Semantic Databases and Semantic Machines

Introduction

In my previous paper [2] I introduced ideas with the goal of constructing a database model based on the highest level of abstraction, i.e. based on the conceptual model. Because of this, the conceptual model is independent of any other data model. Additionally, the conceptual model greatly influences the construction of the semantic model, for which we will show that meaning is not only related to a sentence. Meaning has a much more complex structure and is related to concepts, thoughts and knowledge.

Due to the constraint of the number of pages imposed by the publishers, [2] was limited to a small number of pages and too few explanations. For this reason, we will now elaborate on the main ideas of [2], to which we will add improvements and new solutions.

In this paper I work with terms of a most general character. Of course, my intention was not to give a complete theory of these complex terms but instead to begin with some ideas, constructions and solutions in this field.
I have introduced some important scientific fields into the domain of database theory because I think that the theory of databases is best suited for these fields. My data model especially opens up possibilities for these other fields.

In my papers I generally accept the results obtained by G. Frege. For the part of Frege's work that is disputable, I accept the approach that holds that this part needs to be fixed, rather than dismissed.

1 Abstract Objects

In [2] I introduce m-attributes, m-entities, m-relationships and m-states and create a more formal approach to these very complex objects. These objects are interpretations and abstractions of their corresponding real world objects. I name them abstract objects. Note that the four mentioned "m" objects are objects of a most general character.

For the purpose of modeling real world objects, concepts are introduced. They are abstract objects constructed in our minds.

The prefix "m" is introduced to indicate that m-attributes, m-entities, m-relationships and m-states are in memory (computer memory, human memory, etc.) Given that this memorization takes place in "real memory", it follows that these objects are located in the real world. Aside from this, abstract objects have another shared trait: they have meaning for a corresponding subject. Therefore, in this paper, abstract objects are understood to be real objects from the real world. In contrast to other objects from the real world, we determine and construct these abstract objects using our mental activities, in which concepts and semantics play a large role.

In my model when I say that abstract objects are written to memory, I mean they are written in both database memory and a subject's memory. By subject, I most often mean the person working with the database. However, in a general case this can mean a machine, robot, processor,

etc. Obviously, the point is not that an abstract object needs to be literally written in a subject's memory, but rather that the subject always knows how to retrieve or come to the corresponding abstract object in a memory. A subject can always "retrieve" an abstract object from memory; identify it; place it within a linguistic construct; write it into an auxiliary memory; and operate with it. Thus, data in a database always:
(i) is a subjective image
(ii) depends on the abilities of the subject

The link between the knowledge of a subject and the abstractions that the subject makes from the attributes of the corresponding real object is mathematically formalized in relation (3.3.3) of [2].

This relation refers to m-attributes, for which the following holds true:
(i) m-attributes are direct (perceptual) interpretations of attributes of a real world entity.
In contrast to m-attributes, constructions of m-entities, m-relationships and m-states are neither direct interpretations nor direct abstractions of the corresponding entities. They are complex constructions, made up of other abstract objects.
(ii) m-attributes are constructed with the help of "universal" and "particular" attributes. I introduce universal attributes as constructs of our abilities, which enable us to identify attributes of entities (or relationships). The attributes of a particular entity or relationship are named "particular" attributes (See Section 3 of [2]).
Note that each person gets most of his knowledge from other people.

1.1 Properties of Abstract Objects
In [2] some new properties of abstract objects are defined. These properties determine a new approach to the theory and understanding of abstract objects. These newly introduced properties are:
(i) Abstract objects are recorded, meaning that they are permanent.
(ii) Abstract objects hold meaning for corresponding subjects.
(iii) Abstract objects differ in the way they are constructed.
1. The simplest of them – m-attributes, are direct abstractions of the real world through our perceptual abilities.
2. More complex abstract objects are constructed from simpler ones. For instance, m-entities are constructed from m-attributes.

Therefore, hierarchy among abstract objects is determined by the level of abstraction of their corresponding real world objects.
(iv) We identify abstract objects through their identifiers.

1.2 The Meaning of Abstract Objects
On a semantic level we use G. Frege's notation of an object. Frege's notation of an object characterizes an object as the kind of thing which can be the referent of a name.

Abstract entities have meaning for us because an abstract entity is recorded and this record can be actual or not actual for a subject. Paying attention to an entity makes it actual and allows it to be meaningful (an actual entity is defined in 4.2.3). In my approach, meaning is not only the relation

of words and sentences to reality, but also the relation of sentences to facts, knowledge and thoughts.
Meaning is correlated to man's ability to recognize the truth value of a sentence. Relation (3.3.3) from [2] is also determined in this sense.
The relation (3.3.3) represents a formalization of the identification of certain entity's attribute. In [2] identification of entities, relationships and states is similarly defined using a recursive definition.

Conclusion: A subject has permanently recorded abstract objects. These abstract objects denote corresponding real world objects. Abstract objects are parts of facts, and facts determine meaning of the factual sentence.

1.3 Entities, the World and Propositions

In [2] I wrote "We accept that the world is discrete, i.e. that it consists of things or individuals. These things or individuals are called entities." Propositions are about entities (or about individual things).

