Flexible Querying Using Criterion Trees: A Bipolar Approach

Guy De Tré[1], Jozo Dujmović[2], Joachim Nielandt[1], and Antoon Bronselaer[1]

[1] Dept. of Telecommunications and Information Processing, Ghent University,
Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
{Guy.DeTre,Joachim.Nielandt,Antoon.Bronselaer}@UGent.be
[2] Dept. of Computer Science, San Francisco State University,
1600 Holloway Ave, San Francisco, CA 94132, U.S.A.
jozo@sfsu.edu

Abstract. Full exploration of databases requires advanced querying facilities. This is especially the case if user preferences related to expected results are complex. Traditional query languages like SQL and OQL only have limited facilities for expressing query criteria that are composed of simple criteria. So, while searching for information, users often have to translate their complex requirements (which are typical for human reasoning) into simpler queries, which in many cases can only partly reflect what the user is actually looking for. As a potential solution, we recently proposed a query language extension that is based on soft computing techniques and supports the use of so-called criterion trees. In this paper, we further extend criterion trees so that they can contain both mandatory and optional query conditions. More specifically, we study optional query conditions from a bipolar point of view and propose and illustrate a framework for handling them in query processing.

Keywords: Fuzzy querying, criterion trees, GCD, partial absorption.

1 Introduction

When users want to search for information in a database system, their needs and preferences have to be specified in so-called WHERE-clauses of queries or similar constructs. Traditional query languages like SQL [9] and OQL [2] only support WHERE-clauses in which the query criteria are specified by a Boolean expression that consists of simple expressions connected by logical connectives ($\wedge$, $\vee$ and $\neg$). Parentheses can be used to alter the sequence of evaluation.

Such Boolean expressions do not offer the facilities and flexibility that are required to fulfill complex information needs as often encountered in real-life situations. Indeed, humans often tend to express criteria in a soft way. They structure and group criteria, assign a different importance to different criteria or subgroups of criteria, make a distinction between mandatory and optional criteria, etc. For example, if somebody is searching for a house in a real estate database, it is quite natural to require affordability (acceptable price and maintenance costs) and suitability (e.g., good comfort and a good location). Most

M. Jamshidi et al. (eds.), *Advance Trends in Soft Computing WCSC 2013*,
Studies in Fuzziness and Soft Computing 312,
DOI: 10.1007/978-3-319-03674-8_5,

homebuyers require simultaneous satisfaction of affordability and suitability criteria and would NOT accept homes where either affordability or suitability is not satisfied, requiring a hard (partial) conjunction operator. If this is not the case, then the aggregator must be a soft partial conjunction. In addition, for some homebuyers affordability is more important than suitability; an opposite criterion is also possible. If the query language criterion cannot express these fundamental requirements, it is not going to be acceptable for most homebuyers. Furthermore, good comfort might be further specified by living comfort and basic facilities. Living comfort refers to at least two bathrooms, three bedrooms, garage, etc., whereas basic facilities refer to gas, electricity, sewage, etc. Good location might be subdivided by accessibility, healthy environment, nearby facilities, etc. Some of these criteria (e.g., garage) might be mandatory, while others (e.g., dentist at close distance) might be optional. Traditional query languages have no specific facilities to deal with such complex search conditions.

Soft computing techniques help to overcome these shortcomings. Soft criteria specifications and criteria preferences can be dealt with by using 'fuzzy' querying techniques of which an overview is, among others, given in [12]. In order to efficiently cope with queries where special aggregators are required or users need to generalize or specialize their criteria for obtaining better insight in what they are looking for, we recently proposed a hierarchically structured criteria specification that is called a *criterion tree* [4]. As originally proposed, criterion trees cannot cope with optional query criteria. Nevertheless, such a facility is required if one wants to adequately support human consistent searching and decision making [1,6]. It is currently subject to a more general research topic that is commonly known as bipolarity in flexible querying (see, e.g., [5]).

In this paper, we propose to extend criterion trees with facilities for handling optional query criteria. The paper is further structured as follows. In Section 2, some preliminaries of criterion trees and bipolar querying are given. Next, we study the aggregation of mandatory and optional query criteria in Section 3. Two aggregation operators *'and optionally'* and *'or optionally'* are proposed. The main advantage of these operators is that they assign a bonus, resp. penalty (or malus), to the query satisfaction depending on the satisfaction, resp. dissatisfaction, of the optional criterion. Next, in Section 4, the extension and evaluation of criterion trees is presented, considering the novel operators. An illustrative example is given in Section 5. Section 6 concludes the paper.

