INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 12(2):166-182, April 2017.

Domain/Mapping Model: A Novel Data Warehouse Data Mode

I. Bojičić, Z. Marjanović, N. Turajlić, M. Petrović, M. Vučković, V. Jovanović

Ivan Bojičić*, Zoran Marjanović, Nina Turajlić, Marko Petrović, Milica Vučković
Faculty of Organizational Sciences, University of Belgrade
Jove Ilića 154, 11000 Belgrade, Serbia
ivan.bojicic, marjanovic.zoran, turajlic.nina, petrovic.marko, vuckovic.milica@fon.bg.ac.rs
*Corresponding author: ivan.bojicic@fon.bg.ac.rs

Vladan Jovanović
Allen E. Paulson College of Engineering and Information Technology, Georgia Southern University
Statesboro, USA
vladan@georgiasouthern.edu

Abstract: In order for a data warehouse to be able to adequately fulfill its integrative and historical purpose, its data model must enable the appropriate and consistent representation of the different states of a system. In effect, a DW data model, representing the physical structure of the DW, must be general enough, to be able to consume data from heterogeneous data sources and reconcile the semantic differences of the data source models, and, at the same time, be resilient to the constant changes in the structure of the data sources. One of the main problems related to DW development is the absence of a standardized DW data model. In this paper a comparative analysis of the four most prominent DW data models (namely the relational/normalized model, data vault model, anchor model and dimensional model) will be given. On the basis of the results of [1][a], the new DW data model (the Domain/Mapping model-DMM) which would more adequately fulfill the posed requirements is presented.
Keywords: data warehouse, data models, relational/normalized model, data vault model, anchor model, dimensional model, domain/mapping model

[a]Reprinted (partial) and extended, with permission based on License Number 4057540167908 [2016] ©IEEE, from "Computers Communications and Control (ICCCC), 2016 6th International Conference on".

1 Introduction

A data warehouse can be defined as a model of a concrete business system representing a set of all of the states of that system during a given interval of time. The constant changes (organizational, legislative, functional, etc.) that a business system faces also reflect on the supporting data warehouse. Hence, one of the main issues, related to DW development and maintenance, is the inconsistency, between the actual system and its supporting data warehouse, which increases over time. Overcoming this discrepancy requires a flexible DW data model i.e. a data model which could be easily adaptable to the frequent changes in the business system.

An additional issue in the field of DW development is the absence of a standardized model for representing the structure of a data warehouse (i.e. a standardized DW data model). Existing approaches propose that the data should be organized in compliance with the third normal form (3NF) [2] or the multi-dimensional model [3]. Both approaches exhibit some limitations related to the difficulty in maintaining the data warehouse when the structure of the data sources changes. On the other hand both approaches are standardized by means of corresponding metamodels defined in the Common Warehouse Metamodel (CWM) [4]. Two additional approaches, aimed at addressing these limitations, have emerged in recent years, namely the Anchor model [5], based on data that has been normalized into the sixth normal form (6NF), and the Data Vault

Table 1: Fundamental concepts of the the data models

	Object	**Relationship**	**Attribute**	**Identifier**
Normalized model	Relation	Foreign Key	Domain	Primary Key
Data Vault model	Hub	Link	Satellite	Business / Primary Key
Anchor model	Anchor / Knot	Tie	Attribute	Primary Key
Dimensional model	Dimension	Fact	Attribute	Business / Primary Key

model [6] which can (but need not) also store data normalized into the 6NF. It should be noted that the Anchor and Data Vault models are not standard extensions of CWM.

By identifying the strengths and weaknesses of each of these models it is possible to establish the foundation for a new DW data model which would more adequately fulfill the posed requirements. The comparative analysis of these models is given in [1] (doi: 10.1109/ICCCC.2016.7496754) and based on those results and as an extension, this paper proposes a novel data warehouse data model: the Domain/Mapping Model (DMM).

The remainder of the paper is organized as follows: first the fundamental concepts of the DW data models will be identified and elaborated. Sections 2 is devoted to the analysis of the four most prominent DW data models. Section 3 details the groundwork for the proposed model, which is introduced in Section 4. Several examples illustrating the usage of the proposed DMM are given in Section 5. The final Section gives a brief summary of the work.

2 Comparative analysis of the Data Models

Data models are intellectual instruments for specifying the static characteristics of systems, i.e. for describing the objects, their attributes and their relationships in a stationary state [7]. As a data warehouse is defined as a model of a concrete business system representing a set of all of the states of that system during a given period of time, it is imperative that the underlying data model, be able to, not only support the specification of the system as it transitions through states, but also withstand changes in the business system or data sources.

The four most prominent models are analyzed in this paper: the Normalized model [2, 8, 9], the Data Vault model [6, 10–12], the Anchor model [5, 13] and the Dimensional model [3, 14, 15].

At the highest level of abstraction, all of the described models are based on several fundamental concepts, as depicted in Table 1.

2.1 Built-in semantics

The main point of difference among the models is the level of built-in semantics they provide.

The *Normalized model* does not presume any semantic constraints and, as such is extremely general, as the development of any given business model is based on mappings between sets. Furthermore, it does not provide any implicit concepts which would enable maintaining the history of changes of an object nor the values of its attributes.

The *Data Vault model* assumes that business objects have a stable identifier and somewhat alters the structure of the source by allowing for an object to be arbitrarily structured (i.e. its structure can be split into several *Satellites*). The *Data Vault model* is suited for tracing changes in the values of attributes, except for the *Hub* identifier (i.e. the *Business Key*).

The *Anchor model* is highly normalized. It provides two concepts, the *Anchor* and the *Knot*, for representing business objects. In addition, it enables the tracing of the history of all concepts, save for *Knots*.

The *Dimensional model* is based on the events that take place within a business system and the *Dimensions* which define them. Furthermore, it is possible to define numerical properties for expressing the quantitative aspects of the events. The tracking of the history of changes is based on the complex rules pertaining to changes in dimensions.

All of the models provide a single representation of an object (or entity) except for the *Anchor model* in which concepts of a concrete system can be represented by *Anchor* or *Knot* concepts. The main difference is that the *Normalized*, *Data Vault* and *Anchor models* are normalized, while the *Dimensional model* is denormalized.

A relationship between objects is represented through the *Foreign Key* concept in the *Normalized model*, which establishes a "tight relationship". The *Data Vault*, *Anchor* and *Dimensional models* define the relationship between objects through the *Link*, *Tie* and *Fact* concepts, respectively, wherein the relationship is realized as a table which stores the primary keys of the objects in the relationship. In addition, in *Dimensional model* it is customary to store additional derived or aggregated attributes in the structure of a *Fact*.

