Automated Tutoring for a Database Skills Training Environment

Claire Kenny
Bachelor of Science in Computer Applications

A dissertation submitted in fulfilment of the
requirements for the award of
Master of Science (M.Sc.)

to the

Dublin City University
School of Computing

Supervisor: Dr. Claus Pahl

September 2006

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Master of Science is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: Claire Kenny

ID No.: 99495481

Date: September 2006

Acknowledgments

I wish to thank all those that supported me during my research time.

In particular, sincere thanks to my supervisor, Dr. Claus Pahl, for his unending patience and invaluable guidance.

Thanks also to my friends, both inside and outside DCU. I am especially grateful to those - you know who you are - who gave me much encouragement and feedback while I was writing up, and afterwards.

Finally, a special thanks to my family; my parents Fintan and Margaret, brother Ray and sister Maria. Words cannot say how appreciative I am.

Contents

List of Figures

List of Tables

Abstract

The emergence of educational technology and the growth of the Internet, coupled with the rise in the number of students entering third level education, has led to a surge of online courses offered by universities. These online courses may be part of a traditional classroom based course, or they may act as an entire course by themselves. Student engagement, assessment, feedback and guidance are important parts of any course, but have an added importance for one that is presented online. Together, in the absence of a human tutor, they can greatly aid the student in the learning process.

We present an automated skills training system for a database programming environment that will promote procedural knowledge acquisition and skills training. An SQL (Structured Query Language) select statement tutoring tool is an integral part of this. Targeted at students with a prior knowledge of database theory, and as part of a blended learning strategy, the system allows the student to practice SQL querying at his own time and pace. This is achieved by providing pedagogical actions that would be offered by a human tutor. Specifically, we refer to synchronous feedback and guidance based on a personalised assessment. Each of these features is automated and includes a level of personalisation and adaptation. A high-level of interaction and engagement exists between the student and the system. Students assume control of their learning experience.

Chapter 1

Introduction

1.1 Introduction

This work is based on the two expansive domains of university education and Internet technologies. Specifically, our work deals with using Internet technologies to teach Structured Query Language, a database programming language, to university students. Student-system interaction, feedback and guidance are important features of our tutoring system, as is an individually tailored learning experience. Together they will aid the system to promote procedural knowledge acquisition and skills training.

In this chapter we will provide a background to the work, beginning with an overview of educational technology and continuing with a look at database systems. Our motivation and key objectives of the work will then be outlined. The chapter will conclude with a synopsis of the chapters that follow.

1.2 Background

1.2.1 Educational Technology

The field of teaching and learning has been developed, modified, studied and evaluated throughout the ages to the present day. Various methods of teaching have been suggested, and many theories have been put forward as to how students learn. All of these methods and theories, however, have a common goal: to help people in education to work to the best of their ability and to reach their greatest potential.

In modern times vast advances in the technological world have also been made. The computer has become a feature in the daily lives of a large number of individuals and organisations across the world. There can be little doubt that technology and computation methods will continue to grow into the future.

Both education and technology are large, interesting fields which, despite immense research being completed, will always lend themselves to studies and improvements. Education can be viewed as something that is both a basic right and a vital asset.

Technology and computing are often regarded as a powerful means of making tasks easier for the individual. It is little surprise then that education and technology have been fused together to form the area of educational technology. We are specifically concerned with education through Internet technologies.

The growth in this area, coupled with the boom in Internet usage, has led to an increasing number of universities and training colleges using educational technology, specifically computing and the Internet, to provide online courses to students. Some offer online modules as part of a traditional lecture-based structure. Others allow students to access entire courses online which are complemented with occasional classroom-based tutorials.

1.2.2 Database Systems

A database is "a shared collection of logically related data, and a description of this data, designed to meet the information needs of an organisation" [Connolly and Begg, 2002]. In essence it is a central repository of information that is necessary or relevant to a person or persons. Databases are very important to organisations as they facilitate the storage and retrieval of a wide array of information.

A database management system (DBMS) is a software system that allows users to define and create access to a database, and then control and maintain this access [Connolly and Begg, 2002]. The user should therefore be able to define and manipulate data. A database can be defined and created using a Data Definition Language (DDL). This lets users specify data types and structures, and the constraints that are applied to them. Data manipulation is done using a Data Manipulation Language (DML). This means that users can insert, update, retrieve, and delete data from the database. SQL is both a DDL and a DML.

1.2.3 Structured Query Language

Structured Query Language, or SQL, is a formal declarative database programming language composed of about thirty English words such as select, from, where, delete, insert, etc. It is non-procedural, meaning the user specifies the information that is required as opposed to indicating how to get it. SQL is essentially free-format. Developed in the late 1970s, it was originally an IBM concept but was soon adapted by other vendors. In the present day there are hundreds of SQL-based products on the market. SQL is the most common of database query languages, and is

the formal and de facto standard language for relational DBMSs [Connolly and Begg, 2002].

The aims of SQL are [Connolly and Begg, 2002]:

- To create a database and its relation structures
- To carry out basic data management jobs, including the insertion, modification, and removal of data
- To perform queries, ranging from the simple to the complex. This should be done with minimal user effort, so the language's syntax and command structure should not be to difficult to learn
- To remain portable – it must adhere to a certain standard, so that the same syntax and command structure can be used if DBMS is changed

1.2.4 Introduction to Databases CA218

The School of Computing in Dublin City University (DCU) offers a four-year degree course in Computer Applications. For the most part this course is given in a traditional lecture-based format.

