Kỷ yếu Hội nghị Khoa học Quốc gia lần thứ IX "Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR'9)"; Cần Thơ, ngày 4-5/8/2016
DOI: 10.15625/vap.2016.0006

ANALYSIS OF FUZZY QUERY PROCESSING AND OPTIMIZATION IN FUZZY OBJECT ORIENTED DATABASE

Nguyen Tan Thuan[1], Doan Van Ban[2],Truong Ngoc Chau[3], Tran Thi Thuy Trinh[1]

[1]Information Technology Department – Duy Tan University

[2]Institute of Information Technology – Vietnamese Academy of Science and Technology

[3]Information Technology Department – Danang University of technology

nguyentanthuan2008@yahoo.com,dvban@ioit.ac.vn,truongngocchau@yahoo.com,thuytrinh85@gmail.com

ABSTRACT— *Modern Fuzzy object-oriented database are complex programs which get user queries, translate them in internal representation necessary for data access and provide results in best possible way; that is, results which take less execution-time and consume fewer resources. However, as the data is increasing at rapid pace, there is always a need of systems which meet organizational requirements. Better systems can only be developed if internal working of (existing) systems is understood very well. This paper presents a new approach fuzzy object algebra expression optimization that helps in understanding how (high-level) user queries are translated in internal representation necessary for data access. Furthermore, the study also implements few optimization techniques to produce alternative execution plans and their corresponding execution time. Paper also presents a discussion on which plans are better based on the computational analysis.*

Keywords *— Fuzzy object algebra expression, Fuzzy OODB, Query Processing, Query Optimizatio, FOQL.*

I. INTRODUCTION

Fuzzy object-oriented database management systems are complex programs responsible for performing all this query processing and optimization. The pace at which data recording needs are increasing one can confidently say that future systems will be even more complex as they have to deal with voluminous amounts of data. Therefore, for meeting future requirements it is very important that existing working of FOODBM software must be understood very well. In this paper, we present a new approach that helps in understanding how user queries provided in fuzzy object query language (FOQL) are translated in internal representation (fuzzy object algebra). Furthermore, for each input query we implements three optimization strategies (simple, elimination of Cartesian product, push selection) to produce alternative execution plans and their corresponding execution time. Based on the study results, paper presents discussion on which optimization strategies are better.

Much research has been carried out on query translation and optimization for e.g., in [2] a detailed discussion on fuzzy query processing is provided. Similar research has also been provided in [3] and [4] which have remained very useful for the development of the part of our study which translates user queries from FOQL to fuzzy object algebra. Similar to our work, Bendre M. et. al., in [6] have developed a tool that translates fuzzy object algebraic expressions to relational calculus (FOQL). Our inspiration came also from the work presented in [6].

The various sections of the paper are organized as follows. In first section we discuss the existing definitions of fuzzy object oriented database models. In second section, we introduct the Fuzzy Query Processing Model in a more generalized manner and propose an efficient method for query optimization based on fuzzy object algebra expression. In third section, we implement performance evaluation. In last section we provide the conclusion of this study and discuss some future research directions.

II. FUZZY OBJECT-ORIENTED DATABASE MODELS

A. Fuzzy object-oriented database models.

Based on the discussion above, we have known that the classes in the fuzzy OODB may be fuzzy. Accordingly, in the fuzzy OODB [7][8][9], an object belongs to a class with a membership degree of [0, 1] and a class is the subclass of another class with degree of [0, 1] too [9]. In the OODB, the specification of a class includes the definition of ISA relationships, attributes and methods implementations. In order to specify a fuzzy class, some additional definitions are needed. First, the weights of attributes to the class must be given. In addition to these common attributes, a new attribute should be added into the class to indicate the membership degree to which an object belongs to the class. If the class is a fuzzy subclass, its super classes and the degree that the class is the subclass of the super classes should be illustrated in the specification of the class. Finally, in the definition of a fuzzy class, fuzzy attributes may be explicitly indicated.

A new F-model is defined as an enhanced OO data model by replacing objects with fuzzy objects, classes with fuzzy classes and associations with fuzzy associations on the essential characteristics of OO paradigm. Objects are physical entities, abstract concepts, events, processes or whatever of interest. Each object is assigned a system-defined, unique object identifier (OID). Fuzzy objects are objects which have fuzzy attribute values and fuzzy associations with other objects and grouped to form fuzzy classes.

1. Fuzzy Attribute Values

We call both imprecise values and vague values as fuzzy values. Every certain and precise values can be extended fuzzy values. We define three different kinds of fuzzy values.