2 Preliminaries

2.1 Criterion Trees

A *criterion tree* is a tree structure of which each node can be seen as a container for information. Each leaf node contains an elementary query criterion c_A, which is defined on a single database attribute A and expresses the user's preferences with respect to the acceptable values for that attribute while computing the answer set of the query. In general, a fuzzy set with membership function μ_A over the domain dom_A of A can be used to define the criterion. The membership

grade $\mu_A(v) \in [0,1]$ of a domain value $v \in dom_A$ then expresses the extent to which v is preferred by the user.

All non-leaf nodes of a criterion tree contain a symbol representing an aggregation operator. Moreover, each child node n_i of a non-leaf node n has an associated weight w_i, reflecting its relative importance within the subset of all child nodes of the non-leaf node. Hereby, for a non-leaf node with k child nodes it must hold that $\sum_{i=1}^{k} w_i = 1$. With this choice, we follow the semantics of weights as used in the LSP methodology [7]. The supported basic aggregators are conjunction (C), hard partial conjunction (HPC), soft partial conjunction (SPC), neutrality (A), soft partial disjunction (SPD), hard partial disjunction (HPD) and disjunction (D). This set is in fact a selection of seven special cases from the infinite range of generalized conjunction/disjunction (GCD) functions and can be easily extended when required [7].

Once specified, criterion trees can be used in the specification of the WHERE-clause of a query. This is illustrated in Section 4. Their evaluation for a relevant database record r results in a criterion satisfaction specification, which can then be used in the further evaluation and processing of the query. Criterion trees are evaluated in a bottom-up way. This means that, when considering a relevant database tuple r, the elementary criteria c_i of the leaf nodes are first evaluated. When specified by a membership function μ_A, the evaluation $\gamma_{c_i}(r)$ of c_i boils down to determining the membership value of the actual value $r[A]$ of A for r, i.e., $\gamma_{c_i}(r) = \mu_A(r[A])$. Next, non-leaf nodes (if any) are evaluated in a bottom-up fashion. A non-leaf node n can be evaluated as soon as all its child nodes n_i, $i = 1, \ldots, k$ have been evaluated. For evaluation purposes, the following implementation of GCD is used [8]:

$$M(x_1, \ldots, x_n; q) = \begin{cases} (\sum_{i=1}^{n} w_i x_i^q)^{1/q} & \text{, if } 0 < |q| < +\infty \\ \prod_{i=1}^{n} x_i^{w_i} & \text{, if } q = 0 \\ \min(x_1, \ldots, x_n) & \text{, if } q = -\infty \\ \max(x_1, \ldots, x_n) & \text{, if } q = +\infty \end{cases} \tag{1}$$

where $x_i \in [0,1]$, $1 \leq i \leq n$ are the input values which represent query satisfaction degrees (hereby, 0 and 1 respectively denote 'not satisfied at all' and 'fully satisfied'). The normalized weights $0 < w_i \leq 1$, $1 \leq i \leq n$, $\sum_{i=1}^{n} w_i = 1$ specify the desired relative importance of the inputs. Furthermore, the computed exponent $q \in [-\infty, +\infty]$ determines the logic properties of the aggregator. Special cases of exponent values are: $+\infty$ corresponding to full disjunction D, $-\infty$ corresponding to full conjunction C and 1 corresponding to weighted average A. The other exponent values q allow to model other aggregators, ranging continuously from full conjunction to full disjunction, and can be computed from a desired value of orness (ω). For aggregation in criterion trees, we used the following numeric approximation for q [7]:

$$q = \frac{0.25 + 1.89425x + 1.7044x^2 + 1.47532x^3 - 1.42532x^4}{\omega(1-\omega)} \tag{2}$$

where

$$x = \omega - 1/2.$$

Suitable orness-values are the following: $\omega = 1/6$ for HPC, $\omega = 5/12$ for SPC, $\omega = 7/12$ for SPD and $\omega = 5/6$ for HPD, assuming that partial disjunction is modeled as the De Morgan dual of partial conjunction.

