

DATABASE VERSUS KNOWLEDGEBASE - A COMPARATIVE STUDY

¹E. C. Okafor, ²C. C. Osuagwu

¹*Department of Computer Engineering, Enugu State University of Science and Technology, Enugu, Nigeria*

²*Department of Electronic Engineering, University of Nigeria, Nsukka, Nigeria*
e-mail: ericokafor@cee-esut-ng.org, duruchiosuagwu@yahoo.com

Abstract

Databases and knowledgebases have become ubiquitous as tools for organizing and managing information with the support of knowledgebase and database management systems respectively. An intelligent knowledgebase system has the capacity to acquire, store, retrieve, communicate, process and use knowledge for the purpose of solving problems. A database management system is also designed to assist in maintaining and utilizing large collection of data. However, a knowledgebase system differs by performing and supporting tasks which usually require human expertise to complete. This study introduces descriptions of current databases and knowledgebases, and compares them in terms of structure, design procedure and management, techniques for content acquisition, storage and retrieval mechanisms, and application.

Keywords: *database, knowledgebase, data models, relational database management system, rule-based expert system.*

1.0 Introduction

High information throughput cannot be achieved through techniques that do not carry over from one application to another. Some of the inherent problems are:

- Finding and using data is time consuming and cumbersome, hence information generated is rarely useful.
- Report generation is quite slow.
- Available security options are very limited in scope and impractical to apply.

Computerized data processing is the best technique of overcoming these problems. This trend led to today's database (DB) and knowledgebase (KB).

Simply put, a DB is a collection or repository housing data and metadata in a structured manner that allows for easy access.

A KB should not be seen to be the same. Unlike a DB that describes simple information, a KB is a repository describing complex information that

encapsulates and encompasses DBs. It is a model of human knowledge that is used by expert systems.

In the succeeding pages, we critically compare DBs and KBs in terms of structure, design and management, techniques of content acquisition, storage and retrieval, as well as application.

2.0 General Features

2.1 Database

A DB houses data and metadata where metadata provides descriptions of data characteristics and a set of relationships that link data, that is, it is 'data' about data. Each DB is controlled using a collection of programs that manage the structure and control access to data stored in it. A general name for this set of programs is Database Management System (DBMS), and the functions include:

- Storing definitions of data relationships in a data dictionary, which it uses to control access to data.
- Transforming entered data to conform to specified data structures.

- Implementing/enforcing security and privacy within a DB.
- Allowing for multi-user access to data.
- Providing backup and data recovery.
- Enforcing data integrity.

A data model is a collection of high-level data description constructs that hide many low-level storage details. With the help of DBMS, the user defines data to be stored in terms of data models. Data models in use today include [1, 2, 3]:

- Relational data model – proposed by Edgar Codd in 1970 [4].
- Hierarchical model – based on IBM's Information Management System developed in the late 1960s.
- Network model – introduced by Charles Bachman in 1960s.
- Object-oriented models.

Object-oriented models evolved from increased emphasis on process integration. Initial areas of focus by several object-oriented DB vendors have been in computer-aided design (CAD), computer-aided manufacturing (CAM), computer-aided software engineering (CASE) applications, and factory and office automation. A common denominator to all applications of object-oriented models is the need to manage very complex information efficiently [5,6].

An even more abstract high-level data model that allows user to provide good descriptions of data, especially in a real application scenario is the *semantic data model*. A good example of semantic data model is the entity-relationship (ER) model. This model provides a means of pictorially describing data involved in a real world enterprise. The descriptions are presented in terms of objects (entities) and their relationships. ER is very useful in developing initial DB designs.

Since most DBMS today are based on a relational data model, discussions in this paper will focus almost entirely on relational DBs.

2.2 Knowledgebase

Knowledgebases are usually domain-specific. A domain here refers to the subject area or field of specialization which is explicitly included within the scope of discussion. KB is a repository (in a computer) of an expert's knowledge in a given domain. The repository includes rules, assertions, procedures, descriptions and other knowledge structures that represent domain knowledge. When presented with a problem, a computer manipulates this knowledge repository to solve the problem in the same way an expert would have. Essentially, the computer mimics the reasoning process and in the end, effectively replaces the expert.

The *inference engine* controls the interaction between a user and a KB. The combination of a KB and an inference engine which processes it, gives a knowledgebase system (KBS). KB technology offers an appealing alternative to flat files and relational data models.

