UP-TO-DATE STORAGE AND DATA MODELS

Krassimira Ivanova, Stefan Karastanev, Vitalii Velychko, Krassimir Markov

Abstract: *A short survey on several up-to-date storage and data models is outlined in this paper. Mainly they are graph as well as Resource Description Framework (RDF) models. During the eighties of the last century, the total growing of the research and developments in the computers' field, especially in image processing, data mining and mobile support, cause impetuous progress of establishing convenient "spatial information structures" and "spatial-temporal information structures" and corresponding access methods. Important cases of spatial representation of information are Graph models. Because of this, Graph models and databases will be discussed more deeply. The need to manage information with graph-like nature, especially in RDF-databases, has reestablished the relevance of this area. In accordance with this, the analyses of RDF databases as well as of the storage and retrieval technologies for RDF structures will be in the center of our attention.*

Keywords*: storage and data models, databases, Resource Description Framework, RDF-databases*

ACM Classification Keywords*: H.2 Database Management; H.2.8 Database Applications*

Introduction

Storage models and data models are closely interconnected and in the same time they are quite different. Storage models are the basic level. They appear long before the data models had been developed. From another point of view, the Data models are the core of modern information systems.

Because of this, in this survey we outline the main characteristics of both types of models. Firstly we discuss the Storage models and theirs main features. Then, we will remember the main types of Data models. Finally we will pay special attention to graph models and their using for storing of semi-structured data.

Storage models

Let remember that the "***data storage***" is a part of a computer that stores information for subsequent use or retrieval [AHD, 2009]. It is a device consisting of electronic, electrostatic, electrical, hardware, or other elements into which data may be entered, and from which data may be obtained as desired. For instance it may be magnetic tapes, hard drive storage, network storage, removable media (USB devices, flash drives, SD cards, DVDs), and online storage (Cloud storage [Mell & Grance, 2011]) [Greenwood, 2012].

The "***storage model***" is a model that captures key *physical* aspects of data structure in a data store. The ***storage schema*** (internal schema) is a specification of how the data relationships and rules specified in the logical schema of a database will be mapped to the physical storage level in terms of the available constructs, such as aggregation into records, clustering on pages, indexing, and page sizing and caching for transfer between secondary and primary storage. Storage schema facilities vary widely between different **D**ata **B**ase **M**anagement **S**ystems (DBMS) [Daintith, 2004].

Memory management is a complex field of computer science. Over the years, many techniques have been developed to make it more efficient [Ravenbrook, 2010]. Memory management is usually divided into three areas: *hardware*, *operating system*, and *applications*, although the distinctions are a little fuzzy. In most computer systems, all three are presented to some extent, forming layers between the user's program and the actual memory hardware:

- **Memory management at the hardware level** is concerned with the electronic devices that actually store data. This includes things like RAM, Associative memory, and memory caches [Mano, 1993];
- **Memory in the operating system** must be allocated to user programs, and reused by other programs when it is no longer required. The operating system can pretend that the computer has more memory than it actually does, and that each program has the machine's memory to itself. Both of these are features of *virtual memory* systems;
- **Application memory management** involves supplying the memory needed for a program's objects and data structures from the limited resources available, and recycling that memory for reuse when it is no longer required. Because in general, application programs cannot predict in advance how much memory they are going to require, they need additional code to handle their changing memory requirements.

Application memory management combines two related tasks:

- **Allocation**: when the program requests a block of memory, the memory manager must allocate that block out of the larger blocks it has received from the operating system. The part of the memory manager that does this is known as the *allocator*;
- **Recycling**: when memory blocks have been allocated, but the data they contain is no longer required by the program, the blocks can be recycled for reuse. There are two approaches to recycling memory: either the programmer must decide when memory can be reused (known as *manual memory management*); or the memory manager must be able to work it out (known as *automatic memory management*).

The progress in memory management gives the possibility to allocate and recycle not directly blocks of the memory but structured regions or fields corresponding to some types of data. In such case, we talk about corresponded "*access methods*".

The **Access Methods (AM)** had been available from the beginning of the development of computer peripheral devices. As many devices so many possibilities for developing different AM there exist. Our attention is focused only to the access methods for devices for permanently storing the information with direct access such as magnetic discs, flash memories, etc. [Markov et al, 2008].

In the beginning, the AM were functions of the Operational Systems' Core or so called Supervisor, and were executed via corresponding macro-commands in the assembler languages [Stably, 1970] or via corresponding input/output operators in the high level programming languages like FORTRAN, COBOL, PL/I, etc.

The establishment of the first databases in the sixties of the previous century caused gradually accepting the concepts "physical" as well as "logical" organization of the data [CODASYL, 1971; Martin, 1975]. In 1975, the concepts "access method", "physical organization" and "logical organization" became clearly separated. In the same time Christopher Date [Date, 1977] wrote:

"The Data Base Management System (DBMS) does not know anything about:

a) Physical records (blocks);
b) How the stored fields are integrated in the records (nevertheless that in many cases it is obviously because of their physical disposition);
c) How the sorting is realized (for instance it may be realized on the base of physical sequence, using an index or by a chain of pointers);
d) How is realized the direct access (i.e. by index, sequential scanning or hash addressing).

This information is a part of the structures for data storing but it is used by the access method but not by the DBMS".

Every access method presumes an exact organization of the file, which it is operating with and is not related to the interconnections between the files, respectively, – between the records of one file and that in the others files. These interconnections are controlled by the physical organization of the DBMS [Date, 2004].

Therefore, in the DBMS we may distinguish four levels:

- Basic access methods of the core (supervisor) of the operation system;
- Specialized access methods realized using basic access methods;
- Physical organization of the DBMS;
- Logical organization of the DBMS.

During the eighties of the last century, the total growing of the research and developments in the computers' field, especially in image processing, data mining and mobile support cause impetuous progress of establishing convenient "spatial information structures" and "spatial-temporal information structures" and corresponding access methods. From different points of view, this period has been presented in [Ooi et al, 1993; Gaede & Günther, 1998; Arge, 2002; Mokbel et al, 2003; Moënne-Loccoz, 2005; Markov et al, 2008]. Usually, the "one-dimensional" (linear) AM are used in the classical applications, based on the alphanumerical information, whereas the "multi-dimensional" (spatial) methods are aimed to serve the work with graphical, visual, multimedia information [Markov et al, 2013].

Maybe one of the most popular analyses of the genesis of the access methods is given in [Gaede & Günther, 1998]. The authors presented a scheme of the genesis of the basic multi-dimensional AM and theirs modifications. This scheme firstly was proposed in [Ooi et al, 1993] and it was expanded in [Gaede & Günther, 1998]. An extension in direction to the multi-dimensional spatio-temporal access methods was given in [Mokbel et al, 2003].

The survey [Markov et al, 2008] presents a new variant of this scheme, where the new access methods, created after 1998, are added. A comprehensive bibliography of corresponded articles, where the methods are firstly presented, is given.

Data models

Data model is a model that captures key *logical* aspects of data structure in a database, i.e. the underlying structure of a database is a *data model*. A data model is a collection of conceptual tools for describing the real-world entities to be modeled in the database and the relationships among these entities. Data models differ in the primitives available for describing data and in the amount of semantic detail that can be expressed. The various data models that have been proposed fall into three different groups: *object-based* logical models, *record-based* logical models, and *physical* data models. Physical data models are used to describe data at the lowest level. [Silberschatz et al, 1996].

There is multitude of reviews and taxonomies of data models [Silberschatz et al, 1996; Navathe, 1992; Beeri, 1988; Kerschberg et al, 1976]. An evolutionary scheme of the most important and widely accepted DataBase (DB) models is outlined in [Angles & Gutierrez, 2008] (see 0 - rectangles denote database models (db-models), arrows indicate influences, and circles denote theoretical developments; a time-line in years is shown on the left [Angles & Gutierrez, 2008]).

From a database point of view, the conceptual tools that make up a database model (db-model) should at least address data structuring, description, maintenance, and a way to retrieve or query the data. According to these criteria, a db-model consists of three components [Codd, 1970]:

— A set of data structure types;

— A set of operators or inference rules;

— A set of integrity rules.

Several proposals for db-models only define the data structures, sometimes omitting operators and/or integrity rules. In addition, each db-model proposal is based on certain theoretical principles, and serves as base for the development of related models. The short overview of Database Models (db-models) below follows one given in [Angles & Gutierrez, 2008].