1.4 Binary Structures

I call a structure that has an identifier of an entity (or relationship) and one attribute a binary structure. I also use the term binary structure when we associate knowledge about an attribute to that attribute. Thus, a binary structure has two wholes – an identifier and an attribute, along with possible associated knowledge. Note that in a general case my model can include associated knowledge about the identifier.
In my papers I use binary entities, binary relations and binary files.
When we use the term attribute from a data structure, we actually refer to the name of the attribute. I use names of objects as determined by G. Frege.

2 Facts

In my model, an m-attribute recorded into a database is called data.
If we memorize this m-attribute in our (subject's) memory, then we consider it a fact because it is an m-attribute of a corresponding m-entity.
A subject is aware of facts. Therefore, we define facts as something of which we are aware.

In [2] (see Sections 3.7, 3.8) two more types of facts are defined. In 3.7, I define facts that are related to data (i.e. that are related to events in memory), while in the 3.8 I define facts that are related to events that happen to attributes of real entities or relationships. These two types of facts also create elementary sentences, which are the smallest complete semantic units.

2.1 Facts and Factual Sentences

In my model facts are on the thought level. We construct factual sentences from facts. Thus, I differentiate facts and factual sentences. Factual sentences in this data model are sentences that correspond to and are determined by binary structures of entities or relationships. Factual sentences express the corresponding facts.

In my data model, a fact is "awareness" about how a binary structure is constructed. Thus, a fact "knows" that a certain real entity is constructed from attributes and that the attributes are basic components of that construction. Generally speaking, a fact carries knowledge about the construction of real world entities.

2.2 Existence of Extensions. Russell's Paradox.

Let us note that (3.3.3) from [2] always warrants the existence of an extension for the given concept. Relation (3.3.3) makes Russell's paradox impossible and gives conditions under which we can construct extensions using properties. Relation (3.3.3) says that we cannot build a concept for which we do not have the corresponding abilities.

Naturally, we can change (3.3.3) in the sense that this link requires other conditions for the extension of a given concept. However, that which is important is that (3.3.3) shows that a concept is not a one-dimensional construction, but rather a complex construction linked to a series of other constructs.
As we know, Russell's paradox was overcome in Set Theory, with the creation of a very complex axiomatic system.

2.3 Construction of a Fact about an Entity

In previous sections we defined the construction of a fact. Note that in these definitions there is a "Universal" attribute that we call an m-attribute and a corresponding attribute of the entity, which upon memorizing, we also call an m-attribute. The attribute of the entity (i.e., the Particular Attribute) matches the Universal Attribute. Both attributes are abstract objects. We can note that the Universal Attribute is the identifier for the corresponding entity's attributes. We call this entity's attributes Particular Attributes. The definition for facts about relationships is similar.

Conclusion: A fact is constructed from an entity's binary structure. The construction of a binary structure is clear and simple. It consists of one entity and one element of knowledge about this entity (or one element of knowledge about the entity's attribute). This entity is determined by its identifier. The construction for facts about relationships is similar.

3 Relationships between Facts and Natural Language

We can note that an attribute which becomes the component of a fact is always related to its entity (or relationship).
Thus, "red" is a universal attribute, while in the sentence: "The color of the car is red", red refers to the entity car. A fact about an entity is determined by three things: an attribute, a property and an entity. These three things are components which form the corresponding factual sentence. "The color of the car is red" expresses the corresponding fact about the entity. Note that properties and entities are concepts in our data model.

3.1 Procedure for the Construction of a Factual Sentence

The procedure that constructs a factual sentence is as follows: first, we memorize the fact in our mind, and then we can form a factual sentence that expresses the fact. Factual sentences always express something about the world. A fact is a possessor of truth and becomes stored i.e. becomes permanent. In contrast to facts, factual sentences only denote the truth. Facts about entities are atomic, they are not composed. We also say facts about entities are primitive.

3.1.1 Language Machine

We will now formalize the construction of two types of sentences:

(i) A sentence that expresses a fact about an entity
As we already said, a fact carries knowledge about the construction of real world entities. Because of this, a factual sentence has a grammatical structure that is determined by the construction of a real entity; that is, a factual sentence is a simple sentence that has a subject (entity-with-identifier) and a predicate (is-attribute).
We can represent above sentence with a function-argument procedure. For example the following sentence: The color of the car is red can be processed as:

is-red (car-with-identifier-id1) = T

This procedure is applied to the entity car-with-identifier-id1 identified by id1 and it gives "true" or "false".

(ii) A sentence that expresses a fact about a relationship (between two entities).
In this case, a factual sentence is a simple sentence that has a subject, predicate and object. As an example of "relationship" we take "marriage". Thus the sentence "John is married to Mary" expresses the relationship "marriage".

The above two cases show that the construction of factual sentences is simple. This construction is based on binary structures from databases.
In contrast to my solution, databases that have complex structures are not suitable for the construction of sentences.