KB models exist as high-level templates for expert knowledge. These models serve to highlight the main points that are often obscured. They may also partition a large KB into smaller, easier-to-verify pieces. For instance, [7] proposed a design domain KB model structured as a hierarchical tree with classes representing leaves. The virtual class, which represents domains, appears at the top level of the structure, while subclasses define the domain classes. The subclasses are composed of objects defined with attributes. There are mathematical properties of a knowledge model that help establish the correctness of a KB.

Generally speaking, KB as a data store involves three types of meta-information, or information about the structure of information. First, is a *hierarchical taxonomy or classification of concepts* in the domain of discourse. The most general concepts are at the top, with specializations fanning out as we move down the tree. These relations allow child

concepts to inherit properties from their parents.

Next in the order are the *relations between concepts*. These are essentially attributes whose values are constrained to be other concepts in the taxonomy, thus creating a network of concepts. The final type of meta-information associated with KBs is *instances*. These are concepts that fall at the leaves of a hierarchical tree, and usually represent real-world objects that have values associated with all their attributes. Very often, actual data is represented as instances, but their semantics are encoded by a conceptual network. T

Together, these three types of meta-information produce a KB. Then, the ontology becomes the concepts and relations available for modeling data. In organizational memory systems, concepts and relations usually have explicit semantics; hence higher levels of analysis may be used to find new connections between concepts and new relationships between people. This process essentially creates new (organizational) knowledge [8,9].

3.0 Structure

3.1 Relational Database Management System (RDBMS)

The initial relational database model may be traced to papers by Codd [1,10,11]. This model is implemented through a very sophisticated Relational Database Management System (RDBMS). Because of this, an RDBMS enables the user/designer operate in a human logical environment while it takes care of all the complex physical details. A relational database is perceived by the user to be a collection of tables in which data should be; however, how they are physically stored is of no concern to the user or designer. From Figure 1, three levels of abstraction are usually used to describe data in a DBMS – the *physical*, *logical* and *external schemas* [12, 13]. A schema is a description of data in terms of a data

model. Use of data models in the description of real world data is illustrated by [14]. Data definition language (DDL) is used in defining the conceptual and external schemas. An external schema consists of a collection of one or more *views* and relations from the logical schema. The records in a *view* are not stored in the DBMS. The external schema is designed just to suit end-user requirements.

The physical schema specifies additional storage details, especially how relations described in the logical (or conceptual) schema are stored on a storage media.

The physical DB architecture consists of **data files** and **transaction log files**.

Data Files are of two types: primary data file and secondary data file.

- **Primary Data File:** stores data in DB objects such as tables and indexes. It contains information about system tables, objects and location of additional files e.g. secondary data files and transaction log files. Only one primary data file can exist in a DB.
- **Secondary Data File:** stores data in user-defined tables. There can be several in a DB.

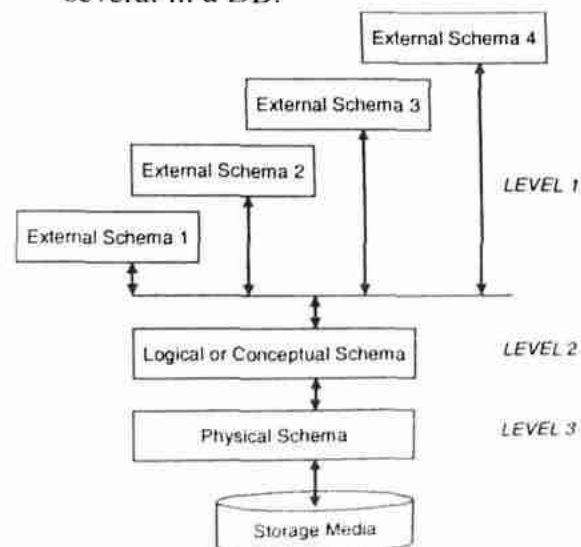


Figure 1: Levels of abstraction in a DBMS

Transaction Log Files record the transactions or modifications that are performed within a DB and which operations made them. Each DB must have

at least one transaction log file and they can be spread over several storage media. A transaction log file also contains information about file identification, file size, file growth and information which can be used to recover the DB in case of a crash or malfunction. Each data file and transaction log file has a user-defined logical name which can be used for reference it.

