

Conceptual Modeling of Electricity Consumption Tracking with Semantically Complete Model

Vladimir Ovchinnikov (ovch@lipetsk.ru)

Semantically complete modeling technique has a row of essential distinctions from other conceptual modeling techniques: a designer and a user of a model may work in the plane of object types only, not using relation designations; each relation has complete information about interconnection of object types included in it, there is no necessity to rebuild formed idea of object type interconnections during model study; formulation of conceptual queries may be fulfilled without using proper relation designations.

This article considers application of semantically complete modeling technique for conceptual modeling of electricity consumption tracking during creation of an appropriate information system.

1. Introduction

Information systems find applications practically in all fields of human activity of present-day world. Requirements to information systems are being continuously developed, it results in increasing of their complexity. For hastening and quality improving of complex information systems, methods of data conceptual modeling are used. They allow to abstract one's mind from inessential system implementation details during the designing stage and to concentrate on UoD (Universe of Discourse) modeling [3].

A new approach to conceptual modeling was proposed by the author [5, 6]. It was named as semantically complete one and has a row of distinctive features in comparison with others [1, 2, 4]:

- While using relations of a semantically complete model, one does not need to memorize and use their designations. For each relation, it is sufficient to know object types underlying it.
- Model analysis may be fulfilled in the plane of object types exclusively. For any connected object types, it is sufficient to state a connectivity fact. More complex statements about connectivity of object types are not necessary.
- Each relation has a complete semantics within a model. While studying 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 the different way.
- A model may be studied sequentially, without reconstruction of already formed idea about interconnections of object types during study.
- For each such model, a query language may be developed, expressions of which do not refer to relations in the apparent way. Relations being used are defined by sets of object types, an interconnection of which is requested. Let us note other important properties of such a query language:
 - a query has structure and way of writing that are close to its natural verbal formulation (without artificial using of relation designations);
 - a query is compact, whereas for object type designation the same word-combinations are used as in a verbal formulation.
- Expressions of propositional logic as applied to this model may be written down in a simpler way (from the structural point of view).

This article describes an application of semantically complete modeling to UoD of electricity consumption tracking. The main part of it is contained in the sections 2 and 3. The section 2 represents a short description of UoD; introduces its semantically complete model stepwise; gives explanations of used semantically complete modeling constructions. In the section 3 a semantically complete model of electricity consumption calculation for installations is introduced. In the last section, as a result of application of this conceptual modeling technique to electricity consumption tracking, conclusions about practical features of the technique are given.

The article does not contain an exhaustive description of the semantically complete modeling technique. For more detailed familiarization with its constructions, it is necessary to resort to the paper [7].

2. Modeling of Electricity Consumption

The process of semantically complete modeling consists of two components: uncovering of UoD concepts, uncovering and formalization of connections between concepts (relations), uncovering and formalization of constraints applied to the relations.

During study of UoD of electricity consumption tracking, the following main concepts was outcropped: gage point, installation, measurement. A gage point implies a sensor measuring electricity consumption. A sensor fixes consumption during some time periods and regularly transmits measurements into the tracking system.

Measurements of a certain gage point may be received from different sources: a sensor directly (in the case of normal operation), an approximation algorithm (in the case of fail of short duration), a system user (in the case of sensor fail of long duration). At that, several measurements of the same gage point may pertain to the same time point. It means that, for example, a user does not agree with sensor measurements. In this case, a user enters alternative measurements. And within a system, for all gage points, it should be true the rule that, for any time point, there is only one measurement of alternative ones that is confirmed. Just confirmed measurements are reflected in accounts of electricity consumption. Other measurements are used for analysis of history and reasons of value divergence from different sources.

An installation implies a unit as well as a unit part or a unit group. Installations are organized into a hierarchy according to the principle “part-whole”. A shop as a whole is considered as a special case of an installation including all other installations of the shop. Each gage point is placed on only one installation. At that, gage points may be placed not only on installations that are leaves in an installation tree. For instance, any plant may have auxiliary rooms not included into any installation of a shop. Gage points fixing electricity consumption within such rooms are considered to be placed within the shop itself and out of any installations constituting it.

