Adaptive Evaluation Techniques for Querying XML-based E-Catalogs

Georg Lausen and Pedro José Marrón
Universität Freiburg, Institut für Informatik
Georges-Koehler-Allee, Geb. 51, 79110 Freiburg, Germany
{lausen,pjmarron}@informatik.uni-freiburg.de

Abstract

The integration of electronic catalogs (eCatalogs) is one of the most important aspects for the successful deployment of electronic commerce systems, since they are usually the only communication channel between buyers and suppliers. In this paper, we propose an XML-based global eCatalog integration platform whose query model allows us to avoid the costly problem of finding rewritings for each local eCatalog in the system, while at the same time, providing reliable answers to a wide range of XPath queries. Our implementation relies on the following characteristics to achieve its goals: the intrinsic properties of the XPath model; the applicability and efficiency of an extensible fitness function used to evaluate each answer; and the hierarchical nature of product catalogs.

Keywords: *E-Commerce, E-Catalog, XML, XPath, LDAP.*

1 Introduction

Electronic Catalogs (eCatalogs) are a crucial component for Electronic Commerce (eCommerce) on the Internet, since they list, describe and, more importantly, categorize the kinds of products and services suppliers offer to their customers. ECatalogs are, therefore, the main communication channel between buyers and suppliers, only surpassed in importance by the proper and correct administration of diverse groups of catalogs from different suppliers, whose transparent integration constitutes one of the key aspects of the successful deployment of electronic commerce nowadays.

Various forms of catalog integration have been proposed in the literature [4, 1, 7] and some of them are available as commercial products, but whether they propose a centralized integrated platform as their solution, or merely a means to establish a direct channel between buyers and suppliers, the importance of projecting a uniform, integrated image to the potential customers is crucial for electronic commerce [6].

In this paper, we propose an eCatalog integration platform that can be considered similar in spirit to the *local-as-view* schema, where queries are formulated following a predefined global catalog (gCatalog) and forwarded to the appropriate local catalogs for evaluation. As opposed to the classical view approach to catalog integration [3], our methodology does not rely on finding rewritings to perform individual queries on local catalogs, but in an adaptive query evaluation strategy that allows us to perform the same query on all local catalogs and still obtain reliable answers. The gist of our approach relies on three main factors:

1. The intrinsic properties of XPath queries over XML-based catalogs;
2. the flexibility of a fitness function that allows us to discriminate more accurate solutions from others; and
3. the nature of the conceptual hierarchy on which product catalogs are based.

The combination of these factors allows us to perform the evaluation of XPath queries on top of XML-based catalogs in a very efficient way.

The rest of this paper is organized as follows. Section 2 briefly describes the XPath and XML models and introduces the running example we will use throughout this paper. Section 3 explains the details of our adaptive query evaluation model and how it can be used to provide catalog integration capabilities, leaving the discussion of the advantages and limitations of our approach, as well as some appropriate possible solutions for such disadvantages for section 4. Finally, section 5 concludes this paper.

2 XML, XPath and eCatalogs

Consider the scenario, originally described in [8], where a group of electronics companies decide to offer their products on-line by means of a common, integrated site. The differences in the range and type of products they sell, are reflected in their individual local catalogs, but they manage to agree on a common global catalog to implement on the site. Company SESP [2], for example, only sells mobile phone jammers, as can be seen in its local catalog (left-hand side of figure 1), whereas company BIGGER contains a richer variety of products ranging from computers, to phone jammers, etc. (right-hand side of figure 1).

After several meetings, they agree that the global catalog should contain a combination of the individual concepts (represented by XML tags) that provide a generic representation of their business model. They agree that they have *products*, organized in different *departments*, and each product may contain internally a description of its features in terms of *company* names, product *name*s, *price*, etc. Figure 2 contains the hierarchical representation of the concepts found in their local catalogs, and figure 3 represents the global catalog.

If we were to use the classic XPath model to perform a query on the global catalog, we would need to know how to rewrite this query in terms of the local catalogs so that we can obtain the desired answers.

Example Let the query $Q_{global} =$

$$/department/mobile/products/jammer[price < 200]$$

be a valid XPath query performed on the global catalog to request all jammers in the system whose price lies below 200 EUR. Its evaluation using the classic approach involves the creation of two equivalent queries Q_{SESP} and Q_{BIGGER} and their evaluation in their respective local catalogs, collecting the answers for further presentation to the user.

