Personalised Correction, Feedback, and Guidance in an Automated Tutoring System for Skills Training

Abstract
In addition to knowledge, in various domains skills are equally important. Active learning and training are effective forms of education. We present an automated skills training system for a database programming environment that promotes procedural knowledge acquisition and skills training. The system provides support features such as correction of solutions, feedback and personalised guidance, similar to interactions with a human tutor. Specifically, we address synchronous feedback and guidance based on personalised assessment. Each of these features is automated and includes a level of personalisation and adaptation. At the core of the system is a pattern-based error classification and correction component that analyses student input.

Keywords
automated tutoring; correction; personalisation; feedback; guidance; skills training; database programming.

1 Introduction

The delivery of feedback is an integral part of learning processes (Heaney and Daly, 2004). A human tutor offers the student various forms of feedback; an automated online tutor should also do this. This feedback should be relevant, precise and understandable. The level a student reaches when learning is often proportional to a student's engagement with a teacher or an activity. In computer-aided learning, feedback is of central importance in particular if a human tutor is not always available.

We present an automated, computer-based tutoring system that supports a skills training environment for the database language SQL. Database programming and querying is one of the core skills for computer scientists and software developers, but is also important for other information technology users. In the computing domain, formalisable languages such as SQL are particularly suitable to be supported by automated tutoring systems. SQL is vital for the definition, manipulation and retrieval of information from a database system (Pahl, Barrett and Kenny, 2005). We refer to this system as a tutor as it offers features that a human tutor would offer:

- In particular, it provides feedback for the student that is of a contextually high quality. The system allows a knowledge- or skills-level interaction with the content through programming activity and synchronous contextual feedback (Ravenscroft, Tait, and Hughes, 1998).
- Furthermore, by automating the tutoring process, students can individually tailor their learning environment by defining feedback preferences and choosing their own learning paths through the system's curriculum.

The student benefits from a system that is always available and that analyses and corrects a submission and offers feedback and personalised guidance based on the analysis results.

Our primary objective is to investigate an integrated approach to correction, domain-specific feedback and personalised guidance features. Tutoring systems have often neglected the guidance dimensions, which supports a learner over longer sessions and beyond. At the core

of this approach is a correction technique that allows personalised domain-specific feedback and guidance.

- We develop techniques to analyse the SQL select statement in order to ascertain difficulties a typical novice student might encounter while creating these statements. These errors are categorised according to a multi-dimensional error classification scheme.
- We determine adaptivity techniques for use in a knowledge-based feedback and guidance system. These are used to develop a system that lets content and navigation be adapted by feedback and guidance features.

The system is evaluated through a combination of different instruments. SQL is a suitable topic to explore these issues, but they apply equally to other computer-processable languages, ranging from textual to graphical languages (Kenny, 2006).

2 Background

Guidance, Feedback, and Personalisation

In the educational context, feedback is strategically useful advice given to a student based on tasks previously attempted. Its objectives are self-reliant learning and competency in a domain. Feedback can be defined as being local or global [Melis and Ullrich, 2003). Local feedback aims to help correct the student's attempt at solving a specific problem. Global feedback uses several aspects of the whole learning process to coach the student in a more general manner.

The challenge in computer-aided learning is to achieve personalisation in a learning experience based on skills training and activity. Adaptive hypermedia systems are personalisation systems that build a model of the goals, preferences and knowledge of each individual user, and use this model throughout the interaction with the user in order to adapt to the needs of the user (Brusilovsky, 2000). Adaptive teaching and learning can be provided at two main levels (Boyle, 1997). The first is the adaptation of presentation of content. The second is navigational guidance at the link level (De Bra and Stash, 2004). Guided discovery places an emphasis on the student's ability to proactively seek information. It is a move from teacher-controlled learning to student-controlled learning and learning organisation (Stephenson, 2001).

Intelligent Tutoring Systems and Pattern Matching

An Intelligent Tutoring System (ITS) is a computer-based instructional system with models of instructional content that specify what to teach, along with teaching strategies that specify how to teach (Murray, 1999). ITSs have been shown to be highly effective at increasing students' performance and motivation (Beck, 1996), although ITS in the past have often been restrictive, limiting the student's control of the learning experience. A traditional ITS has four distinct components – an expert model, a student model, an instructional or pedagogical model, and an instructional environment or user interface (Boyle, 1997).

- The expert model, or domain model, contains knowledge of the domain or subject area that needs to be communicated to the student. Knowledge is normally represented as a set of facts or rules (Chou, 2002).
- The student model holds information about the student (personal details, learning style, learning preferences, etc.), along with a representation of the knowledge s/he holds (which is matched against the expert model).

- The pedagogical model has knowledge of teaching strategies to determine when and how to instruct the student. It makes decisions about the topic, the problem, and feedback.
- The interface acts as the means of communication between the student and the ITS, receiving input and presenting information.

Pattern matching or pattern recognition is a method that can be used in ITS to define ideal solutions or ideal learning paths. For instance, it can be used as a means of correcting student work. CAPIT's student modeller, for instance, is a pattern matcher that takes a learner's solution to a problem and determines which constraints are violated (Mayo and Mitrovic, 2001). Pattern matching can also be used to ascertain a higher level of student understanding. The Tactical Action Officer TAO applies pattern-matching rules to detect sequences of actions that indicate whether the student understands an activity (Stottler and Vinkavich, 2000).