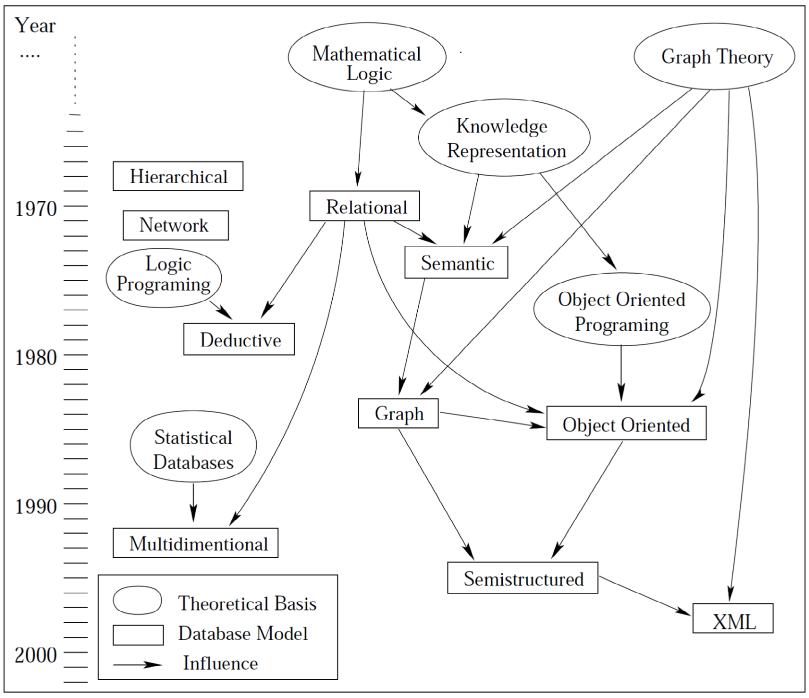


Figure 1. Evolutionary scheme of DB-models [Angles & Gutierrez, 2008]

Before the advent of the relational model, most db-models focused essentially on the specification of data structures on actual file systems (0). At this time the main information structure is the "record". Let remember that the "record" is a logical sequence of fields which contain data eventually connected to unique identifier (a "key"). The identifier (key) is aimed to distinguish one sequence from another [Stably, 1970]. The records are united in the sets, called "files". There exist three basic formats of the records – with *fixed*, *variable* and *undefined* length.

In the ***context-free file models***, storing of the records is not connected to their content and depends only on external factors – the sequence, disk address or position in the file. The main idea of the ***context-depended file models*** is that the part of the record is selected as a key which is used for making decision where to store the record and how to search it. This way the content of the record influences on the access to the record [Markov et al, 2008].

Modern Data Base Management Systems (DBMS) are built using context-depended file models such as: unsorted sequential files with records with keys; sorted files with fixed record length; static or dynamic hash files; index files and files with data; clustered indexed tables [Connolly & Begg, 2002].

Two representative database models are the ***hierarchical*** [Tsichritzis & Lochovsky, 1976] and the ***network*** [Taylor & Frank, 1976] models, both of which place emphasis on the physical level.

The ***relational db-model*** was introduced by Codd [Codd, 1970] and highlights the concept of abstraction levels by introducing the idea of separation between physical and logical levels. It is based on the notions of sets and relations.

As opposed to previous models, ***semantic db-models*** [Peckham & Maryanski, 1988] allow database designers to represent objects and their relations in a natural and clear manner, providing users with tools to faithfully capture the desired domain semantics. A well-known example is the entity-relationship model [Chen, 1976].

Object-oriented db-models [Kim, 1990] appeared in the eighties, when most of the research was concerned with so-called “advanced systems for new types of applications” [Beeri, 1988]. These db-models are based on the object-oriented paradigm and their goal is to represent data as a collection of objects, which are organized into classes, and are assigned complex values.

Graph db-models made their appearance alongside object-oriented db-models. These models attempt to overcome the limitations imposed by traditional db-models with respect to capturing the inherent graph structure of data appearing in applications such as hypertext or geographic information systems, where the interconnectivity of data is an important aspect. This type of models is outlined further.

Semi-structured db-models [Buneman, 1997] are designed to model data with a flexible structure, for example, documents and Web pages. Semi-structured data is neither raw nor strictly typed, as in conventional database systems. These db-models appeared in the nineties. Further in this chapter we will outline such type model called *Resource Description Framework (**RDF**).*

Closely related to them is the ***XML model*** *(eXtensible Markup Language)* [Bray et al, 1998], which did not originate in the database community. Although originally introduced as a document exchange standard, it soon became a general purpose model, focusing on information with tree-like structure [Angles & Gutierrez, 2008].

Mapping of the data models to storage models is based on program tools called “***access methods***”.

Semi-structured data models

Traditional database systems rely on the relational data model.

When it was proposed in the early 1970’s by Codd, a logician [Codd, 1970], the relational model generated a true revolution in data management. In this simple model data is represented as relations in first order structures and queries as first order logic formulas. It enabled researchers and implementers to separate the logical aspect of the data from its physical implementation. Thirty years of research and development followed, and they led to today's mature and highly performance relational database systems [Mendelzon et al, 2001].

The age of the Internet brought new data management applications and challenges. Data is now accessed over the Web, and is available in a variety of formats, including HTML, XML, as well as several applications specific data formats. Often data is mixed with free text, and the boundary between data and text is sometimes blurred. The way the data can be retrieved also varies considerably: some instances can be downloaded entirely; others can only be accessed through limited capabilities. To accommodate all forms and kinds of data, the database research community has introduced the ***"semi-structured data model"***, ***where data is self-describing, irregular***, ***and graph-like***. The new model captures naturally Web data, such as HTML, XML, or other application specific formats [Mendelzon et al, 2001].

The topic of semi-structured data is relatively recent [Buneman, 2001]. Applications that manage semi-structured data are becoming increasingly commonplace. Current approaches for storing semi-structured data use existing storage machinery - they either map the data to ***relational databases, or use a combination of flat files and indexes*** [Bhadkamkar et al, 2009].

In semi-structured data, the information that is normally associated with a schema contained within the data, which is sometimes called "self-describing". In some forms of semi-structured data, there is no separate schema, in others it exists but only places loose constraints on the data, Semi-structured data has recently emerged as an important topic of study for a variety of reasons. First, there are data sources such as the Web, which we would like to treat as databases but which cannot be constrained by a schema. Second, it may be desirable to have an extremely flexible format for data exchange between disparate databases. Third, even when dealing with structured data, it may be helpful to view it as semi-structured for the purposes of browsing [Buneman, 2001].

The importance of semi-structured models which are "graph-like" revived the interest to *Graph models*.

Graph models and databases

Graph database model is a model in which the data structures for the schema and/or instances are modeled as a directed, possibly labeled, graph, or generalizations of the graph data structure, where data manipulation is expressed by graph-oriented operations and type constructors, and appropriate integrity constraints can be defined over the graph structure [Angles & Gutierrez, 2008].

Graph database model can be defined as those in which data structures for the schema and instances are modeled as graphs or generalizations of them, and data manipulation is expressed by graph-oriented operations and type constructors.

The graph database models are divided into two categories:

- *Graph models with explicit schema*: *Logical Data Model (LDM)* [Kuper & Vardi, 1984, 1993]; *Hypernode Mode (HyM)* [Levene & Poulovassilis, 1990; Poulovassilis & Levene, 1994; Levene & Loizou, 1995]; *GROOVY* [Levene & Poulovassilis, 1991]; *GOOD* [Gyssens et al, 1990; Gemis & Paredaens, 1993]; *GMOD* [Andries et al, 1992]; *PaMaL* [Gemis & Paredaens, 1993]; *GOAL* [Hidders & Paredaens, 1993]; *GDM* [Hidders, 2001, 2002]; *Gram* [Amann & Scholl, 1992].
- *Graph models with implicit schema*: *Object Exchange Model (OEM)* [Papakonstantinou et al, 1995]; *GGL* [Graves, 1993; Graves et al, 1994; 1995a; 1995b]; *RDF* [Klyne & Carroll, 2004; Hayes & Gutierrez, 2004; Angles & Gutierrez, 2005]; *Simatic-XT* [Mainguenaud, 1992].

The notion of graph database model can be conceptualized with respect to three basic components, namely:

- Data structures;
- Transformation language;
- Integrity constraints.

Hence, a graph database model is characterized as follows:

- Data and/or the schema are represented by graphs, or by data structures generalizing the notion of graph (hypergraphs or hypernodes) [Guting, 1994; Levene & Loizou, 1995; Kuper & Vardi, 1984; Paredaens et al, 1995; Kunii, 1987; Graves et al, 1995a; Gyssens et al, 1990];
- Data manipulation is expressed by graph transformations, or by operations whose main primitives are on graph features like paths, neighborhoods, subgraphs, graph patterns, connectivity, and graph statistics (diameter, centrality, etc.) [Gyssens et al, 1990; Graves et al, 1995a; Guting, 1994];
- Integrity constraints enforce data consistency. These constraints can be grouped in schema-instance consistency, identity and referential integrity, and functional and inclusion dependencies. Examples of these are: labels with unique names, typing constraints on nodes, functional dependencies, domain and range of properties [Graves et al, 1995b; Kuper & Vardi, 1993; Klyne & Carroll, 2004; Levene & Poulovassilis, 1991].

Advantages of Graph database models

Graph database models are applied in areas where information about data interconnectivity or topology is more important, or as important, as the data itself. In these applications, the data and relations among the data are usually at the same level.

Introducing graphs as a modeling tool has several advantages for this type of data.

- It allows for a more natural modeling of data. Graph structures are visible to the user and they allow a natural way of handling applications data, for example, hypertext or geographic data. Graphs have the advantage of being able to keep all the information about an entity in a single node and showing related information by edges connected to it [Paredaens et al, 1995]. Graph objects (like paths and neighborhoods) may have first order citizenship; a user can define some part of the database explicitly as a graph structure [Guting, 1994], allowing encapsulation and context definition [Levene & Poulovassilis, 1990];
- Queries can refer directly to this graph structure. Associated with graphs are specific graph operations in the query language algebra, such as finding shortest paths, determining certain subgraphs, and so forth. Explicit graphs and graph operations allow users to express a query at a high level of abstraction. To some extent, this is the opposite of graph manipulation in deductive databases, where often, fairly complex rules need to be written [Guting, 1994]. It is not important to require full knowledge of the structure to express meaningful queries [Abiteboul et al, 1997]. Finally, for purposes of browsing it may be convenient to forget the schema [Buneman et al, 1996];
- For implementation, graph databases may provide special graph storage structures, and efficient graph algorithms for realizing specific operations [Guting, 1994].

