Querying Databases with Taxonomies

Davide Martinenghi[1]and Riccardo Torlone[2]

[1] Dip. di Elettronica e Informazione
Politecnico di Milano, Italy
martinen@elet.polimi.it

[2] Dip. di Informatica e Automazione
Università Roma Tre, Italy
torlone@dia.uniroma3.it

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 an abstract query language as a foundation for querying data sets with vague schemas. Our approach takes advantages of 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. The query language is a conservative extension of relational algebra where special operators allow the specification of relaxed queries over vaguely structured information. We study equivalence and rewriting properties of the query language that can be used for query optimization.

1 Introduction

There are today many application scenarios in which user queries do not match the structure and the content of data repositories, given the nature of the application domain or just because the schema is not available. This happens for instance in location-based search (find an opera concert in Paris next summer), multifaceted 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, the query is usually relaxed to accommodate user's needs, and query answering relies on finding the best matching 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 [1]. 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.

J. Parsons et al. (Eds.): ER 2010, LNCS 6412, pp. 433–446, 2010.

To this aim, we propose in this paper a logical data model and an abstract query language supporting query relaxation over relational data. Our approach relies on 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 query language is called Taxonomy-based Relational Algebra (TRA) and 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.

In sum, the contributions of this paper are the following: (i) 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; (ii) a simple but powerful algebraic language for supporting query relaxation: the query language makes it possible to formulate complex searches over vague schemas in different application domains; (iii) 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.

Because of space limitation, we do not address the issue of implementing the formal framework proposed in this paper and we disregard the orthogonal problem of taxonomy design. Both issues will be addressed in forthcoming works.

2 A Data Model with Taxonomies

2.1 Partial Orders and Lattices

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 $b' \leq b$ ($b \leq b'$).

The *greatest lower bound* or *glb* (*least upper bound* or *lub*) is a lower bound (upper bound) b of two elements v_1 and v_2 in a poset $(V, \leq)$ such that $b' \leq b$ $(b \leq b')$ for any other lower bound (upper bound) b' of v_1 and v_2. It easily follows that if a lub (glb) exists, then it is unique. The glb and the lub are also called *meet* and *join*, respectively.

A *lattice* is a poset in which any two elements have both a glb and a lub. The glb and lub can also be defined over a set of elements. By induction, it follows that every non-empty finite subset of a lattice has a glb and a lub.

2.2 Hierarchical Domains 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 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). *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* CM *of* containment mappings $\text{CMAP}_{l_1}^{l_2} : M(l_1) \to M(l_2)$ *for each pair of levels $l_1 \leq_L l_2$ satisfying the following* consistency conditions*:*
  - *for each level l, the function CMAP_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: $\text{CMAP}_l^{l_2}(\text{CMAP}_{l_1}^l(m)) = \text{CMAP}_{l'}^{l_2}(\text{CMAP}_{l_1}^{l'}(m))$ for each member m of l_1.*

Example 1. 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 23/07/2010, which is mapped by the containment mappings to the member 07/2010 of the level month and to the member Summer of the level season.

As should be clear from Definition 1, in this paper we consider a general notion of taxonomy in which, whenever $l_1 \leq_L l_2$ for two levels in an h-domain, then the set of ground members for l_1 is contained in the set of ground members for l_2.

The following result can be easily shown.

Proposition 1. *The poset $(L, \leq_L)$ is a lattice and therefore every pair of levels l_1 and l_2 in L has both a glb and a lub.*

Actually, a partial order $\leq_M$ can also be defined on the members M of an h-domain h: it is induced by the containment 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) $\text{CMAP}_{l_1}^{l_2}(m_1) = m_2$.*

Example 2. Consider the h-domain of Example 1. Given the members $m_1 =$ 29/06/2010 and $m_2 =$ 23/08/2010 of the level day, $m_3 =$ 06/2010 and $m_4 =$ 08/2010 of the level month, $m_5 =$ 2Q 2010 and $m_6 =$ 3Q 2010 of the level quarter, $m_7 =$ 2010 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 $m_2 \leq_M m_8$.

Example 2 shows an interesting property: differently from the poset on the levels on an h-domain, the poset on the members of an h-domain is not a lattice in general. Consider for instance the members m_1 and m_2 of the example above: they have no lower bound, since their intersection is empty (more precisely, the intersection of the ground members that they represent is empty), and have two incomparable minimal upper bounds: 2010 at the year level and Summer at the season level. Indeed, it is possible to show that the poset $(M, \leq_M)$ can be converted into a lattice by adding to M all the elements of the *powerset* of the ground members (including the empty set, which would become the bottom level). This however would imply an explosion of the number of members and an unnatural representation of an h-domain.

