A New Approach for Query Processing and Optimization in Fuzzy Object-Oriented Database

Thuan T. Nguyen[1,2](✉), Ban V. Doan[3], Chau N. Truong[4], and Trinh T. T. Tran[1,2]

[1] Graduate University of Science and Technology—Vietnam Academy of Science and Technology (VAST), Da Nang, Vietnam
nguyentanthuan2008@yahoo.com, thuytrinh85@gmail.com
[2] Duy Tan University, Da Nang, Vietnam
[3] Institute of Information Technology—VAST, Da Nang, Vietnam
doanban@gmail.com
[4] Da Nang University of Technology, Da Nang, Vietnam
truongngocchau@yahoo.com

Abstract. For enhancing the efficiency of processing users' queries, all database management systems (DBMSs) must conduct query preprocessing, or query optimizing. This paper proposes a new model for the Fuzzy Object-Oriented DBMS (FOO-DBMS), which optimizes the query statements and processes the data before returning back to users based on fuzzy object algebra and equivalent transformation rules. Discussions on this model are also presented with computation and analysis.

1 Introduction

The fuzzy object-oriented database (FOO-DB) model contains unclear, uncertain, and inaccurate information. Hence, in order to deal with the ever-increasing and complex data, it is always time- and resource-consuming. Typically, a database management system works well if its components perform efficient processing (less processing time, less resources). This paper proposes a new approach for optimizing and processing query based on the fuzzy object algebra (FOA) expression and the rules of equivalent transformation. This approach is the main idea of our proposed model which is analyzed and discussed later in the paper.

A few propositions have been introduced on query translation and optimization [1]. Similar research in OODB has also been presented in [2, 3] and is still very useful for the development of the concerning part of our study which is used to translate user queries from FOQL to fuzzy object algebra.

V. Bhateja et al. (eds.), *Information Systems Design and Intelligent Applications*, Advances in Intelligent Systems and Computing 672,
https://doi.org/10.1007/978-981-10-7512-4_6

The remaining part of the paper includes: part 2 that introduces about fuzzy algebraic operations in FOODB, part 3 proposes a new model for the fuzzy object-oriented DBMS (FOO-DBMS), which optimizes the query statements and processes the data before returning back to users based on fuzzy object algebra and equivalent transformation rules, the analysis on several experiments using the proposed algorithm shows better performance of query processing, which proves the efficiency enhancement of our method are presented in part 4, and the conclusion is stated in part 5.

2 Algebraic Operations in FOODB

2.1 Association Operators

A new fuzzy class $\left(\widetilde{FC}\right)$ can be created by combining two existing fuzzy classes. Depending on the relationships between the attributes of the combination classes, there are six types of binary combination and two types of unary operations which are: (1) fuzzy union $\left(\widetilde{\cup}\right)$, (2) fuzzy intersection $\left(\widetilde{\cap}\right)$, (3) fuzzy join $(\widetilde{\bowtie})$, (4) fuzzy cross-product/Cartesian $(\widetilde{\times})$, (5) fuzzy difference $(\simeq)$, (6) fuzzy division $(\widetilde{\div})$, (7) fuzzy projection $\left(\widetilde{\Pi}\right)$, and (8) fuzzy selection $(\widetilde{\sigma})$ and are denoted in [4]. Let $\widetilde{FC_1}\mathrm{and}\widetilde{FC_2}$ be any two fuzzy class with sets of attributes $F\mathrm{A}(fA_1)\mathrm{andFA}(fA_2)$, respectively. Thus, a new fuzzy class $\widetilde{FC}$ is created from the combination of two classes $\widetilde{FC_1}\&\widetilde{FC_2}$ and is defined as follows:

1. $\widetilde{FC} = \widetilde{FC}_1 \,\widetilde{\cup}\, \widetilde{FC}_2,\ \mathit{if}\ FA'\left(\widetilde{FC}_1\right) = FA'\left(\widetilde{FC}_2\right)$
2. $\widetilde{FC} = \widetilde{FC}_1 \,\widetilde{\cap}\, \widetilde{FC}_2,\ \mathit{if}\ FA'\left(\widetilde{FC}_1\right) = FA'\left(\widetilde{FC}_2\right)$
3. $\widetilde{FC} = \widetilde{FC}_1 \,\widetilde{\bowtie}\, \widetilde{FC}_2,\ \mathit{if}\ FA'\left(\widetilde{FC}_1\right) \cap FA'\left(\widetilde{FC}_2\right)$
4. $\widetilde{FC} = \widetilde{FC}_1 \,\widetilde{\times}\, \widetilde{FC}_2,\ \mathit{if}\ FA'\left(\widetilde{FC}_1\right) \cap FA'\left(\widetilde{FC}_2\right) = \emptyset\ \mathit{and}\ FA'\left(\widetilde{FC}_1\right) \neq FA'\left(\widetilde{FC}_2\right)$
5. $\widetilde{FC} = \widetilde{FC}_1 \simeq \widetilde{FC}_2,\ \mathit{if}\ FA'\left(\widetilde{FC}_1\right) = FA'\left(\widetilde{FC}_2\right)$
6. $\widetilde{FC} = \widetilde{FC}_1 \,\widetilde{\div}\, \widetilde{FC}_2,\ \mathit{if}\ FA'\left(\widetilde{FC}_1\right) = FA'\left(\widetilde{FC}_2\right)$
7. $\widetilde{FC''} = \widetilde{\Pi}_{sub}\left(\widetilde{FC'}\right)$
8. $\widetilde{FC''} = \widetilde{\sigma}_{sub}\left(\widetilde{FC'}\right)$

