Helping Wine Lovers With Taxonomies

Paolo Ciaccia[1], Davide Martinenghi[2] and Riccardo Torlone[3]

[1]*Dipartimento di Informatica - Scienza e Ingegneria, Università di Bologna, Italy*
[2]*Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy*
[3]*Dipartimento di Ingegneria, Università Roma Tre, Italy*

Abstract
We formally investigate the problem of retrieving the best results complying with multiple preferences expressed in a logic-based language when data are stored in relational tables with taxonomic domains. We introduce two operators that rewrite preferences for enforcing transitivity, which guarantees soundness of the result, and specificity, which solves conflicts among preferences. We show that these two properties cannot be achieved together and identify the only two possibilities to ensure transitivity and minimize conflicts. Our approach proves effective when tested over both synthetic and real-world datasets.

1. Introduction

Preferences are typically expressed in generic terms (e.g., I prefer pasta to beef), whereas available data is more specific (the menu might contain lasagne and hamburger). This mismatch causes difficulties when trying to automatically suggest the best solutions. The problem becomes even more involved when several preferences at different levels of granularity and possibly conflicting with each other are specified.

Example 1. We would like to select some bottles of wine from the list in Figure 1. While we prefer white wines to red ones, we prefer Amarone (a red wine) to white wine; we prefer Tuscan wineries in the province of Siena to those in the Piedmont province of Asti. Moreover, if the winery lies in the Langhe (which spans both the Asti and Cuneo provinces) we prefer an aged wine (i.e., produced before 2017) to a more recent one. ■

In order to support the mentioned preferences, we need further information, as, e.g., provided by the taxonomies in Figure 1. The example also shows that conflicts can occur when preferences are defined at different levels of detail (e.g., the preference for Amarone, which is a red wine, is in contrast with the more generic preference for white wines). Also observe that further preferences can be naturally (transitively) derived from those that are stated explicitly (e.g., from the preference for wines from Siena to those from Asti and the preference for aged wines when they are from the Langhe region, we can also derive a preference for wines from Siena to young wines from Langhe).

In this paper, unlike previous approaches that have only tackled the problem of mapping preferences to data (see, e.g., [1]), we formally investigate the problem of modifying input

SEBD 2022: The 30th Italian Symposium on Advanced Database Systems, June 19-22, 2022, Tirrenia (PI), Italy
paolo.ciaccia@unibo.it (P. Ciaccia); davide.martinenghi@polimi.it (D. Martinenghi); riccardo.torlone@uniroma3.it (R. Torlone)

Wines

Wine	Winery	Year	
Arneis	Correggia	2019	a
Amarone	Masi	2014	b
Amarone	Bertani	2013	c
Canaiolo	Montenidoli	2015	d
Barolo	Laficaia	2014	e
Arneis	Ceretto	2019	f

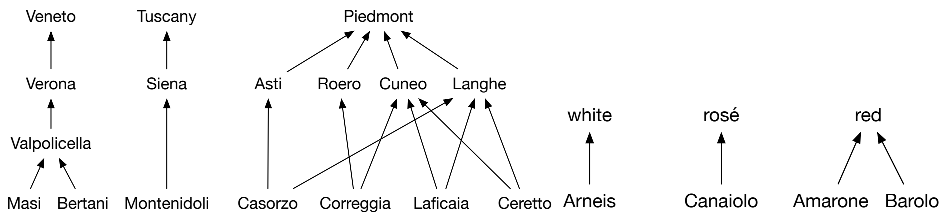


Figure 1: Taxonomies for the running example.

preferences so as to guarantee that they are *transitive* and *specific*. The latter property means that, in case of conflicts, the more specific preference overrides the more generic one. For instance the specific preference for Amarone over white wines counts more than the that for white wines over red ones.

We therefore study, from both a theoretical and a practical point of view, how transitivity and specificity can be obtained by suitable *rewritings* of the initial preferences. We tackle the problem by introducing two operators that rewrite preferences expressed as FO formulas: T to enforce transitivity and S to remove conflicts between preferences. We prove that it is unfortunately *impossible* to guarantee at the same time transitivity and a complete absence of conflicts, no matter the order in which T and S are considered and how many times they are applied. Intuitively, the removal of conflicts may compromise transitivity, whereas enforcement of transitivity may (re-)introduce conflicts. In spite of this, we formally show that: (i) the set of all possible *sequences* of operators can be reduced to a finite (and small) set, (ii) there are only two sequences that guarantee transitivity and minimize residual conflicts between preferences, and (iii) the best results obtainable via these two sequences may be very different.