Second year students take an Introduction to Databases module (CA218). This module is both compulsory and examinable. It differs from most other modules in that for the last number of years students used purpose-built online courseware to learn. Human-presented tutorials and lab sessions are also scheduled as part of the module. This combination results in a blended learning module, meaning both computer aided learning and traditional classroom-based teaching are provided.

Courseware is comprised of lectures, tutorials and labs covering database topics such as ER modelling, normalisation, database creation and querying, and so on. Lectures are available in textual and audio format. Tutorials are made up of a series of animations. Interactive labs are also present, one of which is an SQL execution environment. The extension of this execution environment is the area our work is most concerned with.

1.3 Motivation

Computer-Aided Learning systems are becoming acceptable ways of teaching various topics to a range of students. Naturally, it is important that technical issues are addressed during the development of such a system. What is sometimes forgotten, however, is that pedagogical details must also be given adequate thought. With these issues in mind, we want to develop an automated SQL tutoring system to be delivered online. This system will attempt to mimic typical actions of a human tutor.

The delivery of feedback is very important; it is an integral part of the learning process [Heaney and Daly, 2004]. A human tutor offers the student various forms of feedback; an online tutor should do this also. This feedback should be relevant, precise and understandable.

The level a student reaches when learning a skill is often proportional to the student's engagement with a teacher or the skill. Interaction between the student and the online tutor is important, therefore. Ravenscroft et al. [Ravenscroft, Tait, and Hughes, 1998] describe a system where the student can interact with the topic being taught. In their system, the student interacts with knowledge (by annotating text, etc.) to create tailored knowledge representations. This concept can also be applied to the teaching of procedural knowledge. In our tutoring system, instead of interacting with text, the student will interact with the system through the submission of answers and the receipt of feedback. Thus, as with Ravenscroft et al.'s system, the student can tailor knowledge representations.

The above issues can be applied to any online tutoring system. In the computing domain, formal languages are particularly suitable for addressing these issues; we will use SQL. SQL is vital for the definition, manipulation and retrieval of information from a database system. Therefore, it is natural that it will be part of an introductory database course, such as CA218. Indeed, by studying SQL the student will acquire a conceptual understanding of database engineering by working within a data model's structures, operations and constraints [Pahl and Kenny, 2005].

We will focus on the fundamental SQL select statement. This statement can be quite simple but also has the capacity to become considerably complex. Thus, the teaching and learning of the SQL select statement suits a large group of students with a wide range of ability. CA218 students are such a group, and class numbers typically exceed one hundred students. SQL will therefore suit those that can work with simple

or complex queries. Also, because of such (possibly unexpected) chance of complexity, students can form misconceptions about the language early on, which leads to difficulties with it at a later time.

A simple SQL execution environment currently exists as part of the CA218 online courseware. This system allows the student to select a question from menu and then submit his answer to the tutor. His result and the correct result are then displayed in table format. The student must himself determine if his answer is correct or incorrect. In cases, the student's solution may accidentally produce the correct output even though the actual query is incorrect. It would be helpful if the system could explicitly mark the student's query as being correct or incorrect. In addition, the current system offers feedback only if the student makes an error with the query syntax. There is no automated assistance available if the student makes a non-syntax error. We wish, therefore, to produce an online system that will help students to learn SQL. We will refer to this system as a tutor as it will offer features that a human tutor would offer. In particular, it will provide feedback for the student that will be of a contextually high quality. 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. SQL is a suitable topic to explore these issues, but they apply equally to other formal languages. In summary, in such a learning system, execution and manipulation of feedback are paramount [Kenny, McMullen, Melia and Pahl, 2005].

The design and implementation of our new system should benefit a range of stakeholders:

- The learner will benefit from a system that corrects a submission and offers feedback and guidance based on the results of this. He will be able to use the system at times when a human tutor would not normally be available for consultation.
- The instructor will benefit from the information he can access from student records and web access logs. Through these he will be able to tell, for example, the questions students are having most and least difficulty with.
- The instructional designer will benefit because the design of the system will allow the system's questions and solutions to be updated with relative ease.

- The software developer will benefit from an architecture that is created to be modular and maintainable.

1.4 Objectives

Our project is guided by six objectives. Our primary objective is to extend the SQL execution environment currently in use as part of the CA218 module. Specifically, this means designing and implementing an automated tutoring system to teach the SQL select statement. This system should include domain specific feedback and personalised guidance features. Before beginning the first objective we need to fulfil three others – our second, third and fourth objectives. The second objective is to develop an architecture for an automated tutoring environment. This architecture should be based on research into various online learning systems, thus allowing us to use the positive aspects of standard architectures currently functioning. Following this, as our third objective, we must 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 will be categorised according to a multi-level error classification scheme that we will define. This will allow us to alert the student to an error in a very specific manner, thus minimising confusion and leading to a quicker and less frustrating learning experience. Our fourth objective aims to determine adaptivity techniques for use in a knowledge-based feedback and guidance system. These will be used to develop a system that lets content and navigation to be adapted by feedback and guidance features. Following the design, implementation, and testing of the system it will be made available for use as part of the CA218 module. Our fifth objective is to evaluate the system, which will begin after the mid-semester SQL exam. This stage of the project involves evaluating the results, behaviour, and opinion of this year's CA218 students, along with assessing the system based on the fulfilment of our objectives and general software engineering goals. Our final objective consists of carrying out the all of the tasks outlined above in accordance with educational principles and pedagogical considerations. This is to ensure that the student has access to a complete and pedagogically sound learning experience.

1.5 Outline of Work

This chapter has outlined the background to and the core objectives of the project.

In chapter two we will discuss the wider context in which the project is situated by reviewing the relevant literature encountered throughout this research time. This chapter will include a review of online tutoring methods and traditional classroom-based pedagogical theories.

