

# Ontology-based Integration of Adaptive Educational Systems

Sergey Sosnovsky<sup>1</sup>, Antonija Mitrovic<sup>2</sup>,  
Danielle Lee<sup>1</sup>, Peter Brusilovsky<sup>1</sup>, Michael Yudelson<sup>1</sup>

<sup>1</sup>University of Pittsburgh, School of Information Sciences,  
135 North Bellefield ave., Pittsburgh, PA, 15260, USA  
{sas15, hyl12, peterb, mvy3}@pitt.edu,  
<http://www.sis.pitt.edu/~paws/>

<sup>2</sup>Dept. of Computer Science and Software Engineering, University of Canterbury  
Private Bag 4800, Christchurch 8140, New Zealand  
Tanja.Mitrovic@canterbury.ac.nz  
<http://www.cosc.canterbury.ac.nz/tanja.mitrovic/ictg.html>

**Abstract.** With the growth of adaptive educational systems available to students, semantic integration of user modeling information from these systems is emerging into an important practical task. Ontologies can serve as the major representational framework for such integration. In this paper, we report an experiment on integration of domain models of two different adaptive systems. The differences in domain representations require use of manual mappings provided by human experts. The structure of domain ontology helps to refine the resulting mappings and align human expertise.

## 1 Introduction

The expansion of WWW, as a major infrastructure for information and service delivery brings unmatched opportunities for dissemination of adaptive educational technologies. With the growing number of adaptive Web-based educational systems (AWBES) it becomes realistic to have several AWBES available to assist a student in the same domain. This opportunity comes with challenges for several AWBES working with the same student to exchange information about the student. Traditionally, adaptive educational systems focus on the modeling of student's knowledge in the domain, which includes a particular representation of the domain structure in terms of its elementary units and evaluation of student's knowledge of these units. The mediation of such user modeling components will require target systems to achieve a certain level of mutual understanding of the domain semantics. Once the systems agree on the domain model, they can exchange student models for equivalent or related parts of the domain and include it into the adaptive inference.

One of the first steps in this direction would be the implementation of the domain models with the help of ontologies, which express the shared view on the domain semantics and come with a full package of technologies developed within the framework of the Semantic Web initiative. If the student models of two systems rely on the common domain ontology, they can be exchanged and consistently interpreted when necessary. OntoAIMS project provides a good example of such integration [1].

Several research teams have generalized this approach to the level of architectures for user model semantic integration. Mitrovic and Devedzic [2] propose M-OBLIGE, an

architecture for ITS interoperability based on the central ontology used as a reference point by the local ITS ontologies. Niederée et al. [3] describe the Multi-Dimensional Unified User Context Model (*UUMC*) and the creation of a user modeling server providing the commonly-represented user information to multiple adaptive systems. Heckmann [4] developed *UbisWorld*, an infrastructure for cross-system ubiquitous personalization on the basis of shared ontologies.

Unfortunately, the practice of AWBES is still far from the use of common ontologies. Although the designers of AWBES more and more frequently choose to represent the domain models as ontologies, they tend to employ different ontologies for the same domain. In this case, to perform semantic integration of the user models, one has to apply ontology mapping techniques [5]. The mapping between the domain ontologies can be later used as a translation component for user model mediation. Sosnovsky et al. [6] describe the implementation of such approach for translation between two overlay models of student knowledge based on two different ontologies.

In addition, many AWBES do not yet employ ontologies for domain representation. In many cases, implemented technologies for adaptation and user modeling rely on formalisms, different from the conceptual networks, which are the core components of ontologies. The semantic integration of such systems requires manual mapping of underlying domain models into each other. However, we argue that even in this case ontologies could be useful as a common denominator and facilitate future integration.

In this paper, we present a case of domain model mapping in the area of database programming. To support students working with two AWBES for SQL language, we have to integrate two very different domain models. One of the AWBES uses an overlay model of students' knowledge based on an ontology, while the other implements the domain model as a set of constraints. During integration, we used the ontology underlying the first system as the reference model for constraint mapping. We report the results of a small experiment evaluating the results of manual mapping provided by several experts and show how the ontology structure can help to align the manually provided expertise.

The rest of the paper is structured as follows. In Sections 2 we briefly describe the two systems used in the study. Sections 3 and 4 discuss the ontology and the constraint-based domain model. Section 5 presents the mapping experiment. Finally, Section 6 concludes the paper with the discussion and future plans.

## **2 System descriptions**

Two existing adaptive educational systems for the SQL domain have been chosen as an example of AWBES semantic integration.

