Multiyear Evaluation of SQL-Tutor: Results and Experiences

Antonija Mitrovic, Brent Martin, Michael Mayo

Intelligent Computer Tutoring Group
Department of Computer Science, University of Canterbury
Private Bag 4800, Christchurch, New Zealand
{tanja,bim20,mmayo}@cosc.canterbury.ac.nz

Abstract
We report on three evaluation studies performed during 1998 and 1999 on SQL-Tutor, an intelligent educational system for the SQL database language. The main goal of our research has been the exploration and extension of Constraint-Based Modeling, a student modeling approach proposed by Ohlsson. SQL-Tutor provided us with experiences of using CBM and also provided opportunities to extend the approach in several important ways. The main goal of all three evaluation studies was to determine how well CBM supports student learning. We have obtained extremely positive results. Students who learnt with SQL-Tutor performed significantly better than those who did not on a subsequent classroom examination. Furthermore, the analysis of students’ learning also shows that CBM has a sound psychological foundation. In the second study, we evaluated the effectiveness of feedback provided to the students. This study showed that high-level advice is most beneficial to students’ learning. The focus of the third study was different. We extended CBM to support long-term modeling of student knowledge, and used this extension to develop an adaptive problem-selection strategy. The study revealed the benefits of this strategy. We also reflect on our experiences in evaluating SQL-Tutor.
Keywords
Student modeling, constraint based modeling, evaluation, intelligent educational systems, probabilistic student model, pedagogical decision making

This paper has not been submitted elsewhere in identical or similar form, nor will it be during the first three months after its submission to UMUAI.
1. Introduction

Evaluating any AI-based project is a difficult task. Intelligent Educational Systems (IES) fit this statement perfectly, being the application of AI methods to educational environments. The underlying theories are either new or still under development. As a result, there is no widespread agreement as to how the tasks these systems must perform, such as student modeling, adaptive pedagogical decision making, generation of instructional dialogues and others, are to be performed. Since the area is quite young, it is logical for researchers to focus on establishing the basic methodology first. However, the importance of evaluation cannot be overlooked. Evaluation is fundamental in all stages of a research project, as it provides guidance for future stages. It is not only within projects that evaluation can benefit research in IESs. Across-system evaluation is also of extreme importance, because it allows for comparisons of effectiveness and suitability of various approaches. This kind of evaluation provides us with the relative benefits of different approaches and identifies avenues for future research, therefore shaping the whole research area.

Most of the older IES research projects involved no evaluation at all or performed limited formative evaluations. Some more recent papers, fortunately, include data on carefully designed and performed evaluations, even the summative ones. We hope this trend will continue, as the area matures.

In this paper, we report on three evaluation studies performed in 1998 and 1999 on SQL-Tutor, an IES for the SQL database language. All three studies involved elements of summative and formative evaluation. We present the system in section 2, followed by a brief overview of Constraint Based Modeling (Ohlsson 1994) in section 3. Section 4 describes the evaluation studies. The analysis of students’ responses to the user questionnaire is given in section 5, and shows that the students perceived SQL-Tutor as user-friendly, and judged the
feedback helpful, and the interface easy to learn. The main goal of our research has been the
evolution and extension of Constraint-Based Modeling. One of the aims of developing
**SQL-Tutor** was to provide a test bed for the methodology in a fairly complex domain.
Therefore, **SQL-Tutor** provided us with experiences of using CBM and also provided
opportunities to extend the approach in several important ways. The main goal of all three
evaluation studies was to determine how well CBM supports student learning. We have
obtained extremely positive results, presented in section 6. In the second study, discussed in
section 7, we evaluated the effectiveness of feedback provided to the students. Section 8
presents the third study, which revealed the benefits of the adaptive problem-selection
strategy based on a probabilistic student model. We present the conclusions in the final
section.

### 2. **SQL-Tutor**

**SQL-Tutor** is a problem-solving environment intended to complement classroom instruction,
and we assume that students are already familiar with database theory and the fundamentals
of SQL. The need for an intelligent tutoring system in the area of SQL is discussed elsewhere
(Mitrovic 1998a). In **SQL-Tutor**, students work on their own as much as possible and the
system only intervenes when the student is stuck or asks for help. There are three functionally
identical versions of the system, for Solaris, MS Windows and the Web. Here we give only a
brief description of the system, and the interested reader is referred to other papers (Mitrovic
1998b, Mitrovic & Ohlsson 1999) and the system’s Web page\(^1\) for details.

The architecture of the stand-alone version\(^2\) of the system is illustrated in Figure 1. At the
beginning of a session, **SQL-Tutor** selects a problem for the student to work on. When the
student enters a solution, the pedagogical module sends it to the student modeler, which

---

\(^1\) [http://www.cosc.canterbury.ac.nz/~tanja/sql-tut.html](http://www.cosc.canterbury.ac.nz/~tanja/sql-tut.html)

\(^2\) For details of the architecture of the Web-enabled version, please see (Mitrovic & Hausler 2000).
analyzes the solution, identifies mistakes (if there are any) and updates the student model appropriately. On the basis of the student model, the pedagogical module generates an appropriate pedagogical action (i.e. feedback). After the first attempt, the student is only told whether his/her solution is correct or not. The level of detail in feedback messages increases if the student is not able to correct the solution. We discuss different levels of feedback in section 7. When the current problem is solved, SQL-Tutor offers the possibilities of logging off, or going on to the next problem. There are two ways to select the next problem in SQL-Tutor. Students can work through a pre-specified sequence of problems, or turn problem selection over to the system. In the latter case, the system will select an appropriate problem on the basis of the student model. We discuss the problem-selection strategies used in section 8.

The system contains definitions of several databases, a set of problems for each database and the ideal solutions to them. Each problem is assigned a difficulty level, which depends on many features, such as the wording of the problem, the constructs needed for its solution, the number of required tables/attributes etc. New databases can easily be added to SQL-Tutor, by supplying the same SQL files used to create the database in a database management system.

The ideal solutions are necessary because SQL-Tutor has no domain module, and therefore must evaluate student solutions by comparison to correct ones. There are two reasons for not having a domain module. Firstly, database queries are given in a natural language; however, the current state-of-the-art in Natural Language Processing (NLP) is still far from being able to handle various inherent problems, such as references and synonyms. There is a possibility to avoid NLP: the text of the problem may be represented not in its natural-language form, but in a form that could be the product of NLP, as done in (Anderson et al. 1995). However, it is difficult to avoid building parts of the solution into such a
representation. Furthermore, even if we overlook the NLP problem, the knowledge required to write SQL queries is ill defined and it would be very difficult to develop a problem solver in this area.

Nevertheless, an ITS must be able to evaluate student answers. SQL-Tutor does that by comparing student solutions to correct solutions. In order to be able to check the correctness of the student's solution, SQL-Tutor uses domain knowledge represented in the form of constraints, described in more detail below. This constraint-based model handles differences between the student and ideal solutions, even major ones such as differing solution strategies.

The interface of SQL-Tutor, illustrated in Figure 2, has been designed to be robust, flexible, and easy to use and understand. It reduces memory load by displaying the database schema and the text of a problem, by providing the basic structure of the query, and by providing explanations of the elements of SQL. The main page is divided into four areas. The upper part shows the text of the problem being solved so students can remind themselves easily of the elements requested in queries. The middle left part contains the clauses of the

---

**Fig. 1. Architecture of SQL-Tutor**
SQL SELECT statement, thus visualizing the goal structure. Students need not remember the exact keywords used and the relative order of clauses. The middle right section of the window is the feedback area. In Figure 2, the student has submitted an incorrect solution, and the current feedback informs the student that the solution is erroneous. The lowest part displays the schema of the currently 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. Students can ask the system for descriptions of databases, tables and attributes, as well as the descriptions of the SQL constructs. 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** uses Constraint-Based Modeling (Ohlsson 1994) to diagnose students’ solutions. The conceptual domain knowledge is represented in terms of over 500 constraints. We discuss CBM and give examples of constraints included in **SQL-Tutor** in the next section. A student’s solution is matched to constraints to identify any that are violated. Long-term student’s knowledge is represented as an overlay model, in which a tally for each constraint shows the frequency of correct and incorrect use. Both students and problems in **SQL–Tutor** are assigned a level. The student’s level is incremented if he/she solves two or more problems consecutively at or above the student’s current level, within three attempts each.