With regard to attributes, it should be noted that the *Data Vault* and *Anchor models* separate the structure of an object from the object itself, by using *Satellites* and *Attributes*, respectively, which reference the object via a foreign key. The difference between these two models is that, in the *Anchor model* a separate table is created for each attribute, while, in the *Data Vault model*, the attributes can be grouped according to various criteria into multiple *Satellites* of one object. The *Normalized* and *Dimensional models* keep the attributes within the structure of the object.

In all of the models each concept, which is used for representing an object type, has an identifier. The *Data Vault model* assumes that the identifier is the actual business key. Somewhat similarly, the *Dimensional model*, when using *Type 2 SCDs*, also expects the existence of a business key on the basis of which the dimension values will be grouped.

2.2 Resilience to change

As previously mentioned, the constant changes that a business system faces also reflect on the supporting data warehouse. Hence, one of the primary requirements related to data warehouse development is to provide the ability of absorbing changes in the structure of the data sources, without changing the structure of the underlying data model, in order to facilitate future maintenance and extensions of the data warehouse.

In this section the data models will be analyzed from the viewpoint of their adaptability and extensibility. More precisely, the evaluation of this aspect will be focused on establishing whether changes in the structure of the data sources can be handled simply through the addition of new concepts, without requiring modifications of the DW physical layer.

The *Normalized model* has proven to be the optimal data model for transactional systems since, on the one hand, it guarantees minimal data redundancy and, on the other, decomposes the structure of the system down to the level of its fundamental objects. However, in the data warehouse domain it demonstrates certain weaknesses, as data warehouse requirements differ considerably from transactional system requirements. The main weakness of the *Normalized model* lies in the fact that the attributes and relationships are built into the structure of the system's objects, which leads to a number of maintenance-related issues. Namely, any change in the structure of the source (attributes or relationships) will require changes in the structure of the Normalized model, as depicted in Fig.1, where the cardinality of the relationship between the *Employee* and *Organizational Unit* concepts is adjusted in order to allow for an *Employee* to be assigned to more than one *Organizational Unit*. In the depicted example it is necessary to create the *Position* table, transfer the existing data (pertaining to the relationship between the *Employee* and *Organizational Unit* entities) into the *Position* table, and delete the foreign key,

referencing the *Organizational Unit*, from the *Employee table*.

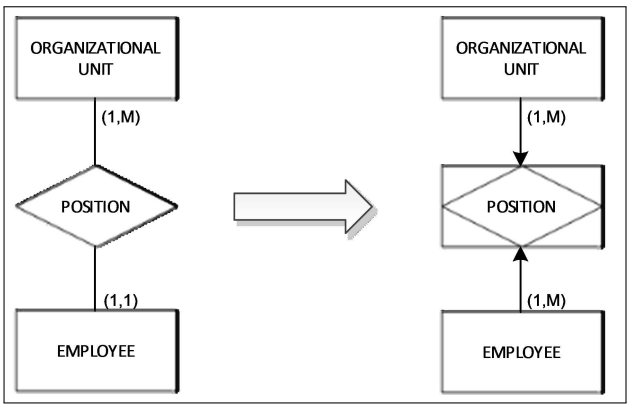

Figure 1: Cardinality change in the *Normalized model*

In addition, the *Normalized model* has some shortcomings when it comes to the reconciliation of source models since, due to the potential semantic heterogeneity of the source models, it may not be possible to design a single reconciled model, as demonstrated by *Golfarelli* in [15].

The *Data Vault model* exhibits much greater flexibility with regard to changes in the structure of the sources. Its flexibility derives from the underlying language, as the structure of an object is decoupled from the object itself, and placed in a physically separate concept: a *Satellite*. Furthermore, every relationship (i.e. *Link*), regardless of its cardinality, is always created as a table representing an aggregation of the related objects.

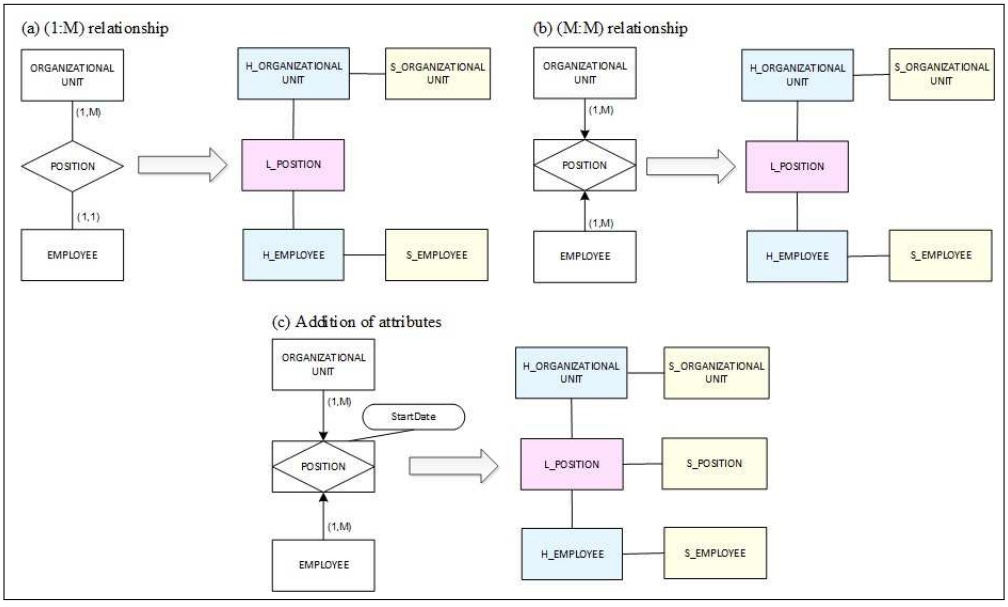

Figure 2: Flexibility of the Data Vault model

As depicted in Fig.2(a), in the initial structure of the source system an *Employee* could be assigned to one and only one *Organizational Unit*. Consequently, the corresponding *DV model* contains the *H_ Employee* and *H_ OrganizationalUnit Hubs* with the associated *Satellites*. The Hubs are related through the *L_ Position Link*. If the structure of the source changes to allow for an *Employee* to be assigned to more than one *Organizational Unit*, the target *DV model* will not undergo any changes, as shown in Fig.2(b).

Furthermore, if the *Position* concept is enhanced with an additional property, e.g. *Start-Date*, the *DV model* once again proves its flexibility, given that it is only necessary to extend (not modify) the existing model by adding an *S_ Position Satellite* (containing the introduced property) to the *L_ Position Link*, as illustrated in Fig.2(c).

Moreover, further modifications, e.g. relating the *Position* concept to other concepts of the source model, will not require changing the existing structure of the *DV model*, given that it allows for two *Links* to be related.