In which, $FA'\left(\widetilde{FC}_1\right)\&FAtr'\left(\widetilde{FC}_2\right)$ are achieved from $FA\left(\widetilde{FC}_1\right)\&FA\left(\widetilde{FC}_2\right)$, by removing the degree of their respective fuzzy attributes $FA\left(\widetilde{FC}_1\right)\&FA\left(\widetilde{FC}_2\right)$. The $\mu_{\widetilde{FC}}$ represents the membership degree of attributes which belong to class $\widetilde{FC}$. Assume we have an object $\mu_{\widetilde{FC}}(o)$ of $\widetilde{FC}$, and $\mu_{\widetilde{FC}}(o)$ is used to represent the value of o on $\mu_{\widetilde{FC}}$. Let fA_i be the attribute of class $\widetilde{FC}$ $\cdot\ \mu_{\widetilde{FC}}(o)(A_i)$ is the fuzzy value of object $\mu_{\widetilde{FC}}(o)$ on FA_i. If $FA = \{fA_i, fA_i, \ldots, fA_m\}$ is a set of fuzzy attributes, then $\mu_{\widetilde{FC}}(o)(FA)$ stands for all values of $\mu_{\widetilde{FC}}(o)$ object on attributes in FA. More generally, $\mu_{\widetilde{FC}}(o)\left(\widetilde{FC}\right)$ represents all values of o on attributes of $\mu_{\widetilde{FC}}(o)$ on attributes of $\widetilde{FC}$. The formal conditions of fuzzy association operations are shown as follows:

1. Fuzzy union $\left(\widetilde{\cup}\right)$: Through fuzzy union, new class $\widetilde{FC}$ is made up of two fuzzy classes $\widetilde{FC}_1\&\widetilde{FC}_2$, requires $FA'(fA_1) = FA'(fA_2)$, it means that all related attributes of $\widetilde{FC}_1\&\widetilde{FC}_2$ have the identical weights. Suppose, the new $\widetilde{FC}$ is the fuzzy union of $\widetilde{FC}_1\&\widetilde{FC}_2$. Then the $\mu_{\widetilde{FC}}(o)$ of $\widetilde{FC}$ are created from three types of $\mu_{\widetilde{FC}}(o)$. First, two types of $\mu_{\widetilde{FC}}(o)$ are derived from the component class (for example, class $\widetilde{FC}_1$) and do not coincide with any $\mu_{\widetilde{FC}}(o)$ in the rest of component class (e.g., $\widetilde{FC}_2$) that satisfies the given threshold. Let δ be the threshold.

$$\widetilde{FC} = \widetilde{FC}_1\ \widetilde{\cup}\ \widetilde{FC}_2$$
$$= \left\{ \mu_{\widetilde{FC}}(o) \;\middle|\; \begin{array}{c} \left(\forall \mu_{\widetilde{FC}}(o)''\right)\begin{pmatrix} \mu_{\widetilde{FC}}(o)'' \in \widetilde{FC}_2 \wedge \mu_{\widetilde{FC}}(o) \in \widetilde{FC}_1 \wedge \\ SE\left(\mu_{\widetilde{FC}}(o)\left(\widetilde{FC}_1\right), \mu_{\widetilde{FC}}(o)''\left(\widetilde{FC}_2\right)\right) < \delta \end{pmatrix} \vee \\ \left(\forall o\mu_{\widetilde{FC}}(o)'\right)\begin{pmatrix} \mu_{\widetilde{FC}}(o)' \in \widetilde{FC}_1 \wedge \mu_{\widetilde{FC}}(o) \in \widetilde{FC}_2 \wedge \\ SE\left(\mu_{\widetilde{FC}}(o)\left(\widetilde{FC}_2\right), \mu_{\widetilde{FC}}(o)'\left(\widetilde{FC}_1\right)\right) < \delta \end{pmatrix} \vee \\ \begin{pmatrix} \left(\exists \mu_{\widetilde{FC}}(o)'\right)\left(\exists \mu_{\widetilde{FC}}(o)''\right)\left(\mu_{\widetilde{FC}}(o)' \in \widetilde{FC}_1 \wedge \mu_{\widetilde{FC}}(o)'' \in \widetilde{FC}_2\right) \wedge \\ SE\left(\mu_{\widetilde{FC}}(o)'\left(\widetilde{FC}_1\right), o\mu_{\widetilde{FC}}(o)''\left(\widetilde{FC}_2\right)\right) \geq \\ \delta \wedge \mu_{\widetilde{FC}}(o) = merge\left(\mu_{\widetilde{FC}}(o)', \mu_{\widetilde{FC}}(o)''\right) \end{pmatrix} \end{array} \right\} \tag{1}$$

In which, $\mu_{\widetilde{FC}}(o)', \mu_{\widetilde{FC}}(o)''$ are two $\mu_{\widetilde{FC}}(o)$ of $\widetilde{FC}$, and $\mu_{\widetilde{FC}}(o) = merge\left(\mu_{\widetilde{FC}}(o)', \mu_{\widetilde{FC}}(o)''\right)$. Then $\mu_{\widetilde{FC}}(o)\left(\widetilde{FC}\right) = \mu_{\widetilde{FC}}(o)'\left(\widetilde{FC}\right)$ or $\mu_{\widetilde{FC}}(o)\left(\widetilde{FC}\right) = \mu_{\widetilde{FC}}(o)''\left(\widetilde{FC}\right)$ and $\mu_{\widetilde{FC}}(o) = max\left(\mu_{\widetilde{FC}_1}(o)', \mu_{\widetilde{FC}_2}(o)''\right)$.

