Abstract
In order to make more flexible database access the query language SQLf has been previously proposed. This language allows the expression of user preferences in queries giving discriminated answers. Such preferences are specified by means of fuzzy logic conditions. User may also specify the quality of the desired answer giving a minimum satisfaction level. One of the SQLf features is the use of Fuzzy Quantifiers in the WHERE clause as an operator for criteria combination. This use is known as Horizontal Quantification. An open issue for flexible querying is the problem of providing efficient evaluation mechanisms. As ever, these queries may be processed with a Naïve Strategy processing the fuzzy quantified sentence over the base relation obtained by the FROM clause. We propose here two improved strategies based in the application of the Derivation Principle. This principle consists in the distribution of a minimum user desired satisfaction degree over query involved conditions. One mechanism uses an external program in order to process the fuzzy query over the result of a SQL query. The other one performs the fuzzy query processing via a function calls decorated SQL query. Such functions perform the computation fuzzy sentences satisfaction degrees. We present in this paper a formal performance study of these three mechanisms. This study has been made using a SQLf prototype build on top of a RDBMS.

Keywords: Query Processing, Fuzzy Querying, Querying Performance, Fuzzy Quantifiers, SQLf.
1. INTRODUCTION

SQLf provides a large variety of fuzzy querying constructions, allowing the use of fuzzy conditions in any place where SQL allows a Boolean condition [3]. An interesting feature of SQLf is allowing the use of fuzzy quantifiers in querying. Liétard [5] and Tineo [8] have made contributions in the semantics of fuzzy quantified sentences in database querying.

Nevertheless, the problem of database fuzzy querying not is only a semantics issue but also a practical reality. The interest of some previous works has been to provide efficient evaluation mechanisms for fuzzy querying language SQLf [1], [3], [5]. At present time, evaluation mechanisms for fuzzy quantified queries have been proposed and studied only for vertical quantification where the quantification is made over the number of rows satisfying a fuzzy condition. In this context: Bosc et al [1] have presented a strategy based in Sugeno Integral properties for improve query evaluation, we name it Sugeno Strategy; Tineo [9] has presented the application of the Derivation Principle to such queries. This principle consists in distributing the minim satisfaction degree of desired solution deriving form the fuzzy query a crisp. The fuzzy query is then evaluated over the result of the regular derived query. This strategy has been proposed to be used several querying SQLf structures [4], [9]. López and Tineo [6] have made a formal statistics study of the performance some vertical fuzzy quantified queries evaluation mechanisms: Sugeno, Derivation and Naïve.

We are interested here in queries with horizontal quantification. These queries, given fuzzy criteria list, quantify the number of criteria that are satisfied for each row. We present here evaluation mechanisms for horizontal quantified query processing. For such mechanisms, we make a system performance analysis that is based in experimentation and use of statistics models [7]. The analysis of such experiments will lead us to distinguish the effects of factors that may inside in the performance of query evaluation mechanisms, pointing to the selection of the evaluation mechanisms with best performance.

Fuzzy quantifiers may be classified according to their nature in absolute and proportional. According to the behaviour of their membership functions, fuzzy quantifiers may be classified in increasing, decreasing and unimodal ones [11]. We restrict our work to increasing absolute fuzzy quantifiers. This restriction is made with loose of generality, because for the kind of queries studied here, the use of different kinds of quantifiers are equivalent [10].

The rest of this paper is organized as follows: In section 2 we present the syntax and semantic of the SQLf querying structure matter of this work, we give there an illustrative example that will be used in further sections of this paper. Section 3 is devoted to present the evaluations mechanisms that have been proposed for SQLf query evaluation and our new proposal for applying them in the evaluation of horizontal quantified queries. In order to study the performance of these evaluation mechanisms, in section 4 we set down our experiments. Results obtained form these experiments are shown and analyzed in section 5. Finally, in section 6 we present conclusions that we have arrived form this work, in that section, we also point to further works.

