i+2}]\mu[W_{i+1}Y_{i+1}] \\
& \{W_{i+1}Y_{i+1}.Y_{i+2}.\cdots.Y_{n-1}(X_{n-1}, Y_n'); Y_n' := \sigma[x_k \text{ op } z_j](Y_n)\} \\
& \{(U_0, W_{i+1}Y_{i+1}); W_{i+1}Y_{i+1} := W_{i+1} \times Y_{i+1}\}(u) \\
& \text{where } U_0 = attr(u) - W_{i+1}Y_{i+1} \\
= \; & \mu[Y_n']\mu[Y_{n-1}]\ldots\mu[Y_{i+2}]\mu[W_{i+1}Y_{i+1}] \\
& \{W_{i+1}Y_{i+1}.Y_{i+2}.\cdots.Y_{n-1}(X_{n-1}, Y_n'); Y_n' := \sigma[x_k \text{ op } z_j](Y_n)\} \\
& \{(U_0, W_{i+1}Y_{i+1}); W_{i+1}Y_{i+1} := W_{i+1} \times Y_{i+1}\} \\
& \mu[W_iY_i]\ldots\mu[W_1Y_1] \\
& \quad \{(V_{i-1}, X_{i-1}, W_iY_i); W_iY_i := W_i \times Y_i\} \\
& \quad \vdots \\
& \quad \{(X_0, W_1Y_1); W_1Y_1 := W_1 \times Y_1\}(r) \\
= \; & \mu[Y_n']\mu[Y_{n-1}]\ldots\mu[Y_{i+2}]\mu[W_{i+1}Y_{i+1}]\mu[W_iY_i]\ldots\mu[W_1Y_1] \\
& \{W_1Y_1.\cdots.W_{i+1}Y_{i+1}.Y_{i+2}.\cdots.Y_{n-1}(X_{n-1}, Y_n'); Y_n' := \sigma[x_k \text{ op } z_j](Y_n)\} \\
& \quad \{(X_i, W_{i+1}Y_{i+1}); W_{i+1}Y_{i+1} := W_{i+1} \times Y_{i+1}\} \\
& \quad \vdots \\
& \quad \{(X_0, W_1Y_1); W_1Y_1 := W_1 \times Y_1\}(r)
\end{aligned}
$$

□

The following theorem looks at the interaction of the select and unnest operators. Specifically, when the selection operation involves two attributes which may not be compared because they are not in the same scope, the structure is *unnested* along the shortest path until the attributes are on the same path. This is an alternative to performing Cartesian products to bring the attributes on the same path.

Theorem 5.5 *Let R be a nested relation such that it contains a path $P_1 = W_1.\cdots.W_m$, and a path $P_2 = Y_1\cdots.Y_n$. Without loss of generality, assume $m \leq n$, and that all subrelation names in the scheme tree of R are unique. Let $x_k \in sattr(W_m), z_j \in sattr(Y_n)$. Let $V_l = attr(W_l) - W_{l+1}, l = 1, 2, \ldots, m-1$. Let $X_k = attr(Y_k) - Y_{k+1}, k = 1, 2, \ldots, n-1$, and $X_0 = attr(R) - \{W_1, Y_1\}$. Then*

$$
\begin{aligned}
& \sigma[x_k \;\theta\; z_j]\mu[W_m]\ldots\mu[W_1]\mu[Y_n]\ldots\mu[Y_1](r) \\
= \; & \mu[Y_n']\mu[Y_{n-1}]\ldots\mu[Y_1]\{Y_1.\cdots.Y_{n-1}(X_{n-1}, Y_n'); Y_n' := \sigma[x_k \;\theta\; z_j](Y_n)\} \\
& (\mu[W_m]\ldots\mu[W_1](r))
\end{aligned}
$$