Logical (or conceptual) schema describes stored data in terms of the data model supported by the DBMS. For a relational database model, a logical schema describes the relations stored in the DB. A typical DBMS based on this model is shown in Figure 2. The DBMS accepts SQL commands from a variety of sources (or the user interfaces) and produces query evaluation plans using the query evaluation engine. These plans are executed against the DB and results returned to user. In response to the user requests, the buffer manager retrieves pages from disk into main memory. DBMS components also exist for concurrency control and crash recovery, and are based on careful scheduling of these user requests and maintaining of log of all changes to the DB.

One of the reasons for the rise to dominance of relational database model is its powerful and flexible query capability. RDBMSs use query language (e.g., Structured Query Language, SQL) to take care of translating user request into the technical details required to retrieve requested data.

SQL-based relational DB software involves three parts: a *user interface*, a *set of tables* within the DB and an *SQL engine*.

- The *user interface* may include menus, query operations, report generators and so on. It allows a user interact with the data.
- The *data tables*, or *data files*, simply store data.

- The *SQL engine*, largely hidden from a user, does the tough DB jobs. It is responsible for creating schema structures, maintaining data dictionary and system catalog, controlling DB table access and maintenance, and translating user requests into a format that the computer can handle.

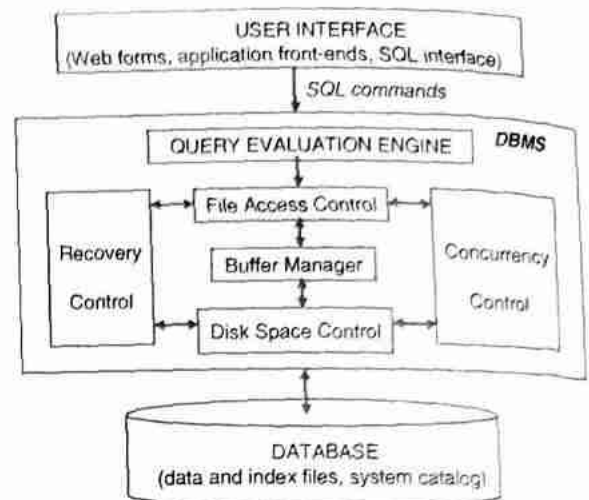


Figure 2: Structure of a DBMS based on the relational database model

3.2 Knowledgebase System (KBS)

KB is a repository of human knowledge that is used by KBSs. KBS has been touted as a better alternative to flat-file and relational DB because of the following unique features:

- It can store collection of knowledge and manipulate it in response to use enquiries. In some cases, it can explain responses to the user.
- It can cope with data, which may be uncertain or partial (i.e., incomplete).
- It can come to uncertain, and where necessary, multiple conclusions when there is no definite answer.
- It can generally decide on the next step to be taken in a problem solving mission.

Most KBSs are rule-based expert systems. A *rule-base* is a collection or chain of knowledge that is represented in the form of IF-THEN rules. A rule-based KBS has a large number of interconnected and nested

IF-THEN statements or rules that are the basis for the knowledge in the system.

A rule-based system is composed of three parts: the *data level*, the *control level* and the *knowledge level*.

- **Data Level:** refers to the human-computer interface or user interface technology which allows users to interact with a system. This interaction is either completely controlled by the system, which uses prompts to request user input, or by a user able to enter any number of inputs in a natural language form. Occasionally, control is shared between user and system.
- **Control Level:** refers to the inference engine, which is the active component of an expert system that reasons and determines how to apply the knowledge in the KB to the facts and premises presented at the user interface. It attempts to deduce new facts which are then used to draw further conclusions. Its aim is to mimic human reasoning so that a user can understand why the system has chosen certain steps. There are various inferencing techniques which model different reasoning methods:
 - **Forward Chaining:** a strategy for searching the rule base in an expert system that begins with the information entered by the user and searches the rule base to arrive at a conclusion.
 - **Backward Chaining:** is a strategy for searching the rule base in an expert system that acts like a problem solver by beginning with a hypothesis and seeking more information to derive its initial conditions.
 - **Mixed Chaining:** involves using forward and backward chaining strategies at the same time.
- **Knowledge Level:** refers to the KB. It is the most important element of an expert system since it holds the experts' problem-solving knowledge. It is where knowledge harvested from experts is stored. The separation of

control level from knowledge level implies modularity (in the system), and makes it easier to manipulate knowledge without altering the control.