2. Fuzzy intersection $\left(\widetilde{\cap}\right)$: Through fuzzy intersection, new class $\widetilde{FC}$ is made up of two fuzzy classes $\widetilde{FC}_1\,\&\,\widetilde{FC}_2$, and requests $FA^{'}(fA_1) = FA^{'}(fA_2)$. Let δ be the threshold.

$$\widetilde{FC} = \widetilde{FC}_1 \widetilde{\cap} \widetilde{FC}_2 = \left\{ \mu_{\widetilde{FC}}(o) \middle| \begin{array}{c} \left(\forall\mu_{\widetilde{FC}}(o)''\right)\begin{pmatrix} \mu_{\widetilde{FC}}(o)'' \in \widetilde{FC_2} \vee \mu_{\widetilde{FC}}(o) \in \widetilde{FC_1} \wedge \\ SE\left(\mu_{\widetilde{FC}}(o)\left(\widetilde{FC_1}\right), \mu_{\widetilde{FC}}(o)''\left(\widetilde{FC_2}\right)\right) < \delta \end{pmatrix} \wedge \\ \left(\forall\mu_{\widetilde{FC}}(o)'\right)\begin{pmatrix} \mu_{\widetilde{FC}}(o)' \in \widetilde{FC_1} \vee \mu_{\widetilde{FC}}(o) \in \widetilde{FC_2} \vee \\ SE\left(\mu_{\widetilde{FC}}(o)\left(\widetilde{FC_2}\right), \mu_{\widetilde{FC}}(o)'\left(\widetilde{FC_1}\right)\right) < \delta \end{pmatrix} \wedge \\ \left(\begin{array}{c} \left(\exists\mu_{\widetilde{FC}}(o)'\right)\left(\exists\mu_{\widetilde{FC}}(o)''\right)\begin{pmatrix} \mu_{\widetilde{FC}}(o)' \in \widetilde{FC_1} \vee \mu_{\widetilde{FC}}(o)'' \in \widetilde{FC_2} \vee \\ SE\left(\mu_{\widetilde{FC}}(o)'\left(\widetilde{FC_1}\right), \mu_{\widetilde{FC}}(o)''\left(\widetilde{FC_2}\right)\right) \end{pmatrix} \geq \\ \delta \wedge \mu_{\widetilde{FC}}(o) = minimize\left(\mu_{\widetilde{FC}}(o)', \mu_{\widetilde{FC}}(o)''\right) \end{array} \right) \end{array} \right\} \tag{2}$$

In which, $\mu_{\widetilde{FC}}(o)', \mu_{\widetilde{FC}}(o)''$ are two $\mu_{\widetilde{FC}}(o)$ of $\widetilde{FC}$ and $\mu_{\widetilde{FC}}(o) = merge\left(\mu_{\widetilde{FC}}(o)^{'}, \mu_{\widetilde{FC}}(o)''\right)$. We have $\mu_{\widetilde{FC}}(o)\left(\widetilde{FC}\right) = \mu_{\widetilde{FC}}(o)^{'}\left(\widetilde{FC}\right)$ or $\mu_{\widetilde{FC}}(o)\left(\widetilde{FC}\right) = \mu_{\widetilde{FC}}(o)''\left(\widetilde{FC}\right)$ and $\mu_{\widetilde{FC}}(o) = minimize\left(\mu_{\widetilde{FC}_1}(o)', \mu_{\widetilde{FC}_2}(o)''\right)$.

3. Fuzzy join $(\widetilde{\bowtie})$: Through fuzzy join, new class $\widetilde{FC}$ is made up of two fuzzy classes $\widetilde{FC}_1\&\widetilde{FC}_2$, in which $FA^{'}\left(\widetilde{FC}_1\right) \cap FA^{'}\left(\widetilde{FC}_2\right) \neq \emptyset$ *and* $FA^{'}\left(\widetilde{FC}_1\right) \neq FA^{'}\left(\widetilde{FC}_2\right)$. $FA^{'}\left(\widetilde{FC}_1\right) \cup \left(FA^{'}\left(\widetilde{FC}_2\right) - \left(FA^{'}\left(\widetilde{FC}_1\right) \cap FA^{'}\left(\widetilde{FC}_2\right)\right)\right)$ are attributes for $\widetilde{FC}$ and the membership degree of the attributes. $\mu_{\widetilde{FC}}(o)$ of $\widetilde{FC}$ are made up of a combination of $\mu_{\widetilde{FC}}(o)$ in $\widetilde{FC}_1\&\widetilde{FC}_2$ that are equivalent in semantics on $FA^{'}\left(\widetilde{FC}_1\right) \cap FA^{'}\left(\widetilde{FC}_2\right)$ under a given threshold. Let δ be the threshold.

