ISBN 978-981-11-0008-6
Proceedings of 2016 6th International Workshop on Computer Science and Engineering (WCSE 2016)
Tokyo, Japan, 17-19 June, 2016, pp. 79-87
doi: 10.18178/wcse.2016.06.015

Using Periodic Patterns for Dynamic Estimation of Future Database Workloads

Janusz R. Getta [1] and Marcin Zimniak [2]

[1] School of Computing and Information Technology, University of Wollongong, Wollongong, Australia
[2] Faculty of Computer Science, TU Chemnitz, Chemnitz, Germany

Abstract. Analysis of historical database workloads can be used to improve the future performance of database systems. This work investigates a problem how to use information about periodic processing of database applications recorded in the past for static and dynamic estimation of the future database workloads. The new concepts of anchor events and event expressions are used to create the mappings of past periodic patterns in data processing into the future database workloads. We show that event expressions allow for both static and dynamic estimation of future database workloads. The paper presents the algorithms for static a dynamic estimation of future database workloads from information about time distributions of periodic processing of database applications and information about the events that trigger processing of the applications.

Keywords: mining periodic patterns, database workloads, automated performance tuning process mining.

1. Introduction

One of the main objectives of automated database performance tuning is detection and elimination of workload peaks that significantly slow down a database system in the critical periods of time. Usually, the database administrators try to utilize their experience from the past management of database system to anticipate the workload peaks and to prepare a system for the processing of more resource demanding database applications. An automated system of physical database design and performance tuning attempts to emulate the database administrators through detection of the periods of time when the database workloads are either exceptionally low or exceptionally high. Then, the low workload periods of time can be utilized to automatically restructure data, to reduce access paths, to cluster the groups of related data, and to apply other performance tuning techniques, that prepare a database system for more efficient processing in the future [1].

The workload peaks and the longer periods of very high or very low workload are inevitable due to the interferences of periodically processed database tasks and due to the randomly occurring ad-hoc data processing tasks reflecting the real world random events. To implement an automated database performance tuning system two important research problems must be solved. First, it should be possible to automatically and with a high level of accuracy predict the future levels of workload in a database system. Second, it should be possible to find the best ways to "prepare" a system at the periods of low workload time for the periods of high workload time through re-scheduling of the less important activities, through creating better access paths to data resources, and through allocation of more computational resources to the complex processes, etc. This work attempts to solve the first problem, i.e. it shows how to use information about periodic processing of database applications to estimate the future workload levels in a database system.

Is it possible to predict the future database workloads? In a general case it is hard to give a confident and positive answer to such question. However, in some cases, depending on the existing circumstances it is

+ Corresponding author. Tel.: +61 242214339.
E-mail address: jrg@uow.edu.au.

possible to give a positive answer with a pretty high level of confidence. For instance, one of the ways the future database workload levels can be anticipated is based on identification of so called *periodic patterns* in historical information about processing of database applications. The majority of data processing is a reflection of the real world processes. Many of these processes are repeated every more or less the same period of time and because of that many of database applications have quite predictable occurrences and behaviour. For example, football games occur almost every week in a given period of time, payroll information must be generated every months or fortnight, supermarkets have higher workloads every week on a long trading day, etc.

To prepare a database system for the workload peaks it is possible to get more information on the periodicity of database applications and implementation details of database operations from the logs, traces, audit trails, etc., [2]. In particular, we need information about the sequences and frequencies of elementary database operations executed during the computations of queries and data manipulation statements. For instance, it frequently happens that processing of several different events requires the periodic computations of the same elementary operations and because of that it creates a clear processing pattern.

Unfortunately, identification of periodic patterns in the database workloads alone is not sufficient to predict the future levels of the workloads. Information about the behaviour and properties of database applications discovered from data recorded in the past is not enough to more precisely predict the future workloads. This is because the distributions of the real world events that trigger database applications may change in time. For example, a new session starts at a university on a different day each year. Periodically organized sport events do not always happen on the same date in the next year. Therefore, to be able to utilize information about periodical processing of database applications in the past we need to find a mapping of the past data processing into *periods of time* between *real world events* and later on to find another mapping of such periods into a time line in the future. Then, composition of the mappings may provide a static estimation of future database workloads. This idea is applied in he present work to answers a question how to predict the future workload levels from information about periodical changes of workload levels in the past.

