

Improving Controllability of Vast Conceptual Models

Vladimir Ovchinnikov (ovch@lipetsk.ru)

High-quality designing of present-day information systems is not possible without using conceptual modeling techniques. They allow to decrease complexity of processed information noticeably, and to focus on key aspects of a system being created. But now complexity of systems is so high that their representation as a conceptual model has high complexity too.

There is several approaches to control of conceptual model complexity, for example, adding abstraction layers. In this paper we suggest an original method that may be used along with existing ones. Essence of the method lies in not using self-dependent identification of relations. An object type set underlying a certain relation is suggested to be used as its identifier.

The article considers the consequences of this step: simplification of conceptual model analysis, completeness of relation's semantics; it is shown possibility of development of a conceptual query language expressions of which do not refer to relations in an explicit way, are compact, and, at the same time, have structure that is very close to phrases of natural language (without any artificial additions). The paper gives prove that the suggested approach to control of conceptual model complexity does not result in significant effort increasing in the course of conceptual designing, and that it is applicable in all the cases in which other conceptual techniques are applicable.

Keywords: conceptual model, conceptual model complexity, object-role model, conceptual query, semantically complete model

1. Introduction

Developing of up-to-date information systems requires ample quantity of specialists of different professions. Project participants' grounding levels in the field of information system designing may differ greatly. It is necessary a mean that allows discuss common problems regardless of skills in database projection. As such a mean conceptual models are used. Being intuitively clear after relatively short familiarization, they permit to establish a productive contact among project members quickly.

One of the most advanced conceptual modeling techniques is recognized to be object-role modeling technique (ORM) [3, 4, 11, 13, 20, 23, 24, 25]. We do not dwell on the technique as there is sufficient quantity of works on this topic (for instance, [4] and [8]). Let us only note that we will further use the concept "relation" as the generalization of object-role model's concepts "fact type", "specialization type", and "generalization type".

For conceptual modeling techniques the conceptualization principle holds true [14, 18]. According to it a conceptual model should contain only Universe of Discourse (UoD) description without implementation aspects. Absence of implementation aspects conditions relative simplicity and controllability of conceptual models, constraints, and queries.

Nevertheless, a description of large information system, even as a conceptual model, still remains complex one for all project participants regardless of skill level. Controlling of complex conceptual models is one of tasks being solved in investigations of conceptual model abstracting [6, 8, 9, 10, 22, 26]. The abstracting essence lies in adding abstraction layers into an initial high-detailed conceptual model for increasing its controllability and clearness. Abstracting realizes the strategy "divide and conquer" with respect to complex conceptual models. In this paper an alternative approach to simplification and increase of conceptual model controllability will be suggested.

2. Problem Statement

By investigating how one (a professional designer or an unskilled man) creates and studies conceptual models and queries we found three circumstances:

- P1: being of alternative relations (based on the same object types) provokes appreciable complication; for one who has no experience in projection of databases it is not evident how two object types may be simultaneously connected by two (or more) different ways;
- P2: queries, as a rule, are initially formulated by designer as a text without using relation designations; for translation of a query into formal form it is necessary to memorize designations of all essential relations;
- P3: memorizing relation designations requires significant effort; it partially diverts from essence of Universe of Discourse being modeled (what is more, designations are easily forgettable).

A method of solving P3 (only for query formulation) was suggested in works [1, 2, 5]. They suggest an interactive method of query building:

- on the first step a user selects an interesting object type;
- the system of interactive query building offers to choose one or several roles in which the given object type participates; all object types that are included into a fact type together with chosen role become subordinate to the initial object type;
- the operation of role choice may be iteratively repeated for any subordinate object type.

Acquired access path (in the form of tree) is used for joining appropriate tables. The system allows filter resulting join on any object type. In the papers [16, 17] a similar approach is suggested with the following difference: several linear access paths are initially created and are further combined to more complex query.

Both methods permit query building for one who is not familiar with a conceptual model, and do not require to memorize relation designations. They are convenient for users of conceptual model. But a designer, modifying and checking up a conceptual model, have to remember all relations of model part that he is processing now. Therefore these methods are convenient for a designer only when he stands as a user of finished conceptual model.

Consider the process of model analysis during its familiarization or anamnesis by a man. In the course of analysis a man thinks by means of object types (terms) until he meets alternative relations. Indeed, if one works on the fact type “a material piece was produced by a unit”, he simply ascertains that the object types “material piece” and “unit” are connected. But if that model additionally contains the alternative fact type “a material piece was consumed by a unit”, then the picture becomes more complicated. Firstly, a man has to attribute (in his mind) the object types “material piece” and “unit” to category of complex-connected object types. Secondly, he has to memorize both connections as separate concepts. In this situation it is not enough to be familiar with object types, for correct using of model it is necessary to manage structure and semantics of relations.

