What's new in Querying, Query Processing and Optimization in PBMS?

Ilaria Bartolini[1], Paolo Ciaccia[1], Marco Patella[1], and Yannis Theodoridis[2]

[1] DEIS – University of Bologna, Italy
e-mail: {ibartolini, pciaccia, mpatella}@deis.unibo.it

[2] CTI and University of Piraeus, Greece
e-mail: ytheod@cti.gr

Abstract: Representing, storing, and manipulating useful patterns extracted from raw data is an emerging field of research in data management The architecture of a so-called Pattern-Base Management System (PBMS) should definitely include modules for pattern definition/manipulation languages, appropriate indexing mechanisms and efficient query processing, evaluation and optimization techniques. In this paper, we provide a preliminary overview of the above issues and discuss efficient solutions.

1. Introduction

In this paper, we consider the major issues related to querying and query processing in a, so called, Pattern-Base Management System (PBMS), that handles patterns extracted from large databases.

The outline of the paper is as follows: In Section 2, we briefly review what has emerged so far concerning the logical modeling of patterns and then, in Section 3, we classify operation types depending on the type of patterns they apply to and on the type of relationships between patterns and between patterns and raw data they exploit. In Section 4, issues concerning the declarative languages that supported by the PBMS, are reported. A set of preliminary examples of pattern types, pattern instances and queries is also presented in that section. Section 0 introduces some query optimization issues based on query examples presented earlier, in Section 4. Finally, Section 6 highlights the relevance of some operation types which, according to our view, are peculiar to PBMSs and as such deserve careful consideration, and summarizes some open issues for future work.

2. Basics from a Logical Model for Patterns

In [6], a logical model for patterns is proposed and the concept of Pattern-Base Management Systems (PBMS) is introduced. Restricting our attention to those aspects which can have a relevant impact on the physical level, we list the following:

1. The PBMS will manage a variety of *pattern types*, which cannot be a-priori defined; the extension of each pattern type will consist of a collection of *patterns* sharing similar characteristics (i.e. they fit the pattern type definition).
2. A pattern type *pt* is a quintuple (*n*, *ss*, *ds*, *ms*, *f*), where *n* is the name of the pattern type; *ss*, *ds*, and *ms* (called, respectively, structure schema, data source schema, and measure schema) are types in T (the set T of types includes all the base types together with all the types recursively defined by applying a type constructor to one or more other types); *f* is an formula, written in a given language, which refers to type names appearing in the source and in the pattern schemas.
3. A pattern *p* instance of a pattern type PT is a quintuple (*pid*, *s*, *d*, *m*, *e*), where *pid* (pattern identifier) is a unique identifier for *p*; *s* (structure) is a value for type *ss*; *d* (data source) is a dataset whose type conforms to type *ds*; *m* (measure) is a value for type *ms*; *e* is obtained by the formula *f* by instantiating each type name appearing in *ss* with the corresponding value specified in *s*.

Besides managing patterns, a PBMS should also take a set of *pattern (semantic) relationship types* (e.g. aggregation, refinement, generalization) into account, which will be used to model a pattern base according to the specific application needs.

It is also convenient to take into account the fact that some interesting relationships between patterns arise because of the presence of *data source relationships* that hold between the corresponding data source schemata *ds*. For instance, two sets of association rules might be related because they have been mined from two data sets corresponding to different stores of a same chain.

3. Classification of Pattern Operations

For query processing purposes, patterns are classified according to how they are constructed and what they are used for (Fig. 1).

