Query Unnesting in Object Oriented Database

Bharti Joshi[1], Rostom D Morena[2]

[1]Dept of Information Technology,

Saraswati College of Engineering, Kharghar, Navi Mumbai

[2]Dept of Computer Science and Technology,

Veer Narmad South Gujarat University, Surat, Gujarat

bhartijoshi1@rediffmail.com, rdmorena@rediffmail.com

Abstract: *Both relational and Object-oriented query languages allow nested queries (queries with subqueries). Nested queries are typically inefficient to execute in their original form and are difficult to optimize. Most existing query Unnesting techniques are based on source-to source, query graph, or calculus transformations. Few are based on algebraic transformations. Algebraic Unnesting is desirable for cost-based algebraic optimization, because it allows such optimizers to be more readily extended with Unnesting capabilities. This paper concentrates on a new algorithm for the execution of nested queries , which covers unnesting techniques for projection, predicate and range dependency which commonly occurs in OODBMS.*

Keywords: *Object-Oriented database, Nested Query, O-algebra, Monoid Calculus, Unnesting queries, Access Support Relations.*

I. Introduction

Object-oriented database systems, which can be considered fifth-generation database technology, began developing in the mid-80's out of a necessity to meet the requirements of applications beyond the data processing applications, which characterized relational database systems (fourth-generation database technology). Attempts to use relational database technology for advanced applications like computer aided design (CAD), computer aided manufacturing (CAM), software engineering, knowledge-based systems, and multimedia systems, quickly exposed the shortcomings of relational database systems[5], [6]. The need to perform complex manipulations on existing databases and a new generation of database applications generated a need that would be better satisfied by object-oriented databases (OODBs).Many definitions of object orientation and object-oriented databases have been developed over the years [7], but we will define object oriented databases as databases that integrate object orientation with database capabilities. Object orientation allows a more direct representation and modelling of real world problems, and database functionality is needed to ensure persistence and concurrent sharing of information in applications.

Today there are over 25 object-oriented database products on the market, including, GemStone from Servio Corporation, ONTOS from ONTOS, ObjectStore from Object Design, Inc., and many others [8]. In addition, relational database management systems from Oracle, Microsoft, Borland, Informix, and others have incorporated object-oriented features into their relational systems. A lot of these products have been around since the mid to late ‘80’s, and after almost one and half decades of development, the lack of maturity of a lot of these products has contributed to the slow acceptance of OODBs into the real worldwide market of today. Most current OODBs are still not full-fledged database systems comparable to current relational database systems (RDBs) [8].

Comparing the relational model with an object-oriented model shows that the latter is more powerful in the modelling stage but does not support a standard formal query model. While the non-atomic domain concept is supported by the nested relational model, while simple message expressions make object-oriented systems superior to the relational model, an object-oriented query language is still needed for more complex situations and to support associative access [9].

One essential feature of declarative query languages is the nesting of queries. In SQL, nested queries are used to express complex conditions. This is also true in object-oriented SQL like query languages. But object oriented nested queries serve other purposes as well: They are used to access nested structures and to produce nested result. There are many recent proposals on OODB query optimization that are focused on unnesting nested queries. Query unnesting appears more often in OODB queries than in relational queries, because OODB query languages allow complex expressions at any point in a query. Current OODB systems evaluate nested queries in a nested loop fashion, which does not leave many opportunities for optimization. Most unnesting techniques for OODB queries are actually based on similar techniques for relational queries. If considered in isolation, query unnesting itself does not result in performance improvement; instead, it makes possible other optimizations, which would not be possible otherwise.

II. Related Work:

The purpose of this section is familiarizing the reader briefly with past work.

A) Jie Lin O-Algebra:

Jie Lin used an Object algebra, called O-algebra, designed for processing OODB queries[1]. The operands of O-algebra are collections of objects of the same internal type. Each object has the form (o,v) where o is an object id and v is a tuple of properties.. In present method only one level of nesting is allowed. However, tuple components can be sets but not sets of sets. The front end bulk data type objects are mapped to internal objects with one bulk data type property and the front end complex objects can be decomposed to objects with atmost one level nesting. Due to uniformity of operands and the simplicity of operators, a powerful set of algebraic laws can be obtained.