The "function-argument" construction from 3.1.1.(i). is G Frege's idea. He also defined the denotation of a sentence to be a truth value. However I use the identifier of the entity in 3.1.1.(i), in contrast to Frege, who uses the name of the entity. Note that the identifier in my data model can identify (or help to identify) the meaning of an entity (or relationship).
G. Frege's "function-argument" approach is important for another result of his, which has to do with semantic compositionality.

3.1.2 Construction of Elementary Facts

We construct elementary facts in relation to real constructions from the real world.
We show how to decompose an entity (or relationship) into binary structures. We also show how to decompose states of entities and relationships into binary structures and how to maintain their history. Here we show that the corresponding atomic propositions are generated from binary structures of entities (or relationships).
All of these facts are related to entities or relationships. Thus, we construct elementary facts in relation to real constructions from the real world. We create elementary sentences on the basis of these facts. A compound proposition is made up of factual sentences. We determine the truth value of compound propositions using logic. We determine the truth value of a factual sentence as is explained in Section 3.2.1.
On the basis of our model i.e. on the basis of binary structures, we determine that a factual sentence is an elementary semantic unit. Our model shows that factual sentences are formed through a complex process of interpreting the real world using concepts, facts and awareness.

Conclusion: In this section, the basic thesis is that a factual sentence is constructed based on the subject's construction of binary structures of entities, relationships and their states. We know how an entity is constructed and using factual sentence we actually describe this construction.

This thesis explains certain important problems in human communication and in the use of language. For instance, it can be used to explain how ideas, thoughts and semantic content are conveyed between two people.
This thesis helps to understand how human beings might learn language and how we understand new sentences which we have never heard before**.**
It also explains how different people understand which thought is expressed by a sentence.
Note that we describe only factual sentences that relate to entities, relationships, attributes, states, and events.

3.2 Meaning and Truth of Factual Sentence

Our data model determines states and histories of entities and relationships. It shows that the construction of a sentence is based on the following:
(i) A factual sentence is a basic logical-semantic unit that corresponds to binary structures.
(ii) The construction of a factual sentence is based on the real world and the way we interpret it.
(iii) With one fact, the following is determined:
 (a) The construction of a factual sentence
 (b) The construction of the truth value of a factual sentence
 (c) The construction of the meaning of a factual sentence
 (d) The construction of the relation between the truth value of a factual sentence and its meaning

We will explain these constructions using the definition of truth given by A. Tarski.

3.2.1 Definition of Truth (A. Tarski)

This definition states: “P” is true if, and only if, P.

We will use the following example, with which Tarski explains this definition:
“Snow is white” is true if, and only if, snow is white.
In this definition, the meaning of the above sentence tells us that we are speaking about the entity snow from the real world and its attribute, white. Thus we conclude that the sentence (under quotations) is true, iff it is true that the real entity snow is white in color.

However, according to (i) – (iii) in 3.2, the following details are missing from Tarski’s definition:

1. We (human beings) inherently know (understand) the construction of *binary* structures of entities and relationships. Tarski’s example is actually a construction which consists of the entity snow and its property, color.
2. Based on this construction, we know that *snow* refers to the real entity snow, and that *white* refers to its real color. In our data model we have the corresponding concept of the entity snow. We also know that this concept has the property color. With knowledge that is based on this construction a subject can generate the meaning of the corresponding factual sentence.
3. Now that we know the real objects to which this sentence refers, we can denote its truth value. Generally speaking, the meaning of this sentence determines its truth value.

In the construction of the sentence and its meaning, the construction of the entity itself plays an important role. Factual sentences describe the construction of the corresponding entity (or

relationship). In papers [1] and [2] we showed how to construct entities using binary structures, mostly through binary concepts and binary relations.
Binary structures also link truth value and meaning; they state which objects from the real world are in a binary structure with each other.

Thus, the meaning of each factual sentence is determined by the construction of its corresponding fact. The meaning of the words that make up the factual sentence is also determined by the construction of the fact.
For instance, in the sentence "The color of the car is red", color is a concept of the property of the entity car. The car is a concept. Red is the name of a particular color. The corresponding binary structure completely determines the construction of the sentence. It also determines the meaning of the sentence. We clearly know what each word in the sentence means in the real world. A fact possesses the truth value of a sentence.
In place of the above sentence we can write: "The car is red". In this instance, we can note that by leaving out the abstract object "color" (i.e. the concept) the meaning of the sentence changes. The construction of the sentence also changes – instead of the noun "red", we now use "red" as an adjective.

Conclusion: A subject's knowledge about the construction of an entity generates meaning and truth of the corresponding sentence.

4. Awareness

Perhaps it may seem questionable to include a theory of awareness in the domain of database and mathematical theory.
However, awareness is above all related to data, facts and information. In only 20 years, since the appearance of the Internet, the amount of data and information has greatly increased.
Clearly many aspects of awareness belong to the domain of cognitive science and neuroscience; we will not talk about these here.
Database theory and mathematics can contribute results of a general character, which can be integrity tools for all sciences related to awareness.
I will first introduce the concept of awareness, upon which I will try to formalize it. In my data model we deal with perceptual awareness and the awareness of complex abstract objects, i.e. awareness of m-entities, m-relationships, m-states, and events.
Thus, perceptual awareness is the awareness of attributes of objects from the real world as opposed to awareness of abstract objects. Of course, abstract objects can have different levels of complexity.

