$\mathcal{P}$at$\mathcal{M}$an: A Visual Database System to Manipulate Path Patterns and Data in Hierarchical Catalogs*

Aggeliki Koukoutsaki, Theodore Dalamagas, Timos Sellis, Panagiotis Bouros

Knowledge and Database Systems Lab
School of Electrical and Computer Engineering
National Technical University of Athens
Zographou 157 73, Athens, Hellas
{akoukou,dalamag,timos,pbour}@dblab.ece.ntua.gr

Abstract. Hierarchical structures is a popular means of organizing data in Digital libraries for browsing and querying. Examples of such structures are the catalogs which provide data organized in thematic hierarchies on a category/subcategory basis. Such hierarchies group data under certain properties. Navigational paths are the knowledge artifacts to represent such groups. We consider paths in hierarchies as patterns which provide a conceptual clustering of data in groups sharing common properties. Under this perspective, we have designed and implemented $\mathcal{P}$at$\mathcal{M}$an, a visual database system to manage hierarchical catalogs. The system can store navigational paths and data from hierarchical catalogs, and provides a pictorial query-by-example language to manipulate path patterns and data in a uniform way.

1 Introduction

Hierarchical structures is a popular means of organizing data for browsing and querying. Hierarchical catalogs in Digital libraries is such an example. These catalogs provide data organized in thematic hierarchies on a category/subcategory basis. Users can navigate these hierarchies to identify data of their preference. Such hierarchies group data under certain properties. Navigational *paths* in these structures are the *knowledge artifacts* to represent such groups, and act as semantic guides to reach the data of each group either through browsing or through path expression query languages.

We consider paths in hierarchies as patterns which provide a conceptual clustering of raw data in groups sharing common properties. We focus on the extraction and manipulation of these patterns, combined with the manipulation of data. Consider for example two hierarchical catalogs which maintain information about photo equipment. Searching for lenses in the first catalog may need to follow the path /cameras&lenses/lenses, while in the second catalog may need

* Work supported in part by DELOS Network of Excellence on Digital Libraries, EC FP6 IST programme, no G038-507618.

to follow the path /photo/35mm systems/lenses. Such paths can be seen as alternative pattern versions for the same group of data. On the other hand, there are cases where the owner of a catalog may need to provide integrated photo systems, with camera bodies from Adorama and the appropriate lenses from B&H. In such case, the two paths /cameras&lenses/35mm SLR and /photo/35mm systems/lenses form a kind of complex pattern representing different raw data, i.e. bodies and lenses, that should be grouped together.

Our work addresses part of the requirements defined for pattern management in [10], [12], emphasizing on the management of path-like patterns and data in hierarchical catalogs. Similar requirements have been also introduced in the inductive database framework [5], where patterns are integrated within the database environment as full-fledged objects. Under this context, a number of specialized inductive query languages have been proposed but most of these concern descriptive rules [4], [8]. Path manipulation techniques for biochemical pathways treated as knowledge artifacts, i.e. biochemical reactions, are discussed in [6] and [11]. In these works, database systems to manipulate and query biochemical pathways are presented. The problem of manipulating paths (and tree-like structures, in general) appears also in XML data management [3, 7].

In our work, we manipulate the descriptions of data as path-like patterns. We capture the notion of alternative path-like patterns and complex patterns to provide a framework to query raw data from many catalogs together with their patterns. Under this perspective, we have already suggested models to represent hierarchical catalogs, emphasizing on the role of paths as knowledge artifacts in such structures, and defined the $\mathcal{P}$at$\mathcal{M}$anQL language to manipulate path-like patterns together with raw data [1].

This paper extents the work in [1] and presents $\mathcal{P}$at$\mathcal{M}$an, a fully-functionable prototype system that implements the aforementioned framework. The prototype assists the user to form queries using a visual interface that offers pictorial query-by-example capabilities. Also, it can visualize the query results, showing navigational paths for hierarchical catalogs as well as the raw data organized in these catalogs. The rest of the paper is organized as follows. Section 2 discusses representation issues for hierarchical catalogs. Section 3 presents an overview of the $\mathcal{P}$at$\mathcal{M}$anQL language. Section 4 gives an overall description of $\mathcal{P}$at$\mathcal{M}$an system. Section 5 describes and application scenario, and, finally, Section 6 concludes this paper.

