Noname manuscript No.
(will be inserted by the editor)

Taxonomy-based Relaxation of Query Answering in Relational Databases

Davide Martinenghi · Riccardo Torlone

Received: date / Accepted: date

Abstract Traditional information search in which queries are posed against a known and rigid schema over a structured database is shifting towards a Web scenario in which exposed schemas are vague or absent and data comes from heterogeneous sources. In this framework, query answering cannot be precise and needs to be relaxed, with the goal of matching user requests with accessible data. In this paper, we propose a logical model and a class of abstract query languages as a foundation for querying relational data sets with vague schemas. Our approach relies on the availability of taxonomies, that is, simple classifications of terms arranged in a hierarchical structure. The model is a natural extension of the relational model in which data domains are organized in hierarchies, according to different levels of generalization between terms. We first propose a conservative extension of the relational algebra for this model in which special operators allow the specification of relaxed queries over vaguely structured information. We also study equivalence and rewriting properties of the algebra that can be used for query optimization. We then illustrate a logic-based query language that can provide a basis for expressing relaxed queries in a declarative way. We finally investigate the expressive power of the proposed query languages and the independence of the taxonomy in this context.

D. Martinenghi
Politecnico di Milano, Italy
Tel.: +39 02 2399 3669
Fax: +39 02 2399 3411
E-mail: martinen@elet.polimi.it

R. Torlone
Università Roma Tre, Italy
Tel.: +39 06 5733 3377
Fax: +39 06 5733 3612
E-mail: torlone@dia.uniroma3.it

1 Introduction

There are today many application scenarios in which user queries do not match the structure and the content of data repositories. This may happen due to the nature of the application domain or simply because the schema is not available. Examples of mismatch between query and data occur in location-based search (find an opera concert in Paris next summer), multi-faceted product search (find a cheap blu-ray player with an adequate user rating), multi-domain search (find a database conference held in a seaside location), and social search (find the objects that my friends like). In these situations, given the complexity and heterogeneity of data sources, data structure and organization is usually made transparent to the user. Therefore, the query needs to be relaxed to accommodate user's needs, while query answering relies on finding the best match between the request and the available data.

In spite of this trend towards "schema-agnostic" applications, the support of current database technology for query relaxation is quite limited. The only examples are in the context of semi-structured information, in which schemas and values are varied and/or missing [6], and semantic data, where data may be highly diverse [22]. Conversely, the above mentioned applications can greatly benefit from applying traditional relational database technology enhanced with a comprehensive support for the management of query relaxation.

To this end, we propose in this paper a logical data model and a number of abstract query languages supporting query relaxation over relational data. Our approach takes advantages of the availability of taxonomies, that is, simple ontologies in which terms used in schemas and data are arranged in a hierarchical

structure according to a generalization-specialization relationship. The data model is a natural extension of the relational model in which data domains are organized in hierarchies, according to different levels of detail: this guarantees a smooth implementation of the approach with current database technology. In this model, data and metadata can be expressed at different levels of detail. This is made possible by a partial order relationship defined both at the schema and at the instance level.

The first query language we propose, termed TRA (Taxonomy-based Relational Algebra), is a conservative extension of relational algebra. TRA includes two special operators that extend the capabilities of standard selection and join by relating values occurring in tuples with values in the query using the taxonomy. In this way, we can formulate relaxed queries that refer to attributes and terms different from those occurring in the actual database. We also present general algebraic rules governing the operators over taxonomies and their interactions with standard relational algebra operators. The rules provide a formal foundation for query equivalence and for the algebraic optimization of queries over vague schemas.

We then present HDRC (H-Domain Relational Calculus), a logic-based query language that provides a basis for expressing relaxed queries over relational databases in a declarative way. The comparison between TRA and HDRC provides insights on the strengths and weaknesses of these languages in terms of expressive power and finiteness of query answers. To this end, we investigate the notion of domain independence in this context, extend it to the more general notion of taxonomy independence, and characterize the expressive power and taxonomy independence of TRA and HDRC by comparing several variants thereof.

In sum, the contributions of this paper are the following:

- a simple but solid framework for embedding taxonomies into relational databases: the framework does not depend on a specific domain of application and makes the comparison of heterogeneous data possible and straightforward;
- a simple but powerful algebraic language for supporting query relaxation: this language makes it possible to formulate, in a procedural way, complex searches over vague schemas in different application domains;
- the investigation of the relationships between the query language operators and the identification of a number of equivalence rules: the rules provide a formal foundation for the algebraic optimization of relaxed queries;
- a declarative, logic-based language for supporting query relaxation: this language provides a basis for an extension of SQL able to exploit taxonomies for expressing relaxed queries over relational data;
- the extension of the notion of domain independence (termed taxonomy independence) suitable for this context, and the precise characterization of the expressive power of various logical and algebraic versions of the query language;
- a discussion on implementation concerns, on the completeness of the apparatus of languages (algebraic and logic-based) that are formalized in the paper, as well as on further extensions of the work that might be taken into account in order to provide users with a yet more flexible and usable querying tool.

In this paper we focus on the general aspects regarding the model and the query languages, whereas we do not address the issue of implementing the formal framework proposed and we disregard the orthogonal problem of taxonomy design.

To our knowledge, this is the first attempt to provide a foundation to taxonomy-based query relaxation in relational database systems. Indeed, as we will discuss in the related work section, many approaches have been proposed to the problem of supporting user searches with non-traditional techniques of query processing. However, our taxonomy-based relaxation is a mechanism for query evaluation that is orthogonal to previous attempts, and therefore might even be combined with other proposals present in the literature. Indeed, we do not claim that taxonomies can solve all the problems related to query answering over unknown schemas, but we intend to show that they can provide a significant contribution to this issue with a rather limited effort.

The rest of the paper is organized as follows. In Section 2, we present several application scenarios that motivate our work. In Section 3 we introduce some preliminary notions and present the data model. The algebraic query language for this model, TRA, is illustrated in Section 4, where we also provide a number of equivalence rules that can be used for query optimization. Section 5 is devoted to the presentation of HDRC and the investigation of the expressive power and of the taxonomy independence of TRA, HDRC, and variants thereof. In Section 6, we discuss several issues regarding the impact and possible extensions of our work. In Section 7 we compare our approach with related works and finally, in Section 8, we draw some conclusions and sketch future works.

Concerts

Title	Style	Day	Place	
Rigoletto	Classical	09/08/2013	Verona Arena	t_a
Bruce Springsteen	Rock	29/06/2013	Stade de France	t_b
Robbie Williams	Pop	12/07/2013	Wembley Stadium	t_c
George Benson	Jazz	09/04/2013	Olympia	t_d

Fig. 1 A table storing a list of events.

2 Motivating examples and applications

In this section we illustrate a number of different real-world scenarios in which the availability of a support to query relaxation can make user searches more flexible and effective. The first one defines the context that will be used for our running examples throughout the paper.

2.1 Location-based search

There is today a large availability of location-based information sources in a variety of contexts, such as entertainment, real estate, business directory, health care, weather reporting, and more. Very often, however, data are organized and accessed on the basis of a specific schema that does not match the needs of users to search that data.

Consider for instance a catalog of concerts in Europe stored in a relational table organized as illustrated in Figure 1. Assume now that we are planning a trip to Italy next summer, where we would also like to attend a musical performance of Italian opera. Tuple t_a actually satisfies the request (Arena is a Roman amphitheater in Verona, Italy, which hosts concerts during the summer, and Rigoletto is a famous Italian opera), but such a result can be very hard to retrieve because of the mismatch between the query we would like to formulate and the way in which data are represented and stored. Indeed, this kind of information is usually accessible only via rigid Web interfaces that hardly capture user needs and often return too many answers or no answer at all.

This problem can be alleviated if the storage system is able to answer relaxed queries that are reasonable in the given application domain even if they do not match the database schema exactly. In particular, we consider the frequent case in which attributes and values in the query are more general or more specific than those in the database, as in the following example, which expresses the request above.

```
SELECT *
FROM Concerts
WHERE Country = 'Italy' AND
      Season = 'Summer 2013' AND
      Genre = 'Opera'
```

Our approach aims at supporting this kind of relaxed queries by taking advantage of taxonomies between values defined on the various domains of the attributes. For instance, by extending the domain of the attribute Place of the relation in Figure 1 with a geographical taxonomy of the kind shown in Figure 2, we can easily verify that tuple t_a satisfies, in a relaxed way, the first condition of the query.

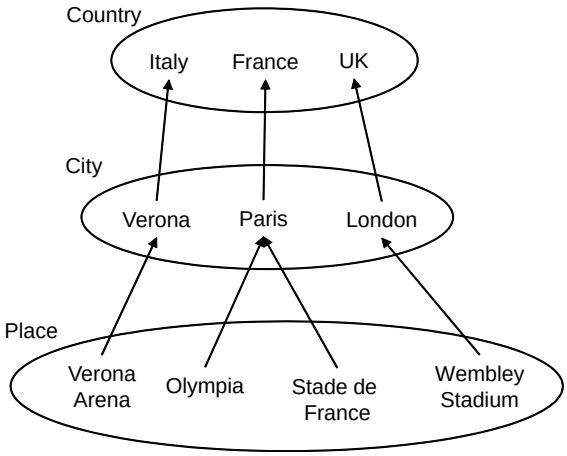

Fig. 2 A geographical taxonomy

Similarly, by using a temporal taxonomy and a classification of musical genres we can verify that t_a also satisfies the second and the third condition of the query. Note that, while testing the first two conditions requires to traverse the taxonomy upward (from a more specific term occurring in the database to a more general term occurring in the query), the last one needs to traverse the taxonomy in the opposite direction (from the more general notion of "Classical" used in the database to the more specific "Opera" term specified in the query). This suggests a need for two modalities of taxonomy-based relaxation, which we shall adopt in this paper, according to the direction of the relaxation.

Conferences

Name	Period	City	
SEBD	30/06–03/07, 2013	Roccella	$t_{c,1}$
CIKM	27/10–01/11, 2013	San Francisco	$t_{c,2}$

Temperatures

Month	Region	Average	
June	Calabria	28	$t_{t,1}$
July	Calabria	30	$t_{t,2}$

Fig. 3 Two non-matching relations.

2.2 Multi-domain search

Currently, there is a huge number of specialized data sources that cover specific domains very well, but that need to be integrated with others in order to obtain valuable information.

Assume, for instance, that we are looking for a computer science conference taking place in a period with an expected local average temperature of at least 24°C. This query can be answered by matching a catalog of conferences with a weather database containing seasonal average temperatures. The problem is that, even if the content of both data sources is gathered and stored in two tables of a single database, it is very likely that common information is represented at different detail levels.

For instance, as shown in Figure 3, the timeline could be represented as a start and end date in the conference catalog and on a monthly base in the weather database. Moreover, it may well be the case that the former lists the city of the conference, whereas, in the latter, average temperatures refer to regional areas. Again, it can be hard to realize that tuple $t_{c,1}$ in relation Conferences matches with both tuples $t_{t,1}$ and $t_{t,2}$ in relation Temperatures (Roccella is a village located in Calabria, a region of southwestern Italy, and the SEBD conference starts in June and ends in July).

In this case, we would need a relaxed join operation that is able to match, respectively, values from the attributes Period and City of the relation Conferences with values from the attributes Month and Region of the relation Temperatures. Again, the availability of a geographical and a temporal taxonomy can be very helpful to fulfill this task.

Note that, while the join condition on City and Region needs to find a common ancestor in the geographical taxonomy, which requires to traverse the taxonomy upwards, the join condition on Period and Month needs to find a common descendant in the temporal taxonomy, which requires to traverse the taxonomy downwards. This suggests that also the taxonomic join should come in two versions, according to the way in which the taxonomies are visited to check the join condition.

2.3 Trip planners

A trip planner finds one or more suggested journeys between an origin and a destination. In public transportation, such points are typically described by specific addresses, and the trip details include the sequence of transport steps to be taken by the traveler to reach the destination, usually in the form of bus/metro stops and paths on a map.

The MOKA project[1] addresses urban mobility in Italy through heterogeneous transport modes, with emphasis on the Milan area, with the goal of providing travelers with estimates of trip durations in real time, expressed as minimum-duration/maximum-duration pairs. Particular care needs to be taken when computing, in real time, long-distance journeys involving several different kinds of transportation at the finest level of granularity (typically, that of the single bus/metro stops), since solving mobility problems (e.g., by Dijkstra's algorithm) may take too long for a graph with many nodes. Therefore, in such cases it is of uttermost importance to be able to scale the problem up to a coarser representation level involving fewer nodes. For example, when planning a trip from Politecnico di Milano to Roma Tre University, it is convenient to start by finding the best way to go from Milan's to Rome's area (say, high-speed rail). Then, the remaining part of the plan amounts to computing two new sub-journeys: one that moves from Politecnico to the train station in Milan, and another one from the train station in Rome to the university. An alternative solution is to fly from Milan to Rome, which then amounts to finding connections with the city airports. Similarly, the sub-journeys may be analyzed at different levels of granularity, from single stops of public transportation (finest level) up to neighborhoods, quarters, cities, countries, and so on. In this context, taxonomies are the appropriate tool for conveying information at the desired level of granularity.

We remark that leveling the structure of the data into taxonomies is unavoidable in order to preserve tractability of the problem solution as the problem size scales to a larger, possibly world-wide scenario. In this

[1] "MOKA: an infrastructure for public transit integrated car pooling", a project funded by Politecnico di Milano. Website: `http://moka.necst.it/app/index.html`

respect, the MOKA system currently covers parts of the Italian transportation network, but a new version is under development, which aims at covering transportation at the European level. To this end, the MOKA project needs to make use of the Taxonomic Relational Algebra operators that are going to be introduced in the following sections.

According to a procedural approach being investigated in MOKA, a trip planning query may be conceived as a sequence of relaxed selection queries invoking the planning service (i.e., a virtual relation computed on the fly) with a start and an end point at a given level of granularity. In the first invocation, the selected level is the coarsest level for which source and destination differ (e.g., the city level in the example with Politecnico di Milano and Roma Tre University). Then, the plan is split into sub-journeys, as mentioned earlier, each of which is computed in a similar way, until completion of the plan.

This taxonomy-based approach proves particularly useful when travel information needs to be matched with tourism-related data, such as points of interest and events that vary so quickly (e.g., festivals, exhibitions, and seasonal events of culturally lively areas) that they are typically unavailable in the trip planner's database but are rather stored in external sources. For all these cases, the availability of relaxed taxonomic joins, as those discussed in the previous subsection, would be able to capture space and/or time proximity.

2.4 Genometric queries

DNA sequencing is a technology that is changing biological research and will change medical practice; availability of individual genomes may soon become the biggest and most important "big data" problem of mankind. Within the GenData 2020 project[2] we are defining a new paradigm for raising the level of abstraction in genome data management by introducing a genometric query language based on the framework presented in this paper.

In the current bioinformatic practice, many different groups of researchers annotate data resulting from their experiments. When researchers from other groups need to query these data (possibly coming from several such repositories), they are typically unaware of the level of specificity of the annotations, and thus need to manually cope with possible mismatches between the granularity of the query they have in mind and the granularity of the annotations in the data. Fortunately, the community may count on largely agreed-upon ontologies such as the Gene Ontology[3] – a data source independent and decoupled from the annotated data, which mostly consist of *is-a* and *part-of* relationships between terms, and thus conveys significant taxonomic information. This information may be used by researchers to relax their queries and allow the results to match the intended meaning despite syntactic mismatches. This need is notably found in genometric queries, currently being investigated in Gendata 2020, like the following: "Find DNA-seq datasets showing mutations within the exons of a given gene G, considering only samples obtained from human brain cells". Note that here "human brain" is an element of a taxonomy, a part of which is, e.g., the "frontal lobe": clearly, data annotated with the latter are also potentially relevant for the query, although the "frontal lobe" is never mentioned in it.