Considering a database record r, the query satisfaction degree $\gamma_n(r)$ corresponding to n is computed using Eq. (1) and the following arguments: $\gamma_{n_i}(r)$, $i = 1, \ldots, k$, w_i being the weight that has been associated with n_i, $i = 1, \ldots, k$, and q being the value that models the aggregator that is associated with n. The overall satisfaction degree for a record r using a criterion tree is obtained when the root node n_{root} of the tree is evaluated, i.e., by computing $\gamma_{n_{root}}(r)$.

2.2 Bipolar Querying

An important issue in bipolar querying concerns the handling of constraints and wishes (see, e.g., [10,5,1]). Bipolarity hereby refers to the fact that users might distinguish between mandatory and desired criteria while specifying their query preferences. For handling desired criteria, two aggregators 'and if possible' and 'or else' have been proposed in [1] and defined as follows (with $k \in [0, 1]$):

$$\gamma_{c_1 \text{ and if possible } c_2}(r) = \min(\gamma_{c_1}(r), k\gamma_{c_1}(r) + (1 - k)\gamma_{c_2}(r)) \tag{3}$$

and

$$\gamma_{c_1 \text{ or else } c_2}(r) = \max(\gamma_{c_1}(r), k\gamma_{c_1}(r) + (1 - k)\gamma_{c_2}(r)). \tag{4}$$

In what follows, we propose a generalization of these operators which uses slightly different semantics and is based on the conjunctive and disjunctive partial absorption operators as originally proposed and studied in [6].

3 Aggregation of Optional Criteria

The behavior and aggregation of optional criteria has been studied and analyzed in [6] and resulted in the following 'and optionally' (conjunctive partial absorption) and 'or optionally' (disjunctive partial absorption) operators [3].

$$\gamma_{(c_1 \text{ and optionally } c_2)}(r) = w_2\gamma_{c_1}(r) \mathit{\Delta} (1 - w_2)[w_1\gamma_{c_1}(r) \nabla (1 - w_1)\gamma_{c_2}(r)] \tag{5}$$

where $\mathit{\Delta} \in \{C, HPC\}$ and $\nabla \in \{D, SPD, HPD, A\}$, and

$$\gamma_{(c_1 \text{ or optionally } c_2)}(r) = w_2\gamma_{c_1}(r) \nabla (1 - w_2)[w_1\gamma_{c_1}(r) \mathit{\Delta} (1 - w_1)\gamma_{c_2}(r)] \tag{6}$$

where $\nabla \in \{D, HPD\}$ and $\mathit{\Delta} \in \{C, SPC, HPC, A\}$. Both operators are asymmetric.

Using weighted power means as in Eq. (1), both operators can be implemented by

$$M(x_1, x_2; q_1, q_2) = [(1 - w_2)[w_1 x_1^{q_2} + (1 - w_1)x_2^{q_2}]^{q_1/q_2} + w_2 x_1^{q_1}]^{1/q_1} \tag{7}$$

where $x_1 \in [0,1]$ is the mandatory input, $x_2 \in [0,1]$ is the desired input and the exponents q_1 and q_2 are those reflecting the selected aggregators for Δ and ∇ (being computed as described in Section 2).

The weights w_1 and w_2 are computed so as to reflect as adequately as possible the impact of the mean penalty P and mean reward R percentages provided by the user. Hereby the underlying semantics of P and R are defined by the following border conditions [6] (and their dual counterparts for 'or optionally'):

$$\forall\, 0 < x \leq 1 : (x \text{ and optionally } 0) = x(1-p), 0 \leq p < 1 \tag{8}$$

(hence if the optional condition is not satisfied at all, then criterion satisfaction is decreased with a penalty of p)

$$\forall\, 0 < x < 1 : (x \text{ and optionally } 1) = x(1+r), 0 \leq r < 1/x - 1 \tag{9}$$

(hence if the optional condition is fully satisfied, then criterion satisfaction is increased with a reward of r). Note that p and r can be zero. The values P and R are (approximately) the mean values of p and r and usually expressed as percentages. Decision-makers select desired values of P and R and use them to compute the corresponding weights w_1 and w_2. More details on this computation can be found in [6].

By taking $\Delta = C$ and $\nabla = A$, Eq. (3) is obtained as a special case of Eq. (5). Likewise, with $\nabla = D$ and $\Delta = A$, Eq. (6) yields Eq. (4). The main advantage of the 'and optionally' and 'or optionally' operators is that they enable the use of both a reward and a penalty, whereas Eq. (3) and (4) by definition only assign a reward in case of (partial) satisfaction of the optional condition. Such a penalty facility is however required if we want to adequately reflect human reasoning. Consider for example two house descriptions in a database where a mandatory condition 'proximity of bus stop' is perfectly satisfied for both. If an optional condition 'proximity of dentist' is only satisfied for the first house and there is no penalty facility available, then it would not be possible to distinguish between the overall satisfaction of both houses. However, humans would naturally assign a penalty to the second house and prefer the first one.