Given the catalogs in figures 2 and 3, we cannot simply forward $Q_{SESP} = Q_{BIGGER} = Q_{global}$ to SESP and BIGGER, because SESP does not contain a *department* node and BIGGER does not have a *products* node. Therefore, we need to make $Q_{SESP} = /products/jammer[price < 200]$ and $Q_{BIGGER} = /department/mobile/jammer[price < 200]$ if we want to obtain an answer. This is the classic problem of query rewriting in mediator systems that, in the semistructured world, would imply analyzing the current query and adapt it to either a DTD that describes the contents of each local data repository, or to the data itself, if the DTD is non-existent. □

The reason we need to provide a rewriting in the previous example lies on the XPath query evaluation model, where an XPath query Q is formed by the concatenation of path expressions that perform walk-like operations on the document tree, retrieving a set of nodes that conform to the requirements of the query. Each expression is joined with the next by means of the character '/'.

A formal characterization of this model, originally proposed in [9], is as follows:

Definition (XPath Query) An XPath Query Q_X is defined as:

$$Q_X = q_0/q_1/\ldots/q_n$$

where q_i is an XPath subquery defined below, and '/' the XPath subquery separator. □

Furthermore,

Definition (XPath Subquery) An XPath Subquery q_i is a 3-tuple

$$q_i = (C_i, w_i, C_{i+1})$$

where:

- C_i is a set of XML nodes that determine the input context;
- w_i is the Path Expression to be applied to each node of the input context (defined below); and
- C_{i+1} is a set of XML nodes resulting from the application of the path expression w_i onto the input context C_i. C_{i+1} is also called the output context. □

Finally,

Definition (XPath Path Expression) A Path Expression w_i is a 3-tuple

$$w_i = a_i :: e_i[c_i]$$

such that:

- a_i is an axis along which the navigation of the path expression takes place (see table 1 for a complete description)[1];
- e_i is a node expression that tests either the name of the node or its content type; and
- c_i is a boolean expression of conditional predicates that must be fulfilled by all nodes along the path. □

Example The XPath query $Q_X = /child :: mondial/child :: country[attribute :: car\_code = \text{“}D\text{”}]$ is composed of two subqueries whose combination selects all country nodes directly connected to the mondial node having an attribute car_code with value “D”. □

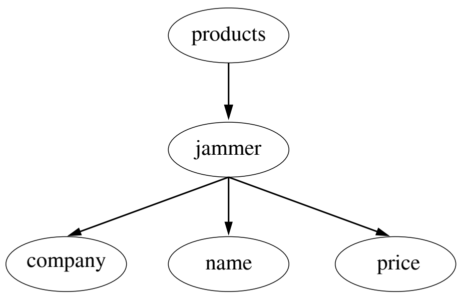
[1]Only the child and ancestor axis are considered in our adaptive evaluation algorithm

```
<products>
  <jammer>
    <company>SESP</company>
    <name>VHP Jammer </name>
    <price><onrequest/></price>
    <case><type>Mobile Attache Case
    </type></case>
  </jammer>

  <jammer>
    <company>SESP</company>
    <name>Full Milspec. Portable
          HP Jammer</name>
    <price><onrequest/></price>
    <case><type>Rugged military
    </type></case>
    <booster><range>1km</range>
    </booster>
    <supplement>39</supplement>
  </jammer>
  ...
```

```
<department>
  <mobile>
    <jammer>
      <name>Static HP Jammer</name>
      <company>BIGGER</company>
      <price>250 EUR</price>
    </jammer>
    <jammer>
      <name>Personal Jammer</name>
      <company>JamLogic</company>
      <price><onrequest/></price>
    </jammer>...
  </mobile>
  <computing>
    <computer>
      <make>Dell Infinion</make>
      <cpu>Pentium 4</cpu>
      <memory>128 MB</memory>...
    </computer>...
  </computing>
  ...
```

Figure 1. Local Catalogs for companies SESP **(left) and** BIGGER **(right)**



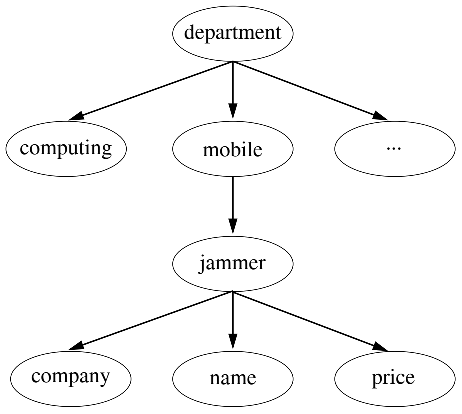


Figure 2. Concept hierarchy for SESP **(left), and** BIGGER **(right)**

The usual processing methodology of XPath queries implies the evaluation of each subquery q_i on each element of its corresponding input context C_i in order to generate each element of the output context C_{i+1}. However, as a special case, if the path expression w_i of a given subquery q_i is empty, represented as $w_i = \varepsilon$, the evaluation of $q_i = (C_i, \varepsilon, C_{i+1})$ is simply performed by assigning the output context to the contents of the input context: $C_{i+1} = C_i$.