The following chapter will describe related work and similar systems that are currently available. We will comment on how these systems relate to our proposals.

In chapter four we will elaborate on our set of objectives. We will also list a number of requirements relating to the project.

Chapter five contains information regarding the actual design and implementation of our project. Here will we describe the architecture of the system before moving onto a detailed analysis of each of the system's components. We will also define our multi-level error classification scheme and detail the manner in which the student interacts with the automated tutor.

The system will be evaluated in chapter six. This will be done by providing the motivation of our evaluation, outlining our evaluation methods, and then presenting and discussing our results.

Finally, this work will be concluded with a discussion of the entire project, along with recommendations for future work.

Chapter 2
Literature Review

2.1 Introduction

Many studies on online tutoring have been made. A wide range of online learning technologies and teaching methods exist, some of which have unique merits and hence stand on their own, and some that cross-over. We will look at some of these online learning technologies and teaching methods to determine if they can be of benefit to our project.

Online learning systems are often developed without much thought being given to pedagogical issues that apply either to an online or traditional environment. The primary aim of any learning system is for people to learn. Learning technologies as well as pedagogical theories need to be addressed.

In this chapter we will first look at Computer Aided Learning theories and techniques. This will be followed by a description of Intelligent Tutoring Systems and Human-Computer Interaction issues. Finally we will study traditional classroom-based pedagogical theories that could be applied to an online domain.

This chapter deals primarily with theory. Implementations of Computer Aided Learning systems and Intelligent Tutoring Systems will be outlined in chapter three.

2.2 Learning Technology Systems

The IEEE has drafted a learning technology systems architecture (LTSA) [IEEE P1484.1/D9] that specifies a high level architecture for information technology-supported learning, education, and training systems, such as computer assisted instruction and intelligent tutoring. It describes the high-level system design and the components of these systems. Being a standard, it is pedagogically, content, and platform neutral. It claims to be neither prescriptive nor exclusive.

Its chief aims are to:

- Provide a framework for understanding existing and future systems

- Promote interoperability and portability by identifying critical system interfaces.

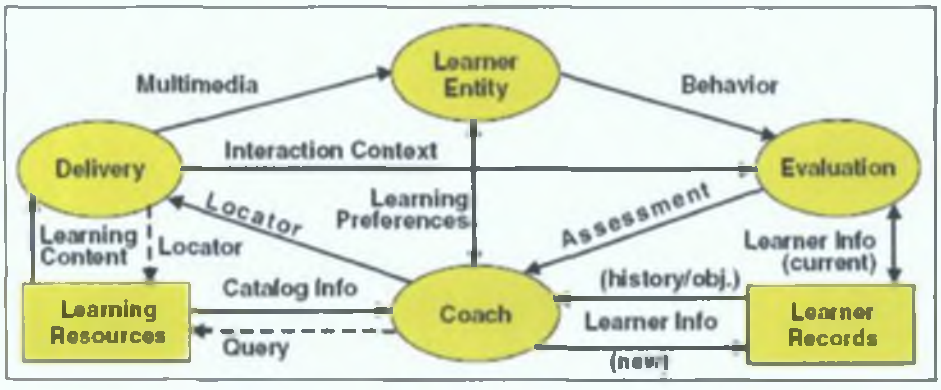

Figure 2.1. LTSA [IEEE P1484.1/D9]

The architecture of this standard (Figure 2.1) is comprised of three various components, which are defined as processes, stores and flows. These components are conceptual – there is no need for every component to be identifiable in actual implementations.

The components are [IEEE P1484.1/D9]:

- Processes – learner entity, evaluation, coach, delivery
- Stores – learner records, learning resources
- Flows – learning preferences, behaviour, assessment information, learner information, query, catalogue information, locator, learning content, multimedia, interaction context

A combination of these components can be used to design any Computer Aided Learning (CAL) environment. In particular, we will be drawing inspiration from the LTSA definition of a coach: "A conceptual process that may incorporate information from several sources, such as the learner (learning preferences), evaluation process (assessment information), learner records (performance, preference, and other learner information), and learning resource (query and catalogue information), and may use this information to search (query) and select (locator) learning content (via the delivery process and multimedia) for learning experiences".

As stated previously, the LTSA is a good starting point for the design of a CAL system. The architecture it describes is considerably generic however – we will not be

able to create our own architecture directly from it, but we can take inspiration from it.

Both CAL and Intelligent Tutoring Systems (ITS) are examples of learning technologies. We will outline both in sections 2.3 and 2.4 respectively.

2.3 Computer Aided Learning (CAL)

Recent emerging learning theories, such as situated learning and cognitive apprenticeship (discussed in section 2.6.2) have caused the design of instructional systems to change from teaching to learning, and for learning environments to become more facilitative. Such theories have also greatly influenced the design of technology-based learning environments [Feng-Kwei and Bonk, 2001].

Computer Aided Learning refers to learning where the Internet is used as the main tool. It excludes aspects that can be part of "distance learning" but are not electronic (books, for example) [Hadjerrouit, 2004]. This area is undergoing rapid developments in terms of pedagogy and technology [Pahl, 2003a], hence gaining importance in both education and industry.

Positive characteristics of computer-aided learning environments include [Abdelraheem, 2003]:

- Enabling active engagement in construction of knowledge
- A powerful navigation system
- Self-paced student learning to suit the individual needs of each student
- Student autonomy is encouraged as he is effectively in charge of his own learning.