4 Extended Criterion Trees

Criterion trees can be extended with the 'and optionally' ($ANDOP$) and 'or optionally' ($OROP$) operators. This can be described using Extended BNF (EBNF) [11] by:

```
nabla_conjunction = "D" | "SPD" | "HPD" | "A"
delta_conjunction = "C" | "HPC"
nabla_disjunction = "D" | "HPD"
delta_disjunction = "C" | "SPC" | "HPC" | "A"
aggregator = "C" | "HPC" | "SPC" | "A" | "SPD" | "HPD" | "D"
criterion tree = elementary criterion | composed criterion
composed criterion = aggregator "(" criterion tree":"weight","
```

```
   criterion tree":"weight {"," criterion tree":"weight}")" |
    "ANDOP("nabla_conjunction"," delta_conjunction"," P"," R")
      (Mandatory:" criterion tree", Optional:" criterion tree")" |
    "OROP("nabla_disjunction"," delta_disjunction"," P"," R")
      (Sufficient:" criterion tree", Optional:" criterion tree")"
elementary criterion = attribute "IS {("min value"," suitability")"
   {",(" value"," suitability")" } ",("max value"," suitability")}"
```

where { } means 'repeat 0 or more times'.

The values in the specification of an elementary criterion must form a strictly increasing sequence. Together they specify the piecewise linear membership function μ_A of a fuzzy set that reflects the user's preferred values (suitability) for the attribute A under consideration.

For the asymmetric $ANDOP$ and $OROP$ operators, the first criterion tree reflects the mandatory/sufficient criterion whereas the second criterion tree defines the optional criterion.

Once specified, extended criterion trees can be used in the specification of the WHERE-clause of a query. Extended criterion trees are evaluated bottom-up, by first evaluating the elementary criteria. For a database record r this is done by determining the membership value $\mu_A(r[A])$ of the actual value $r[A]$ of A in r. The composed criteria are evaluated as soon as all their components have been evaluated. Eq. (7) is used for the evaluation of the $ANDOP$ and $OROP$ operators, whereas Eq. (1) is used for the evaluation of the other aggregators.

5 An Illustrative Example

Specifying search criteria for a house in a real estate database is often a complex task. It boils down to specifying weighted criteria and subcriteria, followed by carefully considering penalties and rewards that might result from the satisfaction and dissatisfaction of optional criteria. Assume that the user is looking for an affordable house with good comfort, condition and location. Such a search can be specified using the following SQL statement for regular relational databases.

```
SELECT id, address, price, TREE(c_suitability) AS satisfaction
FROM real_estates r, location l
WHERE (r.location_id=l.id) AND satisfaction>0.5
ORDER By satisfaction
```

The query uses a predefined function $TREE$ which takes an extended criterion tree as argument and computes the overall satisfaction degree ($satisfaction$) of the database records being processed by the query. The criterion tree $c\_suitability$ is specified by

```
c_suitability=HPC(c_comfort:0.4, c_condition:0.4, c_location:0.2)
```

with subtrees $c\_comfort$, $c\_condition$ and $c\_location$. It specifies that a house is considered to be suitable if it is comfortable, the overall condition of the house

is good and its location is adequate. For the aggregation, hard partial conjunction (HPC) is used and comfort and condition are considered to be more important than comfort and location. Typically, users will then specify in more detail what they expect with respect to comfort, condition and location. This is done by specifying each of the three subtrees in more details. For example, *c_location* =

```
ANDOP(A,C,15,10)(Mandatory:
      A(SPC(c_railway_station:0.3, c_road:0.5, c_highway:0.2):0.5,
        HPC(c_sport:0.4, c_doctor:0.2, c_restaurant/bar:0.4):0.5),
            Optional:green_area)
```

This specification reflects that according to the user, a good location is determined by two mandatory criteria and one optional criterion. The mandatory criteria respectively reflect good accessibility and proximity of facilities which are of equal importance to the user. Good accessibility is in this case expressed by proximity of a railway station, proximity of a regional road and proximity of a highway. These three subcriteria are aggregated with a soft partial conjunction operator (SPC) and proximity of a regional road is considered to be more important than the other two criteria. Proximity of facilities is further specified as proximity of sport facilities, proximity of medical practitioners and proximity of bars and restaurants. Hard partial conjunction (HPC) is used for the aggregation. Proximity of green area is considered to be optional. Hence, the 'and optionally' operator ($ANDOP$) is used to combine the mandatory and optional criteria. A penalty of 15% is considered for houses with a complete lack of green area in their environment. Houses that fully satisfy the green area criterion will earn a reward of 10%.

