INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

EXPLORING QUERY PROCESSING USING SBA IN OBJECT ORIENTED DATABASE

M. C. NIKOSE[1], S.O.MUNDHADA[2], N.A.RATHOD[3]

1. Asst Professor, Computer Science & Engineering, Sipna College of Engg & Tech, Amravati, Maharashtra.
2. Asst Professor, Computer Science & Engineering, Sipna College of Engg & Tech, Amravati, Maharashtra.
3. Asst Professor, Computer Science & Engineering, HVPM College of Engg & Tech, Amravati, Maharashtra.

Accepted Date:

27/02/2013

Publish Date:

01/04/2013

Keywords

Query Processing,

Query model,

OQL,

Query optimization,

Rewriting,

SBA

Corresponding Author

Ms. M. C. Nikose

Abstract

Query processing is the sequence of actions that takes as input a query formulated in the user language and delivers as result the data asked for. An important task in query processing is query optimization. Query optimization is the refining process in database administration and it helps to bring down speed of execution. Some object-oriented languages allow expressing queries explicitly in the code, which are optimized using the query optimization techniques from the database domain. With respect to this, a formalized object query language (OQL) has been developed that performs optimization of queries at compile time. We use a universal object data model referred to as the stack based approach, which is responsible for naming-scoping-binding principle. In this paper we proposed one of the methods of query optimization depending on rewriting. Query rewriting deal with optimization method at textual level. Optimization by rewriting concerns queries containing so called independent sub queries. It consists in detecting them and then factoring outside the loops implied by query operators.

I INTRODUCTION

In recent years, database research has concentrated on object-oriented data models, which allow storing highly structured data. With regard to the data structuring concepts offered, an object-oriented data model can be looked upon as an extension of the nested relational model, [5] which allows storing relations as attribute values. However, the relational model only permits the alphanumeric data management. A similar role in object-oriented database is fulfilled by object query languages (OQL). The usefulness of these languages strongly depends on query optimization. With growing complexity of data structuring concepts, the complexity of the accompanying query language grows as well and thus also the complexity of query processing and optimization.

A query language is special tool used in contemporary database management systems (DBMSs) to retrieve information from data storage. Although finding information is probably the most important application of modern query languages, they are also used to insert, update, and remove data stored in a database, to identify constraints, active rules, etc. According to specialists modern query languages should be Declarative, of high-level, macroscopic, and Independent of the data organization, Interpreted, Efficient, Objectives of query optimization are,

1. **To retrieve data quickly:** Query optimization is the refining process in database administration and it helps to bring down speed of execution. Most of the databases after are built and filled with data, and used come down on speed. The time taken to execute a query and return results exponentially grows as the amount of data increases in the database leading to more waiting times on the user, and application sides. Sometimes the wait times could range from minutes, to hours, and days as well in worst cases. Technically one of the reasons of slow speeds of executions could be excess normalization leading to multiple tables. More the number of tables more are the complex nature of joins, and thus leading to more execution times. Sometimes, the complex nature of joins could worse the situation by bring the execution into deadlock.

ISSN: 2319-507X
IJPRET

2. To reduce the system resources: The resources required to fulfill a query are reduces, and ultimately provide the user with the correct result set faster.

3. The goal of the query optimizer is the best throughput: This means that it chooses the least amount of resources necessary to process all rows accessed by the statement.

For the correct and precise formalization of object query language the concept of naming-scoping-binding paradigm must take into consideration. Hence we fallow the Stack Based Approach (SBA)[3]. In this paper we are exploring, how query processing is done using SBA in object oriented database.

II. OVERVIEW OF OODBMS

An OODBMS is the result of combining object oriented programming principles with database management principles. Object oriented programming concepts such as encapsulation, polymorphism and inheritance are enforced along with regular database management concepts such as the Atomicity, Consistency, Isolation and Durability (ACID properties) which lead to system integrity, support for an *ad hoc* query language and secondary storage management systems which allow for managing very large amounts of data.. OODB [6] is a system while supporting all the functionality of a relational database system (including queries, transactions, and backup and recovery mechanisms), also offers an Object oriented programming language interface, user defined data types, object identifiers and the ability to manage objects persistently. Features that are common in the RDBMS world such as transactions, the ability to handle large amounts of data, indexes, deadlock detection, backup and restoration features and data recovery mechanisms also exist in the OODBMS world. A primary feature of an OODBMS is that accessing objects in the database is done in a transparent manner such that interaction with persistent objects is no different from interacting with in-memory objects.