Graph database models took off in the eighties and early nineties alongside object-oriented models. Their influence gradually died out with the emergence of other database models, in particular geographical, spatial, semi structured, and XML.

Recently, the need to manage information with graph-like nature especially in *RDF-databases* has reestablished the relevance of this area [Angles & Gutierrez, 2008].

RDF databases

Resource Description Framework (RDF) is the W3C recommendation for semantic annotations in the Semantic Web. RDF is a standard syntax for Semantic Web annotations and languages [Klyne & Carroll, 2004].

The design of a traditional database is guided by the discovery of regularity or uniformity. The principle of regularity is a standardization of design relying on an abstract view of the world, where exceptions to the rule are not taken into account, since they are considered as insignificant in the design of an advantageous structured schema. The popularity of relational database management systems (RDBMS) is due to their ability to support many data management problems dealt by applications. *However, a priori uniformity required by relational model can lead to hardness when modeling a not static world such as Semantic Web data* [Faye et al, 2012].

The primary goal of RDF is to handle ***non regular or semi-structured data***. The research community has early recognized that there is an increasing amount of data that is insufficiently structured to support traditional database techniques, but does contain a sufficiently regular structure exploitable in the formulation and execution of queries [Muys, 2007].

It is widely acknowledged that information access can benefit from the use of ***ontologies***. For this purpose, available data has to be linked to concepts and relations in the corresponding ontology and access mechanisms have to be provided that support the integrated model consisting of ontology and data. The most common approach for linking data to ontologies is via RDF representation of available data that describes the data as instances of the corresponding ontology that is represented in terms of RDF Schema. Due to the practical relevance of data access based on RDF and RDF Schema, a lot of effort has been spent on the development of corresponding storage and retrieval infrastructures [Hertel et al, 2009].

The underlying structure of any expression in RDF is a collection of triples, each consisting of a subject, a predicate and an object. A set of such triples is called RDF graph [RDF, 2013]. This can be illustrated by a node and directed-edge diagram, in which each triple is represented as a “node-edge-node” link (hence the term "graph") (Figure 2).

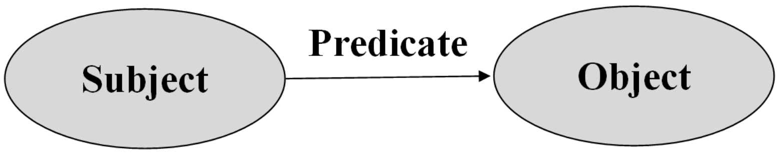


Figure 2. RDF triple

Each triple represents a statement of a relationship between the things denoted by the nodes that it links. It has three parts:

- Subject;
- A predicate (also called a property) that denotes a relationship;
- Object.

The direction of the edge (predicate) is significant: it always points toward the object. The nodes of RDF graph are its subjects and objects.

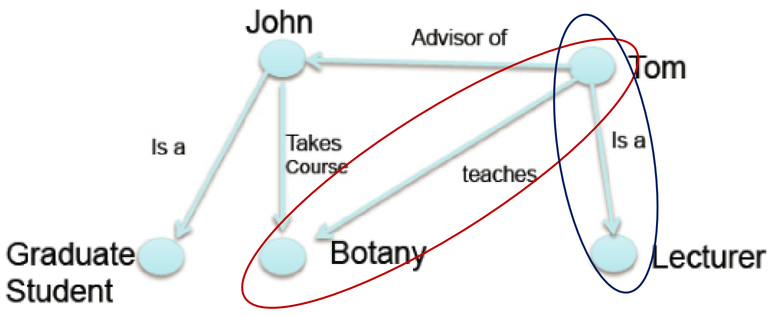
The assertion of RDF triple says that some relationship, indicated by the predicate, holds between the things denoted by subject and object of the triple. The assertion of RDF graph amounts to asserting all the triples in it, so the meaning of RDF graph is the conjunction (logical AND) of the statements corresponding to all the triples it contains. A formal account of the meaning of RDF graphs is given in [Hayes, 2004]. In other words, RDF provides a general method to decompose any information into pieces called triples [Briggs, 2012]:

- Each triple is of the form “Subject”, “Predicate”, “Object”;
- Subject and Object are the names for two things in the world. Predicate is the relationship between them;
- Subject, Predicate, Object are given as URI’s (stand-ins for things in the real world);
- Object can additionally be raw text.

In technical terms the RDF-triples’ set form labeled directed graph, where each edge is a triple, for instance, the triples:

Subject	Predicate	Object
<Tom>	<is a>	<Lecturer>
<Tom>	<teaches>	<Botany>

define some of the elements of the next graph [Briggs, 2012]:



The research community has early recognized the natural flexibility and expressivity of triples. Indeed, triples consider both objects and relationships as first-class citizens; thus, allowing on-the-fly generation of data. The power of RDF relies on the flexibility in representing arbitrary structure without a priori schemas. Each edge in the graph is a single fact, a single statement, similar to the relationship between a single cell in a relational table and its row’s primary key. RDF offers the ability to specify concepts and link them together into a graph of data [Faye et al, 2012].

RDF advantages

As a storage language, RDF has several advantages [Owens, 2009]. First, it is possible to link different data sources together by adding a few additional triples specifying relationships between the concepts. This would be more difficult in the case of an RDBMS in which schema realignment or matching may be necessary. Then, RDF offers a great deal of flexibility due to the variety of the underlying graph-based model (i.e. almost any type of data can be expressed in this format with no needs for data to be present). There is no restriction on the graph size, as opposed to RDBMS field where schema must be concise. This a significant gains when the structure of the data is not well known in advance. Last, any kind of knowledge can be expressed in RDF, authorizing extraction and reuse of knowledge by various applications.

Consequently RDF offers a very useful data format, for which efficient management is needed. This becomes a hard issue for application dealing with RDF and known as ***RDF (or Triple) Stores***, due to the irregularity of the data. RDF Stores must allow the following fundamental operations on repository of RDF data: *performing a query*, *updating*, *inserting (assertion),* and *deleting (retraction) triples* [Owens, 2009]. In addition, there are issues that may require an extension of the triple-based schemas and thus are affecting the design of the database tables:

- Storing multiple ontologies in one database;
- Storing statements from multiple documents in one database.

Both points are concerning the aspect of provenance, which means keeping track of the source an RDF statement is coming from.

When storing multiple ontologies in one database it should be considered that classes, and consequently the corresponding tables, can have the same name. Therefore, either the tables have to be named with a prefix referring to the source ontology or this reference is stored in an additional attribute for every statement [Pan & Heflin, 2004].

A similar situation arises for storing multiple documents in one database. Especially, when there are contradicting statements it is important to know the source of each statement. Again, an additional attribute denoting the source document helps solving the problem [Pan & Heflin, 2004].

The concept of "***named graphs***" [Caroll et al, 2004] is including both issues. The main idea is that each document or ontology is modeled as a graph with a distinct name, mostly an URI. This name is stored as an additional attribute, thus extending RDF statements from triples to so-called quads. For the database schemas described above this means adding a fourth column to the tables and potentially storing the names of all graphs in a further table.

RDF disadvantages

Different authors report different and specific RDF disadvantages. For instance, in [Costello & Jacobs, 2003] is noted that disadvantages of using the RDF format are:

- RDF uses namespaces to uniquely identify types (classes), properties, and resources. Thus, one must have a solid understanding of namespaces;

- Constrained: the RDF format constrains one on how he design his XML (i.e., one can't design his XML in any arbitrary fashion);
- Another XML vocabulary to learn: to use the RDF format one must learn the RDF vocabulary.

Other point of view we see in [Baidu, 2013]. RDF disadvantages are:

- Generic triple storage often (but not always) implies less efficient lookups (special indexes can still be built, but this moves away from schema flexibility);
- Certain data cannot easily be represented in RDF;
- Practical disadvantages with respect to (relatively) immature RDF storage systems and tools and porting over existing systems;
- High overhead for developers to get the necessary expertise to do a good job;
- Only non-standard solutions available for declaratively specifying (common types of CWA) constraints;
- The RDF triple is ontology based, always need the same schema;
- Not easy to do some complex reasoning;
- Low efficient to query data in the RDF triples, compared against RDBMS.

From our point of view, it is important to discuss the problem of ***numbering*** large RDF triple's elements (strings). Developers generally make special provisions for storing RDF resources efficiently. Indeed, rather than storing each Internationalized Resource Identifier (IRI) or literal value directly as a string, implementations usually associate a *unique numerical identifier* to each resource and store this identifier instead [Yongming et al, 2012].

There are two motivations for this strategy. First, since there is no a priori bound on the length of the IRIs or literal values that can occur in RDF graphs, it is necessary to support variable-length records when storing resources directly as strings. By storing the numerical identifiers instead, fixed-length records can be used. Second, and more importantly, RDF graphs typically contain very long IRI strings and literal values that, in addition, are frequently repeated in the same RDF graph.

Unique identifiers can be computed in two general ways [Yongming et al, 2012]:

- ***Hash-based approaches*** obtain a unique identifier by applying a hash function to the resource string, where the hash function used for IRIs may differ from the hash function used for literal values. Of course, care must be taken to deal with possible hash collisions. In the extreme, the system may reject addition of new RDF triples when a collision is detected. To translate hash values back into the corresponding IRI or literal value when answering queries, a distinguished dictionary table is constructed;
- ***Counter-based approaches*** obtain a unique identifier by simply maintaining a counter that is incremented whenever a new resource is added. To answer queries, dictionary

tables that map from identifiers to resources and vice versa are constructed. Typically, these dictionary tables are stored as B-Trees for efficient retrieval. A variant on this technique that is applicable when the RDF graph is static is to first sort the resource strings in lexicographic order, and to assign the identifier n to the nth resource in this order. In such a case, a single dictionary table suffices to perform the mapping from identifiers to resources and vice versa [Yongming et al, 2012].

