

A Semantically Complete Conceptual Modeling Technique

Vladimir Ovchinnikov (ovch@lipetsk.ru)

Existent up-to-date conceptual modeling techniques have a row of features that complicate the process of modeling and model study: a designer and a model user have to work in the plane of relations and in the plane of object types simultaneously; to comprehend interconnection of certain object types completely, it is necessary to study a model as a whole; formulation of conceptual queries can not be done without using proper relation designations.

A semantically complete model is proposed as a basis of a conceptual modeling technique that is free of the enumerated peculiarities. For practical using of the semantically complete conceptual modeling technique it is necessary to define notations for models, constraints and conceptual queries. That is the topic of the paper.

Keywords: conceptual modeling, conceptual query language, semantically complete model, semantically complete query language

1. Introduction

Conceptual modeling methods play the key role during designing complex information systems since they allow to cover a Universe of Discourse (UoD) as a whole without deepening to system implementation details. It is just the conceptualization principle [12] holding for any data modeling technique that is treated as a conceptual one. The conceptualization principle permits to abstract one's mind from inessential aspects for the conceptual designing stage which are connected with, for instance, an efficiency of an ultimate system architecture. It makes possible for a designer to comprehend UoD as a whole and to prepare high-quality system architecture specification that has decisive effect for content, quality and time constraints for all subsequent stages of system creation.

Set-theoretic modeling with constraints formulated with propositional logic expressions is often used as a conceptual modeling technique. This approach has significant theoretical grounds since it is based on set theory and propositional logic. But as a result of commonality, applicability for solving any modeling task, its using for data modeling leads to some model unhandiness in comparison with specialized techniques. For example, to ascertain a mandatory constraint for the attribute x within the relation $R \subseteq X \times Y$, it is necessary to formulate the constraint as the expression $\forall x \in X \exists y \in Y \{(x, y) \in R\}$ instead of, for example, (\underline{x}, Y) .

Another often used conceptual modeling technique is Entity-Relationship (ER) one [7, 8]. The key conceptions of the technique are entity, representing an object of real world, entity attribute, describing an object characteristic, and relationship between entities. For constraint formalization it is used a special notation that covers the most fundamental constraints (uniqueness, primary key, mandatory, referential integrity constraints). During designing of an ER model one decides about attribute grouping into entities. It partially represents deciding about ultimate system implementation and is one of reasons of research continuation in the field of data conceptual modeling. Researches did not stop on Entity-Relationship Model and kept on investigation for more complete fulfillment of the conceptualization principle.

Object-Role Modeling (ORM) [4, 14] was created as a result of further investigations. It evolved from a binary modeling technique [19] to a technique generalizing [5] majority of known conceptual modeling technique [3, 13, 16, 19, 23] including Entity-Relationship one [8]. The key conception of object-role modeling is also entity (the second name – object type). But unlike to ER model, this entity is not characterized by an attribute set – all attributes are considered as independent entities that may have proper interconnections with other entities (in the framework of this technique interconnections are named as fact types). Such approach allows to abstract one's mind from partitioning attributes among entities. The question which one has to solve during ER modeling: “am I to allocate the given attribute as an entity or keep it as an attribute”, – this question is postponed to the system implementation stage and is not affected during conceptual designing. Moreover, object-role modeling has rather more wide constraint language that contains, for instance, such constraint as inclusion of one connection (fact type) to another. For object-role modeling technique there are methods of adding abstraction layers [6, 9, 10, 11, 22, 24] that are used, as well, during an ER model forming from an ORM model.

Quality of a conceptual modeling technique may be estimated as maturity of three composing means: model description, constraint description and conceptual query formulation. The more advanced conceptual modeling technique: the more it permits to describe UoD and customer requirements in detail, the less it forces designer to deepening into system implementation details, the more it is clear for the colleagues and customer which are unskilled in the field of information system designing.

In the view of UoD description minuteness, the set-theoretic modeling is unattainable for the rest techniques. But in the part of clarity for unskilled men, it is not stand up to criticism. Comparing an ER model and an ORM model, we may state that ORM has a more powerful constraint description language and has rather better clarity for unskilled men. Simultaneously, ER has no conceptual query languages while ORM has several [1, 2, 17, 18]. It means that object role modeling is more preferable than ER modeling for conceptual UoD description. An ER model may be formed from an ORM model during gradual movement from a requirement specification to a system implementation [15].

All enumerated techniques have a row of disadvantages. To describe them we use the conception of object type as generalization of set of set-theoretic modeling and attribute of ER modeling, and the conception of relation as generalization of fact type of object-role modeling and entity of ER modeling. These disadvantages are the following:

- A designer and a user of a model have to work in two planes simultaneously: in the plane of object types and in the plane of relations. Relations have proper semantics and are to be memorized along with object types.
- By studying a separate relation, one can not state that interconnection of object types underlying the relation is studied too: a model may contain other relations based on the same object types. It is the reason of fundamental incompleteness of studying any model part. To be sure that an interconnection between object types are understood correctly, it is necessary to study all model as a whole.
- Formalization of conceptual queries can not be done without using proper relation designations.

- A formal query has a structure that significantly differs from its natural verbal formulation.

The enumerated disadvantages add complexity into the process of conceptual model creation and study.

As a development of conceptual modeling the papers [20, 21] suggest semantically complete model that has the following distinctive feature: relations (interconnections) are identified by means of sets of object types (entities) underlying their. The feature allows to attain a high degree of abstracting from implementation details by dint of acquisition the following additional properties:

- During working with model relations, one has not to memorize and use their designations. For each relation it is necessary to know about object types underlying it.
- Model analysis may be fulfilled in the plane of object types only. For any connected object types it is sufficient to state the connectivity fact. More complex statements about connectivity of object types are not necessary.
- Each relation has complete semantics within a model. During study of a certain relation one may be sure that another diagram of the same model does not contain another relation defining the interconnection between the same object types in a different way.
- Any model may be studied sequentially without rebuilding of already formed representations about interconnections of object types.
- For such model a query language may be developed, expressions of which do not refer to relations in the apparent way. Used relations are defined by sets of object types, an connection of which is requested. Let us list the other important properties of such a query language:
 - a query has a structure that is close to its natural verbal formulation (i.e. without artificial using of relation designations);
 - a query is compact and, at that, for object type designation it is used the same word-combination as in the verbal formulation.
- Propositional logic expressions as applied to the given model may be recorded in the simpler way from the structural point of view.

It follows from the enumerated properties that semantically complete model application for conceptual modeling is more preferable than of other models of conceptual class.

However, the papers [20, 21] do not contain a description of notations for semantically complete model and constraints, do not contain a structure and notation of a conceptual query language. It does not allow to use this technique in practice as an independent method of conceptual modeling. The paper bridges this gap: we introduce base constraints of a semantically complete model, suggest a notation for it and its constraints, state a structure and a notation of a semantically complete query language.