### **2.1 SQL-Guide**

SQL-Guide is an AWBES helping students to practice SQL skills. A typical SQL-Guide problem description contains a set of predefined databases and a desired output, for which a student is asked to write a matching query (see Fig. 1). The system evaluates student's answer and provides simple feedback. All problems in SQL-Guide are dynamically generated using a set of parameterized templates. An average template is capable of generating several dozens of unique SQL problems with the predefined level of difficulty and the same set of related concepts.

To assist students in choosing the appropriate problem to practice, SQL-Guide employs an adaptive hypermedia technique called adaptive annotation. Every problem in SQL-Guide is annotated with an adaptive icon reflecting the progress of the student with the learning material underlying this problem. The CUMULATE user modeling server keeps

track of all answers the student has given to SQL-Guide's problems and computes the long-term model of his/her knowledge for the related concepts. SQL-Guide requests the state of the model and dynamically annotates problems with the appropriate icons. The student's progress is double-coded: as the knowledge level grows, the icon fades and the bar level rises. By means of this abstraction, SQL-Guide delivers to a student two kinds of information: where the progress has been made (higher bar level) and where the attention should be focused (brighter target color). The checkmarks over the problem icons designate problems that at least once have been solved correctly. To help a student understand the meaning of annotations, QuizGuide dynamically generates mouse-over hints for all icons. A more complete description of the system can be found in [7, 8].

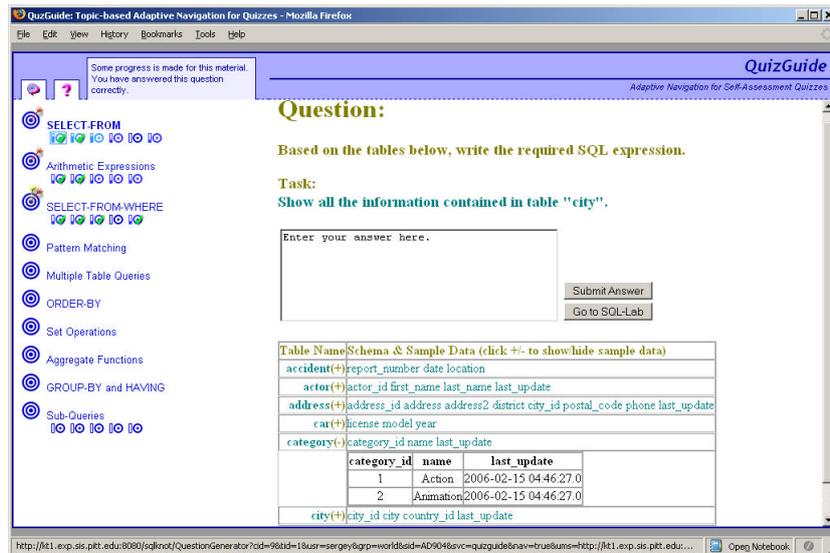


Fig. 1 The interface of SQL-Guide

## 2.2 SQL Tutor

SQL-Tutor is a constraint-based intelligent tutoring system that helps university-level students learn SQL. The system contains definitions of several databases, and a set of problems and the ideal solutions to them. In order to check the student's solution, SQL-Tutor compares it to the correct solution, using domain knowledge represented in the form of constraints. At the beginning of a learning session, SQL-Tutor selects a problem for the student to work on. When the student submits a solution, the system analyzes the solution, identifies mistakes (if there are any), and provides adaptive feedback. When the current problem is solved, or the student requires a new problem to work on, the pedagogical module selects an appropriate problem based on the student model.

The interface, illustrated in Fig. 2, has been designed to be robust, flexible, and easy to use. It reduces the memory load by displaying the database schema and the problem text, by providing the basic structure of the query, and also by providing explanations of the elements of SQL. The main page is divided into three areas. The upper part displays the text of the problem being solved, while the middle part contains the clauses of the SELECT statement, thus visualizing the goal structure. Students need not remember the exact keywords used and the relative order of clauses. The lowest part displays the schema of the chosen database. Schema visualization is very important; all database users are painfully aware of the constant need to remember table and attribute names and the corresponding semantics as well. The motivation here is to remove from the student some of the cognitive load required for checking the low-level syntax, and to enable the student to focus on higher-level, query definition problems. SQL-Tutor was evaluated in 11 studies

since 1998, which proved the system’s effectiveness [9-11]. All the studies showed that SQL-Tutor is successful in supporting students’ learning.