Figure 4 contains a graphical representation of the evaluation process of a query consisting of six subqueries. The ovals inside each context between two subqueries indicate the individual XML nodes that satisfy the subquery at each point.

Therefore, given the serial evaluation model of XPath and the structure of the SESP local catalog that lacks the *department* and *mobile* nodes, the execution of Q_{global} with no modifications onto the SESP local catalog produces no answers, since the output context C_1 of the $/department/$ subquery is empty. The same problem appears querying BIGGER.

3 Adaptive Query Evaluation Strategy

Our adaptive XPath query evaluation strategy has been designed with the purpose of avoiding the need for the kind of rewriting mechanisms detailed in the previous example,

Axis Name	Considered Nodes
ancestor	Any node along the path to the root
ancestor-or-self	Same, but including the current node
attribute	Consider only attribute nodes in the tree
child	Any node directly connected to the current node
descendant	Any node from the subtree rooted at the current node
descendant-or-self	Same, but including the current node
following	Any node with id greater than the current node, excluding its descendents
following-sibling	Any same-level node with id greater than the current node
parent	The direct predecessor of the current node
preceding	Any node with id lower than the current node, excluding its ancestors
preceding-sibling	Any same-level node with id lower than the current node
self	The current node

Table 1. Allowed Axis Expressions in XPath

while at the same time, obtaining the set of answers we would retrieve had we performed the appropriate rewriting.

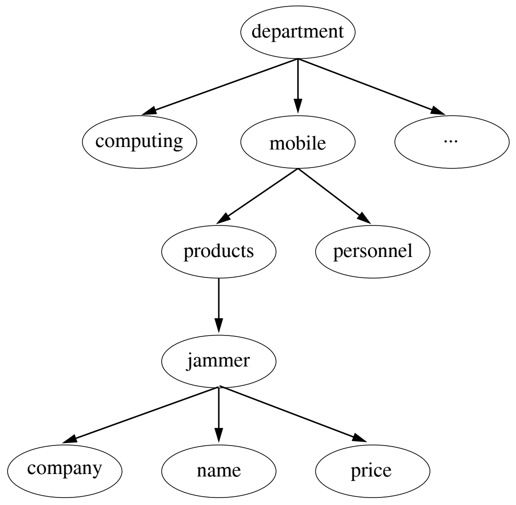


Figure 3. Concept hierarchy: global catalog

We assume that each local catalog provides a mapping between the concepts found in the glogal catalog and its own local representation, so that discrepancies that might appear as a result of using synonyms, or other languages for the same concepts are easily solved. The benefit of such a mapping mechanism is manyfold:

- Each concept can be translated independently of its location in the submitted query, so that no structural reorganization needs to be performed.
- Each local catalog may decide entirely on its own how to direct a specific concept or query inside its catalog, allowing the support of synonyms or other languages simply by the implementation of a dictionary-like system.

For example, if the `BIGGER` catalog in figure 2 had a *portable* node, instead of *mobile*, but wanted that all queries directed towards *mobile* products go through the *portable* node, it would only have to replace every occurrence of *mobile* with *portable* without any further structural or semantical consideration.

The gist of our strategy is the application and evaluation of three possible subquery transformations on each of the subqueries in the original XPath query, and on the metric provided by a fitness function for every possible answer to enable the discrimination of one set of solutions over another.

Definition (Subquery Transformations) The three possible transformations performed on a given subquery by our adaptive query evaluation algorithm are:

No transformation: Where the query is evaluated as it was originally specified by the user;

Subquery generalization: Where the axis of a particular subquery is augmented according to the contents of table 2;

Subquery elimination: Where a subquery is eliminated from the original XPath query if its result set (output context) is empty, allowing for the further evaluation of the following queries as if it had never been on the original query in the first place. □

Predicates that appear as part of a subquery are evaluated in the normal way and are, therefore, not subject to the transformations we just described. We further assume that predicates contain only simple path expressions that

Original Axis	Augmented Axis
`child`	`descendent`
`parent`	`ancestor`

Table 2. Subquery Generalization Strategy

test properties and attributes of a node that, following the DOM model [11], are stored as direct descendents of such elements. We could, of course, allow the adaptive search of path expressions in predicates, but then we need to define what it means for a predicate match to happen after a generalization, which could lead to inconsistencies in the result set. For this reason, and taking into account that the purpose of the use of predicates and the above mentioned transformations is opposite to each other, we have opted to only allow simple path expressions at the predicate level.