The paper has the following structure. The section 2 introduces base constraints of a semantically complete model and suggests its graphical notation. The section 3 introduces a textual notation, grounds its necessity. The section 4 states a description of a semantically complete expression structure, the conception of representation is introduced during it. The section 5 proposes a notation of a semantically complete query language, and the section 6 proposes Backus-Naur Form of it. The following section contains an example of practical application of the semantically complete modeling

technique including the proposed query language. The last section suggests the summary and the closest research directions for your attention.

2. A Graphical Notation of Semantically Complete Model

To make it possible to apply semantically complete modeling as an independent conceptual modeling technique, it is enough to introduce the following row of base constraints: functional constraint, equal constraint, mandatory constraint and fullness constraint. These constraints exist within all conceptual techniques in one way or another. Let us explain the essence of the enumerated constraints.

Let the relation R exists and is based on the following object types:

$R : (A_1, \dots, A_n, B_1, \dots, B_m)$. Then:

- Functional constraint $A_1, \dots, A_n \rightarrow B_1, \dots, B_m$ implies the predicate $\forall a \in \pi_{A_1, \dots, A_n} R \exists b \in \pi_{B_1, \dots, B_m} R \{(a, b) \in R\}$ stating that any existent instance combination for the types A_1, \dots, A_n corresponds to only one instance combination for the types B_1, \dots, B_m .
- Equal constraint $A_1, \dots, A_n \equiv B_1, \dots, B_m$ implies the predicate $\forall a \in \pi_{A_1, \dots, A_n} R \exists b \in \pi_{B_1, \dots, B_m} R \{(a, b) \in R\} \wedge \forall b \in \pi_{B_1, \dots, B_m} R \exists a \in \pi_{A_1, \dots, A_n} R \{(a, b) \in R\}$ stating that any existent instance combination for the types A_1, \dots, A_n corresponds to only one instance combination for the types B_1, \dots, B_m and vice versa. This constraint is equivalent to two counter functional constraints: $A_1, \dots, A_n \rightarrow B_1, \dots, B_m \wedge B_1, \dots, B_m \rightarrow A_1, \dots, A_n$.
- Mandatory constraint $(A_1, \dots, A_n, B_1, \dots, B_m)$ implies the predicate $\forall a \in \pi_{A_1, \dots, A_n} R \exists b \in \pi_{B_1, \dots, B_m} R \{(a, b) \in R\}$ stating that any existent instance combination for the types A_1, \dots, A_n corresponds to at least one instance combination for the types B_1, \dots, B_m . At that, if the model contains other relations based on the object types A_1, \dots, A_n , the part $\forall a \in \pi_{A_1, \dots, A_n}$ of the previous expression should be modified with the aim at covering all instance combinations for the object types A_1, \dots, A_n that participate in all relations based on them.
- Fullness constraint $(A_1, \dots, A_n | B_1, \dots, B_m)$ implies the predicate $\forall a \in \pi_{A_1, \dots, A_n} R \{a_1 \neq \emptyset \wedge \dots \wedge a_n \neq \emptyset\}$ stating that all existent instance combinations for the types A_1, \dots, A_n contain nonempty instances of all these types. Fullness constraint is introduced in consequence of possibility of empty object instances existence within conceptual relations.

The enumerated constraints are exhaustive for many conceptual modeling techniques, for instance, for ER modeling: primary key constraint may be modeled by combination of functional constraint and mandatory constraint, referential integrity constraint may be modeled by mandatory constraint and so on. Within the proposing modeling technique there is possibility to form more complex constraints based on queries to a model. In this case a constraint is formulated not for a relation included to the model, but for a relation calculated with an expression. A structure and a syntax of semantically complete query language expression will be stated in the appropriate sections further.

Then, on the basis of described constraints of semantically complete modeling, a graphical notation for it will be introduced. Addition of a new graphical notation into the row of already existent ones is conditioned by the following reasons:

- Semantically complete model has the line of properties that significantly differ from other conceptual models. For example, object-role model allows to define several fact types (relations) based on the same object type set [4]. Semantically complete model does not permit it since any relation is identified by an object type set [21]. Therefore, the appropriate possibilities of the object-role model graphical notation, and its particular cases [4, 5], become redundant.
- Semantically complete model does not include the aspect of grouping attributes to entities. Therefore, the ER model graphical notation [8] is not applicable to it in principle.

Let us take the following intuitively clear designations as the basis of proposing graphical notation:

- a binary relation without functional or equal constraints is designated by a line: –;
- a binary relation with functional constraint – by an arrow: →;
- a binary relation with equal constraint – by a triple line: ≡;
- a mandatory constraint – by bold point from the side of a mandatory object type: •; a mandatory constraint may be used along with functional or equal constraints;
- a fullness constraint is implied by default; if a relation has a certain object type that is not full, on a diagram this fact is designated by rhomb from the side of this object type; a fullness constraint may be used along with other constraints;
- a n-ary relation is designated by a contour outlining object types underlying it.

The examples of all six cases are shown on the figure 1.

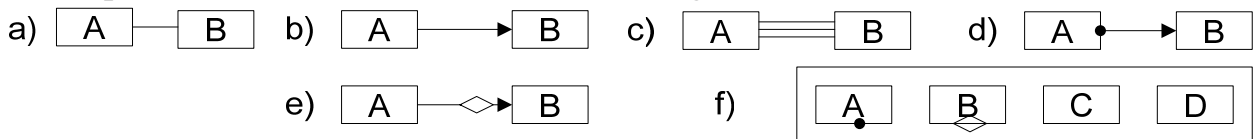


Figure 1 Elements of Semantically Complete Model Graphical Notation

Compare, for instance, representation of the case d) in the proposed notation with its representation in the notation of object-role model (1) and ER model (2) on the figure 2.

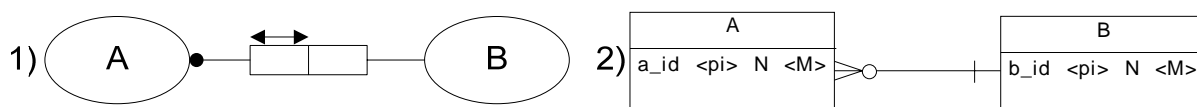


Figure 2 Representation of Functional and Mandatory Constraints in the Notations of Object-Role and ER models

As we can see from the figure 2 the semantically complete model notation is more compact. It is attained owing to its properties [21]:

- Each relation may contain a certain object type only once.
- A model can not contain two or more alternative relations simultaneously. Relations are considered to be alternative if they are based on object type sets included (or equal) one to another.

The graphical notation of semantically complete model does not require means of reflection of alternative connections between object types and those connections that contain one object type more than once. At that, the paper [21] shows that expressiveness of a semantically complete model is not worse than expressiveness of any other conceptual model. For example, any UoD modeled with an object-role model may be

modeled with a semantically complete model too. To provide it, a special transformation technique for recursive and alternative relations is used.

