DE GRUYTER OPEN
1862
RIGA TECHNICAL UNIVERSITY
Applied Computer Systems
ISSN 2255-8691 (online)
ISSN 2255-8683 (print)
May 2018, vol. 23, no. 1, pp. 28–36
doi: 10.2478/acss-2018-0004
https://www.degruyter.com/view/j/acss

Object-Relational Database Structure Model and Structure Optimisation

Ainārs Auziņš[1*], Jānis Eiduks[2], Alīna Vasiļevska[3], Reinis Dzenis[4]
[1–4] *Riga Technical University, Riga, Latvia*

Abstract **– During object-relational database physical structure design, problems are caused by three factors: ambiguity of transformations of conceptual model, multiplicity of quality assessment criteria, and a lack of constructive model. In the present study a constructive hierarchical model of physical database structure has been developed. Implementations are used in XML, SQL and Java languages. Multi-criterial structure optimisation method has also been developed. Structure variation space is generated using transformation rule database. Prototype has been implemented within the framework of the research.**

Keywords **– Multi-criterial optimisation, object-relational database, Pareto solution, structure optimisation.**

I. Introduction

Object-relational database is a hybrid solution. It combines the possibilities of a relational database with the most important technological solutions of an object database. The most comprehensive theoretical investment in the technology of the database was the development of the theoretical basis – relational algebra – for the relational database [1]. The following models of the database (including, the object database) were only engineering solutions. Nowadays, the systems of the relational database are still the most popular systems in the development of information systems due to the fact that they are simple and their designing technology by the use of CASE (Computer Aided System Engineering) is well developed [2]. Unfortunately, it also has a lot of deficiencies. The author of the relational database concept, E. F. Codd, also made the conclusion that the structure of its storage and the language of obtaining data from it, SQL, had to be supplemented with more complicated table constructions using domain mechanism (unfortunately, it was not done) [3].

The concept of object-relational database was developed in technical projects. POSTGRES and its branch Illustra can be mentioned as the most significant of them. An outstanding database practitioner and scientist M. R. Stonebreaker was the leader and initiator of both projects. When developing various information systems and database systems, he was one of the first to understand how to improve the relational database, including the use of objects in it. Many other database technology specialists were also analysing the possibilities of improving relational database and the trends in the language of programming (object-oriented programming). Conclusions were formulated in two manifests [4], [5]. Common idea was to develop a hybrid system combining the technologies of relational and database systems. On the basis of their proposals, there was standard SQL:1999 (SQL3) developed, which defined the basic structures of object-relational database. Later, additional adjustments were included in standards SQL:2003 and SQL:2006.

Companies developing the largest database systems such as Oracle, IBM and Microsoft have updated their relational database systems, including object-relational technology. It allows developing databases for very complicated objects and ensuring efficient retrieval of data by the use of extended SQL language. Why object-relational database systems are not dominant in such a case? It is due to undeveloped theoretical basis. Object-relational database algebra, which may solve this problem, is being gradually developed only for some years [6].

During the construction of object-relational database, problems are caused by three factors: ambiguity of transformations of conceptual model, multiplicity of criteria in quality assessment, and a lack of constructive model. There are various engineering developments and recommendations that allow slightly systemising the process of the construction of object-relational database, but they have not given substantial results. The authors of the present study have attempted to understand the existing problems deeper and propose their solutions.

II. Fundamentals of Object-Relational Database Construction

In object-relational database, relational database table structures are used as the data storage framework. Their qualities are extended for the storage of various types of objects (simple objects, collection type objects, composed objects). The following tables are used:

1) Relational table in the columns of which various type of data objects can be stored;
2) Object table in which only row type objects are stored.

Row type objects have identifiers OI. They are used for addressing of the objects. When performing data retrieval from the table, data can be obtained as a single object and as a tuple in relational algebra. It allows performing an easy exchange of data between relational and object-relational database structures.

By specialising object-relational database technology, a data storage structure for XML and graphic data has also been

[*] Corresponding authors e-mail: ainars.auzins@rtu.lv

developed. Object-relational database type XMLType has been developed for the storage of XML data. By using it:

1) XML data table (table with XMLType row type objects) can be developed;
2) XMLType objects can be also included in relational table columns.

In the most popular database systems such as Oracle, DB2 IBM, PostgreSQL, SQL Server MS, the possibilities of relational database are extended by an object-relational technology. Relational, object and XML tables (further, the developments of the leading database system Oracle are analysed [7]) are used. They are **basic containers**, in which data objects and scalar data are included.

Object-relational database offers wide possibilities to create **data association**.

1. Relational table is used, including data objects in its columns, and consequently ensuring their association.
2. Special construction relational tables have been developed, which apply links "one-to-many" in the case of object collections. These tables are called nested tables (correct name would be "a table with a collection").
3. Object identifiers can be included in other tables as references. They can be used for addressing or indication on necessary objects. It can perform linking of various table objects. The use of object identifiers ensures much faster reactivity of table object connection. The types of the link "one-to-one" (1 : 1) and "one-to-many" (1 : *N*) can be implemented. Object links are one direction links. It is also possible to develop two-direction links. In this case, it is also possible to implement the link "many-to-many", which is not possible in the case of relational database.
4. Object association can be used for the development of composed objects (object hierarchy).
5. For the creation of object inheritance constructions, an object table version with heterogenic objects has been developed. It allows storing in one object table the objects with various attributes and corresponding data processing methods.

Data **processing methods** are defined for objects. Object-relational database technology has mainly been developed due to these methods. There is no sense to group data in objects without having a possibility to process them together. Composed objects are defined for the use of exactly necessary methods. The use of these methods:

1) allows extending the programming possibilities of database server and decreasing the load of computing networks;

2) includes database languages in SQL queries. It allows implementing the slogan for information system development of last 10 years, which is "data processing mainly in database" in a more comprehensive way. This slogan was brought forward by leading database technology development companies Oracle, IBM and SAS. When making complex SQL queries by the use of methods, the optimisation of query fulfilment reactivity is performed in the server. When data processing function is included in applications, the optimisation of reactivity is more problematic.