A semantically complete model consists of relations each of which is represented by a phrase that is best understood for a UoD expert. Within a phrase, separate concepts are represented by word-combinations that are full and clear for experts; word-combinations, used as copulas between concepts, also reflect the essence of UoD from an expert’s point of view. Within a semantically complete model all relations are based on unique sets of concepts, therefore, using of copulas between concepts is not obligatory for formal accuracy of the model, but is obligatory for simplification of model usage and its representation in an intuitively clear form for an expert and a designer himself. For example, the fact that each measurement is characterized by a certain consumption value is recorded within a semantically complete model as “A Measurement is characterized by a Consumption Value”. Here, “Measurement” and “Consumption Value” represent concepts, the word-combination “is characterized by” represents the copula between them, and the phrase as a whole is the semantically complete relation. We have used the term “concept” as a synonym of the accepted conceptual modeling term “object type” (also known as “entity” or “set”) since within semantically complete modeling these terms represent one thing, as it was shown in [7]. Here it is necessary to note that relations may consists of more than two concepts.

Further, we identify and formalize relations and constraints of electricity consumption tracking UoD, anticipating them with necessary explanations. Constraints may be shown in a relation phrase as well as below it in square brackets with indent. By underlining a concept in a phrase, a mandatory constraint are imposed on an appropriate relation, for example, “A Measurement is characterized by a Consumption Value”. A mandatory constraint implies that within all states of a certain relation, each existent instance of a mandatory concept (underlined one) corresponds to at least one combination of the rest concepts of the relation. A concept instance is considered to be existent if it meets in at least one state of a relation of the semantically complete model.

A functional constraint implies that, within all states of a certain relation, each instance of a concept-determinant corresponds to at most one instance of a constrained concept. A functional constraint is formulated below a phrase in square brackets with the help of the arrow symbol “→”

from a concept-determinant to a constrained concept, for example, “[Measurement → Consumption Value]”.

In the framework of investigated UoD it is known that each measurement is obligatory characterized by one consumption value. This fact is written down in the following way:

A Measurement is characterized by a Consumption Value

[Measurement → Consumption Value]

Here and further, a text with a bold font represents the semantically complete model of the electricity consumption UoD.

An equivalent constraint implies that, within all states of a certain relation, one instance of a concept-determinant corresponds to at most one instance of a constrained concept and vice versa. An equivalent constraint is formulated below a phrase in square brackets with the help of the identity symbol “≡”, for instance, “[Consumption Value ≡ Positive Integer]”. It is necessary to note that an equivalent constraint is equal to two counter functional constraints.

In our case, each consumption value represents, obligatory and biuniquely, a positive integer. In the semantically complete model it is written down as:

A Consumption Value represents a Positive Integer

[Consumption Value ≡ Positive Integer]

Further, let us reflect the fact that each measurement obligatory pertains to a certain time period and one gage point:

A Measurement pertains to a Time Period

[Measurement → Time Period]

A Measurement pertains to a Gage Point

[Measurement → Gage Point]

Let us formulate some trivial relations for the concept “Time Period”, which will be necessary when we will formalize complex constraints below. A time period has a beginning time point and an end time point which are, obviously, time points:

A Time Period has a Beginning Time Point

[Time Period → Beginning Time Point]

A Time Period has an End Time Point

[Time Period → End Time Point]

A Beginning Time Point is a Time Point

[Beginning Time Point ≡ Time Point]

An End Time Point is a Time Point

[End Time Point ≡ Time Point]

If a concept has a fixed quantity of possible instances, it is reflected in square brackets after a phrase by means of enumeration of instances in the braces after the equality symbol “=”.

Therefore, the fact that each measurement obligatory pertains to one of three measurement sources (“Sensor”, “Algorithm”, “User”) is represented in the following way:

A Measurement pertains to a Measurement Source

[Measurement → Measurement Source]

[Measurement Source = {“Sensor”, “Algorithm”, “User”}]

Each measurement obligatory has an confirmation of one of two types: “yes” or “no”:

A Measurement has a Measurement Confirmation

[Measurement → Measurement Confirmation]

[Measurement Confirmation = {“Yes”, “No”}]

For the given relation, the following constraint is fulfilled: among all measurements of one gage point pertaining to one time point, only one measurement may be confirmed. In other words, for all time points, it is true that a time point may correspond to only one confirmed measurement

taken in this time point (and vice versa). Within the semantically complete model it may be formulated as follows:

[((Time Period–Measurement–Gage Point), (Measurement–Measurement Confirmation = “Yes”), (Time Point(2)–Beginning Time Point–Time Period–End Time Point–Time Point(3)), (Time Point(1) >= Time Point(2)), (Time Point(1) <= Time Point(3))). (Measurement, Gage Point, Time Point(1)): Gage Point ≡(Time Point(1))≡ Measurement]

The formulated constraint consists of two parts: before a colon, a semantically complete query calculating a relation is given, and after a colon, a constraint imposed on the calculable relation is represented. Let us explain the process of calculation of the relation (more detailed description of structure of semantically complete queries is given in [7]). At that, a relation will be designated shortly with enumeration of concepts in round brackets, for instance, (Measurement, Time Period) is designation of a relation defined by the phrase “A Measurement pertains to a Time Period”.

The phrase (Time Period–Measurement–Gage Point) represents a composition of two binary relations: (Time Period–Measurement) and (Measurement, Gage Point). Similarly, (Measurement–Measurement Confirmation) is a composition of one binary relation (that is the relation itself), and (Time Point(2)–Beginning Time Point–Time Period–End Time Point–Time Point(3)) is a composition of four binary relations.

In this query the concept “Time Point” is used in two roles: as a time point of period beginning and as a time point of period end. Therefore, within the query, there is role indexes allowing to use the object type “Time Point” as two role object types: “Time Point(2)” and “Time Point(3)”. Role object types are considered independently; it allows to formulate complex queries in the more compact form.

A relation on condition represents a relation which has a state that consists of cohesions (rows, records, tuples) satisfying a certain condition. For example, (Measurement Confirmation = “Yes”) is a relation on condition based on the single concept “Measurement Confirmation” and consisted of a single cohesion containing one concept instance: “Yes”. In our case, the phrase (Measurement–Measurement Confirmation = “Yes”) should be considered as composition of two relations: the relation on condition (Measurement Confirmation = “Yes”) and the relation (Measurement–Measurement Confirmation). Actually, it represents a selection from the last relation of all cohesions satisfying the condition.

We have considered one method of composition designation: with the help of the dash symbol ‘-’. This method may be applied to binary relations only. A more general way of composition designation is relation enumeration via comma. For example, ((Time Period–Measurement–Gage Point), (Measurement–Measurement Confirmation = “Yes”), (Time Point(2)–Beginning Time Point–Time Period–End Time Point–Time Point(3))) is a relation calculated as a composition of resulting relations of intermediate compositions discussed above: (Time Period–Measurement–Gage Point), (Measurement–Measurement Confirmation = “Yes”), and (Time Point(2)–Beginning Time Point–Time Period–End Time Point–Time Point(3)). This method allows to define a composition of relations of arbitrary arities.

It is necessary to note that a criterion of column superposition is coincidence of role object types in all composition cases. For example, during fulfillment of the composition ((Time Period–Measurement–Gage Point), (Measurement–Measurement Confirmation)) the object type “Measurement” superposes, in the resulting relation of the composition, all the object types “Measurement” of base relations. But during fulfillment of the composition (Time Point(2)–Beginning Time Point–Time Period–End Time Point–Time Point(3)), the superposition of the object type “Time Point” does not take place since it is represented as two different role object types.

A state of the relation (Time Point(1) >= Time Point(2)) consists of all possible cohesions (on the logical level) within which the first time point follows after the second one or is equal to it. A state of the relation (Time Point(1) <= Time Point(3)), conversely, consists of all cohesions

within which the first time point precedes the third one or is equal to it. A state of the composition of these relations ((Time Point(1) \geq Time Point(2)), (Time Point(1) \leq Time Point(3))) consists of cohesions for which it is true that the first point time is between the second one and the third one, including them.

Thus, the composition of all described relations ((Time Period–Measurement–Gage Point), (Measurement–Measurement Confirmation = “Yes”), (Time Point(2)–Beginning Time Point–Time Period–End Time Point–Time Point(3)), (Time Point(1) \geq Time Point(2)), (Time Point(1) \leq Time Point(3))) calculates, for each confirmed measurement, those time points (with the index 1) which are between a beginning and an end of the confirmed measurement, including boundary time points.

A projection of a relation is designated by a point with a following enumeration of appropriate role object types via comma in round brackets. For instance, “. (Measurement, Gage Point, Time Point(1))” is the projection of a relation on the role object types “Measurement”, “Gage Point”, “Time Point(1)”. As a result of application of this projection to the last foregoing composition, we obtain a relation containing, for each gage point, all confirmed measurements of it and, for each confirmed measurement, all time points between its beginning and end.