Various optimizations can be devised to further improve storage space. For example, when literal values are small enough to serve directly as unique identifiers (e.g., literal integer values), there is no need to assign unique identifiers, provided that the storage medium can distinguish between the system-generated identifiers and the small literal values. Also, it is frequent that many IRIs in an RDF graph share the same namespace prefix. By separately encoding this namespace prefix, one can further reduce the storage requirements [Yongming et al, 2012].

In other words, ***the bottleneck problem for RDF is numbering of very great amount of strings from RDF triples, sometimes up to several billion instances***.

For goal of this research we chose the second approach for solving the problem, i.e. to use counters. The new idea is that the process of numbering does not use B-Trees or any variant of traditional hashing. We use NL-addressing to assign numbers and co-ordinate access to restore string which corresponds to given number. The algorithm is presented in [Ivanova, 2015]. It has constant complexity which is important for very large datasets.

Storage and retrieval technologies for RDF

The state of the art with respect to existing storage and retrieval technologies for RDF data is given in [Hertel et al, 2009] as well as in [Faye et al, 2012]. Different repositories are imaginable, e.g. main memory, files or databases.

RDF schemas and instances can be efficiently accessed and manipulated in main memory. Storing everything ***in-memory cannot be a serious method*** for storing extremely large volumes of data. However, they can act as useful benchmark and can be used for performing certain operations like caching data from remote sites or for performing inference. Most of the in-memory stores have efficient reasoners available and can help solve the problem of performing inference in persistent RDF stores, which otherwise can be very difficult to perform [CTS, 2012].

For persistent storage, the data can be serialized to files, but for large amounts of data the use of database management system is more reasonable. Examining currently existing RDF stores we found that they have used relational and object-relational database management systems.

Storing RDF data in a (relational) database requires an appropriate table design. There are different approaches that can be classified in:

- **Generic schemas**, i.e. schemas that do not depend on the ontology and run on third party databases (For instance, Jena SDB which can be coupled with almost all relational databases like MySQL, PostsgreSQL, and Oracle);
- **Ontology specific schemas**, for instance, the native triple stores which provide persistent storage with their own implementation of the databases (Virtuoso, Mulgara, AllegroGraph, and Garlik JXT).

Main characteristics of several known RDF triple stores are presented in Table 1.

Storing ontology generic schemas

Vertical representation

The simplest RDF generic schema is a triple store with only one table required in the database.

The table contains three columns named *Subject*, *Predicate* and *Object*, thus reflecting the triple nature of RDF statements. Indexes are added for each of the columns in order to make joins less expensive. This corresponds to the *vertical representation* for storing objects in a table [Agrawal et al, 2001].

In this case, no restructuring is required if the ontology changes. This is the greatest advantage of this schema. Adding the new classes and properties to ontology can be realized by a simple ***INSERT*** command in the table. On the other hand, performing a query means searching the whole database and queries involving joins become very expensive. Another aspect is that the class hierarchy cannot be modeled in this schema, what makes queries for all instances of a class rather complex [Hertel et al, 2009].

In other words, since the collections of triples are stored in one single RDF table, *the queries may be very slow to execute*. Indeed, when the number of triples scales, the RDF table may exceed main memory size. Additionally, simple statement-based queries can be satisfactorily processed by such systems, although they do not represent the most important way of querying RDF data. Nevertheless, RDF triples store scales poorly because complex queries with multiple triple patterns require many self-joins over this single large table as pointed out in [Faye et al, 2012].

The triple table approach has been used by systems like Oracle [oracledb, 2012; Chong et al, 2005], 3store [Harris & Gibbins, 2003], Redland [Beckett, 2001], RDFStore [RDFStore, 2012] and rdfDB [Guha, 2013].

Normalized triple store (vertical partitioning)

The triple store can be used in its pure form [Oldakowski et al, 2005], but most existing systems add several modifications to improve performance or maintainability. A common approach, the so-called *normalized triple store*, is adding two further tables to store resource URIs and literals separately as shown in Figure 3, which requires significantly less storage space [Harris & Gibbins, 2003]. Furthermore, a hybrid of the simple and the normalized triple store can be used, allowing storing the values themselves either in the triple table or in the resources table [Jena2, 2012].

Triples:

Subject	Predicate	IsLiteral	Object
r1	*r2*	*False*	*r3*
r1	*r4*	*True*	*l1*
…	…	…	…

Resources:

ID	URI
r1	*…#1*
r2	*…#2*
…	…

Literals:

ID	Value
l1	*Value1*
…	…
…	…

Figure 3. Normalized triple store

In a further refinement, the Triples table can be *split horizontally* into several tables, each modeling an RDF property. These tables need only two columns for *Subject* and *Object*. The table names implicitly contain the predicates. This schema separates the ontology schema from its instances, explicitly models class and property hierarchies and distinguishes between class-valued and literal-valued properties [Broekstra, 2005; Gabel et al, 2004].

To realize the vertical partitioning approach, the tables have to be stored by using a *column-oriented DBMS* (i.e., a DBMS designed especially for the vertically partitioned case, as opposed to a row oriented DBMS, gaining benefits of compressibility and performance), as collections of columns rather than collections of rows. The goal is to avoid reading entire row into memory from disk, like in row-oriented databases, if only a few attributes are accessed per query. Consequently, in column oriented databases only those columns relevant to a query will be read. The approach creates materialized views for frequent joins. Furthermore, the object columns of tables in their scheme can also be optionally indexed (e.g., using an unclustered B+ tree), or a second copy of the table can be created clustered on the object column. One of the primary benefits of vertical partitioning is the support for rapid subject joins. This benefit is achieved by sorting the tables via subject. The tables being sorted by subject, one has a way to use fast merge joins to reconstruct information about multiple properties for subsets of subjects.

Index-all approach is a poor way to simulate a column-store. The vertical partitioning approach offers a support for multi-valued attributes. Indeed, if a subject has more than one object value for a given property, each distinct value is listed in a successive row in the table for that property. For a given query, only the properties involved in that query need to be read and no clustering algorithm is needed to divide the triples table into two-column tables.

Inserts can be slow in vertically partitioned tables since multiple tables need to be accessed for statement about the same subject. With a larger number of properties, the triple store solution manages to outperform the vertically partitioned approach [Faye et al, 2012].

Storing ontology specific schemas

Horizontal representation

Ontology specific schemas are changing when the ontology changes, i.e. when classes or properties are added or removed. The basic schema consists of one table with one column for the instance identificator (ID), one for the class name and one for each property in the ontology. Thus, one row in the table corresponds to one instance. This schema is corresponding to the *horizontal representation* [Agrawal et al, 2001] and obviously has several drawbacks:

- Large number of columns;
- High sparsity;
- Inability to handle multi-valued properties;
- The need to add columns to the table when adding new properties to the ontology,

etc.

Horizontally splitting the schema results in the so called one-table-per class schema, i.e. one table for each class in the ontology is created. A class table provides columns for all properties whose domain contains this class. This is tending to the classic entity-relationship-model in database design and benefits queries about all attributes and properties of an instance.

However, in this form the schema still lacks the ability to handle multi-valued properties, and properties that do not define an explicit domain must then be included in each table. Furthermore, adding new properties to the ontology again requires restructuring existing tables [Hertel et al, 2009].

Decomposition storage model

Another approach is vertically splitting the schema, what results in the *one-table-per-property schema*, also called the *decomposition storage model*.

In this schema one table for each property is created with only two columns for *Subject* and *Object*. RDF properties are also stored in such tables, e.g. the table for rdf:type contains the relationships between instances and their classes.

This approach is reflecting the particular aspect of RDF that properties are not defined inside a class. However, complex queries considering many properties have to perform many joins, and queries for all instances of a class are similarly expensive as in the generic triple schema [Hertel et al, 2009].

In practice, a *hybrid schema* is used to benefit from advantages of combining both the table-per-class and table-per property schemas. This schema contains one table for each class, only storing there a unique ID for the specific instance. This replaces the modeling of the rdf:type property. For all other properties tables are created as described in the table-per-property approach (Figure 4) [Pan & Heflin, 2004]. Thus, changes to the ontology do not require changing existing tables, as adding a new class or property results in creating a new table in the database.

ClassA:

ID
…#1
…

Property1:

Subject	Object
…#1	…#3
…	…

ClassB:

ID
…#3
…

Figure 4. RDF Hybrid schema (the table-per-property approach)

A possible modification of this schema is separating the ontology from the instances. In this case, only instances are stored in the tables described above.

Information about the ontology schema is stored separately in four additional tables *Class*, *Property*, *SubClass* and *SubProperty* [Alexaki et al, 2001]. These tables can be further refined storing only the property ID in the Property table and the domain and range of the property in own tables Domain and Range [Broekstra, 2005]. This approach is similar to refined generic schema, where ontology is stored the same way and only storage of instances is different.

To reduce the number of tables, single-valued properties with a literal as range can be stored in the class tables [Wilkinson, 2006; Broekstra et al, 2002]. Adding new attributes would then require changing existing tables. Another variation is to store all class instances in one table called Instances. This is especially useful for ontologies where there are many classes with only few or no instances [Alexaki et al, 2001; Wilkinson, 2006; Inseok et al, 2005].

