Intelligent Data Analysis 21 (2017) 117–133
DOI 10.3233/IDA-150434
IOS Press

Incremental maintenance of discovered association rules and approximate dependencies

Alain Pérez-Alonso[a,*], Ignacio J. Blanco Medina[b], Luisa M. González-González[a] and José M. Serrano Chica[b]
[a]*Department of Computer Science, University "Marta Abreu" of Las Villas, Villa Clara, Cuba*
[b]*Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain*
[c]*Department of Computer Science, University of Jaén, Spain*

Abstract. Association Rules (ARs) and Approximate Dependencies (ADs) are significant fields in data mining and the focus of many research efforts. This knowledge, extracted by traditional mining algorithms becomes inexact when new data operations are executed, a common problem in real-world applications. Incremental mining methods arise to avoid re-runs of those algorithms from scratch by re-using information that is systematically maintained. These methods are useful to extract knowledge in dynamic environments. However, the implementation of algorithms only to maintain previously discovered information creates inefficiencies. In this paper, two active algorithms are proposed for incremental maintenance of previous discovered ARs and ADs, inspired by efficient computation of changes. These algorithms operate over a generic form of measures to efficiently maintain a wide range of rule metrics simultaneously. We also propose to compute data operations at real-time, in order to create a reduced relevant instance set. The algorithms presented do not discover new knowledge; they are just created to efficiently maintain previously extracted valuable information. Experimental results in real education data and repository datasets show that our methods achieve a good performance. In fact, they can significantly improve traditional mining, incremental mining, and a naïve approach.

Keywords: Association rules, approximate dependencies, knowledge maintenance and active databases

1. Introduction

Mining Association Rules (ARs) and Approximate Dependencies (ADs) are important research fields among several data mining techniques. It aims at discovering useful correlations or tendencies among attributes, hidden in data repositories. Many algorithms have been proposed to improve the mining process and create more efficient methods [5,14,21,32,44]. However, these proposed algorithms could become expensive when dealing with huge amounts of data, commonly stored in data warehouses or very large and big databases.

The knowledge discovered by ARs and ADs methods is specific for the current stage of the repository in which they were run. In real-world applications, data repository is not static and records are commonly

*Corresponding author: Alain Pérez-Alonso, Center for Informatics Studies, Faculty of Mathematic, Physic and Computer Science, University "Marta Abreu" of Las Villas, Villa Clara, Cuba. Tel.: +535 321 3880; E-mail: apa@uclv.edu.cu.

inserted, updated or deleted, following real activities in the universe of discourse. These continuous changes can render the measures of rules inexact and eventually invalid [24].

The need for incremental mining of rules arises to avoid having algorithms to re-run from scratch and re-scan the whole data. This is specifically useful when real-time data information is required. Example applications can be found in the field of data streams like web click stream data, sensor networks data, and network traffic data [27,28,41]. Emerging research in big data offers similar issues in association with velocity and volume [35,43]. At this time, many research efforts are being made to improve the performance [18,22,24,26]. These efforts will reduce the problem of updating ARs to find the new set of large itemsets and share an intermediate maintenance form (frequent itemsets) for this goal [13].

In this work, the update problem is focused on maintaining the measures of previously discovered rules, covering the decision-makers needs for real-time data information. This approach can also be helpful to the refine rules processes, at post-mining stage [6]. Our perspective neither removes previously discovered rules or adds new ones; it just efficiently update initial mined rules measures. A splitter form of rule is defined, allowing rules direct maintenance in a wide range of literature metrics [15,23], and simultaneously, maintaining these metrics in an efficient way. The maintenance problem is handled from a change computation point of view. The process of change computation deals with modifications induced by data operations; it is an important field in active database systems [42]. Additionally, the materialized view maintenance and the integrity constraint checking are significant fields of active systems that provide multiple methods [8,9,20,31]. Two algorithms, inspired in these methods, are proposed in this paper. One of them is intended for rules immediate maintenance and the other for rules deferred maintenance. Both algorithms reuse previous results incrementally to avoid measures calculations from scratch. So far, there is no method to maintain ARs and ADs from this perspective.

Experimental results are obtained from active relational databases with real educational data and repository datasets. They show that the proposed algorithms achieve good performance and improve classical mining, incremental mining and a naïve approach significantly. The proposed algorithms are being implemented and compared in two of the most used open source database management systems.

The main contribution of this paper is twofold. First, we propose two efficient algorithms to directly maintain the previously discovered rules, a very well addressed research from [11]. Second, from a deferred perspective, we propose to consider at real-time the interactions between data operations in order to create a reduced relevant instance set. A common characteristic of the proposed algorithms is the efficient maintenance of existing rules, keeping their real-time measures available. It is made without having access to the database itself, making self-maintainable approaches [12,20].

The remainder of this paper is organized as follows. The next section defines basic concepts and describes the current research problem. Section 3 presents a change computation scope with integrity constraints and materialized views for ARs and ADs. Section 4 describes the proposed algorithms, including an initial naïve approach. In Section 5 we briefly review related work and compare it with our approach. Section 6 presents the experimental results of the proposed and reviewed methods for the performance evaluation. Finally, Section 7 concludes this paper summarizing the results of our work.

2. Problem statement

This section defines concepts used in this paper and describes the research problem. ARs and ADs are different data relationships that share some similarities [30]. These two data dependencies are referred to as Data Rules (DRs) in the remainder of this paper for a common reference.

2.1. Association rules definition

Association rules can formally be represented as an implication of itemsets (sets of items) in transactional databases [1]. Let $It = \{It_1, It_2, \ldots, It_m\}$ be a non-empty set of m distinct attributes. Let T be the transaction scheme that contains a set of items such that $It \subseteq T$. An AR is an implication of the form $X \Rightarrow Y$ where $X, Y \subset It$ such that $X \neq \emptyset$, $Y \neq \emptyset$ and $X \cap Y = \emptyset$. In this statement X and Y are called rule itemsets and they are the antecedent and consequent of the rule, respectively.

There are two important classical parameters to measure using the association rules: support and confidence. Support is defined as the fraction of records that contains $X \cup Y$ in all records. Confidence is the fraction of the transactions that contains $X \cup Y$ in records that contain X.

Alternative metrics are very well established [23] and they solve some drawbacks associated with the original indicators. An example is the certainty factor (*CF*) [4] in Eq. (1) that is a confidence alternative. The certainty factor takes values in $[-1, 1]$. It is positive when the dependence between X and Y is positive, 0 when they are independent, and a negative value represents negative dependence.

$$CF(X \Rightarrow Y) = \begin{cases} \dfrac{Conf(X \Rightarrow Y) - Supp(Y)}{1 - Supp(Y)}, & Conf(X \Rightarrow Y) > Supp(Y) \\ \dfrac{Conf(X \Rightarrow Y) - Supp(Y)}{Supp(Y)}, & Conf(X \Rightarrow Y) < Supp(Y) \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Association Rules were first studied in market basket data, where each basket is a transaction containing the set of items bought by a client. In a relational database context, it is usual to consider that items are pairs <*attribute*, *value*> and transactions are tuples in a relation.