The difficulty in the evaluation process lies in determining the correct sequence of transformations to apply at each subquery so that accurate results are produced. In order to do this, given a subquery q_i with input context C_i, we evaluate it with respect to each subquery transformation, so that from each input context C_i, we generate three output contexts: C^n_{i+1}, C^g_{i+1} and C^e_{i+1} that correspond, respectively, to the application of the *no transformation*, *subquery generalization* and *subquery elimination* transformations defined above.

Following this technique, each context $C_{i+1} = C^n_{i+1} \cup C^g_{i+1} \cup C^e_{i+1}$ is formed by the union of the result contexts obtained by the application of each one of the three transformations. Figure 5 contains a graphical representation of the evaluation of a query with six subqueries using our technique. The dark-shaded nodes are the ones generated by the evaluation of a subquery as such, the light-shaded ones by the generalized subquery, and the white ones by the elimination of the subquery.

The purpose of the fitness function, defined below, is to assign a metric to each one of the nodes generated in the process above so that we can distinguish which one of the transformations provides us with more accurate results.

Definition (Fitness function) Let q_i be the current subquery to be evaluated, and C_i its associated input context. Each node $n \in C_i$ is augmented to have a value v_n resulting from the evaluation of the previous subqueries $q_0 \ldots q_{i-1}$. We already know that the new context $C_{i+1} = C^n_{i+1} \cup C^g_{i+1} \cup C^e_{i+1}$ is composed by the union of the independent result sets of the three possible evaluation techniques. Furthermore, we know that each new node in C_{i+1} is the result of the evaluation of q_i on a node $n \in C_i$.

Then, the fitness function assigns a value v_m to each new node in $m \in C_{i+1}$ generated by a node $n \in C_i$ as follows:

- if $m \in C^n_{i+1}$, then $v_m = b_l^2 + v_n$;
- if $m \in C^g_{i+1}$, then $v_m = b_l + v_n$; and
- if $m \in C^e_{i+1}$, then $v_m = 1 + v_n$.

where $b_l > 0$ is an integer that represents the base of the fitness function at a given level in the computation.

If a node n has been generated by the application of more than one strategy, for example, $n \in C^n_{i+1}$ and $n \in C^g_{i+1}$, the final context C_{i+1} only contains the instance of n with the biggest value v_n. Originally, the value of the root node is initialized to 1: $v_{root} = 1$. □

Of course, this definition is simply one of many possible implementations of a strictly monotonic function, whose purpose is the correct ranking of intermediate and final results generated by the evaluation procedure. A more involved representation of such a fitness function where the term b_l becomes a function of the current node $b_l = f(m)$, as opposed to a function of the current level in the evaluation procedure, would allow a specific local catalog to direct the query towards certain nodes and to rank "weekly specials" higher simply by selecting the appropriate values for those particular nodes.

As can be deduced from the definition of the fitness function, nodes generated as a result of the evaluation of a subquery in its original state have a higher fitness value than those generated by the generalization of the subquery, or those obtained by its elimination. The reason behind this choice is to favor the original query over transformation operations that might abstract away the intention of the user. Therefore, the elimination of a subquery is valued as the worst possible alternative in the series of transformations because if a particular subquery has been specified, it should be used before it is tossed away. Otherwise, the empty query would always have a good chance of becoming part of the answer set, which makes no real sense. The final result of the query is, therefore, the node or set of nodes whose score at the end of the evaluation process is highest.

Using this definition, our original example is evaluated as follows:

Example In order to evaluate $Q_{global} =$

$$/department/mobile/products/jammer[price < 200]$$

using the technique just explained, the two local catalogs `SESP` and `BIGGER` would have to perform the following operations:

Assuming that $root$ denotes the root node of the `SESP` catalog, the first step is to set Q_{SESP} to Q_{global} without performing any transformation on it. Then, $q_0 = /department$ is evaluated on the initial input context $C_0 = \{root\}$, that contains only the root node, yielding $C^n_1 = \emptyset$, since there are no $department$ nodes as direct children from the root. The next step is to evaluate the generalized

