Contextual snowflake modelling for pattern warehouse logical design

VIVEK TIWARI* and RAMJEEVAN SINGH THAKUR

Maulana Azad National Institute of Technology (MA-NIT),
Bhopal 462 007, India
e-mail: vivek.vktonline@gmail.com; ramthakur2000@yahoo.com

MS received 10 March 2014; revised 22 August 2014; accepted 15 September 2014

Abstract. Pattern warehouse provides the infrastructure for knowledge representation and mining by allowing the patterns to be stored permanently. The goal of this paper is to discuss the pattern warehouse design and related quality issues. In the present work, we focus on conceptual and logical design of pattern warehouse, by introducing a context and ‘kind of knowledge’ hierarchy to this end. For the simplicity, association kinds of patterns are considered for running examples. We have extended well-known ‘snowflake’ schema for pattern warehouse logical design. We have introduced a new concept hierarchy ‘*kind of knowledge*’ which helps to arrange patterns, the four quality forms (QF) are also discussed which will work as guidelines for pattern warehouse conceptual and logical design to minimize the evaluation and maintenance cost. In particular, we address the three main issues: (i) conceptual design, (ii) snowflake schema and (iii) pattern refreshment.

Keywords. Pattern warehouse; pattern warehouse management systems (PWMS); data models; knowledge warehousing; conceptual modelling; context modelling; quality forms.

1. Introduction

Data management can be considered in three ways, management of daily transaction data, management of historical data (Barbara & Anna 2005; Zdenka 2012) and management of patterns. Transactional data are managed and maintained by operational databases (Michael 2010) which are also known as database management systems (DBMS). Historical data are managed by data warehouses and used for decision making (Batra 2005). Data in the data warehouse are in huge amount so the user cannot get anything from observation of data. It is clear that business users do not want massive data, but they are interested in trends hidden within data (Golfarelli *et al* 2004). This trend is also known as pattern. In the recent evolution of database technology, patterns are being managed by the pattern warehouse management system (PWMS) (Tiwari & Thakur 2014).

*For correspondence

The evolution of database technology is depicted in figure 1. Tiwari & Thakur (2014) have presented the architecture of PWMS where patterns are managed in type tier and pattern tier layers. In this paper, we try to further divide the patterns into groups in type tier layer according to their underlying context of raw data. Importantly, context of the data and snowflake based logical modelling is presented in this work. There are no standard, or even widely accepted, patterns management techniques, languages or design methodologies for pattern warehouse. The concept of making the pattern as persistent is new. The pattern is a candidate for generic representation was first time introduced by a PANDA report in 2003 (Ilaria *et al* 2003). Due to huge availability of data, many techniques have been developed to extract knowledge, especially in the context of data mining (Batra 2005; Vazirgiannis *et al* 2003). The results of such operations are abstract and compact representations of the original data, which called patterns (Catania *et al* 2004). The pattern gives the semantic representation of raw data. The volume of extracting patterns from various knowledge discovery applications is increasing rapidly, so there is a need for effective and efficient pattern management system (Jaesoon *et al* 2002; Mohammad *et al* 2009). The extracted patterns are stored in the pattern warehouse through Pattern Warehouse Management system (PWMS) (Catania *et al* 2004; Manolis *et al* 2007). Pattern warehouse is a new concept and little emphasis has been given till date. A pattern warehouse is as attractive as data warehouse as the main repository of an organization's pattern and can be optimized for reporting and analysis (Manolis & Vassiliadis 2003). By nature, patterns are not persistent. It means each time when you need patterns, you need to execute pattern generating method again and again (Tiwari & Thakur 2014). Pattern warehouse is a way to make the pattern persistent by storing them permanently. In this work, we try to bring the attention on pattern warehouse conceptual and logical design. Since patterns are very semantic rich, so we have to take attention on patterns individually or contextually and then design systems accordingly (Riccardo *et al* 2011). We restricted our attention to association kinds of patterns in examples. We have introduced quality forms (QF) as guidelines for good schema design for the first time.

Figure 1. Evolution of database technology.

The four quality-forms have been discovered to work as a road map in this work. These quality forms would help for designers to design well-robust, reliable and efficient pattern warehouse.

2. Literature survey

Ilaria *et al* (2003) has shown for the first time that the concept of pattern is a good candidate for generic representation. They discussed the main issue related to pattern handling and pattern representation. The work also outlined the architecture of pattern base management system (PBMS). Authors have insisted on the use of dedicated pattern storage system by discussing a variety of patterns available in huge amount nowadays. They introduced a new idea of persistent pattern. The presented work was very abstract and lacking the issues regarding implementation. The authors tried to extend SQL to retrieve the patterns, but it is not sufficient because patterns are semantically rich. Several important specific implementation issues still need to be investigated. The work has little emphasis on raw data behaviour and nature.

Manolis *et al* (2007) considered the modelling of language for querying patterns. Specifically, they define the logical foundations and mapping that covers data, patterns, and their intermediate mappings. They introduced query operators and predicates for comparing patterns. Authors represented the fact to support that volume, diversity and complexity of pattern to make their management by a DBMS like environment imperative. The authors explained that data to pattern and vice-versa mapping is important, but they failed to offer any underlying mechanism to achieve. The authors pointed out that the necessity to find out the relationship between patterns with respect to raw data, but they did not introduce any method. The work did not cover important pattern retrieval part. The authors argued that query operators are more appropriate than data mining techniques for pattern retrieval, but discussion was lacking to support this mythology. The work was required to discuss on actual way of pattern storage and how generic data structure should accommodate all kinds of patterns. There is some discussion about bottleneck of existing database systems like relational databases, XML based database with respect to pattern storage. The presented model only allowed for designer to organize and compare semantically similar patterns. They offered pointer based mapping to relate the patterns.