### 3. CBM

Constraint-Based Modelling is a student modelling approach proposed by Ohlsson (1994), as a way of overcoming the intractable nature of student modelling. CBM arises from Ohlsson’s theory of learning from errors (1996), which proposes that we often make mistakes when performing a task, even when we have been taught the correct way to do it. According to this theory, we make mistakes because the declarative knowledge we have learned has not been internalised in our procedural knowledge, and so the number of decisions we must make while performing the procedure is sufficiently large that we make mistakes. By practicing the task, however, and catching ourselves (or being caught by a mentor) making mistakes, we modify our procedure to incorporate the appropriate rule that we have violated. Over time, we internalise all of the declarative knowledge about the task, and so the number of mistakes we make is reduced. Ohlsson therefore explains the process of learning from errors as consisting of two phases: *error recognition* and *error correction*. A student needs declarative knowledge in order to detect an error; only then, the error can be corrected by specializing faulty knowledge so that it is applicable only in situations in which it is appropriate.
Procedure-tracing domain models (Anderson et al 1995) check whether or not the student is performing correctly by comparing the students procedure directly with one or more “correct” ones. In CBM, we are not interested in what the student has done, but in what *state* they are currently in. As long as the student never reaches a state that is known to be wrong, they are free to perform whatever actions they please. State constraints define equivalence classes of problem states. An equivalence class triggers the same instructional action; hence all states in an equivalence class are pedagogically equivalent. Therefore, it is possible to attach feedback messages directly to constraints. A violated constraint signals an error, which translates to incomplete/incorrect knowledge. The domain model is therefore a collection of state descriptions of the form:

“If <relevance condition> is true, then <satisfaction condition> had better also be true, otherwise something has gone wrong.”

In other words, if the student solution falls into the state defined by the relevance condition, it must also be in the state defined by the satisfaction condition in order to be correct. An example3 of a constraint in the domain of SQL is:

(p 34

"If there is an ANY or ALL predicate in the WHERE clause, then the attribute in question must be of the same type as the only expression of the SELECT clause of the subquery."

(and (not (null (where ss)))

(match '(?*d1 ?a (?or "" "" "" "" ""

(or "ANY" "ALL") 

("SELECT" ?*la "FROM" ?*d2 "") ?*d3)

(where ss) bindings))

(and (equal (length ?la) 1))

---

3 For more examples of constraints in SQL-Tutor see (Mitrovic & Ohlsson 1999)
(equalp (find-type ?a) (find-type (car ?la))))

"WHERE")

The first part of the constraint is a unique number, followed by the hint message that will be displayed to the student if the constraint is violated. The relevance condition specified that the constraint is important for solutions in which the WHERE clause is not empty, and it contains a condition based on the ANY/ALL keyword. In such cases, the solution is correct if the SELECT clause of the nested query contains only one expression, which is an attribute of the same type as the attribute preceding the subquery. These two tests make up the satisfaction condition. The last part of the constraint identifies the part of the solution the constraint is dealing with (the WHERE clause in constraint 34). The constraint base of SQL-Tutor contains more than 500 constraints, but this number is likely to increase as new problems requiring new situations are added to the system. As can be seen, the relevance and satisfaction conditions are LISP clauses. They can contain any LISP predicate, but the most frequent predicate is match, which enables pattern matching.

A very important feature of CBM is its computational simplicity. Instead of using complex reasoning as required by other student modeling approaches, CBM reduces student modeling to pattern matching. Conditions are combinations of patterns, and can therefore be represented in compiled forms, such as RETE networks (Forgy 1982), which are very fast and for which off-the-shelf software is available. In the first step all relevance patterns are matched against the problem state. In the second step, the satisfaction components of constraints that matched the problem state in the first step (i.e., the relevant constraints) are matched. If a satisfaction pattern matches the state, then the constraint is satisfied, and the system is not to take any action. In the opposite case, the constraint is violated. The student model consists of all violated constraints.
CBM does not require extensive studies of students bugs required by enumerative modeling, as in (Anderson et al 1995). Furthermore, Ohlsson's approach is not sensitive to the radical strategy variability phenomenon, as it completely ignores procedures used by students to solve problem, thus allowing for student's inconsistencies in choosing problem-solving strategy. CBM is neutral with respect to the pedagogy, since different pedagogical actions (immediate or delayed ones) may be generated on the basis of the model.

Another advantage of CBM is that it allows for a simpler architecture, since there is no need for a runnable expert module. CBM-based systems are able to generate instructional actions even without being able to solve problems on their own, by focusing on violated constraints. Of course, CBM does not prevent us from having a domain module; on the contrary, the existence of a domain module can be very beneficial to the student, as it can provide the answer to questions such as "What do I do next?".

As proposed by Ohlsson (1994), CBM is a method for diagnosing a student’s solution. The approach identifies errors, which is extremely important for students lacking declarative knowledge, who are unable to detect errors themselves. As stated earlier, one of the goals of our research is to evaluate how well CBM supports learning. We also discuss in this paper how CBM can be extended to allow for long-term modelling of student’s knowledge, and generation of pedagogical actions.

4. Evaluation Studies

Three evaluation studies of SQL-Tutor have been performed so far, with a new study planned for September 2000. We present the common details of all three studies here, and the specifics are discussed in later sections. All studies were carried out at the University of Canterbury, with Computer Science students enrolled in database courses. The students had attended six lectures about SQL and they had completed at least eight hours of hands-on
experience of query definition prior to using the system. The students used SQL-Tutor in a
two-hour session, during their normal lab time. All students' actions were recorded and the
students filled out a questionnaire at the end of the session. There were several observers
present in all the evaluations, who agreed among themselves that the students were quite
interested in interacting with the system and exploring its various functions. In all three
studies, all actions performed by the students were recorded in logs, and students filled in a
user questionnaire at the end of the session.

Study 1 was performed in April 1998. The participation was voluntary and anonymous.
Out of the 49 students enrolled in the course, twenty choose to participate. The goal of this
study was twofold: to evaluate how well CBM supports student learning and to evaluate the
interface and the constraint base of SQL-Tutor.

Study 2 was carried out in May 1999, and involved all senior-year students enrolled in a
database course. The goal of this study was to evaluate the effectiveness of feedback provided
by the system. The students were randomly allocated to one of two versions of the system:
one version gave restricted feedback, and the other version generated all levels of feedback.

Study 3 was performed in October 1999, and involved all second-year students taking an
introductory database course. In addition to the questionnaire, study 3 students sat a pre-test
and a post-test. Three versions of the system were used in the study: the basic version, a
version which generated probabilistic student models and used them to select problems, and a
version in which feedback was presented via an animated pedagogical agent. The students
were randomly assigned to one of the versions. Since the evaluation of the pedagogical agent
is irrelevant to this paper, we focus on the other two versions only. For details of the analysis
of the pedagogical agent, please see (Mitrovic and Suraweera 2000). In this paper, we report
on the evaluation of the probabilistic student model and the appropriateness of problems
selected on the basis of such student models.
We present the results of subjective analysis first, by summarizing the responses to user questionnaires from all three studies. When questionnaires included specific question relevant to a single study only, the responses are summarized in a section devoted to the corresponding evaluation study.

5. Subjective evaluation

This section presents a summary of the students’ answers to the user questionnaire in all three studies. The purpose of the questionnaire given to students at the end of the session was to evaluate the students' perception of SQL-Tutor. The questionnaire, which is included in Appendix A*, consisted of 16 questions, most of which were based on the Likert scale with five responses ranging from very good (5) to very poor (1). Students were also allowed to give free-form responses.

Table I gives the number of students involved in each study, and their responses to whether they would recommend SQL-Tutor to other students. Please note that the percentages do not add up to 100%, as some students have not answered all questions. As illustrated there, students appreciated the style of learning with the system. Figure 3 also illustrates the students’ attitude towards the system. The majority of the students appreciated the learning experiences with SQL-Tutor, the feedback received and liked the interface.

