2006-1829: LEVELS OF ABSTRACTION IN DATABASE QUERY DEFINITION

Fani Zlatarova, Elizabethtown College
Associate Professor of CS, CS Department, Elizabethtown, PA 17022

Levels of Abstraction in Database Query Definition

Abstract

One of the most challenging steps in the database design and processing is the definition of queries. The planning and analysis of systems development are critical for the entire creation process. However, real computer-oriented aspects emerge in the design phase. Building up the optimum structure of an information system will determine the quality of its implementation. An important role is assigned to the database query definition, because answering queries is the final goal of an information system. Teaching different levels of abstractions and the respective strategies for query definition, such as: natural language descriptions, relational algebra expressions, *QBE* relational language expressions, *SQL* language expressions, query optimization tools, and miscellaneous recent query applications, would prepare our students for efficient and effective management of information systems.

1. Introduction

Currently, using information systems is part of the daily activities of the global economic, social, and political environments. Many academic institutions introduce new graduate interdisciplinary programs involving a significant amount of information systems-related issues. The number of information systems users increases every day.

How do we know if our students are well prepared to develop software applications meeting the required standards? Do they have the right knowledge and skills beyond the technological level? What would be the optimum mix of theoretical and practical material for them?

Curricula designed for engineering students include courses related to the development, management, and processing of database (DB) systems and information systems (ISs). Query definition and processing is one of the most important topics considered in such courses. The entire organization of an IS is oriented towards the efficient and effective execution of corresponding queries, which extract knowledge from the data contained in the respective DB. The way of processing queries in an IS determines the success or the failure of this system[2]. There is a variety of textbooks[3, 5, 6, 10] offering a detailed appropriate description at different levels of abstraction of query-oriented topics. Impressions obtained during professional academic forums show that college instructors are tempted to adopt textbooks, which do not include important aspects relevant to query definition and processing, issued by leading publishing houses. Usually, such books do not contain the formal description of queries based on the *relational algebra* (*RA*) and all *RA*-related topics. However, this is the alphabet and the heart of the relational data model used in the most popular commercial database management systems (DBMSs). Such books have their value, advantages, and readers in other cases but are not appropriate for an academic auditorium. Science is not equivalent to technology.

The main purpose of the paper is to present a vision about the right approach to teaching query definition by considering different levels of abstraction related to this problem. These levels are related to:

- Natural language descriptions;
- *RA* expressions;
- *QBE* relational language;
- *SQL* language; and
- Query optimization tools.

These topics have been taught in the *Introduction to Database Systems* course offered at the author's academic institution. They do not cover the complete set of views of abstraction. The newly emerging DB applications dealing with Web-oriented queries and multimedia queries (audiovisual queries, special queries, XML queries, queries performed in mobile environment, and others considered in[1, 7, 8, 9, 11, 13]) have been considered in the next DB-oriented course, *Database Systems Development and Applications*. Similar issues have been researched in the *Readings and Projects in Computer Science and Information Systems* course, and in the out-of-class student seminar on *Information Systems*, organized by the author.

In the introductory DB course, students should be able to easily switch from one to another type of the expression opportunities, listed above, and to write optimized queries[4]. This will lead to obtaining competence in understanding the intermediate steps of the query processing and transaction management. The miscellaneous examples, exercises, and projects (especially the service learning projects) offered and assigned to students by using DBMSs, such as: *Oracle*, *Microsoft SQL Server*, *Microsoft Access*, *MySQL*, and appropriate CASE and Web design-oriented tools, help to improve the students' practical skills and create the right vision about their professional ability to develop information systems.

2. Strategies for Query Definition

One of the major criticisms related to the relational model was inadequate performance of queries[3]. Currently, the existence of efficient algorithms for processing queries allows cost effective solutions. However, without possessing the right knowledge and skills in writing queries, it would not be possible to achieve a good performance in the information retrieval process. Students should be acquainted with the multilateral nature of query definition and processing. Only the deep understanding of all aspects related to query creation would allow writing optimized queries. The material taught should be offered in an appropriate sequence. Traditionally, it would be recommended to start with the intuitive level of query description before reaching the more abstract level of definition. Different levels of abstraction in query formulation are needed during the established phases of the systems development life cycle. They correspond to the different groups of people involved in each phase. The user interface used in a specific DBMS environment plays a significant role and imposes the requirement to know how to use it. Being acquainted with the physical characteristics of the specific DB organization would lead to creating cost effective solutions. However, understanding the logical DB organization is critical for the proper query definition. The existing functional dependencies between attributes require a good understanding, consideration of integrity rules, and normalization of relational schemas. All this would have a positive effect not only on the query formulation and processing, but also on the final systems efficiency.

