Multiple Paths Join for Nested Relational Databases*

Hong-Cheu Liu
Kotagiri Ramamohanarao

Technical Report 93/23

Department of Computer Science
The University of Melbourne, Parkville 3052, Australia
E-mail: hong@cs.mu.OZ.AU rao@cs.mu.OZ.AU

October, 1993

Abstract: This paper presents a solution to the problem of how to efficiently process queries which include join operations in the nested relational model. We focus on the join operation, and introduce a new join operator called P-join. The P-join operator merges two scheme trees. It combines the advantages of the extended natural join and recursive join for efficient data access. The algebraic expression of queries, when expressed using P-join, is more succinct and more easily optimized than it is when using other join operators. We also present a series of algebraic equivalences which are useful for query optimization in the nested relational model and prove the correctness of the P-join operation by adapting the criteria of faithfulness and precision of generalization.

*To appear in *Proceedings of Fifth Australasian Database Conference*, Christchurch, New Zealand, January 1994.

1. Introduction

The relational data model is a widely accepted model for conventional database management systems. However, it has been generally recognized that the 1NF relational model, although simple and mathematically tractable, is not rich enough to model emerging advanced database applications which involve complex objects. In order to model these applications the nested relational model has been developed and in the last decade much research has been carried out in this area. It use hierarchical structures rather than flat tables to enable the representation of complex objects.

The nested relational model provides the structural core of objected-oriented databases, and plays the role of an intermediate stage in an evolutionary path from the relational model to the object-oriented data model and query language [12]. By transforming object queries into an object algebra in the spirit of (nested) relational algebra, optimization techniques can be applied to query processing in object-oriented systems [3, 12]. However, there has been relatively little work done on query processing for the nested relational model. Algebraic optimization techniques for relational database languages, though well-understood and quite successful, cannot be fully applied to the nested relational model [2, 4]. In this paper, we consider this issue.

Like in the relational model, the join operator is one of the most expensive operators in nested relational query processing. The definitions of join which have been proposed for the nested relational model are of four types: standard natural join, extended natural join [8], unnest-join [3], and recursive join [1]. Many natural queries cannot be expressed by these join operators without restructuring operations. We introduce, within a restricted set of nested scheme trees, a more general form of join (P-join) which does not require as many restructuring operators and combines the advantages of the extended natural join and recursive join for efficient data access. The work presented in this paper extends that of [5].

The structure of this paper is as follows. In the remainder of this section, we present a motivating example. Section 2 briefly reviews the nested relational model and recalls some well-known concepts. In Section 3, we present the path-dependent join (i.e. P-join). In Section 4, we derive some algebraic equivalences. In Section 5, we discuss the correctness of the P-join operator. Finally, we draw some conclusions and discuss future work in Section 6.

1.1 Motivating Example

Let us introduce the following example.

Example 1.1 Consider the following database which has nested relations Product and Part.

Product = (prodname, Warranty(premium, country, w-period), Composition(c-name, c-id, Parts(p-name, quantity)), Distributors(company, fee))
Part = (p-name, weight, Warranty(country, w-period), Source(company, cost))

Consider also the following query.

Q1: Find those countries and the corresponding w-periods which are same in some product and one of its parts, together with companies that are both a product distributor and a part source. (Presumably, this information might be used to reduce premium.) Group the result on prodname and p-name.

Due to the fact that the intending join attributes company, p-name and w-period appear on different subscheme levels, the query Q1, cannot be directly expressed by any of the four join operators mentioned above *without* restructuring the data.

When the algebraic expression of queries includes restructuring operators, they are not easily optimized [1]. The problem of how to optimize queries which include join operations in the NF^2 relational model is main topic discussed in the paper.

2. The Nested Relational Model

In this section, we briefly review some well-known concepts and present the basic definitions of NF^2 relational data structures and NF^2 relational algebra used throughout this paper. A detailed formalism can be found in [8, 14]. First, we introduce some notational conventions. We assume all attributes of our relations are contained in a universal finite set $\mathcal{U}$. The elementary values are in the countable set $\mathcal{V}$. The symbol $\mathcal{P}(S)$ denotes the powerset of a set S. We also assume relations considered in this paper do not contain null values. However the results of this paper can be extended to handle relations with null values.

2.1. Nested Relational Schemes

We assume familiarity with 1NF relations and relational algebra on these relations. Because the data types of the nested relational model are conceptually simple extensions of 1NF relational data types, the basic definitions which follow will contain the 1NF case as a base case.

Definition 2.1 A *relation scheme* R is recursively defined by the following rules:
(1) if $\{A_1, ..., A_k\} \subset \mathcal{U}$ and $A_1, ..., A_k$ are atomic-valued attributes, then $R = (A_1, ..., A_k)$ is a relation scheme.
(2) if $\{A_1, ..., A_k\} \subset \mathcal{U}$, $A_1, ..., A_k$ are atomic-valued attributes, and $R_1, ..., R_n$ are relation schemes, then R=$(A_1, ..., A_k, R_1, ..., R_n)$ is a relation scheme. □

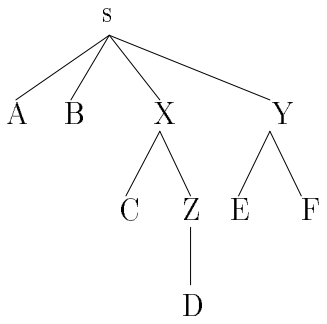
The atomic-valued attributes $A_1, ..., A_k$ are called *zero-order* attributes; $R_1, ..., R_n$ are called relation-valued attributes or *higher-order* attributes. The set of top-level attributes in a relation scheme (or a subrelation scheme) R is denoted by E_R.