2.2. Approximate dependencies definition

Approximate dependency rules (also called partial determinations or approximate functional dependencies) are related to the satisfaction of a normal function dependency f: $X \rightarrow Y$ over a relation r. An AD requires that f is hold in almost all records of r. In other words, the ADs allow very small portions of records of r that violates f [29]. This exception or error of f is commonly used to calculate the approximate measures, representing the fraction of records that must be removed for $X \rightarrow Y$ to be assured [19].

There are many possible ways for defining the approximateness of data dependencies. These methods are summarized and compared in [14] being g_3 measure [21] widely used. Other studies exist that unify ADs and ARs [30,37]. In [37] a new definition of ADs is shown by the representation of ADs as ARs in the new transaction set. Here, the introduced set T_r contains r^2 tuples, $|T_r| = |r \times r| = n^2$ considering $|r| = n$. This measures ADs more accurately based in pairs of tuples. We consider ADs under this definition without losing any generality.

2.3. Data rules maintenance problem

Real-world applications can be affected by many data operations. A common approach in active databases known as event-condition-action rules [34] defines these data operations at event occurrences that can be primitive or composite. The primitive type, called primitive structural event (PSE), is a single low-level event. A composite type is a combination of multiple primitive or composite structural events

(CSE). These events produce a database transition (DB_{s0}, DB_{s1}) where DB_{s0} is the initial state and DB_{s1} is the final state.

Three types of PSE are considered: inserting a record Δt^{+}, deleting a record Δt^{-}, and updating a record Δt^{-+}. An update event can be seen like an independent event in which t_0^{-+} is the record before the update event corresponding at DB_{s0} state and t_1^{-+} is the record after update in the DB_{s1} state.

Let us consider an initial database state DB_i and a composite structural event over this state that produces a database transition (DB_i, DB_f). Let $DRS_i = \{(DR_1, DRM_1^i), (DR_2, DRM_2^i), \ldots, (DR_n, DRM_n^i)\}$ be a set of mined data rules and their measure value, discovered in DB_i state. This paper focuses on efficiently finding the new measures of discovered data rules in the final state DB_f. In other words, the data rules maintenance problem can be reduced to find a data rule set $DRS_f = \{(DR_1, DRM_1^f), (DR_2, DRM_2^f), \ldots, (DR_n, DRM_n^f)\}$.

3. Rules maintenance under the change computation scope

Change computation is the capability to compute data modification operations. Their methods have been accepted in active databases, materialized view maintenance, and integrity constraint checking [42]. All these methods share similar characteristics like definition of modifications to be monitored, computation of changes, and reaction to defined changes.

Materialized views, integrity constraints, and DRs share the capacity of reflecting data information, but for different purposes. In this section we present DRs under the change computation scope, allowing and formalizing the use of those methods in the data rules maintenance problem.

3.1. Data rules and materialized view

Views define derived data, which can be materialized in database systems. The process of keeping these views up-to-date in database transitions is called materialized view maintenance [16]. Let a view be defined by query Q and materialized in MV. Any correct materialization of MV in a database state DB_s always returns the same data as Q: $MV(DB_s) = Q(DB_s)$. Specifically, the materialization of a view must be equivalent to its querying: $MV \equiv Q$. An important aspect to materialize a view is the speed of its querying, a desirable quality when response time is critical. If tm_1 is the time for computing $MV(DB_s)$ and tm_2 the time for computing $Q(DB_s)$ then, in general terms, $tm_1 \ll tm_2$.

Just like a view DRs define some data information, but in a particular way because they expose attributes correlated in an implication form. A rule of the form $X \Rightarrow Y$ establishes, with some measure, that when X occurs so does Y. If quantitative ARs [39] are considered, then their attributes are the triplet $<l, a, u>$ representing $l \leqslant a \leqslant u$ with r, s numbers of attributes in X, Y respectively. An example of a such rule can be defined by a selection condition SC in Eq. (2).

$$SC \equiv \bigwedge_{i=1}^{r} l_i \leqslant a_X i \leqslant u_i \wedge \bigwedge_{j=1}^{s} l_j \leqslant a_Y j \leqslant u_j \tag{2}$$

Such a condition defines a query over the data repository that selects the co-occurrence of rule antecedent and consequent. Materializing similar queries can be used for directly maintaining DRs measures in the rule base and not to materialize an intermediate form (itemsets) or data mining views [3].

3.2. Data rules and integrity constraint

Integrity is a mandatory property for a relational database and it is associated with two components: validity and completeness. Validity represents the truth of data, completeness represents the totality of relevant data, and integrity constraints are conditions that guarantee its satisfaction at any time [33].

Each state of a database must satisfy all integrity constraints. Let *ICS* be a set of integrity constraints in denial form and *F* a boolean function that evaluates if any constraint violates or not a database state. A consistent database state of DB_s guarantees that all integrity constraints in *ICS* evaluated with *F* must return false: $\forall ic \in ICS\ (F(ic, DB_s) = false)$. A correct database transition must have a final consistent state. Integrity constraints checking methods are aimed at holding the database's integrity [9].

DRs only represents data knowledge and never deny any database state. Nevertheless, they have an inherent restriction: they measure threshold. The measure evaluates the level of interest of the rules while its perspectives establish a minimum acceptable value that defines DRs existence. Let DRS_s be a set of mined DRs in a database state DB_s, *G* a function that evaluates the measure of a DR and δ the minimum threshold. A correct *DRS* set guarantees that all DR evaluated with *G* must return a value equal to or greater than δ: $\forall dr \in DRS\ (G(dr, DB_s) \geqslant \delta)$. The DRs incremental maintenance goal is not to maintain the measure threshold, but it can be part of the efficient evaluation of *G*.

4. Data rules maintenance proposals

Many research activities propose measures of rules with different properties, and their number is overwhelming [15,23]. Existing measures for DRs are usually defined by counting a total number of records that satisfy some condition. These conditions are generally associated with the antecedent, consequent, rule examples, and counterexamples among others [15,23].

In our proposals, the DR measure is considered a set of k distinct measure-parts $DRM = \{Mp_1, Mp_2, \ldots, Mp_k\}$ in which each item represents a different part of the measure formula. Measure parts must be atomic, it means that they cannot be divided into smaller items and still bring the same measure value. For example, confidence can be split in two parts: count of (antecedent $\cup$ consequent) and count of records. On the other hand, the certainty factor Eq. (1) needs four parts: count of antecedent, count of consequent, count of (antecedent $\cup$ consequent), and count of records. In this way, it is possible to efficiently maintain several metrics at the same time because metrics share some measure-parts. For example, following [23] is possible with only five distinct measure-parts to maintain 20 measures simultaneously. The final data rule measure is a formula over *DRM* parts.

Three methods are considered for rule maintenance. The first one consists of a naïve approach. This initial strategy is later enhanced from a change computation perspective in an immediate and a deferred way thus becoming the second and the third methods. The improvements are oriented towards two essential points: rules to be maintained and data instances to be analyzed.

