Модел. и анализ информ. систем. Т. **22**, № **2** (2015) 158–175
©*Antonov D. V., Roublev V. S.,* 2015

UDC 519.682; 519.688; 004.65

Access Efficiency to Data in DIM DBMS

Antonov D. V., Roublev V. S.

P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia

e-mail: dmitrii.antonov@gmail.com, roublev@mail.ru

received March 1, 2015

Keywords: DBMS, RDBMS, OODBMS, DIM, discrete determined models, OD-model, query technologies, ODQL-queries, DB converting

In the article is provided the review of tools used in new type object DBMS for increasing the efficiency of access to data. Described object DIM DBMS features based on the use of the classes of objects relations as objects sets (inheritance, inclusion, interaction and history) and objects relations (inheritance, internal inheritance, inclusion, internal inclusion, interaction and history). The description of subject domain is entered by means of an object and dynamic data model (OD-model), and DIM DBMS completeness for any OD-model is justified. ODQL object query language allowing to combine the exact description complexity with the simplicity of use due to two query levels introduction is described. For the elucidation of the most effective way of the appeal to DIM DBMS the study of various query technologies for this environment is conducted, and mechanisms for users work with it are developed and realized. For this purpose a software complex necessary for work with DIM DBMS is developed. The description of the "DIM Navigator" main software development concepts, necessary for data manipulation opportunity in the available DB by means of the graphic interface is provided. Software development "The Generator of ODQL-queries" is considered which is necessary for simplification of query creation to DIM DBMS, needlessly for the user to know the syntax of a modern query language. Problems of converting data from the existing DBMS in DIM DBMS are considered.

The article is published in the author's wording.

Introduction

The improvement of design technologies and the use of databases gained the new development connected with computer equipment progress in regard to speed and memory growth. The last scientific achievements (for example, in astrophysics) demand storage and processing rather large information volumes which already approached petabytes [1]. The necessity of continuous changes of not only data but also algorithms is "the invariable characteristics of the modern world" [2]. Shortcomings of the available DBMS models allowed to set the task about the creation of a new DBMS technology which uses the advantages of the available technologies: temporal, relational, object-oriented and object-relational ([3]–[8],[16]).

1. Dynamic information DIM model

In [9] the new object approach to DBMS creation is described which assumes not only the change of these objects, but also types of objects change possibility, i.e. database schemes called a dynamic information model (DIM). In this approach 6 basic objects relations were allocated: inheritance, inclusion, internal inheritance, internal inclusion, history and interaction. For the description of subject domain is entered the general definition of the Discrete determined model whose objects can evolve, formalization is also given to such models (OD-model) and DIM completeness is shown: any OD-model and its evolution can be described by means of DIM. Described in [10] the language of object and dynamic queries ODQL for the dynamic information DIM model possesses the completeness – any DIM objects group (with any properties) can be allocated with ODQL query [11]. In [12] is described the organization of queries performance at which the optimum labor input is reached. But time of queries performance depends as on the organization of data storage and manipulation with them, and on a realization platform choice. Thus questions of data dynamic change are considered. Consider at first ways of receiving such an organization which does not depend on a platform choice for realization.

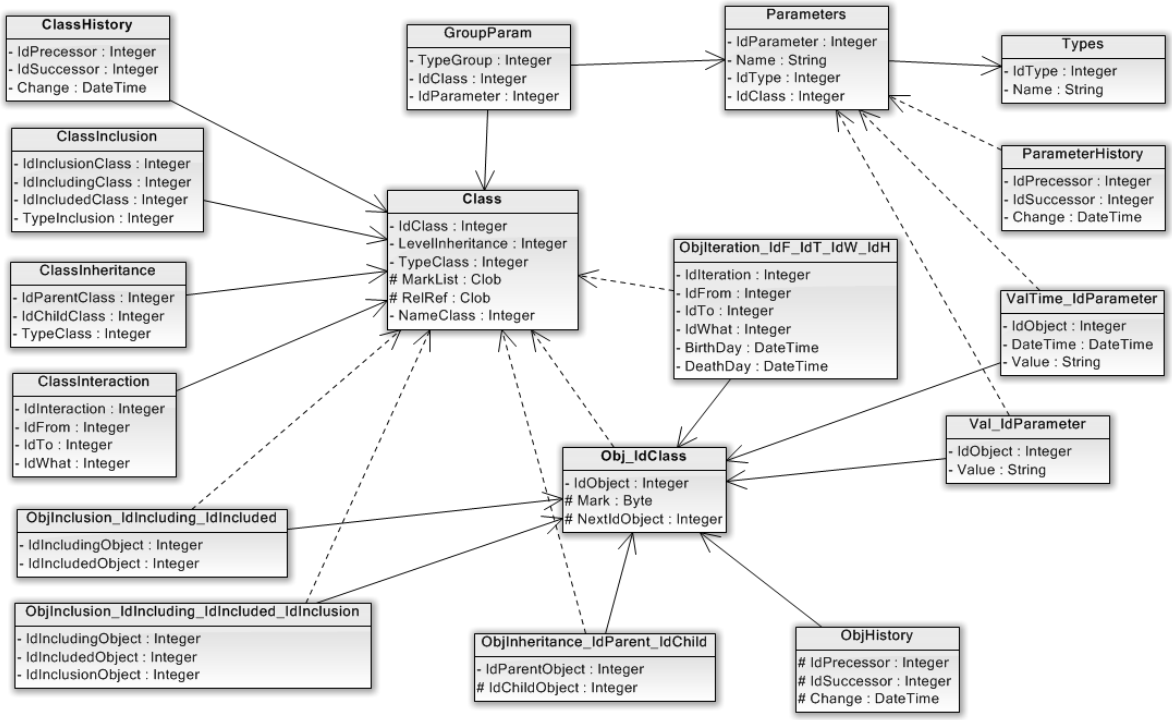

Fig. 1: Scheme of the DIM metalevel

The natural way of the dynamic DBMS organization can be defined by the metalevel description of objects, classes of objects, parameters and relational communications of these entities. The metalevel represents a relational database with a set of tables comprising all necessary values. Thus, the choice as a platform of relational DBMS can be made later on the basis of the correctness analysis of a choice of this or that platform for the metalevel. Information on classes is provided in tables of metalevel as about objects of metalevel. As parameters of each class can change, information on classes