A number of experiments shows that the computation of the best results largely benefits from the minimization of conflicts between preferences, while incurring a low overhead due to the rewriting process.

The full version of this paper has appeared in [2].

2. Operations on Preferences

We consider a simple extension of the relational model in which the values of an attribute can be arranged in a hierarchical *taxonomy*, i.e., a poset $T = (V, \leq_V)$, where V is a set of *values* and $\leq_V$ is a partial order on V.

Given a set of attribute-taxonomy pairs $A_1 : T_1, \ldots, A_d : T_d$, let $\mathcal{T}$ denote the set of all possible tuples over any schema that can be defined using such pairs. A *preference relation* is a relation $\succeq$ on $\mathcal{T} \times \mathcal{T}$. Given t_1 and t_2 in $\mathcal{T}$, if $t_1 \succeq t_2$ then t_1 is *(weakly) preferable* to t_2. If $t_1 \succeq t_2$ but $t_2 \not\succeq t_1$, then t_1 is *strictly preferable* to t_2, denoted by $t_1 \succ t_2$.

We consider that preferences are expressed via an *intrinsic preference formula* [3] F such that $t_1 \succeq t_2 \Leftrightarrow F(t_1, t_2)$. A formula is a disjunction of DNFs, each of which is called a *preference statement*, formed by one or more *clauses.*

Example 2. The preferences in Example 1 can be expressed as $P_1(x,y) \vee P_2(x,y) \vee P_3(x,y) \vee P_4(x,y)$, which can be compactly written as: $P_1 = \mathsf{white} \succeq \mathsf{red}$ (short form of $P_1(x,y) = (x[\mathsf{Wine}] \leq \mathsf{white}) \wedge (y[\mathsf{Wine}] \leq \mathsf{red})$), $P_2 = \mathsf{Amarone} \succeq \mathsf{white}$, $P_3 = \mathsf{Siena} \succeq \mathsf{Asti}$, $P_4 = \mathsf{Langhe} \wedge \mathsf{aged} \succeq \mathsf{Langhe} \wedge \mathsf{young}$. ■

Now we introduce two operators that can be applied to a preference relation: Transitive closure (T) and Specificity-based refinement (S). Let $\succeq$ denote the initial preference relation; the resulting relation is indicated $\succeq_{\mathsf{T}}$ for T and $\succeq_{\mathsf{S}}$ for S. Multiple application of operators, e.g., first T and then S, leads to the relation $(\succeq_{\mathsf{T}})_{\mathsf{S}}$, which we compactly denote as $\succeq_{\mathsf{TS}}$. In general, for any *sequence* $X \in \{\mathsf{T}, \mathsf{S}\}^*$, $\succeq_{\mathsf{X}}$ is the preference relation obtained from the initial preference relation $\succeq$ by applying the operators in the order in which they appear in X. Notice that $\succeq_{\varepsilon} = \succeq$, where ε denotes the empty sequence.

We describe the behavior of the two operators by means of suitable rewritings of a preference formula. Given a sequence X of operators, and an initial (input) formula $F(x,y)$ inducing the preference relation $\succeq$, $F^{\mathsf{X}}(x,y)$ denotes the rewriting of F that accounts for the application of the X sequence, yielding $\succeq_{\mathsf{X}}$.

Transitive Closure. Transitivity of $\succeq$, and consequently of $\succ$, is a basic requirement of any sound preference-based system. If $\succeq$ is not transitive then $\succ$ might contain cycles, a fact that could easily lead either to empty or non-stable results [2].

The transitive closure operator, denoted T, given an input preference relation $\succeq_{\mathsf{X}}$ yields the preference relation $\succeq_{\mathsf{XT}}$. The transitive closure F^{XT} of an ipf F^{X} with n statements $P_1, \ldots, P_n$ is still a finite ipf that can be computed as described in [3]. In particular, two predicates of the form $(x[A_i] \leq_{V_i} v_1)$ and $(x[A_i] \leq_{V_i} v_2)$, over the same attribute A_i and using the same variable x, are contradictory if values v_1 and v_2 are different and have no common descendant in the taxonomy V_i. If the predicates are of the form $(x[A_i] \leq_{V_i} v_1)$ and $(x[A_i] \not\leq_{V_i} v_2)$, then they are contradictory in case there is a path from v_1 to v_2 in V_i (or $v_1 = v_2$).