Rizzi (2004) provided a basic foundation for the design of pattern base by introducing UML based conceptual modelling. During the last few years, UML has been gradually superseding Entity/Relation in database domain. UML based conceptual modelling for pattern representation was the first introduced. They addressed the main issues in static modelling, including the representation of relationships between patterns, and briefly presented some issues related to functional and dynamic modelling. Author just shown how it would be possible to conceptually model a pattern-base of the static, functional, and dynamic points of view through extending UML. The author believed that adopting UML is still preferable since it was a standard de facto for most software engineering applications. The work was limited to mainly focus on static modelling. There is a little discussion on how patterns are distinguished according to static, dynamic and functional point of view. There is a need for more discussion on the necessity of functional and dynamic analysis of patterns. The authors introduced new pattern relationship, such as specification, composition and refinement, but failed to make it clear for working and operation of such relations. There is a need to introduce some operators which can carry out and find those relations. The authors have given little emphasis on raw data and source schema.

Manolis & Vassiliadis (2003) have presented the architecture of a pattern base management system that can be used to efficiently store and querying patterns. The authors introduced the intuition and mathematical foundations for pattern management. There is a need to discuss

that how presented architecture can be converted into conceptual and logical way. The authors assumed that the mapping between raw data and patterns already present, but they failed to introduce any technique or method to support this. The discussion is lacking to prove that mapping is possible in any ways. The authors also assumed that the patterns must 'qualify' as compact, but did not describe any parameters for the qualification. In a similar way, there needs to be more discussion to determine the degree of semantically rich patterns. The presented work just introduced data and pattern space and tried to make the mathematical relationship without any clear objective. We have not given the attention on developing methods to store, manipulation and retrieval of patterns.

Evangelos & Irene (2005) have studied the problem of the efficient representation and storage of patterns in a so-called pattern-base management system. They looked at three well-known models from the database domain, the relational, the object-relational and the semi-structured (XML) model. The three alternative models were presented and compared based on criteria like the generality, extensibility and querying effectiveness. The comparison showed that the semi-structure representation was more appropriate for a pattern-base. The authors just tried to extend existing database design approaches like relational, object-oriented and XML to make an efficient pattern management system. The work was limited to pattern representation only rather than to discuss pattern retrieval processes in detail. The presented work pointed that indexing is an important need for pattern retrieval, but did not describe how indexing would work on patterns. There was a very little discussion on pattern storage schema. The authors also extended query based retrieval method, but it worked well with structured data and could not fit on patterns efficiently. The work was limited to data mining pattern validation only.

Bartolini *et al* (2004) presented a framework for comparing patterns. Patterns are grouped in two ways: patterns and complex pattern, i.e., patterns built up from other patterns. Similarity operation is valuable whenever patterns are extracted from different data source using the same method and to know different behaviour of algorithm over a same dataset. The authors proposed the similarity operator, *SIM*, which has to take into account both the similarity between the patterns' structures and the similarity between the measures. They have formulized the similarity operator by taking simple patterns without considering the issues with complex patterns like how to reconcile the structure, making them comparable, etc. The work also lacks to cover the working of aggregation function with respect to combined structure and measure similarity. There is a need to take a working example for better understanding. They also need to cover the applicability of these operators with respect to pattern retrieval. There are still several issues need to be taken into consideration for making pattern retrieval more feasible.

Mazón *et al* (2008) discussed that facts and dimension hierarchy was important to explore the information at different levels of details. They represented a conceptual model to accommodate summarizability by adopting the normalization method. The authors introduced eclipse-based implementation of this normalization process. The presented work is more concentrated on normalization process rather than central issues of summarizability. There is a need for more detailed discussion on logical and implementation issue of summarizability by taking running example. The presented work is good to give basic guidelines for the data integration process so that it may develop summarizability compliant data processing method. We found that summarizability issue is important and its inadequate handling may cause to erroneous output of pattern aggregation.

Catania *et al* (2004) presented their work was based on PANDA (Ilaria *et al* 2003) theme. They tried to draw attention on more advanced issues like heterogeneity, temporal, querying, etc. of patterns management. Authors discussed important issue regarding variability of source or raw data, validation and synchronization of patterns. The works also discussed more general pattern

retrieval process to accommodate all kinds of patterns. The work was failed to determine pattern validation in case of source data has been changed or updated. There must have been some specific operator to check pattern validity. This work had little discussion on temporal pattern manipulation language (TPML) and it did not make any clear relation with pattern retrieval.

Vazirgiannis *et al* (2003) have reviewed the concept of patterns and their applicability in several areas. They examined the various types of patterns that were extracted from the dataset, in order to gather the necessary requirements for the definition of a pattern model. This model formed the heart of the pattern base management system. The authors tried to integrate the existing approaches towards a novel logical integration of patterns into a data model, language and base management system support.

3. Significance of pattern warehouse

The pattern, despite being already the result of some elaboration on raw data, is not, usually, in a form that can lead us directly to real life results (Manolis & Vassiliadis 2003). We need tools that will permit us to compare, query and store the pattern so that patterns can be retrieved on-demand when needed (Rizzi *et al* 2003). Pattern warehouse is being considered as a solution. Following section draws the attention on the necessity to separate pattern repository system and its benefits.