4.1. Naïve approach

One way in which a naïve strategy may arise is via database queries. By this form each item of *DRM* is obtained following Eq. (2) as a query over all data: $Mp_1 = Qp_1, Mp_2 = Qp_2, \ldots, Mp_k = Qp_k$. As a result, previous information of measure-parts is not used to recalculate new values. Rule base is updated from scratch after each primitive or composite structural event takes place.

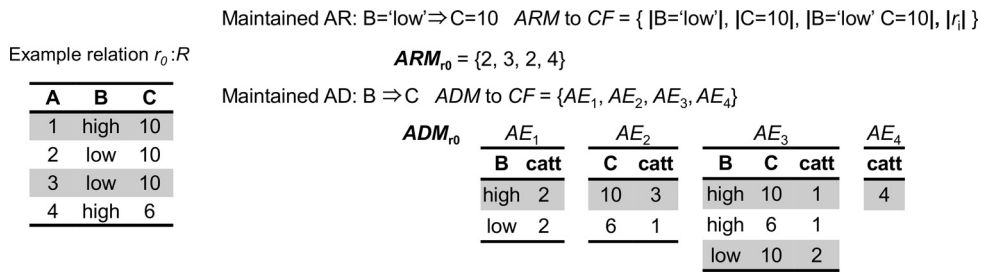

Fig. 1. Example of measure-parts structures.

These queries are quite different depending on if they are devoted to maintain ARs or ADs. In the case of ARs, *count* queries represent how many records satisfy a part select condition. For ADs *group by* queries are needed according to the efficient calculation of measures in [37]. A *group by* query allows exploring in the set of n records instead of exploring the corresponding set of n^2. For both, measure-parts take an integer positive value $Mp_1, Mp_2, \ldots, Mp_k \in \mathbb{Z}^+$.

The naïve approach involves maintaining all rules after each data operation and querying the entire data. Additionally, many irrelevant instances are analyzed which results in a waste of time. The only attractive aspect of the naïve approach involves not introducing extra manipulation as a reaction of database operations and the simplicity of their implementation. However, this is irrelevant when the stored data is large and *count* or *group by* queries become highly inefficient.

4.2. Measure parts structures for ARs and ADs

This issue is handled by maintaining measure-parts incrementally. However, due to the differences between ARs and ADs, naïve queries are necessary to define two measure-parts structures. For the remainder of this paper, measure-parts defined specifically for one AR and one AD will be *ARM* and *ADM*, respectfully. Measure parts for an AR have the same definition as the general DR measure-parts, $ARM = \{Mp_1, Mp_2, \ldots, Mp_k\}$ taking each measure part an integer positive value $Mp_1, Mp_2, \ldots, Mp_k \in \mathbb{Z}^+$.

Incremental view maintenance provides interesting solutions for *group by* queries [36]. These solutions are redefined for specifically maintaining *ADM*. $\forall MP_k \in ADM$ one auxiliary relation AE_k is defined, then $ADM = \{AE_1, AE_2, \ldots, AE_k\}$. The attribute list of each *AE* scheme contains the attributes of the base relation involved in that measure part plus a count attribute (*catt*). The *catt* attribute reflects how many groups of attribute values are in the base relation.

An example of ARs and ADs structures are provided in Fig. 1. The r_0 base relation will share all the examples of this paper, and maintenance will be achieved over these structures in an immediate and a deferred way.

The incremental maintenance of measure-parts structures allow obtaining the level of interest of the rule without re-scanning the whole data, an important aspect when managing large amount of data.

4.3. Immediate incremental maintenance proposal

An immediate approach is oriented to update the rule base immediately after the event takes place, in an active fashion. This approach verifies the specific rules that must be updated and it computes only the changes made by a primitive structural event. It means that only one record can be checked at a time. Incremental view maintenance algorithms offer multiple solutions. Specifically, a counting algorithm

for view maintenance [16] provides an interesting perspective. The following Algorithm 1 presents the proposed immediate incremental maintenance where rule measures are updated for data operations.

Algorithm 1 Immediate incremental maintenance for a DR

Input: A composite structural event *CSE* that modifies the attributes related in list *L*, and measure-parts *DRM* of $X \Rightarrow Y$ data rule.
Output: Updated measure-parts *DRM*.
Method:

```
for all PSE ∈ CSE do
    if (PSE = Δt^{-+}) then                                   ▷ update event
        if (L ∩ {X ∪ Y} ≠ ∅) then
            for all Mp_k ∈ DRM do
                if (L ∩ {involved attributes in Mp_k} ≠ ∅) then
                update Mp_k, increment with t_0^{-+};
                update Mp_k, decrement with t_1^{-+};
                end if
            end for
        end if
    else if (PSE = Δt^{+}) then                               ▷ insert event
        for all Mp_k ∈ DRM do
          update Mp_k, increment with t^{+};
        end for
    else                                                      ▷ delete event (PSE = Δt^{-})
        for all Mp_k ∈ DRM do
          update Mp_k, decrement with t^{-};
        end for
    end if
end for
```

Not all rules must be checked in update events. It is only when the values of the attributes are changed or their unknown value changes, especially $L = \{a \in R | t_0^{-+}[a] \neq t_1^{-+}[a]\}$. Not all parts of *DRM* need to be recalculated either. For example, if an update operation modifies only the antecedent attributes of a rule, then it is not necessary in certainty factor measure Eq. (1) to recalculate the part of the consequent. Through these conditions, maintenance can avoid unnecessary rules updates. In an insert or delete event, all record attributes are affected and always change measures of rules (except when they have unknown status).

The measure-parts MP_k are constantly updated in the proposed algorithm. However, the incrementing of measure-parts is quite different depending on the DR type. To illustrate our immediate proposal for ARs let us look at the following example in Fig. 2. Here, three structural events modify the r_0 relation and immediately, for each event, a previous algorithm maintains the *ARM* set. These structural events share all the examples of this paper.

Another example, in Fig. 3, illustrates the immediate proposal for ADs. It is not difficult to follow the Algorithm 1 for each PSE and the modification of all AE_k auxiliary relations in $ADMr_i$ set.

Finally, to obtain AD measure each measure part $MP_k \in ADM$ is calculated by aggregating the *catt* attribute in AE_k. For this immediate proposal, rule base refreshing is made without accessing base relation.

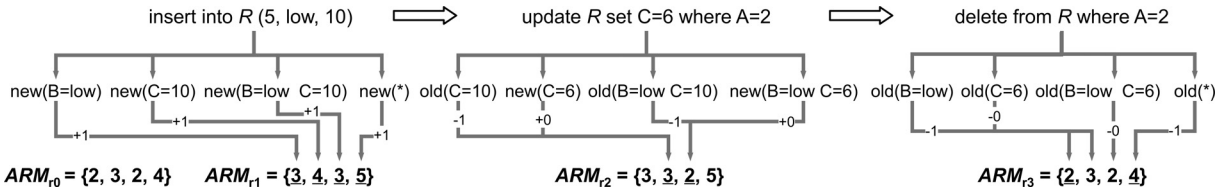

Fig. 2. Example of immediate incremental maintenance for an AR.