However, it should be noted that, even though in the previous examples the *DV model* did not undergo any changes at the physical level (i.e. it was only extended), the semantics of the source system have implicitly been changed by representing a strong entity (i.e. the *Position* aggregation) as a Link (i.e. event or relationship) in the *DV model*.

An additional drawback of the *DV model* is the fact that, the transformation process is irreversible, i.e. obtaining the structure of the original source model from an existing *DV model* is not possible. Because the *DV model* is semantically much poorer than the source models, certain information contained in the source models will inherently be lost in the transformation. For instance, in the previous examples (Figs.2(b) and 2(c)) it would be impossible to determine into which of the two source models the *DV model* would be reversibly transformed.

In keeping with the same example, the adaptability of the Anchor model, to changes in the structure of the sources, will be examined. All of the models have been created using the original Anchor modeling tool [16].

Initial model (Fig.3(a)) consists of two Anchors: *OU_ OrganizationalUnit* and *EM_ Employee*. The Tie between the two *Anchors* (*OU_ engages_ EM_ assignedTo*), wherein an Employee can be assigned to at most one *Organizational Unit*, is implicitly named on the basis of the roles of the related *Anchors*. Following the transformation the Tie becomes a table whose primary key is the identifier of the concept that participates in the relationship with a maximum cardinality of 1.

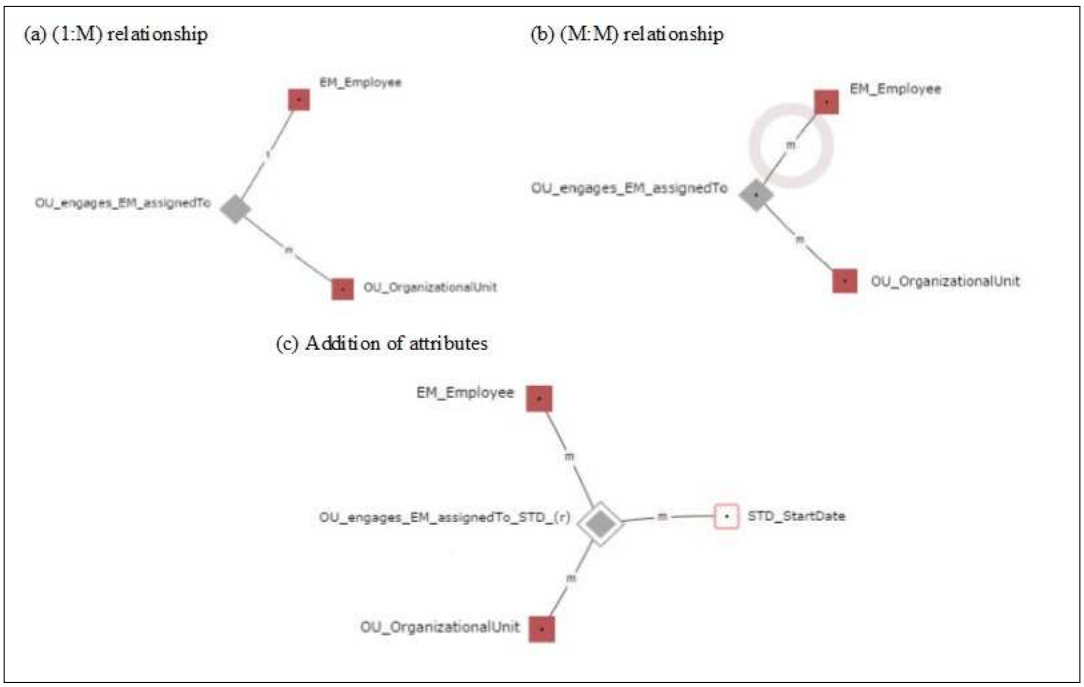

Figure 3: Flexibility of the *Anchor model*

If the cardinality changes, in order to allow for an *Employee* to hold more than one *Position*

(as depicted in Fig.3(b)), the only alteration of the model is extension of the primary key of the *Organizational Unit* with the primary key of the table representing the *Tie*.

However, if the structure of the source system changes so that a *StartDate* is added to the relationship (*OU_ engages_ EM_ assignedTo*), the *Anchor model* does not support this feature. Namely, the transformation of a relationship into an object, or the addition of structural features to a relationship, is not allowed. Instead, the *Tie* can reference a *Knot* representing a set of all of the start dates pertaining to the given *Position*, as depicted in Fig.3(c).

Consequently, at the physical level the table, representing the *Tie*, is modified by adding attribute (the primary key of the *Knot*) which becomes part of the composite key of the *Tie*.

The *Dimensional model* exhibits the same weaknesses as the *Normalized model*, as the addition of new attributes in the source model requires changing the structure of corresponding data warehouse *Dimension*. Yet the relationships in the *Dimensional model* are more stable, given that this model is process-oriented, i.e. it is designed on the basis of the core business processes of an enterprise. Nonetheless, the introduction of additional concepts in the source model (which become *Dimensions* in the data warehouse model) also requires the modification of the corresponding *Fact* in order to include a foreign key referencing the newly formed *Dimension*.

2.3 Temporal aspects

By defining a data warehouse as a collection of Time-Variant data [2], *Inmon* established one of the main requirements for data warehouses, namely that they must enable the tracking of past, current and future states of objects and events, as well as the points in time at which the changes occurred [17]. The notion that a transactional system should support more than one time dimension was first put forward in [18] when *Snodgrass* proposed the *Bitemporal* concept, splitting the time aspect into two dimensions: *Valid time* and *Transaction Time*.

Nowadays, as stated in [19], it is customary for the time aspect to be treated through three dimensions *Valid Time*, *Transaction Time* and *User-Defined Time*. It can be said that a data model supports the modeling of temporal aspects if it provides built-in mechanisms for capturing both the *Valid Time* and the *Transaction Time*.

The aim of this section is to explore whether the data models support the tracking of the entire history of business system objects, and if so, through which mechanisms.

The *Normalized model* does not provide built-in mechanisms for capturing temporal aspects, rather it is left to the data warehouse designer to decide which temporal concepts will be included and how they will be modeled [20].

The *Data Vault model* implicitly incorporates the *Transaction Time* into each concept (*Hub*, *Link* and *Satellite*) via the *Load Date* and *Load End Date* concepts. As this model assumes that the structure of objects will change over time, *Satellites* can be used to capture the changes in values which occur during the life-cycle of an object. Hence the *Valid Time* concept can only be applied to the structure of an object, not the object itself nor its relationships. Consequently, and in light of the underlying premise that business keys are immutable, the *Data Vault model* does not provide a built-in mechanism for tracking the *Transaction Time* for business keys. Nevertheless, such cases can be handled by using the *Same-As Link* whereby two *Hubs*, with different business keys, are asserted to be identical [6].