1. Pattern semantics are much richer than the raw dataset so the dedicated system needs to preserve it.
2. Patterns' behaviour/functionality is significantly more complex (Ilaria *et al* 2003). There involves complex multiple dimensions of similarity, such as (i) intra-pattern vs. inter-pattern similarity; (ii) Structural vs. value based similarity etc.
3. Since raw data may be very heterogeneous, so several kinds of artifacts exist that represent hidden knowledge (Inmon 2005). Clusters, association rules are common examples of such knowledge artifacts, generated by data mining applications (Tiwari *et al* 2010). So the dedicated pattern warehouse management system is required to handle this heterogeneity.
4. Patterns are a special kind of data. So we need to put them in a very specialized storage system that is called in this paper 'Pattern Warehouse'. This system must be able to handle all kinds of patterns.
5. PWMS is a specific system to store and reuse the patterns in order to fulfill requirements of the users for decision making.
6. PWMS system provides a valid mapping between the pattern warehouse and the raw data to be able to switch between.
7. Require a specific data structure or schema to store various kinds of patterns.
8. An intelligent pattern retrieval language needs to be incorporated in PWMS.
9. PWMS gives the ability to compare patterns with specified operations.
10. PWMS incorporates the clear policy for updating the patterns timely without creating inconsistencies.

4. Candidate patterns of pattern warehouse: Proposed context

Since the patterns are semantically rich and diverse (Riccardo *et al* 2011), therefore satisfying the user's interest is dependent on 'how and what kinds of pattern are being stored in a pattern warehouse' (Mazón *et al* 2008; Giorgini *et al* 2005). Inherently, pattern warehouse is also

subject-oriented. It is not at all feasible to store all possible patterns collectively in a pattern warehouse because managing the patterns is far more complex and complicated compare to data. In view of this, we are introducing 'context' term as a virtual separator among patterns. The following section describes four contexts with examples. Context helps to distinguish clearly among patterns and improve user's satisfaction. When the user puts the query at dashboard, underlying query manager identifies the context of the query and then forwarded its concern context wise arranged patterns. Context based pattern separating approach improves the searching by reducing the search space. One or more context can be hybridized for increasing the span of user's queries. We have presented a hybrid context based approach in section -5. Let us understand what context means:

Ex: User put the queries:

Then system must be able to identify:

- (i) Context of query: What kinds of patterns can satisfy the query like medical data pattern, university data pattern, stock data pattern, etc.
- (ii) What kind of data mining techniques able to give the answers.

The query manager receives the query and tries to give satisfactory answer. Efficiency and easiness depend on the way the patterns are stored. Pattern storage is not so easy as storing the raw data. We try to draw attention to what pattern are going to be stored and which kinds of pattern will be able to satisfy user queries. In this view, we are introducing four contexts:

Case 1: Global data context: Patterns are created and stored in a pattern warehouse (PW) without concerning the domain of underlying raw data, i.e., patterns from medical data, the university data, stock data, transactional data and from many more are stored collectively without any separation. This method loses the isolation of patterns.

Benefits:

- (i) Easy to store
- (ii) Easy to define schema for pattern storage.

Problems

- (i) Difficult to extract patterns, domain-wise
- (ii) Lose the isolation
- (iii) Query results may not be satisfactory
- (iv) Pattern retrieval will not be efficient.

Case 2: Domain data context: Patterns are created and stored in PW with concern domain of underlying raw data, i.e., patterns from medical data, the university data, stock data, transactional data and from many more are stored in such a way that they can be recognized and access specifically.

Benefits:

- (i) Easy to extract patterns, domain-wise
- (ii) Query results will be satisfactory to some extent

- (iii) Pattern retrieval will be efficient to some extent.
- (iv) Maintains the isolation at an abstract level.

Problems

- (i) Difficult to define schema for pattern storage.

Case 3: Scenario context: Patterns are created and stored in PW with concern domain of underlying raw data and its scenario also i.e., suppose, we have patterns from medical data. These patterns can be further separated scenario-wise like heart, cancer, diabetes or from any other scenario. We need to store in such a way that they can be recognized and access specifically scenario-wise.

Benefits:

- (i) Easy to extract patterns, scenario-wise
- (ii) Query results will satisfy the customer need
- (iii) Pattern retrieval will be efficient
- (iv) Maintains the isolation at a deep level.

Problems

- (i) Very difficult to define schema for such pattern storage.

Case 4: Techniques and kind of knowledge context: Patterns are created and stored in PW with concern underlying pattern retrieval techniques. i.e., patterns can be separated according to techniques like association patterns, clustering patterns, classification patterns, etc. We need to store in such a way that they can be recognized and accessed specifically techniques-wise.

Benefits:

- (i) Some customer queries can only be satisfied by specific DM technique
- (ii) Query results will satisfy the customer need.

Problems

- (i) Very difficult to define schema for pattern storage.

In some cases such as the data mart (it contains data of limited scope and focused on specific business function or region), inherently, patterns are extracted from data mart also represent that focuses business function only. So, we do not need to separate such patterns as per context-wise (i.e., case 1, 2, 3). In such cases, various kinds of pattern can be generated through different techniques like association, cluster, classification, etc. So we have introduced the fourth case (techniques-wise). The decision on selection of context is dependent on underlying application, user requirement, domain and data. The context can be hybridized to full fill application requirement.

5. Conceptual and logical modelling: Proposed

Pattern warehouse design process is a sequence of phases. It is common to start with *requirements analysis* and *specification*, then do conceptual design and logical design (Hüsemann *et al*

2000; Bouzeghoub *et al* 1999). We are giving our attention on the central issues: conceptual and logical schema design only. Context based conceptual or logical schema are not found. We proposed here conceptual designs (figure 2) with clear goals and objectives, such as completeness (all kinds of patterns), summarizability (ability to compute aggregate or derived pattern), and knowledge Independence (every pattern can be answered using the pattern warehouse only) (Mazón *et al* 2008).

Initially, pattern management concept and its issues were introduced in the PANDA report (Ilaria *et al* 2003). We are extending the definition and concept of pattern representation of PANDA report and incorporating in the proposed conceptual modelling as presented in figure 2. In the proposed schema, patterns are represented with triple (Pattern_Type, Pattern, Context):

Pattern_Type : A pattern type *pt* is a quintuple –
 $pt = (n, ss, ds, ms, f),$

where, *n* is the name of pattern type, *ss* (structure schema) is a definition of pattern space, *ds* (source schema) define related raw data space, *ms* (measure schema) quantify the quality, *f* is a formula that describes the relationship between context space and pattern space.