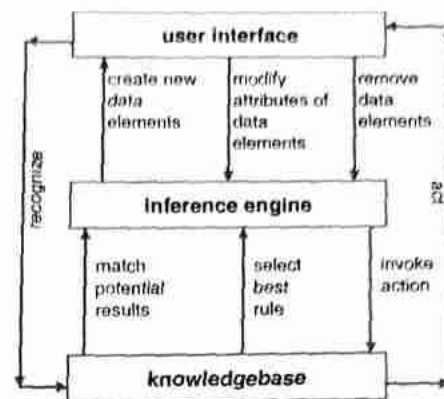


Figure 3: Organization of a rule-based expert system

Rules and other structures for representing knowledge (semantic networks, frames, logic and sets) are illustrated by [15].

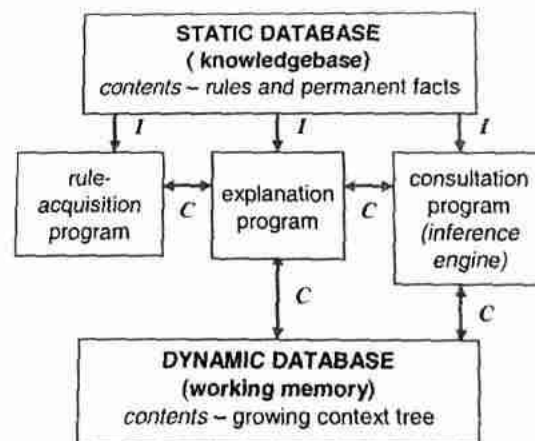


Figure 4: Organization of the MYCIN expert system (I - information; C - control).

Knowledge acquired from an expert has to be represented formally, that is, structured in such a way that it allows for manipulation to yield additional data and knowledge. The power of a system tends to be related to the depth and breadth of the knowledge in the KB. Figure 3 presents the structure of a KBS and Figure 4 shows the organization of MYCIN – the first functional expert system.

4.0 Design Procedure

Management techniques, personnel, and various stages of the design process for RDBMS and KBS will be discussed.

4.1 RDBMS

To effectively manage a relational DB, especially in a distributed environment, the services of *system administrator* (SA) and a *database administrator* (DBA) are required.

The DBA is responsible for the following:

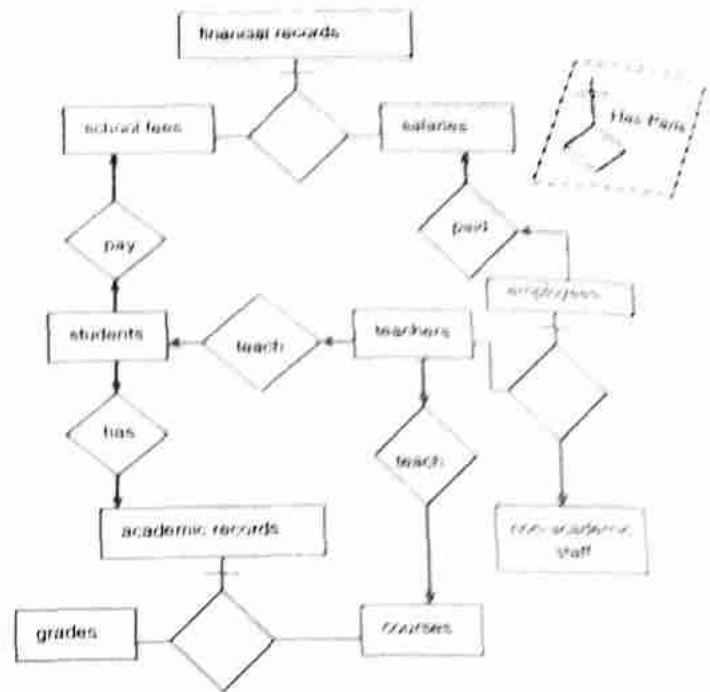
- Designing the logical and physical DB structure using any available RDBMS product such as Microsoft SQL Server, Oracle, IBM DB2.
- Creating DB users and assigning permissions and usage rights to them.
- Implementing security for the DB.
- Deploying the DB for use.