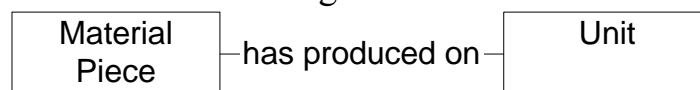
3. A Textual Notation of Semantically Complete Model

As a rule, existent data conceptual modeling techniques use only the base constraints for UoD description. Formalization of more complex constraints defers to the application system implementation stage. Nevertheless, complex constraints are parts of requirement designing and have a determinative influence on all subsequent stages of system creation. Their formalization, for instance, with ER model, is not possible, and the designer have to resort to external means, most often to verbal description.

For formalizing complex constraints a graphical notation should not be used as complex constraints in graphical form is bulky and unreadable. Therefore, along with a graphical notation we suggest to use a textual notation for more detailed requirement specification.

The second reason of textual notation introduction is necessity to develop effective means of discussion of intermediate formalization process results with ones not being specialists in the field of information system designing, and being experts in a studied UoD. It should be stated that the hopes pinned on graphical notations, as means of communication with non-specialists in information system designing, are not justified in full measure: to master the skill of graphical diagram reading it is necessary a separate training. In this regard, textual notations, having the property of closeness to natural language phrases, is more preferable since a specialist of any UoD wield textual information.

Semantically complete model in its textual notation represents a set of phrases that are understandable to a UoD expert. Each phrase is a statement of existence of a semantically complete relation based on a set of object types mentioned in the phrase. For example, the phrase “A Material Piece has produced on a Unit” states about existence of the object types “Material Piece” and “Unit” and existence of the semantically complete relation (Material Piece, Unit) within the model. The given phrase is the analogue of formulation of the same semantically complete relation in the graphical notation, as it is shown on the figure 3:



**Figure 3 Graphical Notation of the Phrase
“A Material Piece has produced on a Unit”**

For the proposed modeling technique it is important to name object types in the way that is fully corresponded to their senses from an expert’s point of view. In this case, a designer have not to switch between two rows of terms: terms for thought expression and terms for modeling; both rows flow together to one used in both processes. In such interpretation an object type acquires the meaning of UoD concept. Therefore, within a semantically complete model we use the conception “concept” as a synonym for the conception “object type” [21].

Definition of textual notation phrases should not contradict to the properties of a semantically complete model:

- a concept can not be a part of a phrase more than once;

- within a model there are no two phrases based on concept sets that are included or equal one to another.

If during designing necessity of violating these rules arises, a method of putting a model to the semantically complete form described in [21] should be used. This method comes to adding new concepts to a model that are equivalently connected with a certain existent concept, and allows to eliminate semantically completeness violations. At that, all properties of a semantically complete model described in the introduction comes into force.

Base constraints of the textual notation may be defined in a phrase itself as well as below the phrase in square brackets with intend. For example, in the following way:

A Produced Material Piece is a Material Piece
 [Produced Material Piece \equiv Material Piece]
A Produced Material Piece is produced on a Unit
 [Produced Material Piece \rightarrow Unit]

From consideration of intuitive clarity we use the following constraint designations in the textual notation:

- functional constraint – in square brackets below a phrase with the symbol \rightarrow ;
- equal constraint – in square brackets below phrase with the symbol \equiv ;
- mandatory constraint – by means of underlining of appropriate concepts in a phrase;
- fullness constraint is considered to be active if a concept does not included to vertical lines, for instance, in the phrase “A Produced Material Piece has produced on a |Unit|” the concept “Produced Material Piece” has the fullness constraint and the concept “Unit” has no it.

The proposed textual notation allows to represent a model in the way that is transparent for experts of any UoD, who are not specialists in the field of information system designing. Within it strict formal foundations of a model harmonizes with an easy readable representation.

In square brackets below a phrase, more complex constraints than enumerated before may be represented. For instance, the constraints based on complex queries. For defining such a constraint it is necessary to specify a query language for semantically complete model. We will do it in the following sections.

4. A Structure of Semantically Complete Expression

An expression of any query language represents a way of calculation of a resulting relation on basis of one or several base relations. Base relations may be relations included to a model directly as well as relations calculated by means of another expression – subexpression. To define the structure of semantically complete expression, we use the conception “relation” as generalization of calculable relations and relations constituting a model (semantically complete relations [20]). While semantically complete relations have some constraints [20], relations in the general case, conversely, have no such constraints, for example, one object type may participate in a calculable relation several times.

To maximize minuteness of a semantically complete expression structure describing, we define the conception “relation” in the way somewhat widened in comparison with commonly accepted. As a constitutive conception we use the

conception “object type” that is more typical for conceptual modeling than the analogous conceptions “set”, “entity”, and others.

We consider that any relation is based on a set of positions. Each of them contains only one object type. Let us detail object types according to a role which they play in a certain expression. For that, we define an additional role token for relation positions, represent it as a positive integer index, and formulate the constraint that a relation may contain an object type with the same role (index) only once. For designation of an object type in a role we use the conception “role object type” that corresponds to a combination of an object type and its role index within a relation.

Note that not any relation may be represented as an object type set since object types may repeat within a relation. But any relation may be represented as a role object type set since there is prohibition of role object type repetition. The property will further allow us to simplify the definition of a semantically complete expression structure.

A set of role object types represents a relation header. Further, we define the conception “relation state” (“relation body”). A relation state implies a set of cohesions (records, connections, tuples) each of which contains a set of role object type instances. Within cohesion there may be all role object types of the relation as well as only part of them. For each state it is true that all cohesions included in it contain different sets of role object type instances.

A header and a relation state may be represented as a table, as it is shown on the figure 4.

A(1)	B(1)	B(2)
3	5	6
7	8	6
A		1

Figure 4 Example of Relation Header and Relation State

On this figure {A, B} are object types, {A(1), B(1), B(2)} are role object types, {(3,5,6), (7,8,6), ('A', 1)} are cohesions of current state of this relation.

A key distinction of proposed relation definition from existent ones is that the conception “role object type” is used. Also, there is a distinction from relational model, that is while a relational attribute can not be shared between several relations, belongs to only one relation, actually being a column or a position of the relation, an object type can participate in several relations in the same role as well as in different roles.

Let us introduce the conception “representation” to define elemental transformations that may be applied to relations within the proposed query language. A representation is a relation having the following properties:

- it is based on object types being states of certain relations; one of these relations is considered to be calculable, the other are considered to be base;
- within a representation a state of calculable relation functionally depends on states of base relations.

Let us underline that all relations-representations are based on object types being states of certain relations and a calculable relation state is unambiguously determined by states of base relations. A representation have such name because of it describes a way of relation calculation, in other words, it represents a calculable relation.

In the framework of representations, object types do not repeat and have the same index taken by default for all representations (for example, 1). Therefore, as applied to representations, we can state that object types and role object types are the same.

On the figure 5 a representation example is shown. It is based on role object types $\{R1(1), R2(1)\}$ that are relation states at the same time. On the figure the relations are designated in the same way as the object types – R1 and R2. The relation R1 is based on the role object types $\{C(1), A(1)\}$, and the relation R2 is based on the role object types $\{A(1), B(1), B(2)\}$. The state of the given representation consists of a set of cohesions, each object instance of which is a state of the relation R1 or the relation R2.