Skills Training, Apprenticeships, and Scaffolding

Stephenson (2001) argues that experience is the foundation of and the stimulus for learning. Learning is primarily developed through activity. Skills training involves higher levels of activity and interaction than typical acquisition of declarative knowledge. The student trains mainly by practising a task. An apprenticeship is concerned with procedural knowledge acquisition and skills training. Traditional apprenticeship is a form of teaching and learning that has been used successfully throughout the ages, primarily for practical tasks. Apprenticeship is a three-step process involving a master and an apprentice. Initially the master demonstrates the completion of the various stages of a task while the apprentice observes. The apprentice works at the task while the master observes and offers advice. The apprentice practises in a controlled environment. The apprentice eventually achieves competency and self-reliance. Traditional apprenticeship is a blend of scaffolding, fading and coaching.

- Scaffolding is a temporary support while completing a task or activity. The key idea behind scaffolding is to provide a student with timely support at an appropriate level. Collins et al. (1991) refer to scaffolding as being a set of limited hints and feedback.
- Ideally, scaffolding will be faded, meaning it will be removed gradually, thus encouraging the student to work in a self-reliant manner.
- Coaching is the process of overseeing the student's learning (Collins et al., 1991). It involves formulating the course of work the learner should take, providing timely scaffolding at the appropriate level and fading it accordingly, and offering meaningful feedback and encouragement.

Cognitive apprenticeship moves the traditional apprenticeship into the classroom and the cognitive domain (Collins et al., 1991). A major principle of cognitive apprenticeship is collaboration and conversation with a master (Winnips, 2003). Cognitive apprenticeship uses the idea of situated learning, whereby the learner is placed in a real-world environment (Collins, 1988). The virtual apprenticeship model (Murray et al., 2003) applies cognitive apprenticeship to the Web context, and is therefore a suitable concept for web-based learning. This model uses scaffolding and activity-based learning to allow the student to construct knowledge, practise skills and gain experience. The construction of artefacts and a realistic or even authentic setting are vital (Herrington and Oliver, 2000).

3 System Architecture and Component Functionality

System Architecture

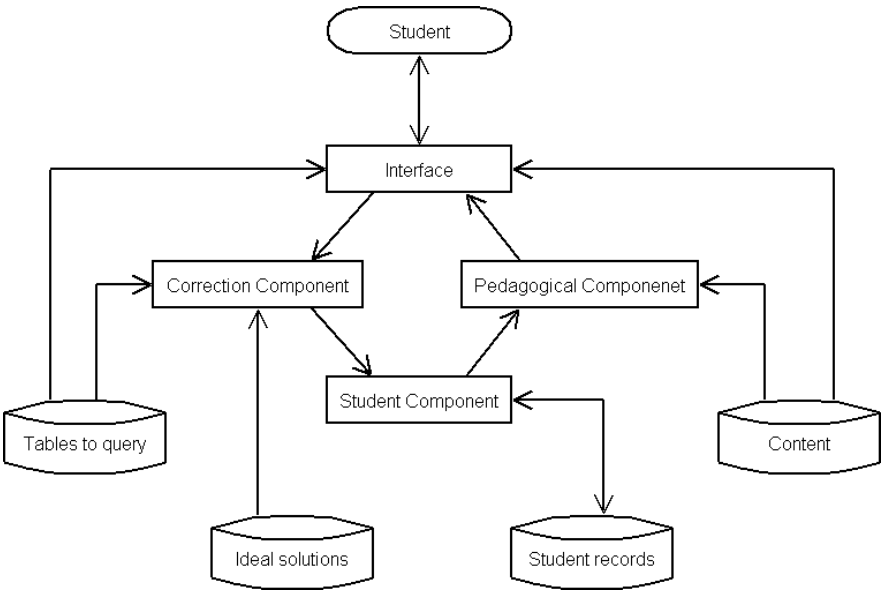

Figure 1. Architecture of SQL Tutoring System

The tutoring system is a learning coach for students studying SQL that offers a range of exercises (SQL questions). We use a component-based architecture similar to ITS architectures, see Figure 1, with an interface, an expert model (here called the correction component), a student component and a pedagogical component. The main components are supported by information objects, which store:

- the tables that the student should query in the various exercises,
- a collection of ideal solutions that are used by the correction component,
- student records to store general and topic-specific information about the student,
- content that the student may be directed towards by the pedagogical component.

The student component is responsible for arranging information about the student for storage, and for retrieving relevant student information from the database. The correction component sends to the student component information regarding the total number of attempts and the number and types of errors made by the student for each question, which can be retrieved per session or per set of sessions. We outline interface and pedagogical component now and discuss the correction component later on in detail.

Interface Component

The navigation approach is based on a virtual apprenticeship format and guided discovery. Our tutoring system accommodates the second and third stages of the apprenticeship model by allowing students to practise a task with and without help from the system. Virtual apprenticeship dictates that the student's learning should be supported by scaffolding. Within the guided discovery paradigm, the student assumes control of some learning experience factors. The student can control and view feedback and guidance at a chosen level. A default system setting in the pedagogical component, which implements scaffolding and fading, changes the levels of feedback and guidance based on number and success of student attempts.