4.1 Aspects of Awareness That Can Be Maintained and Supported by Databases

(i) One important property of awareness is the ability to identify real and abstract objects. In my data model, every object can be identified.
(ii) We hold that awareness is closely related to concept, because a concept means to have an idea about a thing. Considering that my data model is composed of facts, and that each fact is determined with a concept, then my model supports awareness in the sense that we are aware of each fact.
(iii) My model is an entirely temporal data model, which can maintain history of all data. This model especially maintains current and non current data. So in my model, awareness is not only about "Now".

4.2 The Formalization of the Notation of Awareness

I differentiate facts and factual sentences. In my data model facts are on the thought level, while factual sentences are on the language level. Therefore, the awareness of a fact is on the thought level.
However, we express awareness using language. We can note that the facts in my data model are represented by binary structures, which are determined by the reality. In Section 3, I explain that binary structures determine the corresponding atomic sentences. Therefore, atomic sentences are basic units about reality. Aside from atomic structures, my model also includes complex structures such as the states of an entity. Yet, I show that complex structures from my model can be decomposed into binary structures. Note that my model only contains attributes, entities, relationships, their states, and events. Now we can conclude that a factual sentence refers to reality and expresses a corresponding fact. This conclusion is in compliance with G. Frege's notation of three realms: the physical realm, the realm of semantics and the realm of thoughts.

4.2.2 Awareness of a Fact

We will call facts of which a subject is aware "actual facts". This is because a subject cannot be aware of all the facts he knows in a single moment.

In this data model all the knowledge about an entity is linked. The way this knowledge is linked is through the identifier of an entity. From the moment we get to know the identifier, we have access to the entire knowledge about the entity.
This identifier is linked to every attribute, entity and relationship that is linked to this entity. So if a subject is aware of a fact f, he is aware of all other facts related to f.

4.2.3 Actual Facts

There is a particular mechanism which enables a subject to be aware of facts. This mechanism has events that periodically happen and that cause the facts to become "actual". These events occur in different points in time. Facts are recorded in memory (database memory, subject's memory …), which means that they are permanent.

4.2.4 Expressing Awareness with a Factual Sentence

In my data model, each fact has a corresponding factual sentence. On the level of language I introduce a formalism of the factual sentence which expresses awareness of the corresponding fact.
A factual sentence that expresses awareness of the corresponding fact is a sequence of one reflexive sentence. Thus for a given factual sentence P, the following algorithm holds:

(reflexivity) … $(P \Rightarrow P) \rightarrow (P \Rightarrow P) \rightarrow ,\ldots, \rightarrow (P \Rightarrow P)$
where each $(P \Rightarrow P)$ corresponds to one event from 4.2.3. The symbol $\rightarrow$ means "next" in the sequence.

5. Identification

5.1 Misrepresentation or Lack of Representation of the Following in Current Database Theory:

(i) The term "key" has been used in an exclusive way. I consider that this term has been wrongly interpreted. The current database theory sees a key as part/set of parts of an entity. However, in my model the key is part of the subject's knowledge. The key in my data model is treated as a unique whole. It has a dual role - it identifies the corresponding unique entity in the real world and the corresponding unique structure in a database. Therefore, I introduce the term identifier and in my data model the identifier is a subject's addition to a entity. Therefore, a subject can add an identifier to a real entity.

In my data model, the identifier identifies the knowledge about a corresponding entity, relationship or state. It also can identify the meaning of the entity, relationship or state.

(ii) Today's database theory holds that an entity (that is, the corresponding relation) must be normalized. Contrastingly, in our data model the general case is that the entity has intrinsic properties. This means that properties of the entity take values freely. This entity corresponds to a relation which has mutually independent attributes. Once again, this is a general case in database design. If the database designer needs to introduce constraints in his application, then he needs to define those limitations on the mentioned entity with intrinsic properties. For example, he can define functional dependencies, after he has defined the entity (in the general case).

(iii) The identifier should be in accordance with the corresponding key. The identifier should have the same number of the possible values as the key does. We can write the attributes as a tuple or as a concatenated string. In both cases, the number of possible keys is the same. Current database theory does not have any rules for identifiers.

(iv) In our model data does not get deleted nor updated. All the data is saved so that there are no anomalies resulting from deleting, adding and updating.

(v) A concept and its extension are defined in data model for the first time. Our definition of a concept is not based only on attributes. There are no paradoxes in our extensions. For the first time, the following concepts of a most general character are introduced: concepts of properties, entities, relationships and states. Identifiers of the above concepts (with the exception of attributes) are also introduced.

I assume that there are two types of concepts. The first one is perceptual and it belongs to concepts of properties. The other concepts are more complex and constructed from concepts of a lower order, that is from abstract objects. In our model, knowledge belongs to the extensions of concepts.