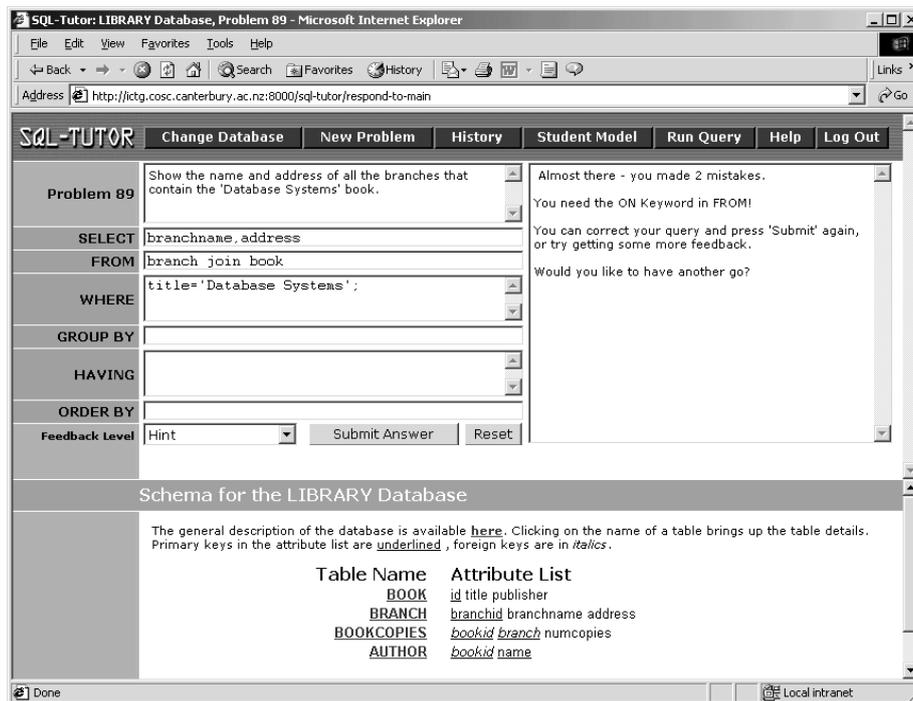


Fig. 2. The interface of SQL-Tutor

### 3 Ontology used by SQL-Guide

Every problem template (and naturally every problem) in SQL-Guide is indexed with several concepts from the SQL Ontology, which was developed as a collaborative effort between the PAWS Lab of the University of Pittsburgh, and the ICT Group of the University of Canterbury. The main purpose of this ontology is to support the development of adaptive educational content for SQL and facilitate the integration of educational systems in this domain, while ensuring the objective modeling of SQL semantics. The ontology can be accessed at <http://www.sis.pitt.edu/~paws/ont/sql.owl>. It is a light-weight OWL-Full ontology, with more than 200 classes connected via three relations: standard *rdfs:subClassOf* (hyponymy relation) and a transitive relation pair *sql:isUsedIn* – *sql:uses*, which models the connection between two concepts, where one concept utilizes another. Fig. 3 gives some examples of relations.

<sql:WhereClause>	<rdfs:subClassOf>	<sql:Clause>
<sql>SelectStatement>	<rdfs:subClassOf>	<sql:Statement>
<sql:WhereClause>	<sql:isUsedIn>	<sql>SelectStatement>
<sql>SelectStatement>	<sql:uses>	<sql:WhereClause>

Fig. 3. Example of relations from SQL Ontology

The level of granularity of the terminal concept in the SQL ontology was chosen to support the adequate modeling of students’ knowledge with the necessary details. At the same time, our goal was not the comprehensive representation of the current SQL standard, therefore certain parts of the domain stay out of the scope of this ontology. We designed the ontology in Protégé 3.3.1 (Fig. 4). It is still being refined, although the main structure of the ontology is determined.

