DATABASE DESIGN AND DATA MODEL FOUNDED ON CONCEPT AND KNOWLEDGE CONSTRUCTS

ABSTRACT

In this paper a new data model is introduced. The data model is based on concept constructs. We present a novel approach to and a novel definition of a concept. Attention is paid to solutions of complex databases which can maintain states of entities or relationships, as well as changes made to these states. We consider that the existing theory does not have satisfactory solutions for these databases.
The approach in this paper is not a kind of formal axiomatic system; rather we define the necessary and important constructs in the process of database design.

1. INTRODUCTION

In brief, this data model is devoted to data-databases which can maintain changes of attributes and databases which can maintain knowledge related to their data. Temporal databases and databases that maintain history, belong to this group of databases. Databases, in which attention is placed on data in its broadest sense, also belong to this group of databases. An example of one such database is a database which can always determine which person or procedure created any piece of data in the database. Other examples are online applications such as online web applications where data should be available and public as soon as it is entered. We find that these databases are of a general character, i.e. they are not special cases.
Note that the special case of the databases mentioned above are databases that have Insert, Delete and Update operations, but in which changes are not maintained, i.e. they always have one state – the state after the last changes. Simple databases, which have no changes at all, also belong here. Because the existing database theory has tediously researched this group of databases, we have not devoted any attention to these simple cases.

2. PRELIMINARIES

We accept that the world is discrete, i.e. that it consists of things or individuals. These things or individuals are called entities [3]. This means that we use the term entity only for real world objects. During the process of database design we do one very important thing - we interpret (a part of) the real world. Because they are basic in interpreting the real world, we 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.
We also introduce concepts, which we use to model these real world objects and their relationships. Concepts are abstract objects constructed in our mind. Concepts are presented in detail in Section 4.
Extension of a Concept. We accept G. Frege's assumption that an object may be associated to every concept. This object is called the extension of the concept [4], [5].
Intrinsic Properties. We assume that the concepts of entities have only intrinsic properties. By intrinsic properties, we mean properties which an entity (or relationship) has itself; intrinsic properties are independent of other concepts. Extrinsic properties, on the other hand, are all other properties related to the concept of the entity.

3. PROPERTIES AND ATTRIBUTES

Data which is related to attributes and properties makes up the majority of data in every database. This data comprises almost all of a database's data. In Section 3 we present a novel approach to properties and attributes.

Limitation of Interpretation. Our assumption related to real world objects is that we can only recognize or match those objects for which we have perceptual, inferential or rational abilities. In the remaining text instead of perceptual, inferential, or rational abilities, we will use just the term "abilities".

3.1 Events

In this paper it is assumed that there are two kinds of events in the real world which are related to information about an entity:

(i) An event which causes new information about the entity.

(ii) An event which causes existing information to be invalid after this event. We will also say this event *closes* existing information about the entity.

3.2 Information about an Entity

When we speak of information about an entity in this paper, we mean information about the entity's attributes. In our terminology, a property is a concept and a property has multiple attributes.

Example 1:

The color of an entity is a property of that entity, while red, blue or green are attributes of the entity. Of course, we must have the ability to match the real world color red to the corresponding attribute red in our mind. The color of an entity is the concept of color in our mind – as we mentioned, a concept is an abstract object. Attributes in our mind correspond to an entity's real world attributes carried by information■

3.2.1 Matching of attributes.

The term *match* is used in the sense that information about an entity's attribute generates (by means of our abilities) a corresponding image of the attribute in our mind. This attribute's image matches the entity's attribute■

Definition. An m-attribute can be built up by the following two steps:

(i) The m-attribute is created by the match between an entity's attribute and the corresponding attribute in our mind.

(ii) The m-attribute is the interpreted and abstracted entity's attribute brought to us by information■

3.3 Definition of the Concept of an Entity's Property.

The concept of an entity's property is an abstract object which is generated by information about the entity's attributes and which is satisfied by its m-attributes and by nothing else■

Universal and Particular Attributes. We assume that attributes are universal, and not particular, i.e. that attributes do not depend on an entity■

Example 2: For two cars whose color is red we say that they have the same color – red. The principle of universality of an attribute enables the attribute red in our mind to match the color red in all cars which are red. The color red does not depend on a particular car■

Let S be the relation *satisfy* between a concept of a property and the corresponding m-attributes. If an m-attribute satisfies the corresponding concept, then it can be said that they are in a relation and we write that: S (the m-attribute, the concept of the property) = T.

By definition of relation S, only the members of the extension of this property satisfy the concept of the property. On the other hand, every m-attribute is created with the help of its corresponding entity's attribute. Considering 3.2.1, 4.2.1 and 4.2.2 the following holds true:

S (the m-attribute, the concept of the property) = T iff the m-attribute matches the entity's attribute. … (3.3.3)

Loosely speaking, (3.3.3) can be written as: The color of the entity car is red iff the information "the color of car is red" matches the m-attribute.
Relation S implies that there is subordination in which the satisfaction of the concept for the m-attribute comes first, while its extension comes second.

