# On some heuristic method for optimal database workload reconstruction

Marcin Zimniak[1], Marta Burzańska[2], and Bogdan Franczyk[1]

[1] Information Systems Institute
Leipzig University, Germany
{zimniak,franczyk}@wifa.uni-leipzig.de
[2] Faculty of Mathematics and Computer Science
Nicolaus Copernicus University, Toruń, Poland
quintria@mat.umk.pl

**Abstract.** The paper deals with the problem of database workload and its reconstruction. It is partially related to workload as a sequence of SQL statements in physical database design problem. An efficient algorithm based on greedy heuristic method for workload reconstruction using periodic patterns is provided. The quality of reconstruction is estimated by proposed reconstruction quality indicator.

**Keywords:** workload and workload reconstruction · periodic patterns· periodic patterns discovery· heuristic methods · optimization in physical database design .

## 1 Introduction

The problem of physical database design and tuning often requires detailed workload analysis. The paper "Automatic physical design tuning: workload as a sequence" [1] published in 2006 defines the structure of the workload as a sequence of SQL queries. This paper influenced a lot of research on workload, however, the topic of studying workload on an abstract plane to boost the performance of a database management system has not been fully addressed. The following paper introduces a new approach to the database workload based on the multiset concept. In this approach we do not analyze a sequence of SQL queries, instead, we take into account the multisets of queries abstract syntax trees (AST). Query syntax tree is viewed as the implementation of the SQL query under specific conditions at a specific time in a DBMS. Through the use of the AST concept, the problem of the equivalence of SQL queries in the process of generating the workload has been avoided. Similar problems have been discussed in previous papers [11, 12]. Both dealt with application of periodic patterns methods to a series of SQL queries. However, despite analyzing the workload on a physical level, both papers lack (among other things) the analysis of transition cost between queries and DBMS states. Such costs are important when working with bigger workloads and calculating their total cost, which then is used in various recommendation systems.

Prediction and reconstruction of the workload can become a useful tool for optimizing recommendation modules. At a later stage, they can be used in the development of some form of automated physical database design tools.
The following paper presents a new concept for the reconstruction of the workload. This concept is a result of combining of the data mining of periodic patterns with elements of physical database design. Those elements include cost models used in DBMS optimization. The aim of the article is to provide an effective heuristic method to search for the optimal workload reconstruction and also to provide the reconstruction quality measure. Both elements will be used in workload prediction and determination of the workload prediction degree.

This paper is organized as follows. The second section presents the concept of workload, including cost issues related to the physical workload model. The third section deals with defining periodic patterns and their derivation rules. This section also defines the workload reconstruction and reconstruction quality measure proposal. Section 4 introduces the algorithm which uses one of the heuristic methods in order to generate an optimal reconstruction. Section 5 concludes and discusses further research plans.

## 2 Workload

The article examines the workload on two essentially independent planes. On the abstract plane, we do not include cost relations between database objects and costs resulting from the transition from one database configuration to another. As for the physical workload model, it reflects the behavior of the database system for a given time period during which the aforementioned costs are taken into account.

### 2.1 Database processing model

We consider a typical relational database system where the relational model of data is used to represent data containers. Let $x$ be a nonempty set of attribute names later on called as a *relational schema* and let $dom(a)$ denotes a domain of attribute $a \in x$. A *tuple* $t$ defined over a schema $x$ is a full mapping $t : x \to \bigcup_{a \in x} dom(a)$ and such that $\forall a \in x,\ t(a) \in dom(a)$. A *relational table* $r$ created on a schema $x$ is a set of tuples over a schema $x$.

Query processor transforms SQL statements submitted by the user applications into the query execution plans formulated as the expressions of extended relational algebra. The operations of extended relational algebra include the implementation dependent variants of operations of standard relational algebra such as *selection*, *projection*, *join*, *antijoin*, *set operations*, and other operations like *grouping*, *sorting*, and *aggregate functions*. Due to the different implementation techniques, the operations included in the basic system of relational algebra, e.g. *selection* or *join* contribute to an number of different elementary operations depending on their implementations, e.g. *index based selection*, *full scan selection*, *hash based join*, *index based join*, etc.