In the 1NF case, for each attribute A there is an associated set of values $DOM(A)$. In the nested relation, without loss of generality, each atomic-valued attribute $A \in \mathcal{U}$

s

A	B	X		Y	
		C	Z	E	F
			D		
a_1	b_1	c_1	d_1	e_1	f_1
a_2	b_2	c_1	d_1	e_1	f_2
			d_2	e_2	f_2
		c_2	d_2		
a_3	b_3	c_3	d_3	e_3	f_3

(a) nested relation s



(b) nested scheme tree of s

Figure 1: An example of nested relation and nested scheme tree

may assume values drawn from a basic set $\mathcal{V}$. The domain of a given attribute X of a nested relation is recursively defined by

$$DOM(X) = \begin{cases} \mathcal{V} & \text{if } X \text{ is zero-order attribute.} \\ \mathcal{P}(\times_{S \in E_X} DOM(S)) & \text{if } X \text{ is higher-order attribute.} \end{cases}$$

Definition 2.2 An *instance* r defined on a relation scheme $R = \{A_1, ..., A_k, R_1, ..., R_l\}$ is a set of ordered $(k+l)$-tuples, $r \in \mathcal{P}(DOM(A_1) \times ... \times DOM(A_k) \times DOM(R_1) \times ... \times DOM(R_l))$. The *family of instances* defined on scheme R is denoted by I_R; $I_R = \mathcal{P}(DOM(A_1) \times ... \times DOM(A_k) \times DOM(R_1) \times ... \times DOM(R_l))$. □

The projection of relation r onto attributes N is denoted $r[N]$, and similarly, the projection of tuple $t \in r$ onto attributes N is denoted $t[N]$. A relation structure $\mathcal{R}$ consists of a relation scheme R and an instance r defined on R, and is denoted $< R, r >$.

Restructuring operators *nest* and *unnest* are used to add one level of nesting to a relation and flatten a relation by one level respectively. Nest (ν) takes a relation structure $\mathcal{R} =< R, r >$ and groups together tuples with common values in some subset of the attributes in R. Unnest (μ), the inverse of the nest operator, takes a relation structure nested on some set of attributes and ungroups it by one level [14].

2.2. Nested Scheme Tree

A nested relation scheme is structured as a rooted tree in which the nodes are labeled with elements of $\mathcal{U}$. Such a tree is called a scheme tree. We write T_R to represent the scheme tree of the relation scheme R. When there is no ambiguity, we simply denote it by T. The set of nodes of scheme tree T is denoted by $node(T)$. The set of leaf nodes of scheme tree T is denoted by $leaf(T)$. Figure 1 shows a relation structure and the corresponding scheme tree.

Definition 2.3 In a scheme tree we define the following concepts.

- If N_1, N_2, ..., N_k is a sequence of nodes in a scheme tree T such that N_i is the parent of N_{i+1}, for $1 \leq i < k$, then this sequence is called the *path* from node N_1 to node N_k.
- The *length* of a path is one less than the number of nodes in the path.
- If there is a path from node N_i to node N_j, $1 \leq i < j \leq k$, then N_i is an *ancestor* of N_j and N_j is a *descendant* of N_i. □

In order to define selection operator in the next section, we give the concept of selection-comparable nodes.

Definition 2.4 For all nodes $N_a, N_b \in leaf(T)$, where $N_a \neq N_b$, if node N_a is a child of an ancestor of a node N_b, then N_a and N_b are called *selection-comparable* nodes. We denote it by $N_a \xrightarrow{\sigma} N_b$. □

For example, in Figure 1 (b), $B \xrightarrow{\sigma} D$, $A \xrightarrow{\sigma} C$, but C and E are not selection-comparable nodes.

Definition 2.5 The *expression* of a node N in the scheme tree T_R is denoted by $R_i \cdot P^i$, where $R_i \in E_R$; $P^i = \emptyset$ if $R_i \in leaf(T_R)$ or $P^i =$ the expression of node N in subtree T_{R_i}. □

For example the expression of node D in the scheme tree T_s in Figure 1 is $X \cdot Z \cdot D$.

2.3. Selection Operator and Projection Operator

We need more sophisticated selection and projection operators which can work on nested relations.

Selection Operator[1]

A predicate involved in a selection condition is expressed by $e_1 \phi e_2$, where ϕ is a comparison operator; e_1 and e_2 are the expressions for selection-comparable nodes in the scheme tree, or one is an expression of a node and the other is a constant value (including a relation-valued constant). Comparison operators include set comparison and set membership operators, in addition to the usual arithmetic operators i.e. $\phi \in \{<, \leq, =, \neq, >, \geq, \in, \notin, \subset, \subseteq, \supset, \supseteq, =\}$.

Definition 2.6 Let r be a relation with relation scheme R. Let $\theta = e_1 \phi e_2$ be a selection condition; c be a constant which belongs to $\mathcal{C}$, where $\mathcal{C}$ is set of all constants including relation-valued constants. Then $\sigma_\theta(r)$ is defined as follows:

[1] This selection operator is more general than the selection operator defined in [1, 11].