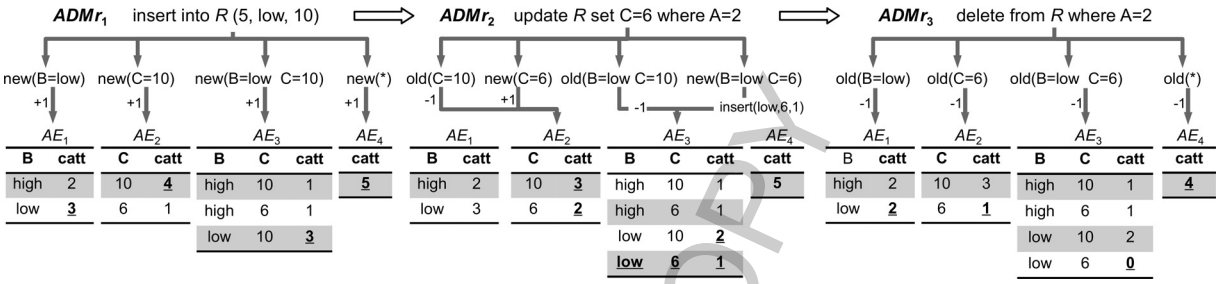

Fig. 3. Example of immediate incremental maintenance for an AD.

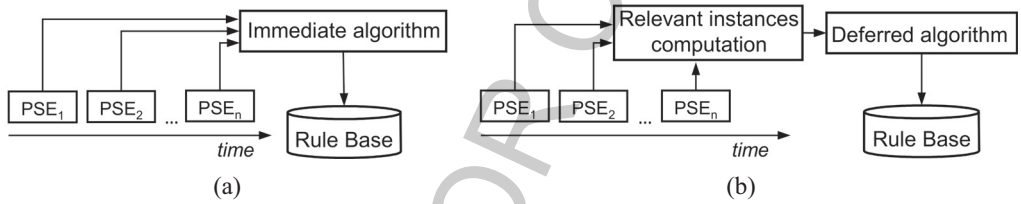

Fig. 4. Immediate incremental maintenance (a), and deferred incremental maintenance approach (b).

4.4. Deferred incremental maintenance method

A deferred approach efficiently maintains a rule base up-to-date but not for each data operation like immediate approach. This method computes modified instances in a data transition and updates the rule base for these relevant instances. Principal differences of immediate and deferred maintenance approaches are illustrated in Fig. 4.

Typically, incremental mining algorithms consider different types of operations but do not consider the interactions between such operations [24,26]. We propose an algorithm specifically for these interactions; in the best scenario it reduces the number of operations significantly and in the worst case it only maintains the same original number.

The deferred proposal is divided in two subproblems as Fig. 4 highlights. The first subproblem consists of computing the relevant instances affected in a database transition. In this step, a relevant operation set at real-time is built, after each primitive structural event takes place. A different approach would be to scan the original operation set to reduce their number. The second one is related to incrementally update the rule base with those relevant instances. The integrity constraint checking handle these subproblems in its field [8,9].

The computation of relevant instances in a transition must consider the relationships among primitive structural events. These interactions are controlled by net effect policy [8,10,34]. For example, if a record is inserted and deleted in the same database transition, then these events do not cause any variation on the

final database state and its measures of rules. Thus, the following policies are considered for structural event interactions. If a record is:

- Inserted and later deleted, then it does not count.
- Inserted and later updated, then it counts as inserted.
- Several times updated, then it counts as one update.
- Updated and later deleted, then it counts as deleted.
- Deleted and later inserted, then it counts as updated.

Generally, non-modeled update events such as deletion followed by insertion are registered by an auxiliary relation [8,10]. In this approach a different consideration is assessed, each database relation related to any rule has only two auxiliary relations. These auxiliary relations register the insert, update, and delete events. Their relation schemas are copies with different names of the base relation scheme and they must store the newest and oldest record values. Insert and delete auxiliary relations store $t^+ \cup t_1^{-+}$ and $t^- \cup t_0^{-+}$ records respectively, according with net effect considerations. These structural event interactions are applied over relations at real-time by the active Algorithm 2.

Algorithm 2 Compute relevant instances that may modify DRs

Input: A composite structural event *CSE*, I and D the auxiliary relations of base relation.
Output: Auxiliary relations I and D updated for a *CSE*.
Method:

for all $PSE \in CSE$ **do**
  if ($PSE = \Delta t^{-+}$) **then** ▷ update event
    if ($\{t_0^{-+} \cap I\} = \emptyset$) **then**
      insert into I values t_1^{-+};
      insert into D values t_0^{-+};
    else
      update $u \in I$ set $u = t_1^{-+}$ where $u = t_0^{-+}$;
    end if
  else if ($PSE = \Delta t^{+}$) **then** ▷ insert event
    insert into I values t^+;
  else ▷ delete event ($PSE = \Delta t^{-}$)
    if ($\{t^- \cap I\} = \emptyset$) **then**
      insert into D values t^-;
    else
      delete $u \in I$ where $u = t^-$;
    end if
  end if
end for

This active process adds a minimum activity over regular data operations, just the necessary ones to store relevant instances and to apply net effect policy. A small example is demonstrated by the Fig. 5 where for each transition of a relation, the transition of their auxiliary relation is visible. In this example, the original three structural events are reduced to two relevant instances.

The rule base is updated only with these instances, by incrementing previous rule's information. These rule base updates could be made automatically with a decision-maker's rule base access or scheduled.

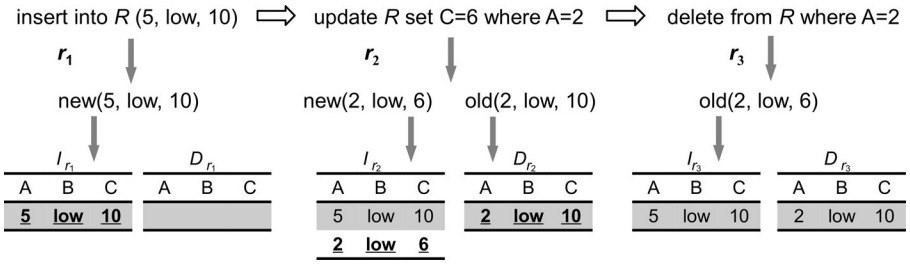

Fig. 5. Example of compute relevant instances for primitive structural events.

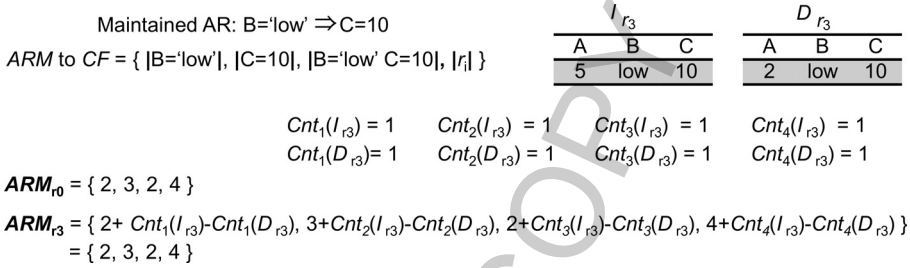

Fig. 6. Example of *ARM* deferred actualization for relevant instances.

In addition, the rule base should be updated if new data rules need to be added or some of the existing ones must be deleted. In this step, the Algorithm 3 is presented in order to update *DRM*.