The remaining criteria in the specification of *c_location* are all examples of elementary criteria. Elementary criteria are specified by a membership function. For example, *c_doctor* can be specified by

```
r.distance_to_doctor IS {(5,1), (20,0)}
```

which denotes that travel distances of more than 20 minutes to reach a doctor are unacceptable.

Evaluation of *c_suitability* for a given record r is done with the function $TREE$. This function first evaluates the elementary criteria and then evaluates the internal nodes of the criterion tree in a bottom-up approach. Hereby, an internal node can be evaluated as soon as all of its child nodes have been evaluated. The satisfaction degree resulting from the evaluation of the root node (*c_suitability*) is returned by the function $TREE$. In the query, only records with a satisfaction degree larger than 0.5 will be returned. Of course, in practice, criterion trees can be much more complex than the one given in this example.

6 Conclusions

Criterion trees offer flexible facilities for specifying complex query conditions. In this paper we extended criterion trees with 'and optionally' and 'or optionally'

operators which allow to properly deal with optional query criteria. These operators consistently reflect human reasoning and enable the use of both a reward and a penalty for cases where the optional criteria are satisfied, resp. dissatisfied.

The proposed work is currently being implemented within the framework of the open source PostgreSQL object-relational database system. In further research, we plan to focus on performance and optimization issues.

References

1. Bosc, P., Pivert, O.: On Four Noncommutative Fuzzy Connectives and their Axiomatization. Fuzzy Sets and Systems 202, 42–60 (2012)
2. Cattell, R.G.G., Barry, D.K.: The Object Data Standard: ODMG 3.0. Morgan Kaufmann, San Francisco (2000)
3. De Tré, G., Dujmović, J.J., Bronselaer, A., Matthé, T.: On the Applicability of Multi-criteria Decision Making Techniques in Fuzzy Querying. Communications in Computer and Information Sciences 297, 130–139 (2012)
4. De Tré, G., Dujmović, J., Nielandt, J., Bronselaer, A.: Enhancing flexible querying using criterion trees. In: Larsen, H.L., Martin-Bautista, M.J., Vila, M.A., Andreasen, T., Christiansen, H. (eds.) FQAS 2013. LNCS (LNAI), vol. 8132, pp. 364–375. Springer, Heidelberg (2013)
5. Dubois, D., Prade, H.: Handling bipolar queries in fuzzy information processing. In: Galindo, J. (ed.) Handbook of Research on Fuzzy Information Processing in Databases, pp. 97–114. IGI Global, Hershey (2008)
6. Dujmović, J.J.: Partial Absorption Function. Journal of the University of Belgrade, EE Dept. Series Mathematics and Physics 659, 156–163 (1979)
7. Dujmović, J.J.: Preference Logic for System Evaluation. IEEE Transactions on Fuzzy Systems, vol 15(6), 1082–1099 (2007)
8. Dujmović, J.J.: Characteristic Forms of Generalized Conjunction/Disjunction. In: IEEE World Congress on Computational Intelligence, Hong Kong (2008)
9. ISO/IEC 9075-1:2011: Information technology – Database languages – SQL – Part 1: Framework, SQL/Framework (2011)
10. Lacroix, M., Lavency, P.: Preferences: Putting more knowledge into queries. In: VLDB 1987 Conference, Brighton, UK, pp. 217–225 (1987)
11. Wirth, N.: What Can We Do About the Unnecessary Diversity of Notation for Syntactic Definitions. Communications of the ACM 20(11), 822–823 (1977)
12. Zadrozny, S., De Tré, G., De Caluwe, R., Kacprzyk, J.: An Overview of Fuzzy Approaches to Flexible Database Querying. In: Galindo, J. (ed.) Handbook of Research on Fuzzy Information Processing in Databases, pp. 34–54. IGI Global, USA (2008)