parameters is also provided in other tables connected with the classes table by relational communications. Objects of each class will be organized in the separate tables connected with classes of the classes table. In much the same way, values of object parameters are placed in the separate tables connected with the corresponding parameters and objects tables by relational communications. Note that each class, parameter, object has a unique identifier (IdClass, IdParameter, IdObj) for establishment of the mentioned communications between tables. The scheme of the DIM metalevel organization is given in Figure 1. In this scheme, ClassInheritance, ClassInclusion, ClassInteraction, ClassHistory of the classes communications table corresponding to the relations of inheritance, inclusion, interaction and hierarchy are shown. The table Obj_IdClass of class objects defines for each object an object choice "Mark" mark in a class index-selection [12], and the index-selections of classes and the classes relations are realized as external indexes of this table and tables of objects communications.

2. OD-model

Now in many areas of human activity for the description of various processes the discrete determined models are often used. Model discretization in this case is understood as a final, though potentially unlimited number of model objects, and determinancy of model is understood as the determined laws of the model objects behavior. Formalization of the discrete determined model has led to creation of object and dynamic model (OD-model), and for the adequate description of its data formalization of the classes scheme DIM and formalization of the OD-model static description by the classes scheme DIM [30] are entered into DIM. We will call a group of elements

$$(O, A, \bar{A}(o), V(o), L_p, L_o, L_f, \bar{A}_{L_f}(o_l^j), V_{L_f}(o_l^f), F, T) \tag{1}$$

an OD-model where

O — a final objects set,

$A = \bigcup_o A_o$ — a final set of objects properties with types of these properties (this set element of pair (a,V^a) – property, property type),

$\bar{A}(o)$ — a cortege function of object properties o,

$V(o)$ — a cortege function of objects properties values (orderliness of object properties values of o corresponds to orderliness of this object properties in a cortege $\bar{A}(o)$),

$L_p = \bigcup_{j\in\ L_p}\{l_j^p = \{o, o1\}\}$ — a set of objects simple communications,

L_o — an objects-communications set$(O \cap\ L_o = \emptyset)$,

$L_f = \bigcup_{j\in\ L_f}\{(l_j^f, o_l^j \in\ L_o)\}$ — a set of objects functional communications,

$\bar{A}_{L_f}(o_l^j)$ — a cortege function of object-communication attributes o_l^j functional communications L_f,

$V_{L_f}(o_l^f)$ — a cortege function of object-communication attributes values o_l^f functional communications L_f,

F — a final set of algorithmic procedures of objects properties values change and objects change,

T — a discrete time scale

3. DIM DBMS advantages and a problem of data converting in DIM DBMS

The available DBMS technologies possess the following shortcomings:

1. Relational — is universal, effective on the realization but very difficult for the use, as it is necessary to project in terms of a large number of tables, but not objects.
2. Object-oriented — is object, but has methods shortcomings: are convenient for the objects interactions description of one class, but for objects interactions of different classes, that is more often used in a DB, it is inconvenient as it is compelled to use the asymmetrical device of "friendly" functions.
3. Object-relational — has advantages in comparison with relational, but also it has shortcomings of both technologies.
4. Temporal — has advantages in opportunities of data change preservation history, but not their types.

DIM DBMS has some advantages in comparison with other systems:

1) is object, and the objects relations device (including internal inheritance and internal inclusion) allows to describe adequately OD-models data, and the interactions device allows to describe symmetrically laws of data change and data types;
2) unlike the temporal one allows to keep data change history, and their types.

The DB transformation from the existing DBMS to DIM puts a problem of data converting.

As there are DBMS of different types, no uniform algorithm for data converting from any DBMS to DIM can be written, but it is possible to use OD-model, that is, at first to transform the available DB to OD-model, and then to use the available OD-model transformation algorithm to structure of DIM DBMS, following the theorem of the static completeness described in [30].

Theorem of static completeness.
Any OD-model OD for any moment $t \in T$ *in it can be statically described by means of some scheme S of the DIM classes which is in a normal form.*

3.1. Algorithm of receiving display for any model and its realization

For a start, it is necessary to receive a display for any model. The algorithm of receiving the display appears as follows:

1. A series of queries for obtaining the list of the tables, the fields corresponding to them, communications between tables presented in DB, and also sets of the values which are available there, is carried out.
2. The arrays which are responsible for the sets corresponding to sets of OD-model are filled.

3.2. Display algorithm of relational model data to the OD-model

The algorithm of receiving the display for a relational model looks in many respects similar to the general algorithm, but has some differences, namely, has such operations procedure:

1. The query for obtaining the tables list and the fields corresponding to them presented in a DB is performed.

2. On the basis of the obtained data, the array which is responsible for a table name set of a transferable DB is filled up.

3. The two-dimensional array is filled up (as it is necessary to compare the name of each field with the type corresponding to it) which is responsible for a field set of all tables (information on the name of fields and their type is registered in the array).

4. A series of queries to tables for obtaining information about the data written down in their fields is performed (the data sets corresponding to each field are read out).

5. The array which is responsible for a set of the values which are written down in the table is filled (each set is divided by a special tag for opportunity further to distinguish sets from each other).

6. A series of queries with the purpose to find out existence of external indexes, therefore, of communications between tables is performed.

