Optimization of Queries using Nested Indices

Elisa Bertino*
Istituto di Elaborazione dell'Informazione
Via S.Maria 46 - 56100 Pisa (Italy)

Abstract
The notion of nested object is a basic concept of the object-oriented paradigm. It allows the value of an object attribute to be another object or a set of other objects. This means that a class consists of a set of attributes, and the values of the attributes are objects that belong to other classes; that is, the definition of a class forms a hierarchy of classes. All attributes of the nested classes are nested attributes of the root of the hierarchy. In a previous paper [Bert 89], we have introduced the notion of nested index that associates the values of a nested attribute with the objects instances of the root of the hierarchy. In that paper, we have evaluated the performance of this indexing mechanism in the case of queries containing a single predicate. In the present paper, we consider the usage of nested indices in the framework of more general queries containing several predicates.

1 Introduction
Object-oriented databases have recently emerged as the new database generation. Their overall goal is to provide data management facilities for advanced applications, such as CAD/CAM, software engineering, office automation, that need handling objects more complex than tuples of relations. This has resulted in the development of several object-oriented DBMSs. These systems, because of the increased complexity of the data model, have had to address new issues and requirements in the design and analysis of suitable access mechanisms [Ditt 86]. In fact, to be viable an object-oriented approach to data management must be supported by an architecture that directly implements the basic concepts of the object-oriented paradigm.

One important element in the object-oriented paradigm is the view that the value of an attribute of an object is an object or a set of objects. A class C consists of a number of attributes, and the value of an attribute A of an object belonging to the class C is an object or a set of objects belonging to some other class C'. The class C' is the domain of the attribute A of the class C. The class C' in turn consists of a number of attributes, and their domains are other classes. In other words, in object-oriented databases a class is in general a hierarchy of classes. This hierarchy is called a class-attribute hierarchy in [Kim 88]. Figure 1 is an example class-attribute hierarchy; the hierarchy is rooted at the class Vehicle, and the * symbol next to an attribute indicates that the attribute is multi-valued. An attribute of any class in a class-attribute hierarchy is logically an attribute of the root of the hierarchy, that is, the attribute is a nested attribute of the root class. For example, in Figure 1, the Location attribute of the class Division is a nested attribute of the class Vehicle.

In normalized relational databases, the search conditions in a query are expressed as a boolean combination of predicates of the form <attribute operator value>. The query is directed to one or more relations, and the attribute specified in a predicate is an attribute of one of the relations.

*This work was partially performed while the author was visiting the Microelectronics and Computer Technology Corporation, Austin (Texas)
In object-oriented databases, the search conditions in a query on a class can still be expressed as a boolean combination of predicates of the form <attribute operator value>. However, the attribute may be a nested attribute of the class. For example, the following is a query that may be issued against the class Vehicle as defined in the class-attribute hierarchy of Figure 1.

*Retrieve all 2-doored red vehicles manufactured by Fiat* (Q1).

The query consists of a predicate against the non-nested attribute Color, and two predicates against the nested attributes Name and Doors. We call a predicate on a nested attribute a nested predicate and a predicate on a non-nested attribute a simple predicate. An important issue in supporting queries in object-oriented databases is the efficient evaluation of queries involving nested predicates.

Given a branch, or path, in a class-attribute hierarchy a nested index provides a direct association between the objects (or values) at the end of the path and the objects instances of the class which is the root of the path. For example, a nested index on the path P=Vehicle.Manufacturer.Name will associate a distinct value of the Name attribute with a list of object identifiers of Vehicle whose Manufacturer is an instance of the Company class whose Name is the key value. For the objects shown in Figure 2, the nested index will contain the following entries:

(Fiat, {Vehicle[i], Vehicle[j]}) (Renault, {Vehicle[k]}).

The attribute Name is the ending attribute of path P.

In [Bert 89], the performance and update costs of the nested index have been evaluated and compared with the costs of other organizations. The nested index has the best retrieval performance, while the update costs may be quite high depending on how the updates are distributed on the classes along the path. An overall conclusion was that the nested index organization can be convenient, compared to others, for path lengths 2 or 3 provided that reverse references among objects are supported. For greater path lengths, however, it is better to split a path into several subpaths and allocate a nested index on each path if the frequency of updates is not negligible.

In [Bert 89], however, the performance of the nested index has been evaluated in the case of queries containing a single predicate. In the present paper, we provide an analysis of the usage of the nested index when queries contain several predicates. In particular, we identify the types of queries for which a nested index is more suited. The goals of this analysis are: (1) to show additional opportunities provided by the usage of nested indices in optimizing queries; (2) to derive useful indications for index configuration strategies in the framework of object-oriented DBMSs. The remainder of this paper is organized as follows. In Section 2, we summarize query processing strategies in object-oriented DBMSs from [Kimk 88] and we introduce a graphical representation of query strategies that will be used throughout the paper. In Section 3 we present the analysis of query strategies where the nested index is used.