3.4 Definition of a Fact about an Entity
If an attribute of an entity brought by information is matched with its corresponding m-attribute and is memorized in our mind, then we call this entity's attribute a fact about the entity■
Note that an attribute which is a fact is always related to its entity (or relationship). Thus, "red" is a universal attribute, while "red car" refers to the entity car and can be a fact. A fact about an entity is determined by three things: an attribute, a property and an entity. So when we say "The car is red" then this fact is determined by the following: the entity car, attribute red and property color. The sentence "The car is red" expresses a fact about the entity. First, we memorize the fact in our mind, and then we can form a sentence that expresses the fact. Factual statements always express something about the world.
Facts about entities are atomic, they are not composed. We also say facts about entities are primitive. However, we can construct a compound structure from facts.
A fact retains the meaning of that which the information is about. We define facts as something of which we are aware, something connected to an entity and the entity's property.

3.5 Definition of an Entity's Attribute.
An entity's attribute is presented by the following:
(i) Information about the attribute of the entity;
(ii) One fact about the entity■
Example 3. This example is related to the work of French philosopher Maurice Merleau-Ponty [6]. When we look at a wheel from an angle, we see the wheel in the shape of an ellipse. However, we describe the wheel as being a circle.
This example is another confirmation that a concept is not determined only by a conjunction of properties.

3.6 Primitive Knowledge About an Entity
Those entity's attributes which are facts are stored in a database. They are primitive knowledge about an entity. A stored or memorized fact becomes permanent.
Definition. A set of all facts about one entity is primitive knowledge about the entity. We denote it as $KE = \{F_1, F_2,\ldots,F_k\}$ where F_i, are the facts defined in 3.4

3.7 Knowledge about an Attribute
Knowledge about an attribute is a set of facts which are related to one individual attribute. We will denote knowledge about an attribute in the following way:

$$Ka = \{Fa_1, Fa_2,\ldots,Fa_m\}$$

For instance, if John provided information about attribute An, then John is a fact related to attribute An. If attribute An was created in the real world on 12 December 2007, then this date is a fact about attribute An.

3.8 Knowledge about Data
When a fact about an entity is stored in a database, we name it data. Data is represented by a value from a domain. Facts about data from a database establish knowledge about this data. We will denote this knowledge as:

$Kd = \{Fd_1, Fd_{2,}\ldots,Fd_n\}$

For example, if Sam entered the data Dn into the database to represent the attribute An, then Sam is a fact related to Dn. If the data Dn was entered on 1/12/08, then this date is a fact about Dn.■
Knowledge defined in 3.6 – 3.8 is related to basic database elements and our intention is to define this basic knowledge. The facts from 3.7 and 3.8 are atomic, similar to facts about an entity.
Regarding the above defined three terms of knowledge, we can say that all data (i.e. the information which is stored in a database) represents facts or that we believe it represents facts.

3.9 Knowledge about an Entity.
Knowledge about an entity at some point in time is denoted by K and
$K = KE \cup KA \cup KD$; where KA is the knowledge about all the attributes of one entity and KD is knowledge about all the data about the entity. We conclude that:
(i) There are different kinds of facts; facts about entities, attributes or data
(ii) We use attributes together with associated knowledge related to them.
We define knowledge about a relationship similarly to defining knowledge about an entity.

4 THE CONCEPTUAL MODEL

4.1 Introduction to conceptual model
In Section 2 we introduced concepts as abstract objects which are abstracted by our rational activities. The process of data modeling is divided into two levels; a conceptual level, and a logical level or database level. There is a mapping between the conceptual schema and the database schema. We determine schema mapping as a mapping between a source schema and a target schema. The source schema is the schema of a concept. On the conceptual level we only consider schemas of concepts, schema mappings and extensions.

4.2 Concepts
We construct concepts so that they are satisfied by the intended things. We derive the schema of a concept and its semantic properties from a corresponding concept and we use this schema to express the concept in language. For the purpose of database theory and practical use, we will define the following types of concepts: the concept of an entity, the concept of an m-n relationship and the concept of a state of an entity (or relationship). The concept of an entity's property is defined in 3.3. Every schema of the above concepts denotes a key that uniquely identifies the members of the concept's extensions. The process of identifying plurality and individuals is defined in Section 5.
We do not specify a schema definition language. Instead, we define a simple and intuitive schema-level notation.

4.2.1 G. Frege's Assumption
In Section 2 we introduced the extension of a concept. Now we determine the extension of a concept in the way it was done by Gottlob Frege: The extensions of two concepts are identical objects iff the two concepts are coextensive■
According to G. Frege, two concepts are coextensive if every thing that satisfies either satisfies the other.
Although we use G. Frege's approach to extensions, we have a different approach to concepts. We define a concept in Section 5.3.