7. The array which is responsible for a set of communications between tables is filled.

As a result of the algorithm execution we receive a set of arrays containing information on a set of properties, objects and communications. On the basis of these data the model is formed which corresponds to the description of OD-model from which by means of a special program it is possible to receive a structure corresponding to the DIM DBMS metalevel.

4. Effective realization of the DIM system

4.1. ODQL object queries language

The object concept is difficult as for the allocation of its properties and their values it is required to work both with object class properties, and with the properties received on inheritance. Therefore, an actual task is to introduce such a language, by using which a user could set objects of one class (or several classes with their communications), considering not only parameters and the properties of a class inclusion but also all the inherited properties.

The SQL query language for RSUBD is evident, but is not objective. The ODMG group, being the founder of one of the OOBD technologies, developed the standard of object OQL query language (see [10]). But, first, this technology does not pursue the aim of adaptive DB creation which will be able to change dynamically the data scheme,

and secondly, the classes relations entered in it do not allow to describe adequately any discrete determined models that is also the property of the DIM technology proved in the same place. Therefore, the object query language allowing to carry out manipulations with data to DIM is necessary. This language by means of the constructions must define precisely what we wish to allocate, and be simple enough in use, it was possible to set visually that information which needs to be allocated with a small amount of clear constructions.

4.1.1. Two levels of the ODQL query language

The specified requirements imposed to the ODQL language are inconsistent. According to the first of them it must contain the constructions answering to the entered DIM relations and, in particular, to be object, i.e.

- it can allocate an object group of all objects set of this or that class by restrictions on these objects properties;
- it can mark out object properties, whether they are inborn or inherited;
- it can allocate the included objects (both included in object, and included in its parental objects);
- it can mark out properties of the included objects.

But such constructions are numerous and therefore can make request very difficult and not evident that contradicts the second requirement.

The second requirement is very important in the cases when the user must be able visually and simply get access to data and manipulate them, without knowing in details the classes relation of the corresponding objects. At the same time in other cases the user needs to have opportunity to apply knowledge of the objects and classes relations to produce more sophisticated queries.

To resolve this contradiction we introduce 2 query levels of ODQL:

- the lower level of language allowing the user, by using the knowledge of the classes relations, to describe precisely a query to data, specifying all objects and their classes relations;
- the top level of the language where only the constructions which limit setting property values of an object group and a class (or several connected classes) of the allocated objects group participate in the query description, and also the list of the marked-out properties for these objects (all other information can be received from the metalevel of the classes DIM description).

The complete description of both levels of the ODQL language can be found in [10].

4.1.2. Description of query language

The query generally may contain phrases of *from*, *select*, *for*, *links*, *where*. As well as for SQL queries the *select* phrase defines that will get out, the *from* phrase – for what

entities follows a choice. Phrases of *for*, *links* and *where* define choice conditions, i.e. correspond to the phrase of *where* SQL.

The main query phrase is *select* which allocates data of necessary entities: – or classes and properties of these classes (classes parameters and classes communications); – or values of objects parameters (and expressions from such values); – or objects (in subqueries).

The *from* phrase contains or the indication of a objects set of one class determined by his name, or a subquery of a class allocation, either a class objects, or transfer through a comma of names of objects classes and (or) subqueries of objects allocation or subqueries of classes allocation. The first class specified in this phrase by a name or a subquery (the first class of a subquery or a class allocated subquery), is called basic: the choice is made by the *select* phrase or for objects of this class, or for "trees" of the connected objects of all classes of the *from* phrase, each of which has "root" in a basic class. The *from* phrase can be lowered that will mean a choice of a class objects which possesses the first specified by the *select* phrase parameter. To specify a class of the chosen parameter, we will enter a name of this class before a parameter name, dividing them a point (or an alias of a class name or a subquery). It isn't obligatory to do it as the class of each property unambiguously is determined by his name owing to unambiguity restriction.

Restrictions on data and communications which in SQL are set by the *where* phrase, in ODQL are divided into restrictions of objects communication which we will describe the *links* phrase, and restrictions of values of classes parameters which we will determine by *for* phrase (restriction on parameters values of this or that class or on cumulative expressions values from parameters of such class) or *where* (restriction on cumulative parameters values of different classes). Such restrictions division can promote more effective query realization:

- restrictions of the *links* phrase allow to allocate the classes connected with a basic class and to define an order of step-by-step query performance upon transition from objects group of one class to objects group of the related class;
- restrictions of the *for* phrase reduce number of the considered objects at each stage of query performance;
- restrictions of the *where* phrase allow to finish allocation of the considered objects.

The combination of query and subquery has the following rules of an performance order:

1. If the subquery doesn't depend on external query, at first the subquery is performed.
2. If the subquery depends on a choice of external query object (the correlated subquery, at first external query is performed and after object (objects) definition from which (whiches) the subquery depends, the subquery is performed).

Query performance for classes consists in the classes group definition having the certain types of communication set by the *links* phrase.

Query performance for classes properties consists in determination of the classes group properties having certain communications.

Query performance for objects consists in objects allocation of a basic class, for each of which the related the classes objects specified by the *links* phrase and satisfying to restrictions of *for* and *where* phrases at their existence are defined.

Query performance for parameters and expressions values from them consists in objects allocation of a basic class and the related objects of other classes determined by the *links* phrase and satisfying to restrictions of *for* and *where* phrases at their existence, and the subsequent expressions calculation of the *select* phrase for each object of a basic class and the related objects [10].

4.2. Ways of ODQL queries realization and their comparative efficiency

Development of the DIM DBMS [9] new object technology is directed on work efficiency with the databases (DB) [6],[13]. So, ODQL query object language [10] is the convenient means for manipulation with data allowing it is easier to describe difficult queries, also it possesses request completeness [11]. However it is necessary that and on time of queries performance this language was effective [15]. In [12] the algorithm of ODQL object queries performance which is also effective on temporary laboriousness was developed.