The property table technique has the drawback of generating many NULL values since, for a given cluster, not all properties will be defined for all subjects. This is due to the fact that RDF data may not be very structured. A second disadvantage of property table is that multi-valued attributes, that are furthermore frequent in RDF data, are hard to express. In a data model without a fixed schema like RDF, it’s common to seek for all defined properties of a given subject, which, in the property table approach, requires scanning all tables.

In this approach, including new properties requires also adding new tables; which is clearly a limitation for applications dealing with arbitrary RDF content. Thus schema flexibility is lost and this approach limits the benefits of using RDF. Moreover, queries with triples patterns that involve multiple property tables are still expensive because they may require many union clauses and joins to combine data from several tables. This consequently complicates query translation and plan generation. In summary, property tables are rarely used due to their complexity and inability to handle multi-valued attributes [Faye et al, 2012].

This approach has been used by tools like Sesame [Sesame, 2012; Broekstra et al, 2002], Jena2 [Jena2, 2012; Wilkinson et al, 2003], RDFSuite [Alexaki et al, 2001] and 4store [Harris et al, 2009].

Multiple indexing frameworks

The idea of multi-indexing is based on the fact that queries bound on property value are not necessarily the most interesting or popular type of queries encountered in real world Semantic Web applications.

Due to the triple nature of RDF data, the goal is to handle equally the following type of queries:

- Triples having the same subject;
- Triples having the same property;
- List of subjects or properties related to a given object.

For achieving this goal, these approaches maintain a set of six indices covering all possible access schemes an RDF query may require. These indexes are PSO, POS, SPO, SOP, OPS, and OSP (P stands for property, O for object and S for subject). These indices materialize all possible orders of precedence of the three RDF elements. At first sight, such a multiple-indexing would result into a combinatorial explosion for an ordinary relational table. Nevertheless, it is quite practical in the case of RDF data [Weiss et al, 2008; RDF, 2013]. The approach does not treat property attributes specially, but pays equal attention to all RDF items [Faye et al, 2012].

This approach has been used by tools like Kowari system [Wood et al, 2005], Virtuoso [Erling & Mikhailov, 2007], RDF-3X [Neumann & Weikum, 2008], Hexastore [Weiss et al, 2008], RDFCube [Matono et al, 2007], BitMat [Atre et al, 2009], BRAHMS [Janik & Kochut, 2005], RDFJoin [McGlothlin & Khan, 2009], RDFKB [McGlothlin & Khan, 2009a], TripleT [Fletcher & Beck, 2009], iStore [Tran et al, 2009], Parliament [Kolas et al, 2009].

Storing models for popular ontologies

Storing models for nine popular linguistic, conceptual or mixed ontologies are outlined in Table 1. These models are similar and practically are based on the well-known file systems or relational databases (RDBMS). In the case of RDBMS, orientation is mainly toward SPARQL. The ontologies are described by high-level languages (e.g. KIF, CycL, SubL, RDF, XML), which can be interpreted and/or stored in relational structures (e.g. MySQL), ER-model (e.g. FreeBase) and others.

In general, the systems for storing ontologies and, in particular, RDF data are based on (see also [Magkanaraki et al, 2002]):

– ***Structures in memory*** (e.g. TRIPLE [Sintek & Decker, 2001]);
– Popular ***relational databases*** (e.g. ICS-FORTH RDF Suite [Alexaki et al, 2001; 2001a], Semantics Platform 2.0 of Intellidimension Inc. [ISP2.0, 2012], Ontopia Knowledge Suite [Ontopia, 2012]);
– ***Non-relational file systems***, indexed by key B-trees, such as Oracle Berkeley DB (e.g. rdfDB [Dumbill, 2000], RDF Store [RDFStore, 2012], Redland [Beckett, 2001], Jena [McBride, 2001]).

Table 1. Methods for storing data in nine ontologies

	ontology name	developer	quantity of terms	storing models	integration with other ontologies
1	**WordNet** [Fellbaum et al, 1998; Miller, 1995]	Princeton University	about 100 000	files	SUMO, FrameNet
2	**Sensus** [ISI, 2012]	ISI USC	more than 70 000	files, relational databases	subset of the WordNet
3	**Omega** [Philpot et al, 2005]	ISI USC	about 120 000	relational databases (MySQL)	WordNet, Mikrokosmos
4	**Mikrokosmos** [Beale et al, 1996; Mikr, 2012]	CLR UNMS	more than 7 000	relational database	WordNet, Omega
5	**OpenCyc** [OpenCyc, 2012]	Cycorp	more than 100 000	files (CycL, SubL, RDF)	WordNet
6	**DOLCE** [Masolo et al, 2003]	LAO ICST	about 4 000	files (KIF)	No
7	**PropBank** [Giuglea & Moschitti, 2004; Kingsbury & Palmer, 2003]	University PennState	more than 4 300	frame files	FrameNet, VerbNet
8	**FrameNet** [Fillmore, 1976; Baker et al, 1998; FrameNet, 2012]	ISI, Berkeley, CA	about 900 frames	files (XML)	WordNet, PropBank, SUMO
9	**SUMO** [SUMO, 2012]	Teknowledge Corporation, SUO WG	more than 1 000	SUO-KIF files	FrameNet, WordNet, EMELD

Conclusion

A short survey on several up-to-date storage and data models was outlined in this paper. Mainly they are graph as well as Resource Description Framework (RDF) models. During the eighties of the last century, the total growing of the research and developments in the computers' field, especially in image processing, data mining and mobile support, cause impetuous progress of establishing convenient "spatial information structures" and "spatial-temporal information structures" and corresponding access methods. Important cases of spatial representation of information are Graph models. Because of this, Graph models and databases were discussed more deeply. The need to manage information with graph-like nature, especially in RDF-databases, has reestablished the relevance of this area. In accordance with this, the analyses of RDF databases as well as of the storage and retrieval technologies for RDF structures were in the center of our attention.

Acknowledgements

The paper is published with partial support by the project ITHEA XXI of the ITHEA ISS (www.ithea.org) and the ADUIS (www.aduis.com.ua).

Bibliography

[Abiteboul et al, 1997] Abiteboul S., Quass D., McHugh J., Widom J., and Wiener J. L "*The Lorel query language for semistructured data*", International Journal on Digital Libraries (JODL) 1, 1, 1997, pp. 68–88.

[Agrawal et al, 2001] Agrawal R., Somani A, Xu Y., "*Storage and querying of e-commerce data*", In: Proceedings of the 27th Conference on Very Large Data Bases, VLDB 2001, and Roma, Italy.

[AHD, 2009] The American Heritage® "*Dictionary of the English Language*" Fourth Edition copyright© 2000 by Houghton Mifflin Company, Updated in 2009; Published by Houghton Mifflin Company. All rights reserved.

[Alexaki et al, 2001] Sofia Alexaki, Vassilis Christophides, Gregory Karvounarakis, Dimitris Plexousakis, Karsten Tolle "*The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases*", 2nd International Workshop on the Semantic Web (SemWeb'01), Hongkong, 2001.

[Alexaki et al, 2001a] Alexaki S., V. Christophides, G. Karvounarakis, D. Plexousakis, "*On Storing Voluminous RDF Descriptions: The case of Web Portal Catalogs*", In Proceedings of the 4th International Workshop on the Web and Databases (WebDB'01) - In conjunction with ACM SIGMOD/PODS, Santa Barbara, CA. May 24-25, 2001.

[Amann & Scholl, 1992] Amann B. and Scholl, M. "*Gram: A Graph Data Model and Query Language*", In European Conference on Hypertext Technology (ECHT), ACM, 1992, pp. 201–211.

[Andries et al, 1992] Andries M., Gemis M., Paredaens J., Thyssens I., and den Bussche, J. V. "*Concepts for Graph-Oriented Object Manipulation*", In Proc. of the 3rd Int. Conf. on Extending Database Technology (EDBT) LNCS, vol. 580, Springer, 1992, pp. 21–38.

[Angles & Gutierrez, 2005] Angles, R. and Gutierrez, C, "*Querying RDF Data from a Graph Database Perspective*", In Proc. 2nd European Semantic Web Conference (ESWC), Number 3532 in LNCS. 2005, pp. 346–360.

[Angles & Gutierrez, 2008] Angles R., C. Gutierrez, "*Survey of Graph Database Models*", ACM Computing Surveys, Vol. 40, No. 1, Article 1, Publication date: February 2008, DOI 10.1145/1322432.1322433, http://doi.acm.org/10.1145/1322432.1322433, pp. 1-39

[Arge, 2002] Arge, L., "*External memory data structures*", In: Handbook of Massive Datasets, Part 4, ch. 9. Kluwer Academic Publishers, 2002. pp. 313-357.

[Atre et al, 2009] Medha Atre, Jagannathan Srinivasan, James A. Hendler, "*BitMat: A Main Memory RDF Triple Store*", Technical Report, Tetherless World Constellation, Rensselaer Polytechnic Institute, Troy NY, USA, 2009.

[Baidu, 2013] http://hi.baidu.com/huyangtree/item/5993ece1c094e1bc2f140b86 (accessed: 16.12.2013)

[Baker et al, 1998] Baker F. C., C. J. Fillmore, J. B. Lowe, "*The Berkeley FrameNet Project*", COLING–ACL, Montreal, Canada, 1998, pp. 86-90, http://acl.ldc.upenn.edu/C/C98/C98-1013.pdf (accessed: 21.07.2012)

[Beale et al, 1996] Beale S., S. Nirenburg and K. Mahesh, „*Semantic Analysis in the Mikrokosmos Machine Translation Project*", 1996, pp. 1-11, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.9053 (accessed: 21.07.2012)

[Beckett, 2001] Beckett David, "*The design and implementation of the Redland RDF Application Framework*", WWW10, 2001, Hong Kong, ACM 1-58113-348-0/01/0005, Redland - URL: http://www.redland.opensource.ac.uk/ (accessed: 15.10.2012).