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-relation). *Let H be a set of h-domains. We denote by $S = \{A_1 : l_1, \ldots, A_k : l_k\}$ a* t-schema *(schema over taxonomies), where each A_i is a distinct* attribute *name and each l_i is a level of some h-domain in H. A* t-tuple *t over a t-schema $S = \{A_1 : l_1, \ldots, A_k : l_k\}$ 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.*

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. As an example, a t-schema over the h-domains *time*, *location* and *weather conditions* can be the following: $S = \{Time : \text{day}, Location : \text{city}, Weather : \text{brief}\}$. A possible t-relation over this schema is the following:

	Time: day	Location: city	Weather: brief	
$r_1 =$	11/05/2010	Rome	Sunny	$t_{1,1}$
	24/04/2009	Milan	Cloudy	$t_{1,2}$
	24/07/2010	New York	Showers	$t_{1,3}$

Then we have: $t_{1,1}[\mathsf{Location{:}city}] =$ Rome.

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

Definition 4 (Poset on t-schemas). *Let S_1 and S_2 be t-schemas over a set of h-domains H_1 and H_2 respectively. We have that $S_1 \leq_S S_2$ if: (i) $H_2 \subseteq H_1$, and (ii) for each $A_i : l_i \in S_2$ there is an element $A_i : l_j \in S_1$ such that $l_j \leq_L l_i$.*

Definition 5 (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 6 (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 following t-relations:

$S_1 =$	Title:cultural-event	Author:artist	Time:day	Location:theater	
$r_1 =$	Romeo & Juliet	Prokofiev	13/04/2010	La Scala	$t_{1,1}$
	Carmen	Bizet	24/05/2010	Opéra Garnier	$t_{1,2}$
	Requiem	Verdi	28/03/2010	La Scala	$t_{1,3}$
	La bohème	Puccini	09/01/2010	Opéra Garnier	$t_{1,4}$

$S_2 =$	Title:event	Time:quarter	Location:city	
$r_2 =$	Concert	1Q 2010	Milan	$t_{2,1}$
	Ballet	2Q 2010	Milan	$t_{2,2}$
	Sport	3Q 2010	Rome	$t_{2,3}$
	Opera	2Q 2010	Paris	$t_{2,4}$

Then, 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 $r_1 \leq_r r_2$.

The same considerations done for the poset on levels also apply to the poset on t-schemas.

Proposition 2. *Let $\mathcal{S}$ be the set of all possible t-schemas over a set of h-domains H. Then, the poset $(\mathcal{S}, \leq_S)$ is a lattice.*

Conversely, the poset on t-relations is not a lattice in general since, it is easy to show that, given two t-relations, they can have more than one minimal upper bound (but necessarily at least one) as well as more than one maximal lower bound (possibly none).

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

3 Querying with Taxonomies

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.

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), and natural join ($\bowtie$). It also includes some variants of these operators that are obtained by combining them with the following two new operators.

Definition 7 (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:*

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

Definition 8 (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:*

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

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.

Example 5. Consider the t-relations r_1 and r_2 from Example 4. The result of $\hat{\varepsilon}_{\mathsf{theater}}^{\mathsf{city}}(r_1)$ is the following t-relation.

$S_3 =$	Title:cultural-event	Author:artist	Time:day	Location:theater	Location:city	
$r_3 =$	Romeo & Juliet	Prokofiev	13/04/2010	La Scala	Milan	$t_{3,1}$
	Carmen	Bizet	24/05/2010	Opéra Garnier	Paris	$t_{3,2}$
	Requiem	Verdi	28/03/2010	La Scala	Milan	$t_{3,3}$
	La bohème	Puccini	09/01/2010	Opéra Garnier	Paris	$t_{3,4}$

The result of $\check{\varepsilon}_{\mathsf{month}}^{\mathsf{quarter}}(r_2)$ is the following t-relation.

$S_4 =$	Title:event	Time:quarter	Location:city	Time:month	
$r_4 =$	Concert	1Q 2010	Milan	Jan 2010	$t_{4,1}$
	Concert	1Q 2010	Milan	Feb 2010	$t_{4,2}$
	Concert	1Q 2010	Milan	Mar 2010	$t_{4,3}$
	Sport	3Q 2010	Rome	Jul 2010	$t_{4,4}$
	Sport	3Q 2010	Rome	Aug 2010	$t_{4,5}$
	Sport	3Q 2010	Rome	Sep 2010	$t_{4,6}$
	...	...	...	...	