This algorithm has in a basis the analysis of difficult data communications in this DBMS, and for ensuring efficiency it uses the new device an index-selections of classes objects and an index-selections of the classes relations. The question in, whether is it the best on query performance time in comparison with use approach relational tables of metalevel, been the basis for a data structure [14]? The last approach is based on broadcast of object ODQL query in relational SQL query to the DB metalevel and performance of this SQL query.

For the answer to this question it is necessary to conduct researches on a choice of query technology. For objectivity also necessary comparison of both technologies query time with the best time which can be received for SQL query of a relational DB. The relational technology is supposed as the best in this sense, but the relational query corresponding by result to completely object ODQL query can be many times more difficult. Therefore if SQL query time no more than several times better on ODQL query time, it is quite acceptable for a choice of object technology ([23]–[26]).

4.2.1. Test example

To a choice of a test example is necessary to impose requirement of sufficient laboriousness that time of the specified request technologies differed. Therefore the query with laboriousness $\theta(n^3)$ is suitable for this purpose: performance time of such query will grow quickly enough that will allow to distinguish efficiency of technologies.

Therefore we will take the following task from [10] as a test example of the scheme DB for these queries:

The ships from a *Ships* set, n_s which number, transport freights from a *Cargoes* set, n_c which number of names, to ports from a *Ports* set, n_p which number. Thus the ship s is carry freight c in port p in number of k (s, p).

It is necessary to write the following query:

The list of freights names which carry the most loaded ships and thus to the smallest number of ports among such ships.

At first we will define the scheme of a relational DB and query to it. In this case we will allocate tables:

Ships with the fields IdShip, ShipName,

Cargoes with the fields IdCargo, CargoName and

Ports with the fields IdPort, PortName, and also the communication table

ShipsCargoes with the fields IdShip, IdCargo, IdPort, Quantity.

The SQL query, optimum on laboriousness, can look as follows:

```
Select Distinct Cargoes.Name from
Cargoes,(select ShipsCargoes.IdShip,ShipsCargoes.IdCargo from ShipsCargoes,
 (select IdShip from
  (select min(ShM1.Cntp) as minCntp from
    (Select Count (Ship_Port.IdPort) as CntP, Ship_Port.IdShip From (Select
sc.IdShip,sc.IdPort From ShipsCargoes sc,
     ( Select Sum (Quantity) as SummC, IdShip
      From ShipsCargoes
      Group by IdShip
     ) Ship_Quantity,
     ( Select Max (SummC) as maxSum
      From ( Select Sum (Quantity) as SummC, IdShip
       From ShipsCargoes
       Group by IdShip
      ) Sh1
     )
    Where Ship_Quantity.SummC = maxSum and sc.IdShip = Ship_Quantity.IdShip
     Group by sc.IdShip, sc.IdPort
    ) Ship_Port
   Group by Ship_Port.IdShip) ShM1) ShM,
  (Select Count (Ship_Port.IdPort) as CntP, Ship_Port.IdShip
   From (Select sc.IdShip,sc.IdPort From ShipsCargoes sc,
    ( Select Sum (Quantity) as SummC, IdShip
     From ShipsCargoes
     Group by IdShip
    ) Ship_Quantity,
    ( Select Max (SummC) as maxSum
     From ( Select Sum (Quantity) as SummC, IdShip
      From ShipsCargoes
      Group by IdShip
     ) Sh1
    )
   Where Ship_Quantity.SummC = maxSum and sc.IdShip = Ship_Quantity.IdShip
    Group by sc.IdShip, sc.IdPort
   ) Ship_Port
  Group by Ship_Port.IdShip) ShM2
 where ShM.minCntp=ShM2.cntp) ShN
```

where ShipsCargoes.IdShip=*ShN*.IdShip) *ShY*
where Cargoes.IdCargo=*ShY*.IdCargo

We will define a DIM DB and query to it now. It is described by three classes (like tables of a relational DB):

Ships with the parameters IdShip, ShipName,

Cargoes with the parameters IdCargo, CargoName and

Ports with the parameters IdPort, PortName, and also a communication class

Ships_Cargoes with the parameters IdShip_Cargo, IdPort, Quantity which has the parent the class Ports.

Between the classes Ships and Cargoes the inclusion relation with a communication class of Ships_Cargoes is established. The query on ODQL has rather clear (from the point of view of a task) structure:

select *c.CargoName* **from** *Cargoes c*, *Ships_ Cargoes sc*,
((**select objmaxsum** (sc.Quantity) **on** *s* **from** *Ships s* **links** *s* **contains**(sc) c) *s1*
intersection
(**select objmincount** (*p.obj*) **on** *s1* **from** *s1*, *Ports p* **links** *s1* **contains**(*sc*) *c*, *p* **parent** *sc*)
) *s2*
links *s2* **contains**(*sc*) *c*

