DATABASE DESIGN AND DATA MODEL FOUNDED ON CONCEPT CONSTRUCTS AND KNOWLEDGE CONSTRUCTS

1. Introduction

In brief, this data model is intended for databases that can maintain changes and their history, and to databases that can maintain knowledge related to their data. We begin database design at the conceptual level. The schema mapping from the conceptual model to a logical model is realized using a new approach: binary concepts.

The data model consists of abstract objects, which are based on our interpretations and abstractions of the real world. We introduce m-attributes, m-entities, m-relationships and m-states, as well as the construction and identification of these abstract objects. (see for example 3.3.3) Knowledge about an entity is introduced as a set of facts. In this paper however, facts and factual sentences are treated as two very different things. The data model introduces two new kinds of knowledge: knowledge about attributes and knowledge about data. This enables work with states and with changes of states. Therefore, the solution is completely new and effective for General databases.

We divide databases into two kinds: Simple databases and General databases. This paper is about General databases. Solutions and constructions for Simple and General databases are different. Understanding the different nature of these two kinds of databases enables us to introduce solutions for General databases, such as temporal, historical, online and data oriented databases. In contrast to Simple databases, General databases maintain changes. Defining states, changes of states, history of changes and their maintenance enables solutions for General databases. Section 2 is about abstract objects, the interpretation of the world and intrinsic attributes. Here we define concepts. Section 3 describes a new semantics related to attributes and properties. The data model and solutions are introduced as event based. This section describes concepts of properties. Section 4 describes concepts of entities, relationships and states. The construction of binary concepts, binary relations and binary files is defined here. The semantics of changes is also defined. In Section 5 we present two processes: distinguishing and identifying. Section 6 is devoted to data and its construction in General databases. This Section describes concepts of states. Using simple examples we here present how to determine who or which procedure constructed any data. The simple form is also defined here.

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]. 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 interpretations of 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 name them abstract objects.

We introduce concepts, which we use to model these real world objects. Concepts are abstract objects constructed in our mind.

Definition of a 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.
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]■
Our Entity-Relationship model for General databases consists of four concepts: the concept of an entity's property, the concept of an entity, the concept of an m-n relationship and the concept of a state of an entity or relationship.
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.■

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

3.2 Information about an Entity

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

Example1: 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 to us by information■
Universal and Particular Attributes. We introduce universal attributes as constructs of our abilities, which enable us to identify attributes of entities or relationships.
These attributes of entity or relationship we named the particular attributes■

Example2: 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. However, this car does have the particular attribute: red color■

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 of an m-attribute. 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■
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. (T stands for true)
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, 3.3, 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 to us 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 (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 color of the car is red" then this fact is determined by the following: the entity car, the attribute red and the property color. The sentence "The color of the car is red" expresses a fact about the entity. First, we memorize the fact in our mind, then we can form a sentence that expresses the fact. Factual statements always express something about the world. A fact is the possessor of truth and a fact 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 , therefore a fact is the smallest complete semantic unit related to an entity or relationship.

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.

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■

Example3: 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 = $\{F_1,\ldots,F_k\}$, 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 A_n, then John is a fact related to attribute An. If attribute A_n was created in the real world on 12/12/2007, then this date is a fact about attribute A_n.

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 create knowledge about this data. We can denote this knowledge as: $K_d = \{Fd_1, Fd_2,\ldots,Fd_n\}$. For example, if Sam entered the data D_n into the database to represent the attribute An, then Sam is a fact related to Dn. If the data D_n was entered on 1/12/08, then this date is a fact about D_n■
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 three above defined 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 total knowledge about all the attributes of one entity and where KD is total 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.
In the same way, we can define knowledge about a relationship.

4 The conceptual model

4.1 Introduction to the 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 this schema of a concept and its semantic properties from the corresponding concept and we use the 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 concepts denotes a key that uniquely identifies the members of the concept's extension. The process of identifying plurality and individuals is defined in Sections 2 and 5. We do not specify a schema definition language. Instead, we define a simple and intuitive schema 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 everything that satisfies either concept satisfies the other one also■
Although we use G. Frege's approach to extensions, we have a different approach to concepts.
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 was 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 or 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 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. 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 this one entity.

