Computer Science and Information Systems 11(2):569–599 DOI:10.2298/CSIS130523034K

# A Direct Approach to Physical Data Vault Design

Dragoljub Krneta[1], Vladan Jovanović[2], and
Zoran Marjanović[3]

[1] Lanaco Information Technologies, Knjaza Milosa 15,
78000 Banja Luka, Bosnia and Herzegovina
dragoljub.krneta@lanaco.com
[2] Georgia Southern University, 1500 IT Drive,
Statesboro, Georgia 30458, USA
vladan@georgiasouthern.edu
[3] Faculty of Organizational Sciences, Jove Ilica 154,
11000 Belgrade, Serbia
zoran.marjanovic@fon.rs

**Abstract.** The paper presents a novel agile approach to large scale design of enterprise data warehouses based on a Data Vault model. An original, simple and direct algorithm is defined for the incremental design of physical Data Vault type enterprise data warehouses, using source data meta-model and rules, and used in developing a prototype case tool for Data Vault design. This approach solves primary requirements for a system of record, that is, preservation of all source information, and fully addresses flexibility and scalability expectations. Our approach benefits from Data Vault dependencies minimizations and rapid loads opportunities enabling greatly simplified ETL transformations in a way not possible with traditional (i.e. non data vault based) data warehouse designs. The approach is illustrated using a realistic example from the healthcare domain.

**Keywords:** Enterprise data warehouse, system of records, design automation, Data Vault model.

## 1. Introduction

In this section we delineate the scope of the work, introduce baseline terminology, present selected requirements for design automation of a Data Vault (DV) type data warehouse, and explain the organization of this paper. The scope of the work is the design of DV based Enterprise Data Warehouse (EDW) / Data Mart (DM) content. The paper does not addresses storage and distribution opportunities i.e. recent advances such as (Hadop, NoSQL, etc.).

Data warehouse (DW) is a subject oriented, nonvolatile, integrated, time variant collection of data in support of management's decisions [1]. A DW may be used to support permanent systems of records and information management (compliance and improved data governance). Data marts are small data warehouses that contain only a subset of the EDW [2]. The data mart provides the platform for Online Analytical Processing (OLAP) analysis. Therefore, OLAP is a natural extension of the data

warehouse. The results from OLAP analysis can be presented visually, which enables improved comprehension [3]. The data mart is a part of data storage, but usually contains summarized data [4]. The Extract, Transform, Load (ETL) process involves fetching data from transactional systems, cleaning the data, transforming data into appropriate formats and loading the result to a warehouse [5]. In the ETL process, data from data sources is extracted by extraction routines. Data are then propagated to the Data Staging area where they are transformed and cleaned before being loaded to the data warehouse [6], [7]. An Important element of the DW is metadata, including definitions and rules for creating data [8]. Metadata play an important role in data warehousing. Before a data warehouse can be accessed efficiently, it is necessary to understand what data is available in the warehouse and where the data is located [2].

The design of data warehouses is a difficult task. There are several problems designers have to tackle. First of all, they have to come up with semantic reconciliation of the information lying in the sources and the production of an enterprise model for the data warehouse [2]. Data warehouses can be distinguished by the type of architecture. Bill Inmon [9], [1], [10] proposed the CIF (Corporate Information Factory) as an integrated data warehouse, i.e. database in the third normal form (3NF), from which multidimensional data marts are to be derived. The second option is bus architecture, defined by Ralph Kimball [11], [12], [13] where a data warehouse is just a collection of data marts with conformant dimensions. Data Warehousing 2.0 (DW 2.0) is a second-generation attempt to define a standard Data Warehouse architecture. One of the advantages introduced in DW 2.0 is its ability to support changes of data over time [10].

Data modeling techniques for the data warehouse differ from the modeling techniques used for operational systems and for data marts. This is due to the unique set of requirements, variables and constraints related to the modern data warehouse layer. Some of these include the need for an integrated, non-volatile, time-variant, subject oriented, auditable, agile, and complete store of data. To address these needs several new modeling approaches have been introduced within the Data Warehouse/Business Intelligence industry. Among these are Data Vault modeling and Anchor Modeling [14].

The Data Vault approach is introduced by Daniel Linstedt [15], [16], [17], to solve the problems of flexibility and performance, enabling maintenance of a permanent system of records [18]. Data Vault (DV) model is recognized by the modeling style using Hub, Link and Satellite entities. In terms of entity relationship data models, a Hub entity holds identities, typically business keys (or their combinations), Link entities are representation of foreign key references (typically used to represent transactions between two or more business components i.e. Hubs). A Satellite entity shows context information i.e. attributes of a Hub or a Link. We can split out satellites by: rate of change, type of data and source system. [19]. See the example of a Data Vault in figure 1. The main difference from traditional ER data modeling style is in representing attributes, namely Hub/Link entities which have relationship/identify Satellite entities (representing traditional attributes in time).

Anchor Modeling is a database modeling technique built on the premise that the environment surrounding a data warehouse is in a constant state of change, and furthermore that a large change on the outside of the model should result in only a

small change on the inside of the model [20], [21]. The goal of using the anchor modeling technique is to “achieve a highly decomposed implementation that can efficiently handle growth of the data warehouse without having to redo any previous work” [21], mimicking the ideas of isolated semantic temporal data put forth in DW 2.0 architecture [22].

Data Vault and Anchor models are characterized by strong normalized data and insensitivity to changes in the business environment, and adapted to frequent changes, as well as the modeling of small parts of the data model, without the need for redesign.

Sixth Normal Form (6NF) is a term used in relational database theory by Christopher Date to describe databases which decompose relational variables to irreducible elements. While this form may be unimportant for non-temporal data, it is certainly important when maintaining data containing temporal variables of a point-in-time or interval nature [22], [23]. In [23], Chris Date points out several shortcomings when attempting to model fully temporalized data using standard Fifth Normal Form (5NF) principles and introduces Sixth Normal Form (6NF) as "A relvar (table) R is in 6NF if and only if R satisfies no nontrivial join dependencies at all, in which case R is said to be irreducible" [23].

The Data Vault model, in the extreme case where a satellite table consists of one attribute, becomes a 6NF design.

According to [16], [24], a Data Vault makes most sense in the case of distributed data sources. To ensure traceability of data, the Data Vault is not transforming data from different sources before they are loaded into the warehouse, thus enabling permanent system of records (i.e. Data Vault). This differs from the CIF where data are consolidated up front.

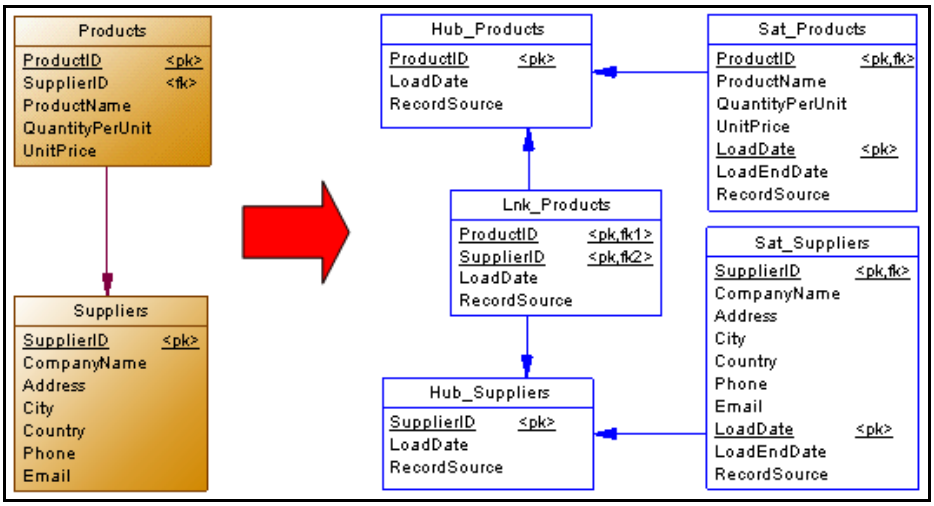


**Fig. 1.** An example of mapping a relational to the data vault physical model

A system theoretic block diagram representation, figure 2 from [25], shows explicit separation of data warehouse data into permanent original data (DW staging or to put it more precisely, a Data Vault) and derived data in data marts. The DB mimics

system's state at the last updated moment in time while DW tracks the system's state over time.

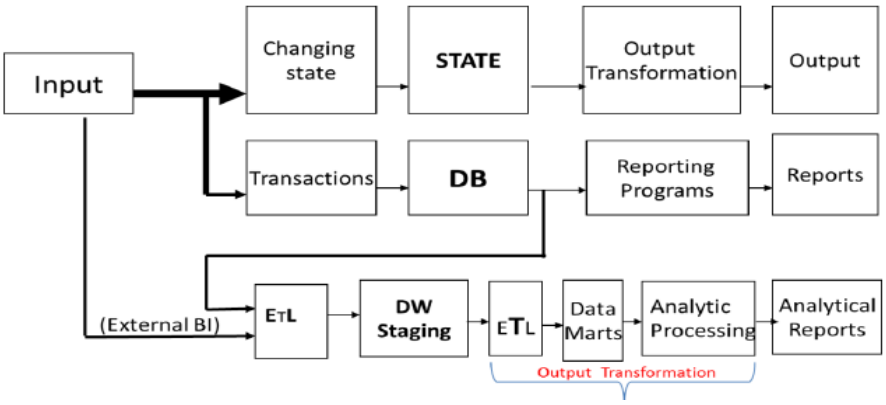


**Fig. 2.** DW State history (Data Vault) and DW Reporting Area

Methodologies of data warehouse design, such as ([1], [13], [26], [27], [19]) are not fully formalized. Based on the starting point to be used in determining the DW content, the following four classes of approaches to DW design can be recognized:

1. Data-driven (or supply-driven) [9], [28], [29], [30], [31] is based on an analysis of the data, the database schema and the identification of available data.
2. Demand-driven (also user-requirements driven) [11], [28], [30], [31] starts with the determination of requests for information from the user, but mapping that to available data remains a problem.
3. Goal-driven [32], [33], [30] is based on a business strategy i.e. begins with an analysis of key business processes, but according [28], [34] is a subset of Demand-driven approach.
4. The Hybrid approach combines user requirements with analysis of management and operational data. It distinguishes the interleaved and sequential access. For sequential access, the data-driven and demand-driven paradigms are applied independently of each other, one after the other, while in the interleaved hybrid approach both paradigms are applied in parallel [34].

This work is part of a larger research program, the DW21 dealing with the next generation DW design technology [35] where Data Vault is a baseline approach to DW design and most of the requirements are derived from a developer's standpoint as shown in figure 3.