3 A Data Model with Taxonomies

In this section, we present an extension of the relational model in which domains are organized in simple taxonomies of generalization-specialization relationships. We start with some preliminary notions on partial orders, which are basic ingredients of our model.

3.1 Preliminaries

A (weak) *partial order* $\leq$ on a domain V is a subset of $V \times V$ whose elements are denoted by $v_1 \leq v_2$ that is: reflexive ($v \leq v$ for all $v \in V$), antisymmetric (if $v_1 \leq v_2$ and $v_2 \leq v_1$ then $v_1 = v_2$), and transitive (if $v_1 \leq v_2$ and $v_2 \leq v_3$ then $v_1 \leq v_3$). If $v_1 \leq v_2$ we say that v_1 *is included in* v_2. A set of values V with a partial order $\leq$ is called a *poset*.

A lower bound (upper bound) of two elements v_1 and v_2 in a poset $(V, \leq)$ is an element $b \in V$ such that $b \leq v_1$ and $b \leq v_2$ ($v_1 \leq b$ and $v_2 \leq b$). A *maximal lower bound* (*minimal upper bound*) is a lower bound (upper bound) b of two elements v_1 and v_2 in a poset $(V, \leq)$ such that there is no lower bound (upper bound) b' of v_1 and v_2 such that $b' \leq b$ ($b \leq b'$).

3.2 Taxonomies and t-relations

The basic construct of our model is the *hierarchical domain* or simply the *h-domain*, a collection of values arranged in a containment hierarchy. Each h-domain is described by means of a set of *levels* representing the

[2] GenData 2020 ("Data-Driven Genomic Computing") is a project funded by MIUR (Italian Ministry of Education, University and Research) involving a large consortium of Italian universities: see `http://gendata.weebly.com/`

[3] `http://www.geneontology.org`

domain of interest at different degrees of granularity. For instance, the h-domain *time* can be organized in levels like *day*, *week*, *month*, and *year*.

Definition 1 (H-domain and taxonomy) An *h-domain* h is composed of:

- a finite set $L = \{l_1, \ldots, l_k\}$ of *levels*, each of which is associated with a set of values called the *members* of the level and denoted by $M(l)$;
- a partial order $\leq_L$ on L having a bottom element, denoted by $\bot_L$, and a top element, denoted by $\top_L$, such that:
  - $M(\bot_L)$ contains a set of *ground* members whereas all the other levels contain members that represent groups of ground members;
  - $M(\top_L)$ contains only a special member $m_\top$ that represents all the ground members;
- a family LM of functions $\text{LMAP}_{l_1}^{l_2} : M(l_1) \to M(l_2)$, called *level mappings*, for each pair of levels $l_1 \leq_L l_2$ satisfying the following *consistency conditions*:
  - for each level l, the function LMAP_l^l is the identity on the members of l;
  - for each pair of levels l_1 and l_2 such that $l_1 \leq_L l \leq_L l_2$ and $l_1 \leq_L l' \leq_L l_2$ for some $l \neq l'$, we have: $\text{LMAP}_l^{l_2}(\text{LMAP}_{l_1}^{l}(m)) = \text{LMAP}_{l'}^{l_2}(\text{LMAP}_{l_1}^{l'}(m))$ for each member m of l_1.

A *taxonomy* is a finite set of h-domains.

Example 1 An example of a possible taxonomic organization of the h-domain *location* is reported in Figure 2, where the levels Place, City, and Country are represented. As another example, the h-domain *time* has a bottom level whose (ground) members are timestamps and a top level whose only member, `anytime`, represents all possible timestamps. Other levels can be day, week, month, quarter, season and year, where day $\leq_L$ month $\leq_L$ quarter $\leq_L$ year and day $\leq_L$ season. A possible member of the day level is `23/07/2012`, which is mapped by the level mappings to the member `07/2012` of the level month and to the member `Summer` of the level season.

As should be clear from Definition 1 and Example 1, in this paper we consider a general notion of taxonomy that involves terms arranged in a containment hierarchy: this allows the representation of both subsumptive (is–a) and compositional (part–of) relationships between values.

A partial order $\leq_M$ can also be defined on the members M of an h-domain h: it is induced by the level mappings as follows.

Definition 2 (Poset on members) Let h be an h-domain and m_1 and m_2 be members of levels l_1 and l_2 of h, respectively. We have that $m_1 \leq_M m_2$ if: (i) $l_1 \leq_L l_2$ and (ii) $\text{LMAP}_{l_1}^{l_2}(m_1) = m_2$.

Example 2 Consider the h-domain of Example 1. Given the members $m_1 =$ `29/06/2012` and $m_2 =$ `23/08/2012` of the level day, $m_3 =$ `06/2012` and $m_4 =$ `08/2012` of the level month, $m_5 =$ `2Q 2012` and $m_6 =$ `3Q 2012` of the level quarter, $m_7 =$ `2012` of the level year, and $m_8 =$ `Summer` of the level season, we have: $m_1 \leq_M m_3 \leq_M m_5 \leq_M m_7$, $m_2 \leq_M m_4 \leq_M m_6 \leq_M m_7$, and $m_1 \leq_M m_8$ and $m_2 \leq_M m_8$.

We are ready to introduce the main construct of the data model: the t-relation, a natural extension of a relational table built over taxonomies of values.

Definition 3 (T-schema) Let T be a taxonomy. We denote by $S = \{A_1 : l_1, \ldots, A_k : l_k\}$ a *t-schema* (schema over taxonomies) for T, where each A_i is a distinct *attribute* name and each l_i is a level of some h-domain in T.

Definition 4 (T-relation and t-database) A *t-tuple* t over a t-schema $S = \{A_1 : l_1, \ldots, A_k : l_k\}$ for a taxonomy T is a function mapping each attribute A_i to a member of l_i. A *t-relation* r over S is a set of t-tuples over S. Finally, a *t-database* d over a set of t-schemas $\mathbf{S} = \{S_1, \ldots, S_n\}$ for T is a set of t-relations $r_1, \ldots, r_k$ over $S_1, \ldots, S_n$, respectively.

Given a t-tuple t over a t-schema S and an attribute A_i occurring in S on level l_i, we will denote by $t[A_i : l_i]$ the member of level l_i associated with t on A_i. Following common practice in relational database literature, we use the same notation $A : l$ to indicate both the single attribute-level pair $A : l$ and the singleton set $\{A : l\}$; also, we indicate the union of attribute-level pairs (or sets thereof) by means of the juxtaposition of their names. For a subset S' of S, we will denote by $t[S']$ the restriction of t to S'. Finally, for the sake of simplicity, often in the following we will not make any distinction between the name of an attribute of a t-relation and the name of the corresponding h-domain, when no ambiguities can arise.

Example 3 The example discussed in Section 2.1 can be represented by the t-schema $S = \{Title : \text{Title}, Category : \text{Style}, Time : \text{Day}, Location : \text{Place}\}$. A possible t-relation over this schema is shown in Figure 4. Then we have: t_a[Category:Style] = Classical.

A partial order relation on both t-schemas and t-relations can be also defined in a natural way.

Definition 5 (Poset on t-schemas) Let S_1 and S_2 be t-schemas over a taxonomy T. We have that $S_1 \leq_S S_2$ if for each $A_i : l_i \in S_2$ there is an element $A_i : l_j \in S_1$ such that $l_j \leq_L l_i$.

Concerts

Title:Title	Category:Style	Time:Day	Location:Place	
Rigoletto	Classical	09/08/2013	Verona Arena	t_a
Bruce Springsteen	Rock	29/06/2013	Stade de France	t_b
Robbie Williams	Pop	12/07/2013	Wembley Stadium	t_c
George Benson	Jazz	09/04/2013	Olympia	t_d

Fig. 4 A t-relation for the schema of Example 3.

Definition 6 (Poset on t-tuples) Let t_1 and t_2 be t-tuples over S_1 and S_2 respectively. We have that $t_1 \leq_t t_2$ if: (i) $S_1 \leq_S S_2$, and (ii) for each $A_i : l_i \in S_2$ there is an element $A_i : l_j \in S_1$ such that $t_1[A_i : l_j] \leq_M t_2[A_i : l_i]$.

Definition 7 (Poset on t-relations) Let r_1 and r_2 be t-relations over S_1 and S_2 respectively. We have that $r_1 \leq_r r_2$ if for each t-tuple $t \in r_1$ there is a t-tuple $t' \in r_2$ such that $t \leq_t t'$.

Note that, in these definitions, we assume that levels of the same h-domain occur in different t-schemas with the same attribute name: this strongly simplifies the notation that follows without loss of expressibility. Basically, it suffices to use as attribute name the role played by the h-domain in the application scenario modeled by the t-schema.

Example 4 Consider the t-relations and t-schemas in Figure 5. It is easy to see that: (i) $S_1 \leq_S S_2$, and (ii) $t_{1,1} \leq_t t_{2,2}$, $t_{1,2} \leq_t t_{2,4}$, $t_{1,3} \leq_t t_{2,1}$, and $t_{1,4} \leq_t t_{2,4}$. It follows that $r_1 \leq_r r_2$.

In the following, for the sake of simplicity, we will often make no distinction between the name of an attribute and the corresponding level.

4 An Algebraic Query Language

In this section we present TRA (Taxonomy-based Relational Algebra) an extension of relational algebra over t-relations. This language provides insights on the way in which data can be manipulated taking advantage of available taxonomies over those data. Moreover, for its procedural nature, it can be profitably used to specify query optimization. The goal is to provide a solid foundation to querying databases with taxonomies.

4.1 TRA: syntax and semantics

Similarly to what happens with the standard relational algebra, the operators of TRA are closed, that is, they apply to t-relations and produce a t-relation as result. In this way, the various operators can be composed to form the *t-expressions* of the language.

TRA is a conservative extension of basic relational algebra (RA) and so it includes its standard operators: selection (σ), projection (π), natural join ($\bowtie$), union ($\cup$), difference ($-$), and renaming (ρ). It also includes some variants of these operators that are obtained by combining them with the following two new operators.

Definition 8 (Upward extension) Let r be a t-relation over S, A be an attribute in S defined over a level l, and l' be a level such that $l \leq_L l'$. The *upward extension* of r to l', denoted by $\hat{\varepsilon}^{A:l'}_{A:l}(r)$, is the t-relation over $S \cup \{A : l'\}$ defined as follows:

$$\hat{\varepsilon}^{A:l'}_{A:l}(r) = \{t \mid \exists t' \in r : t[S] = t', t[A : l'] = \textsc{lmap}^{l'}_{l}(t'[A : l])\}$$

Definition 9 (Downward extension) Let r be a t-relation over S, A be an attribute in S defined over a level l, and l' be a level such that $l' \leq_L l$. The *downward extension* of r to l', denoted by $\check{\varepsilon}^{A:l}_{A:l'}(r)$, is the t-relation over $S \cup \{A : l'\}$ defined as follows:

$$\check{\varepsilon}^{A:l}_{A:l'}(r) = \{t \mid \exists t' \in r : t[S] = t', t'[A : l] = \textsc{lmap}^{l}_{l'}(t[A : l'])\}$$

For simplicity, in the following we will often simply write $\hat{\varepsilon}^{l'}_{l}$ or $\check{\varepsilon}^{l'}_{l}$, when there is no ambiguity on the attribute name associated with the corresponding levels. In addition, for a sequence $\hat{\varepsilon}^{A:l'_1}_{A:l_1} \cdots \hat{\varepsilon}^{A:l'_n}_{A:l_n}$ of applications upward extension, we will use the shorthand notation $\hat{\varepsilon}^{A:l'_1 \cdots A:l'_n}_{A:l_1 \cdots A:l_n}$. Similarly for downward extension.

Example 5 Consider the t-relations r_1 and r_2 from Example 4 (Figure 5). The result of $\hat{\varepsilon}^{\mathsf{city}}_{\mathsf{theater}}(r_1)$ is the t-relation r_3 shown in Figure 6. The result of $\check{\varepsilon}^{\mathsf{quarter}}_{\mathsf{month}}(r_2)$ is the t-relation r_4 shown in Figure 6.

The main rationale behind the introduction of the upward extension is the need to relax a query with respect to the level of detail of the queried information. For example, one might want to find events taking place in a given country, even though the events might be

$S_1 =$

Title:cultural-event	Author:artist	Time:day	Location:theater	
Romeo & Juliet	Prokofiev	13/04/2012	La Scala	$t_{1,1}$
Carmen	Bizet	24/05/2012	Opéra Garnier	$t_{1,2}$
Requiem	Verdi	28/03/2012	La Scala	$t_{1,3}$
La bohème	Puccini	09/04/2012	Opéra Garnier	$t_{1,4}$

$r_1 =$

$S_2 =$

Title:event	Time:quarter	Location:city	
Concert	1Q 2012	Milan	$t_{2,1}$
Ballet	2Q 2012	Milan	$t_{2,2}$
Sport	3Q 2012	Rome	$t_{2,3}$
Opera	2Q 2012	Paris	$t_{2,4}$

$r_2 =$

Fig. 5 T-relations and t-schemas for Example 4.

$S_3 =$

Title:cultural-event	Author:artist	Time:day	Location:theater	Location:city	
Romeo & Juliet	Prokofiev	13/04/2012	La Scala	Milan	$t_{3,1}$
Carmen	Bizet	24/05/2012	Opéra Garnier	Paris	$t_{3,2}$
Requiem	Verdi	28/03/2012	La Scala	Milan	$t_{3,3}$
La bohème	Puccini	09/04/2012	Opéra Garnier	Paris	$t_{3,4}$

$r_3 =$

$S_4 =$

Title:event	Time:quarter	Location:city	Time:month	
Concert	1Q 2012	Milan	Jan 2012	$t_{4,1}$
Concert	1Q 2012	Milan	Feb 2012	$t_{4,2}$
Concert	1Q 2012	Milan	Mar 2012	$t_{4,3}$
Ballet	2Q 2012	Milan	Apr 2012	$t_{4,4}$
Ballet	2Q 2012	Milan	May 2012	$t_{4,5}$
Ballet	2Q 2012	Milan	Jun 2012	$t_{4,6}$
...	...	...	...	

$r_4 =$

Fig. 6 T-relations and t-schemas for Example 5.

stored with a finer granularity (e.g., city). Similarly, the downward extension allows the relaxation of the answer with respect to the level of detail of the query. For instance, a query about products available in a given day may return the products available in that day's month. Both kinds of extensions meet needs that arise naturally in several application domains.

For this purpose, we introduce two new operators for the selection that leverage the available taxonomies; they can reference an h-domain that is more general or more specific than that occurring in its tuples.

Definition 10 (Upward selection) Let r be a t-relation over S, A be an attribute in S defined over l, and m be a member of l' with $l \leq_L l'$: the *upward selection* of r with respect to $A = m$ on level l, denoted by $\hat{\sigma}_{A:l=m}(r)$, is the t-relation over S defined as follows:

$$\hat{\sigma}_{A:l=m}(r) = \{t \in r \mid \textsc{lmap}_l^{l'}(t[A:l]) = m\}$$

Definition 11 (Downward selection) Let r be a t-relation over S, A be an attribute in S defined over l, and m be a member of l' with $l' \leq_L l$: the *downward selection* of r with respect to $A = m$ on level l, denoted by $\check{\sigma}_{A:l=m}(r)$, is the t-relation over S defined as follows:

$$\check{\sigma}_{A:l=m}(r) = \{t \in r \mid \textsc{lmap}_{l'}^{l}(m) = t[A:l]\}$$

In the following, we will often simply write $\hat{\sigma}_{A=m}$ and $\check{\sigma}_{A=m}$, without explicitly indicating the name of the level, when this is unambiguously determined by the corresponding attribute. Also, we will call these operators t-selections, to distinguish them from the standard selection operator.