(ii) The schema of the concept of the entity can be also defined as:
$Schema_e$ (IdOfEntity, $Property_1$,…, $Property_n$)■
These two definitions show that one entity can be represented either by a concept or by a set of concepts. Example6 in 4.2.3 shows the way in which an entity represented by a set of binary concepts can also be represented by one corresponding concept.
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 its 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 kinds of 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 this schema:
Key = ($KeyOfEntity_1$, $KeyOfEntity_2$).
(iii) The involved entities are represented by the schemas of their concepts■

We define relationships among k entities in the same way. The key of this relationship is represented by keys of the involved entities: Key = ($KeyOfEntity_1$, …,$KeyOfEntity_n$). We assume that $Property_1$,…,$Property_m$ are intrinsic. This implies that the relationship's properties are related only to the concept of the relationship and to nothing else■
Regarding relationships, our database design is based on binary relationships. We use binary relationships to represents ternary or higher degree relationships. In case when business rules define the usage of only ternary or higher degree relationships then obviously there can be no decomposition of higher degree relationships to binary ones. 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 4.2.2 we define that:
S (the m-relationship, the concept of the m-n relationship) = T iff the m-relationship matches the m-n relationship.

Example4: 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.

Example5: 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 (PersonId, CarId), has 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)■
Example5 illustrates that binary files should be constructed using two fields: simple key and attribute. M-N relationships in the file model need to have a key constructed in this manner.

Example6: We can apply 1-1 relationships to extensions and entities 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 a 1-1 relationship to an extension and entity which have the following schemas:
(i) $Schema_1$ (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. This relationship is between the same entities. Therefore the attributes of this entity are attributes of the relationship. 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 between entities, where m, n ≥ 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	properties	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 all these states are from the same 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 were 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 (or 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 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. Similarly, this holds true for relationships. When we work with the concept of a state of an entity or relationship, 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	properties	knowledge about attributes	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 or relationship. The concept of a state of an entity or relationship 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 regarding the concept of a state of an entity or relationship 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 entities satisfy the concept. We hold that concepts are abstract objects i.e. concepts are on the thought level, and thereofre 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 determines a new state every time a change of the state of the entity 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.
(i), (ii), (iii) also apply to relationships.

4.2.5 Schema of the Concept of a State.
(i) 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:
StateEntity (P, E, $A_1 \ldots A_n$, $K_{p1} \ldots K_{nr}$, $D_{p1},\ldots,D_{ns}$) 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; $A_1,\ldots,A_n$ 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 $K_{p1}, K_{p2}\ldots K_{pi}$;
E has knowledge $K_{e1}, K_{e2}\ldots K_{ej}$;
A1 has knowledge $K_{11}, K_{12}\ldots K_{1k}$;
…
An has knowledge $K_{n1}, K_{n2}\ldots K_{nr}$.
Knowledge $D_{p1},\ldots,D_{ns}$ is defined in 3.8■
(ii) The schema of a concept of a state of an m-n relationship between states of two entities is as follows:
StateRelationship (P, P_{e1}, P_{e2}, $A_1\ldots A_m$, $K_{p1}\ldots K_{mr}$, $D_{p1},\ldots,D_{ms}$), where P is an identifier of the state of the relationship and P_{e1}, P_{e2} are identifiers of states of entities e1, e2. The time intervals of states of a relationship and states of its involved entities must be the same■

In General databases, relationships are always between states. … (4.2.5.1)

4.2.6 Binary Concepts
(i) The schema of a concept of a state of an entity can be represented by schemas of the following binary concepts: (P, E), $(P, A_1)\ldots (P, A_n)$ to which we associate the corresponding knowledge and get the following concepts:
(4.2.6.1) Schemas of K-concepts;
C_{k1} (P,A_1, $K_{11},\ldots,K_{1k}$,$D_{11},\ldots,D_{1m}$);
…
C_{kn} (P, A_n, $K_{n1},\ldots,K_{nr}$, $D_{n1},\ldots,D_{ns}$);
(b) Schema of the E-concept
C_e (P, E, $K_{p1},\ldots,K_{ej}$, $D_{p1},\ldots,D_{ps}$)■
We can construct binary concepts for a state of a relationship in the same way as we do for states of entities, but instead of E we use P_{e1}, P_{e2} :
(4.2.6.2) Schemas of K-concepts;
C_{k1} (P,A_1, $K_{11},\ldots,K_{1h}$,$D_1,\ldots,D_{1j}$);
…
C_{km} (P, A_m, $K_{m1},\ldots,K_{mr}$, $D_{m1},\ldots,D_{ms}$);
(b) Schema of the E-concepte
C_e (P, P_{e1}, P_{e2}, $K_{p1},\ldots,K_{pw}$, $D_{p1},\ldots,D_{pv}$)■
In this modeling of states, there is another kind of redundancy which we want to maintain. Here, the operations delete and update do not exist. We want to keep and maintain all the entered data in a database. 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 links between one entity and all of its states.
(ii) Note that in order for us to obtain m-attributes, they must not only satisfy their concepts, but also match the corresponding real world attributes, as defined in (3.3.3). The binary concepts of a state suggest that the attributes they contain belong to that state which is determined by the identifier of the state. The concept of a state and its binary concepts have the same identifier of state. Therefore by our definition of a concept, the schemas in Sections 4.2.5 and 4.2.6 are schemas of the same state. 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. However, we can apply the more formal approach: Using relationships between the binary relations from (4.2.6.2) we can get StateRelationship from 4.2.5. For example the relationship between C_e (P, P_{e1}, P_{e2}, K_{p1}, $\ldots,K_{pw}$, $D_{p1},\ldots,D_{pv}$) and C_{k1} (P, A_1, K_{11}, $\ldots,K_{1k}$, $D_{11},\ldots,D_{1m}$) gives: C_2 (P, P_{e1}, P_{e2} , $K_{p1},\ldots,K_{pw}$, A_1, $K_{11},\ldots$, K_{1k}, $D_{p1},\ldots$, D_{pv} , $D_{11},\ldots,D_{1m}$) . Here we applied (4.2.5.1), then we used the fact that the concepts C_e and C_{k1} are in the same state P. In the same way we can construct concepts C3,