In the same way in which we defined m-attributes we define m-entities, m-relationships and m-states. These objects are interpretations and abstractions of their corresponding real world objects, as is determined in Section 2. M-entities satisfy the concept of an entity, m-relationships satisfy the concept of an m-n relationship, and m-states satisfy the concept of a state of an entity (relationship). We also consider the process of matching real world objects with their interpretations and abstractions. During the process of interpreting a real world object, there is another process which is involved in the construction of the interpreted object. We call this process matching. The following two things are important in the matching:
(i) The interpreted object matches the real world object in compliance with its concept;
(ii) The interpreted object can be used to identify the corresponding real world object.

4.2.2 Concept of an Entity

We determine the concept of an entity by its properties. We assume that these properties are intrinsic. This implies that the entity's properties are related only to the concept of the entity and to nothing else. Therefore, we use an identifier of an entity as a key in this paper. The schema of the concept of an entity is defined by the following two forms:
(i) By 3.3, properties are concepts. Therefore, the schema of the concept of an entity can be presented as follows: $Schema_1$ (IdOfEntity, $Property_1$),…, $Schema_n$ (IdOfEntity, $Property_n$), where IdOfEntity is the identifier of the entity which denotes that all the properties belong to one entity.
(ii) The schema of the concept of the entity can be also determined as: $Schema_e$ (IdOfEntity, $Property_1$,…, $Property_n$)■
These two schemas show that one entity can be represented either by a concept or by a set of concepts. In Example 6 in 4.2.3 the way in which an entity represented by binary concepts can be represented by one concept will be shown.
M-entity. The m-entity consists of m-attributes which correspond to the real world entity's attributes. We assume that the m-entity matches an entity if all it's the m-attributes match the corresponding entity's attributes carried by information. We say that the m-entity satisfies the concept of an entity if all of its m-attributes satisfy the concepts of the corresponding properties. We define the relation *satisfy* as a relation between the concept and the m-entities which satisfy the concept. Similarly to (3.3.3), we can say that the following holds true:
S (the m-entity, the concept of the entity) = T iff the m-entity matches the entity.

4.2.3 Concept of an m-n Relationship

This is a concept which models relationships between two entities. We construct this concept so that it determines an m-n relationship and the properties which are related to it, $m,n \geq 1$. The schema of a concept of an m-n relationship between two entities includes the following three components:
(i) $Schema_r$ (KeyOf $Entity_1$, $KeyOfEntity_2$, $Property_1$,…,$Property_m$)
(ii) The key of a relationship is represented by keys of the involved entities. In the schema: Key = ($KeyOfEntity_1$, $KeyOfEntity_2$).
(iii) The involved entities are represented by the schemas of their concepts. So the concepts of these entities are involved in the concept of the m-n relationship■
The concept of an m-n relationship is satisfied by the corresponding m-relationships.

The components of the m-relationship satisfy 4.2.3.(i), 4.2.3.(ii) or 4.2.3.(iii).
In a similar way to (3.3.3) we define that: S (the m-relationship, the concept of the m-n relationship) = T iff the m-relationship matches the m-n relationship.
Example 4. The schema: TeacherBookCourse (Teacher, Book, Course) is not a schema for a concept of an m-n relationship, because the schemas for corresponding entities are not given. In other words, it is not appropriate that the construction of a concept be based on unknown concepts.
Example 5. Owner (PersonId, CarId), Key = (PersonId, CarId);
Person (PersonId, PersonName), Key = (PersonId);
Car (CarId, Maker, Color), Key = (CarId).
Here we have a schema of the concept of the m-n relationship Owner between the entities Person and Car. Now for example, we can map the conceptual schema to a file schema. In this case we will construct the following files: FileOwner, FilePerson and FileCar.
(i) The FileOwner file has the schema: (PersonId, CarId), and the following two keys:
(ii) K= (PersonId, CarId) and –K = (CarId, PersonId).
We named the keys "K" and "–K" to emphasize that they are inverses of each other.
(iii) File Person has the schema: Person (PersonId, PersonName), Key = (PersonId)
File Car has the schema: Car (CarId, Maker, Color), Key = (CarId)■
Example 6. We can apply 1-1 relationships to the extensions and entity which correspond to the concepts in 4.2.2(i) and then we will get the schemas in 4.2.2.(ii). Thus, if we apply 1-1 relationship to the extensions and entity which have the following schemas: $Schema_1$ (IdOfEntity, $Property_1$), $Schema_2$ (IdOfEntity, $Property_2$) then we have $Schema_{e2}$ (IdOfEntity, IdOfEntity, $Property_1$, $Property_2$) = $Schema_{e2}$ (IdOfEntity, $Property_1$, $Property_2$) and this is by definition the schema of the entity. Similarly, we can get $Schema_{e3}$, or $Schema_{en}$, i.e. schema of an entity with n properties.