One practical problem for organizations building data warehouses using different methodological approaches is that the process of designing a data warehouse including a process of identifying and/or generating measures and dimensions (or Hub, Link and Satellite concepts in case of Data Vault approach) is not sufficiently automated.

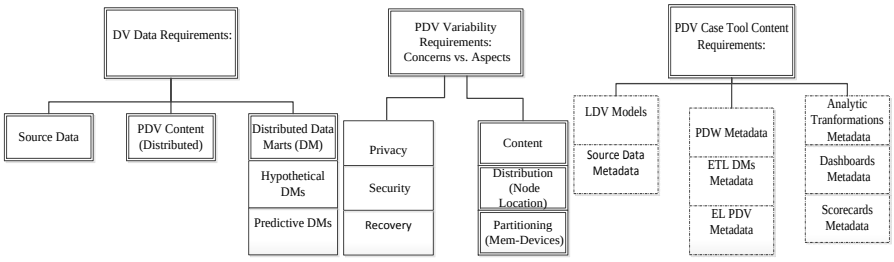


**Fig. 3.** Some Requirements of DW 2.1 Research Program

The rest of the paper is structured as follows: section 2 reviews work on data warehouse design automation, section 3 presents our approach to physical Data Vault design, section 4 illustrates related Physical Data Vault (PDV) design tool we developed to automate prototyping and implementation of data vaults based data warehouses, section 5 illustrates the use of the tool on a real example for distributed data base sources, and section 6 outlines future work.

## 2. A Review of Data Warehouse Design Automation

The starting point for review is selection of relevant works in the field of design automation for DW content (to narrow the scope somewhat, we did not emphasize a large body of work dealing primarily with ETL). Our choices are based on perceived contributions to the theory and practice of design, confirmed by the number of citations (at www.harzing.com).

A semiautomatic sequential hybrid approach of conceptual modeling of a data warehouse that starts with E/R model is proposed in [36]. After the analysis and design of the conceptual/logical model, the identification of facts is followed by semi-automatic creation of attribute trees in accordance with the requirements. The next step is to identify the dimensions, measures and aggregate functions after which logical and physical schema of data warehouse is performed. Improvement of the procedure is given in a textbook form in the [26].

The Hybrid approach in designing multidimensional database warehouse on the basis of transactional normalized systems is proposed in [37]. In this approach an ER diagram is transformed into a Structured Entity Relationship Models (SERM) diagram.

Phipps and Davis [38] start with a data-driven approach, and later uses the demand-driven approach (sequential hybrid approach). The initial scheme is obtained using a multidimensional E/R model. This paper proposes an algorithm for obtaining a conceptual scheme of transactional schemas. The algorithm uses the numeric fields and relationships between entities as the basis for the creation of Multidimensional E/R schemes. The algorithm is applied to each data model where data can be divided into numeric, date/time, and text data types. In addition, this approach can be used to evaluate candidates for the conceptual schema using custom queries.

Peralta (et al.) [39] represents a step forward in the automation of data warehouse based on the rules applying existing knowledge in the data warehouse design. The

proposed rules are built into the design strategy that is run in accordance with the requirements and data extraction based on the transactional database schema. The framework consists of: rules, guidelines for the design based on non-functional requirements, the mapping scheme of the original database to a conceptual scheme, and a number of schema transformations. This approach allows the use of existing design techniques, with improved productivity.

Romero and Abello [40] propose a semi-automatic method for recognizing business multidimensional concepts from domain ontology representing different and potentially heterogeneous data. With the use of ontologies and tools it can search for multi-dimensional patterns. Simulations of real cases to verify the algorithm are performed as well as theoretical analysis of the algorithm. This approach is able to integrate data from heterogeneous sources that describe their domains through ontologies. One of the most promising areas where the method can be applied is the semantic Web, which contributes to the extraction and integration of external data from the Web.

Zepeda (et al) [41] introduces a semi-automatic process to build a DW based on MDA (Model Driven Architecture). It starts by collecting and consolidating user requirements. Other steps include recognizing structural information from the original database schema supported by an automation technique. This approach uses a tool to guide designers toward effective solutions in line with customer requirements.

Nazri (et al) [42] is based on a model of data source and proposes a semi-automatic tool that uses lexical ontology as the knowledge base. Once you identify the facts and dimensions, the generation of a multidimensional model is made with minimal involvement of a user. The method is illustrated using a semantic lexicon WorldNet as knowledge domain, to identify measures and dimensions.

Zekri (et al) [43] introduces a semi-automatic approach for the design of multidimensional diagrams. Conceptual Data Model (CDM) is first translated into the multivariate pattern, and then refined with user requirements. Both steps use graphs as a formalism for the representation of the data model for decision making.

Design automation is not an easy process because, in addition to the identification of the original database schema, it attempts to formalize the requirements of end users. However, certain steps must be performed manually, for example, the identification of business keys and measures. Table 1 provides a comparative overview of the various approaches, including a new one based on the Data Vault. Note that the public domain literature concerning the design of Data Vault systems [15], [16], [24], [44], [45], [46] does not elaborate on actual design process automation.

Our direct approach to automatic generation of Data Vault physical models is based on metadata schemas of structured data sources (transactional databases) taking into account some semi-structured and unstructured sources. In addition, compared to the alternatives (Table 1), our direct approach to the physical design automation of data warehouse is achieved through the use of rules. It should be noted that majority of listed approaches are academic and that only Golfarelli [26] has a long tradition of industrial use which is also a characteristic of the Data Vault approach itself [15], [16], [44]. A direct physical design for data warehouses is practical only in the case of a data vault type of data warehouses and is made possible by separating storage of identities (permanent keys) from evolving relationships and characteristics (attributes).

**Table 1.** Comparison of data warehouse design approaches

| Approach | Generating | | | Data Source | | Using Rules |
|---|---|---|---|---|---|---|
| | Conceptual DW model | Physical DW model | Data Vault | Structured | Semi-structured | |
| [36] | X | X | | X | | |
| [37] | X | | | X | | |
| [38] | X | | | X | | |
| [39] | X | | | X | | X |
| [40] | X | X | | X | X | |
| [41] | X | | | X | | |
| [43] | X | X | | X | | |
| [42] | | X | | X | X | |
| Direct Physical DV | | X | X | X | X | X |

An important contribution of our approach is the realization of a possibility to directly derive Data Vault schema from source RDBMS schemas. The direct approach to physical Data Vault design is now possible thanks to the feature of the Data Vault models i.e. the separation of unchangeable identities of entities in real systems (Hubs) from time variant relationships (Links) and the characteristics of such entities and relationship (Satellites).

There are many approaches to modeling and/or generating ETL process and code for example [45, 46, 47, 48, 49, 50, 51 and 52]. In Muñoz [51] an approach to the automatic code generation of ETL processes is given. The modeling of ETL processes used MDA (Model Driven Architecture) with the formal definition of a set of QVT (Query, View, Transformation) transformation. The problem of defining ETL activities and securing their formal conceptual representation is given in Simitsis and Vassiliadis [48]. The proposed conceptual model provides custom attributes for monitoring and appropriate ETL activities in the early stages of project data warehouse, and flexibility, extensibility and reuse patterns for ETL activities. Jovanovic (et al) [53] presents a tool GEM that from a given set of business requirements and data source descriptions semi-automatically produces multidimensional and ETL conceptual designs. In addition, many commercial ETL design tools exists [54, 55, 56, 57, etc.]. Nevertheless none of the before mentioned approaches and tools consider the ETL process in a Data Vault based data warehouse context that is for a second generation of data warehousing models.

Sources listed in Table 1 above mainly represent academic contributions. Of interest to practitioners and to our research are also several additional sources illustrating the development and/or use of tools to automate the design of data warehouses.

Golfarelli (et al) [58] presented the prototype CASE tool WAND. This tool assists the designer in structuring a data mart, carries out conceptual design in a semi-automatic fashion, allows for a core workload to be defined on the conceptual scheme and carries out logical design to produce the data mart scheme.

QBX is a CASE tool for data mart design resulting from a close collaboration between academy and industry. It supports designers during conceptual design, logical design, and deployment of ROLAP data marts in the form of star/snowflake schemata, and it can also be used by business users to interactively explore project-related knowledge at different levels of abstraction. [59].

BIReady's Model-driven Data Warehouse generation engine [60] is one extension of CA Erwin tool that allegedly uses best practices and a stable architecture so that one can easily manage and evolve data warehouse. BIReady is able to generate a Data Warehouse automatically from a business model. BIReady even generates the ETL and loads data. BIReady builds data architectures from the ground up based on best-practices. For example, the Corporate Information Factory (CIF) pattern by Bill Inmon, the father of Data Warehousing, and Dan Linstedt's Data Vault modelling approach. Even BIReady's datamarts are real Kimball Star-schemas.

The Birst [61] tool automatically compiles a logical dimensional model into a star schema design and then generates and maintains fact and dimension tables. Logical measures are automatically analyzed for calculation grain, while logical dimensions are analyzed for levels requiring persistence. This tool generates and manages all key relationships, including surrogate keys where necessary and provides data connectivity and extract options for a wide variety of databases, both flat and structured files. The Birst also supports scheduled data extraction from all major relational database management systems, including Oracle, DB2, SQLServer, MySQL, Sybase, and PostgreSQL.

Quipu [62] is an open source data warehouse generation system that creates data warehouses and supports Data Vault architectures. Quipu automates the data warehouse data model design and generates the ETL load code to fill the data warehouse from source systems. With Quipu you can simply and quickly generate and implement a source driven Data Vault, often referred to as source or raw Data Vault. Additional business value can be achieved by implementing a business Data Vault, where source data is combined in a Data Vault implementation of a single business model. Quipu supports both Data Vault architectures. Quipu's repository holds all relevant metadata of the source data elements and generates a data warehouse model based on the Data Vault modeling methodology. Quipu provides the functions to build a Data Vault data warehouse quickly and reliably. Quipu fills the gap in the tool sets available today to implement the data warehouse architecture by generating, maintaining and populating (database) structures and code to capture changes in data, both transactional and reference [62]. According to a white paper available on [62], functionality to assist the construction of Data Marts is not yet available in the Version 1.1 release of Quipu.

Pentaho Data Integration (PDI, also called Kettle) [46] is the component of Pentaho responsible for the ETL processes. Though ETL tools are most frequently used in data warehouses environments, PDI can also be used for other purposes: migrating data between applications or databases, exporting data from databases to flat files, parallel loading of data into databases, data cleansing, integrating applications. PDI can be used as a standalone application, or it can be used as part of the larger Pentaho Suite. As an ETL tool, it is the most popular open source tool available Moreover, the transformation capabilities of PDI allow you to manipulate data with very few

limitations [46]. The ETL process is fully automated, but data vault and data mart processes are not part of the framework [63].