Example (Association rule): Pattern type for association rule is defined as

n: Association rule

ss: TUPLE(head: SET(STRING), body: SET(STRING))

Figure 2. PW conceptual design.

ds: BAG(transaction: SET(STRING))
 ms: TUPLE(confidence: REAL, support: REAL)
 $f: \forall x (x \in \text{transaction and } x \in \text{context source, i.e., transaction} \in \text{context source}).$

Pattern : Let $pt = (n, ss, ds, ms, f)$ is a pattern type . A pattern p instance of pt is a quintuple:

$$P = (pid, s, d, m, e),$$

where, pid - pattern identifier, s - is a value for type ss , d - dataset, m - is a value of type ms ,
 e - region of the source space.

Example:

pid: 001
 s: (head = { 'Laptop' }, body = { 'P3', 'SONY' })
 d: 'SELECT SETOF(article) AS transaction FROM
 sales GROUP BY transactionId'
 m: (confidence = 0.75, support = 0.55)
 e: {transaction: { ' Laptop', 'P3', 'SONY' } }

Context: It is defined as:

$$c = (cid, cn, cs, \text{pattern-type}, pc)$$

where,

cid- context identifier
 Pattern type
 cn – context name
 cs – context source
 pc- collection of pattern of type pt .

'Context' and 'Pattern type' are directly related to each other. In general, this relationship has the cardinality one -to-many, i.e., a context can correspond to more than one pattern type. On the other hand, 'Context and 'Pattern' are related indirectly through 'Context–Pattern' relationship. Context contains generic information about the pattern, such as the identifier, source, feature's name, etc. Pattern is specialized, according to the pattern type it belongs to, for example association rule patterns, cluster patterns, etc. we say that the data that are represented by a pattern form the image of the corresponding context. This Context oriented modelling of patterns is shown in figure 3.

Pattern warehouse cannot be designed the same ways as transactional-oriented operational database. The classical requirement gathering system cannot benefit much for the pattern warehouse conceptual design, but requirement driven is still important. Although the design process of pattern warehouse and OLAP are quite different (Inmon 2005). In this research work, we have extended the well known data warehouse schema 'Snowflake Schema' to this end (Levene & Loizou 2003). We have considered a medical database (as shown in table 1 (a)) which represents the patient and their symptoms of particular disease. For the simplicity, we have considered 'diabetes'. ID represents the patient unique identification number and S_i represents the symptoms associated with patients regarding diabetes only. Table 1(b) shows the frequency of each symptom. It helps to know which symptoms are most likely to appear. This medical database and

Figure 3. Context oriented modelling of patterns.

concern outcomes are used throughout the paper as an example. Diabetes patterns are generated through applying data mining techniques (association mining) on this database and then stored in a pattern warehouse.

The association type of diabetes patterns are represented as per the proposed conceptual schema in following ways: Pattern type for association rule and context: diabetes is defined as

n : Association rule
 ss :TUPLE(head: SET(STRING), body: SET(STRING))
 ds :Medical_DB (ID & Symptoms : SET(STRING))
 ms :TUPLE(confidence: REAL, support: REAL)
 f: $\forall x (x \in \text{Medical_DB and } x \in \text{Diabetes})$

Table 1. A medical database with frequency count.

ID	Symptoms	Symptom	Count
01	S ₁ , S ₂ , S ₃ , S ₅	S ₁	4
02	S ₂ , S ₃ , S ₄ , S ₅	S ₂	6
03	S ₁ , S ₃ , S ₅	S ₃	5
04	S ₁ , S ₂	S ₄	4
05	S ₁ , S ₃ , S ₅	S ₅	5
06	S ₂ , S ₄ , S ₅	S ₆	2
07	S ₂ , S ₄ , S ₆		
08	S ₂ , S ₄ , S ₆ , S ₃		

(a) Medical Database;

(b) Frequency count of 1- itemset

Table 2. Pattern warehouse with association patterns.

P_ID	P_SIZE	P_CONF	PATTERN
P101	1	3	S_1
P103	1	3	S_2
P104	2	2	$S_1 - S_2$
P105	2	2	$S_1 - S_3$
P106	3	2	$S_1 - S_3 - S_5$
P107	3	2	$S_2 - S_4 - S_5$
P108	3	3	$S_2 - S_4 - S_6$

Example :

pid: P101
s: (S_i)
d: 'SELECT S FROM Medical_DB GROUP BY 'PID'
m: (P_SIZE=1, P_CONF=3)
e: {Medical_DB, Context: diabetes}

The elementary view of the pattern warehouse for the association type pattern is shown in table 2 according to the initial pattern schema (*Ex. Association Rule*)- (P_ID , P_SIZE , P_CONF , $P_Patterns$). Table 2 contains four columns (P_id , P_Size , P_Conf and $P_Pattern$). P_ID (P101, P102, ...) represents the unique identification number of patterns. The last column 'Pattern' represents the real frequent patterns which satisfied measures like 'size' and 'confidence' as per column 2 and 3, respectively. Patterns with each value of measures (i.e. Size: 1 itemset, 2-item set, 3- itemset ... s ; confidence: 1,2,3, ... m) is stored in pattern warehouse. For simplicity, table 2 represents the association kinds of pattern with size (1,2,3) and confidence (2,3). End users can access patterns with any combination of measures as per their need. Pattern warehouse represented in table 2 is as per context 4 (kind of knowledge). These patterns can be considered as scenario wise (context 3) as well. In other words, patterns of table 2 are created from medical data and more specifically represents 'diabetes' concerning patterns. We are presenting hybrid (context 3 and context 4) context-wise patterns. Patterns contain knowledge like 'diabetes – association' patterns. The main objective of this section is only to present the clear picture of patterns, context and how it will then represent as snowflake schema. What kinds of diabetes knowledge are represented by patterns is out of scope of this work.