Object-oriented databases attempt to provide a seamless join between program and database and hence overcome the impedance mismatch. To make this possible the data manipulation language of an object-oriented database should be computationally complete.

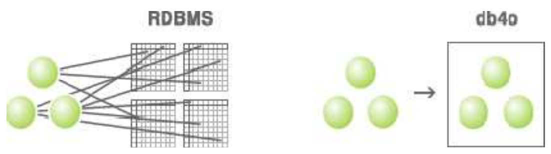

Fig 2.1.1.2 No Impedance Mismatch.

III QUERY PROCESSING IN OODBMS

Query processing is the sequence of actions that takes as input a query formulated in the user language and delivers as result the data asked for. Query processing involves query transformation and query execution. Query transformation is the mapping of queries and query results back and forth through the different levels of the DBMS. Query execution is the actual data retrieval according to some access plan. An important task in query processing is query optimization**.** Query optimization techniques are dependent upon the query model and language. For example, a functional query language lends itself to functional optimization which is quite different from the algebraic, cost-based optimization techniques employed in relational as well as a number of object-oriented systems. The query model, in turn, is based on the data (or object) model since the latter defines the access primitives which are used by the query model. These primitives determine the power of the query model. Usually, user languages are high-level, declarative languages allowing to state what data should be retrieved, not how to retrieve them. For each user query, many different execution plans exist, each having its own associated costs. The task of query optimization ideally is to find the best execution plan, i.e. the execution plan that costs the least, according to some performance measure. Usually, one has to accept just feasible execution plans, because the number of semantically equivalent plans is too large to allow for enumerative search.

3.1 Query Processing Architecture

During static analysis we simulate run-time query evaluation to gather information that we need to optimized the queries. The general architecture of query processing is shown below. A parser of queries and programs takes a query source as input, makes syntactic analysis and returns a query/program syntactic tree. A query/program syntactic tree is a data structure which keeps the abstract query syntax in a well-structured form, allowing for easy manipulation (e.g. inserting new nodes or sub trees, moving some sub tree to another part of the tree, removing some sub trees, etc.). Each node of the tree contains a free space for writing various query optimization information.

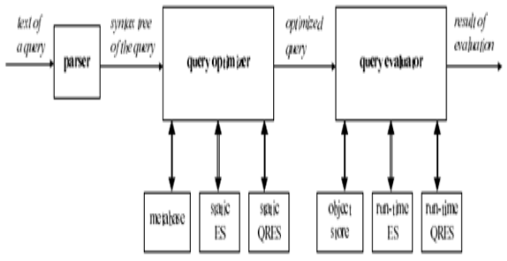

Fig. 1. Architecture of query processing

Syntactic trees are the most convenient way to represent and to process query that can be optimized through rewriting to be used during query evaluation. Static analysis involves,

- Metabase (counterpart of an object store), which is a formal data structure representing database schema.
- Static environmental stack (counterpart of a run-time environmental stack) that stores so-called static binders, i.e. named signatures. The structure during compile time simulates all the operations that are made on the environmental stack by query/program run-time mechanism.
- Static result stack: the structure that accumulates signatures being the type description of corresponding run-time temporary and final query results.

Query processors are usually combination of two components namely,

- Query parser: checks whether a query is syntactically and semantically correct, and transform it to some internal form.
- Query optimizer: determines query evaluation plans. A query optimizer should generate a set of plans, among these only one of the plans is chosen.
- Query plan evaluator: evaluates query in accordance with the chosen evaluation plan.

IV Overview of Stack Based Approach for Query Optimization

The ideas of stack-based approach contribute to a very hot research and technological area, thus have to compete with a tremendous amount of proposals: formal and informal, implemented and speculated .The comparison of our approach with ones is a bit difficult, as the discussion has to involve a taste, belief and backgrounds of competing parties, which are incompatible as a rule. We can only indicate lack of some kinds of queries and/or data structures in a particular approach. Besides the universality, this approach is regular, minimal and semantically clean, which is not the case of informal proposals as a rule.