4.2.3.1 Complex Concepts

The above examples show us how we can build complex concepts from basic concepts. We construct complex concepts by applying the concept of m-n relationships among entities, where m, n $\geq$ 1.

4.2.4 Concept of a State of an Entity or Relationship

In Section 1 we determined which databases can be effectively solved using the results from this paper. Here in 4.2.4 we present our solutions for these databases. Later in examples 7, 8 and 9 we show a small fraction of practical and technical possibilities based on our solutions. By using the concept of a state of an entity or relationship we construct solutions for the databases mentioned in section 1 as well as many other complex databases. The concept of a state of an entity has the following components: properties, knowledge about attributes and knowledge about the data. The concept of a state of an entity also contains an identifier of the entity and an identifier of the state of the entity. One entity can have many states. A concept of a state of an entity has the following form:

identifier of the state of an entity	identifier of the entity	attributes	knowledge about attributes	knowledge about data

In the states of an entity, the identifier of the entity stays unchanged through all the states of the entity, because the states are from one entity (however if we want, then we can decide and determine which of the states belong to one entity). The identifier of the entity determines which states belong to the entity. If an entity only has one state, then the identifier of the entity is semantically equal to the identifier of the state.
Knowledge about the entity's attributes and knowledge about data are defined in 3.7 and 3.8. Knowledge about an entity (or relationship) is defined in 3.9

4.2.4.1 Definition of a State of an Entity or Relationship.
A state of an entity (or relationship) is knowledge about the entity (relationship) ■ The identifier of a state of an entity is the key of the state of the entity.

4.2.4.2 Definition of a Change of a State
A change of a state of an entity is any change of knowledge about the entity■
A change of a state of an entity (or relationship) is always caused by an event from the real world. So in the case that a real world event causes a change of the state of an entity, we will create a new identifier of the new state of the entity.
When we work with the concept of a state of an entity, this means that we keep all the data in a database. It also means that no updating or deleting of data from the database occurs (i.e. the database is always expanding), because we define only two operations on the data entry level; adding new data and *closing* existing data.

A concept of a state of a relationship has the following form:

identifier of the state of a relationship	key of a relationship	attributes	knowledge about attributes or the key	knowledge about data

Here the "key of a relationship" is a set of identifiers of states of the involved entities.
Now we will analyze in more detail the concept of a state of an entity. The concept of a state of an entity is a definite departure from the idea that a concept is determined only by a conjunction of properties. This concept has an associated structure which generates the meaning of the m-entity (or m-relationship) as the totality of the entity's (or relationship's) states and the corresponding knowledge. We assume the following principles regarding concepts and especially the concept of a state of an entity to be true:
(i) When we are constructing a concept, then we know in advance how that concept has to look, i.e. which kinds of the entities satisfy the concept. As concepts are abstract objects i.e. concepts are on the thought level, we actually know what kind of concept we want even before we have formed a sentence which expresses this concept.
(ii) The concept of a state of an entity is based on the definition of state in 4.2.4.1. This concept determines a new state every time a change of the state of the entity (or relationship) happens in the real world.
(iii) The concept of a state of an entity enables us to identify the plurality of states, and also enables us to identify individual states. This concept determines the construction of individuals as well as the identification of the constructor which constructed these individuals.

4.2.5 Schema of the Concept of a State.
Now we will present the schema of the concept of a state of an entity in detail. The schema of the concept of a state of an entity takes the form:
ConceptStateName (P, E, A_1… A_n, K_{p1}… K_{nr}, D_{p1},…,D_{ns})
where P is the concept of the identifier of a state of the entity (or relationship);
E is the concept of the identifier of the entity;
A_1,…,A_n are concepts of the properties of an entity (or relationship);
Each property, including E and P, can have different sets of knowledge K associated to them and defined in 3.6 – 3.9. Thus:
P has knowledge K_{p1}, K_{p2}… K_{pi};
E has knowledge K_{e1}, K_{e2}… K_{ej};
A1 has knowledge K_{11}, K_{12}…K_{1k};
….
An has knowledge K_{n1}, K_{n2}… K_{nr}.
Knowledge D_{p1},…,D_{ns} is defined in 3.8.

4.2.6 Binary Concepts.
The schema of a concept of a state can be represented by schemas of the following binary concepts: (P, E), (P, A1)… (P, An) to which we associate the corresponding knowledge and get the following concepts:
(i) Schemas of K-concepts;
C_{k1} (P,A_1, K_{11},…,K_{1k},D_1,…,D_{1m});
…
C_{kn} (P, A_n, K_{n1},…,K_{nr}, Dn1,…,D_{nq});
(ii) Schema of the E-concept
C_e (P, E, K_{p1},…,K_{pi}, D_{p1},…,D_{ps});
(iii) Schema of the E-concept for the state of an m-n relationship between two entities will be as follows: C_{emn} (P, P_{e1}, $P_{e2,}$ K_{k1},…,K_{kt}, D_{k1},…,D_{ku}) where P_{e1}, P_{e2} are identifiers of states of entities e1, e2.
The state of any entity can be represented in two ways: the one described in 4.2.5 and by the one described in 4.2.6. The schemas in sections 4.2.5 and 4.2.6 represent the same entity. The constructs applied in these schemas enable the direct construction of schemas in 4.2.6 from the schemas in 4.2.5, and vice versa. These constructs enable every change of state to be recorded. We prefer this recording to be done by constructors and the advantages of this are explained in Section 6. E-concepts determine the relationships between one entity and all of its states.