The fact that the transitive closure is computed with respect to the (possibly infinite) domain $\mathcal{T}$ of the tuples, and *not* with respect to a (finite) relation r of tuples, is quite standard for preference relations (see e.g., [3]), and has the advantage of yielding a relation $\succeq_{\mathsf{XT}}$ that does not change with r.

Example 3. Continuing with Example 2, the transitive closure of F is the formula F^{T} that, among others, adds the following statements to F:

$$P_5 = \mathsf{Amarone} \succeq_{\mathsf{T}} \mathsf{red} \qquad P_6 = \mathsf{Siena} \succeq_{\mathsf{T}} \mathsf{Langhe} \wedge \mathsf{young}$$

Statement P_5 clearly follows from P_2 and P_1, whereas P_6 is obtained from P_3 and P_4, since there exists at least one winery that is both in the Asti province and in the Langhe region (Casorzo is one of them). ■

After applying the T operator, we simplify the formula as needed, and, in particular, we remove statements that are subsumed by other statements. Similarly, we also simplify statements by removing contradictory clauses and clauses subsumed within the same statement.

Specificity-based Refinement. The most intriguing of our operators is S. As apparent from Example 2, *conflicting preferences*, such as $a \succeq b$ and $b \succeq a$ (induced by P_1 and P_2, resp.), may hold. To solve this problem we resort to a *specificity principle*, borrowed from non-monotonic reasoning, stating that more specific information should prevail over more generic one. Therefore, giving the same importance to, e.g., P_1 and P_2, the former being more generic, contradicts the intuition.

The specificity principle we adopt for analyzing conflicting preferences is based on the *extension* of preferences statements, i.e., on the set of pairs of tuples in $\mathcal{T}$ for which a statement is true.

Definition 1 (Specificity principle). *Let $\succeq_{\mathsf{X}}$ be a preference relation, and let F^{X} be the corresponding formula. Let P_i and P_j be two preference statements in F^{X}. We say that P_i is* more specific *than P_j if, for any pair of tuples $t_1, t_2 \in \mathcal{T}$ such that $P_i(t_1, t_2)$ is true, then $P_j(t_2, t_1)$ is also true, and the opposite does not hold.*

From Definition 1 we can immediately determine how a less specific statement has to be rewritten so as to solve conflicts.

Lemma 1. *A preference statement $P_i(x, y)$ is more specific than $P_j(y, x)$ iff $P_i(x, y)$ implies $P_j(y, x)$ (written $P_i(x, y) \to P_j(y, x)$) and the opposite does not hold.*

According to Lemma 1, when $P_i(x, y)$ implies $P_j(y, x)$ the S operator replaces $P_j(y, x)$ with $P'_j(y, x) = P_j(y, x) \wedge \neg P_i(x, y)$, so that P_i and P'_j do not induce any conflicting preferences.

Example 4. Continuing with Example 3, the application of the S operator amounts to rewriting formula F^{T} by replacing the clause $P_1(x, y)$ with $P_1(x, y) \wedge \neg P_2(y, x)$, since $P_2(y, x) \to P_1(x, y)$. This, after distributing $\neg$ over the two predicates in P_2 and simplifying, leads to the new clause: $P_7 = \mathsf{white} \succeq_{\mathsf{TS}} \mathsf{red} \wedge \neg \mathsf{Amarone}$. ■

3. Sequences of Operators

In this section we analyze the effect of performing the operations described in the previous section, and prove some fundamental properties.

In order to clarify the relationships between the results of the different operations, we introduce the notions of equivalence and containment between sequences of operators.

Definition 2 (Equivalence and containment). *Let $\mathsf{X}, \mathsf{Y} \in \{\mathsf{T}, \mathsf{S}\}^*$; X is* contained *in Y, denoted $\mathsf{X} \sqsubseteq \mathsf{Y}$, if for every initial preference relation $\succeq$, $\succeq_{\mathsf{X}} \subseteq \succeq_{\mathsf{Y}}$; X and Y are* equivalent, *denoted $\mathsf{X} \equiv \mathsf{Y}$, if both $\mathsf{X} \sqsubseteq \mathsf{Y}$ and $\mathsf{Y} \sqsubseteq \mathsf{X}$.*

We observe that T and S are idempotent, T is monotone and cannot remove preferences, while S cannot add preferences. In addition, the preference relation obtained after applying T on the initial preference relation $\succeq$ is maximal, in that it includes all other relations obtained from $\succeq$ by applying T and S in any way.