Lin used procedures to convert OQL queries to O-Algebra queries. Usually in OODB queries, bulk data (set, list and bag) is commonly constructed and the constructions are expressed by nested queries. He presented a general solution to reduce nested queries in OODB queries. As

O-Algebra does not allow nested sub expressions, when OQL queries are converted to O-Algebra queries, nested queries are already reduced.

Lin introduced an SQL –like intermediate language, called SOQL, which serves as a bridge for facilitating the transformation from OQL to O-Algebra. SOQL does not have any nested queries. A given OQL query can always be converted into this form by name substituting. In OQL, the nested queries can appear in any clauses. SOQL does not allow nested queries. Instead it represents sub queries by methods. For nested queries in OQL, it is transformed in to a method of SOQL and a method call is generated in main query. Notice that the methods in SOQL are independent queries. Therefore when the OQL queries are translated into the SOQL queries, the nested queries have actually been reduced.

This approach is more general because it is not restricted by the patterns of nested queries.

B) Fegaras Monoid Algebra:

Fegaras presented a new query unnesting algorithm that generalizes many unnesting techniques. Method proposed by Fegaras is capable of removing any form of query nesting. He has utilized monoid comprehension calculus for removing queries. Fegaras claims that his algorithm unnests the queries that cannot be normalized by the normalization algorithm using two rewrite rules only. Queries are translated into monoid comprehensions, which serve as an intermediate form, and then translated into version of the nested relational algebra that supports aggregation, quantification, outer joins and outer unnestings. Query unnesting is performed during the translation of monoid comprehensions into algebraic forms. He used both calculus and algebras as an intermediate forms because the calculus resembles current OODB languages and easy to normalize, while the algebra is lower level and can be directly translated into the execution algorithms.

Fegaras unnesting algorithm is quite simple. For each box that corresponds to a nested query it converts reductions into nests, joins into outer joins and unnest into outer unnest. At the same time it embeds the resulting boxes at the points immediately before they are used. He has given various transformation rules for unnesting OODB queries by translating terms in monoid calculus.

C) Horng ASR Technique**:**

He has used query processing architecture. In this the declarative query expression is first parsed to internal form which is composed of parsing trees and several tables such as variable table, path expression table, class table etc..The internal form is then translated into a query graph by Query Graph Generator. Execution plan Generator is the process of query mapping query graph to a sequence of data manipulation operators. Finally, the Runtime Database Executor runs the execute plan to generate query results. He proposed an algorithm to transform a query Expression into a query graph. Queries are expressed in OQL. The idea of the transformation is based on the internal storage structure, Access support Relation. He mapped a path expression in a query to an ASR storage structure. A path expression maps to a combination of Simple nodes and Navigate-Links. A path decides the order of execution of navigation. Access Support Relations techniques are introduced as a means for optimizing query processing in OODBMS. The general idea is to maintain separate Structures to redundantly store those object references that are frequently traversed in database queries. He defined a set of operators to manipulate ASRs in the query processing. These operators are I/O Operators, Basic Operators and Extended operators All and Exist. Execution plan generation is the process of mapping a Query graph to a set of operators. Above method can handle large amount path expressions that exist in an object oriented queries.

III. Our Algorithm for Processing a nested query:

We distinguish three kinds of dependencies in OODBMS

Projection dependency (a reference to an outer variables occurs in the select clause)

Range dependency(a reference to an outer variables occurs in the from clause)

Predicate dependency(a reference to an outer variables occurs in the where clause)

Our algorithm is based on cluet [3] classification of nested queries. His classification extends the relational classification of which four types of nested queries are more added.

Type A nested queries have a constant inner block that returns a single element.

Type N nested queries have a constant inner block that returns a set.

Type J nested queries have an inner block that is dependent on the outer block and returns a set.

Type JA nested queries have an inner block that is dependent on the outer block and returns a single element.