Alternative relations add an extra degree of freedom. A man not only notes a fact of connectivity, but also forms additional concepts (in his mind) for each alternative relation. Formed concepts are not reflected among object types of model and have rather non-trivial semantics. The process of analysis of alternative relations requires certain skills in designing information systems and does not come easily to novices.

Complexity of alternative relations studying becomes also apparent from their conditional character. Considering a certain relation, no one can say that it is “completed” interconnection description for object types included in it. In other words, for understanding essence of object types interconnection it is necessary to know semantics of a given relation and aim of its creation. This knowledge is not contained in a model itself and is its environment, condition of its creation. For example, we should know that the object types “material piece” and “unit” are connected by two different ways, and that the semantics of the first connection is “produces” and of the second – “consumes”.

During study of model, even if one knows that a certain relation is not alternative, one can not be sure that the relation will not become alternative as a result of model evolution in the future.

Therefore, a student has to form an additional concept for each relation regardless of its alternativeness at the moment. He has to analyze a model in two dimensions: as a set of connected object types and as a set of relations having its own designation and semantics.

As we note before, a man tend towards thinking about Universe of Discourse and formulating queries without using relation designations. In the course of query formulation a designer has to recall designations. If a conceptual model is of large size, this work can not be done in mind without using documentation. As a result, productivity decreases, attention is diverted from Universe of Discourse essence. After formalization completion, relation designations become unnecessary and are soon forgotten. When one returns to work after some interruption, many things should be recalled anew.

These complexities are seemed to be unavoidable. Are not we ever going to forbid creation of alternative relations in a model?.. In this paper we make an attempt to put such a veto and analyze its consequences.

The article aims at further development of conceptual model and queries controllability, and represents a try to soften enumerated complexities of existent conceptual modeling techniques. A method suggested here affects both aspects (model and queries) and may be applied along with abstracting approaches. It may be considered as modification of one of existent approaches to conceptual modeling (for instance, object-role or set-theoretic) or as self-dependent method beyond context of other approaches. Examples in the article, for better clearness, will be made on the basis of object-role model. For deep understanding of the paper it is necessary to be familiar with ORM notation.

In the following section the idea of suggested method is present. Then its practical application is proposed to your attention. In the conclusion we accent your attention at main properties of the suggestion and consider possible ways of further investigations.

3. Idea of Proposed Method

The idea of the method lies in refusing self-dependent identification of relations. We suggest to use a set of object types underlying a relation as its identifier. In this case a relation, in a certain sense, loses its "independence" and becomes a set of object types. Such a way of posing inevitably causes the questions:

- May all UoDs be described with a model constrained such a way?
- Is this step the reason of effort increase in projecting and using a model?
- Does this step harm intuitive clearness of conceptual model?
- Does it result in model unhandiness and other inconveniences in its projection and usage?

Let us discuss these questions later, after more detailed describing of proposal's essence.

Identification of relation by means of object type set implies two constraints:

C1: a relation can not be based on the same object type more than once since, in that case, we attain to relation identification through multi-set of object types, and not through set;

C2: a model can not contain alternative relations.

Relations are considered to be alternative in the narrow sense if they are based on the same set of object types. In this case, as we can see, relations have the same identifier (a set of object types), and so they can not exist in a model simultaneously.

But in C2 we intend the alternativeness in the wide sense: relations are considered to be alternative if they are based on object type sets included one into another. Here we widen the concept of alternativeness (and thus thoughen the constraint C2), aiming at simplification of model study process. If one knows that a model being under study does not contain alternative relations in the wide sense then he may be sure that each relation represents complete knowledge about interconnection of object types included in it. He has not to begin study of object type interconnection each time anew when he meets just another alternative relation (we consider the case of complex and vast model that can not be grasped at first sight). The examples of inadmissible relations are shown in the figure 1:

- a) a relation based on an object type more than once;
- b) relations based on the same set of object types (alternativeness in the narrow sense);
- c) relations based on object type sets included one into another (alternativeness in the wide sense).

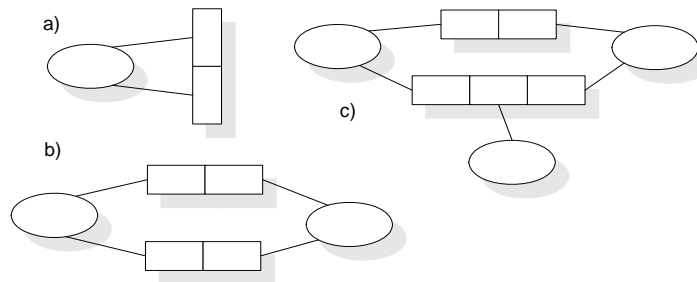


Figure 1 Examples of relations that are inadmissible within the suggested approach

Let us consider nascent anxieties about vitality of the proposed approach. The first anxiety: introduction of such tough constraints might involve impossibility of conceptual modeling for quite a number of information systems. This anxiety is refuted if all the cases shown in the figure 1 could be modeled in the framework of suggested approach. For that, modify each case by way of classifying one of object types with the help of inheritance mechanism. As a result, the introduced constraints cease to infringe (figure 2). Consequently, we conclude that the first anxiety is refuted and any UoD may be modeled without violating the introduced constraints.