(vi) My model is event oriented and is based on two events.

(vii) The model is not solely based on the real world because everything is recorded through a subject's interpretation. Subjectivity in the model is represented by knowledge.

(viii) This data model allows simple mapping between different data models. This is solved through two mappings – the first is a mapping of schemas, and the second is a mapping of data. Both of these mappings are maximally simplified because they are realized through binary structures and because they are completely determined by the corresponding identifier of state.

5.2 Decomposition into Binary Structures

In Section 4.2.2 of [2] we introduce binary concepts. In Section 6.5 we introduce Simple Form, which defines the conditions for the decomposition of arbitrary relation into binary relations. In my data model, the design of entities and relationships is always decomposed into binary structures. The above mentioned identifiers introduced in [1] and [2] play an important role in binary structures.

This has to do with an entire field that has been ignored in current database theory. In this paper, identifiers are of a fundamental character. In our model, we work on the identification of attributes, entities, relationships and states. We also work on the identification of real and abstract objects. The process of identification in our data model is integrated into the process of meaning.

5.3 Basic Terms

In this section we will attempt to define a few fundamentals related to identification. The following definitions of identification and of an identifier are from Sections 5 from [2] and Section 1.1 from [1].

Definition of Identification

In [2] we defined that the process of identifying goes from a subject to an entity (or relationship). In order to identify an entity by an identifier, two things are necessary: an identifier of the entity and a subject who has knowledge about the identifier of this entity.

Definition of an Identifier

Every entity has an attribute which is the identifier of the entity or can provide identification of the entity. This identifier has one value for all the states of an entity or relationship. See Section 1.1 from [1].

In this definition, "can provide identification of the entity" refers only to a surrogate key that can provide identification for the corresponding real entity. We do not pay attention to the surrogate key because it is not significant. Surrogate keys also have restrictions on the level of database design and semantics.

5.4 Three Possible Ways of Designing a Database with Regard to Identification

In database design one of the most important components is to solve the identification of database objects and real world objects. The next important step is how to find the corresponding real world object if we have identified its corresponding database object, and vice verse. In database design we can roughly observe three situations related to identifiers:

1. Identifiers only used for database objects
   We use the identifiers for database objects, but we do not use the same identifier for corresponding objects from the real world. Some call this identifier the "surrogate key", they think that all things are "unique acts of nature". However, the majority of entities that databases use are made by people, not by nature.
2. Identifiers which are used for both database objects and real world objects
   There are identifiers which are used both for real objects and for corresponding objects in the database. These identifiers are used as industrial standards that often are defined on an international level, for instance VIN numbers.
3. Identifiers which are used only for real world objects
   There are identifiers which are used for objects from the real world, but are not used in the database. This case is used when we identify real entities by identifiers rather than by attributes.

5.5 Leibniz's Law

We use Leibniz's Law on the level of database design. We determine the difference between two entities using Leibniz's Law.

In our model we only accept a limited form of Leibniz's Law. We will say that objects x and y are "logically" equal if the following is true $(\forall x, y)[x=y \Leftrightarrow (\forall P)(Px \Leftrightarrow Py)]$, where P is an intrinsic property of those objects.

5.6 A General Law That Determining the Difference between Entities

Leibniz's Law cannot give solutions for certain problems in database theory and practice. For instance, in most cases using Leibniz's Law we cannot differentiate between products with the same barcode. These products have the same properties and corresponding attributes, but are still different entities.
Therefore we accept that it is possible that two or more objects have all the same intrinsic properties. For this case we use General Law $(\forall x, y)[x=y \Leftrightarrow (\forall P)(Px \Leftrightarrow Py)]$, in which P stands for extrinsic and intrinsic properties of given object.
The extrinsic property is the property of a given object that is in relationships with other objects, for example, a location, or distance, etc. Most often we associate an identifier to the entities which have the same attributes, but which are physically different. In this way, by assigning different identifiers to these entities, the entities will become mutually different.

5.7 Keys and Identifiers

When every property in a relation is part of the primary key for that relation, the relation is referred to as an all-key relation. These attributes are mutually independent. Similarly, when every property of an entity is part of the entity's primary key, the entity is referred to as an all-key entity. These properties are intrinsic properties of the entity.

The general case in our data model is when all the attributes of entities are intrinsic, i.e. when the corresponding relations have mutually independent attributes. The primary key of these entities is an all-key. All other cases are special cases. In the database design phase, these special cases must be treated as constraints and should be precisely defined and solved. This general case makes an important difference between my approach and the current database theory.

If a relation is an all-key relation, then the relation can be rewritten in the form:
$R(id, a_1,\dots,a_n)$, where K = id. Note that we can apply the Simple Form to the relation R, i.e. the relation R can be written as n binary relations.
Of course, an id can have a set of values that is different from the set of values that has the key K. However these two sets must have the same cardinality.

Note that 5.1(iv) strictly holds true in our model; that we have introduced states and knowledge about entities and relationships; and that we exclusively use binary structures.

