Catalog Integration for Electronic Commerce through Category-Hierarchy Merging Technique

Dongkyu Kim, Jaebum Kim, Sang-goo Lee
School of Computer Science and Engineering
Seoul National University, ENG4190
Shillimdong Kwanakgu, Seoul, Korea
{dkkim, jbkim, sglee}@europa.snu.ac.kr

Abstract

Internet marketplaces are now faced with new challenges that arise from the need to seamlessly integrate enormous number of product catalogs from different sources. In order to help users find products efficiently, Internet shops provide hierarchies of product catalogs (called category hierarchies). However, the absence of robust models (and well understood semantics) for product catalogs and their hierarchies severely impairs our ability to systematically support structured operations.

In this paper, we present an extended catalog model whose purpose is to provide a universal product catalog repository with browsing, searching, and downloading capabilities in order to facilitate catalog sharing and interoperability. In addition, we present a model for category hierarchies with well defined semantics which allows flexible representation of product hierarchies as well as automatic merging of different category hierarchies as a special case of catalog integration.

1. Introduction

The Internet is no longer an academic and research oriented network but an open public network with endless commercial opportunities. The exciting environment of electronic commerce (or e-commerce, for short) on this new mode of Internet encompasses a broad range of interaction processes among the various market participants; order, transport and delivery, mediation, invoice and payment, etc. These processes can be grouped into three basic phases of a commercial transaction: information, negotiation, and settlement [1]. The information phase of a typical e-commerce transaction is where suppliers and consumers gather information and explore potential market partners for goods and services. Electronic catalogs (or e-catalogs), like their printed counterparts, hold information about the goods and services offered or requested by the participants and, consequently, form the basis of this phase of a transaction. Certain parts of the catalogs, such as pricing and terms and conditions, are important in the negotiation and settlement phases as well.

The computerized nature of e-catalogs allows for opportunities to automate and streamline many of the processes involved in an e-commerce transaction. Catalog interoperability is essential in realizing these opportunities and it will also provide for further benefits, including the abilities to compare products from different suppliers, to locate complementary products, and to customize catalogs for each individual user and situation [2]. Another potential benefit is in catalog sharing. Most Internet shops spend a significant amount of resources in building and maintaining their e-catalogs. The manufacturers of the products must send their product information to each and every shop that carries their products. The overall process can be facilitated if certain parts of the efforts can be shared among the participants.

2. Catalog sharing and interoperability

Figure 1 presents our business model showing the interactions between three kinds of participants: manufacturers, Internet shops, and customers. Its purpose is to provide a universal e-catalog repository with browsing, searching, and downloading capabilities in order to facilitate catalog sharing and interoperability. By the nature of its business model, it must deal with catalogs from multiple sources that may be heterogeneous.

There can be a spectrum of approaches to managing multi-source catalogs, where at the one end is the fully materialized approach and at the other end is the virtual catalog approach. The fully materialized approach is somewhat of a brute-force approach where all catalogs are physically stored in one site using the catalog model

* This work was supported by the Brain Korea 21 Project.
provided by that site. As simple as it may seem, this is quite a practical approach where interoperability is not a big priority relative to other e-commerce functions such as order processing, billing, and customer management. Most of the commercial e-commerce software systems and hosting service providers, such as Open Market [5], take this approach. The virtual catalog approach provides some mechanism for catalog sources to dynamically respond to specific queries by mediators or agents, which will then restructure and combine the results for the customer. This approach is proposed in MEPC (Mediating Electronic Product Catalogs) [6] and Virtual Catalogs [7]. There are merits to having physical catalogs materialized in repository and also to having the ability to query relevant sources and construct catalogs on the fly. A catalog object in our model is a virtual catalog that can be bound directly to the contents in the catalog repository or to a query that will retrieve the contents from possibly distributed and heterogeneous catalog sources using a standard protocol.

One of the interoperability issues not actively pursued in the literature is that of product categorization. An e-catalog may be an HTML document describing a set of products or product groups with hyperlinks as well as a single product item such as a certain model of a TV set. Each participant dealing with e catalogs has its own perspective of the whole product universe. This perspective, in most cases, takes the shape of a categorization hierarchy. Without the consolidation of the categorization hierarchies, we can only achieve partial catalog interoperability; only for the individual product items. Our solution to this issue is to extend the catalog model to include categorization hierarchies as searchable and interoperable objects [3][4]. Also as part of the repository, it can provide a number of different perspectives that can be used in searching and browsing.

3. Live Catalog: the extended catalog model

3.1. Product catalog and category catalog

Users tend to use navigation even when they know what they want. Therefore, it is important to maintain a good product categorization hierarchy. The hierarchy must be robust enough to accommodate possibly different perspectives of potential users and be easily modifiable to adapt to changing trends and products.