[Beeri, 1988] Beeri, C., "*Data models and languages for databases*", In Proceedings of the 2nd International Conference on Database Theory (ICDT), LNCS, vol. 326, Springer, 1988, pp. 19–40.

[Bhadkamkar et al, 2009] Medha Bhadkamkar, Fernando Farfan, Vagelis Hristidis, and Raju Rangaswami, "*Storing Semi-structured Data on Disk Drives*", ACM Transactions on Storage, Vol. 5, No. 2, Article 6, Publication date: June 2009, pp. 6.1–6.35, ACM New York, NY, USA ISSN: 1553-3077 EISSN: 1553-3093 doi>10.1145/1534912.1534915 (accessed: 20.07.2013)

[Bray et al, 1998] Bray, T., Paoli, J., and Sperberg-Mcqueen, C. M., "*Extensible Markup Language (XML) 1.0*", W3C Recommendation 10, (February), 1998. http://www.w3.org/TR/1998/REC-xml-19980210 (accessed: 20.07.2013).

[Briggs, 2012] Mario Briggs, "*DB2 NoSQL Graph Store*", What, Why & Overview, A presentation, Information Management software IBM, 2012, https://www.ibm.com/developerworks/mydeveloperworks/blogs/nlp/resource/DB2_NoSQLGraphStore.pdf?lang=en (accessed: 01.12.2012)

[Broekstra et al, 2002] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen, "*Sesame: A Generic Architecture for Storing and Querying RDF and RDF*", 2002.

[Broekstra, 2005] Broekstra J., "*Storage, querying and inferencing for Semantic Web languages*", PhD Thesis, Vrije Universiteit, Amsterdam (2005)

[Buneman et al, 1996] Buneman, P., Davidson, S., Hillebrand, G., and Suciu, D., "*A Query Language and Optimization Techniques for Unstructured Data*", SIGMOD Record. 25, 2, 1996, pp. 505-516.

[Buneman, 1997] Buneman, P, "*Semistructured data*", In Proceedings of the 16th Symposium on Principles of Database Systems (PODS), ACM Press, 1997, pp. 117-121.

[Buneman, 2001] Peter Buneman, "*Semistructured Data*", Department of Computer and Information Science, University of Pennsylvania http://homepages.inf.ed.ac.uk/opb/papers/PODS1997a.pdf (accessed: 20.07.2013)

[Caroll et al, 2004] Caroll J, Bizer C, Hayes P, Stickler P., "*Semantic Web publishing using named graphs*", In: Proceedings of Workshop on Trust, Security, and Reputation on the SemanticWeb, at the 3rd International SemanticWeb Conference, ISWC 2004, Hiroshima, Japan.

[Chen, 1976] Chen, P. P. S, "*The entity-relationship model—toward a unified view of data*", ACM Trans. Database Syst., 1, 1, 1976, pp. 9–36

[Chong et al, 2005] Eugene Inseok Chong, Souripriya Das, George Eadon, Jagannathan Srinivasan, "*An efficient SQL-based RDF querying scheme*", VLDB '05: Proceedings of the 31stinternational conference on Very large data bases, Trondheim, Norway, 2005.

[CODASYL, 1971] Codasyl Systems Committee, "*Feature Analysis of Generalized Data Base Management Systems*", Technical Report, May, 1971.

[Codd, 1970] Codd, E., "*A relation model of data for large shared data banks*", Magazine Communications of the ACM, 13/6, 1970, pp. 377-387

[Connolly & Begg, 2002] T.M. Connolly, C.E.Begg, "*Database Systems*", A Practical Approach to Design, Implementation, and Management, Third Edition, Addison-Wesley Longman, Inc. – Pearson Education Ltd., 1995, 2002

[Costello & Jacobs, 2003] Roger L. Costello, David B. Jacobs, "*XML Design*", (A Gentle Transition from XML to RDF), The MITRE Corporation, 2003, http://www.csee.umbc.edu/courses/771/current/presentations/rdf.ppt (accessed: 16.12.2013)

[CTS, 2012] Comparison of Triple Stores http://www.bioontology.org/wiki/images/6/6a/Triple_Stores.pdf (accessed: 11.01.2013).

[Daintith, 2004] John Daintith, "*Storage Schema*", A Dictionary of Computing, and 2004, Retrieved November 18, 2012, from Encyclopedia.com: http://www.encyclopedia.com/doc/1O11-storageschema.html (accessed: 26.11.2012)

[Date, 1977] Date C. J., "*An Introduction to Database Systems*", Addison-Wesley Inc., 1975.

[Date, 2004] Date C. J., "*An Introduction to Database Systems*", 8th Edition, Pearson Education, Inc, ISBN 0-324-18956-6, 2004.

[Dumbill, 2000] Dumbill E., "*Putting RDF to Work*", Article on XML.com, 09.08.2000. (http://www.xml.com/pub/a/2000/08/09/rdfdb/); rdfDB URL: http://guha.com/rdfdb/ (accessed: 15.10.2012).

[Erling & Mikhailov, 2007] Orri Erling, Ivan Mikhailov, "*RDF Support in the Virtuoso DBMS*", Conference on Social Semantic Web, 2007.

[Faye et al, 2012] David C. Faye, Olivier Cure, Guillaume Blin, "*A survey of RDF storage approaches*", Received, December 12, 2011, Accepted, February 7, 2012, ARIMA Journal, vol. 15, 2012, pp. 11-35.

[Fellbaum et al, 1998] Fellbaum, Christiane, ed., "*WordNet: An Electronic Lexical Database*", MIT Press, Cambridge, MA, 1998, pp. 422

[Fillmore, 1976] Fillmore C. J., "*Frame semantics and the nature of language*", Annals of the New York Academy of Sciences, Volume 280, 1976, pp. 20–32.

[Fletcher & Beck, 2009] George H. L. Fletcher, Peter W. Beck, "*Scalable indexing of RDF graphs for efficient joins processing*", CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management, New York, NY, USA, 2009.

[FrameNet, 2012] FrameNet II FrameGrapher. http://framenet.icsi.berkeley.edu/FrameGrapher (accessed: 21.07.2012)

[Gabel et al, 2004] Gabel T, Sure Y, Voelker J., "*KAON – An overview*", Insititute AIFB, University of Karlsruhe, 2004, http://www.aifb.kit.edu/web/KAON/en (accessed: 11.08.2012).

[Gaede & Günther, 1998] Gaede V. and Günther O, "*Multidimensional access methods*", ACM Computing Surveys, 30(2), 1998

[Gemis & Paredaens, 1993] Gemis, M. and Paredaens, J., "*An Object-Oriented Pattern Matching Language*", In Proc. of the First JSSST Int. Symposium on Object Technologies for Advanced Software, Springer- Verlag, 1993, pp. 339–355.

[Giuglea & Moschitti, 2004] Giuglea A, A. Moschitti, "*Knowledge Discovering using FrameNet*", VerbNet and PropBank, 2004, pp. 6, http://olp.dfki.de/pkdd04/giuglea-final.pdf (accessed: 21.07.2012)

[Graves et al, 1994] Graves, M., Bergeman, E. R., and Lawrence, C. B., "*Querying a Genome Database using Graphs*", In In Proc. of the 3th Int. Conf. on Bioinformatics and Genome Research, 1994.

[Graves et al, 1995a] Graves, M., Bergeman, E. R., and Lawrence, C. B., "*A Graph-Theoretic Data Model for Genome Mapping Databases*", In Proc. of the 28th Hawaii Int. Conf. on System Sciences (HICSS), IEEE Computer Society, 32, 1995a.

[Graves et al, 1995b] Graves, M., Bergeman, E. R., and Lawrence, C. B., "*Graph Database Systems for Genomics*", IEEE Engineering in Medicine and Biology, Special issue on Managing Data for the Human Genome Project 11, 6, 1995b.

[Graves, 1993] Graves, M, "*Theories and Tools for Designing Application-Specific Knowledge Base Data Models*", PhD thesis - University of Michigan, 1993

[Greenwood, 2012] Eric Greenwood, "*Storage Models and their Most Glaring Vulnerabilities*", Tweak and Trick, http://www.tweakandtrick.com/2011/08/data-storage-model-risk.html (accessed: 26.11.2012)

[Guha, 2013] R. V. Guha, "*rdfDB: An RDF Database*", http://www.guha.com/rdfdb/ (accessed: 16.03.2013).

[Guting, 1994] Guting, R. H., "*GraphDB: Modeling and Querying Graphs in Databases*", in: Proc. of 20th, Int. Conf. on Very Large Data Bases (VLDB). Morgan Kaufmann, 1994, pp. 297–308.

[Gyssens et al, 1990] Gyssens, M., Paredaens, J., den Bussche, J. V., and Gucht, D. V. A, "*Graph-Oriented Object Database Model*", in: Proc. of the 9th Symposium on Principles of Database Systems (PODS), ACM Press, 1990, pp. 417–424.

[Harris & Gibbins, 2003] Harris S, Gibbins N., "*3store: Efficient bulk RDF storage*", in: Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems, PSSS 2003, Sanibel, and Island, FL, USA, 2003.