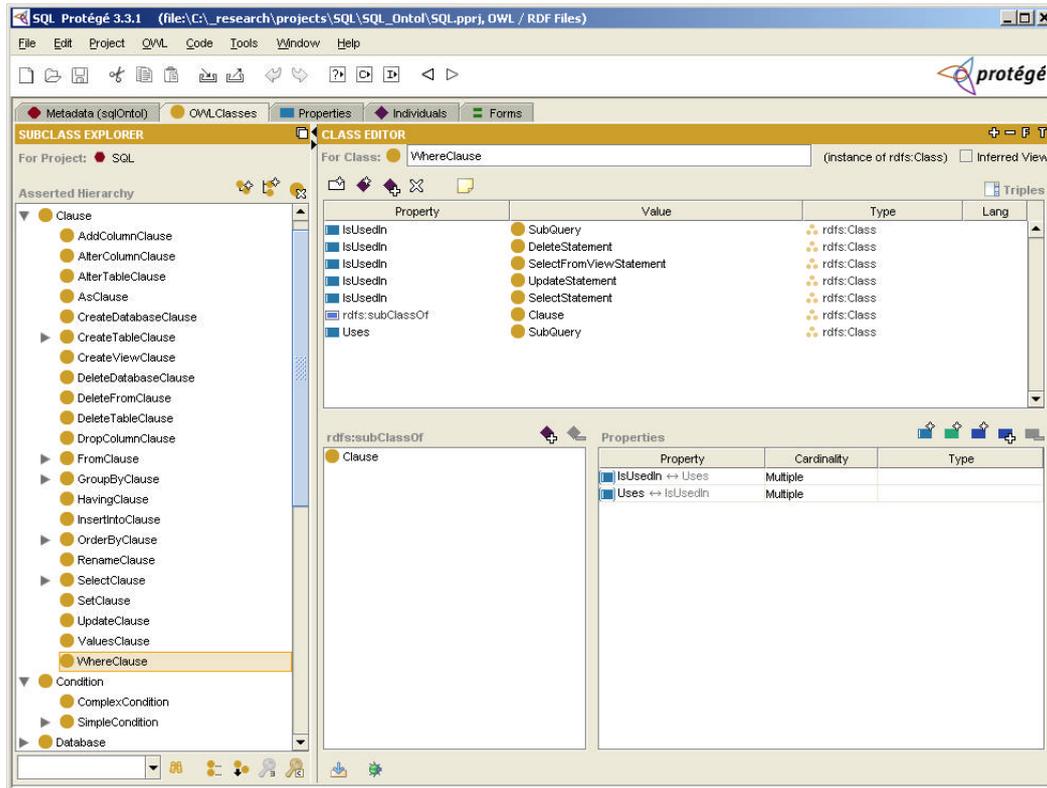


Fig. 4. SQL Ontology

#### 4. Constraint-based model of SQL-Tutor

SQL-Tutor is the first in the family of systems developed at the Intelligent Computer Tutoring Group (ICTG), which use Constraint-Based Modeling (CBM) to model domain and their students [12]. Domain knowledge is represented in form of constraints, which specify basic domain principles that must be satisfied by any correct solution. A constraint consists of two conditions: the *relevance* condition specifies solution features for which the constraint is relevant, while the *satisfaction condition* specifies additional features which a solution must possess in order to be correct. An example constraint is “If you are driving in New Zealand (the relevance condition), you should be on the left side of the road (the satisfaction condition)”. Each condition may be a logical combination of simple tests.

The constraint set in SQL-Tutor contains about 700 constraints, which check for syntactic and semantic correctness of the solution. Fig. 5 illustrates two constraints, used in the study reported in Section 5. Constraint 16 is a syntactic constraint; it is relevant when the student has specified the HAVING clause in his/her query (referred to as *ss*). In this case, the solution must also contain the GROUP BY clause (tested in the satisfaction condition). The constraint contains a message that may be displayed to the student if the constraint is violated. The last element of the constraint is the name of the concept (in this case, GROUP BY) that the constraint covers.

SQL-Tutor is not capable of solving the problems posed to the student. Instead, it evaluates solutions on their semantics by comparing the student’s solution to the pre-specified, ideal solution (*is*). Constraint 635 is relevant if the WHERE clause exists in both *ss* and *is*, with *is* containing a condition using the NOT IN predicate and a nested SELECT statement, but when the student has an alternative way of specifying the same condition, based on the EXISTS predicate (and a nested SELECT). The satisfaction condition of the constraint makes sure that the student is using the negative version of the

predicate. There are additional constraints that will check other aspects of the solution; each constraint focuses on a very narrow part of the domain.