<table>
<thead>
<tr>
<th>Study</th>
<th>No of students</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>75%</td>
<td>0%</td>
</tr>
<tr>
<td>2</td>
<td>33</td>
<td>84%</td>
<td>3%</td>
</tr>
<tr>
<td>3</td>
<td>48</td>
<td>94%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Table I. Some statistics about the subjective evaluation

* Appendix A contains the questionnaire used in Study 3. Some of the questions in the questionnaires for the previous two studies were slightly different.
Study 1: Learning constraints

Study 1 included elements of formative and summative evaluation. As stated earlier, we wanted to evaluate some components of the system (the interface and the constraint base), and we also wanted to evaluate CBM. Students’ reactions to the system were summarized in the previous section. In this section, we present an analysis of how well CBM supports learning. We also report on the effect of learning with the system on subsequent classroom performance.

Fig. 3. Responses from the user questionnaire (in percentages)
6.1. Mastery of Constraints

In a previous paper (Mitrovic & Ohlsson 1999) we showed that state constraints represent psychologically appropriate units of knowledge. When students’ learning is plotted in terms of constraints, we get a smooth curve that closely approximates a so-called power law (Anderson 1993). The evaluation data reported in (Mitrovic & Ohlsson 1999) was collected in study 1. The analysis of students’ learning published in that paper concentrated on a subset of 100 randomly selected constraints, which were relevant at least once during the study. Here we perform the same type of analysis, but this time using all the constraints that were used by students, and we report on the findings for all three evaluation studies.

For each constraint, we identified the problem states in which that constraint was relevant in each student's record and rank ordered from 1 through R. We refer to these as occasions of

![Graph showing the power law relationship for three studies](image)

**Fig. 4. Mastery of constraints**

Study 1
\[ y = 0.0753x^{-0.9904} \]
\[ R^2 = 0.6764 \]

Study 2
\[ y = 0.0407x^{-0.7266} \]
\[ R^2 = 0.8375 \]

Study 3
\[ y = 0.0773x^{-0.7304} \]
\[ R^2 = 0.9408 \]
application. For each occasion, it was recorded whether the relevant constraint was violated or satisfied. This analysis was repeated for each student. From this transformation of the computerized records we can compute the probability of violating a given constraint C. To estimate this quantity, we computed, for each student, the proportion of constraints that he or she violated on the first occasion of application, the second occasion, and so on. These proportions were averaged across all students and plotted as a function of the number of occasions when C was relevant. Figure 4 shows the results of this analysis for all three evaluation studies.

As the number of uses increases, the set of constraints that were relevant for that number of times diminishes in size. At n=10, the constraint set has, on average, dropped to 32% of the original set, while at the end of each series (when the number of failures is zero) the set can be as low as 3% of the original. Hence, a single failure will have 30 times the impact on probability as at the start of the curve. We have arbitrarily chosen n=10 to reduce this effect. As can be seen, in all three studies students learnt the constraints in the same way. These results provide additional proof of the soundness of CBM.

6.2. Classroom performance

Study 1 was voluntary; out of the 49 students enrolled in the course, twenty chose to participate. Therefore, study 1 was not a controlled study. At the end of the course, students sat an examination, which contained questions relevant to the domain of SQL-Tutor. This allowed for a comparison of competence of the two groups of students. The students who used SQL-Tutor achieved higher marks compared to the students in the control group, as illustrated in Table II.
The difference in means is significant (t = 2.908, p = .006). We computed the values for the effect size and power, the two measures commonly used to determine the effects and validity of experiments. In the IES world, the common way to calculate the effect size is to divide the increase in means by the standard deviation of the control group (Bloom 1984). Using this formula, we get the effect size of 0.66. This result is comparable to the results published in (Albacete & VanLehn 2000); they report on the effect size of 0.63 in a similar setting, with students using their system in a single, 2-hour session. The effect size of this magnitude is common in remedial tutoring (Albacete & VanLehn 2000), while better results are obtained in longer studies. For example, (Anderson et al. 1995) report an effect size of 1.0 in studies that lasted for one semester, and Bloom (1984) reports an effect size of 2.0 when one-on-one human tutoring replaces traditional classroom learning. Therefore the effect size of 0.66 after only a single session is remarkable.

Another way of measuring the effect size is advocated for in (Chin 2000). The effect size is computed as the omega squared value, which gives the magnitude of change in the dependent variable due to changes in the independent variables. In our case, we get the effect size of 0.141, which is quite large.

The other measure, power or sensitivity, gives an indication of how repeatable the experiment is. It gives the percentage of repeated experiments for the same design, the same effect size and the same number of subjects that would produce the given significance. Chin (2000) recommends that researchers strive for power of 0.8. In our case, the power at

<table>
<thead>
<tr>
<th>Group</th>
<th>Mean</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experimental</td>
<td>82.75</td>
<td>8.76</td>
</tr>
<tr>
<td>Control</td>
<td>71.23</td>
<td>17.56</td>
</tr>
<tr>
<td>Total</td>
<td>76.24</td>
<td>15.39</td>
</tr>
</tbody>
</table>

Table II Competence of the two groups in study 1
significance threshold of 0.05 is 0.75, and we get the power of 0.8 for the significance threshold of 0.08.

Because this experiment was not controlled and some student kept using the system after the study, we cannot claim to have a definite proof of the quality of the system from these results. However, the competence of the experimental group is significantly higher.

We had hoped to perform the same analysis in a controlled experiment. However, this kind of study also introduces an ethical dilemma. If the teacher is aware that a particular instructional approach (in this case, the use of SQL-Tutor as problem-solving environment) is good for students, what is the justification to prevent one group of students from using it? In our case, the students learned about the system from the students who were using it, and demanded to be given access to the system. We decided that it is more important for students to learn SQL than to evaluate the system, and consequently, we are not able to report on a controlled version of the same experiment. This illustrates the complexities of evaluating educational systems.

7. Study 2: Evaluating feedback

The goal of the second study was to evaluate the effect of feedback given to students. The level of feedback determines how much information is provided to the student. There are six levels of feedback in SQL-Tutor: positive/negative feedback, error flag, hint, all errors, partial solution and complete solution. At the lowest level (positive/negative feedback), the message simply informs the student whether the solution is correct or not. This type of feedback is illustrated in Figure 2. An error flag message informs the student about the clause in which the error occurred. A hint-type message gives more information about the type of error, by specifying the general principle that has been violated. This description is

---

5 In case that there are several mistakes in various clauses, the pedagogical module will select one of them to start with.
directly taken from the constraint. A message of type *all errors* presents the hint messages for all errors the student has made. *Partial solution* feedback displays the correct content of the clause relevant to the first violated constraint, while the *complete solution* simply displays the pre-specified ideal solution of the current problem. Table III illustrates the messages that the student would get in the situation illustrated in Figure 2.

<table>
<thead>
<tr>
<th>Feedback type</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error flag</td>
<td>Almost there – a few mistakes though. One of them is in the WHERE clause.</td>
</tr>
<tr>
<td>Hint</td>
<td>Make sure that you have listed all the necessary tables for this query. Consider all the attributes necessary in join conditions, search conditions, expressions to be retrieved, grouping and restricting grouping, and sorting.</td>
</tr>
<tr>
<td>All errors</td>
<td>1. You have to use another table in this query!</td>
</tr>
<tr>
<td></td>
<td>2. When you compare the value of an attribute to a constant in WHERE, they must be of the same type.</td>
</tr>
<tr>
<td></td>
<td>3. You need another search condition, using a string constant!</td>
</tr>
<tr>
<td></td>
<td>4. Check whether you are using the appropriate string constants in WHERE!</td>
</tr>
<tr>
<td></td>
<td>5. Check that you have all the necessary string constants in WHERE! You need to specify more.</td>
</tr>
<tr>
<td></td>
<td>6. Check that you have all the necessary string constants in WHERE! You need to specify more.</td>
</tr>
<tr>
<td>Partial solution</td>
<td>WHERE DIRECTOR=(SELECT NUMBER FROM DIRECTOR WHERE FNAME='Stanley' AND LNAME='Kubrick')</td>
</tr>
<tr>
<td>Complete solution</td>
<td>The correct solution of this problem is:</td>
</tr>
<tr>
<td></td>
<td>SELECT TITLE</td>
</tr>
<tr>
<td></td>
<td>FROM MOVIE</td>
</tr>
<tr>
<td></td>
<td>WHERE DIRECTOR=(SELECT NUMBER FROM DIRECTOR WHERE FNAME='Stanley' AND LNAME='Kubrick')</td>
</tr>
</tbody>
</table>

*Table III.* Examples of various types of feedback messages
The level of feedback is adjusted in the following way. When a student starts working on a new problem, he/she receives only feedback of the positive/negative type. If the student goes through several unsuccessful solution attempts, the feedback is upgraded to the error flag level and then to the hint level. The system never volunteers more than a hint, but the student can ask for partial and complete solutions by clicking on a feedback button and selecting the desired level.