The SA is charged with the following responsibilities:

- Evaluating DB design.
- Maintaining the DB software after deployment.
- Assigning right to use the DB to selected individuals.
- Coordinating the development of applications based on the data source.

The design process for a DB usually involves four stages: *problem definition*, *modeling*, *normalization* and *implementation*.

- **Problem definition** - Refers to the process of determining what the requirements or specifications or functions of a DB will be. The DBA identifies the problem domain and individual objects. For example, in a secondary school DB, an initial study by the DBA may likely unearth the following unique entities or objects: students, employees (teachers and other staff), academic records (courses, grades), financial records (salaries, school fees).



for the tables. From our example, assume the DBA has put forward

Figure 6 as the structure of some of these tables.

STUDENT		EMPLOYEE		COUNSELIST	
NAME	Character(50)	NAME	Character(50)	COURSE TITLE	Character(20)
AGE	Numeric	EMPID	Character(10)	TEACHER	Character(50)
REGNO	Character(10)	ADDRESS	Character(50)	COURSE CODE	Character(16)
ADDR	Character(50)	SALARY	Numeric	CLASS OFFERED	Numeric

Figure 6: Structure of the some of the tables for the hypothetical secondary school database.

- **Implementation** – This involves creating DB objects using SQL. When the SQL codes are executed, the RDBMS creates the underlying physical structures which will store data. SQL codes that will create the tables of Figure 6 are presented in Figure 7.

```
create table STUDENT( name varchar2(50), age number(2), regno varchar2(10), address varchar2(50));
create table EMPLOYEE( name varchar2(50), empid varchar2(10), address varchar2(50), salary number(8,2));
create table COUNSELIST(title varchar2(20),teacher varchar2(50),code varchar2(16), class_offered
number(2));
```

Figure 7: SQL codes implementing tables of Figure 6.

4.2 KBS

To design and manage a KBS, the services of a knowledge engineer is required. Knowledge engineering is the competence or know-how a person requires before he or she can effectively represent human expert knowledge in computer memory. A knowledge engineer is a specialist who elicits or harvests expert knowledge from professionals (and/or other sources) and translates same into a repository of rules or frames for an expert system.

The crucial aspects of an expert system design process are the *acquisition*, *representation* and *utilization* of the expertise. Crafting an expert system involves three stages:

- **Stage 1** - Specification of the problem domain and selection of experts with respect to that domain. In the design of an expert system for the diagnosis of a disease, this first stage involves identifying the disease and selecting medical personnel with expert knowledge in the domain of interest.
- **Stage 2** - Acquiring the required knowledge from experts and expressing it in a form suitable for

translation into a program. Here, the knowledge engineer uses different techniques to elicit the required knowledge from selected experts. Broadly speaking, two methods of eliciting knowledge from experts are *handcrafting* and *induction*. These are treated later.

- **Stage 3** – Translating acquired expert knowledge into suitable KBS using appropriate programming language. Popular programming languages used in designing KBSs are LISP, Prolog and Java. New KBS shells can also be developed by tailoring generalized versions to suit specific requirements. KBS shells do exist that can be filled with problem-specific rules derived from experts.

The first two stages fall under the functions of a knowledge engineer while the last is the job of a software engineer.

5.0 Content Acquisition

Data or knowledge acquisition refers to the processes or ways of collecting data to be stored in a DB or KB.

For every DB, there are two primary means of acquiring data.

- The **first method** involves the use of independent or third-party software such as a management information system (MIS) or web-based application to enter data into a DB. The software communicates with the DB via its SQL engine (Figure 2). This method requires that the software be designed by software engineers using SQL driver to map SQL commands to a format the SQL engine can understand.
- The **second method** involves entering data directly into the DB. This can be done through the user interface that comes with DB software. This method is usually available to only DBAs or SAs who have unlimited access to a DB.

Knowledge acquisition is the process of obtaining rules of thumb or heuristics from a human expert and storing them in a computer. The two main methods used are *handcrafting* and *induction*.

- In **handcrafting**, the rules of thumb defining a knowledge domain are developed directly based on interviews between a knowledge engineer and domain experts. This method is adopted if:
 - Access to articulate experts is readily available.
 - The domain is large.
 - The knowledge is specific and conclusions are definite.
 - Domain knowledge has already been captured in the form of books, manuals, etc.
- The **induction** method is based on the premise that the application area can be described in terms of examples characterizing the expert's skill in solving problems. Computer-based tools can sometimes be used to induce

the rules from examples supplied by domain experts. This method is adopted if:

- Representative set of example data can be obtained.
- Domain experts are not readily available.
- Experts have difficulty in explaining their reasoning.
- The domain is finite and well-defined.