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

$$\hat{\sigma}_{A:l\,\theta\,m}(r) = \{t \in r \mid \text{CMAP}_l^{l'}(t[A:l])\,\theta\,m\}$$

Definition 10 (Downward selection). *Let r be a t-relation over S, A be an attribute in S defined over l, m be a member of l' with $l' \leq_L l$, and $\theta \in \{=,<,>,\leq,\geq,\neq\}$: the* downward selection *of r with respect to $A\,\theta\,m$ on level l, denoted by $\check{\sigma}_{A:l\,\theta\,m}(r)$, is the t-relation over S defined as follows:*

$$\check{\sigma}_{A:l\,\theta\,m}(r) = \{t \in r \mid \text{CMAP}_{l'}^{l}(m)\,\theta\,t[A:l]\}$$

In the following, we will often simply write $\hat{\sigma}_{A\,\theta\,m}$ and $\check{\sigma}_{A\,\theta\,m}$, 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 that: $\hat{\sigma}_{\mathsf{City}=\mathtt{Milan}}(r_1) = \{t_{1,1}, t_{1,3}\}$ and $\check{\sigma}_{\mathsf{Day}=\mathtt{13/03/2010}}(r_2) = \{t_{2,1}\}$.

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.

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

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

Finally, we introduce two new join operators. Their main purpose is to combine information stored at different levels of granularity.

Definition 11 (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:*

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

Definition 12 (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:*

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

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 and the following t-relation.

$S_5 =$	Company:airline-company	Location:airport	
$r_5 =$	Alitalia	Linate	$t_{5,1}$
	Air France	Roissy	$t_{5,2}$

The result of $r_1 \hat{\bowtie}_{\mathsf{city}} r_5$ is the following t-relation:

$S_6 =$	Event:cultural-event	Author:artist	Time:day	Location:theater	Company:airline-company	Location:airport	
$r_6 =$	Romeo & Juliet	Prokofiev	24/04/2010	La Scala	Alitalia	Linate	$t_{6,1}$
	Carmen	Bizet	24/05/2010	Opéra Garnier	Air France	Roissy	$t_{6,2}$
	Requiem	Verdi	24/03/2010	La Scala	Alitalia	Linate	$t_{6,3}$
	La bohème	Puccini	09/01/2010	Opéra Garnier	Air France	Roissy	$t_{6,4}$

Now, consider the following t-relations.

$S_7 =$	Loc:theater	Time:year	Price:money	
$r_7 =$	La Scala	2010	150	$t_{7,1}$

$S_8 =$	Loc:theater	Time:month	Discount:perc.	
$r_8 =$	La Scala	03/2010	10%	$t_{8,1}$
	La Scala	06/2010	20%	$t_{8,2}$

The result of $r_7 \check{\bowtie}_{\mathsf{theater,day}} r_8$ is the following t-relation:

$S_9 =$	Loc:theater	Time:year	Price:money	Time:month	Discount:perc.	
$r_9 =$	La Scala	2010	150	03/2010	10%	$t_{9,1}$
	La Scala	2010	150	06/2010	20%	$t_{9,2}$

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 $S = \{A^1 : l^1, \ldots, A^n : l^n\}$, $S_i \supseteq \bar{S}_i \supseteq \{A^1 : l^1_i, \ldots, A^n : l^n_i\}$ for $i = 1, 2$, and P is a predicate requiring pairwise equality in both sides of the join for all fields added by the extensions.

$$r_1 \hat{\bowtie}_{S:\bar{S}_1,\bar{S}_2} r_2 = \pi_{S_1 S_2}(\hat{\varepsilon}^{A^1:l^1}_{A^1:l^1_1} \cdots \hat{\varepsilon}^{A^n:l^n}_{A^n:l^n_1}(r_1) \bowtie_P \hat{\varepsilon}^{A^1:l^1}_{A^1:l^1_2} \cdots \hat{\varepsilon}^{A^n:l^n}_{A^n:l^n_2}(r_2)) \qquad (3)$$

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

$$r_1 \check{\bowtie}_{S:\bar{S}_1,\bar{S}_2} r_2 = \pi_{S_1 S_2}(\check{\varepsilon}^{A^1:l_1^1}_{A^1:l^1} \cdots \check{\varepsilon}^{A^n:l_1^n}_{A^n:l^n}(r_1) \bowtie_P \check{\varepsilon}^{A^1:l_2^1}_{A^1:l^1} \cdots \check{\varepsilon}^{A^n:l_2^n}_{A^n:l^n}(r_2)) \tag{4}$$

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 7-12 with a t-expression. Similar extensions are possible for other RA operators (e.g., difference); we omit them in the interest of space.