Table 2 summarizes information about relevant tools.

**Table 2.** Comparison of data warehouse design tools

| | **Type of DW tool** | **Physical DW model** | **Data Vault design** | **Generate ETL code** | **Data Mart design** |
|---|---|---|---|---|---|
| **WAND** | Academic | Automatic | Not supported | Automatic | Automatic |
| **QBX** | Academic | Automatic | Not supported | Automatic | Automatic |
| **Quipu** | Open source | Automatic | Automatic | Automatic | Manual |
| **Pentaho Kettle** | Open source | Manual | Manual | Automatic | Manual |
| **BIReady** | Commercial | Automatic | Manual | Automatic | Automatic |
| **Birst** | Commercial | Automatic | Not supported | Automatic | Automatic |
| **Direct PDV** | Prototype | Automatic | Automatic | Automatic (in progress) | Automatic (in progress) |

The advantage of our approach is the availability of a graphical view of the physical diagrams for Data Vaults and data marts during the design process. Furthermore, except for WAND and QBX, these tools do not present formalized and publically available information in detail.

## 3. Direct Physical Data Vault Design

The aim of physical design is to assist in implementing a DW on a selected database management platform. In the process of storing data, ones take large amounts of data from variety of sources: ERP (Enterprise Resource Planning) systems, relational databases, Excel files, DBF files, TXT and XML files or data from the Web. The data in databases are called structured as they are defined with the data schemas. Information contained in file systems is considered unstructured and exists in two basic forms: as textual and as non-textual. Existing technology allows for easy loading of textual data in the staging database or data warehouse, which is not the case with non-textual. Some textual data (documents) can be structured and are known as semi-structured.

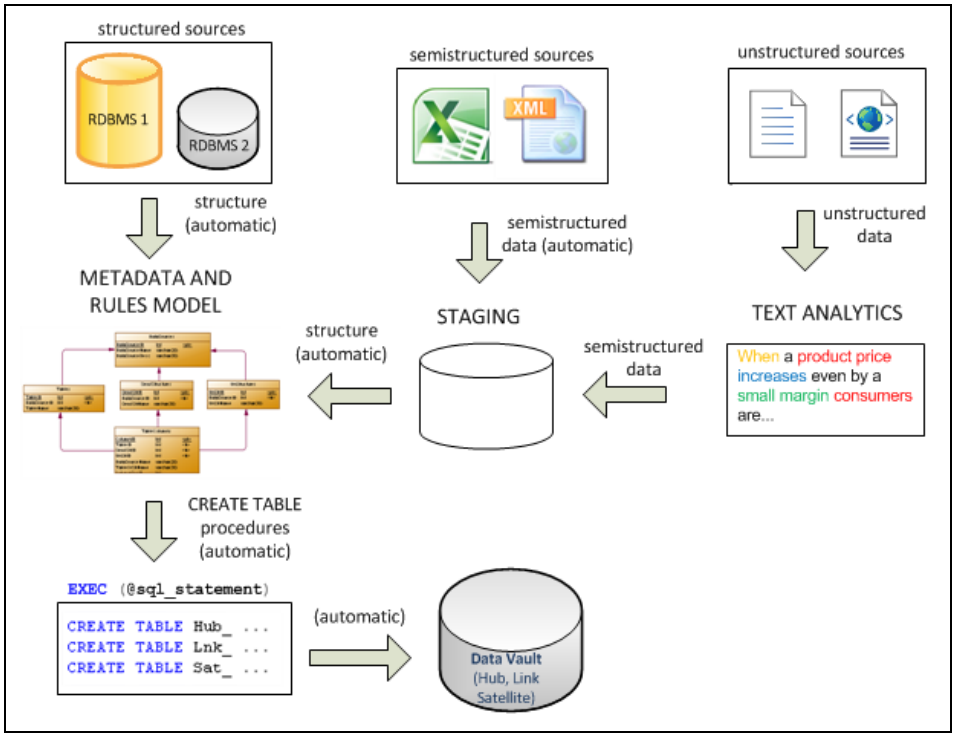


**Fig. 4.** Overview of proposed approach

We propose a direct approach to the design of physical Data Vault data warehouses based on the historical preservation of the original data. An important methodological contribution of our approach is that conceptually, at the metadata level, data sources (DS) of all types are presented with relational model schema equivalents. The relational model is treated as the common factor for Data Vault implementation and all the source data and therefore all types of sources (structured and some semi-structured) are the first to be abstracted via meta-model into relational model equivalents. The term direct approach means that an approach directly leads to Data Vault models based on a physical model of sources in a relational form. The approach is facilitated by the fact that, no matter how many sources of operational data are drawn from, it is the aim of each information system to integrate basic information of wider significance and establish control using only one or few primary database(s) that include most of the data (and these databases, in most cases, reside in relational database engines). On the other hand, theoretical analysis [64] considers relational data models as general models, in which all data can be represented and so reduced. This work covers only the first stage from Figure 5 of the full scope of our research on Data Vault approach automation. The phases of initial Data Vault design automation (for the first stage) are shown in Figure 6.

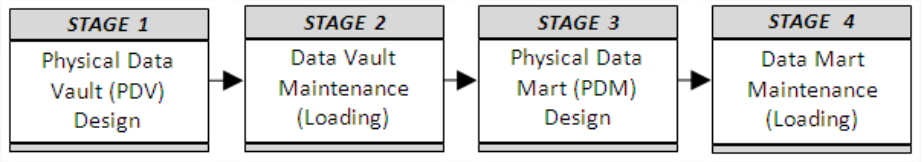


**Fig. 5.** Stages of Data Vault data warehouse design and exploitation

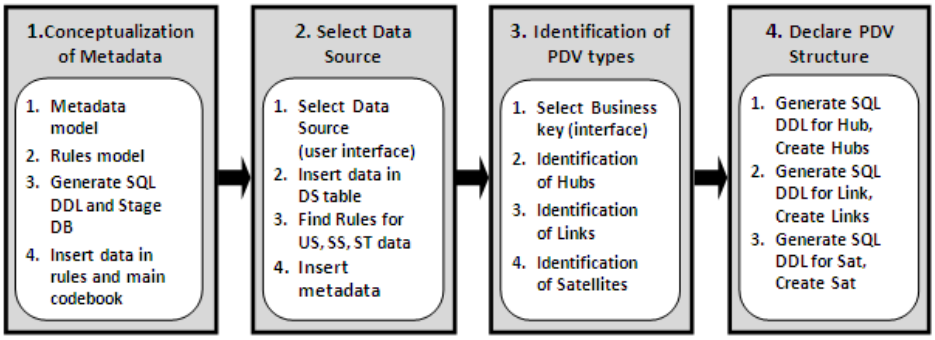


**Fig. 6.** Phases of initial Physical Data Vault design automation

The direct approach to Physical Data Vault (PDV) design automation of a data warehouse presented in this paper includes the use of rules for data warehouse design. To understand the detailed rules we first outline a general algorithm for recognition of Hub, Link and Satellite relations based on mapped original data (equivalent relational model tables). In a nutshell the algorithm consists of the following:

```
I- For each source:
      a) For each Table: the user confirms Hubs by
         selecting a Business Key
      b) For each Table, if not already selected as
         a Hub, create Link
      c) For each Table:
            i. for each FK create  Link
      d) for each non-key attribute: create
         Satellite
II-   For each Hub (PK)
   a) For each source
    i.   For each Table (search for matching PK): if
         found, create Link (to Hub ad II; this only
         illustrate possible integration).
```

The effect of the algorithm is to create a Link for each FK (except for tables whose PK are completely composed of FKs as all such tables will became Links in step I- b- if not confirmed Hubs in step I- a). The proof of completeness of the algorithm follows from the assumption that, no matter the source, every table must have a PK, and the fact that tables are originally valid i.e. have been made from real data so that no

contradictions in terms of circular dependencies on other tables could prevent their formation.

1. Every PK is either composed (partially or fully) from the PKs from other tables or is a separate PK (possibly, but rarely, composed of several fields concatenated or not. See [19]).
2. Tables with separate PKs are designated as Hubs
3. Tables with fully composed PKs are designated as Links.
4. Remove all Links tables whose PK are not used as FK
5. Remove all Hubs tables whose PK are not part of any PK in any remaining table
6. Represent all remaining FK references with separate Links tables (each of those Links will now have two FK references); this is a crucial step separating mutually dependent tables
7. Represent what is left, treating it as a directed graph expressing FK references (as precedence relations) in a matrix form
    a. Nodes are tables (including Links created in step 6),
    b. Branches are FK references whose direction follows the FK reference i.e. branches are pointing to tables whose PK is referenced by each FK. The result is a connected directed graph.
8. We can fully reduce the matrix using Warshall's algorithm following precedence relations (as adjacencies) on a connected directed acyclic graph (in a finite number of steps) where the number of steps determines the length of the longest precedence chain.
9. The key for a proof that the above procedure finishes is that the graph obtained by carving initial, i.e. root, Hubs and Links must be acyclic (proof by contradiction)
    a. If the precedence (FK path) forms a cycle the table has a PK that depends on itself but this contradicts the initial requirement that tables can exists in extension, meaning that records in the tables exist i.e. not only can be formed with complete PK in the time of schema creation but also effectively populated in the time of record

Next figure shows how introduction of Links breaks cycle in ER models.

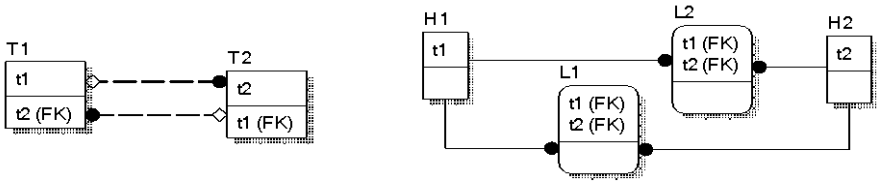


**Fig. 7.** Links for FK relationship

### 3.1. Conceptualization of Metadata – Phase 1

The main phase is the Conceptualization of Metadata (general Physical Data Vault structure) resulting in a physical data model that can be used in designing individual Data Vault data warehouses.