$$\begin{aligned} &\widetilde{FC} = \widetilde{FC}_1 \widetilde{\bowtie} \widetilde{FC}_2 \\ &= \left\{ \mu_{\widetilde{FC}}(o) \middle| \begin{array}{c} \left(\exists\mu_{\widetilde{FC}}(o)'\right)\left(\exists\mu_{\widetilde{FC}}(o)''\right)\mu_{\widetilde{FC}}(o)' \in \widetilde{FC}_1 \wedge \mu_{\widetilde{FC}}(o)''\widetilde{FC}_2 \wedge \\ \left(\begin{array}{c} SE\begin{pmatrix} \mu_{\widetilde{FC}}(o)'\left(FA'\left(\widetilde{FC}_1\right) \cap FA'\left(\widetilde{FC}_2\right)\right), \\ \mu_{\widetilde{FC}}(o)''\left(FA'\left(\widetilde{FC}_1\right) \cap FA'\left(\widetilde{FC}_2\right)\right) \end{pmatrix} \geq \delta \wedge \mu_{\widetilde{FC}}(o)\left(FA'\left(\widetilde{FC}_1\right)\right) = \\ \mu_{\widetilde{FC}}(o)'\left(\widetilde{FC}_1\right) \wedge \mu_{\widetilde{FC}}(o)\left(FA'\left(\widetilde{FC}_2\right) - \left(FA'\left(\widetilde{FC}_1\right) \cap FA'\left(\widetilde{FC}_2\right)\right)\right) = \\ \mu_{\widetilde{FC}}(o)''\left(FA'\left(\widetilde{FC}_2\right) - \left(FA'\left(\widetilde{FC}_1\right) \cap FA'\left(\widetilde{FC}_2\right)\right)\right) \wedge \mu_{\widetilde{FC}}(o) = \\ op\left(\left(\mu_{\widetilde{FC}_1}(o)', \mu_{\widetilde{FC}_2}(o)''\right)\right) \end{array} \right) \end{array} \right\} \end{aligned} \tag{3}$$

In which, operations of op are not defined. In most cases, $op\left(\mu_{\widetilde{FC}_1}(o)', \mu_{\widetilde{FC}_2}(o)''\right)$ either $min\left(\mu_{\widetilde{FC}_1}(o)', \mu_{\widetilde{FC}_2}(o)''\right)$ or $\mu_{\widetilde{FC}_1}(o)' \times \mu_{\widetilde{FC}_2}(o)'$.

4. Fuzzy cross-product/Cartesian $(\widetilde{\times})$: Through fuzzy product, new class $\widetilde{FC}$ is made up of two fuzzy classes $\widetilde{FC}_1 \& \widetilde{FC}_2$ and membership degree attribute.

$$\widetilde{FC} = \widetilde{FC}_1 \,\widetilde{\times}\, \widetilde{FC}_2 = \left\{ \mu_{\widetilde{FC}}(o) \left| \begin{array}{c} \left(\forall \mu_{\widetilde{FC}}(o)'\right)\left(\forall \mu_{\widetilde{FC}}(o)''\right)\mu_{\widetilde{FC}}(o)' \in \widetilde{FC}_1 \wedge \mu_{\widetilde{FC}}(o)\left(FA'\left(\widetilde{FC}_1\right)\right) = \\ \left(\begin{array}{c} \mu_{\widetilde{FC}}(o)'\left(\widetilde{FC}_1\right) \wedge \mu_{\widetilde{FC}}(o)\left(FA'\left(F\widetilde{C}_2\right)\right) = \\ \mu_{\widetilde{FC}}(o)''\left(\widetilde{FC}_1\right) \wedge \mu_{\widetilde{FC}}(o) = op\left(\mu_{\widetilde{FC}_1}(o)', \mu_{\widetilde{FC}_2}(o)''\right) \end{array} \right) \end{array} \right. \right\} \tag{4}$$

In which, operations of op are not defined. In most cases, $op\left(\mu_{\widetilde{FC}_1}(o)', \mu_{\widetilde{FC}_2}(o)''\right)$ either $min\left(\mu_{\widetilde{FC}_1}(o)', \mu_{\widetilde{FC}_2}(o)''\right)$ or $\mu_{\widetilde{FC}_1}(o)' \times \mu_{\widetilde{FC}_2}(o)'$.

5. Fuzzy difference $(\simeq)$: The fuzzy difference of $\widetilde{FC}_1 \& \widetilde{FC}_2$ requests $FA'(fA_1) = FA'(fA_2)$, which pulls all the properties in $\widetilde{FC}_1 \& \widetilde{FC}_2$ with the same weight numbers. For new class, $\widetilde{FC}$ is the fuzzy subtraction of $\widetilde{FC}_1 \& \widetilde{FC}_2$. Then, the objects of $\widetilde{FC}$ are made up of two types of objects. The first type of object is derived from class $\widetilde{FC}_1$ and does not appear in $\widetilde{FC}_2$, satisfying the given threshold. The second type of object obtained by moving duplicate objects in $\widetilde{FC}_1$ satisfies the given threshold. Let δ be the threshold.

$$\widetilde{FC} = \widetilde{FC_1} \simeq \widetilde{FC}_2 = \left\{ \mu_{\widetilde{FC}}(o) \left| \begin{array}{c} \left(\forall \mu_{\widetilde{FC}}(o)''\right) \left(\begin{array}{c} \mu_{\widetilde{FC}}(o)'' \in \widetilde{FC}_2 \wedge \mu_{\widetilde{FC}}(o) \in \widetilde{FC}_1 \wedge \\ SE\left(\mu_{\widetilde{FC}}(o)\left(\widetilde{FC}_1\right), \mu_{\widetilde{FC}}(o)''\left(\widetilde{FC}_2\right)\right) < \delta \end{array} \right) \vee \\ \left(\begin{array}{c} \left(\exists \mu_{\widetilde{FC}}(o)'\right)\left(\exists \mu_{\widetilde{FC}}(o)''\right)\left(\mu_{\widetilde{FC}}(o)' \in \widetilde{FC}_1 \wedge \mu_{\widetilde{FC}}(o)'' \in \widetilde{FC}_2\right) \wedge \\ SE\left(\mu_{\widetilde{FC}}(o)'\left(\widetilde{FC}_1\right), \mu_{\widetilde{FC}}(o)''\left(\widetilde{FC}_2\right)\right) \geq \delta \wedge \mu_{\widetilde{FC}}(o) = \\ remove\left(\mu_{\widetilde{FC}}(o)', \mu_{\widetilde{FC}}(o)''\right) \end{array} \right) \end{array} \right. \right\} \tag{5}$$

