PSYCHO: A Prototype System for Pattern Management

Barbara Catania Anna Maddalena Maurizio Mazza

Department of Computer and Information Science - University of Genoa
Via Dodecaneso 35, 16146, Genoa, Italy
{catania,maddalena,mazza}@disi.unige.it

Abstract

Patterns represent in a compact and rich in semantics way huge quantity of heterogeneous data. Due to their characteristics, specific systems are required for pattern management, in order to model and manipulate patterns, with a possibly user-defined structure, in an efficient and effective way. In this demonstration we present *PSYCHO*, a pattern based management system prototype. PSYCHO allows the user to: (i) use standard pattern types or define new ones; (ii) generate or import patterns, represented according to existing standards; (iii) manipulate possibly heterogeneous patterns under an integrated environment.

1 Introduction

The huge quantity of heterogeneous raw data collected from modern, data-intensive environments are not easily manageable from users. Knowledge extraction and data management techniques are therefore required to extract from them concise and relevant information that can be interpreted and manipulated by humans in order to discover interesting data correlations. Several kinds of *patterns* exist that can represent such hidden knowledge. In general, a pattern can be defined as a *compact* and *rich in semantics* representation of raw data. Clusters, association rules, frequent itemsets, symptom-diagnosis correlations are common examples of patterns.

Pattern management is an important issue in many domains. Data mining is certainly the most important context in which pattern management is required. Examples of other domains where patterns can be useful are information retrieval and image processing.

The specific characteristics of patterns make traditional DBMSs unsuitable for pattern management. In particular, patterns can be generated from different application contexts resulting in very heterogeneous structures. Moreover, patterns can be generated by using some data mining tools (a-posteriori patterns) but also known by the users and used for example to check how well some data source is represented by them (a-priori patterns). Since source data change with high frequency, another issue consists in determining whether existing patterns, after a certain time, still represent the data source from which they have been generated, possibly being able to change pattern information when the quality of the representation changes. Finally, patterns should be manipulated (e.g. extracted, synchronized, deleted) and queried through dedicated languages. All the previous considerations motivate the need for the design of ad hoc *Pattern Based Management Systems (PBMSs)* [4].

Many efforts have been devoted towards designing a PBMS. Existing proposals mainly differ in: (i) the chosen architecture for pattern and data management; (ii) the considered pattern model; (iii) the languages for pattern retrieval and management. The architecture of a PBMS can be integrated or separated. In the first case, raw data and patterns are stored together and the mining process is usually seen as a particular type of query. In the second case, raw data and patterns are logically stored and managed by two distinct systems, and mining operations are not queries but manipulation operations.

Several approaches have been provided for pattern management. Scientific community efforts usually provide general frameworks for pattern extraction and management. Among them, we recall the Inductive Database approach, relying on an integrated architecture and mainly investigated in the context of the CINQ project [5, 3], the 3W Model [2] and the PANDA framework [1, 4, 9], relying on a separated architecture. CINQ has provided solutions (and prototypes) for the management of specific data mining patterns (mainly association rules), together with SQL-like languages to

Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005

deal with pattern extraction and manipulation. Differently, the 3W Model and PANDA frameworks guarantee the representation and the management of heterogeneous and possibly user-defined patterns, represented as linear constraints in 3W Model and according to an object-relational model in PANDA. Various standards have also been defined with the aim of providing standard representation and manipulation of patterns resulting from data mining and data warehousing processes, in order to support their exchange between heterogeneous architectures. Among them, we recall PMML [10], providing an XML-based format for representing data mining results and the used mining algorithm, Common Warehouse Metamodel (CWM) [6], an important standardization effort for data warehousing (and data mining) metadata, Java Data Mining (JDM) [7], a Java API supporting pattern manipulation and guaranteeing interoperability between data mining applications. From the side of commercial systems, the most important DBMSs address the pattern management problem by providing an application layer offering features for representing and managing typical data mining patterns.

Unfortunately, none of the proposed approaches provides all the capabilities introduced above for pattern management. All approaches, except PANDA, 3W Model and, in a restricted way, PMML, support just the management of patterns of the same type, usually generated by a mining process. Moreover, pattern types are often predefined and cannot be extended by the user. Finally, synchronization issues are only partially taken into account (and not explicitly considered by existing standards).