A solution proposed in this work is based on an observation that certain real world events are repeated every certain and not always the same period of time. Such events can be repeated everyday, every month every year, or every few years. Of course, the real world events are repeated with not necessarily the same period of time, sometimes they are not repeated at all and a relative order of the events may change from time to time. A basic idea is such that if it is possible to associate the periodic patterns in workload with the real world events or with the periods of time between the events then no matter what periods and orders of the events are it is possible to find a mapping of the periodic patterns into the future plans for the same events.

An entire procedure consists of the following steps. First, we collect information about the past workload levels from database logs, traces, audit trails, etc. Next, such information is analyzed and the periodic patterns of database operations are discovered. The starting points of periodic patterns are associated with the real world events or with the periods of time between such events creating a mapping of periodic patterns into a set of events or periods of real world events. Next, we consider the plans for the same events in the future such that we find a mapping of the real world events or periods of time between the events into the pints in time or periods of time in the future. Finally, we find a composition of the mappings of histories of periodic patterns in the past into real world events and a mapping of real world events into the periods of time in the future. The composition provides us with a static estimation of the future database workloads.

Of course both mappings are not the ultimately precise. Some periodic patterns are bound with several possible periods of time determined by the real world events with different evaluation of chances to happen in each one of them. Then, it may happen that some of these estimations are not correct and periodical processing of certain operations is not initiated on the predicted time slots. Dynamic or adaptive estimation of future database workloads is proposed to solve the problem. We assume, that several predictions for periodical data processing are proposed and when some of them do not happen then the future plans are automatically adjusted. Dynamic estimation of the future workloads requires the algorithms very similar to

the algorithms used for *conformance checking*. *Conformance checking* compares an already created process model with the logs of its executions recorded at the testing stages to make sure that unexpected situations do not happen when the process runs at a production stage.

The paper is organized in the following way. The next section reviews the earlier research works related to workload estimation, conformance checking in process mining, and discovering periodic patterns. A section 3 defines the concepts of time units and *CRP* periodic patterns. In section 4 introduces the new concepts of *anchor* events, *event expression* and we show how the *event expressions* can be evaluated over the histories and plans. A section 5 presents an algorithm for a static estimation of future workloads. A dynamic algorithm for workload estimation is discussed in a section 6. Finally, a section 7 concludes the paper.

2. Previous Work

This work is based on the research results in the areas of *workload estimation*, *conformance checking in process mining*, and *mining of periodic patterns*. The works on *workload estimation* address a problem how the past database workload analysis can be used to predict the workload levels in the future. Analysis of database workload goes back to [3] where the traces from processing of database applications have been used to investigate database performance on SMT processors.

A workload analysis performed in [4] demonstrated the repetitive nature of enterprise workloads. The results proved that it is possible to predict future workload with a high level accuracy.

A problem of predicting the execution time of a query on a loaded data warehouse with a dynamically changing workload has been investigated in [5] This approach combined a query plan with en earlier observed workload to predict the execution time of the query.

In [6] statistical models have been used to find the resource requirements for cloud computing applications in order to provide maximal performance for external users.

Performance predictability for concurrent query execution is needed for cloud and multi-tenant database that face diverse and heterogeneous workloads. A solution proposed in [7] performs analysis each query behaviour in isolation, and then investigates the pairwise query interactions to predict the future resource utilization at different concurrency levels.

A similar problem of estimation of workload completion times in the presence of significant query interactions has been investigated in [8]. The solution proposed in this work applies an experimental approach by testing many possible query co-occurrences.

A more general problem of dynamic concurrent workloads where the workload queries may dynamically change has been considered in [9]. An approach proposed there is based on analytic modelling of I/O and CPU requirements for each query. The experiments show that analytic-model based approach leads to better prediction accuracy than machine-learning based approaches.