The student selects a question and submits a solution attempt. The number of attempts and successes is recorded in the student records. The student's query is sent to the correction model for correction. A correct query is sent to a database server and a result is returned. The correction model returns an error flag to the interface. If the question is answered incorrectly, the student may, depending on chosen preferences, receive scaffolding in the form of feedback or guidance, which includes hints and, after a number of unsuccessful attempts, also correct solutions.

Pedagogical Component

The pedagogical component contains the teaching strategies of the system. This component decides what level of feedback and guidance to return to the student by examining the personal preferences and the errors made in the questions attempted. It is the element in charge of the personalisation of scaffolding.

The pedagogical process is initiated by the interface. If the student has made one or more errors in a particular question the interface sends the student's feedback and guidance preferences to the pedagogical model. The system offers a mixed locus of control of feedback and guidance, meaning the student or the system can choose the type and level of feedback to be displayed. The pedagogical model determines, according to preferred feedback and guidance levels, the further actions to be taken. In the default option, the system gradually increases the levels from minimum to maximum as the student repeatedly unsuccessfully attempts a question. The scaffolding will plateau at the maximum levels possible. This increase of feedback and guidance is a form of scaffolding, and so it is also faded. Consequently, both forms of scaffolding are gradually decreased.

4 Error Classification and Correction Component

Error classification is the first step to providing the student with adequate and constructive personalised feedback. The SQL select statement is the focus of our error classification. Multiple concepts covered by select statements can be applied to other SQL statements, which make it a suitable topic from an educational point of view.

SQL Problems and Solutions

The select statement is a fundamental SQL statement, used to query and extract information from a database (Ramakrishnan and Gehrke, 2003). It can be made up of six clauses, but we focus here on the three central ones – SELECT, FROM, WHERE – whereas the others are dealt with in (Kenny, 2006). These map to input elements (from), a condition that is the extraction filter (where), and an output description (select). An example is:

SELECT colour FROM parts WHERE weight > 10 and city = "Paris".

A list of database table names can be provided in the FROM clause. The SELECT clause can contain a list of column names of tables named in the from-list, possibly combined with aggregation operators such as minimum or average. The qualification in the WHERE clause is a condition or filter constraint based on logical and comparison operators.

The nature of SQL select statements means there is not always a single correct solution for a given question. However, we can produce an ideal solution that will contain the minimum number of elements possible in a correct solution. Some semantically equivalent variations between the student solution and the ideal solution are, however, accepted:

- the re-ordering of attributes and table names within the one clause,
- syntactical equivalence or operators,
- sometimes, using attribute prefixes is semantically equivalent to the absence of prefixes,
- redundant elements that are not part of the select clause do not affect output,
- using table aliases results in the same output as their non-use.

An example for the first case are SELECT A, B FROM C, D and SELECT B, A FROM D, C, an example for the second are <> and != as inequality operators.

Abstract Notation for Ideal Solutions

Each question posed by our tutoring system addresses a specific SQL problem or SQL construct. Targeted constructs and problems are usually only part of a full select statement. Consequently, the statement, be it the ideal solution or the student attempt, needs to be decomposed into individual elements. An abstract notation was developed to allow the system to identify element types in a solution.

We chose a pattern matching approach as the classification and correction technique. An abstract notation is used to classify any errors the student has made by matching the student's solution with an abstracted form of an ideal solution. The proposed ideal solution is a statement that contains the minimum amount of constructs needed to correctly answer the question. The aim of the abstraction is to cater for semantically equivalent solutions. We refer to basic expressions in the ideal solution schema (e.g. attributes or tables) as elements. Each relates to a string token in the ideal solution. The grammar of the SQL language serves as the basis of the pattern-based matching by providing these element types as abstractions. Every element in the abstract notation has a particular purpose within the query and is derived from the grammar. We can therefore directly equate our SQL elements with an SQL grammar.

Table 1. SQL Query Example with Ideal Solution and Solution Schema

Question text	Get distinct colour and city for non-Paris parts with weight greater than 10
Proposed ideal solution	*SELECT distinct colour, city* *FROM p* *WHERE city <> 'Paris' and weight > 10*
Ideal solution schema	*SELECT misc attr attr* *FROM table* *WHERE att cop misc lop att cop misc* with *table*: a table name *attr*: an attribute name *prefix*: a table name prefix *cop*: an SQL comparison operator *lop*: an SQL logical operator *misc*: a miscellaneous item

We can see from the ideal solution in the example in Table 1 that there are a minimum number of elements present to fulfil the question requirements. Each element maps directly to

its type in the solution. The correction model can ascertain the specific elements that need to be part of the student's solution, and the types of these elements. In the example, the system would know that *p* is the name of a table, *colour* and *city* are attributes, <> is a comparison operator, and *distinct*, *'Paris'* and *10* are miscellaneous items.

Error Classification for SQL Select Statements

We introduce a multi-dimensional error categorisation scheme (summarised in Figure 2) that incorporates the dimensions formal language facets, clause types and common elements to locate where and how the student has made a mistake. This allows us, using the instructor's experience, to put the error classification in an educational perspective. It is important to base error categories on typical difficulties the students may encounter. The correction component uses the error categorisation scheme to determine the type of mistake made by the student and the pedagogical model presents these to the student in a personalised and educationally sound way.