4.1 The Stack-Based Approach (SBA)

SBA is a formal frame addressing object-oriented database query and programming languages (PLs). The approach is motivated by the belief that there is no definite border line between querying and programming; thus there should be a universal theory that uniformly covers both aspects. SBA [5] offers a unified and universal conceptual and semantic basis for queries and programs involving queries, including programming abstractions such as procedures, functions, classes, types, methods, views, etc. SBA [7] treats query language as a special kind of programming language; it is an attempt to build a uniform semantic foundation for integrated query and languages. SBA allows to precisely determining the semantics of query languages; there relation with object oriented concepts, with imperative programming constructs and with programming abstraction, including procedures, functional procedures views, modules etc. its main features are the following

- The naming-scoping-binding principle is assumed, which means that each name occurring in a query is bound to the appropriate run-time entity (an object, attribute, method, parameter, etc. according to the scope of this name).
- One of its basic mechanisms is an environmental stack (ES). The stack is responsible for scope control and for binding names. In contrast to classical stack it does not store objects, but some structures built upon object identifiers, names, and values.

- In contrast to relational languages and OQL, the relativity principle is assumed, i.e. the syntax, semantics, and pragmatics are identical at an arbitrary level of data hierarchy.
- Types are a mechanism to determine whether objects are built in a proper way (i.e. in accordance with the database schema).
- For objects the principle of internal identification is assumed (i.e., each run time entity has a unique internal identifier).

4.2 Abstract Object-Oriented Store Model

An object store model is simply an abstract view on data structured stored in the database and is orthogonal to any ideologies such as the relational model or XML. The main components of object store model are its location, the name that can be used to denote it, the value stored there i.e. its contents. Atomic values, records ,conceptual object are the example of object store, which resides somewhere in memory and needs location for internal identiction to perform operation like update, retrive, delete. These location does not have meaning or structure,instede they can be used to access object. Location are some times called as identifires[5].

In SBA each object has the following features;

- **Internal object identifier (OID):** It is used to identify internally an object from the object store model; in particular, it will be used as a reference or pointer to an object. For each object stored in the object store its internal object identifier is unique.
- **External object name**: They are explicitly assigned to objects by a database designer, a database administrator, an application programmer, or another human agent, e.g. *Person, Department, etc*.
- **Contents** of an object, which can be a value, a link or a set of objects.

Following three sets are used to define object,

- set of internal identifiers
- N- set of external names

- V- set of atomic values

Formally let, i, i1, i2 ∈ I, n ∈ N and v ∈ V. Objects are formulated as three triples as,

- Atomic object: <i, n, v> where i is the ID of the object, n is an external name assigned to the object, and v is the value of the atomic object (e.g. an integer, a string, etc.)
- Link object: <i1, n, i2 > where i 1 is the ID of the reference object, n is an external name assigned to the object, and i2 is the ID of the object referred to.
- Complex object <i, n, S> where i is the ID of the object, n is an external name assigned to the object, and S is a set of objects.

4.3 ENVIRONMENTAL STACK

Environmental stack (ENVS) is a basic concept of semantics and implementation of majority of well-known programming languages. The stack employs following roles

- Control over scopes of variable names occurring in a program and binding these names to run-time entities;
- Storing values of local variables of procedures;
- Storing values of actual parameters of procedures;
- Storing a *return track*, i.e. an address of the program code where the control should be returned after the given procedure is terminated (usually the next address after the procedure call).

ENVS is subdivided into sections, which are ordered, with newest section known as top and oldest one as bottom. A section is associated with a particular *procedure call* or an executed *program block.* When the control is transfer to a procedure call, a new section of volatile objects (activation record) is pushed on the top of the stack. The section is popped from the sack when the procedure or program block is terminated. For the procedure that is currently running all values of parameters, local variables, objects and any other local entities are stored within the top stack section. Activation record possess the abstraction principle, which allows the programmer to consider the currently being written piece of code to be independent of the context of its possible uses. The stack allows to associate parameters and local variables to particular procedures invocation. The stack also used to accomplish strong typing, encapsulation, inheritance and overriding. In SBA the stack has a new function: processing queries acting on the object store. It makes it possible to control scopes of all names occurring in a query in a simple and uniform way. It also makes it easier to understand the precise semantics of queries [1], [5].