Example 6 Consider again the t-relations r_1 and r_2 from Example 4. We have:

$$\hat{\sigma}_{\text{Location=Milan}}(r_1) = \{t_{1,1}, t_{1,3}\}$$
$$\check{\sigma}_{\text{Time=13/03/2012}}(r_2) = \{t_{2,1}\}.$$

It can be easily seen that these operators can be obtained by composing the upward or downward extension, the (standard) selection, and the projection operators, as shown in (1) and (2) below.

$$\hat{\sigma}_{A:l=m}(r) = \pi_S(\sigma_{A:l'=m}(\hat{\varepsilon}^{A:l'}_{A:l}(r))) \quad (1)$$

$$\check{\sigma}_{A:l=m}(r) = \pi_S(\sigma_{A:l'=m}(\check{\varepsilon}^{A:l}_{A:l'}(r))) \quad (2)$$

We now introduce two new join operators. Their main purpose is to combine information stored at different levels of granularity.

Definition 12 (Upward join) Let r_1 and r_2 be two t-relations over S_1 and S_2 respectively, and let S be an upper bound of a subset $\bar{S}_1$ of S_1 and a subset $\bar{S}_2$ of S_2. The *upward join* of r_1 and r_2 with respect to S on $\bar{S}_1$ and $\bar{S}_2$, denoted by $r_1 \hat{\bowtie}_{S:\bar{S}_1,\bar{S}_2} r_2$, is the t-relation over $S_1 \cup S_2$ defined as follows:

$$r_1 \hat{\bowtie}_{S:\bar{S}_1,\bar{S}_2} r_2 = \{\, t \mid \exists t_1 \in r_1, \exists t_2 \in r_2, \exists t' \text{over } S : t_1[\bar{S}_1] \leq_t t', t_2[\bar{S}_2] \leq_t t', t[S_1] = t_1, t[S_2] = t_2 \}$$

Definition 13 (Downward join) Let r_1 and r_2 be two t-relations over S_1 and S_2 respectively, and let S be a lower bound of a subset $\bar{S}_1$ of S_1 and a subset $\bar{S}_2$ of S_2. The *downward join* of r_1 and r_2 with respect to S on $\bar{S}_1$ and $\bar{S}_2$, denoted by $r_1 \check{\bowtie}_{S:\bar{S}_1,\bar{S}_2} r_2$, is the t-relation over $S_1 \cup S_2$ defined as follows:

$$r_1 \check{\bowtie}_{S:\bar{S}_1,\bar{S}_2} r_2 = \{\, t \mid \exists t_1 \in r_1, \exists t_2 \in r_2, \exists t' \text{over } S : t' \leq_t t_1[\bar{S}_1], t' \leq_t t_2[\bar{S}_2], t[S_1] = t_1, t[S_2] = t_2 \}$$

In the following, we will omit the indication of $\bar{S}_1$ and $\bar{S}_2$ when evident from the context. Also, we will call these operators t-joins, to distinguish them from the standard join operator.

Example 7 Consider the t-relation r_1 from Example 4 (Figure 5) and the t-relation r_5 shown in Figure 7. The result of $r_1 \hat{\bowtie}_{\mathsf{city}} r_5$ is the t-relation r_6, also shown in Figure 7. Now, consider the t-relations r_7 and r_8 shown in Figure 7. The result of $r_7 \check{\bowtie}_{\mathsf{theater,day}} r_8$ is the t-relation r_9 shown in Figure 7.

Also in this case, both the upward join and the downward join can be obtained by combining the upward extension or the downward extension, and the (standard) join. Equation (3) below shows this for the upward join, where $S = \{A^1 : l^1, \ldots, A^n : l^n\}$, and $S_i \supseteq \bar{S}_i \supseteq \{A^1 : l^1_i, \ldots, A^n : l^n_i\}$ for $i = 1, 2$.

$$r_1 \hat{\bowtie}_{S:\bar{S}_1,\bar{S}_2} r_2 = \pi_{S_1 S_2}(\hat{\varepsilon}^{S}_{\bar{S}_1}(r_1) \bowtie_{1.S=2.S} \hat{\varepsilon}^{S}_{\bar{S}_2}(r_2)) \quad (3)$$

Equation (4) below shows this for the downward join, where $S \supseteq \{A^1 : l^1, \ldots, A^n : l^n\}$, and $S_i \supseteq \bar{S}_i \supseteq \{A^1 : l^1_i, \ldots, A^n : l^n_i\}$ for $i = 1, 2$.

$$r_1 \check{\bowtie}_{S:\bar{S}_1,\bar{S}_2} r_2 = \pi_{S_1 S_2}(\check{\varepsilon}^{\bar{S}_1}_{S}(r_1) \bowtie_{1.S=2.S} \check{\varepsilon}^{\bar{S}_2}_{S}(r_2)) \quad (4)$$

Finally, we introduce two new operators for computing the difference between two t-relations whose information is stored at different levels of granularity.

Definition 14 (Upward difference) Let r_1 and r_2 be two t-relations over S_1 and S_2 respectively, and let S be an upper bound of S_1 and S_2. The *upward difference* of r_1 and r_2 with respect to S, denoted by $r_1 \hat{-}_S r_2$, is the t-relation over S_1 defined as follows:

$$r_1 \hat{-}_S r_2 = \{\, t \in r_1 \mid \exists t' \text{over } S, \neg\exists t'' \in r_2 : t \leq_t t', t'' \leq_t t' \}$$

Definition 15 (Downward difference) Let r_1 and r_2 be two t-relations over S_1 and S_2 respectively, and let S be a lower bound of S_1 and S_2. The *downward difference* of r_1 and r_2 with respect to S, denoted by $r_1 \check{-}_S r_2$, is the t-relation over S_1 defined as follows:

$$r_1 \check{-}_S r_2 = \{\, t \in r_1 \mid \exists t' \text{over } S, \neg\exists t'' \in r_2 : t' \leq_t t, t' \leq_t t'' \}$$

In this case, too, one can express these new operators by combining the upward or downward extension with standard operators (difference, join and projection). Equations (5) and (6) below show this for the upward difference and, respectively, downward difference.

$$r_1 \hat{-}_S r_2 = \pi_{S_1}\big((\hat{\varepsilon}^{S}_{S_1}(r_1)) \bowtie_{1.S=2.S} (\pi_S(\hat{\varepsilon}^{S}_{S_1}(r_1)) - \pi_S(\hat{\varepsilon}^{S}_{S_2}(r_2)))\big) \quad (5)$$

$$r_1 \check{-}_S r_2 = \pi_{S_1}\big((\check{\varepsilon}^{S_1}_{S}(r_1)) \bowtie_{1.S=2.S} (\pi_S(\check{\varepsilon}^{S_1}_{S}(r_1)) - \pi_S(\check{\varepsilon}^{S_2}_{S}(r_2)))\big) \quad (6)$$

We will call these operators t-differences, to distinguish them from the standard difference operator.

Example 8 Consider the t-relations r_{10} and r_{11} shown in Figure 8. After upward extension from theater to city, we have $\hat{\varepsilon}^{\mathsf{city}}_{\mathsf{theater}} r_{10} = r_{12}$ and $\hat{\varepsilon}^{\mathsf{city}}_{\mathsf{theater}} r_{11} = r_{13}$, where r_{12} and r_{13} are also shown in Figure 8. Therefore, the result of the upward difference $r_{10} \hat{-}_{\mathsf{city}} r_{11}$ is the t-relation r_{14} shown in Figure 8.

Note that the result of the upward difference r_{14} differs from the result of applying the standard difference to the upward extensions of the relations and then projecting out the attribute of the extension (city):

$$r_{14} = r_{10} \hat{-}_{\mathsf{city}} r_{11} \neq \pi_{\mathsf{theater}}(r_{12} - r_{13}) = r_{10}.$$

Albeit possible, extending the standard union operation to taxonomies seems less natural than in the cases described so far. Namely, the union of two t-relations r_1 and r_2 over the same attributes but at different levels of granularity would require fixing an arbitrary schema for the result (r_1's or r_2's schema or an upper or lower bound thereof). In turn, this would amount to having a result that includes tuples that did not exist in either r_1

$S_5 =$

Company:airline-company	Location:airport	
Alitalia	Linate	$t_{5,1}$
Air France	Roissy	$t_{5,2}$

$r_5 =$

$S_6 =$

Event:cultural-event	Author:artist	Time:day	Location:theater	Company:airline-company	Location:airport	
Romeo & Juliet	Prokofiev	24/04/2012	La Scala	Alitalia	Linate	$t_{6,1}$
Carmen	Bizet	24/05/2012	Opéra Garnier	Air France	Roissy	$t_{6,2}$
Requiem	Verdi	24/03/2012	La Scala	Alitalia	Linate	$t_{6,3}$
La bohème	Puccini	09/04/2012	Opéra Garnier	Air France	Roissy	$t_{6,4}$

$r_6 =$

$S_7 =$

Location:theater	Time:year	Price:money	
La Scala	2012	150	$t_{7,1}$

$r_7 =$

$S_8 =$

Location:theater	Time:month	Discount:perc.	
La Scala	03/2012	10%	$t_{8,1}$
La Scala	06/2012	20%	$t_{8,2}$

$r_8 =$

$S_9 =$

Location:theater	Time:year	Price:money	Time:month	Discount:perc.	
La Scala	2012	150	03/2012	10%	$t_{9,1}$
La Scala	2012	150	06/2012	20%	$t_{9,2}$

$r_9 =$

Fig. 7 T-relations and t-schemas for Example 7.

$S_{10} =$

Location:theater	
Arcimboldi	$t_{10,1}$
Massimo	$t_{10,2}$
Opéra Bastille	$t_{10,3}$

$r_{10} =$

$S_{10} =$

Location:theater	
La Scala	$t_{11,1}$
Opéra Garnier	$t_{11,2}$

$r_{11} =$

$S_{11} =$

Location:theater	Location:city	
Arcimboldi	Milan	$t_{12,1}$
Massimo	Palermo	$t_{12,2}$
Opéra Bastille	Paris	$t_{12,3}$

$\hat{\varepsilon}^{\mathsf{city}}_{\mathsf{theater}}(r_{10}) = r_{12} =$

$S_{11} =$

Location:theater	Location:city	
La Scala	Milan	$t_{13,1}$
Opéra Garnier	Paris	$t_{13,2}$
		$t_{13,3}$

$\hat{\varepsilon}^{\mathsf{city}}_{\mathsf{theater}}(r_{11}) = r_{13} =$

$S_{10} =$

Location:theater	
Massimo	$t_{14,1}$

$r_{10} \hat{-}_{\mathsf{city}} r_{11} = r_{14} =$

Fig. 8 T-relations and t-schemas for Example 8.

or r_2, which seems less desirable. Similarly, taxonomical versions of projection and renaming do not seem particularly meaningful.

As in the standard relational algebra, it is possible to build complex expressions combining several TRA operators thanks to the fact that TRA is closed, i.e., the result of every application of an operator is a t-relation. Formally, one can define and build the expressions of TRA, called t-expressions, by assuming that t-relations themselves are t-expressions, and by substituting the t-relations appearing in Definitions 8-15 with a t-expression.

Similar extensions are possible for other RA operators and combinations thereof. As an example, with the same conventions adopted for the upward and downward join, we mention the *upward semijoin* $r_1 \hat{\ltimes}_{S:\bar{S}_1,\bar{S}_2} r_2$ and the *downward semijoin* $r_1 \check{\ltimes}_{S:\bar{S}_1,\bar{S}_2} r_2$, defined in Equations (7) and (8), respectively:

$$r_1 \hat{\ltimes}_{S:\bar{S}_1,\bar{S}_2} r_2 = \pi_{S_1}(r_1 \hat{\bowtie}_{S:\bar{S}_1,\bar{S}_2} r_2) \tag{7}$$

$$r_1 \check{\ltimes}_{S:\bar{S}_1,\bar{S}_2} r_2 = \pi_{S_1}(r_1 \check{\bowtie}_{S:\bar{S}_1,\bar{S}_2} r_2) \tag{8}$$

From these, one can, e.g., define the *upward antijoin* $r_1 \hat{\triangleright}_{S:\bar{S}_1,\bar{S}_2} r_2$ and the *downward antijoin* $r_1 \check{\triangleright}_{S:\bar{S}_1,\bar{S}_2} r_2$ as shown in Equations (9) and (10), respectively:

$$r_1 \hat{\triangleright}_{S:\bar{S}_1,\bar{S}_2} r_2 = r_1 - r_1 \hat{\ltimes}_{S:\bar{S}_1,\bar{S}_2} r_2 \tag{9}$$

$$r_1 \check{\triangleright}_{S:\bar{S}_1,\bar{S}_2} r_2 = r_1 - r_1 \check{\ltimes}_{S:\bar{S}_1,\bar{S}_2} r_2 \tag{10}$$

$S_2 =$

Title:event	Time:quarter	Location:city
Sport	3Q 2012	Rome

$r_{15} =$ $t_{15,1}$

Fig. 9 A t-relation for Example 9.

Example 9 Consider t-relations r_2 from Example 4 (Figure 5) and r_5 from Example 7 (Figure 7). Their upward antijoin $r_2 \hat{\triangleright}_{\mathsf{city}} r_5$ is the t-relation r_{15} shown in Figure 9: Similarly, the downward antijoin $r_5 \check{\triangleright}_{\mathsf{airport}} r_2$ is an empty t-relation with schema S_5.

4.2 Query equivalence in TRA

One of the main benefits of Relational Algebra is the use of algebraic properties for query optimization. In particular, equivalences allow transforming a relational expression into an equivalent expression in which the average size of the relations yielded by subexpressions is smaller. Rewritings may be used, e.g., to break up an application of an operator into several, smaller applications, or to move operators to more convenient places in the expression (e.g., pushing selection and projection through join). In analogy with the standard case, we are now going to describe a collection of new equivalences that can be used for query optimization in Taxonomy-based Relational Algebra.

In the remainder of this section, we shall use, together with possible subscripts and primes, the letter r to denote a t-relation, l for a level, A for a set of attributes, and P for a (selection or join) predicate.

4.2.1 Upward and downward extension

Border cases
Let l be the level of an attribute in r. Then:

$$\hat{\varepsilon}^{l}_{l}(r) = \check{\varepsilon}^{l}_{l}(r) = r \tag{11}$$

Equivalence (11) shows that if the upper and lower level of an extension coincide, then the extension is idle, both for the upward and for the downward case. The proof of (11) follows immediately from Definitions 8 and 9.

Idempotency
Let l be the level of an attribute in r such that $l \leq_L l'$ and $l'' \leq_L l$. Then:

$$\hat{\varepsilon}^{l'}_{l}(\hat{\varepsilon}^{l'}_{l}(r)) = \hat{\varepsilon}^{l'}_{l}(r) \tag{12}$$

$$\check{\varepsilon}^{l}_{l''}(\check{\varepsilon}^{l}_{l''}(r)) = \check{\varepsilon}^{l}_{l''}(r) \tag{13}$$

Equivalences (12) and (13) state that repeated applications of the same extension are idle, both for the upward and for the downward case. Here, too, the proof follows immediately from Definitions 8 and 9.

Duality
Let l be the level of an attribute in r such that $l' \leq_L l$. Then:

$$\hat{\varepsilon}^{l}_{l'}(\check{\varepsilon}^{l}_{l'}(r)) = \check{\varepsilon}^{l}_{l'}(r) \tag{14}$$

The above Equivalence (14) shows that an upward extension is always idle after a downward extension on the same levels. To prove (14), it suffices to consider that the mapping from members of a lower level to members of an upper level is many-to-one, so no new tuple can be generated by the upward extension. Note, however, that the downward extension after an upward extension on the same levels is generally not redundant, since the mapping from members of an upper level to members of a lower level is one-to-many.