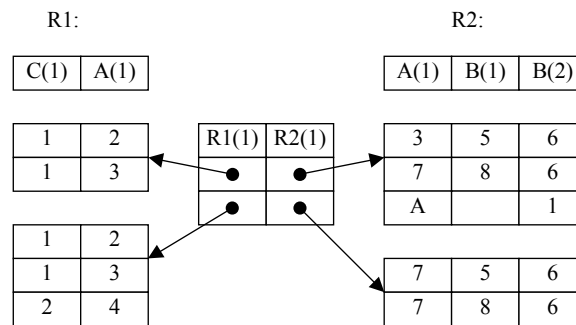


Figure 5 Example of Representation Based on Two Relations

Let us enumerate the main representations of suggested query language. It is necessary to note that the list of main representations intersects with analogous lists of other conceptual languages but there are significant distinctions in the structure of common representations. The key distinction is strict differentiation of object types: the criterion of matching (superposition) of base relation positions, during joining, uniting, and differencing, is always fixed and based on coincidence of role object types, and not on coincidence of names or positions, as it is within existent approaches.

Let us define composition (join) of several relations. A composition is a representation complying with the following statements:

- Base relations form a connected multigraph on role object types underlying them.
- A calculable relation is based on all role object types of base relations (without repetition).
- A state of a calculable relation consists of cohesions based on allowable combinations of base relation cohesions in the following way: each instance of a role object type participated in cohesions of the allowable combination is included into an appropriate cohesion of the calculable relation under the same role object type. A combination of base relation cohesions is allowable if
 - it contains one cohesion for each base relation;
 - role object types repeated in several base relation cohesions have the same instance in all these cohesions or have no instances in all cohesions.

Composition has the definition that is analogous to natural join of relational algebra with the difference that natural connectivity of role object types is defined not by name coincidence, but by coincidence of role object types themselves. This distinction is decisive, since a name coincidence does not guarantee the same semantics of relational attributes, and is principal for acquiring the semantic component by suggested language.

The second distinction is that a composition is defined for any quantity of base relations, and not for two as it is within other approaches. The last distinction is not

principal since a n-ary composition may be decomposed on a set of binary compositions without information loss.

An example of a composition of two relations based on the role object types {A(1), B(1), B(2)} and {B(2), C(1)} is shown on figure 6.

A(1)	B(1)	B(2)
3	5	6
7	8	6
A		1

B(2)	C(1)
6	5
8	4
1	1

⇒

A(1)	B(1)	B(2)	C(1)
3	5	6	5
7	8	6	5
A		1	1

Figure 6 Example of Composition of Two Relations

A union is a representation complying with the following statements:

- A calculable relation is based on all role object types of base relations without repetition.
- For each cohesion from any base relation state there is one cohesion in a calculable relation state that consists of the same instances of the same role object types. All role object types of a calculable cohesion, that do not participate in a base cohesion, possess an empty value, in other words, are absent in the calculable cohesion. Cohesions of different base relations, containing the same combination of object role type instances, are included into the calculable relation state as a single cohesion, in the unduplicated way.

The intensional difference of the proposed union from existent ones is the strategy of relation column (position) matching: matching of base relation columns is fulfilled only from coinciding of role object types, and not column positions. It represents semantic orientation of suggested union. The next distinctions are not so principal: there may be more than two of uniting relations; coinciding of dimensions of uniting relations is not necessary.

The figure 7 represents an example of uniting of two relations having different header. Pay attention to the peculiarity that the cohesion (,,3) of the first relation and the cohesion (3,) of the second relation are reflected to one cohesion of the calculable relation while all other cohesions are reflected in the one-to-one way as their instance combinations are not repeated.

A(1)	B(1)	B(2)
3	5	6
7	8	6
A		1
		3

B(2)	C(1)
6	5
8	4
1	1
3	

⇒

A(1)	B(1)	B(2)	C(1)
3	5	6	
7	8	6	
A		1	
		6	5
		8	4
		1	1
		3	

Figure 7 Example of Uniting of Two Relations

A difference is such a representation that complies with the following statements:

- It is based on two base relations.

- A calculable relation is based on the same role object types as the first base relation.
- A calculable relation state consists of those cohesions of the first base relation state for which there are no adjacent cohesions in the second base relation state. Two cohesions are considered to be adjacent if all common role object types either contain the same instances or do not contain instances simultaneously.

Like composition and union, the proposed way of difference definition differs from existent variants: its fulfillment does not depend on position of columns within the base relations or their naming, and depends on role object type coinciding only.

An example of difference of two relations is shown on the figure 8.

A(1)	B(1)	B(2)
3	5	6
7	8	6
A		1
		3
B	5	

⇒

A(1)	B(1)	B(2)
A		1
		3

B(2)	C(1)
6	5
8	4
	4

Figure 8 Example of Difference of Two Relations

Let us give a definition of projection within the suggested query language. A projection is such a representation that complies with the following statements:

- It is based on one base relation.
- A calculable relation is based on subset of role object types of a base relation.
- A calculable relation state consists of cohesions formed by means of projection of each cohesion of a base relation state by role object types that constitute the calculable relation. In other words, those role object type instances of a base relation cohesion, role object types of which are included into a calculable relation, form a calculable relation cohesion.

The proposed projection definition practically does not differ from existent ones. The only distinction is in the form, the projection is defined with using role object types, and not attributes or relation positions.

Introduction of a next representation, a positioning, is conditioned by the peculiarities of a semantically complete model and the proposed query language. Existent conceptual query languages have no analogues for positioning. A positioning is such a representation that complies with the following statements:

- It is based on one base relation.
- Between role object types of a base relation and a calculable relation there is an one-to-one correspondence: each role object type of a base relation corresponds to one role object type of a calculable relation and vice versa. Simultaneously, there is a constraint that corresponding role object types relate to the same object type.
- A calculable relation state consists of role analogues for cohesions of a base relation state. A role analogue of a cohesion contains the same object type instances as the cohesion itself. The distinction is that role object types on which the instances are fastened may differ. Change of instance fastening is fulfilled according to the defined correspondence of role object types of base and calculable relations.

Within a query an object type may be used in different roles. Therefore, a mechanism of assigning different indexes for the same object type, i.e. forming different role object types on the basis of the same object type, is necessary. A positioning is such a mechanism. As a result of positioning application, a base relation does not change its content, but the header of the relation changes: other indexes may be assigned to several or all object types.

During application of other representations to a result of positioning (for instance, a composition) all role object types are considered to be different in spite of the same object type may form their basis. It allows to formulate compact and complex queries, not referring to relation designations, with participation of the same object type in different roles.

An example of a positioning is shown on the figure 9. We can see from the figure that the initial relation $\{A(1), B(1), B(2)\}$ is positioned to the relation $\{A(2), B(1), B(2)\}$, and the relation content remains unchanged.