The constraints are modular and problem-independent. There is no one-to-one mapping between problems and constraints; a large number of constraints would be relevant for each problem. Constraint violations indicate errors in the solution, and the system provides the messages attached to violated constraints as feedback to students. SQL-Tutor uses the student model to adaptively select problems at the right level of complexity for the student.

```
(16 "You have to specify the grouping in the GROUP BY clause before you can
specify how to restrict grouping in the HAVING clause!"
(not (null (having ss)))
(not (null (slot-value ss 'group-by)))
"GROUP BY")

(635 "Check the condition involving the nested SELECT!"
(and (not (null (where ss))) (not (null (where is))))
  (match '(?d1 ?a1 "NOT" "IN" "(" "SELECT" ?d5 "FROM" ?t ?*d2)
    (where is) bindings)
  (not (member "IN" (where ss) :test 'equalp))
  (member "EXISTS" (where ss) :test 'equalp)
  (match '(?d3 ?n "EXISTS" "(" "SELECT" ?a2 "FROM" ?t ?*d4)
    (where ss) bindings))
(equalp ?n "NOT")
"WHERE")
```

Fig. 5. Two example constraints

## 5 Semantic Integration Experiment

The fundamental differences in the domain models of the two systems make reliable automatic alignment of these models rather impractical. A well-established set of ontology mapping techniques cannot be applied to this task due to the unique nature of SQL-Tutor's constraints. A constraint is not directly related to a single concept or a sub-tree of the ontology; instead it models the syntactic or semantic relations between various concepts. The development of the algorithm even for partial resolution of the modeling discrepancies between ontologies and constraint-based models is not a trivial task.

To investigate the feasibility of integrating the two systems, we decided first to determine the mapping provided by several experts for a limited set of constraints into concepts of the SQL ontology. Six experts participated in the experiment: two instructors teaching relational database courses, two PhD students who are teaching assistants, and two PhD students regularly using SQL. Each expert was asked to find the most relevant concepts for 20 constraints, which were selected to cover both different parts of SQL and the variety of constraint types. Every expert had hierarchical layout of the ontology. The task was to pick for every provided constraint relevant concepts from the ontology and assign them with the weights (low/medium/large) designating the importance of the relation between the constraints and the concept. The time for this task was not limited. The experts also specified their levels of expertise in SQL, knowledge engineering and constraint-based modeling, summarized in Table 1.

Table 1. Levels of relevant expertise

Expertise in...	SQL	Knowledge Engineering	Constraints
Expert1	Medium	Low	Low
Expert2	High	Low	Low
Expert3	High	Low	Low
Expert4	Medium	High	Low
Expert5	Medium	High	Low
Expert6	High	High	High

## 5.1 Evaluation of Manual Mapping

Although manual mapping by experts is regarded as the golden standard comparing to the mapping acquired automatically, there are several important problems with this approach. Two, arguably, most important of these problems are: it is error-prone and subjective. As a result, there is generally a high level of disagreement between experts.

The data analysis confirmed this phenomenon once again. For example, for constraint 16 (Figure 5) the two experts with the highest knowledge of SQL specified different sets of concepts: Expert A specified five concepts (*Having Clause*, *Group By Clause*, *Where Clause*, *Aggregate Functions* and *Condition*), while expert B only specified the initial two concepts. Furthermore, they did not agree even on the weights assigned to those two concepts, with expert A assigning large weights to both of them, while expert B assigned a large weight to the *Having Clause*, and a medium weight to the *Group By Clause*. Four experts only specified those two concepts (but with different weights), while two remaining experts had additional concepts too.

Overall, our experts used 61 ontology concepts for mapping. Although the average number of concepts a single expert used per constraint was only 3.1 (sd=1.1), the number of unique concepts used by all experts per constraint varied from 6 to 12. For ten constraints more than half mappings have been provided only by one of the six experts, which means, for the half of the constraints no agreement have been reached on more than a half of the mappings. Moreover, for two constraints no single mappings have been found, that would be agreed upon by the majority of the experts (4 out of 6).

To numerically express the agreement between the experts, we computed the matching ratios for every pair. The ratio is essentially the percentage of mapping cases provided by the first expert, on which he/she has agreed with the second expert. For example, if the expert1 found 100 mappings, out of which 50 have been confirmed by the expert2, the rating of agreement of the first expert to the second is  $50/100 = 0.5$ . The average rating of agreement varied from only 40% to 66%.

## 5.2 Evaluation of Ontologically-Enhanced Expertise

More detailed analysis showed that the experts approached mapping in two different ways: some experts provided very laconic set of mappings, others tended to over-specify the conceptual maps of constraints. The total number of mappings varied from 40 to 81. In order to align the manually created mappings, we employed the taxonomic structure of the SQL ontology. We used three main heuristics to refine mappings:

- If an expert maps a constraint to all children of a single concept, such mappings are substituted by the mapping “constraint – parent concept”;
- If the minority of experts map a constraint to a child of the concept chosen for the same constraint by the majority of experts, the majority vote is taken: the mappings are modified to the “constraint – parent concept”
- If the minority of experts map a constraint to a parent of the concept chosen by the majority for the same constraint, and there is no sibling concepts mapped to the same constraint, the majority vote is taken: the mappings are modified to the “constraint – child concept”.

