Storing Data in the Trial of Complex Technical Products

Igor Shmidt
Perm National Research Polytechnic University - Electrotechnical Department
Komsomolsky Ave. 29, 614990, Perm, Russia
E-mail: shmidt@msa.pstu.ac.ru

***Abstract*—The article considers the ways of organization of databases for the storage of the results obtained during testing. A new variant of the organization of the data to ensure the ability to write to the database different sets of parameters in the form of chronological series. The required set of parameters depends on the modification of the tested technical installation.**

***Keywords:* testing of complex technical products, noSQL, chronological database, document-oriented DBMS.**

I. Introduction

Among the various tasks that require high-volume data storage is allocated the task of gathering and storing of data relating to tests of complex technical products [1,2,3]. A feature of such tests is:

- You need to store a large amount of information that describes the trends of measured parameters during the test. Some tests require the speed of data collection to a few KHz and the duration of the test can be up to several days;
- There is no strict structure of the data collected. Various modifications of products can have different sets of parameters. Thus, over the entire life cycle of the product can be tested with various modifications;
- Data retention period should correspond to the total time of the product life cycle;
- In addition to storing the measured during testing the parameters you want to store data pertaining to the operation of the test items during its life cycle.

Much of the information that accumulates when testing are the results of observation over parameters of the process of testing. When observing the values of the parameters are recorded and linked to the moment of observation. The result is an ordered in chronological sequence of measured values is called a time series.

II. Why RDBMS is Not Fit For Storage of Test Results?

Traditional industrial relational DBMS are not adapted for efficient storage and processing of time series. This is because the relational DBMS include the following mechanisms:

- Normalization, which on the one hand ensures the absence of redundancy and logical errors when updating and sample data, but other decreases the capacity of
- Support of the transaction, which ensures the reliability and consistency of data, but requires substantial resources.

These mechanisms are useful when solving business problems, but they are not needed in the problems of collection and storage of information about the tests. As a result of the performance of processing time series is very low, and the complexity of the description of the application logic high.

To solve the problem of storing temporary data, there are special temporal extension RDBMS, but most of the developments in this direction are intended for storage of slowly varying data.

It is proved that the storage time series in a relational form causes some redundancy in the volume of content and information. Storage of temporal data in a relational is the amount, which is calculated by the formula [4,5].

$$T = \sum_{i=1}^{K} \sum_{t=1}^{Z_i} \left[\sum_{j=1}^{N} V_{ij} + V_t \right], \quad (1)$$

where

T - the total volume of information;

Vij - the amount of information contained in the j attribute tuple I;

K and N - number of records and columns in the table;

Vt - size of the temporary attribute;

Zi is the number of states tuple i.

While the amount of data in the same table excluding temporality is expressed by the formula:

$$T = \sum_{i=1}^{K} \sum_{t=1}^{Z_i} V_{ij}, \quad (2)$$

where

T - the total volume of information;

Vij - the amount of information contained in the j attribute tuple i;

K and N - number of records and columns in the table.

Indeed, it is easy to see redundancy even if only saves the current value of the entity in the temporal RDBMS.

Possible decrease in the volume Vt by changing its type and the creation of a «cheap» surrogate primary keys for the identification of a tuple and calculate its temporary values relative to the parent of a tuple. Decrease in volume Vt through the use of surrogate key is based on the records of

relative time values for each tuple.

The total volume of the stored information will be presented to the expression of the form (3). This method will always be much more economic than presented by the expression (1).

$$T = \sum_{i=1}^{K}\left[V_t + \sum_{t=1}^{Z_i}\sum_{j=1}^{N} V_{ij}\right] \quad (3)$$

On the basis of comparison of expressions (1) and (3) we can construct a mapping of the sets being a description of the database structure. So, for the formula (1) corresponds to the set of (4) and formula (3) corresponds to the set of (5).

$$\left\{\left\{T_{i,}, V_j\right\}\right\} \quad (4)$$

$$\left\{T_i, \left\{V_j\right\}\right\}, \quad (5)$$

where

Ti - timestamp;

Vj - the value of the tag.

When creating the system of tests of important factors that you need to evaluate the DBMS for storage of large amounts of data, are: the complexity of the structure, the volume occupied by the data and the speed of receiving data.

III. Using NOSQL Storage of Test Data

An alternative approach is the use of noSQL [6]. This approach will build a system capable of adapting to the increasing amounts of data and effectively handle them. NoSQL solutions provide much higher data throughput than traditional DBMS.

Storage time series are most adapted store known as «key-value». Such databases are the easiest way to store multiple values associated with the same timestamp tag. Often such databases are included with the software for measuring or checking or integrated into the SCADA system [8]. Examples include the database Citadel, which is part of the graphical programming platform National Instruments - LabView. To improve the characteristics of the storage of historical data in databases is usually possible to buffering and compression.