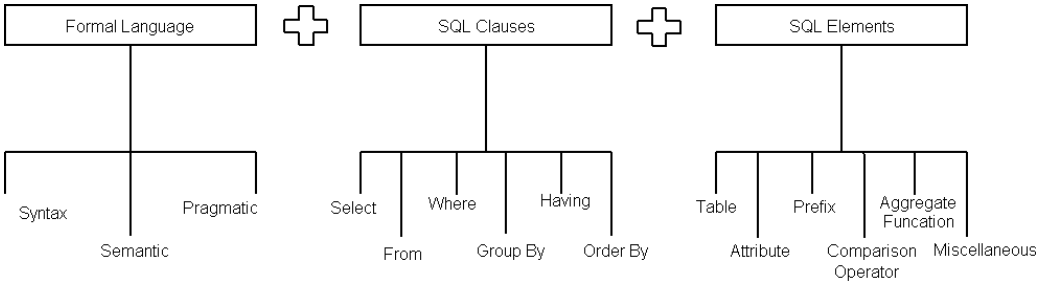

Figure 2. Multi-dimensional Error Classification Scheme

As SQL is a formal language, its aspects can be classified into syntactic or semantic. Errors made by the student can be categorised in this first dimension into two central categories:

- Syntax errors are caused by problems with the actual syntax of the language; for example, misspelling a keyword or selecting an attribute that does not exist.
- Semantic errors occur when a statement is created that is syntactically correct, but which does not reflect the student's intentions correctly. Suppose a student wishes to select the names of employees who are over the age of thirty. Instead of using the greater-than symbol (>) the less-than symbol (<) might be used. The names of employees under the age of thirty instead of over would be selected.

The second dimension is based on the clauses of the SQL statement. Each clause performs a particular function and so addresses a different semantic issue. For instance, the select clause specifies the output of the query. We categorise any errors made by the student based on these SQL select statement clauses, which allows us to identify the semantic difficulty that a student is having in relation to the specific function of a clause.

In addition to the clause-specific error dimension, we also categorise any errors made by the student based on these elements used in the clauses in this third dimension, which complements the clause-specific errors:

- Table names: Students may select the wrong tables, particularly when more than one table is to be selected, or when a table needs to be selected but will not appear in the result.
- Attributes: Students may select incorrect attributes, particularly when the attribute will not appear in the result.

- Prefixes: Prefixes may be needed in a statement where two or more attributes with the same name exist in different tables.
- Comparison operators: A student may inappropriately use a comparison operator.
- Aggregate functions: The student may use an incorrect aggregate function.
- Miscellaneous items: This category is included for minor aspects that do not adhere to the above categories.

The element errors are object-specific, whereas the clause errors are function-specific.

The multi-dimensional error categorisation scheme shall be illustrated. Consider two tables, s and sp, that capture information about suppliers and shipments of parts.

- *Question:* Get the numbers and names of all suppliers.
  Student Answer (incorrect): SELECT sno, name FROM s.
  System Diagnosis: This statement is incorrect because the student has tried to select the attribute "name". We assume that the correct attribute is "sname". This can be identified as primarily a function-specific error in the select clause, i.e. the output of the statement is affected through an attribute element.
- *Question:* Get the maximum status of suppliers in Paris.
  Student Answer (incorrect): SELECT min(status) FROM s WHERE city = 'Paris' .
  System Diagnosis: In this case, the student has selected the minimum status instead of the maximum status. Here, the system identifies a primarily element-specific semantic error due to the misuse of the aggregate function "min" on element "status" in the select clause.

The identification of the primary error is based on heuristics defined by database instructors. The system also identifies the categories of errors when the student has made more that one mistake. The number and ordering of displayed errors depends on the student's scaffolding preferences and the heuristics-based prioritisation of errors by the instructor.

Correction Component

The correction component (Figure 3) is a central component. Developing an accurate means of correction is the main challenge. We have chosen a pattern matching-based method of correction. This method permits us to create a flexible correction component that is more efficient to implement than constraint-based solutions, which depend on extensive rule repositories. Since SQL is a relatively small-scale formal language, pattern matching techniques are sufficient to correct a range of query variations. Pre-defined ideal solutions are dynamically matched against the solution submitted by the student and inconsistencies are identified.

1. A syntactically correct query is matched against an ideal solution for possible semantic errors. A segmenter splits the query into segments based on the clauses. The segmenter also partitions the ideal solution from a solution repository.
2. Having been split into segments, the student's and the ideal query are normalised for easier matching. This step consists of removing an extraneous elements and standardising certain miscellaneous items and operators.
3. When normalised, all segments are then sent to the pattern matching component. Queries are matched segment by segment. This allows the correction model to identify the clauses and elements that are violated. Each token in the ideal solution is necessary and needs to appear in the student solution. Any extra words that appear in the student's solution may affect the resulting table, which is analysed as well. The type of missing elements is recorded.

The correction component deals with semantic errors, as syntactic errors are identified by an incorporated database system in our system.

