DE GRUYTER OPEN
RIGA TECHNICAL UNIVERSITY 1862
Information Technology and Management Science
ISSN 2255-9094 (online)
ISSN 2255-9086 (print)
December 2017, vol. 20, pp. 109–115
doi: 10.1515/itms-2017-0019
https://www.degruyter.com/view/j/itms

Supporting Problem Solving Process of Expert System Architecture in Database Administration

Andrejs Kauliņš
Riga Technical University, Latvia

***Abstract* – The present article describes the supporting problem solving process of expert system architecture in database administration. The proposed expert system produces solutions using input data from users, also recommends a better solution according to the defined task, restrictions and given optimisation goal. To generate solutions, an expert system uses the combined method to produce a solution. The present article also describes a problem domain, where an expert system could be used. In the implementation section, system design and implementation are also introduced. Conclusion section comprises further research and development steps.**

***Keywords* – Expert system architecture, decision support, ontology.**

I. Introduction

The problem solving process in database administration is a complex process. Database administrators direct or perform all activities to maintain a successful database environment. Many regulations indicate implementation of data security, high availability and strong database requirements. The present article describes expert system architecture to help users take right decisions in the database administration field. The system generates solutions based on input data from users. A user shell provides task definition, infrastructure parameters, restrictions and optimisation goal. The algorithm of generating solutions is constructed based on ontology and genetic algorithms. Users get the best solution according to the provided goal and estimates for implementation time and costs for this solution. The system uses different components – user interface, task handler, solution generator and optimiser, task estimator, relational database, knowledge database and report subsystem. The system is designed to support Oracle Database administration process, but could be adapted to support a decision-making process in other fields.

II. Problem Domain

Database administration includes different aspects and DBA administrators have many duties:

- To identify database requirements by interviewing customers, analysing department applications, programming, and operations, as well as evaluating existing systems and designing the proposed systems;
- To recommend solutions by defining database physical structure and functional capabilities, database security, data back-up, and recovery specifications;
- To install the revised or new systems by proposing specifications and flowcharts, recommending optimum access techniques, coordinating installation requirements;
- To maintain database performance by calculating optimum values for database parameters, implementing new releases, completing maintenance requirements, evaluating computer operating systems and hardware products.
- To prepare users by conducting training, providing information, resolving problems;
- To provide information by answering questions and requests;
- To support database functions by designing and coding utilities;
- To maintain quality service by establishing and enforcing organisation standards;
- To maintain professional and technical knowledge by attending educational workshops, reviewing professional publications, establishing personal networks, benchmarking state-of-the-art practices, participating in professional societies;
- To contribute to team effort by accomplishing the related results as needed.

New strict regulations of data security – General Data Protection Regulation (GDPR) [2] –, force database administrators to encrypt and protect data in a database. How to solve and implement this requirement in practice? Often online encryption is impossible and we have conflicts between implementing regulations and business system availability. Even more, the encryption task can be implemented in different ways – using SQL, PL/SQL packages, data export/import, etc. Which is the right solution in our environment with given business requirements? The described expert system helps users to generate solutions and choose the best one based on the estimates and goal defined.

III. Expert System Architecture

The main goal of the expert system is to generate potential solutions to a given problem, estimate them and recommend an appropriate resolution according to a given optimisation rule. To perform these tasks, a system will accept input data and produce output results as a final report (Fig. 1).

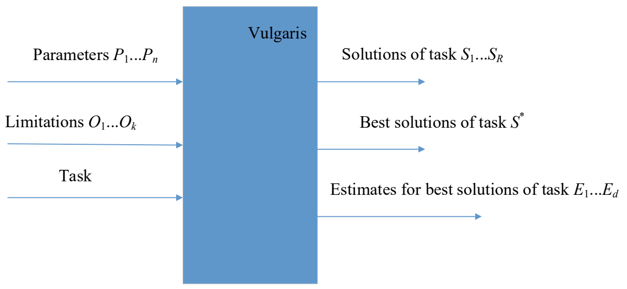
Vulgaris
Parameters $P_1 \ldots P_n$
Limitations $O_1 \ldots O_k$
Task
Solutions of task $S_1 \ldots S_R$
Best solutions of task S^*
Estimates for best solutions of task $E_1 \ldots E_d$

Fig. 1. Input/output data.