The described mechanism of selecting feedback is overly simple, and is not adaptive. One of our goals is to develop an adaptive mechanism for selecting feedback types. As an initial step towards this goal, we performed an evaluation of the effectiveness of various types of feedback available to students. Our initial hypothesis was that constraint-level feedback (hint or all-errors) would be most effective (that is, best support students’ learning). We hypothesized that positive/negative and error-flag feedback would be too general to be informative for students, and that partial-solution and complete-solution feedback would be contra-productive in many cases. Although the student might directly copy latter types of feedback, which maps onto correct solutions in the next submission, we thought that such feedback would not help students to correct misconceptions in the long term.

Study 2 involved two versions of the system; one version provided feedback of type positive/negative and error flag only (we refer to this version as limited), and the other version generated all levels of feedback (full version). We analyzed the logs collected during the evaluation session in several ways, presented in the following subsections.

7.1. Probability of constraint violation

The first analysis we performed focused on the learning performance. Study 1 showed that the degree of mastery of a given constraint is a function of the amount of practice on that unit. We wanted to determine whether feedback would also influence mastery of constraints, and
analysed the probability of violating a given constraint $C$ for the $n$th problem for which the constraint is relevant. To estimate this quantity, we computed, for each student, the proportion of all constraints that he/she violated in the first problem, the second problem, and so on. These proportions were averaged across all subjects and all constraints.

Figure 5 illustrates the learning performances of the two groups of students. As in section 6.1, we have used a cutoff of $n=10$ to reduce the statistical effects that result from the number of constraints in each set becoming very small.

When the full group is compared to limited, the latter has the higher learning rate. However, the existence of the two groups does not allow us to evaluate our hypothesis, as the full group received partial/complete solution (the detrimental feedback according to our hypothesis) as well as the “good” feedback (hints/all-errors). We therefore post hoc split the full group into two groups: the detailed group who used partial and complete feedback
predominantly, while the general group who used the hint and all-errors messages predominantly.

The analysis of learning of the three groups (limited, general and detailed) is given in Figure 6. These results suggest that detailed feedback (i.e. being shown a solution) is detrimental to the rate of learning. However, it is important here to consider possible sources of extraneous effects. In most cases, the group that begins with the highest error rate also has the highest learning rate. The group with the highest initial error rate (which is independent on the feedback) will therefore display the highest initial learning rate.

![Graph showing learning rates for three groups](image)

**Fig. 6.** The probability of constraint violation for the three groups

Furthermore, for the general and detailed groups, the feedback level was chosen by them, while for the limited group the level was artificially determined. It is possible that any trends observed are not because of the effects of feedback, but reflect a characteristic of the students that choose that feedback level.
We also gathered a few statistics on the three groups, given in Table IV. The detailed group solved most problems on average; however, this is due to the fact that the students in this group predominantly selected partial and complete solutions. It is much more important that the students in the general group needed only 2.17 attempts per problem, compared to 2.21 and 2.25 attempts on average for the detailed and limited groups. Also, the amount of time per attempt is shortest for the general group, which is in favour of our hypothesis. The difference in times for the limited and general groups is significant (t=1.87, p=0.08), with the effect size of 0.063, and the power of 0.33. This suggests that the “good” feedback, i.e. the feedback messages provided from the constraints, was easier to absorb, and so the time required to understand the feedback and make the necessary changes is substantially reduced. The variation in time required is also heavily reduced, so the worst examples from both the limited and detailed groups lie many standard deviations outside the distribution for the general group.

<table>
<thead>
<tr>
<th>Group</th>
<th>Solved</th>
<th>NoAttempts</th>
<th>Time/attempt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>mean</td>
<td>SD</td>
</tr>
<tr>
<td>Detailed</td>
<td>87.07%</td>
<td>2.21</td>
<td>0.49</td>
</tr>
<tr>
<td>General</td>
<td>83.49%</td>
<td>2.17</td>
<td>0.65</td>
</tr>
<tr>
<td>Limited</td>
<td>84.10%</td>
<td>2.25</td>
<td>1.38</td>
</tr>
</tbody>
</table>

Table IV. Statistics for the three groups

7.2. Effect of the feedback on violated constraints

Our hypothesis has a corollary that effective feedback on a violated constraint will increase the chance of that constraint being used successfully the next time. We therefore focused on
the effect of feedback received on a violated constraint on the next attempt/problem for which the same constraint is relevant. If a particular type of feedback is better than another, we expect to see an increase in the probability that the constraint is used correctly the next time, because the student is more likely to have learned the constraint.

We determined the frequency of a constraint being used successfully after being violated, with respect to a particular level of feedback received on it. Because some feedback types are intended to refer only to the first violated constraint (error flag, hint, partial solution), the other violated constraints were treated as having received a level of feedback higher than positive/negative, but lower than any of the other feedback types. This is because the other constraints may indirectly receive feedback (e.g. if they relate to the same clause, and so the same partial solution applies), but at a level which is unknown and variable.

Table V presents the frequencies of successful application of a constraint on the next attempt in solving the current problem, after receiving feedback of a specific type. The Const column gives the total number of constraints that were violated and followed by feedback of a certain type. Success is the number of successful applications of the same constraint in the next attempt, while Failure specifies the number of times the same constraint was violated following the feedback. Note that the two columns do not add up to Const, as there may be several instances of the same constraint violated in student logs. The most frequent type of feedback was positive/negative (a total of 890 messages), while full solution was only given on 22 occasions.
Learned gives the percentage of successful application of the constraint in the next attempt following the feedback. The highest value of Learned is obtained for partial solution; however, this does not mean that the students have learnt the constraint from such a feedback message. Instead, students typically retype the given solution and submit it. Therefore, there is no real learning involved. If we ignore partial solution, the next best feedback type is all errors, followed closely by error flag and hint. However, these three types of feedback were offered in vary different proportions, with 224 error flag messages, 153 messages of the all errors type, and only 76 hint messages. Only 27% of the solutions made in the attempt following the full solution are correct, and therefore this type of feedback is contra-productive. The last column (Correct) gives the percentage of correct applications of the constraint following the feedback in any future problem. Partial solution again has the highest percentage here, but it has only been offered 32 times, which is much less than the number of messages generated for other types of feedback.

<table>
<thead>
<tr>
<th>Feedback</th>
<th>Const</th>
<th>Success</th>
<th>Failure</th>
<th>Learned</th>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pos/neg</td>
<td>436</td>
<td>254</td>
<td>636</td>
<td>29%</td>
<td>78.0%</td>
</tr>
<tr>
<td>Error flag</td>
<td>116</td>
<td>98</td>
<td>126</td>
<td>44%</td>
<td>81.8%</td>
</tr>
<tr>
<td>Hint</td>
<td>43</td>
<td>33</td>
<td>43</td>
<td>43%</td>
<td>74.4%</td>
</tr>
<tr>
<td>All errors</td>
<td>64</td>
<td>72</td>
<td>81</td>
<td>47%</td>
<td>80.0%</td>
</tr>
<tr>
<td>Partial sol</td>
<td>26</td>
<td>22</td>
<td>10</td>
<td>69%</td>
<td>91.6%</td>
</tr>
<tr>
<td>Full sol</td>
<td>18</td>
<td>6</td>
<td>16</td>
<td>27%</td>
<td>44.2%</td>
</tr>
</tbody>
</table>

Table V. The effect of feedback on whole sessions
7.3. Focusing on single feedback type

In the previous subsection, we divided all the students logs into three groups (limited, general and detailed) according to the predominant type of feedback used. The three groups were then compared. Due to the problems in the experimental design, we cannot reach definite conclusions from such an analysis, as the students in general and detailed groups received messages of other types in addition to the predominant ones. The other problem encountered was the shortness of sessions, resulting in a relatively small number of occasions where the same constraint was used.