For our purpose, sharing is another feature that must be supported for product categorization hierarchies. When an Internet shop obtains a set of e-catalogs from the repository, it should be able to obtain the sub-structure of the product hierarchy pertinent to those products. Therefore, the product hierarchy is not only the means for searching for product catalogs but also itself the target of search and extraction. We achieve this by modeling product categories as catalog objects with encapsulated structure and methods.

Live Catalog is our extended catalog model for modeling the set of products and categories and their relationships. There are two types of catalog objects in Live Catalog, product catalog and category catalog. A product catalog represents typical product information containing product number, title, manufacturer, descriptions, and prices. For each product, there is a set of attributes that are used to describe it. A category catalog is defined as a set of products that share a common property represented as a propositional condition over the attributes of the products. A category may have one or more subcategories whose conditions are more specific, and the set of categories forms a directed acyclic graph.

Definition 3.1 Live Catalog Graph is a tuple \((N, L_h, L_r, T, CP)\), where

- \(N\) is a set of catalog nodes: Product or Category,
- \(L_h\) is a set of links connecting nodes, which represent hierarchical relationships between nodes,
- \(L_r\) is a set of links connecting nodes, which represent general relationships (non-hierarchical) between nodes,
- \(T\) is a set of term vectors, which are used for dimension factors
- \(CP\) is a set of predicates on category nodes: Category Predicate

An example of a Live Catalog is shown in figure 2. Each catalog node (category or product) consists of a binary tuple \(<\text{content, rule}>\), where content is a structured object containing data on the product or category, and rule is a set of category predicates defining the products (or subcategories) that belong to the node. The relationship between two nodes can be hierarchical if it relates a category and its subcategory or general for all other kinds of relationships such as similar-to or complementary-to.
A term vector is composed of a set of keywords on the basis of common criterion and is used to determine the memberships of products in categories. There must be one initial term vector so that every product should be included in at least one category. For example, \( T_0 = \{\text{Desktop PC}, \text{Laptop}, \text{Air conditioner}, \text{Microwave oven}, \ldots\}\) can be an initial term vector on \( \text{PRODUCT\_FAMILY} \) and there can be \( T_1 = \{\text{Home}, \text{Office}\} \) on \( \text{PLACE} \) or \( T_2 = \{\text{Samsung}, \text{LG}, \text{Sony}, \ldots\}\) on \( \text{BRAND} \).

### 3.2. Category hierarchy and rule

A category is a set of products that share a common property. Three types of information need to be modeled in a category catalog object; data describing the category itself, its immediate sub-categories, and the products that belong to it. The content portion of the catalog object is used for representing the descriptions and direct links to its children. Two common approaches are found in commercial products for specifying the memberships of products in categories. One is where a category contains a set of pointers to its products, and the other is where a product contains pointers to the categories it belongs to. Both approaches are simple but problematic for inserts and deletes. A declarative specification of conditions for the membership is used in Live Catalog. The rule portion of the catalog object holds this condition on attributes of products. When it is necessary to retrieve the set of products for a certain category, the rule can be used as a query to retrieve the members at that point of time. Products and categories can be added or deleted without affecting the other. Examples of categories and corresponding rules are shown below.

#### Example 3.1

**Figure 2. A catalog hierarchy**

- **AirConditioner** : \( \text{IsMemberOf} (x, (\phi, \phi, 1, \phi, \cdots), T_i) \)
- **Microwave** : \( \text{IsMemberOf} (x, (\phi, \phi, 1, \cdots), T_i) \)
- **Appliances** : \( \text{IsMemberOf} (x, (\phi, \phi, 1, \cdots), T_i) \land \text{IsMemberOf} (x, (1, \phi), T_i) \)

**Definition 3.2** \( \text{IsMemberOf}(x, c, t) \in CP \)

\[
\begin{cases}
\text{TRUE}, & \text{if } x^t = c^t \\
\text{FALSE}, & \text{otherwise}
\end{cases}
\]

where, \( x^t \) is the vector corresponding to term vector \( t \in \mathcal{T} \) for an individual product \( c^t \) is a vector corresponding to term vector \( t \in \mathcal{T} \) for the category.

The rule construction of Live Catalog is very simple but powerful because the hierarchical relationship between catalog nodes is described by subsumption conditions based on the Boolean model. Thus, one could construct Live Catalog through the following steps:

1. Compose first level categories based on an initial term vector (\( \text{PRODUCT\_FAMILY} \) in our example): Air Conditioner, Microwave Oven.
2. Generalize a new category through union of lower level categories: Appliances.
3. Use two types of specialization:
   a. Specialize categories on a specific term vector (\( \text{PLACE} \) in our examples): Home appliances, Office Appliances.
   b. Specialize a new category through intersection of higher level categories: Air conditioner For Home.
4. Repeat the processes 2-3

#### Figure 3. An example of a category hierarchy and category rules
Figure 3 shows a detailed example of a Live Catalog with rules (from figure 2). Now we can derive Property 3.1 from the previous discussions and use it to identify the hierarchical (parent-child) relationships of categories in a category hierarchy.