### 2.2 Abstract workload model

k SQL statements submitted by $M$ users within the *user applications* $a_1, \ldots, a_n$ are recorded in an *application trace.* A *trace of an application* $a_i$ is a finite sequence of pairs $<c_i{:}t_{c_i}, s_{i_1}{:}t_{i_1}, \ldots, s_{i_n}{:}t_{i_n}, d_i{:}t_{d_i}>$ where $c_i$ is a *connect* statement, $t_{c_i}$ is a timestamp when the statement has been processed, each $s_{i_j}$ is SQL statement with a timestamps $t_{i_j}$ attached, and $d_i$ is a *disconnect* statement with its timestamp $t_{d_i}$. Processing of an application $a_i$ starts from processing of a connect statement $c_i$, the processing of SQL statements $s_{i_j}$, and it finally ends with processing of a disconnect statement $d_i$.

An *audit trail* is a sequence of interleaved trails of user applications. For example, a sequence $<c_i{:}t_{c_i}, s_{i_1}{:}t_{i_1}, c_j{:}t_{c_j}, s_{j_1}{:}t_{j_1}, s_{i_2}{:}t_{i_2}, d_i{:}t_{d_i}\ d_j{:}t_{d_j}>$ is a sample audit trail from the processing of applications $a_i$, and $a_j$.

In the subsequent text the implementation record of each of the $k$ SQL queries is placed within a non-empty period of time [a,b] comming from M users . It follows that syntactically equivalent `SELECT` queries can have different implementations. The problem of SQL query equivalence is a complex problem [3, 2].

We can circumvent this problem by using query execution plans accessible through the use of mentioned `EXPLAIN PLAN` command. Such plans usually take the form of enhanced syntax trees and are treated as query implementations. `SELECT` query analysis on a non-empty period of time [a,b] results in extraction of the syntax trees which are then placed in a *syntax tree table* [12]. This table contains a complete and compressed information about the syntax trees of SQL statements and the number of their occurrences in the analyzed workload. The paper [12] contains detailed information about the construction of such tables. It is worth noting, that a syntax tree is represented in a syntax tree table only once, no matter how many times it is included in the other syntax trees as a subtree. The cases of shared subtrees resulted in the adoption of the multisets theory. In the following text we define a multiset $M$ is defined as a pair $<S, f>$ where $S$ is a set of values and $f : S \to N^+$ is a function that determines multiplicity of each element in $S$ and $N^+$ is a set of positive integers [9]. We also assume that the syntax trees have been unambiguously labeled by the letters of a fixed alphabet - a set of natural numbers.

For simplicity in further definitions, we assume the condition that the execution time of each query together with the generated load was recorded unambiguously for a given workload.

For the given time period $[a, b]$ and the number of analyzed SQL queries $k$, let $n \leq k$ be a minimal number of the time period's equal divisions such that the total execution time for each of the queries (including all implementational costs) fits in exactly one time segment with the length $|[a, b]/n|$. Such time period division generates $n$ time segments of equal length called the *time units.* Each time unit has its established length and a start point in time [12].

Let $U$ be a nonempty sequence of $n$ disjoint time units over which a workload of k queries is recorded and let $|U| = n$ denote the total number of time units in $U$. Then $U[m]$ denotes the $m$-th time unit in $U$ for $m = 1..n$. Let $V$ be a mapping of a subset $1..k$ of natural numbers representing workload queries (or more precisely query syntax trees) into a subset $1..n$ of natural numbers representing time units. This syntax tree-to-time unit mapping allows for registrations of syntax trees in syntax tree tables. Those tables are later used to locate similar syntax trees whose location in the [a,b] only slightly differs from the "ideal" periodicity.

Let $L$ be a set of all syntax trees (including all syntax subtrees) generated from a given set of $k$ SQL queries executed in a specified time period $[a, b]$ with a given sequence of time units $U$. For each $T \in L$, a *workload trace of a syntax tree* $T$ is a multiset $W_T$ of time units such that $W_T[i] =<\{T\}, f_i>$ and $f_i(T)$ is equal to the total number of times the syntax tree $T$ was processed in the $i$-th time unit $U[i]$.