A. Input Data

1. Problem domain infrastructure description:
   1.1. Server type, OS platform version;
   1.2. Number of CPU installed;
   1.3. RAM size installed on the server;
   1.4. Database version;
   1.5. SGA/PGA size parameters;
   1.6. Storage description (number of disks, RAID arrays if used, disk parameters – rpm, volume).

All parameters are defined as single value parameters P_1, ..., P_n, as shown in Fig. 1 (Table I).

TABLE I
PARAMETERS FOR INFRASTRUCTURE DESCRIPTION

Class/Subclass name	**Parameter**	**Value**
Business System	SLA	Gold/Silver/Bronze
Server	Model	T5
Database	Manufacturer, version	Oracle, 12.1.0.2.0
Linux	OS version, CPU, RAM	Rhel 7.4, 4,8 Gb
Storage	Model	G1000
Storage controller	Firmware version	4.1.2.3
LUN	Pool name, RAID type	ASMD, RAID1
Disk	Type, rpm, cache	Spined, 5400, 2 Mb

2. Restrictions which apply to solutions. Later we will see an example of restrictions. For example, if a business system should be online all the time, or can be at the downtime state, for a limited amount of time “data should be available”.

3. Task definition. System should generate solutions to a task. “Encrypt Data in LOB segment”, “Compress tablespace”, “Upgrade database” are typical DBA tasks, which should be solved.

4. Goal for optimal solution.

There are two practical goals for the best solution – by implementation time (reducing implementation time) and implementation costs (minimising implementation costs).

B. Output Data

1. Possible list of all solutions (for internal purposes, and if a user requires the whole list with estimates).
2. Solutions that are applicable to our restrictions.
3. Better recommended solution, which fits our optimisation goal.
4. Estimates for each solution (implementation times, costs).

To formulate solutions and recommendations, the system has modules (Fig. 2).

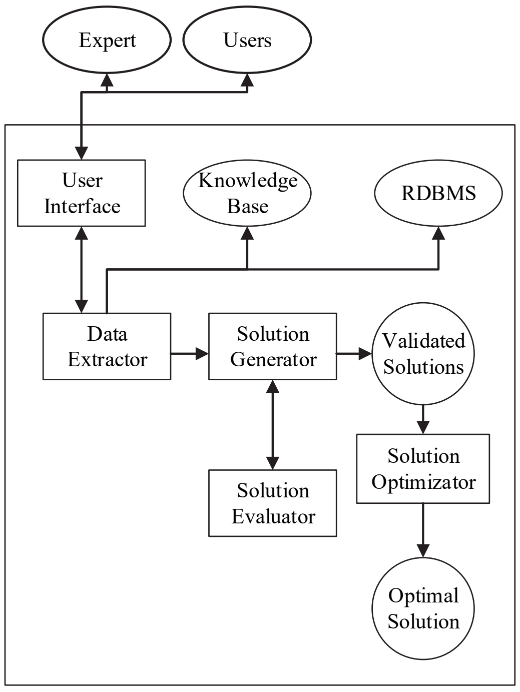

Fig. 2. Expert system architecture.

C. Data Extractor

Data extractor writes all parameters in a relational database in tables and extracts restrictions, task and optimisation goal. Ontology is used to fulfil the rest of the information on the infrastructure. For example, ontology contains information on a relationship of server and memory, storage, and CPU that could be installed in a given type of server. Informing the system that we have server type SPARC T5-2, it must have one or two processors installed in it, and even more – if one processor is installed, it has to have 128 Gb / 256 Gb / 512 Gb of RAM, and if it has two processors installed, then it can have 256 Gb / 512 Gb / 1024 Gb of RAM. All these details about the given type of server are described in ontology (Fig. 3) [7]–[9].