**Property 3.1** A category subsumes each of its children categories in a category hierarchy based on the rules. That is, the subsumption relation is a partial order on the set of all categories in a category hierarchy.

4. Catalog Integration through Category Hierarchy Merging

The merging of two category hierarchies is the process of creating one big category hierarchy that contains all the information from each of the category hierarchies while still maintaining the hierarchical information of the original hierarchies. During this process, it is essential to identify the parent-child relationships. Rules, described in section 3, are used for this process. It is important that the automatic category-hierarchy merging process remains a lossless process. That is, all original information for each source category hierarchy must be preserved after a merge.

The automatic category-hierarchy merging algorithm consists of three phases. The first phase is designed to reduce the search space of the algorithm. The next process determines the most suitable place for a node to be merged. Finally, the third phase applies a case-by-case merging strategy to two nodes. The proposed algorithm receives two category hierarchies as inputs and selects one to be the base category hierarchy, and merges the other (target hierarchy) into the base hierarchy.

4.1. Phase 1: search space reduction

An exhaustive search of all nodes is the simplest way of merging category hierarchies. However, we can reduce the number of nodes that must be considered by comparing the rules of the root nodes in the base and the target category hierarchies.

In the first phase, the algorithm checks for the following four cases.

- **Case 1:** The whole target category hierarchy is subsumed by the base category hierarchy. The algorithm must consider all nodes of the target category hierarchy in the second and third phase. For the base category hierarchy, however, it only considers the subcategory hierarchy that has a root node rule that subsumes a root node rule for the target category hierarchy. Therefore, the search space of the algorithm is reduced to a subcategory hierarchy of the base category hierarchy.

- **Case 2:** Inversely, the whole base category hierarchy is subsumed by the target category hierarchy. The algorithm then restarts and changes the input order.

- **Case 3:** Rules for the base category hierarchy and the target category hierarchy overlap. That is, a subcategory hierarchy of the target hierarchy is subsumed by the base category hierarchy. In this case, the algorithm proceeds to the next phase with the subcategory hierarchy as the target hierarchy. After algorithm completion, it simply inserts all the nodes of the original target category hierarchy, excluding the previous subcategory hierarchy, into the base category hierarchy. In this case, the search space of the algorithm is reduced to just a subcategory hierarchy.

- **Case 4:** The base category hierarchy and the target category hierarchy are distinct. The algorithm simply inserts all the nodes of the target category hierarchy into the base category hierarchy.

After this phase, the algorithm has determined minimal subcategory hierarchies to work from, both the base and the target category hierarchies. The inputs for next two phases are the (new) base hierarchy, $C_1$, and the (new) target hierarchy, $C_2$. The algorithm iterates $C_2$ from the root node to leaf nodes and determines the most suitable node in $C_1$ for merging. For each iteration, both phases are processed sequentially.

4.2. Phase 2: determining the most suitable nodes for merging

After Phase 1, the root node of $C_1$ is guaranteed to subsume all nodes in $C_2$. Based on this, we have the following property.

**Property 4.1** Let the root node of $C_1$ be $S_0$. Then, for all nodes $T_i \in C_2$, $S_0$ subsumes $T_i$. That is, $S_0$ is the maximum upper bound of $T_i$.

Let the current node in the loop on $C_2$ be $T_i$. The most suitable place for $T_i$ to be inserted in $C_1$ is its minimum upper bound within $C_1$. The algorithm starts with the root node of $C_2$, and searches every path from the root $S_0$ to the leaf nodes in $C_1$ in order to find the minimum upper bound of $T_i$. During this process, there are four cases that the algorithm can encounter. The following property explains these cases.

**Property 4.2** For all nodes $T_i \in C_2$, during the search process, it goes through one of the following cases.
based on their rule as defined by Properties 3.1 and 4.1.

- Case 1: In the middle or at the end of the path, it meets a lower bound node without any overlap node on the way to that node.
- Case 2: In the middle or at the end of the path, it meets a completely different node without any overlap node or lower bound node on the way to that node.
- Case 3: In the middle or at the end of the path, it meets an overlap node.
- Case 4: At the end of the path, it meets an upper bound node.