…, Cn as relationships between 3, 4,…, or n binary concepts from (4.2.6.2).

4.2.7 States in the 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 StateEntity 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}, D_{n1},…,D_{ns});

(ii) Schema of the E-relation

R_e (P, E, K_{p1},…,K_{ej}, D_{p1},…,D_{ps})■ 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

(i) Let Rstate and R_{k1},…,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… A_n, $K_{p1,}$…, K_{nr}, D_{p1},…, D_{nq}) = (P, E, K_{p1} ,…,K_{ej}, D_{p1} ,…, D_{ps}) join R_{k1} (P,A_1, K_{11},…, K_{1k}, D_1, …, D_{1m}) join,…, join R_{kn} (P, A_n, K_{n1},…, K_{nr}, D_{n1} ,… ,D_{ns})■ This equation apparently holds true always because of the construction of the simple key, attributes, 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 binary relations are joined using common column P. The equation holds true for relations that represent both entities and relationships.

(ii) We can define the mapping f from the schemas of binary concepts of states to the schemas of binary relations of states and similarly we can define an inverse mapping for f. This mapping is determined by the corresponding identifiers of states. Now we can note that the decomposition of a concept of a state into a set of binary concepts has a corresponding decomposition of a corresponding relation into binary relations. Here, the relationship between two binary concepts corresponds to the join between a corresponding two binary relations. We link these binary structures with identifiers of state.

Example7: This example shows how certain complex databases, including "temporal databases" and "databases which maintain history", should be solved. The solution is related to two entityes 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.

5 Identifying and distinguishing entities

(i) The process of identifying goes from a subject to an entity. 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. An identifier is an attribute, i.e. it is simple and belongs to a property of an entity or relationship. Therefore we do not need total knowledge about the entity to identify it; an identifier is enough.
(ii) The process of distinguishing entities is important for the identifying of the same. However, these are two different processes.

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■
We use the construction 5.1 to construct:
a) A concept that is different from an 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 attributes by which we can identify the corresponding entity. If we need we can add an identifier to this set of attributes.
(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".
5.1 and 5.2 also apply to relationships.

5.3 Identification of a whole through its parts.

Regarding our definition of Particular Attributes and 4.2.2. we conclude:
Each intrinsic attribute of an entity has the same identifier as that entity.

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 Example9. 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 and 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. In this way we have direct access to attributes. 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 procedures which carry out this entry of new data.

6.3 Constructor

To construct this new data, we can use the following two constructors: the Constructor and the Closing Constructor. We create the new data using the Constructor, while we *close* the data with the Closing Constructor. These two constructors in some way correspond to the Constructor and Destructor from OOP. The differences are the following: the Closing Constructor does not delete or destroy or change data; it just says that the data is not valid from some point in time. Regarding 3.1, these two constructors are initiated by real world events. Using Constructor and Closing Constructor, 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 General 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 Simple 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 Boyce-Codd Normal Form■