Starting from these limitations and taking into account the results presented in the context of the PANDA project [9], we have designed and implemented a PBMS coping with most of the features previously introduced. The system, called PSYCHO, from *Pattern base management SYstem arCHitecture prOtotype*, is based on a separated architecture and provides the following features: (i) manipulation of heterogeneous patterns; (ii) definition of user-defined patterns, not necessarily coming from a data mining context; (iii) a Pattern Manipulation Language (PML) supporting the management of both a-posteriori and a-priori patterns and pattern synchronization; (iv) a Pattern Query Language (PQL), offering query capabilities for selecting and combining patterns, possibly of different types, and for combining patterns and data in order to get a deeper knowledge of their correlations (cross-over queries).

In the following, we briefly review the PSYCHO pattern model (Section 2), we sketch the PSYCHO architecture (Section 3), and we finally outline the context of the demonstration (Section 4).

2 The Pattern Model

The PSYCHO logical model relies on the PANDA model [1, 4, 9] and is based on three main concepts: *pattern type*, *pattern*, and *class*.

A *pattern type* gives a formal description of the pattern structure. It is a record with six elements: (i) the *pattern name* n; (ii) the *structure schema* s, which defines the structure of the patterns instances of the pattern type; (iii) the *source schema* d, which describes the dataset from which patterns, instances of the pattern type being defined, are constructed; (iv) the *measure schema* m, which is a tuple describing the measures which quantify the quality of the source data representation achieved by the pattern; (v) the *formula* f, carrying the semantics of the pattern. f is a constraint-based formula describing, possibly in an approximated way, the relation between data represented by the pattern and the pattern structure. Inside f, attributes are interpreted as free variables ranging over the components of either the source or the pattern space; (vi) the *validity period schema* v, defining the schema of the temporal validity interval associated with each instance of the pattern type.

Patterns are instances of a specific pattern type. Thus, they are record values with identifiers containing the proper instantiation of the corresponding schema elements in the pattern type. In a pattern, the formula component is obtained from the one in the pattern type by instantiating each attribute appearing in s with the corresponding value, and letting the attributes appearing in d range over the source space. An example of a pattern type representing circular clusters of items (represented by quantity and price) and one specific pattern of that type are shown in Fig. 1. We remark that the data source represents the overall data set the pattern is related to. On the other hand, the formula represents, in an intensional and possibly approximated way, the subset of data represented by the pattern. The extensional set of data exactly represented by the pattern, when needed, can be stored in the system as a sort of metadata. In Fig. 2, assuming to represent the cluster of Fig. 1, the intensional mapping is represented by all the points inside the circle, the extensional one by all the black points, and the source dataset by all black and white points.

A *class* is a set of semantically related patterns and constitutes the key concept in defining a pattern query language. A class is defined for a given pattern type and contains only patterns of that type.

Based on the considered pattern model, we designed three languages for pattern management. The *Pattern Definition Language (PDL)* is used for defining new pattern types, classes, and *mining functions*, used for pattern extraction. The *Pattern Manipulation Language (PML)* is used to perform operations such as insertions, extraction, deletions, updates, synchronization of patterns. Moreover, it allows the user to insert

n: ItemCluster
s: TUPLE(repr:TUPLE(id:STRING, price:REAL,qty:REAL), max_dist:REAL)
d: items:SET(TUPLE(id:STRING,price:REAL,qty:REAL))
m: AvgIntraClusterDist:REAL
f: $\forall x \in$ items (dist(repr, x) $\leq$ max_dist)
v: [start:DAY, end:DAY)

pid: 221
s: [repr:[id:A12, price:10,qty:20], max_dist:0.5]
d: itemsVIEW:SELECT(id,price,qty) FROM Products)
m: AvgIntraClusterDist:0.75
f: $x \in$ itemsVIEW : (dist(repr, x) $\leq$ 0.5)
v: [1-JAN-2005,31-MAR-2005)

Figure 1: The cluster pattern type and one pattern

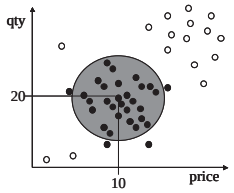

Figure 2: Source data set and mappings

(remove) a pattern into (from) a certain pattern class. Finally, the *Pattern Query Language (PQL)* allows the user to query the PBMS in order to retrieve patterns and correlate them with data they represent (*cross-over queries*). For all the three languages, an SQL-like syntax has been provided to simplify the request specification.