2. QUERY’S SEMANTICS

SQLf provides a kind of horizontal quantified queries which general form is:

\[
\text{select A from R where } Q(\text{fc}_1,\ldots,\text{fc}_n) \text{ with calibration } t.
\]

Being \( t \) a threshold of the minim satisfaction degree for user desired rows in the answer, \( A \) an attribute (attributes list) of the relation (relations list) \( R \). \( Q \) a fuzzy quantifier, and \( \text{fc}_1,\ldots,\text{fc}_n \) a list of fuzzy conditions. This query returns the fuzzy relation \( R' \) on \( \{a / (\exists y \in R / y.a=a) \land (\mu Q(X,fc)\geq t) \} \), being the membership degree of each element \( a: \mu R(a)=\mu Q(X,fc) \) (the truth degree of fuzzy quantified sentence \( Q \)’s are \( \mu fy \)) where \( X=[\text{fc}_1,\ldots,\text{fc}_n] \), \( \mu fy \) is a fuzzy predicate on \( X \) whose satisfaction degree is \( \mu fy(\text{fc})=\mu fc(y) \) (the satisfaction degree of the row \( y \) to the fuzzy condition \( \text{fc} \)). The sentence \( Q \)’s are \( \mu fy \) is interpreted with the Yager’s decomposition interpretation [11]. This interpretation coincides with Tineo’s [8] one in this case. The satisfaction degree of the quantified sentence is \( \mu (Q(X,fc))=\sup (\min (\mu Q(i),\mu fy)) \) where \( \mu fy(i) \) is the \( i \)-th higher value of the \( \mu fy(\text{fc}) \) degrees.

For example, let’s consider the employee relation of Table 1. We search for employees that meet at least three of the criteria: be around 34 years old, tall, heavy, of a high level study and with a regular salary. Fuzzy terms involved in this requirement may be defied by fuzzy sets in Fig. 1. We ask only for answers with satisfaction degree greater or equal to 0.5. This query is expressed in SQLf as:

\[
\text{select * from employee where at_least_3(}
\text{\quad age=aprox}_34\text{\_years, height=tall, weight=heavy, studies=high_level, salary=regular_salary}
\text{\quad )}
\text{with calibration 0.5.}
\]
In order to show the semantics of the example query, we present here some tables with the computation of the query result:

First, for each row in employee relation (Table 1), we show (Table 2) the satisfaction degree of fuzzy predicates defined in Fig. 1.

Applying the Yager’s decomposition interpretation [11] for quantified propositions, the computation of the satisfaction degree for the fuzzy quantified sentence is made for each row of the employee relation, sorting in decreasing order the satisfaction degree of the fuzzy criteria (Table 2) and mixing them with the satisfaction degrees of the fuzzy quantifier according to its definition given in Fig. 1. This computation of satisfaction degrees for fuzzy quantified sentences is shown in Table 3.

Finally, the query returns the fuzzy relation in Table 4. It contains those rows which satisfaction degree is greater or equal to the minimum desired satisfaction level, in this case 0.5. The satisfaction degree of each row is part of the result. Rows are decreasing ordered by this degree.
Table 2. Satisfaction Degree of Fuzzy Criteria

<table>
<thead>
<tr>
<th>y.id_num</th>
<th>µfy(age = aprox_34_years)</th>
<th>µfy(height = tall)</th>
<th>µfy(weight = heavy)</th>
<th>µfy(studies = high_level)</th>
<th>µfy(salary = regular_salary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emp01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0,5</td>
<td>1</td>
</tr>
<tr>
<td>Emp02</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0,75</td>
<td>1</td>
</tr>
<tr>
<td>Emp03</td>
<td>1</td>
<td>0,15</td>
<td>0,3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Emp04</td>
<td>1</td>
<td>0</td>
<td>0,25</td>
<td>0,5</td>
<td>1</td>
</tr>
<tr>
<td>Emp05</td>
<td>0,5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0,4</td>
</tr>
<tr>
<td>Emp06</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0,25</td>
<td>1</td>
</tr>
<tr>
<td>Emp07</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Emp08</td>
<td>1</td>
<td>0,6</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Emp09</td>
<td>1</td>
<td>0</td>
<td>0,75</td>
<td>0,5</td>
<td>1</td>
</tr>
<tr>
<td>Emp10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3. Satisfaction Degree of Horizontal Fuzzy Quantified Sentence