In this subsection we report on another kind of analysis, performed on the level of individual attempts. Instead of taking the whole session as a unit, we now take each attempt at solving a problem, and classify all attempts in accordance to the type of feedback obtained on it. Therefore, the set containing all instances of hint messages consists of attempts made by various students, with no regard to the version of the system they used in the study.

We then performed the same kind of analysis reported in section 6: we analysed the probability of violating a constraint each time that it was relevant, for different types of feedback obtained on the constraint on previous occasions. Some feedback types (error flag, hint, partial solution) apply only to a subset of the constraints, and are intended to target the first constraint failed. In such cases, any other constraints failed during this attempt are assumed a feedback level of positive/negative. The cut-off points are set at 33% of the original number of instances (at n=1).

As illustrated in Figure 7, the initial learning rate (i.e. the slope at n=1) is highest for all errors (0.44) and error flag (0.40) messages, closely followed by positive/negative (0.29) and hint (0.26). The learning rates for partial (0.15) and full solution (0.13) are low. This supports our hypothesis that our CBM-based general feedback is superior to offering a correct solution.
Fig. 7. The probability of constraint violation after feedback.
However, it is important to note that the number of instances in each set was very small, because most attempts involved multiple types of feedback, and so could not be analysed.

7.4. Discussion

As shown, the data gathered in study 2 prefers feedback that presents information about the general domain principles that are violated by the student’s solution (e.g., hint and all errors). The same feedback levels give the shortest time per attempt and the fewest attempts per solved problem. When analysing the individual attempts, these two feedback levels also give the highest rate of learning.

Due to the discussed problems in experimental design and the high level of uncertainty inherent in all projects dealing with human subjects, our conclusions are not irrefutable. However, we believe that it is absolutely critical to perform evaluations of this kind in all ITS-related projects and that we have made a small contribution in identifying and dealing with the caveats that await researchers. We plan to extend SQL-Tutor with an adaptive mechanism that will monitor the student during interaction, and adapt the level of feedback automatically, based on the observations of the study presented here.

8. Study 3: Evaluating a probabilistic student model

As described previously, the initial extension of CBM for use in SQL-Tutor involved a long-term model of student’s knowledge expressed as an overlay over the constraint base. This simple model was used to develop a problem-selection strategy. When the student asks the system to select a problem, SQL-Tutor examines the student model, identifies the focus constraint (which is the constraint that has been violated most often) and selects a problem that is relevant to the focus constraint from the pool of unsolved problems whose level is within +1 or −1 of the student’s current level.
This problem selection strategy is overly simple. In studies 1 and 2, it was often the case that selected problems were too complex or simple for the student, or they jumped to another part of the domain seemingly not connected to the previous problem. We therefore wanted to explore other approaches to long-term modeling of student’s knowledge and to develop new strategies of generating pedagogical actions based on such models.

As a solution, we have developed a probabilistic model of student’s long-term knowledge. Bayesian networks (Charniak 1991, Pearl 1988) are tools for representing and reasoning with uncertain knowledge using Bayesian probability theory. In the next subsection, we describe the probabilistic student model developed for SQL-Tutor. On the basis of this model, we have designed a problem selection strategy, which is described in the rest of this section.

8.1. The probabilistic student model

Before Bayesian networks could be applied to the task of problem selection, SQL-Tutor’s student model had to be reformulated in probabilistic terms. The new student model consists of a set of binary variables \( \text{Mastered}_1, \text{Mastered}_2, \ldots, \text{Mastered}_n \), where \( n \) is the total number of constraints. Each variable can be in the state \( \text{YES} \) or \( \text{NO} \) with a certain probability, indicating whether or not the student has mastered the constraint.

Initial values for \( \Pr(\text{Mastered}_c = \text{YES}) \) were determined by counting the frequencies with which \( c \) was both satisfied and relevant (i.e. either satisfied or violated) in SQL-Tutor logs from previous evaluation studies, and by then dividing the former frequency by the latter. The logs were only analysed up to the point where the user gets the first constraint-specific feedback about \( c \). This ensured that the effects of learning did not bias the initial probabilities. Some constraints that did not appear in the past SQL-Tutor logs either because they were new or they had never been used. For these constraints, \( \Pr(\text{Mastered}_c = \text{YES}) \) was initialised to 0.5.
The student model is updated after the student submits his/her solution to a problem and receives feedback. The system currently uses the heuristics in Table VI to update the probabilities. The reason for manipulating the probabilities by percentages rather than real values is that percentage changes are discounted near the extremes of the probability interval. For example, if \( P(Mastered_c=\text{YES}) = 0.4 \), then satisfying \( c \) would result in a much more significant change to the student model than if \( P(Mastered_c=\text{YES}) \) had been, for example, 0.95.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Update Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>If constraint ( c ) is satisfied, then ( P(Mastered_c = \text{YES}) ) increases by 10% of ((1- P(Mastered_c=\text{YES}))).</td>
<td></td>
</tr>
<tr>
<td>If constraint ( c ) is violated and no feedback about ( c ) is given, then ( P(Mastered_c = \text{YES}) ) decreases by 20%.</td>
<td></td>
</tr>
<tr>
<td>If constraint ( c ) is violated but feedback is given about ( c ), then ( P(Mastered_c = \text{YES}) ) increases by 20% of ((1-P(Mastered_c=\text{YES}))).</td>
<td></td>
</tr>
</tbody>
</table>

**Table VI.** Heuristics used for updating the student model

### 8.2. Predicting student performance on single constraints

We use a simple Bayesian network given in Figure 8 to predict the performance of a student given a problem \( p \) on a single constraint \( c \). \( Mastered_c \) comes from the student model. Both \( \text{RelevantIS}_{c,p} \) and \( \text{RelevantSS}_{c,p} \) are \text{YES}/NO variables. \( \text{RelevantIS}_{c,p} \) is \text{YES} if constraint \( c \) is relevant to problem \( p \)’s ideal solution. Because this can be determined from the problem database, \( \text{RelevantIS}_{c,p} \) is always known with certainty. \( \text{RelevantSS}_{c,p} \) is \text{YES} if constraint \( c \) is relevant to the student’s solution to problem \( p \). \( \text{Performance}_{c,p} \) is a three-valued node taking values \text{SATISFIED}, \text{VIOLATED} or \text{NOT-RELEVANT}. The arcs indicate that \( \text{RelevantSS}_{c,p} \) is dependent on \( \text{RelevantIS}_{c,p} \). \( \text{Performance}_{c,p} \) is dependent on whether or not the student has mastered \( c \), and \( c \)’s relevance to the student solution.
Fig. 8. A Bayesian network for predicting student performance on a single constraint.

A full specification of this Bayesian network requires prior and conditional probabilities. $P(Mastered_c)$ and $P(RelevantIS_{c,p})$ are the prior probabilities, which are already available from the student model and problem database respectively. In Table VII, $\alpha_c$ and $\beta_c$ are properties of the constraint $c$. $\alpha_c$ ($\beta_c$) is the probability of a constraint being relevant to the student’s solution if it is (not) relevant to $p$’s ideal solution. Effectively, $\alpha_c$ and $\beta_c$ provide a measure of the “predictive usefulness” of the ideal solution. For example, when $\alpha_c = \beta_c = 0.5$, the relevance of $c$ to the ideal solution tells us nothing about the relevance of $c$ to a potential student solution. However, if $\alpha_c = 0.9$ and $\beta_c = 0.1$, there is a high probability that constraints

<table>
<thead>
<tr>
<th>RelevantIS$_{c,p}$</th>
<th>YES</th>
<th>NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>RelSS$_{c,p}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>YES</td>
<td>$\alpha_c$</td>
<td>$\beta_c$</td>
</tr>
<tr>
<td>NO</td>
<td>$1-\alpha_c$</td>
<td>$1-\beta_c$</td>
</tr>
</tbody>
</table>

Table VII. $P(RelevantSS_{c,p}|RelevantIS_{c,p})$
relevant to the ideal solution will also be relevant to a student solution, and vice versa.