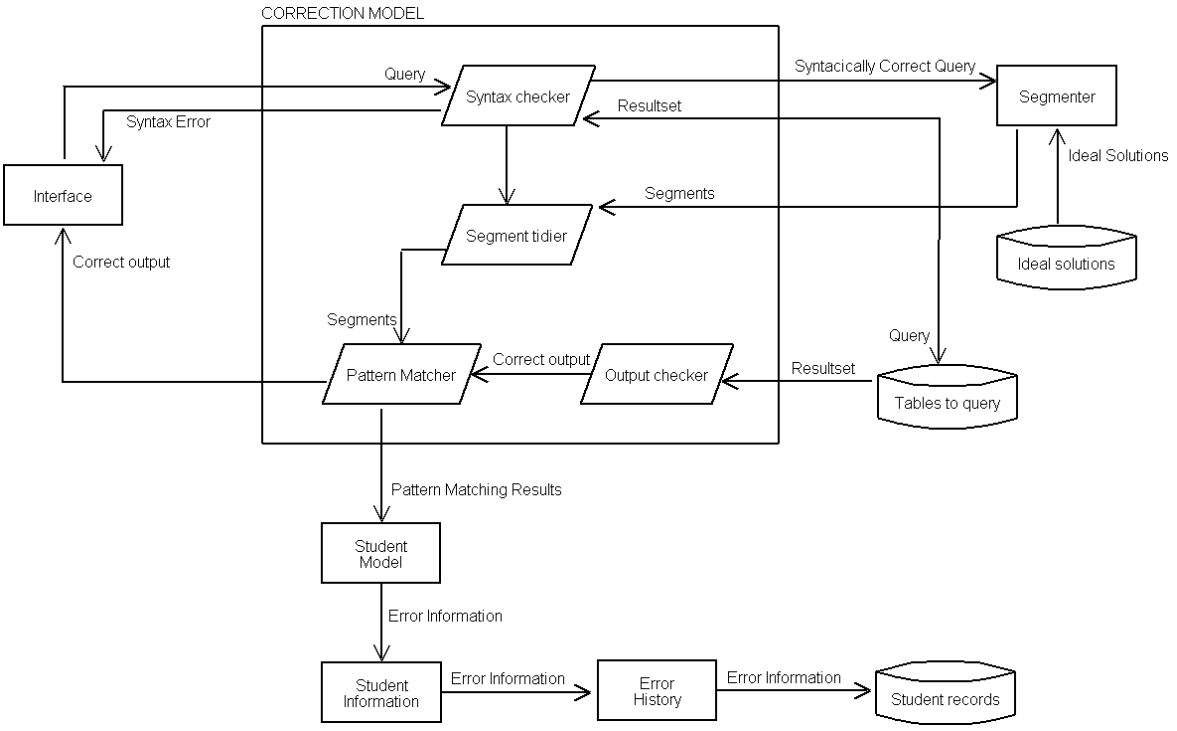

Figure 3. Correction Component Zoom

With this method of correction, a student query is marked correct if it has the correct attributes even if the ordering is different from the ideal solution.

- Representation. Based on an SQL grammar, the ideal solution can be represented in form of an abstract syntax, as depicted in Figure 4. Concrete ideal solution elements are associated with the abstract schema elements.
- Normalisation. The student solution is normalised. The representation in the form of an abstract syntax tree identifies already some syntactically different, but semantically equivalent formulations. A simple example is the same meaning of the keywords UNIQUE and DISTINCT. Further equivalences are captured by abstract grammar-based rules that describe semantically equivalent reformulations of the abstract syntax tree through, for instance, reordering or truncation of irrelevant subtrees.
- Pattern Matching. The segmented and normalised student solution (and its rule-based variations) needs to fit into the solution schema-based tree structure given by the ideal solution tree. Redundant expressions from the student solution are removed in the normalisation. Therefore, a 1:1 association between ideal schema and student solution is needed for semantically correct answers.

Different orderings of attributes, tables, or operands and operators, as in the attribute list in the example, are addressed by the pattern matcher through the equivalence rules. The component also checks the output of the two solutions when actually executed. This acts as verifier for the pattern matcher and increases the accuracy of the correction component.

In the example, SELECT distinct pname, colour FROM p WHERE city = ‘Paris’ and weight > 10, the student has made two errors. Instead of selecting colour and city, the selected attributes are colour and pname. S/he has also specified Paris parts instead of non-Paris parts (i.e. wrongly used the equality operator).

