ГОДИШНИК НА МИННО-ГЕОЛОЖКИЯ УНИВЕРСИТЕТ “СВ. ИВАН РИЛСКИ”, Том 50, Св. II, Добив и преработка на минерални суровини, 2007
ANNUAL OF THE UNIVERSITY OF MINING AND GEOLOGY “ST. IVAN RILSKI”, Vol. 50, Part II, Mining and Mineral processing, 2007

ABOUT DATA FRAGMENTATION AND ALLOCATION IN DISTRIBUTED OBJECT ORIENTED DATABASES

Adrian Runceanu, Marian Popescu,

University Constantin Brâncuşi, Targu-Jiu, Romania; adrian_r@utgjiu.ro; marian@utgjiu.ro

ABSTRACT. In this paper, we present fragmentation and allocation problems in object oriented distributed. Hybrid fragments are defined by applying horizontal fragmentation followed by vertical fragmentation to each database class.
keywords: object oriented databases, link graph, data fragmentation, data allocation.

ОТНОСНО ФРАГМЕНТИРАНЕ И РАЗПРЕДЕЛЕНИЕ НА ИНФОРМАЦИОННИ ПОТОЦИ В ОПРЕДЕЛЕНИ ОБЕКТНООРИЕНТИРАНИ БАЗА ДАННИ

Адриан Рункеану, Мариан Попеску
Университет „Константин Бранкуши”, Търгу Жил, Румъния; adrian_r@utgjiu.ro, marian@utgjiu.ro

РЕЗЮМЕ: В тази статия ние представяме фрагментирането и разпределението на проблемите в определени обектно ориентирани. Части от хибрид са дефинирани чрез прилагането на хоризонтална фрагментация, която е последвана от вертикална фрагментация на всеки клас база-данни.
Ключови думи: обектно ориентирани база данни, свързваща диаграма, фрагментиране (раздробяване) на данни, разпределение на данни

1. Introduction

Data distribution has largely been studied in the relational model, but the complex structure of objects and their relationships make it more difficult for object oriented databases. Data distribution is carried out by first fragmenting data and then allocating the obtained fragments to the network sites. Fragmentation allows grouping data items used together by applications so as to minimize the I/O ratio. The allocation of fragments has the primary goal of minimizing the number of remote accesses which are performed by applications.

As proposed in [KNM94], the fragmentation methods used for the relational model may be extended to the object model. However, as classes are closely related to each other, much more importance must be accorded to derived horizontal fragmentation.

In a relational database, if the instances of a relation *R1* are referenced by those of a relation *R2*, then *R1* are fragmented based on the predicates of *R2*. Fragmenting *R2* if it is referenced by more than one relation is however more difficult. In such a case, must *R2* be fragmented based on the properties of one, some or of all the referencing relations?

This problem is much more complex when dealing with object oriented databases because relationships are not only more numerous but also of different types. Indeed, as mentioned above, the principle contribution of the object model is its ability to reflect reality as accurately as possible through the objects and their relationships. The facilities offered by the object model in moving from the real-world to the model leads the designer to a fine “modularization”, which creates in our sense a great number of classes and relationships and consequently complicates the fragmentation process.

The allocation problem is also more complex in distributed object oriented databases because it must not only deal with data allocation but with method allocation too. As a method manipulates only a subset of the attributes and the objects of a class, it must naturally reside at the site that contains the data it manipulates.

2. Database fragmentation

Since the beginning of the 80's, many researchers have been interested in distributed database design. Technological and organizational reasons justify this tendency: distributed databases eliminate many limitations of centralized databases, and naturally correspond to the decentralized structure of several organizations. A distributed database can be defined as a collection of data, which logically belong to the same system but are distributed over the sites of a computer network [CP84].

The design of a distributed database is a complex task because it requires the comprehension and the resolution of several related subproblems as data fragmentation (horizontal, vertical, hybrid), data allocation (with or without redundancy), optimization and allocation of operations (request transformation, selection of the best execution strategy, allocation of operations to sites). The subproblems we consider are data fragmentation and allocation.

Objects and fragmentation

[KNM94] is to our knowledge the first paper which analyses the problem of object distribution and proposes general algorithms. It gives an overview of the different problems which must be solved in the distribution design. As general solution, it proposes the extension of the relational fragmentation methods. Other works present algorithm s for fragmenting object oriented databases. We will describe briefly the approach followed in each of the algorithms we know.

Horizontal fragmentation. The proposed algorithm uses the principle of minterm predicates. Both of the inheritance and aggregation hierarchies are preserved. The principle consists of producing in a first step primary horizontal fragments for each of the database classes. The result of this fragmentation is then used to induce derived fragmentation via inheritance, composition and method call links. Finally, the algorithm merges primary and derived fragments with which it has the highest affinity. The affinity between a primary and a derived fragment is defined as the frequency at which both are used simultaneously by applications.