<table>
<thead>
<tr>
<th>y.id_num</th>
<th>min(µQ(1),µfy1)</th>
<th>min(µQ(2),µfy2)</th>
<th>Min(µQ(3),µfy3)</th>
<th>min(µQ(4),µfy4)</th>
<th>min(µQ(5),µfy5)</th>
<th>µ(Q(X,fy))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emp01</td>
<td>0</td>
<td>0,5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0,5</td>
</tr>
<tr>
<td>Emp02</td>
<td>0</td>
<td>0,5</td>
<td>0</td>
<td>0,75</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Emp03</td>
<td>0</td>
<td>0,3</td>
<td>0,15</td>
<td>0</td>
<td>0</td>
<td>0,3</td>
</tr>
<tr>
<td>Emp04</td>
<td>0</td>
<td>0,5</td>
<td>0,5</td>
<td>0,25</td>
<td>0</td>
<td>0,5</td>
</tr>
<tr>
<td>Emp05</td>
<td>0</td>
<td>0,4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0,4</td>
</tr>
<tr>
<td>Emp06</td>
<td>0</td>
<td>0,25</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0,25</td>
</tr>
<tr>
<td>Emp07</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Emp08</td>
<td>0</td>
<td>0,5</td>
<td>0,6</td>
<td>0</td>
<td>0</td>
<td>0,6</td>
</tr>
<tr>
<td>Emp09</td>
<td>0</td>
<td>0,5</td>
<td>0,75</td>
<td>0,5</td>
<td>0</td>
<td>0,75</td>
</tr>
<tr>
<td>Emp10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4. Fuzzy Relation resulting form the Fuzzy Query

<table>
<thead>
<tr>
<th>Id_num</th>
<th>first_name</th>
<th>last_name</th>
<th>Age</th>
<th>height</th>
<th>Weight</th>
<th>studies</th>
<th>salary</th>
<th>Profession</th>
<th>µRf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emp02</td>
<td>Cri</td>
<td>Sto</td>
<td>33</td>
<td>200</td>
<td>77</td>
<td>5</td>
<td>1500</td>
<td>Engineer</td>
<td>1</td>
</tr>
<tr>
<td>Emp09</td>
<td>Tez</td>
<td>Mum</td>
<td>35</td>
<td>175</td>
<td>110</td>
<td>4</td>
<td>1000</td>
<td>Nurse</td>
<td>0,75</td>
</tr>
<tr>
<td>Emp08</td>
<td>Rod</td>
<td>Bin</td>
<td>34</td>
<td>192</td>
<td>120</td>
<td>1</td>
<td>500</td>
<td>Watchman</td>
<td>0,6</td>
</tr>
<tr>
<td>Emp01</td>
<td>Ann</td>
<td>Tao</td>
<td>40</td>
<td>168</td>
<td>58</td>
<td>4</td>
<td>1250</td>
<td>Physician</td>
<td>0,5</td>
</tr>
<tr>
<td>Emp04</td>
<td>Ken</td>
<td>Cha</td>
<td>36</td>
<td>171</td>
<td>90</td>
<td>4</td>
<td>1250</td>
<td>Architect</td>
<td>0,5</td>
</tr>
</tbody>
</table>

3. EVALUATION MECHANISMS

Proposed SQLf processing mechanisms are intended for adding fuzzy querying capabilities on top of an existing RDBMS. Such mechanisms perform fuzzy query evaluation on the result of an underlying regular query addressed to the RDBMS. In this context a theoretical measure of mechanisms behaviour is the number of accessed database rows. One may think that whenever more rows are accessed more time is spent. Evaluation mechanisms have been proposed in order to keep low the number of accessed rows. It will be interesting to show the application of such mechanisms in case of horizontal fuzzy quantified queries. On the other hand, in some cases, regular selection criteria may be of high complexity and it may have a bad incidence in performance. Therefore it will be helpful to make performance study.