The present work is based on a simple technique of workload estimation originally developed in [10]. The works on *conformance checking in process mining* has been triggered by the applications of process mining to analysis of transaction logs and audit trails. *Conformance checking* verifies consistency of a process discovered from the past logs with the present behaviour of a software system. The book [11] is at the moment the most comprehensive source of information on the present state of process mining. Since its introduction the problem of process mining has been formally defined and the first solutions have been proposed in a number of practical applications. Application of process mining to conformance checking has been investigated in [12] and [13].

The model of periodicity considered in this paper is an extension of the model introduced in [2] for the analysis of database audit trails. In this work we consider a class periodic patterns defined in [14] with periodic pattern discovery algorithms described in [15].

3. Basic Concepts

A period of time in the past $<t_{start}, t_{end}>$ over which the periodic patterns have been discovered and a

future period of time in the future $<t'_{start}, t'_{end}>$ over which we try to predict the workloads are both divided into a contiguous sequence of *n* disjoint and fixed size *time units* $<t_i,\tau_i>$ and $<t'_i, ,\tau_i>$, $i = 1, \ldots, n$ where each t_i and t'_i are the timestamps when an elementary time unit starts and $,\tau_i$ is its length. Both sequences of time units satisfy the following properties: $t_{start}=t_1$ and $t_i + \tau_i \leq t_{i+1}$ and $t_{n-1} + \tau_n = t_{end}$ and $t'_{start}=t'_1$ *and* $\$t'_i + \tau_i \leq t_{i+1}$ *and* $t_{n-1} + \tau_n = t_{end}$. Both periods of time are divided into exactly the same number of time units with the same length of the respective time units in the periods, however the individual time units in a sequence may have different length.

As a simple example consider a period of time that starts on $t_{01:01:2007:0:00am}$ and ends on $t_{31:01:2007:12:00pm}$. Then, a sequence of disjoint time units called as *morning tea time* consists of the time units $<t_{01:01:2007:10:30am}, 30>$, $<t_{02:01:2007:10:30am}, 30>$, …, $<t_{31:01:2007:10:30am}, 20>$ where a time unit on the last day of month is a bit shorter than the previous ones.

A sequence of time units $<t_i, \tau_i>$, $i = 1, \ldots, n$ is denoted by *H* (history) where *H[j]* is the *j*-th time unit in the sequence. In the same way a sequence $<t'_i, \tau_i>$ for $i=1, \ldots, n$ is denoted by *F* (future plan) where *F[j]* is the *j*-th time unit in the sequence.

A class of *CRP* periodic patterns is defined as a triple *<C, R, P>* where

(1) a *carrier C* defines a structure of periodically repeated events, computations, queries, etc.,

(2) a *range R* determines a time scope of periodic repetitions of a *carrier* measured in time units, for example from time unit *H[i] to* a time unit *H[j]*,

(3) a *periodicity P* determines when the next periodic repetition of a *carrier* may happen, for example after *p* time units from the latest occurrence of a *carrier* with possible delay by *k* time units.

In [10] a carrier *C* is a nonempty, finite sequence of multisets of syntax trees of relational algebra expressions implementing SQL statements. A range *R* is a pair of the ordinal numbers of time unit where the repetitions of a carrier start and end. A periodicity *P* is a total number of time units between two adjacent repetitions plus one. In [15] *C* is defined as a e^k of an event *e*, *R* is defined as a pair of numbers *f:t* that determine the first and the last repetition of a carrier and *P* is defined as a pair of numbers *n:x* that determine the minimal and maximum distance between any two adjacent repetitions of a carrier.

In this work we consider a subclass of *CRP* periodic patterns [14] defined as a triple *<C, f:t, p>* where:

(1) a *carrier C* is a nonempty sequence of at least one nonempty multisets of operations where a *multiset M* is a pair *<S, f >* where *S* is a finite set and $f: S \rightarrow N^+$ is a function such that $\Sigma_{s \in S} f(s) < \infty$ which is called as a *cardinality* and it is denoted by $|M|$.

The function *f* determines multiplicity of each element in *S* [16].