Proof:

$$
\begin{aligned}
& \sigma[x_k\ \theta\ z_j]\mu[W_m]\ldots\mu[W_1]\mu[Y_n]\ldots\mu[Y_1](r) \\
= \ & \sigma[x_k\ \theta\ z_j]\mu[Y_n]\ldots\mu[Y_1](\mu[W_m]\ldots\mu[W_1](r)) \\
& \text{by Theorem 2 in [TF86]} \\
= \ & \mu[Y'_n]\mu[Y_{n-1}]\ldots\mu[Y_1]\{Y_1.\cdots.Y_{n-1}(X_{n-1}, Y'_n); Y'_n := \sigma[x_k\ \theta\ z_j](Y_n)\} \\
& (\mu[W_m]\ldots\mu[W_1](r)) \\
& \text{by Theorem 5.3}
\end{aligned}
$$

□

Next, we need a theorem which allows us to push a selection which is a disjunction of atomic conditions past unnests. Let the selection conjunct C_i consist of a disjunction of atomic conditions $c_1 \vee c_2 \vee \cdots \vee c_l$. Notice in the previous theorems that pushing a single atomic condition past unnests consists of two phases: a structure change on R so that both attributes come on the same path from the root; a selection on the subrelation containing the most deeply nested of the attributes. When we are dealing with a disjunction, we need to bring *all* the attributes in the disjunction on the same path from the root. The selection can then be applied on the subrelation containing the most deeply nested of the attributes. We introduce a new operator, CP, to simplify the notation.

Let c be an atomic selection condition. We define $\mathrm{CP}(c, r)$ to be the nested relation r' that results from structure changes that are performed on r in order for the attributes in c to lie on the same path. These structure changes consist exclusively of Cartesian products between subrelations at the same scheme tree level in R.

Definition 5.1 [$\mathrm{CP}(c, r)$] Let R be a nested relation. Let c be an atomic selection condition posed as $\sigma[c]\mu^*(r)$. Let x_i, x_j be attributes in $\mu^*(r)$.

1. If c is of the form x_i op K, then
$$\mathrm{CP}(c, r) = r.$$
2. If c is of the form x_i op $x_j + K$, then there are two cases to consider.
   (a) If x_i, x_j are on the same path from the root in R, then
   $$\mathrm{CP}(c, r) = r.$$
   (b) If x_i, x_j are on different paths from the root, then for $k \geq 0$ let
   $$\begin{aligned} x_i &\in \mathrm{sattr}(Y_1.Y_2.\cdots.Y_k.Z_1.Z_2.\cdots.Z_n), \\ x_j &\in \mathrm{sattr}(Y_1.Y_2.\cdots.Y_k.W_1.W_2.\cdots.W_m). \end{aligned}$$

Without loss of generality, let $n \geq m > 0$. Let $X_l = attr(Z_l) - Z_{l+1}$, $V_l = attr(W_l) - W_{l+1}$, $l = 1, 2, \ldots, m-1$. Let $X_0 = attr(Y_k) - \{W_1, Z_1\}$ if $k > 0$, and $X_0 = attr(R) - \{W_1, Z_1\}$ if $k = 0$. Then

$$\begin{aligned} \mathrm{CP}(c, r) \;=\; & \{(X_{m-1}, V_{m-1}, Z_m W_m); Z_m W_m := Z_m \times W_m\} \\ & \vdots \\ & \{(X_0, Z_1 W_1); Z_1 W_1 := Z_1 \times W_1\}(r) \end{aligned}$$

□

From Theorems 5.1 - 5.4 and the definition of $\mathrm{CP}(c, r)$, it is clear that

$$\sigma[c]\mu^*(r) \;=\; \mu^*\{P_1.\cdots.P_{d-1}(\mathrm{attr}(P_{d-1}) - P_d, P'_d); P'_d := \sigma[c](P_d)\}(\mathrm{CP}(c, r))$$

where $P_1.\cdots.P_d$ is the pathname of the subrelation containing the most deeply nested of the attributes in the atomic selection condition c.

We now extend the function CP to work on a selection conjunct C_i.