Generic architecture of CAL systems

Pahl [Pahl, 2003a] characterises the evolution of teaching and learning environments along four dimensions, each of which provides a different perspective of such environments. They are:

- Content – the subject-oriented perspective
- Format – the organisational perspective
- Infrastructure – the technical perspective

- Pedagogy – the educational perspective

While originally used to describe the changes that can occur in online courses, the same categories can be used to define the areas that need to be considered when designing an online learning system. Attention needs to be given to each perspective during the design and implementation of a CAL system. The last perspective (pedagogy) is of notable interest. Currently, technologies, not pedagogical considerations, are the driving force behind web-based learning [Hadjerrouit, 2004]; there is little understanding of the relevant learning issues. This problem needs to be addressed when designing and developing a CAL system.

Figure 2.2 shows the components of a typical Web-based learning system. The final system should lend itself to management, modification (including updates) and reusability.

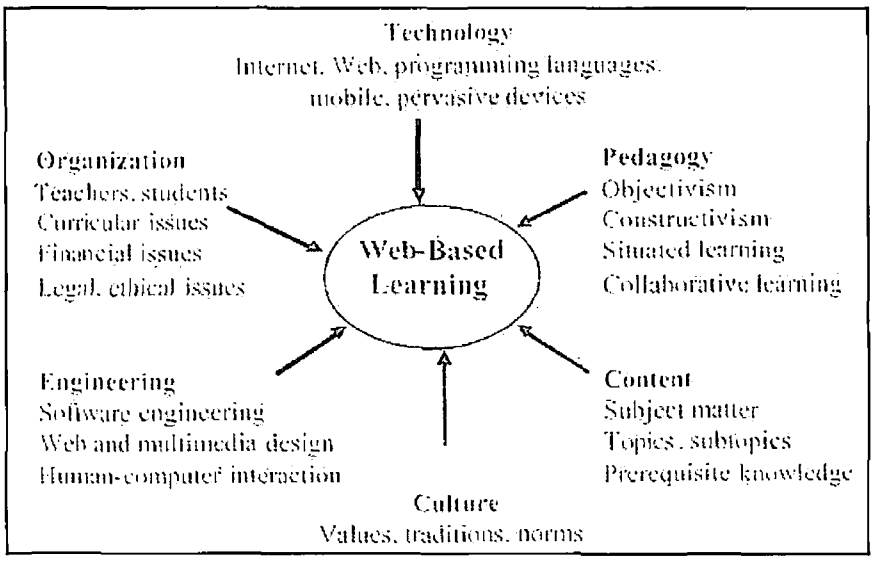

Figure 2.2. Components of a Web-based Learning System [Hadjerrouit, 2004]

2.3.1 Guided Discovery

Guided discovery refers to a form of navigation within a learning system. With this the emphasis is placed on the student and his ability to seek information, rather than the system overwhelming him with data. The student's intelligence is thus both exploited and supported. It is a move from teacher controlled learning to student controlled learning. The control we refer to includes, but is not limited to, pace and timing, choice of content, management of learning activities, and responses to exercises [Stephenson, 2001]. In particular, the student can customise the pace of

learning to his own ability. Students should have the option of autonomous learning [Pahl, 2003a]; the student should have the opportunity to take control of his own learning organisation and to self-navigate to any part of the course.

A mixed locus of control in a CAL environment has been advocated [Masthoff, 2002]. With this, both the student and the system can be the chief navigator throughout the courseware. Only one of these can be in control at any single time. This controlling decision is made based on the general interaction and the student's performance.

2.3.2 Personalisation and Adaptive Hypermedia

2.3.2.1 Personalisation

The trend in Computer Aided Learning environments has turned towards personalisation in recent times. By this we mean a teaching and learning strategy that is tailored to the personal needs of the learner. According to Dagger et al. [Dagger, Wade, and Conlan, 2005], the goal of personalisation in e-learning is "to support e-learning content, activities and collaboration, adapted to the specific needs and influenced by specific preferences of the learner and built on sound pedagogic strategies". Ideally, the e-learning experience should be based on activity and, because pedagogy is fundamental for the success of an online course [Dagger, Wade, and Conlan, 2004], it needs to be created around sound teaching and learning principles. A personalised method of instruction is characterised by [Liegle and Woo, 2000]:

- Pace controlled by the learner
- The ability to retake assessments until mastery is demonstrated
- Immediate feedback
- Small units of instructional material
- The use of peer proctors to deliver feedback and testing
- Optional lectures.

Personalisation prevents content and navigation from being common to all users, and allows the presentation of non-static learning material.

2.3.2.2 Adaptive Hypermedia

Adaptive hypermedia is a method of personalisation. Brusilovsky [Brusilovsky, 2000] defines adaptive hypermedia systems as those 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 the hypertext to the needs of the user". Hypertext here refers to, for the most part, page content and hyperlinks. This process is repeated as the user's goals and preferences change. So, for instance, an adaptive hypermedia system may customise a selection of examples and hints, and recommend appropriate hyperlinks.

Adaptive teaching and learning can be provided at two main levels [Boyle, 1997]. The first is the selection of presentation of content. The second is navigational guidance at the link level.

Adaptive navigation support (the second of the two named just previously) aims to help users find relevant paths in hyperspace based on the user's individual characteristics. This can be done in the following ways [De Bra and Stash, 2004]:

- *Direct guidance*: The system visually indicates which of the links on the page is the best link to follow. The system may also dynamically generate an additional link (usually called "next") which links to the "next best" page.
- *Link sorting*: The system utilises the user model to sort all links on a particular page. The most relevant links appear at the top of the page whereas less relevant links appear towards the bottom of the page.
- *Link annotation*: Links are augmented with comments to help the user to make an informed decision himself. These annotations are usually provided in the form of visual cues, such as icons, font colours, sizes and types.
- *Link hiding, disabling, and removal*: A user is prevented from following links that are not relevant for him at a particular point in time. For non-contextual links, this may be achieved through link hiding (for example, by changing a hotword into a normal word), or by disabling the link so that there is no outcome from clicking on a hotword. When a link is non-contextual it can simple be removed.