(2) a *range f:t* is a pair of natural numbers that determine a location of the *first* cycle and the *total* number of cycles in the pattern,

(3) a *periodicity p* is a natural number which determines a distance between every two adjacent cycles,

(4) and the values of *f:t* and *p* must satisfy the conditions $f \geq 1$ and $t \geq 1$ and $p \geq 0$ and $f+(t-1)*p \leq |U|$ and if $t=1$ then $p=0$.

4. Anchor Events

Information obtained from the algorithms discovering periodic patterns determines the "locations", i.e. start time unit, total number of cycles, and period, of the patterns in the past time units. Unfortunately such information does not say anything about the distribution of periodic patterns in the future. To find the "locations" of periodic patterns in the future we have to express the past histories in a context of events that have happened in the past and the same events which will happen in the future. Such events are called as *anchor events* of periodic patterns.

An *anchor event* is an event that has happened in the past in a period of time $<t_{start}, t_{end}>$ and that will happen in the future period of time $<t'_{start},t'_{end}>$. A time unit when an *anchor event* has happened in the past does not need to have the same index as a time unit when the event will happen in the future. For example, the first day of spring session in 2015 is on a different date that in 2016 or the same sport event is scheduled

by different days in different years. Additionally and order of *anchor events* can be different in two different periods of time. In the future we shall call *anchor events* as simply *events*.

A *history of events* over a sequence of time units H in the past is a set of pairs $<e_i,k_i>$ for $i=1, \ldots, n$ such that an event e_i happened in the past in a time unit k_i, i.e. an event e_i happened in a period of time $H[k_i]$.

A *plan of events* over a sequence of time units F in the future is a set of pairs $<e_i, k'_i>$ for $i=1, \ldots, n$ such that it is planned that an event e_i will happen in the future in a time unit k'_i, i.e. an event e_i is planned to happen in a period of time $F[k'_i]$.

For example, consider a set of events $\{e_1, e_2, e_3, e_4\}$, and a sequence of 5 time units H in the past and 5 time units F in the future. A sample history of the events in H is a set of pairs $\{<e_1,2>, <e_2,3>, < e_3,3>, <e_4,5>\}$. A sample plan of the events in F is a set of pairs $\{<e_1,2>, <e_2,4>, < e_3,3>, <e_4,3>\}$. The events e_1 and e_3 are planned for the same periods of time as it happened in the past, however time units and an order of the events e_2 and e_4 will change in the future.

A language of *event expressions* is defined to determine the possible starting points of periodic patterns. An *event expression* determines a set of time units either in H or in F.

The language of *event expressions* is defined in the following way. Let $E = \{e_1, \ldots, e_m\}$ be a set of m names of events. An *event term* is either a string *"before* e_i*"* or *"at* e_i*"* or *"after* e_i*"* where $e_i \in E$.

Let t be an *event term*. Then, the simplest *event expression* is *"t"*. Let x_i and x_j be the *event expressions*. Then *"*x_i*"*, *"not* x_i*"*, *"*x_i *and* x_j*"*, *"*x_i *or* x_j*"* are the event expressions.

As a simple example assume the following set of events: *Uni-events = {"start of enrolment", "start of session", "end of enrolment", "end of session"}*. A period of time when the students are allowed to enrol the courses is determined by an event expression *"after start of enrolment and before end of enrolment"*. An expression *"(after start of enrolment and before start of session) or (after end of enrolment and before end of session)"* defines a union of periods of time that include an early enrolment and a period of time when enrolment is not possible. Any late enrolment is defined by an expression *"after start of session and before end of enrolment"*.