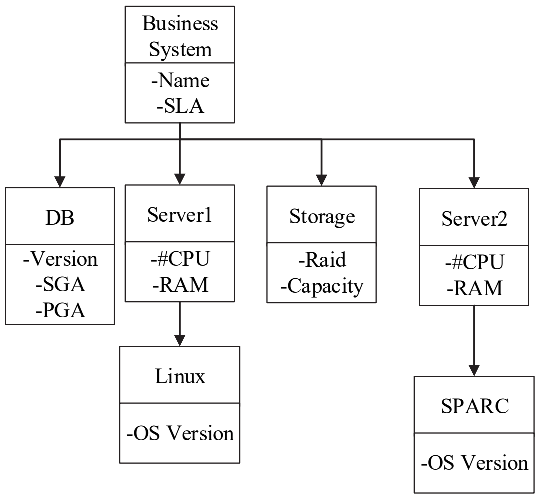

Fig. 3. Infrastructure ontology.

After all input data are fixed, we can generate solutions, for that purpose “Solution Generator” takes control.

D. Solution Generator

Solution generator reads knowledge base, for solutions, which are generated and saved in a similar problem solution process. For a given task, it manipulates with atomic operations, for example, in the task of encrypting data, atomic operations:

- Upgrade a database;
- Add CPU;
- Migrate a database;
- Move a table;
- Redefine a table;
- Export a piece of data;
- Import a piece of data;
- Unplug tablespace;
- Plug tablespace.

Solutions to problems in our domain could be simple, just consisting of one atomic operation: Problem – Solution and complex:

Problem: $Sol_1 \rightarrow Sol_2 \rightarrow \ldots \rightarrow Sol_n$ (Fig. 4).

On the other hand, each atomic operation can take some time for implementation and will require system or data unavailability in each step of implementing a complex solution (Fig. 5).

Each version of database has some features implemented in it. Version 12.2 has Transportable Data Encryption ONLINE feature, which can be used to encrypt LOB segment. All features for a specific version are documented in oracle manual, for example, for Oracle Database 12.2 one can reference Oracle® Database New Features Guide 12c Release 2 [12].

Solutions are dependent on features used, so we need to store information about these features in the knowledge base (Table II).

TABLE II
FEATURE AVAILABILITY IN DB VERSION

No.	Feature	11.1	11.2	12.1	12.2
1	Transport-able tablespace	Yes	Yes	Yes	Yes
2	Transparent data encryption (online)	N/A	N/A	N/A	Yes
3	LOB operations	Basic	Basic	Advanced	Advanced

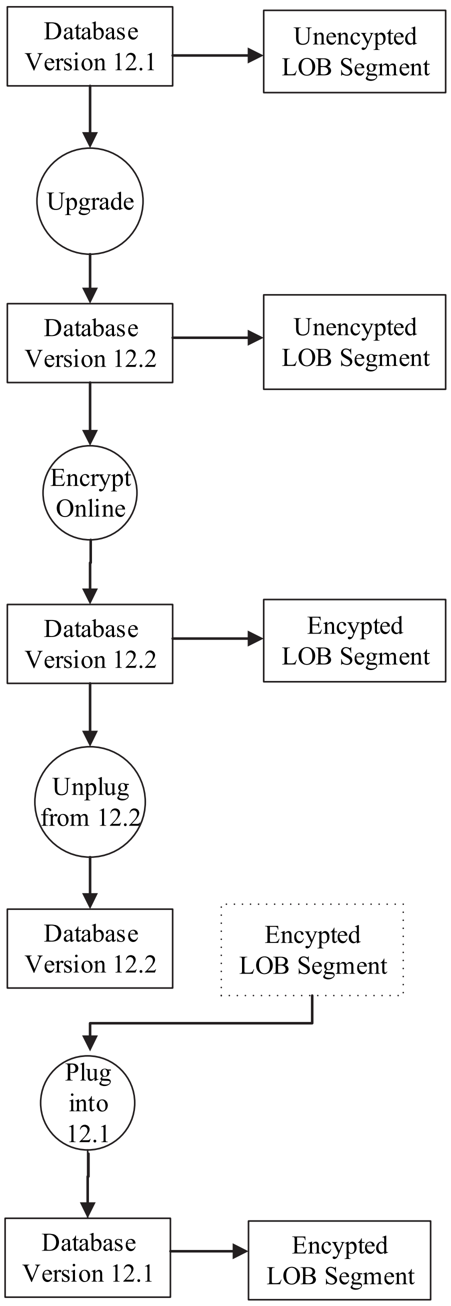

Fig. 4. Complex solution to a simple problem.