2 Representing Hierarchical Catalogs

We consider data in hierarchical catalogs to be records in relations, organized in the leaves of the hierarchy. These relations are called *resource items*. Every resource item is characterized by a number of *attributes*. For example, SLR cameras is a resource item with attributes 'brand', 'model' and 'price'. Resource items can be reached following paths in a hierarchical catalog. For instance, SLR cameras might be reached through the path /cameras&lenses/35mmSLR.

We can combine several common resource items, creating *tree-structured relations (TSRs)*. Common resource items are items with a similar semantic interpretation in the knowledge domain that catalogs refer to. Intuitively, a TSR represents the different ways of accessing a resource item in a set of catalog schemas and can be modelled using an AND/OR-like graph. This graph can be seen as a set of patterns, where each pattern is one of the different ways to access a resource item. Figure 1(a) presents an example of a TSR with two OR components, one of which is an AND group denoted by the curved line crossing the involved paths /photo/35mmSLR/bodies and /photo/lenses. Intuitively, an OR component captures a way to access a resource item. For example, /photo/35mm systems is one of the two alternative patterns for the resource item SLR cameras. An AND group corresponds to complex resource items that can be constructed using items from one or many catalog schemas. To access such items, one should exploit all paths of the group. For example, the AND group in Figure 1(a) corresponds to SLR systems built from camera bodies and lenses, two resource items that are in different catalog schemas. The construction of AND groups is closely related to the cartesian product operator that we present in the next section. The formal definitions of catalog schemas and resource items can be found in [1].

Note that in this work we do not consider schema matching issues. We assume that a schema matching pre-processing resolves name mismatches and other inconsistencies [9].

3 The $\mathcal{P}$at$\mathcal{M}$anQL Language

In this section, we briefly discuss the $\mathcal{P}$at$\mathcal{M}$anQL language (a detailed description of the language is presented in [1]). The $\mathcal{P}$at$\mathcal{M}$anQL language manipulates paths together with raw data in hierarchical catalogs. It is based on a set of operators (*select, project, cartesian product, union, intersection and difference*) to manipulate TSRs from catalog schemas.

Select (σ). Select operates on a TSR to produce a new one, selecting instances of resource items and OR components whose paths satisfy some defined predicates. Figure 1(a) shows an example involving a select operator in the query construct a TSR to include all cameras from TSR 'SLR systems', other than Pentax, with price greater than 200, having '/photo/35mm systems' in their paths:

$$\sigma_{<brand \neq \text{“Pentax”},\ price>200><\text{“/photo/35mm systems”} \subset \$\_>}(SLR\ systems)$$

Results are presented in Figure 1(b). Notice that "/photo/35mm systems" $\subset$ \$_ holds if there is a path p in an OR component, such that /photo/35mm systems $\subset p$ (i.e. the sequence of nodes in /photo/35mm systems appears identically in p). In this case, the whole OR component is retrieved.

Project (π). Project operates on a TSR to produce a new one, keeping only some of the paths of each OR component or OR components on the whole, and some of the attributes of the related resource item. Figure 2(a) presents an example of a project operator in the query construct a new TSR from the TSR

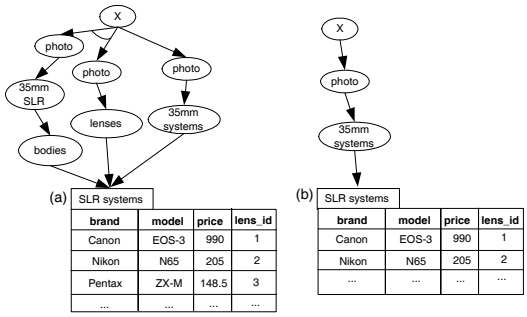

Fig. 1. *Select* operator.

'SLR systems' keeping only the rightmost component (i.e. #2) and the attributes 'model' and 'lens_id' of the resource item:

$$\pi_{<model,\ lens\_id><\#2>}(SLR\ systems)$$

Results are presented in Figure 2(b).

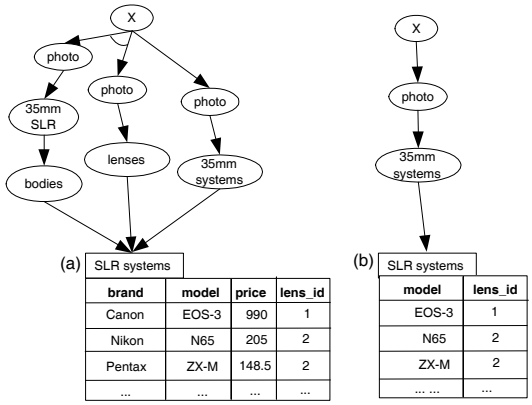

Fig. 2. *Project* operator.