Commutativity
Let l_1, l_2 be levels of attributes of r, s.t. $l_i \leq_L l'_i$ and $l''_i \leq_L l_i$, for $i = 1, 2$. Then:

$$\hat{\varepsilon}^{l'_2}_{l_2}(\hat{\varepsilon}^{l'_1}_{l_1}(r)) = \hat{\varepsilon}^{l'_1}_{l_1}(\hat{\varepsilon}^{l'_2}_{l_2}(r)) \tag{15}$$

$$\check{\varepsilon}^{l_2}_{l''_2}(\check{\varepsilon}^{l_1}_{l''_1}(r)) = \check{\varepsilon}^{l_1}_{l''_1}(\check{\varepsilon}^{l_2}_{l''_2}(r)) \tag{16}$$

The above Equivalences (15) and (16) state that two extensions of the same kind can be swapped. Both follow straightforwardly from Definitions 8 and 9.

Interplay with standard projection
Let l be the level of an attribute A in a relation r over S s.t. $l \leq_L l'_1 \leq_L l'_2$ and $l_2 \leq_L l_1 \leq_L l$, and let $A_p \subseteq S$ s.t. $A_p \not\ni A : l_1$ and $A_p \not\ni A : l'_1$. Then:

$$\pi_{A_p} \hat{\varepsilon}^{A:l'_2}_{A:l}(r) = \pi_{A_p} \hat{\varepsilon}^{A:l'_2}_{A:l'_1}(\hat{\varepsilon}^{A:l'_1}_{A:l}(r)) \tag{17}$$

$$\pi_{A_p} \check{\varepsilon}^{A:l}_{A:l_2}(r) = \pi_{A_p} \check{\varepsilon}^{A:l_1}_{A:l_2}(\check{\varepsilon}^{A:l}_{A:l_1}(r)) \tag{18}$$

Note that the outer π_{A_p} in Equivalence (17) is necessary, because, in case $l \neq l'_1 \neq l'_2$, the left-hand sides of the equivalences would be t-relations that do not include the attribute-level pair $A : l'_1$, whereas the right-hand sides would; therefore, projecting away $A : l'_1$ is essential. Similarly for Equivalence (18).

Let l be the level of an attribute A in a relation r over S s.t. $l \leq_L l'$ and $l'' \leq_L l$, and $A_p \subseteq S$ s.t. $A_p \not\ni A : l'$ and $A_p \not\ni A : l''$. Then:

$$\pi_{A_p}(\hat{\varepsilon}^{A:l'}_{A:l}(r)) = \hat{\varepsilon}^{A:l'}_{A:l}(\pi_{A_p}(r)) \tag{19}$$

$$\pi_{A_p}(\check{\varepsilon}^{A:l}_{A:l''}(r)) = \check{\varepsilon}^{A:l}_{A:l''}(\pi_{A_p}(r)) \tag{20}$$

Equivalences (19) and (20) show that, similarly to Equivalences (17) and (18), it is also possible to swap extension and standard projection provided that the projection does not retain the added attribute.

Interplay with standard selection
Let l be the level of an attribute A in r s.t. $l \leq_L l'$ and $l'' \leq_L l$, and P be a selection predicate that does not refer either to $A : l'$ or $A : l''$. Then:

$$\sigma_P(\hat{\varepsilon}^{A:l'}_{A:l}(r)) = \hat{\varepsilon}^{A:l'}_{A:l}(\sigma_P(r)) \tag{21}$$
$$\sigma_P(\check{\varepsilon}^{A:l}_{A:l''}(r)) = \check{\varepsilon}^{A:l}_{A:l''}(\sigma_P(r)) \tag{22}$$

Equivalences (21) and (22) show swapping of extension and standard selection, when the added attribute-level pair is immaterial to the selection predicate.

Interplay with standard join
Let l be the level of an attribute A in r_1 but not r_2 s.t. $l \leq_L l'$, $l'' \leq_L l$, and P be a join predicate that does not refer either to $A : l'$ or $A : l''$. Then:

$$\hat{\varepsilon}^{A:l'}_{A:l}(r_1 \bowtie_P r_2) = (\hat{\varepsilon}^{A:l'}_{A:l}(r_1)) \bowtie_P r_2 \tag{23}$$
$$\check{\varepsilon}^{A:l}_{A:l''}(r_1 \bowtie_P r_2) = (\check{\varepsilon}^{A:l}_{A:l''}(r_1)) \bowtie_P r_2 \tag{24}$$

Equivalences (23) and (24) show that extension can be "pushed" through standard join. (Note that, if $A : l$ was in the schema of both r_1 and r_2, the extension should be "pushed" through both sides of the join.)

Interplay with standard difference
Let l be the level of an attribute of r_1 and r_2, s.t. $l \leq_L l'$ and $l'' \leq_L l$. Then:

$$\hat{\varepsilon}^{l'}_{l}(r_1) - \hat{\varepsilon}^{l'}_{l}(r_2) = \hat{\varepsilon}^{l'}_{l}(r_1 - r_2) \tag{25}$$
$$\check{\varepsilon}^{l}_{l''}(r_1) - \check{\varepsilon}^{l}_{l''}(r_2) = \check{\varepsilon}^{l}_{l''}(r_1 - r_2) \tag{26}$$

The above Equivalences (25) and (26) state that extension can be "pushed" through standard difference, too. To prove these equivalences, it suffices to observe that upward extension *adds* an attribute, and that the single value (upward case) or set of values (downward case) added for that attribute functionally depends on the value at level l.

4.2.2 Upward and downward selection

Idempotency
Let l be the level of an attribute A of r s.t. $l \leq_L l'$ and $l'' \leq_L l$, where l' is the level of m' and l'' of m''. Then:

$$\hat{\sigma}_{A:l=m'}(\hat{\sigma}_{A:l=m'}(r)) = \hat{\sigma}_{A:l=m'}(r) \tag{27}$$
$$\check{\sigma}_{A:l=m''}(\check{\sigma}_{A:l=m''}(r)) = \check{\sigma}_{A:l=m''}(r) \tag{28}$$

Equivalences (27) and (28) state that repeated applications of the same t-selection are idle, both for the upward and for the downward case. To prove (27), consider that, by (1), the left-hand side of the equivalence can be written as:

$$\pi_S(\sigma_{A:l'=m'}(\hat{\varepsilon}^{A:l'}_{A:l}(\pi_S(\sigma_{A:l'=m'}(\hat{\varepsilon}^{A:l'}_{A:l}(r))))))$$

where S is the schema of r. The innermost π_S can be moved outside the upward selection by using equivalence (19) if $l \neq l'$ or equivalence (11) if $l = l'$. By using standard properties of the relational operators, the innermost π_S can also be moved outside the outermost selection, and eliminated by idempotency:

$$\pi_S(\sigma_{A:l'=m'}(\hat{\varepsilon}^{A:l'}_{A:l}(\sigma_{A:l'=m'}(\hat{\varepsilon}^{A:l'}_{A:l}(r)))))$$

Now, equivalence (21) allows swapping selection and upward extension provided that the selection predicate does not refer to the attribute-level pair introduced by $\hat{\varepsilon}$. This condition is only required to make sure that, after the swap, the selection refers to an existing attribute-level pair. Therefore, equivalence (21) can be used here to move the innermost selection outside the outermost $\hat{\varepsilon}$, although $A : l' = m'$ is a predicate that clearly refers to $A : l'$ (l' being the level of m'), since $A : l'$ is already introduced by the innermost $\hat{\varepsilon}$. By idempotency of both standard selection and upward extension (as of equivalence (12)), we obtain:

$$\pi_S(\sigma_{A:l'=m'}(\hat{\varepsilon}^{A:l'}_{A:l}(r)))$$

which, by (1), corresponds to the right-hand side of (27). Analogously for (28).

Commutativity

$$\hat{\sigma}_{l_2:A_2=m_2}(\hat{\sigma}_{l_1:A_1=m_1}(r)) = \hat{\sigma}_{l_1:A_1=m_1}(\hat{\sigma}_{l_2:A_2=m_2}(r)) \tag{29}$$
$$\check{\sigma}_{l_2:A_2=m_2}(\check{\sigma}_{l_1:A_1=m_1}(r)) = \check{\sigma}_{l_1:A_1=m_1}(\check{\sigma}_{l_2:A_2=m_2}(r)) \tag{30}$$
$$\check{\sigma}_{l_2:A_2=m_2}(\hat{\sigma}_{l_1:A_1=m_1}(r)) = \hat{\sigma}_{l_1:A_1=m_1}(\check{\sigma}_{l_2:A_2=m_2}(r)) \tag{31}$$

The above equivalences state that t-selection is commutative, both for the upward and the downward case. Moreover, an upward selection can be swapped with a downward selection (and vice versa), as shown in equivalence (31). The proof of these follows straightforwardly from commutativity of standard selection and interplay of extension and standard selection.

4.2.3 Upward and downward join

Relationship between upward and downward join
The first equivalence easily follows from the definitions and refers to a special but important case in which the join is over t-relations whose schemas are ordered. Let r_1 and r_2 be a pair of t-relations defined over S_1 and S_2, respectively. If $S_1 \leq_S S_2$ then:

$$r_1 \check{\bowtie}_{S_1} r_2 = r_1 \hat{\bowtie}_{S_2} r_2 \tag{32}$$

Pushing upward and downward selection through upward and downward join
Let $A : l$ be in the schema S_1 of r_1 but not in the schema S_2 of r_2, and C_{low} and C_{up} be a lower and an upper bound of $C_1 \subseteq S_1$ and $C_2 \subseteq S_2$. Then:

$$\hat{\sigma}_{A:l=m}(r_1 \hat{\bowtie}_{C_{up}:C_1,C_2} r_2) = (\hat{\sigma}_{A:l=m} r_1) \hat{\bowtie}_{C_{up}:C_1,C_2} r_2 \tag{33}$$

$$\hat{\sigma}_{A:l=m}(r_1 \check{\bowtie}_{C_{low}:C_1,C_2} r_2) = (\hat{\sigma}_{A:l=m} r_1) \check{\bowtie}_{C_{low}:C_1,C_2} r_2 \tag{34}$$

$$\check{\sigma}_{A:l=m}(r_1 \hat{\bowtie}_{C_{up}:C_1,C_2} r_2) = (\check{\sigma}_{A:l=m} r_1) \hat{\bowtie}_{C_{up}:C_1,C_2} r_2 \tag{35}$$

$$\check{\sigma}_{A:l=m}(r_1 \check{\bowtie}_{C_{low}:C_1,C_2} r_2) = (\check{\sigma}_{A:l=m} r_1) \check{\bowtie}_{C_{low}:C_1,C_2} r_2 \tag{36}$$

The above equivalences (33)-(36) indicate that a t-selection can be "pushed" through a t-join on the side that involves the attribute-level pair used in the selection. To prove the equivalences, it suffices to use (1)-(4) and the properties of standard operators.

Pushing standard projection through upward and downward join
Let r_i be a t-relation over S_i for $i = 1, 2$, C_{low} and C_{up} be a lower and an upper bound of $C_1 \subseteq S_1$ and $C_2 \subseteq S_2$, L be a subset of $S_1 \cup S_2$, and $L_i = C_i \cup (L \setminus S_i)$ for $i = 1, 2$. Then:

$$\pi_L(r_1 \hat{\bowtie}_{C_{up}:C_1,C_2} r_2) = \pi_L((\pi_{L_1} r_1) \hat{\bowtie}_{C_{up}:C_1,C_2} (\pi_{L_2} r_2)) \tag{37}$$

$$\pi_L(r_1 \check{\bowtie}_{C_{low}:C_1,C_2} r_2) = \pi_L((\pi_{L_1} r_1) \check{\bowtie}_{C_{low}:C_1,C_2} (\pi_{L_2} r_2)) \tag{38}$$

Equivalences (37) and (38) show how standard projection can be "pushed" through an upward or downward join to both sides of the join by properly breaking up the projection attributes into smaller sets. Again, the equivalences follow immediately by applying (3) and (4) together with the standard "push" of projection through join and through extension (as of equivalences (19) and (20)).

From the above discussion, we have the following correctness result.

Theorem 1 *Equivalences (11)-(38) hold for any possible t-relation.*

Theorem 1 together with the fact that TRA is closed entails that equivalences (11)-(38) can also be used to test equivalence of complex t-expressions.

Preservation of partial order
Finally, we observe that some applications of the TRA operators preserve partial order between relations.

$$\text{If } r_1 \leq_r r_2 \text{ then } (\hat{\varepsilon}^{l'}_{l}(r_1)) \leq_r r_2 \tag{39}$$

$$\text{If } r_1 \leq_r r_2 \text{ then } (\check{\varepsilon}^{l}_{l'}(r_1)) \leq_r r_2 \tag{40}$$

$$\text{If } r_1 \leq_r r_2 \text{ then } r_1 \leq_r (\hat{\varepsilon}^{l'}_{l}(r_2)) \tag{41}$$

The implications (39)–(41) show that a partial order $r_1 \leq_r r_2$ is preserved both if r_1 is (upward or downward) extended and if r_2 is upward extended. Note that downward extending r_2, instead, may not preserve the order.

5 A Logical Query Language

In this section we present HDRC (H-Domain Relational Calculus) an extension of the Domain Relational Calculus (DRC) over t-relations. This language provides a basis for a declarative query language over relational databases, similar to SQL, that exploits taxonomies defined on data domains and allows the relaxation of query answering. Moreover, the comparison between the logic language and the algebraic one allows us to better understand their strengths and weaknesses in terms of expressive power and finiteness of query answers.

5.1 HDRC by examples

Intuitively, a *t-query* is a function from t-relations over a set of input t-schemas to a t-relation over an output t-schema, where the input and output t-schemas are defined over the same taxonomy.

In Section 4 we have shown how a t-query can be expressed in TRA , a procedural language. Conversely, an HDRC a t-query is specified by means of a declarative expression of the following form:

$$\{A_1 : x_1, \ldots, A_n : x_n \mid \psi(x_1, \ldots, x_n)\}$$

where $A_1 : x_1, \ldots, A_n : x_n$ is called the *target list*, $A_1, \ldots, A_n$ are distinct attribute names, $x_1, \ldots, x_n$ are distinct *variables*, and $\psi(x, x_1, \ldots, x_n)$ is a first-order formula in which $x_1, \ldots, x_n$ are the only free variables. As it happens in DRC, the formula ψ is composed by

t-relations and atoms comparing variables with either variables or constants. In addition, taxonomies between values of data domains are taken into account by allowing in ψ equality atoms that involve level mappings. As for DRC, the result of a HDRC query is the set of t-tuples $c_1, \ldots, c_n$ that, respectively substituted to $x_1, \ldots, x_n$, satisfy the formula ψ.

Example 10 Let us consider the t-relations r_1 and r_2 of Example 4, reported also in Figure 10 for convenience. The following query q_1^{HDRC} involves r_1 and retrieves information about cultural events located in Milan.

$$q_1^{\text{HDRC}} = \{\ \mathit{Title} : x_1, \mathit{Author} : x_2, \mathit{Time} : x_3, \mathit{Location} : x_4 \mid \exists x_3(r_1(\mathit{Title} : x_1, \mathit{Author} : x_2, \mathit{Time} : x_3, \mathit{Location} : x_4) \wedge \text{LMAP}^{\mathsf{city}}_{\mathsf{theater}}(x_4) = \text{Milan})\ \}$$

Query q_1^{HDRC} returns $\{t_{1,1}, t_{1,3}\}$, the same result as the TRA query

$$\hat{\sigma}_{\text{Location}=\texttt{Milan}}(r_1)$$

from Example 6.