Conclusion: For the all-key relation R in our data model the following holds:

$$R(id, a_1,\dots,a_n) = R_1(id, a_1) \text{ join } R_2(id, a_2) \text{ join },\dots, \text{ join } R_n(id, a_n)$$

5.8 Identification of Entities

We defined identification for three groups of objects and in each we used a unique identifier (see Section 5 in [2]). They are the following:
(a) Entities (or relationships) determined by their intrinsic properties. In this case, we introduce an identifier as a property of each of these entities (or relationships).

(b) Entities (or relationships) which cannot be uniquely determined by their intrinsic properties, because some of them may have the same values for properties. Here we introduce the identifier, which is determined in 5.6. This identifier is also a property of the entity (or relationship).

(c) Wholes identified through their parts. Regarding our definition of Particular Attributes, we conclude: each intrinsic attribute of an entity has the same identifier associated it as does that entity.
This type of identification is done using the relationship of each of these attributes to the entity's identifier. This relationship determines that the attributes belong to the same entity. Thus, each of the entity's attributes has the same identifier.
We also say that each of these attributes is in a relationship with its entity. Therefore, if we have SchemaE (IdOfEntity, Property1 ,…, PropertyN), then each of Schema1 (IdOfEntity, Property1), Schema2 (IdOfEntity, Property2) ,…, SchemaN (IdOfEntity, PropertyN) is a schema of a relationship. Now, the construction of a new relationship between two relationships is determined with Schema1, Schema2,…,SchemaN. Finally, we show that the following holds:

SchemaE (IdOfEntity, Property1 ,…, PropertyN) =
Schema1 (IdOfEntity, Property1) relationship… relationship SchemaN (IdOfEntity, PropertyN)

Here, schemas have been used to represent concepts. The equality of the schemas, or concepts, is defined with the equality of the corresponding extensions (see 4.2.1, 4.2.2 and 4.2.3) of [2].

We can note that case (c) reminds us of Leibnitz's Law. However, here we have both the process of the identification of an entity (or relationship) and the construction of the entity's identifier. In Section 4.2.2 of [2] we implement this idea. We build entities with their intrinsic properties. This means an entity itself has these properties and that they are independent of other concepts. For instance, when we say, "the color of the car is red" we are talking about the entity's property. The term "intrinsic properties" corresponds to the term "mutually independent attributes" from the Relational Model.

We can now effectively realize an entity in the following two ways:
1. As a whole consisting of smaller groups, which contain one property and the entity's identifier
2. As a whole consisting of the entity's intrinsic properties and the entity's identifier

In 4.2.2 from [2], we show that these two representations are equal using relationships between entities, i.e. we demonstrate this using only a technique that belongs to the conceptual level.
In Example6 from [2], we use relationships between binary entities. This technique (see 5.12(ii)) is presented as a learning mechanism. Therefore, Example6 is a formalization of this learning mechanism, while 5.8.c is a formalization on the conceptual level.

5.9 Design of Entities and Relationships

5.9.1 Possible Cases in the Design Phase of an Entity

Case 1. All-key entity. As we already said, in our model this is a general case. Let the entity have n properties: a1, a2, … an. According to 5.7 and 5.8 (c) the following holds: E(id, a1,…,an) = E1 (id, a1) relationship…relationship E(id, an).

Case 2. All the entities from a set of entities have the same attributes. For instance, they are all Honda Civics with the same attributes. In this case we introduce the identifier id = VIN as a property of the entity. Now we have the entity E (id, a1, a2,…an) where a1,a2,…an are intrinsic properties, and we again use 5.7 and 5.8 (c) to get binary structures. We have the same situation with almost all industrial and business products. Note that we here use the law from Section 5.6.

Case1 and Case2 are general cases. Case1 uses the law defined in 5.5, while Case2 uses the law defined in 5.6.

Case3. The above mentioned Case1 can have constraints. For instance, a database designer can define functional dependence. In this case, we will construct an entity so that the corresponding relation is in BCNF. Now, the only new possible case is: An entity has n attributes, and the key K has m attributes (m<n). Then we put id =K and get:

E (id, a1,...,an) = E1 (id, a1) relationship ,…, relationship E (id, an)

(Note that 5.1(iv) strictly holds true in our model)

5.9.2 Possible Cases in the Design Phase of a Relationship

The construction of a key in relationships is always the same; it is determined in advance. The general scheme of a relationship has the following form: Rship (K, a1,…,an), which can be written as Rship (id, a1,…,an). Considering 5.1(iv), this scheme can be decomposed into binary structures.

In [2] a relationship between two or more entities is defined. It is understood that the following relationships are constructed in the same way:

(i) relationships between entities and relationships
(ii) relationships between relationships.

We conclude this section noting that here we have not considered states. As we mentioned, our design is based on Simple Form. In the following section we will discuss the states of entities or relationships.

5.10 Design of the state of an entity or relationship

In the previous sections we defined how to construct entities (relationships), so that the entity (or relationship) is "well" designed and that this design enables decomposition into binary structures. The current database theory has not dealt with this question, especially not on the conceptual level.