Cartesian product ($\times$). Cartesian product operates on two TSRs to produce a new TSR, combining (a) every OR component of the first TSR with every OR component of the second TSR, constructing an AND group of paths, and (b) every instance of the resource item in the first TSR with every instance of the resource item in the second TSR. Figure 3(c) shows the cartesian product of the two TSRs of Figure 3(a) and (b). The AND group of paths /photo/35mmSLR/bodies and /photo/lenses (i.e. the leftmost OR component of

TSR SLR systems) is combined with the one and only OR component /camera & lenses/lenses of TSR Lenses, to construct a new AND group of three paths. Similarly, the rightmost OR component of TSR SLR systems is combined with /camera & lenses/lenses of TSR Lenses, to construct a new AND group of two paths. The new TSR has a resource item with attributes from both TSRs and combinations of instances.

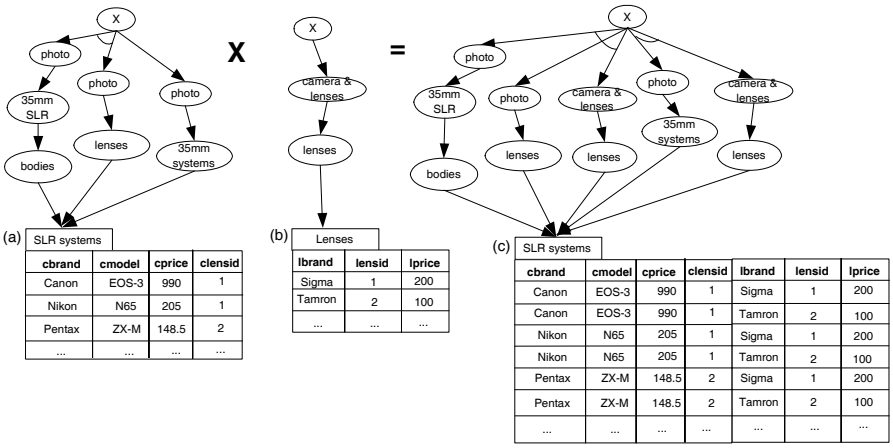

Fig. 3. *Cartesian product* operator.

Union ($\bigcup$) / **Intersection** ($\bigcap$). Union operates on two TSRs with the same set of resource item attributes. It produces a new TSR with all the OR components of the two TSRs and the union of their instances. Similarly to the union operator, intersection operates on two TSRs with the same set of resource item attributes. It produces a new TSR with all the OR components of the two TSRs and their common instances. Figure 4(c) shows the union of the two TSRs of Figure 4(a) and (b). The TSR constructed has all OR components of both TSRs. Also, the result contains instances either from the first or the second TSR.
Difference (-). Difference operates on two TSRs with the same set of resource item attributes. It produces a new TSR with the OR components of the first TSR and the instances of the first which do not exist in the second.

4 The $\mathcal{P}$at$\mathcal{M}$an System

We have developed $\mathcal{P}$at$\mathcal{M}$an, a prototype system that manipulates paths and raw data in hierarchical catalogs in a uniform way. $\mathcal{P}$at$\mathcal{M}$an provides modules to import TSRs from hierarchical catalogs coming in various forms, and store them in a relational database system. It also helps the user posing queries with a visual interface that offers pictorial query-by-example capabilities. Finally, it

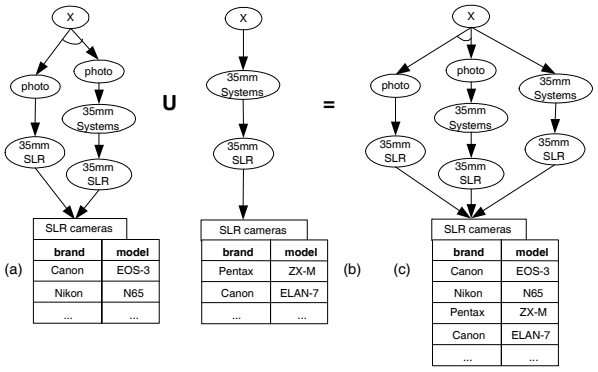

Fig. 4. *Union* operator.

can visualize the query results, showing the structure of the resulting TSRs as well as the raw data that satisfy the query predicates. We next describe briefly each one of system modules.