In addition, let the syntax tree table comprising all syntax trees and subtrees be given. A *workload* of the set $L$ is denoted by $W_L$ and $W_L = \biguplus_{T \in \mathbf{L}} W_T$

## 2.3 Physical workload model

In this paper, the aforementioned $W_L$ structure was used instead of the earlier model of the physical workload considered in [1]. In addition, the following extension was adopted. Instead of the sequence of `SELECT` expressions, the sequences of multisets of syntax trees were used. Due to the use of the SQL query execution plans it was possible to register on-the-fly: operations, containers and access paths with costs (and workloads) at the level of each operation in the execution plan.

Let the enumeration of the syntax trees be monotonic through the set of natural numbers with accordance with the timestamp values. Let $\{S_k\}$ be a multiset of syntax trees with a given $U(t)$. Let $(S_1, S_2, ..., S_N)$ be a sequence of $N$ multisets registered in $W_L$.

There are many methods for registering the `SELECT` queries in relational databases. An example of such a method in the Oracle DBMS is the so-called *audit trail* applied in [12]. In addition, there are built-in workload logging tools (eg. *Profiler* tool in Microsoft SQL Server). A *physical structure*should be understood as any access path supported by the database server. Those structures include, among others: indexes, materialized views, multidimensional clustering of tables, etc. A *configuration* of the workload $W_L$ is the set of possible-to-use physical database structures that can be materialized. A physical structure is considered *significant* if it can potentially be used in the execution plan of a `SELECT` query (even if it was not used in the final execution plan at the defined time period $[a, b]$). The topic of costs in database systems is a very broad subject, simplified in this paper. In order to have comprehensive knowledge about costs, eg. in the Oracle database system, we refer the reader to [6].

The following notation was used in the further part of the work. `COST` $(\{S\}, C)$ means the total cost of operations in the `EXPLAIN PLAN` expression encoded with the appropriate syntactic trees at the given $C$ database configuration. Let `TRANSITION-COST` $(C_i, C_j)$ be the minimum cost of the transition between the $C_i$ and $C_j$ configurations. These costs include costs related to the creation /

removal of indexes and other physical structures. We assume the available optimization mechanisms that estimate costs on an ongoing basis, perform without unnecessary overhead using built-in extensions such as what-if, etc.

*Representation* of the $(\{S_1\}, \{S_2\}...\{S_N\})$ sequence execution is defined as a sequence $(C_1$, $\{S_1\}$, $C_2$, $\{S_2\}$ ... $C_N$, $\{S_N\}$, $C_{N+1})$. It is a sequence in which each multiset of syntax trees has a pre-configuration and post-configuration (we allow for empty configurations).

We define the *sequence execution cost* $<C_1, \{S_1\}, C_2, \{S_2\}...C_N, \{S_N\}, C_{N+1}>$, as $\sum_{k=1}^{N}$ ($\texttt{COST}$ $(\{S_k\}, C_k)$ + $\texttt{TRANSITION-COST}$ $(C_{k-1}, C_k)$)+ $\texttt{TRANSITION-COST}$ $(C_N, C_{N+1})$.
The zero state $C_0$ can be, for example, the initial state of the database or its value can be set by built-in what-if applications. All the costs discussed so far are accompanied by workloads and time units. In the further part of the article, we assume that in each $U(i)$, $i = 1, 2, ..., n$, the total workload is directly proportional to the total costs, treating the concepts of costs and workloads interchangeably.

## 3 Workload reconstruction using periodic pattern theory

Workload reconstruction plays an important role in the automated physical database design and in physical design optimization mechanisms. In this paper out of all possible reconstructions, we investigate only those most probable and, at the same time, the most acceptable when it comes to costs. It means we study those $W_L$ into $W_L$ mappings for which the total costs during reconstruction does not exceed initial total workload costs. Those mappings maintain the consistency of the subsequences implemented through a minimal set of periodic patterns with an emphasis on maximizing quality indicators of periodic patterns

### 3.1 Periodic patterns

The theory and applications of the concept of periodic patterns to the workload prediction problem were discussed in the previous works of one of the authors [11, 12]. The theory of periodic patterns is well known. It grew out of, among others, the periodic sets [7] as well as periodic events [8].