The ontology-enhanced mappings result an increases expert agreement. The pairwise t-test demonstrated a statistically significant difference between the original agreement ratings ( $M = 0.52 \pm 0.046$ ) and the agreement ratings for ontology-aligned expertise ( $M = 0.59 \pm 0.032$ );  $t(5) = 3.705$ ;  $p = 0.014$ .

## 6 Discussion

We have presented an example of semantic integration of two AWBES employing different adaptation technologies and different domain models. The magnitude of diversity of the underlying domain models prevents us from using existing automatic techniques for semantic integration. Instead we rely on the manual mapping of domain models provided by human experts. We conducted a small experiment and analyzed the results of manual mapping. The experiment showed a very high level of disagreement between the experts. However the application of very basic inference rules exploiting taxonomic relation between the ontology concepts allows us to significantly improve the mapping results. Such ontological inference overcomes overly specific/general mappings and align the manually-provided expertise. The resulting ratings of agreement are still far from acceptable. We plan to further refine the mapping and extend it to map the entire domain models. Another important direction of this project is the implementation of protocols for user authentication, remote application call, user modeling information exchange etc.

## References

1. Denaux, R., Dimitrova, V., Aroyo, L. Integrating Open User Modeling and Learning Content Management for the Semantic Web. In, Ardissono, L., Brna, P., and Mitrovic, A., (eds.), *10th Int. Conf. User Modeling*, Edinburgh, Scotland, UK: Springer, 2005, pp. 9-18.
2. Mitrovic, A., Devedzic, V. A Model of Multitutor Ontology-based Learning Environments. *Continuing Engineering Education and Life-Long Learning*, 14, 3 (2004), 229-245.
3. Niederée, C., Stewart, A., Mehta, B., Hemmje, M. A Multi-Dimensional, Unified User Model for Cross-System Personalization. *Workshop on Environments for Personalized Information Access at AVT2004*, Gallipoli, Italy, 2004, pp. 34-54.
4. Heckmann, D. *Ubiquitous User Modeling*. Berlin, Germany: Akademische Verlagsgesellschaft Aka GmbH, 2006.
5. Kalfoglou, Y., Schorelmmmer, M. Ontology Mapping: the State of the Art. *The Knowledge Engineering Review*, 18, 1 (2003), 1-31.
6. Sosnovsky, S., Dolog, P., Henze, N., Brusilovsky, P., Nejd, W. Translation of Overlay Models of Student Knowledge for Relative Domains Based on Domain Ontology Mapping. In Luckin, R., Koedinger, K.R., and Greer, J., (eds.), *13th Int. Conf. Artificial Intelligence in Education*, Marina Del Ray, CA, USA: IOS Press, 2007, pp. 289-296.
7. Sosnovsky, S., Brusilovsky, P., Lee, D.H., Zadorozhny V., Zhou X. Re-assessing the value of adaptive navigation support in e-learning context. In: J. Kay & P. Pearl (eds.) Proc. 5th Int. Conf. Adaptive Hypermedia and Adaptive Web-Based Systems, 2008 (accepted).
8. Brusilovsky, P., Sosnovsky S., Lee, D.H., Yudelso, M., Zadorozhny V., Zhou X. An open integrated exploratorium for database courses. In: E. Menasalvas & A. Young (eds.) Proc. 13th Conf. Innovation and Technology in Computer Science Education, 2008 (accepted).
9. Mitrovic, A., Martin, B., Suraweera, P. Intelligent tutors for all: Constraint-based modeling methodology, systems and authoring. *IEEE Intelligent Systems*, special issue on Intelligent Educational Systems, 22(4), 38-45, July/August 2007.
10. Mitrovic, A., Suraweera, P., Martin, B. and Weerasinghe, A. (2004) DB-suite: Experiences with Three Intelligent, Web-based Database Tutors. *Journal of Interactive Learning Research*, 15(4), 409-432, 2004.
11. Mitrovic, A., Ohlsson, S.: Evaluation of a Constraint-based Tutor for a Database Language. *Int. J. on Artificial Intelligence in Education*, 10(3-4), 238-256, 1999.
12. Ohlsson, S.: Constraint-based student modeling. In: Greer, J.E., McCalla, G (eds): *Student modeling: the key to individualized knowledge-based instruction*, 167-189, 1994.