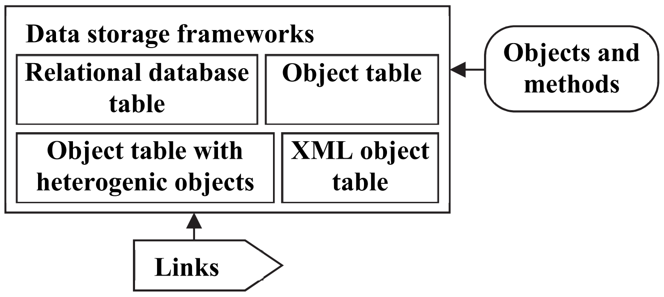

Fig. 1. Data storage structure creation of object-relational database.

Figure 1 shows the creation of the structure of object-relational database. Taking into account the methods that are necessary for making data retrieval queries, the respective object types and subordinate types (including methods) are defined. When necessary, the object type subordination hierarchy for the implementation of inheritance is defined. By the use of various object linking mechanisms, the objects are placed in containers (tables).

Sometimes, an alternative variant is used for the implementation of methods and objects in a database. There is an object view mechanism used. Initially, there is a relational database created. Then, object types and methods are defined. Virtual object table is created with the assistance of object views from the columns of relational table. These objects also have identifiers, and the necessary methods are defined. Certainly the speed of query fulfilment is smaller for such an alternative variant. However, the obviousness of the structure of basic data is often better understandable and there is a good flexibility of the structure for making changes. For performing the analysis and choice of the structure of object-relational database, a constructive model of it is necessary.

III. Related Research

The development of designing and construction methodology of object-relational databases started at the beginning of the 21th century. Initial considerations and studies were published in works of E. Marcos [8], E. Pardede [9], and Wai Yin Mok [10]. The methodology offered by the scholars divided the development of object-relational databases into 3 stages: analysis, designing and constructing in the same way as it is for relational database. In the designing stage, it was recommended to use UML (Unified Modelling Language) class diagram instead of extended EER (Extended Entity Relationship) model. UML class diagram:

1) also includes the indications of the necessary functionality (methods);
2) UML is an extended language that allows for the creation of new stereotypes of existing elements in class diagram.

The constructing stage was divided into two steps:

1) standard or logical construction, which is independent of the management system of the specific database used (orientation towards standards SQL:1999, SQL:2003);
2) specific construction, which is made for a specific selected database management system without including configuration and the reactivity optimisation of SQL queries.

Such a division is useful in cases of database migration but it is difficult and unnecessary for the constructing object-relational databases (the first step is unnecessary). Excluding of performance analysis from specific constructing is not a successful solution because it leaves a considerable influence on the structure of the database [11].

SQL:1999, SQL:2003 standard database structures as well as the structures of the databases developed by company Oracle have been analysed in studies. New stereotypes are defined for UML class diagram, which would correspond more precisely to possible structures of object-relational database. It is frequently noted in publications that object-relational database is suitable for the creation of spatial data, scientific research data and technical equipment databases [12], [13]. Due to this reason, two approaches for the construction of object-relational databases have been proposed:

1) **General approach**. In this case, the transformation of conceptual model is discussed:
   a) SQL:1999 or SQL:2003 standard data storage in structures [12], [13];
   b) specific object-relational database systems in data structures (in most cases, in the structures of company Oracle database systems) have been selected. The second option is better because each object-relational database system has its own specific structure and their implementation. The transformation from conceptual model to the structures of standard SQL:1999 or SQL:2003, and after that to the structures of selected database system can cause inaccuracy in the implementation of the directions of conceptual model [14], [15];
2) **Domain–specific approach**. In this case, it is possible to use already existing analogue structures that have been developed for the system of the database as well as more detailed information about the area of use.

Corresponding software constructing tools are necessary for constructing object-relational database. There are tens of CASE (Computer Aided System Engineering) tools that can be used for constructing relational databases. This process has been formalised in a detailed way for them. Object-relational databases do not have such tools. However, in some projects this issue is being solved. These are the possible solutions:

1. Extension of tool ArgoUML. UML class diagram is suitable for obtaining the structure of object-relational database. Two program modules have been developed performing the transformation of conceptual model in SQL:2003 standard code and in the Oracle11g data defining language [16].
2. Tool O-ODM, which transforms object-oriented use objects in the structures of object-relational databases [17].
3. UML class diagrams and corresponding SQL:2003 standard constructions are analysed in studies [18].

The analysed studies do not offer any proposals for solving the matter regarding the comparison of alternative structures of object-relational databases.

IV. Variety of Object-Relational Database Structures and Structure Model

Object-relational database structure S is formed by basic containers K_i (i = 1, ..., k), data objects O_j (j = 1, ..., z), object methods M_r (r = 1, ..., m) and object linking mechanisms R_t (t = 1, ..., h):

$$S = S(K, O, M, R), \qquad (1)$$

where $K = (K_1, ..., K_k)$, $O = (O_1, ..., O_z)$, $M = (M_1, ..., M_m)$, $R = (R_1, ..., R_h)$.

Using the variation of these structure elements, it is possible to obtain various object-relational database solutions with different qualities. When analysing problem environment and creating a conceptual model of the data, in most cases linking of two attribute clusters is performed by using the link one-to-many. This situation is shown in Fig. 2 by using UML class diagram. The necessary data processing methods are a_1, a_2, a_3, a_4 – attributes and M_1, M_2. The authors of the present paper will use this model to demonstrate the alternative variants of the structure implementation in object-relational database.

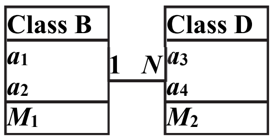

Fig. 2. UML diagram of two classes with a one-to-many association.