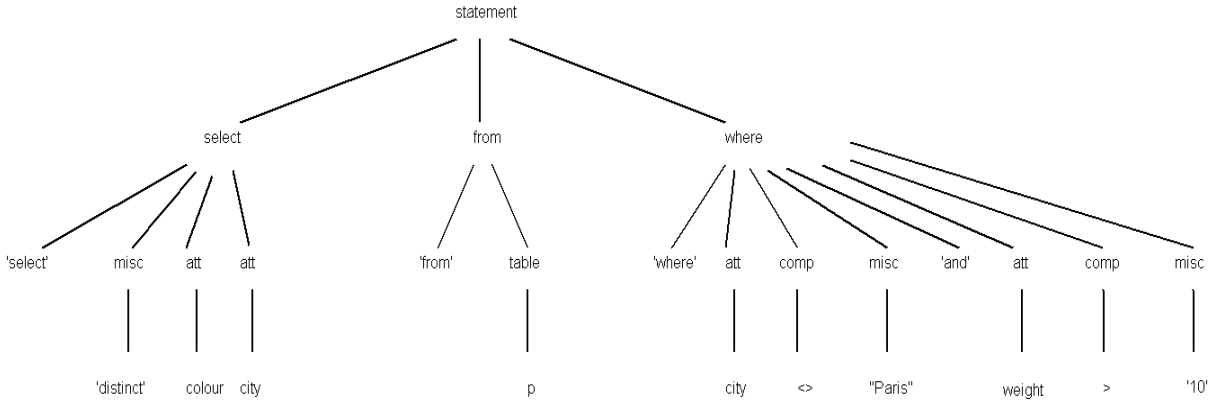

Figure 4. Ideal Solution Tree Structure

The accuracy of error classification for languages such as SQL is a central challenge. While SQL is a relatively simple programming language, many variations of a single statement correctly answer a question – which we have captured with the abstract syntax tree representation and the equivalence rules. With larger numbers of correct answers, the accuracy of the correction technique is the key difficulty as all equivalences have to be fully captured. The method of correction can lead to an error in the diagnosis when permitted variations are not adequately dealt with. The combination of a larger number of these variations significantly increases the complexity, making it difficult in practice to actually produce a fully accurate technique, whether it is based on pattern matching or on any other technique like constraint-based reasoning. A heuristic approach to accuracy improvement is often necessary.

A high degree of accuracy would even allow the correction component to be used to grade student solutions. The tutoring system could be used to return percentage marks and potentially to mark coursework. This could be implemented by linking a severity level to errors and linking an importance level to elements in the clause. Students could be given full marks initially, which are decreased in varying amounts by the correction model based on the severity of errors and the absence of vital elements.

5 Feedback and Guidance

Feedback

Feedback can be defined in the educational context as immediate, or synchronous advice given to a student, based on a solution s/he has submitted, that aims to support the student's learning experience. In our system, feedback is an umbrella term for hints about mistakes along with links to relevant background material.

Based on the correction component and the representation of error categories, our tutoring system provides five levels of increasing feedback. Hints and links to supplementary material are predefined in order to make pedagogical sense when combined. The principle is that the student examines the hint, reassesses the requirements of the question, and re-formulates the solution.

- While the hints aim to be helpful they do not reveal the complete answer. The formulation of hints requires the expertise of an instructor to reflect common errors and their cause.
- The hints are weighted, reflecting the different severities of individual errors. For semantic errors, we consider clause hints being the most important, then element hints – which is reflected in the organisation of levels.

The feedback decision procedure is directly based on the error diagnosis, i.e. is based on the three error categories and the student's preferred feedback level.

- Level 1: one SQL clause hint.
- Level 2: one SQL clause hint + one SQL element hint.
- Level 3: all SQL clause hints + all SQL element hints.
- Level 4: all SQL clause hints + all SQL element hints + required elements.
- Level 5: all SQL clause hints + all SQL element hints + required elements + links.

Solutions are additionally available as part of the scaffolding after a number of unsuccessful attempts – which is an adaptive feedback element. For our example from the previous section, for each level the following hints are added:

- Level 1: there was an error in the select clause.
- Level 2: there was an error with an attribute that was selected.
- Level 3: there was an error in the where clause; there was an error with a symbol that was used.
- Level 4: you need to include the following element: "colour … <> …" .
- Level 5: try the following links: – 1 – 2 – 3 .

Feedback is a synchronous reply, only determined by the most recent student-system interaction. In order to maximise the potential of tutoring systems, a more long-term personalised form of feedback is needed, but which is often lacking, in particular in SQL tutoring systems. This form of global feedback is called guidance.

Guidance

Guidance means to offer the student advice and recommendations about subject areas to concentrate on based on a more comprehensive assessment than a single correction. This determination may be a single session, but typically it is data collected over a number of sessions. Our system uses the student records to offer personalised guidance based on a student's overall activity in the system. The virtual apprenticeship model and the guided discovery format, both of which we are incorporated in this guidance approach, assume that the student will take a degree of responsibility for her/his learning experience. Guidance only suggests areas that the student should relearn or pay particular attention to.

The pedagogical model retrieves the information on all errors made by the student from the student model. There are four levels of guidance. As with the feedback feature, each level builds on the previous level. The decision is based on the complete error diagnosis and the preferred guidance level.

- Level 1: one SQL clause judgement.
- Level 2: one SQL clause judgement + one SQL element judgement.
- Level 3: all SQL clause judgement (graded) + all SQL element judgement (graded).
- Level 4: all SQL clause judgement (graded) + all SQL element judgement (graded) + menu of relevant questions to practise.

For instance, the following guidance advice could be given:

- Level 1: your greatest amount of errors involves the WHERE clause.

- Level 2: your greatest amount of errors involves the WHERE clause, using symbols.
- Level 3: you are having major problems with the WHERE clause; you are having minor problems with the SELECT clause.
- Level 4: try the following questions: here (follow the link).

Guidance can be given on request or at the beginning or end of a session. Internally, based on the activity log, error categories are ranked and the highest ranking ones are given first in the guidance to the student.

7 Discussion

Evaluation of the Tutoring System

Student attainment is one of the factors that determine the system's effectiveness and success. Over the last years, we have achieved an exam mark increase by 2% annually through regular improvements. While during the introduction of our tutoring system, all other factors remained constant, the relevant exam results have increased by 5.4% on the previous year.