Evaluation of event expressions over a history (or a future plan) of events provides a set of time units included in a sequence of time units H (or F). Let $L(E)$ be a language of event expressions that can be created over a set of events E, let $P(h(E))$ be a powerset set of all histories of events in E, and let $P(H)$ be a powerset of all sequences of time units. Then, evaluation of event expression is defined as a function $eval: L(E) \times P(h(E)) \rightarrow P(H)$. Let h be a history of events over a sequence of time units H. Let e be an event and let x and y be the event expressions. Then, evaluation function *eval* satisfies the following properties:

(i) $eval("before\ e",h) = \{H[i]: \exists <e,k> \in h \text{ and } \$1 \leq i < k\}$

(ii) $eval("after\ ",h) = \{H[i]: \exists <e,k> \in h \text{ and } k > i \geq m\}$

(iii) $eval("at\ e",h) = \{H[i]: <e,i> \in h\}$

(iv) $eval("(x)"\$,h)=eval("x",h)$

(v) $eval("not\ x",h)=H-eval("x",h)$

(vi) $eval("x\ and\ y",h)=eval("x",h) \cap eval("y",h)$

(vii) $eval("x\ or\ y",h)=eval("x",h) \cup eval("y",h)$

For example, evaluation of an expression *"after start of session and before end of enrolment"* over a sequence of time units overlapping on a period of one session returns all time units included in a period of late enrolment.

Event expressions are used to determine the typical "locations" of periodic patterns in time depending on the locations of anchor events. The algorithms discovering periodic patterns in the logs, traces, audit trails, etc. obtained in the past provide information about "locations" or something what can be called as a "map" of periodic patterns in time. Then, such a "map" is enhanced with the locations of anchor events in the same period of time. Comparison of the location of anchor events and "locations" of periodic patterns in time allows for specification of event expressions that determine possible occurrences of periodic patterns relative to the "positions" of anchor events. The event expressions can be evaluated on a set of planned events

providing information about future distribution of periodic patterns and about future workloads.

5. Static Workload Estimation

An algorithm for static estimation of future workloads is the following. At the beginning of the future workload estimation we consider a sequence H of n time units in the past and a sequence F of n time units in the future. We assume that both sequences have the same structure, i.e. the total number of time units is the same, $|H|=|F|$, the respective time units have the same length $\tau_i = \tau'_i$, and the length of time between the respective time units is the same, $(t_i - t_{i+1})=(t'_i - t'_{i+1})$. Next, we consider a set of anchor events E and a *history* h of events in E over H and a *plan* f of events in E over F. It is important to note, that a history h does not need to be exactly the same as a plan f, i.e. the anchor events in the future do not need to be assigned to the same time units as in the past. Next, we consider a set of m periodic patterns $P=\{p_1, \ldots, p_m\}$ discovered from information included in the logs, traces, audit trails, etc., and recorded over the time units in H. For each periodic pattern $p_i \in P$ a start time unit in H, total number of instances, periodicity and carrier are recorded.

A static workload estimation uses information about start points for periodic patterns found in the sequence of time units H in the past to estimate start points of the same periodic patterns in a sequence of time units F in the future.

Static workload estimation is performed in two steps. In the first step we create a set of triples $<p_i, x_i, b_i>$ where $p_i \in P$, x_i is an event expression that uses anchor events to determine the periods of time when a periodic pattern p_i has happened and b_i is a measure of belief that the same periodic pattern will happen in the same periods of time in the future. For each periodic pattern $p_i \in P$ it is possible to create many such triples that determine our beliefs of its occurrence in certain periods of time determined by event expressions.

As an example consider a set of periodic patterns $\{p_1, p_2, p_3\}$ such that p_1 started in a time unit $H[1]$, p_2 started in a time unit $H[3]$, and p_3 started in a time units $H[4]$. Next, assume that an anchor event e_1 has happened in a time unit $H[1]$, an event e_2 has happened in $H[2]$ and an event e_3 has happened in $H[4]$. Therefore, we obtain the following history of anchor events $\{<e_1,H[1]>, <e_2,H[2]>, <e_3,H[4]>\}$. Next, we create event expressions and we assign the periodic patterns to the event expressions and we estimate the measures of beliefs for each assignment. We consider the following 4 event expressions: *"before e_1", "at e_1", "at e_2 or (after e_2 and before e_3)", "at e_3 or after e_3"*.

The locations of time units together with the start points of periodic patterns and anchor events are given in Fig. 1.