Similar to Algorithm 1, the incrementing of measure-parts of Algorithm 3 is quite different depending on the DR type. After the measure-parts are updated, the algorithm truncates all data from the auxiliary relations, a special feature in constant measure access. To illustrate our deferred proposal for ARs, let us look at the following example in Fig. 6.

The incrementing measure-parts of Algorithm 3 for ADs keeps the *ADM* auxiliary relations up-to-date. Notice that in this strategy it is not necessary to delete records in auxiliary relations. Instances that do not already exist in base relation have zero value in *catt* attribute. This deferred proposal, like the immediate one, updates rule base without access to the base relation. Also, it entails the benefits of having only two auxiliary relations instead of more. Heuristically, the time for querying I and D auxiliary relations is still much lower than the one of querying the whole data like naïve approach. In rare occasions, this is not accomplished. For example, if the entire base data is deleted, then querying it is very fast and querying D auxiliary relation is highly inefficient.

Algorithm 3 Deferred incremental maintenance for relevant instances

Input: I, D auxiliary relations of Algorithm 2 output, and measure-parts *DRM*.
Output: Updated measure-parts *DRM*.
Method:

for all $MP_k \in DRM$ **do**
  update Mp_k, increment with I;
  update Mp_k, decrement with D;
end for
empty I;
empty D;

Table 1
Computational complexity of proposed algorithms

	ARs		ADs	
	Active	Measure update	Active	Measure update
Immediate approach	$O(n)$	$O(1)$	$O(n * \log m)$	$O(\log m)$
Deferred approach	$O(n * \log n)$	$O(\log n)$	$O(n * \log n)$	$O(\log m * \log n)$

4.5. Complexity analysis

Since our proposals consider an active component, algorithms' analysis are split to consider the complexity imposed on the regular data operations and complexity of measure update. It is also necessary to separate the proposed algorithms according to DRs type. The final complexities are related in Table 1.

In this analysis n is considered the number of primitive structural events and m represents the greatest cardinality of measures auxiliary relations in ADs structures. Heuristically, m should not be too large in huge amounts of data since it is part of an approximate dependency. However, m may be gradually increased over time with new data operations. For all auxiliary relations, a common database B-tree structure is considered in which data operations can be evaluated in $O(\log n)$. Nevertheless, it is possible to consider a hash table implementation having complexity $O(n)$ as the worst case but an expected performance of $O(1)$.

The active component of the proposed algorithm is very important because it implies the overhead imposed on system regular operations. The overall algorithm complexity is obtained by adding both results, which coincide with the active complexity because it has the highest activity.

The principal advantage of immediate proposal is speed increasing in update measures. The instant actualization of rule base after each primitive structural event avoid any manipulation at measure access, except their calculation with measure-parts. However, ADs active step of the immediate proposal presents a complexity based on *m* that reduces its efficiency.

We do not compare immediate and deferred approaches, they are different solutions and the best method must be selected according to system requirements. For example, if the system has a huge number of structural events and needs a few accesses to the rule base, then deferred proposal is preferable. Moreover, if the system has a good performance for structural events and needs a lot of accesses to the rule base, then immediate approach must be chosen.

5. Related work and comparison with our approach

Numerous algorithms for mining ARs have been proposed at this time based on Apriori approach [1]. These Apriori-like algorithms generate candidate itemsets level-by-level, which might cause multiple scans of the database and high computational costs. In order to avoid re-scanning the whole data and breaking Apriori bottlenecks, many algorithms have been proposed by using tree-structures [22,38]. The frequent-pattern tree (FP-tree) proposed by [17] is a milestone in the development of ARs based on this method. The FP-tree is used to compress a database into a tree structure which stores only large items. After the FP-tree is constructed, a mining algorithm called FP-growth derives all large itemsets in a second step [17].

In real-world applications, data repository is not static. Generally, data will increase with time. Traditional batch mining algorithms solve this problem by re-scanning the whole data when new transactions are inserted, deleted or modified. This is clearly inefficient because all previous mined information is

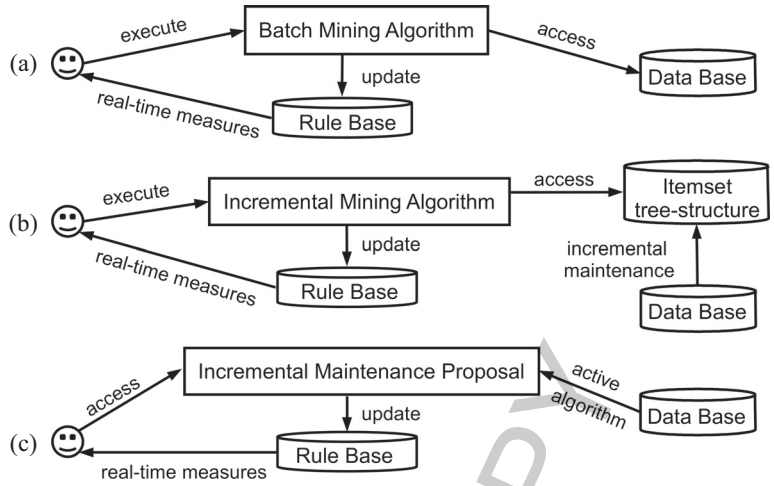

Fig. 7. Batch mining method (a), incremental mining method (b), and incremental maintenance proposal (c) for real-time measures.

wasted. The incremental mining defines this issue as an update problem and reduces it to find the new set of large itemsets incrementally. Algorithm FUP (Fast UPdate) [11], proposed by Cheung and his coworkers, is the first algorithm for incremental mining of association rules when new data transactions are added to a database.

Although the FUP approach improves a mining performance in dynamic environments, the original database is still required to be re-scanned. Extended tree structures are being designed for FP-tree to efficiently handle this problem [18,22,24,26]. These proposals improve pioneer tree-structure in different ways but ma intain the execution of FP-growth algorithm in a second step. Some related researches are still in progress.

Unlike incremental mining methods, we handle the update problem by maintaining the measures of previous discovered rules. That does not lead to maintain itemsets information, instead, existing rules measures are directly updated in an incremental way. After the rules discovering process, we keep only the extracted rules and no rule is removed or added, thus allowing the still expensive incremental mining algorithm. In Fig. 7 three scenarios illustrate when a system decision-makers needs the real-time measures of previously discovered rules. That includes the batch mining method, the incremental mining method, and our proposal.

As can be appreciated in Fig. 7, discovering new knowledge is out of our scope, maintaining efficiently up-to-date measures of rules is our main objective.

6. Experimental results

Several experiments have being performed on real data and real structural events obtained from SWAD, a web system for education support at the University of Granada [7]. The studied dataset consists of information about students' courses containing nine attributes over 5 K instances. The most relevant attributes are the student's sex (*sex*), the average of questions answered in all student exams (*avg_aqst*), the sum of visits to signature web files (*sum_vfiles*), the count of clicks in the platform (*cnt_clicks*), the count of downloads files (*cnt_dfiles*), and the average of all exam scores (*avg_score*).