A crisp value a on an universe U is characterized by the following function:

$$\mu_a(x) = \begin{bmatrix} 1, & if\ x = a \\ 0, & if\ x \neq a \end{bmatrix}$$

An *imprecise value a* is expressed as an interval which contains two elements at least, and characterized by the following function:

$$\mu_a(x) = \begin{bmatrix} 1, & if\ x \in a \\ 0, & if\ x \notin a \end{bmatrix}$$

A vague value a on an universe U is defined by a fuzzy set and characterized by the function such as $0 \leq \mu_a(x) \leq 1$, for all$x \in$ U.Linguistic descriptions for objects are vague values.

2. Fuzzy Object and Fuzzy Class

A fuzzy object is an object whether it belongs to or not to a class or has fuzzy associations. A fuzzy class is such a class which has an uncertain boundary. For example, classis PERSON has a certain boundary and class HIGHLY EDUCATED F'ERSON has an uncertain boundary. A fuzzy class (FC) is defined as the followings:

$$FC_i=\{(O_{ij}, \mu(O_{ij}))|\ O_{ij} \text{ is an object and } \mu(O_{ij})>0 \text{ and } \mu(O_{ij})\in [0,1]\}.$$

where $\mu(O_{ij})$ is a degree of membership of jth object in fuzzy class FC_i, O_{ij}. As the value is closer to 1, the object is most likely a certain member. A conventional class in OO databases can be extended lo a fuzzy class which is a set of fuzzy objects with membership value 1. We call a pair (Oij, $\mu(O_{ij})$) a fuzzy object.

B. Algebra operators.

When we are combining two existing classes, for creating a new class. It's depending on the relationships between two combining classes/entities, and set of attribute are combining with classes also. There are six types of binary combination operations can be defined such as: fuzzy union ($\tilde{\cup}$), fuzzy intersection ($\tilde{\cap}$), fuzzy join $\tilde{\bowtie}$, fuzzy cross product $\tilde{\times}$, fuzzy difference ($\simeq$), and fuzzy division ($\tilde{\div}$).

C. Equivalent transformation rules

Assume that $\mu C(o), \mu C1(o), \mu C2(o), \mu C3(o)$ are the set for fuzzy object: e, f, g, h are algebraic expressions, arithmetic $op \in \{union, diff\}$. These rules apply only on the fuzzy object operations, math sets, set operations and multi-set operations (bag). On signs, we only use math Notations in a form [pp207,1] operations can be setup with a changes in a number of different models.

R1. Selection operations are commutative: $\sigma_{\lambda t.g}\left(\sigma_{\lambda s.f}(\mu C(o))\right) = \sigma_{\lambda s.f}\left(\sigma_{\lambda s.g}(\mu C(o))\right)$

R2. Conjunctive selection operations can be deconstructed into a sequence of individual selections; cascade of σ.

$$\sigma_{\lambda s.(f \wedge g\ \wedge \ldots h)}(\mu C(o)) = \sigma_{\lambda s.f}\left(\sigma_{\lambda t.g}\left(\ldots\left(\sigma_{\lambda u.h}(\mu C(o))\right)\ldots\right)\right)$$

R3. Only the final operations in a sequence of projection operations is needed, the others can be omitted; cascade of Π

$$\Pi_{(a_1 \ldots a_n)}\left(\Pi_{(b_1 \ldots b_n)}(\mu C(o))\right) = \Pi_{(a_1 \ldots a_n)}(\mu C(o))|\ \{a_1, \ldots, a_n\} \subset \{b_1, \ldots, b_n\}$$

R4. Permutation selection and projection: $\sigma_{\lambda s.e}\left(\Pi_{(a_1,\ldots,a_n)}(\mu C(o))\right) = \Pi_{(a_1,\ldots,a_n)}\left(\sigma_{\lambda s.e}(\mu C(o))\right)$

R5. Permutation and a projection over union, on a set / multiset:

$$\Pi_{(a_1,\ldots,a_n)}(\mu C1(o)\ op\ \mu C2(o)) = \Pi_{(a_1,\ldots,a_n)}\mu C1(o)\ op\ \Pi_{(a_1,\ldots,a_n)}\mu C2(o)$$

R6. The selection operation distributes over the union, and Difference, on on a set / multiset

$$\sigma_{\lambda s.f}(\mu C1(o)\ op\ \mu C2(o)) = \sigma_{\lambda s.f}(\mu C1(o)\ )\ op\ \mu C2(o), if\ f\ is\ \ related\ to\ \mu C1(o)$$

Generality:

$$\sigma_{\lambda s.(f \wedge g \wedge h}(\mu C1(o)\ op\ \mu C2(o)) = \sigma_{\lambda u.h}\left(\sigma_{\lambda s.f}(\mu C1(o))\ op\ \sigma_{\lambda t.g}(\mu C2(o))\right), if\ f\ is\ \ related\ to\ \mu C1(o),$$

$$g\ is\ \ related\ to\ \mu C2(o)\ and \text{ h related to both } \mu C1(o) \text{ and } \mu C2(o)$$