Let the workload $W_L$ and the sequence of time units U be given. The sequences $C, C' \subseteq W_L$ of the same length are called *equivalent* if $C = C'$ occurs for all corresponding coordinates.

A *periodic pattern* in a workload $W_L$ is a tuple $<C,\ f,\ t,\ p,>$ where:

1. the *carrier* $C$ determines a non empty subsequence $C \subseteq W_L$
2. $f$ is a number of time unit in $U$ where the repetitions of $C$ start
3. $t$ is a total number of occurrences of equivalent sequences $C \subseteq W_L$, such that $p$ denotes the number of consecutive time unit elements after which the $t$ pairs of neighboring sequences are equivalent.
4. Parameters $f$, $t$, $p$ satisfy the following inequality: $f, t \geq 1$, $p \geq 0$, $f + (t - 1) * p + |C| - 1 \leq |U|$

Also, if $t = 1$ then $p = 0$ and the pattern $<C,f,1,0>$ is called the *trivial periodic pattern* (*trivial* pattern)

Let $<C,\ f,\ t,\ p,>$ be a periodic patterns in $W_L$ with a given $U$.
A *trace of a carrier* $C$ is a subsequence $C \subseteq W_L$, denoted $tr(C,f,n)$, in which the first $f-1$ elements are the empty multisets.
A *trace of a periodic pattern* $<C,\ f,\ t,\ p,>$ over the time unit sequence $U$, under the condition $f+(t-1)*p+|C|-1 \leq n$, is a subsequence $TR(<C,f,t,p>,n)$ of a sequence $W_L$ such, that $TR(<C,f,t,p>,n) = tr(C,f,n) \uplus tr(C,f+p,n) \uplus \ldots \uplus tr(C,f+(t-1)*p,n)$

## 3.2 Derivation rules

According to the work [4], for the periodic patterns we define the derivation rules by means of which new periodic patterns can be generated. Given the $W_L$ and $U$, the following rules take place:
**Rule 0 (Triviality)** Let $C$ be a submultiset, such that $C \subseteq W_L[f]$ for $f \in \{1,..,n\}$. Then $<C,f,1,0>$ is a (trivial) periodic pattern in $W_L$. This rule states that in any non-empty workload $W_L$ , you can find all the trivial patterns of the form $<C,f,1,0>$.
**Rule 1 (Normalization)** Let $<C,f,t,p>$ be a periodic pattern in $W_L$. Then $<C',f',t,p>$, where $f' = f+i$, is a periodic pattern in $W_L$, such that $C'$ is formed from $C$ by the elimination of all of the i-empty multisets preceding $C$ and/or the elimination of all of the empty multisets trailing $C$.
**Rule 2 (Exclusion/Duality)** Let $<C,f,t,p>$ be a periodic pattern in $W_L$. If $f_{split} = f+i*p$ for $0 \leq i \leq t-1$, then only one of the following patterns is a periodic pattern: a) $P = <C,f,i-1,p>$ with $W_L = W_L \setminus TR(P',n)$ is a periodic pattern in $W_L$ such that $P' = <C,\ f_{split},\ t-i+1,\ p>$, b) $P' = <C,\ f_{split},\ t-i+1,\ p>$ with $W_L = W_L \setminus TR(P,n)$ is a periodic pattern in $W_L$ such that $P = <C,f,i-1,p>$

Contrary to the previously mentioned research, in this paper we omit the concept of periodic patterns "validity". As a result, the process of building and applying derivation rules may result in "depletion" of the workload that takes place in the Exclusion/Duality rule
**Rule 3 (Elimination)** Let $<C,f_i,t_i,p_i>$, $<C,f_j,t_j,p_j>$ be periodic patterns in $W_L$, such that $f_i < f_j$. Then the following cases hold:

(1) If $t_i = t_j = 1$ then $\langle C,\ f_i,2,\ f_j - f_i\rangle$ cannot be a periodic pattern in $W_L$ (the carrier C starting from position $f_i$ can occur a maximum of 1 time in $W_L$ - in accordance with the definition. The following sub-rules stating the maximum of $t_i$, $t_j$ times starting from $f_i$, $f_j$ respectively)
(2) If $t_i = 1$, $t_j > 1$ and $f_j - f_i = p_j$, then $\langle C,\ f_i,t_j+1,\ p_j\rangle$ is not a periodic pattern in $W_L$.
(3) If $t_j = 1$, $t_i > 1$ and $f_j = f_i + t_i * p_i$, then $\langle C,\ f_i,t_i+1,\ p_i\rangle$ is not a periodic pattern in $W_L$.
(4) If $t_j \neq 1$, $t_i \neq 1$, $p_i = p_j$ and $f_j = f_i + t_i * p_i$, then $\langle C,\ f_i,t_i+t_j,\ p_i\rangle$ is not a periodic pattern in $W_L$.

**Rule 4 (Decomposition)** Let $<C,f,t,p>$ be a periodic pattern in $W_L$. Then $\langle C', f,t, p\rangle$, where a carrier $C'$ is a subsequence of a carrier $C$, is a periodic pattern in $W_L$.
**Rule 5 (Composition)** Let $<C_i,f_i,t,p>$, $<C_j,f_j,t,p>$ be periodic patterns in $W_L$, such that $f_i \leq f_j$ and $\uplus_{s\in\{i,j\}} TR(< C_s, f_s, t, p >, n) \subseteq W_L$. Then $\langle C_k, f_i,t, p\rangle$ is a periodic pattern in $W_L$ such that $C_k = tr(C_i, 1, f_j - f_i + |C_j|) \uplus tr(C_j, f_j - f_i, f_j - f_i + |C_j|)$.

For example, given the periodic patterns $\langle TV^2$,1,3,4$\rangle$ and $\langle T$,4,3,4$\rangle$ in a workload $W_L$ with given $U$ then $\langle TV^2\emptyset T$, 1,3, 4$\rangle$ is a periodic pattern in $W_L$ as well.

### 3.3 Reconstruction and workload reconstruction quality measure

The model theory, in George Polya's view, deals with the equivalence classes of similar periodic sequences. The motivation behind the reconstruction concept is the fact that for each sequence of determined processes (and with such we are working) there exists a period and pre-period [5]. The theory of shifts, in terms of periodic patterns for sequences, makes it possible to indicate the minimal sets of generators and their calculation is possible with the help of efficient algorithms. The problem raised in the work relates to parallel processes that interact with each other in real time. The study of the periodicity of such structures is close to the study of symbolic dynamics in particular of groups of automorphisms of similar structures.

Let $R$ be a non-empty set of periodic pattens in a $W_L$ given time unit sequence $U(n)$. We say that $R$ is a *reconstruction of the workload* $W_L$ in $U(n)$ if:

i. $\uplus_{s=1}^{|R|} TR(< C_s, f_s, t_s, p_s >, n) = W_L$
ii. all TRs implementing connect-disconnect processes remain consistent in relation to each other. We allow duplication of database connect/disconnect processes in case of hypothetical processes, assuming that logging in and logging out does not involve costs.

As a *quality measure of the reconstruction* $R$ is a real value $0 \leq m_R < 1$ defined as:
$m_R = 1 - (1/\sum_{i=1}^{|R|}(\|C_i\| * t_i))^{1/|R|}$
where $\|C_i\|$ is the length of the carrier $C_i$, $|R|$ is the cardinality of $R$. When $R = R_0 = \{< W_L, 1, 1, 0 >\}$ we assume that $m_{R_0} = 0$.