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 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,…,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 $R_n(K, A_n)$ if and only if every relation that is a legal value for R(K, A_1, A_2,…,A_n) is equal to the join of its corresponding binary relations■
Definition Relational schema R(K, A_1, A_2,…,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 $R_n(K, A_n)$
if and only if
(i) Key K is simple
(ii) A_1, A_2,…,A_n are mutually independent attributes■
$R_1(K, A_1)$, $R_2(K, A_2)$,…, $R_n(K, A_n)$ are the corresponding binary schemas.
Simple Form has the following advantages over existing relational theory:
1. We have the conditions which a relation must satisfy in order to be in BCNF;
2. We do not need to put a relation into 2NF and 3NF to get it into BCNF.
3. The binary schemas can be immediately constructed■
(i) Let the relation schema R (K, P_1, …, P_m represent relationships between 2 entities where K is the key composed of the identifiers of the involved entities. P_1,…, P_m are intrinsic attributes of the relationship. Note that our design is always based on binary n-m relationships, except when business rules strictly demand n-ary relationships. If we introduce an identifier id, then the relation's schema becomes a set of the following binary schemas: R_1(id, K), R_2(id, P_1), …,R_{m+1} (id, P_m). Similarly we can decompose n-ary relationships to the binary relationships■

Example8:
Now from the table Car in Example7 we will construct the following four tables:

Table1		Table2		Table3		Table4			
CarKey	CarId	CarKey	Maker	CarKey	Type	CarKey	Color	DateFrom	DateTo
23	vin1	23	Buick	23	sedan	23	silver	1.1.2000.	12.20.2000.
24	vin1	24	Buick	24	sedan	24	blue	12.21.2000.	8.1.2001.
25	vin1	25	Buick	25	sedan	25	red	8.2.2001.	1.1.2005.
26	vin1	26	Buick	26	sedan	26	silver	1.2.2005.	999999
27	vin2	27	Honda	27	sedan	27	silver	3.15.2006.	999999
28	vin3	28	Ford	28	sedan	28	black	3.15.2006.	999999

Here we constructed four binary relations from the relation represented by the table Car in Example7. The first three tables each have two columns, one of which is for attributes and the other for key. In addition, Table4 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 Example8 we have the relation Car from Example7 represented in Simple Form with the corresponding binary relations■
(ii) 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 Example9 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 Example8 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. 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-concept and K-concepts. We say also that the meaning of the m-entity or m-relationship is determined as the totality of the entity's or 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 proc-

ess 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.

Example9: Here we will consider more than two 'knowledge-columns' related to the property Color from Example8. Actually, the "knowledge-columns" are related to the construction of data which represents individuals that fall under the concept Color. We modified Table4 from Example8 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	1.1.2000	999999	Mike
24	silver	1.1.2000	12.20.2000	Paul	1.1.2000	999999	Mike
25	blue	12.20.2000	999999	Paul	1.1.2000	999999	Mike
26	blue	12.20.2000	999999	Paul	1.1.2000	12.26.2001	Bill
27	blue	12.20.2000	999999	Paul	12.27.2001	999999	Bill

The first three new columns form a logical whole 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, 25, 27 are created by the Constructor while the rows containing CarKey = 24, 26 is created by the Closing Constructor. 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.

7. Conclusions

Regarding that the paper has a certain number of new contributions to the database theory, we added this section with the aim to clearly present this paper.

7.1 New Solutions for Database Design

(i) By introducing intrinsic attributes and entity identifiers, we define the procedure of constructing an m-entity and its m-attributes on a conceptual level. The benefit of this is that concepts of entities constructed in this manner can be mapped to relations that are in the Boyce-Code normal form.

(ii) Using our definition of m-n relationships and the m-entity construction procedure we always get relations which satisfy all normal forms. The term relation used in this Section refers to relations in the relational model. One of the paper's contributions to database design is that the construction of abstract objects is determined, i.e. the construction of m-entities, m-relationships, m-attributes and m-state is determined. We have also established how to identify these abstract objects. Another improvement is that the corresponding database objects are well constructed from the beginning. In contrast, the current database theory allows the construction of badly designed relations that must later be fixed using normal forms. Current database theories do not use the term abstract objects in the sense that they define abstract objects, explain their nature, construction or identification.
(iii) We have divided all databases into Simple and General databases. By introducing new data structures for General databases, we provide solutions for Temporal, Historic, and complex data oriented databases. Prior to this paper, these databases did not have general solutions. We apply General databases when we model states of entities or relationships. We apply Simple databases when we model entities and relationships. In other words we distinguish modeling of entities and relationships from modeling of their states. Thus, these objects are different types and have their corresponding operations. General databases do not have delete and update operations. According to (3.1), General database modeling supports an insertion only of a new primitive information.
(iv) With our solution we can determine who has created any of the data. The theoretical background for this problem was presented in Section 6.
A crucial question related to objects is their identification. We have determined the identification for both abstract and the corresponding real world objects. For example, identification of attributes is defined in (3.3.3).
(v) Identifiers are presented in section5. By using 5.2.(i) we can construct only identifiers for database's objects. In today theory, these identifiers are called surrogates, what is caused by poor understanding of identifiers.
By using the corresponding attributes, we can determine the real entity to which this identifier corresponds.
By using 5.2.(ii), we construct both, real identifiers and their corresponding identifiers in a database. Today, almost every real database uses this kind of identifiers.
By using 5.2.(iii), we can determine the entity for a given attribute.