The relational query to the DIM DBMS metalevel (we will call it SQL_DIM) looks as follows:

```
select v.value from
 validcargo v,
 (select distinct cargoes1.id_cargo from
  (select scq.id_ship,scq.id_cargo from
   ship_cargo_quantity scq) cargoes1,
  (select p3.id_ship from
   (select count(id_port) ports,id_ship from
    (select distinct p.id_port,p1.id_ship from
     ports_quantity p,
     (select s2.id_ship,s2.id_quantity from
      (select scq.id_ship,scq.id_quantity from
      ship_cargo_quantity scq) s2,
     (select s1.id_ship from
     (select scq.id_ship,sum(v.value) summ from
      validquantity v,ship_cargo_quantity scq
      where scq.id_quantity=v.idobject group by id_ship) s1,
     (select max(summ) as max from
      (select scq.id_ship,sum(v.value) summ from
       validquantity v,ship_cargo_quantity scq
       where scq.id_quantity=v.idobject group by id_ship) s) m
     where s1.summ=m.max) s3
    where s2.id_ship=s3.id_ship) p1
   where p.id_quantity=p1.id_quantity)
  group by id_ship) p3,
```

```
(select min(ports) ports from
 (select count(id_port) ports,id_ship from
  (select distinct p.id_port,p1.id_ship from
   ports_quantity p,
   (select s2.id_ship,s2.id_quantity from
    (select scq.id_ship,scq.id_quantity from
    ship_cargo_quantity scq) s2,
   (select s1.id_ship from
    (select scq.id_ship,sum(v.value) summ from
     validquantity v,ship_cargo_quantity scq
     where scq.id_quantity=v.idobject group by id_ship) s1,
    (select max(summ) as max from
     (select scq.id_ship,sum(v.value) summ from
      validquantity v,ship_cargo_quantity scq
      where scq.id_quantity=v.idobject group by id_ship) s) m
     where s1.summ=m.max) s3
    where s2.id_ship=s3.id_ship) p1
   where p.id_quantity=p1.id_quantity)
  group by id_ship)) p4
 where p3.ports=p4.ports) ships1
where cargoes1.id_ship=ships1.id_ship) cargoes2
where cargoes2.id_cargo=v.idobject
```

We will notice that for transformation of ODQL query to SQL_DIM-query it is possible to write transformation procedure. But it will make sense if performance time of such query is significantly better than performance time of ODQL query.

4.2.2. Choice of tests parameters for perform computing experiments

At a choice of computing experiments performing parameters it is necessary to allocate three levels of complexity of filling of tables: the minimum filling (each ship carry 1 freight to 1 port), average filling (each ship carry 20% of freights names to 20% of ports), the maximum filling (each ship carry freights of all names to all ports). Further when performing tests it is necessary to increase gradually entries number in the tables containing information on the ships, freights and ports.

4.2.3. Results of computing experiments and their analysis

For a start we will enter designations for the performed queries:

1. Query for a relational DB we will call «SQL».

2. ODQL-query to a DIM DB which is performed by means of technology an index-selections we will call «ODQL».

3. The same ODQL-query which is performed by means of SQL query to metalevel we will call «SQL_DIM».

Comparative tables of queries performance time for different complexity of tables filling DB can be found in [25]. By results of the made computing experiments it is possible to see that:

- for a case with the minimum filling the relation of queries average time of «SQL_DIM»\«SQL» makes 1.62, and «ODQL»\«SQL» — 3.06,
- with average filling of «SQL_DIM»\«SQL» — 2.1, «ODQL»\«SQL» — 22.05,
- with the maximum filling of «SQL_DIM»\«SQL» — 3.73, «ODQL»\«SQL» — 320

As time of «ODQL» grows catastrophically quickly, was decided to make repeated experiment for «SQL» and «SQL_DIM» with average filling.

In figure 2 is represented the graph on which it is possible to see time relation of the «SQL_DIM» to «SQL» (data are taken from tables in [25]). On abscissa axis the value equal to $\log_2 scp$, where s — number of the ships, c — freights quantity, p — ports number is specified. From testing results follows that «ODQL» significantly loses «SQL», but «SQL_DIM» more slowly on average 3-4 times, though has a similar tendency of query time growth. But nevertheless with increase in the tables size time growth is slowed down. It allows to make a choice for «SQL_DIM», but in spite of the fact that there is a loss on time, thus we get a number of advantages which to us are provided by ODQL DIM.

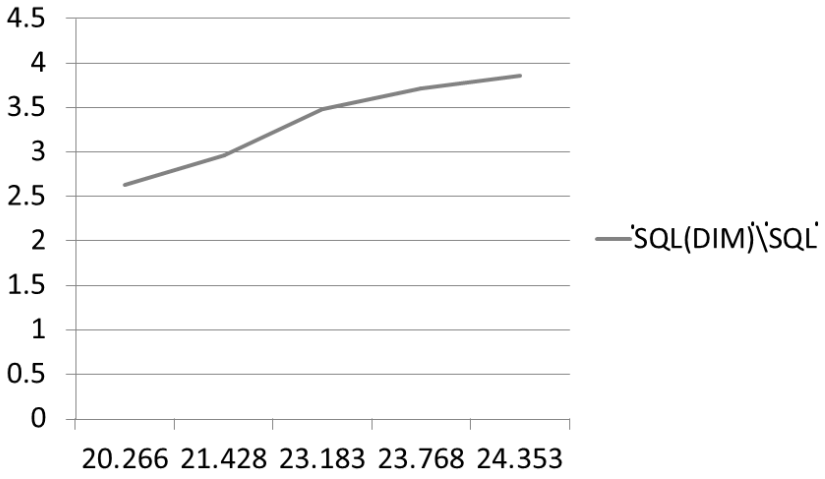

Fig. 2: Relation of queries time

5. Realization of system program components

5.1. DIM navigator

For interaction with metalevel it is necessary to create a program component which will allow to work with the tables presented in it. We will call its "DIM Navigator" [31]. It is also necessary to make so that this component was convenient to the user. For a start we will decide on the concept "user" there are 2 types of users which can interact with DIM

DBMS: "administrator" and "the ordinary user". The administrator can make changes to structure of the classes placed in DBMS, and the ordinary user can look through them only.

For editing the administrator needs to have possibility of creation, editing and removal of classes, communications between them, parameters and other structural elements. The ordinary user doesn't need to edit these data, but with DBMS he can be necessary for work to see data structure for what the navigator is also provided.