Naïve Strategy \[1, 3\] consists in a program scanning the whole database relation and computing the satisfaction degrees. For the example in previous section, Naïve strategy will access all rows in Table 1, computing all fuzzy criteria degrees in Table 2 and all the fuzzy quantified sentence satisfaction degrees in Table 3. Each row must be retained if pass the desired threshold. In the example, this mechanism accesses the double quantity of rows that actually are in the solution. For each one computes fuzzy sentence satisfaction degrees which is also time consuming. The pseudo code for this evaluation strategy applied to horizontal quantified queries’ processing is:
Procedure Naïve
Begin
Result ← ⊘
For y in (select * from R) do
  M ← SatisfactionDegree(y, Q(fc1, ..., fcn))
  If M ≥ t then
    Result ← Result ∪ {<y.A,M as μ>}
  select * from Result order by μ desc
End

The pseudo code of the function for computing fuzzy quantified sentence satisfaction degree for each row is:

Function SatisfactionDegree(y, Q(fc1, ..., fcn))
Begin
  Compute µfy(fc1), for i ∈ {1, ..., n}
  Sort µfy(fc1), ..., µfy(fcn) in decreasing order
  Let’s <µfy1, ..., µfyn> be the resulting sort
  Return sup min(µQ(i), µfyi), i ∈ {1, ..., n}
End

Sugeno Strategy [1] consists in scanning the whole table as in Naïve one, but with halting conditions for each group of elements under the quantification. The idea is avoid access to some elements. Nevertheless, as in horizontal quantification, elements under quantification are not rows but fuzzy criteria, these heuristics lose their sense since each row must actually be accessed. We consider by this reason that this strategy have no benefit for this kind of queries. Therefore, it will not be considered in the following of this work.

Derivation Principle [9] consists in processing rows retrieved by a regular query intended for selecting rows whose satisfaction degree is greater or equal to the given threshold t. This set of rows is known in fuzzy sets theory as the t-cut [2]. Given the fuzzy quantified query, we derive a regular query intended for retrieving the t-cut of the fuzzy query answer set. Some times, it is impossible to derive such query due to fuzzy condition nature, but a query for a superset of the t-cut. In this case we say that the derivation is weak otherwise is said to be strong.

For a SQLf horizontal quantified query of form:

select A from R where Q(fc1, ..., fcn) with calibration t.

we obtain the derived classical SQL query:

select * from R where OAC({DNC(fc1, ≥, t), ..., DNC(fcn, ≥, t)}, LB).

Being OAC( {DNC(fc1, ≥, t), ..., DNC(fcn, ≥, t)}, LB) the normal disjunctive from containing as terms all conjunction combinations of LB elements from {DNC(fc1, ≥, t), ..., DNC(fcn, ≥, t)}, LB is the lowest natural which satisfaction degree to the quantifier is greater or equal to t, DNC(fc, ≥, t) is the derived necessary condition for the t-cut of fc.

Definition of DNC(fc, ≥, t) has been matter of previous works [2], [4]. If all DNC(fc, ≥, t) are strong derivation, then the derived query presented here is also a strong derivation. We have made formal proof of this query derivation that will appear in [10]. Proof is hard and very long so it could not be included in this paper. Interested reader might contact the author.