As another example, the following query q_2^{HDRC} involves the t-relation r_2 and retrieves the events located in Milan on the same quarter as day 13/03/2012.

$$q_2^{\text{HDRC}} = \{\ \mathit{Title} : x_1, \mathit{Location} : x_3 \mid \exists x_2(r_2(\mathit{Title} : x_1, \mathit{Time} : x_2, \mathit{Location} : x_3) \wedge x_2 = \text{LMAP}^{\mathsf{quarter}}_{\mathsf{day}}(13/03/2012))\ \}$$

The result is the same as that of the TRA query

$$\check{\sigma}_{\text{Time}=\texttt{13/03/2012}}(r_2)$$

from Example 6, but without including the *Time* attribute.

Finally, the following query q_3^{HDRC} involves, again, the t-relation r_1 and the t-relation r_5 of Example 7, reported here for convenience:

$S_5 =$

Company:airline-company	Location:airport	
Alitalia	Linate	$t_{5,1}$
Air France	Roissy	$t_{5,2}$

$r_5 =$

$$q_3^{\text{HDRC}} = \{\ \mathit{Event} : x_1, \mathit{With} : x_5 \mid \exists x_2, x_3, x_4, x_6(r_1(\mathit{Title} : x_1, \mathit{Author} : x_2, \mathit{Time} : x_3, \mathit{Location} : x_4) \wedge r_5(\mathit{Company} : x_5, \mathit{Location} : x_6) \wedge \text{LMAP}^{\mathsf{city}}_{\mathsf{theater}}(x_4) = \text{LMAP}^{\mathsf{city}}_{\mathsf{airport}}(x_6))\ \}$$

This query combines cultural events with airlines that have flights to the same city as the events and is similar in spirit to query $r_1 \hat{\bowtie}_{\mathsf{city}} r_5$ from Example 7 (its result only includes the *Title* and *Company* attributes, which are renamed as *Event* and *With*, respectively).

5.2 HDRC: syntax and semantics

We now formally introduce the language HDRC.

Let us fix a set of t-schemas $\mathbf{S}$ for a taxonomy T. For each h-domain $h = \{L, \leq_L, \text{LM}\}$ in T and for each level l in L, we assume the existence of a countable set of *variables of type* l. The *terms* and their respective types are inductively defined as follows.

- A variable of type l is a term of type l;
- a value in $M(l)$ (the members of l) is a term of type l;
- if t is a term of type l' and $l' \leq_L l$, then $\text{LMAP}^l_{l'}(t)$ is a term of type l;
- nothing else is a term of type l.

Atoms are defined as follows:

- if t and t' are terms of the same type then $t = t'$ is an atom,
- if r is a t-relation over a t-schema $S = \{A_1 : l_1, \ldots, A_n : l_n\} \in \mathbf{S}$ and $x_1, \ldots, x_n$ are variables of type $l_1, \ldots, l_n$, respectively, then $r(A_1 : x_1, \ldots, A_n : x_n)$ is an atom;
- nothing else is an atom.

Finally, the *formulas* are defined as follows.

- An atom is a formula in which all variables are free;
- if ψ_1 and ψ_2 are formulas, then $(\psi_1) \wedge (\psi_2)$, $(\psi_1) \vee (\psi_2)$, and $\neg(\psi_1)$ are formulas (where parentheses are omitted when no ambiguity may arise); each variable is free (bound) in them if it is free (bound) in the subformula where it appears;
- if ψ is a formula and x is a variable, then $\exists x(\psi)$ and $\forall x(\psi)$ are formulas; the variable x is bound in them, any other variable is free (bound) if it is free (bound) in the subformula where it appears;
- nothing else is a formula.

A HDRC *query* over $\mathbf{S}$ is an expression of the form

$$\{A_1 : x_1, \ldots, A_n : x_n \mid \psi(x_1, \ldots, x_n)\}$$

where $\psi(x_1, \ldots, x_n)$ is a formula having $x_1, \ldots, x_n$ as distinct free variables. The expression $A_1 : x_1, \ldots, A_n : x_n$ is called the *target list*.

The *result* of a HDRC query q of the above form with respect to a t-database over $\mathbf{S}$ is the set of t-tuples $c_1, \ldots, c_n$ that, respectively substituted to $x_1, \ldots, x_n$, satisfy the formula ψ. The notion of *satisfaction* of a formula with respect to a substitution s and a set of t-relations is defined in the usual way, with the only observation that variables must vary over values of the corresponding types.

$S_1 =$ / $r_1 =$

Title:cultural-event	Author:artist	Time:day	Location:theater	
Romeo & Juliet	Prokofiev	13/04/2012	La Scala	$t_{1,1}$
Carmen	Bizet	24/05/2012	Opéra Garnier	$t_{1,2}$
Requiem	Verdi	28/03/2012	La Scala	$t_{1,3}$
La bohème	Puccini	09/04/2012	Opéra Garnier	$t_{1,4}$

$S_2 =$ / $r_2 =$

Title:event	Time:quarter	Location:city	
Concert	1Q 2012	Milan	$t_{2,1}$
Ballet	2Q 2012	Milan	$t_{2,2}$
Sport	3Q 2012	Rome	$t_{2,3}$
Opera	2Q 2012	Paris	$t_{2,4}$

Fig. 10 T-relations and t-schemas for Example 10.

5.3 Taxonomy independence and expressive power

As in traditional Domain Relational Calculus, there are HDRC expressions involving negation that depend on the domain.

Example 11 The result of the expression q_{dep}^{HDRC}, shown below, defined over the input t-schema:

$$S_1 = \{ Title : \mathsf{culturalEvent}, Author : \mathsf{artist}, Time : \mathsf{day}, Location : \mathsf{theater}\}$$

is a t-relation with one tuple for each of the members of the culturalEvent level of the Event h-domain that does not occur in Milan, according to the actual content of the queried t-database.

$$q_{dep}^{\text{HDRC}} = \{\ Event : x_1 \mid \neg(\exists x_2(\exists x_3(\exists x_4(r_1(Title : x_1, Author : x_2, Time : x_3, Location : x_4) \wedge \text{LMAP}_{\mathsf{theater}}^{\mathsf{city}}(x_4) = \text{Milan})))) \}$$

It is well known that this property is highly undesirable since, if the domain changes without affecting the database, the result may change. Moreover, and even worse, if the domain is infinite, the result may be an infinite set of t-tuples. Thus, since t-queries are defined as functions on the set of t-relations, it follows that the expression q_{dep}^{HDRC} from Example 11 defines a different t-query for each different domain.

Let us then introduce a notion of domain independence in our framework. We say that a taxonomy T is *compatible* with a t-database d and an expression E of a query language L if: (i) d is a t-database for T and (ii) T includes all the values occurring in E. We then denote by $E_T(d)$ the application of an expression E of L to a t-database d for a taxonomy T.

Definition 16 (H-domain independence) We say that an expression E of a query language L is *h-domain independent* if for any t-database d and for any pair of taxonomies T and T' compatible with d and E, $E_T(d) = E_{T'}(d)$. A language is h-domain independent if all its expressions are h-domain independent.

The expression q_{dep}^{HDRC} from Example 11 shows that HDRC is not h-domain independent. Unfortunately, differently from traditional Relational Algebra, which is a domain-independent query language, it turns out that TRA is not a h-domain independent language either.

Example 12 Let us consider t-relation r_1 of Example 4 and assume that the theater "La Scala", located in Milan, is moved to Venice. Then, the result of the expression $\hat{\sigma}_{\mathsf{City=Milan}}(r_1)$ would change even if the actual content of r_1 does not change.

Actually, this behavior is not surprising: it depends on the fact that the upward and downward extension operators generate new values, not occurring in the original t-database d. However, it turns out that, in a significant number of cases, this situation is somehow under control since the answer to a query depends exclusively on a small set of values outside d: those that can be obtained by applying a bounded number of times the level mappings of the taxonomy on which d is defined. We will make this more precise by replacing the above notion of h-domain independence by a new notion of taxonomy independence. Some preliminary concepts are needed.

Definition 17 (Induced mapping) Let T be a taxonomy and let C be a set of values taken from T. The *mapping* $\text{LM}_T(C)$ *induced by* T *on* C is defined as the following set of pairs:
$\text{LM}_T(C) = \{v_1{:}l_1 \mapsto v_2{:}l_2 \mid \exists v_1 \in C, \exists \text{LMAP}_{l_1}^{l_2}$ *in* T : $\text{LMAP}_{l_1}^{l_2}(v_1) = v_2\}$

Note that the induced mapping obtained as in Definition 17 refers to all possible mappings between any two levels l_1 and l_2 in the taxonomy such that $l_1 \leq_L l_2$.

Moreover, since in our data model a taxonomy is composed by a finite number of h-domains each of which is organized into a finite number of levels (see Definition 1), as long as C is finite, the induced mapping is also a finite set, even if the h-domains are populated by infinite members.

Following common database practice, we call *active domain* of a t-database d for a taxonomy T the set of values in T that appear in some t-tuple of some t-relation of d. Moreover, given a t-database d and an expression E, we call the active domain of d and E, denoted by $\text{ADOM}(d, E)$, the union of the active domain of d and the set of values that appear in E.

Now we want to define an expression E to be *taxonomy-independent* if it depends only on the mapping induced by T on the values occurring in $\text{ADOM}(d, E)$. A technical notion is needed.

Definition 18 (Agreement) Given a t-database d, an expression E, and a pair of taxonomies T and T', we say that T *agrees with* T' *on* d *and* E if: *(i)* T and T' are compatible with d and E, and *(ii)* $\text{LM}_T(\text{ADOM}(d, E)) = \text{LM}_{T'}(\text{ADOM}(d, E))$.

Example 13 Consider two taxonomies T and T', both with a single h-domain on two levels l_1 and l_2 such that $l_1 \leq_L l_2$ and with

$$M(l_1) = \{1, 2, 3\}$$
$$M(l_2) = \{a, b, c\}.$$

In T the level mappings are such that

$$\text{LMAP}_{l_1}^{l_2}(1) = a$$
$$\text{LMAP}_{l_1}^{l_2}(2) = b$$
$$\text{LMAP}_{l_1}^{l_2}(3) = c,$$

whereas in T' we have

$$\text{LMAP}_{l_1}^{l_2}(1) = a$$
$$\text{LMAP}_{l_1}^{l_2}(2) = c$$
$$\text{LMAP}_{l_1}^{l_2}(3) = b.$$

Consider the two databases d and d' consisting of a single t-relation r over a schema $S = A : l_1$ shown below:

$$d = \left\{ r = \begin{array}{|c|}\hline A : l_1 \\ \hline 1 \\ \hline \end{array} \right\} \qquad d' = \left\{ r = \begin{array}{|c|}\hline A : l_1 \\ \hline 2 \\ \hline \end{array} \right\}$$

Finally, consider the expression $E = \hat{\varepsilon}_{A:l_1}^{A:l_2}(r)$. T and T' agree on d and E, since *(i)* T and T' are compatible with d and E, and *(ii)* the mapping induced by T on $\text{ADOM}(d, E)$ (note that E involves no constant) is $\{1{:}l_1 \mapsto a{:}l_2\}$ and coincides with the mapping induced by T'. However, T and T' do not agree on d' and E, since the mapping induced by T on $\text{ADOM}(d, E)$ is $\{2{:}l_1 \mapsto b{:}l_2\}$, while the mapping induced by T' is $\{2{:}l_1 \mapsto c{:}l_2\}$.

We are now ready to introduce our notion of independence of the taxonomy.

Definition 19 (Taxonomy independence) An expression E of a query language L is *taxonomy-independent* if, for any t-database d and for any pair of taxonomies T and T' that agree on d and E, we have $E_T(d) = E_{T'}(d)$. A language is taxonomy-independent if all its expressions are taxonomy-independent.

Yet, we have a negative result for TRA.

Lemma 1 TRA *is not taxonomy-independent.*

Proof Consider a taxonomy T consisting of just one h-domain h with two levels l_1 and l_2 such that: $l_1 \leq_L l_2$, $M(l_1) = \{1, 2\}$, $M(l_2) = \{a, b\}$, and $\text{LMAP}_{l_1}^{l_2}$ maps 1 to a and 2 to b. Now consider a t-database d consisting of a single t-relation r over a t-schema $S = A : l_2$ as shown below:

$$r = \begin{array}{|c|}\hline A : l_2 \\ \hline a \\ b \\ \hline \end{array}$$

Then the expression $E = \check{\varepsilon}_{A:l_1}^{A:l_2}(r)$ is not taxonomy-independent. To show this it is sufficient to consider another taxonomy T' that is identical to T, except for the fact that $\text{LMAP}_{l_1}^{l_2}$ maps 1 to b and 2 to a. Then, T and T' agree on d and E (since there is no level l in h such that $l_2 \leq_L l$) but $E_T(d) = \begin{array}{|cc|}\hline A : l_2 & A : l_1 \\ \hline a & 1 \\ b & 2 \\ \hline \end{array}$

$$\neq \begin{array}{|cc|}\hline A : l_2 & A : l_1 \\ \hline a & 2 \\ b & 1 \\ \hline \end{array} = E_{T'}(d).$$

The consequences of the lack of taxonomy independence become even more dramatic if, in the example of the proof of Lemma 1, the lower level l_1 has infinitely many members, each of which maps to either a or b (e.g., the members of l_1 are positive integers with the even numbers mapping to a and the odd numbers mapping to b) since, in this case, the result of E would be an infinite number of tuples.

The proof above shows that the downward extension can make an expression dependent of the taxonomy. This is indeed true also for the downward join and the downward difference, but not for the downward selection and for all the upward versions of the taxonomic operators. In fact, we have the following positive result that precisely defines the safe portion of TRA.

Lemma 2 $\text{TRA}^- = \text{TRA} - \{\check{\varepsilon}, \check{\bowtie}, \check{-}\}$ *is the maximal subset of* TRA *that is taxonomy-independent.*

Proof In the proof of Lemma 1 we showed that there are expressions using the downward extension operator $\check{\varepsilon}$ that are not taxonomy-independent. A similar argument can be used for the downward join $\check{\bowtie}$ and the downward difference $\check{-}$. In particular, consider two taxonomies T and T' over a single h-domain h with levels l_1, l_2, and l_3 such that $l_3 \leq_L l_1$ and $l_3 \leq_L l_2$, with $M(l_1) = \{a_1, b_1\}$, $M(l_2) = \{a_2, b_2\}$, and $M(l_3) = \{a_3, b_3\}$. In T, we have $\text{LMAP}_{l_3}^{l_1}(a_3) = a_1$, $\text{LMAP}_{l_3}^{l_2}(a_3) = a_2$, $\text{LMAP}_{l_3}^{l_1}(b_3) = b_1$, and $\text{LMAP}_{l_3}^{l_2}(b_3) = b_2$; in T', we have $\text{LMAP}_{l_3}^{l_1}(a_3) = a_1$, $\text{LMAP}_{l_3}^{l_2}(a_3) = b_2$, $\text{LMAP}_{l_3}^{l_1}(b_3) = b_1$, and $\text{LMAP}_{l_3}^{l_2}(b_3) = a_2$. Consider now the expression $E = r_1 \check{\bowtie}_{l_3:l_1,l_2} r_2$ over the following database d:

$$d = \left\{ r_1 = \begin{array}{|c|}\hline A:l_1 \\ \hline a_1 \\ \hline \end{array}, \qquad r_2 = \begin{array}{|c|}\hline A:l_2 \\ \hline a_2 \\ \hline \end{array} \right\}$$

Clearly, T and T' agree on d and E, since there is no level l in h such that $l_1 \leq_L l$ or $l_2 \leq_L l$. However, $E_T(d) = \begin{array}{|cc|}\hline A:l_1 & A:l_2 \\ \hline a_1 & a_2 \\ \hline \end{array} \neq \emptyset = E_{T'}(d)$, which proves that $\check{\bowtie}$ is not taxonomy-independent. Along the same lines, consider the expression $E' = r_1 \check{-}_{l_3} r_2$. We have $E'_T(d) = \emptyset \neq \begin{array}{|c|}\hline A:l_1 \\ \hline a_1 \\ \hline \end{array} = E'_{T'}(d)$, which proves that $\check{-}$ is not taxonomy-independent either.