Theorem 1. *Let* $\mathsf{X}, \mathsf{Y} \in \{\mathsf{T}, \mathsf{S}\}^*$, *with* $\mathsf{X} \sqsubseteq \mathsf{Y}$. *Then:*

$$\begin{array}{rlll}
\mathsf{XTT} \equiv \mathsf{XT} & \mathsf{XSS} \equiv \mathsf{XS} & \textit{idempotence} & (1)\\
\mathsf{XT} \sqsubseteq \mathsf{YT} & & \textit{monotonicity} & (2)\\
\mathsf{X} \sqsubseteq \mathsf{XT} & \mathsf{XS} \sqsubseteq \mathsf{X} & \textit{inflation / deflation} & (3)\\
\mathsf{X} \sqsubseteq \mathsf{T} & & \textit{maximality} & (4)
\end{array}$$

We call *complete* those sequences that include both T and S. A sequence X is *transitive* if, for every initial preference relation $\succeq$, $\succ_{\mathsf{X}}$ is transitive. Eventually, our goal is to drop conflicting and less specific preferences while preserving transitivity. To this end, we add minimality with respect to $\sqsubseteq$ as a desideratum.

The following, non-trivial result shows that we can focus on just eight sequences, shown in Figure 2, because any sequence is equivalent to one of them.

Theorem 2. *Let* $\mathsf{X} \in \{\mathsf{T}, \mathsf{S}\}^*$. *Then* $\exists \mathsf{Y} \in \{\varepsilon, \mathsf{T}, \mathsf{S}, \mathsf{TS}, \mathsf{ST}, \mathsf{TST}, \mathsf{STS}, \mathsf{STST}\}$ *such that* $\mathsf{X} \equiv \mathsf{Y}$.

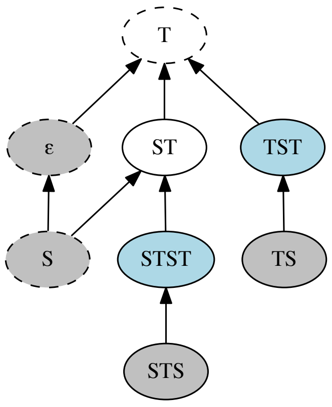


Figure 2: A transitively reduced graph showing containment between sequences. Dashed border for incomplete sequences; grey background for non-transitive sequences; blue background for minimal-transitive sequences.

To give an intuition, we observe that *i)* consecutive repetitions of the same operator are idle (via idempotence) and *ii)* the repeated application of a TS suffix does not change the semantics of a sequence (i.e., $\mathsf{XTS} \equiv \mathsf{XTSTS}$).

Minimality and transitivity. Generally, any complete sequence not ending with S is non-minimal, in that it may contain conflicting preferences (possibly introduced by T) that turn out to be in contrast with other, more specific preferences. We exemplify this on ST using a single taxonomy about time.

Example 5. Let F consist of $P_1 = \mathsf{autumn} \succeq \mathsf{sep}$ and the more specific $P_2 = \mathsf{sep10} \succeq \mathsf{oct10}$. By specificity, in F^{S}, P_1 is replaced by the statement P_3 consisting of two clauses: $\mathsf{autumn} \succeq \mathsf{sep} \wedge \neg\mathsf{sep10}$ and $\mathsf{autumn} \wedge \neg\mathsf{oct10} \succeq \mathsf{sep}$. In F^{ST}, the clauses in P_3 transitively combine into P_1 again, since, e.g., the value $\mathsf{sep30}$ is below sep but not $\mathsf{sep10}$ and below autumn but not $\mathsf{oct10}$; therefore $\mathsf{oct10} \succeq_{\mathsf{ST}} \mathsf{sep10}$ holds. However, in F^{STS}, P_1 is again replaced by P_3, so that $\mathsf{oct10} \not\succeq_{\mathsf{STS}} \mathsf{sep10}$, which shows that ST is not minimal. ■

All the containments indicated in Figure 2 are strict, as can be shown through constructions similar to that of Example 5, so no sequence ending with T is minimal in $\{\mathsf{T}, \mathsf{S}\}^*$.