The *Anchor model* provides the Historized option for representing time-variant *Attributes* and *Ties* which, however, only allows for capturing their *Valid Time*. When this option is used the structure of each of the chosen concepts is supplemented with a *ValidFrom* attribute [21] which captures the beginning of the validity period of the value (e.g. Anchor modeling tool generates a *ChangedAt* attribute for capturing the *Valid Time*). It is implicitly assumed that the end of the validity period of a previous value corresponds to the beginning of the validity period of the

new value. However, such an assumption may not always be valid. In addition, if the Static option is used when creating the model, the possibility of tracking the history of value changes for such *Attributes*, or *Ties*, will be lost. The stated reason is that this option is to be used for values which are considered to be stable, e.g. the *Date of Birth*. Yet, this does not take into account the possibility that the value of an attribute may be incorrectly recorded in the source, or the possibility that several sources, storing different values of the same attribute, may exist. *Anchors* and *Knots* are considered to be immutable, which may not always be the case.

The *Transaction Time* (i.e. *RecordedAt*) is not incorporated into the structure of the concepts representing business system objects, but is stored as metadata in separate tables instead [13].

The Dimensional model enables the tracking of *Valid Time* through the various *SCD types* it offers (excluding *Type 1* and *Type 3 SCDs* which do not allow for tracking the *Valid Time* of an object). However, one of the main issues, related to tracking changes in dimensions, is the choice of the *SCD Type*, as switching to a different *SCD Type* at a later point in time inevitably leads to changes in the structure of the dimension. In addition, if the *Type 2 SCD* is chosen, changes is in the *Business Key* of an object cannot be tracked, as it is the basis for tracking changes in the other attributes.

In the *Dimensional model* the *Transaction Time* is not incorporated into the structure of the concepts representing business objects, instead it is recorded in separate log tables.

2.4 Completeness and traceability of data

In accordance with *Inmon's* definition [2] the second crucial characteristic of data warehouses is the Non-Volatility of data, which assumes that all of the data that has been integrated into a data warehouse must be retained in the data warehouse, unmodified, in order to ensure that a given query, executed at any point in time, will always produce the same, consistent result [22].

The completeness and traceability requirements are in direct collision with the *Single Version of the Truth* concept which assumes that the data, that is to be stored in the data warehouse, is prepared and filtered in advance. The *Single Version of the Truth* is, thus, the result of integrating data from multiple sources with the aim of providing a uniform basis for analysis and avoiding redundancy [23]. However, in order to achieve this aim, it is customary when designing a data warehouse, to choose, out of all of the data relating to a single concept and extracted from various sources, which one will represent the "truth" and, as such, be stored in the data warehouse. All other occurrences (i.e. those which do not conform to the "truth") are discarded and will not be loaded into the data warehouse. As a result, given that the data warehouse stores only part of the source data, it is impossible to conduct analyses on all of the values which exist in the source systems.

Consequently, the *Single Version of the Facts* concept emerged which assumes that the data warehouse stores all of the source data [24], thereby making it possible to provide different views for different users according to their specific needs. In essence, this approach promotes an *ELT* (*Extract-Load-Transform*) process in which the transformation of data takes place after the data warehouse has been loaded, as opposed to the traditional ETL approach in which the data is transformed prior to its being loaded into the data warehouse. As a result, the data warehouse will store all of the available data from all of the sources spanning the entire history of the business system. Therefore, the analytical processing of the data will be performed on-demand depending on the business users needs [24].

The aim of this section is to explore whether the data models provide mechanisms for, on the one hand, ensuring the completeness of the data which is transferred from the sources and, on the other, for tracing the stored data back to the sources from which it originated.

The *Normalized model*, as was the case with the temporal aspect, does not implicitly provide

Table 2: Comparison of DW Data Models

	Normalized	**Data Vault**	**Anchor**	**Dimensional**
Built/in semantics	No	Yes	Yes	Yes
Resilience to change	No	Partially	Partially	No
Temporal aspect	No	Partially	Partially	Yes
Completness	No	Yes	Partially	No
Traceability	No	Yes	Yes	No

concepts to support the traceability of data or the recording of data originating from multiple sources. Traceability can, for example, be accomplished by recording additional metadata for each n-tuple, if the *Single Version of the Truth* approach is adopted. However, if the *Single Version of the Facts* approach is adopted the *Normalized model* demonstrates certain weaknesses. Namely, storing data from several sources in the same model would require storing one n-tuple per source in which the same business concept is present. This leads to the issue of relating these n-tuples, i.e. it requires using additional concepts for relating n-tuples representing the same business object. Furthermore, it introduces redundancy (as the n-tuples representing the same business object contain a number of attributes with the same values) thereby eliminating one of the good traits of the *Normalized model*.

The *Data Vault model* supports the traceability of data by requiring that the source, from which the *Hub* was initially loaded, be recorded. Likewise, the *Satellite* and *Link* concepts include built-in attributes for recording the source from which the loaded values originated (i.e. the *RecordSource* attribute) [11]. However, the main shortcoming of this model lies in the fact that the structure of a Hub contains the business key, the value of which is assumed to be stable or rarely changes. Moreover, the structure of the business key might also change. These situations are resolved by introducing a new *Hub* which will be related to the original Hub via a *"Same-As"Link*. Consequently, more than one *Hub*, with the same attributes and the same relationships with other model elements, will exist. In addition, the *Data Vault model* enables tracing data from multiple sources, by recording the data source in a *Satellite*.

The *Anchor model* supports the traceability of data, via the metadata concept, as all time-variant concepts (thus excluding *Anchors* and *Knots*) can reference the metadata capturing the source of the data. In effect, this also means that it is not possible to record that, for a single object two identical values, originating from two different sources, are both valid at the same point it time. Reason for this is that, at the implementation level, there exists a *Unique Constraint* over a group of attributes: identifier, *Valid Time* and the value of the attribute.

The *Dimensional model* does not provide built-in mechanisms for tracing the stored data back to the sources. Furthermore, it does not allow for recording multiple values (originating from multiple sources) for a single object, during the same period of time.

3 Groundwork

In light of the previous discussion it can be concluded that none of the analyzed models completely fulfill all of the necessary requirements pertaining to DW data models (as summarized in Table 2). By defining and studying the issues, recognized in the previous section, a number of conclusions will be made which will set the grounds for designing a new DW data model.

Extensibility and adaptability of the data model

As previously demonstrated, data models with a higher degree of integration among objects and attributes or objects and relationships (i.e. the *Normalized* and *Dimensional models*) exhibit

less flexibility and adaptability to changes in the structure of the data sources in comparison to models with a lower degree of integration (i.e. *Anchor* and *Data Vault models*).