The existing environments and strategies for creating queries corresponding to specific levels of abstraction offer different aspects, complexity, and detailing. Possessing knowledge about the basic levels of abstraction and their analogs in the ISs application area would allow students to formulate equivalent queries (i.e. to transform queries from a specific 4GL environment to another). Two queries, created by using different development strategies, are *equivalent queries* if they produce the same result that could contain permutation of rows and/or columns. Students would also be able to accomplish a comparative analysis about the software used when designing concrete ISs.

The basic strategies for query definition corresponding to the different levels of abstraction are considered below. The examples, described in the following subsections, use a sample logical DB named *MUSIC*, *MUSIC = {C, S, A}, which* contains information about recording companies (*C*), singers (S), and albums issued (*A*), is build up of the following relational schemas:

C (<u>CNo</u>, CName, URL, City);
S (<u>SNo</u>, SName, Address, Phone, Email);
A (<u>ANo</u>, CNo, SNo, AName, AType, Price, Qty, Year).

For simplicity of the DB structure, it is assumed that the DB contains only albums of solo performers. However, it is possible for a singer to have more than one album produced by one company, but a maximum of one album per year.

2.1. Defining Queries by Using Natural Languages

First, the query formulation is performed by using natural languages. This level of representation is primarily oriented toward the end users, non-programmers. The correct formulation of the query is critical when solving the corresponding retrieval problem. Students are advised to disregard the DB terminology, data structures, and data types. They should play the role of the end users and should try to express themselves in a correct clear natural language without ambiguities. An example for such a query would be:

"Find the names of singers collaborating with companies from London that have produced albums of these singers in 2005."

The formulation of the query explains clearly *what* is required, but does not contain details showing *how* to find the corresponding answer.

2.2. Defining Queries as *RA* Expressions

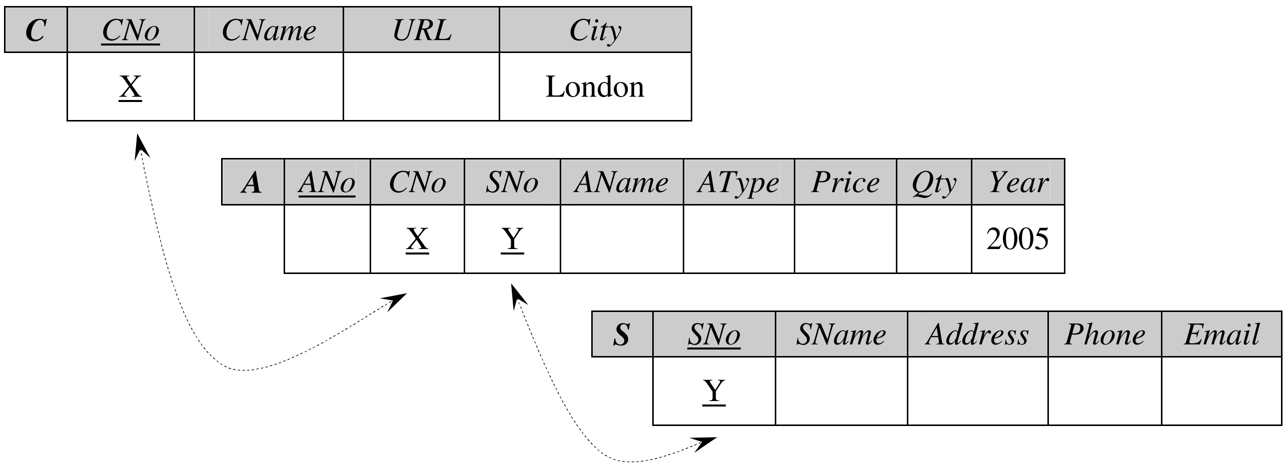
The formal definition of the operations upon tables, in terms of the *RA*, seems to be the most confusing part for a significant number of students. They need the corresponding background that could be obtained from courses such as *Mathematical Calculus* and *Discrete Mathematics*. This does not create difficulties when teaching the DB course at academic institutions offering programs that are more rigid. In this case, selection of elective courses usually occurs after the second year of study. The instructor knows that the necessary courses have been taken by all students. At other academic institutions, instructors cannot rely on the appropriate student preparation neither from the high school, nor from the previous college courses. This could result from the high degree of flexibility in course selection and from the fact that sometimes, curricula do not include the required mathematics-oriented courses. In such cases, students feel confused when they start working with *RA* operators. However, a little bit more effort and time spent by the instructor and a significant number of appropriate exercises would be worthwhile and useful. The discussion of multiple examples and exercises explaining the intermediate results in the query processing is very helpful. Students should easily formulate queries similar to the following query that is equivalent to the query formulated as an example in section 2.1:

$$\pi_{SName}(\, S \underset{SNo}{\bowtie} \sigma_{Year\,=\,2005}(A) \underset{CNo}{\bowtie} \sigma_{City\,=\,\text{'London'}}(C)\,)$$

Using RA operations is important for the detailed understanding of the query nature and prepares students for the next DB-oriented courses. Most of the recent query applications require their formal description. In this way, students would have a good background needed by a possible graduate program. Textbooks that do not include RA issues are not recommended even for introductory DB courses at college level. Working with RA queries develops the logical thinking and provides students with a tool for writing complex queries that extract valuable knowledge from the DB data.

2.3. Defining Queries in *QBE* Relational Language

Teaching the *QBE* environment for query definition requires a small amount of time. However, the respective *QBE* bi-dimensional syntax shows clearly how relationships between tables work. Students learn how to use foreign keys properly.



C	*CNo*	*CName*	*URL*	*City*
	X			London

A	*ANo*	*CNo*	*SNo*	*AName*	*AType*	*Price*	*Qty*	*Year*
		X	Y					2005

S	*SNo*	*SName*	*Address*	*Phone*	*Email*
	Y				

Figure 1. A sample query written in QBE environment

QBE queries are related to their implementation in the *Microsoft Access* DBMS. Because students are usually familiar with the interfaces of *Microsoft Word* and *Microsoft Excel*, teaching them how to use *Microsoft Access* is rather effortless. This provides them with an additional tool for data processing. The relatively low degree of complexity of this application program offers a good starting point in the practical students' experience related to query definition. The existing *SQL* link allows further comparative analysis with more sophisticated DBMSs, such as *Microsoft SQL Server* and *Oracle*. Working with integrated packages similar to *Microsoft Office* could be useful later when teaching advanced *SQL* features. Recent research projects developed by software experts working for Oracle Corporation introduce new features in SQL simulating spreadsheet operations[14, 15].

Textbooks, which do not include introduction to *QBE*, fail to provide students with an easy tool for query definition and an opportunity that enables the better understanding of the DB logical structure.

2.4. Defining Queries in *SQL* Language

The most important practice-oriented environment for query definition is offered by *SQL*, because of its adoption as a standard language for relational DBMSs. Students should be familiar with the syntax and semantics of this language and able to formulate miscellaneous queries such as the two equivalent *SQL* queries written below, similar to the queries considered in sections 2.1-2.3:

```
SELECT SName
FROM   C,S,A
WHERE  C.CNo = A.CNo
AND    S.SNo = A.SNo
AND    Location = 'London'
AND    Year = 2005;
```

or

```
SELECT SName
FROM   S
WHERE  SNo IN
       (SELECT SNo
        FROM   A
        WHERE  Year = 2005
        AND    CNo IN
               (SELECT CNo
                FROM   C
                WHERE  City = 'London'));
```

Practical work with DBMSs, such as *Microsoft SQL Server*, *Oracle*, or *MySQL*, is highly recommended. If students are not acquainted with the levels of abstraction respective to the strategies describes in sections 2.1-2.3, they are not ready to understand the interaction among tables in multi-table queries, for example. They also would experience significant difficulties when writing queries with high complexity and be unprepared to work in a real environment requiring query processing. Their employers would need more time for training purposes and likely, would not be willing to hire such specialists.