Firstly, special construction relational table implementing associations in the case of object collections will be used as basic container K_1. B class objects O_1 will be included in the first column of the table and collection type objects O_3 including D class object O_2 collections – in the second column of the table. Figure 3 shows one alternative of a structure using relational table. Objects O_2 included in the collection also have object identifiers OI. An indication on them can also be made from other tables. The structure of the table implements a one-to-many link. To place the included object Q_2 cluster instead of one row collection object Q_3, the object-relational database function Table() is used. Table() is one of many object extensions in SQL language.

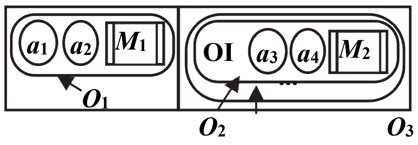

Fig. 3. Relation table with ordinary and collection type objects.

An important advantage of the use of objects in database is the possibility of increasing the query reactivity by the use of object addressing mechanism – object identifiers OI and reference R. By using of object-relational database object associations, the structure of conceptual model shown in Fig. 2 can be implemented as three mutually linked tables (Fig. 4). The first table A can be both relational table and the table with row type objects or object columns. It includes object references R indicating the respective collection type object in object table AB. Such an object has recorded references R on all associable objects of table B. One table A object can be associated with many table B objects. Figure 4 shows an example of one-directional associations. To implement two-directional associations, A and B tables must have row type objects and references. There must be another association table used, which is analogue to table AB.

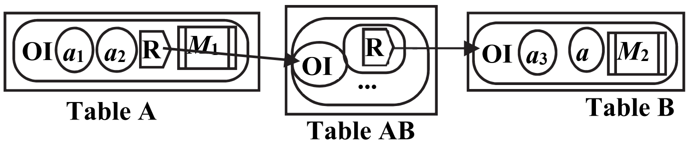

Fig. 4. Use of object addressing mechanism.

Object-relational databases are developed for the storage and efficient retrieval of data in case of complex objects. SDO_GEOMETRY object type for the storage of graphical data can be mentioned as a typical example. Initially, database systems Oracle, PostgreSQL, IBM DB2 used the database structure consisting of 5 relational tables for the storage of graphical data. When using the possibilities of object-relational databases, a complex hierarchical three level SDO_GEOMETRY object type was developed for the storage of graphical data. It is used for defining of relational table column type. Tens of methods have been developed for SDO_GEOMETRY object type allowing for a simple creation of all necessary queries, i.e., both topology and scalar attribute.

The realisation in the form of a complex object (Fig. 5) can be created for the conceptual scheme in Fig. 2. The object, whose components is the first type object O_1 and its corresponding collection O_3 of the second type objects O_2 is included in object table. This structure has an additional advantage, i.e., methods can also be defined for common complex object O_4 that creates the composition of the examined elements. Object O_4 has an object identifier; due to this reason, it is possible to make indications on it from other objects or relational tables.

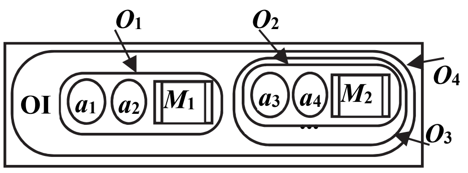

Fig. 5. Composite row type object.

One of the criteria of database quality is the simplicity of the perception of database data semantics. Many designers of object-relational database structures, including one of the leading specialists of company Oracle Thomas Kyt, noted that they prefer simple relational structures for data storage [19]. Object views are used to implement the necessary data processing methods. The views of (virtual tables) in relational database systems only simplify data retrieval orders. Object views allow making new virtually complex object structures and necessary object methods in object-relational database systems. Figure 6 shows a physical structure of object-relational database, which ensures common processing of several type object data in the method.

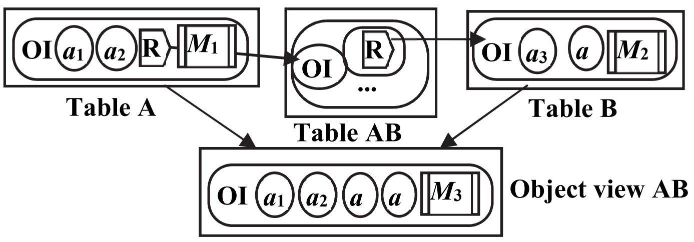

Fig. 6. Use of object view.

Object-relational databases are oriented towards the use with complicated data structure and method subordination. The principle of specialisation is applied to understand the behaviour of these objects and to model it. Specialisation separates object specific aspects and includes them in one or several subordinate types of objects. Object subordination hierarchy is created: the whole – a part – a part of the part and etc. In the database, it can be various and complicated because among data there are much more variants of subordination possible than among programs (Fig. 7).

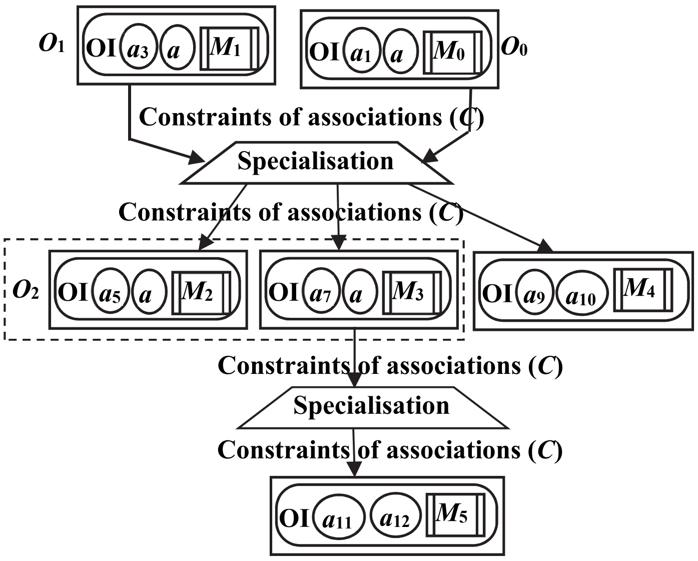

Fig. 7. Specialisation structure of database objects.