Definition 5.2 [$\mathrm{CP}(C_i, r)$] Let $C_i = c_1 \vee \cdots \vee c_l$ be a conjunct consisting of a disjunction of atomic selection conditions. Then we define

$$\begin{aligned} \mathrm{CP}(C_i, r) \;&=\; \mathrm{CP}(c_1 \vee \cdots \vee c_l, r) \\ &=\; \mathrm{CP}(c_l, \mathrm{CP}(\ldots, \mathrm{CP}(c_2, \mathrm{CP}(c_1, r))\ldots)). \end{aligned}$$

□

Now $\mathrm{CP}(C_i, r)$ returns a relation which has all the attributes in C_i on the same path. There will be some attribute(s) in C_i which is the *most deeply* nested on the path. Let $P_1.P_2.\cdots.P_d$ be the common path of all the attributes in C_i, where P_d is the subrelation in $\mathrm{CP}(C_i, r)$ that contains the most deeply nested of the attribute(s). (We assume that $d \geq 1$, for if all attributes are contained at the root, then trivially $\sigma[C_i]\mu^*(r) = \mu^*\sigma[C_i](r)$).

We are now ready to state the following theorem.

Theorem 5.6 *Let R be a nested relation, and let $C_i = c_1 \vee \cdots \vee c_l$ be a conjunct consisting of a disjunction of atomic conditions. $P_1.\cdots.P_d$ is the pathname in $CP(C_i, r)$ as defined above. Then*

$$\sigma[C_i]\mu^*(r) \;=\; \mu^*\{P_1.\cdots.P_{d-1}(attr(P_{d-1}) - P_d, P'_d); P'_d := \sigma[C_i](P_d)\}(CP(C_i, r)).$$

The proof is easily seen. All attributes being tested in C_i are brought into a common path first, as required by the scoping rules of the algebra, and the testing is done at the level of the most deeply nested attribute which is in C_i.

6 The Transformation Procedure

We are now in a position to describe a procedure for transforming the query Q_e to an equivalent query Q_r that operates on a nested relation. The query we begin with, Q_e, has the form given in equation (1)

$$\sigma[C_1 \wedge \cdots \wedge C_s](e_1 \times \cdots \times e_k).$$

We have already shown that this is equivalent to the query in equation (4)

$$\sigma[C_1]\sigma[C_2]\cdots\sigma[C_s]\mu^*(r)$$

where $r = r_1 \times \cdots \times r_k$. A high level view of the procedure to transform the query is: 1. Begin with the query

$$\sigma[C_1]\sigma[C_2]\cdots\sigma[C_s]\mu^*(r)$$

2. WHILE there are still conjuncts C_i to the left of μ^* DO
3. Choose a conjunct C_i
4. $r \leftarrow \mathrm{CP}(C_i, r)$
5. Let $P_1.\cdots.P_d$ be the pathname identifying the subrelation containing the most deeply nested attribute(s) in r. Transform the expression to:

$$\sigma[C_1]\cdots\sigma[C_{i-1}]\sigma[C_{i+1}]\cdots\sigma[C_s]\mu^* \{P_1.\cdots.P_{d-1}(\mathrm{attr}(P_{d-1}) - P_d, P'_d); P'_d := \sigma[C_i](P_d)\}(\mathrm{CP}(C_i, r))$$

6. END WHILE

The efficiency of the loop, of course, depends on the choice of the conjunct chosen in step 3. The main thing to note is that a Cartesian product always increases the number of tuples, and applying a selection condition always reduces the number of tuples. Thus it is desirable to first choose all conjuncts C_j such that $r = \mathrm{CP}(C_j, r)$. That is, apply all the selection conjuncts for which no Cartesian products need to be performed. Furthermore, a selection condition applied to attributes closer to the root eliminates more data than a selection condition applied deeper down in the structure. So among all the selection conjuncts C_j such that $r = \mathrm{CP}(C_j, r)$, we choose the one whose most deeply nested attribute(s) is as close to the root as possible.