R7. Permutation between selection orperation and apply orperation: if conditions only contain attributes selected by the operation returns apply: $apply_{\lambda_{\mu s.e}}\left(\sigma_{\lambda t.f}(\mu C(o))\right) = \sigma_{\lambda t.f}\left(apply_{\lambda_{\mu s.e}}(\mu C(o))\right)$

R8. Permutation between flat and apply on set and multiset: Suppose that $\mu C(O)$ is an instance of a class and x is a complex set of attributes of the class.

$$flat\left(apply_{\lambda s.\left(apply_{\lambda t.e}\left(\Pi_{(X)}\left(\Pi_V(\mu C(o))\right)\right)\right)}(\mu C(o))\right) = apply_{\lambda t.e}\left(flat\left(apply_{\lambda s.\Pi(X)}\left(\Pi V(\mu C(o))\right)(\mu C(o))\right)\right)$$

R9. Set union is associative: $(\mu C1(o)\ union\ \mu C2(o))\ union\ \mu C3(o) = \mu C1(o)\ union\ (\mu C2(o)\ union\ \mu C3(o))$

R10. The inheritance laws to allow the selection and apply: if C2 is a subclass of C1, instance of $\mu C2(o)$ is a subset of instance of $\mu C1(o)$.

$$\sigma_{\lambda s.f}(\mu C1(o))\ union\ \sigma_{\lambda s.f}(\mu C2(o)) = \sigma_{\lambda s.f}(\mu C1(o)) apply_{\lambda s.e}(\mu C1(o))\ union\ apply_{\lambda s.e}(\mu C2(o)) = apply_{\lambda s.e}(\mu C1(o))$$

III. FUZZY QUERY PROCESSING MODEL AND OPTIMIZATION

A. An Optimization Methodology

A query processing methodology similar to relational DBMSs, but modified to deal with the difficulties discussed in the previous section, can be followed in FOODBMSs. Figure 1 depicts such a methodology proposed in [10].

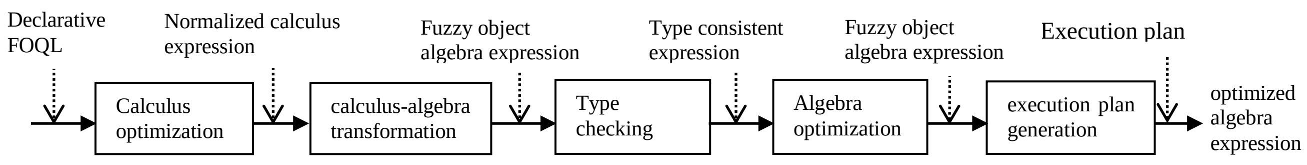

Declarative FOQL
Normalized calculus expression
Fuzzy object algebra expression
Type consistent expression
Fuzzy object algebra expression
Execution plan
Calculus optimization
calculus-algebra transformation
Type checking
Algebra optimization
execution plan generation
optimized algebra expression

Figure 1. Fuzzy Query Processing Methodology

In this methodology, the query-rewrite optimization is a high-level process where general-purpose heuristics drive the application of rewrite rules, and plan optimization is a lower level process which generates execution plans based on knowledge of relative costs, statistics and physical structure. A declarative query is thus optimized as follows:

- The calculus expression is first reduced to a normalized form by eliminating duplicate predicates, applying identities and rewriting.
- The normalized expression is then converted to an equivalent fuzzy object algebra expression. The algebra form of the query is a nested expression which can be viewed as a tree whose nodes are algebra operators and whose leaves represent extents of classes in the fuzzy object-oriented databases.
- The algebra expression is next checked for type consistency to insure that predicates and methods are not applied to objects which do not support them. This is complicated by the potential heterogenous nature of fuzzy query results.
- Next the type-checked expression is rewritten using an equivalence-preserving rule system.

Lastly an execution plan, or perhaps several different execution plans, are generated, which take into account the actual fuzzy object implementations.

B. Query translation and optimization

1. Structured Fuzzy Object Query Language

Structured fuzzy object query language (FOQL) is declarative query language developed for users' comfort that tells the database what user wants.

Structure of FOQL Query is based on three clauses:

SELECT <*attribute list*>FROM <*Class*WITH *threshold Class* WITH *threshold*>WHERE <*query condition* WITH *threshold*>.

Here, <query condition> is a fuzzy condition and all thresholds are crisp numbers in [0; 1]. Utilizing such SQL, one can get such objects that belong to the class under the given thresholds and also satisfy the query condition under the given thresholds at the same time. Note that the item WITH threshold can be committed. The default of the threshold is exactly 1 for such a case. Now we give a fuzzy query example. Assume we have a fuzzy class Young Salespersons as follows.