Student opinion is another crucial success criterion. Over 90% agreed, some strongly, that the tutoring system was a useful teaching and learning tool in its own right. The majority of students agreed or strongly agreed that the system is easy to use in general. The feedback and personalised guidance features are seen as an important part of the tutoring system. Our survey results imply that the idea of providing both feedback and guidance has its merits. One critical aspect, however, is worth noting. Unintended higher levels of inaccuracy of the initial prototype have led to a more critical evaluation by students. Accuracy is the crucial property of the correction feature, even if it is not used for grading.

Students were asked to rate five statements about the system. Students indicated that the most important aspect of the system is its focus on active skill-oriented learning rather than a passive lecture-based approach. The second most important aspect is that the system is always available and enables self-paced learning, which would otherwise only have been possible during supervised lab times.

General information about student behaviour and usage of the system can be determined through web usage mining, which we have used to validate and complement the results of both the attainment evaluation and student survey. Although this evaluation often shows examples of just-in-time learning, we observed that this is complemented by long-term and pre-emptive use. The majority of usage occurred outside of the scheduled lab sessions, which demonstrates its value as a self-study tool following the apprenticeship philosophy.

Related Work

Our system is based on the virtual apprenticeship model, enabling activity-based learning and training based on an ITS-style architecture. We describe three systems that combine personalisation and ITS properties with SQL tutoring. These systems are similar in their aims and interfaces, but they each have differing architectures and methods of correction. We are particularly interested in the correction, scaffolding, and feedback/guidance methods employed by the systems.

- SQLator (Sadiq, Orlowska, Sadiq, and Lin, 2004) corrects a student's submission by equating it with the corresponding English question – the authors refer to this as

evaluating the student's submission rather than correcting it. The system's equivalence engine judges if the student's SQL solution correctly corresponds to the given English question, without actually executing the query at correction stage. This correction approach is currently limited to a selection of select statement errors.

- Acharya (Bhagat, Bhagat, Kavalan, and Sasikumar, 2002) uses a three-step process – pre-processing, atom processing and truth table processing – to correct the student's proposed solution. This process assumes that the sets of atoms in two expressions are the same. The process will fail if one of the expressions is made up of more atoms than the other, so the correction method is not as robust as it might need to be. Also, the where clause is the only clause that is analysed, which leads to accuracy problems.
- SQL-Tutor (Mitrovic and Ohlson, 1999) uses constraint-based reasoning to correct queries submitted by the student. The system's constraints deal with syntax errors as well as semantic errors, as the student's proposed solution is not actually executed. The system checks each submitted query for relevant constraints that have been violated. This method of correction can yield a high level of accuracy, depending on the extent of the constraint base. In order to provide for the large range of possible errors the developers have created a large number of constraints over a number of years and versions of the system. The required investment for this one component is not always a feasible option.

Another, related approach to correcting SQL queries exist. Based on (Brass and Goldberg, 2006), the authors have developed a correction tool that gives feedback on general queries without an ideal solution using heuristics about the general consistency of the query. While this would provide a useful complement to a training or development environment for databases, it does not provide an adequate degree of feedback in an automated tutoring setting.

The three systems offer some form of scaffolding in the form of feedback to the student.

- SQLator's automatic feedback consists of an error flag telling the student if an answer has been marked as correct or incorrect. Asynchronous feedback is offered by allowing staff members to email or post messages to address submissions. A synchronous hint-based feedback system, which we consider essential for automated tutoring, is proposed as an extension.
- Acharya provides feedback in the form of error flagging and hints. Hints are comprised of text and links. Only one hint is displayed at a time. Allowing the user to have an option here would be preferable. This system does not offer guidance based on the student's previous actions.
- SQL-Tutor provides advanced feedback, offering both hints and partial solutions. There are five levels of feedback ranging from a simple correctness indication to offering the complete solution. Feedback is in text form only and relevant links are not offered. Links to relevant topics are often useful to students and put errors into context. There is no additional guidance based on the student's previous actions.

Both Acharya and SQL-Tutor suggest the next question for the student to attempt. The reasons for this system recommendation are, however, not made explicit to the student as in our guidance component and the student is not given a choice of recommended questions.

8 Conclusions

Automated tutoring systems have become an accepted method of instruction. Students reach a higher level of understanding when they are actively engaged in these systems. Our automated tutoring system provides a realistic training environment for database

programming. The apprenticeship theory is particularly suited to skills training. An important feature of this theory is scaffolding, of which feedback is a classic example. Feedback can be global or local, which we have addressed through synchronous local feedback and global guidance based on the student's overall performance. Automated tutoring is time and location independent. It can be developed to mimic the action of a human tutor – correcting, providing feedback, presenting an assortment of questions, etc. In addition, tutoring systems are moving towards adaptive flexible tutoring rather than imposing a strict learning path on the student.

Some difficulties need to be addressed in the implementation of automated tutoring systems. A system's accuracy and the student's trust level are important for its success. Designing and implementing a flawless correction method is, however, the challenge. Trust in a system will increase as correction errors are lessened; students will be confident that their errors are accurately diagnosed and the scaffolding provided is relevant. For certain topics it may be difficult to create a perfect method of correction. This leads to either fallibility within the system or high development costs. While these limitations exist, an automated tutoring system is nonetheless a beneficial learning tool. Our main aim here was to introduce techniques and to demonstrate:

- the potential of advanced tutoring for a computer language based on a pattern matching approach to automated correction,
- the benefits of integrated feedback and personalised guidance based on pattern-based correction.