Whenever a structure change must be made, it is desirable to choose a conjunct C_j such that the number of Cartesian products caused by the operation $\mathrm{CP}(C_j, r)$ is minimal. In this way, the least amount of new tuples are generated. Once we get the new $r = \mathrm{CP}(C_j, r)$, there may be several selection conjuncts C_i such that $r = \mathrm{CP}(C_i, r)$ for the new r.

With the above points in mind, we modify step 3 of the algorithm to:

3. Choose a conjunct C_i such that $r = \mathrm{CP}(C_i, r)$, and where the most deeply nested attribute in C_i is closer to the root than in any of the other qualifying conjuncts. If none exists, then choose any C_j where $r \neq \mathrm{CP}(C_i, r)$ such that the number of Cartesian products required is less than for any other qualifying conjuncts.

7 Concluding Remarks

Many possible expressions can be generated for Q_r, some more efficient than others. We have presented an algorithm which tries to minimize the number of new tuples generated at each iteration, and tries to select out qualifying tuples as early as possible. No attempt is made at generating the optimal expression. There is always the tradeoff between doing extensive preprocessing to get the optimal expression, and just finding a "good" expression faster. For now, we have chosen this second route.

This work is currently being extended to include projections. The rules for pushing projections past unnests will be added in a forthcoming paper. We expect this to be a straightforward procedure.

The algebra used in the paper will be the internal representation for queries in the *LauRel* project [L88]. LauRel is a prototype database management system being developed at the University of Waterloo. It will support an extended relational model, an SQL-based language called SQL/W, and also serve as a testbed for research on parallelism in database systems. We expect the theoretical results of this paper to be of direct use in LauRel.

References

[DL87] V. Deshpande and P.-Å. Larson, An Algebra for Nested Relations, *Technical Report, CS-87-65*, University of Waterloo, 1987.

[JS82] G. Jaeschke and H.-J. Schek, Remarks on the algebra of non-first-normal-form relations, *Proc. ACM SIGACT/SIGMOD Symposium on Principles of Database Systems*, 1982, p. 124-138.

[L88] P.-Å. Larson, The Data Model and Query Language of LauRel, *Data Engineering Bulletin,* Vol. 11 No. 3, 1988, p. 23-30.

[LY87] P.-Å. Larson and H.Z. Yang, Computing Queries from Derived Relations: Theoretical Foundation, *Technical Report, CS-87-35*, University of Waterloo, 1987.

[RK87] M.A. Roth and H.F.Korth, The Design of ¬1NF Relational Databases into Nested Normal Form, *Proc. of ACM/SIGMOD Annual Conference*, San Francisco, 1987, p. 143-159.

[RKS84] M. A. Roth and H. F. Korth and A. Silberschatz, Extended Algebra and Calculus for ¬1NF Relational Databases, *Technical Report, TR-84-36*, University of Texas at Austin, 1984.

[SS80] M. Schkolnik and P. Sorenson, Denormalization: A Performance Oriented Database Design Technique, *Proc. AICA Conference*, Bologna, Italy, 1980.

[S86] M.H. Scholl, Theoretical Foundation of Algebraic Optimization Utilizing Unnormalized Relations, *Proceedings of the International Conference on Database Theory*, Rome, Italy, September, 1986, Springer-Verlag, p. 380-396.

[SS81] M. Schkolnik and P. Sorenson, *The Effects of Denormalization on Database Performance*, Research Report RJ3082 (38128), IBM Research Lab, San Jose, CA, 1981.

[SS86] H.-J. Schek and M.H. Scholl, The Relational Model with Relation-Valued Attributes, *Information Systems*, Volume 11, No. 2, 1986, p. 137-147.

[TF86] Stan J. Thomas and Patrick C. Fischer, Nested Relational Structures, *Advances in Computing Research*, Vol. 3, 1986, JAI Press Inc., p. 269-307.

[Y87] H. Z. Yang, Query Transformation, Ph.D. Thesis, University of Waterloo, 1987.