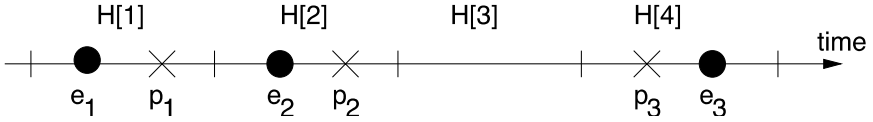

Fig. 1: Past time units, anchor events, and start points of periodic patterns.

Finally, we create the following triples that determine the possibilities of occurrence of periodic patterns in the periods of time determined by the event expressions: $<p_1,$"before e_1",0.3>, $<p_1,$"at e_1",0.7>, $<p_2,$"at e_2 or (after e_2 and before e_3)",0.9>, $<p_2,$"at e_3 or after e_3",0.1>, $<p_3,$"at e_3 or after e_3",1.0>.

In the second step of static workload estimation we iterate over the triples created in the previous step and for each triple $<p_i, x_i, b_i>$ we evaluate an event expression x_i in a future plan f, i.e. we find $eval(x_i, f)$ and we assign time units found with a pair $<p_i, b_i>$. At the end of this process each time unit in F has a set of pairs that determine which periodic patterns will start in a given time unit and with what chances.

Assume, that in the example above, a future plan f of anchor events is the following $\{<e_1, F[1]>, <e_2, F[4]>, <e_3, F[3]>\}$. Then, evaluations of anchor expressions return the following results: *eval("at e_1", f)={F[1]}, eval("at e_2 or (after e_2) and before e_3)", f)={F[4]}, and eval("at e_3 or after e_3", f)={F[3],F[4]}.* Therefore, a periodic patterns p_1 has *0.3* chances to start before a time unit *F[1]* and *0.7* chances to start during a time unit *F[1]*. A periodic pattern p_2 has *0.1* chances to start in a time unit *F[3]* and *0.9* chances to start in a time unit *F[4]*. Finally, a periodic pattern has *1.0* chances to start in the time units *F[3]* or *F[4]* or after *F[4]*.

The locations of time units together with the estimated start points of periodic patterns and anchor events are given in Fig. 2.

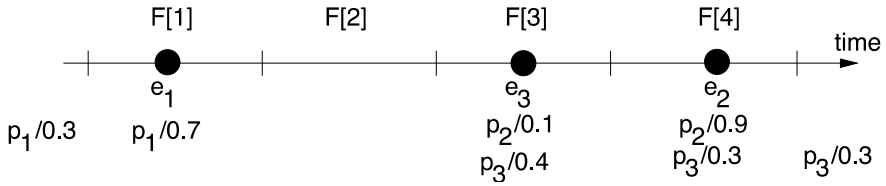

Fig. 2: Future time units, anchor events, and start points of periodic patterns.

Finally, static workload estimation is created as an aggregation of workload traces of each periodic pattern. Consider a periodic pattern $<C, f{:}t, p>$ where C is a sequence of multisets of database operations where $|C| \leq n$. A *trace of C* spanning over n multisets and starting at a time unit f where $f+|C|-1 \leq n$ is denoted by $tr(C, f, n)$ and it is defined as sequence of $f-1$ empty multisets followed by a sequence of multisets C and and followed by $n-(f+1)-|C|$ empty multisets. For example, $trace(e_1e_2^2,3,5)$ is a sequence of multisets $\varnothing\varnothing e_1e_2^2\varnothing$. A *trace of a periodic pattern* $<C, f{:}t, p>$ over n time units where $f+(t-1)*p+|C|-1 \leq n$ is denoted by $TR(<C, f{:}t, p>, n)$ and it is defined as a union $tr(C, f, n) \oplus tr(C, f+p, n) \oplus \ldots \oplus tr(C, f+(t-1)*p, n)$. In the other words, a trace of periodic pattern is a union of traces of its carrier over n multisets such that each trace starts at the time units $f, f+p, \ldots, f+(t-1)*p$. For example, a trace of periodic pattern $<e_1e_2^2, 2{:}2, 1>$ over 5 time units is the following union of sequences of multisets $\varnothing e_1e_2^2\varnothing\varnothing \oplus \varnothing\varnothing e_1e_2^2\varnothing = \varnothing e_1(e_1,e_2^2)e_2^2\varnothing$.