Example 1: The illustrative examples given in Section 2.2.

Conclusion 1: The data model should reinforce a "loose" coupling between the objects and their attributes and relationships, and support the automated transformation of source model concepts into the 6NF.

Resilience to changes in the values of object identifiers

Although this issue was not explicitly described in the previous analysis of data models (because the identifier is in fact part of the structure of an object), it is relevant because the analyzed models differ in the way that they handle object identifiers. An object identifier is defined as an attribute with a unique value where the inverse *Domain→Object mapping* also has a (0 or 1:1) cardinality. While such a viewpoint is justified in transactional systems, it cannot be applied to the business identifiers of objects when it comes to temporal systems with, potentially, several data sources.

Example 2.1: The Pension and Disability Insurance (PIO) Fund of the Republic of Serbia, established by the Law on Pension and Disability Insurance has two, in the most part, independently maintained information systems in its Belgrade and Novi Sad branches. Consequently, there are a number of cases in which the two branches assigned two different valid *Personal Numbers* (representing business identifiers) to the same person. Given that the integration of these two information systems requires the storing of both identifiers, the same object will have two different, yet simultaneously valid, values for the same identifier. On the other hand, in several cases a single *Personal Number* was assigned to more than one person.

Example 2.2: The Ministry of Interior of the Republic of Serbia is responsible for assigning *Unique Master Citizen Numbers* (in Serbian: *Jedinstveni Matični Broj Gradjana*- JMBG) to all citizens. However, there are some cases in which a citizen was mistakenly assigned two different JMBG numbers which are both valid at the same time. In addition there are a few cases of duplicate JMBG numbers, i.e. the same JMBG was assigned to different citizens.

Conclusion 2: The data model should provide for defining an object identifier, but also allow for a 1:M cardinality not only for the *Object→Domain mapping*, but for the inverse *Domain→Object mapping* as well. In other words, object identifiers should be attributes with multiple, possibly shared, values. The same holds for all other attributes of an object.

Resilience to changes in the structure of object identifiers

The structure of an object can, over time, undergo changes with regard to the attributes (or group of attributes) representing the business identifier. This issue is related to those models which presume the existence of a business identifier (the *Data Vault* and *Dimensional models*).

Example 3.1: Up till 1982, the business identifier of an *Insured Person* object in the PIO Fund of the Republic of Serbia was the *Personal Number*. With changes in legislation, JMBG were introduced as object identifiers.

Example 3.2: Up till 2003, the business identifier of a *Contributor* object in the PIO Fund was the *Registration Code*. With changes in legislation, a group of attributes was introduced as a composite object identifier: the *Tax Identification Number* (in Serbian: *Poreski Identifikacioni Broj* PIB), *Municipality* and *Street*.

Conclusion 3: The data warehouse data model should enable storing semantically different business identifiers for a single object in different periods of time.

Data redundancy

Even though most of the analyzed data models (save for the *Dimensional model*) dedicate special attention to this issue, data redundancy will inevitably occur as new data sources are added to the data warehouse at various points in time. This issues stems from the fact that all of the models structurally relate the attributes to their corresponding objects.

Example 4: The information system of the Ministry of Interior of the Republic of Serbia has, among many others, two subsystems: *Register of Births* (responsible for assigning and processing *Unique Master Citizen Numbers*) and *Human Resources*. When the first subsystem data is loaded, which ever one that may be, the JMBG will, as an attribute, become part of the object it belongs to (e.g. an *Individual*). The loading of the second subsystem data (e.g. *Human Resources* and the *Employee* object which also contains a JMBG attribute) will lead to the storing of the same JMBG values in two different places, independent of one another, in the same DW.

Conclusion 4: The domains, i.e. the sets from which the attributes of an object take their values, should be structurally independent, so as to allow for the same values to be used by the attributes of different objects in order to keep the data redundancy as minimal as possible.

Capturing the temporal aspect

As discussed in Section 2.3, the capturing of temporal aspects is only partly supported in most of the analyzed data models.

Example 5: The information system of the Customs Administration, which is a part of the Ministry of Finance of the Republic of Serbia, has, among others, two subsystems: the *Transit system* (*New Computerized Transit System* - NCTS) and *Reference Data* RD. The NCTS subsystem regularly relies on the RD codelists. The RD codelists have a validity period, and new codelists are introduced into the system at least one month before the beginning of their validity period. It is obvious that all of the system's objects are subject to the temporal aspect.

Conclusion 5: The data model should allow for capturing the temporal dimension, i.e. the valid time and transaction time for every object, attribute and relationship within the system. Furthermore, the capturing of the temporal aspect should be implicit, i.e. it should not depend on the expertise of the designer nor the degree of knowledge about the real system.

Capturing the temporal aspect

This issue was described in Section 2.4. While none of the analyzed data models were, for the most part, initially designed to satisfy this requirement, the *Data Vault* and *Anchor models* do exhibit certain flexibility due to the fact that attributes and relationships are not structurally integrated into the objects. However, the shortcomings of these models are exposed when the temporal dimension is introduced. Namely, when the source model is replaced by another model version the mechanisms provided by the two data models do not allow for tracking the continuity of the two sources, i.e. they cannot be perceived as two versions of the same model.

Example 6: It has been illustrated, through several examples, that it may be expected for the data warehouse to store several different, yet simultaneously valid, values for the same attribute.

Conclusion 6: The data model should implicitly allow for simultaneously storing several different values for a single object characteristic (relationship or attribute) while also maintaining a reference to the version of the source (model) from which the value was retrieved.

All data is equal

Example 7: Two of the analyzed models, namely the *Data Vault* and *Anchor* models, introduce novel concepts which differ semantically from the concepts which are customarily used for describing business objects. The *Data Vault model* provides the *Reference data* concept, while the *Anchor model* offers the *Knot* concept. Both concepts play a role in representing static, immutable objects.

The Ministry of Health of the Republic of Serbia maintains various sets of objects related, for example, to the organizational structure, human resources or medical equipment of the health facilities within its jurisdiction. The processing of such objects relies on the reference data which belongs to a separate *CLASSIFICATIONS* submodel, which represents a unique codelist system accessible by all business objects.

Such codelists would be represented as *Reference data* in *Data Vault* models or *Knots* in

Anchor models. The main limitation of such an approach is that codelists are regarded as immutable, even though they are managed by another business function, i.e. they are managed, like any other object in the system.

Conclusion 7: The data model should give equal importance to each type of data, regardless of whether it represents metadata, whether it originates from within the business system or the frequency with which it changes.