3 The PSYCHO architecture

The PSYCHO architecture relies on Oracle [8] and Java technologies and can exploit Oracle Data Mining (ODM) server functionalities when dealing with standard data mining patterns. The architecture is composed of three distinct layers (Fig. 3). The physical layer contains both the *Pattern Base* and the *Data Source*. The Pattern Base stores pattern types, patterns, and classes; the Data Source stores all raw data from which patterns have been extracted. It is in general distributed and various types of repositories (relational, XML, etc.) can be considered. The middle layer, that we call *PBMS Engine*, coincides with the kernel of the system, and it supports all functionalities for pattern manipulation and retrieval. The PBMS Engine and the Pattern Base represent the core of the PSYCHO prototype. The external layer corresponds to a set of user interfaces (a shell and a GUI) from which the user can send requests to the engine and import/export data in other formats. The communication between the PBMS engine and the physical layer is performed in Java. To be more flexible in the implementation of the external layer modules, communication between the PBMS Engine and the external layer is established using sockets and implementing requests as serializable objects. The result is a completely distributed architecture, where the pattern base, source data, PBMS Engine, and external modules can reside on different hosts.

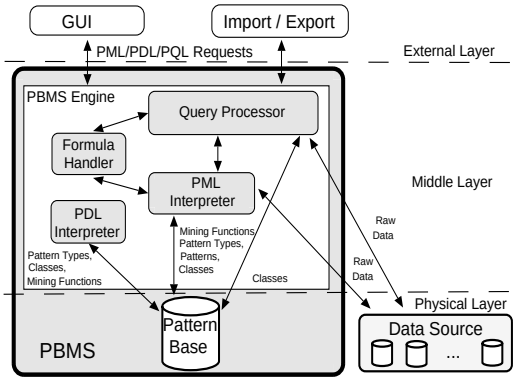

Figure 3: The 3 layers architecture of PSYCHO

3.1 Physical Layer

Pattern Base. The *Pattern Base* component contains pattern type, pattern, and class definitions. In PSYCHO, we used the object-relational model of Oracle 10g DBMS [8] for pattern storage. Concerning the pattern formula, we consider two distinct representations: an operational one, by which the formula is interpreted as a predicate over data source elements implemented as an Oracle PL/SQL stored function; a declarative one, by which the formula is just a representation of a linear constraint formula (see below the Formula Handler for additional details). Since the provided implementation of the Pattern Base exploits the Oracle object-relational logical model, the PML and PQL interfaces are realized using PL/SQL functions and procedures which are invoked by the Java application implementing the PBMS Engine (see below).

Data Source. The *Data Source* is a distributed database containing raw data from which patterns have been extracted. Various technology can be used to store the source datasets: relational or object-relational DBMSs, XML dataset, streams, etc. In the current PSYCHO version, we assume raw data are stored in an Oracle 10g DBMS.

3.2 Middle layer

The middle layer consists of the *PBMS Engine* component, whose aim is to execute requests sent to the PBMS. The PBMS Engine has been implemented in Java and is logically divided into three main sub-modules, each of which is dedicated to parse PQL, PML, and PDL requests, respectively, and to execute them through calls to the right functions and procedures defined in the Pattern Base. A fourth component is dedicated to the management of the intensional mapping. Such modules are described in the following.

PDL Interpreter. It takes in input a PDL request for a pattern type or class definition and translates it into calls to the right functions and procedures defined in the Pattern Base.