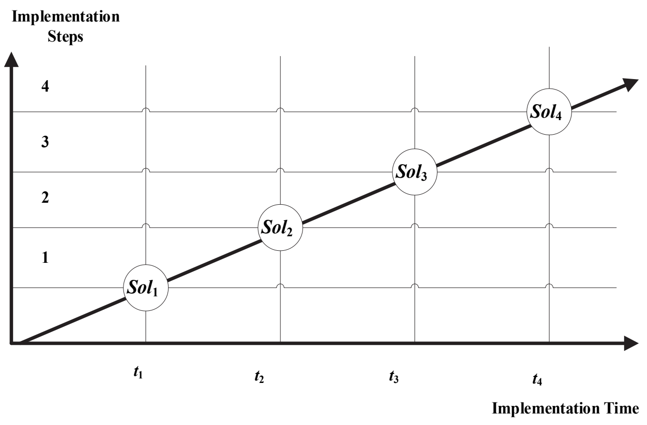

Fig. 5. Complex solution implementation steps.

E. Solution Evaluator

To get evaluations and validate solutions, before implementing any change in production, these solutions have to be implemented in the test environment. That could be a difficult task, in terms of costs and time. Test environment should be the same as the production one; data should be the same as in production. All solutions have to be implemented in the test environment to validate the process of implementation and fix the time, which all operations take in the test environment. Another issue is to validate the process for errors. Database upgrade scripts could fail due to errors, so it is necessary to fix them, before implementing solutions in the production environment. To evaluate atomic operation execution time, Markov chain [1] is used to model the test environment and estimate results.

After obtaining validated and estimated solutions, it is possible to choose the best one as a final recommended solution.

F. Solution Optimiser

Optimiser estimates validated solutions and chooses the best one, which fits by the optimisation goal given to the system.

Finally, a report is generated to a user with a recommended solution.

G. Knowledge Base

Knowledge base contains different ontologies:

- Ontology of infrastructure;
- Ontology of database features;
- Ontology of PL/SQL statement semantics.

Knowledge base is based on triples; an engine is a property graph of Oracle Database (Fig. 9). Knowledge base is populated by an expert of database administration domain. All ontologies are built by oracle commands described in [10]. After each generated solution, the main steps are saved into a relational database for future re-usage. Input data are compared with history and if the entered data are the same, the existing solutions are reused.

IV. Simple Example

Let us consider a simple task for database administrators. To implement European Union regulations, we need to encrypt documents, which are stored in oracle 12.1 version database. Documents (pdf, pictures, and scanned documents) are stored in Large Objects (LOB) as separate segments from tables. Business system is critical and should be available 24 × 7 (Gold support agreement level by ITIL). Maintenance window is defined on Sunday from 8:00 am to 12:00 (noon), so allowable downtime is 4 hours. Database is located on SPARC T5-2 server with 2 CPU and 1TB RAM memory is installed. Data files are located on external storage HITACHI USP, for data protection RAID1 is used, each logical unit (LUN) contains 20 disks (256 GB × 15000 rpm); total volume is 5 GB. Our task is to "encrypt 1GB of data placed in LOB segment". We have a restriction – database downtime is 4 hours. There are a plenty of solutions to this task:

- To move data with a single SQL command (alter table move to encrypted tablespace);
- To switch parallelism on the table, so the encryption process will proceed in parallel, at a degree defined by a database administrator.
- If CPU is bottleneck, we can add additional CPU cores to a virtual server (LDOM or local zone), where a database is located.
- If Oracle version supports DBMS_REDEFINITION feature, we can use table redefinition online to complete the task [6].
- In Oracle 12.2 version, we can use online encryption feature without downtime.
- Unencrypted data can be exported to a file system, and then imported in the encrypted tablespace. This operation should be made, when the system is unavailable to users (new transactions due to the import/export process).

How to choose the best solution? If we have an older version of Oracle Database, we cannot implement encryption without downtime (technical issues). Even in such a case, it is possible to upgrade the database to a newer version, encrypt data, then downgrade version to the original one.