Our algorithm can unnest any type of nested queries.

```
Procedure Unnest(query_block)
{   for each predicate in the WHERE clause of query_block    if predicate is a nested predicate(i.e contains inner query block)

Unnest(inner_query_block)
  { //Determine type of nesting (A,  N, J and  JA)
  if SELECT clause of inner_query_block contains join predicate referencing a relation which is  not in it's
        FROM  clause
      //Nesting is type_JA or type J
      unnest_JA(query_block, inner_query_block)
      { these queries are unnested using  binary  grouping operator or outer join followed by  a unary grouping operator. queries have membership predicates can be evaluated using semi_joins and anti_joins.
      Queries have inner and outer block  have  common domain can be evaluated  simple  grouping
      }
else
   //nesting is type_A or N
    unnest_A(inner_query_block)
      { can be unnested using constant  factroization. Using temp variable inner query can be evaluated
      independently and this value is substituted in outer block}
      for each predicate in the SELECT clause of query_block if predicate is a nested predicate(i.e contains inner query block)
    // type A or N rarely occurs in SELECT  Clause
    Reduce the range dependency to predicate dependency  using type
based rewriting
    Then call  Unnest_JA(query_block,inner_query_block)
    for each predicate in the FROM clause of query_block
     if predicate is a nested predicate(i.e contains inner query block)
        // this type of nesting rarely occurs can be unnest using map operation, nested
loops and cross products.
return
}
```

In explaining procedure unnest (), it is useful to model a nested query with a multi-way tree whose nodes are query blocks, where the outermost query block is the root and the innermost query blocks are the leaves. Procedure unnest () searches down through the levels of a nested query from the outermost query block until it finds the innermost query blocks. It then examines the leaf block to determine the type of nesting present, and transforms the parent to canonical form by calling the appropriate transformation procedures. After this is done for all nested predicates in query_block, the recursion then unwinds one level and the query block immediately above is processed in the same way, continuing the unwinding until lastly the outermost, or root, query block is transformed. The algorithm represented in procedure unnest () solves the problem of correctly transforming a type-JA query with multiple levels of nesting.

Predicate dependency is very popular in RDBMS. In OODBMS any three dependency can occur. To illustrate projection dependency lets see example

```
Select  tuple(dept: d,
                  Emps: select e
                        From   e in Employee
                        Where e.dept=d)
 From d in department
After  rewriting
    Select      tuple(dept: d, emps:des)
    From       d in Department
```

```
Define    des=select  e
            From  e in employee
             Where ed=d
            Define  ed=e.dept)
```

Now this can be easily converted into algebraic form same as predicate dependency using our algorithm.

IV. Conclusion and future scope :

In this paper, we have presented a new algorithm that enhances and incorporates the previously known techniques for Unnesting various types of queries. It appears that we can unnest the various OODBMS predicates using the techniques presented in this paper.
We are in process of implementing our algorithm in C#.NET.

References

[1] J. Lin and Z M Ozsoyogulu,” Processing OODB queries by O-Algebra” Technical report, Case western Reserve University, Computer Eng., and Science Department,1995.

[2]. J. T. Horng, J. C. Wang,J.T. Lin and B,J, Liu,” Nested Query Processing Techniques in Object Oriented Databases”,Proceeding of International Conference on Distributed Systems, Software Engineering and Database Systems, Inst of Computer Sc and Engineering Taiwan1996.

[3]. S. Cluet and G. Moerkotte,” Nested in object bases”,In Fifth International Workshop on database Programming Languages, gubbio,Italy,September 1995

[4]. Leonidas Fegaras,”Query Unnesting in Object Oriented Databases”(extended version) Available at http://www-cse.uta.edu/~fegaras/sigmoid98.ps, January 1998.

[5]. Khoshafian, S., “Insight Into Object-Oriented Databases,” *Information and Software Technology*, vol. 32, no.4, 1990.

[6]. Bertino, E., Negri, M., Pelagatti, G., and Sbattella, L., “Object-Oriented Query Languages: The Notion and the Issues,” *IEEE Transactions on Knowledge and Data Engineering,* Vol. 4, No. 3, 1992

[7]. Atkinson, M., Bancilhon, F., DeWitt, D., Dittrich, K., Maier, D, and Zdonik, S."The Object-Oriented Database System Manifesto," in Bancilhon et. al. (eds.)*Building an Object-Oriented Database System: The Story of O2*. Morgan Kaufman, 1992.

[8]. Kim, W., "Object-Oriented Database Systems: Promises, Reality, and Future," in *Modern Database Systems: The Object Model, Interoperability and Beyond*.

[9]. Reda Alhajj,"An Object-Oriented Query Model: An Algebraic Approach with Closure." Journal of Information Science and Engineering 16, 499-533(2000)

* * * * *