For our example query presented in previous section, with the definition of fuzzy terms given in Fig. 1, we obtain the derived query:

select * from employee where
(31<age and age<=37 and 190<=height) or
(31<age and age<=37 and 100<=weight) or
(31<age and age<=37 and 4<=studies) or
(31<age and age<=37 and 875<=salary and salary<=1625) or
(190<=height and 100<=weight) or
(190<=height and 4<=studies) or
(190<=height and 875<=salary and salary<=1625) or
(100<=weight and 4<=studies) or
(100<=weight and 875<=salary and salary<=1625) or
(4<=studies and 875<=salary and salary<=1625).

With the Derivation Principle in this example, just rows from the employee relation (Table 1) that are in the final result (Table 4) must be accessed in this fuzzy query processing. This shows that Derivation Principle keeps low the number of rows accessed in query processing.

The Derivation Principle may be used into two query evaluation mechanisms: First is the Program Derivation Strategy consists in processing the fuzzy query over the result of derived query by means of an external program. The pseudo code for processing horizontal fuzzy quantified queries with this evaluation mechanism is the following:
Procedure Derived Program
Begin
  Result ← ∅
  For y in (select * from R where OAC({DNC(fc1,≥,t),...,DNC(fc_n,≥,t),LB}) do
    M ← SatisfactionDegree(y, Q(fc1,...,fc_n))
    Result ← Result ∪ {<y,A,M as mu>}
  select * from Result order by mu desc
End

Second is the Query Derivation Strategy consists in adding to the “select” clause a call to a function that computes the satisfaction degree of the fuzzy sentence. Also an “order by” clause is added for giving the result in decreasing order of the computed satisfaction degrees. We may appreciate this mechanism in the pseudo code:

```
select A, SatisfactionDegree(y, Q(fc1,...,fc_n)) as mu from R
where OAC({DNC(fc1,≥,t),...,DNC(fc_n,≥,t),LB}) order by mu desc
```

Despite the Derivation Strategy requires the computation of satisfaction degrees for a low number of rows than Naïve Strategy, we must to show experimentally if it really gives better performance. It may be natural to think that the use of a rather complex condition in the “were” clause of derived query might involve high cost giving advantage to the Naïve strategy. On the other hand it is necessary also to consider two proposed mechanisms based in the Derivation Principle because some DMBS do not allow function calls in querying structure.

4. Experiments’ Design
The performance evaluation will be made using formal model statistic method. The idea of this method is to explain the influence of several considered factors in the observed values from experiments. The importance of a factor is measured by the proportion of the total variation in the response that is explained by the factor. The queries where addressed to the database relation: employee(id_number, first_name, last_name, age, height, weight, studies, salary, profession, emp_dep_num). Database relations are populated using a program in PL/SQL that generates database extension. Values for id_number are sequential generated. Attributes first_name, last_name and profession are single function of id_number. Foreign key emp_dep_num values are random generated into a range. Rest of employee relation’s attributes are uniform random generated. Attribute age ranges between 18 and 65 years old. Attribute height ranks between 120 and 210 centimeters tall. Attribute weight ranks between 50 and 150 kilograms. Attribute studies ranks between 1st and 6th academic level. Attribute salary ranks between 100 and 3000$. The variables that are observed in the experiments are called answer variables. They usually are measures of the system behavior. In our case we are interested in the performance of the fuzzy query answer system. Therefore we want to observe the total spent time in solving the fuzzy query.