4.2.7 States in Relational Model
In the relational model we represent knowledge by columns. We represent a state of an entity in the relational model as the following relation schema: Rstate (P, E, A_1… A_n, K_{p1}… K_{nr}, D_{p1},…,D_{nq}). Here relation schema Rstate is a target schema and the corresponding source schema is in the form of ConceptStateName from 4.2.5. We accept that a relation has, aside from properties columns, those columns which represent knowledge and identifiers.

4.2.8 Binary Relations
The schema Rstate can be represented by schemas of the following binary relation schemas: (P, E), (P, A1)… (P, An) to which we associate the corresponding knowledge and we get the following relation schemas:
(i) Schemas of K-relations;
R_{k1} (P,A_1, K_{11},…,K_{1k},D_1,…,D_{1m});
…
R_{kn} (P, A_n, K_{n1},…,K_{nr}, Dn1,…,D_{nq});
(ii) Schema of the E-relation
R_e (P, E, K_{p1},…,K_{pi}, D_{p1},…,D_{pq});
If we have a schema of the state of an m-n relationship between two entities, then instead of E we will put P_{e1},P_{e2}, in R_e , where P_{e1}, P_{e2} are identifiers of states of entities e1, e2■
We call these schemas corresponding binary schemas because each of them has one attribute and the simple key. E-relation and K-relations are types of binary relations because if we omit the columns

of knowledge, then the relations become binary relations.

4.2.9 An Effective Solution Which Decomposes Any Relation of State to Binary Relations

Let Rstate and $R_{k1},\ldots,R_{kn}$, R_e are the relation schemas defined in 4.2.7 and 4.2.8 respectively. We will say that relational schema Rstate is equal to join of its corresponding binary schemas and denote it as Rstate $(P, E, A_1 \ldots A_n, K_{p1} \ldots K_{nr}, D_{p1},\ldots,D_{nq}) =$
$R_e (P, E, K_{p1},\ldots,K_{pi},D_{p1},\ldots,D_{pq})$ join $R_{k1} (P,A_1, K_{11},\ldots,K_{1k}, D_1,\ldots,D_{1m})$ join,…, join $R_{kn} (P, A_n, K_{n1},\ldots,K_{nr}, D_{n1},\ldots,D_{nq})$ if and only if every relation that is a legal value for Rstate is equal to the join of its corresponding binary relations■

This equation holds always because of the construction of the simple key, states and E-relation. One identifier of a state determines all the components of the state. One identifier of an entity determines all the states of the entity. The relations are joined using common column P. The equation holds true for both entities and relationships.

Example 7

This example shows how certain complex databases, including "temporal databases" and "databases which maintain history", should be solved. The solution is related to two entities and one relationship, but each of these three data structures changes its state. We begin with the fact that the concept of a state of the entity Car is given by the schema: Car (CarKey, CarId, Maker, Type, Color, DateFrom, DateTo). Using the mapping from the schema of the concept to the schema of the relation we have the following relation schema: Car (CarKey CarId, Maker Type, Color DateFrom, DateTo). We will use the schema to form the following table Car:

Table Car

CarKey	CarId	Maker	Type	Color	DateFrom	DateTo
23	vin1	Buick	sedan	silver	1.1.2000.	12.20.2000
24	vin1	Buick	sedan	blue	12.21.2000	8.1.2001
25	vin1	Buick	sedan	red	8.2. 2001	1.1.2005
26	vin1	Buick	sedan	silver	1.2. 2005	999999
27	vin2	Honda	sedan	silver	3.15.2006	999999
28	vin3	Ford	sedan	black	3.15.2006	999999

…

CarKey is the identifier of the state of the entity Car, this is the only property of Car that has unique values. CarId is an identifier of the entity Car. VIN (vehicle identification number) values are used for this property. In this example CarKey's values 23, 24, 25, and 26 denote four states of the one car identified with CarId = vin1. Date "999999" represents the maximum date in the used software and means that the corresponding data is current. In this table, the columns DateFrom and DateTo are strictly related to one attribute from the column Color. DateFrom and DateTo are not properties of the entity Car. Instead, they are a part of our actual knowledge about one particular attribute from the column Color. The entity Car also represents knowledge about a particular attribute from the column Color. Therefore, besides columns which represent properties, the table Car also has

columns which represent knowledge about attributes.