A(1)	B(1)	B(2)
3	5	6
7	8	6
A		1
		3
B	5	

⇒

A(2)	B(1)	B(2)
3	5	6
7	8	6
A		1
		3
B	5	

Figure 9 Example of Relation Positioning

Practically all conceptual query languages have the conception of selection from a relation by a condition. Within the proposed language a condition is considered as an independent relation based on role object types underlying it. At that, a cohesion set (perhaps infinite) complying with the condition is taken as a state of such a relation. Certainly, the given way of representation is logical only; we can not deal with physical storing of such relation state as a cohesion set. A relation that is formed according to a certain condition is named as “relation on condition”.

The definition of a condition as a relation allows to depart from using a selection as an independent representation. In this case, a selection is reduced to a composition of two relations: a base relation and a relation on condition. It becomes possible due to the following feature of the proposed composition: it does not require to specify a join condition in the apparent way, the join criterion is fixed and lies in coincidence of role object types (see above). Further we will show how it affects the syntax of the proposed query language.

An example of a selection from a relation is shown on the figure 10. On this figure a selection from the base relation $\{A(1), B(1), B(2), C(1)\}$ is represented as its composition with a relation on condition. The last relation consists of infinite quantity of cohesions (at logical level) that comply with the condition “ $B(2) > C(1)$ ”. As a result of the composition only cohesions complying with the given condition are selected from the base relation state.

A(1)	B(1)	B(2)	C(1)
3	5	6	8
7	8	6	2
A		1	1
B	2	2	1

⇒

A(1)	B(1)	B(2)	C(1)
7	8	6	2
B	2	2	1

B(2) > C(1):

B(2)	C(1)
2	1
3	1
4	1
...	

Figure 10 Example of Selection from Relation

Relations defined with the help of a formula on the basis of role object types may participate in a semantically complete expression. Such relation is based not only on role object types participating in the formula, but on an additional role object type that is absent at an initial model and represents a formula calculation result. A separate cohesion of such relation state represents a combination of instances of initial role object types and an instance of calculation result. For example, if a relation is based on the formula $B(2) - C(1)$, its state may be represented as consisting of infinite quantity of cohesions (at the logical level) as it is shown on the figure 11. Relations defined with formulas we name as relations on formulas. Relations on formulas and relations on conditions may be used within expression along with other relations.

B(2) - C(1):

B(2) - C(1)	B(2)	C(1)
1	2	1
2	3	1
3	4	1
...		

Figure 11 Example of Relation on Formula

With the aim of attraction of reader's attention to the main aspects of the proposed query language, within the paper we do not consider such possibilities as grouping, analytic functions, and other transformations exceeded the bounds of the base representation set. The author engages himself to discuss these aspects of the language in the next papers.

Further we define the essence of semantically complete query language expression. An expression implies such representation that complies with the following statements:

- Base relations is the representations: composition, union, difference, projection, positioning (and other representations not included into this paper). A calculable relation is a representation too. Thus, instances of role object types of base and calculable representations are states of certain relations (see definition of representation).
- Base representations form connected multigraph on role object types underlying it.
- A calculable representation consists of role object types that participate in only one base representation. If a role object type participates in more than one base representation, it is not included into a calculable relation.
- A calculable representation state is determined by way of composition of all base representations and subsequent projection on role object types underlying only one of

base representations. A relation that is calculated with the help of a calculable representation is named as target relation of an expression.

An example of an expression is shown on the figure 12. This figure contains an expression based on four representations: $\{D, C, B, A\}$, where D is the calculable representation, and $\{C, B, A\}$ are the base representations. Each base representation is based on its own set of object types (that are relation states simultaneously): the representation A is based on $\{A0, A1\}$, $B - \{B0, B1\}$, $C - \{C0, B0, A0\}$. The base representations form connected multigraph. The calculable representation is based on the object types that participate in only one of base representations: $\{C0, B1, A1\}$.

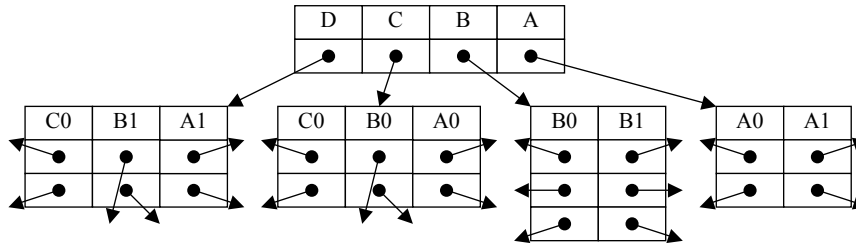


Figure 12 Example of Expression Based on Three Base Representations and One Calculable Representation

According to the proposed definition, an expression may be represented as a set of base representations. For formalization of an expression it is sufficient to list a set of representations without apparent describing of complex structural interconnections between them. High degree of semantically complete expression simplicity is based on the single rule of relation column superposition within representations: columns are combined if they relate to the same role object type.

Taking into account that all representations are constrained by a functional dependency of a calculable relation from base relations, and an expression calculates only one representation, we conclude that functional connections between base and calculable relations form a tree from initial relations to a target relation through a row of intermediate relations. Initial relations may be relations of semantically complete model as well as relations on conditions or relations on formulas (see above).

In our example (the figure 12) the expression structure may be represented as a tree of calculation of the target relation $C0$ from the initial relations $A1$ and $B1$, as it is shown on the figure 13.

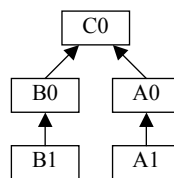


Figure 13 Tree of Calculation of Target Relation C1 from Initial Relations A1 and B1

Since relations of a semantically complete model may be used without apparent referring to their proper designations (through enumeration of object types only [21]), any semantically complete expression may be based on relations without apparent referring to them. How this fact reflects on a notation of a proposed query language we will show below.

5. A Notation of Semantically Complete Query Language

For the described structure of a semantically complete expression a row of alternative notations may be developed. We suggest one of possible notations that is selected by reasoning from intuitive transparency and compactness.

A semantically complete model consists of relations each of which is identified by set of object types underlying it [21]. Therefore, to refer to a semantically complete relation we further will use not a proper relation designation, but a set of object types enumerated with comma in round brackets. For example, (A, B, C) is the reference to the relation based on the object types A, B, and C.

Positioning of semantically complete relation is designated in the analogous way as the referencing with the distinction that in round brackets after an object type name its role index within a relation is shown. For example, (A(1), B(2), C(1)) is the positioning of the relation (A, B, C). At that, we consider that if a role index is not shown, it is equal to one. Therefore, the positioning (A(1), B(2), C(1)) may be equivalently recorded as (A, B(2), C). Within a semantically complete expression we will not use direct references to a model relation, and the record (A, B, C) will be considered as the positioning (A(1), B(1), C(1)) of an appropriate semantically complete relation since roles of object types are important for expression fulfillment.

A relation composition is denoted with two methods. The first method is enumeration of relations with comma in round brackets. For instance, ((A, B), (B, C)) is the composition of the relations (A, B) and (B, C), or rather the composition of two relations received as the positioning (A(1), B(1)) of the relation (A, B) and as the positioning (B(1), C(1)) of the relation (B, C). This method of denoting is applicable for relations of an arbitrary arity.