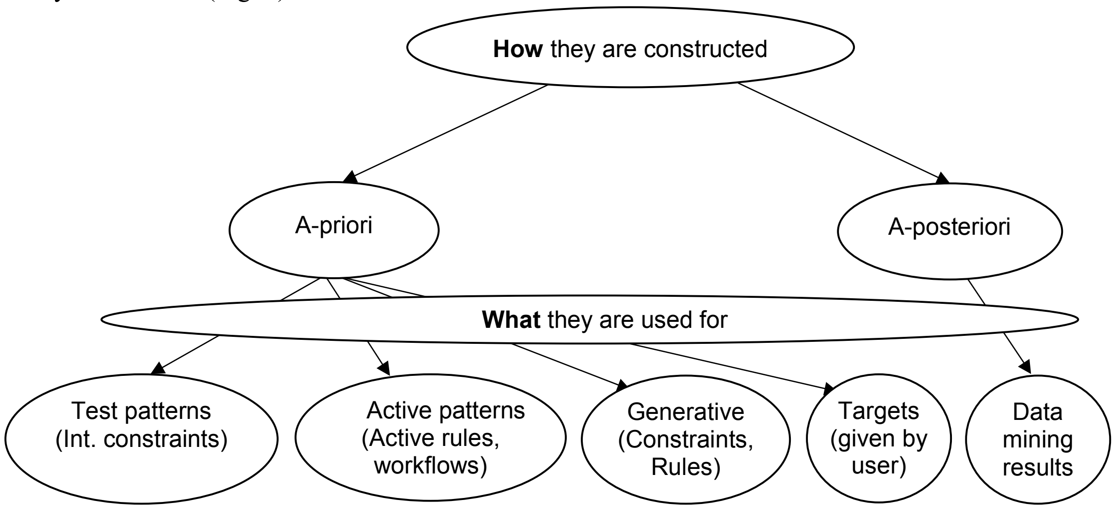
How they are constructed
A-priori
A-posteriori
What they are used for
Test patterns (Int. constraints)
Active patterns (Active rules, workflows)
Generative (Constraints, Rules)
Targets (given by user)
Data mining results

Fig. 1. Pattern classification schema

- *A-posteriori patterns*: patterns created in the PBMS by means of a processing on the raw data, typically as results of a data mining task.
- *A-priori patterns*: patterns existing a priori in the raw dataset (e.g. integrity constraints), being enforced by DBMS administrator or user.

From the query processing point of view, a-posteriori patterns typically require some kind of processing to be performed on the raw data in order to extract their structure and measure components, whereas a-priori patterns are usually compared against raw data to produce a score assessing how well a pattern is representative of (a part of the) raw data.

We now analyze the main patterns relationship types we are interested in for query processing issues (see also [6]).

- *Composition*: A complex pattern can be obtained by assembling other patterns; more precisely, an aggregated pattern has a structure schema which is defined in terms of one or more pattern types.
- *Refinement*: A refinement is a relationship existing between patterns at different abstraction or granularity levels.
- *Specialization*: A pattern type pt1 specializes another pattern type pt2 when the structure schema, the source schema, and the measure schema of pt1 specialize the structure schema, the source schema, and the measure schema, respectively, of pt2.
- *Induced relationships*: Relationships existing between subsets of the data space can induce relationships on patterns related to such data. Relationships of this type can also lead to refinement relationships. As an example, consider a functional dependency between two features product and category in the data space: product $\rightarrow$ category. Such relationship would naturally induce a number

of relationships between patterns generated over the data, e.g. between rules built on categories and rules built on products. This relationship can also be viewed as a refinement.

Such relationships between patterns, along with the obvious relationship between patterns and the data they are extracted from (for a-posteriori patterns) or against which they have to be matched (for a-priori patterns), define a classification of the different operation types a PBMS has to deal with. In the following we will present examples of such classification.

3.1 Basic operations

A basic operation between patterns is that of **comparison**: Two patterns of the same (simple) type can be compared to compute a score s, $s \in [0,1]$, assessing their mutual similarity. The similarity between two simple patterns is computed as a function of the similarity between both the structure and the measure components:

$$pattern\_similarity = f(structure\_similarity, measure\_similarity)$$

Supposed two patterns have the same structure, the measure of similarity naturally corresponds to a comparison of the patterns' measures, e.g. by aggregating differences between each measure [2]. In the general case, however, the patterns to be compared have different structures, thus a preliminary step is needed to *reconcile* the two structures to make them comparable.

3.2 Operations on PB-DB relationships

These operations allow crossing the boundary between the PBMS and the DBMS, relating patterns to data from which they are generated or with which they have to be matched.

- *PB-DB relationships for a-posteriori patterns*. This class contains all the operations which are used to create patterns from the raw data. Typical examples include data mining algorithms as extraction of association rules for market-basket analysis, clustering, etc. Such operations are typically used to populate the Pattern Layer of the PBMS.
- *PB-DB relationships for a-priori patterns*. As above anticipated, a-priori patterns are patterns which already exist in the PBMS and that are compared (e.g. matched) to the raw data at hand. Basic operations belonging to this class include:
  a. Matching a pattern against a subset of the data space (*matching*), obtaining the data objects along with their score expressing how well they fit the given pattern.
  b. Matching a data object against a set of patterns of a given type (*classification*), obtaining a score for each pattern, expressing how well each pattern is suited to represent the given datum.
  c. Obtaining the measures for a given pattern with respect to a set of data (*measuring*); given the pattern structure, the user could ask to compute its measures with respect to a subset of the raw data. As an example of this class of operations, just consider the computation of support and confidence measures for an association rule over a set of data.