Figure 4 depicts how Snowflake schema is used for logical designing of pattern warehouse. The scenario of presented pattern is 'diabetes' and patterns are association type. The presented snowflake schema is well suitable to accommodate both scenario and pattern type in hybrid way to give logical design. So this schema can be viewed as 'association – diabetes' pattern schema. The following section describes each term in view of pattern warehouse only.

Dimension Table (Pattern Semantic): A dimension table and its normalized tables store *patterns*. In the proposed schema, each dimension represents a specific category of patterns. In contrast to data warehouse snowflake schema, here dimension table is normalized as per 'kind of knowledge' wise. This way of normalization allows making hybridization of various contextual based categories of pattern. It helps to represent the problems in a more realistic way.

Let us consider the proposed snowflake schema in figure 4. We are introducing two levels of hierarchy for '*kind of knowledge*'. First, kind of pattern, i.e., patterns is categorized

Figure 4. Snowflake schema for hybrid ‘association-diabetes’ patterns.

according to their underlying techniques (association rule, classification, clustering, etc.). Second, scenario of patterns, i.e., patterns is sub-categorized as per their underlying specific data context (scenario: heart, diabetes, blood, cancer, etc.). The presented hierarchy is backbone for normalization in this work. The kind of knowledge based normalization is flexible in terms of ordering. We can also categorize patterns first scenario and then techniques-wise. The hierarchy can be extended up to n- number of levels, but it may create problems at pattern access and maintenance time. Inherently, the warehouse is not designed for fine normalization so subdivision up to 2- levels is considerable. It must be noticed that the presented concept hierarchy of patterns is not as same as normalization in the transactional database. Typical normalization is a kind of vertical data partitioning, but the presented concept hierarchy is to group the patterns according to what kinds of knowledge they are contained. This concept is explained in figure 4 by first patterns are grouped as per techniques and then scenario-wise. For each technique (association, clustering, classification, etc.), there is a separate pattern table in pattern warehouse. Each table is uniquely identified by their primary key. So we have given the name of primary as same as concerned technique (association_key, clustering_key, classification_key, etc.). Next pattern tables are subdivided into scenarios. There can be n-number of scenario like cancer, diabetes, etc. So scenario tables are identified by their primary key (scenario_key_1, etc.). As we have mentioned that patterns are very semantic rich. We have to design PW system or pattern table specifically for individual type of patterns (association, clustering, classification, etc.). So, for the simplicity we have taken ‘association pattern’ as running example throughout the paper. This is why we have not discussed Cluster dimension and Classification dimension in details. As the way, discussed for association patterns, can be extended for cluster and classification. Cluster and other patterns can be subdivided into scenario-wise.

Fact table (Fact semantic): It is a central table in his schema. Fact table contains the primary keys of dimension tables. The primary key of the fact table is composite key that is made up of all of its foreign keys. In contrast with a fact table of data warehouse, here the values of fact table depend on the order of hierarchy. The presented snowflake schema can be used in a variety of ways to represent real world problems.

6. Quality forms

This section introduces four quality forms which are supposed to be considered as guidelines for the good schema design of pattern warehouse. This quality form concept can be considered as quality factors for pattern warehouse design (Vassiliadis 2000). Following section covers each quality forms in details.

1QF: Summarizability — First quality form ensures summarizability by giving the ability to compute aggregate or derived pattern from other existing patterns. Summarizability issue becomes important when patterns are aggregated during decision making. We insist to maintain the summarizability as 1QF in conceptual level so that the problems can be avoided when querying the pattern. There are two major issues with proposed first quality form: (1) the adequate representation of mapping between pattern semantics and (2) level of aggregation within the pattern semantic hierarchy. 1QF reduces the underlying computational cost and make PW more independent from source data (Data Warehouse). 1QF can be achieved through sequence of roll-up, roll-down, and aggregation etc. operations. These operations can be achieved through summarizability operator (SO).

Suppose:

$Ptn_{(30)}$: Represents the all patterns having threshold values equal to more than 30%.

Lets us consider, we need patterns with threshold value equal to more than 20% and PW does not contain such patterns. At this stage summarizability ensures to compute patterns with threshold value between 20–30 % because patterns with threshold value equal to more than 30% already available in PW. So asked pattern $DPtn_{(20)}$ can be derived by aggregating new pattern $NPtn_{(20-30)}$ with already available pattern $Ptn_{(30)}$.

$$\begin{aligned} \text{Derived Pattern} &= \{ (\text{New Pattern})\text{Summarizability Operator} (\text{Old Pattern}) \} \\ DPtn_{(20)} &= \{ (NPtn_{(20-30)})SO(Ptn_{(30)}) \} \end{aligned}$$

We are extending the concept of summarizability presented by (Lenz & Shoshani 1997) so that it can be accommodated in PW design as a quality factor. The necessary conditions for summarizability are:

1. Many-to-one relationship between semantic hierarchies must be modelled.
2. Many-to-one relationship should be full. This means, all values of parent level must be presented at lower levels.
3. Summarizability must be performed on type compatible semantics.
4. Guarantee to get consistent and reliable result after summarization.
5. Summarization is only pattern retrieval concerning property.
6. Violation of this property must be expressed in the schema.
7. Summarization is preferably important and to be implemented in the application layer.

8. Summarized pattern should be cached to improve performance. Cached patterns can be long lasting until underlying source data updated.
9. 1QF ensures that summarized pattern can be represented as pattern view.