6. Fuzzy division $(\widetilde{\div})$: The fuzzy division is written as $(\widetilde{\div})$, to extract data tuples of a relational relation with all remaining relations.

$$\widetilde{FC} = \widetilde{FC}_1 \tilde{\div} \widetilde{FC}_2 = \left\{\mu_{\widetilde{FC}}(o)[fA_1, \ldots, fA_n] : \mu_{\widetilde{FC}}(o) \in \widetilde{FC}_1 \wedge \forall \widetilde{FC}_2 \in \widetilde{FC}_2\left(\left(\mu_{\widetilde{FC}}(o)\left[fA_1, \ldots, fA_n \cup \widetilde{FC}_2\right) \in \widetilde{FC}_1\right)\right\} \tag{6}$$

In which, the $\tilde{\div}$ of $\widetilde{FC}_1$ *and* $\widetilde{FC}_2$ requires $FA'(fA_1) = FA'(fA_2)$, and $\widetilde{FC}$ is composed with attribute $[fA_1, \ldots, fA_n]$ and $\mu_{\widetilde{FC}}(o)[fA_1, \ldots, fA_n]$ is the limitation of $\mu_{\widetilde{FC}}(o)$.

7. Fuzzy projection $\left(\tilde{\Pi}\right)$: The new $\widetilde{FC}''$ composed with membership degree attribute is created from the fuzzy projection on the attribute subset sub.

$$\begin{aligned}\widetilde{FC}'' &= \tilde{\Pi}_{sub}\left(\widetilde{FC}'\right)\\ &= \left\{\mu_{\widetilde{FC}}(o) | \left(\forall \mu_{\widetilde{FC}}(o)'\right)\left(\mu_{\widetilde{FC}}(o)' \in \widetilde{FC}' \wedge \mu_{\widetilde{FC}}(o) = \cup_{\Im} \mu_{\widetilde{FC}}(o)'\right)\right\}\end{aligned} \tag{7}$$

Here, the removal of redundant $\mu_{\widetilde{FC}}(o)$ in the fuzzy set of $\mu_{\widetilde{FC}}(o)'$ is used by the $\cup_{\Im} \mu_{\widetilde{FC}}(o)'$ operation.

8. Fuzzy selection ($\tilde{\boldsymbol{\sigma}}$): The new $\widetilde{FC}''$ composed with membership degree attribute is created from the fuzzy selection on the attribute subset sub.

$$\widetilde{FC}'' = \tilde{\sigma}_{sub}\left(\widetilde{FC}'\right) = \left\{\mu_{\widetilde{FC}}(o) | \mu_{\widetilde{FC}}(o) \in \widetilde{FC}', \varphi\left(\mu_{\widetilde{FC}}(o)\right)\right\} \tag{8}$$

Here, the removal of redundant $\mu_{\widetilde{FC}}(o)$ in the fuzzy set of $\mu_{\widetilde{FC}}(o)'$ is used by the $\varphi o'$ operation.

2.2 Structured Fuzzy Object Query Language

Structure of a fuzzy object-oriented query consisting of three clauses:

SELECT < attribute list > FROM < Class WITH threshold Class WITH threshold>

WHERE < query condition WITH threshold > .

Example for a $\widetilde{FC}$ OldSalesPersons as follows:

CLASS OldSalesPersons 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, very old}:

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) = 0.1w (Name) = 0.1w (Age) = 0.9w (Sex) = 0.1w (DOB) = 0.6

METHODS END

A query based on the class is issued by using:

SELECT Name

FROM OldSalesPerson as O, SalesPersons as S WITH 0.6

WHERE O.foid = S. foid AND O. Age = ‘very old’ WITH 0.7.

3 Proposed Fuzzy Query Processing Architectures and Optimization

To unify the execution steps of a query in a certain process, we propose a query processing architecture for FOODB, represented in Fig. 1.

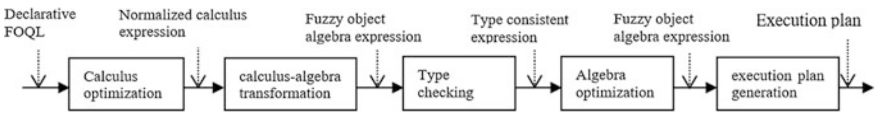

Declarative FOQL
Normalized calculus expression
Fuzzy object algebra expression
Type consistent expression
Fuzzy object algebra expression
Execution plan
Calculus optimization
calculus-algebra transformation
Type checking
Algebra optimization
execution plan generation

Fig. 1. Fuzzy query processing architecture

When the user submits a FOQL, it is first parsed by the FOODBSs, which verify the syntax and type correctness of the FOQL. Being a declarative language, FOQL does not suggest concrete ways to evaluate its queries. Therefore, a parsed FOQL has to be converted into a fuzzy object algebra expression, which can be evaluated directly using the algorithms. A typical FOQL query such as

SELECT Name FROM OldSalesPerson as O, SalesPersons as S WITH 0.6 WHERE O.foid = S.foid AND O. Age = ‘very old’ WITH 0.7.