Let the $R_i$, $R_j$ be reconstructions in $W_L$ with a given $U(n)$. We say that the reconstruction $R_i$ is *better* (*more feasible*) than the reconstruction $R_j$ (denoted $R_i > R_j$) if:

a) $m_{R_i} \geq m_{R_j}$,
b) $|R_i| \leq |R_j|$
c) the total sum of the sequence execution costs in $R_i$ is not greater than the total sum of the sequence execution costs in $R_j$.
d) The number of the corresponding predictive patterns quality measures in the reconstruction $R_i$ is greater than the respective number of measures in $R_j$, with at least one quality measure being taken into account.

The predictive patterns quality measures have been described in [4] and may easily be adapted to the generalized concept of predictive patterns described in [10].

There is one more qualitative measure of periodic patterns. Namely, the absolute number of different syntax trees in the C carrier of the given periodic pattern. If we have two different periodic patters $P$ and $P'$ generating the same costs, with $P$ being more feasible then $P'$ for most of the quality measures from [4], we say that $P$ *dominates* over $P'$ if $|supp(\{C_P\})| > |supp(\{C_{P'}\})|$ where $\{C_P\}$ is a multiset of the carrier C in a periodic pattern P.
The total cost of sequence execution in a reconstruction is the sum of the costs generated by the sequence of traces of all periodic patterns of the given reconstruction.

The concept of the database workload reconstruction presented in this paper was developed to predict the future database load. The benefits of estimating the optimal prediction are the databases optimization possibilities. Based on the database load forecast, one can create, for example, indexes, materialized views, etc. These structures can then be used at the right time in the future in such a way that, with their help, one can reconfigure significant structures even better than those proposed by existing advisory devices. Another possible application of the database workload reconstruction is the prediction of configuration. The encoding of `SELECT` queries using execution plan syntax trees enabled the current registration of important physical structures used in the query implementations. Using this fact at a further stage, it is possible to reconstruct the configuration in the given $W_L$, and thus to assess the quality of selection of physical structures used in the configurations.

The optimization issue for the reconstruction and thus for the estimation of optimal prediction is to determine the best reconstruction in the sense of the $>$ relationship described above.

Example 1. Let $W_L = <\{1\},\{1^2\},\{21^2\},\{21\},\{2\}>$. We may have a trivial reconstruction $R_0 = << \{1\},\{1^2\},\{21^2\},\{21\},\{2\} >, 1, 1, 0 >$ along with another cardinality 1 reconstruction $R_1 = << \{1\},\{1\},\{2\} >, 1, 3, 1 >$. Then $m_{R_0} = 0 < \frac{8}{9} = m_{R_1}$, wherein the traces of the reconstructions $R_0$ and $R_1$ are identical and thus the total costs of the sequence execution overlap in both reconstructions. From this it follows that $R_1$ is better than $R_0$ reconstruction.

Example 2. Let $W_L = <\{1\},\{1^2\},\{21^2\},\{21\},\{2\}>$ be a workload with a given $U(n)$ and a pair of indexes $I_1$ and $I_2$ used in the implementation of certain `SELECT` queries stored as syntax trees 1 and 2. The coded information in the syntactic tree is, among others, the access path, i.e. a complete set of relevant physical structures used in implementations. In this example it means that in the trees 1 and 2 the indexes $I_1$, $I_2$ were used respectively. In addition, we assume that the database storage is not affected in any time unit $U(i)$. Assume that the costs of creating indexes $I_1$, $I_2$ are the same regardless of where they are created. In addition, assume that based on the value of costs stored in the syntactic tree table, the costs of implementing the syntactic trees 1, 2 are respectively 3: 4. The deviation of this ratio does not exceed 10% if the expressions are performed together in the same $U(i)$. In the case when the syntax trees 1 and 2 are executed separately (ie in different $U(i), U(j)$) then the cost of the syntax tree 1 is twice as high as the cost of the syntax tree 2 with a deviation not exceeding 5%. The benefits of using both indexes are the same for both syntactic trees regardless of whether 1 or 2 are executed together or separately. In addition, the costs of removing indexes $I_1$ and $I_2$ are equal 0. The calculations of the total reconstruction sequence costs show that the sum of the execution costs of the periodic patterns sequence in $R_1 = \{<< \{1\}, \{1\}, \{2\} >, 1, 3, 1 >\}$ is lesser than a respective sum in $R_2 = \{<< \{1\} >, 1, 3, 1 >\}$ $\{<< \{1\}, \{2\} >, 2, 3, 1 >\}$. Moreover the conditions a) $m_{R_0} = \frac{8}{9} > \frac{2}{3} = m_{R_1}$ with b) bring that $R_1 > R_2$.