4 The Domain/Mapping model

Taking into account the issues and conclusions described in the previous section, a new model - the **Domain/Mapping model** (DMM) is proposed. The proposed DMM, depicted in Fig.4 is a general model which has been specifically designed to reconcile the semantic differences of existing data warehouse conceptual models, eliminate redundancy, be resilient to changes, allow for the capturing of temporal aspects, enable traceability, extensibility and adaptability, as well as to maintain a single version of the facts.

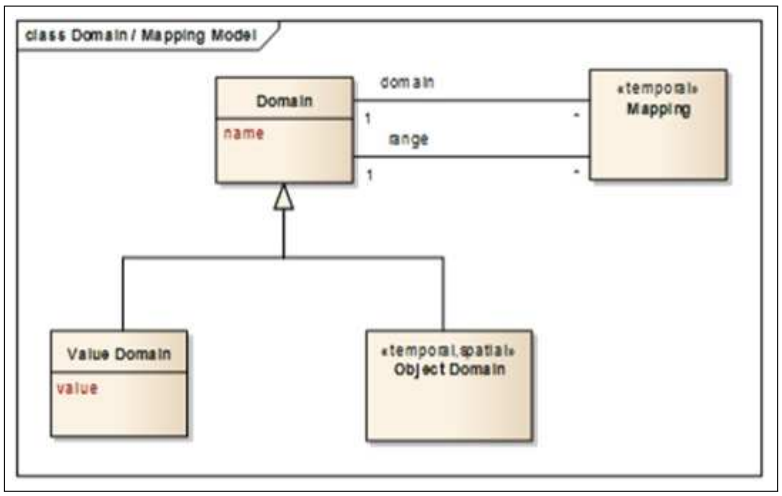

Figure 4: The Domain/Mapping model conceptual model

The core concept of the proposed model is the *Domain* which represents a set of values or objects. A set of values represents a *Value Domain* which is universal and does not depend on temporal and spatial aspects. It is a set of a finite number of atomic elements, from which the attributes of an object take their values. Each such set is predefined to include an additional element: the unknown (i.e. *null*) value.

Definition 1. A Value Domain $D_v = (I, V)$ is an ordered pair, where:

- I is a finite set of identifiers and
- V is a finite set of values.

A value domain is represented by an ellipse symbol with the name of the value domain inside the ellipse.

An *Object Domain* represents a set of real-world objects which are influenced by temporal and spatial aspects, i.e. the objects are mutable in time and space.

Definition 2. A Object Domain $D_o = (I, T, S)$ is an ordered triplet, where:

- I is a finite set of identifiers,

- T is a temporal indicator and
- S is a spatial indicator.

An object domain is represented by a double-lined ellipse symbol with the name of the object domain inside.

A *Mapping* is a concept which allows for forming the structure of the objects and their relationships. When an attribute is defined, then the mapping is between an *Object Domain* and a *Value Domain*. If a relationship between objects is defined, then the mapping is between the *Object Domains*. The forming of the structure of the objects and their relationships is influenced by the temporal aspects. Furthermore, the data warehouse structure is defined in accordance with a particular model version (different sources or versions of the same model).

Definition 3. A Mapping $M = (F, T, V)$ is an ordered triplet, where:

- M is the mapping between two domains,
- T is a temporal indicator and
- V is the version of the model in which the mapping is defined.

A mapping is represented by a solid undirected line with the name of the mapping on it.

Definition 4. F is a pair consisting of a mapping function i.e. a mapping $f(x) = D_d \rightarrow D_r$ and its inverse mapping $f\prime(x) = D_r \rightarrow D_d$, where:

- D_d is the domain of the mapping and,
- D_r is the codomain (i.e. range) of the mapping.

Definition 5. Mappings between two value domains are forbidden.

Definition 6. A Temporal Aspect $T = (T_{tt}, T_{vf}, T_{vt})$ is an ordered triplet of temporal dimensions, where:

- T_{tt} is the transaction time,
- T_{vf} is the beginning of the validity period and
- T_{vt} is the end of the validity period.

Definition 7. A Spatial Aspect $S = (L_t, L_g)$ is an ordered triplet of temporal dimensions, where:

- L_t is the latitude and
- L_g is the longitude.

Definition 8. A Latitude $L_t = (G_d, G_m, G_s, G_{dw})$ is an ordered quadruple, where:

- G_d = the degree ranging from 0-90,
- G_m = the minutes,
- G_s = the seconds and
- G_{dw} = the direction (north/south).

Definition 9. A Longitude $L_g = (G_d, G_m, G_s, G_{dh})$ is an ordered quadruple, where:

- G_d = the degree ranging from 0-180,
- G_m = the minutes,
- G_s = the seconds and
- G_{dh} = the direction (west/east).

5 DMM examples

In this section several examples, demonstrating the utilization of the proposed DMM model, will be given. The depicted examples also illustrate the aptness of the proposed model with regard to the issues and the fulfillment of the requirements postulated in Section 3.

Two simplified submodels of the PIO Fund model, depicting changes in the structure of the *Insured Person* concept during a given period of time, are shown in Fig.5. As depicted, the initial *Insured Person* concept was later modified by introducing the JMBG, which then also became the business identifier of the object due to changes in legislation. Up till then the business identifier was the *Personal Number* (PN). The fulfillment of ***Conclusions 1-3*** will be demonstrated through this example.

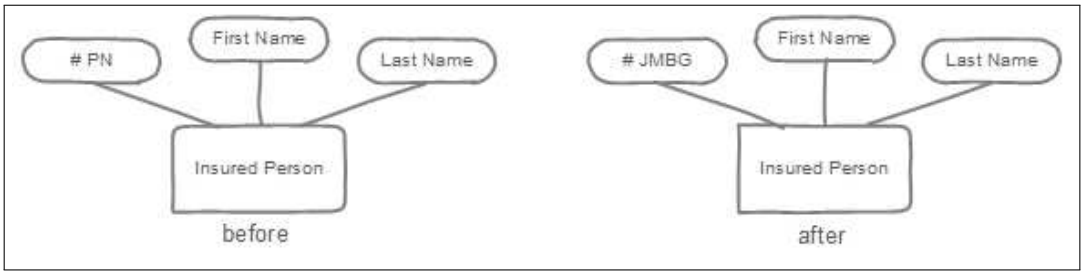

Figure 5: Business identifier replacement

The following Fig.6(a) depicts the DMM model with three domains: *Personal Numbers*, *First Names* and *Surnames*. It also contains the *Insured Person* concept, whose concrete instances are related to the defined domains, via the *Mapping* concept.