The *PQBE module* helps the user posing queries in $\mathcal{P}$at$\mathcal{M}$anQL. The visual interface provided offers Pictorial-Query-by-Example capabilities, and, thus, minimizes user errors coming from queries that do not conform to the schema of a TSR, or lack certain attributes from the resource items used, etc. The user may apply the unary operators *select* and *project* to manipulate a single TSR. For example, Figure 5 shows how one can apply a selection operator on a TSR. Figure 5(a) shows how to restrict the attributes that the user can select (e.g. brand) , according to the involved resource item, and Figure 5(b) how to restrict the paths that the user can select by retrieving and showing automatically the paths of the involved TSR (e.g /35mm SLR/lenses). Similarly, during the projection operator, user's options are restricted by retrieving and showing the available attributes, paths and *OR* components of the involved TSR.

The user may also apply the binary operators *union*, *intersection*, *difference*, *cartesian product* to manipulate a set of TSRs. In this case, the user should initially select all the TSR schemas that will participate in the query. Then, she can select a pair of TSRs and apply one of the binary operators available. The new TSR constructed as the result of this operation can be further processed as a single entity using a unary operator or can participate in a new query using a binary operator and another TSR. The task continues till the user constructs a TSR of her preference using the binary and unary operators available. As Figure 6 shows, the user can apply a unary operator in each TSR as well as a binary operator to combine the two selected TSR schemas. We note that in any step of the query formulation, the user can call the output module and see the structure of the hierarchy maintained in all TSR participating as well as the data organized in its hierarchy.

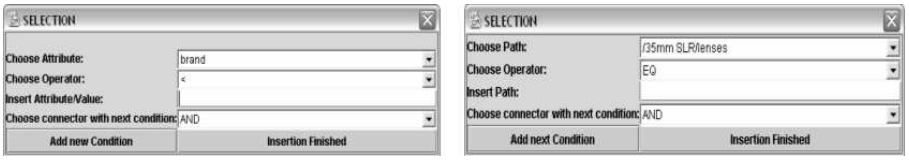

Fig. 5. Applying the *select* operator.

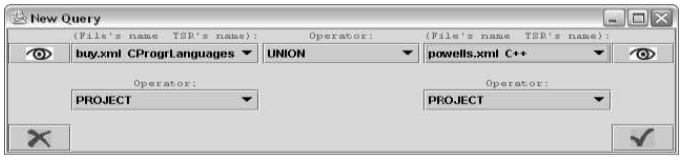

Fig. 6. Creating a query.

Query engine and *retrieval module* control the actual evaluation of the queries. Queries are reformulated to $\mathcal{P}$at$\mathcal{M}$anQL expressions using the operators *select*, *project*, *cartesian product*, *union*, *intersection* and *difference* discussed in Section 3. The TSRs involved in the query are retrieved either from the XML files or the tables of the RDBMS creating instances of TSR structures in the main memory. These structures are created dynamically, and capture the form of OR components, the attributes and the records of resource items. Depending on the operator, the OR components are processed separately from data records. The manipulation of OR components is based either on selecting/pruning single paths or groups of them, or by combining paths, or by unifying them under a common root. Data management functionality is provided, implementing SQL-like querying operations.

The *output module* presents the query results, visualizing the structure of the hierarchies maintained in the TSRs and the data records organized in the resource items. It also helps the user to explore TSRs which are already stored either in plain XML files or in the RDBMS. Figure 9 shows a TSR schema which is the result of a query. The AND groups are denoted by the curved line crossing the involved paths (in the original implementation paths in a certain AND group have identical colors). For example, /camera-lenses/35mmSLR/SLR systems and /photo/35mm Systems/lenses/SLR Systems cameras belong to an AND group. Resource items are represented by the DATA nodes. Selecting such a node, a new window opens, showing the attributes and the data records of the resource item (see Figure 9).

The $\mathcal{P}$at$\mathcal{M}$an system, beside its core functions for querying and storing TSRs from hierarchical catalogs, offers a set of tools supporting the user to perform other administration and manipulation tasks. Queries can be still written (following original $\mathcal{P}$at$\mathcal{M}$anQL's syntax) in text files and loaded in the system to be

executed on the TSRs stored. Also, attributes and data records can be added to empty resource items of stored TSRs, given relational tables filled with records or plain record-oriented delimited text files. Finally, TRSs which are stored as plain XML files can be imported in the RDBMS, and TRSs which are stored in the RDBMS can be exported as plain XML files.