From property 4.1, there exists at least one upper bound node of $T_i$ on each path. And then, in case 1-3, during the search of each path, the parent node of the first node not being upper bound that it meets is its minimum upper bound within the path of $C_j$. So, the parent node can be a candidate node for the most suitable node of $T_i$. In Case 4, however, the algorithm saves the leaf node, not the parent node, as a candidate node. After searching all the paths, the algorithm finds the minimum node among all the candidate nodes to be the most suitable node of $T_i$ and proceeds to the third phase with $T_i$ and the selected node of $C_j$.

### 4.3. Phase 3: merge two nodes on a case-by-case basis

In the third phase, the algorithm inserts the target category-hierarchy node information into the base category hierarchy. In the second phase, the algorithm will have already determined which two nodes are to be merged. Let them be $S_i, T_j \in C_j$. There are only two remaining subsumption relationships: one is for when both nodes are the same and the other is for when $S_i$ is not same but subsumes $T_j$. In the first case, the only thing to do is to modify hierarchical information of the base category hierarchy. In the second, however, the algorithm inserts $T_j$ into the base category hierarchy as a child of $S_i$. Doing this, it also adds hierarchical information of the target category including direct relationships between other nodes.

Especially, in Case 3 of phase 2, the algorithm processing is more complex. In this case, the algorithm continues to search up to a leaf node in the path even after it found a candidate node. It then determines a minimum overlap node of $T_i$ and puts it in the set of overlap nodes, $O_i$. Assume that $S_i$ is the node of $C_j$ merged or inserted for $T_i$. While processing a child node of $T_j$, called $T_j$, if $O_i$ is empty then the algorithm simply searches only the paths starting from $S_i$ to the leaf nodes in $C_j$. Every other path is pruned, since there exists at least one node in other paths that subsumes $T_j$ or they are completely different from $T_j$ based on Property 4.2. However, if $O_i$ is not empty, then it must search not only every path from $S_i$ down to the leaf nodes, but also, additionally, if there exist some nodes in $O_i$ that are the upper bound of $T_j$, search every path from those nodes down to the leaf nodes in $C_j$. This must be done because some nodes on the paths may subsume $T_j$.

An alternative method in Case 3 of Property 4.2 is to split an overlap node and create a compact category hierarchy as a merge result. However, the algorithm does not split an overlap node because that is one of the hierarchical structures of the base category hierarchy. That is, the primary principle of the algorithm is lossless merging. This is an important consideration for users of Internet shops. Another drawback of splitting is that the algorithm becomes more complex due to the increased number of nodes.

### 4.4. An example of category-hierarchy merging

![Example of category-hierarchy merging](image)

This example is a case in which the rules of each root node in a base category hierarchy and a target category hierarchy overlap. In figure 4, ‘source category hierarchy 1’ is a base category hierarchy and ‘source category hierarchy 2’ is a target category hierarchy.

At first, the algorithm determines the subcategory hierarchies enclosed by the ellipse so that search space could be confined to them. And then, the candidate nodes for node 1 are node 10 and node 4. However, node 4 is the minimum upper bound of node 1, so node 1 is then merged into node 4. In this case, because node 4 already exists in the base category hierarchy, the algorithm only needs to insert hierarchical information among nodes 1, 2, and 3. The sole candidate node for node 2 is node 4, so node 2 is merged into node 6 as a child node of node 4.
Node 4 is the only candidate for node 3, so node 3 is merged into node 5 as a child node of node 4 and a parent node of node 7. In these cases, because nodes 5 and 6 are new nodes of the base category hierarchy, the algorithm inserts hierarchical information as well as node information into the base category hierarchy. Finally, the algorithm simply adds nodes 8 and 9 to create new nodes of the base category hierarchy.

5. Conclusions

Our approach to multi-vendor catalogs is, on the one hand, similar to Virtual Catalog in that catalogs can be distributed over multiple sites and, on the other hand, similar to the integrated approach used by Open Market in that users can be provided with global views on all product catalogs. It provides users with consistent and well-categorized perspectives, which can be used in searches and to customize the views on-line. With the versatility and flexibility, we believe the proposed extended data model can effectively support a shared repository of e-catalogs as well as the role of a virtual catalog provider, either in the business-to-business or the business-to-customer settings.

Additionally, we proposed an automatic category-hierarchy merging technique for catalog integration. The algorithm uses subsumption relationships already maintained in each category node. Using these subsumption relationships, the algorithm merges categories, while maintaining the lossless merging property. Using the model, we may provide customers with a global view of categories merged from a number of different Internet shops, create an individually customized product category hierarchy for each customer, and provide a unified interface for mutual transactions among Internet shops based on a common category hierarchy. Moreover, the model and the algorithm are applicable to almost all domains that use category hierarchies, such as web community sites, unified newspaper service sites, and VOD/AOD sites.

6. References