Brusilovsky [Brusilovsky, 2000] suggests that adaptive presentation and adaptive navigation support both form an adaptive hypermedia system.

2.3.3 Recommender Systems and Web Usage Mining

2.3.3.1 Recommender Systems

According to Patel [Patel, Russell, and Kinshuk, 1999] an e-learning task recommender is "a recommendation system that would recommend a learning task to a learner based on the tasks already done by the learner and their successes, and based on tasks made by other 'similar' learners." In short, it is a system that produces individualised recommendations, or guides the user in a personalised manner to objects that might interest him.

In a web-based learning environment, the course administrator could track the activities occurring in the course website, find patterns and behaviours in these, and use them to change, improve or adapt the contents of the course [Zaïane, 2001]. This idea is normally used in the area of e-business, but in the last number of years the theory is being applied to the area of education online. In this e-learning context, a recommender system may also utilise information about other users. Thus, such a system takes the added role of "a software agent that tries to 'intelligently' recommend actions to a learner based on the actions of previous learners" [Zaïane, 2002]. These previous learners would normally be deemed to be 'successful' learners. Web usage mining (discussed in the following section) could be used to develop this agent.

Knowledge-based recommendation systems [Manouselis and Sampson, 2004], while slightly different, may also be of benefit to the learner. Such systems form a representation of how a particular item matches a certain user need, and can connect this need with an appropriate recommendation.

2.3.3.2 Web Usage Mining

Information about all pages accessed on our website is stored by the web server in a web access log. Each entry is ordered chronologically. The log gives information about:

- The IP address or domain name of the request
- The username (in some cases)
- The method of the request (Get or Post)
- The name of the requested file
- The result of the request (success, error, and so on)

- The amount of data sent back
- The URL of the referring page

These logs typically need to be reformatted for them to be easily read by a human. These cleansed logs can then be used for the purposes of web usage mining.

Web usage mining is a form a data mining that is suited to the web context. It is defined as "the process of applying data mining techniques to the discovery of usage patterns from web logs data, to identify users' behaviour" [Batista and Silva, 2002].

Data mining (or knowledge discovery of data, as it is sometimes called) refers to the process of extracting information or knowledge from a data set for the purposes of decision making. The whole process of knowledge discovery generally consists of the following steps [Chang, Healey, McHugh and Wang, 2001]: data cleansing, data integration, data transformation, data reduction, data mining, pattern evaluation, and knowledge presentation.

The information gleaned from this can then be used to improve the effectiveness of a website, by adapting the relevant pages to suit the users' behaviour.

Integrated web mining [Zaïane, 2002] uses any patterns found on-the-fly to improve a system, by inputting them into an intelligent software system.

2.4 Intelligent Tutoring

2.4.1 Intelligent Tutoring Systems (ITSs)

An Intelligent Tutoring System (ITS) can be defined as 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]. Further benefits, according to McArthur et al. [McArthur, Lewis, and Bishay, 1993], include offering micro-tutoring (meaning they can produce very detailed feedback for the user), their ability to merge into the classroom environment, their adoption a popular method of teaching and learning, and the tutor controlling the learning.

This last point, regarding tutor control, can be seen as both an advantage and disadvantage. Dagger et al. [Dagger et al., 2004] explain that early ITSs "imposed a

strict learning path" on the user, thus limiting his control of his own learning experience, which is not in keeping with constructivist theory. Further disadvantages are that they have been developed for a relatively small amount of topic areas, so learning outcomes can be limited, and that they have limited pedagogical expertise [McArthur et al., 1993].

2.4.1.1 Architecture

A traditional ITS has four distinct components – an expert model, a student model, an instructional, tutorial or pedagogical model, and an instructional environment or user interface (Figure 2.3).

Expert model

The expert model, sometimes known as the domain model, contains knowledge of the domain or subject area. This forms a basis for what needs to be communicated to the student as well as the standard the student will be compared to. Knowledge is normally represented as a set of facts or rules [Chou, 2002].

Student model

The student model holds information about the student (personal details, learning style, learning preferences, etc.), along with a representation of the knowledge he holds (which will have been or will be matched against that in the expert model). Ideally, this will be updated every time the student answers a question or solves a problem that has been presented by the ITS.

Student modelling is sometimes done using an overlay model [Chou, Chan, and Lin, 2002]. With this, the student's knowledge is looked at as being a subset of the expert / domain knowledge.

Pedagogical model

The pedagogical model makes pedagogical decisions such as what needs to be presented or stated at a particular time, when an intervention is needed, and so on. In short, the pedagogical model has knowledge of teaching strategies to determine when and in what manner to instruct the student. Hence, it needs to make decisions about the topic, the problem, and feedback.

Interface

The interface acts as the means of communication between the student and the ITS, receiving input and presenting information.