5 Application Scenario

This section presents an application scenario to show how our system can be used to manipulate navigational path patterns and data from hierarchical catalogs. Figure 7 presents three TSRs from catalogs related to photo equipment. Specifically, (b) and (c) are TSRs from catalogs called Adorama and B&H, while (a) is a TSR of an imaginary catalog owned by X.

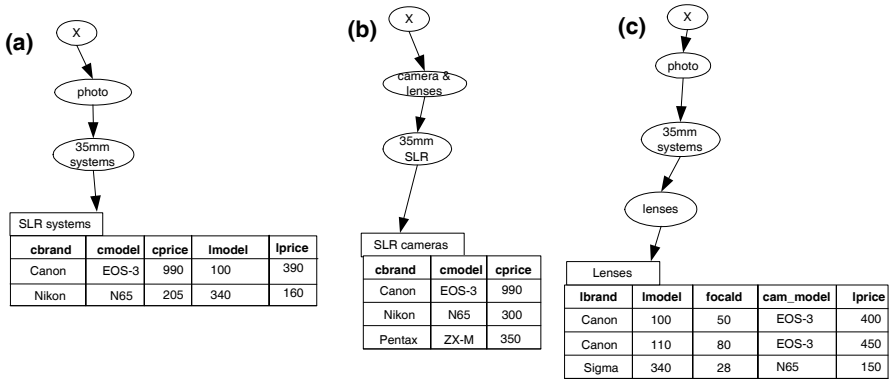

Fig. 7. TSR examples.

X sells integrated photo equipment, that is camera bodies and lenses as one package. Since new lenses are out in the market, X needs to find among the lenses provided by B&H, those that fit in 'Canon' bodies provided by Adorama, and are not in her stock as integrated systems. To create complex resource items (i.e. camera bodies and lenses as an integrated package), the user applies the *cartesian product* operator on SLR cameras and lenses to construct TSR1 TSR1 includes paths from both TSRs in an AND group and data for integrated photo equipment (i.e. camera bodies and lenses). From the records of the resource item node of TSR1, the user keeps only those (a) referring to 'Canon' camera bodies and (b) having lenses that fit these bodies. From all the attributes, the user keeps cbrand, cmodel, lmodel. Figure 8 illustrates TSR1. This TSR refers to integrated systems having 'Canon' bodies from Adorama and fitted lenses from B&H. To find integrated systems which are not in her stock, the user applies the *difference* operator between TSR1 and SLR systems, resulting in a new TSR having systems

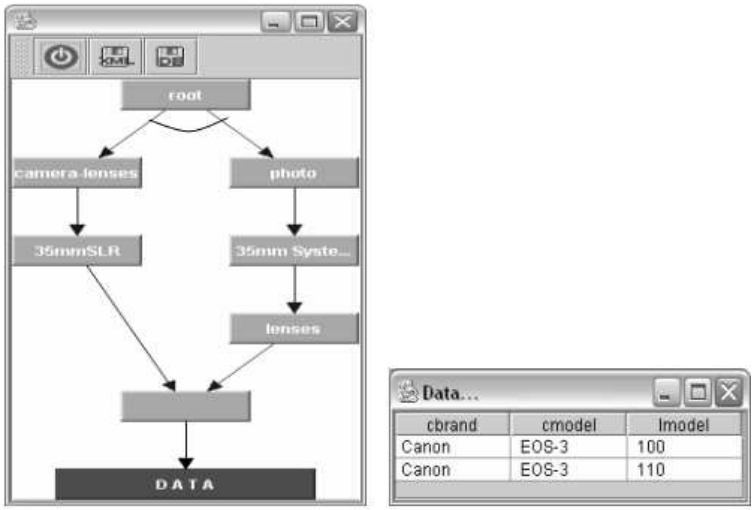

Fig. 8. TSR1

with 'Canon' bodies from Adorama and lenses from B&H which are not in X's catalog. Since difference operates on two TSRs with the same set of resource item attributes, a project operator is necessary on SLR systems to keep only attributes cbrand, cmodel and lmodel before applying the difference operator. The resulting TSR (see Figure 9) shows that there is an integrated photo system, including a body 'EOS-3' and lens '110' not offered by X Having as a result only the lenses without the appropriate camera bodies, requires a projection operation which keeps only the attribute lmodel and the path /photo/35mm systems/lenses.

