An Efficient Join for Nested Relational Databases

Hong-Chen Liu and Chaiyaporn Chirathamjaree

Department of Computer Science
Edith Cowan University
2 Bradford Street, Mount Lawley
W.A. 6050 Australia

Abstract

The join operation is one of the most expensive and critical issues in nested relational query processing. Many natural queries cannot be expressed by extended join operators proposed for the nested relational model so far without restructuring operations. In this paper, we consider a more general form of join, called P-join, which does not require as many restructuring operators and combines the advantages of the extended natural join and the recursive join for efficient data access. We propose an algorithm for computing the P-join and estimate cost required by using various join techniques developed in relational database systems. The complexity of the P-join algorithm is not more than other join algorithms with expensive restructuring operators involved and additional block shuffle for reading unnecessary data files.

1. Introduction

By relaxing the first normal form (1NF) assumption, the resulting nested relational model can support such new application as office automation, multimedia systems, scientific data processing systems, engineering design systems, and so forth. In the last decade, much research has been carried out on nested relations and complex object databases. This strand of research has been maturing, and as a result systems based on nested relations and complex objects have been developed. However, query optimisation and processing for the nested relational model is still an area of research in the database community.

The object-oriented database shows great promise in serving as a model for next generation database systems [3,14]. The nested relational model provides the structural core of object-oriented databases. Since strong similarities exist between the query languages proposed so far for object-oriented models and the nested relational model, and complex structures are an important component of object-oriented models, query languages that manipulate hierarchical structure are worthy of study [12,20]. This motivates us to investigate the issue of query optimisation and processing in nested relational databases.

The join operation is one of the most frequently executed and expensive query processing operations in relational databases. The nested relational model provides a better way to represent complex objects than the 1NF relational model. The information about complex objects will not be distributed over several relations in the nested relational model, and many frequent join operations (as is necessary under the 1NF relational model) can be avoided. However, no one storage structure will adequately serve all query types, hence join operation is still necessary in certain types of queries in nested relational databases. It is meaningful to study how join operation techniques used in relational databases can be extended to nested relational database systems. In this paper, we will focus on the issue of how to efficiently process join operation in nested relational databases.
The rest of this paper is organised as follows. In section 2, we briefly review the P-join operator. Section 3 will establish a cost model and propose a mechanism of computing the P-join. In section 4, we will present analysis on various methods for computing the P-join based on our cost model. In section 5, we propose a generalised P-join algorithm. The brief discussion and main advantages of the P-join are presented in section 6. Finally, we will draw conclusions in section 7.

2. Basic Concepts and Definitions

In this section, we first briefly review some well-known concepts and basic definitions of the nested relational model used throughout this paper. Then we review the P-join operator which has been proposed in [15].

2.1 Nested relational Scheme

A nested relational scheme is structured as a rooted tree in which the nodes are labelled with attribute names. Such a tree is called a scheme tree. We write \( T_R \) to represent the scheme tree of the relation \( R \). The set of nodes of a scheme tree \( T \) is denoted by \( \text{node}(T) \). The set of leaf nodes of a scheme tree \( T \) is denoted by \( \text{leaf}(T) \).

In a scheme tree, we consider nodes to be comparable if a comparison between them can be computed using a single selection scan. We define the concept of selection-comparable nodes as follows.

**Definition 2.1** For all nodes \( N_a, N_b \in \text{node}(T) \), where \( N_a \) and \( N_b \) with the same scheme structure and \( N_a \neq N_b \), 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 \( N_a \to N_b \).

For example, in the scheme of the relation \( R(A, B, X(C, Z(D)), Y(E, F)) \), \( B \to D \), \( A \to C \) but \( C \) and \( E \); \( X \) and \( Y \) are not selection-comparable nodes.

We need recursive definition of selection and projection operators which can work on nested relations [5,18]. In particular, we extend the selection operator to apply to selection-comparable nodes. The comparison operators include set comparison and set membership operators, in addition to the usual arithmetic operators.

2.2 Extended Cartesian Product and Path-dependent Extended Cartesian Product

We will now introduce extended Cartesian product and path-dependent extended Cartesian product which are used in the P-join definition described in the next subsection.

The idea behind the extended Cartesian product, \( \times' \), 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. For example, let \( r \) and \( q \) be two relations with schemes \((A, X(B, C), Y(D))\) and \((E, X(B, C), Y(D))\) respectively. The scheme of the extended Cartesian product of \( r \) and \( q \) is \((A, E, X(B_r, C_r, B_q, C_q), Y(D_r, D_q))\). We distinguish between the two Bs with suffixes \( B_r, B_q \); two Cs with suffixes \( C_r, C_q \).

Extended Cartesian Product provides the base domain for the extended natural join operator [18]. This is like Cartesian product which provides a base domain for the natural join operator.

---

1. Two nodes with the same scheme structure mean that they have the same scheme tree structure but may have different attribute names.