Like the initial probabilities of mastery, we determined values for $\alpha_c$ and $\beta_c$ from past SQL-Tutor logs. However, these conditional probabilities were not available directly from the data. All that can be determined from the logs was the frequencies with which constraints were relevant to the IS, the SS or both. Derivation (1) shows how $\alpha_c$ was calculated using the chain rule. A similar calculation was done for $\beta_c$. For new or previously unused constraints, $\alpha_c$ and $\beta_c$ were initialised to 0.5.

$$
\alpha_c = P(\text{RelevantSS}_{p,c} = \text{YES} | \text{RelevantIS}_{p,c} = \text{YES})
$$

(1)

$$
= P(\text{RelevantSS}_{p,c} = \text{YES} \& \text{RelevantIS}_{p,c} = \text{YES}) / P(\text{RelevantIS}_{p,c} = \text{YES})
$$

$$
= \frac{\text{# times } c \text{ is relevant to both SS and IS in the logs}}{\text{# times } c \text{ is relevant to IS}}
$$

<table>
<thead>
<tr>
<th>RelevantSS$_{c,p}$</th>
<th>Mastered$_c$</th>
<th>YES</th>
<th>NO</th>
<th>YES</th>
<th>NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>SATISFIED</td>
<td>1-$Slip_c$</td>
<td>$Guess_c$</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>VIOLATED</td>
<td>$Slip_c$</td>
<td>1-$Guess_c$</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>NOT-RELEVANT</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Table VIII. $P(\text{Performance}_{c,p}|\text{RelevantSS}_{c,p},\text{Mastered}_c)$

Table VIII is the conditional probability distribution of $\text{Performance}_{c,p}$ given its parent variables $\text{RelevantSS}_{c,p}$ and $\text{Mastered}_c$. $Slip_c$ ($Guess_c$) are defined as the probability of a student who has mastered (not mastered) $c$ slipping (guessing) and violating (satisfying) the constraint. In the third and fourth columns of Table VIII, $P(\text{Performance}_{c,p} = \text{NOT-RELEVANT}) = 1.0$ and the other entries are 0, because these represent the two scenarios where $\text{RelevantSS}_{c,p} = \text{NO}$ (i.e. $c$ is not relevant to the student solution). The four columns
represent situations where the values of the parent nodes are known with certainty. In practice, the values of the parents will not be known with certainty.

The Bayesian network is used to predict the probabilities of the student violating, satisfying or not using $c$ in his/her solution to $p$. A simple example will illustrate the evaluation process. Let us take the following constants: $\alpha_p = 0.9, \beta_p = 0.1, Slip_c = 0.3, Guess_c = 0.05$. Now, suppose that $c$ is relevant to problem $p$’s ideal solution (i.e. $P(\text{RelevantIS}_{c,p} = YES) = 1$) and the student is not likely to have mastered $c$ (e.g. $P(\text{Mastered}_c = YES) = 0.25$). An evaluation of the network yields the probability distribution $[P(\text{Performance}_c = VIOLATED) = 0.709, P(\text{Performance}_c = SATISFIED) = 0.191, P(\text{Performance}_c = NOT-RELEVANT) = 0.1]$.

### 8.3. Selecting problems

A single problem requires mastery of many constraints before it can be solved. The number of relevant constraints per problem ranges in SQL-Tutor from 78 for the simplest problems, to more than two hundred for complex ones. It is therefore necessary to select an appropriate problem for a student on the basis of his or her current knowledge.

We determine the value of a problem by predicting its effect on the student. If the student is given a problem that is too difficult, he/she will violate many constraints. When given a simple problem, they are not likely to violate any constraints. A problem of appropriate complexity is the one that falls into the zone of proximal development, defined by Vigotsky (1978) as “the distance between the actual development level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or collaboration of more capable peers”. Therefore, a student should be given a problem that is slightly above their current level but not so difficult as to discourage the student.
Let us discuss the strategy we propose for selecting problems. Each violated constraint triggers a feedback message. If the system poses a problem that is too difficult, there will be many feedback messages coming from various violated constraints, and it is unlikely that the student will be able to cope with them all. If the problem is too easy, there will be no feedback messages, as all constraints will be satisfied. A problem of appropriate complexity will generate an optimal number of feedback messages. This is the basis of the evaluation function we propose.

The algorithm for evaluating problems is given in Figure 9. The function takes two parameters, the problem $p$ to be evaluated and an integer, $OptimalFeedback$. It returns the value of $p$. $OptimalFeedback$ is an argument specifying the optimal number of feedback messages the student should see regarding the current problem. Its value is currently set to the student’s level + 2, reflecting the fact that novices are likely to cope well with a small number of messages at a time, while advanced students are able to resolve several deficiencies in their solutions simultaneously.

```c
int Evaluate(problem p, int OptimalFeedback) {
  int Feedbacks = 0;
  For every constraint c {
    Evaluate the Bayesian network;
    If P(Performance\(c_p = VIOLATED\)) > 0.45
      Then Feedbacks := Feedbacks + 1; }
  Return (- |OptimalFeedback – Feedbacks|); }
```

**Fig. 9.** The problem evaluation function.

The evaluation function assumes that feedback will be generated for every constraint where P(Performance\(c_p = VIOLATED\)) > 0.45. This heuristic is used because it is intractable to calculate the exact probability of a problem producing the optimal number of feedback messages. The 0.45 value was chosen because initial tests showed that it gave best the results.
The problem with the highest value is selected from the pool of unsolved problems within 1 level of the student’s level.

8.4. Results

All actions students performed in the study were logged, and later used to analyse the effect of the proposed problem-selec**tion** approach on learning. Both groups of students had two ways of selecting problems; they could go through all problems in order, or they could ask the system to select an appropriate problem based on the student model. In the case of the control group, the problem is selected based on the simple heuristic discussed in section 8, while the Bayesian approach was used for the experimental groups.

In order to evaluate the proposed problem selection method, we identified the logs of students who used system's choice in both groups. Six students from the experimental group attempted 36 problems selected by next problem and 38 problems selected by system’s choice using the new Bayesian approach. Thirteen students from the control group worked on 106

<table>
<thead>
<tr>
<th>Average attempts</th>
<th>Exper. group</th>
<th>Control group</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>SD</td>
</tr>
<tr>
<td>Next problem</td>
<td>3.96</td>
<td>1.5</td>
</tr>
<tr>
<td>System’s choice</td>
<td>3.49</td>
<td>1.12</td>
</tr>
<tr>
<td>Gain</td>
<td>0.47</td>
<td>1.26</td>
</tr>
</tbody>
</table>

**Table IX.** Average number of attempts per solved problem
and 79 problems selected by next problem and the original system’s choice respectively.

Both problem selection strategies try to keep the student focused on a concept they are having difficulty with. However, as stated earlier, the focus of the heuristic-based problem selection strategy is too narrow, and it often selects problems which are either of the wrong difficulty, or which introduce new concepts as well as the target concept. We would therefore expect it to cause an increase in the problem solving effort, when compared to "next problem", since the default ordering is of (roughly) increasing difficulty, and introduces new concepts in a fairly orderly fashion. Conversely, we would expect a decrease in required effort when students use the Bayesian problem selection strategy, because it overcomes these difficulties, while introducing concepts in an order appropriate to the student. The results in Table IX illustrate this: for the students in the control group, the problems selected by the heuristic problem selector ("system’s choice") are significantly more difficult than those selected by the “next problem” option (t=2.69, p=.015, power is 0.73 for the significance threshold of 0.05). Conversely, the effort required by the students in the experimental group when using the Bayesian method decreases by a small, not significant, amount (0.47). Table IX also reports the "gain" caused by using "system choice", i.e. the decrease in effort that resulted. When we compare this for the two groups, we get a significant difference (p=.007, t=3.08). This statistic has a large effect size (0.36) and power of 0.83 at significance threshold of 0.05. Therefore, the results strongly suggest that the new problem solving strategy is superior in its ability to select problems of appropriate difficulty.