Is normally translated into the following fuzzy object algebraic expression:

$$\tilde{\pi}_{Name}\left(\begin{array}{c}\tilde{\sigma}_{\text{OldSalesPersons}.\boldsymbol{foid}=SalesPersons.\boldsymbol{foid}\wedge\text{OldSalesPersons}.\boldsymbol{Age}='\boldsymbol{very}\ old'}\\(\text{OldSalesPersons}\ \tilde{\bowtie}\ SalesPersons)\end{array}\right)$$

The above architecture is described through the steps as follows. First, users submit a fuzzy query that they do not need knowledge of fuzzy objects. Next, the system receives the fuzzy query, performs the removal of duplicate predicates, and applies the identities and rewriting. Next, the system performs the conversion of the query to the fuzzy algebraic expression. Next, the system expresses this fuzzy algebraic expression a nested expression in the form of a fuzzy algebraic tree. Next in the fuzzy query processing process is to apply the same rewrite rules equivalent to preserving the equivalent of algebraic expressions for fuzzy–fuzzy objects. Finally, an implementation plan that takes into account the implementation of fuzzy objects is generated from the optimized fuzzy algebraic expression.

3.1 Equivalent Transformation Rules

Assume that $\mu_{\widetilde{FC}}(o), \mu_{\widetilde{FC}_1}(o), \mu_{\widetilde{FC}_2}(o), \mu_{\widetilde{FC}_3}(o)$ are the set for fuzzy object: e, f, g, h are algebraic expressions, operations $op \in \{union, diff\}$. These rules apply only on the fuzzy object operations, math sets, set operations, and multiset operations (bag). On signs, we only use math notations in a form [4] operations can be setup with changes in a number of different models.

R1. Selection operations are commutative:

$$\sigma_{\lambda t.g}\left(\sigma_{\lambda s.f}\left(\mu_{\widetilde{FC}}(o)\right)\right) = \sigma_{\lambda s.f}\left(\sigma_{\lambda s.g}\left(\mu_{\widetilde{FC}}(o)\right)\right)$$

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

$$\sigma_{\lambda s.(f \wedge g \wedge \ldots h)}\left(\mu_{\widetilde{FC}}(o)\right) = \sigma_{\lambda s.f}\left(\sigma_{\lambda t.g}\left(\cdots\left(\sigma_{\lambda u.h}\left(\mu_{\widetilde{FC}}(o)\right)\right)\cdots\right)\right)$$

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

$$\Pi_{(a_1,\,\ldots,\,a_n)}\left(\Pi_{(b_1,\,\ldots,\,b_n)}\left(\mu_{\widetilde{FC}}(o)\right)\right) = \Pi(a_1,\ \ldots,\ a_n)\left(\mu_{\widetilde{FC}}(o)\right)|\{a_1,\ldots,a_n\} \subset \{b_1,\ldots,b_n\}$$

R4. Permutation selection and projection:

$$\sigma_{\lambda s.e}\left(\Pi_{(a_1,\ldots,a_n)}\left(\mu_{\widetilde{FC}}(o)\right)\right) = \Pi_{(a_1,\ldots,a_n)}\left(\sigma_{\lambda s.e}\left(\mu_{\widetilde{FC}}(o)\right)\right)$$

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

$$\Pi_{(a_1,\ldots,a_n)}\left(\mu_{\widetilde{FC}_1}(o) op\, \mu_{\widetilde{FC}_2}(o)\right) = \Pi_{(a_1,\ldots,a_n)}\mu_{\widetilde{FC}_1}(o)\; op\; \Pi_{(a_1,\ldots,a_n)}\mu_{\widetilde{FC}_2}(o)$$

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

$$\begin{aligned}&\sigma_{\lambda s\cdot f}\left(\mu_{\widetilde{FC}_1}(o)op\,\mu_{\widetilde{FC}_2}(o)\right)\\&=\sigma_{\lambda s\cdot f}\left(\mu_{\widetilde{FC}_1}(o)\right)op\,\mu_{\widetilde{FC}_2}(o),\ \textit{if f is related to}\ \mu_{\widetilde{FC}_1}(o)\end{aligned}$$

Generality:

$$\begin{aligned}&\sigma_{\lambda s\cdot(f\wedge g\wedge h}\left(\mu_{\widetilde{FC}_1}(o)op\,\mu_{\widetilde{FC}_2}(o)\right)\\&=\sigma_{\lambda u\cdot h}\left(\sigma_{\lambda s\cdot f}\left(\mu_{\widetilde{FC}_1}(o)\right)op\,\sigma_{\lambda t\cdot g}\left(\mu_{\widetilde{FC}_2}(o)\right)\right),\ \textit{if f is related to}\ \mu_{\widetilde{FC}_1}(o),\\&\textit{g is related to}\ \mu_{\widetilde{FC}_2}(o)\ \textit{and}\ \text{h related to both}\ \mu_{\widetilde{FC}_1}(o)\ \text{and}\ \mu_{\widetilde{FC}_2}(o)\end{aligned}$$

R7. Permutation between selection operation and apply operation: if conditions only contain attributes selected by the operation returns apply:

$$apply_{\lambda_{\mu s\cdot e}}\left(\sigma_{\lambda t\cdot f}\left(\mu_{\widetilde{FC}}(o)\right)\right)=\sigma_{\lambda t\cdot f}\left(apply_{\lambda_{\mu s\cdot e}}\left(\mu_{\widetilde{FC}}(o)\right)\right)$$