Transitivity is achieved for any sequence ending with T, while no sequence ending with S is transitive. Therefore, we summarize our finding in the following major result.

Theorem 3. *Let* $\mathsf{X} \in \{\mathsf{T}, \mathsf{S}\}^*$. *Then* XT *is not minimal and* XS *is not transitive, thus no sequence is both transitive and minimal.*

Since transitivity and minimality cannot be both achieved at the same time, we enforce transitivity (which cannot be waived) and look for the minimal sequences among those that are transitive, which we call *minimal-transitive* sequences.

We first observe that all complete sequences starting with S are incomparable with those starting with T (also refer to Figure 2). Therefore, only three sequences are both complete and transitive: ST, TST and STST, the first of which contains the last one and is therefore not minimal. The remaining two sequences are transitive, incomparable, and, therefore, minimal in the set of complete and transitive sequences.

Theorem 4. *The only minimal-transitive sequences are* TST *and* STST.

The result of Theorem 3 is inherent and no finer granularity in the interleaving of T and S (e.g., by making S resolve one conflict at a time instead of all together) would remove this limitation. Furthermore, this limitation is unavoidable and we can prove that no method whatsoever (not just those based on the T and S operators) could solve it.

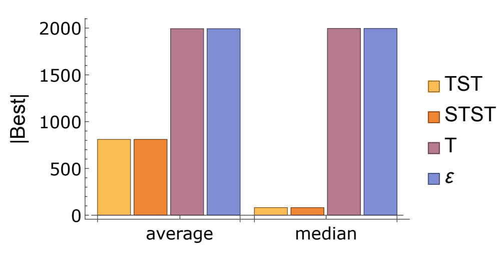
4. Obtaining the Best Results

Given a relation $r \subseteq \mathcal{T}$, the "best" tuples in r according to the preference relation $\succeq$ can be selected by means of the *Best* operator β [4], which returns the tuples t_1 of r such that there is no other tuple t_2 in r that is *strictly* preferable to t_1, i.e., $\beta_{\succ}(r) = \{t_1 \in r \mid \nexists t_2 \in r, t_2 \succ t_1\}$.

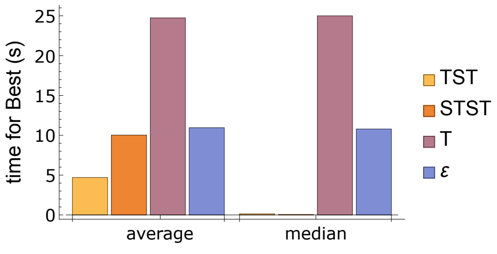
As shown in Section 3, TST and STST are incomparable, thus there will be relations r and input preference relations $\succeq$ for which the best results delivered by the two semantics will differ. In order to quantify the difference between these results, we define, for any two sequences X and Y, $\textsc{Diff}_\beta(\mathsf{X}, \mathsf{Y}, n)$ as the worst-case size of the difference in the results delivered by X with respect to those due to Y, for any given cardinality of the input relation r. We have:

Theorem 5. $\textsc{Diff}_\beta(\mathsf{TST}, \mathsf{STST}, n) = \Theta(n)$; $\textsc{Diff}_\beta(\mathsf{STST}, \mathsf{TST}, n) = \Theta(n)$.

From a practical point of view, Theorem 5 shows that there is no all-seasons minimal-transitive semantics. Furthermore, there can be cases (used in the proof of the theorem) in which the



(a) Cardinality of β.



(b) Time for computing β.

Figure 3: Computing β with default parameter values.

number of best results from any of the two semantics is comparable to n, whereas the other semantics returns $\mathcal{O}(1)$ tuples.

For space reasons, we only give hints at our experimental results and refer the reader to [2] for details. As for the rewriting of the input formula, in all cases we incur a low overhead, negligible with respect to the time required for computation of β.

In our experiments, as recognized in the germane literature [5], we only consider *relevant* tuples, i.e., those that satisfy either side of at least one clause in the rewritten formula, since the other tuples correspond to those objects that the formula does not talk about.

The algorithm we adopt for computing β is an improved version of BNL, exploiting a heuristics tailored to our scenario, which pre-sorts the input relation so that the tuples matching the left side of a clause and corresponding to more specific values in the taxonomies come first. The rationale is that these values are likely associated with a smaller amount of tuples, so that a smaller β partial result can be found before scanning large amounts of data. Furthermore, such tuples are likely to be preferred to many others, in particular when specificity is a concern.