The most important types of specialisation used in a database are the following:

1) Total specialisation – every base type object O_1 has at least one subordinate type object O_2 ($O_1 \rightarrow 1 \ldots nO_2$);
2) Partial specialisation – not every base type object O_1 has a corresponding subordinate type object O_2 ($O_1 \rightarrow 0 \ldots nO_2$);
3) Disjoint inheritance – every base type object can have no more than one subordinate type object O_2 ($O_1 \rightarrow 0 \ldots 1O_2$);
4) In mutual exclusion specialisation, subordinate type object O_2 cannot belong at the same time to several base types of objects O_1 and O_0 ($O_1 \rightarrow O_2$ xor $O_0 \rightarrow O_2$); besides, the base type of object O_1 can have no one subordinate type object O_2 ($O_1 \rightarrow 0 \ldots nO_2$);
5) Overlapping specialisation – every base type object O_1 can have several subordinate type objects O_2, O_3 ($O_1 \rightarrow O_2$ and O_3);
6) Partition specialisation is a combination of overlapping and mutual exclusion specialisation – subordinate type objects form mutually excluding groups;
7) Union specialisation – every base type object mandatory has a subordinate type object, and a base type object can have objects from several subordinated types simultaneously.

For the development of object specialisation type, there are database restrictive mechanisms C used: declarative restrictions and triggers. They implement the necessary object association logics.

The specialisation of the object is not an aim in itself. It is used to develop even more logically complicated constructions: inheritance, aggregation and composition.

Inheritance is an association between base type object and one or several subordinate type object(-s). It indicates that not only these objects have an association but also determines the mechanism of their association:

1) Subordinate type object can automatically be supplemented with base type attributes;
2) The methods of base type object can be used for subordinate type objects, in which base type attributes can be additionally included.

Inheritance is the top type of specialisation. A lot of inheritance types are formed due to a large number of the types of specialisation: single inheritance – one base type; complex multiple; repeated inheritance – several base types; multilevel inheritance – there are more than 2 levels in the hierarchy of types; hierarchical inheritance – base type has several subordinated types; multipath inheritance – complex inheritance + multilevel inheritance; hybrid inheritance – single inheritance + complex inheritance; base type object union inheritance – the sub-type object is associated with several base type objects and subtype object simultaneously inherits attributes and methods only from one base type object; in union inheritance of subordinated objects – every base type object is mandatory linked with a subtype object and the main type object can be linked with several subtype objects (the subtype objects can also be another subtype objects).

To ensure the functioning of inheritance mechanism, not only object association must be indicated but also object type hierarchy H. Consequently, the description of object-relational database structure (1) can be supplemented with association restriction C and type hierarchy components:

$$S = S(K, O, M, R, C, H). \qquad (2)$$

When constructing the structure of object-relational database, various possible variants are developed and their analysis is performed. For the automation of this process, a constructive structure model is necessary that can be used in software. XML language is used for data exchange among systems and their parts. It is the official data exchange standard. XML language allows creating the descriptions of hierarchical data structures. They can be easily used in Java and C# programming languages. XML language is also used for the transformation of conceptual model into object relational model [20], [21].

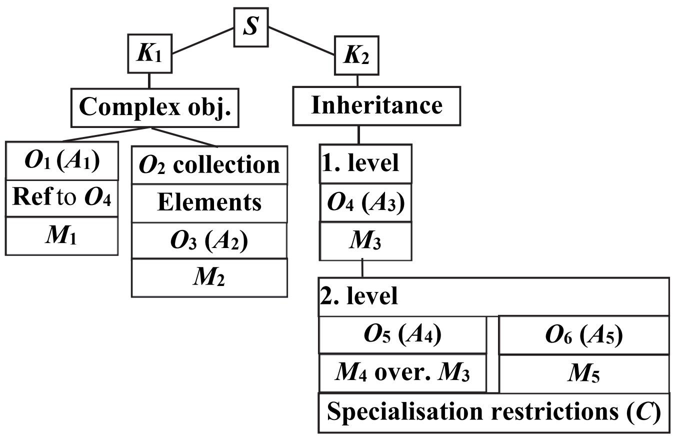

Fig. 8. Model of database structure components.

Figure 8 shows the example of object-relational database model structure S. It is a hierarchical structure of data and their descriptions that can be easily defined in XML language. The structure is formed by two object table frameworks K_1 and K_2. In the first table of objects, a complex writing type object is formed by O_1 type object and collection type object O_2. In the second object table framework K_2, inheritance is being implemented by using heterogenic writing type objects O_4, O_5, O_6. Object type hierarchy second level method M_4 override first level method M3 code. The restrictions C are also defined for the use of specialised objects O_5 and O_6.

Framework K_1 complex objects have an object link with framework K_2 base objects. References with object O_4 identifiers recorded are included to define association in objects O_1.

Object-relational database structure XML models can be easily used in software to generate structures and make their transformation and visualisation.

V. Optimisation of Object-Relational Database Structure

Object-relational database technology allows for a large variety of the transformation of information system conceptual model into the physical model of database (this depiction is practically unequivocal for relational databases). The designer of a database must examine and assess a large variety of different potential solutions (Section IV). In many cases, the designer is even not aware of all possible solutions. Due to this reason, the following database structure optimisation problem must be formulated and solved:

$$S\,(K, O, M, R, C, H) \rightarrow \text{optimum} \Rightarrow S_{\text{opt}}, \qquad (3)$$
$$K, O, M, R, C, H \in \Omega,$$

where Ω is admissible variation space of the database structure.

Structure optimisation problems are solved by using parametric optimisation in many sectors, for example, mechanics, construction, shipbuilding, etc. The mathematic model of the structure to be constructed is developed with the parameters characterising it, and model optimisation is performed by changing the values of parameters. The experience in the optimisation of topologies and forms has revealed the disadvantages and inefficiency of parametric optimisation in the operation with structures [22]. There is the analysis provided of a set of parameters rather than the structure, and in many cases it is not the same. For example, the values of the construction parameters of a building are very good but their look and usability (subjective criteria) are unsatisfactory. It is really difficult to adjust such evaluations of criteria for the values of a parameter cluster. Non-parametric or structure optimisation is used in these areas [23]. It is necessary that structure S variation space Ω is created by possible structures rather than their parametric models. When moving in this space from one structure to the next structure, there would be the structures and not their parametric models analysed. It would allow for better use of both official and unofficial assessments. It is necessary to define structure variation space Ω to see it. It is not always possible, and often it is not easy.