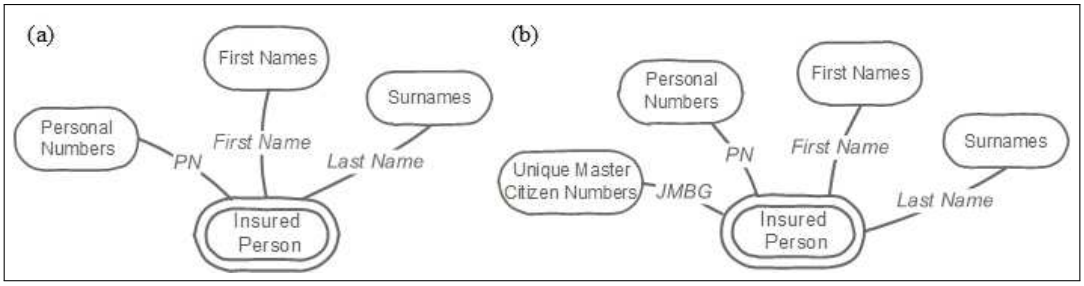

Figure 6: The initial (a) and modified (b) *Insured Person* DMM model

All mappings have an M:M cardinality (i.e. all attributes are multi-valued) which satisfies the requirement that several values of the same attribute can be stored at the same time, across two dimensions: the temporal and source dimensions. Namely, if several different systems store different values of a single attribute, and if the value of the attribute may change over time, the proposed model would be capable of upholding these changes. This is possible due to the

fact that the mapping of domains is accomplished through the *Mapping* concept which implicitly stores both the valid time and the version of the source metamodel. Consequently, the data warehouse stores all values of all data source model versions, without loss of information.

Fig.6(b) illustrates this scenario, when a new attribute, which also becomes a new object identifier, is added. At the physical level, all of the introduced concepts are implemented as tables. As depicted in Fig.6(b), the initial model was modified by simply adding new structures, without altering or deleting existing ones, i.e. by mapping corresponding domains.

The next example demonstrates how the elimination of data redundancy is achieved by keeping the domains (from which the attributes take their values) structurally independent, thereby allowing for several different attributes to use the defined domains at the same time. The depicted example shows that different business functions from different data sources use the same value domains. Fig.7 depicts a simplified *HUMAN RESOURCES* submodel (of the model of the Ministry of Interior of the Republic of Serbia) which contains an *Employee* concept consisting of the *JMBG*, *First Name*, *Last name* and *Birth Date* attributes. The first three attributes are mappings onto value domains, while the *Birth Date* is a mapping onto an object domain *DATE*, which is itself composed of three value domains: *Days*, *Months* and *Years*.

If a new data source is introduced, it is possible to reuse the existing domains, in order to eliminate redundancy, without altering the existing data warehouse structure. The following Fig.7 depicts a *REGISTER OF BIRTHS* submodel of the same business system, which describes the part of the system related to the JMBG of citizens. This submodel maintains the *JMBG* numbers of all citizens and includes all of the JMBG numbers from the human resources department, i.e. those belonging to the employees of the Ministry of Interior. It is customary for those *JMBG* numbers to be stored redundantly, which leads to data anomalies. The redundancy is eliminated by mapping both concepts (*Employee* and *Individual*) to the value domain *Unique Master Citizen Numbers*.

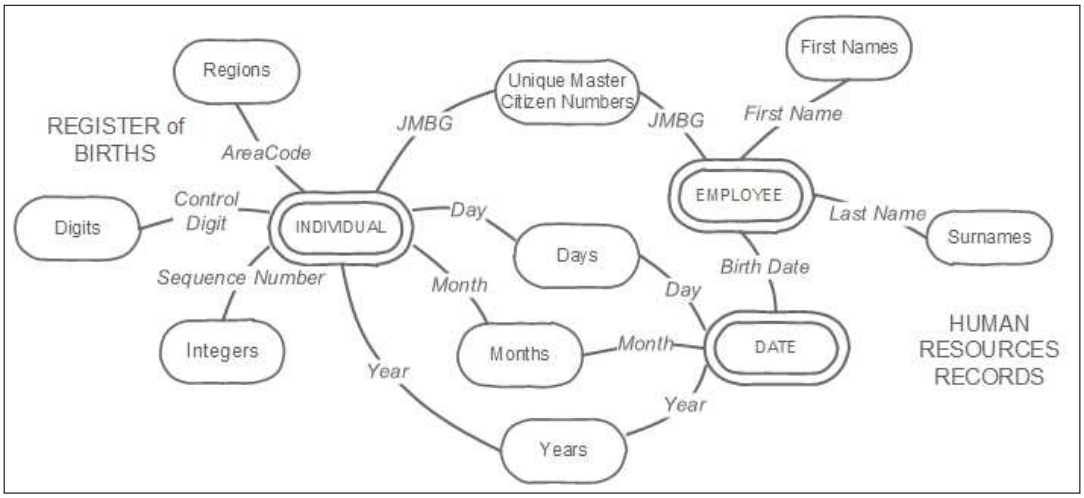

Figure 7: The absence of data redundancy in the DMM model

Furthermore, the Individual concept also reuses the *Days*, *Months* and *Years* value domains, as the concrete values of these domains are utilized for constructing the *Unique Master Citizen Numbers* (the first 7 digits of the JMBG correspond to the date of birth of an individual: two digits for the day, two for the month and the last three digits of the year of birth). By building a net of domains, which can be used for various concepts, data redundancy is kept minimal, which addresses ***Issue 4*** described in Section 3.

In addition, the DMM fulfills the requirement postulated in ***Conclusion 5***, by implicitly

capturing the valid time and transaction time for all objects stored in the data warehouse, as well as for the mappings they participate in.

The fulfillment of the requirement postulated in ***Conclusion 6***, is accomplished by using an implicit M:M cardinality for object-attribute or object-object mappings. Consequently, it is possible for a single object to have several different values for the same attribute or to be simultaneously related to several objects of the same type.

Finally, the fulfillment of the requirement postulated in ***Conclusion 7***, is depicted by the following two submodels (Fig.8). The first shows a simplified *MEDICAL EQUIPMENT* submodel (which is part of the Ministry of Health model).

If the *Equipment Codelist* is modeled without the capability of tracing changes in its values through time, it would not be possible to maintain the classification model without changing its structure. The following submodel *CLASSIFICATIONS*, shows how a new source can be easily introduced into the data warehouse without requiring changes in the existing model, even if it the two models are not at the same level of abstraction.