By Definition 8, $\hat{\varepsilon}$ is clearly a taxonomy-independent operator, since the level mappings are only used upwards, starting from the values in the input relation. Therefore, the evaluation of an upward extension will necessarily be the same with respect to any two taxonomies that agree on the input relation. In addition, by using the equivalences (1), (3) and (5) in Section 4, we can see that $\hat{\sigma}$, $\hat{\bowtie}$ and $\hat{-}$ are also taxonomy-independent, since they can be rewritten in terms of $\hat{\varepsilon}$ and the classical (trivially taxonomy-independent) RA operators. Finally, as in the case of $\hat{\varepsilon}$, the downward selection operator $\check{\sigma}$ is also taxonomy-independent, since, as of Definition 11, the level mappings are only used upwards, starting from a value (m) present in the expression.

Let us call TI-HDRC the language formed by the HDRC expressions that are taxonomy-independent. Unfortunately, from the fact that the problem of testing domain-independence of classical relational calculus expressions is undecidable, it easily follows that it is also undecidable to determine whether an HDRC expression is taxonomy-independent. However, as it happens in the traditional setting, we can define a restricted version of HDRC, called *safe* HDRC and denoted by HDRC^{safe}, that allows us to formulate taxonomy-independent expressions only.

Definition 20 (hdrc^{safe}) HDRC^{safe} consists of exactly the HDRC expressions $\{A_1 : x_1, \ldots, A_n : x_n \mid \psi\}$ such that the formula ψ satisfies the following conditions:

1. the $\forall$ quantifier does not occur in ψ;
2. in any disjunctive subformula of ψ of the form $\psi_1 \vee \psi_2$, ψ_1 and ψ_2 have the same set of free variables;
3. in any maximal conjunctive subformula of ψ of the form $\psi_1 \wedge \ldots \wedge \psi_n$ all free variables are bounded, in the following sense:
   - if a variable x occurs in a formula ψ_i, where ψ_i is not negated (i.e., is not of the form $\neg\psi'_i$) and is not an equality atom (i.e., is not of the form $t = t'$), then x is bounded;
   - if a variable x occurs in a formula ψ_i, where ψ_i is of the form $x = a$ or $a = x$, where a is a constant value, then x is bounded;
   - if a variable x occurs in a formula ψ_i, where ψ_i is of the form $x = y$ or $y = x$, where y is a bounded variable, then x is bounded;
   - if a variable x occurs in a formula ψ_i, where ψ_i is of the form $x = \text{LMAP}_l^{l'}(y)$ or $\text{LMAP}_l^{l'}(y) = x$, where y is a bounded variable, then x is bounded.
4. a negated subformula $\neg\psi'$ only occurs in ψ in a conjunction of formulas, as discussed in item 3 (that is, it must be part of a larger subformula of the form: $\psi_1 \wedge \ldots \wedge \psi_i \wedge \neg\psi' \wedge \psi_{i+1} \wedge \ldots \wedge \psi_n$).

The following result shows that the restriction to safe query expressions does not limit the expressiveness of the language.

Lemma 3 *Any taxonomy-independent* HDRC *expression can be expressed in* HDRC^{safe}.

Proof Let $E = \{\tau \mid \psi\}$ be a HDRC query over a set of t-schemas $\mathbf{S}$ for a taxonomy T. We prove the claim by showing that if E is taxonomy independent then it can be converted into a query $\hat{E} = \{\tau \mid \hat{\psi}\}$ such that: (i) $\hat{\psi}$ satisfies the conditions of Definition 20 and (ii) $\hat{E}$ is equivalent to E, i.e., $E_T(d) = \hat{E}_T(d)$ for every t-database d for T over $\mathbf{S}$.

First of all, by considering the known equivalences existing between logical connectives and quantifiers, we can transform the formula ψ into an equivalent formula ψ' in which the $\forall$ quantifier and the $\vee$ connective do not occur. Specifically, this can be done by recursively substituting in ψ the expressions $\phi_1 \vee \phi_2$ by $\neg(\neg\phi_1 \wedge \neg\phi_2)$ and the expressions $\forall x(\phi)$ by $\neg(\exists x(\neg(\phi)))$. Note that this step eliminates all disjunctions from ψ; however, new disjunctions may appear in subsequent transformations of the formula, which we shall describe next. Yet, all new occurrences of disjunctions will comply with the safeness requirements specified in condition 2 of Definition 20.

Our proof can now proceed as follows. First we show how to build a logical formula $\eta^l(x)$ that holds if and only if x is substituted with any of the members of level l that can be obtained from $\text{ADOM}(d, E)$ via the level mappings: such members are precisely the values of level l that can be "reached" from the values in the active domain of d or from the constants used in the expression E. Then, we use η^l to modify ψ' so as to force every variable of type l to range over the reachable members of level l. In this way, all boundedness requirements from Definition 20 are met. Finally, we show that, if, as assumed, the initial expression E is taxonomy independent, then it is equivalent to the obtained expression $\hat{E}$.

For convenience, we first introduce a technical notion. Given a set of values C and a level l of an h-domain of T, let us call the *projection on l of C*, denoted by $\text{PROJ}^l_\uparrow(C)$, the set of members v of l such that there is a value in C that is mapped to v by some level mapping of T. In symbols:

$$\text{PROJ}^l_\uparrow(C) = \{v \in M(l) \mid \exists v' \in C, \exists \text{LMAP}^{l_2}_{l_1} \text{ in } T : \text{LMAP}^{l_2}_{l_1}(v') = v\}.$$

With this, we can now build a HDRC formula $\eta^l(x)$ with only one free variable x of type l such that, for every t-database d for T, $\eta^l(x)$ is true on a substitution s if and only if the value of s on x is an element of $\text{PROJ}^l_\uparrow(\text{ADOM}(d, E))$. This is done as follows.

For every t-schema $S_i = \{A_{i,1} : l_{i,1}, \ldots, A_{i,j} : l_{i,j}, \ldots, A_{i,k_i} : l_{i,k_i}\}$ in $\mathbf{S} = \{S_1, \ldots, S_r\}$ such that $l_{i,j} \leq_L l$, we define a formula $\phi^l_{A_{i,j}}(x)$ with free variable x that, for every t-database, is true on a substitution s if and only if $s(x)$ is in the projection on l of the values occurring in $r.A_{i,j}$, that is: $s(x) \in \text{PROJ}^l_\uparrow(\pi_{A_{i,j}}(r))$. As an example, if $S_1 = \{A_{1,1} : l_{1,1}, \ldots, A_{1,k} : l_{1,k}\}$ and $l_{1,1} \leq_L l$, then the formula $\phi^l_{A_{1,1}}(x)$ is:

$$\phi^l_{A_{1,1}}(x) = \exists x_{1,2}(\ldots \exists x_{1,k}(r(A_{1,1} : x_{1,1}, \ldots, A_{1,k} : x_{1,k})) \ldots) \wedge (\text{LMAP}^l_{l_{1,1}}(x_{1,1}) = x)$$

Clearly, the formula

$$\eta^l_d(x) = \bigvee_{\substack{1 \leq i \leq r \\ 1 \leq j \leq k_i}} \phi^l_{A_{i,j}}(x),$$

defined as the disjunction of all the $\phi^l_{A_{i,j}}$'s, is true on s if and only if $s(x)$ belongs to the projection on l of the active domain of d.

Similarly, if E involves the values $v_1, \ldots, v_q$ of levels $l_1, \ldots, l_q$, respectively, such that $l_i \leq_L l$ for $1 \leq i \leq q$, the formula $\eta^l_E(x)$ defined as:

$$\eta^l_E(x) = (x = \text{LMAP}^l_{l_1}(v_1)) \vee \ldots \vee (x = \text{LMAP}^l_{l_q}(v_q))$$

is true on s if and only if $s(x)$ is one of $\text{LMAP}^l_{l_1}(v_1), \ldots, \text{LMAP}^l_{l_q}(v_q)$.

With this, we can introduce the notation $\eta^l(x) = \eta^l_d(x) \vee \eta^l_E(x)$, which is true on s if and only if the value of s on x is an element of $\text{PROJ}^l_\uparrow(\text{ADOM}(d, E))$.

Now we modify formula ψ' so that the range of each variable x of type l is specified by the formula $\eta^l(x)$. Let $x_1, \ldots, x_n$ be the free variables of the query $E = \{\tau \mid \psi'\}$ of types $l_1, \ldots, l_n$, respectively, and let

$$\hat{E} = \{\tau \mid \eta^{l_1}(x_1) \wedge \ldots \wedge \eta^{l_n}(x_n) \wedge \psi''\},$$

where ψ'' is obtained from ψ' as follows: (i) the quantified subformulas $\exists x(\phi)$, where x is of type l, are recursively substituted by $\exists x(\eta^l(x) \wedge \phi)$, and (ii) the negated subformulas $\neg\phi$ that do not occur in a conjunction of formulas are recursively substituted by $\eta^{l_1}(y_1) \wedge \ldots \wedge \eta^{l_p}(y_p) \wedge \neg\phi$, where $y_1, \ldots, y_p$ are the free variables of ϕ of types $l_1, \ldots, l_p$, respectively.

We claim that $\hat{E}$ is the desired result since: (a) $\hat{E}$ is an HDRC^{safe} expression and (b) if E is taxonomy independent, then $\hat{E}$ is equivalent to E.

Part (a) follows from the fact that, by construction: (i) the $\forall$ quantifiers have been eliminated in the first step, (ii) disjunctions only occur in the subformulas $\eta^l(x)$ that involve the same free variable x of type l, (iii) the maximal conjunctive subformula of ψ'' is ψ'' itself and all the free variables occurring in it are bounded by the formulas $\phi^l_{A_{i,j}}$, and (iv) all negated subformulas occur in a conjunction of formulas, according to the last step of the transformation.

For part (b) of the claim above we observe that if E is taxonomy independent, then it can be equivalently evaluated on any two taxonomies that agree on E and the underlying t-database. Then, for each t-database d for a taxonomy T, let T' be the restriction of T involving only the values in the active domain of d and the mappings induced by T on $\text{ADOM}(d, E)$. Clearly T' agrees with T on d and E and therefore the evaluation of E on T' produces the same result as the evaluation of E on T. Now, the difference between E and $\hat{E}$ is in the subformulas that force the variables to vary only in the projections of $\text{ADOM}(d, E)$, which are clearly included in T' by definition. It follows that $E_{T'}(d) = \hat{E}_{T'}(d)$ and that $\hat{E}$ is also taxonomy independent. Therefore, for every t-database, the results of the E and E'' coincide, and so they are equivalent.

As usual, we say that a query language L_1 is *at least as expressive as* another query language L_2, in symbols $L_1 \sqsupseteq L_2$, if for each query q of L_2 there exists an equivalent query q' of L_1. If both $L_1 \sqsupseteq L_2$ and $L_2 \sqsupseteq L_1$ then we say that L_1 and L_2 are *equivalent*.

We have the following "completeness" result that summarizes the relationships between the safe portions of the various query languages that we have defined.

Theorem 2 *The following languages are equivalent.*

- TRA$^-$
- TI-HDRC
- HDRC^{safe}

Proof We show that TI-HDRC $\sqsupseteq$ TRA$^-$ and that TRA$^-$ $\sqsupseteq$ HDRC^{safe}. The claim then follows from the fact that, by Lemma 3, HDRC^{safe} $\sqsupseteq$ TI-HDRC.

TI-HDRC $\sqsupseteq$ TRA$^-$. Let E_a be a TRA$^-$ expression. The proof proceeds by induction on the number of operators used in E_a and derives an HDRC expression E_c that is equivalent to E_a. Since E_a is taxonomy independent by Lemma 2, it follows that E_c is also taxonomy independent and so it actually belongs to TI-HDRC.
Basis. In the base case E_a does not involve any operator and so $E_a = r$, where r is a t-relation over a t-schema $S = \{A_1 : l_1, \ldots, A_n : l_n\}$. Then, the HDRC expression equivalent to E_a is trivially: $E_c = \{A_1 : x_1, \ldots, A_n : x_n \mid r(A_1 : x_1, \ldots, A_n : x_n)\}$.
Induction. We consider the various cases for the top-level operator assuming that the TRA$^-$ subexpressions on which the operator is applied have equivalent HDRC expressions. We also assume that E_a only involves the operators of renaming, projection, union, (standard) selection, (standard) join, (standard) difference, upward extension, and downward selection. This assumption entails no loss of generality since, using the equivalences (1)-(6) in Section 4, we can transform any TRA$^-$ expression into an equivalent expression in which all the other operators are not present. We also assume that in E_a the operands of the join operator (if any) do not have attributes in common (and so the join is actually a cartesian product) since any join involving operands with common attributes can be transformed into a cartesian product followed by a selection.

Let us start with the unary operators. Let $E'_c = \{A_1 : x_1, \ldots, A_n : x_n \mid \psi\}$ be the HDRC expression equivalent to the TRA$^-$ expression E'_a.

- $E_a = \rho_f(E'_a)$: then $E_c = \{f(A_1) : x_1, \ldots, f(A_n) : x_n \mid \psi\}$;
- $E_a = \pi_{A_2,\ldots,A_n}(E'_a)$: then $E_c = \{A_2 : x_2, \ldots, A_n : x_n \mid \psi\}$;
- $E_a = \sigma_{A_i=c}(E'_a)$: then $E_c = \{A_1 : x_1, \ldots, A_n : x_n \mid \psi \wedge (x_i = c)\}$;
- $E_a = \sigma_{A_i=A_j}(E'_a)$: then $E_c = \{A_1 : x_1, \ldots, A_n : x_n \mid \psi \wedge (x_i = x_j)\}$;
- $E_a = \hat{\varepsilon}^{A_i:l_j}_{A_i:l_i}(E'_a)$: then $E_c = \{A_1 : x_1, \ldots, A_n : x_n, A_i : x \mid \psi \wedge (x = \textsc{lmap}^{l_j}_{l_i}(x_i))\}$;
- $E_a = \check{\sigma}_{A_i:l=c}(E'_a)$: then $E_c = \{A_1 : x_1, \ldots, A_n : x_n \mid \psi \wedge (x_i = \textsc{lmap}^{l_i}_{l}(c))\}$.

For union and difference let $E'_c = \{A_1 : x_1, \ldots, A_n : x_n \mid \psi'\}$ and $E''_c = \{A_1 : y_1, \ldots, A_n : y_n \mid \psi''\}$ be the HDRC expressions equivalent to the TRA$^-$ expressions E'_a and E''_a, respectively.

- If $E_a = E'_a \cup E''_a$ then $E_c = \{A_1 : x_1, \ldots, A_n : x_n \mid \psi' \vee \psi'''\}$ where ψ''' is obtained from ψ'' by renaming $x_1, \ldots, x_n$ to $y_1, \ldots, y_n$ respectively;
- If $E_a = E'_a - E''_a$ then $E_c = \{A_1 : x_1, \ldots, A_n : x_n \mid \psi' \wedge \neg\psi'''\}$ where ψ''' is obtained from ψ'' by renaming $y_1, \ldots, y_n$ to $x_1, \ldots, x_n$ respectively.

For the join operator let $E'_c = \{A_1 : x_1, \ldots, A_n : x_n \mid \psi'\}$ and $E''_c = \{B_1 : y_1, \ldots, B_m : y_m \mid \psi''\}$ be the HDRC expressions equivalent to the TRA$^-$ expressions E'_a and E''_a, respectively (since they do not have attributes in common, we can assume that E'_c and E''_c have disjoint sets of free variables).