3.3 Operations involving composition

In a complex pattern, the structural component includes the schemes of simple component patterns, whereas the measure part may include values defined as expressions of the component pattern schemes [6]; an obvious generalization allows to aggregate complex patterns, thus defining a multi-level pattern hierarchy. A basic operation belonging to this class is the comparison of complex patterns. The similarity score s between two complex patterns of the same type is computed starting from the similarity between component patterns, then the scores obtained for each sub-pattern are aggregated, using an *aggregation logic*, to determine the overall similarity of the two patterns [2]. The aggregation logic may be very simple, just an expression combining numerical values, or a more complex one, if constraints and/or transformation costs are to be considered: For example, a suitable "matching" between components patterns might be needed. The aggregation logic can also involve a variety of transformations, each with an associated cost, and the overall similarity score is obtained as the maximum score obtained by applying all the possible transformations to the component patterns.

3.4 Operations involving refinement of patterns

Here, the most common operations are zooming in/out on different granularity/abstraction levels. For a-priori patterns, these correspond to multi-resolution queries: Processing of such queries may require access to raw data, depending on the relationships existing between patterns. As an example, consider a hierarchical clustering algorithm yields a *dendrogram*, where the nested grouping of clusters is represented [3]. At each granularity level, different clusters can be determined, see Fig. 2. Measures for clusters at higher abstraction levels (i.e. with lower granularity) can be obtained with simple computations from measures for clusters at lower levels, thus without need to access the raw data.

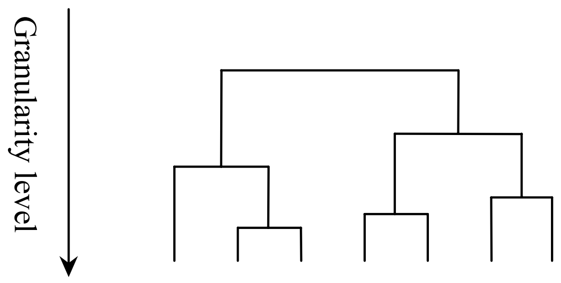

Fig. 2. A dendrogram

3.5 Operations involving specializations (or hierarchies)

Operations in this class allow the user to navigate the types hierarchy by generalizing (i.e. moving up the types hierarchy) or specializing (i.e. moving down the types hierarchy) a pattern. These are simple operations and we will not detail them further.

3.6 Operations on induced relationships

Operations belonging to this class act on patterns which have no direct relationship, but whose relationship is induced by relationships existing on the data such patterns are associated to. This may require to access the raw data in order to compute measures for patterns. For example, consider the relationship between association rules on categories and rules on products: if we want to compute the exact confidence of a rule on a given category, we cannot devise a formula to obtain it from measures of rules built on products of that category, but the original data have to be accessed.

4. Querying a Pattern Base

A DBMS (PBMS) is basically a system that stores and retrieves data (respectively, patterns) upon user's requests. The two fundamental features that are supported by a DBMS are the data model and the high-level languages for defining, manipulating and retrieving data. We are considering that the PBMS must provide the same set of declarative languages for defining, manipulating and retrieving pattern types and patterns.

4.1 Pattern Definition Language (PDL)

PDL is placed in the pattern type layer according to the PBMS architecture proposed in [6]. This language enables PBMS users to manipulate pattern types that are stored in the PBMS catalog. PDL must implement a set of constraints and checks that are related to the underlying patterns. For example, if a user posts a pattern type deletion the PBMS must check for existing pattern instances and abort the operation if associated patterns are found. The definition of PDL is an open issue.

4.2 Pattern Manipulation Language (PML)

PML should be able to support insertions, deletions and updates for patterns. The usage of PML in manipulating a-priori patterns is essential. Users should be given the tools to insert an a-priori pattern through a declarative way and PBMS should perform all the semantic checks before storing the pattern in the pattern base. For example, the condensed expression of a pattern type has its own structure and every value that is assigned to it must conform to that structure. Thus, the syntax and the semantic checks must be performed in multiple levels before a pattern acquires the grant for storing.