The experimental factors are those parameters whose values are changed in the experiments in order to determine their effect in the answer variable. First considered factor is the strategy used. We expect this factor to have a high influence in the performance of the query evaluation. The levels for this factor are: Naïve, Derivation (Program) and Query (Derived), respectively. The volume of data is a factor that must be considered. The access time is always depending of the volume of data. Despite in the different strategies, the proportion of accessed registers does not depend on how large is the database, it is reasonable to think that an interaction might exists between the volume factor and the strategy factor. We define six levels for the volume factor, varying the quantity of rows in the employee table of the database: Very (400 rows), Tiny (800 rows), Small (1600 rows), Medium (3500 rows), Large (7000 rows) and Extra (14 rows). The selectivity of a query may influence its performance. A query is more selective in the measure that retrieves a smaller set of result. We may vary the selectivity of a SQL query changing its desired satisfaction degree level t. We have chosen three levels for selectivity factor: Low (t = 0.25), Middle (t = 0.5) and High (t = 0.75). A condition that could vary in a database experiment is the physical storage structure for the database tables. We prefer in this study to avoid this kind of consideration. We adopt to have simple table structures provided of an index for each attribute in the relation. We must remember that evaluation mechanisms proposed and analyzed in this work are designed as a logic layer on top of an existing RDBMS. We will fix conditions and fuzzy quantifier in the query. The design of fuzzy terms involved is made with care to be really representative. As data is generated with uniform distributions, definitions of such fuzzy terms are made in a way of be in some sense “uniform”. Fuzzy terms definitions in Fig. 1 have this desired characteristic. For the experimentation, we use the same query presented as example here before, varying the value of t in the “with calibration” clause:

```
select * from employee where
  at_least_3( age=aprox_34_years, height=tall, weight=heavy,
               studies=high_level, salary=regular_salary
          )
with calibration t.
```
The repetitions of experiments are called replicas. As we use a dedicated server, replicas are unnecessary. Nevertheless, we perform three replicas of each experiment in order to prove this hypothesis. Replicas will no be considered in the model.

We have chosen a full factorial design for our experimental study. That is, we will consider all the mentioned factors and all their levels. We have the hypothesis that all factors and their interactions have significant influence in the performance. This kind of design allows study the influence of each factor and all theirs interactions. We must take value for the answer variable for each possible combination of the factors in all their levels.

The model is an expression of the observed values for the response variable as a combination of experimental factors levels influences. The model for our study is:

\[ y_{ijk} = \mu + \tau_i + \pi_j + \rho_k + \tau_i \pi_j + \tau_i \rho_k + \pi_j \rho_k + \tau_i \pi_j \rho_k + \epsilon_{ijkl} \]

Being: \( y_{ijk} \) the observed value for the levels \( i, j, k \) of the factors \( E, V, T \), respectively; \( \mu \) the arithmetic mean of the observed values of all experiments; \( \tau_i \) the effect of the factor \( F \) at the level \( m \), \( F \in \{E, V, T\}; \)

\( \tau_{m'} \) the effect of the Interaction between factors \( F' \) and \( F'' \) at the levels \( m' \) and \( m'' \), respectively, with different \( F', F'' \in \{E, V, T\} \); and \( \epsilon_{ijkl} \) the effect of the Interaction between all the factors \( E, V \) and \( T \) for the levels \( i, j \) and \( k \), respectively.

5. **EXPERIMENTAL RESULTS**

We have run the experiments in a SQLf prototype on top of Oracle 9i RDBMS. Experimentation platform was a desktop computer with 895MHz Pentium III processor, 512MB RAM and 20GB hard disk, running Red Hat Linux 8.0, Query transformation mechanisms where coded in SWI Prolog.

Experiments have given the results summarized in **Table 5**. We have loaded these results in the R statistical software with the experimental model. **Table 6** shows the analysis of variance for this model with the observer data.