- If $E_a = E'_a \bowtie E''_a$ then $E_c = \{A_1 : x_1, \ldots, A_n : x_n, B_1 : y_1, \ldots, B_m : y_m \mid \psi' \wedge \psi''\}$.

This completes the first part of the proof.

TRA$^-$ $\sqsupseteq$ HDRC^{safe}. Let $E_c = \{A_1 : x_1, \ldots, A_n : x_n \mid \psi\}$ be a HDRC^{safe} expression. Also in this case, the proof proceeds by induction on the number of connectives and quantifiers used in ψ and derives a TRA$^-$ expression E_a that is equivalent to E_c. We also assume, without loss of generality, that in ψ the constant values only occur in atoms of the form $(x = v)$.

A preliminary concept is needed. For each free variable x of type l occurring in ψ, we denote by E_x the TRA$^-$ expression that, for each t-database d, produces as a result a t-relation over a single attribute named x that includes the projection on l of the active domain of d and E_c, as defined in the proof of Lemma 3, i.e.: $E_x = \textsc{proj}^l_{\uparrow}(\textsc{adom}(d, E_c))$. This expression can be built as follows: we first apply the upward extension $\hat{\varepsilon}^{A:l}_{A:l'}$ to all the t-relations having an attribute A over a level $l' \leq_L l$, we rename the new attribute $A : l$ as $x : l$, we then project on the new attribute x, and we finally take the union of all the results and, if any, of the members of level l reachable from the values occurring in E_c. For example, if the t-database contains two t-relations r_1 and r_2 over $S_1 = \{A_1 : l_1\}$ and $S_2 = \{A_2 : l_2\}$, respectively, E_c involves the value 3 of level $l_3 \leq_L l$ such that $\textsc{lmap}^l_{l_3}(3) = 4$, and the type of a variable x in E_c is a level l such that $l_1 \leq_L l$ and $l_2 \leq_L l$, then $E_x = \pi_x(\rho_{x:l \leftarrow A_1:l}(\hat{\varepsilon}^{A_1:l}_{A_1:l_1}(r_1))) \cup \pi_x(\rho_{x:l \leftarrow A_2:l}(\hat{\varepsilon}^{A_2:l}_{A_2:l_2}(r_2))) \cup \{\langle 4 \rangle\}$.

We can start now with the inductive proof. To simplify the construction, in the induction we first build

a TRA$^-$ expression equivalent to E_c with an output t-schema whose attributes are named as the free variables $x_1, \ldots, x_n$ occurring in ψ. Then, we shall obtain E_a from E by a straightforward renaming of the attributes.

Basis. In the base case ψ is a maximal conjunction $\psi_1 \wedge \ldots \wedge \psi_k$ of atoms. We build for each ψ_i a TRA$^-$ expression E_i as follows:

- if $\psi_i = r(A_1 : x_1 \ldots A_n : x_n)$ then $E_i = \rho_{x_1 \ldots x_n \leftarrow A_1 \ldots A_n}(r)$,
- if $\psi_i = (x = v)$ then $E_i = \sigma_{x=v}(E_x)$,
- if $\psi_i = (x = y)$ then $E_i = \sigma_{x=y}(E_x \bowtie E_y)$,
- if $\psi_i = (\text{LMAP}_l^{l'}(x) = y)$ then $E_i = \rho_{y:l' \leftarrow x:l'}(\hat{\varepsilon}_{x:l}^{x:l'}(E_x)) \bowtie E_y$, and
- if $\psi_i = (\text{LMAP}_{l_1}^{l}(x) = \text{LMAP}_{l_2}^{l}(y))$ then

$$E_i = \pi_{xy}(\rho_{z:l \leftarrow x:l}(\hat{\varepsilon}_{x:l_1}^{x:l}(E_x)) \bowtie \rho_{z:l \leftarrow y:l}(\hat{\varepsilon}_{y:l_2}^{y:l}(E_y))).$$

Then, we have that $E_a = E_1 \bowtie \ldots \bowtie E_k$.

Induction. Since universal quantifiers are not allowed in HDRC^{safe} expressions and negation can only appear within conjunctions, we have only three cases:

- $\psi = \exists x(\psi')$: let $x, x_1, \ldots, x_n$ be the free variables of ψ' and, according to the inductive hypothesis, let E' be the TRA$^-$ expression equivalent to $\{x : x, x_1 : x_1, \ldots, x_n : x_n \mid \psi'\}$; then: $E_a = \pi_{x_1 \ldots x_n}(E')$;
- $\psi = \psi_1 \vee \psi_2$: let $x_1 \ldots x_n$ be the free variables of ψ_1 and ψ_2 (they are the same by Definition 20) and let E_1 and E_2 be the TRA$^-$ expressions equivalent to $\{x_1 : x_1, \ldots, x_n : x_n \mid \psi_1\}$ and $\{x_1 : x_1, \ldots, x_n : x_n \mid \psi_2\}$ respectively; then $E_a = E_1 \cup E_2$;
- ψ is a maximal conjunction of the form $\psi_1 \wedge \ldots \wedge \psi_n$: intuitively, given the TRA$^-$ expressions $E_1, \ldots, E_n$ equivalent to $\{x_{1,1} : x_{1,1}, \ldots, x_{1,l_1} : x_{1,l_1} \mid \psi_1\}, \ldots, \{x_{n,1} : x_{n,1}, \ldots, x_{n,l_n} : x_{n,l_n} \mid \psi_n\}$ respectively, the expression E_a corresponds to the intersection of $E_1, \ldots, E_n$; however, the formulas $\psi_1, \ldots, \psi_n$ are not necessarily defined on the same free variables; therefore, in order to obtain compatible expressions, if a formula ψ_i has a set of free variables $Z = \{z_1, \ldots, z_k\}$, we need to extend the equivalent TRA$^-$ expression E_i to the set of all free variables X occurring in $\psi_1, \ldots, \psi_n$ by means of an expression of the form: $\bar{E}_i = E_i \bowtie E_{w_1} \bowtie \ldots \bowtie E_{w_l}$, where $\{w_1 \ldots w_l\} = X - Z$; in the case in which ψ_i is negative (i.e., it is of the form $\neg\psi_i'$), then $\bar{E}_i = ((E_{z_1} \bowtie \ldots \bowtie E_{z_k}) - E_i') \bowtie E_{w_1} \bowtie \ldots \bowtie E_{w_l}$ where E_i' is the TRA$^-$ expression equivalent to $\{z_1 : z_1, \ldots, z_k : z_k \mid \psi_i'\}$; the final TRA$^-$ expression is then obtained by joining all the subexpressions obtained in this way, i.e.: $E_a = \bar{E}_1 \bowtie \ldots \bowtie \bar{E}_n$.

This completes the induction. Let now E be the TRA$^-$ expression obtained as a result: it is indeed defined over a set of attributes named as the free variables of the original HDRC^{safe} expression $E_c = \{A_1 : x_1, \ldots, A_n : x_n \mid \psi\}$. The final TRA$^-$ expression can be obtained from E by a straightforward renaming:

$$E_a = \rho_{A_1 \ldots A_n \leftarrow x_1 \ldots x_n}(E).$$

6 Discussion

In this section, we report a few observations that emphasize the impact, the consequences and the possible further developments of our work.

6.1 Efficient implementation of taxonomic operators

We have introduced several operators offering a full-fledged suite of tools for querying databases with taxonomies. Upward and downward extension are the central operators in that they are sufficient to capture all the other taxonomic operators (see Equations (1)-(6)). However, for certain operators, a direct implementation that is not based on the extension operators may be preferable for efficiency reasons.

For example, this is apparent for the downward selection operator, whose rewriting shown in Equation (2) uses an operator that is not taxonomy-independent ($\check{\varepsilon}$), although $\check{\sigma}$ itself is taxonomy-independent. Indeed, instead of first extending the relation downwards to then perform a classical selection, as suggested by the equivalence shown in Equation (2), it may be more convenient to map the member of the lower level into the upper level and then use the obtained value for comparison, as suggested in the very definition of downward selection (Definition 11).

Taxonomy independence may not be the only issue. Think for example of an upward selection of the form $\hat{\sigma}_{A:l=m}(r)$, where m is an extremely selective value for the upper level and r is a large relation; if statistics about value distribution in the taxonomy are available at the DBMS site, we might prefer an execution of the selection that first maps m downwards and then compares the (few) obtained values with those in r.

In general, statistical knowledge about the data and the taxonomy may enable an efficiency-aware use of the equivalence rewritings discussed in Section 4.

6.2 Schema agnosticism

A language supporting query relaxation may need to provide users with a tool that allows them to formulate their queries without fully knowing the schema of the

database and all the possible levels of representation of the data.

Along these lines, a first step towards offering users a high-level and transparent language, with the ability to simplify the specification of queries, would consist of giving the possibility to write queries in which some of the levels are left unspecified. As a first, very simple example consider a t-selection written, e.g., as $\hat{\sigma}_{A=m}(r)$, i.e., where the user did not specify the level l at which attribute A is stored in the database. Here we are not merely resorting to a shorthand for the full notation $\hat{\sigma}_{A:l=m}(r)$, as we did in the previous sections, but are rather emphasizing that the user is indeed unaware of the level l used in the database and just wants to match a value m with whatever is stored for attribute A.

Another example comes from the notion of natural join, which we may revisit in a relaxed, taxonomic sense. In their purely relational version, natural joins leave it to the "system" to properly match corresponding attributes in the joined relations. In a taxonomic version of natural join, the user may not want or not be able to specify the level at which the attributes for the joined relations must be compared. A taxonomic natural join of the form $r_1 \hat{\bowtie} r_2$ could therefore be conceived as an instance of the standard taxonomic join $r_1 \hat{\bowtie}_{S:\bar{S}_1,\bar{S}_2} r_2$ in which the system is in charge both of identifying two corresponding sets $\bar{S}_1$ and $\bar{S}_2$ with equally named attributes in the joined relations (as happens in the standard natural join) and also of determining the upper bound S of $\bar{S}_1$ and $\bar{S}_2$ to be used for the t-join, which defines the level(s) at which the attributes of the joined relations must be compared. This requires a way to automatically determine such a level S. The most natural choice is to use minimal upper bounds for upward joins and maximal lower bounds for downward joins. As an example, one might want to do a natural join between two relations, each with a *Location* attribute, one at the theater level, the other at the airport level, with no need to specify that their upper bound is *Location* : city. In this sense, the natural upper join between relation r_1 of Figure 5 and relation r_5 of Figure 7 can be simply written as $r_1 \hat{\bowtie} r_5$; the result is relation r_6, also shown in Figure 7.

As an aside, note that very often the minimal upper bound is unique (and therefore a *least* upper bound) and similarly for the maximal (thus *greatest*) lower bound. However, in general, there might be more than one such bound. In turn, this determines two possible semantics for the natural join: a looser existential semantics and a stricter universal semantics. In the former case, a join tuple is considered to be part of the result if it satisfies the join condition for *at least one* minimal upper bound (respectively, maximal lower bound); in the latter case, a join tuple is in the result if it satisfies the join condition for *all* the minimal upper bounds (respectively, maximal lower bounds). As further hinted at in the next paragraph, the choice of the semantics or, more generally, of the upper/lower bounds to use for query relaxation may be determined through an interaction with the user, who is the ultimate judge of the query intent and result.

A further step towards providing users with a high-level access to a relational database would consist in allowing users to write queries that are much closer to natural language. For instance, a very common way to express searches today relies only on a collection of keywords. Our framework can actually provide an effective support to keyword-based queries over relational databases by "chasing" the underlying relations with all the applicable extension operators until some keyword occurs in the result. Consider for instance the example in Section 2.1 and assume that the query just consists of the keyword "Italy". If we iteratively tested all the possible applications of the upward extension operators for the geographical taxonomy in Figure 2 to the table in Figure 1, we would extend tuple t_a with the values "Verona" for City and "Italy" for Country. The tuple obtained in this way is then a suitable candidate answer to the given keyword-based query.

6.3 Interactive relaxation

Taxonomic versions of natural joins and other operators in which users are not required to refer to the level of granularity of the stored data are certainly of help when the details regarding the structure of the h-domains in use are not known. However, users might be willing to relax their queries beyond the levels automatically proposed by a system. For example, assume that the natural join between two *Location* attributes at the theater and airport levels returns no tuples if the upper bound in use is city. A taxonomy-aware system might interactively propose that the query should undergo an extra round of relaxation by using a coarser level as an upper bound, which might be satisfactory for the user in certain scenarios. For example, a low-cost flight to Orio al Serio airport, which refers to Bergamo at the city level, might be a suitable solution for an opera enthusiast wanting to reach La Scala in Milan, although the locations Orio al Serio and La Scala only match at the region level (both being in the Lombardy region), but not at the city level.

6.4 Completeness of the taxonomic paradigm

In Theorem 2, we have shown that two different languages (the algebraic TRA$^-$ and the logic-based HDRC^{safe}) have the same expressive power and therefore can implement the same set of queries. This leads to a notion of *completeness* that has also been used by Codd in the relational setting [19]: a query language is complete if it is at least as expressive as relational algebra (in our case, TRA$^-$). The fact that the query languages we have studied were defined independently but are nonetheless equivalent gives evidence to the fact that they are sufficiently expressive within the considered framework.

In addition, the proof of Theorem 2 is constructive, in the sense that, given a query in one of the two languages, we show how to construct an equivalent query in the other language. This provides an obvious hint at the way in which a declaratively specified query may be implemented.

A salient feature of the considered languages TRA$^-$ and its logical counterpart HDRC^{safe} is that, according to the results of Section 5, they guarantee finiteness of query answering. In particular, to do so, these languages need to exclude the downward versions of extension, join, and difference, but allow one to express downward selections.

6.5 Materialization of taxonomies

The use of existing taxonomies is a crucial aspect concerning the practicality of our framework. One of the main aims of our work is indeed that of extending current relational technology without the need of building everything from scratch. According to our proposed model, existing relational databases may accommodate already existing taxonomies either by completely materializing them in the database, or by keeping them as an external part of the system in which the different steps from one level to another in the taxonomy are computed on the fly.

For instance, an agreed upon taxonomy that is limited in size and stable, such as certain kinds of geographical taxonomies, lends itself well to being materialized. On the other hand, there are taxonomies which are not bounded in size (e.g., a taxonomy including an h-domain describing a timeline) or whose members at the finest level are too numerous to be materialized (again, think of the level of timestamps for a time h-domain). Another example of taxonomies that cannot be conveniently materialized comes from trip planners: at the finest level, their organization in stops and schedules varies too often to be considered stable; in addition to that, some integrators for trip planning services may not own all the data they need in order to compute a plan, but rather are only allowed to incrementally query external services as their plan is being calculated.

6.6 Idiosyncrasies of upward and downward operators

The examples we have discussed throughout the paper show that both forms of relaxation (upward and downward) are well defined and useful. Yet, we now emphasize that the kind of relaxation offered by downward operators is somewhat stronger than that of their upward counterparts. For example, consider a geographical taxonomy with cities, states, and countries, and assume that in our relation (say, r) the tuples are stored at the state level. Now, if we pose the query $\hat{\sigma}_{\text{Location}=\texttt{USA}}(r)$, the answer will, e.g., contain a tuple, say t, in which the state is California. This is meaningful, since all California locations are also USA locations. If we pose, instead, the query $\check{\sigma}_{\text{Location}=\texttt{Sacramento}}(r)$ the answer will anyhow also contain the same tuple t, even if `Sacramento` is just *one* city in California, not covering the entire territory.