Vertical fragmentation. Vertical fragmentation aims to fragment a class such as attributes and methods frequently used together are grouped together. The general approach consists of first grouping the methods frequently used together in each class [EB94]. Grouping techniques similar to those used for attribute grouping in relational are used: method usage matrix, applications' frequencies, method affinity matrix. Each method group is then extended to incorporate the attributes used by the group methods. If an attribute is referenced by methods which belong to different groups, it is assigned to the one with which it has the highest affinity.

A. The object data model

The data in an object based system consists of a set of encapsulate objects. The concept of object represents an encapsulation of the attributes that describe data and the methods that manipulate them. A unique identifier is associated to each object. Objects with common attributes and methods belong to the same class, and every class has a unique identifier. Inheritance allows reuse and incremental redefinition of classes in terms of existing ones. Parent classes are called superclasses while classes that inherit attributes and methods from them are called subclasses. Composition relationships allow the representation of composite objects which include other objects as part of them. These are called component objects. Due to the encapsulation concept, the access to objects can be performed only by method invocations. The set of objects that belong to a class represents its extension. In the inheritance hierarchy, objects are stored in the most specialized class.

B. The link graph

The global conceptual schema is composed of a set of classes and inter-class links. The existence of a link arises due to some real world relationship which exists between two classes. Two types of links characterize the object model: the inheritance links and the composition links.

We explicit such informations in a “Link Graph” that we construct from the global conceptual schema and which constitutes the input to our distribution process. The nodes in the link graph represent classes and the edges composition relationships. Given a class C_j and the set of all its subclasses SUB(C_j), if C_j has an attribute of type C_k, then there exists an edge from each class belonging to $C_j \cup SUB(C_j)$ to C_k. Indeed, as mentioned above, subclasses of C_j inherit both simple and complex attributes of C_j. An example of link graph is given in figure 1.

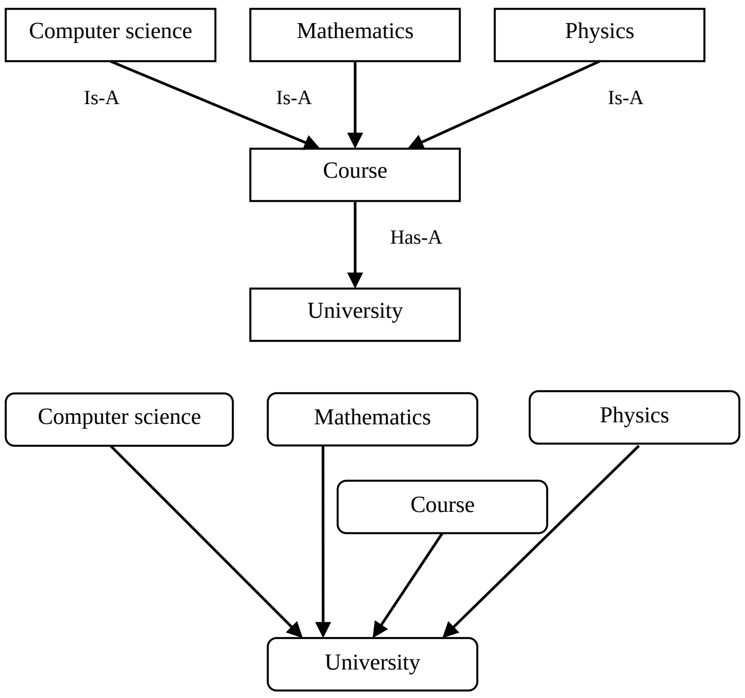

Figure 1: A Link Graph Example

C. Transaction modeling

A user query accessing database objects is defined as a sequence of method invocations on an object or a set of objects of classes. A user query Q_k is represented by $\{M^{i_1 j}, M^{i_2 k}, ..., M^{i_n l}\}$, where each M in a user query refers to an invocation of a method of a class object. It is assumed that the user has a good notion of the important transactions that will run against the database, and of the methods involved in each transaction. We suppose that each transaction has known execution frequencies for each of the sites where it may originate, and that data volumes transmitted between transactions' methods (parameters and results) can easily be estimated.

3. An allocation model

Given a set *S* of *n* sites $\{S_1, S_2, ..., S_n\}$ communicating via a network and a set *F* of *k* fragments $\{F_1, F_2, ..., F_k\}$ communicating by method calls, the allocation problem may be formally described by a function from the set of fragments to the set of sites, $\Pi : F \rightarrow S$. If fragments are not replicated, there exists n^k possible allocations. A performance criterion used for comparing the n^k allocations is a function $f : \Pi \rightarrow R$ which associates a cost to each allocation. An optimal allocation is the one which minimize the cost function. In general, the optimization process aims to minimize the transactions' response time which depends on the I/O ratio and the communication delays. The expression of the cost function must be sufficiently simple in order to be easily evaluated and eventually modern networks.

However, taking into account all the network characteristics, especially modern networks, to develop general models yields generally to non linear functions and consequently complicates the problem. The allocation process must also take into account some imposed constraints such as disk capacity. One may impose for example that the total size of fragments assigned to a site must not exceed the disk capacity.