4.3 Pattern Retrieval (or Query) Language (PRL or PQL)

Queries in PBMS are divided in two main categories: queries that apply to pattern types and queries that apply to pattern instances. The query processor mostly derives pattern type queries at the semantic analysis phase, that is, after a query has been syntactically analyzed it is further checked for inconsistencies between the entities that it contains and the entities that exist in the PBMS catalog. Next, we give some examples for queries of that type, partially taken from [6].

Name	**Structure Schema**	**Data source Schema**	**Measure Schema**	**Function**
Association Rule	RECORD(head: SET(STRING), body: SET(STRING))	BAG(transaction: SET(STRING))	RECORD(confidence: REAL, support: REAL)	head $\cup$ body $\subseteq$ transaction
Interpolating Line	SET(sample: RECORD(x: REAL, y: REAL))	RECORD(a: REAL, b: REAL)	fitting: [0..1]	$y = a \cdot x + b$
TimeSeries	RECORD(curve: LIST(y: REAL), position: TIMESTAMP, shift: REAL, gain: REAL)	LIST(sample: REAL)	similarity: [0..1]	sample[position + i $-$ 1] = shift + gain $\times$ curve[i], $\forall i$: $1 \leq i \leq$ *length*(curve)

Table 1. Examples of pattern types

Pattern Identifier (pattern type)	**Structure**	**Data source**	**Measure**	**Expression**
512 (Association Rule)	(head = {'Boots'}, body = {'Socks', 'Hat'})	'SELECT SETOF(article) FROM sales_in_US GROUP BY transactionId'	(confidence = 0.75, support = 0.15)	{'Boots', 'Socks', 'Hat'} $\subseteq$ transaction
513 (Association Rule)	(head = {'Hat'}, body = {'Boots'})	'SELECT SETOF(article) FROM sales_in_Canada GROUP BY transactionId'	(confidence = 0.60, support = 0.35)	{'Boots', 'Hat'} $\subseteq$ transaction
456 (TimeSeries)	(curve = (y = 0, y = 0.8, y = 1, y = 0.8, y = 0), position = 12, shift = 2.0, gain = 1.5)	'colorado.txt'	similarity = 0.83	sample[12] = 2.0, sample[13] = 3.2, sample[14] = 3.5, sample[15] = 3.2, sample[16] = 2.0

Table 2. Examples of patterns

Apart from the query processor that submits queries to the system catalog during the semantic checking phase, users may also query the system catalog of the PBMS. The PBMS can be seen as a repository for pattern types and patterns, thus queries submitted to the system catalog may be of interest for a group of users. Examples of system catalog queries posted by the query processor will be presented along with queries on patterns. We use a table to demonstrate the result of every query but this does not imply that the query result will be a set of tuples as in the relational model.

4.3.1 Queries on pattern types

- *Q1: Retrieve all pattern types of the PBMS:* Q1 will return the entire system catalog. The type of this query result is unknown. Will it be a set of objects of type "pattern type" or a set of pattern type names that will be materialized upon user's or application's request? The result, according to the first option, would be identical to Table 1.
- *Q2: Retrieve the measures of pattern types:* Q2 shall return the measure schema of every pattern as shown in Fig. 3 (a).
- *Q3: Retrieve names and structure schemata of pattern types extracted by datasets of type bag (i.e. the Data source schema is like BAG%):* This query has a selection predicate on the value of the Data source schema component. The query language should have the ability to express such queries that refer to the internal structure and to the values of the pattern type components. The query result of this query is depicted in Fig. 3 (b).

Measure Schema
RECORD(confidence: REAL, support: REAL)
fitting: [0..1]
similarity: [0..1]

(a) result of Q2

Name	Data source Schema
Association Rule	BAG(transaction: SET(STRING))

(b) result of Q3

Fig. 3. Results of queries on pattern types

4.3.2 Queries on patterns

- *Q4: Retrieve patterns of type "Association Rule":* This query returns patterns that are instances of the "Association Rule" type and can be determined in two alternative ways. We will use the notation of SQL to illustrate the example.

  ```
  select * from "Association Rule";
  ```

  The above query implies a relational flavor, that is, every pattern type is a first-class object in the PBMS.

  ```
        select * from patterns
  where pattern_type = "Association Rule";
  ```

  On the other hand, this alternative query implies that all pattern instances are members of the unique and general entity "Pattern". Thus, the part "select * from Pattern" is always implied in every query.
- *Q5: Retrieve the name of types and the value of all Measures named "confidence" where value of confidence must be greater than 0.70:* The query result is shown in Fig. 4 (a).
- *Q6: Retrieve the head of patterns with type "Association Rule" (AR1) and the body of patterns with type "Association Rule" (AR2) having the body of the first to be equal with the head of the second:* The expected query result is shown in Fig. 4 (b).