The second method is applicable to binary relations only and lies in designation of a composition in round brackets with the symbol ‘-’. In this case, a composition is fulfilled for those binary relations that are formed by a pair of role object types gathered round each symbol ‘-’. For instance, (A-B-C) is the composition of the relations (A, B) and (B, C). It is equivalent to the record ((A, B), (B, C)). This method of composition denoting allows to formulate a composition of a binary relation set as a chain of role object types. It is compact and intuitively clear.

A union of two relations is denoted in round brackets with the help of the keyword ‘union’ or the symbol ‘U’ between relations: ((A, B) U (B, C)) or ((A, B) union (B, C)). To denote union of more than two relations we do not introduce an additional syntax and use chains of unions: ((A, B) U (B, C) U (A, C, D)).

A difference of two relations is denoted in round brackets with the help of the keyword ‘minus’ or the symbol ‘/’ between relations: ((A, B) / (B, C)) or ((A, B) minus (B, C)).

A relation projection is denoted by enumeration of role object types being projected in round brackets, via a dot after a projecting relation. For instance (A, C, D).(C, D) is the projection of the relation (A, C, D) by the role object types C and D.

We consider that a projection is formulated with a subsequent positioning if a list of role object types for projection is not included into an initial relation (nevertheless, all object types should be included in any case). For example, (A, C, D).(C(2), D) denotes not only the projection (A, C, D).(C, D), but also the subsequent positioning (C(2), D).

At that, a record of a projection with a positioning that contains ambiguity is not allowable: when a role object type of a calculable relation is not included into a base relation and a base relation simultaneously contains several other role object types for the same object type. For instance, the record $(A, C(1), C(2), D).(C(3), D)$ is not allowable because of ambiguity: which role object type ($C(1)$ or $C(2)$) is positioned. In this case, at first, it is necessary to define a projection without positioning, and then a positioning: $(A, C(1), C(2), D).(C(2), D).(C(3), D)$. If a projection with a positioning is fulfilled on all role object types of a base relation, the projection is reduced to repetition of the base relation and may be neglected. For example, in the case of $(A, B).(A, B(2))$ we may consider that for the relation (A, B) the positioning $(A, B(2))$ is fulfilled only.

Within the proposed semantically complete expression a selection does not differ from a composition of two relations: one of relations is formed according to a logical condition. For example, $((A, B), (B > C))$ is the composition of two relations, the second relation is the relation on condition. It follows from the composition properties:

- The order of relations is not important: $((A, B), (B > C))$ and $((B > C), (A, B))$ are equivalent.
- Within a composition there may be an arbitrary quantity of relations on conditions. At that, a row of such relations is equivalent to one relation based on condition concatenating all conditions of the row.

A relation calculated with formula is denoted as formula in round brackets. For instance, $(B(2) - C(1))$. Note that within the given language it is necessary to distinguish the symbol minus ‘-’ and the symbol dash ‘-’ used for denoting a composition of binary relations.

To use the peculiarities of the suggested query language in full measure it is necessary to fulfill the requirement of complete conformity of object type naming and its sense [21]. Object types should be designated by word-combinations used by experts of UoD for discussing between themselves. It allows to attain such a structure of semantically complete expression that practically coincides with its verbal formulation.

6. Formalization of Syntax of Semantically Complete Query Language

Let us formalize the syntax of the proposed query language using Backus-Naur Form. We will forestall formal statements with their verbal interpretations.

A query of the semantically complete language is formulated as a semantically complete expression that is a description of relation calculation way:

semantically_complete_expression ::= relation

Calculation of a relation may be described with the help of the phrases: positioning, composition, binary composition, projection, union, or difference; a relation may be defined with a logical condition or a formula:

relation ::= positioning | composition | binary_composition | projection | union | minus | logical_relation | formula_relation

A positioning phrase represents an enumeration of role object types in round brackets via a comma:

positioning ::= (role_object_type , role_object_type_list)

role_object_type_list ::= role_object_type | role_object_type , role_object_type_list

A role object type is recorded as an object type with the subsequent optional role index in round brackets:

role_object_type ::= object_type [(role_index)]

A role index represents a positive integer:

role_index ::= positive_integer

A union phrase represents an enumeration of relations in round brackets via the keyword ‘union’ or the symbol ‘U’:

union ::= (relation U union_chain) | (relation union union_chain)

union_chain ::= relation | relation U union_chain | relation union union_chain

A difference phrase represents two relations in round brackets separated by the keyword ‘minus’ or the symbol ‘/’:

minus ::= (relation / relation) | (relation minus relation)

A projection phrase consists of a relation and a positioning phrase right after, separated by a dot:

projection ::= relation . positioning

A binary composition represents an enumeration of role object types in round brackets separated by the symbol ‘-’:

binary_composition ::= (role_object_type – role_object_type_chain)

**role_object_type_chain ::= role_object_type | role_object_type –
role_object_type_chain**

A composition phrase represents an enumeration of relations in round brackets via a comma:

composition ::= (relation , relation_list)

relation_list ::= relation | relation , relation_list

A relation on condition is recorded as a logical condition based on role object types in round brackets:

logical_relation ::= (logical_expression)

**logical_expression ::= logical_atom | (logical_expression) | logical_expression AND
logical_expression | logical_expression OR logical_expression | NOT
logical_expression**

**logical_atom ::= role_object_type > role_object_type | role_object_type <
role_object_type | role_object_type = role_object_type | role_object_type >=
role_object_type | role_object_type <= role_object_type**

A relation on formula is recorded as a formula based on role object types in round brackets:

formula_relation ::= (formula_expression)

**formula_expression ::= role_object_type | (formula_expression) |
formula_expression + formula_expression | formula_expression -
formula_expression | formula_expression * formula_expression |
formula_expression / formula_expression | formula_expression ^
formula_expression | sin (formula_expression) | cos (formula_expression) |
exp (formula_expression)**

7. An Example of Semantically Complete Modeling Technique Application

As an example of proposed technique application we consider MES (Manufacturing Execution System) as applied to plate and coil rolling, namely that part of the system which tracks actual production. The metallurgical production peculiarity is that making a final product can not be represented as a hierarchy BOM since during

processing a metal may be cut and welded in the arbitrary way. The key requirement made for systems of such class in the tracking part is the following: a system should contain data about all existent material pieces (raw materials, intermediate and final products) and full history of their processing.

During production a material piece undergoes the following transformations: slabs can be cut, coils can be cut or welded, plate packs can be formed from different coils as it is shown on the figure 14. Transformation operations occur on units, one material piece can be produced on only one unit and can be consumed for making other material pieces on only one unit too.