The first two steps of this stage assume building a model of the metadata (Figure 8) and of the modeling rules (Figure 9). When making the meta model and rules we distinguish between part of the model that is independent of the data source (source-independent design) and part of the model, which depends on the source (source-specific design). The independent part refers to the metadata tables and rules, and the dependent part involves procedures related to different data sources. The purpose and a method of how to update the metadata table are shown in Table 3**.**

The RuleTypes table contains information about the types of policies, such as rules for different data sources, rules for identifying Hub, Link and Satellite tables and rules for creating Hub, Link and Satellite tables. The Rules Table contains information about the rules for a particular type of rules and actions to be performed when a certain condition is satisfied. Rules in column Rules allow easier execution of commands that are stored in the column RuleAction. The following example is given to show the SQL (Structured Query Language) rule to describe a set of constraints that are applied to identify candidate Hub based on the value of the column BusinessKey.

```
UPDATE TableColumns SET TableColumns.HubCandidate = 1
WHERE TableColumns.BusinessKey = 1
```

The third step in this phase is generating code to create a staging database and creating tables and stored procedures from the model (Figure 8 and 9). From this presentation of the physical model we can generate a script to create a table in the staging database and insert initial data into tables StructureType and SourceType. The fourth step is to insert data into the rules table and basic codebook (reference data).

After the first stage and creation of the tables and appropriate procedures conditions are created for the development of an application that allows one to automate the design of the physical model of Data Vault data warehouse, with minimal user interaction. Second, third and fourth phase of Figure 6 needs to be done for each specific system.

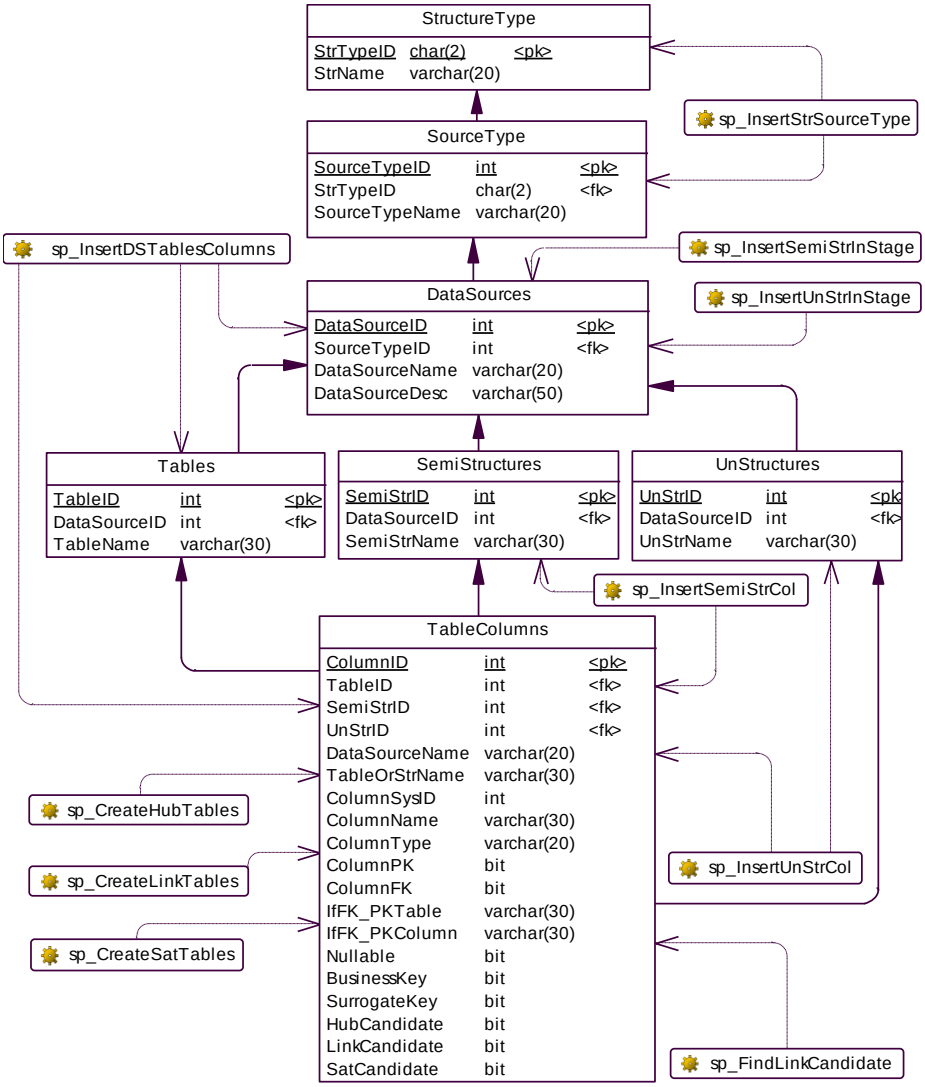


**Fig. 8.** Physical metadata model

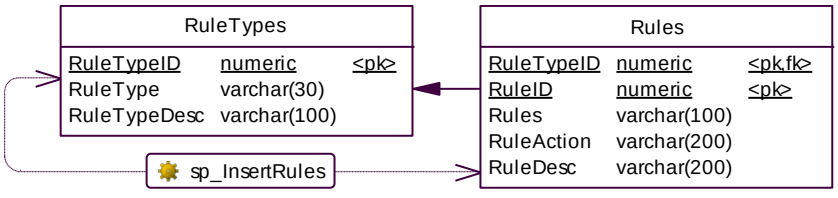


**Fig. 9.** Model of Rules

**Table 3.** Purpose of the generated tables

| Table | Purpose | Insert method |
|---|---|---|
| StructureType | Codebook data: structured (ST), semi-structured (SS) or unstructured (US) | From model, example: INSERT INTO StructureType (StrTypeID, StrName) VALUES ('ST', 'Structured') |
| SourceType | Data sources codebook (example: MS SQL Server, Oracle, xls, XML, txt, etc.) | From model, example: INSERT INTO SourceType (SourceTypeID, StrTypeID, SourceTypeName) VALUES (1, 'ST', 'MS SQL Server') |
| DataSource | Data source that will be used by data warehouse | Automatically after identification all data sources through the user interface |
| Tables | Table from operational data | Automatically after identification all data sources through the user interface |
| SemiStructures | Semi-structured data source (xls, XML, etc.) | Semi-automatic, after importing and structuring (if possible) in the staging database |
| UnStructures | Unstructured data source (txt, etc.) | Semi-automatic, after textual analysis and importing and structuring in the staging DB |
| TableColumns | Columns of data tables from databases and other sources that can by structured | Automatically, except BusinessKey fields that will be selected through the user interface |
| RuleTypes | Types of rules for creating a data warehouse | From model |
| Rules | Contain rules and appropriate action if condition is satisfied | From model |

### 3.2. Identification of Data Sources – Phase 2

This phase is applied to each individual DW (further development of CASE tools will take into account experiences with the prototype). The first step involves the identification of specific applications and systems (databases, structured and unstructured data sources) for data warehouse by a user selecting specific data sources (through the user interface). The second step in this phase is to insert data into table data sources based on the selected data source.

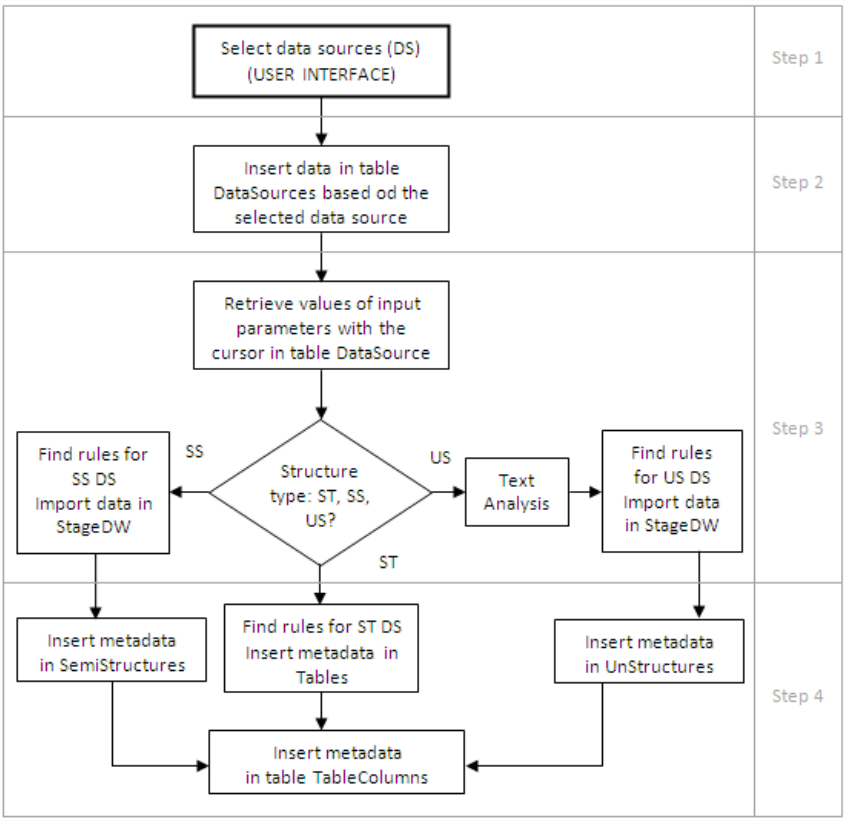


**Fig. 10.** Flowchart of the second phase

The third step defines rules for inserting metadata for structured, semi-structured and unstructured data sources into a staging database. The rules defined in the model are stored in a table Rules. For identification of the rules, it is necessary to find data type (ST, SS, US) from all sources of data based on input parameters. These parameters will be obtained through the mechanism of the cursor through tables from data sources. The cursor is introduced as a set of records which is attached to the pointer at current row. Commands in SQL statements include moving the cursor to work with the current row. If it is semi-structured data sources, in the absence of a clear structure, their operational data need to be imported into the staging database. Semi-structured data such as Excel files can be imported into a database using ETL processes [13].

Import metadata in the staging database will be prepared using a stored procedure as indicated in the model *sp_InsertSemiStrInStage* (with appropriate parameter). In case of unstructured data sources, it is necessary to first identify their metadata. Unstructured data can be placed into the database in the traditional way by using the files metadata, the path to the file or URL, and attributes and links between files placed in the database. The newer method of storing unstructured data in the database is known as text analytics. It is the process of converting unstructured documents into

structured documents by analysis of the structure of the text in the document [65]. Text analytics is the process of enabling a computer to extract meaning from text and is performed as a series of iterative process prior to loading into the database [66]. Some of the procedures are simple editing, stop-word removal, replacement of synonym or concatenation, homographic resolution, thematic clustering, external glossary/taxonomy overlay, stemming, alternate spelling resolution, foreign language accommodation, search direct and indirect support. A description of each of the procedures is given in [67].