Pattern Identifier (pattern type)	**Measure**
512 (Association Rule)	(confidence = 0.75, support = 0.55)

(a) result of Q5

AR1.head	**AR2.body**
{'Hat'}	{'Socks', 'Hat'}

(b) result of Q6

Fig. 4. Results of queries on patterns

5. Query optimization

In this section we present some preliminary issues concerning the query evaluation process in the PBMS. Again, the use of terms like "scan", "select" and "project" are taken from the relational lingua franca and do not impose any restriction on future definitions of the PBMS data model operations. We will use example Q5, taken from the previous section, to demonstrate some preliminary query evaluation plans. For example, Fig. 5 (a) and (b) illustrate a primitive and an improved, respectively, execution plan for Q5.

step 1. Scan the Pattern Base to find patterns that satisfy the condition "Measure.confidence > 0.70".

step 2. Project the value of measure from the patterns returned in Step 2.

step 3. Construct the query result

(a) naïve execution plan for Q5

step 1. Scan system catalog to find pattern types having "confidence" in their measure schema.

step 2. In the retrieved pattern types from Step 1, identify the type of measure "confidence" and check if the comparison with the const value "0.70" is valid. This implies static type checking.

step 3. Assuming an index on patterns.PTName, perform index scan in the pattern space and retrieve only those patterns that are instances of pattern type(s) of Step 1 and satisfy the condition "Measure.confidence > 0.70".

step 4. Project the value of measure from the patterns returned in Step 3.

step 5. Construct the query result.

(b) improved execution plan for Q5

Fig. 5. Results of queries on pattern types

6. Final Remarks and Summary

In this paper, we have presented some preliminary aspects on querying and query processing in PBMS. In the following, we address some issues that should be addressed in future work.

Pattern comparison: As anticipated in Section 3.1, comparison between patterns requires matching both the structural and the measure components of patterns [2].

Comparison of complex patterns: The comparison of complex patterns (see Section 3.3) entails the use of aggregation logic. Efficient evaluation of aggregation algorithms may require the indexing of component patterns, for example a metric index like the M-tree [1] can be used if the (dis-)similarity between component patterns is a metric. Other issues arise if one considers complex matches between component patterns, e.g. *M*-to-*N* matching, matching based only on a subset of component patterns, etc. Moreover, one should also consider that a number of transformations can be applied in order to obtain a better matching. Such transformations can considerably alter the complexity of matching. In this complex scenario, the use of approximate strategies could significantly reduce evaluation costs.

Obtaining measures for finer/coarser patterns: These operations allow the user to zoom in/out on different granularity/abstraction levels. Measures for coarser patterns can be obtained by appropriately combining the measure components of finer patterns. In order to reduce accesses to the DBMS, approximate results and additional knowledge on the object domain can be used. Similar arguments also apply to the case where measures are requested for finer patterns, since they can be estimated starting from measures for coarser patterns.

Other issues include:

- Operators similar to "Project", "Select", "Join", etc. should be defined for the structure and values of pattern types and patterns.
- Type checking and casting mechanism for PBMS need to be defined, as well as the level of strictness that it will provide.
- Definition of the intermediate query results types should be addressed.
- Additional metadata for pattern types and patterns could be considered.

References

[1] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. *Proceedings of the 23rd VLDB International Conference*, pp. 426-435, 1997

[2] V. Ganti, J. Gehrke, R. Ramakrishnan, Wei-Yin Loh. A Framework for Measuring Changes in Data Characteristics. *Proceedings ACM Symposium on Principles of Database Systems*, Philadelphia, PA, pp. 126-137, 1999.

[3] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31 (3), pp. 264-323, 1999.

[4] E. Keogh. Exact Indexing of Dynamic Time Warping. *Proceedings of the 28th VLDB International Conference*, pp. 406-417, 2002.

[5] E. Keogh, M. J. Pazzani: Relevance Feedback Retrieval of Time Series Data. *Proceedings of ACM SIGIR Conference*, pp. 183-190, 1999.

[6] S. Rizzi et al. Towards a Logical Model for Patterns. *Under submission*, 2003.