For work of the program with DIM DBMS there has to be a possibility of reading and record in a DB. That data could be transferred from one environment to another, it is necessary to create structure for interaction with DBMS. It will be in many respects similar to structure of metalevel.

Data storage is realized in lists, each of which corresponds to a certain table in metalevel. Each list is structured with the help classes according to fields of tables of the scheme of metalevel that allows to transfer data to a DB and to load information from it.

For convenience of the user in the interface presented the ordered structure in which it is possible to add, edit, delete classes (fig. 3), for each of them it is possible to adjust parameters which this class contains. Also there is a creation possibility of all communications types used in DIM DBMS, this inheritance, inclusion, a choice, history and interaction. At communications control between classes the check algorithm of their structure that allows to bypass logical mistakes which can arise at the user is used. It is possible to give a case with a wrong definition of incorrect inheritance as an example. In such situation the algorithm of check of communications which recursively checks interrelations of a class is used and in case of a logical mistake doesn't apply the changes made by the user. It allows to support structural integrity of lists, and respectively and tables. There is an opportunity in the presence of the rights to add and change objects of each of classes. The tree of classes objects is represented in fig. 4. Also at addition or data removal it is necessary to watch data integrity: definiteness restrictions, unambiguity and choice ([17],[18],[19],[20],[27],[28]).

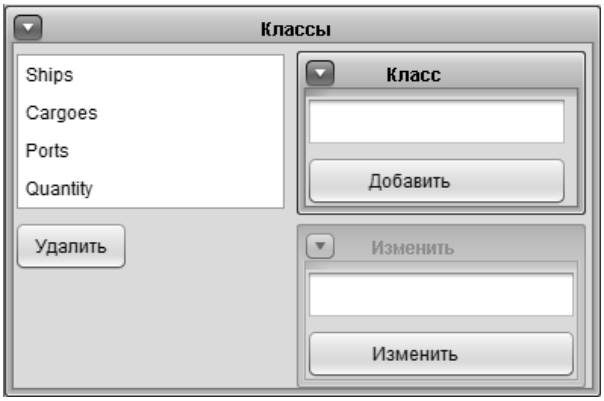

Fig. 3: DIM navigator, work with classes

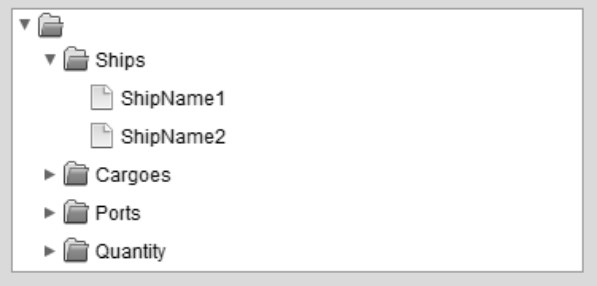

Fig. 4: DIM navigator, tree of classes objects

5.2. The generator of ODQL-queries

As drafting object query demands from the user of objects classes knowledge and their communications that not always he knows precisely, creation of intellectual system which will ease queries creation in object DIM DBMS is necessary. We will call this system "The Generator of ODQL-queries for DIM DBMS".

The generator represents a set of the components helping the user to orient visually in structure of a DB and to make query, using the interactive interface. For ensuring interactivity the system allowing to choose from the presented DB elements necessary for the user is used. At the initial stage the user chooses the necessary parameters, then specifies a class if such parameters meet at several classes. If necessary specifies conditions for this choice. Also the user can specify at this stage, whether performance of the conditions connected with other classes or parameters of the chosen class is necessary. If need is available, the user is offered to choose with what of classes or parameters he wants to connect query conditions, and the list of classes contains only what are connected with the class chosen at the moment. Thus, in system is realized possibility of the additional conditions indication taking into account interrelations between the DIM DBMS elements. At a stage of drawing up query it is controlled onto a correctness of required data therefore the problem is solved with creation of difficult queries. The example of a query generation parameters choice is presented in figure 5.

Development of the intellectual system allowing to make queries for DIM DBMS in a form, comfortable for the user, was result of work ([21],[22],[29]).

5.3. Converter

Transfer of the existing DB on DIM DBMS requires creation of the program which will be able to transform data from different types of DBMS, such as: temporal, relational, object-oriented and object-relational. Such transformation requires two stages. On the first the DB is converted in OD-model, then the model is converted in DIM.

Was developed the program "DIM DBMS Converter" which first stage of work is data transformation from a relational DB to OD-model.

The principle of converting in OD-model consists in data transfer from any DBMS in the general structure from which further there is a transformation to DIM DBMS. For compliance to structure of OD-model the converter possesses a set of arrays which emulate this model. Thus, when reading from DBMS all data are filtered at the program level and make OD-model (see section 2) though they aren't connected among themselves

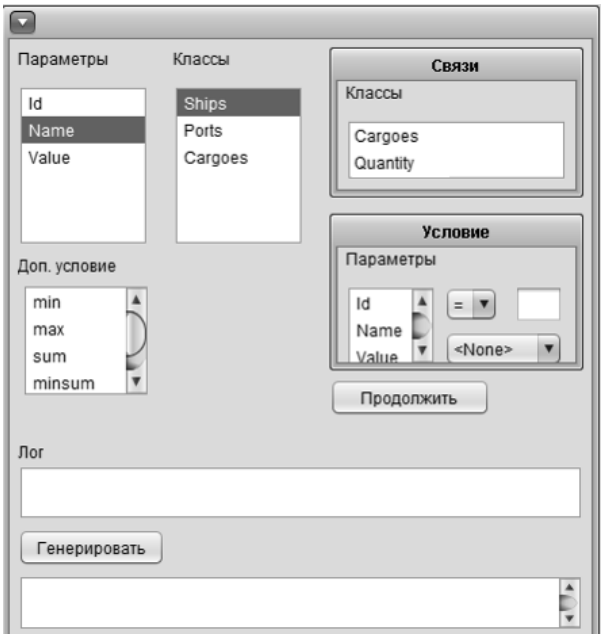

Fig. 5: Generator of ODQL-queries, example of a query generation parameters choice