Object-relational database structure variation premise Ω is formed by its possible implementation structures. Every structure S_X has its "closest" structures S_X^i, (i = 1, …, v) that are obtained by making the simplest alterations. They also have their "closest" structures. In such a way, object-relational database structure variation space Ω is formed. It is a space with a definitive number of structures.

First of all, there must be successive moving in structure space Ω ensured for finding an optimum structure. It can be implemented in the following way (Fig. 9):

1. Alteration rules are found for initial structure S_0 in the Transformation Rule Database (TRDB) that can be used for it (selected rules – SR). Activities 1 and 2 are performed, Fig. 9.
2. Selected rules are used for performing transformations (T) for structure S_0 (Activities 3, 4, 5, 6). New structures S_1 and S_2 are obtained (the number of these structures can vary a lot).
3. Suitable rules are selected again in the Transformation Rule Database for every new structure S_1 un S_2.
4. The selected rules are used for the transformation of structures S_1 and S_2. New structures S_3, S_4 and S_5 are obtained. After that, there is a transition to the fulfilment of Step 3. Sequential structure generation cycle is created.

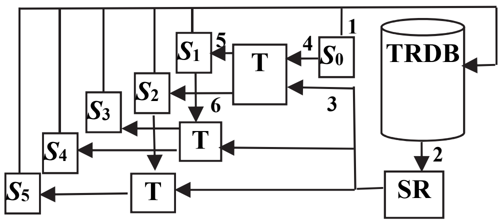

Fig. 9. Sequential generation of object-relational database structures.

When constructing the database structure, it is not necessary to generate all sequential database structures. The process must be oriented towards obtaining better and better structure. Object-relational database potential structure space is a discrete space. Its examination allowed drawing a conclusion that global optimisation methods (local optimums might exist) must be used for solving an optimisation problem in this space. When performing an iterative optimisation process, the number of structures to be analysed in any iteration is not large. It is important to ensure the "inertia" of the process so that it does not stop in a local, optimum. One of the best methods that can be adapted and used in this situation (according to the authors) is the variant of random search method group [24] – random search with trend vector (RS Trend Vector method)). The operation of the algorithm of the method is shown in Fig. 10.

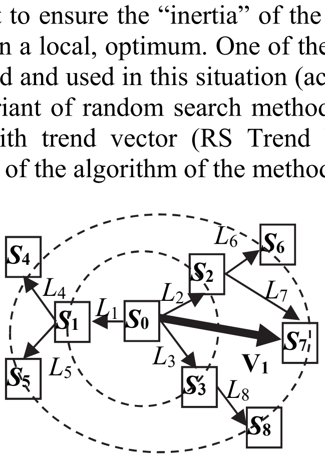

Fig. 10. Global optimum search with RS Trend Vector method.

Suitable transformation rules (L_1, L_2, L_3) are applied to the initial structure of the database at the beginning of constructing S_0. New structures S_1, S_2, S_3 are obtained, which slightly differ from the initial structure (transformation rules perform minimum transformation of the structure):

$$S_1 = L_1(S_0),\ S_2 = L_2(S_0),\ S_3 = L_3(S_0). \quad (4)$$

These are the structures that are "closest" to S_0 (from the perspective of structure comparison). The structure examination space must be extended because it is necessary to ensure the operation of algorithm to perform global optimisation. Possible structure transformation rules are applied to structures S_1, S_2, S_3. New structures are obtained S_4, S_5, S_6, S_7, S_8. Depending on the complexity of optimised structure, there might be a necessity for a larger expansion of the space of the structure to be analysed.

When "closest" surrounding structures (S_1, S_2, ..., S_8) of initial structure S_0 have been obtained, they must be assessed. The "best one" is chosen as the beginning of the following iteration:

$$\text{If optimum } \{S_1, S_2, ..., S_8\} = S_7, \text{ then } S_0 \rightarrow S_7. \quad (5)$$

When several iterations are performed, it is possible to find out which transformation rules are most frequently used, and if it is possible to see the trend, they can be stressed in further search process.

Object-relational databases include possibilities to store large amount of data and multiplicity of the creation of data retrieval structure. Database structure quality criteria or metrics must be defined and used to compare alternative variants and make a choice. The quality of data storage and retrieval is influenced by several factors:

1) Usability: how fast and how easy the developer of the information system or the database user can understand the possibilities of the use of the database;
2) Productivity or efficiency: how simply and efficiently the user can implement his/her wishes;
3) Functionality: how wide the possibilities of the use are;
4) Maintainability: possibilities to make modifications and improve the performance.

International standard ISO/IEC 25010:11 "Systems and Software Quality Requirements and Evaluation" of 2011 defines five most essential qualities of maintainability [25]: analysability of the structure, changeability, stability (every activity of the database is performed completely or, in case of a mistake, the activity is completely annulled, testability, compliance (possibilities to extend or replace the structure).

A lot of efforts for the research and development of database structure quality metrics have been devoted by joint research group ALARCOS from the University of Castile-La Mancha in Spain [26]. They have analysed quality metrics for relational, object-relational and active databases. The following metrics have been used for object-relational databases: metrics referring to the quality of the table, coupling metrics, complexity metrics, and data referential integrity metrics.

Database quality metrics have also been analysed in works of other authors [27], [28]. Additionally, subjective weight or importance coefficients are introduced in examined metrics.

Nevertheless, the most important conclusion of specialists is that there is no common criterion, which can evaluate overall quality of a database by the use of a scalar value. Due to this reason, to obtain a good structure of a database it is necessary to use not only metric (evaluation criteria) but also automated process. A compromise solution must be found in it by sequential examining and assessing various database structures.