(1) $\sigma_\theta(r) = \{t \mid (t \in r) \wedge (\theta(t) = true)\}$ if θ is a condition on E_R i.e., $e_i \in (E_R \cup \mathcal{C})$.

(2) $\sigma_\theta(r) = \{t \mid (\exists t_r \in r),\ (t[E_R - R_i] = t_r[E_R - R_i]) \wedge (t[R_i] = \sigma_{\theta'}(t_r[R_i]) \neq \emptyset)\}$

where $\theta' = \begin{cases} P^i \phi c & \text{if } \theta = R_i \cdot P^i \phi c \\ P^i \phi t_r[R_j] & \text{if } \theta = R_i \cdot P^i \phi R_j \\ P^{i_1} \phi P^{i_2} & \text{if } \theta = R_i \cdot P^{i_1} \phi R_i \cdot P^{i_2} \end{cases}$

$R_i, R_j \in E_R$; P^{i_1}, P^{i_2} are the expressions for selection-comparable nodes in subtree T_{R_i}.

For example Figure 2(b) shows a recursive selection, $\sigma_{X \cdot C = c_1}$, on the relation s.

Projection Operator[2]

Definition 2.7 Let r be a relation with relation scheme R. The project-list L of R has the form (1) L is empty, or (2) $L = (R_1 L_1, ..., R_n L_n)$, where $R_i \in E_R$; L_i is a project-list of R_i. Then $\pi_L(r)$ is defined as follows:

(1) $\pi(r) = r$

(2) $\pi_{(R_1 L_1, ..., R_n L_n)}(r) = \{t \mid (\exists t_r \in r) \wedge (t[R_i] = \pi_{L_i}(t_r[R_i])),\ i \in \{1, ..., n\}\}$ □

For example Figure 2(c) shows a recursive projection, $\pi_{(A, X(Z(D)))}$, on s.

(a) nested relation s

A	B	X		Y	
		C	Z	E	F
			D		
a_1	b_1	c_1	d_1	e_1	f_1
a_2	b_2	c_1	d_1	e_1	f_2
			d_2	e_2	f_2
		c_2	d_2		
a_3	b_3	c_3	d_3	e_3	f_3

(b) $\sigma_{X \cdot C = c_1}(s)$

A	B	X		Y	
		C	Z	E	F
			D		
a_1	b_1	c_1	d_1	e_1	f_1
a_2	b_2	c_1	d_1	e_1	f_2
			d_2	e_2	f_2

(c) $\pi_{(A, X(Z(D)))}(s)$

A	X
	Z
	D
a_1	d_1
a_2	d_1
	d_2
	d_2
a_3	d_3

Figure 2: Examples of recursive selection and projection on s

3. Path-dependent Join

First we briefly review some auxiliary concepts and describe the P-join operator with single join-path which has been proposed in [5]. We then extend this to multiple join-paths.

[2]The definition of projection operator has the same meaning as the projection operator defined in [1].

(a) r

A	X		Y
	B	C	D
1	1	1	5
	3	3	10
2	2	2	10

(b) q

B	X		Y
	B	C	D
2	1	1	5
	2	2	
3	4	4	5
			10

(c) $r \times^e q$

A	B	X				Y	
		B_r	C_r	B_q	C_q	D_r	D_q
1	2	1	1	1	1	5	5
		1	1	2	2	10	5
		3	3	1	1		
		3	3	2	2		
1	3	1	1	4	4	5	5
		3	3	4	4	5	10
						10	5
						10	10
2	2	2	2	1	1	10	5
		2	2	2	2		
2	3	2	2	4	4	10	5
						10	10

Figure 3: An example of extended Cartesian product

3.1. Extended Cartesian Product and Path-dependent Cartesian Product

The idea behind the extended Cartesian product $\times^e$ is that we form the Cartesian product by combining two relational operands with common higher-order attributes not only at the top level but also at the subscheme levels.

Let "$\circ$" denote the concatenation operator for tuples as well as relation schemes. For example, let two tuples be $t_1 = (1, 2)$, $t_2 = (2, 4)$, then $t_1 \circ t_2 = (1, 2, 2, 4)$; let $E_R = (A, X)$, $E_Q = (B, X)$ be two schemes of relations r, q respectively, then $E_R \circ E_Q = (A, X, B, X)$. We distinguish between the two Xs with suffixes X_r, X_q.

Definition 3.1 Let r and q be two nested relations, with schemes R and Q respectively. Then the extended Cartesian product, $\times^e$, is defined as follows.

(1) if r and q contain no common higher-order attributes
$r \times^e q = r \times q$
(2) if r and q contain common higher-order attributes
$r \times^e q = \{t | (\exists t_r \in r, \exists t_q \in q),\ t[(E_R - \bar{X}) \circ (E_Q - \bar{X})] = t_r[E_R - \bar{X}] \circ t_q[E_Q - \bar{X}]$
$\wedge\ t[X_i] = t_r[X_i] \times^e t_q[X_i], \forall 1 \leq i \leq k\}$
where $\bar{X} = \{X_1, ..., X_k\}$ are common higher-order attributes. □

Example 3.1 Let r and q be the two relations given in Figure 3(a) and (b) respectively. Figure 3(c) shows the extended Cartesian product of r and q.