R8. Permutation between flat and apply on set and multiset: suppose that $\mu_{\widetilde{FC}}(o)$ is an instance of a class and x is a complex set of attributes of the class:

$$\begin{aligned}&flat\left(apply_{\lambda s\cdot\left(apply_{\lambda t\cdot e}\left(\Pi_{(X)}\left(\Pi_V\left(\mu_{\widetilde{FC}}(o)\right)\right)\right)\right)}\left(\mu_{\widetilde{FC}}(o)\right)\right)\\&=apply_{\lambda t\cdot e}\left(flat\left(apply_{\lambda s\cdot\,\Pi(X)}\left(\Pi V\left(\mu_{\widetilde{FC}}(o)\right)\right)\left(\mu_{\widetilde{FC}}(o)\right)\right)\right)\end{aligned}$$

R9. Set union is associative:

$$\left(\mu_{\widetilde{FC}_1}(o)\,union\,\mu_{\widetilde{FC}_2}(o)\right)union\mu_{\widetilde{FC}_3}(o)\mu_{\widetilde{FC}_1}(o)\,union\left(\mu_{\widetilde{FC}_2}(o)\,union\mu_{\widetilde{FC}_3}(o)\right)$$

R10. The inheritance laws to allow the selection and apply: if $\widetilde{FC}_2$ is a subclass of $\widetilde{FC}_1$, instance of $\mu_{\widetilde{FC}_2}(o)$ is a subset of instance of $\mu_{\widetilde{FC}_1}(o)$.

$$\begin{aligned}&\sigma_{\lambda s\cdot f}\left(\mu_{\widetilde{FC}_1}(o)\right)union\sigma_{\lambda s\cdot f}\left(\mu_{\widetilde{FC}_1}(o)\right)=\sigma_{\lambda s\cdot f}\left(\mu_{\widetilde{FC}_1}(o)\right)apply_{\lambda s\cdot e}\left(\mu_{\widetilde{FC}_1}(o)\right)\\&union\,apply_{\lambda s\cdot e}\left(\mu_{\widetilde{FC}_2}(o)\right)=apply_{\lambda s\cdot e}\left(\mu_{\widetilde{FC}_1}(o)\right)\end{aligned}$$

3.2 FOQL to Fuzzy Object Algebra Translation

The transformation equivalence between FOQL queries and fuzzy object algebra.

Definition. If E is a fuzzy object algebraic expression and Q fuzzy query object is FOQL together define sets of fuzzy object, we say Q represent E and the opposite; we call E equivalent to Q. Symbol $E \approx Q$.

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

Theorems 1. Every algebraic expression is fuzzy object represented by the object query in FOQL.

Theorems 2. Every fuzzy object in FOQL queries are represented by algebraic expressions fuzzy object

Thus, 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, 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. So, we wanted to prove that the transformation preserved on a basis equivalent algebraic fuzzy objects that may be acceptable. Some transformation rules is presented [4].

3.3 Heuristic Optimization Based on Algebraic Equivalences

1. Search space and transformation rules.

The most important advantage of processing and optimizing fuzzy algebra is that through the algebraic expression of the object, we can use algebraic properties such as transformation, distribution. Therefore, each fuzzy query has the number of different equivalent expressions that depend on the input of the query from the user's request. These expressions are corresponding to the results they created, but different from their costs. However, fuzzy query optimizers modify fuzzy query expressions by using algebraic transformation rules to achieve the same results at the possible lowest cost. The transformation rules depend a lot on specific objects, as they are determined properly for each object algebra and their combinations.

2. Search algorithm

Heuristic

1. The parser of a high-level fuzzy query creates an initial internal representation;
2. Apply heuristics rules to optimize the internal representation.

3. A fuzzy query execution plan is generated to execute groups of operations based on the access paths available on the files involved in the fuzzy query.
   The main heuristic is to apply first the operations that reduce the size of intermediate results.
   For example, 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 the selection components to classes or "through" connection nodes and allows creation group.
4. For each projection (objects, sets, sets), use 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.

3.4 Fuzzy Query Execution Plans

After translating the fuzzy query into a fuzzy algebra expression, the query processor passes the expression to the query optimizer, generates different execution plans, or a combination of operators.

There are some of algebraic transformations that are performed in the query optimizer to create equivalent (rational) query plans. By removing $(\widetilde{\times})$ and Push $(\widetilde{\boldsymbol{\sigma}})$.

1. Execution plan

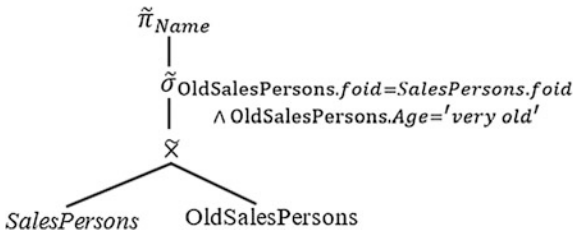
The query processor converts the query to an equivalent fuzzy algebra expression for the input query and forward it to the query optimizer. The first fuzzy algebraic object created by the query processor involves in the fuzzy Cartesian product called execution plan execution.

Example 1: returns the list of names of old salespersons whose Age are 'very old.'
In FOQL, it can be represented as:
SELECT Name FROM OldSalesPerson as O, SalesPersons as S WITH 0.6
WHERE O.foid = S.foid AND O. Age = 'very old' WITH 0.7.