The fourth step in this phase is to fill the tables Tables, SemiStructures, UnStructures TableColumns and information about the tables and their columns on transactional databases. This step will be automatically generated from the procedure for importing data by using data from the target system schema repository containing the tables, columns, indexes, constraints and relationships for each specific system for database management. Inserting metadata into the table TableColumns looks more complex due to the large number of metadata and a number of columns. A similar approach is taken with other structured data sources that import metadata into tables and TableColumns. Tables will be prepared using a stored procedure as indicated in the model *sp_InsertDSTablesColumns*. Based on data sources, this procedure will be transmitted as a parameter for a specific structured data source (MS SQL Server, Oracle, IBM DB2 and others), on the basis of the rules in the table of rules.

In a nutshell, the algorithm for this phase consists of the following:

1. Loop over data sources (incrementally or in sets of DBs as sources): Select data source and Insert metadata in table *DataSources*
2. For each row in table *DataSources*: find structure type, source type and rules
   a) For each structured type:
      *i.* Insert metadata in *Tables*
      ii. For each row in table *Tables:* insert metadata in table *TableColumns*
   b) For each semistructured type:
      i. Extract and load data in database StageDW
      ii. Insert metadata in table SemiStructures
      iii. For each row in table *SemiStructures:* insert metadata in *TableColumns*
   c) For each unstructured type:
      i. Text analytics process
      ii. Extract and load data in database StageDW
      iii. Insert metadata in table UnStructures
      iv. For each row in table *UnStructures*: insert metadata in *TableColumns*

Identification of PDV types (Hubs, Links, Satellites) for each DW is initially reduced to the identification of a business key until the Hub, Link and Satellite are automatically generated. On the model in Figure 8, the meaning of the columns in the tables StructureType, SourceType, DataSource, Tables, SemiStructures, UnStructures

clear,only the structure and purpose of the specific TableColumns column (Table 4) will be explained.

This table should contain metadata from databases, semi-structured metadata and unstructured data sources that can be appropriately structured. Our approach provides the automatic loading of metadata from all TableColumns structured data sources and partly from some unstructured and semi-structured data sources. In addition to the data that will be filled on the basis of a database schema, this table will contain a column indicator that will give us information as to whether it is a business key, surrogate key, or if the table Hub, Link or Satellite candidate.

**Table 4.** TableColumns Table structure

| Column | Type | Description | Load automatic | Load through UI |
|---|---|---|---|---|
| ColumnID | int | Primary key | ✓ | |
| TableID | int | Table ID | ✓ | |
| SemiStrID | int | ID from semistructured data source | ✓ | |
| UnStrID | int | ID from unstructured data source | ✓ | |
| DataSourceName | varchar | Data source name | ✓ | |
| TableOrStrName | varchar | Table name or semi/un struct. file name | ✓ | |
| ColumnSysId | integer | Sys column ID | ✓ | |
| ColumnName | varchar | Column name | ✓ | |
| ColumnType | varchar | Data type | ✓ | |
| ColumnPK | bit | PK column? | ✓ | |
| ColumnFK | bit | FK column? | ✓ | |
| Nullable | bit | Nullable column? | ✓ | |
| IfFK_PKTable | varchar | Table name on PK side (if column FK) | ✓ | |
| IfFK_PKColumn | varchar | Column name on PK side (if column FK) | ✓ | |
| BusinessKey | bit | Business key column? | | ✓ |
| SurrogateKey | bit | Is column Surrogat for corresponding Business key? | ✓ | |
| HubCandidate | bit | Hub candidate? | ✓ | |
| LinkCandidate | bit | Link candidate? | ✓ | |
| SatCandidate | bit | Satellite candidate? | ✓ | |

This phase involves the following steps:

*1. Identification of business key through a user interface.* Through the appropriate user interface, based on the data, the user should check the business keys. Based on the business key, the column with the appropriate BusinessKey value (0 or 1) will be filled. Filling in this section can be automatically based on rules stored in the table Rules,

according to [15], [16], [24], [44], and based on BusinessKey the column SurrogateKey will be filled.

*2. Identification of hubs.* The Hub entity table contains a single list of business keys. These are the keys that organizations use in their daily operations, such as customer number, code of the employee, account number, and so on [15]. According to [15], [16], [24], Hub candidates can be identified using the filled BusinessKey and SurrogateKey on the basis of rules for identifying hubs which are found in the tables RuleTypes and Rules.

*3. Identification of links.* Link is the physical representation of references, foreign keys and many-to-many relationships in third normal form [15]. Links can be identified using the procedure *sp_FindLinkCandidate* (which is an integral part of the model and that is called from the table Rules and RuleTypes) that includes the following steps:

a) Find many-to-many table

b) Set to true LinkCandidate field in many-to-many table

c) Set 1 value in the column LinkCandidate for tables that have a foreign key

*4. Identification of satellites. The* Satellite entity contains context data of hub and contains attributes that are not primary or foreign keys [15]. Satellites are identified on the rules in tables RuleTypes and Rules, according to [15], [16], [24], [46].

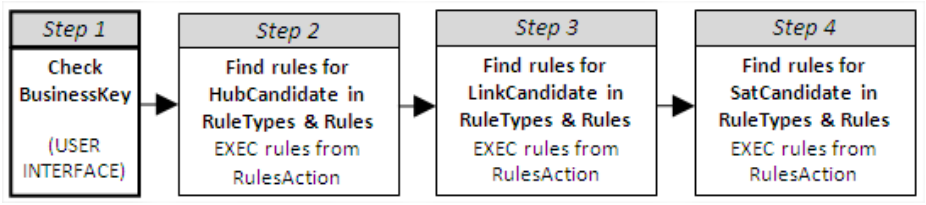


**Fig. 11.** Flowchart of the third phase

### 3.4. Initial Declaration of PDV Structure- Phase 4

The last phase in the process of automation of the physical design of a data warehouse is the initial declaration of the PDV structure. This phase include following steps:

*1. Generate and execute the script to create a hub table.* When we have the business key, appropriate tables can be identified (by setting the value 1 in column HubCandidate). It is possible to generate a script to create a hub table in a Data Vault, based on the rules in the Rules table. This step provides the following:

a) Forming a cursor to go through the TableColumns Where HubCandidate=1

b) Retrieve data from a table and assign variables

c) In each iteration, use dynamic SQL to supplement sql_statement

d) Perform an sql_statement, creating a hub table

*2. Generate and execute the script to create the link table.* The link table is used to represent relationships or transactions between two or more business components (two or more hubs). We identified the appropriate link tables containing value 1 in column

LinkCandidate which enables the generation script to automatically generate a link table, based on the rules in the Rules table. It is possible to generate a script to create a link table in a data warehouse, based on the rules in Table Rules. This step involves:

a) Forming a cursor to go through the table TableColumns Where LinkCandidate=1
b) Retrieve data from a table and assign variable
c) In each iteration, use dynamic SQL to supplement sql_statement
d) Performing an sql_statement, creating a link table

*3. Generate and execute script to create the satellite table.* Satellite entity shows how context hub data. Satellite table created for each hub table will contain non-key attributes in a transactional database. We identified the appropriate Satellite tables by setting the value 1 in column SatCandidate which enables the generation script to automatically generate the satellite table based on the rules in the Rules table. In this case, one table is generated for each hub table (or link table, if the transaction has only a link table). This step involves:

a) Forming a cursor to go through the table TableColumns Where SatCandidate=1
c) Retrieving data from a table and assign variable
d) In each iteration, use dynamic SQL to supplement sql_statement
e) Performing an sql_statement, thus creating satellite tables

If the source of simple unstructured data, (Excel or TXT file), the data on clients who are not in a transactional database (or other contact information, information about market position, business data, etc.), then an additional Satellite table will be created that will include data from Excel or TXT file.

Procedures described in the fourth phase make it possible to automatically create a complete data warehouse based on Data Vault concepts. Among meta-requirements for any design approach and its automation at least the following four are obvious: performance, scalability, flexibility and agility. In order to fully appreciate potential of our PDV approach one have to first realize what DV as such (comparing to traditional alternatives namely normalized EDW and dimensional Data Marts) contributes to satisfying such meta-requirements. The DV separation of Identities and Links from attributes (Satellites) by design creates a scale-free network [17] and thus greatly reduces stress of incremental expansion (scalability) this supports expansion of scope. The structural changes are also additions (no deletes) so flexibility of designs is assured. The DV by design foster higher levels of performance by (decoupling dependencies and) allowing parallel data loads all the time from all source systems. By requiring input without any irreversible data alterations (ELT as opposed to ETL) from a data source into a raw data vault (as a permanent fully auditable system of records) Linstedt suggest typical loading speed of 100K rows per minute is normal (as a benchmark). Nature of the DV model preserves scalability and flexibility as well as performance of the data vault designs weather manual or automated.

## 4. Physical Data Vault Design Tool

The goal of this section is to present the prototype CASE tool the authors have implemented to support their methodology. The PDV (Physical Data Vault) design tool

assists the designer in structuring a data vault, carries out physical design in a semi-automatic fashion, and allows for a core workload to be defined on the physical scheme.

The tool was developed using Larman's [68] method. Specifications of requirements can be presented as verbal descriptions of the model and use cases. Verbal Description: Need to make an application that will provide support to the process of designing a data warehouse. The data warehouse should provide an analysis of data from structured data sources and simple unstructured and semi-structured (xls, txt). Structured data should not be loaded in a separate staging database, but directly loaded into the data warehouse. Semi-structured data has to be further structured in the staging database and then loaded into the data warehouse. Unstructured data has to go through the process of textual analysis, and then loaded into staging database for some structuring, and then loaded into the data warehouse. The Figure 12 shows the observed use-cases for the PDV tool.