During the questioning of more than 20 experienced database designers, the criteria used by them to evaluate object-relational database structure have been identified.

1. **Semantic clarity** of a database was dominant. All data units, their groups (objects), association among groups and separate data units must be clearly and possibly simply defined and implemented. Namely, it must be easy to understand them.
2. The next priority was **performance** or reactivity of the fulfilment of the main queries. The fulfilment of user queries must fall within certain time limits. It is the only quantitative criterion for the evaluation of database systems, which are used by SQL optimizer and test database. Its structure is automatically adjusted in compliance with the analysed structure. Necessary indices are also created.
3. It was noted that the **simplicity of database changes and additions** were also very important. A database "lives a long life". New information is entered; the data structure is adjusted and supplemented. It is not possible to design and implement a completely finished database. There will always be changes in it. It must be ensured that there is a possibility to make these changes in possibly simplest way.

Empiric models were developed to determine the values for the criteria of semantic clarity and simplicity of database changes and additions. Opinion of professional database designers was used for their implementation. It must be noted that in the course of iterative database constructing, it is possible to adjust these models.

The mathematic formulation of multi-criterial optimisation problem of object-relational database structure is the following:

$$\left.\begin{array}{l} q_1(S) \rightarrow \text{optimum} \\ \quad S \in \Omega \\ \dots \\ q_n(S) \rightarrow \text{optimum} \\ \quad S \in \Omega \end{array}\right\} \rightarrow \{S^*_{\mathrm{p}}\} \in P, \tag{6}$$

where $q_1(S), ..., q_n(S)$ – quality criteria, optimum – criteria value priority direction (min or max), P – formally the cluster of the best structures S^*_{p} (Pareto cluster). The "best" structure is to be found in Pareto cluster.

Various algorithms and approaches have been developed [29] for solving multi-criterial optimisation problem (6). The following approaches are used:

1) A priori approach: the priorities of criteria are clearly defined;
2) A posterior approach: experiments are initially performed to define the priorities of criteria more precisely;
3) Adaptive or interactive approach: sequential, directed analysis of database structures is performed in dialogue regime; at the end of each iteration, the designer provides the structure assessment and wishes I_t ($t = 1, ...$):

$$q_1(S_0), ..., q_n(S_0) \xrightarrow{I_1} q_1(S_1), ..., q_n(S_1) \xrightarrow{I_2} q_1(S_2), ..., q_n(S_2) \xrightarrow{I_3} ... \rightarrow q_1(S^{**}), ..., q_n(S^{**}). \tag{7}$$

Moreover, (in the opinion of the designer) $S^{**} \in P$ is the "best" database structure. It is good if the structure revision can be made in Pareto P cluster, but the solving process (7) can also be implemented starting with non-Pareto cluster structure.

In most cases, an adaptive approach is used in the problems of designers. By using it for designing object-relational database construction, the visualisation of graphic variants of the database obtained in iterations can be made. It allows adjusting the values of models of empiric quality criteria and precision of these models can be made.

The issue of formalising of the structure assessment I_t ($t = 1, ...$) is problematic. It includes the following information:

1) The values of the quality criteria of the structure (how good or bad they are);
2) The indications about changes to be made in the structure (the number or tables that must be decreased (increased), more object addressing mechanisms must be used, object type hierarchy must be increased (decreased)).

The designer of a database must understand the correlation of this information very well to direct the process of optimisation into the desirable direction.

VI. Approbation of the Developed Technology

The system prototype (Fig. 11) was developed for the approbation of the construction technology of the physical object-relational database structure. It has the following components:

1) Conceptual model diagram designing software: an extended UML model has been used. The extensions are linked with more expanded reproduction of object specialisation. It is necessary for inheritance and aggregation constructions (database specifics in comparison with object-oriented programming);
2) Conceptual model transformation rules in initial structures of object-relational database construction: 3 database structure models are developed in XML and object SQL languages, as well as a model for graphic reproduction of the structure;
3) Adaptive (iterative) multi-criterial optimisation algorithm, which uses the method of random search trend vector. The database structure graphic model is also used for the visualisation of structures.

The prototype was developed using database management system Oracle11 with database language object SQL and programming languages PL/SQL, Java, XML, JavaScript. The recommendations of technologies "in database processing" and "in database analytics" were used for the implementation of the system [30].

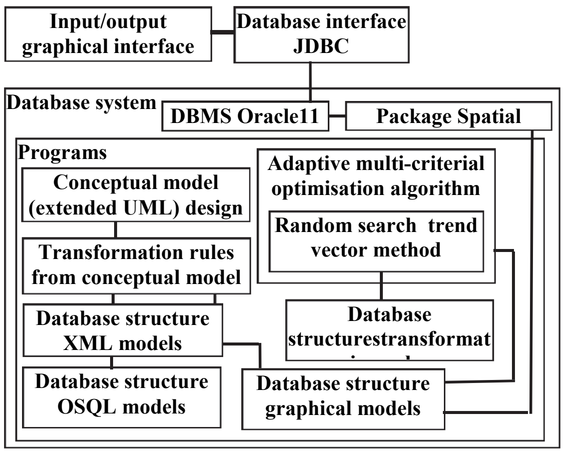
Input/output graphical interface
Database interface JDBC
Database system
DBMS Oracle11
Package Spatial
Programs
Conceptual model (extended UML) design
Adaptive multi-criterial optimisation algorithm
Random search trend vector method
Transformation rules from conceptual model
Database structure XML models
Database structurestransformat
Database structure OSQL models
Database structure graphical models

Fig. 11. System prototype of object-relational database structure construction.

The method of object-relational database designing was approbated by developing the database of fruit-tree qualities for the Latvian State Institute of Horticulture. The initial structure consisted of 40 tables and 320 attributes. An average depth of object hierarchy was 1.3, and the maximum average depth of object hierarchy was 3. In the final structure, there are 18 tables, an average depth of object hierarchy is 2.1, and the maximum average depth of object hierarchy is 7.