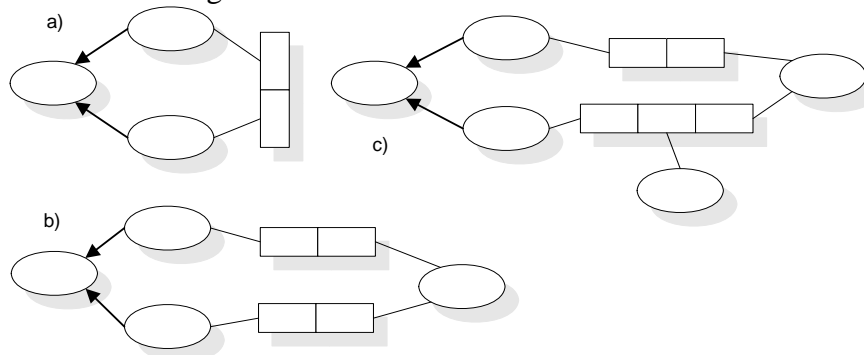


Figure 2 Transformation of inadmissible relations to admissible ones

The second anxiety: the introduced constraints might involve effort increase, clearness loss, or other inconveniences of this approach. Indeed, additional effort are necessary for classifying object types, but they are little. At that, as it will be shown further, clearness of model is increased, and model study process is simplified.

When one works with a model satisfying the introduced constraints, he needs not memorize relation designations. It is sufficient to ascertain connectivity of object types included in a relation. A student may constantly be within the plane of object types, that significantly simplifies his task in the case of complex model. Since a man tends to think by means of object types (in particular, non-specialist in the field of information system design), he needs not turn his attention to relation designations.

If a model satisfies to the constraint C2 (i.e. it has no alternative relations in the wide sense), then one may consider each relation as complete description of object type connection. For each arbitrary small part of a model one may be sure that during the next study he has not to change his view about connection of object types contained in the part being under study now (certainly, on the assumption of the model does not evolve). The property of semantic stability for any part of a model involves additional simplification and controllability for conceptual modeling. The strategy “divide and conquer” develops further.

Existent conceptual languages do not allow to formulate expressions based on object types only [1, 2, 14, 15]. For instance, to formulate path expression in LISA-D [14, 15] it is necessary to point at used relations explicitly. This property results from admissibility of alternative relations in object-role model.

Identification of relations by means of object types permits to develop a query language that does not use relations’ designations for referring to them. In the context of the suggested approach a designer or a user may formulate a query using object types only. Which relation should be used in a

certain case is automatically determined by a system on the basis of simple rules. Illustrative explanation of these rules will be given below when an example of modeling will be considered.

During initial formulating of query on natural language a designer, as a rule, does not use relations. Within this approach, if projecting goes adequately, a query formulation on natural language may practically be the same as a formal query. It will be shown below when an example of modeling will be considered. As a result, we achieve simplicity and controllability increase for conceptual queries.

We believe that the enumerated features of the suggested model allow us to name it as “semantically complete”. Perhaps this name is partially loud, but it conforms with essence of the model since each relation has a complete semantics: the model can not simultaneously contain several relations setting interconnection of the same object types in different ways.

4. Use Case

As an example consider MES (Manufacturing Execution System) as applied to plate rolling, namely, the part of the system that makes tracing of actual production. Peculiarity of metallurgic manufacture is that making a final product can not be represented as hierarchical BOM (Bill of Materials) since during processing metal can be cut or welded in an arbitrary way. The key requirement to tracing, having made to systems of such a class, is the following: a system should contain all data about all existent material pieces (row materials, intermediate products, or final products) and complete history of their processing. The ORM diagram shown in the figure 3 contains the part of conceptual model that is responsible for compliance with the given requirement.