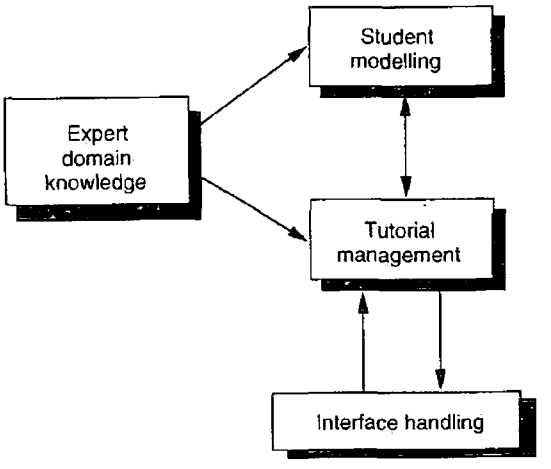

Figure 2.3. Typical ITS Architecture [Boyle, 1997]

2.4.1.2 Pattern Matching

Pattern matching or pattern recognition is a method that can be used in ITSs or intelligent teaching and learning environments. The Free Online Dictionary of Computing [FOLDOC] defines pattern recognition as "a branch of artificial intelligence concerned with classification or description of observations". Such a system can compute numeric or symbolic information from these observations and then classify or describe them. Wikipedia [Wikipedia] defines pattern matching as "the act of taking in raw data and taking an action based on the category of the data".

Pattern matching can be used in ITSs and general CAL systems for a variety of purposes. For instance, it can be used as a means of correcting work the student has done. CAPIT's [Mayo and Mitrovic, 2001] student modeller is a pattern matcher that takes the student's solution to a problem and determines which constraints are violated. Pattern matching can also be used to ascertain a higher level of student understanding. For example, the Tactical Action Officer (TAO) ITS [Stottler and LCDR Vinkavich, 2000], developed for the U.S. Navy by SHAI, teaches tactical rules of engagement to officers who direct the sensors and weapons aboard cruisers. This system applies pattern-matching rules to detect sequences of actions that indicate whether the student does or does not understand.

2.4.1.3 Adaptivity

An ITS aims to provide the user with individual and appropriate help while he is learning, such as suggesting the next step to take in the learning cycle or presenting material in such a manner as to match his preferences. It should model the knowledge the student currently has and support the move to a new (and hopefully higher) knowledge state [Urban-Lurain, 1996] by generating content and teaching methods that adapt to the individual needs of users [Boyle, 1997]. Ideally, the computer should act as the human tutor.

2.4.2 Variations of the ITS

The benefits of an ITS can be altered slightly and used in what is referred to as an Intelligent Learning Environment (ILE). Briefly, [McArthur et al., 1993]:

- Both focus on construction over instruction
- Students can control the pace of learning, and the tutor acts as a guide only
- Personalised feedback and information are important.

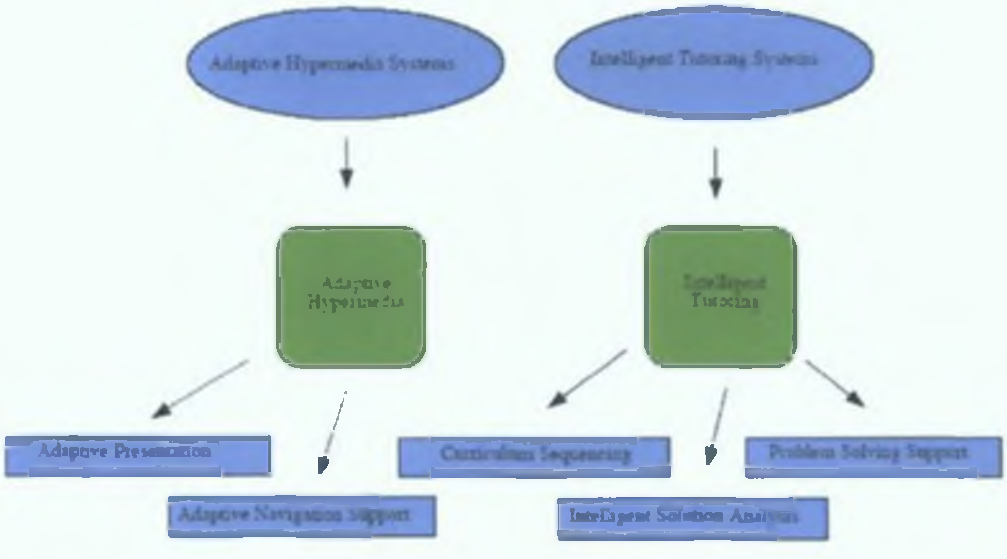

Figure 2.4. AIWBES [Brusilovsky and Peylo, 2003]

An adaptive and intelligent Web-based educational system (AIWBES) [Brusilovsky and Peylo, 2003] is a learning system formed by mixing features from adaptive systems and intelligent tutoring systems (Figure 2.4). Adaptivity is obtained by

"building a model of the goals, preferences and knowledge of each individual student and using this model throughout the interaction with the student in order to adapt to the needs of that student". A level of intelligence is included by acting as a human tutor would in some cases, for example diagnosing the student's misconceptions.

2.5 Human-Computer Interaction (HCI)

HCI involves the design, implementation and evaluation of interactive systems in the context of the user's task and work [Dix, Finlay, Abowd, and Beale 1998]. Systems used by humans, including the various systems described previously in this chapter, should strive to meet a set of usability goals [Preece, Rogers, and Sharp, 2002]. These are:

- Effectiveness – how good the system is at doing what it was designed to do
- Efficiency – efficient when supporting the user as they perform various tasks
- Safety – protects the user from unsafe or undesirable situations, such as causing an action accidentally
- Utility – provide the right kind of functionality, allowing users to do what they want or need to do
- Learnability – how quickly the use of the system can be learned. Preece et al. [Preece et al., 2002] refer to the "ten-minute-rule" – the system fails if the novice user cannot learn how to use it in ten minutes or less
- Memorability – once learned, is the system easy to remember how to use?