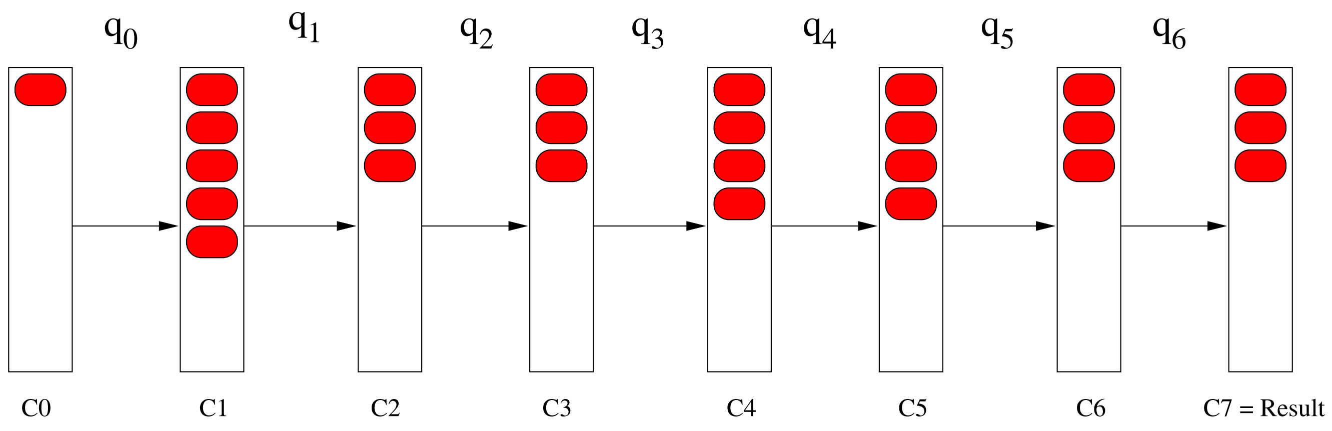


Figure 4. Example of a classic XPath query evaluation

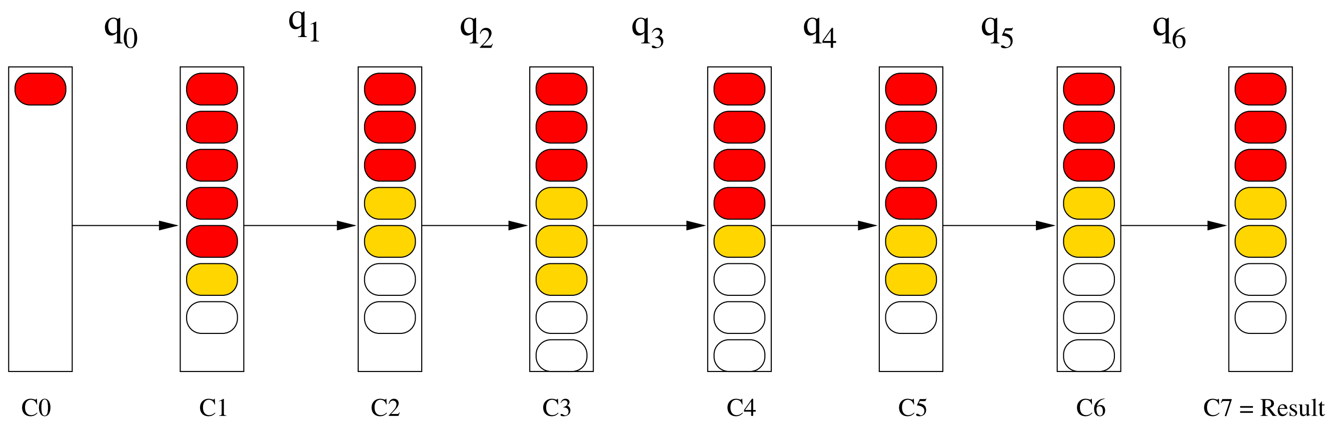


Figure 5. Example of our adaptive query evaluation technique

subquery $q_0 = //department$ on C_0, that also produces no results $C_1^g = \emptyset$, since there are no $department$ nodes anywhere in the SESP catalog. Finally, $q_0 = \varepsilon$ is evaluated on C_0, generating $C_1^e = \{root\}$, where $v_{root} = 2$ at this point. The same exact process happens in the evaluation of $q_1 = /mobile$ since there are no $mobile$ nodes in the catalog, thus producing $C_2 = \{root\}$, with $v_{root} = 3$. The evaluation of $q_2 = /products$, on the other hand, has a non-empty C_3^n, since as it can be seen in the catalog of figure 1, $products$ is a direct child of the root. The value of each node $n \in C_3^n$ is, according to our fitness function definition, $v_n = 10^2 + 3 = 103$, assuming that $b_l = 10$ at all levels and that, as it is the case, all nodes in C_3^n come from the only ($root$) node in C_2. The evaluation of the generalized query $q_2 = //products$ produces $C_3^g = C_3^n$, that is, since there are no other nodes in the catalog, the descendents of the root node are just its direct children. Furthermore, since the value of the nodes in C_3^n is greater than that of the nodes in C_3^g, we merge the result set to only contain the one instance of a given node with the bigger value. Finally, the evaluation of $q_2 = \varepsilon$ yields $C_3^e = \{root\}$, where $v_{root} = 4$.