Figure 12a depicts a fragment of this database: kinds of fruit-tree (table A), samples (table B) and projects (table C). After transformation from conceptual model depicted in Fig. 12a, there were 7 tables, an average depth of object hierarchy was 1.6. When performing single iteration of multi-criterial optimisation with indication that it was desirable to decrease the number of tables, the structure depicted in Fig. 12b (number of tables – 3, an average depth of object hierarchy – 3) was obtained.

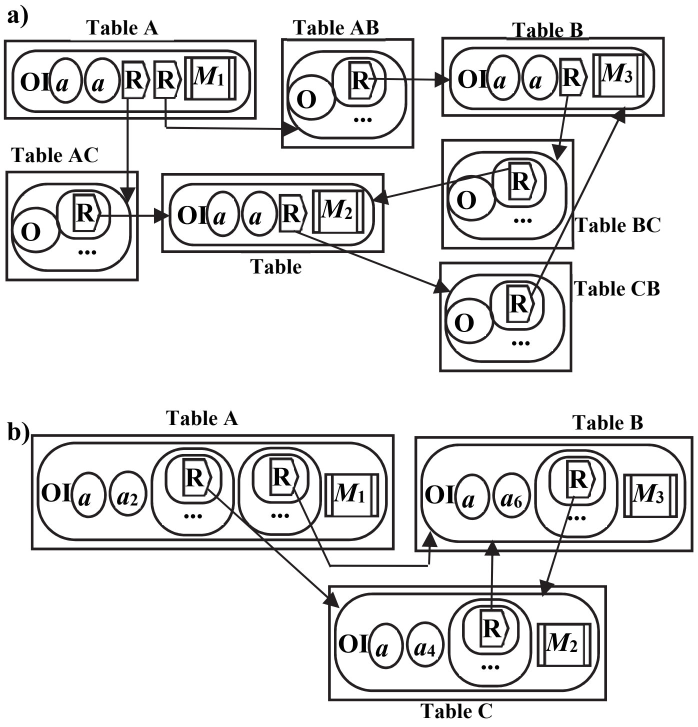

Fig. 12. Fruit tree property database structure at the beginning (a) and end (b) of construction.

VII. Conclusion

The development of object-relational database construction method and the system prototype, as well as their approbation allowed making several conclusions.

1. It was established and demonstrated that the following factors made the construction of object-relational database difficult:
   a) Multiplicity of alternative structures;
   b) Multiplicity of criteria in the assessment of the quality of structure;
   c) A lack of a constructive physical model of object-relational database.
2. The constructed physical model of object-relational database and the developed rules for its transformation allow generating automatically the possible structure variation space of the database, which provides a possibility of structure overview when performing the search for the "best" structure.
3. By the use of structure variation space, it is possible to perform the optimisation of the structure instead of parametric optimisation. Such an approach can be used not only for the construction of the database structure, but also in building, engineering industry and other sectors.
4. When constructing object-relational database, global multi-criterial optimisation must be used. In the meanwhile, the values of many criteria must be taken into account, and there are also local structure criteria in the variation space of possible structures.
5. Additional solutions must be still sought for the use of database structure transformation rules to generate the structures that are "closest" to the selected database structure. Use order and links of these rules must be improved to develop a united, compact, and coordinated system of rules.
6. Experts in database design assessed that the visualisation possibilities of the designed object-relational database structure in combination with possibilities to make changes while moving in the possible structure space were very useful. Consequently, it provides a possibility to make a visual analysis of "closest" alternative variants and to gain better understanding of the possibilities of the change in the structure. Such a visual revision tool of database possible structure was recognised as a valuable improvement in construction.
7. Only system prototype was developed. A lot still has to be done to use it completely. The process of database creation takes place with the participation of system analyst together with potential users (customers), as well as designers of object-relational database. Each group has its own tasks. The first group must provide the description of database semantics, which is as precise and extended as possible. It must be included in the conceptual model. In the meanwhile if there are any indications that might be useful for the physical development of the database, it also must be possible to include them in this model (additional work for system analyst). Previous opinion stating that the data conceptual model is not completely focused on a specific database system is not always correct.

References

[1] E. F. Codd, “A Relational Model of Data for Large Shared Data Banks,” *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, Jun. 1970. https://doi.org/10.1145/362384.362685

[2] J. Pokorny, “Databases in the 3rd Millennium: Trends and Research Directions,” *Journal of Systems Integration*, vol. 1, pp. 3–15, 2010.

[3] E. F. Codd, *Relational Model for Database Management: Version 2*. London: Addison–Wesley, 1990.

[4] The Committee for Advanced DBMS Function, “Third-Generation Database Systems Manifesto,” *Computer Standards and Interfaces*, vol. 13, issue 1–3, pp. 41–54, Oct. 1991. https://doi.org/10.1016/0920-5489(91)90008-n

[5] H. Darwen and C. J. Date, Databases, types and the Relational Model, 3rd ed. London: Addison–Wesley, 1998.

[6] C. Travers, “DZone/Big Data Zone. Object-Relational Algebra: Definitions and Assumptions,” Oct. 22, 2012. [Online]. Available: https://dzone.com/articles/object-relational-algebra. [Accessed: 21 Nov. 2017].

[7] S. K. Gupta, *Oracle Advanced PL/SQL Developers Prefessional Guide*. Birmingham: Packt Publishing, 2012.

[8] E. Marcos, B. Vela, and J. M. Cavero, “A Methodological Approach for Object-Relational Database Design using UML,” vol. 2, no. 1, pp. 59–72, Mar. 2003. https://doi.org/10.1007/s10270-002-0001-y

[9] E. Pardede, W. Rahayu, and D. Taniar, “Mapping Methods and Query for Aggregation and Association in Object-Relational Database using Collection,” in *Proc. International Conference on Information Technology: Coding and Computing*, 2004, pp. 539–543. https://doi.org/10.1109/itcc.2004.1286513

[10] W. Y. Mok and D. P. Paper, “On Transformations From UML Models to Object-Relational Database,” in *Proc. 34th Hawaii International Conferenceon System Sciences*, USA: Hawaii, 2001, pp. 1–10. https://doi.org/10.1109/hicss.2001.926341

[11] N. Keivani, A. M. Maatuk, S. Aljawarneh, and M. A. Ali, “Towards the Maturity of Object-Relational Database Technology: Promises and Reality,” *International Journal of Technology Diffusio*, vol. 6, no. 4, pp. 112–120, Oct. 2015. https://doi.org/10.4018/ijtd.2015100101

[12] Ch. Parent, S. Spaccapietra, and E. Zimányi, *Conceptual Modeling for Traditional and Spatio-Temporal Applications: The MADS Approach*. Berlin: Verlag Berlin Heidelberg, 2007, pp. 194–205.