Besides testing real taxonomies, we experimented on a variety of synthetic datasets, generated according to various taxonomy topologies (regular, random, and scale-free), dataset sizes (up to 1M tuples), attributes (up to 5), and input preferences (up to 10). Figure 3a shows our results using default parameter values (regular taxonomy on 1 attribute, two conflicting preference statements, 10K tuples). The amount of relevant tuples is roughly 40% of the size of the dataset. Both T (i.e., only enforcing transitivity) and ε (i.e., no rewriting of the preference formula) retain about half of the relevant tuples (which is both the average and the median value we obtained), while TST and STST retain less than 2% in the median case (the average value goes up to 20% due to runs with unfocused input formulas referring to values not in the dataset). This is reflected in the computation times, shown in Figure 3b, which are consistently around $24s$ for T and $10s$ for ε, but nearly two orders of magnitude smaller in the median case for TST and STST.

This confirms that our approach is effective both in reducing the cardinality of β and in achieving substantial speedup with respect to baseline strategies. Similar results are obtained in all other scenarios we tested.

We observe that the application of T alone corresponds to the work performed by preference evaluation methods that only aim at guaranteeing transitivity, e.g., [3], which are therefore outperformed by our approach. The inability of T to deal with conflicting preferences, thus generating many indifferent tuples, which in turn induce (very) large result sets, indeed applies to all our scenarios. Similar observations apply to ε (i.e., the empty sequence, corresponding to the input formula), which represents the action of works on preference evaluation using no rewriting whatsoever, such as [1]. Additionally, the results obtained via ε would be totally unreliable, due to lack of transitivity.

5. Conclusions

In this paper we have tackled the problem of finding the best elements from a repository on the basis of preferences referring to values that are more generic than the underlying data and may involve conflicts. To this aim, we have introduced and formally investigated two operators for enforcing, in a given collection of preferences, the properties of specificity, which can solve conflicts, and transitivity, which guarantees the soundness of the final result. We have then characterized the limitations that can arise from their combination and identified the best ways in which they can be used together. We have finally shown, with experiments over both synthetic and real-world datasets, the effectiveness and practical feasibility of the overall approach. We remark that the need to address conflicts arising from preferences was also observed in [6, 7]. The framework proposed there allows for a restricted form of taxonomies [8] (with all values organized into distinct, named levels) and hints at an ad hoc procedure with very limited support for conflict resolution; the focus of [6, 7] is, however, on the downward propagation of preferences.

References

[1] T. Lukasiewicz, M. V. Martinez, G. I. Simari, Preference-based query answering in datalog+/- ontologies, in: IJCAI 2013, pp. 1017–1023. URL: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6505.

[2] P. Ciaccia, D. Martinenghi, R. Torlone, Preference queries over taxonomic domains, Proc. VLDB Endow. 14 (2021) 1859–1871. URL: http://www.vldb.org/pvldb/vol14/p1859-martinenghi.pdf.

[3] J. Chomicki, Preference formulas in relational queries, ACM Trans. Database Syst. 28 (2003) 427–466. URL: https://doi.org/10.1145/958942.958946. doi:10.1145/958942.958946.

[4] P. Ciaccia, D. Martinenghi, R. Torlone, Foundations of context-aware preference propagation, J. ACM 67 (2020) 4:1–4:43. URL: https://doi.org/10.1145/3375713. doi:10.1145/3375713.

[5] P. Georgiadis et al., Efficient rewriting algorithms for preference queries, in: ICDE,2008, pp. 1101–1110. URL: https://doi.org/10.1109/ICDE.2008.4497519.

[6] P. Ciaccia, D. Martinenghi, R. Torlone, Finding preferred objects with taxonomies, in: ER, 2019, pp. 397–411. URL: https://doi.org/10.1007/978-3-030-33223-5_33.

[7] P. Ciaccia, D. Martinenghi, R. Torlone, The POOR-MAD approach: Preferred objects over rich, multi-attribute data, in: SEBD, 2021, pp. 283–290. URL: http://ceur-ws.org/Vol-2994/paper30.pdf.

[8] D. Martinenghi, R. Torlone, Taxonomy-based relaxation of query answering in relational databases, VLDB J. 23 (2014) 747–769. URL: https://doi.org/10.1007/s00778-013-0350-x.