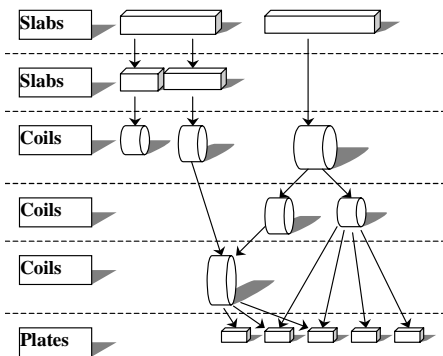


Figure 14 Transformation of Material Pieces During Processing

During projection of requirements to the system, it was developed and discussed with UoD experts in the given UoD a semantically complete model. A textual notation of its part are shown below:

- A Unit has an Average Capacity
[Unit → Average Capacity]
- A Unit has produced Produced Material Pieces
[Produced Material Piece → Unit]
- A Unit has consumed Consumed Material Pieces
[Consumed Material Piece → Unit]
- A Produced Material Piece is a Material Piece
[Produced Material Piece ≡ Material Piece]
- A Consumed Material Piece is a Material Piece
[Consumed Material Piece ≡ Material Piece]
- A Material Piece Transition has fulfilled from Consumed Material Piece
[Material Piece Transition → Consumed Material Piece]
- A Material Piece Transition has fulfilled to Produced Material Piece
[Material Piece Transition → Produced Material Piece]
- A Material Piece is assigned to a Production Order
[Material Piece → Production Order]
- A Material Piece is characterized by a Thickness
[Material Piece → Thickness]
- A Slab is a Material Piece
[Slab ≡ Material Piece]
- A Coil is a Material Piece
[Coil ≡ Material Piece]
- A Pack is a Material Piece

[Pack \equiv Material Piece]

In spite of the noticeable length of the relation and constraint list, such representation of the semantically complete model is a good mean for discussion of the formalized UoD structure with experts. In this case, the presence of a directed graph of transition of material piece parts is uncovered within the UoD.

General form graphs are structures that are the most complex for representation in information systems. We have chosen this example deliberately to demonstrate how this problem may be solved with semantically complete modeling. As we can see from the model, its solution lies in definition of concepts equivalently connected with a base concept that forms separate nodes or arcs of a graph. In our case, for the concept “Material Piece” representing graph nodes, we define two concepts equivalently connected with it: “Produced Material Piece” and “Consumed Material Piece”. The incidence of graph nodes and arcs, represented with the concept “Material Piece Transition”, is defined with the help of the relation “A Material Piece Transition has fulfilled to a Produced Material Piece” for incoming arcs, and with the help of the relation “A Material Piece Transition has fulfilled from a Consumed Material Piece” for outgoing arcs.

The graphical notation of the given semantically complete model is represented on the figure 15.

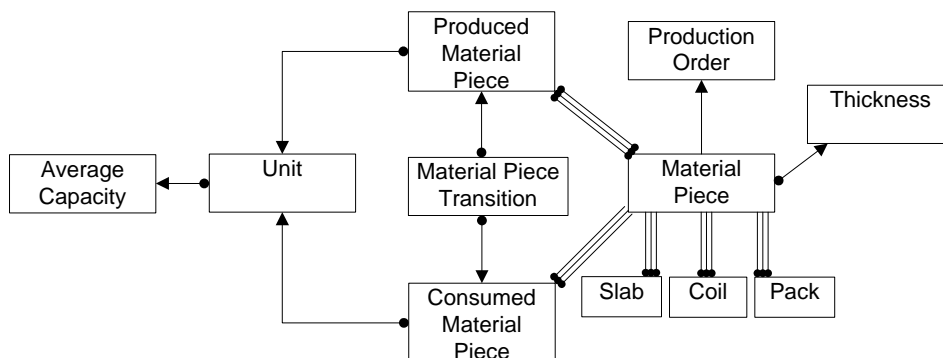


Figure 15 Graphical Notation of Semantically Complete Model of Material Piece Transformation During Processing

For comparison, let us give the object-role model of the same UoD (the figure 16).

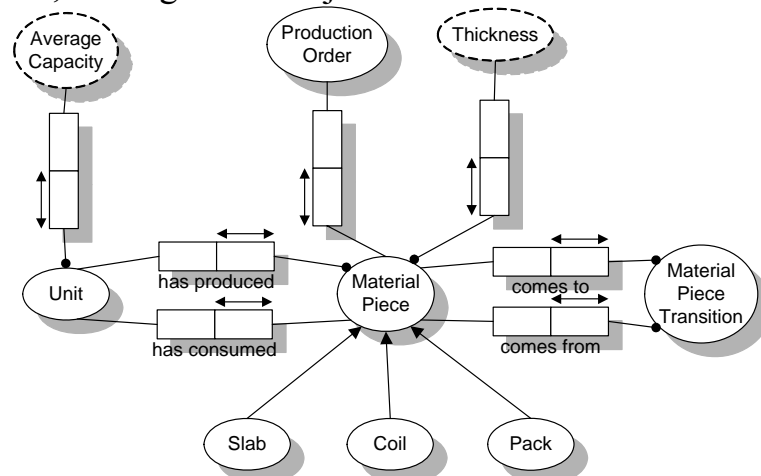


Figure 16 Object-Role Model of Material Piece Transformation During Processing

Due to semantically complete model properties, there is no necessity to use bulky designations of fact types in the form of partitioned rectangles within its graphical notation. At that, should we reduce the object-role model to the form satisfying the rules

of semantically complete model forming (see above), the unhandiness of the object-role model will increase. Therefore, we consider that, as applied to the semantically complete model, the proposed graphical notation has more simplicity and information capacity than the graphical notation of object-role model. We ascertain that introduction of independent graphical notation for semantically complete model is defensible.

Let us formulate a query example in the verbal form: “for each produced material piece, calculate percent reduction of all material pieces consumed during its production”. By disclosing the conception “percent reduction”, let us formulate this query in the more apparent form: “calculate a difference between a thickness of each consumed material piece and a thickness of each produced material piece made from the consumed one, divide the difference by the thickness of consumed material piece”.

Let us formalize the formulated query with the proposed query language:
(Thickness(1)–Material Piece(1)–Consumed Material Piece–Material Piece Transition–Produced Material Piece–Material Piece(2)–Thickness(2)),
((Thickness(1) - Thickness(2))/Thickness(1))

The given semantically complete expression represents the composition of two relations. The first relation has the arity 7 and is calculated as the binary composition of the relations (Thickness(1), Material Piece(1)), (Material Piece(1), Consumed Material Piece), (Consumed Material Piece, Material Piece Transition), (Material Piece Transition, Produced Material Piece), (Produced Material Piece, Material Piece(2)), and (Material Piece(2), Thickness(2)). Each of the enumerated relations is calculated as a positioning of an appropriate model relation, for instance, the relation (Thickness(1), Material Piece(1)) is calculated as the positioning of the semantically complete relation (Thickness, Material Piece).