yet. Further the algorithm on which these tables contact the relevant fields groups is used (as a rule, each group begins with the field Id). Further the program analyzes fields names regarding partial coincidence, and on the basis of it forms communications between the tables corresponding to this field. As in the existing DB there can be features of communications between tables, the user is given opportunity of preview and correction of OD-model elements. For convenience to the user the preliminary list of parameters, classes and communications between them (fig. 6) as will look received from the DB OD-model in DIM DBMS is displayed. There is a possibility of editing communications and other objects of model. If in the course of editing it is required to return to the initial version of the model received by the program processing of a DB, the user can cancel the made changes.

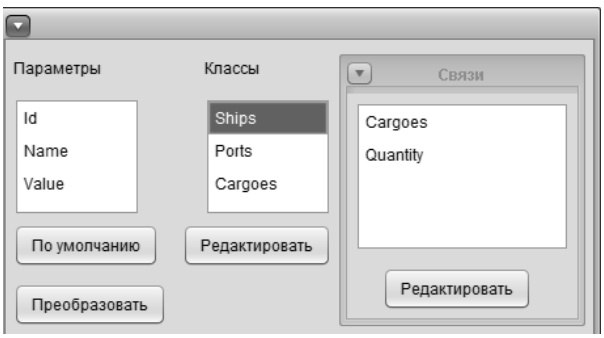

Fig. 6: DIM converter, application form

6. Conclusion

For access efficiency to data to DIM DBMS it is developed software "DIM DBMS Converter", "DIM Navigator", "The Generator of ODQL-queries" and the comparative analysis of query technologies for DIM [25] is performed. The developed algorithms are used in the created program for transformation of relational DB to DIM DBMS DB.

References

[1] Gray J., Liu D. T., Nieto-Santisteban M., Szalay A., Dewitt D. J., Heber G., "Scientific Data Management in the Coming Deacade", *SIGMOD Recjrd*, **34**, 2005.

[2] Sivtsov A., "Shest komponentov uspeshnykh proektov na primere DW/BV. Korporativnye bazy dannykh-2011", *Materialy 16-y ezhegodnoy tekhnicheskoy konferentsii*, 2011, (in Russian).

[3] Codd E. F., "A relational model for large shared data banks", *Comm. ACM*, **13**:6 (1970), 377–387.

[4] Codd E. F., *Further normalization of the database relational model, in Database Systems*, ed. R. Rustin, Prentice Hall, Englewood Cliffs, NJ, 1972.

[5] Atkinson M. i dr., *Manifest sistem obektno-orientirovannykh baz dannykh. SUBD*, 1995, (in Russian).

[6] Garsia-Molina G. et al., *Sistemy baz dannykh. Polny kurs*, Vilyams, 2003, (in Russian).

[7] Robert Greene, "OODBMS ARCHITECTURES. An examination of implementations", http://www.odbms.org/wp-content/uploads/2006/10/028.01-Greene-OODBMS-Architectures-September-2006.pdf.

[8] Kostenko B. B., Kuznetsov S. D., "Istoriya i aktualnye problemy temporalnykh baz dannykh", 2007, http://www.citforum.ru/database/articles/temporal, (in Russian).

[9] Pisarenko D. S., Roublev V. S., "Object DBMS DIM and its main concepts", *Modeling and analysis of information systems*, **16**:1 (2009), 60–87, (in Russian).

[10] Roublev V. S., "The Object Query Language of the Dynamic Information Model DIM", *Modeling and analysis of information systems*, **17**:3 (2010), 144–161, (in Russian).

[11] Rublev V. S., "Zaprosnaya polnota yazyka ODQL dinamicheskoy informatsionnoy modeli DIM", *Yaroslavsky pedagogichesky vestnik*, Fiziko-matematicheskie i estestvennye nauki, **1**, Yaroslavl, 2011, 69–75, (in Russian).

[12] Roublev V. S., "Object Query Computing Optimization in the Dynamic Information Model DIM", *Modeling and analysis of information systems*, **18**:2 (2011), 39–51, (in Russian).

[13] Deyt K. Dzh, Khyu Darven, "Osnovy budushchikh sistem baz dannykh: trety manifest", 2004, (in Russian).

[14] Rublev V. S., Kaybyshev A. Sh., "Organizatsiya khraneniya dannykh i vypolneniya zaprosov v dinamicheskoy informatsionnoy modeli DIM", *Yaroslavsky pedagogichesky vestnik*: *Estestvennye nauki*, **III**, YaGPU, Yaroslavl, 2012, 70, (in Russian).

[15] Jarke M., Koch J., "Query Optimization in Database Systems", *Computing Surveys*, **16**:2 (1984).

[16] Darvin Kh., Deyt K., "Trety manifest", *SUBD*, 1996, №1, 110–123, (in Russian).

[17] Yusupov A. R., "Yazyki obektno-dinamicheskikh zaprosov i reshenie problem dostupa k informatsii v Dinamicheskoy informatsionnoy modeli DIM", *Sovremennye problemy matematiki i informatiki*, **6** (2004), 157–163, (in Russian).

[18] Yusupov A. R., "Interfeys polzovatelya dinamicheskoy informatsionnoy modeli DIM i navigator obektov", *Diskretnye modeli v teorii upravlyayushchikh sistem. Tezisy dokladov V nauchnoy konferentsii*, MGU, M., 2003, 92–93, (in Russian).