Definition 3.2 Let R be a relation scheme, and T be the scheme tree of R. A path $P_r = (N_1 \cdots N_k)$ is a *join-path* of R if N_1 is a child of $root(T)$ and N_k is a non-leaf node of T. □

Definition 3.3 Let r and q be two relations with schemes R and Q respectively. Then the path-dependent Cartesian product, $\overset{p}{\times}$, is defined as follows.

(1) if $P_r = P_q = \emptyset$,
$r(P_r) \overset{p}{\times} q(P_q) = r \times^e q$.

(2) if $P_r = R_i \cdot P_r^i$ and $P_q = Q_j \cdot P_q^j$ are two join-paths of R and Q respectively,
$r(P_r) \overset{p}{\times} q(P_q) =$

(a) $\{t|(\exists t_r \in r), (t[E_R - \{R_i\}] = t_r[E_R - \{R_i\}])$
$\wedge (t[R_i] = t_r[R_i](P_r^i) \overset{p}{\times} q(P_q))\}$,
if length of $P_r >$ length of P_q.
(b) $\{t|(\exists t_r \in r, t_q \in q), (t[(E_R - \{R_i\}) \circ (E_Q - \{Q_j\})] = t_r[E_R - \{R_i\}] \times^e$
$t_q[E_Q - \{Q_j\}]) \wedge (t[R_i Q_j] = t_r[R_i](P_r^i) \overset{p}{\times} t_q[Q_j](P_q^j))\}$,
if length of $P_r =$ length of P_q.
(c) $q(P_q) \overset{p}{\times} r(P_r)$,
if length of $P_r <$ length of P_q. □

Example 3.2 Let r and q be two relations of Figure 4(a) and (b) respectively. Figure 4(c) shows the path-dependent extended Cartesian product of $r(X \cdot Y)$ and $q(Z)$.

(a) r

A	X		
	B	Y	
		D	E
15	10	1	1
		2	2
25	20	3	2

(b) q

C	Z	
	F	G
20	2	2
	3	3
40	2	2

(c) $r(X \cdot Y) \overset{p}{\times} q(Z)$

A	X					
	B	C	YZ			
			D	E	F	G
15	10	20	1	1	2	2
			1	1	3	3
			2	2	2	2
			2	2	3	3
	10	40	1	1	2	2
			2	2	2	2
25	20	20	3	2	2	2
			3	2	3	3
	20	40	3	2	2	2

Figure 4: An example of path-dependent Cartesian product

3.2. P-join Operator with Single Join-path

The P-join operator with single join-path has been presented in [5]. In this section, we review the basic definitions and concepts regarding this new join operator.

Definition 3.4 We use $r(N_k)$ to denote $r(P_r)$ if the join-path $P_r = (N_1 \cdots N_k)$. N_k is called the *path-determining* node of P_r.

Suppose N_R and N_Q are the path-determining nodes of P_r and P_q respectively, then the scheme of relation $r(N_R) \overset{p}{\times} q(N_Q)$ is denoted by $R(N_R) \overset{p}{\times} Q(N_Q)$.

Natural P-join
The natural P-join, written $r(N_R) \bowtie^p q(N_Q)$, is applicable only when the same atomic attributes appear on the selection-comparable nodes in the scheme tree of $R(N_R) \overset{p}{\times} Q(N_Q)$.

Definition 3.5 The predicate $\Phi(T_R)$ is true when the condition $\{\forall N_a, N_b \in leaf(T_R)$, if N_a and N_b have same name then $(N_a \xrightarrow{\sigma} N_b) \vee (N_b \xrightarrow{\sigma} N_a)\}$ holds.

Definition 3.6 Let r and q be two relations with two join-paths P_r, P_q respectively. In the scheme $R(N_R) \overset{p}{\times} Q(N_Q)$, if (1) $A^1, ..., A^k$ are the names used for both schemes R and Q, and A_r^i, A_q^i have the same parent node, $1 \leq i \leq k$, (2) $B^1, ..., B^l$ are the names used for both R and Q and B_r^j, B_q^j have different parent nodes, $1 \leq j \leq l$, and (3) $\Phi(T_{R(N_R) \overset{p}{\times} Q(N_Q)})$, then

$$r(N_R) \bowtie^p q(N_Q) = \sigma_{\theta_B}(\pi_{\bar{X}}[\sigma_{\theta_A}(r(N_R) \overset{p}{\times} q(N_Q))]),$$

where N_R, N_Q are the path-determining nodes of paths P_r, P_q respectively and

θ_A is $(P^1 \cdot A_r^1 = P^1 \cdot A_q^1) \wedge (P^2 \cdot A_r^2 = P^2 \cdot A_q^2) \wedge \cdots \wedge (P^k \cdot A_r^k = P^k \cdot A_q^k)$, where P^i is the expression of the parent node of A^i,

$\bar{X}$ is the scheme of $r(N_R) \overset{p}{\times} q(N_Q)$ except that the components of $P^1 \cdot A_q^1$,...,$P^k \cdot A_q^k$,