The evaluation of $q_3 = /jammer[price < 200]$ uses C_3 as its input context, where some of the nodes, namely those that have satisfied the previous subquery, contain a higher value than the $root$ node, for example, that has been generated simply by the elimination of all previous subqueries $q_0 \ldots q_2$. Therefore, C_4^n will be composed of the direct $jammer$ children from C_3 whose price is less than 200 EUR. So if $n \in C_4^n$ and its preceding node $m \in C_3^n$, as it will probably be the case, the value of n is: $v_n = 10^2 + 103 = 203$. On the other hand, if a node n is generated by the $root$, whose value is $v_{root} = 4$, as we have seen, its new value would be $v_n = 10^2 + 4 = 104$.

At the end of this process, the result of the query is represented by the group of nodes in C_4 whose value is greatest. It is not very hard to see from the example that the best obtained value for each of the nodes in C_4 correspond to the nodes generated by the query $Q_{SESP} = /products/jammer[price < 200]$, which is exactly the optimal rewriting obtained intuitively by inspection of the local SESP catalog.

The processing and evaluation of Q_{BIGGER} is analogous. □

4 Properties of our Approach

The simplicity of our approach is both, its main strength and its major limitation, since the structural complexity of catalogs in the real world may lead our algorithm to produce incorrect results. Nevertheless, the advantages of using the system we just detailed outweigh the possible limitations, as can be seen in the next sections.

4.1 Advantages

Due to space constraints, we cannot develop each advantage of our approach in detail in this paper, but the following list gives an overview of the benefits the deployment of such a system as the core technology of an integrated electronic catalog system would produce.

- A rewriting for a query formulated agains a global catalog does not need to take into account the structural differences present in each local catalog. Therefore, the evaluation of queries is independent of the number of local catalogs registered in the system, making our approach efficient and, more importantly, scalable to any number of local catalogs.

- By selecting an appropriate fitness function and a concept translation mechanism, each local catalog can tweak a specific query to produce different results depending on the current state of their catalog. This property enables local catalogs to very easily implement strategies that direct customers to special offers, or new products.

- Each local catalog is able to perform the evaluation of the query in parallel, thus benefiting from a distributed architecture.

- The use of a base for the exponential part of the fitness function that depends on the level at a given catalog, allows for the flexible tailoring of the query evaluation process at the local catalogs in an independent fashion.

- If a group of local catalogs agree on the parameters of the fitness function, a homogeneous ranking for the evaluation of independent answers is achieved at no extra cost, directing the user naturally to the more accurate answers of her query.

- A number of semantic relationships among related elements in the catalog can be implemented by means of the fitness function. If, for example, the concept *accessories* should always be related to its parent category *CPU*, the local catalog could select a relatively small value for the *accessories* node, so that *accessories* would only be selected as part of the result set if *CPU* is also in the set.

- The use of the fitness function and our evaluation strategy enables the efficient search of the right combination of transformation operations without incurring in the exponential overhead typical of such a search, where at each step of the computation, one out of three choices needs to be made. By means of the result unification process where only one instance of a given node with the highest score is maintained as part of the result set, we are able to reduce an exponential search problem ($O(3^n)$, where n is the number of subqueries) into a linear one ($O(3 \cdot n)$).

  Although it could be possible to conceive a general evaluation strategy where, for example, all queries are generalized and if no result is found, eliminated, in order to avoid the cost of searching for a better strategy, general strategies are likely to produce inferior results, since they only explore a subset of the solution space, and might skip nodes that are important for the correct evaluation of the query.

- The nature and efficiency of our adaptive evaluation strategy eliminates the need to check each query against one or more DTDs that might describe the contents of each local catalog, which leads to further advantages:

  1. DTDs are no longer indispensable, allowing the integration of "legacy" catalogs that do not have a DTD;
  2. local catalogs can be maintained more efficiently without the need to check each new addition to the catalog against a DTD; and
  3. different document structures may coexist in the same catalog without incurring in the additional overhead needed to reformulate each query in the terms of each DTD.