In any case, a solution should be tested in the test environment due to the following reasons:

- Time estimation in each step;
- Quality check (software errors can happen, administrators should fix them before implementing a solution in the production environment).

As you can see, implementing a solution for data encryption is easy to define from the business side, but hard to implement technically.

When all the details are provided for an expert system, a solution generator takes control and tries to generate solutions for a given task and defined goal. Solution estimator eliminates solutions which are not valid or compatible with our restrictions. In our case, a simple command "alter table emp.scott_lob move tablespace tbs_encrypted" executes as one process, and takes on a given server for 10 hours with tablespace offline, so data in the table are not available, and this solution is not valid. To take more powerful CPU to reduce time, we have to add additional cores, which are additional costs for software licenses. If we have more powerful servers (for example, T7, which is 30 % faster), we could migrate the database to them, and this solution would be valid. Then it took 3.3 hours. But before encrypting data, we should test and migrate the database from T5 server to T7. It would take another downtime.

V. System Implementation

For conceptual system design and modelling, Petri net was used. System design is an iterative process, for the existing system Petri net model is designed, then this model is modified with details of the modelled system, changes are made in architecture (Fig. 6).

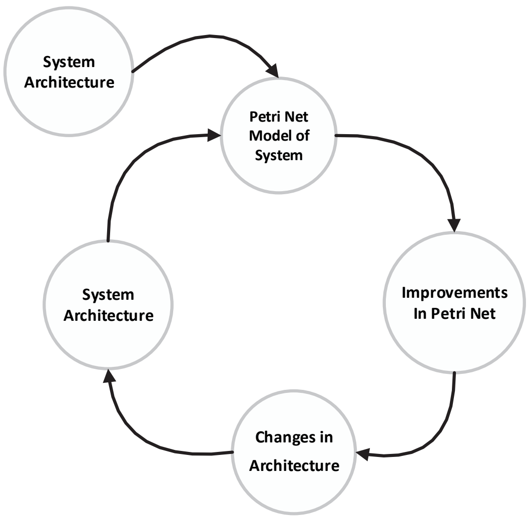

Fig. 6. System design using Petri net.

A Petri net consists of places, transitions, and arcs. Arcs run from a place to a transition or vice versa, never between places or between transitions. The places from which an arc runs to a transition are called the input places of the transition; the places to which arcs run from a transition are called the output places of the transition (Fig. 7).

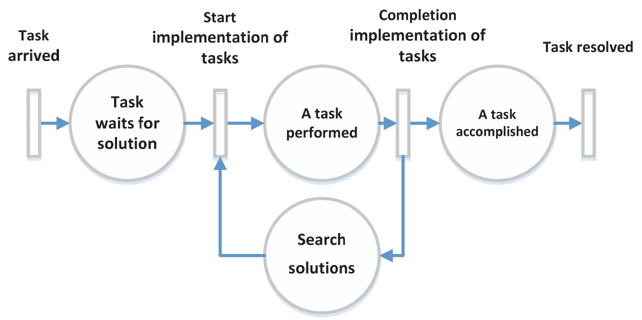

Fig. 7. Petri net model of the designed system.

On next iteration we add details to Petri net, and then make changes in system architecture (Fig. 8).

Graphically, places in a Petri net may contain a discrete number of marks called tokens. Any distribution of tokens over the places will represent a configuration of the net called a marking. In an abstract sense, relating to a Petri net diagram, a transition of a Petri net may fire if it is enabled, i.e., there are sufficient tokens in all of its input places; when the transition fires, it consumes the required input tokens, and creates tokens in its output places. A firing is atomic, i.e., a single non-interruptible step.

Unless an execution policy is defined (Table III), the execution of Petri nets is nondeterministic: when multiple transitions are enabled at the same time, any of them may fire.