[13] N. N. Karanikolas and M. Vassilakopoulos, “Comparison of Post-Relational and Object-Relational Modelling for Real-World Database Applications,” *Journal of Systems and Information Technology*, vol. 16, no. 4, pp. 313–334, Nov. 2014. https://doi.org/10.1108/jsit-05-2014-0034

[14] A. Alami and M. Bahaj, “Framework for a Complete Migration From a Relational Database RDB to an Object Relational Database ORDB,” *International Journal of Scientific Engineering and Applied Science*, vol. 1, issue 6, pp. 134–138, Sep. 2015.

[15] V. L. Malykh, A. N. Kalinin, and T. Sh. Yusufov, “Object-Relational Approach to Building a Data Storage”, *Program Systems: Theory and Applications*, vol. 8, issue 3, pp. 169–187, 2017.

[16] T. R. de Castro, S. N. A. de Souza, and L. S. de Souza, “CASE Tool for Object – Relational database Designs”, in *7th Iberian Conference on Information Systems and Technologies*, 2012, pp. 156–161.

[17] C. A. Rombaldo Jr, S. N. A. Souza, and L. S. de Souza, “O-ODM Framework for Object-Relational Databases,” *International Journal of Artificial Intelligence and Interactive Multimedia*, vol. 1, no. 6, pp. 29–35, 2012. https://doi.org/10.9781/ijimai.2012.164

[18] M. F. Golobisky and A. Vecchietti, “Fundamentals for the Automation of Object-Relational Database Design,” *International Journal of Computer Science Issues*, vol. 8, issue 3, no. 2, pp. 9–22, May 2011.

[19] *ORACLE. Ask The Oracle Masters (ask Tom)*. [Online]. Available: https://asktom.oracle.com/pls/asktom. [Accessed: 12 Sept. 2017].

[20] M. Machkour and K. Afdel, “Transforming XML Into Object-Relational Schema,” *IOSR Journal of Computer Engineering*, vol. 18, no. 5, pp. 40–52, May 2016. https://doi.org/10.9790/0661-1805014052

[21] L. Sang, J. Xiao, and X. Cui, “Converting XML Schema Data to Object-Relational Data With DOM,” in *Proc. 2011 International Conference on Web Information Systems and Mining,* vol. 2, Springer-Verlag, 2011, pp. 452–460.

[22] J. Haslinger and R. A. E. Makinen, *Introduction to Shape Optimization: Theory, Approzimation and Computation.* USA, Philadelphia: Industrial and Applied Mathematics, 2008. https://doi.org/10.1137/1.9780898718690

[23] J. Chen, V. Shapiro, K. Suresh, and I. Tsukanov, “Parametric and Topological Control in Shape Optimization,” in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, USA, Philadelphia, September 10–13, 2006, vol. 1, USA: Pennsylvania, 2010, pp. 234–256.

[24] J. Eiduks, “Control of Comlicated Systems in Case of Vectorial Indicator of Quality,” in *Forth Formator Symposium on Mathematical Methods for the Analysis of Large-Scale Systems*, Prague: Publishing House of the Czechoslovak Academy of Sciences, 1983, pp. 165–178.

[25] *ISO/IEC 25010:2011.* March 2011. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en. [Accessed: 7 Sept. 2017].

[26] A. L. Baroni, C. Calero, M. Piattini, F. Brito e Abreu, “A Formal Definition for Object-Relational Database Metrics,” in *Proc. Seventh International Conference of Enterprise Information Systems*, USA: Miami, 2005, pp. 334–339. https://doi.org/10.5220/0002510503340339

[27] S. Justus and K. Iyakutti, “A Formal Suit of Object Relational Database Metrics,” World Academy of Science, Engineering and Technology, vol. 24, pp. 750–759, 2008.

[28] S. M. Suleiman, Q. A. Al-Radaideh, B. A. Abulhuda, and I. N. AlSmadi, “Automating the Collection of Object Relational Database Metrics,” *International Journal of Advanced Computer Science and Applications*, vol. 2, no. 6, pp. 19–27, 2011. https://doi.org/10.14569/ijacsa.2011.020603

[29] L. A. Rastrigin and Ja. Ju. Ejduk, “Adaptivnye Metody Mnogokriterialnoj Optimizacii,” Avtomatika i Telemehanika, Vipusk 1, 1985, s. 26–38. (in Russian).

[30] D. M. Kroenke and D. J. Auer, *Database Processing: Fundamentals, Design and Implementation*, 14th ed. UK: Pearson Higher Education, 2016.

Ainārs Auziņš received *Mg. sc. ing.* degree in 2003 from Riga Technical University. His current research interests include database technologies and optimisation.
Since 2011, he has been working as a System Analyst, Researcher and Lecturer at RTU.
E-mail: ainars.auzins@rtu.lv

Jānis Eiduks holds *Dr. sc. ing.* degree. He is a Professor at Riga Technical University. His current research interests include database technologies and optimisation.
E-mail: Zane_matematika@inbox.lv

Alīna Vasiļevska received *Mg. sc. ing.* degree in 2017 from Riga Technical University. His current research interests include artificial intelligence.
E-mail: vasilek93@inbox.lv

Reinis Dzenis received *Bc. sc. ing.* degree in 2017 from Riga Technical University. His current research interests include database technologies and programming.
E-mail: reinisdzeniso@gmail.com

.