Table Person

PersonKey	PersonId	Name
208	ssn1	Mary Jones
209	ssn1	Mary Adams
210	ssn2	John Stewart

Table Owner

OwnerKey	PersonKey	CarKey	DateFrom	DateTo
54	210	26	1.2.2005	3.15.2006
55	210	27	3.16.2006	10.9.2006
56	210	26	10.10.2006	999999

In the table Person, PersonKey is an Identifier of the state of the entity Person, PersonId is the Identifier of the entity Person, and Name is the name of the person. Here Mrs. Mary Jones changed her last name because she had gotten married to Mr. Adams. The table Owner represents the relationship between the entities Person and Car where OwnerKey is the Identifier of a state of the relationship Owner, PersonKey is the Identifier of a state of the entity Person, CarKey is the Identifier of the state of the entity Car, and DateFrom, DateTo determine the period of ownership. Here, Mr. John Stewart bought a Buick in 2005 and then sold it to his friend. He bought a Honda in 2006. In 2006 he bought his old Buick back from his friend. An identifier of a state of an entity is always initiated by a real world event. Formally it can be said that the identifier of an entity determines one set of its identifiers of state. For example in table Car, vin1 determines the following set: $A = \{x \mid \tau E(x, vin1) = T\}$ where E(x, vin1) is the sentence "x is in the E-relation with vin1". The E-relation is defined in 4.2.6.(ii). The above mentioned E-relation does not have columns Kij and Dkl.

---

5 DETERMINING PLURALITY - IDENTIFYING AND DISTINGUISHING ENTITIES.

The process of identifying goes from a subject to the real world and this implies that the subject has some knowledge about the entity which it tries to identify. In the process of identifying there are two constructions.

5.1 Construction of a Unique Concept of an Entity and of Unique Members of the Extension of the entity's Concept - Distinguishing Entities

To construct unique concepts of entities we will use corresponding properties. This construction satisfies the definition of a concept of an entity from 4.2.2 and Frege's assumption. To construct unique members of the extension of the concept, we must consider the following two cases:

(i) We can construct a unique concept using properties which are generated by entities whose concept we want to construct. If the properties in the concept construction enable the extension of the concept to have unique members, then we have a construction which satisfies the conditions in 5.1, i.e. we have the construction we want.

(ii) If the concept's properties can not establish a uniqueness of the extension's members, then we will add a new property to the concept of the entity, called the identifier of the entity. The new property will be used for the construction of the unique entities' identifiers. So the entity's identifiers by their construction will allow the members of the extension of the entity's concept to be unique. On the other

hand, the entity's identifiers form unique entities in the real world. Therefore, we use the same identifiers for both the formation of unique members of the corresponding concept's extension and for the formation of unique entities■ So we use the construction 5.1 to construct:

a) A concept that is different from any other concept.
b) Members of the concept's extension which are mutually distinguishable■

We will call the construction described in 5.1 "distinguishing of entities". In conclusion, we can say that we use the properties of the entity or the additional identifier of the entity to form distinct entities.

5.2 Identification of Entity

The following constructions enable the identification of entities whose concept constructions were described in 5.1.(i) and 5.1.(ii) respectively.

(i) To identify an entity which has the concept construction described in 5.1.(i) we use a construction based on a minimal set of attributes by which we can identify the corresponding entity.

(ii) To identify an entity which has the concept construction described in 5.1.(ii) we use identifiers of the entities whose concept is constructed in 5.1(ii)■

We will call the construction described in 5.2 "identifying of entity".

5.3 Definition of Concept

A concept is a construct which determines one or both of the following:

(i) A plurality of things in which all the things satisfy the concept;

(ii) A particular thing from the plurality determined by (i)■

In order to identify an entity we use the following procedures:

Procedure1: Identifying the plurality.

Procedure2: Identifying individuals.

Procedure2 is not effective without Procedure1.

6. CONSTRUCTIONS OF DATA THAT REPRESENTS SINGLE OBJECTS OR INDIVIDUALS

In this section we will generally consider the construction of data that represents individuals; this construction will be shown in detail in Example 9. The construction of the data described in this section is intended for databases which use concepts of state, i.e. databases in which all the data is saved. By an individual, we usually mean an attribute which is represented by data. More generally, individuals are not sets. On the other hand, a set is a plurality regarded as a single object. We consider the entry of data which represents individuals or single objects a separate unit in database design. Therefore we have developed effective solutions which enable the representation of data by applying Binary Concepts and Binary Relations. Similarly we can construct binary files for a file schema. Though there have been researchers who have expressed the desire to represent data by means of binary relations, they have not yet shown how this should be done.

6.1 Derived data

Derived data is data which is obtained from the existing data in the database. For example, this is data which we can get from a report, display, view, or query, as well as data which we can get by applying operations to existing data in the database. Relational Algebra, for example, uses a collection of operations to relations.