4 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.

4.1 Upward and Downward Extension

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

$$\hat{\varepsilon}_l^l(r) = \check{\varepsilon}_l^l(r) = r \tag{5}$$

Equivalence (5) 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 (5) follows immediately from Definitions 7 and 8.

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

$$\hat{\varepsilon}_l^{l'}(\hat{\varepsilon}_l^{l'}(r)) = \hat{\varepsilon}_l^{l'}(r) \tag{6}$$

$$\check{\varepsilon}_{l''}^{l}(\check{\varepsilon}_{l''}^{l}(r)) = \check{\varepsilon}_{l''}^{l}(r) \tag{7}$$

Equivalences (6) and (7) 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 7 and 8.

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

$$\hat{\varepsilon}^{l}_{l'}(\check{\varepsilon}^{l}_{l'}(r)) = \check{\varepsilon}^{l}_{l'}(r) \tag{8}$$

The above Equivalence (8) shows that an upward extension is always idle after a downward extension on the same levels. To prove (8), 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:

$$\hat{\varepsilon}^{l'_2}_{l_2}(\hat{\varepsilon}^{l'_1}_{l_1}(r)) = \hat{\varepsilon}^{l'_1}_{l_1}(\hat{\varepsilon}^{l'_2}_{l_2}(r)) \tag{9}$$

$$\check{\varepsilon}^{l_2}_{l''_2}(\check{\varepsilon}^{l_1}_{l''_1}(r)) = \check{\varepsilon}^{l_1}_{l''_1}(\check{\varepsilon}^{l_2}_{l''_2}(r)) \tag{10}$$

The above Equivalences (9) and (10) state that two extensions of the same kind can be swapped. Both follow straightforwardly from Definitions 7 and 8.

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:

$$\pi_{A_p} \hat{\varepsilon}^{A:l'_2}_{A:l}(r) = \pi_{A_p} \hat{\varepsilon}^{A:l'_2}_{A:l'_1}(\hat{\varepsilon}^{A:l'_1}_{A:l}(r)) \tag{11}$$

$$\pi_{A_p} \check{\varepsilon}^{A:l}_{A:l_2}(r) = \pi_{A_p} \check{\varepsilon}^{A:l_1}_{A:l_2}(\check{\varepsilon}^{A:l}_{A:l_1}(r)) \tag{12}$$

Note that the outerπ_{A_p} in Equivalence (11) 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 (12).

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:

$$\pi_{A_p}(\hat{\varepsilon}^{A:l'}_{A:l}(r)) = \hat{\varepsilon}^{A:l'}_{A:l}(\pi_{A_p}(r)) \tag{13}$$

$$\pi_{A_p}(\check{\varepsilon}^{A:l}_{A:l''}(r)) = \check{\varepsilon}^{A:l}_{A:l''}(\pi_{A_p}(r)) \tag{14}$$

Equivalences (13) and (14) show that, similarly to Equivalences (11) and (12), 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:

$$\sigma_P(\hat{\varepsilon}^{A:l'}_{A:l}(r)) = \hat{\varepsilon}^{A:l'}_{A:l}(\sigma_P(r)) \tag{15}$$

$$\sigma_P(\check{\varepsilon}^{A:l}_{A:l''}(r)) = \check{\varepsilon}^{A:l}_{A:l''}(\sigma_P(r)) \tag{16}$$

Equivalences (15) and (16) 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:

$$\hat{\varepsilon}^{A:l'}_{A:l}(r_1 \bowtie_P r_2) = (\hat{\varepsilon}^{A:l'}_{A:l}(r_1)) \bowtie_P r_2 \tag{17}$$

$$\check{\varepsilon}^{A:l}_{A:l''}(r_1 \bowtie_P r_2) = (\check{\varepsilon}^{A:l}_{A:l''}(r_1)) \bowtie_P r_2 \tag{18}$$

Equivalences (17) and (18) 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.)