User experience goals can also be looked at. In short, these goals determine if the system is satisfying, enjoyable, helpful, motivating, aesthetically pleasing, supportive of creativity, rewarding and emotionally fulfilling.

Users' memories can often become overloaded with detail about how to carry out certain procedures. This should be avoided. Interfaces should be designed to be intuitive and consistent. Exploration should be encouraged to some extent, but sometimes interfaces need to contain the user slightly so that they are guided towards selecting the appropriate actions [Preece et al., 2002].

2.6 Pedagogy and Types of Learning

Learning, the fundamental aim of our system, is a broad term used to describe a considerably complex domain. Indeed, the complexity of the subject area has resulted, as Biggs [Biggs, 1999] commented, in its being an area of interest for psychologists for the entire twentieth century. Put briefly, it can be defined as a change in one's conceptual understanding [Herrington, Oliver, and Stoney, 2000]. This however, is a quite an abstract description of what happens in the learning process. Let us look at the area in more detail.

Theory / design

Traditionally, learning was largely divided into two types – didactic teaching and learning (for example, based in the classroom) and practical teaching and learning (such as when a trade is being studied). These we can refer to knowledge and skills, respectively. Let us be clear that while many similarities exist between the two there is also a distinct difference between the elements involved, and thus different educational approaches are required [Pahl, Barrett, and Kenny, 2004]. Knowledge acquisition is much more of a passive act than skills training. A student would read, assimilate, and probably memorise. There does not have to be interaction between the student and the teacher. Skills training involves a level of interaction however. The student learns mainly by practising a task. There is usually some amount of interaction between the student and the teacher.

John Stephenson suggests that "experience is the foundation of, and the stimulus for, learning", and that learning itself is never a passive act. "Learning is primarily developed through activity" [Stephenson, 2001]. The ultimate aim of our system is for students to learn the SQL database programming language. This is a practical task, therefore we focus on skills training as opposed to knowledge acquisition.

Throughout the ages, practical tasks have been learnt through apprenticeships. We will look at apprenticeship theory in more detail later in this chapter.

2.6.1 Constructivism

Constructivism has emerged as one of the most important learning theories to date. Specifically, it means that that a learner learns through the active construction of knowledge [Herrington and Oliver, 2000]. Rather than acquiring knowledge through

transmission from some medium (such as instruction or teaching), knowledge is constructed by the individual learner. This knowledge will normally be unique to each person. Boyle [Boyle, 1997] suggests that the learner understands the world by interacting with it to construct, test, and refine various cognitive representations of it. The key emphasis is not on the synthesis of facts and detail, but on various processes and skills needed to become an expert in the domain. Kunz [Kunz, 2004] elaborates on this by stating that the basic elements of constructivist teaching and learning environments are active construction of knowledge, teamed with social collaboration and negotiation, along with meta-cognition and awareness of the thinking and learning process itself, all in a contextually situated and authentic environment. As such, interaction is deemed to be a major part of constructivism. It allows learners to obtain experience through active participation, as described by Pahl et al. [Pahl et al., 2004]. The process of constructing, interpreting and modifying representations of the world is continuous [Hedberg, 2002].

Very often, construction of knowledge is deemed to be deeper than instruction or transmission of knowledge [Pahl et al., 2004]. Deep and surface approaches to learning, explained by Biggs [Biggs, 1999], are outlined in more detail later in this chapter. Surface learning means students remember a set of disjoint but have little comprehension of the inner meaning or point of the topic. Deep learning on the other hand is when the student has a true understanding of the topic; they see the "big picture".

Theory / design

Boyle [Boyle, 1997] outlines seven principles for constructivist design:

- Provide experience of the knowledge construction process
- Provide experience in and appreciation of multiple perspectives
- Embed learning in realistic and relevant contexts
- Encourage ownership and voice in the learning process
- Embed in the social experience
- Encourage the use of multiple models of representation
- Encourage self-awareness of the knowledge construction process

From this, we can summarise that constructivism is a multi-faceted learning theory of some complexity that is based on experience and realism. It has been

acknowledged as being the most suitable theory for computer-based learning environments [Abdelraheem, 2003].

2.6.2 Apprenticeships

Let us now look at the concept of apprenticeships in more detail. As outlined previously, an apprenticeship is mostly concerned with procedural knowledge acquisition and skills training. We will look at three types of apprenticeship – traditional apprenticeship, cognitive apprenticeship and virtual apprenticeship.

2.6.2.1 Traditional Apprenticeship

Traditional apprenticeship is a form of teaching and learning that has been used successfully throughout the ages, primarily for practical tasks.

Theory / design

Apprenticeship is, for the most part, a three-step process involving a master and an apprentice, the former being the teacher and the latter the student. Initially the master demonstrates the completion of the various stages of the task while the apprentice observes. For example, this step is achieved in the CA218 module by a series of animated tutorials. In time the apprentice works at the task while the master observes and offers advice when he feels the apprentice needs it, or when the apprentice requests aid. By doing this the apprentice can practise the various parts of the task in a controlled and safe environment. Finally the apprentice will achieve competency in the domain and can complete the task unaided. He is ready to assume responsibility and work in a self-reliant and independent manner. The apprenticeship environment allows learners to receive individual attention; this can enhance their learning experience.

Traditional apprenticeship is a blend of modelling, scaffolding, fading and coaching.

Modelling

Modelling is the demonstration and observation process outlined above. The apprentice observes the master as he demonstrates the various portions of the task, often explicitly showing the apprentice what to do.

Scaffolding

Often, when a structure is being built or repaired, metal scaffolding is placed around it as a means of supporting the workers and the building. This is removed when the structure is strong enough to stand by itself.