CLASS *YoungSalesPersons* WITH DEGREE OF *1.0* INHERITS *SalesPersons* WITH DEGREE OF *1.0*

ATTRIBUTES *ID*: TYPE OF *string* WITH DEGREE OF *1.0*

Name: TYPE OF *string* WITH DEGREE OF *1.0 Age*: FUZZY DOMAIN {*very young*, *young*, *old*, *veryold*}:

TYPE OF *integer* WITH DEGREE OF *1.0 Sex*: FUZZY DOMAIN {*male*, *female*}: TYPE OF

character WITH DEGREE OF *1.0 DOB:* FUZZY DOMAIN *{day, month, year}:* TYPE OF

integer WITH DEGREE OF *1.0 Membership_Attribute name*

WEIGHT w (*ID*) = *0.1*w (*Name*) = *0.1*w (*Age*) = *0.9*w (*Sex*) = *0.1w (DOB) = 0.5* METHODS END

A query based on the class is issued by using:

SELECT Name FROM YoungSalesPerson, *SalesPersons* WITH 0.5

WHERE YoungSalesPerson.FIOD= *SalesPersons.FOID* AND YoungSalesPersons. Age = very young WITH 0.8.

2. Optimization of Query Execution Plans

Given a query, obviously there are many equivalent processing trees, that is, alternatives to execute the query. These alternatives result from a number of open choices, some of which are listed in the following.

Join Ordering. A sequence of join operations is freely reordered able, this result in many alternatives. In order to reduce the number of orderings to consider, one may exclude the ones resulting in Cartesian products, or restrict the join orderings to the ones resulting in linear join trees, instead of considering all possible, bushy ones (this is the well-known problem studied, for example, in [13,14,15].

The reason for join operations to occur is the following: First, reassembling objects from several relations requires a join operation for each partition. Object partitioning is introduced for example, in the mapping of inheritance hierarchies. Second, each application of an object valued function (which is not materialized) results in a corresponding join. Reordering these joins corresponds to changes to the order of function application, that is whether we do forward or backward traversals, or starting in the middle of a path query [12].

Pushing Selections. Selections may be performed at various times. For example, one could perform a given selection before or after applying an object value function (move selection into join, or vice versa). Additionally, selections with conjunctively combined selection predicates may be split into two selections (or vice versa), and the order of selections, as well as the order of the terms in selection predicates, may be changed.

Pushing Projections. Projections, in the same way as selections, may be performed at various times. In addition, the order of applying projections and selections may be changed.

Method Selection. For each logical operation there may be several methods, that is physical implementations. For example, a join operation may be performed by nested loops join, hash join, or sort merge join. In addition, for implicit joins which are supported by link fields, pointer based join versions may be selected. Further, selections may be performed by using indexes, or by performing a relation scan, followed by predicate testing.

Index Selection. If a selection is supported by multiple indices, one has the opportunity to use all, some, or just one of the available indices.

3. Fuzzy object algebra

As opposed to FOQL, fuzzy object algebra is procedural language that not only tells the FOOBD ***what*** user wants but also tells ***how*** to compute the answer. Relational algebra is based on relations and operators that operate on relations; see Figure 2. Most commonly used relational operators are:

- Select ($\tilde{\sigma}$): Returns tuples that satisfy a given predicate (condition or formula).
- Project ($\tilde{\pi}$): Returns attributes listed
- Join ($\tilde{\bowtie}$): Returns a filtered cross product of its arguments.

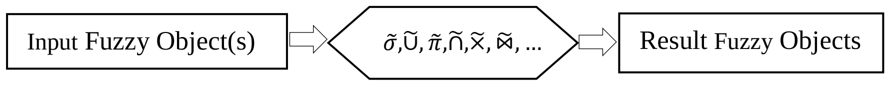

Input Fuzzy Object(s)
$\tilde{\sigma}, \tilde{\cup}, \tilde{\pi}, \tilde{\cap}, \tilde{\times}, \tilde{\bowtie}, \ldots$
Result Fuzzy Objects

Figure 2. Relational operators take one or two relations as input and produce a resultant relation.

4. FOQL to fuzzy object algebra Translation

The transformation equivalence between FOQL queries and Fuzzy object Algebra.

Definition. If E is an fuzzy object algebraic expression and Q fuzzy query object is FOQL together define a sets fuzzy object , we say Q represent E and the opposite, we call E equivalent to Q. Symbol E ≈ Q.

Equal representation between the query language and algebra FOQL fuzzy object is expressed through two theorems 1 and 2 as follows:

Theorems 1. [1] Every algebraic expressions are fuzzy object represented by the object query in FOQL.