TABLE III
Execution Policies for Petri Net (Fig. 7)

Event	**Precondition policy**	**Postcondition policy**
Task arrived	–	A task performed
Start implementation of task	Task waits for a solution, a task performed	A task performed
Completion implementation of task	A task performed	Task waits for solution, a task accomplished
Task resolved	A task accomplished	–

Let us consider an example of firing of transition (Fig. 8), the place p_1 is an input place of transition t; whereas, the place p_2 is an output place to the same transition. Let PN_0 be a Petri Net with a marking configured M_0 and PN_1 be a Petri Net with a marking configured M_1. The configuration of PN_0 allows for transition t through the property that all input places have a sufficient number of tokens (shown in the figures as dots) "equal to or greater" than the multiplicities on their respective arcs to t. Once and only once a transition is enabled, the transition will fire. In this example, the firing of transition t generates a map that has the marking configured M_1 in the image of M_0 and results in Petri Net PN_1, seen in Fig. 8. In the diagram, the firing rule for a transition can be characterised by subtracting a number of tokens from its input places equal to the multiplicity of the respective input arcs and accumulating a new number of tokens at the output places equal to the multiplicity of the respective output arcs.

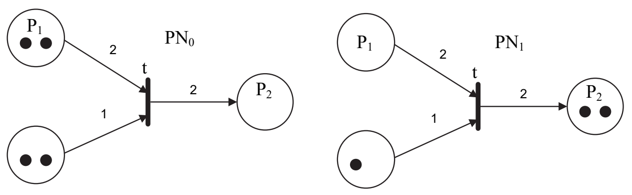

Fig. 8. Petri net firing transition example.

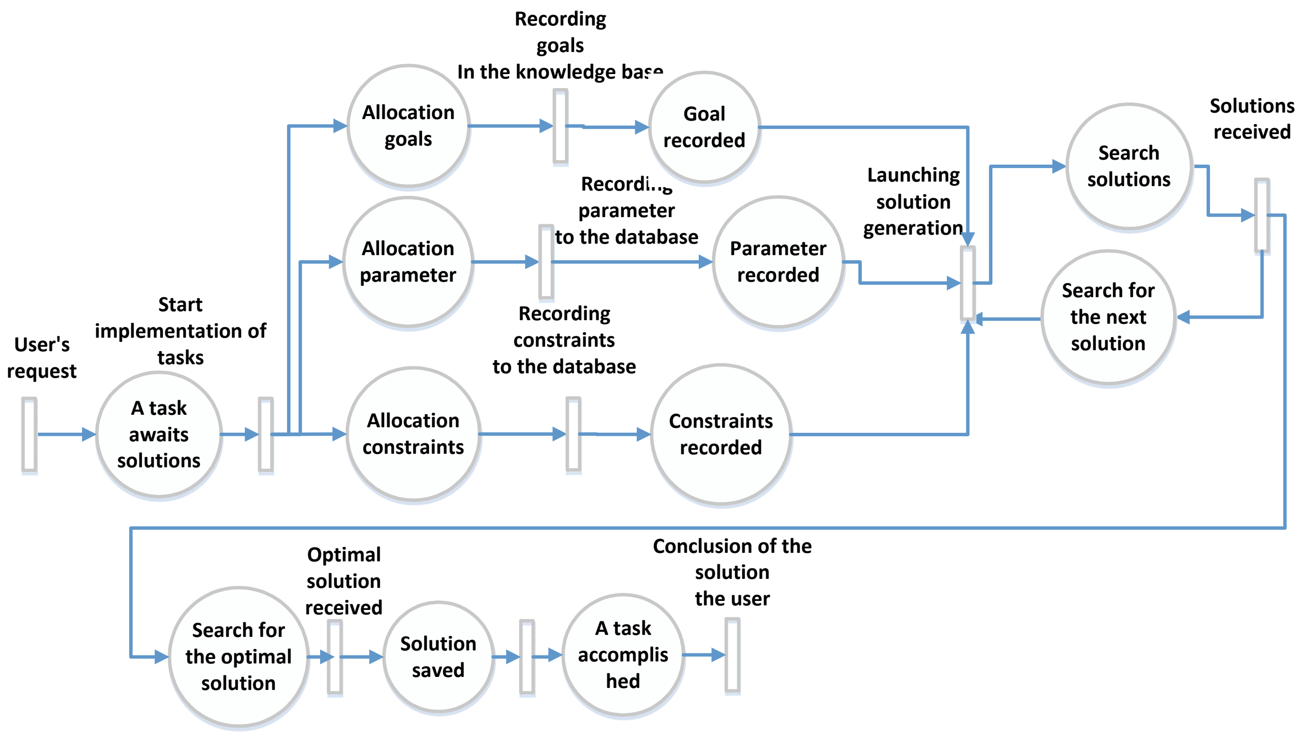
Recording goals In the knowledge base
Allocation goals
Goal recorded
Recording parameter to the database
Allocation parameter
Parameter recorded
Launching solution generation
Search solutions
Solutions received
Search for the next solution
User's request
Start implementation of tasks
A task awaits solutions
Recording constraints to the database
Allocation constraints
Constraints recorded
Optimal solution received
Conclusion of the solution the user
Search for the optimal solution
Solution saved
A task accomplis hed