6.2. New Data

Data which is entered into a database is new data. This data cannot be derived from existing data in the database. Often, it is of interest to us how this type of data is

constructed. Mainly, the new data represents individuals. We might, for instance, be interested in knowing how the new data was entered into the database and who entered it (who is responsible for this data). We can also be interested in the constructions of various procedures which carry out this entry of new data.

6.3 Constructors

To construct this new data, we can use the following two constructors – the Constructor and the ClosingConstructor. We create the new data using the Constructor, while we *close* the data with the ClosingConstructor. These two constructors in some way correspond to the Constructor and Destructor from OOP. The difference is that ClosingConstructor does not delete or destroy or change data; it just says that the data is not valid from some point in time. The second difference is that these two constructors are initiated by real world events. By 3.1 we consider only two kinds of events, the first one causes the creation of new data in the database and the second causes the *closing* of current data. The third difference is that, using Constructor and ClosingConstructor, we create keys and knowledge in the database. The use of the constructors is one of the possible solutions. However constructors can construct complex structures.

6.4 Necessary conditions for binary representation

In 4.2.2.(i) we show that the concept of an entity can be presented as a set of schemas of binary concepts, i.e. as concepts that have one property and one identifier of the entity to which this property belongs. We will now consider the conditions necessary for binary representation. These conditions are as follows:

(i) The entity's properties should be intrinsic.

(ii) The key is an identifier of the entity. Thus, the key uniquely determines a member of the extension and at the same time identifies the entity, as it is defined in Section 5.

In 4.2.9 we construct solutions for complex databases. Now we will consider the relational model and the construction of binary relations which represent an entity, but without entity's states (This is for simpler databases). We can define a schema mapping where the source schema is a set of symbols for the schemas of binary concepts and where the target schema is the corresponding set of binary relation symbols. We can also define another 1-1 mapping, which is from the members of an extension of a binary concept to the tuples of the corresponding binary relation. These two mappings determine a starting schema for binary relations of an entity.

We can apply another approach to binary relations. We can construct a relation which is based on 4.2.2.(ii). This relation represents an entity, which has an identifier and intrinsic properties. If we translate these two conditions to relational terminology, then we have a relation with a simple key and mutually independent attributes. Obviously this relation is in BCNF. Formally we can say: If an entity satisfies the following conditions:

(i) The entity has an identifier;

(ii) All the other properties of the entity are intrinsic, then the relation that represents this entity is in BCNF■

6.5 Definition of Simple Form

Let R (K, A_1,…,A_n) be a relation schema, where

(a) key K is simple

(b) A_1, A_2,…, A_n are the attributes which are mutually independent

(c) R_1 (K, A_1), R_2 (K, A_2), …, R_n (K, A_n) are the corresponding binary schemas.

We will say that relational schema

$R (K, A_1, A_2,\ldots,A_n)$ is equal to join of its corresponding binary schemas and denote it as $R (K, A_1, A_2, \ldots,A_n) = R_1 (K, A_1)$ join $R_2 (K, A_2)$, join … join $R_n (K, A_n)$ iff every relation that is a legal value for $R (K, A_1, A_2,\ldots,A_n)$ is equal to the join of its corresponding binary relations■

Definition

Relational schema $R (K, A_1, A_2,\ldots,A_n)$ which represents an entity is in Simple Form if R satisfies the following:

$R (K, A_1, A_2,\ldots,A_n) = R_1 (K, A_1)$ join $R_2 (K, A_2)$ join…join $R_n (K, A_n)$ iff

(i) Key K is simple

(ii) $A_1, A_2,\ldots,A_n$ are mutually independent attributes■

$R_1 (K, A_1), R_2 (K, A_2) ,\ldots, R_n (K, A_n)$ are the corresponding binary schemas. In a similar way we can define Simple Form for m-n relationships.

Simple Form has the following advantages over existing relational theory:

(i) We have the conditions which a relation must satisfy in order to be in BCNF;

(ii) We do not need to put a relation into 2NF and 3NF to get it into BCNF.

(iii) The binary schemas can be immediately constructed■

Example 8 Now from the table Car in Example 7 we will construct the following four tables:

Table1

CarKey	CarId
23	vin1
24	vin1
25	vin1
26	vin1
27	vin2
28	vin3

Table2

CarKey	Maker
23	Buick
24	Buick
25	Buick
26	Buick
27	Honda
28	Ford

Table3

CarKey	Type
23	sedan
24	sedan
25	sedan
26	sedan
27	sedan
28	sedan

Table4

CarKey	Color	DateFrom	DateTo
23	silver	1.1.2000	12.20.2000
24	blue	12.21.2001	8.1.2001
25	red	8.2.2001	1.1.2005
26	silver	1.2.2005	999999
27	silver	3.15.2006	999999
28	black	3.15.2006	999999

Here in Example 8 we have constructed four “attribute-based” or binary relations from the relation represented by the table Car in Example 7. The first three tables each have two columns, one of which is for attributes and the other for key. However, Table4 in addition to these two columns has knowledge about the property Color which is represented by two columns (Datefrom and Dateto). One can add some other “knowledge-columns” related to Color. Now in Example 8 we have the relation Car from Example 7 represented in Simple Form■