Theorems 2. [1]Every Fuzzy object in FOQL queries are represented by algebraic expressions fuzzy object

Thus, the rewrite a given query into algebraic expressions with algebraic set objects are equivalent. The algebraic expressions can be estimated with different abatement costs. So theoretically we wanted to find an algebraic expression equivalent to a query so that it can achieve a plan for more effective enforcement. However, in the solution installed, because the number of queries equivalent too large, that we only need a subset of this query. Therefore, in order to find other similar queries, we will need a set of rules to transform the equivalent algebraic expressions. However, the model fuzzy object oriented data does not have a standard fuzzy algebraic objects applicable to all models of fuzzy object-oriented, so the expectation to have a normal training include modified protection laws Full equivalent does not exist. So, we wanted to prove that the transformation preserved on a basis equivalent algebraic fuzzy objects that may be acceptable. Some law transform is presented [pp207,1].

In order to translate a FOQL query into *Fuzzy Object Algebra*, query processor translates each **SELECT** clause to **Project ($\widetilde{\pi}$), FROM** clause to **objects name(s)** or their **Cartesian**, and **WHERE** clause to **Select ($\widetilde{\sigma}$).**

The query mentioned earlier to "return the names of YoungSalespersons who earned Age ' very young" can be written in Fuzzy Object Algebra as follows:

$$\widetilde{\boldsymbol{\pi}}_{\boldsymbol{Name}}\left(\begin{array}{c}\widetilde{\boldsymbol{\sigma}}_{\text{YoungSalesPersons}.\boldsymbol{foid}=SalesPersons.\boldsymbol{foid}\ \wedge\ \text{YoungSalesPersons}.\boldsymbol{Age}='\boldsymbol{very\ young}'}\\(\text{YoungSalesPersons}\ \widetilde{\times}\ SalesPersons)\end{array}\right)$$

However, query optimizer produces other equivalent alternative execution plans implemented in it so that a better plan in terms of execution time and resources may be found.

5. Algebraic Optimization

We previously identified the components of an algebraic optimizer. In this section, we discuss each of these components in some detail.

a) Search Space and Transformation Rules.

A major advantage of algebraic optimization is that an algebraic query expression can be transformed using well defined algebraic properties such as transitivity, commutatively and distributive. Therefore, each query has a (potentially large) number of equivalent expressions, which make up the search space. These expressions are equivalent in terms of the results that they generate, but may be widely different in terms of their costs. Thus, the query optimizers modify the query expressions, by means of algebraic transformation rules, in an attempt to obtain one which generates the same result with the lowest possible cost. The transformation rules are very much dependent upon the specific object algebra, since they are defined individually for each object algebra and for their combinations. The lack of a standard object algebra definition is particularly troubling since the community cannot benefit from generalizations of numerous object algebra studies. The general considerations for the definition of transformation are rules and the manipulation of query expressions is quite similar to relational systems, with one particularly important difference. Relational query expressions are defined on flat relations, whereas object queries are defined on classes (or collections or sets of objects) that have inheritance relationships among them. It is, therefore, possible to use the semantics of these relationships in object-oriented query optimizers to achieve some additional transformations.

b) Search Algorithm

Since the enumerative search algorithms are based on evaluating the cost of the entire search space, their overhead is quite high. In relational systems, the number of join operations typically determines the complexity of enumerative search. If there are N join operations and there are two choices for join ordering (for inner and outer relations), then there are $O(2^N)$ alternative query expressions to evaluate. Heuristics such as performing selections and projections before joins (to reduce the sizes of the join operands) do not change the combinatorial nature of the problem. Therefore, the value of N and the threshold beyond which combinatorial nature of the problem makes enumerative solutions infeasible becomes an important issue. N=10 has been suggested as an empirical threshold value [Ioannidis and Wong 1987].

Heuristic:

1. The parser of a high-level query generates an *initial internal representation*;
2. Apply heuristics rules to optimize the internal representation.
3. A query execution plan is generated to execute groups of operations based on the access paths available on the files involved in the query.

The **main heuristic** is to apply first the operations that reduce the size of intermediate results.

E.g., Apply fuzzy SELECT and fuzzy PROJECT operations before applying the fuzzy JOIN, or other binary operations.

Outline of a Heuristic Fuzzy Object Algebraic Optimization Algorithm:

1). Using rule R2, break up any select operations with conjunctive conditions into a cascade of select operations.

2). Using inheritance laws for projection (R3), the selection and allows apply (R10) combination of projection, select a projection and a selection.

3). For each selection, use the law (R4, R6, R7, R10) "pushed" to allow select components to classes or "through" connection nodes and allows creation group.

4). For each projection (objects, sets, sets), using legislation (R3, R4, R5) to projection move down as far as possible. If the projected attributes include all the attributes of the expression, we remove that projection.

5). Using the law (R8,R9, R10) on the object class, to remove duplicate elements in the object class; move allows flattened (flat), lets remove duplicates in multiple files (bagtoset) ahead of the group or connection operations.

6). Creating a sequence of steps for estimating change in an order every star team for no group is evaluated, its subgroups.
7). Identify sub trees that represent groups of operations that can be executed by a single algorithm.