Content domains focusing on formal languages lend themselves to automated tutoring – their structure makes them relatively easy to analyse. The student can make submissions to the system and receive results based on automated correction. The presented correction approach using grammar-based pattern matching can in general be transferred to similar learning and training domains where computer-processable languages, both textual and diagrammatic, can be processed automatically based on an explicitly formulated grammar.

An automated system can never fully replace human tutoring in terms of its quality. This can only be alleviated to some extent by providing an interesting and, importantly, useful system with greatly improved accessibility and availability. Personalised guidance complements immediate correction and feedback – and should be present in any tutoring system.

References

Bhagat, S., Bhagat, L., Kavalan, J. and Sasikumar, M. (2002). Acharya: An intelligent tutoring environment for learning SQL. *Proceedings of Vidyakash 2002 – International Conference on Online Learning*.

Beck, J., Stern, M. and Haugsjaa, E. (1996). Applications of AI in education. *ACM Crossroads, The Student Journal of the association for Computing Machinery*, 3 (1).

Boyle, T. (1997). *Design for multimedia learning*. Prentice Hall Europe.

Brass, S. and Goldberg, C. (2006). Semantic errors in SQL queries: A complete list. *Journal of Systems and Software* 79:630-644.

Brusilovsky, P. (2000). Adaptive Hypermedia: From Intelligent Tutoring Systems to Web-based education. *Proceedings of 5th International Conference on Intelligent Tutoring Systems, ITS 2000*. Springer Verlag. pp. 1-7.

Chou, C.-Y., Chan, T.-W. and Lin, C.-J. (2002). Redefining the learning companion: the past, present, and future of educational agents. *Computers & Education,* 40 (3), pp. 255-26.

Collins, A. (1988). *Cognitive apprenticeship and instructional technology*. Technical Report No. 6899. BBN Labs Inc., Cambridge, MA, USA.

Collins, A., Brown, J.S. and Holum, A. (1991). Cognitive Apprenticeship: Making thinking visible. *American Educator*, Winter edition.

De Bra, P. and Stash, N. (2004). Multimedia adaptation using AHA! *Proceedings of ED-MEDIA 2004 World Conference on Educational Multimedia, Hypermedia & Telecommunications*. pp. 563-570.

Heaney, D. and Daly, C. (2004). Mass production of individual feedback. *Proceedings of ITiCSE'04*. ACM Press. pp. 117-121.

Herrington, J. and Oliver, R. (2000). An instructional design framework for authentic learning environments. *Educational Technology Research and Development,* 48 (3), pp. 23-48.

Kenny, C. (2006). *Automated Tutoring for a Database Skills Training Environment.* M.Sc. Thesis. Dublin City University, School of Computing.

Kenny, C. and Pahl, C. (2005). Automated tutoring for a database skills training environment. *Proceedings of ACM SIGCSE Symposium 2005*. ACM Press. pp. 58-62.

Mayo, M. and Mitrovic, A. (2001). Optimising ITS behaviour with Bayesian Networks and Decision Theory. *International Journal of Artificial Intelligence in Education*, 12, pp. 124-153.

Melis, E. and Ullrich, C. (2003). Local and global feedback. *Proceedings of AIED2003, 11th International Conference in Artificial Intelligence in Education*.

Mitrovic, A. and Ohlsson, S. (1999). Evaluation of a Constraint-Based tutor for a database language. *International Journal of Artificial Intelligence in Education*, 10, pp. 238-256.

Murray, T. (1999). Authoring Intelligent Tutoring Systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, 10, pp. 98-129.

Murray, S., Ryan, J. and Pahl, C. (2003). A tool-mediated Cognitive Apprenticeship approach for a computer engineering course. *Proceedings of International Conference on Advanced Learning Technologies ICALT2003*. IEEE Press. pp. 2-6.

Pahl, C., Barrett, R. and Kenny, C. (2004). Supporting active database learning and training through interactive multimedia. *Proceedings of ITiCSE'04, The 9th Annual Conference on Innovation and Technology in Computer Science Education,*

Ramakrishnan, R. and Gehrke, J. (2003). *Database management systems*. McGraw Hill.

Ravenscroft, A., Tait, K. and Hughes, I. (1998). Beyond the media: Knowledge level interaction and guided integration for CBL systems. *Computers in Education*, 30 (1/2), pp. 49-56.

Sadiq, S., Orlowska, M., Sadiq, W. and Lin, J. (2004). SQLator – an online SQL learning workbench. *Proceedings of ITiCSE'04*, June 2004. pp. 223-227. ACM Press.

Stephenson, J. (ed) (2001). *Teaching and learning online*. Kogan Page. London.

Stottler, R.H. and Vinkavich, M. (2000). Tactical Action Officer Intelligent Tutoring System (TAO ITS). *Proceedings of the 2000 Interservice/Industry Training, Simulation and Education Conference (I/ITSEC-2000).*

Winnips, K. (2001). *Scaffolding-by-design as a model for online learner support*. Ph.D. thesis, Faculty of Educational Science and Technology, University of Twente, Netherlands.