We are proposing 1QF as most important for querying the patterns. Importantly, summarization in pattern retrieval depends on the pattern's (i) structure (ii) characteristics and (iii) semantic. This work is proposed to classify patterns according to context of data so a context dependency (Hurtado *et al* 2005) can be considered as a restricted kind of dimension constraint. Finally, we state that 1st condition indicates the deal with conceptual level and 2nd is data level.

2QF: Knowledge Independence — A pattern warehouse is in 2QF, if every pattern of the data warehouse can be answered using the pattern warehouse only. This quality form ensures that every knowledge is available 'on-demand'. This quality form also guarantees to zero knowledge loss. The motivation behind knowledge-independence property is to enable knowledge on-demand rather than analysis on-demand. Most of the time, analysis is time and resource consuming and too expensive. 2QF improves user experience and satisfaction.

Inherently, PW is also subject-oriented or more specifically context-oriented. PW is designed for satisfying specific queries. Variety of patterns can be extracted and stored in PW, but for achieving knowledge-independence PW, we have to be specific.

Let,

$$\begin{aligned} V_{pt} \sum_{k=1}^n C_k &= \{pt_1, pt_2, pt_3, \dots, pt_p\} \\ &= \sum_{k=1}^n \{pt_{1k}, pt_{2k}, pt_{3k}, \dots, pt_{pk}\} \end{aligned}$$

where

$\sum_{k=1}^n C_k$ = Set of n-numbers of contexts V_{pt} = Set of p-numbers of patterns and C = Context of patterns.

We are assuming that V_{pt} is able to definitely answer any kinds of user's need. Items in the set V_{pt} will play an important role to achieve a knowledge-independence property. Now the question is 'How will you decide that what patterns must be in V_{pt} ?', i.e., which and what kinds of patterns need to be stored in V_{pt} ? The simple solution is 'efficient requirement analyses'. Normally, PW is designed for specific domain or context (as presented in section 4). So by proper meetings with target users, and understanding their needs, we can easily find out probable patterns of V_{pt} . Next, the elements and their context of V_{pt} must be verified as schema at the stage of conceptual design. Then definitely, user queries can be answered using PW i.e., PW becomes knowledge-independence.

3QF: Self-Materialability — A PW is in 3QF, if the system is able to compute new instance of pattern after every source data update only through: (i) an older instance of the pattern and (ii) new updated information.

This quality form makes the pattern warehouse more independent from data warehouse. Materialability is also known as update-independence quality. To achieve the materialability is a computational intensive task. We are presenting a clear picture of the update-independence in view of the very semantic pattern warehouse by taking association patterns.

Let's consider,

$$Pt = \sigma_{Dmm} Measures.$$

Measures are a set of 'k' elements. Measures may vary as per underlying data mining methods (Dmm). Suppose there are two measures (size and confidence) for association rule mining (ARM) patterns (Tiwari *et al* 2010).

$$\text{Measure} = \{\text{Size}, \text{Confidence}\}_{K=2}$$

Then,

$$\text{Pt} = \sigma_{ARM} \{(\text{Size}) \bowtie (\text{Conf})\}$$

Size and *Conf* are set of 'n' and 'm' elements, respectively.

$$S_{\text{size}} = \{s_1, s_2, s_3, \dots, s_n\} \quad (\text{i})$$

$$C_{\text{confi}} = \{c_1, c_2, c_3, \dots, c_m\} \quad (\text{ii})$$

$$\begin{aligned} \{(\text{Size}) \bowtie (\text{Conf})\} &= \{s_1, s_2, s_3, \dots, s_n\} \bowtie \{c_1, c_2, c_3, \dots, c_m\} \\ &= \{(s_1, c_1), (s_1, c_2), \dots, (s_1, c_m), (s_2, c_1), \\ &\quad (s_2, c_2), \dots, (s_2, c_m), \dots, (s_n, c_1), \\ &\quad (s_n, c_2), \dots, (s_n, c_m)\} \end{aligned} \quad (\text{iii})$$

We are storing values of each set as matrix form in the presented pattern warehouse, as shown below. Measure matrix can be extended as multidimensional matrix to accommodate more measures.

$$\begin{pmatrix} (s_1, c_1) & (s_1, c_2) & \dots & (s_1, c_m) \\ (s_2, c_1) & (s_2, c_2) & \dots & (s_2, c_m) \\ \vdots & \vdots & \ddots & \vdots \\ (s_n, c_1) & (s_n, c_2) & \dots & (s_n, c_m) \end{pmatrix} n \times m$$

The above matrix represents the view with two measures, i.e., size and confidence. The numbers of measures are variable and dependent on kinds of techniques applied to extract patterns or applications. Let's consider Eq. 1, one more measure (support) can be added like:

$$\text{Pt} = \sigma_{ARM} \{(\text{Size}) \bowtie (\text{Conf}) \bowtie (\text{Sup})\}_{k=3}$$

So, the presented matrix needs to be extended in a multidimensional way to accommodate additional measures.

Update representation

When an update (ΔS or ΔC) received then:

- (i) The context of the update is identified
- (ii) Only concerned patterns need to re-compute in pattern warehouse
- (iii) Updates the patterns and make changes permanently as a batch

Since the pattern warehouse consists the patterns in context-wise, so a small section of pattern warehouse need to access without disturbing rest of the parts.

Let us consider,

ΔS_x : Represents update in term of size, i.e., x-itemset patterns are populated.