6. Generation of Query Execution Plans

After the query processor translates given FOQL statement to Fuzzy object Algebra it forwards that expression to query optimizer which generates various executions plans representing different orders or combinations of operators.

There are a number of algebraic laws implemented in query optimizer for generating equivalent (logical) query plans. However, following are the most commonly used techniques that we also consider and implement in our tool: Simple, Elimination of Cartesian product, and Push selection.

a) Simple Execution plan

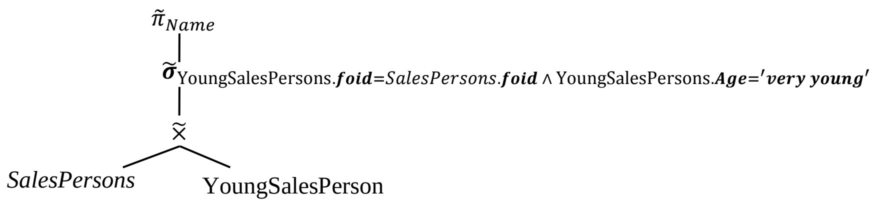
Query processor generates an equivalent relational algebraic expression for the input query and forwards it to the query optimizer. The first algebraic expression generated by the query processor involves Cartesian product that we call simple execution plan.

Example 1: ***return names of*** YoungSalesPersons ***who earned Age 'very young'***
In FOQL it can be represented as: SELECT YoungSalesPersons FROM YoungSalesPerson, *SalesPersons* WITH 0.5
WHERE YoungSalesPerson.FIOD= *SalesPersons.FOID* AND YoungSalesPersons. Age = very young WITH 0.8.

In *Fuzzy object Algebra* above FOQL statement is represented as:

$$\tilde{\pi}_{Name}\left(\begin{array}{c}\tilde{\sigma}_{\text{YoungSalesPersons}.\boldsymbol{foid}=SalesPersons.\boldsymbol{foid}\,\wedge\,\text{YoungSalesPersons}.\boldsymbol{Age}='\boldsymbol{veryyoung}'}\\ (\text{YoungSalesPersons}\ \tilde{\times}\ SalesPersons)\end{array}\right)$$

Figure. 3 show the algebraic tree of the above expression:



$\tilde{\pi}_{Name}$
$\tilde{\sigma}_{\text{YoungSalesPersons}.\boldsymbol{foid}=SalesPersons.\boldsymbol{foid}\,\wedge\,\text{YoungSalesPersons}.\boldsymbol{Age}='\boldsymbol{very\ young}'}$
$\tilde{\times}$
SalesPersons
YoungSalesPerson

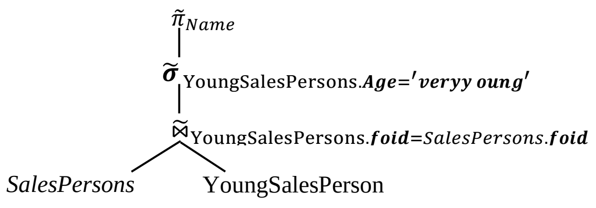
Figure 3. Simple execution plan involving Cartesian product

b) Elimination of Cartesian Product

Cartesian product operations can be combined with selection operations (and sometimes, with projection operations) which use data from both relations to form joins. After replacing Cartesian product with Join, relational algebra for the query given in example 1 can be represented as:

$$\tilde{\pi}_{Name}\left(\tilde{\sigma}_{\text{YoungSalesPersons}.\boldsymbol{Age}='\boldsymbol{veryyoung}'}\left(\begin{array}{c}\text{YoungSalesPersons}\ \tilde{\bowtie}_{\text{YoungSalesPersons}.\boldsymbol{foid}=SalesPersons.\text{fo}\boldsymbol{id}}\\ (\text{YoungSalesPersons})\end{array}\right)\right)$$

Figure 4 below gives the operator tree of the above algebraic expression:



$\tilde{\pi}_{Name}$
$\tilde{\sigma}_{\text{YoungSalesPersons}.\boldsymbol{Age}='\boldsymbol{veryy\ oung}'}$
$\tilde{\bowtie}_{\text{YoungSalesPersons}.\boldsymbol{foid}=SalesPersons.\boldsymbol{foid}}$
SalesPersons
YoungSalesPerson

Figure 4. Execution plan using Join

c) Push Selection

Selections can be pushed down the expression tree as far as they can be pushed. By pushing selection operation down, we are actually decreasing the size of relations with which we need to work earlier. Relational algebra for the query written in example 1under push selection strategy can be represented as:

$$\tilde{\pi}_{Name}\left(\text{YoungSalesPersons}\ \tilde{\bowtie}_{\text{YoungSalesPersons}.\boldsymbol{foid}=SalesPersons.\text{fo}\boldsymbol{id}}\left(\begin{array}{c}\tilde{\sigma}_{\text{YoungSalesPersons}.\boldsymbol{Age}='\boldsymbol{veryyoung}'}\\ (\text{YoungSalesPersons})\end{array}\right)\right)$$