6 Conclusions & Further Work

In this paper we described $\mathcal{P}$at$\mathcal{M}$an, a prototype visual database system to manage hierarchical catalogs. The $\mathcal{P}$at$\mathcal{M}$an system manipulates navigational path patterns and data from hierarchical catalogs in a uniform way, providing (a) mechanisms to import structures from hierarchical catalogs coming in various forms, (b) pictorial query-by-example capabilities based on the $\mathcal{P}$at$\mathcal{M}$anQL, a query language to manipulate TSRs from hierarchical catalogs, and (c) visualization capabilities to explore navigational paths for hierarchical catalogs as well as the raw data organized in these catalogs.

References

1. P. Bouros, T. Dalamagas, T. Sellis, M. Terrovitis, *PatManQL: A language to Manipulate Patterns and Data in Hierarchical Catalogs*, Proc. of EDBT PaRMa'04 Workshop, Heraklion, Greece, 2004.

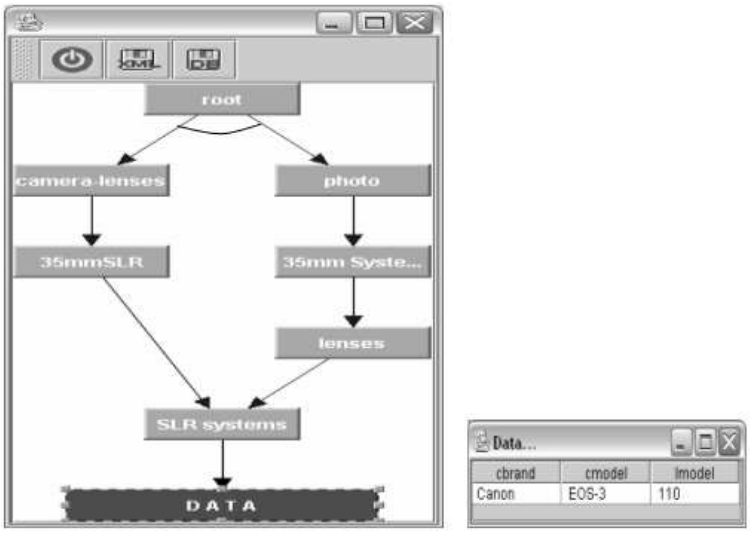

Fig. 9. Resulting TSR after the difference operation.

2. A. Chaudhri, A. Rashid, R. Zicari, *XML Data Management: Native XML and XML-Enabled Database Systems*, Addison Wesley, 2003.
3. V. Christophides, S. Cluet, J. Simeon, *On wrapping query languages and efficient XML integration*, Proc. of the ACM SIGMOD Conf., p141-152, 2000.
4. J. Han, Y. Fu, W. Wang, K. Koperski, O. Zaiane. *DMQL: A Data Mining Query Language for Relational Databases*, Proc. of the SIGMOD'96 DKMD Workshop, Mondreal, Canada, 1996.
5. T. Imielinski, H. Mannila, *A Database Perspective on Knowledge Discovery*, Commun. ACM, 39(11), 1996.
6. L. Krishnamurthy, J. Nadeau, Z. M. Ozsoyoglu, G. Ozsoyoglu, G. Schaeffer, M. Tasan, W. M. Xu, *Pathways Database System: an Integrated System for Biological Pathways*, Journal of Bioinformatics, 19, 2003.
7. B. Ludascher, Y. Papakonstantinou, P. Velikhov, *Navigation-Driven Evaluation of Virtual Mediated Views*, Lecture Notes in Computer Science, 1777, 2000.
8. R. Meo, G. Psaila, S. Ceri, *An Extension to SQL for Mining Association Rules in SQL*, Data Mining and Knowledge Discovery, 2(2), p195-224, 1998.
9. E. Rahm, P. A. Bernstein, *A survey of approaches to automatic schema matching*, VLDB Journal, 10(4), 2001.
10. S. Rizzi, E. Bertino, B. Catania, M. Golfarelli, M. Halkidi, M. Terrovitis, P. Vassiliadis, M. Vazirgiannis, E. Vrachnos, *Towards a logical model for patterns*, Proc. of ER'03 Conference, 2003.
11. I. Rojas, L. Bernardi, E. Ratsch, R. Kania, U. Wittig, J. Saric, *A database system for analysis of biochemical pathways*, Silico Biology, 2(2), p75-86, 2002.
12. M. Terrovitis, P. Vassiliadis, S. Skiadopoulos, E. Bertino, B. Catania, A. Maddalena, *Modeling and Language Support for the Management of Pattern-bases*, Proc. of SSDBM'04 conference, 2004.