6. Dynamic Workload Estimation

An important problem in dynamic workload estimation is identification of periodic patterns in a stream of operations submitted for processing in each time unit. The problem is very similar to a problem of *conformance checking* in a context of process mining where consistency of a discovered model is verified with a stream of coming events. However, in our case matching of earlier discovered periodic patterns in a stream of incoming operations must be done as soon as it is possible due to early workload estimation requirement.

Identification of periodic patterns in a stream of operations is performed by the computation of a signature of the pattern and a signature of a stream of operations and then comparison of the signatures. Due to the strict time constraints we apply a simple way of finding a signature of periodic pattern by taking the first element of a carrier $C[1]$ from the pattern. A signature of a stream of operations in a given time unit $S[F[i]]$ is equal to all operations included in the time unit except all operations that belong to the earlier identified and still running periodic patterns. If in a time unit $F[i]$ the first element of a carrier $C[1] \subseteq S[F[i]]$ then we assume that the periodic pattern starts in a time unit $F[i]$. Of course correctness of identification of a periodic pattern in a running stream of operations is a compromise between correctness of identification and time spent on identification. A higher level of correctness can be achieved when more time units from a stream of current operations are compared with a carrier of a pattern. On the other hand more time units used for identification of a pattern means more time spent on data processing until the pattern is identified or rejected. In the present work we use a single time unit signature to identify or to reject the patterns as soon as it is possible.

A dynamic workload estimation algorithm is an *online algorithm* [17] that processes a set of pairs $<F[i], P[i]>$ where $P[i]$ is a set of pairs like $<p_i,b_i>$ that represent periodic patterns that suppose to start at time unit $F[i]$.

A dynamic workload estimation algorithm operates in the following way. At the end of each current time unit the algorithm uses the signatures of periodic patterns that supposed to start in the time unit and all operations processed to find which planned periodic patterns started in the time unit. If a pair $<p_i, b_i>$ is included in a set of pairs associated with $F[i]$ and comparison of signatures indicates that a periodic pattern p_i started in a time unit $F[i]$ then all pairs with the first element p_i are removed from all set of pairs. If a pair $<p_i, b_i>$ is included in a set of pairs associated with $F[i]$ and comparison of signatures indicates that a periodic pattern p_i did not start in a time unit $F[i]$ then a value of b_i is divided by the total number of

remaining pairs $<p_i, b_j>$ and the results as added to each b_j. All pairs with the first element p_i are removed from all set of pairs. It increases the chances for a periodic pattern to happen in the remaining periods of time determined by its event expression for p_i. If there are no other pairs like $<p_i, b_j>$ then no changes to the plans are performed.

As a simple example consider a set of pairs *{<F[0], {<p_1,0.3>}>, <F[1], {<p_1,0.7>}>, <F[3], {<p_2,0.1>, <p_3,0.4>}>, <F[4], {<p_2,0.9>,<p_3,0.3>}>, <F[5], {<p_3,0.3>}>}* obtained from the previous example (see Fig. 2). If at the end of time unit *F[0]* a periodic pattern p_1 has not started yet then its chances to occur are increased to *1.0* in a time unit *F[1]*. If at the end of *F[3]* a periodic pattern p_2 has not started yet then its chances to occur in *F[4]* are increased to *1.0*. If p_3 started in *F[3]* then all other occurrences of p_3 are removed from the plans.

7. Summary and Future Work

This work shows how to use the periodic patterns discovered in the processing of database applications for estimation of the future levels of workload in a database system. The periodic patterns are discovered through analysis of logs, traces, audit trails of database operations recorded in the past. A mapping of periodic patterns into *anchor events* and the periods of time between the *anchor events* is created by a database administrator. Then, the *anchor events* and periods of time between the *anchor events* are mapped into the future history. A composition of both mappings provides information about a distribution of periodic patterns in the future. Information about the future workload levels is obtained through expansion of the patterns into the workload traces.