[Harris et al, 2009] Steve Harris, Nick Lamb, and Nigel Shadbolt, "*4store: The design and implementation of a clustered RDF store*", In SSWS2009: Proceedings of the 5th International Workshop on Scalable Semantic Web Knowledge Base Systems, 2009.

[Hayes & Gutierrez, 2004] Hayes, J. and Gutierrez, C., "*Bipartite Graphs as Intermediate Model for RDF*", in: Proc. of the 3th Int. Semantic Web Conference (ISWC), Number 3298 in LNCS, Springer-Verlag, 2004, pp. 47–61.

[Hayes, 2004] Hayes P., "*RDF Semantics*", W3C Recommendation, ed., 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-mt-20040210/; Latest version available at http://www.w3.org/TR/rdf-mt/ (accessed: 28.08.2012)

[Hertel et al, 2009] Hertel A., J. Broekstra, and H. Stuckenschmidt, "*RDF Storage and Retrieval Systems*", In: S. Staab and R. Studer (eds.), Handbook on Ontologies, International Handbooks on Information Systems, DOI 10.1007/978-3-540-92673-3, Springer-Verlag Berlin Heidelberg 2009. pp 489-508.

[Hidders & Paredaens, 1993] Hidders, J. and Paredaens, J., "*GOAL A Graph-Based Object and Association Language*", Advances in Database Systems: Implementations and Applications, CISM, 1993, pp. 247–265.

[Hidders, 2001] Hidders, J., "*A Graph-based Update Language for Object-Oriented Data Models*", PhD thesis in Technische Universiteit, Eindhoven, 2001

[Hidders, 2002] Hidders, J., "*Typing Graph-Manipulation Operations*", In: Proc. of the 9th Int. Conf. on Database Theory (ICDT), Springer-Verlag, 2002, pp. 394–409.

[Inseok et al, 2005] Eugene Inseok Chong, Souripriya Das, George Eadon, Jagannathan Srinivasan, "*An efficient SQL-based RDF querying scheme*", VLDB '05: Proceedings of the 31stinternational conference on Very large data bases, Trondheim, Norway, 2005.

[ISI, 2012] http://www.isi.edu (accessed: 21.07.2012)

[ISP2.0, 2012] Intellidimension Inc., Semantics Platform 2.0. http://www.intellidimension.com/products/semantics-platform/ (accessed: 15.10.2012)

[Ivanova, 2015] Krassimira Ivanova, "Algorithm for Quick Numbering of Large Volumes of Data", International Journal "Information Theories and Applications", Vol. 22, Number 4, 2015, ISSN 1310-0513 (printed), ISSN 1313-0463 (online), pp. 303 - 313.

[Janik & Kochut, 2005] Maciej Janik and Krys Kochut, "*BRAHMS: A WorkBench RDF Store and High Performance Memory System for Semantic Association Discovery*", In Fourth International Semantic Web Conference, 2005.

[Jena2, 2012] Jena2 database interface – database layout, http://jena.sourceforge.net/DB/layout.html (accessed: 22.08.2012)

[Kerschberg et al, 1976] Kerschberg, L., Klug, A. C., and Tsichritzis, D, "*A Taxonomy of Data Models*", In: Proc. of Systems for Large Data Bases (VLDB), North Holland and IFIP, 1976, pp. 43–64.

[Kim, 1990] Kim, W, "*Object-oriented databases: definition and research directions*", IEEE Trans, Knowl. Data Eng. 2, 3, 1990, pp. 327–341.

[Kingsbury & Palmer, 2003] P. Kingsbury, M. Palmer, "*PropBank: the Next Level of the TreeBank*", University of Pennsylvania, Department of Computer and Information Science, 2003, pp. 12, http://w3.msi.vxu.se/~rics/TLT2003/doc/kingsbury_palmer.pdf (accessed: 21.07.2012)

[Klyne & Carroll, 2004] G. Klyne and J. J. Carroll Editors, "*Resource Description Framework (RDF): Concepts and Abstract Syntax*", W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ Latest version available at http://www.w3.org/TR/rdf-concepts/ (accessed: 22.08.2012)

[Kolas et al, 2009] Dave Kolas, Ian Emmons, Mike Dean, "*E_cient Linked-List RDF Indexing*", in Parliament. http://parliament.semwebcentral.org/ISWC2009ParliamentPaper.pdf. See also: http://parliament.semwebcentral.org/ (accessed: 23.03.2013)

[Kunii, 1987] Kunii, H. S., “*DBMS with Graph Data Model for Knowledge Handling*”, In Proc. of the 1987 Fall Joint Computer Conference on exploring technology: today and tomorrow, IEEE Computer Society Press, 1987, pp. 138–142.

[Kuper & Vardi, 1984] Kuper, G. M. and Vardi, M. Y., “*A New Approach to Database Logic*”, In: Proc. of the 3th Symposium on Principles of Database Systems (PODS), ACM Press, 1984, pp. 86 96.

[Kuper & Vardi, 1993] Kuper, G. M. and Vardi, M. Y., “*The Logical Data Model*”, ACM Transactions on Database Systems (TODS) 18, 3, 1993, pp. 379–413.

[Levene & Loizou, 1995] Levene, M. and Loizou, G., “*A Graph-Based Data Model and its Ramifications*”, IEEE Transactions on Knowledge and Data Engineering (TKDE) 7, 5, 1995, pp. 809–823.

[Levene & Poulovassilis, 1990] Levene, M. and Poulovassilis, A., “*The Hypernode Model and its Associated Query Language*”, In: Proc. of the 5th Jerusalem Conf. on Information technology. IEEE Computer Society Press, 1990, pp. 520–530.

[Levene & Poulovassilis, 1991] Levene, M. and Poulovassilis, A., “*An Object-Oriented Data Model Formalised Through Hypergraphs*”, Data & Knowledge Engineering, (DKE) 6, 3, 1991, pp. 205 - 224.

[Magkanaraki et al, 2002] Magkanaraki A., G. Karvounarakis, Ta Tuan Anh, V. Christophides, D. Plexousakis, “*Ontology Storage And Querying, Technical Report*”, No 308, Foundation for Research and Technology, Hellas Institute of Computer Science, Information Systems Laboratory, April 2002. http://xml.coverpages.org/MagkanarakiOnt.pdf (accessed: 15.10.2012)

[Mainguenaud, 1992] Mainguenaud, M., “*Simatic XT: A Data Model to Deal with Multi-scaled Networks*”, Computer, Environment and Urban Systems 16, 1992, pp. 281–288

[Mano, 1993] M. Morris Mano, “*Computer System Architecture*”, Third edition. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, USA, ISBN 0-13-175563-3, 1993, 530 p.

[Markov et al, 2008] Markov, K., Ivanova, K., Mitov, I., & Karastanev, S., “Advance *of the access methods*”, International Journal of Information Technologies and Knowledge, 2(2), 2008, pp. 123–135.

[Markov et al, 2013] Markov, Krassimir, Koen Vanhoof, Iliya Mitov, Benoit Depaire, Krassimira Ivanova, Vitalii Velychko and Victor Gladun, "*Intelligent Data Processing Based on Multi-Dimensional Numbered Memory Structures*", Diagnostic Test Approaches to Machine Learning and Commonsense Reasoning Systems, IGI Global, 2013, pp. 156-184, doi:10.4018/978-1-4666-1900-5.ch007, ISBN: 978 1-4666-1900-5, EISBN: 978-1-4666-1901-2
Reprinted in: Markov, Krassimir, Koen Vanhoof, Iliya Mitov, Benoit Depaire, Krassimira Ivanova, Vitalii Velychko and Victor Gladun, "*Intelligent Data Processing Based on Multi-Dimensional Numbered Memory Structures*", Data Mining: Concepts, Methodologies, Tools, and Applications, IGI Global, 2013, pp. 445-473, doi:10.4018/978-1-4666-2455-9.ch022, ISBN13: 978-1-4666-2455-9, EISBN13: 978-1-4666-2456-6

[Martin, 1975] J. Martin, „*Computer Data-Base Organization*”, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[Masolo et al, 2003] Masolo C., Borgo S., Gangemi A., Guarino N., Oltramari A., “*WonderWeb Deliverable D18: Ontology Library (final)*”, Laboratory for Applied Ontology – ISTC–CNR, 2003, pp. 349, http://www.loa-cnr.it/Papers/D18.pdf (accessed: 21.07.2012)

[Matono et al, 2007] Akiyoshi Matono, Said Mirza Pahlevi, Isao Kojima, “*RDFCube: A P2P-Based Three- Dimensional Index for Structural Joins on Distributed Triple Stores*”, SpringerLink – Book Chapter Databases, Information Systems, and Peer-to-Peer Computing, 2007.

[McBride, 2001] McBride B., “*Jena: Implementing the RDF Model and Syntax Specification*”, In: Steffen Staab et al (eds.), Proc. of the Second International Workshop on the Semantic Web-SemWeb2001, May 2001, http://ceur-ws.org/Vol-40/mcbride.pdf, Jena URL:http://www.hpl.hp.com/semweb/jena-top.html (accessed: 15.10.2012)

[McGlothlin & Khan, 2009] James P. McGlothlin, Latifur R. Khan “RDFJoin: A Scalable of Data Model for Persistence and Efficient Querying of RDF Datasets”, UTDCS-08-09, 2009.