2.5. Query Optimization

Teaching query optimization helps students to perform effective and efficient information retrieval. Writing optimized queries requires knowledge and skills related to *RA*. Student should also be familiar with developing RA trees (query graphs) and writing the corresponding RA expressions. The transformation rules and the heuristical processing strategies should also be well known. Fig. 5 shows possible optimization steps of the query considered as an example in the previous sections 2.1-2.4.

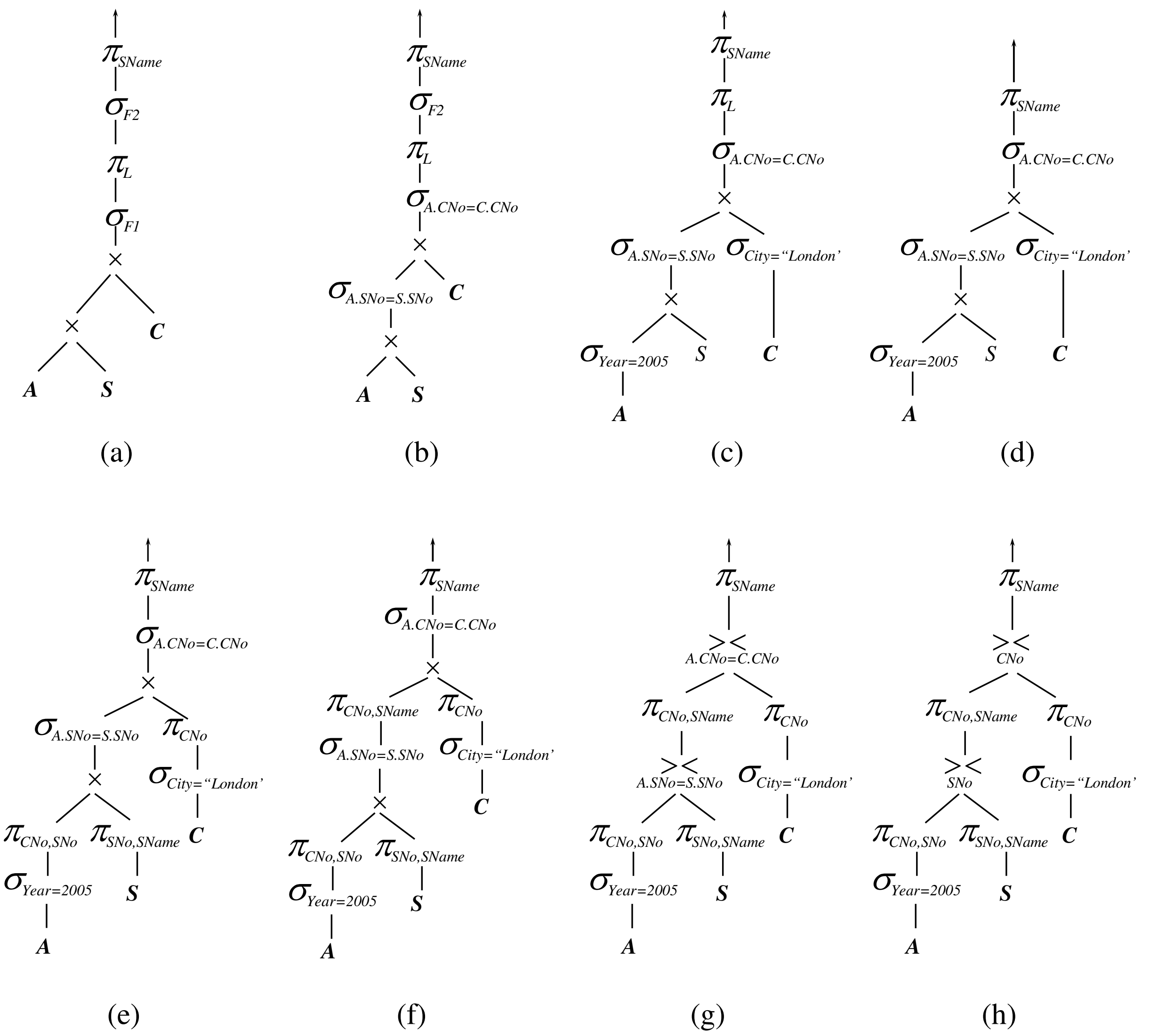


Figure 2. A sample sequence of optimization steps represented through RA trees

Legend: The following abbreviations have been used:

L = {CNo, CName, URL, City, SNo, SName, Address, Phone, Email, ANo, AName, AType, Price, Qty, Year}

F1 ≡ (A.SNo=S.SNo and A.CNo=C.CNo)

F2 ≡ (Year=2005 and City='London')

The respective *RA* expressions follow. Query optimization topics should be included in textbooks adopted by instructors of DB courses as well.