Then, Suppose , $x = 2$,

$$S'_2 = S_2 + \Delta S_2,$$

where S'_2 is now re-computed pattern

The Eq. (i) becomes as

$$S_{size} = \{s_1, s'_2, s_3, \dots, s_n\} \quad (iv)$$

The equation (iii) become as :

$$\begin{aligned} \{(Size) \bowtie (Confi)\} &= \{s_1, s'_2, s_3, \dots, s_n\} \{c_1, c_2, c_3, \dots, c_m\} \\ &= \{(s_1, c_1), (s_1, c_2), \dots, (s_1, c_m), (s'_2, c_1), (s'_2, c_2), \dots, \\ &\quad (s'_2, c_m), \dots, (s_n, c_1), (s_n, c_2), \dots, (s_n, c_m)\} \end{aligned} \quad (v)$$

$$\text{Already computed patterns: } \left\{ \begin{array}{l} (s_1, c_1), (s_1, c_2), \dots, (s_1, c_m), (s_3, c_1), (s_3, c_2), \dots, \\ (s_3, c_m), \dots, (s_n, c_1), (s_n, c_2), \dots, (s_n, c_m) \end{array} \right\} \quad (vi)$$

$$\text{Newly re-computed patterns (Pt')} : \{(s'_2, c_1), (s'_2, c_2), \dots, (s'_2, c_m)\} \quad (vii)$$

The proposed context-wise method allows re-computing only for few patterns.

$$Pt' = \{S'_2\} \bowtie \{c_1, c_2, c_3, \dots, c_m\}$$

$$Pt' = (s'_2, c_1), (s'_2, c_2), \dots, (s'_2, c_m)$$

$$Pt' = \sigma_{Dmm} \{S'_2\} \bowtie \{c_1, c_2, c_3, \dots, c_m\}$$

The proposed method for updating pattern warehouse as described in the above section allows populating only recomputed patterns (Pt'). We do not need to compute the remaining patterns as in equation (vi). So the presented method is very efficient.

4 QF (Pattern ->Source Mapping): Source data and pattern are two end points of PW design. 4QF enables the system to define mapping between pattern to source data and vice versa. There are various complexity and constraints to implement 4QF. This quality form is simply a reverse engineering. 4QF allows us to go from pattern to source data. There are various complexity and constraints to implement 4QF.

7. Discussion

We have presented context-based conceptual and logical modelling for pattern warehouse which serve as the foundation for physical design and make business decisions to better understand and forecast. The context-based pattern separation helps us to manage and retrieve more specific patterns efficiently. We argue that it is better to design context-oriented pattern warehouse for maximizing user satisfaction because it represents real world problem in a better way to both users and designers. The span of pattern warehouse can be increased by adopting a hybrid context approach. Hybrid context modelling can be implemented through snowflake schema because inherently, snowflake allows to normalization. We have extended this normalization as a way of context separation. This is not like a vertical partition of data, but it marks a fine separation among contextual pattern. There are also introduced basic but important guidelines as quality

forms (QF). The pattern refreshment issue is discussed by introducing a matrix-based approach. The matrix allows identifying updated patterns efficiently. The presented approach makes the clear distinction between newly re-computed patterns and old one. The presented approach identifies the portion of the pattern warehouse which needs to be updated without disturbing remaining portions.

8. Problems associated with pattern warehouse

As the pattern warehousing is a new emerging technology, it has too many risks. We have listed the risks with pattern warehouse in the initial phase as given below:

- (i) The scope and objective of pattern warehouse must be clear. Like data warehouse, pattern warehouse is also subject-oriented. We must be ware about what kinds of patterns are going to be stored. Patterns are created for specific purpose i.e., patterns of sales, association among the sold items, sales patterns geographical-wise, etc. This means, various kinds of patterns extracted from same data. So if the scope and objective is clear, then its helps to design more efficient pattern warehouse management system.
- (ii) Patterns are semantically very rich. Extra care is required for its management. Meta data of pattern warehouse must be organized in more efficient way.
- (iii) Pattern representation must be realistic. Wrong pattern representation leads to big failure at the end. The adopted schema design must be tested and validated. Adopted schema design should fulfill the scope of the project.
- (iv) Missing of end user communication may lead to big failure. So end user communication must be involved in the design of pattern warehouse. User's requirement must be properly understood and drafted (Inmon 2005).
- (v) Data source to pattern mapping must be implemented in pattern schema in realistic way to validate and update patterns time to time. As data updated, pattern must be updated accordingly and this is called pattern refreshment.
- (vi) Poor quality of data can cause problems.

9. Conclusions and future work

The research work has shown that even if several proposals exist, but in terms of practical feasibility, the conceptual and logical design of pattern warehouse is still missing. We have presented a context-based conceptual and snowflake-based logical modelling in this paper. We have discovered four quality-forms as a road map for better pattern warehouse design and help to minimize the evaluation and maintenance cost. Research work helps and guides to develop pattern management system in an effective and efficient way. This paper tries to make clear understanding about the need of pattern warehouse. We cover all the aspects about how pattern warehouse is different from data warehouse and the current research progress of pattern warehouse. We have introduced '*kind of knowledge*' wise context based hierarchy which is a backbone behind the proposed snowflake-based logical design of pattern warehouse. We have extended a well-known and tested snowflake schema to accommodate persistent patterns in a logical way. We have also tried to draw attention on pattern refreshment issue and introduced a matrix based approach. The presented method is efficient because it re-computes only concerned patterns and allows other patterns to continue to be available for users. More detailed discussion

is required in terms of physical implementation feasibility and techniques. For simplicity, association kinds of patterns are taken in example. The work can be further extended to incorporate other data mining patterns such as classification, cluster, decision tree. The architecture is presented in such a way that it can also handle or incorporate other kinds of pattern like pattern in sequence, in number, in graph, in image, in signal, etc.