The mappings provide a set of possible time slots where each periodic pattern can start possibly start together with the estimations of chances for each one of them. Many possible time slots for each periodic pattern allow for the dynamic estimation of the future workloads in a situation when the first and each successive case does not happen. Dynamic estimation of the workloads requires fast conformance checking of periodic patterns with the current database workload. At the moment a set of database operations included in the first element of the pattern's carrier is used to identify the patterns in a flowing stream of database operations. It is a common sense solution, which unfortunately does not provide a better level of accuracy in identification of the periodic patterns. The fast solution to a conformance problem in identification of the patterns is one of the major problems left for the future research in this topic.

The main objective of automated performance tuning is to eliminate significant imbalances of database workload and in a consequence to "flatten" a workload over a longer period of time. One more remaining research problem addresses a merge for the past distributions of periodic patterns before "flattening" the workloads with the most up to day distribution of the patterns after automated performance tuning has been applied. It also includes the research questions on how to create a new distribution of periodic patterns and how to make a process of automated performance tuning stable.

8. References

[1] N. Bruno, Ed., *Automated Physical Database Design and Tuning.* CRC Press Taylor and Francis Group, 2011.

[2] M. Zimniak, J. Getta, and W. Benn, "Mining periodic workload patterns in database audit trails," *International Journal of Database Theory and Application,* vol. 6, no. 6, pp. 63–74, 2013.

[3] J. L. Lo, L. A. Barroso, S. J. Eggers, K. Gharachorloo, H. M. Levy, and S. S. Parekh, "An analysis of database workload performance on simultaneous multithreaded processors," in *ACM SIGARCH Computer Architecture News,* vol. 26, no. 3, 1998, pp. 39–50.

[4] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysis and demand prediction of enterprise data center applications," in *Workload Characterization, 2007. IISWC 2007. IEEE 10th International Symposium on,* 2007, pp. 171–180.

[5] C. Gupta, A. Mehta, and U. Dayal, "PQR: Predicting query execution times for autonomous workload management," in *Autonomic Computing, 2008. ICAC'08. International Conference on,* 2008, pp. 13–22.

[6] A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson, "Statistics- driven workload modeling for the cloud," in *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on,* 2010, pp. 87–92.

[7] J. Duggan, U. Cetintemel, O. Papaemmanouil, and E. Upfal, “Performance prediction for concurrent database workloads,” in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data,* 2011, pp. 337–348.

[8] M. Ahmad, S. Duan, A. Aboulnaga, and S. Babu, “Predicting completion times of batch query workloads using interaction aware models and simulation,” in *Proceedings of the 14th International Conference on Extending Database Technology,* 2011, pp. 449–460.

[9] W. Wu, Y. Chi, H. Hacigumus, and J. F. Naughton, “Towards predicting query execution time for concurrent and dynamic database workloads,” *Proceedings of the VLDB Endowment,* vol. 6, no. 10, pp. 925–936, 2013.

[10] M. Zimniak, J. Getta, and W. Benn, “Predicting database workloads through mining periodic patterns in database audit trails,” *Vietnam Journal of Computer Science,* vol. 2, no. 4, pp. 201–211, 2015.

[11] W. M. P. van der Aalst, *Process Mining Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[12] A. Rozinat and W. M. van der Aalst, “Conformance checking of processes based on monitoring real behavior,” *Information Systems,* vol. 33, no. 1, pp. 64–95, 2008.

[13] M. de Leoni and W. M. P. van der Aalst, “Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming,” in *Proceedings of the 10th International Conference on Business Process Management,* 2013.

[14] J. Getta and M. Zimniak, “Deriving complex periodic patterns from nested logs of events,” *International Journal of Knowledge Engineering,* vol. 1, no. 3, pp. 223–229, 2015.

[15] ——, “Discovering complex incomplete periodic patterns through logical derivations,” *Open Journal of Social Sciences,* vol. 3, no. 1, pp. 8–15, 2015.

[16] D. A. Simovici and C. Djeraba, *Mathematical tools for data mining: set theory, partial orders, combinatorics.* Advanced Information and Knowledge Processing. Springer, 2008.

[17] A. Fiat, *Online Algorithms: The State of the Art* (Lecture Notes in Computer Science). Springer, 1998.