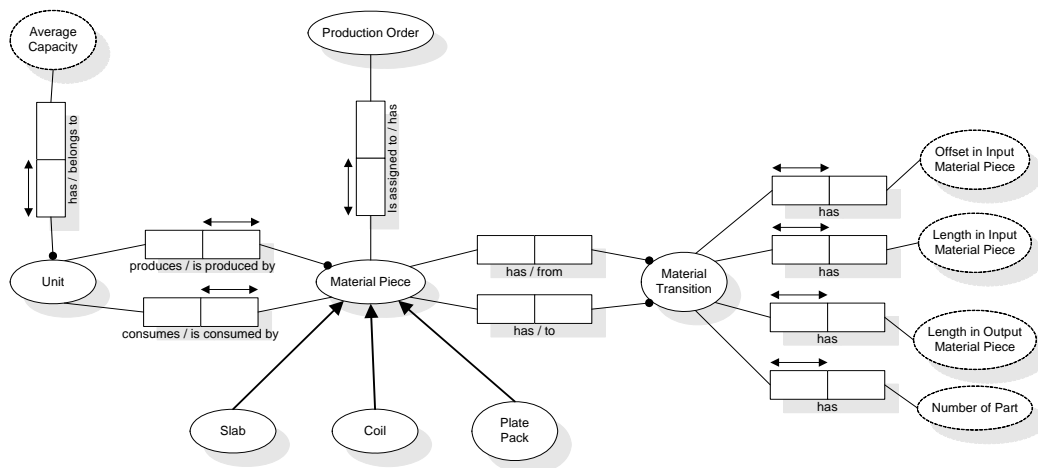


Figure 3 ORM diagram of material piece transformation

On the diagram the object types “Material Piece” and “Material Transition” reflect the key process of material transformation. While producing, slabs may be cut, coils may be cut or welded, and packs of plates may be formed from different coils as the figure 4 shows. Operations of transformation are executed on units, one material piece must be produced on an only unit and may be consumed for producing other material pieces on an only unit too.

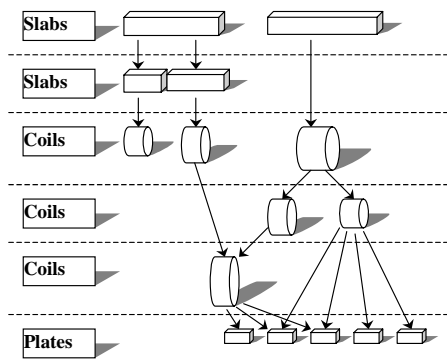


Figure 4 Transformation of material pieces during processing

The shown ORM diagram contains two alternative relations based on the object types “Unit” and “Material Piece” and two alternative relations based on “Material Piece” and “Material Transition”. The complexity for non-specialists in the field of database projection is that it is not sufficient to say “the object types “Unit” and “Material Piece” are connected”. It is necessary to realize that these object types are connected by two different ways. If we talk about unit’s production, we use the relation “produces”, if we talk about unit’s consumption, we use the relation “consumes”.

Furthermore, for all the rest relations we can not also be sure that they have not (and will not have in the future) alternative relations. We have to remember not only a fact of connectivity of the object types “Material Piece” and “Production Order”, but also to create a separate cell (in our mind) for this relation, to give it our internal unique tag (designation, name).

Let us transform the above ORM diagram to the form without alternative relations (figure 5). For alternative relations based on object types “Unit” and “Material Piece” do the following. Introduce two subtypes “Produced Material Piece” and “Consumed Material Piece” for the object type “Material Piece”. Then, recreate both alternative relations by building it on subtypes of the object type “Material Piece”. As a result, the object types implied in the figure 3 become apparent in the figure 5. Do the same thing for the object type “Material Transition”. The obtained diagram does not contain alternative relations, and therefore, one may analyze it in the plane of object types only. Relations have not self-dependent semantics any more and hold information about connectivity of object types only. In one’s perception they are completely identified by object types constituting them without additional tags.