θ_B is $(P_1^1 \cdot B_r^1 = P_2^1 \cdot B_q^1) \wedge (P_1^2 \cdot B_r^2 = P_2^2 \cdot B_q^2) \wedge \cdots \wedge (P_1^l \cdot B_r^l = P_2^l \cdot B_q^l)$, where P_1^i, P_2^i are the expressions of the parent nodes of B_r^i, B_q^i respectively. □

Example 3.3 Figure 5(c) shows an example of natural P-join between r and q, which are given in Figure 5(a),(b) where
$r(Y) \bowtie^p q(Z) = \sigma_{X \cdot B_r = B_q}\ (\pi_{A, X(B_r, C), B_q, YZ(D_r, E, F)}\ [\sigma_{YZ \cdot D_r = YZ \cdot D_q}(r(Y) \overset{p}{\times} q(Z))])$

3.3. Decomposition P-join Operator

Due to the fact that the same attribute names in two join relations may appear in multiple subtrees, we can extend P-join with multiple join-paths which exploits the more general situation. The following is our approach to extend P-join :

- Firstly, decompose each scheme into subschemes.

(a) r

A	X		Y	
	B	C	D	E
10	5	7	1	1
	15	8	2	2
			4	4
20	15	8	3	3
	15	9	4	4

(b) q

B	Z	
	D	F
5	1	1
	2	2
15	4	4

(c) $r(Y) \bowtie^p q(Z)$

A	X		B	YZ		
	B	C		D	E	F
10	5	7	5	1	1	1
				2	2	2
10	15	8	15	4	4	4
20	15	8	15	4	4	4
	15	9				

Figure 5: An example of natural P-join between r and q

- Select one subscheme from each relation to make pairs which contain the same attributes and satisfy P-join condition Φ, then apply P-join on each pair.
- Combine these joined relations from each pair and the remaining subrelations which correspond to the remaining subschemes.

Of course, the choice of join paths is highly dependent on the relation scheme structure and queries. We give the following example to illustrate this decomposition P-join concept.

Example 3.4 Figure 6 (c) shows an example of decomposition P-join between r and q, shown in Figure 6 (a), (b). The mechanism of computing this decomposition P-join is:

1. We decompose scheme R into two subschemes $R_1(A, X, Y)$, $R_2(A, U)$; scheme Q into two subschemes $Q_1(I, Y')$, $Q_2(I, V)$.
2. The two pair of subschemes (R_1, Q_1) and (R_2, Q_2) satisfy P-join condition $\Phi(T_{R_1(Y) \overset{p}{\times} Q_1(Y')})$ and $\Phi(T_{R_2(U) \overset{p}{\times} Q_2(V)})$ respectively. We apply P-join on each pair of subrelations $(r_1[A, X, Y], q_1[I, Y'])$, $(r_2[A, U], q_2[I, V])$.
3. Combine two joined relations using the standard natural join ie., $(r_1(Y) \bowtie^p q_1(Y')) \bowtie (r_2(U) \bowtie^p q_2(V))$.

Note that the schemes of relations of Figure 6 (a), (b) are similar to relations defined in Example 1.1, query Q1 of Example 1.1 can be expressed by multiple paths P-join.

The formal definition of the decomposition P-join follows.

Definition 3.7 Let $(R_1, ..., R_n)$ and $(Q_1, ..., Q_n)$ be lossless-join decompositions on schemes R and Q respectively. The intersection of all subschemes in each decomposition contains at least one common zero-order attribute. ie., $E_R = \cup_{i=1}^n R_i$; $E_Q = \cup_{i=1}^n Q_i$, and $\cap_{i=1}^n R_i = A_r$, $\cap_{i=1}^n Q_i = A_q$, where A_r, A_q are zero-order attributes of R, Q respectively. The decomposition P-join of two relations r and q is:

(a) r

A	X				Y		U
	B	C	Z		D	E	K
			I	J			
a_1	b_1	c_1	i_1	j_1	d_1	e_1	k_1
			i_2	j_2	d_2	e_2	k_2
	b_2	c_2	i_1	j_2			
a_2	b_2	c_2	i_2	j_2	d_1	e_1	k_2
					d_3	e_3	k_3

(b) q

I	Y′		V	
	D	F	K	G
i_1	d_1	f_1	k_1	g_1
	d_3	f_2	k_3	g_3
i_2	d_1	f_2	k_2	g_2
	d_3	f_3		

(c) $r(Y, U) \bowtie^p q(Y', V)$

A	I	X				YY′			UV	
		B	C	Z		D	E	F	K	G
				I	J					
a_1	i_1	b_1	c_1	i_1	j_1	d_1	e_1	f_1	k_1	g_1
		b_2	c_2	i_1	j_2					
a_1	i_2	b_1	c_1	i_2	j_2	d_1	e_1	f_2	k_2	g_2
a_2	i_2	b_2	c_2	i_2	j_2	d_1	e_1	f_2	k_2	g_2
						d_3	e_3	f_3		

Figure 6: An example of decomposition P-join between r and q

$\forall r \in I_R$, $\forall q \in I_Q$, $L_r = (N_1^r, ..., N_n^r)$ and $L_q = (N_1^q, ..., N_n^q)$ are lists of join-paths of r and q respectively. If $\Phi(T_{R_i(N_i^r) \overset{p}{\times} Q_i(N_i^q)})$, $\forall i \in \{1, ..., n\}$, then

$$r(L_r) \bowtie^p q(L_q) = \sigma_\theta[\bowtie (\mathcal{J}_1, ..., \mathcal{J}_n)],$$

where $\mathcal{J}_i = r[R_i](N_i^r) \bowtie^p q[Q_i](N_i^q)$, $1 \leq i \leq n$, $\bowtie$ is the standard natural join, θ is the predicate, with "=" comparison operator, on those selection-comparable nodes with the same attribute names in the scheme tree of $\bowtie (\mathcal{J}_1, ..., \mathcal{J}_n)$.