References

- Barbara C and Anna M 2005 PSYCHO: A prototype system for pattern management. In: *Proceeding of the 31st International Conference on Very Large Data Bases (VLDB)*, (pp)1346–1349, Trondheim, Norway, ACM
- Batra D 2005 Conceptual data modelling patterns: Representation and validation. *J. Database Management (JDM)* 16(2): 84–106 IGI Global
- Bouzeghoub M, Fabret F and Matulovic-Broqué M 1999 Modelling the data warehouse refreshment process as a Workflow Application. In: *Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW)*, 19(6)
- Bartolini I, Ciaccia P, Ntoutsis I, Patella M and Theodoridis Y 2004 A unified and flexible framework for comparing simple and complex patterns. In: *Proceedings of ECML-PKDD'04, LNAI 3202*, 496–499: Springer Berlin Heidelberg
- Catania B, Maddalena A, Maurizio M, Bertino E and Rizzi S 2004 A framework for data mining pattern management. In: *Proceeding of 8th European Conference Knowledge Discovery in Databases: PKDD*, Pisa, Italy, 87–98, Springer, Berlin Heidelberg
- Evangelos K and Irene N 2005 Database support for data mining patterns. In *Proceedings of the 10th Panhellenic conference on Advances in Informatics*, 14–24, Springer, Berlin Heidelberg
- Giorgini P, Rizzi S and Garzetti M 2005 Goal-oriented requirement analysis for data warehouse design. In: *Proceedings of the 8th ACM International Workshop on Data warehousing and OLAP* (pp). 47–56 ACM
- Golfarelli M, Rizzi S and Cella I 2004 Beyond data warehousing: What's next in business intelligence? In: *Proceedings of the 7th ACM International Workshop on Data warehousing and OLAP* (pp). 1–6, Washington, DC, USA, ACM
- Hurtado C A, Gutiérrez C and Mendelzon A O 2005 Capturing summarizability with integrity constraints in OLAP. *ACM Trans. Database Syst.* 30(3): 854–886
- Hüsemann B, Lechtenbörger J and Vossen G 2000 Conceptual data warehouse design. In: *Proceedings of the International Workshop on Design and Management of DataWarehouses (DMDW)*, (pp) 3–9, Stockholm, Sweden
- Ilaria B, Elisa B, Barbara C, Paolo C, Matteo G, Marco P and Rizzi S 2003 Patterns for Next-generation Database systems: preliminary results of the PANDA project. In: *Proceeding the Eleventh Italian Symposium on Advanced Database Systems, SEBD 2003*, Cetraro (CS), Italy
- Inmon W H 2005 Building the Data Warehouse, 4th edition, John Wiley and Sons, Inc., New York
- Jaesoon P, Youngwok K and Youngmin C 2002 The concept of pattern warehouse and contemplate an application in integrated network data ware. *Telecommunication Network Lab., Korea Telecom*, <http://www.knom.or.kr/knom-review/v4n2/1.pdf>, Accessed on 11/Aug/2014
- Levene M and Loizou G 2003 Why is the snowflake schema a good data warehouse design? *Information Systems* 28(3): 225–240
- Lenz H J and Shoshani A 1997 Summarizability in OLAP and statistical data bases. In: *Proceedings of Ninth International Conference on Scientific and Statistical Database Management*. (pp) 132–143). IEEE
- Mazón J N, Lechtenbörger J and Trujillo J 2008 Solving summarizability problems in fact-dimension relationships for multidimensional models. In: *ACM 11th International Workshop on Data Warehousing and OLAP (DOLAP 08)*, Napa Valley, USA, (pp) 57–64
- Michael Eldridge 2010 Enterprise Data Warehouse: A Patterns Approach to Data Integration, Microsoft IT showcase, ©2010, Microsoft Corporation. <http://www.microsoft.com/technet/itshowcase>

- Mohammad R, Keivan K, Reda A and Mick J R 2009 Data modeling for effective data warehouse architecture and design. *Int. J. Inf. Decision Sci.* 1(3): 282–300 Inderscience
- Manolis T, Vassiliadis P and Spiros S 2007 Modelling and language support for the management of pattern-bases. *Data & Knowledge Eng.* 62(2): 368–397
- Manolis T and Vassiliadis P 2003 Architecture for pattern base management systems. Department of Electrical and Computer Engineering. *PANDA workshop, National Technical University of Athens*
- Riccardo A, Elena C, Monica D M, Franca G and Marina M 2011 Context dependent semantic granularity. *Int. J. Data Mining, Modeling and Management (IJDMMM)* 3(2): 189–215 InderScience
- Rizzi S 2004 UML-Based conceptual modeling of pattern-bases. In: *Proceedings of the International Workshop on Pattern Representation and Management (PaRMA)*, Heraklion, Hellas
- Rizzi S, Bertino E, Catania B, Golfarelli M, Halkidi M, Terrovitis M, Vassiliadis P, Vazirgiannis M and Vrahnos E 2003 Towards a logical model for patterns. In: *Proceeding of ER Conference*, Chicago, IL, USA, (pp) 77–90, Springer, Berlin Heidelberg
- Tiwari V, Gupta S and Tiwari R 2010 Association rule mining: A graph based approach for mining frequent itemsets. In: *Networking and Information Technology (ICNIT), 2010 International Conference*, (pp) 309–313, IEEE
- Tiwari V and Thakur R S 2014 P2ms: A Phase-Wise Pattern Management System for Pattern Warehouse. *Int. J. Data Mining, Modeling and Management (IJDMMM)*, Inderscience
- Vazirgiannis M, Halkidi M, Tsatsaronis G and Vrahnos E 2003 A Survey on Pattern Application Domains and Pattern Management Approaches. *PANDA Technical Report TR- 2003-01*, Available at, <http://dke.cti.gr/panda>
- Vassiliadis P 2000 *Data warehouse modeling and quality issues*. National Technical University of Athens Zographou, Athens, GREECE
- Zdenka T 2012 Data modeling and ontological semantics. *Int. J. Data Analysis Techniques and Strategies* 4(3): 237–255 Inderscience