Fig. 9. Petri net model of the designed system in next iteration.

Since firing is nondeterministic, and multiple tokens may be present anywhere in the net (even in the same place), Petri nets are well suited for modelling the concurrent behaviour of distributed systems [3], [4].

Expert System Prototype will be developed and integrated in Oracle Database 12c. User interface will be designed in Oracle Application Express (APEX 5.1) [5].

For algorithms to generate and optimise solutions, Java Run Time engine built in the database and code in the Java language will be used [13].

Oracle Database provides support for developing, storing, and deploying Java applications. Manual [13] introduces the Java language to Oracle PL/SQL developers, who are accustomed to developing server-side applications that are integrated with SQL data. You can develop server-side Java applications that take advantage of the scalability and performance of Oracle Database.

To store ontology of infrastructure, a property graph of Oracle Database was chosen [10], [11].

A property graph consists of a set of objects or vertices, and a set of arrows or edges connecting the objects. Vertices and edges can have multiple properties, which are represented as key-value pairs.

Each vertex has a unique identifier and can have:

- A set of outgoing edges;
- A set of incoming edges;
- A collection of properties.

Each edge has a unique identifier and can have:

- An outgoing vertex;
- An incoming vertex;
- A text label that describes the relationship between the two vertices;
- A collection of properties.

Figure 10 illustrates a very simple property graph with seven vertices and six edges. The vertices have identifiers "Business system", "Database", "Server1", etc. Vertices have properties "Name", "Model", etc. The edge is from the outgoing vertex "Business system" to the incoming vertex "Database". The edges have a text label "Consist of" and a property type identifying the type of relationship between vertices (Fig. 10).

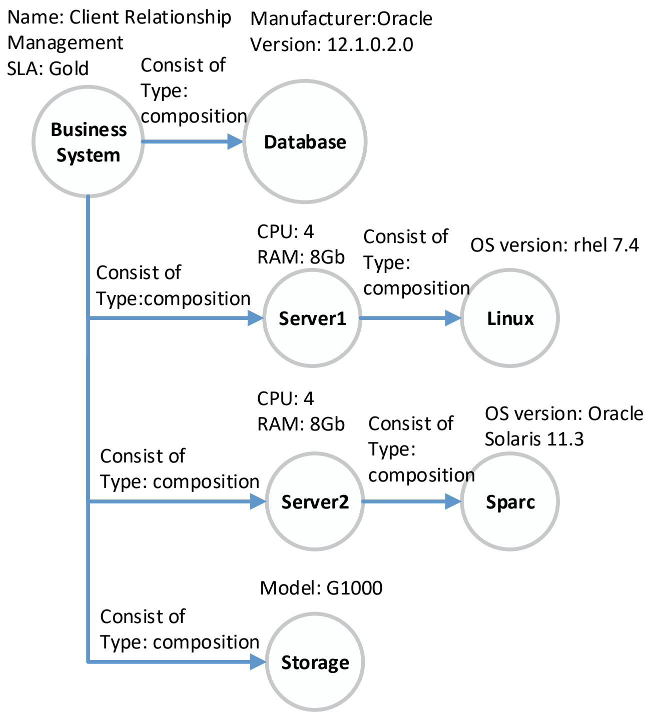
Name: Client Relationship Management
SLA: Gold
Manufacturer:Oracle
Version: 12.1.0.2.0
Consist of Type: composition
Business System
Database
CPU: 4
RAM: 8Gb
Consist of Type: composition
OS version: rhel 7.4
Consist of Type:composition
Server1
Linux
CPU: 4
RAM: 8Gb
Consist of Type: composition
OS version: Oracle Solaris 11.3
Consist of Type: composition
Server2
Sparc
Model: G1000
Consist of Type: composition
Storage

Fig. 10. Ontology of infrastructure of Oracle Database property graph.