6.0 Storage

Data storage refers to how acquired data or knowledge is stored logically and physically. In this respect, RDBMS and KBS are alike in that:

- They use the same physical devices like tape, disk drives etc.
- They have random access to stored information, that is, data or knowledge can be accessed irrespective of when it was stored or where it is located on the drive.
- Both use relational DB as their backend. For a KBS, a static DB is used to hold the rules defining the domain and other permanent facts.

7.0 Content Retrieval

The major point of interaction between user and data store is the user interface. This interface might be local or remote in that it can reside in the same system as the KBS or RDBMS, or in a different computer that connects to the data store via a network. Such user interfaces might be application software (like a spreadsheet program) or web-based software (like an online store or a search engine).

Embedded between the user interface and data store is an engine which controls interaction between them. Because all possible interactions and requests by users must have been predefined by a DBA or application designer using appropriate SQL queries, limitations exist. A user cannot manipulate data as he or she deems fit.

A KBS, on the other hand, uses an *inference engine* to regulate interactions between user and KB. The inference engine determines how to apply knowledge in a KB to facts and premises presented at the user interface. It generates IF-THEN rules to access a KB based on queries or options input by users. This dynamic form of rule generation makes it possible for users to have unlimited number of ways of viewing and manipulating data within a KB.

8.0 Application

KBS duplicates human reasoning in specific areas of problem-solving expertise. They are useful as intelligent assistants and training tools. KBS can also be used in applications requiring qualitative judgment and opportunistic action. The problem-solving tasks performed by such systems are classified into two broad categories [16]:

- Analysis tasks – Characterized by analysis of the behaviour of an object in an application area or the search for an unknown solution. Examples of unknown solution are found in systems on disease or fault diagnosis, prediction of future behaviour, etc.
- Synthesis tasks – Characterized by composition from a set of components or building up from first principles. Typical examples are tasks based on design and planning.

Implementation of analysis-type problem-solving systems is usually easier than synthesis-type systems since expertise associated with such tasks is typically easier to acquire. Current research directions and development determinants for KBs are discussed by [17]. Also, as the world's knowledge repositories become increasingly bigger, a significant number of studies have been devoted to providing effective and efficient strategies for KB management [18,19,20,21,22,23,24,25]. Knowledge management includes all

aspects of creating, acquiring, storing, updating, and sharing knowledge.

DBSs tend to apply to situations requiring procedural and quantitative solutions. For Example, in an MIS, some queries may be predefined during the design stage. The user just fills in parameters for the query system to use in its search for finite solutions.

Current design trend is to integrate KB capability into existing critical DB systems so as to achieve better results.

9.0 Conclusion

The desire to preserve knowledge and information for present and future use led to the development of DBs and KBs. Though both are repositories of sorts, the following summarizes key areas where they differ:

- KB is used to store collection of knowledge for a restricted or specific domain. In other words, its contents cannot exceed the scope of its domain. A DB holds data which may span multiple areas of interest. For example, a KB designed for medical data about ribosome will hold knowledge specific to this domain. On the contrary, DB designed for school management generally holds information about the students' academic records but may hold that of staff payroll, which is not a related field.
- KB is a repository that holds knowledge elicited from experts in an area of specialization. DB does not necessarily hold data acquired from experts, as processes or users can contribute data to it.
- KB systems can manipulate stored knowledge to solve problems in response to user enquiries, and in some cases, can explain responses to the user. This is possible because of the ability of its inference engine to dynamically generate (IF-THEN) rules in response to user requests. A DB cannot manipulate its data except in

response to queries issued to it in a DB query language.

- Generating the contents of a KB requires the services of trained personnel - the knowledge engineer. Only he has the know-how to elicit and represent knowledge in a manner suitable for storage. But the contents of a DB can be generated by user(s) via an appropriate user interface.