| Table 5. Observed Times. We show here the mean of the three replicas |
|---|---|---|---|---|---|---|---|---|
| | **Naïve** | **Derivation** | **Query** |
| | Low | middle | high | low | middle | High | Low | middle | high |
| Very | 2.24 | 2.02 | 1.87 | 2.12 | 2.03 | 1.70 | 1.24 | 1.05 | 0.90 |
| Tiny | 2.61 | 2.56 | 1.98 | 2.52 | 2.14 | 1.74 | 1.31 | 1.14 | 0.95 |
| Small | 3.47 | 3.00 | 2.35 | 3.27 | 3.16 | 2.18 | 1.76 | 1.55 | 1.07 |
| Medium | 5.30 | 4.39 | 2.95 | 5.12 | 4.59 | 2.02 | 2.69 | 2.30 | 1.13 |
| Large | 8.89 | 8.39 | 3.97 | 8.80 | 7.66 | 2.69 | 4.67 | 4.63 | 1.47 |
| Extra | 15.57 | 12.78 | 6.24 | 15.21 | 12.30 | 3.64 | 8.54 | 6.65 | 1.93 |

| Table 6. ANOVA of Full Factorial Model. All factors and thir interaction are very significant |
|---|---|---|---|---|
| | **Df** | **Sum Sq** | **Mean Sq** | F value | **Pr(>F)** |
| Volume | 5 | 1151.09 | 230.22 | 2312.2050 | < 2.2e-16 | *** |
| Selectivity | 2 | 270.70 | 135.35 | 1359.3842 | < 2.2e-16 | *** |
| Strategy | 2 | 198.86 | 99.43 | 998.6075 | < 2.2e-16 | *** |
| Volume:Selectivity | 10 | 287.12 | 28.71 | 288.3684 | < 2.2e-16 | *** |
| Volume:Strategy | 10 | 84.74 | 8.47 | 85.1083 | < 2.2e-16 | *** |
| Selectivity:Strategy | 4 | 14.85 | 3.71 | 37.2855 | < 2.2e-16 | *** |
| Volume:Selectivity:Strategy | 20 | 15.60 | 0.78 | 7.8335 | 2.263e-13 | *** |
| Residuals | 108 | 10.75 | 0.10 | --- |

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
The summary of the analysis of variance (ANOVA) obtained for the full factorial model with experimental data is presented in Table 6. It shows that the stochastic model selected fits very well to the data. This fact tells us that we may take conclusions about the performance using this experiment, because it is stochastically valid. We can see also in the ANOVA table that all interactions between factors are very significant to explain the results. Therefore, we will plot and analyze all interactions of two factors. We have confirmed the hypothesis presented in the experiment design about the significance of selected factors and their interactions.

Let’s analyze the interaction between Volume and Strategy factors, shown in Fig. 2: As we may expect, processing times increases with increasing of database volume. Times for the Query Derivation strategy are considerably lower than others. On the other hand, times for the Naïve and the Derivation Program strategies are very similar being the last lightly lower than the first.

Lowest times for Query Derivation strategy obeys to the fact that this strategy gives the whole control to the DBMS, allowing the computation of satisfaction degrees for each row while they are selected. In this case, due to the application of the Derivation Principle, satisfaction degrees are calculated only for those rows that meet the threshold specified in the calibration. In the other cases the set of rows must be retrieved from the database and after a program must scan this set of rows in order to compute satisfaction degrees. This kind of “re-scan” explains higher costs.

Difference between Derived Program and Naïve strategies is due to the fact that Derived Program works with a subset of rows selected by the derived query while the Naïve strategy works with all rows from the base relation. Naïve strategy involves extra computing due to calculation of satisfaction degrees of rows that will be rejected due to the calibration. On the other hand, in this case, due to the application of the Derivation Principle, Derived Program strategy only computes satisfaction degrees just for those rows that meet the threshold specified in the calibration.

Fig. 2. Volume-Strategy Interaction

Let’s analyze the interaction between Selectivity and Strategy factors (Fig. 3): Query processing time decreases while selectivity increases because a more selective query retrieves a smaller set of answers.

In case of low selectivity we may observe that times for Derived Program strategy and Naïve strategy are very close. It is due to the fact that a low selectivity implies that the set of selected answers is closer to the set of all rows. In this case we have a low number of rows for which the Naïve strategy computes the satisfaction degrees but are then rejected due to the calibration.