A. Concept of Binding

The mechanism, which make it possible to determine the meaning of each name is called binding. The sections of ENVS consist of entities called binders whose role is to relate a name with a run time entity (object, procedure, view). Binding of the name *n* means that the query interpreter looks through the consecutive sections of the ENVS to find the closest binder with the name *n*. Binding is performed on ENVS according to "search from the top" rule. A binder is a pair *(n, x),* where *n* is an external name, and *x* is some value (reference to object). The pair is written as *n(x).* A binding action for some name *Y* is performed according to the following steps:

- The machine checks the top of the stack for an entity named *Y*; if there is such an entity, the binding returns it as the result and the action is terminated.
- If the top does not contain an entity named *Y*, a section below the top is checked.
- Such a process is continued in lower and lower stack sections, till the entity named *Y* is found. Visiting particular stack sections is governed by *scoping rules* that require omitting some sections;
- If *Y* is not found on the stack, then the global environment is searched. The global environment contains static variables, database objects, computer environment variables, procedure libraries, etc. Alternatively, we can assume that the global environment is the lowest stack section in such a case *Y* must be found on the stack, otherwise an error should be reported.

B. Opening a new section of ENVS

In SBA at the beginning of a user session ENVS contains of single section containing binders for all root database objects. During query evaluation the stack is growing and shrinking according to query nesting. The final ENVS state is exactly the same as the initial state. Opening a new scope on the environment stack is caused by entering a new procedure (function, method) or entering a new block. Respectively, removing the scope is performed when the control leaves the body of the procedure or the body of the block. To these classical situations of opening a new scope on the environment stack we add a new one. It is the essence and motive of SBA.

The idea is that some query operators which combines queries behave on the stack similarly to program blocks. These operators are divided into algebraic and non-algebraic. The main difference between them is whether they modify ES during evaluation or not. An operator is algebraic if it does not modify the state of ENVS. The algebraic operators include numerical and string operators and comparisons, Boolean and, or, not, aggregate function, and sequence operators and comparisons, structure constructor, dereferencing etc. Operators which name a query result are unary algebraic operators too. If query q1 ϴ q2 involves a non-algebraic ϴ, then q2 is evaluated in the context of q1. The context is determined by the new section opened by the ϴ operator on ENVS for an element of q1. A new stack section pushed onto ENVS is constructed by a special function. Sub queries q1 and q2 cannot be processed independently, the order of evaluation is important. Non-algebraic operators include projection /navigation (q1.q2), selection (q1 where q2), dependent join (q1 join q2), quantifiers (∃q1q2), transitive closure and ordering.

The following figure present all the stages of the evaluation of simple query P.getSal ()>2500, here the part getSal>2500 behaves like a program block executed in the environment consisting of the interior of an professor object.

ISSN: 2319-507X
IJPRET

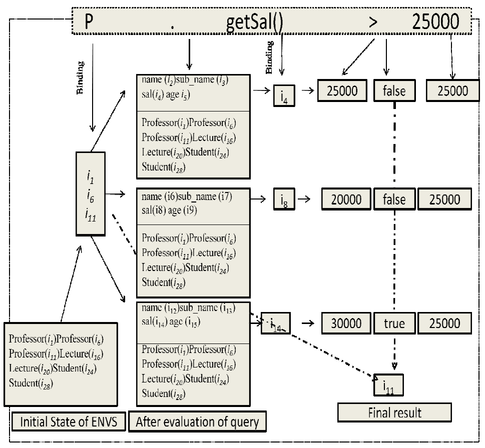
P . getSal() > 25000
Binding
Binding
name (i_2)sub_name (i_3) sal(i_4) age i_5)
Professor(i_1)Professor(i_6) Professor(i_{11})Lecture(i_{16}) Lecture(i_{20})Student(i_{24}) Student(i_{28})
i_4
25000
false
25000
i_1 i_6 i_{11}
name (i6)sub_name (i7) sal(i8) age (i9)
i_8
20000
false
25000
name (i_{12})sub_name (i_{13}) sal(i_{14}) age (i_{15})
i_{14}
30000
true
25000
i_{11}
Final result
Initial State of ENVS
After evaluation of query

Fig. 2: Evaluation of query P.getSal ()>2500

V METHOD OF INDEPENDENT SUBQUERIES

In this paper we are proposing one of the methods of query optimization. The idea of this method is based upon the observation that, if none of the name in sub query is bound in the ENVS section opened by the non-algebraic operator currently being evaluated, then this sub query is independent of this operator. Sub queries are called independent if they can be evaluated outside loops implied by the non-algebraic query operators. Such sub queries are worth analyzing because they usually imply optimization possibilities [1],[2].We use a special technique called the method of independent subqueries to optimize queries. This method is based on query rewriting that deals with optimization method at textual level. Rewriting means transforming a query q1 into a semantically equivalent query q2 promising much better performance.