4. Query Optimization

Query optimization is an important issue in any database system since a good strategy can make query processing faster. In this section we list some algebraic equivalences of the P-join operator. These properties are very useful for query optimization in the nested relational model. We first present the main results of algebraic equivalences. Due to space limitations we omit the proofs. Note that every new node $N_{R_iQ_j}$ corresponding to a new attribute name R_iQ_j as defined in definition 3.3 is called a *joined* node.

Algebraic Equivalences

- *Cascade of* σ:
  $\sigma_{\theta_1}(\sigma_{\theta_2} r) = \sigma_{\theta_1 \wedge \theta_2}(r)$

- *Commutativity law*:
  $r(L_r) \bowtie^p s(L_s) = s(L_s) \bowtie^p r(L_r)$

- *Associativity law*:
  Suppose $L_r = (N_1^r, ..., N_l^r)$, $L_s = (N_1^s, ..., N_l^s)$ are lists of join-paths of r, s respectively; $L_{rs} = (N_1^{rs}, ..., N_o^{rs})$, $L_q = (N_1^q, ..., N_o^q)$ are lists of join-paths of $r(L_r) \bowtie^p s(L_s)$, q respectively. We assume $1 \leq k < m < n < o$ and $m \leq l$. If

  (a) $N_1^{rs}, ..., N_k^{rs}$ are joined nodes and $N_i^{rs} = N_i^r N_i^s$, $1 \leq i \leq k < l$

  (b) $N_{k+1}^{rs}, ..., N_m^{rs}$ are joined nodes and $N_i^{rs} = \tilde{N}_i^r \tilde{N}_i^s$, $k+1 \leq i \leq m \leq l$, $N_i^{rs} \neq N_j^r N_j^s$, $1 \leq j \leq l$,

  (c) $N_{m+1}^{rs}, ..., N_n^{rs}$ are not joined nodes and all $N_i^{rs} \in r$, $m+1 \leq i \leq n$

  (d) $N_{n+1}^{rs}, ..., N_o^{rs}$ are not joined nodes and all $N_i^{rs} \in s$, $n+1 \leq i \leq o$, then

  $$[r(L_r) \bowtie^p s(L_s)](L_{rs}) \bowtie^p q(L_q) = [r(L_r^{'}) \bowtie^p q(L_q^{'})](L_{rq}) \bowtie^p s(L_s^{'})$$

  where $L_r^{'} = (N_1^r, ..., N_k^r, \tilde{N}_{k+1}^r, ..., \tilde{N}_m^r, N_{m+1}^{rs}, ..., N_n^{rs})$; $\quad L_q^{'} = (N_1^q, ..., N_n^q)$;
  $L_{rq} = (N_1^r N_1^q, ..., N_k^r N_k^q, N_{k+1}^r, ..., N_l^r, N_{n+1}^q, ..., N_o^q)$;
  $L_s^{'} = (N_1^s, ..., N_l^s, N_{n+1}^{rs}, ..., N_o^{rs})$.

  Due to the fact that we can reorder the path determining nodes according to a corresponding one-to-one mapping in the two join-path lists without changing the result of the P-join, we assume, for simplicity, all joined nodes in the above equivalence are in consecutive order. The equivalence can be similarly applied to joined nodes in the case of arbitrary order .

- *Commuting a projection with P-join*:
  $\pi_L[r(L_r) \bowtie^p s(L_s)] = \pi_L[\pi_{L_1}(r)(L_r) \bowtie^p \pi_{L_2}(s)(L_s)]$
  if L can be split into L_1 and L_2 such that they contain attributes of r and s in L respectively, and they each contain all common attributes involved in the join.

- *Distributivity of unnest over P-join*:
  Suppose $L_r = (N_1^r, ..., N_l^r)$, $L_s = (N_1^s, ..., N_l^s)$ are the lists of join-paths of r and q respectively.

  (a) Let X be not a joined node.
  $\mu_X(r(L_r) \bowtie^p s(L_s)) =$
  (1) $\mu_X(r)(L_r) \bowtie^p s(L_s)$, if $X \in E_R$.
  (2) $r(L_r) \bowtie^p \mu_X(s)(L_s)$, if $X \in E_S$.

  (b) Let X be a joined node.
  $\mu_X(r(L_r) \bowtie^p s(L_s)) =$
  (1) $\mu_{\tilde{X}^r}(r)(L_r) \bowtie^p \mu_{\tilde{X}^s}(s)(L_s)$, if $X = \tilde{X}^r \tilde{X}^s \neq N_i^r N_i^s$, $1 \leq i \leq l$.
  (2) $\mu_{N_i^r}(r)(L_r^{'}) \bowtie^p \mu_{N_i^s}(s)(L_s^{'})$, if $X = N_i^r N_i^s$, $i \in \{1, ..., l\}$,
  where $L_r^{'} = (N_1^r, ..., N_{i-1}^r, N_{i+1}^r, ..., N_l^r)$; $L_s^{'} = (N_1^s, ..., N_{i-1}^s, N_{i+1}^s, ..., N_l^r)$.