Once an entity (or relationship) can be represented as a set of binary structures, we can design its states. The identifier of a state is the key. Because all the states of the entity (relationship) are different, all the identifiers of state are also mutually different. Therefore, an identifier of state allows a straightforward construction of binary structures that represent states of entities or relationships.

5.11 Evolving Entities?

We will now define some significant matters related to the third group of objects mentioned in the above Section 5.8 (c).

a) Let us first note that we work on the conceptual level, thus we cannot directly use sets or relations (note that many papers use sets on a conceptual level??).
b) There is also the question of the nature of the identifier on the conceptual level. We can ask, "What is the identifier of an entity applied on the conceptual level?"

We define an entity by its intrinsic properties, meaning that the concept of a certain entity is satisfied by those instance**s** which always have the same properties. This implies that a database theory cannot be "evolving" in the sense that we can later add additional properties to one entity and have that entity remain the same entity. Therefore, it is impossible for two entities with different numbers of intrinsic properties to be the same entity.
c) The total number of possible entities that satisfy a particular concept is determined with all intrinsic properties that form the concept. This implies that each identifier of an entity is determined by the number of values of all its intrinsic properties. Therefore, the identifier of an entity and the identifier of its evolved entity cannot be the same. In conclusion, one entity can evolve only into a different entity, and these two entities must satisfy two different concepts. We mention this because existing database theories about "evolving entities" and "agile evolving" sometimes do not consider these facts.

5.12 Learning about an Entity

Another important matter related to entities is the process of a subject's learning about an entity. In the process of learning, one can accept that two different entities are actually one entity. The explanation for this is that the process of learning uses knowledge and concepts that evolve. In our data model, knowledge is a product of concepts (see 3.4 – 3.9 in [2]).

Evolving knowledge about an entity however is very different from the evolving of the entity. The answer to the question, "Can the process of learning about entities or relationships be represented in a database?" is "Yes". Our data model can be understood as a knowledge database. For the first time, a solution is given which only consists of atomic facts represented in a database. We have the past and the present recorded in the database, and we can store future states as well. We can maintain history, states and changes of knowledge on an atomic level.

We will now present one solution that represents the process of "learning about entities" in a database. In the process of defining an entity we predict that the subject gradually builds the entity's concepts. In this process of learning, one constructs a set of "intermediary concepts".

We will define the structure which represents the sequence of identifiers of "intermediary-entities" of an entity. In the learning phase, let this entity be represented with n "intermediary concepts" E1, E2… En. This structure has the following schema:

SchemaSequence (Identifier(i-1), identifier(i), identifier(i+1)); $i \in \{1,2,\ldots n\}$ … (seq)

If the above schema does not have a predecessor or successor, then we will assign a special value to the corresponding identifier.

Note that schema (seq) represents a sequence as a database structure. The scheme also determines the predecessor and successor for each member of a sequence.

The sequence (seq) can be used for saving the history of events. For instance, if we have a situation in which an entity is “stopped”, and then created again with another identifier, then the history of these changes can be solved using sequence (seq). Another example is if an entity is periodically created (for instance every year it gets a new identifier), then we solve history again through the application of (seq).
If we now combine sequence (seq) and schemas (4.2.6) from [2], which create all the states of an entity, we can solve very complex applications that maintain and keep the corresponding history. We can now conclude that learning about an entity contains elements of maintaining history.

5.13 Definition of Learning Mechanism for an Entity’s Concept

(i) Learning Mechanism is defined with the scheme given in (seq) and with the definition of knowledge given in schemas (4.2.6) from [2].

(ii) If a subject does not learn about an entity through a gradual evolution of his knowledge about it, but rather has the knowledge and skills to quickly create a concept of the entity, then he or she can use the following process:
In the first step, the subject determines n binary entities where each of these entities contains one property and one identifier. Considering that a subject knows (and aware) that all the properties of these binary entities are actually properties of one entity, he assigns all the binary entities the same identifier and creates relationships between all of these binary relationships. Each of these relationships has the same function; it interrelates properties to their entity. Thus, in this step, a subject, using the identifier of an entity, ties the properties of the entity into one concept.
We assume that this process is learning, because it is not achieved through a strong theoretical approach nor using an inferential integration. We assume that this process is the learning mechanism for an entity’s concept. Let us note that even an experienced database designer needs time to decide on the concept of an entity.

6 Events

Our model is event oriented. In terms of changes of state of entities or relationships, we define only two events: one that creates a new state of an entity or relationship, and one that “closes” the current state of an entity (relationship). Thus, these two events completely determine the state of every entity or relationship.

6.1 Definition of Time

We can also apply this approach with two events to define time. For instance, we can understand a “second” as a set of two events which occur on the entity “clock”: the event that creates a new second and the event that closes that second.

6.2 Duration of the State of an Entity

If we wish to measure the duration of a state of an entity in seconds, then we say its duration is equal to the corresponding number of events on the entity clock. This way, we use events, not units of time (seconds, minutes, etc) to measure duration. Therefore, the duration of an entity’s state is relative because it is always measured by events defining the states of another entity. Depending on the events we use to define states in the second entity, the duration of the state of the first entity can be different.