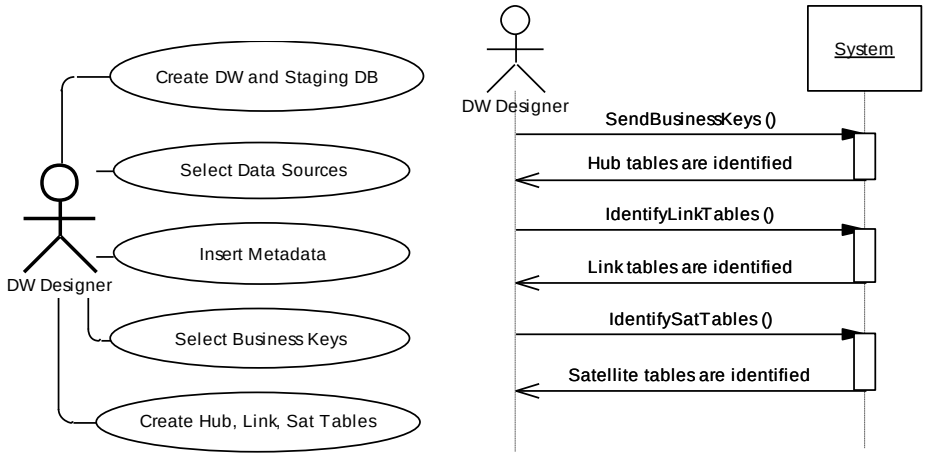


**Fig. 12.** Use-case diagram and Sequence diagram for the use-case select Business Keys

In the analysis phase, the behavior of software systems is determined by using system sequence diagram. For each use-case and for each scenario, a sequence diagram was created. The example in Figure 12 shows the sequence diagram for the use case Select Business Keys. Sequence diagram for use-case Identification of business keys, baseline scenario:

1. Designer sends business keys to system
2. The system returns the information based on business keys and identified Hub tables
3. Designer call the system to identify the Link tables
4. The system returns the information of identified Link table
5. Designer calls the system to identify the Sat tables
6. The system returns the information of identified Satellite tables

The architectural design includes the design of application logic, the user interface and the internal metadata model database. The tool is built in three layer architecture with the database layer, the user interface and business logic layer. The staging

database was designed on the basis of the physical model shown in Figures 8 and 9. We used the Database management system Microsoft SQL Server 2008 R2. Designing the user interface included designing Windows forms. Some examples of the forms are given in the next section. The tool is implemented using Microsoft Visual Studio.NET environment using the C# programming language. The Microsoft's NET Framework was selected as it has a consistent programming model for building diverse applications [69].

## 5. Experimental verification of research results based on a prototype application

Experimental verification of research results was done in the area of health insurance using a prototype tool PDV for a data vault data warehouse design.

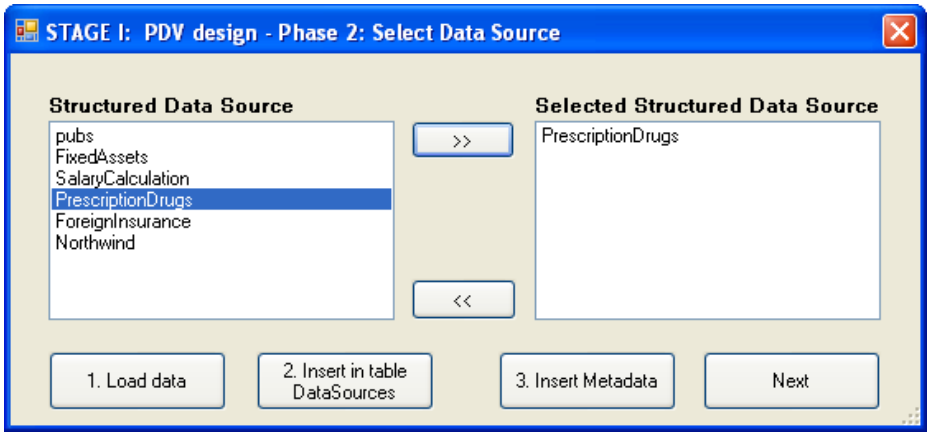


**Fig. 13.** Form for selection of structured data sources

Health Insurance Fund annually enters into contracts with pharmacies to issue prescription drugs to insured patients. Prescription drugs are prescribed in health institution (family doctor's office or hospital). Every fifteen or thirty days (depending on the contract), the pharmacy sends to Fund an invoice for drugs issued on prescriptions. Invoice consists of a header and items. The header contains the name of pharmacy, the date of the invoice, the total amount and the number of receipts. Items invoices contain information about drug, quantity and amount, and patient's information. Invoices are to be submitted electronically. Health insurance Fund distributes databases in offices in each of several regions. The number of records in the table invoice items annually reaches several millions. To reduce the load on the transactional system, view reports (which often changes the format and appearance), to meet the requirements of users (in terms of reports with different grouping and diagrams), it was decided to implement a data warehouse and business intelligence system. When choosing a data warehouse architecture, the Data Vault approach was chosen (emphasizing the need to leave a trail from where and when the information

originated from the databases). Moreover, a Data Vault is designed to model data that can be easily changed following rapid changes in the business environment.

After the first stage (initialized data warehouse and staging databases) associated procedures based on the model are derived from the second stage, the selection of data sources.

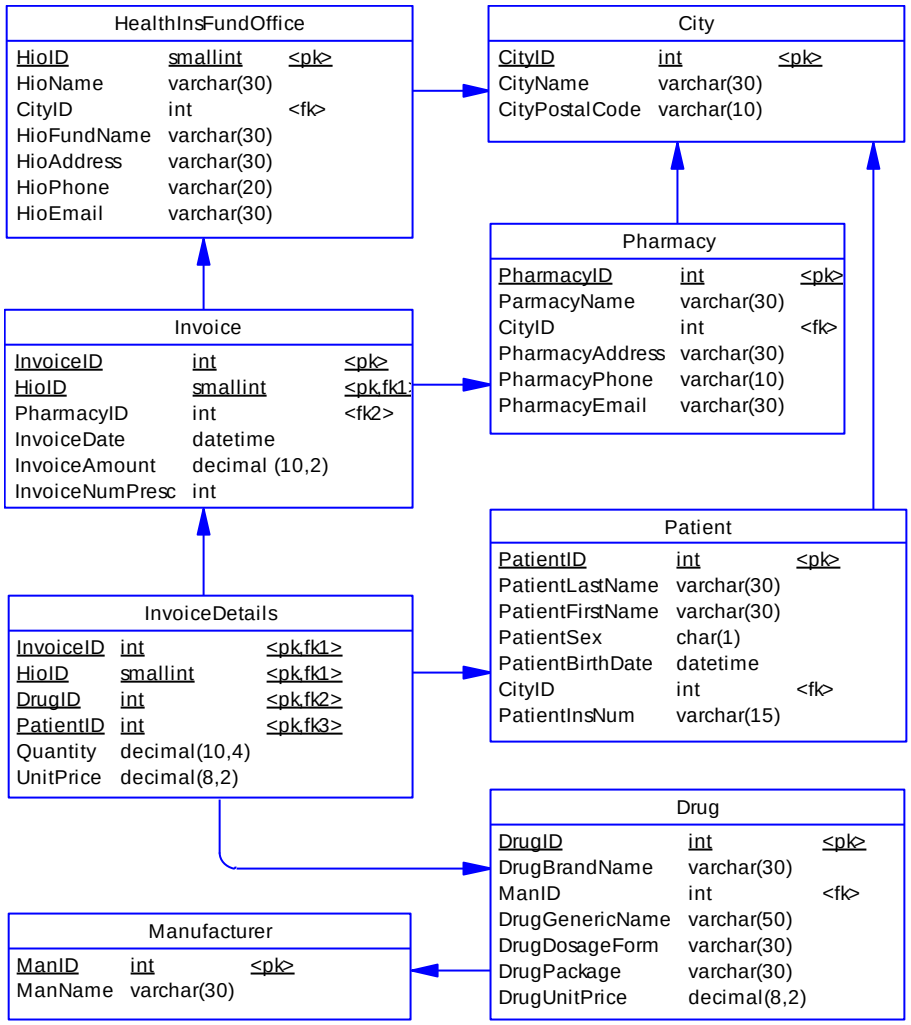


**Fig. 14.** Physical model of transactional database PrescriptionDrugs

After the selection of data sources, the user starts uploading the metadata of selected data sources into the staging database. Part of the physical model of transactional database PrescriptionDrugs is given in figure 14.

In the following form (Fig.15), users simultaneously (on three DataGridView controls) see selected data sources, data source tables and columns. The third DataGridView enables the checking of business keys. On the basis of the business keys

and corresponding rules for Hub tables, the system identifies a Hub table. After that, based on the hub tables and rules, the system identifies link tables. At the end of this phase, system identifies Satellite tables. After clicking the Next button, a form opens to declare a PDV structure as shown in the figure 16.