The identifier of the state of an entity or relationship is not created arbitrarily. It is always initiated by a real world event, as is defined in 4.2.4.2. This connection to a

real world event gives companies great possibilities in creating their own technology. For instance, in Example 9 a company can establish additional paper documentation for any painting of a car with a customer signed agreement and many other options – all of which are associated to the identifier of the state of the entity Car. The identifier of the state of an entity or relationship always goes with the identifier of the entity or relationship. In the above example the identifier 26 is associated with VIN1, so it is not arbitrary at all. Thus the identifier of a state is always related to the real world and usually is associated to documentation.

6.6 m-states

A concept of an entity's state has the following main components: attributes, *knowledge,* and identifiers. We assume that the m-state matches an entity's state if all its components match the entity's state components. The matching of the attributes is already defined. *Knowledge* is defined in 3.7 and 3.8. We match this *knowledge* to a real world entity's state. At last, we match the identifiers of the concept of a state of an entity to the identified entity and the entity's relationships to its states. These processes are explained earlier in the text and examples.

Here we have the relation *satisfy* between the m-states and the corresponding concept of the state of an entity. We say that an m-state satisfies a concept of a state if all the components of the m-state satisfy their corresponding concepts i.e. if every component satisfies its corresponding binary concept defined in 4.2.6. The meaning of an m-entity (or m-relationship) is determined by the corresponding E-relation and K-relations. We say also that the meaning of the m-entity (m-relationship) is determined as the totality of the entity's (relationship's) states and the corresponding knowledge. As mentioned earlier we use constructors when we work with the concepts of states. Both the relation *satisfy* and the process of matching for m-state are defined by their components. In a similar way to (3.3.3) we can define a relationship between the relation *satisfy* and the process of matching for m-states:

S (the concept of an m-state of the entity, the m-state) = T iff the state of the entity matches the m-state.

Example 9. In the following example we will consider more than two "knowledge columns" related to the property Color from Example 8. Actually, the "knowledge columns" are related to the construction of data which represents individuals that fall under the concept Color. Here we modified Table4 from Example 8 and added the six "knowledge columns" related to the property Color: Datefrom1, Dateto1, Datefrom2, Dateto2, Operator1, and Operator2. So, for instance, Table4 can have the following data:

Table 4

CarKey	Color	Datefrom1	Dateto1	Operator1	Datefrom2	Dateto2	Operator2
23	silver	1.1.2000	999999	John	1.2.2000	999999	Mike
24	silver	1.1.2000	12.20.2000	Paul	1.2.2000	12.26.2001	Bill
25	blue	12.21.2001	999999	Paul	12.26.2001	999999	Bill

The first three new columns form a logical whole (unit) and are related to an event in the real world regarding Color. The second three new columns are also a logical whole but they are related to a corresponding event in the database. The first three new columns contain information which John or Paul who work in the garage write down in the form of paper documentation. Mike and Bill enter all the data into a computer. The rows containing CarKey=23 and 25 are created by the Constructor while the row containing CarKey = 24 is created by the ClosingConstructor. Note that we can record the user password and date from the system. Therefore, the constructors can get this data from the system and store it in the database even without the person performing the data-entry knowing this. Thus, we have a solution which can, in a formal way, recognize who created the data and how it was created, for all its data. The goal is for all the data to be saved so that the data that is already entered cannot be changed or destroyed, even if somebody wants it. For example, the data can be used in a court procedure as facts. Of course, there are other practical solutions, but we want to show with this small example that there are many possibilities of solutions using binary relations.

References

[1] J. Barwise, J. Etchemendy, Model-theoretic Semantics, in Posner, Michael (Editor), Foundations of Cognitive Science, MIT Press, (1989), pp. 207-243.

[2] A. Church, The need for Abstract Entities, American Academy of Arts and Sciences Proceedings 80,(1951), pp. 100—113.

[3] P.P. Chen, The Entity-Relationship Model: Toward a Unified View of Data, ACM Trans. on Database Systems, Vol.1, No.1, (March 1976), pp. 9-36.

[4] G. Frege, “Über Sinn und Bedeutung.” Zeitschrift für Philosophie und philosophische Kritik 100, 1892, pp. 25-50.

[5] G. Frege, Grundgesetze der Arithmetik, begriffsschriftlich abgeleitet. 2 vols. Jena, Pohle, (1893/1903).

[6] Merleau-Ponty Maurice, Phenomenology of Perception, Publish by Routledge, 1995

[7] E. Margolis, S. Laurence, Concepts Core readings, The MIT Press, (1999).

[8] A. Tarski, The Semantic Conception of Truth and the Foundations of Semantics’, Philosophy and Phenomenological Research, 4, (1994), pp. 341-76.