Figure. 5 below is the operator tree of the above mentioned *Fuzzy object Algebra* expression:

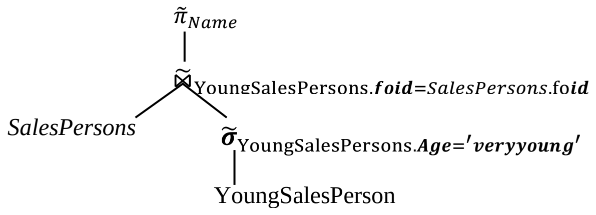

$\tilde{\pi}_{Name}$
$\tilde{\bowtie}_{\text{YoungSalesPersons.}\boldsymbol{foid}=SalesPersons.\text{fo}\boldsymbol{id}}$
SalesPersons
$\tilde{\sigma}_{\text{YoungSalesPersons.}\boldsymbol{Age}='\boldsymbol{veryyoung}'}$
YoungSalesPerson

Figure 5. Pushing selection down the tree

IV. PERFORMANCE EVALUATION

To provide preliminary performance evaluation on implementation of query processing has been proposed based on fuzzy object algebra [1]. We defined the three queries processed condition extract filter data for two cases of single conditions, most conditions and implement them with different sized data. Rating_final.csv data file to use for the evaluation have reference to the UCI web site.

The fuzzy query processor first extract filter data for single condition processing cases. Request query processing engine return all employees age is very young. Such queries are written as follows.

SELECT * FROM YoungSalesPerson, SalesPersons WITH 0.5

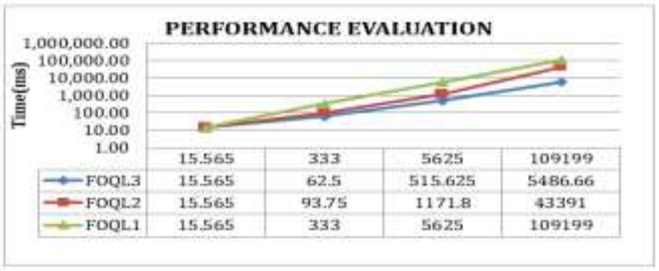
WHERE AND YoungSalesPersons. Age = ‘very young’ WITH 0.8.

The second query processing the extract filter data for single-case conditions and enable a natural join. Request query processing engine return all employees age is very young. Such queries are written as follows.

SELECT Name FROM YoungSalesPerson, SalesPersons WITH 0.5 WHERE YoungSalesPerson.FIOD= SalesPersons.FOID AND YoungSalesPersons. Age = ‘very young’ WITH 0.8.

The third query processing the extract filter data for single-case conditions and enable a natural join. After performing the optimization algebra objects. Request query processing engine return all employees age is very young. Such queries are written as follows.

SELECT Name FROM SalesPersons inner join YoungSalesPerson on YoungSalesperson.FIOD= SalesPersons.FOID WITH 0.5 WHERE YoungSalesPersons. Age = ‘very young’ WITH 0.8.



	15.565	333	5625	109199
FOQL3	15.565	62.5	515.625	5486.66
FOQL2	15.565	93.75	1171.8	43391
FOQL1	15.565	333	5625	109199

Figure 6. Query performance

Three queries to be made against the four data sets of different sizes. The first query processing on file available is 10KB and contains 56 objects; the second file has a file size of about 100 KB and contained 596 objects; the third file contains about 1 MB contains 6052 objects; the fourth file has about 10 MB and contains 60,972 object. Performance assessment testing is done on a computer is intel (R) Core (TM) i3 370 M.

From the above experiments, results achieved confirm that the performance of this method is effective. As an example, we evaluate the query according to this approach from the chart the way the query results shown in Figure. 6. In this case, the graphic shows an increase in linear time with respect to the different data dimensions.

V. CONCLUSION

In this paper, we presented a tool that translates user queries (entered in FOQL) into mathematical representation (Fuzzy object Algebra) necessary for data FOODB access. It has been improved better at: producing syntax and semantic errors, generating query execution plans: (i) Simple, (ii) elimination of Cartesian product with join, and (iii) push selection, allowing connectivity with databases of interest and produces query results. Query optimizer of the tool produces execution times of all three execution plans considered. Based on the obtained study results, it is concluded that push-selection is better than the join and join is better than Cartesian product in terms of execution time.

REFERENCES

[1] Truong Ngoc Chau, Nguyen Tan Thuan, "Một hướng tiếp cận mới trong đại số đối tượng mờ", kỷ yếu, hội thảo Quốc Gia lần thứ XVI một số vấn đề chọn lọc của CNTT & TT – Đà Nẵng, 11-2013,204-209.