7.2. Binary Concepts, Binary Files and Binary Relations

Binary concepts enable new theoretical and practical results. On a theoretical level, they enable a straightforward schema mapping between different data models as well as an inverse of the schema mapping. The advantage of using binary concepts is that the data model is expected to be constructed in elementary form. These elementary structures enable a more convenient and straightforward presentation of knowledge about data, knowledge about attributes, constraints, and transfer of data. On the other hand, we can build up (automatically) compound data structures by combining these elementary structures. They minimize redundancy.
We introduce effective solutions which decompose concepts, relations, and files to corresponding binary data structures.

7.3. New results in semantics

Our model is intended to be a semantic data model and regarding semantics we have the following new results:

(i) We give a general definition of a concept and of four particular concepts, which are of a general character. These definitions make the conceptual model real in the sense that the concepts are defined. We incorporate identifiers into our concepts and have departed from the idea that concepts are constructed only of attributes.

(ii) In (3.3.3) we present relationships among real attributes, m-attributes, universal attributes and the corresponding concepts. Using binary concepts and (3.3.3), we determine the construction of facts. A fact is defined as the smallest complete semantic unit. In contrast to the current database theory, we distinguish a fact from a factual sentence. We also distinguish factual sentences from logical sentences. Therefore, our approach is unique on a sentence level. As we have said in (3.4), we build compound sentences by combining factual sentences. Here, we speak only of compound sentences which express data structures, and we build these sentences by combining the following two processes. The first process combines logic, meaning, concepts, knowledge, facts, identifiers and other mental activities in our mind. The second process combines grammar and syntax to bind words in a sentence in order to express in language the result of the first process. We use the term knowledge for facts that are permanently stored as data in a database. An important property of knowledge is permanence i.e. knowledge is memorized and denotes facts. We also introduce two new kinds of knowledge: knowledge about attributes and knowledge about data. These two types of knowledge are based on corresponding facts about attributes and data.

(iii) The new semantics of the changes is established in this data model. It is related to states of entities and relationships and to the changes of these states. This semantics determines the identity of objects that are changeable, as well as the meaning of objects which have been changed. It enables us to identify states of entities and relationships.

(iv) A fundamental result in our semantics is that we link all changes of states of one entity (relationship) to the identifier of this entity (relationship) . Regarding entities, the following problem existed: How an entity which is changed to another entity is, in fact, the same entity. This problem is solved here. We give procedures, constructions and semantics for this problem.

We incorporated into this paper our ideas presented in [9], 2005.

7.4 Two events about information.

In 3.1 we defined only two events related to real world information. In this paper we assume that events imply time, not vice versa. A very important conesquence of our event approach is that any data in the database corresponds to certain event in the real world, in compliance with our definition in 3.1. In our approach it makes no sense that data that does not correspond to an event in the real world can exist. In case it does exist in the database, we can always determine who gave or stored information about the corresponding entity or relationship. In contrast to the current database theory, we have only two kinds of data about an entity or relationship:

(i) data that is related to an event which causes new information;

(ii) data which is related to an event which causes existing information to be not valid.
In our model, a database and the events of a database also belong to the real world. Therefore we apply only the two events defined in 3.1 for both so called, valid time and transaction time. If we have for example "valid time", system time and more then one "transaction time", we will again use only the two events defined in 3.1.

7.5 Database Design
(i) In the case of a Simple database, concepts of entities should be represented in accordance to 4.2.2, and concepts of relationships should be represented in accordance to 4.2.3.
(ii) In the case of a General database, concepts of entities' states and relationships' states should be constructed in accordance to 4.2.5 or 4.2.6.
(iii) For the events defined in 3.1 the corresponding documentation should be created. This is explained in 6.5(ii).
(iv) Note that a state of an entity (relationship) is defined as knowledge about the entity (relationship). Therefore, an identifier of a state is determined by knowledge about the corresponding entity (relationship)■
With 7.5(iii) and 7.5(iv) we determine a state of an entity in the real world.

References

[1] J. Barwise, J. Etchemendy, Model theoretic Semantics, in Posner, Michael, 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 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] V. Odrljin http://www.dbdesign10.com, created in September of 2005

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