The second relation of the composition represents the ternary relation based on the role object types “Thickness(1)”, “Thickness(2)”, and the calculated role object type “(Thickness(1) - Thickness(2))/Thickness(1)”, that exists only within the given expression, not within the model. As a result of the composition of these two relations, we calculate the resulting relation having the arity 8 and containing all material piece transitions, material pieces, their thicknesses and calculated percent reductions. To decrease the arity of the resulting relation and to keep only significant object types we can project the calculated relation on object types “Consumed Material Piece”, “Produced Material Piece”, and “(Thickness(1) - Thickness(2))/Thickness(1)”. In this case, the resulting relation has the arity 3:

(Thickness(1)–Material Piece(1)–Consumed Material Piece–Material Piece Transition–Produced Material Piece–Material Piece(2)–Thickness(2)),
((Thickness(1) - Thickness(2))/Thickness(1)). (Consumed Material Piece, Produced Material Piece, (Thickness(1) - Thickness(2))/Thickness(1))

If we designate relations as circles, representations – as rectangles, and calculation of one relation state on basis of other relation states – as arrows, the structure of the last expression may be reflected as it is done on the figure 17. On this figure P designates a positioning, C – a composition, Pr – a projection. Filled circles designate the initial expression relations and the target relation being the result of expression calculation. The intermediate relations are nonfilled circles.

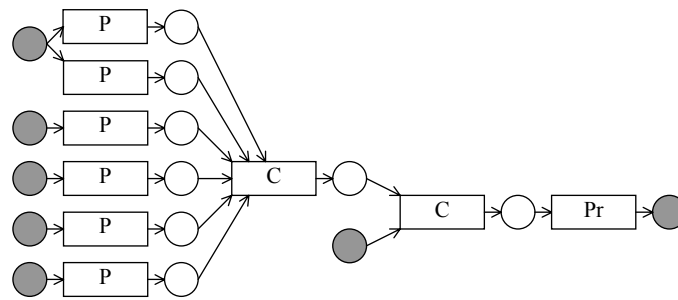


Figure 17 Structure of The Semantically Complete Expression

Since a material piece and a thickness are used in the expression in two roles (for produced and consumed material pieces), the relation (Thickness–Material Piece) participates in the expression twice: as the positioning (Thickness(1)–Material Piece(1)) and as the positioning (Thickness(2)–Material Piece(2)). It is shown on the figure 17.

Using the formal query proposed as an example at first, let us restore its verbalization: “for each material piece transition, calculate a thickness of a material piece consumed during this transition and a thickness of a material piece produced during this transition, divide the difference of the first thickness and the second thickness by the first thickness”. Comparing this verbalization with initial one used for query formalization, we can see that they are very close. It allows to state that a semantically complete expression structure and its verbalization are close.

Thus, the expression proposed in this section is based on object types only and does not use proper relation designations; object types are mediums of UoD semantics and actually are the concepts of UoD; an expression has an intuitively clear structure that is close to a structure of an appropriate natural language phrase, has a compact form in spite of using full word-combinations for object type designation.

8. Summary

The paper proposes a conceptual modeling technique that allows to build semantically complete models in professional activity of an information system designer. At that, all advantages of a semantically complete model (see introduction and [21]) become available for the designer.

Additionally, the textual notation of this technique allows to design and discuss a model in the same terms. The developed graphical notation takes into account peculiarities of a semantically complete model (for instance, that an identifier of a relation is a set of object types underlying it), and gives obvious meaningful representation of a model skeleton included all relations and their base constraints.

A conceptual query language is developed. Its expressions are formulated without using proper relation designations. Within the proposed language, a single principle of base relation superposition is used, according to coinciding of role object types. It is a forward step in comparison with universally recognized superposition based on name or position coinciding. Such approach permits to unify superposition of all representation types, to attain intuitive clarity of language expressions, to free a designer from necessity to formulate a superposition criterion in the apparent way. This fact allows to advance a query structure to the structure of natural language phrases, and, in combination with semantic naming of object types, permits for expression to attain a form that is clear for a non-specialist in information system designing. Simultaneously, the necessity to switch

between terms for modeling and terms for discussion vanishes since both rows of terms coincide into one row.

The relation positioning feature allows to use one object type in different roles within one query. As a result, complex queries attain a more simple and obvious form. A peculiarity of the proposed language is that it does not distinguish a composition and a selection. A selection represents the particular case of a composition of two relations, when one of them is a relation on condition.

An advanced conceptual modeling technique should contain means for describing a model structure, constraints and queries. In this paper we affect all three aspects of semantically complete modeling. The properties of a semantically complete model [21] give grounds to predict a significant effect from its application on practice, that is expressed in hastening and quality increasing of conceptual modeling, facilitation of model familiarization process.

As a nearest research task, the author considers widening of semantically complete query language with additional possibilities (such as grouping, analytical function calculation, and others).

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, 1996, 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, 1997.
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. Brouwer S.J., Martens C.L.J., Bronts G.H.W.M., Proper H.A. Towards a Unifying Object Role Modelling Approach // Proceedings of the First International Conference on Object-Role Modelling (ORM-1), Magnetic Island, Australia, 1994, pp. 259-273.
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. Chen P.P.S. The entity-relationship model – towards a unified view of data // ACM Transactions on Database Systems, 1(1), 1976, pp. 9-36.
8. Chen P.P.S. A Preliminary Framework for Entity-Relationship Models // Entity-Relationship Approach to Information Modeling and Analysis, Saugus, Calif., 1981.
9. Creasy P.N., Proper H.A. A Generic Model for 3-Dimensional Conceptual Modelling // Data & Knowledge Engineering, 20(2), 1996, pp. 119-162.
10. Feldman P., Miller D. Entity Model Clustering: Structuring a Data Model by Abstractions // The Computer Journal, 29(4), 1986, pp. 348-360.
11. 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.

12. van Griethuysen J.J., editor. Concepts and Terminology for the Conceptual Scheme and the Information Base. Publ. nr. ISO/TC97/SC5-N695, 1982.
13. Halpin T.A., Orłowska M.E. Fact-Oriented Modelling for Data Analysis // *Journal of Information Systems*, 2(2), 1992, pp. 1-23.
14. Halpin T.A. *Conceptual Schema and Relational Database Design*. Prentice-Hall, Sydney, Australia, 2nd edition, 1995.
15. Halpin T.A. Entity Relationship modeling from ORM perspective // *Journal of Conceptual Modeling* (www.inconcept.com/jcm), 11, 1999.
16. ter Hofstede A.H.M., van der Weide. Expressiveness in conceptual data modeling // *Data & Knowledge Engineering*, 10(1), 1993, pp. 65-100.
17. 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.
18. 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.
19. Nijssen G.M., Halpin T.A. *Conceptual Schema and Relational Database Design: a fact oriented approach*. Prentice-Hall, Sydney, Australia, 1989.
20. Ovchinnikov V.V. A Conceptual Modeling Technique without Redundant Structural Elements // *Journal of Conceptual Modeling* (www.inconcept.com/jcm), 29, 2003.
21. Ovchinnikov V.V. Improving Controllability of Vast Conceptual Models // *Journal of Conceptual Modeling* (www.inconcept.com/jcm), 31, 2004.
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. 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.
24. Vermeir D. Semantic Hierarchies and Abstractions in Conceptual Schemata // *Information Systems*, 8(2), 1983, pp. 117-124.