7 Database Solution that Manages Programs and Processes

(Section 7 was published on 10. April 2012.)

7.1 Introduction

(i) Here we will deal with imperative programming languages.
(ii) A program is a mathematical function, which has a clearly defined procedure, domain and range.
(iii) Each program has a set of properties, presented by the corresponding variables.

7.2 A New approach to Algorithms and Programs

My new approach is based on concepts. I introduce an algorithm as a concept and I introduce the program that implements (satisfies) the algorithm, as an entity.

7.3 Analysis of the Definitions of a Function

Let's consider the following three definitions of a function:

Definition 1: A function from A to B is a rule that assigns, to each member of set A, exactly one member of set B.

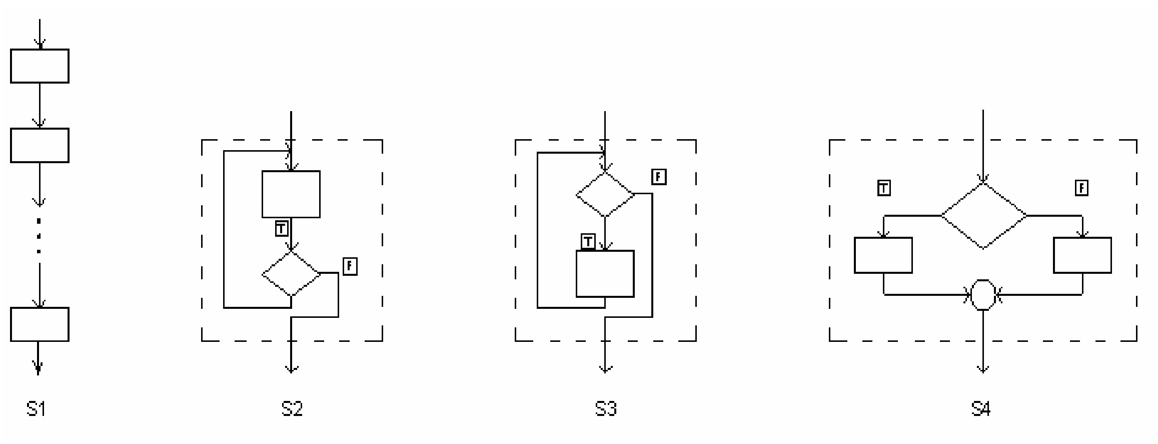
Definition 2: A function from A to B is a relation between A and B that pairs each member of A with exactly one member of B.

Definition 3: A function is any incomplete phrase with specified gaps such that when the gaps are filled with names of entities, the phrase becomes a name. (This is Frege's approach)

We will use Definition 1 for our algorithmic approach. This definition has a drawback – it is intuitive because the term "rule" is undefined. Of course, Mathematics is not an intuitive science. However, we always use algorithms which are finite and clearly defined.
In the sense of Frege's theory of names, we will define an assignment as the binding of names. In particular we will bind the name of a variable to the name of a value.
In programming languages we consider assignment to be the only atomic command.
So, the targets of assignments are locations and the sources of assignments are values.

7.4 Algorithms

My algorithmic concept consists of 4 algorithmic structures (see picture).



The fundamental structure is S1 – named sequence. We can also understand structures S2, S3, S4 as sequences, but we then treat them as a rectangle (with dashed lines in the picture).
Each block has exactly one input and one output; otherwise it is not a sequence. These structures are known as structures of the structural programming.

7.5 Construction of an Algorithm

We use the symbols: A = {→, ▭, ◇, ○ }. The symbol circle marks the spot where the two arrows meet.
(i) We have the set S = {S1, S2, S3, S4} .
(ii) We have one rule. In an algorithm we can place any structure from S instead of a rectangle
(iii) With finite applying of (i) and (ii) we construct sequences.

7.6 Why are sequences so important?

In my opinion, this is because of the following property, which I have not seen clearly presented.

Retaining inherent properties in programs that have changes.
What does this mean? It means that during changes in the logic of a program, we can do the following:

- (i) we can add a new logic to the program
- (ii) we can retain (inherit) parts of the old logic of this program – these can be parts that we determine.
- (iii) We can do (i) and (ii) in formal way presented in 7.5

7.7 Database Solution

In 7.2 a program is defined as an entity which satisfies a certain concept. We can now apply results from my papers [1] i [2].
Thus we can keep and memorize the states of programs in databases.
In [2] I showed how mapping is done between different database models using the identifier of state.

7.8 Conclusion

We may ask why we are doing all this, and where might this approach be applied? I believe this approach enables work with the states of a program in the testing phase and in broader sense, while working with parts of programs.
This approach can be useful for parallel and distributed programming, and in microprocessor and OS development.

Finally, we will define function in a new way: it is the history of states, determined beforehand by the function.

References

[1] Vladimir Odrljin
http://www.dbdesign10.com, created in
September of 2005

[2] Vladimir Odrljin
http://www.dbdesign11com,
Clarified version for a better understanding
of the paper from 21.August 2008.