References

- [1] Vossen, G. *Datamodel, database languages and database management systems*, Addison-Wesley, 1991.
- [2] Elmasri, R., & Navathe, S. *Fundamentals of database systems* (2nd Ed.), Benjamin-Cummings, 1994
- [3] Kerschberg, L., Klug, A., & Tsichritzis, D. *A taxonomy of models*. In: Lockemann, P. C., Neuhold, E. J. (Eds.), *Systems for large databases*, North-Holland, 1977.
- [4] Codd, E. F. *A relational model of data for large shared data banks*. Communications of ACM, 13(6), pp. 377-387, 1970.
- [5] Hellerstein, Joseph M. *Object-database systems*. In: Ramakrishnan R., Gehrke, J., (Eds.), *Database Management Systems*, McGraw-Hill Comp. Inc., Boston, pp. 736-776, 2000.
- [6] Pavkoviæ, N., Štorga, M., & Pavliæ, D. *Two examples of database structures in management of engineering data*. Proc. of 12th International Conference on Design Tools and Methods in Industrial Engineering; Bologna: ADM-Associazione Nazionale Disegno Machine, 89-90, 2001.
- [7] Dekoviæ D., & Bojèetiæ N. *Structure of Design Domain Knowledgebase*. Proc. of the 12th International Conference on Engineering Design (ICED 99), Munich, vol 3, pp. 1401-1406, 1999.
- [8] Davey, B., & Tatnall, A. *Misinforming knowledge through ontology*. Issues in Informing Science and Information Technology, 1, 637-643, 2004.
- [9] Alani, H., Kalfoglou, Y., O'Hara, K., & Shadbolt, N. *Initiating organizational memories using ontology-based network analysis as a bootstrapping tool*. Expert Update, 5(3), 43-53, 2002.
- [10] Codd, E. F. (1972a). *Further normalization of database relational models*. In: Rustin, R (Ed.). *Database systems*, Prentice-Hall, Englewood Cliffs, NJ, 1972.
- [11] Codd, E. F. *Relational completeness of data base sub-languages*. In: Rustin, R (Ed.). *Database systems*, Prentice-Hall, Englewood Cliffs, NJ, 1972.
- [12] Steel, T. (1975). *Interim Report of the ANSI-SPARC Study Group*. In: Proc. ACM SIGMOD conference on the management of data, 1975.
- [13] Ramakrishnan R., & Gehrke, J. (2000). *Database management systems*, McGraw-Hill Comp. Inc. 2000
- [14] Kent, W. *Data and reality, basic assumptions in data processing reconsidered*, North-Holland, 1978.
- [15] Giarrantano, J. C. *Expert systems: principles and programming*, PWS-Kent Publishing Coy, 1998.
- [16] Land, L. *Overview of knowledge-based systems*. In: Land, L. (Ed.). *Knowledge-based systems usage*, McGraw-Hill Book Coy., London, pp. 8-17, 1995
- [17] Owoc, M. L. *Knowledgebases - A management context and development determinants*. Proc. of Informing Science + Information Technology Education Joint Conference, Pori, 1193-1199, 2003.
- [18] Drehev, H. *Hypertext and managing knowledge*. Proc. of Informing Science + Information Technology Education Joint Conference, Pori, 329-339, 2003.
- [19] Hsia, T. L., Lin, L. M., Wu, J. H., & Tsia, H. T. *A framework for designing nursing knowledge management systems*. Interdisciplinary Journal of

- Information, Knowledge and Management, 1, 14-22, 2006.
- [20] Viehland, D. *ISExperNet: Facilitating knowledge sharing in the information systems academic community*. Issues in Informing Science and Information Technology, 2, 441-450, 2005.
- [21] Folorunso, O., & Ogunde, A. O. *Data mining as a technique for knowledge management in business process redesign*. The Electronic Journal of Knowledge Management, 2(1), 33-44, 2004.
- [22] Pandya, K. V., & Xiaoming, C. *Issues of knowledge management in the public sector*. The Electronic Journal of Knowledge Management, 1(2), 25-33, 2003.
- [23] Huttenegger, G. *Knowledge management system building blocks*. The Electronic Journal of Knowledge Management, 1(2), 65-76, 2003.
- [24] Nifco, N. *A conceptualization of knowledge management practices through knowledge, awareness and meaning*. The Electronic Journal of Knowledge Management, 3(1), 45-52, 2005.
- [25] Kane, H., Ragsdell, G., & Oppenheim, C. *Knowledge management methodologies*. The Electronic Journal of Knowledge Management, 4(2), 141-152, 2006.