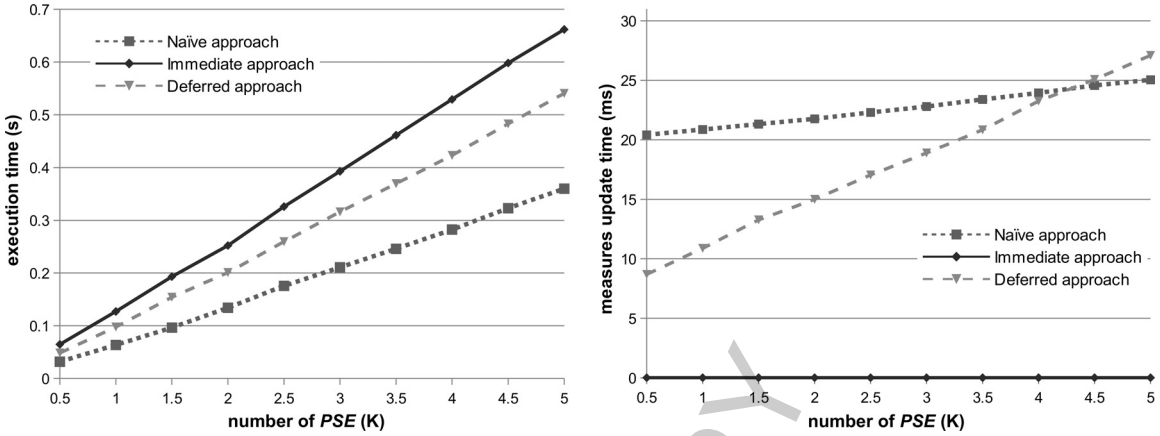

Fig. 8. Proposals comparison for ARs maintenance on execution times (left) and measures update times (right) in PL/pgSQL.

Results illustrate the performance of proposed algorithms in order to maintain seven ARs and seven associated ADs. These ARs were discovered using the KEEL data mining software tool [2]. Maintenance is implemented using the certainty factor metric Eq. (1) from two open source database management systems: PostgreSQL Server version 9.2.2 and MySQL Server version 5.6.13. Both management systems have similar results. For our proposed algorithms, we included PostgreSQL results for ARs and MySQL results for ADs. The experiments were carried out on a dedicated GNU/Linux server with eight processors i7-2600 at 3.4 GHz and 15 GB of main memory.

The experiments have been designed to observe two approaches' behavior: active process execution time and measures update time. The first one provides the execution time when processing different numbers of primitive structural events on studied dataset. Here, naïve approach represents the regular behavior of the database where no active algorithm is executed. The second one exposes the consumed time for update measures of rules, after the same primitive structural events take place. The primitive structural events contain database insert, update, and delete operations extracted from real database transitions. In Figs 8 and 9, PostgreSQL results for ARs maintenance and MySQL results for ADs maintenance are respectively presented.

It can be noticed from Fig. 8 (left) the relatively small extra execution time added to the active parts of our proposals for ARs maintenance. Otherwise, measures updating times for the naïve and immediate approaches (right) do not depend on the structural events quantity and remain almost constant. Just the deferred proposal increases measure update time as expected. This deferred proposal behavior is not worrying since it corresponds with a single measure update time. For constantly measure access, auxiliary relations are always truncated.

Results for ADs in Fig. 9 show similar behavior to those for ARs. Here, the naïve and deferred approaches have small differences in execution time (left). However, the execution time of the immediate approach increased very quickly due to auxiliary structures needed in ADs proposals. This behavior probably compromises database performance in stressed scenarios.

All these experiments are a close look to proposals behavior in a small dataset. In order to observe the scalability of those algorithms, we obtained the total time of update rule base for different database sizes synthetically generated. The complete execution time is calculated as the sum of executing 5 K primitive structural events and updating measures of rules. Results for maintaining the ADs in MySQL are illustrated in Fig. 10.

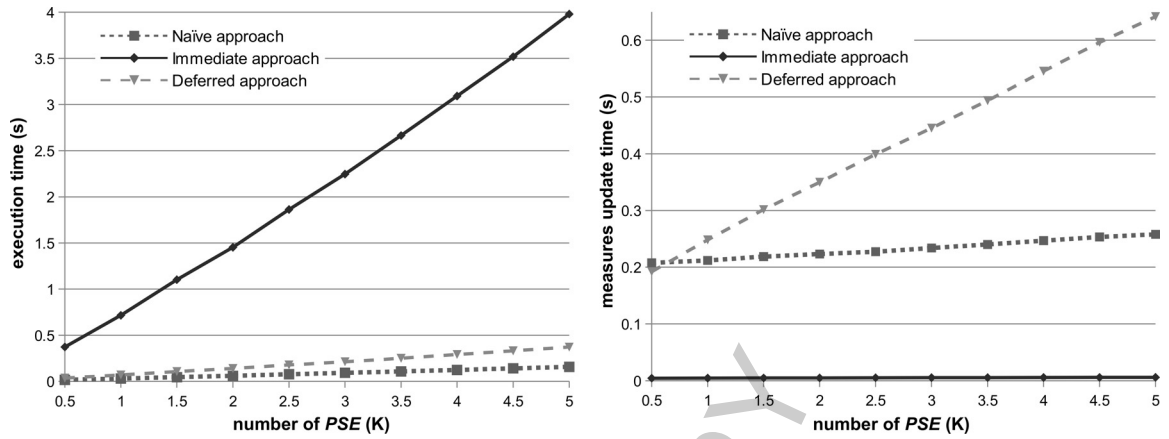

Fig. 9. Proposals comparison for ADs maintenance on execution times (left) and measures update times (right) in MySQL.

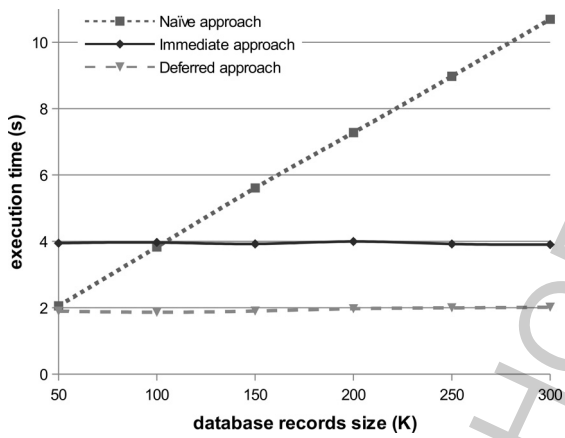

Fig. 10. Proposals comparison of total execution time for ADs maintenance in different database records size.

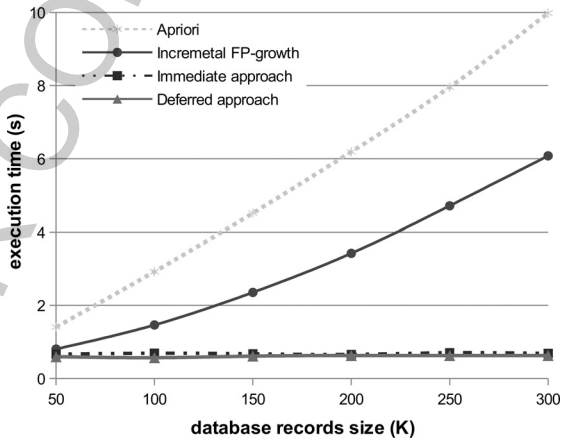

Fig. 11. Related and proposed algorithm comparison of total execution time for ARs maintenance in different database records size.