Finally, it is worth mentioning that our algorithms have been implemented on top of an LDAP-based XML processing system [9, 10, 5] producing quite promising results.

4.2 Limitations and Possible Solutions

The kinds of limitations found in our approach are particularly related to the simplicity of our algorithms, as we have mentioned above. It is exactly for this reason that our adaptive evaluation procedure cannot guarantee the optimal rewriting of a particular query under all conditions, producing under some circumstances less than optimal results, or results that belong to different categories that those meant by the user. The two other key factors that contribute to this problem are:

1. The quality of the fitness function used for the adaptive evaluation is a critical factor in the process that may lead to unexpected results if not configured properly.

2. Structural differences between the global and a particular local catalog can also affect negatively the quality of results produced by our approach.

As an example of the first limitation, suppose that our adaptive evaluation algorithm is instructed to search for *CPU accessories* on a series of local catalogs by issuing the global query: $Q_{global} = /hardware/CPU/accessories$. Let us assume that one of the local catalogs (catalog A) has no explicit category for CPU accessories and that all the CPU related products are listed under $/hardware/CPU$. Let us further assume that another local catalog (catalog B) only contains printer articles and printer accessories, where $/hardware/printers/accessories$ is part of the catalog specification.

The application of the subquery generalization and elimination transformations on each one of these catalogs produces the following two syntactically correct queries: $Q_A = /hardware/CPU$ and $Q_B = /hardware//accessories$, but although Q_A generates the desired result, Q_B returns accessories that have nothing to do with CPU articles. This is in itself not a problem, since our algorithm has found the best possible match for a query that has no correct answer in catalog B. However, even if both catalogs use the fitness function previously described in this paper, the printer accessories of catalog B might receive a higher fitness value than the CPU articles of catalog A.

In order to avoid such semantic inconsistencies, it is necessary to extend the fitness function in the way described in section 3 to allow for the specification of a function value at each node that depends not only on the level it is located with respect to the query, but on the characteristics of the current node.

A fitness function of the form defined above, where $b_l = f(m)$ and m is the current node, could assign *accessories* nodes a very small value compared to their precedessors (in this case *printers* nodes) to convey the semantic relationship between both concepts. In other words, the presence of an *accessories* node in the result set should be almost negligible unless it appears in combination with its parent node.

As for the second inherent limitation of our approach, because of the conceptual simplicity, it is obvious that our algorithm will not be able to capture every structural difference between arbitrary global and local catalogs. We assume that notational or semantic differences that appear as a result of using synonyms, or equivalent names for the same kind of concepts is solved by means of a dictionary-type structure acting on the global query. However, there are other types of structural differences that appear at the tree level as a consecuence of modeling decissions taken by the maintainers of a particular catalog that cannot be so easily solved.

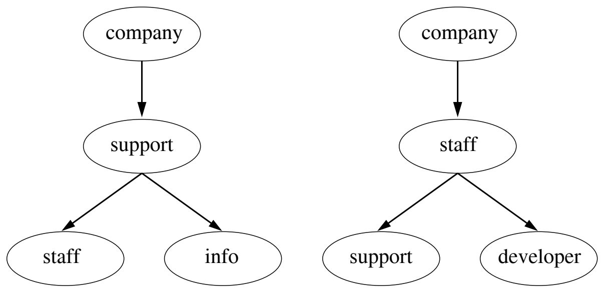


Figure 6. Organization of Company A and B

Figure 6 shows the concept hierarchy of the organization of two fictitional companies. Both organizational representations are correct in themselves, but compared to each other, the nodes *support* and *staff* play completely different roles in each representation. While in the first description the *staff* node is a child of the *support* node, in the second, their relationship is reversed. In the first case, *support* is an aggregation of *staff* and *info*, whereas in the second case *staff* is a generalization of *support* and *developer*.

Such structural discrepancies could arise between the global catalog and one or more local catalogs, which would lead to non-optimal answers from our adaptive algorithm. The bigger problem by such queries is that it is not even clear what the correct answer should be. For example, the query $Q_{global} = /company/support/staff$ would produce the desired result in the first “catalog” in figure 6, returning information about the *staff* nodes, whereas the second catalog generates *staff* nodes by eliminating $/support$ from the query and evaluating $Q_B = /company/staff$, and *support* nodes by generalizing $/support$ to $//support$ and thus evaluating $Q_B = /company//support$. What specific set of nodes has been assigned a bigger value by the fitness function depends on its specific parameters and could lead to the return of *support* nodes as a first fit, where, in reality, the user wanted information about *staff*.