## 4 Greedy heuristic method for workload reconstruction

The goal of the presented heuristic is to find the optimal, in the sense of the aforementioned relation $>$, reconstruction $R$ in a set of all reconstructions for a given $W_L$ workload (not exceeding actual costs) at the given $U(n)$. In the algorithm, the input data is: $W_L$ in the form of sequences of natural numbers multisets, generally understood implementation costs of all of the syntax trees registered in $W_L$, as well as the transition costs between individual neighboring configurations. It is further assumed that the $W_L$ coding is given by a sequence of multisets of natural numbers.

The motivation for the heuristic algorithm adaptation for configurations for $W_L$ heuristic reconstruction was the observation that in the physical workload structure, each multiset is inextricably linked to a certain configuration. As shown by the numerous tests in the paper [1], heuristic solutions for configurations are sub-optimal. It can, therefore, be expected that the heuristics for reconstruction will proceed in the same way. Below we present a heuristic algorithm for reconstruction.

*The Algorithm*
1. Let $S = \{s_1, .., s_M\}$ be a set of physical structures in a given workload $W_L$. Using the exhaustive method to find the shortest path in the cost edge graph [1], a set of optimal solutions $P$ for each of the $s_i$ is calculated separately. As a result, we get a set $P = \{p_1, .., p_M\}$. Let $p_i =< a_{i_1}, W_L[1], .., W_L[n], a_{i_{N+1}} >$.

```
    1.1. Let $R := \emptyset$, while ( $W_L <> \{\emptyset\}$) do:
for i: = 1 to n do:
     for each $j \in \mathrm{supp}(\{W_L\})$ do:
1.1.1. $< W_L[i..n], i, 1, 0 >:=< W_L[i..n] \setminus \{st^i_j\}, i, 1, 0 > \cup < \{st^i_j\}, i, 1, 0 >$,
$< W_L[1..n+1-i], i, 1, 0 >:=< W_L[1..n+1-i] \setminus \{st^{n+1-i}_j\}, n+1-i, 1, 0 >$
$\cup < \{st^{n+1-i}_j\}, n+1-i, 1, 0 >$ where $st^i_j$ is j-support element at $W_L[i]$ and
$W_L[i..n] :=< W_L[i], W_L[i+1], .., W_L[n] >$.
```

In each 1.1.1 execute as follows: $R_i := \emptyset$. For each (pairwise) disjoint sequences (represented by the traces of the corresponding trivial periodic patterns), use (for individual sequences (traces) respectively): decomposition rule which is a preserving cost-based pruning technique and then apply any of the rules of: composition and/or exclusion and/or elimination. Proceed in such a way

that in the final result of this step you get a minimal set of periodic patterns $R_i$ with the minimum total value of the sequence execution costs. This set takes into account the optimal "path" of the solutions given by the current $p_i$ and the maximum value of the quality reconstruction measure $m_{R_i}$.
2. Let $C$ be the set of all configurations over the $p_i$ elements.
3. Greedy heuristics for $R$ runs as follows:

3.1. Let $r {=} < c_1, W_L[i], ..., c_N, W_L[n], c_{N+1} >$ be the best configuration in P in terms of the total costs. Let $rr$ be the best reconstruction in terms of a quality measure in $R$ and such that its cost is the closest to the cost of configuration r. Then $P := P \setminus \{r\}$, $R := R \setminus \{rr\}$. Let $C := C \cup \{c_1, ..., c_{N+1}\}$ .