In Fuzzy object Algebra above FOQL statement is represented as:

$$\tilde{\boldsymbol{\pi}}_{Name}\left(\begin{array}{c}\tilde{\boldsymbol{\sigma}}_{\text{OldSalesPersons}.\boldsymbol{foid}=salesPersons.\boldsymbol{foid}\wedge\text{OldSalesPersons}.\boldsymbol{Age}='\boldsymbol{very}\ old'}\\(SalesPersons\ \tilde{\times}\ \text{OldSalesPersons})\end{array}\right)$$

The above expression is represented by the algebraic tree as Fig. 2:



$\tilde{\pi}_{Name}$
$\tilde{\sigma}_{\text{OldSalesPersons}.foid=SalesPersons.foid\ \wedge\ \text{OldSalesPersons}.Age='very\ old'}$
$\tilde{\times}$
SalesPersons
OldSalesPersons

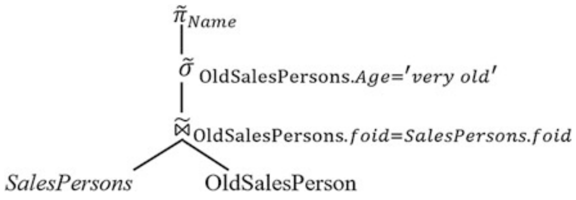
Fig. 2. Implementation plan concern in $(\tilde{\times})$

2. Elimination of $(\tilde{\times})$

The $(\tilde{\times})$ operations can be combined with $(\tilde{\boldsymbol{\sigma}})$ operations (and sometimes, with fuzzy projection operations) which use data from both relations to form joins. After replacing $(\tilde{\times})$ with $(\tilde{\bowtie})$, fuzzy object algebra for the query given in example 1 can be presented as:

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

Figure 3 shows gives the operator tree of the above algebraic expression:



$\tilde{\pi}_{Name}$
$\tilde{\sigma}$ OldSalesPersons.$Age='very\ old'$
$\tilde{\bowtie}$ OldSalesPersons.$foid=SalesPersons.foid$
SalesPersons
OldSalesPerson

Fig. 3. Implementation plan using $(\tilde{\bowtie})$

3. Push $(\tilde{\sigma})$

By pushing $(\tilde{\sigma})$ operation down the expression tree, we actually reduce the size of relations we need to do before. The fuzzy object algebra for the fuzzy query written in example under push fuzzy selection strategy can be presented as:

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

Figure 4 is a description of their above fuzzy object algebra expression:

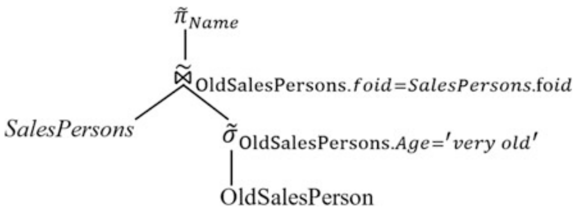

$\tilde{\pi}_{Name}$
$\tilde{\bowtie}$ OldSalesPersons.foid=SalesPersons.foid
SalesPersons
$\tilde{\sigma}$ OldSalesPersons.Age='very old'
OldSalesPerson

Fig. 4. Pushing $(\tilde{\sigma})$ down the tree

4 Performance Evaluation

To provide preliminary performance evaluation on implementation of query, processing has been proposed based on fuzzy object algebra [4]. We defined the three queries processed condition extract filter data for two cases of single conditions, most conditions and implement them on the same dataset.

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

SELECT * FROM OldSalesPersons, SalesPersons WITH 0.6

WHERE AND OldSalesPersons.Age = 'very old' WITH 0.7.

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

SELECT Name FROM OldSalesPerson as O, SalesPersons as S WITH 0.6 WHERE O.FIOD = S.FOID AND O.Age = 'very old' WITH 0.7.

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 old. Such queries are written as follows.

SELECT Name FROM SalesPersons as S inner join OldSalesPersons as O on O. FIOD = S.FOID WITH 0.6 WHERE O.Age = 'very old' WITH 0.7.

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 Fig. 5.

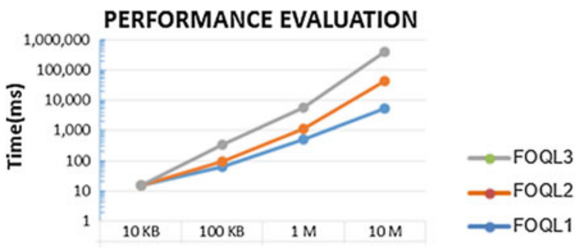


Fig. 5. Query performance

5 Conclusion

This paper presents a new model for optimizing the efficiency of query processing by semantic analyzing and FO algebra transforming. Specifically, we develop a heuristic fuzzy object algebraic optimization algorithm relied on equivalent transformation rules and fuzzy object algebra transformation. Analysis on several experiments using the proposed algorithm shows better performance of query processing, which proves the efficiency enhancement of our method.

References

1. Selee Na.: A Process of Fuzzy Query on New Fuzzy Object Oriented Data Model, In IEEE Tranon Knowledge and Data Engineering. 1(2010), 500–509.
2. Stefano Ceri, Georg Gottlob.: Translating SQL Into Relational Algebra: Optimization, Semantics, and Equivalence of SQL Queries, Software Engineering, IEEE Transactions, vol. SE-11, issue 4, 4(1985). 324–345.

3. 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, (2012). 53–62.
4. Truong Ngoc Chau, Nguyen Tan Thuan.: A Approach New In The Algebra Fuzzy Object, Proceedings of the @ Conference, Viet Nam. 11(2013), 204–209.