In case of middle selectivity, times observed for Derived Program and Naïve strategies are also close. Difference increases just a little. The number of rejected rows for which Naïve strategy makes non useful computation is a bit greater than previous case.

In case of high selectivity, we have a really marked difference between times observed for Derived Program and Naïve Strategies. Here there is high number of rows for which Naïve strategy computes the fuzzy quantified sentence satisfaction degree for being after rejected due to degrees lower than the minim desired degree specified in the calibration of the query. Here times for Derivation Program strategy are going down fast. These times become closer to the Derived Query strategy. It is because the number of desired answers has been reduced with high selectivity, therefore there is a reduced quantity of rows that Derivation Program strategy must re-scan for satisfaction degrees computation.
Let’s analyze the interaction between Volume and Selectivity factors, shown in Fig. 4: As ever, times grow with volume and are lower as selectivity is higher. Curve for high selectivity is softer than curves for low and middle selectivity. It obeys to the fact that high selectivity keeps low the number of retrieved rows. Times for low selectivity and middle selectivity are very close. To explain that we must remember that selectivity in the experiment has been set by means of the minum desired satisfaction degree specified in the calibration of the query. Values selected for this parameter where 0.25 (low selectivity) 0.5 (middle selectivity) and 0.75 (high selectivity). Although 0.5 is in the middle between 0.25 and 0.75, the quantity of retrieved rows for the threshold 0.5 is not just in the middle between the cardinalities of retrieved rows sets for 0.25 and 0.75 thresholds.

According to these results, we conclude that the Query Derivation strategy is the most suitable mechanism for this kind of queries. Interaction between strategy and volume factors (Fig. 2) and interaction between strategy and selectivity factors (Fig. 3) clearly show it. Nevertheless, Query Derivation strategy could not be supported in some DBMS because they not allow user defined function calls in queries. In those cases, we Derivation Program strategy would be the most accurate evaluation mechanism because it have better performance than Naïve strategy.
6. CONCLUDING REMARKS

We have dealt here with SQLf horizontal fuzzy quantified queries. We have shown their intuitive and formal semantics. For this kind of queries we have discussed about the applicability of the known fuzzy query evaluation strategies: Naïve, Sugeno and Derivation Principle. We concluded that Sugeno strategy is not suitable for them. We have introduced the derived query obtained from a horizontal fuzzy quantified query. We have shown with an illustrative example the reduction of row access gained with the Derivation Principle. We have characterized two strategies for query evaluation based in the Derivation Principle: The Derivation Program Strategy and the Derived Query Strategy. We have made a formal statistic study of performance of query evaluation mechanisms presented here. This study involved also the volume and selectivity factors. We have shown with this study that the Derived Query strategy has the best performance between considered mechanisms. It improves considerably the response time respects the Naïve strategy. We have restricted our work to increasing absolute fuzzy quantifiers. Nevertheless, for the kind of queries studied here, the use of different kinds of quantifiers is equivalent. A variant of horizontal fuzzy quantified queries considered here is allowing the fuzzy criteria under quantification to have also preference levels. The study if this kind of queries will be matter of further works. We would also study in further works the application of evaluation strategies presented here in order to build processing mechanisms for other uses of fuzzy quantification in querying such as nesting and division operators. The final objective of our research work would be to implement these query capabilities into a real DBMS.

Acknowledgments

I would like to express a great acknowledgement to the person who has inspirited my work and my life, a very special person who is with me at every moment, giving me love and force for living and guide for doing all things that I do. I have never done this work without the invaluable help that this person has given me. This person is the Lord and Savior of my soul: Jesus Christ, my God.

I also give thanks to my friend Dr. Olivier Pivert who has propose me the initial idea of doing this research and has help with comments about my work for improving it. I also express grateful to my friend Dr. Mariela Curiel who has introduced me to the art of computer systems performance analysis. I beg God to bless theirs lives.

References