3.2. Select the element $s$ from the set $P$ such that $t =$ `UnionPair(r,s)` (defined in [1]) is a configuration such that its value in terms of the total sequence execution costs among all $P$ configuration is the smallest. Similarly, find in $R$ a reconstruction $R_s$ which is the smallest in $R$ in terms of total costs. In addition, the cost of executing the sequence for the configuration $t$ is smaller than the corresponding costs for the configuration $r$. Parallel, find in $R$ such a reconstruction $R_t$ which in terms of total costs is closest to configuration $t$ (costs are not greater than $t$). If there is no $s$ element, then proceed to step 4. Assign $P := P \setminus \{s\}$ , $P := P \cup \{t\}$, $R := R \setminus \{R_s\}$, $R := R \cup \{R_t\}$. Go to step 3.1.

4. Create a graph for all configurations with $C$ at each level (for every support-element in every $W_L[i]$ ). Find the shortest path in this graph. From the set $R$, return the reconstruction that corresponds to the shortest path in this graph.

## 5 Conclusions and future work

The paper presents a new concept of the workload reconstruction along with a measure of reconstruction quality and an efficient algorithm for generating optimal workload reconstruction, thus estimating the workload prediction quality. Unlike previous periodic pattern detection techniques based on the top-down methodology, the presented recursive approach accelerates the periodic pattern detection algorithm through a different derivation system. This system is mostly based on the reducing derivation rules. This results in reduction of the workload elements that need to be analyzed, which influences the speed of workload analysis.

The search for new heuristics, comparative tests, accuracy, and testing the properties of the proposed measure of quality is the next stage of research. Searching for the proposals for other quality measures and thus new criteria for reconstruction is also included in that stage. In order to verify the efficiency and quality of the presented algorithm, an implementation based on a "live" load course is planned. However, acquiring real companies strategic data is extremely difficult. Currently, the development phase includes an extended implementation including cost optimization. Lastly, more extensive research on workloads containing SQL-99 recursive queries should be conducted.

## References

1. Agrawal, S., Chu, E., Narasayya, V.R.: Automatic physical design tuning: workload as a sequence. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data. pp. 683–694 (2006)
2. von Bültzingsloewen, G.: Optimierung von sql-anfragen für parallele bearbeitung. In: Grundlagen von Datenbanken. pp. 20–22 (1990)
3. Ceri, S., Gottlob, G.: Translating sql into relational algebra: Optimization, semantics, and equivalence of sql queries. IEEE Transactions on software engineering **11**(4), 324–345 (1985)
4. Getta, J.R., Zimniak, M., Benn, W.: Mining periodic patterns from nested event logs. In: Proceedings of the 2014 IEEE International Conference on Computer and Information Technology. pp. 160–167. IEEE Computer Society (2014)
5. Goles, E., Martínez, S.: Neural and automata networks: dynamical behavior and applications, vol. 58. Springer Science & Business Media (2013)
6. Lewis, J.: Cost-Based Oracle Fundamentals. Springer (India) Pvt. Limited (2006), `https://books.google.pl/books?id=85iWawYUsVsC`
7. Matos, A.B.: Periodic sets of integers. Theoretical Computer Science **127**(2), 287–312 (1994)
8. Serafini, P., Ukovich, W.: A mathematical model for periodic scheduling problems. SIAM Journal on Discrete Mathematics **2**(4), 550–581 (1989)
9. Simovici, D.A., Djeraba, C.: Mathematical tools for data mining : set theory, partial orders, combinatorics. Advanced information and knowledge processing, Springer, London (2008), `http://opac.inria.fr/record=b1133711`
10. Zimniak, M., Getta, J.R.: On systematic approach to discovering periodic patterns in event logs. In: International Conference on Computational Collective Intelligence. pp. 249–259. Springer (2016)
11. Zimniak, M., Getta, J.R., Benn, W.: Deriving composite periodic patterns from database audit trails. In: Asian Conference on Intelligent Information and Database Systems. pp. 310–321. Springer (2014)
12. Zimniak, M., Getta, J.R., Benn, W.: Predicting database workloads through mining periodic patterns in database audit trails. Vietnam Journal of Computer Science **2**(4), 201–211 (2015)