Notice that our proposals maintain almost the same total execution time, an important consequence of a self-maintainable characteristic. This keeps the lowest execution time in very large or big databases. Clearly, the naïve approach presents a low performance when database grows in size. The experiments for this result only consider a single measure update in the rule base, multiple updates increase the difference between the naïve and proposed methods.

The performance of proposed algorithm was also compared to traditional and incremental algorithms for ARs maintenance. In Fig. 11 a total execution time for proposed algorithms, batch mining, and incremental mining methods, for different database sizes is presented. Our proposal reflects the time of executing 5 K data operations plus measure update of rules. Batch and incremental mining methods reflect the mining execution time for the same measure update goal. The Apriori algorithm stands for batch mining methods. For incremental mining methods, in order to include a wide variety of incremental algorithms [18,22,24,26] we only obtain the FP-growth execution time and depreciate the FP-tree build time, assuming that it was incrementally maintained. This approach is referred as incremental FP-growth. For Apriori and FP-growth algorithms the minimum support threshold was set at 10% and minimum confi-

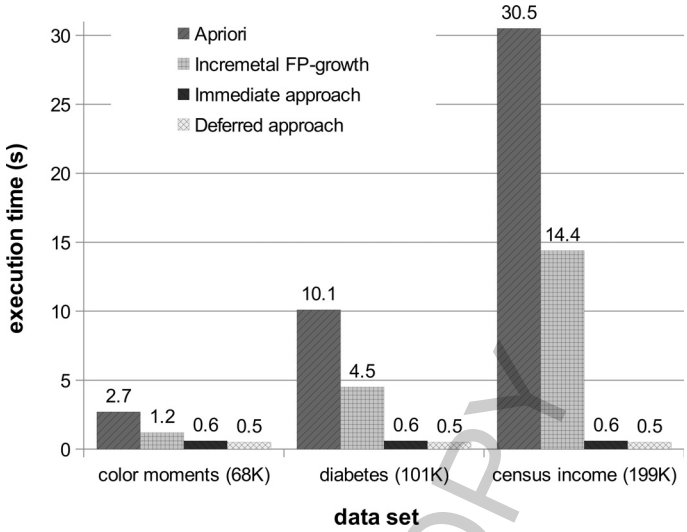

Fig. 12. Related and proposed algorithm comparison of total execution time for ARs maintenance in different datasets.

dence threshold at 80%. Both mining algorithm experiments were created using the KEEL data mining software tool.

Finally, in Fig 12 we also evaluate proposed algorithms in different datasets obtained from the UCI Machine Learning Repository [25]. These datasets are the Color Moments part of Corel Image Features (color moments), the Diabetes 130-US hospitals for years 1999–2008 (diabetes) described in [40], and Census-Income (KDD) data part (census income). Details about these datasets can be found on the UCI Machine Learning website. For diabetes and census datasets nine attributes were selected. Seven ARs were extracted using KEEL from each dataset in order to be incrementally maintained by proposed algorithms. The mining algorithms conditions for this experiment remain constant.

It is obvious to conclude from Figs 11 and 12 that the proposed algorithms are faster than the other two algorithms. In fact, the differences are increased in a way that database size is bigger because they remain almost constant. The above time results are accepted especially in time-critical systems where decision-makers need to make decisions using the existing rule's information as soon as possible.

7. Conclusion and future work

In real-world applications, records are commonly inserted, updated or deleted outdating the previous extracted knowledge as inexact and invalid. In some scenarios, it is necessary to re-run traditional mining or incremental mining algorithms only for updating previous discovered rules. It is possible, from another perspective, to maintain the known rules incrementally by computing data changes efficiently.

In this article, two algorithms have been proposed specifically for maintaining the previous discovered rules. These algorithms operate over a generic form of measures, allowing the maintenance of a wide range of rule metrics in an efficient way. We also propose to consider the interactions between data operations at real-time in order to create a reduced relevant instance set. Experimental results with real data and operations show that our proposals achieve a better performance against the batch mining, incremental mining, and a naïve approach. These improvements are increased when database size is bigger, making it suitable in very large or big database systems.

There are still some interesting research issues related to the contributions of this paper that can be applied to other areas. Specifically, to incrementally maintain other types of data rules like fuzzy association rules, to consider interactions between data operations in existing incremental mining algorithms, and to explore the memory usage of the proposed algorithms in different implementations.

Acknowledgements

The authors would like to thank the members of the Iberoamerican Association of Postgraduate Universities (AUIP) for their international academic mobility program. We also thank Prof. Antonio Cañas Vargas for providing some of the data used in the experiments. We are grateful to all people who have contributed with their suggestions for improving the final version of the manuscript.

References

[1] R. Agrawal, T. Imieliński and A. Swami, Mining association rules between sets of items in large databases, *Sigmod Rec* **22**(2) (1993), 207–216.
[2] J. Alcalá-Fdez, L. Sánchez, S. García, M. Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. Rivas, J. Fernández and F. Herrera, Keel: A software tool to assess evolutionary algorithms for data mining problems, *Soft Comput* **13**(3) (2009), 307–318.
[3] V. Attar and V. Inamdar, Materialized views in data mining, *Lect Notes Comput SC* **7**(12) (2007), 90–94.
[4] F. Berzal, I. Blanco, D. Sánchez and M.-A. Vila, Measuring the accuracy and interest of association rules: A new framework, *Intell Data Anal* **6**(3) (2002), 221–235.
[5] F. Berzal, J.-C. Cubero, N. Marín and J.-M. Serrano, Tbar: An efficient method for association rule mining in relational databases, *Data Knowl Eng* **37**(1) (2001), 47–64.
[6] M. Boettcher, G. Ruß, D. Nauck and R. Kruse, From change mining to relevance feedbacka unified view on assessing rule interestingness, *Post-Mining of Association Rules: Techniques for Effective Knowledge Extraction* (2009), 12–37.
[7] A. Cañas, D. Calandria, E. Ortigosa, E. Ros and A. Díaz, Swad: Web system for education support, in: *Computers and Education*, B. Fernández-Manjón, J. Sánchez-Pérez, J. Gómez-Pulido, M. Vega-Rodríguez and J. Bravo-Rodríguez, eds, Springer Netherlands, 2007, pp. 133–142.
[8] J. Cabot and E. Teniente, Computing the relevant instances that may violate an ocl constraint, in: *Advanced Information Systems Engineering*, O. Pastor and J.A. Falcão e Cunha, eds, volume 3520 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2005, pp. 48–62.
[9] J. Cabot and E. Teniente, Incremental evaluation of ocl constraints, in: *Advanced Information Systems Engineering*, E. Dubois and K. Pohl, eds, volume 4001 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2006, pp. 81–95.
[10] S. Ceri and J. Widom, Deriving production rules for incremental view maintenance, in: *Proceedings of the 17th International Conference on Very Large Data Bases*, VLDB '91, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991, pp. 577–589.
[11] D. Cheung, J. Han, V. Ng and C.Y. Wong, Maintenance of discovered association rules in large databases: An incremental updating technique, in: *Data Engineering, 1996, Proceedings of the Twelfth International Conference on* (1996), 106–114.
[12] L.S. Colby, T. Griffin, L. Libkin, I.S. Mumick and H. Trickey, Algorithms for deferred view maintenance, *Sigmod Rec* **25**(2) (1996), 469–480.
[13] D. Dudek, Rmain: Association rules maintenance without reruns through data, *Inform Sciences* **179**(24) (2009), 4123–4139.
[14] C. Giannella and E. Robertson, On approximation measures for functional dependencies, *Inform Syst* **29**(6) (2004), 483–507, {ADBIS} 2002: Advances in Databases and Information Systems.
[15] S. Greco, R. Słowiński and I. Szczęch, Properties of rule interestingness measures and alternative approaches to normalization of measures, *Inform Sciences* **216**(0) (2012), 1–16.
[16] A. Gupta, I.S. Mumick et al., Maintenance of materialized views: Problems, techniques, and applications, *IEEE Data Eng Bull* **18**(2) (1995), 3–18.
[17] J. Han, J. Pei and Y. Yin, Mining frequent patterns without candidate generation, *Sigmod Rec* **29**(2) (2000), 1–12.