[McGlothlin & Khan, 2009a] James P. McGlothlin, Latifur R. Khan, “*RDFKB: efficient support for RDF inference queries and knowledge management*”, IDEAS ’09: Proceedings of the 2009 International Database Engineering, Applications Symposium, Cetraro - Calabria, Italy, 2009.

[Mell & Grance, 2011] Peter Mell, Timothy Grance, “*The NIST Definition of Cloud Computing*”, NIST Special Publication 800-145, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899-8930, September 2011.

[Mendelzon et al, 2001] Alberto Mendelzon, Thomas Schwentick, Dan Suciu, “*Foundations of Semistructured Data*”, 2001, http://www.dagstuhl.de/Reports/01/01361.pdf (accessed: 20.07.2013).

[Mikr, 2012] Mikrokosmos http://www.ilc.cnr.it/EAGLES96/rep2/node23.html (accessed: 21.07.2012).

[Miller, 1995] Miller G. A., “*WordNet: a lexical database for English*”, G. A. Miller – Communications of the ACM 38: 11, 1995, pp. 39–41

[Moënne-Loccoz, 2005] Moënne-Loccoz, N., “*High-dimensional access methods for efficient similarity queries*”, Technical Report N: 0505, University of Geneva, Computer Vision and Multimedia Laboratory, 2005.

[Mokbel et al, 2003] Mokbel, M., Ghanem, T., & Aref, “*Spatio-temporal access methods*”, A Quarterly Bulletin of the Computer Society of the IEEE Technical Committee on Data Engineering, 26(2), 2003, pp. 40–49.

[Muys, 2007] Andrae Muys, “*Building an Enterprise Scale Database for RDF Data*”, Seminar, Netymon, 2007.

[Navathe, 1992] Navathe, S. B., “*Evolution of Data Modeling for Databases*”, Communications of the ACM 35, 9, 1992, pp. 112–123.

[Neumann & Weikum, 2008] Thomas Neumann, Gerhard Weikum, “*RDF-3X: a RISC-style Engine for RDF*”, JDMR (formely Proc. VLDB) 2008, Auckland, New Zealand, http://www.mpi-inf.mpg.de/~neumann/rdf3x/, https://domino.mpi-inf.mpg.de/intranet/ag5/ag5publ.nsf/AuthorEditorIndividualView/ad3dbafa6fb90dd2c1257593002ff3df/$FILE/rdf3x.pdf?OpenElement (accessed: 23.03.2013).

[Oldakowski et al, 2005] Oldakowski R, Bizer C, Westphal D., “*RAP RDF API for PHP*”, In: Proceedings of Workshop on Scripting for the Semantic Web, SFSW 2005, at 2nd European Semantic Web Conference, ESWC 2005, Heraklion, Greece.

[Ontopia, 2012] “*The Ontopia Knowledge Suite: An introduction*”, White Paper (V. 1.3), 2002 http://www.regnet.org/members/demo/ontopia/doc/misc/atlas-tech.html; URL: http://www.ontopia.net/solutions/products.html (accessed: 15.10.2012)

[Ooi et al, 1993] Ooi B., Sacks-Davis R., Han J, “*Indexing in spatial databases*”, Technical Report, 1993

[OpenCyc, 2012] OpenCyc Documentation http://www.opencyc.org/doc (accessed: 21.07.2012)

[oracledb, 2012] http://www.oracle.com/technetwork/database/options/semantic-tech/index.html (accessed: 11.08.2012)

[Owens, 2009] Alisdair Owens, "*An Investigation into Improving RDF Store Performance an Investigation into Improving RDF Store Performance*", Ph.D. Thesis - University of Southampton, 2009.

[Pan & Heflin, 2004] Pan Z, Heflin J., "*DLDB: Extending relational databases to support Semantic Web queries*", Technical Report LU-CSE-04-006, Department of Computer Science and Engineering, Lehigh University, 2004.

[Papakonstantinou et al, 1995] Papakonstantinou, Y., Garcia-Molina, H., and Widom, J., "*Object Exchange across Heterogeneous Information Sources*", In Proc. of the 11th Int. Conf. on Data Engineering (ICDE). IEEE Computer Society, 1995, pp. 251–260.

[Paredaens et al, 1995] Paredaens, J., Peelman, P., and Tanca, L., "*G-Log: A Graph-Based Query Language*", IEEE Transactions on Knowledge and Data Engineering (TKDE) 7, 3, 1995, pp. 436–453.

[Peckham & Maryanski, 1988] Peckham, J. and Maryanski, F. J, "*Semantic data models*", ACM Comput. Surv., 20, 3, 1988, pp. 153–189

[Philpot et al, 2005] Philpot A., E. Hovy, P. Pantel, "*The Omega Ontology*", Information Sciences Institute of University of Southern California, 2005. pp. 8 http://omega.isi.edu/doc/ (accessed: 21.07.2012)

[Poulovassilis & Levene, 1994] Poulovassilis, A. and Levene, M., "*A Nested-Graph Model for the Representation and Manipulation of Complex Objects*", ACM Transactions on Information Systems (TOIS) 12, 1, 1994, pp. 35–68.

[Ravenbrook, 2010] Ravenbrook, Software engineering consultancy, 2010 Retrieved from http://www.ravenbrook.com/ (accessed: 16.11.2012)

[RDF, 2013] http://www.w3.org/RDF/#specs (accessed: 21.02.2013).

[RDFStore, 2012] RDFStore URL: http://rdfstore.sourceforge.net/documentation/api.html (accessed: 15.10.2012)

[Sesame, 2012] Sesame, OpenRDF, http://www.openrdf.org/index.jsp http://www.openrdf.org/doc/sesame2/2.3.2/users/userguide.html#chapter-sesame2-whats-new (accessed: 01.12.2012)

[Silberschatz et al, 1996] Silberschatz, A., Korth, H. F., and Sudarshan, S. "*Data Models*", ACM Computing Surveys 28, 1, 1996, pp. 105–108.

[Sintek & Decker, 2001] Sintek M., S. Decker, "*TRIPLE-An RDF Query, Inference, and Transformation Language*", In: Proceedings of the Deductive Databases and Knowledge Management Workshop (DDLP' 2001), Japan, October 2001, TRIPLE URL: http://triple.semanticweb.org/ (accessed: 15.10.2012)

[Stably, 1970] Stably D., "*Logical Programming with System*", 360, New York, 1970

[SUMO, 2012] Suggested Upper Merged Ontology (SUMO). http://www.ontologyportal.org/ (accessed: 23.07.2012)

[Taylor & Frank, 1976] Taylor, R. W. and Frank, R. L., "*CODASYL data-base management systems*", ACM Comput. Surv., 8, 1, 1976, pp. 67–103

[Tran et al, 2009] Thanh Tran, Gunter Ladwig, Sebastian Rudolph "*iStore: Efficient RDF Data Management Using Structure Indexes for General graph Structured Data*", Institute AIFB, Karlsruhe Institute of Technology, 2009.

[Tsichritzis & Lochovsky, 1976] Tsichritzis, D. C. and Lochovsky, F. H., "*Hierarchical data-base management: A survey*", ACMComput. Surv. 8, 1, 1976, pp. 105–123.

[Weiss et al, 2008] Weiss, C, Karras, P., Bernstein, A., "*Hexastore: Sextuple Indexing for Semantic Web Data Management*", In: 34th Intl Conf. on Very Large Data Bases (VLDB), Auckland, New Zealand, 28 August 2008, http://www.zora.uzh.ch/8938/2/hexastore.pdf (accessed: 23.03.2013).

[Wilkinson et al, 2003] Kevin Wilkinson, Craig Sayers, Harumi Kuno, Dave Reynolds, "*Efficient RDF Storage and Retrieval in Jena2*", SWDB, 2003.

[Wilkinson, 2006] Kevin Wilkinson, "*Jena Property Table Implementation*", HP Labs, 2006.

[Wood et al, 2005] David Wood, Paul Gearon, Tom Adams, "*Kowari: A Platform for Semantic Web Storage and Analysis*", WWW 2005, May 10--14, 2005, Chiba, Japan

[Yongming et al, 2012] Yongming L., F. Picalausa, G.H.L. Fletcher, J. Hidders, Stijn Vansummeren, "*Chapter 2. Storing and Indexing Massive RDF Data Sets*", In: R. De Virgilio, F. Guerra, Y. Velegrakis (eds), "Semantic Search over the Web". ISBN 978-3-642-25007-1 ISBN 978-3-642-25008-8 (eBook), DOI 10.1007/978-3-642-25008-8. Springer Heidelberg New York Dordrecht London, 2012.

Authors' Information

Krassimira Ivanova *– Assist. prof. Dr.; University of Telecommunications and Posts, Sofia, Bulgaria; Institute of Mathematics and Informatics, BAS, Bulgaria; e-mail: krasy78@mail.bg;*

Основные области научных исследований: Информационные технологии, Многомерные многослойные информационные структуры.

Stefan Karastanev *– Assist. prof.; Institute of Mechanics, BAS, Bulgaria;*

e-mail: stefan@imbm.bas.bg

Основные области научных исследований: Информационные системы и базы данных.

Vitalii Velychko *– Assoc. prof. Dr.; Institute of Cybernetics, NASU, Kiev, Ukraine; e-mail: velychko@aduis.com.ua*

Основные области научных исследований: индуктивный логический вывод, обработка естественно-языковых текстов

Krassimir Markov *– ITHEA ISS IJ, IBS and IRJ Editor in chief, P.O. Box: 775, Sofia-1090, Bulgaria; e-mail: markov@foibg.com*

Major Fields of Scientific Research: General theoretical information research, Multi-dimensional information systems