(a) $\pi_{SName}\ (\sigma_{F2}\ (\pi_{L}\ (\sigma_{F1}\ ((A\ X\ S)\ X\ C))))$

(b) $\pi_{SName}\ (\sigma_{F2}\ (\pi_{L}\ (\sigma_{A.CNo=C.CNo}\ (\sigma_{A.SNo=S.SNo}\ (A\ X\ S)\ X\ C))))$

(c) $\pi_{SName}\ (\pi_{L}\ (\sigma_{A.CNo=C.CNo}\ (\sigma_{A.SNo=S.SNo}\ (\sigma_{Year=2005}\ (A)\ X\ S)\ X\ (\sigma_{City='London'}\ (C)))))$

(d) $\pi_{SName}\ (\sigma_{A.CNo=C.CNo}\ (\sigma_{A.SNo=S.SNo}\ (\sigma_{Year=2005}\ (A)\ X\ S)\ X\ (\sigma_{City='London'}\ (C))))$

(e) $\pi_{SName}\ (\sigma_{A.CNo=C.CNo}\ (\sigma_{A.SNo=S.SNo}\ ((\pi_{CNo,SNo}\ (\sigma_{Year=2005}\ (A))\ X\ \pi_{SNo,SName}\ (S))\ X$
$\pi_{CNo}\ (\sigma_{City='London'}\ (C)))))$

(f) $\pi_{SName}\ (\sigma_{A.CNo=C.CNo}\ (\pi_{CNo}\ (\sigma_{A.SNo=S.SNo}\ ((\pi_{CNo,SNo}\ (\sigma_{Year=2005}\ (A))\ X\ \pi_{SNo,SName}\ (S))\ X$
$\pi_{CNo}\ (\sigma_{City='London'}\ (C)))))$

(g) $\pi_{SName}\ (\pi_{CNo}\ (\pi_{CNo,SNo}\ (\sigma_{Year=2005}\ (A))\ \underset{A.SNo=S.SNo}{\bowtie}\ \pi_{SNo,SName}\ (S))\ \underset{A.CNo=C.CNo}{\bowtie}\ \pi_{CNo}\ (\sigma_{City='London'}\ (C)))$

(h) $\pi_{SName}\ (\pi_{CNo}\ (\pi_{CNo,SNo}\ (\sigma_{Year=2005}\ (A))\ \underset{SNo}{\bowtie}\ \pi_{SNo,SName}\ (S))\ \underset{CNo}{\bowtie}\ \pi_{CNo}\ (\sigma_{City='London'}\ (C)))$

3. Practical Aspects in Teaching Levels of Abstraction in Query Definition

A variety of means to exercise query definition-related issues that are offered to the author's students consists of:

- analyzing examples;
- considering problem-solving exercises in class;
- preparing out-of-class homework assignments;
- development of ISs-oriented class and out-of-class projects;
- participation in the annual ISs student seminar, organized by the author;
- field trips involving discussions about DBMSs applications;
- selecting appropriate internships;
- participating in appropriate professional events.

With the progress of the classes during the semester, more and more interesting practical work could be assigned. Students are able to change the query definition environment corresponding to the levels of abstraction taught in class. Covering *QBE* and *SQL* issues and working with DBMSs, such as *Microsoft Access*, *Microsoft SQL Server*, *Oracle*, or *MySQL*, provides the necessary working basis. By experimenting with practice-oriented problems, students are prepared for their future jobs and for pursuing a higher academic degree. The role of the projects is very important[12]. The variety of projects developed by the author's students includes teaching-oriented projects, research-oriented projects, service-oriented projects, individual and team projects, class and out-of-class projects, senior projects, internship-related projects, and their combination[16, 17]. The multiyear experience from the application of strong requirements for the practice-oriented work shows positive results. When students graduate from their academic institution, they possess the needed knowledge and skills for their future profession and/or graduate studies.