[2] Selee Na, "A Process of Fuzzy Query on New Fuzzy Object Oriented Data Model", In IEEE Tranon Knowledge and Data Engineering, 1, 501-509. 2010.

[3] Stefano Ceri, Georg Gottlob,"Translating SQL Into Relational Algebra: Optimization, Semantics, and Equivalence of SQLQueries", Software Engineering, IEEE Transactions, vol. SE-11, issue 4, pp. 324 – 345, Apr. 1985

[4] XU Silao,HONG Mei,"Translating SQL Into Relational Algebra Tree-Using Object-Oriented Thinking to Obtain Expression Of Relational Algebra", IJEM, vol.2, no.3, pp.53-62, 2012.

[5] Doan Van Ban, Ho Cam Ha, Vu Duc Quang, "Querying Fuzzy Object-Oriented Data Based On Fuzzy Association Algebra", Software Engineering, IEEE Transactions, vol. SE-11, issue 4, pp. 40 – 46, 2011

[6] Bendre M. "Relational algebra translator", 2013. http://www.slinfo.una.ac.cr/rat/rat.html.

[7] Sonia. "Fuzzy Object Oriented Database versus FRDB for Uncertainty Management", International Journal of Computer Applications (0975 – 8887),Volume 74– No.17, July 2013.

[8] Nitesh Kumar,Sumanta Nikhilesh Satpathy,"An algebraic operation in fuzzy object-oriented databases (foodbs)", Journal of Global Research in Computer Science, Volume 4, No. 12, December 2013.

[9] Gloria Bordogna, "A Fuzzy Object-Oriented Data Model for Managing Vague and Uncertain Information", International journal of intelligent systems, vol. 14, 623-651 1999.

[10] STRA UBE, D., AND O ZSU, M. 1990a. Queries and query processing in object-oriented database systems. ACMTransactions on Information Systems 8, 4, (Oct.), 387–430.

[11] Selee Na, "A Fuzzy Association Algebra Based on A Fuzzy Object Oriented Data Model", Software Engineering, IEEE Transactions, 276-281. 1999.

[12] E.Bertino,"Optimization of Queries using Nested Indices", in Proc.EDBT, LNCS, vol.416,pp. 44-59, Springer 1990.

[13] P,G. Selinger, M.M. Astranhan, D.D. Chamberlin, R.A. Lorie, and T.G. Price,"Access Path Selection in a Relational Database Management System", Proc. ACM SIGMOD, pp.23-34, Boston, MA., May-June 1979.

[14] A. Swami,"Optimization of Large Jion Queries: Combining Huristics and Combinatorial Techiques", Proc. ACM SIGMOD, p.367, Porland, OR, May-Jine 1989.

[15] A. Swami and A. Gupta, "Optimization of Large Jion Queries", Proc. ACM SIGMOD,pp.8-17,Chicago, IL., June 1988.

TỐI ƯU HÓA PHÂN TÍCH VÀ XỬ LÝ TRUY VẤN MỜ TRÊN CƠ SỞ DỮ LIỆU HƯỚNG ĐỐI TƯỢNG MỜ

Nguyễn Tấn Thuân, Đoàn Văn Ban, Trương Ngọc Châu, Trần Thị Thúy Trinh

TÓM TẮT— *Mô hình cơ sở dữ liệu hướng đối tượng mờ, là một mô hình phức tạp mà ở đó nó có thể xử lý câu truy vấn từ người sử dụng bằng cách thực hiện việc chuyển đổi nội bộ bên trong hệ thống nhằm truy cập dữ liệu và cung cấp kết quả theo cách tốt nhất có thể: qua đó kết quả trả về mất ít nhất thời gian thực hiện cũng như việc tiêu tốn tài nguyên là ít hơn. Tuy nhiên, với dữ liệu ngày càng gia tăng với tốc độ nhanh chóng, việc đáp ứng các yêu cầu luôn là thách thức cho mọi hệ thống. Do đó hệ thống tốt hơn chỉ có thể phát triển nếu việc xử lý nội bộ của hệ thống đó được đánh giá là rất hiệu quả. Bài báo này đề xuất hướng tiếp cận mới tối ưu hóa biểu thức đại số đối tượng mờ nhằm hỗ trợ hệ thống thực hiện việc chuyển đổi và xử lý truy vấn cần thiết cho việc truy cập dữ liệu. Hơn nữa, các kết quả nghiên cứu của bài báo đã thực hiện một số kỹ thuật tối ưu hóa để nâng cao khả năng thực hiện cho hệ thống. Cuối cùng bài báo trình bày thảo luận kế hoạch thực hiện tốt hơn dựa trên tính toán phân tích.*