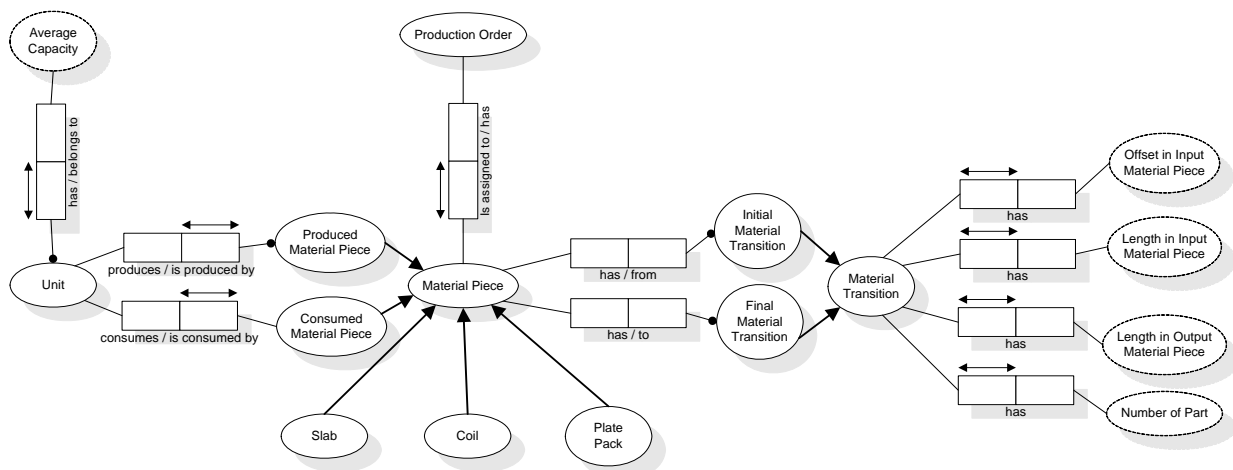


Figure 5 ORM diagram of material piece transformation without alternative relations

Within the proposed approach, during working with relations, one may be sure that each relation completely describes interconnection of object types. Considering the relation on the object types “Production Order” and “Material Piece” separately, we may assert that their interconnection does not depend on other object types or outer (with respect to the model) conditions. These object types are *always* connected in the binary way 1-M (certainly, until the model will be changed). A

student have the full right to count on that there is no a diagram of the same model introducing a new relation on the same object types, and that it will not require to reconstruct the formed view about the interconnection of “Production Order” and “Material Piece” time and again.

Further we consider an example of query to this model: “calculate average proportion of outcome to income for each unit”. Let us reformulate this query to more detailed form: “for each unit calculate a ratio of summary mass of produced material pieces to summary mass of consumed material pieces”. As we can see, the verbal formulation of the query does not refer to relations. And what is more, the verbal formulation is only based on concepts being presented at the model by way of object types. This property allows, within the suggested method, to formulate queries without referring to relations’ designations. For instance, in the following way:

```
((Unit-Produced Material Piece-Material Piece-Mass(1)).(Unit, SUM(Mass(1))),
 (Unit-Consumed Material Piece-Material Piece-Mass(2)).(Unit, SUM(Mass(2))))
(Unit, SUM(Mass(1))/SUM(Mass(2)))
```

Consider syntax and execution of the proposed query in detail. The symbol ‘-’ is used for selecting content of a binary relation based on object types gathered round ‘-’. For example, (Unit-Produced Material Piece) is the selection from the relation based on the object types “Unit” and “Produced Material Piece”.

A chain of object types separated by ‘-’ designates a composition of appropriate relations. (Unit-Produced Material Piece-Material Piece) is a composition of two relations: (Unit-Produced Material Piece) and (Produced Material Piece-Material Piece). If we draw an analogy with SQL, this query may be written as follows:

```
SELECT Unit, Produced Material Piece, Material Piece
FROM (Unit, Produced Material Piece) a,(Produced Material Piece, Material Piece) b
WHERE a. Produced Material Piece = b. Produced Material Piece
```

Here, enumerations of object types is used for designating the relations, for instance, (Unit, Produced Material Piece), rather than proper designations of relations.

If a query uses one object type at several roles, a numeric modifier is applied. At that, one object type with different modifiers is considered to be different in the context of the query. For example, (Material Piece-Mass(1)) and (Material Piece-Mass(2)) are selections from the same relation (Material Piece, Mass), but Mass(1) and Mass(2) are considered independently. In this example, we introduce a modifier for the object type “Mass” since mass is used in two contexts simultaneously: for produced and consumed material pieces.

The symbol ‘.’ denotes projection of relation shown at its left on object types enumerated at its right. For instance, (Unit-Produced Material Piece-Material Piece-Mass).(Unit, Mass) means projection of the composition (Unit-Produced Material Piece-Material Piece-Mass) on the object types “Unit” and “Mass”.

Use of aggregate functions during selection from a certain relation (apparently existent in a model or calculated one) means grouping of all object types not used in these functions. For example, (Unit-Produced Material Piece-Material Piece-Mass).(Unit, SUM(Mass)) is calculation of summary mass of produced material pieces for each unit. If we draw an analogy with SQL, this query may be formulated as:

```
SELECT Unit, SUM(Mass)
FROM (Unit, Produced Material Piece) a,(Produced Material Piece, Material Piece)
b, (Material Piece, Mass) c
WHERE a. Produced Material Piece = b. Produced Material Piece
AND b. Material Piece = c. Material Piece
GROUP BY Unit
```

For denoting a composition of relations may be used an comma-separated enumeration of appropriate relations. This is a single method of composition denoting for relations that are not binary. For binary relations this method is applicable too. For instance, the composition (Unit-Produced Material Piece-Material Piece) may be equally represented as ((Unit, Produced Material Piece),

(Produced Material Piece, Material Piece)). In the concerned query this method is used for complex calculable relations:

```
((Unit-Produced Material Piece-Material Piece-Mass(1)).(Unit, SUM(Mass(1))),
 (Unit-Consumed Material Piece-Material Piece-Mass(2)).(Unit, SUM(Mass(2))))
```

As we can see, the result of the latter composition is the relation based on “Unit”, “SUM(Mass(1))”, and “SUM(Mass(2))”. The final stage of executing the concerned query is calculation of “SUM(Mass(1))/SUM(Mass(2))” and further projection on unit and “SUM(Mass(1))/SUM(Mass(2))”.