To further emphasize this circumstance, consider the following relation:

$$r = \begin{array}{|c|}\hline \textit{Location} : \mathsf{city} \\ \hline \text{San Francisco} \\ \hline \end{array}.$$

Clearly, if we upward extend r to the state level and then project away the city level, we are left with a single tuple whose single attribute has the value `California`:

$$r' = \pi_{\mathsf{state}}(\hat{\varepsilon}^{\mathsf{state}}_{\mathsf{city}}(r)) = \begin{array}{|c|}\hline \textit{Location} : \mathsf{state} \\ \hline \text{California} \\ \hline \end{array}.$$

A downward selection on the result with the condition Location = `Sacramento` will retain the tuple in the answer, although the original city in the database was San Francisco, and not Sacramento, i.e.,

$$\check{\sigma}_{\text{Location}=\texttt{Sacramento}}(r') = \begin{array}{|c|}\hline \textit{Location} : \mathsf{state} \\ \hline \text{California} \\ \hline \end{array}.$$

This phenomenon is caused by the fact that the upward extension has obfuscated the original information, subsequently eliminated by the projection. Therefore, although perhaps surprising at first, the result of the query is certainly compatible with the relaxed query semantics of downward selection.

7 Related work

This work largely extends a preliminary version that appeared in the proceedings of ER 2010 [38]. The main differences with respect to the earlier paper are the following: (i) we have added a new section including several

motivating examples of real-world scenarios in which taxonomy-based query relaxation plays a central role, (ii) we have extended the algebraic language, by introducing the taxonomic versions of the difference operator and by discussing other taxonomic operators, (iii) we have identified several new equivalence rules for algebraic expressions, (iv) we have proposed a declarative, calculus-based query language for querying databases with taxonomies as a basis for a possible extension of SQL, (v) we have investigated the important notion of taxonomy independence of a query language in this context, (vi) we have compared the expressive power of the various languages identifying their strengths and weaknesses in terms of expressive power and finiteness of query answers, (vii) we have discussed in a new section the results of our contribution, a number of issues related to its implementation, and possible future developments, and (viii) we have largely extended the comparison with the related literature.

The approach proposed in this paper is focused on the relaxation of queries with the goal of accommodating user's needs — a problem that has been investigated in several research areas under different perspectives. In the database area, query relaxation has been addressed mainly in the context of XML, RDF, and other semi-structured data models, with different goals in mind: combining database-style querying and keyword search [6], querying databases with natural language interfaces [35], and dealing with the structural heterogeneity of a large number of XML data sources [36]. In [31], queries are relaxed using an ontology that is extracted from the DTDs of XML databases, but the notion of relaxation is different from ours, since it refers to a less restrictive form of matching between path queries and paths of the database. Along the same line, in [5] the authors propose an approach to query relaxation over XML data based on the idea of considering an XPath expression as a template for keyword search, thereby enabling approximate query answers on structure and using it to provide a context for full-text search. Our approach to query relaxation is neither based on structure relaxation nor on full-text search, but it rather relies on relaxing the matching between terms in a query and terms in the database, leveraging existing taxonomies on those terms. Approaches based on relaxing the matching between the query and the structure of data have also been tackled for RDF databases and ontology-based languages, by introducing a measure of distance between paths [29], by reformulating triple-pattern queries by means of statistical language models [23], by combining approximate query answer with full-text search [22], by providing a support to joins based on resource similarity [9], and by exploiting domain knowledge and user preferences [20]. Again, apart from the differences in the data model of reference, our notion of query relaxation is different because it considers neither the structure nor other model-specific features, but only simple taxonomies between values.

The formal notion of malleable schema has been introduced to deal with vagueness and ambiguity in database querying by incorporating imprecise and overlapping definitions of data structures [21,42]. An alternative formal framework relies on multi-structural databases [25,26], where data objects are segmented according to multiple distinct criteria in a lattice structure and queries are formulated in this structure. The idea of making queries more flexible by the logical relaxation of their conditions has also been studied in the context of deductive databases and logic programming queries [28]. A number of operators that have some similarity with the taxonomic operators of our TRA have been proposed for navigating an ontology in a completely different scenario in which the ontology is built over a generic set of concepts and is represented using a lattice-algebraic description language [4]. This model has also been used for query relaxation according to a similarity measure between concepts based on subsumption in the ontology [13]. Although these approaches have some relationship with ours, a direct comparison cannot be done given the diversities in the data model and in the query evaluation process.

A notion of query relaxation is also used in the context of location-based search [16], but in the typical IR scenario in which a query consists of a set of terms, and query evaluation is focused in the ranked retrieval of documents. This is also the case of the approach in [12], where the authors consider the problem of fuzzy matching of queries with items. Actually, in the information retrieval area, which is however clearly different from ours, document taxonomies and, more in general, ontologies have been largely used for query expansion [10], a technique aimed at automatically reformulating a keyword-based user request into a form that is more amenable to information retrieval. For instance, in [27] the authors focus on classifying documents into taxonomy nodes and developing a taxonomy-based scoring function to measure the matching between textual queries and documents, while in [8] the authors propose a framework for relaxing user requests over ontologies to find the most useful Web service.

Many other papers have proposed non-traditional approaches to access a database, in which query conditions are considered as soft and are replaced by constraints that capture additional criteria for satisfying user needs. The goal is to avoid both the empty-answer

problem, where the data do not match the query, and the too-many-answers problem, where too many results match the query. A notable example is preference query processing [7] where returned results are ranked according to the preferences of the users, which are represented as a partial order [17,32] or as a numerical score over the data [2,34] (see [41] for a comprehensive survey of the solutions proposed to the problem of representing user preferences and using them for query processing). Preference query processing is actually an instance of the more general problem of top-k query processing, in which only the most relevant query answers are returned to the user, for some definition of degree of relevance. This field has been addressed by a large body of research during the last years, as surveyed in [30]. Rankings may be established according to a plethora of different multi-dimensional criteria, including proximity [37], diversity [15], context [11], contextual preferences [18], and others. The empty-answer problem has also been addressed by adjusting values occurring in selections and joins [33,39]. Our approach shares the same goal with all of these approaches but it relies on a completely different criterion for query relaxation: the availability of taxonomies on the data domains.

The many approaches to the problem of schema matching [3,40], which focus on finding correspondences between elements of two database schemas, are also related to the problem studied in this paper. Indeed, our query relaxation can be seen as a matching between the schema of the database and a sort of "implicit" schema to which the query refers. However, although some of the proposed techniques could be helpful here, our goal is quite different, since we aim at generating the answer to a query rather than the correspondences between elements of two schemas. Moreover, our approach takes also care of reconciliating possible mismatches existing at the instance level, by suitably finding corresponding members at different levels in each domain.

Summarizing, we can say that many of the above mentioned approaches rely on non-traditional database models, whereas we refer to a natural extension of the relational model and of the classical relational query languages. This guarantees a smooth implementation of the approach with today's most spread database technology. Moreover, none of them strictly considers the problem of query relaxation via taxonomies, which is our concern. In addition, the systematic analysis of query equivalence for optimization purposes and the investigation of the expressiveness of taxonomy-based query languages have never been studied in the relaxed case. Hence, we believe that the approach presented in this paper is complementary to other techniques for query relaxation and, in several cases, it might be used in combination with them.

We finally point out that the problem we have studied in this paper is quite different from the problem of ontology-based data access [14] in which an ontology provides a conceptual description of the content of the data sources and queries over the ontology are rewritten into queries over the underlying databases using the mapping between them.

As a final aside, we mention that the notion of taxonomy independence is partly related to the notion of *bounded depth domain independence* [1] (called "embedded domain independence" in [24]) introduced in the context of query languages with built-in functions for complex-object databases.

8 Conclusion

In this paper, we have presented a logical model and two abstract query languages as a foundation for querying relational databases using taxonomies. In order to facilitate the implementation of the approach with current technology, they rely on a natural extension of the relational model and of the classical relational database languages. A hierarchical organization of data allows the specification of queries that refer to values at varying levels of details, possibly different from those available in the underlying database. We have also studied the interaction between the various operators of the algebraic query language, the expressive power of the various languages, and the important property of taxonomy independence. These results provide a formal foundation for enhancing relational database technology with a comprehensive support for query relaxation with taxonomies.

We believe that several interesting directions of research can be pursued within the framework presented in this paper. Challenging extensions of our approach include: more general forms of relationship between values, suitable distance metrics between queries and answers, and ranking of answers according to some relevance criterion. We are also interested in a deep investigation of general properties of the query languages we have proposed and of their exploitation for simplifying the formulation of queries. In particular, we plan to develop methods for the automatic derivation of expressions on the basis of user queries expressed in a very high-level language, such as a keyword-based one. In addition, a study, in our context, of the classical tools for query optimization, such as query containment and equivalence, seems to be a promising extension of our research.

On the practical side, we plan to study how the presented approach can be implemented, in particular whether materialization of taxonomies is convenient. With this prototype, we plan to develop a quantitative analysis oriented to the optimization of relaxed queries. The equivalence results presented in this paper provide an important contribution in this direction.

References

1. S. Abiteboul and C. Beeri. The Power of Languages for the Manipulation of Complex Values. *VLDB Journal*, 4(4): 727–794, 1995.
2. R. Agrawal and E. L. Wimmers. A Framework for Expressing and Combining Preferences. In *Proc. of SIGMOD*, pag. 297–306, 2000.
3. Z. Bellahsene, A. Bonifati, and E. Rahm (ed.). Schema Matching and Mapping. Springer, 2011.
4. T. Andreasen and H. Bulskov. Conceptual querying through ontologies. *Fuzzy Sets and Systems*, 160(15): 2159–2172, 2009.
5. S. Amer-Yahia, L. V. S. Lakshmanan, and S. Pandit. Flexpath: Flexible structure and full-text querying for XML. In *Proc. of SIGMOD*, pag. 83–94, 2004.
6. S. Amer-Yahia, E. Curtmola, and A. Deutsch. Flexible and efficient XML search with complex full-text predicates. In *Proc. of SIGMOD*, pag. 575–586, 2006.
7. A. Arvanitis and G. Koutrika. PrefDB: bringing preferences closer to the DBMS. In *Proc. of SIGMOD*, pag. 665–668, 2012.
8. W.-T. Balke and M. Wagner. Through different eyes: assessing multiple conceptual views for querying web services. In *Proc. of WWW*, pag. 196–205, 2004.
9. A. Bernstein and C. Kiefer. Imprecise RDQL: towards generic retrieval in ontologies using similarity joins. In *Proc. of SAC*, pag. 1684–1689, 2006.
10. J. Bhogal, A. MacFarlane, and P. Smith. A review of ontology based query expansion. *Information Processing and Management*, 43(4): 866–886, 2007.
11. C. Bolchini, C. Curino, G. Orsi, E. Quintarelli, R. Rossato, F. Schreiber, and L. Tanca. And what Can Context Do for Data? *Com. of ACM*, 52(11): 136–140, 2009.
12. A. Z. Broder, M. Fontoura, V. Josifovski, and L. Riedel. A semantic approach to contextual advertising. In *Proc. of SIGIR*, pag. 559–566, 2007.
13. H. Bulskov, R. Knappe, and T. Andreasen. On querying ontologies and databases. In *Proc. of FQAS*, pag. 191–202, 2004.
14. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, and D. Fabio Savo. The MASTRO system for ontology-based data access. *Semantic Web*, 2(1): 43–53, 2011.
15. I. Catallo, E. Ciceri, P. Fraternali, D. Martinenghi, and M. Tagliasacchi. Top-k diversity queries over bounded regions. *ACM Trans. on Database Syst.*, 38(2): art. 10, 2013.
16. Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In *Proc. of SIGMOD*, pag. 277–288, 2006.
17. J. Chomicki. Preference Formulas in Relational Queries. *ACM Trans. on Database Syst.* 28(4): 427–466, 2003.
18. P. Ciaccia and R. Torlone. Modeling the Propagation of User Preferences. In *Proc. of ER*, pag. 304–317, 2011.
19. E. F. Codd. Relational Completeness of Data Base Sublanguages. In R. Rustin (ed.), *Database Systems*, Prentice Hall and IBM Research Report RJ 987, pag. 65–98, 1972.
20. P. Dolog, H. Stuckenschmidt, H. Wache, and J. Diederich. Relaxing RDF queries based on user and domain preferences. *J. Intell. Inf. Syst.*, 33(3): 239–260, 2009.
21. X. Dong and A. Y. Halevy. Malleable schemas: A preliminary report. In *Proc. of WebDB*, pag. 139–144, 2005.
22. S. Elbassuoni, M. Ramanath, R. Schenkel, and G. Weikum. Searching RDF Graphs with SPARQL and Keywords. *IEEE Data Engin. Bulletin*, 33(1): 16–24, 2010.
23. S. Elbassuoni, M. Ramanath, and G. Weikum. Query Relaxation for Entity-Relationship Search. In *Proc. of ESWC*, pag. 62–76, 2011.
24. M. Escobar-Molano, R. Hull, and D. Jacobs. Safety and Translation of Calculus Queries with Scalar Functions. *Proc. of PODS*, pag. 253-264, 1993.
25. R. Fagin, R. V. Guha, R. Kumar, J. Novak, D. Sivakumar, and A. Tomkins. Multi-structural databases. In *Proc. of PODS*, pag. 184–195, 2005.
26. R. Fagin, P. G. Kolaitis, R. V. Guha, R. Kumar, J. Novak, D. Sivakumar, and A. Tomkins. Efficient Implementation of Large-Scale Multi-Structural Databases. In *Proc. of SIGMOD*, pag. 958–969, 2005.
27. M. Fontoura, V. Josifovski, R. Kumar, C. Olston, A. Tomkins, and S. Vassilvitskii. Relaxation in text search using taxonomies. *Proc. of VLDB*, 1(1): 672–683, 2008.
28. T. Gaasterland, P. Godfrey, and J. Minker. Relaxation as a Platform for Cooperative Answering. *J. Intell. Inf. Syst.*, 1(3/4): 293–321, 1992.
29. C. A. Hurtado, A. Poulovassilis, and P. T. Wood. Query Relaxation in RDF. *J. Data Semantics*, 10: 31–61, 2008.
30. I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Comput. Surv.*, 40(4): art. 11, 2008.
31. Y. Kanza and Y. Sagiv. Flexible Queries Over Semistructured Data. *Proc. of PODS*, 40–51, 2001.
32. W. Kießling. Foundations of Preference in Database Systems. In *Proc. of VLDB*, pag. 311–322, 2005.
33. N. Koudas, C. Li, A. K. H. Tung, and R. Vernica. Relaxing join and selection queries. In *Proc. of VLDB*, pag. 199–210, 2006.
34. G. Koutrika and Y. E. Ioannidis. Personalization of Queries in Database Systems. In *Proc. of ICDE*, pag. 597–608, 2004.
35. Y. Li, H. Yang, and H. V. Jagadish. NaLIX: A generic natural language search environment for XML data. *ACM Trans. on Database Syst.*, 32(4): art. 30, 2007.
36. C. Liu, J. Li, J. Xu Yu, and R. Zhou. NaLIX: Adaptive relaxation for querying heterogeneous XML data sources. *Inf. Syst.*, 35(6): pag. 688–707, 2010.
37. D. Martinenghi and M. Tagliasacchi. Proximity measures for rank join. *ACM Trans. on Database Syst.* 37(1): art. 2, 2012.
38. D. Martinenghi and R. Torlone. Querying Databases with Taxonomies. In *Proc. of ER*, pag. 377–390, 2010.
39. X. Meng, Z. M. Ma, and L. Yan. Answering approximate queries over autonomous web databases. In *Proc. of WWW*, pag. 1021–1030, 2009.
40. E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4): 334–350, 2001.
41. K. Stefanidis, G. Koutrika, and E. Pitoura. A survey on representation, composition and application of preferences in database systems. *ACM Trans. on Database Syst.* 36(3): art. 19, 2011.
42. X. Zhou, J. Gaugaz, W. Balke, and W. Nejdl. Query relaxation using malleable schemas. In *Proc. of SIGMOD*, pag. 545–556, 2007.