VI. Conclusion

The present article has introduced the supporting problem solving process of expert system architecture in database administration. Domain problem has been considered in case of system application. The main architecture has been described and development methodology provided. The present article has not considered an algorithm of solution generation. The method of solution generation is too sophisticated and is outside of scope of this article. Further research will be dedicated to this topic. Some methods are studied in this area:

- Genetic algorithms [14];
- Neural Networks, Deep Learning [15], [16];
- Reinforcement Learning [17].

The considered system has some benefits:

- Users can save time used to search an appropriate certified solution to the given problem;
- System can calculate estimates for solution implementation steps, which reduces testing and saves total solution implementation time.

References

[1] T. Hemdi, *Vvedenie v issledovanie operacij*, 6-e izdanie. Moskva: Izdateljskij dom, 2001.

[2] "The protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing", Regulation (EU) 2016/679 of the european parliament and of the council of 27 April 2016 on Directive 95/46/EC (General Data Protection Regulation).

[3] V. Marahovskij, L.Rozenblum, A. Jakovlev, *Modelirovanie paralleljnyh processov. Seti Petri, Kurs dlja sistemnyh arhitectorov, programmistov, sistemnyh analitikov, projektirovschikov sloznyh sistem upravlenija*, Sankt-Peterburg: Professionaljnaja literatura, IT-Podgotovka, 400 s., 2014.

[4] Dj. Piterson, *Teorija setej Petri i modelirovanie sistem*. Moskva: Mir, 264 s., 1984.

[5] A. Geller, B. Spendolini, *Oracle Application Express: Build Powerful Data-Centric Web Apps with APEX*. Oracle Press, 2017.

[6] J. Greenberg, *Oracle Database Object-Relational Developer's Guide 12c Release 1*. Oracle corp., 2014.

[7] A. Kaulins and A. Borisovs, "Building Ontology from Relational Database", *Information Technology and Management Science*, vol. 17, pp. 45–49, 2014. https://doi.org/10.1515/itms-2014-0006

[8] S. Staab and R. Studer, "Handbook on Ontologies", International Handbooks on Information Systems, Springer Science and Business Media, 2013.

[9] B. Bennett, C. Fellbaum, "Formal Ontology in Information Systems", *Proceedings of the Fourth International Conference FOIS 2006*. IOS Press, 2006.

[10] C. Murray, *Oracle Spatial and Graph Property Graph Developer's Guide, Release 12.2.*, 243 p., 2017.

[11] "Oracle Spatial and Graph Property Graph Java API Reference," 2017. [Online]. Aviable: http://docs.oracle.com/database/122/SPGJV/toc.htm

[12] Oracle® Database New Features Guide 12c Release 2 (12.2), E49697-19, Copyright © 2015, Oracle and/or its affiliates, 2017.

[13] T. Das, "Oracle® Database Java Developer's Guide 12c Release 2 (12.2)", E50047-11, Copyright © Oracle and/or its affiliates, 308 p., 2017.

[14] M. Gen and R. Cheng, *Genetic algorithms and engineering design*. John Wiley & Sons, 411 p., 1997.

[15] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford university press, 2005.

[16] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016.

[17] C. Szepesvari, *Algorithms for Reinforcement Learning*. Morgan and Claypool, 2010.

Andrejs Kauliņš received the *B. sc. ing.* and *M. sc. ing.* degrees in 1993 and 2002 from Riga Technical University and the University of Latvia, respectively. Since 2013 he has been studying at Riga Technical University to obtain a Doctoral degree in Computer Science. Currently the major field of study is complex IT system design based on ontologies.
E-mail: andrejs.kaulins@gmail.com
Phone number: +371 26737122