The initial query formalization does not contain references to relations’ designations, object types are only used. Let us read the suggested formalization as a whole (in the reverse order): “for each unit calculate a ratio of summary mass of material units produced on a given unit to summary mass of material units consumed on a given unit”. This verbalization is practically the same as the initial verbalization (to phrase “on a given unit”).

We believe that nearness of a query’s verbalization and formalization is the most important property of the proposed approach. If one thinks and formalizes his thoughts by means of the same concepts, he needs not map certain concepts to others. Quality and productivity of designing increase.

Why do we suggest an our method of query formalization that rather differs from existent ones? Let us repeat the query formalization shown in the very beginning:

```
((Unit-Produced Material Piece-Material Piece-Mass(1)).(Unit, SUM(Mass(1))),
 (Unit-Consumed Material Piece-Material Piece-Mass(2)).(Unit, SUM(Mass(2))))).
 (Unit, SUM(Mass(1))/SUM(Mass(2)))
```

And now let us formulate the same query on a SQL-like language:

```
SELECT u1. Unit, u1.m1/u2.m2
FROM
 (SELECT Unit, SUM(Mass) m1
 FROM (Unit, Produced Material Piece) a, (Produced Material Piece, Material Piece)
 b, (Material Piece, Mass) c
 WHERE a. Produced Material Piece = b. Produced Material Piece
 AND b. Material Piece = c. Material Piece
 GROUP BY Unit) u1,
 (SELECT Unit, SUM(Mass) m1
 FROM (Unit, Consumed Material Piece) a, (Consumed Material Piece, Material Piece)
 b, (Material Piece, Mass) c
 WHERE a. Consumed Material Piece = b. Consumed Material Piece
 AND b. Material Piece = c. Material Piece
 GROUP BY Unit) u2
 WHERE u1. Unit = u2. Unit
```

Comparing these two formalization methods, we see that instead of three lines we have got fifteen. Absence of necessity to refer to relations’ designations allows to create a new query language by using of which queries will be formulated in the way that is very close to a natural language, and laconic without clearness loss.

Certainly, an expression of propositional logic may be written in the more compact form, but this compactness is attained due to abbreviation usage and not structural simplification. If we use abbreviations, the proposed query formalization becomes simpler and more compact than an appropriate expression of propositional logic. For example, introduce the following abbreviations: U – Unit, PMP – Produced Material Piece, CMP – Consumed Material Piece, MP – Material Piece, M – Mass. Then the concerned query would be written as:

$$\left((U - PMP - MP - M(1))(U, SUM(M(1))), \right) \left(U, \frac{SUM(M(1))}{SUM(M(2))} \right)$$

Formalizing this query as an expression of propositional logic, we acquire the following:

$$f(u \in U) \equiv \frac{\left(\sum_{\forall pmp \in PMP \forall mp \in MP \forall m \in M \{ (u, pmp) \in (U, PMP) \wedge (pmp, mp) \in (PMP, MP) \wedge (mp, m) \in (MP, M) \}} m \right)}{\left(\sum_{\forall cmp \in CMP \forall mp \in MP \forall m \in M \{ (u, cmp) \in (U, CMP) \wedge (cmp, mp) \in (CMP, MP) \wedge (mp, m) \in (MP, M) \}} m \right)}$$

Comparing these two formalization ways, we can see that the expression of propositional logic has more complex structure (though it seems to be more compact due to smaller font size). As applied to the proposed model, an expression of propositional logic may be partially simplified if we introduce the following rule: variables should be named the same way as appropriate sets (perhaps, with using numeric indexes, if it is necessary). In this case the notation (u, cmp) may be considered as membership checking of the combination of variables “u” and “cmp” to the appropriate relation: $(u,cmp) \in (U,CMP)$. Then the simpler form of the expression is attained:

$$f(u \in U) \equiv \frac{\left(\sum_{\forall pmp \in PMP \forall mp \in MP \forall m \in M \{ (u, pmp) \wedge (pmp, mp) \wedge (mp, m) \}} m \right)}{\left(\sum_{\forall cmp \in CMP \forall mp \in MP \forall m \in M \{ (u, cmp) \wedge (cmp, mp) \wedge (mp, m) \}} m \right)}$$

Note that the abbreviation usage causes query clearness decreasing and moves it away from its verbalization on natural language. Therefore, we stand up for formalization method suggested in the very beginning.