Technically, it consists in analyzing in which sections particular names occurring in a query are bound. It turns out that if none of the names in given sub query is bound in the scope opened by the non-algebraic operator currently being evaluated, then that sub query can be evaluated earlier than it results from its textual place in the query it is a part of. The method modifies the textual form of a query so that all its sub queries will be evaluated as soon as possible. To determine in which scopes names occurring in a query are bound at run time [9], we statically analyze that query and during analyzing we additionally do the following:

- Each non-algebraic operator is assigned the number of the scope it opens,
- Each name in the query is assigned two numbers:

- The stack size: the number of scopes thst is on static ENVS when the binding of this name is being performed.
- The binding level: the number of the scope on the stack in which this name is bound.

All those numbers are determined relatively to the bottom scopes of a query.

VI CONCLUSION

One of the biggest problems in Object Oriented Database is the optimization of queries. Due to these problems optimization of object-oriented queries is extremely hard to solve and is still in the research stage. To combat the limitations of RDBMS and meet the challenge of the increasing rise of the internet and the Web, programmers developed object-oriented databases. In this paper we discuss some basic concepts and features of OODBMS. We present a new optimization method for queries involving independent subquery. Hence, planning of an execution strategy may be a more accurate description than query optimization. This proposed work is expected to be a significant contribution to the Database Management area which will not only reduce time or efforts but will also improve the quality.

REFERENCES

1. [JPlod00] J. Płodzień, A. Kraken, "Object Query Optimization through Detecting Independent Subqueries", Information Systems, Elsevier Science, 25(8), 2000, pp. 467-490.

2. [Mich09] Michel Bleja, Krzysztof Stencel, Kazimierz Subeita, Optimization of Object-Oriented Queries Addressing Large and Small Collections, Proc. Of the IMCSIT, 2009, ISBN 978-83-60810-22-4, Vol. 4, pp. 643-680.

3. [Venk10] Venkata Krishna Suhas Nerella, Swetha Surapaneni, Sanjay Kumar Mardria, Thomos Weigert, Department of Computer Science, Missouri Univarsity of Science & Tech, Rolla, MO, 34 th Annual IEEE Computer Software & Applications Conference,2010.

4. [Plod00] J. Plodzien, "Optimization Methods in Object query Languages", Ph. D. Thesis, Institute of Computer Science, Polish Academy of Sciences, 2000.

5. [Subi95] K.Subieta, C.Beeri, F.Matthes, J.W.Schmidt. A Stack-Based Approach to Query Languages. Proc.2nd East-West Database Workshop, 1994, Springer Workshops in Computing, 1995, 159-180.

6. [Adam08] R.Adamus, M.Daczkowski, P.Habela, K.Kaczmarski, T.Kowalski, M.Lentner, T.Pieciukiewicz, K.Stencel, K.Subieta, M.Trzaska, T.Wardziak, J.Wiślicki: Overview of the Project ODRA. Proceedings of the First International Conference on Object Databases, ICOODB 2008, Berlin 13-14 March 2008, ISBN 078-7399-412-9, pp.179-197.

7. [Subi10] K.Subieta. Stack-Based Architecture (SBA) and Stack-Based Query Language(SBQL). http://www.sbql.pl/, 2010.

8. [ClDe92]Cluet, C. Delobel. A General Framework for the Optimization of Queries. Proc. Of SIGMOD Conference, 383-392, 1992.

9. [MAG05] Minyar Sassi, and Amel Grissa-Touzi Contribution to the Query Optimization in the Object-Oriented Databases" Volume 6 ISSN, June 2005 1307-6884

10. [db4oOSODB] db4objects 11.2008, db4o Open Source Object Database.