[19] Lobachev D. I., "Realizatsiya prav dostupa k obektam dinamicheskoy informatsionnoy modeli", *Sovremennye problemy matematiki i informatiki: Sbornik nauchnykh trudov molodykh uchenykh, aspirantov i studentov*, **5**, Yarosl. gos. un-t, Yaroslavl, 2002, (in Russian).

[20] Lobachev D. I., "Organizatsiya dostupa k dannym dinamicheskoy informatsionnoy modeli", *Problemy teoreticheskoy kibernetiki: Tezisy dokladov XIII Mezhdunarodnoy konferentsii*, Moskva, 2002, 111, (in Russian).

[21] Antonov D. V., "Generatsiya ODQL-zaprosov dlya SUBD DIM", *Sbornik dokladov Mezhdunarodnoy konferentsii «II vesennie nauchnye chteniya» (17 maya 2014)*, Nauchno-informatsionny tsentr «Znanie», Donetsk, Ukraina, 2014, ISSN 6827–0151 (in Russian).

[22] Antonov D. V., "Zaprosy SUBD DIM i ikh generatsiya", Shestdesyat sedmaya regionalnaya nauchno-tekhnicheskaya konferentsiya studentov, magistrantov i aspirantov vysshikh uchebnykh zavedeniy s mezhdunarodnym uchastiem, **2** (2014), 284, (in Russian).

[23] Antonov D. V., Rublev V. S., "Sravnitelny analiz zaprosnykh tekhnology dlya skhem baz dannykh SUBD DIM", *Materialy XI mezhdunarodnogo seminara «Diskretnaya matematika i ee prilozheniya», posvyashchennogo 80-letiyu so dnya rozhdeniya akademika O.B.Lupanova (Moskva, MGU, 23 iyunya 2012 g.)*, MGU, Moskva, 2012, 321–323, (in Russian).

[24] Antonov D. V., Rublev V. S., "Sravnenie modeley zaprosov dlya skhem baz dannykh SUBD DIM", *Zametki po informatike i matematike: Sbornik nauchnykh statey*, **4**, Yaroslavl, 2012, 25, (in Russian).

[25] Antonov D. V., Rublev V. S., "Analiz tekhnologiy vychisleniya ODQL-zaprosov SUBD DIM", *Yaroslavsky pedagogichesky vestnik*, **3**, 2013, 93–97, (in Russian).

[26] Antonov D. V., Rublev V. S., "Comparison of query models for schemes of the DIM DBMS databases", *Science Drive-2012*, YarGU, Yaroslavl, 2013.

[27] Antonov D. V., "Navigator of the DIM DBMS databases", *Science Drive-2013*, YarGU, Yaroslavl, 2013.

[28] Antonov D. V., "Navigator DIM", *Zametki po informatike i matematike: Sbornik nauchnykh statey*, **5**, Yaroslavl, 2013, 10, (in Russian).

[29] Antonov D. V., "Generatsiya ODQL-zaprosov dlya SUBD DIM", *Molodaya nauka v klassicheskom universitete: Tezisy dokladov nauchnykh konferentsy festivalya studentov, aspirantov i molodykh uchenykh*, Ivanovo, 2014, Chast I (in Russian).

[30] Rublev V. S., "Teorema o staticheskoy polnote SUBD DIM", *Problemy teoreticheskoy kibernetiki. Materialy XVII mezhdunarodnoy konferentsii (Kazan, 16–20 iyunya 2014g.)*, Otechestvo, Kazan, 2014, 242–245, (in Russian).

[31] Antonov D. V., "Navigator dlya SUBD DIM", Svidetelstvo o gosudarstvennoy registratsii programmy dlya EVM № 2014618475, (in Russian).

Эффективность доступа к данным в СУБД DIM

Антонов Д.В., Рублев В.С.

Ярославский государственный университет им. П. Г. Демидова
150000 Россия, г. Ярославль, ул. Советская, 14

Ключевые слова: СУБД, РСУБД, ООСУБД, DIM, дискретные детерминированные модели, OD-модель, запросные технологии, ODQL-запросы, конвертирование БД

В статье приводится обзор средств, используемых в объектной СУБД нового типа для повышения эффективности доступа к данным. Описываются особенности объектной СУБД DIM, основанные на использовании отношений классов объектов (как множеств объектов): наследования, включения, взаимодействия и истории – и отношений объектов: наследования, внутреннего наследования, включения, внутреннего включения, взаимодействия и истории. Вводится описание предметной области при помощи объектно-динамической модели данных (OD-модели) и обосновывается полнота СУБД DIM для произвольной OD-модели. Описывается объектный язык запросов ODQL, позволяющий совместить сложность точного описания с простотой использования за счет введения двух уровней запросов. В целях выяснения наиболее эффективного способа обращения к СУБД DIM проводится исследование различных запросных технологий для этой среды, а также разрабатываются и реализуются механизмы для работы пользователей с ней. Для этого разрабатывается комплекс программных средств, необходимых для работы с СУБД DIM. Приводится описание основных концепций разработки ПО «Навигатор DIM», необходимого для возможности манипулирования данными в имеющейся БД посредством графического интерфейса. Рассматривается разработка ПО «Генератор ODQL-запросов», который нужен для упрощения построения запросов к СУБД DIM без необходимости для пользователя в обязательном порядке знать синтаксис нового языка запросов. Рассматриваются пути решения проблемы конвертации данных из существующих СУБД в СУБД DIM.

Статья публикуется в авторской редакции.

Сведения об авторе:
Антонов Дмитрий Владимирович,
Ярославский государственный университет им. П.Г. Демидова,
выпускник аспирантуры,
orcid.org/0000-0002-8465-6714
Рублев Вадим Сергеевич,
Ярославский государственный университет им. П.Г. Демидова,
канд. физ.-мат. наук, профессор,
orcid.org/0000-0001-8095-3132