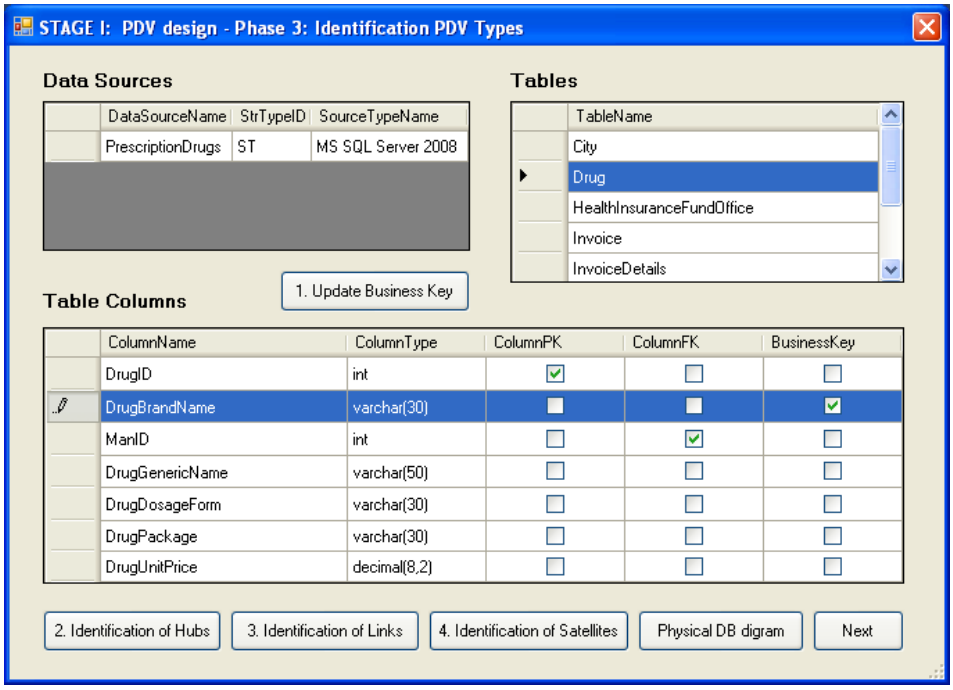


**Fig. 15.** Form for identification PDV types

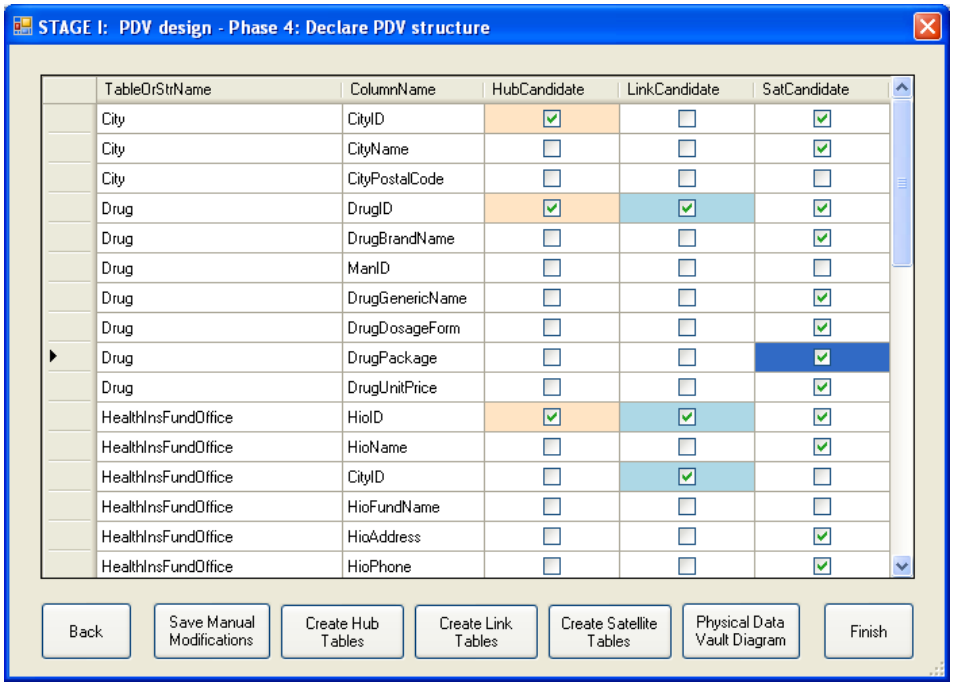


**Fig. 16.** Form for declare PDV structure

This form allows for manual modification of the proposed Hub, Link and Satellite table. After manual modifications, the designer starts the process of creating Hub, Link and Satellite tables that are just being identified. This step completes the process of creating a data warehouse based on the Data Vault concept. The next button is clicked to enable visualization of Data Vault data warehouse structure. The created Data Vault Physical model is shown in figure 17. According [15], in the tables can be generate a surrogate key - optional component, possibly smart key or sequential number, if the composite primary key might cause performance problems.

In our trials and experiments with the use of the PDV design tool (as it was evolving as a prototype itself) it was easy to create a complete data warehouses based on Data Vault concepts. In addition, the tool excelled when used to develop prototypes of data warehouses. In fact, only when a tangible DW prototype was completed, users become more interested in participating and frequently stated new requirements. Automating design for a data warehouse significantly accelerated the development of a robust system by allowing prototyping in the early stages of contact with customers, and customers were more interested in providing information.

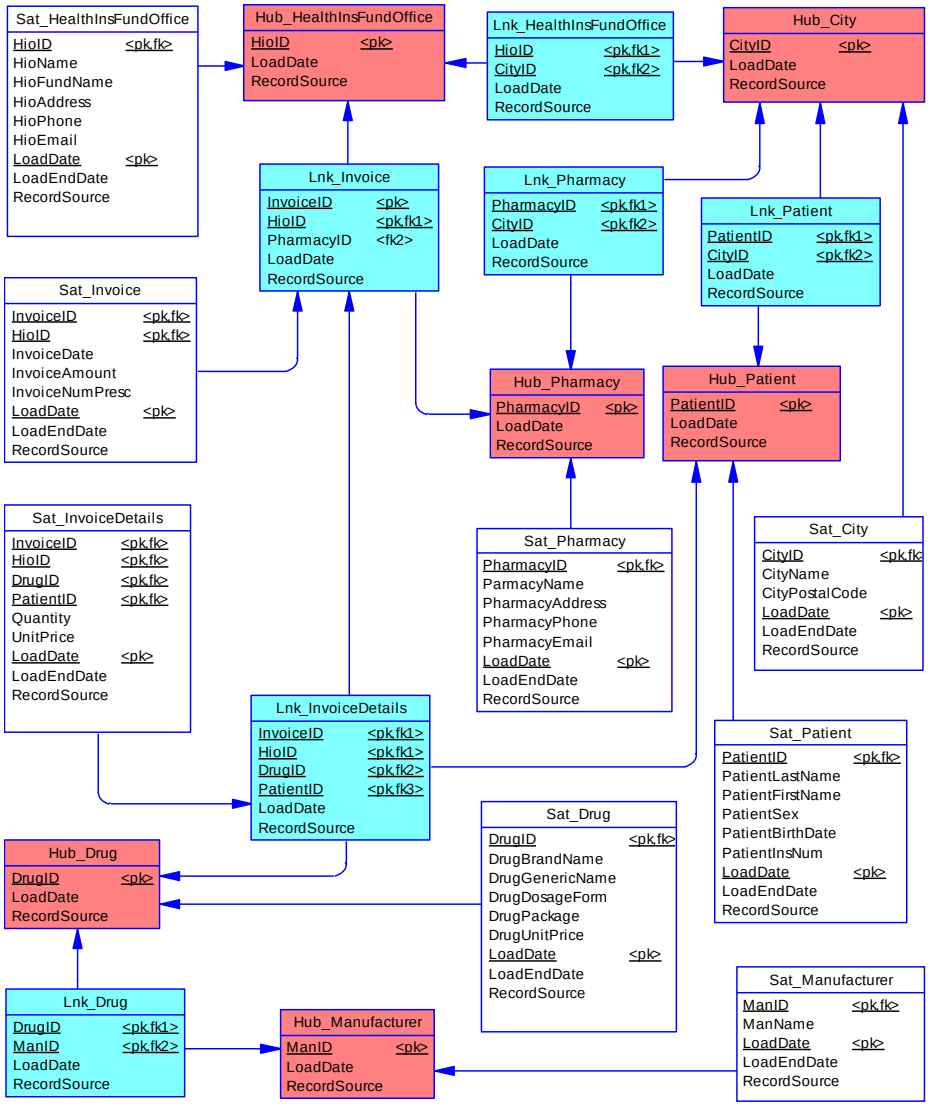


**Fig. 17.** Model of Physical Data Vault data warehouse

## 6. Conclusion

This paper presents the basic algorithm for the initial physical design stage of the Data Vault types of enterprise data warehouses i.e. integrated data warehouses as systems of records not open to end user reporting. The approach is based on the incremental expansion of data warehouse adding new data sources in sets or one at a time. The algorithm utilizes metadata model and rules for the design starting with existing

(mainly) transactional data sources. Relations between entities in transactional systems and rules for the development of a data warehouse based on Data Vault concept are crucial for physical design automation of a data warehouse model. The conceptualization of metadata presented a physical model that can be used in the design of individual data warehouses, and this became the basis for development of a tool. The most important contribution of this paper is realization of Data Vault schema directly from RDBMS schemas. Such a direct approach was possible thanks to the feature of the Data Vault models i.e. separation of unchangeable identities of entities in real systems (Hubs) from time variant relationships among such entities (represented by Links) and the characteristics of such entities and their relationships (represented by Satellites). Traditional approaches to integration used pruning of data from the source and other forms of derivation i.e. consolidation that requires much intervention by experts (due to creative and semantically rich transformations). Data Vaults provide the unique ability to integrate data incrementally by adding links (of 'same as' type essentially 1:1 mappings) between initial and added hubs, while preserving all data in satellites, links and hubs without any reconstructions and deletes (guaranteeing preservation of information necessary for an enterprise size system of records). The subject of ongoing research is detailed specifications of dynamic expansion of Hub, Link (and their Satellite tables) and their additional linking for possible cases of merging Data Vault schemes in operation. Within the achieved scope, work in progress is focused on code generation for the initial loading of the created Data Vault enterprise data warehouses, as well as code generation for the Data Vault updating with new values and/or updates of the code to update (all without slowing down the original system).

The PDV approach is based on available relational schema and this satisfies meta-requirements stated earlier. Loading a schema and transforming it following preprogrammed rules certainly supports design performance, scalability and agility (user intervention is minimal but necessary, and is mainly focused on recognizing major permanent business keys). We claim that any indirect DV design driven by a conceptual or a logical data model (derived from the existing data sources) even when supported by some automation, is less flexible than direct PDV. Furthermore, it increases a danger of losing data from the source, potentially invalidating the central DV EDW purpose - to maintain a system of unaltered records.

Future work relate to the remaining three stages of Figure 5. First is the automation of the ETL process from data sources to feed a Data Vault. The Data Vault type data warehouse is a solution for integrating, storing and protecting data. However, it is not intended, nor suitable, for intensive queries or reports. That is why the data warehouse architecture with a Data Vault layer (persistent system of records with full history of changes) also contains a data mart layer using the star schemas for reporting data access [46]. According to [70] the dimensions of the star schema result from the Hub and Satellite tables, and the fact tables from a Satellite and Link tables. The next item of research is addressing the output area (data marts) with the following steps: Create a model of DMs and DMs materialization code and create metadata for Analytics Tracking (Dashboards/Scorecards) and standard reporting.

The process of designing an enterprise data warehouse based on the Data Vault model can be formalized, generalized and to some extent based on the automated

physical model for structured, semi-structured and simple unstructured data sources, including transactional database. Our direct approach integrates elements of the physical design of enterprise data warehouses based on a data vault model as a system of records. This paper also illustrated the development of a tool for automation of design for data vault based enterprise data warehouses. The tool has been implemented and used on a real case in the field of healthcare and medical insurance and provided satisfactory results.

The paper and the approach presented so far do not address design of data marts, data virtualization, data warehouse schema evolution, master data management, mappings to NoSql data stores or hybrid databases, nor fully elaborates on ELT/ETL transformation automatization as those issues are part of a lager ongoing research program.