We give an example which shows how an expression can be derived from another expression using the associativity of P-join. This can be used to derive more efficient expressions.

Example 4.1
Let r,s and q be three relations with schemes $R = (A, X(B,C), Y(E,F), Z(I,J))$, $S = (A, X^{'}(B,D),\ Y^{'}(E,G), U^{'}(M,P))$, and $Q = (A, X^{''}(C,D), Y^{''}(E), Z^{''}(I,K), U^{''}(P))$. Consider the following nested relational algebraic expression, which includes P-join:

$$[r(L_r) \bowtie^p s(L_s)](L_{rs}) \bowtie^p q(L_q) \text{——(1)}$$

where $L_r = (X, Y)$, $L_s = (X^{'}, Y^{'})$, $L_{rs} = (XX^{'}, YY^{'}, Z, U^{'})$, $L_q = (X^{''}, Y^{''}, Z^{''}, U^{''})$.

If q is likely to have far less matches with r compared to s, then reordering the operands s and q will be more efficient than evaluating expression (1). We can easily derive the following equivalent expression (2) from expression (1) by applying the associative law.

$$[r(L_r^{'}) \bowtie^p q(L_q^{'})](L_{rq}) \bowtie^p s(L_s^{'}) \text{——(2)}$$

where $L_r^{'} = (X, Y, Z)$, $L_q^{'} = (X^{''}, Y^{''}, Z^{''})$, $L_{rq} = (XX^{''}, YY^{''}, U^{''})$, $L_s^{'} = (X^{'}, Y^{'}, U^{'})$.

5. Correctness of decomposition P-Join

In this section we adapt the criteria, defined by Roth et al. [9], to establish the correctness of our P-join for nested relations. The criteria for correctness of an extended operator is that it is faithful and precise.

We state the formal definition of faithfulness from [9].
Definition 5.1 Let P and $P^{'}$ be classes of relations and ψ and $\psi^{'}$ binary operators on P and $P \cup P^{'}$ respectively. We say that $\psi^{'}$ is faithful to ψ if $r\psi^{'}q = r\psi q$ for every $r, q \in P$ for which $r\psi q$ is defined. □

Proposition 1 P-join is faithful to standard natural join.
Proof: By definition, extended natural join ($\bowtie^e$) is P-join with no specified join-path. So P-join is faithful to extended natural join. Since extended natural join is faithful to standard natural join [9], so P-join is faithful to standard natural join.

The following definition of preciseness is from [9].
Definition 5.2 Let P and $P^{'}$ be classes of relations and ψ, $\psi^{'}$ binary operators on P and $P^{'}$ respectively. Let α, β be operators on $P \cup P^{'}$. We say that $\psi^{'}$ is a precise generalization of ψ relative to α, β if one of the following two conditions holds: (1) $\alpha(\beta(r\psi^{'}q)) = \alpha(\beta(r))\psi\alpha(\beta(q))$ for every $r, q \in P^{'}$ for which $r\psi^{'}q$ is defined. (2)

$\beta(\alpha(r\psi' q)) = \beta(\alpha(r))\psi\beta(\alpha(q))$ for every $r, q \in P'$ for which $r\psi' q$ is defined. □

Let ξ be the duplicate attribute elimination operator. We prove that P-join is precise as follows.

Proposition 2 P-join is a precise generalization of the standard natural join with respect to unnesting and the function ξ i.e., $\xi(\mu^*(r(L_r) \bowtie^p q(L_q))) = \xi(\mu^*(r)) \bowtie \xi(\mu^*(q))$ for every r, q for which $r(L_r) \bowtie^p q(L_q)$ is defined, where L_r, L_q are the lists of join-paths of r, q respectively.

Proof: We show inclusion both ways.

- $\subseteq$: Let $t \in LHS$. There is a tuple $\hat{t} \in \mu^*(r(L_r) \bowtie^p q(L_q))$ such that $t = \xi(\hat{t})$. Also, there must be a tuple $u \in (r(L_r) \bowtie^p q(L_q))$ such that $\hat{t} \in \mu^*(u)$. Since $u \in (r(L_r) \bowtie^p q(L_q))$, there exist $t_r \in r$ and $t_q \in q$ such that $u = t_r(L_{t_r}) \bowtie^p t_q(L_{t_q})$, where $L_{t_r} = L_r$; $L_{t_q} = L_q$. By $\bowtie^p$ definition, u must have identical values on those selection-comparable nodes with the same attribute names. So $\hat{t}$ has identical values on those duplicate attributes. Therefore $t = \xi(\hat{t}) \in \xi(\mu^*(u)) = \mu^*(t_r) \bowtie \mu^*(t_q) \subseteq \mu^*(r) \bowtie \mu^*(q) = \xi(\mu^*(r)) \bowtie \xi(\mu^*(q))$=$RHS$. (by distributivity of unnest over P-join illustrated in Section 4)