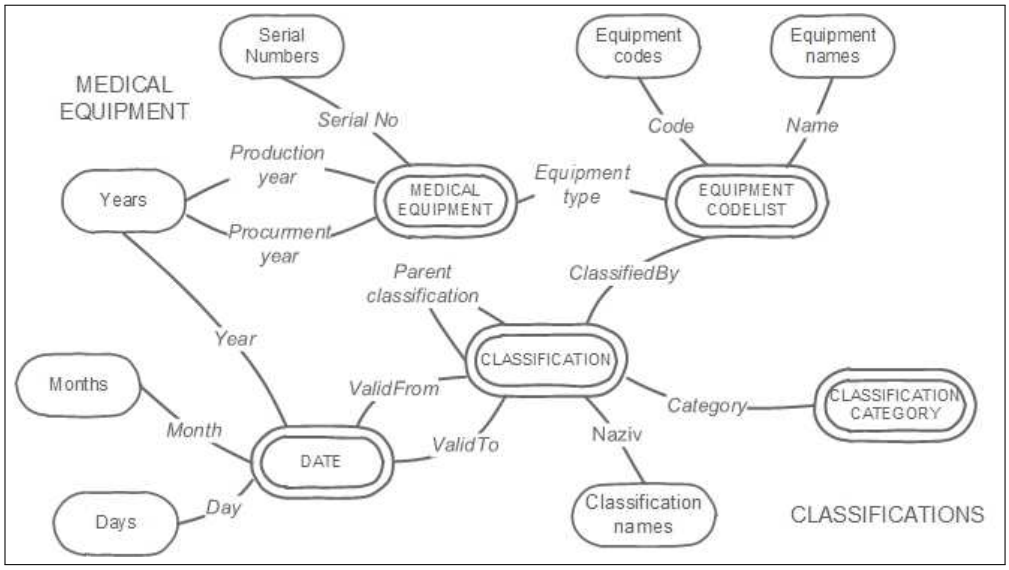

Figure 8: Medical equipment

As demonstrated by the previous examples, a DMM consists of two fundamental concepts, *Domains* and *Mappings*, which provides complete flexibility at the physical level while overcoming the weaknesses of the presented data warehouse data models.

Conclusion

The proposed DMM is capable of absorbing the changes which may occur in the source structures, trace the data back to their sources, and implicitly keep track of both temporal and spatial dimensions, while at the same time providing an integrated view of the data stored in a DW.

In addition, given that the DMM is a general model it can reconcile the semantic differences of the conceptual models used for describing data warehouses, and is, as such, completely independent of the underlying conceptual model.

This opens up the possibility for utilizing the DMM for building the data warehouse modeled by most of the existing conceptual models.

Bibliography

[1] I. Bojičić, Z. Marjanović, N. Turajlić, M. Petrović, M. Vučković and V. Jovanović (2016), A comparative analysis of data warehouse data models, *Computers Communications and Control (ICCCC), 2016 6th International Conference on*, IEEE Xplore, e-ISBN 978-1-5090-1735-5, doi: 10.1109/ICCCC.2016.7496754, 151-159.

[2] W. Inmon, *Building the Data Warehouse*. Wiley, 2002.

[3] R. Kimball, L. Reeves, M. Ross and W. Thornthwaite (1998), *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses*, Wiley, 1998.

[4] Object Management Group (2002); *Common Warehouse Metamodel*, available at: `http://www.omg.org/cgi-bin/doc?formal/03-03-02.pdf`

[5] O. Regardt, L. Ronnback, M. Bergholtz, P. Johannesson and P. Wohed (2009); Anchor Modeling: An Agile Modeling Technique Using the Sixth Normal Form for Structurally and Temporally Evolving Data, *in Proc. of ER09 (Brazil)* , LNCS, 5829(1): 234-250.

[6] D. Linstedt (2010); *Data Vault Modeling Specification v1.0.9.*, available at: `http://danlinstedt.com/datavaultcat/standards/dv-modeling-specification-v1-0-8/`

[7] B. Lazarević, Z. Marjanović, N. Aničić and S. Babarogić (2010). *Baze podataka*. Fakultet organizacionih nauka, 2010 (Textbook in Serbian).

[8] E. F. Codd (1969); Derivability, Redundancy and Consistency of Relations Stored in Large Data Banks, *IBM Research Report*, San Jose, California, 1969.

[9] E. F. Codd (1970); A Relational Model of Data for Large Shared Data Banks, *in Communications of the ACM*, 13(6):377-387.

[10] D. Linstedt (2002); *Data Vault Series 1 - Data Vault Overview*, available at: `http://www.tdan.com/view-articles/5054/`

[11] D. Linstedt (2011); *Super Charge Your Data Warehouse: Invaluable Data Modeling Rules to Implement Your Data Vault*, CreateSpace Independent Publishing Platform, 2011.

[12] D. Linstedt (2003); *Data Vault Series 2 - Data Vault Components*, available at: `http://www.tdan.com/view-articles/5155/`

[13] L. Ronnback, O. Regardt, M. Bergholtz, P. Johannesson, P. Wohed (2010); Anchor modeling - Agile information modeling in evolving data environments, *in Data & Knowledge Engineering*, 69(12):1229-1253.

[14] R. Kimball and M. Ross (2013); *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*, Wiley, 3rd ed., 2013.

[15] M. Golfarelli and S. Rizzi (2009); *Data Warehouse Design - Modern Principles and Methodologies*, McGraw - Hill, 2009.

[16] Anchor Modeling Tool, available at: `http://www.anchormodeling.com/modeler`

[17] E. Malinowski and E. Zimanyi (2008); *Advanced Data Warehouse Design - From Conventional to Spatial and Temporal Applications*, Springer-Verlag Berlin Heidelberg, 2008.

[18] R. T. Snodgrass, I. Ahn (1986); Temporal Databases, *in IEEE Computer*, 19(9):35 - 42.

[19] C. Jensen, J. Clifford, R. Elmasri, S. K. Gadia, P. J. Hayes, S. Jajodia (1994); A Consensus Glossary of Temporal Database Concepts, *SIGMOD Record*, 23(1):52 - 64.

[20] H. Gregersen, J.S. Jensen (1999); Temporal Entity-Relationship models a survey, *IEEE Transactions on Knowledge and Data Engineering*, 11: 464-497.

[21] L. Ronnback, O. Regardt, M. Bergholtz, P. Johannesson, P. Wohed 92010); *From Anchor Model to Relational Database*, available at: `http://www.anchormodeling.com/wp-content/uploads/2010/09/AM-RDB.pdf`

[22] M. Golfarelli, J. Lechtenborger, S. Rizzi, G. Vossen (2006); Schema versioning in data warehouses: Enabling cross-version querying via schema augmentation, *in Data & Knowledge Engineering*, 59(2):435-459.

[23] B. Inmon (2004); *The Single Version of The Truth*, Business Intelligence Network (Powell Media LLC), available at: http://www.b-eye-network.com/view/282.

[24] R. Damhof (2008); *The next generation EDW*, available at: `http://prudenza.typepad.com/files/`