## References

1. Inmon H. W.: Building the Data Warehouse. Wiley Computer Publishing. (1992)
2. Jarke M., Lenzerini M., Vassiliou Y, Vassiliadis P.: Fundamentals of Data Warehouses. Springer-Verlag. (2003)
3. Ćamilović D., Bečejski-Vujaklija D., Gospić N.: A Call Detail Records Data Mart: Data Modelling and OLAP Analysis. Computer Science and Information Systems, Issue: 12, page 87-110. (2009).
4. Krneta D., Radosav D., Radulovic B.: Realization business intelligence in commerce using Microsoft Business Intelligence. 6th International Symposium on Intelligent Systems and Informatics (SISY), pp. 1–6. (2008)
5. Larson B.: Delivering Business Intelligence with MSSQL Server 2008. Mc Graw Hill (2009)
6. Vassiliadis P., Simitsis A., Skiadopoulos S.: Conceptual Modeling for ETL Processes. In Proc. 5th International Workshop on Data Warehousing and OLAP, VA, USA. (2002)
7. Vassiliadis P., Simitsis A, Baikousi E.: A Taxonomy of ETL Activities. InProc. ACM 12th International Workshop on Data Warehousing and OLAP (DOLAP 2009), Hong Kong
8. Adelman S., Moss L., Abai M.: Data Strategy, Addison Wesley, New York. (2005)
9. Inmon W.H.: Building the Data Warehouse: Gettiing Started, White Paper BillInmon.com. (2000)
10. Inmon H.W., Strauss D., Neushloss G.: DW 2.0 The Arcitecture for the Next Generation of Data Warehousing, Morgan Kaufman. (2008)
11. Kimball R.: The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses, Willey. (1996)
12. Kimball R., Ross M.: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd Edition, Wiley. (2002)
13. Mundy J., Thornthwaite W., Kimball R.: The Microsoft Data Warehouse Toolkit, Second edition, Wiley. (2008)
14. Rönnbäck L., Hultgren H.: Comparing Anchor Modeling with Data Vault Modeling, Modeling for the modern Data Warehouse, White paper. (2013)
15. http://www.tdan.com/view-articles, Data Vault Series 1-5 by Dan E. Linstedt. (2002)
16. Linstedt D.: Data Vault Modeling & Methodology, http://www.learndatavault.com. (2011)
17. Linstedt D.: Super Charge your Data Warehouse, Kindle Edition. (2010)
18. Jovanović V., Bojičić I.: Conceptual Data Vault Model, Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA. (March, 2012)

19. Hultgreen H.: Modeling the Agile Data Warehouse with Data Vault. New Hamilton. (2012)
20. Rönnbäck L.: Anchor Modeling – A Technique for Information Under Evolution, GSE Nordic Conference, Stockholm, Sweden. (2001)
21. Regardt, O., Rönnbäck, L., Bergholtz, M., Johannesson, P., Wohed, P.: Analysis of Normal Forms for Anchor Tables. (2010)
22. Knowles C.: 6NF Conceptual Models and Data Warehouses 2.0. Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA. (March 2012)
23. Date, C.J., Darwen, H., Lorentzos, N.: Temporal data and the relational model: A detailed investigation into the application of interval and relation theory to the problem of temporal database management. Morgan Kaufmann Publishers. (Amsterdam, 2002)
24. Graziano K.: Introduction to Data Vault Modeling. True BridgeResources, White paper. (2011)
25. Jovanović V., Bojičić I., Knowles C., Pavlić M.: Persistent staging area models for Data Warehouses. Issues in Information Systems, Volume 13, Issue 1, pp. 121-132. (2012)
26. Golfarelli M., Rizzi S.: Data Warehouse Design Modern Principles and Methodologies. McGraw-Hill. (2009)
27. Corr L., Stagnitto J.: Agile Data Warehouse Design. DecisionOne Press, Leeds. (2011)
28. Winter, R., Strauch, B.: A Method for Demand-Driven Information Requirements Analysis in DW Projects. In Proceedings of 36th Annual Hawaii International Conference on System Sciences. (2003)
29. Jensen, M., Holmgren, T., Pedersen T.: Discovering Multidimensional Structure in Relational Data. In Proceedings of DaWaK. (2004)
30. Guo Y., Tang S., Tong Y., Yang D.: Triple-Driven Data Modeling Methodology in Data Warehousing A Case Study. DOLAP '06 Proceedings of the 9th ACM international workshop on Data warehousing and OLAP. (2006)
31. Golfarelli M.: Data warehouse life-cycle and design. White Paper, DEIS – University of Bologna. (2008)
32. Boehnlein, M., Ulbrich v.E, A.: Business Process Oriented Development of Data Warehouse Structures. Proceedings of Data Warehousing 2000, Physica Verlag. (2000)
33. Westerman P.: Data Warehousing. Morgan Kaufman Publishers. (2001)
34. Romero O., Abello A.: A Survey of Multidimensional Modeling Methodologies. International Journal of Data Warehousing & Mining, 5(2), 1-23. (April-June 2009)
35. Jovanovic V., Marjanovic Z.: DW21 Research Program-Expectations, FON/Breza software engineering, White paper. (February 2013)
36. Golfarelli M., Mario D., Rizzi S.: Conceptual Design of Data Warehouses from E/R Schemes. System Sciences. (1998)
37. Boehnlein M., Ulbrich-vom Ende A.: Deriving initial data warehouse structures from the conceptual data models of the underlying operational information systems. 2nd Int. Workshop on Data Warehousing and OLAP. (1999)
38. Phipps C., Davis K. C.: Automating Data Warehouse Conceptual Schema Design and Evaluation. 4th International Workshop on Design and Management of DW. (2002)
39. Peralta V., Marotta A., Ruggia R.: Towards the Automation of Data Warehouse Design. 15th Conference on Advanced Information Systems Engineering, Velden, Austria. (2003)
40. Romero O., Abello A.: Automating Multidimensional Design from Ontologies, DOLAP'07. (November 2007)
41. Zepeda L., Ceceña E., Quintero R., Zatarain R., Vega L. , Mora Z., Clemente G.: A MDA Tool for Data Warehouse. International Conference on Computational Science and Its Applications. (2010)

42. Nazri M.N.M., Noah S.A., Hamid Z.: Using lexical ontology forsemi-automatic logical data warehouse design. 5th international conference on Rough set and knowledge technology. (2010)
43. Zekri M., Marsit I., Adellatif A.: A new data warehouse approach using graph. Eight IEEE International Conference on e-Business Engineering. (2011)
44. Damhof R.: The next generation EDW. Database Magazine (Netherlands). (2008)
45. Hammergren T. , Simon A.: Data Warehousing for Dummies. 2nd edition. (2009)
46. Casters M., Bouman R., van Dongen J.: Pentaho Kettle Solutions: Building Open Source ETL Solutions with Pentaho Data Integration. Wiley Publishing. (2010)
47. Luján-Mora, S., Vassiliadis, P., Trujillo, J.: Data Mapping Diagrams for Data Warehouse Design with UML. (2004)
48. Vassiliadis, P., Simitsis, A., Skiadopoulos, S.: Conceptual Modeling for ETL Processes. In DOLAP. 2002.
49. Kimball, R., Caserta, J.: The Data Warehouse ETL Toolkit Wiley Publishing, Inc. (2004)
50. Simitsis A., Vassiliadis P., A method for the mapping of conceptual designs to logical blueprints for ETL processes. Decision Support Systems, vol. 45, no. 1. pp. 22–40. (2008)
51. Muñoz L., Mazón J., Trujillo J.: Systematic review and comparison of modeling ETL processes in data warehouse, Inf. Syst. Technol. (CISTI), 5th Iberian Conference. (2010)
52. Oliveira B., Santos V., Belo O.: Pattern-Based ETL Conceptual Modelling. International Conference on Model & Data Engineering, Calabria, Italy. (2013)
53. Jovanovic P., Romero O., Simitsis A., Abelló A., Requirement-driven creation and deployment of multidimensional and ETL designs, Advanced Concept. Model. (2012)
54. http://technet.microsoft.com/en-us/library/ms141026.aspx
55. http://www.oracle.com/us/products/middleware/data-integration/enterprise-edition/overview/index.html
56. http://www-03.ibm.com/software/products/en/ibminfodata
57. http://www.informatica.com/us/products/data-integration/enterprise/powercenter/
58. Golfarelli M., Rizzi S., Saltarelli E.: WAND: A CASE Tool for Workload-Based Design of a Data Mart. SEBD. 2002.
59. Battaglia A., Golfarelli M., Rizzi S., QBX: A CASE tool for Data Mart design, O. De Troyer et al. (Eds.): ER 2011 Workshops, LNCS 6999, pp. 358–363, Springer-Verlag Berlin Heidelberg. (2011)
60. http://biready.com
61. http://www.birst.com/product/technology/data-warehouse-automation
62. http://www.datawarehousemanagement.org/Quipu.aspx
63. http://www.bi-podium.nl/mediaFiles/upload/DWHgen/Pentaho_en_DV_-_KdG.pdf
64. Date, C, Darwen, H., Databases, types and the relational model: the third manifesto (3rd ed.). Reading, MA: Addison-Wesley. (2006)
65. Rainardi V.: Building The Data Warehouse With Examples in SQL Server, Springer, New York. (2008)
66. Nicola M., Sommerlandt M., Zeidenstein K.: From text analytics to data warehousing. http://www.ibm.com/developerworks/data/library/techarticle/dm-0804nicola/ (2008)
67. Inmon W.H., Krishnan K.: Building the Unstructured Data Warehouse, Technic Publications LLC. (2012)
68. Larman Craig: Applying UML and Patterns: An introduction to object-oriented analysis and design, Second edition, Prentice Hall, New Jersey. (1998)
69. http://www.microsoft.com/net
70. Govindarajan S.: Data Vault Modeling The Next Generation DW Approach. http://www.globytes.com. (2010)

**Dragoljub Krneta** received the M.Sc. from Technical Faculty "Mihajlo Pupin" University of Novi Sad. Currently he is a PhD student in Faculty of Organizational Sciences, University of Belgrade, Serbia. He lives in Banja Luka, Bosnia and Herzegovina, where he works as a Software Development Manager in a company Lanaco Information technologies. Throughout his career in IT that lasts more than twenty years, he was actively involved in the design and development of information systems as a programmer, systems analyst, database designer, system architect or project manager. He has published 12 papers on international scientific conferences and journals. His research interests include information systems development, databases, data warehouses and business intelligence systems. He is a member of IEEE.

**Vladan Jovanović,** born in Belgrade Serbia, received all his degrees from the University of Belgrade: a) B. Eng. in Cybernetics Information Systems in 1975, b) M. Sci. in Computer Information Systems in 1978, and c) Ph.D. in Organizational Sciences-Software Engineering in 1982. Since 2001 works as a professor of Computer Sciences at the Georgia Southern University's College of Engineering and IT. Research interests: models, methodology and/or processes of data and software intensive systems design. Professional contributions: a) standardization within the IEEE Systems and Software Engineering Standardization areas of software architecture and design, SQA, and SW Processes, and b) curriculum development in IT, IS, CS, Systems Assurance and Software Engineering, as well as participating in CS and SE program accreditation, as an ABET Program Evaluator.

**Zoran Marjanović** is a full professor at Faculty of Organizational Sciences (FOS), University of Belgrade, Serbia and a president of Breza Software Engineering company. His research interests are IS development methodologies, databases, semantic enterprise application interoperability, and information systems development. Prof Marjanovic is a lead on several on-going projects with government and commercial entities that address design, deployment, and testing of ERP and other business systems. He received his MS and PhD degrees in Information Systems from University of Belgrade. He is a member of ACM and IEEE. E-mail: zoran.marjanovic@brezasoftware.com

*Received: May 23, 2013; Accepted: June 05, 2014*