- $\supseteq$: Let $t \in RHS$. Since $\xi(\mu^*(r)) = \mu^*(r)$ and $\xi(\mu^*(q)) = \mu^*(q)$, so $t \in \mu^*(r) \bowtie \mu^*(q)$. t must be the natural join of some t_1 in $\mu^*(r)$ and some t_2 in $\mu^*(q)$. Also, t_1 was unnested from some t_r in r and t_2 was unnested from some t_q in q. In the join ($\bowtie^p$) of $r(L_r)$ and $q(L_q)$, t_r and t_q will combine to produce w. By $\bowtie^p$ definition, in w, the same attribute names can appear only on those selection-comparable nodes and have identical values. So $t = t_1 \bowtie t_2 \in \xi(\mu^*(t_r \bowtie^p t_q)) = \xi(\mu^*(w))$. Since $w = t_r(L_{t_r}) \bowtie^p t_q(L_{t_q}) \in r(L_r) \bowtie^p q(L_q)$, we have $t \in \xi(\mu^*(r(L_r) \bowtie^p q(L_q)))$=$LHS$.

With these results (propositions 1 and 2) we conclude that P-join is correct for every $r \in I_R, q \in I_Q$ for which $r(L_r) \bowtie^p q(L_q)$ is defined.

6. Conclusion

The nested relational model provides a better way to represent complex objects than the traditional relational model, and allows users to describe their concepts of real world data objects more easily. However, the optimization of queries is quite different from that in the 1NF relational model. Most join operators that have been proposed for this model cannot be used directly to express queries without restructuring operators. In this paper, we have proposed the P-join operator with multiple join-paths, which allows queries including the join operation to be expressed succinctly, without (or with fewer) restructuring operators. Most queries in extended SQL (e.g. SQL/NF [7], TQL [13]) can be written naturally in terms of P-joins and therefore can be optimized more efficiently, since fewer nest and unnest operators are needed.

A series of algebraic equivalences have also been presented, which are useful for query optimization in the nested relational model. We hope the theoretical results

obtained for optimization of nested relational algebra can be carried over to an object algebra.

We have proved the correctness of P-join using the criteria of faithfulness and precision of generalization. This generalized P-join operator has properties that the standard join operator possesses, such as commutativity and associativity. Compared to other joins, the P-join is more powerful in terms of expression and optimization.

This paper offers a solution to the problem of how to efficiently express and optimize queries which include join operations in the nested relational model. Future work involves the design of algorithms for query optimization in nested relational databases, and the formulation of a performance analysis for physical implementation.

Acknowledgment

We thank Gill Dobbie for reading and helpful corrections of this paper.

References

[1] L.S. Colby. A Recursive Algebra and Query Optimization for Nested Relations. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, pages 273-283, 1989.

[2] Y. Jan. Algebraic Optimization for Nested Relations. In *Proceedings of the 23rd Hawaii International Conference on System Sciences,* Vol.2, pages 278-287, 1990.

[3] H.F. Korth. Optimization of Object-Retrieval Queries. In *Proceedings of the 2nd International Workshop on Object-Oriented Database Systems,* pages 352-357, 1988.

[4] H.C. Liu and K. Ramamohanarao. Equivalences of Nested Relational Operators. In *Proceedings of the 3rd Australian Database Conference*, pages 124-138, 1992.

[5] H.C. Liu and K. Ramamohanarao. Path-dependent Nested Relational Algebra. *Technical Report* 93/13, Department of Computer Science, The University of Melbourne, Australia, 1993.

[6] H.C. Liu and K. Ramamohanarao. Multiple Path Join for Nested Relational Databases. *Technical Report*, Department of Computer Science, The University of Melbourne, Australia, 1993.

[7] M.A. Roth, H.F. Korth and D.S. Batory. SQL/NF: A Query Language for ¬1NF Relational Databases. *Information Systems,* 12(1):99-114, 1987.

[8] M.A. Roth, H.F. Korth and A. Silberschatz. Extended Algebra and Calculus for ¬1NF Relational Databases. *ACM Transactions on Database Systems*, 13(4):389-417, 1988.

[9] M.A. Roth, H.F. Korth and A. Silberschatz. Null Values in Nested Relational Databases. *Acta Informatica* 26, pages 615-642, 1989.

[10] R. Sacks-Davis, A. Kent, K. Ramamohanarao, J. Thom, J. Zobel. ATLAS: A Nested Relational Database System for Text Applications. *Technical Report* TR-92-52, Collaborative Information Technology Research Institute, Melbourne, Australia, 1992. To appear in *IEEE Transactions on Knowledge and Data Engineering.*

[11] M.H. Scholl. Theoretical Foundation of Algebraic Optimization Utilizing Unnormalized Relations. In *Proceedings of the International Conference on Database Theory,* pages 380-396, 1986.

[12] M.H. Scholl and H.J. Schek. The Relational Object Model. In *Proceedings of International Conference on Database Theory,* pages 89-105, 1990.

[13] J.A. Thom, A. Kent and R. Sacks-Davis. TQL: Tutorial and User Manual, *Technical Report* TR-92-19, Collaborative Information Technology Research Institute, Melbourne, Australia, 1992.

[14] S.J. Thomas and P.C. Fischer. Nested Relational Structures. In P.C. Kanellakis, editor, *Advances in Computing Research* 3, JAI press, pages 269-307, 1986.

[15] J.D. Ullman. Principles of Database and Knowledge-base Systems. Computer Science Press, Vol.2, 1989.