4. Conclusions

Teaching query definition in an introductory DB course considers different levels of abstraction related to the corresponding definition strategies. The first level is related to the formulation of queries in natural languages and allows the immediate interaction with users who are non-programmers. The second level corresponds to the *QBE* relational language application and provides a friendly environment for operating with tables by using a bi-dimensional editor. It directly shows how relationships among tables work. The next level of abstraction involves *SQL* issues and requires knowledge and skills in using this established standard for a language tool in relational DBMSs. The last level taught in this DB course discusses the optimization strategy in query definition that leads to writing cost effective queries.

A course that follows the introductory DB course could consider further levels of abstraction of query definition and more application features of query processing. For example, a similar course could discuss different types of performing *RA* operators such as the *join* operator: block nested loop join, index nested loop join, sort-merge join, and hash join and the respective cost estimations as well[3]. Discussing alternative approaches to query optimization and query execution would also be appropriate.

Students' background in query definition and processing is highly desirable for their future professional activity or for the next steps of their graduate education. They will have the necessary routine in switching from one development environment to a new one. This is related to a very important characteristic of the software industry, consisting of introducing continuous changes and innovations, and competing aggressively today.

The development of appropriate curricula, including theoretical and practical aspects of software design, would also allow the development and usage of efficient and effective software products by students who would be ready to compete in the international market.

Bibliography

1. Chaudhury, S., Narasayya, V. and Ramamurthy, R. (2004). Estimating progress of execution for SQL queries. *Proc. of the ACM SIGMOD'04 International Conference on Management of Data.* ACM Press
2. Chandra, A. (1988). Theory of database queries. *Proc. of the 7th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems.* ACM Press, 1-9
3. Connolly, T. and Begg, C. (2005). *Database Systems. A Practical Approach to Design, Implementation, and Management*, 4th edition. Addison-Wesley
4. Czejdo, B. and Rusinkiewicz, M. (1984). Query transformation in an instructional database management system. *Proc. of the 15th Technical Symposium.* ACM Press. 217-223
5. Date, C. J. (2003). *An Introduction to Database Systems*, 8th edition. Reading, MA, Addison-Wesley
6. Elmasri R. and Navathe S. (2003). *Fundamentals of Database Systems*, 4th edition. Addison-Wesley
7. Grine, H., Delot. T. and Lecomte, S. (2005) Adaptive query processing in mobile environment. *Proc. of the 3rd Intern. Workshop MPAC'05.* ACM Press
8. Hsieh, S. (2005). Methodically integrating databases and Web applications. *Journal of Computing Sciences in Colleges,* volume 20, issue 4. CCSC, 20-23

9. Meyer, J. and Conry, M. (2002). Design and implementation of a new course: Creating databases for Web applications. *Journal of Computing Sciences in Colleges,* volume 17, issue 6. CCSC, 135-149
10. Ramakrishnan, R. and Gehrke, J. (2003) *Database Management Systems*, 3rd edition. McGraw-Hill
11. Rys, M. (2005). XML and relational database management systems: Inside Microsoft® SQL Server™ 2005. *Proc. of the ACM SIGMOD'05 Intern. Conf. on Management of Data.* ACM Press, 958-962
12. Tuttle, S. (2002). Practical lessons from experience with the database design course project. *ACM Digital Library, CCSC, www.acm.org*
13. Wei-Shinn, K., Zimmermann, R., Wang, H. and Wan, C. (2005) Query Processing and Optimization: Adaptive Nearest Neighbor Queries in Travel Time Networks. *Proc. of the 13th Intern. Workshop GIS'05*. ACM Press, 210-219
14. Witkowski. A and others. (2005). Advanced SQL modeling in RDBMS. *ACM Transactions on Database Systems*, volume 30, issue 1. ACM Press, 83-121
15. Witkowski. A and others. (2005). Query by Excel. *Proc. of the 31st Intern. Conf. VLDB'05,* VLDB Endowment
16. Zlatarova, F. (2004). Computing Projects in Liberal Arts College Environment. *33rd International Symposium IGIP/IEEE/ASEE*, Fribourg, Switzerland, pp. 113-118
17. Zlatarova, F. (2004). Introducing Ethics in Computing Courses and Extra Class Activities. *Proc. of the Frontiers in Education Conference, ASEE/IEEE/IEEE Computer Society*, Savannah, GA, pp. S1E.6–S1E.9