The advantages of the Bayesian approach are clearer when we observe what happens during the problem solving session. The students start with simple problems, and progress to more complex ones. Figure 10 illustrates the average number of attempts students took to solve the $i^{th}$ problem, for the experimental group. It can be seen that the initial problems selected by the next problem option are easier for students than those selected by the Bayesian
This fact is easily explained by the fact that the Bayesian network progresses faster to more complex problems. However, later problems selected by the Bayesian approach are more adapted to the student and therefore require fewer attempts to be solved. Figure 11 illustrates the number of attempts per problem for the students in the control group. The opposite trend is obvious here: students find the system-selected problems (which are selected by using the simple heuristic discussed at the beginning of this section) more challenging than those visited in turn.

**Fig. 10.** The average number of attempts per problem for the experimental group
Pre- and post-tests, given in Appendices B and C, consisted of three multi-choice questions each, of comparable complexity. The marks allocated to the three questions were 1, 5 and 1 respectively. Nine out of fourteen students in the experimental group and sixteen out of eighteen in the control group submitted valid pre-tests, the results of which are given in Table X. There is no significant difference between the mean scores of the two groups ($t=0.656$, $p=.52$), showing that the control and experimental groups contained a comparable cross-section of students. However, a number of factors, such as the short duration of the user study, the holding of the study during the last week of the year etc, conspired to result in a very small number of post-tests being completed. Because some students did not log off, they did not sit the post-test that was administered on a separate Web page. Only one student from the

**Fig. 11.** The average number of attempts per problem for the control group

### 8.5. Pre/post tests
control group and four from the experimental group sat the post-test. As the result, we can draw no conclusions from the posttest results.

### 8.6. Related Work

Other researchers have proposed the use of Bayesian networks in ITSs. ANDES (Conati et al. 1997, Gertner 1998), an ITS for teaching Newtonian physics, uses Bayesian networks for predicting student performance and problem solving behaviour. The ANDES network has a dynamic component, comprising nodes specific to the current problem, and a static component, comprising nodes representing the student’s knowledge. The dynamic component is constructed on-line when a new problem is started. However, this approach relies on the system knowing \textit{a priori} which rules can be relevant to the problem’s solution. This is not the case in the SQL domain where the correct solution known by the system is only one example of a correct solution. The usefulness of the ideal solution in predicting the student solution is determined by the $\alpha_c$ and $\beta_c$ parameters. Thus, in the SQL domain, we are forced to model the entire domain for each problem.

<table>
<thead>
<tr>
<th>Question</th>
<th>Exper. group</th>
<th>Control group</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.20</td>
<td>0.25</td>
</tr>
<tr>
<td>2</td>
<td>3.20</td>
<td>2.73</td>
</tr>
<tr>
<td>3</td>
<td>0.60</td>
<td>0.73</td>
</tr>
<tr>
<td>Total</td>
<td>4.00</td>
<td>3.50</td>
</tr>
</tbody>
</table>

*Table X. Means for the pre-test*
One approach that does model the entire domain is Collins et al.’s (1996) hierarchical Bayesian network model for student modeling and performance prediction on test items. A similar hierarchical model was initially intended for our probabilistic student model. However, the key difference between our domain and Collins' example is that SQL-Tutor contains more than 500 constraints whereas Collins' example consists of only 50 questions. Initial investigations showed that it was infeasible to evaluate a traditional Bayesian network modeling more than 500 constraints on-line. Furthermore, Collins' example domain of elementary arithmetic divides neatly into 10 categories (e.g. addition theory, subtraction theory etc) whereas in SQL there is no such simple classification of constraints.

Finally, Reye (1998) proposes a dynamic Bayesian network model for student modeling. Each variable, corresponding to a single knowledge item, is dynamically updated over time using Bayesian probability theory as the student's performance is observed. Again, this is a similar scheme to our student model where single constraints are represented by single nodes. However, Reye's model makes each knowledge item probabilistically independent. This simplification makes Bayesian student modeling tractable, but for solving decision tasks such as problem selection the probabilities do need to be combined. Reye does not show how this can be done, whereas this is the main emphasis of our approach.

8.7. Discussion

One of the vital tasks an ITS has to perform is to provide problems that are of appropriate complexity for the student’s current knowledge. In our approach, a probabilistic, long-term student model is used to predict student performance on candidate problems. The value of a problem depends on the predicted the number of errors the student is likely to make. Each error would result in a feedback message. Novices are unable to deal with many feedback messages, while advanced students can, and therefore an optimal number of feedback
messages can be established based on the current student’s level. Of all available problems, we select the problem that generates the optimal number of feedback messages.

Initial evaluations indicate that the proposed solution is promising. However, we implemented several heuristics due to the inefficiencies of evaluating large Bayesian networks on-line. For example, both Table VI and Figure 9 depict heuristics used by the system. Ideally the system should use theoretically sound rules based on probability theory and/or decision theory. Future work will look at developing this further. Use of new technologies such as qualitative Bayesian networks (Chao-Lin & Wellman 1998), which are known to be much faster in their evaluation time than traditional Bayesian networks, may also make the development of large-scale Bayesian networks feasible.

Future research will also focus on other decision tasks that an ITS must solve. Problem selection is only one, and other tasks include topic selection, adapting feedback, hint selection, and selective highlighting of text. We are working towards a general framework for solving these types of problems (Mayo 2000, Mayo & Mitrovic 2000).

9. Conclusions

In this paper we reported on three evaluation studies performed on SQL-Tutor. These studies included elements of formative evaluation, which enabled us to improve the system over time. The studies also involved elements of summative evaluation. We analysed Constraint Based Modeling as a student modeling approach, and showed that CBM has sound foundations and that it can be extended successfully to provide for long-term modeling of student’s knowledge and also to support pedagogical decision making.

As Chin (2000) points out, there are many obstacles to evaluating user-adaptive systems. Even when the factors to be studied are identified, and the study is designed properly, there are numerous outside factors that can influence the outcomes. All systems that are aimed at
humans suffer from problems with using different groups of users, who may have different backgrounds, (dis)abilities, learning styles, motivations etc, which are very difficult to account for. Chin also identifies other problems, such as the practice effect, problems caused by the experimental set up and the others. Stern and Sterling (1997) point out to the enthusiasm factor with the opposite effect. They note that the enthusiasm of researchers and students involved in the evaluation of an experimental learning environment will contribute to the results of learning. Students generally react well to the changes in classroom routine.

We have experienced many of these problems. In study 1, we reported on significant improvement of performance of the experimental group in comparison to the control group. However, this study was not controlled, as we asked for volunteers. Furthermore, we cannot be absolutely sure that the students in the control group abstained from using the system, as they might have heard about it from their peers. In study 2, we experienced problems with the experimental design, which offered feedback of mixed types to the experimental group. It was therefore necessary to post-hoc split the experimental group into two subgroups, which also has negative effects on the validity of the results.

We appreciate Chin’s (2000) effort to propose guidelines for designing evaluation studies and believe that many of them are generally applicable. However, in our experience, some of them are difficult or even impossible to apply to educational systems. For example, Chin suggests that subjects are put into groups randomly, with random allocation of times. We were constrained to the scheduled lab times, and it was not an option to change the allocation of times to students. Furthermore, since the studies were carried out in departmental labs, we had no control of the environment, which is another of Chin’s guidelines. In study 3, timing was a constraint, as students needed to get an overall understanding of databases prior to using the system. The only possible time for this study was the last week of lectures, which
meant the very last day (Friday afternoon) of the school year for one set of students. This had a negative effect of the number of participating students, and their motivation.

Another illustration of possible problems comes from study 3. We did not anticipate the problems with students logging off before filling out the post-test and, as the result, collected very few post-tests. Therefore we cannot compare the performances of students on pre- and post-test. Also, we did interact with the students during the sessions, giving additional explanations, because our main goal is to help students to learn better. This illustrates another aspect of the ethical dilemma related to evaluating educational systems, which we discussed in section 6.2. Our primary role as teachers is to help students learn, and that interferes with our goals as researchers.

All three studies involved a small number of students, and consequently we did not have enough data to evaluate the system on. The studies were short, consisting of a single 2-hour session per student, and a big improvement in students’ performance cannot be expected. Furthermore, we do not get a lot of opportunities for evaluation, because there are only one or two relevant courses each year. The existence of a Web-enabled version of the system will provide us with more data, but it will be harder to analyse it, since little or nothing is known about the background of the users.