The distributed database system we consider is assumed to have a set of nodes connected to each other by means of a communication network. Computer hardware (terminal, minicomputer, workstation, personal computer) is located at each node. The hardware need not be identical at each node. This implies that processing and storage capacity may differ from one site to another. The nodes of the network can communicate at a certain cost per unit of data transmitted. Users of the system have access to fragments that can be stored at any of the nodes. Each fragment has a unique identifier, and contains a collection of data and methods. Two fragments F_i and F_j communicate if and only if there exists at least one method of F_i invoking a method of F_j and/or there exists at least one method of F_j invoking a method of F_i. This is a consequence of object encapsulation. Transactions in the database are of two types: read-only queries and update queries. Each query may consist of a series of method calls to extract the data item values and present them to the person sending the request. Similarly, each update could consist of a sequence of methods designed to extract the data item values and write them back into the appropriate database after updating them.

A. The parameters

NF	= number of fragments
NS	= number of sites
i	= fragment index, $i \in \{1,...,NF\}$
l	= site index, $l \in \{1,...,NS\}$
r_i	= method *r* of fragment *i*
$f(r_i,s_j)$	= frequency of invoking method *s* of fragment *j* by method *r* of fragment i
$DT(r_i,s_j)$	= quantity of data transmitted between method *r* of fragment *i* and method *s* of fragment *j* (parameters + results)
$DTF(i,j)$	= quantity of data transmitted between fragment *i* and fragment *j*
MS_i	= set of methods of fragment *i*
FS_i	= size of fragment *i*
d_{lq}	= transmission cost per unit of data between sites *l* and *q*
ST_l	= cost of storing one unit of data at site *l*
Q_l	= storage capacity at node *l*
C_{il}	= cost of storing fragment *i* at site *l* (= $FS_i \times ST_l$)
C_{iljq}	= communication cost between fragment *i* located at site *l* and fragment *j* located at site *q*

B. The decision variables

We define the following decision variables to formulate the problem:

$$x_{il} = \begin{cases} 1 & if \ \ fragment\, i \, is \, allocated \, to \, site\, l, \\ 0 & otherwise. \end{cases}$$

C. The objective function

The objective function to be minimized consists of the sum of communication costs and storage costs.

$$Z_p = \min \sum_{l,q} \sum_{i,j} C_{iljq} x_{il} x_{jq} + \sum_i \sum_l C_{il} x_{il}$$

C_{iljq} is given by the formula :

$$C_{iljq} = d_{lq} \times DTF(i,j)$$

where

$$DTF(i,j) = \sum_{r_i \in MS_i} \sum_{s_j \in MS_j} f(r_i, s_j) DT(r_i, s_j) +$$

$$+ \sum_{s_{ji} \in MS_j} \sum_{r_i \in MS_i} f(s_j, r_i) DT(s_j, r_i)$$

D. The constraints

$$\sum_{l=1}^{NS} x_{il} = 1 \quad \forall i = 1,...,NF \qquad (1)$$

$$\sum_{i=1}^{NF} FS_i \times x_{il} \le Q_l \quad \forall l = 1,...,NS \qquad (2)$$

$$x_{il} \in (0,1) \qquad (3)$$

Constraint (1) states that each fragment is allocated to one site. Constraint (2) ensures that the total size of fragments allocated to one site doesn't exceed the disk capacity of that site. Constraint (3) is the binary constraint on the decision variable.

4. Conclusions

The allocation problem in its generality is NP-Complete. It can be solved by using exact methods or heuristic methods.

Exact methods are based on the exploration of all the possible solutions. Although they result in obtaining optimal solutions, they are very costly and are in exploitable for large problems. An example is the Branch and Bound method.

Heuristic methods yield to suboptimal solutions. The quality of a heuristic is measured uniquely by observing the results it gives, eventually by comparing it with the optimal solution when it is possible to determine it. In this category, several algorithms can be enumerated as greedy algorithms [CLR90], iterative algorithms. Theoretically, these methods do not offer any performance guarantee, however practical experience shows that they generate solutions which are generally close to the optimum, and in reasonable delays.

References

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*, 1990.

[CP84] S. Ceri and G. Pelagatti. *Distributed databases: principles and systems*. McGraw-Hill, 1984.

[EB94] C.I. Ezeife and K. Barker. Vertical class fragmentation in a distributed based system. *Technical Report* 94-03, University of Manitoba, 1994

[KNM94] K. Karlapalem, S.B. Navathe, andM.M.A.Morsi. Issues in distributed design of object oriented databases. In T. Ozsu, U. Dayal, and P. Valduriez, editors, *Distributed object management*, pages 148-164.Morgan Kauffman Publishers, 1994.

[OV99] T. Ozsu and P. Valduriez. Principles of Distributed Databases, Prentice-Hall, second edition, 1999.

[ZO94] Y. Zhang and M. Orlowska. On fragmentation approaches for distributed database design. *Information Sciences*, 1(3):117-132, 1994.

Recommended for publication by the Editorial board