A database that is organized as a «key-value», allow to implement storage of large amount of historical data, providing high speed write and retrieve data. However, if you change the structure of the stored data will be impossible to make a selection of the parameters corresponding to one physical parameter.

The optimal solution is to use a document-oriented database. Data model such storage allows you to merge multiple key-value pairs in an abstraction called an «document». Documents can be nested structure and form collections. Collections can contain other collections. It is this capability that allows to model the relationship is one-to-many.

Such use document-oriented database unusual, because these data are not the document.

Document-oriented architecture DBMS allows you to use a hierarchy of nested and move to a higher level of General information about the entity . Such a data model will eventually permit from the base type of the provision of a set of values (5) go to the form (6)

$$\left\{T_0, \left\{T_m, \left\{T_s, \left\{T_i, V_i\right\}\right\}\right\}\right\}, \quad (6)$$

where

T_0 - the timestamp of the start of the test;

Tm - timestamp minutes;

Ts timestamp seconds;

Ti is the timestamp of the smallest interval of time;

Vi - the value of the tag.

The use of document-oriented database allows identifying a common data structure in the form of fuzzy structures -- structures that can be dynamically expanded. The data is organized as a tree that allows efficient indexing by navigating to the key elements of the tree. This approach is effective because the majority of requests contain the time as the main parameter of search. Thus, a custom look at the data will coincide with the hierarchical structure of the applied model of data in most of the queries. Structure, which is optimal for indexing time, also represents a structure of nested objects, which fully corresponds to the real logic of the objects of the test.

This approach to the implementation of the database structure, there is virtually no duplication of data, and data integrity is ensured by a hierarchical structure.

The proposed flexible structure of the database allows to expand the horizontal functionality of the database (the add parameter types, event types, etc), and also will allow to store the compressed ranks, and automatically, transparently to carry their processing.

For the implementation of such structure will use the database MongoDB [8][9].

IV. The Structure of the Data in Practice

In the end, the overall structure of the developed document-oriented storage will look like the following:

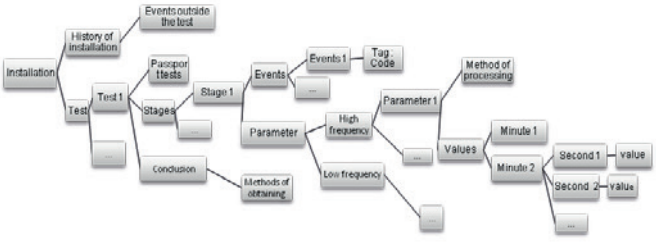

Fig. 1. Example data structures.

Such a structure allows to store:

- the results of observation over parameters of the process of testing;
- meta-information about testing;
- information about the operation of the test items during its life cycle.

Storage is not historical information in a document-oriented database can be organized as a projection of the structure RDBMS containing this information. This

structure will contain all the necessary information, ways of transformation parameters, directly series, information on testing. Storage same time series in the structure of the document requires additional study.

To improve data search and to conserve space is proposed to use the hierarchy of the document and represent time series in the form of a tree. There is a particular task of determining the optimal nesting branches of the tree. In practice, it is necessary to find a compromise between the memory size and speed. For this you must define the target function f and solve optimization problem

$$f(B,C) = k_m f_m (B,C) + k_v f_v (B,C) \qquad (6)$$

$$f(B,C) \to min,$$

where

B - nesting tree branches;

C - the number of elements in the tree;

f_m - function determining the memory footprint;

f_v function determining the speed of access to data;

k_m and f_v – weights.

REFERENCES

[1] S. Schubring and I. Munoz, "Field Performance Testing from an Operators Point of View," Gas Turbine User Symposium 2005. -Las Vegas, Nevada, 2005.

[2] D. Popov and I. Shmidt, "Development of the functional structure of the software complex tests of gas turbine units with a capacity up to 40 MW," Research and innovation, vol. 6, pp. 264–270, 2012.

[3] B. V. Kavalerov, V. P. Kazantsev, I. A. Shmidt, "Simulator and Semi-Nature Testing of Gas-Turbine Power Units Control Systems, Information and Control Systems," St.-Petersburg, Russia, vol 5, pp. 25-31, 2009.

[4] S. Navathe, "Temporal Extensions to the Relational Model and SQL," in A. Tansel, J. Cliord, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass, editors; Temporal Databases: Theory, Design, and Implementation. / S. Navathe, R. Ahmed. – Benjamin/Cummings Publishing Company, 1993. – pp. 92-109.

[5] R. Snodgrass, "The TSQL2 Temporal Query Language," Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061, USA, 1995

[6] G. Vaish, "Getting Started with NoSQL," Packt Publishing, 2013.

[7] D. Bailey and E. Wright, "Practical SCADA for industry," Oxford(GB): Elsevier, 2003. -304 p.

[8] K. Banker, "MongoDB in Action," Manning Publications, 2011. p. 375.

[9] K. Chodorow, "Scaling MongoDB," O'Reilly Media, 2011. p. 62.