If we would not consider advantages of the proposed approach for query formulation, this paper were not complete. But consideration of the suggested query language in detail and all questions concerned it exceed the bounds of this article.

5. Summary

Thus, we proposed a method of simplification and controllability increase of conceptual model. It may be used along with existent approaches (for instance, with abstracting). Essence of the suggestion lie in definition of additional property for relations: to be identified by means of a set of object types. For that two constraints are imposed on conceptual model:

- Each relation may contain a certain object type only once.
- No model can contain two or more alternative relations simultaneously. Relations are considered to be alternative if they are based on object type sets included one to another.

A model, satisfying the enumerated constraints, has the following properties:

- When one works with a model’s relations, he needs not memorize and use their designations. For each relation it is sufficient to know a set of underlying object types.
- Analysis of a model may be exclusively fulfilled in the plane of object types. For any connected object types it is sufficient to ascertain a fact of their connectivity. More complex propositions about connectivity of object types are not necessary.
- Each relation has complete semantics within a model. During familiarization a certain relation one may be sure that another diagram of the same model has no another relation describing interconnection of object types in a different way. One may study a model in a consecutive manner without rebuilding of formed view.
- For such a model it can be developed a query language expressions of which do not apparently refer to relations. Relations that should be used are determined on the basis of object types that are selected. Let us bring other important properties of such a query language:
 - o structure and way of writing of a formalized query are close to its natural verbal formulation (i.e. without artificial usage of relations’ designations);
 - o a formalized query is compact, and what is more object types are designated by means of the same word-combinations as in verbal formulation.
- Expressions of propositional logic as applied to this model may be written in a simpler way.

We named the suggested model as semantically complete one since relations of it have complete semantics. The set-theoretic formalization of the proposed approach was accomplished in [21]. In that paper semantically complete model was introduced in a self-dependent way, and not as an extension of any other conceptual modeling technique.

6. Further Work

As directions of further work we may note the following. The proposed method may be used as a basis for creation of data access interface without apparent relation pointing. Partial solution of this task was accomplished within relational model in the course of working up the concept of universal relation [7, 19]. There it is suggested to use a predefined join of relations as a data access interface. Within that approach it is not possible to build queries that need to use a relation join not participating in a created universal relation. But such a query is inevitable necessary when an initial relational model has relations that are connected in the cyclic way.

Therefore, we propose another approach to organizing a data access interface: as two levels. The first level is a relational model, the second one is a semantically complete model used as superstructure of relational one. In this case relations of relational model (tables of relational scheme) are used for storing semantically complete relations. At that one table may store several semantically complete relations, that brings storing efficiency; and data access is fulfilled through semantically complete model exclusively by means of object types (without apparent usage of relations). This method allows to avoid restrictions of existent approaches since any join, any query may be formulated without using relations. Evidently, before realization of such a data access interface it is necessary to develop the query language discussed above.

Note that properties of introduced model and query language permit to use their as unified tool for data and knowledge representation stored with different formats. Knowledge, to a greater extent than data, need to be processed without using relations. Thinking in the plane of concepts (object types) is natural for a man. And what is more, closeness of verbal formulation of query and its formalization allows to take further steps in developing an algorithm of mapping a verbal query (formulated neatly enough) to formal one.

But all these questions are topics of separate papers.

7. References

1. Bloesch A.C., Halpin T.A. ConQuer: A Conceptual Query Language // Proceedings of ER'96: 15-th International Conference on Conceptual Modeling, LNCS, no. 1157, pp. 121-133.
2. Bloesch A.C., Halpin T.A. Conceptual Queries using ConQuer-II // Proceedings of ER'97: 16-th International Conference on Conceptual Modeling.
3. van Bommel P., ter Hofstede A.H.M., van der Weide. Semantics and verification of object-role models // Information Systems, 16(5), 1991, pp.471-495.
4. Bronts G.H.W.M., Brouwer S.J., Martens C.L.J., Proper H.A. A Unifying Object Role Modelling Approach. Information Systems, 20(3), 1995, pp. 213-235.
5. Burgers C.A.J., Proper H.A., van der Weide Th.P. An Information System Organized as Stratified Hypermedia // In Proceedings of CISM094, International Conference on Information Systems and Management of Data, Madras, India, 1994, pp. 159-183.
6. Campbell L.J., Halpin T.A., Proper H.A. Conceptual Schemas with Abstractions – Making flat conceptual schemas more comprehensible // Data & Knowledge Engineering, 20(1), 1996, pp. 39-85.
7. Carlson C.R., Kaplan R.S. A Generalized Access Path Model and Its Application to a Relational Data Base System // Proceedings of ACM SIGMOD International Conference on Management of Data. Washington, D.C., 1976.
8. Creasy P.N., Proper H.A. A Generic Model for 3-Dimensional Conceptual Modelling // Data & Knowledge Engineering, 20(2), 1996, pp. 119-162.
9. Feldman P., Miller D. Entity Model Clustering: Structuring a Data Model by Abstractions // The Computer Journal, 29(4), 1986, pp. 348-360.
10. Francalanci C., Pernici B. Abstraction Levels for Entity-Relationship Schemas // Proceedings of the 4-th International Conference CAiSE'92 on Advanced Information Systems Engineering, vol. 593 of LNCS, Manchester, United Kingdom, 1992, pp. 456-473.