The last phrase of the considering complex constraint is “Gage Point \equiv (Time Point(1)) \equiv Measurement” following after a semicolon. This phrase states about imposing an equal constraint on the relation obtained by fulfillment of the foregoing query. The equal constraint states that, for all time points, it is true that one gage point may correspond to only one confirmed measurement and vice versa. A role object type, to which a universal quantifier is applied, is placed in round brackets framed by a symbol of a simple constraint; this notation states correctness of this simple constraint in the context of each instance of the role object type placed in the brackets. In our case, within the calculable relation in the context of each instance of the role object type “Time Point(1)”, there exists an equivalence dependency between the role object types “Gage Point” and “Measurement”.

So, the above complex constraint may be formulated as a whole in the following way: for each time point it is true that each gage point corresponds to at most one confirmed measurement, time period of which contains the time point, and each confirmed measurement, time period of which contains the time point, corresponds to at most one gage point (the second part of the statement follows from functional dependence of a gage point upon a measurement). As we can see, the length, and the complexity, and even the structure of the verbal constraint formulation and the formal one are very close. It allows for an information system designer to think and to formulate a UoD model with the same terms, and for UoD experts to comprehend a semantically complete model with a minimal prior training.

Further, using known constructions, we formulate a semantically complete relation stating that each gage point is situated on one installation:

A Gage Point is situated on an Installation

[Gage Point \rightarrow Installation]

A semantically complete model does not allow for one concept (object type) to be included into the same relation more than once. For instance, we can not define a hierarchy of installations with the help of the relation “An Installation is included into an Installation”. For modeling a general form graph or its particular cases (for instance, a tree) within a semantically complete model, it is necessary to introduce concepts that are equivalently connected with nodes or arcs of the graph. In our case, we should define the concepts “Super-Installation” and “Sub-Installation”. It allows to formulate a hierarchy connection between them in the following way:

A Sub-Installation is an Installation

[Sub-Installation \equiv Installation]

A Super-Installation is an Installation

[Super-Installation \equiv Installation]

A Sub-Installation is included into a Super-Installation

[Sub-Installation → Super-Installation]

3. Modeling of Installation Electricity Consumption Calculation

Further we consider the problem of electricity consumption calculation of separate gage points and installations as a whole. Let us begin from a gage point consumption. For each calculated gage point consumption, it is true that it obligatorily relates to a certain gage point, a certain time period, and is characterized by a certain consumption value. Within the semantically complete model, it is formulated in the following way:

A Gage Point Consumption relates to a Gage Point

[Gage Point Consumption → Gage Point]

A Gage Point Consumption relates to a Time Period

[Gage Point Consumption → Time Period]

A Gage Point Consumption is characterized by a Consumption Value

[Gage Point Consumption → Consumption Value]

At that, the following constraint is fulfilled: a consumption value is equal to a sum of consumption portions calculated for all confirmed measurements related to the given gage point and the given time period. Each consumption portion represents a number that is equal to product of two other numbers. The first number is a consumption value of the given confirmed measurement. The second number is a ratio of duration of intersection of two time periods to duration of the second time period, where the first time period represents a time period of the given measurement, and the second one represents a time period for which a consumption of the given gage point is calculated. This constraint is formulated in the following way by means of the semantically complete query language:

**[((Consumption Value(1)–Gage Point Consumption–Time Period(1)), (Gage Point Consumption–Gage Point–Measurement–Measurement Confirmation= “Yes”), (Time Period(2)–Measurement–Consumption Value(2)), (Portion(Time Period(1), Time Period(2)) * Consumption Value(2)))
 (Consumption Value(1), Gage Point Consumption, SUM(Portion(Time Period(1), Time Period(2)) * Consumption Value(2)))):
 Consumption Value(1) = SUM(Portion(Time Period(1), Time Period(2)) * Consumption Value(2))]**

In this constraint, constructions already known for us are used: a binary composition, a general composition, a relation on condition, a role index, a projection. Additionally, it is defined the function “Portion”, a relation on formula and a relation grouping with calculation of the aggregate function “SUM”.