PML Interpreter. It executes PML operations (extraction, deletion, synchronization, operations over

classes). Pattern extraction and synchronization require an interaction with the Data Source to get the data from which patterns have to be generated. Information concerning the mining function can be retrieved from the Pattern Base. In the current PSYCHO implementation, pattern extraction can use either mining functions provided by ODM (e.g., a variant of the A-priori algorithm for association rules, the K-means or the proprietary O-cluster algorithm for clusters) or other mining functions provided by the PBMS. To this purpose, PSYCHO contains a library of predefined mining functions but new ones can be defined by the user. The Query Processor has to be used in case patterns have to be filtered (for example, only patterns with specific measures have to be generated, synchronized, deleted, or inserted in a given class).

Query Processor. It executes queries expressed in PQL, after choosing a query execution plan. For non-cross-over queries, only the Pattern Base, and eventually the Formula Handler, can be involved in the query process. On the other hand, for cross-over queries, Data Source may be required to execute the query. Queries may also use formulas, for example to compare two patterns. When formulas are used under the operational semantics, the queries are executed directly by the Query Processor. On the other hand, when they are used under the declarative semantics, the Formula Handler module is required to execute the query.

Formula Handler. It deals with the declarative representation of formulas, i.e., with constraints. It is used by the PML and PQL interpreters when computations over formula constraints are required. We implement the Formula Handler as a Java module, using the Jasper package for interacting with SICStus Prolog environment [11].

3.3 External Layer

GUI. User requests can be specified through either a GUI, providing both a visual environment or a simple shell, where the user can specify his/her request using an SQL-like syntax. In the first case, the request is translated into the corresponding PDL, PML or PQL request in SQL-like syntax and directly passed to the PBMS Engine that will execute it.

Import/Export. The Import/Export module allows the user to import and export in the PBMS patterns already represented by using standard formats. In the current PSYCHO version, we have already implemented a module for importing/exporting association rules represented in PMML [10], by exploiting, when possible, the import/export functionalities of the ODM server.

4 Outline of the demonstration

The demonstration will highlight the peculiarities of PSYCHO, comparing the supported features with those provided by other existing pattern management solutions. PSYCHO features will be described by using three main pattern management scenarios concerning: (i) management of homogeneous patterns; (ii) management of heterogeneous patterns; (iii) management of user-defined patterns, possibly not related to typical mining processes.

In the context of such scenarios, we will demonstrate the following PSYCHO characteristics, by using either the PSYCHO shell or the PSYCHO GUI: (i) the usage of a-priori and a-posteriori patterns in common data mining processes; (ii) the capability to exploit synchronization to guide the knowledge decision process, a feature which has only been partially considered in other existing proposals; (iii) the expressive power of PQL, by presenting, in the context of the different scenarios, various types of queries, involving patterns and raw data; (iv) the usage of the various pattern-data mapping representations; this issue is quite innovative, since, as far as we know, existing solutions deal with at most a single pattern-data relationship representation; (v) the functionalities of our Import/Export module, which enhances the basic PMML import functionalities provided by Oracle.

References

[1] B. Catania, A. Maddalena, and M. Mazza. A Framework for Data Mining Pattern Management. In *Proc. of ECML/PKDD*, pp. 87-98, 2004.

[2] S. Johnson, L.V.S. Lakshmanan, and R.T. Ng. The 3W Model and Algebra for Unified Data Mining. In *Proc. of VLDB*, pp. 21-32, 2001.

[3] R. Meo, P. Lanzi, and M. Klemettinen. *Database Support for Data Mining Applications.* LNCS 2682. Springer-Verlag. 2004.

[4] S. Rizzi et al. Towards a Logical Model for Patterns. In *Proc. of ER*, pp. 77–90, 2003.

[5] CINQ project. http://www.cinq-project.org.

[6] Common Warehouse Metamodel (CWM). http://www.omg.org/cwm

[7] Java Data Mining API. http://www.jcp.org/jsr/detail/73.prt.

[8] Oracle10g Database. http://www.oracle.com/-technology/products/database/oracle10g/

[9] PANDA Project. http://dke.cti.gr/panda

[10] Predictive Model Markup Language (PMML). http://www.dmg.org/pmml-v3-0.html, 2003.

[11] SICStus Prolog (v.3) http://www.sics.se/isl/-sicstuswww/site/