[18] T.-P. Hong, C.-W. Lin and Y.-L. Wu, Incrementally fast updated frequent pattern trees, *Expert Syst Appl* **34**(4) (2008), 2424–2435.
[19] Y. Huhtala, J. Kärkkäinen, P. Porkka and H. Toivonen, Tane: An efficient algorithm for discovering functional and approximate dependencies, *The Computer Journal* **42**(2) (1999), 100–111.
[20] H. Jain and A. Gosain, A comprehensive study of view maintenance approaches in data warehousing evolution, *ACM Sigsoft* **37**(5) (2012), 1–8.
[21] R.S. King and J.J. Legendre, Discovery of functional and approximate functional dependencies in relational databases, *Journal of Applied Mathematics and Decision Sciences* **7**(1) (2003), 49–59.
[22] Y.-S. Lee and S.-J. Yen, Incrementally mining frequent patterns from large database, in: *Information Granularity, Big Data, and Computational Intelligence*, W. Pedrycz and S.-M. Chen, eds, volume 8 of *Studies in Big Data*, Springer International Publishing, 2015, pp. 121–140.
[23] P. Lenca, P. Meyer, B. Vaillant and S. Lallich, On selecting interestingness measures for association rules: User oriented description and multiple criteria decision aid, *Eur J Oper Res* **184**(2) (2008), 610–626.
[24] X. Li, Z.-H. Deng and S. Tang, A fast algorithm for maintenance of association rules in incremental databases, in: *Advanced Data Mining and Applications*, X. Li, O. Zaïane and Z.-H. Li, eds, volume 4093 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2006, pp. 56–63.
[25] M. Lichman, UCI machine learning repository, 2013.
[26] C.-W. Lin and T.-P. Hong, Maintenance of prelarge trees for data mining with modified records, *Inform Sciences* **278**(0) (2014), 88–103.
[27] C.-Y. Liu, C.-Y. Tseng and M.-S. Chen, Incremental mining of significant urls in real-time and large-scale social streams, in: *Advances in Knowledge Discovery and Data Mining*, J. Pei, V. Tseng, L. Cao, H. Motoda and G. Xu, eds, volume 7819 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2013, pp. 473–484.
[28] H. Liu, Y. Lin and J. Han, Methods for mining frequent items in data streams: an overview, *Knowl Inf Syst* **26**(1) (2011), 1–30.
[29] J. Liu, J. Li, C. Liu and Y. Chen, Discover dependencies from data-a review, *IEEE T Knowl Data EN* **24**(2) (2012), 251–264.
[30] R. Medina and L. Nourine, A unified hierarchy for functional dependencies, conditional functional dependencies and association rules, in: *Formal Concept Analysis*, S. Ferré and S. Rudolph, eds, volume 5548 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2009, pp. 98–113.
[31] A. Mohapatra and M. Genesereth, Incremental maintenance of aggregate views, in: *Foundations of Information and Knowledge Systems*, C. Beierle and C. Meghini, eds, volume 8367 of *Lecture Notes in Computer Science*, Springer International Publishing, 2014, pp. 399–414.
[32] H. Nakayama, A. Hoshino, C. Ito and K. Kanno, Formalization and discovery of approximate conditional functional dependencies, in: *Database and Expert Systems Applications*, H. Decker, L. Lhotská, S. Link, J. Basl and A. Tjoa, eds, volume 8055 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2013, pp. 118–128.
[33] A. Olivé and J. Cabot, A research agenda for conceptual schema-centric development, in: *Conceptual Modelling in Information Systems Engineering*, J. Krogstie, A. Opdahl and S. Brinkkemper, eds, Springer Berlin Heidelberg, 2007, pp. 319–334.
[34] N.W. Paton and O. Díaz, Active database systems, *ACM Comput Surv* **31**(1) (1999), 63–103.
[35] S.J. Qin, Process data analytics in the era of big data, *Aiche J* **60**(9) (2014), 3092–3100.
[36] D. Quass, Maintenance expressions for views with aggregation, in: *Views'96*, 1996.
[37] D. Sánchez, J.M. Serrano, I. Blanco, M.J. Martín-Bautista and M.-A. Vila, Using association rules to mine for strong approximate dependencies, *Data Min Knowl Disc* **16**(3) (2008), 313–348.
[38] S. Shah, N. Chauhan and S. Bhanderi, Incremental mining of association rules: A survey, *International Journal of Computer Science and Information Technologies* **3**(3) (2012), 4071–4074.
[39] R. Srikant and R. Agrawal, Mining quantitative association rules in large relational tables, in: *ACM Sigmod Record*, ACM **25** (1996), 1–12.
[40] B. Strack, J.P. DeShazo, C. Gennings, J.L. Olmo, S. Ventura, K.J. Cios and J.N. Clore, Impact of hba1c measurement on hospital readmission rates: Analysis of 70,000 clinical database patient records, *BioMed Research International* **2014**(781670) (2014), 11.
[41] J. Tan, Y. Bu and H. Zhao, Incremental maintenance of association rules over data streams, in: *Networking and Digital Society (ICNDS), 2010 2nd International Conference on* **2** (2010), 444–447.
[42] T. Urpi and A. Olivé, Semantic change computation optimization in active databases, in: *Research Issues in Data Engineering, 1994, Active Database Systems, Proceedings Fourth International Workshop on*, IEEE (1994), 19–27.
[43] X. Wu, X. Zhu, G.-Q. Wu and W. Ding, Data mining with big data, *IEEE T Knowl Data EN* **26**(1) (2014), 97–107.
[44] Z.K. Zia, S.K. Tipu and M.I. Khan, Research on association rule mining, *Advances in Computational Mathematics and its Applications* **2**(1) (2012), 226–236.