The function “Portion” calculates a number on the basis of two time periods. The number is equal to ratio of a duration of intersection of input time periods to a duration of the second time period. It may be formulated as the following constraint:

**[((Time Period(1)), (Time Period(2)), (Number = Portion(Time Period(1), Time Period(2)))):
 Number = Length(Time Period(1) U Time Period(2)) / Length(Time Period(2))]**

Within the given constraint, the function “Length” is used for calculation of a period duration. The function “Length” calculates, on the basis of input time period, a difference between its end time point and beginning time point. This function is introduced with the following constraint:

**[((Time Point(2)–Beginning Time Point–Time Period–End Time Point–Time Point(3)), (Number = Length(Time Period))):
 Number = Time Point(3) - Time Point(2)]**

Within the function “Portion” definition, the intersection function “U” is used as well. An intersection of two time periods consists of all time points which are included into both time periods:

[((Time Period(1)), (Time Period(2)), (Time Period(3) = Time Period(1) U Time Period(2)), (Inclusion(Time Period(1), Time Point), Inclusion(Time Period(2), Time Point))): Inclusion(Time Period(3), Time Point)]

In the last constraint, for defining relations on conditions (Inclusion(Time Period(1), Time Point)) and (Inclusion(Time Period(2), Time Point)), the predicate “Inclusion” is used which states inclusion of a time point into a time period. This predicate, actually, represents a relation calculated in the following way:

[((Time Point(2)–Beginning Time Point–Time Period–End Time Point–Time Point(3)), (Time Point(2) <= Time Point), (Time Point <= Time Point(3))): Inclusion(Time Period, Time Point)]

Thus, the given predicate states that a time period is constituted of all time points, that are between a beginning of the time period and an end of it. All the introduced functions and the predicate were used for shortening of the initial constraint which, in principle, may be formulated without them, but in a more verbose form. Let us underline the fact that all the functions and the predicate were introduced with the help of the universal mechanism of constraints.

Further, we consider the relation on formula (Portion(Time Period(1), Time Period(2)) * Consumption Value(2)) used in the initial constraint. It is considered that a relation defined with a formula is based not only on role object types mentioned in the formula, but also on an additional role object type being a formula calculation result. Since the given expression contains two functions (“Portion” and “*”), the relation representing a calculation result is built on the base role object types “Time Period(1)”, “Time Period(2)”, “Consumption Value(2)”, and the role object types being the function calculation results: “Portion(Time Period(1), Time Period(2))”, “Portion(Time Period(1), Time Period(2))*Consumption Value(2)”.

Let us consider a grouping with calculation of an aggregate function. In our constraint, it is “.(Consumption Value(1), Gage Point Consumption, SUM(Portion(Time Period(1), Time Period(2)) * Consumption Value(2)))”.

Using of an aggregate function automatically implies fulfillment of grouping on the rest role object types of a relation. In our case, after execution of the projection “.(Consumption Value(1), Gage Point Consumption, SUM(Portion(Time Period(1), Time Period(2)) * Consumption Value(2)))”, the function “SUM” is applied to the last role object type. As a result, the given relation is grouped on the role object types “Consumption Value(1)”, “Gage Point Consumption”; and, for each combination of their instances, it is calculated one instance of the additional role object type “SUM(Portion(Time Period(1), Time Period(2)) * Consumption Value(2))”. Using of several aggregate functions on the same relation is possible as well; at that, the functions are calculated for instance combinations of role object types not covered by the functions.

In our opinion, the proposed method of designation of grouping and aggregate function calculation is more compact and intuitively clear than the method of apparent indication of grouping with the help of the phrase “group by” (as it is accepted, for instance, in SQL).

Further, we consider calculation of electricity consumption for installations as a whole. An installation consumption relates to an installation, a time period, and is characterized by a consumption value:

An Installation Consumption relates to an Installation

[Installation Consumption → Installation]

An Installation Consumption relates to a Time Period

[Installation Consumption → Time Period]

An Installation Consumption is characterized by a Consumption Value

An installation consumption value is calculated on the basis of consumption values of separate gage points and, recursively, on the basis of consumption values of sub-installations: a consumption value is equal to the sum of consumptions of sub-installations for the same time period, plus the sum of consumptions of gage points placed on this installation for the same time period:

[[((Time Period–Installation Consumption(1)–Installation(1)–Super-Installation–Sub-Installation–Installation(2)–Installation Consumption(2)–Time Period), (Installation Consumption(2)–Consumption Value(2))), (Installation Consumption(1), SUM(Consumption Value(2))), ((Time Period–Installation Consumption(1)–Installation–Gage Point–Gage Point Consumption–Time Period), (Gage Point Consumption–Consumption Value(3))), (Installation Consumption(1), SUM(Consumption Value(3))), (Installation Consumption(1)–Consumption Value(1))) : Consumption Value(1) = SUM(Consumption Value(2)) + SUM(Consumption Value(3))]

During formulation of the given constraint, we used already known constructions: a binary composition, a general composition, a role index, a projection, a grouping with calculation of an aggregate function. The given constraint may be decomposed with the help of one or several auxiliary functions. We did not do it to demonstrate the possibility of formulation of complex constraints with using the semantically complete query language without introducing auxiliary functions.

For finishing the formalization of UoD of electricity consumption tracking introduced above, we formulate a relation reflecting the thesis that each shop represents a special case of an installation:

A Shop is an Installation

[Shop ≡ Installation]

Thus, we create the semantically complete model of electricity consumption tracking UoD containing interconnections between concepts only, without any realization details. The model has the intuitively clear form and may be examined by experts after a brief study of principles of its organization. On the basis of the model, it may be developed: a data model, data model constraints, a part of business logic. But these questions are out of the given article.

4. Summary

One of the main properties of semantically complete model is that, for formalization of a model, constraints and queries, there is no necessity to use relation designations. In conjunction with semantic naming of object types (concepts), it allows maximally to advance the structure of a model and queries to the structure of natural language phrases. As a result, a designer may think, communicate with UoD experts, and model in the same terms, and an expert may master a model essence very vast.

Since a semantically complete model can not contain relations that define interconnections of the same concept set in different ways, the model may be studied in the top-down way without rebuilding of already formed representations about concept interconnections. It is necessary to note that this property does not adversely affect the model expressiveness, and any UoD may be modeled by means of this approach.

The mechanism of role indexes allows to formulate complex queries in the compact way, decreasing a quantity of nesting levels of query constructions. At that, the fulfillment of queries are based on the single principle of relation column superposition, according to coincidence of role object types, that is the forward step in comparison with the widespread approaches: with name coincidence or by means of pointing column coincidence in the apparent way. In conjunction with possibility to define relations on logical conditions, it allows to generalize any selection as a composition of two relations (one being selected and one being conditioned). Additionally, it is

possible to define relations on formulas, to group relations and to calculate aggregate functions, that creates conditions for formalization of complex dependences between concepts.

The possibility to define functions and predicates allows to increase the compactness of semantically complete queries. At that, structure and size of queries and constraints extensively approaches to formulation of the same on a natural language. We think that it creates new directions of research in the field of reflection of a text to a semantically complete model and in the field of self-learning algorithms.

In the conclusion, it is necessary to note that, in spite of that semantically complete model is initially positioned as a data conceptual modeling technique, existence of the powerful query and constraint language creates prerequisites to its usage as a knowledge modeling technique. For instance, the semantically complete model within the paper is only partially reflected to a data structure (some of model relations) and data structure constraints (some of functional and equivalent constraints). The rest part is reflected to an applied logic, for instance, calculation of gage point and installation consumption; and to knowledge of UoD, for example, calculation of intersection of two time periods was defined in the declarative way (see above) and can not be directly used for effective intersection calculation, but may be used for creation and verification of such effective calculation algorithm.

References

1. 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.
2. 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.
3. van Griethuysen J.J., editor. *Concepts and Terminology for the Conceptual Scheme and the Information Base*. Publ. nr. ISO/TC97/SC5-N695, 1982.
4. Halpin T.A. *Conceptual Schema and Relational Database Design*. Prentice-Hall, Sydney, Australia, 2nd edition, 1995.
5. Ovchinnikov V.V. A Conceptual Modeling Technique without Redundant Structural Elements // *Journal of Conceptual Modeling* (www.inconcept.com/jcm), 29, 2003.
6. Ovchinnikov V.V. Improving Controllability of Vast Conceptual Models // *Journal of Conceptual Modeling* (www.inconcept.com/jcm), 31, 2004.
7. Ovchinnikov V.V. A Semantically Complete Conceptual Modeling Technique // *Journal of Conceptual Modeling* (www.inconcept.com/jcm), 32, 2004.