In spite of all the above problems, we believe that our results are important, and that they will inspire more research on CBM. We hope that our experiences in evaluating SQL-Tutor will be beneficial to IES researchers and the broader user-adaptive systems area.
Appendix A: User questionnaire

1. What is your previous experience with SQL?
   a) only lectures   b) lectures plus some work   c) extensive use

2. How much time did you need to learn about the system itself and its functions?
   a) most of the session
   b) 30 minutes
   c) 10 minutes
   d) less than 5 minutes

3. How much did you learn about SQL from using the system?
   Nothing     Very much
   1 2 3 4 5

4. Did you enjoy learning with SQL-Tutor?
   Not at all     Very much
   1 2 3 4 5

5. Would you recommend SQL-Tutor to other students?
   a) Yes     b) Do not know     c) No

6. Do you find the interface easy to use?
   Not at all     Very much
   1 2 3 4 5

7. Do you find the display of the schema understandable?
   a) Yes     b) Do not know     c) No

8. Do you find feedback useful?
   Not at all     Very much
   1 2 3 4 5

9. Would you prefer more details in feedback?
   a) Yes     b) Do not know     c) No
10. How often did you use "System's Choice" to have the system select a problem for you to solve?
   Never 1 2 3 4 5

11. If you have used "System's Choice", how do you rate the difficulty of the problems SQL-Tutor selected for you?
   Always too easy 1 2 3 4 5

12. Did the problems selected by "System's Choice" target SQL concepts that you feel were appropriate at the time? Please comment.
   Never appropriate 1 2 3 4 5

13. Did you feel that the problems selected by "System's Choice" were better or worse than those that would have been selected by a human tutor?
   Always worse 1 2 3 4 5

14. Did you encounter any software problems or crashes?
   a) Yes  b) No

15. What do you like in particular about SQL-Tutor?

16. Is there anything you found frustrating about the system?
Appendix B: Pre-test

The questions are based on the MOVIE table. Each movie has a unique number, which is the primary key of the table. Additionally, we have the title, the year of production (YEAR), the critic's rating (CRITICS), the type of movie, the number of Academy awards the movie was nominated for (AANOM) and has won (AAWON), and the number allocated to the director of the movie (DIRECTOR).

Please answer ALL questions:

1. We need to find the titles of all movies other than comedies. Will the following SQL statement achieve that? Yes/No

   SELECT TITLE
   FROM MOVIE
   WHERE TYPE = NOT('comedy')

2. Now, we need to find the title of the movie that has won the most awards. Select ALL correct answers:

   a) SELECT TITLE
      FROM MOVIE
      WHERE AAWON = MAX(AAWON)

   b) SELECT TITLE
      FROM MOVIE
      GROUP BY NUMBER
      HAVING AAWON = MAX(AAWON)

   c) SELECT TITLE
      FROM MOVIE
      WHERE AAWON = (SELECT MAX(AAWON) FROM MOVIE)

   d) SELECT TITLE
      FROM MOVIE
      GROUP BY TITLE
      HAVING AAWON = (SELECT MAX(AAWON) FROM MOVIE)
e) SELECT TITLE
   FROM MOVIE
   WHERE AAWON>=ALL (SELECT AAWON FROM MOVIE WHERE AAWON IS NOT NULL)

3. We need to find the total number of awards won by comedies in 1983. Which of the following statements will achieve that?

a) SELECT SUM(AAWON)
   FROM MOVIE
   GROUP BY TYPE
   HAVING TYPE IN ('comedy') AND YEAR=1983

b) SELECT SUM(AAWON)
   FROM MOVIE
   WHERE TYPE='comedy' AND YEAR=1983

c) SELECT SUM(AAWON)
   FROM MOVIE
   WHERE TYPE='comedy' AND YEAR=1983
   GROUP BY NUMBER
Appendix C: Post-test

The questions are based on the MOVIE table. Each movie has a unique number, which is the primary key of the table. Additionally, we have the title, the year of production (YEAR), the critic's rating (CRITICS), the type of movie, the number of Academy awards the movie was nominated for (AANOM) and has won (AAWON), and the number allocated to the director of the movie (DIRECTOR).

Please answer ALL questions:

1. We need to find the titles of all comedies or dramas. Is the following SQL statement correct? Yes/No

   SELECT TITLE
   FROM MOVIE
   WHERE TYPE='comedy' OR 'drama'

2. What is the type of movie that had the highest number of movies made in 1980? Select ALL correct answers:

   a. SELECT TYPE
      FROM MOVIE
      WHERE YEAR=1980
      GROUP BY TYPE
      HAVING MAX(COUNT(*))

   b. SELECT TYPE
      FROM MOVIE
      WHERE YEAR=1980
      GROUP BY TYPE
      HAVING COUNT(*)>=ALL (SELECT COUNT(*) FROM MOVIE WHERE YEAR=1980 GROUP BY TYPE)

   c. SELECT TYPE
      FROM MOVIE
      WHERE YEAR=1980 AND COUNT(*) = (SELECT MAX(COUNT(*)) FROM MOVIE WHERE YEAR=1980)
      GROUP BY TYPE
d. SELECT TYPE, MAX(COUNT(*))
   FROM MOVIE
   WHERE YEAR=1980
   GROUP BY TYPE

a. SELECT TYPE
   FROM MOVIE
   WHERE YEAR=1980 AND NUMBER = MAX (COUNT(*))

3. We need to find the total number of awards won by comedies in 1983. Select all of the following statements that will achieve that?
   
a) SELECT COUNT(*)
      FROM MOVIE
      WHERE YEAR IN (1981, 1982, 1983) AND TYPE='drama'
   
b) SELECT COUNT(*)
      FROM MOVIE
      WHERE TYPE='drama'
      GROUP BY YEAR
      HAVING YEAR=1983 OR YEAR=1982 OR YEAR=1981
   
c) SELECT COUNT(*)
      FROM MOVIE
      WHERE YEAR>=1981 AND YEAR<=1983
Acknowledgements

The work presented here was supported by the University of Canterbury research grant U6242.

References


Bloom, B.S.:1984, 'The 2 Sigma Problem: the Search for Methods of Group Instruction as Effective as one-to-one Tutoring'. Educational Researcher 13, 4-16.


Content

1. Introduction .................................................................................................................. 3
2. SQL-Tutor ..................................................................................................................... 4
3. CBM ............................................................................................................................ 8
4. Evaluation Studies ....................................................................................................... 11
5. Subjective evaluation ................................................................................................. 13
6. Study 1: Learning constraints...................................................................................... 14
7. Study 2: Evaluating feedback ..................................................................................... 18
8. Study 3: Evaluating a probabilistic student model .................................................... 28
9. Conclusions ................................................................................................................ 41
Appendix A: User questionnaire ..................................................................................... 44
Appendix B: Pre-test ....................................................................................................... 46
Appendix C: Post-test ..................................................................................................... 48
Acknowledgements ......................................................................................................... 50
References ..................................................................................................................... 50

List of Figures

Fig. 1 Architecture of SQL-Tutor
Fig. 2. Interface of the Web-enabled version of SQL-Tutor
Fig. 3. Responses from the user questionnaire
Fig. 4. Mastery of constraints
Fig. 5. The probability of constraint violation for the full and limited groups
Fig. 6. The probability of constraint violation for the three groups
Fig. 7. The probability of constraint violation after feedback
Fig. 8. A Bayesian network for predicting student performance on a single constraint
Fig. 9. The problem evaluation function
Fig. 10. The average number of attempts per problem for the experimental group
Fig. 11. The average number of attempts per problem for the control group
List of Tables

Table I. Some statistics about the subjective evaluation
Table II. Competence of the two groups in study 1
Table III. Examples of various types of feedback messages
Table IV. Statistics for the three groups
Table V. The effect of feedback on whole sessions
Table VI. Heuristics used for updating the student model
Table VII. $P(RelevantSS_{c,p} | RelevantIS_{c,p})$
Table VIII. $P(Performance_{c,p} | RelevantSS_{c,p}, Mastered_{c,p})$
Table IX. Average number of attempts per solved problem
Table X. Means for the pre-test