11. Halpin T.A., Orłowska M.E. Fact-Oriented Modelling for Data Analysis // *Journal of Information Systems*, 2(2), 1992, pp. 1-23.
12. Halpin T.A. *Conceptual Schema and Relational Database Design*. Prentice-Hall, Sydney, Australia, 2nd edition, 1995.
13. ter Hofstede A.H.M., van der Weide. Expressiveness in conceptual data modeling // *Data & Knowledge Engineering*, 10(1), 1993, pp. 65-100.
14. ter Hofstede A.H.M., Proper H.A., van der Weide. Formal Definition of a Conceptual Language for the Description and Manipulation of Information Models // *Information Systems*, 18(7), 1993, pp. 489-523.
15. ter Hofstede A.H.M., Proper H.A., van der Weide. A Conceptual Language for the Description and Manipulation of Complex Information Models. In *Seventeenth Annual Computer Science Conference*, vol. 16 of *Australian Computer Science Communications*, 1994, pp. 157-167.
16. ter Hofstede A.H.M., Proper H.A., van der Weide Th.P. Computer Supported Query Formulation in an Evolving Context // In *Proceedings of ADC'95, 6-th Australian Database Conference*, vol. 17(2) of *Australian Computer Science Communications*, Adelaide, Australia, 1995, pp. 188-202.
17. ter Hofstede A.H.M., Proper H.A., van der Weide. Query Formulation as an Information Retrieval Problem // *The Computer Journal*, 39, 1996, pp. 255-274.
18. van Griethuysen J.J., editor. *Concepts and Terminology for the Conceptual Scheme and the Information Base*. Publ. nr. ISO/TC97/SC5-N695, 1982.
19. Kent W. Consequences of Assuming a Universal Relation // *ACM TODS*, 6(4), 1981.
20. Nijssen G.M., Halpin T.A. *Conceptual Schema and Relational Database Design: a fact oriented approach*. Prentice-Hall, Sydney, Australia, 1989.
21. Ovchinnikov V.V. A Conceptual Modeling Technique without Redundant Structural Elements // *Journal of Conceptual Modeling* (www.inconcept.com/jcm), 29, 2003.
22. Seltviet A. H. An Abstraction-Based Approach to Large-Scale Information System Development // *Proceedings of the 5-th International Conference CAiSE'93 on Advanced Information Systems Engineering*, vol. 685 of *LNCS*, Paris, France, 1993.
23. Shoval P., Zohn S. Binary-relationship integration methodology // *Data & Knowledge Engineering*, 6(3), 1991, pp. 225-250.
24. de Troyer O.M.F. The OO-Binary Relationship Model: A Truly Object Oriented Conceptual Model. // *Proceedings of the Third International Conference CaiSE'91 on Advanced Information Systems Engineering*, vol. 498 of *Lecture Notes in Computer Science*, Trondheim, Norway, 1991, pp. 561-578.
25. de Troyer O.M.F. A formalization of the Binary Object-Role Model based on logic // *Data & Knowledge Engineering*, 19(1), 1996, pp. 1-37.
26. Vermeir D. Semantic Hierarchies and Abstractions in Conceptual Schemata // *Information Systems*, 8(2), 1983, pp. 117-124.

Vladimir Ovchinnikov, PhD in Engineering Science, is the head of main production systems development department of NISCo (Russia, Lipetsk), and assistant professor of Lipetsk State Technical University (LSTU). He has been participating in realization of large information systems, for example, a production tracing system of hot plate rolling manufacture, as a system designer and a project leader. He is lecturer on object-oriented designing and programming at LSTU. PhD thesis is devoted to theoretical and practical issues of query and data storing optimization within relational scheme as applied to creating effective information systems of continuous (non-discrete) production management support. At present, his research activities are concentrated on theoretical and practical issues of conceptual modeling of vast information systems. The author is available at e-mail: ovch@lipetsk.ru, phone: +70742411139.