4.2 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:

$$\hat{\sigma}_{A:l\,\theta\,m'}(\hat{\sigma}_{A:l\,\theta\,m'}(r)) = \hat{\sigma}_{A:l\,\theta\,m'}(r) \tag{19}$$

$$\check{\sigma}_{A:l\,\theta\,m''}(\check{\sigma}_{A:l\,\theta\,m''}(r)) = \check{\sigma}_{A:l\,\theta\,m''}(r) \tag{20}$$

Equivalences (19) and (20) state that repeated applications of the same t-selection are idle, both for the upward and for the downward case. To prove (19), consider that, by (1), the left-hand side of the equivalence can be written as:

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

where S is the schema of r. The innermost π_S can be moved outside the upward selection by using equivalence (13) if $l \neq l'$ or equivalence (5) 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:

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

Now, equivalence (15) 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 (15) can be used here to move the innermost selection outside the outermost $\hat{\varepsilon}$, although $A : l'\,\theta\,m'$ 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 (6)), we obtain

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

which, by (1), corresponds to the right-hand side of (19). Analogously for (20).

Commutativity

$$\hat{\sigma}_{l_2:A_2\,\theta_2\,m_2}(\hat{\sigma}_{l_1:A_1\,\theta_1\,m_1}(r)) = \hat{\sigma}_{l_1:A_1\,\theta_1\,m_1}(\hat{\sigma}_{l_2:A_2\,\theta_2\,m_2}(r)) \tag{21}$$

$$\check{\sigma}_{l_2:A_2\,\theta_2\,m_2}(\check{\sigma}_{l_1:A_1\,\theta_1\,m_1}(r)) = \check{\sigma}_{l_1:A_1\,\theta_1\,m_1}(\check{\sigma}_{l_2:A_2\,\theta_2\,m_2}(r)) \tag{22}$$

$$\check{\sigma}_{l_2:A_2\,\theta_2\,m_2}(\hat{\sigma}_{l_1:A_1\,\theta_1\,m_1}(r)) = \hat{\sigma}_{l_1:A_1\,\theta_1\,m_1}(\check{\sigma}_{l_2:A_2\,\theta_2\,m_2}(r)) \tag{23}$$

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 (23). The proof of these follows straightforwardly from commutativity of standard selection and interplay of extension and standard selection.

4.3 Upward and Downward Join

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:

$$\hat{\sigma}_{A:l\,\theta\,m}(r_1 \hat{\bowtie}_{C_{up}:C_1,C_2} r_2) = (\hat{\sigma}_{A:l\,\theta\,m} r_1) \hat{\bowtie}_{C_{up}:C_1,C_2} r_2 \tag{24}$$

$$\hat{\sigma}_{A:l\,\theta\,m}(r_1 \check{\bowtie}_{C_{low}:C_1,C_2} r_2) = (\hat{\sigma}_{A:l\,\theta\,m} r_1) \check{\bowtie}_{C_{low}:C_1,C_2} r_2 \tag{25}$$

$$\check{\sigma}_{A:l\,\theta\,m}(r_1 \hat{\bowtie}_{C_{up}:C_1,C_2} r_2) = (\check{\sigma}_{A:l\,\theta\,m} r_1) \hat{\bowtie}_{C_{up}:C_1,C_2} r_2 \tag{26}$$

$$\check{\sigma}_{A:l\,\theta\,m}(r_1 \check{\bowtie}_{C_{low}:C_1,C_2} r_2) = (\check{\sigma}_{A:l\,\theta\,m} r_1) \check{\bowtie}_{C_{low}:C_1,C_2} r_2 \tag{27}$$

The above equivalences (24)-(27) 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:

$$\pi_L(r_1 \hat{\bowtie}_{C_{up}:C_1,C_2} r_2) = \pi_L((\pi_{L_1} r_1) \hat{\bowtie}_{C_{up}:C_1,C_2} (\pi_{L_2} r_2)) \tag{28}$$

$$\pi_L(r_1 \check{\bowtie}_{C_{low}:C_1,C_2} r_2) = \pi_L((\pi_{L_1} r_1) \check{\bowtie}_{C_{low}:C_1,C_2} (\pi_{L_2} r_2)) \tag{29}$$

Equivalences (28) and (29) 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 (13) and (14)).

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

Theorem 1. *Equivalences (5)-(29) hold for any possible t-relation.*

Theorem 1 together with the fact that TRA is closed entails that equivalences (5)-(29) can also be used to test equivalence of complex t-expressions.

Finally, we observe that some applications of the TRA operators preserve partial order between relations. For instance, $r_1 \leq_r r_2$ entails (i) $(\hat{\varepsilon}_l^{l'}(r_1)) \leq_r r_2$, (ii) $(\check{\varepsilon}_{l'}^{l}(r_1)) \leq_r r_2$, and (iii) $r_1 \leq_r (\hat{\varepsilon}_l^{l'}(r_2))$, but not (iv) $r_1 \leq_r (\check{\varepsilon}_{l'}^{l}(r_2))$.