Assume we adapted the fitness function by assigning subquery elimination a higher value than subquery generalization. Then *staff* nodes will take precedence over *support* nodes. An interesting question is then, what will happen with a possible continuation of the query, e.g. of the form $/company/support/staff/q_1/\ldots/q_n$. Since the fitness function uses only local information, each node in the output context of $/company//support/staff/q_1$ and $/company/staff/q_1$ will have the same values with respect to $q_1/\ldots/q_n$. In the first case, the elimination of the $/staff$ query will provide us with the expected result, while in the second case the generalization of $/q_1$ to $//q_1$ will do the job.

Another potential problem deals with discrepancies between information stored as either an element or an attribute node in the XML description of a catalog. The same concept can appear as an element in the global catalog and as an attribute in one of the local catalogs. The main problem by such a configuration is that, by definition [11], attributes have no descendants. Therefore, even if we extended our adaptive algorithm to perform subquery generalization and elimination on attributes, instead of just elements, no further searching of the catalog would stem from attribute nodes.

A simple solution to this problem, and one that we are currently in the process of implementing and evaluating, would be to perform an "upgrade" transformation on the corresponding attribute node a to allow for further processing. Such an operation would include the parent of the current attribute node in the intermediate result set C_{i+1} of a given subquery q_i if, searching for concept c, a contains the desired concept c either as part of its name or its value.

Using this technique, it is possible to perform the following kinds of transformations:

- $a/b/c/ \longrightarrow /a[d = \text{“}b\text{”}]/c$; or
- $a/b/c/ \longrightarrow /a[b = \text{“}*\text{”}]/c$,

automatically and with negligible cost for the adaptive procedure described in previous sections, since the application of a conditional transformation at each intermediate step does not significantly slow our algorithm down.

The integration of the "upgrade" operation into our overall procedure, as well as an analysis of the quality of its results is part of the kind of future work planned on the system.

5 Conclusion

In this paper we have introduced the concept of adaptive XPath evaluation that allow us to query XML-based catalogs following the classical *local-as-view* approach, without incurring in the costly query rewriting techniques needed thus far in order to obtain a reasonable answer to a query formulated on top of the global catalog.

Our techniques are based on the application of two non-trivial transformations on the original query: *subquery generalization* and *subquery elimination*, and on the properties of a fitness evaluation function that allows us to distinguish what answers correspond to the best possible rewriting of a given query. The main advantage of our approach lies in the fact that it eliminates the need to provide an a-priori rewriting of the global query before it can be evaluated by each local catalog, thus speeding up the evaluation process. Additionally, there is only a linear overhead in terms of the work each local catalog needs to perform in order to provide an answer to the query, as opposed to the more naive approach where the search for the optimal strategy would be exponential.

Finally, the simplicity of our procedure allows us to easily extend our algorithms to incorporate other transformations that might prove benefitial for our purposes.

References

[1] A. Corporation. *cXML User's Guide*, 2000. White Paper.

[2] S. Corporation. `http://sesp.co.uk/4.htm`.

[3] D. Florescu, A. Levy, and A. Mendelzon. Database techniques for the world-wide-web: A survey. *Sigmod Record*, 27(3), September 1998.

[4] J. Hellerstein. *Technical Requirements for Production-Level B2B E-Catalogs*. Cohera Corporation, 2000. White Paper.

[5] T. A. Howes, M. C. Smith, and G. S. Good. *Understanding and Deploying LDAP Directory Services*. Macmillan Network Architecture and Development. Macmillan Technical Publishing U.S.A., 1999.

[6] A. Jhingran. Moving up the food chain: Supporting e-commerce applications on databases. *Sigmod Record*, 29(4), December 2000.

[7] A. Keller. *Readings in Electronic Commerce*, chapter Smart Catalogs and Virtual Catalogs. Addison Wesley, 1997.

[8] G. M. Kuper and J. Siméon. Subsumption for XML types. In *Proceedings of the 8th International Conference on Database Theory (ICDT)*, pages 331–345, London, UK, January 2001.

[9] P. J. Marrón and G. Lausen. On processing XML in LDAP. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB)*, pages 601–610, Rome, Italy, September 2001. Morgan Kaufmann.

[10] M. Wahl, T. Howes, and S. Kille. Lightweight directory access protocol (v3). RFC 2251, December 1997.

[11] L. Wood, A. L. Hors, V. Apparao, S. Byrne, M. Champion, S. Isaacs, G. Nicol, J. Robie, R. Sutor, and C. Wilson. Document object model (DOM) level 1 specification (second edition). `http://www.w3.org/TR/2000/WD-DOM-Level-1-20000929/`, September 2000.