5 Related Work

The approach proposed in this paper, which extends to a more general scenario a work on modeling context-aware database applications [10], is focused on the relaxation of queries to a less restricted form with the goal of accommodating user's needs. This problem has been investigated in several research areas under different perspectives. In the database area, query relaxation has been addressed in the context of XML and semi-structured databases, with the goal of combining database style querying and keyword search [1] and for querying databases with natural language interfaces [9]. Query relaxation has also been addressed to avoid empty answers to complex queries by adjusting values occurring in selections and joins [8]. Malleable schemas [5,11] deals with vagueness and ambiguity in database querying by incorporating imprecise and overlapping definitions of data structures. An alternative formal framework relies on multi-structural databases [6], where data objects are segmented according to multiple distinct criteria in a lattice structure and queries are formulated in this structure. The majority of these approaches rely on non-traditional data models, whereas we refer on a simple extension of the relational model. Moreover, none of them consider relaxation via taxonomies, which is our concern. In addition, the systematic analysis of query equivalence for optimization purposes has never been studied in the relaxed case.

Query relaxation is also used in location-based search [4], 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 [3], where the authors consider the problem of fuzzy matching queries to items. Actually, in the information retrieval area, which is however clearly different from ours, document taxonomies have been already used in, e.g., [7], where the authors focus on classifying documents into taxonomy nodes and developing the scoring function to make the matching work well in practice, and in [2], where the authors propose a framework for relaxing user requests over ontologies, a notion that is more general than that of taxonomy.

6 Conclusion

In this paper, we have presented a logical model and an algebraic language as a foundation for querying 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. The 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 query language as a formal foundation for the optimization of taxonomy-based queries.

We believe that several interesting directions of research can be pursued within the framework presented in this paper. We are particularly interested into a deep investigation of general properties of the query language. In particular, we plan to develop methods for the automatic identification of the level in which two heterogeneous t-relations can be joined for integration purposes. Also, we are currently studying the impact of our model on the complexity of query answering. 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 quantitative analysis oriented to the optimization of relaxed queries. The equivalence results presented in this paper provide an important contribution in this direction.

Acknowledgments. D. Martinenghi acknowledges support from the Search Computing (SeCo) project, funded by the European Research Council (ERC).

References

1. Amer-Yahia, S., Lakshmanan, L.V.S., Pandit, S.: Flexpath: Flexible structure and full-text querying for XML. In: Proc. of SIGMOD, pp. 83–94 (2004)
2. Balke, W.-T., Wagner, M.: Through different eyes: assessing multiple conceptual views for querying web services. In: Proc. of WWW, pp. 196–205 (2004)
3. Broder, A.Z., Fontoura, M., Josifovski, V., Riedel, L.: A semantic approach to contextual advertising. In: Proc. of SIGIR, pp. 559–566 (2007)
4. Chen, Y.-Y., Suel, T., Markowetz, A.: Efficient query processing in geographic web search engines. In: Proc. of SIGMOD, pp. 277–288 (2006)
5. Dong, X., Halevy, A.Y.: Malleable schemas: A preliminary report. In: Proc. of WebDB, pp. 139–144 (2005)
6. Fagin, R., Guha, R.V., Kumar, R., Novak, J., Sivakumar, D., Tomkins, A.: Multistructural databases. In: Proc. of PODS, pp. 184–195 (2005)
7. Fontoura, M., Josifovski, V., Kumar, R., Olston, C., Tomkins, A., Vassilvitskii, S.: Relaxation in text search using taxonomies. In: Proc. of VLDB, vol. 1(1), pp. 672–683 (2008)
8. Koudas, N., Li, C., Tung, A.K.H., Vernica, R.: Relaxing join and selection queries. In: Proc. of VLDB, pp. 199–210 (2006)
9. Li, Y., Yang, H., Jagadish, H.V.: NaLIX: A generic natural language search environment for XML data. TODS, art. 30 32(4) (2007)
10. Martinenghi, D., Torlone, R.: Querying Context-Aware Databases. In: Proc. of FQAS, pp. 76–87 (2009)
11. Zhou, X., Gaugaz, J., Balke, W., Nejdl, W.: Query relaxation using malleable schemas. In: Proc. of SIGMOD, pp. 545–556 (2007)