Scaffolding in the educational context is similar in ways to the scaffolding described in the previous paragraph. The term was originally coined when used in the area of language learning where it referred to effective interactions between a teacher and a student [Winnips, 2001]. Since then it has gained popularity in many aspects of education, and is now not limited to language learning. Consequently, there are many definitions for what exactly scaffolding is. One definition describes it as "providing learners with just enough assistance to help them construct their own answer to a problem" [Lajoie and Derry, 1993]. Another states it is the supports provided by the teacher to help the student carry out a task [Collins, Brown, and Holum, 1991]. In short, it is a temporary support while completing a task or activity. The key idea behind scaffolding is to provide the learner with timely support at an appropriate level. So, it should be granted when the learner needs it, and it should suit the learner's level of understanding and progress in the domain.

Collins et al [Collins, 1991] refer to scaffolding as being a set of limited hints and feedback. It has been proposed [Greenfield, 1984] that scaffolding has five major characteristics:

- It provides support
- It functions as a tool
- It extends the range of the worker
- It permits the accomplishment of tasks not otherwise possible
- It is used selectively when needed

Scaffolding may also be strategies, guides or tools.

Various levels of scaffolding exist. At the highest level it might be the teacher carrying out most of the task for the learner. At a low to moderate level it can be the teacher giving occasional hints or prompting the learner on what to do next.

As well as this, scaffolds can be deemed to be either soft or hard [Brush and Saye, 2002]. Soft scaffolds are dynamic and situation specific, while hard scaffolds are static and can be anticipated and planned in advance. Soft scaffolding is usually done "on-the-fly" when support is needed. Hard scaffolding, on the other hand, is generally

based on typical student difficulties and can be embedded within multimedia or hypermedia systems.

Ideally, scaffolding will be faded, meaning it will be removed gradually, thus encouraging the learner to work in a self-regulated and self-reliant manner. The level of scaffolding should therefore move from high to low over time. Usually, the level of support given by the teacher is decreased as the learner's ability in the domain increases. This will, ideally, improve the learner's performance [Kelly and Tangney, 2002].

In conclusion, the purpose of scaffolding is to enable learners to complete tasks that they would be unable to perform without this particular support. Scaffolding should be subject to fading.

Feedback

The Oxford English dictionary [Soanes and Hawker, 2005] defines feedback as "information given in response to a product, performance etc. used as a basis for improvement". It can also be described as information regarding the level of success something has or is being done at [Sadler, 1989]. It may also be the conveyance of information about a completed action or something that has been achieved, thus allowing the student to continue with the task in hand [Preece et al., 2002]. In the educational context in question here, feedback is strategically useful advice given to a student based on tasks he has previously attempted or completed. Its objectives are self-reliant learning and competency in a domain. It may be synchronous or asynchronous, that is it may be returned immediately or returned after a period of time.

Sadler [Sadler, 1989] identifies three essential conditions for effective feedback:

- A knowledge of standards
- The necessity to compare these standards to one's own work and
- Taking action to close the gap.

Feedback can be further 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. Local and global feedback differs in terms of content, realm, aim and point in time.

Local feedback alerts the learner to the correctness or incorrectness of his proposed solution to a question, and may provide hints and suggestions specific to that solution. Global feedback, however, can focus on the student's learning process as a whole, and thus provide guidance on topics and further questions that should be viewed to plan for future learning (content and realm). Therefore, local feedback objectives are concerned with one specific exercise whereas global feedback considers a range of exercises (aims). Local feedback is relatively immediate; it can be provided after each of the student's attempts to answer a question. Global feedback is more suited to a delayed delivery, such as when the learner has attempted a range of questions (point in time).

Ideally, the delivery of feedback should be delayed slightly i.e. the student should type his answer and submit it for correction rather than receiving immediate feedback, where the answer is corrected as the learner types. Immediate feedback may have negative motivational consequences and prevent students from learning error-detection skills [Corbett and Anderson, 2001]. This is reinforced by research [Bangert-Drowns, Kulik, Kulik, and Morgan, 1991] indicating that the introduction of a small delay between a learner's response and the system's feedback increases the positive impact of error-correcting feedback. This research also found that feedback is more effective when students make a genuine attempt at answering the question before being able to look ahead at the correct answer. Also, feedback is more effective when it includes details of the correct answer, rather than simply indicating if the answer was correct or incorrect.

In conclusion, feedback is the response to a particular action, with a primary function of correcting future iterations of the action or related actions. It is offered, for example, when the learner responds incorrectly to a question. It helps to encourage the student to attempt tasks and to feel satisfaction on solving these tasks [Daly, 1999]. The learner should use this feedback to re-evaluate the question and make adjustments to his answer accordingly.

Coaching

Briefly, coaching is the process of overseeing the student's learning [Collins et al., 1991]. It involves formulating the course of work the apprentice should take, such as choosing and ordering tasks the student should attempt, providing timely scaffolding at the appropriate level and fading it accordingly, and evaluating the apprentice to

offer meaningful feedback and encouragement. Furthermore, when sequencing the tasks, the master should incorporate three principles in sequencing activities [Collins et al., 1991]:

- Global before local skills
- Increasing complexity
- Increasing diversity.

Effective coaching makes opportunities for the apprentice to practise the task [Kelly and Tangney, 2002] by prompting, questioning and providing motivation.

2.6.2.2 Cognitive Apprenticeship

Cognitive apprenticeship, situated in the social constructivist paradigm, is a method of learning that has its foundations in traditional apprenticeship. It was first proposed by Collins et al. [Collins, Brown, and Newman, 1989], who state that it differs from the traditional format in that "the tasks and problems are chosen to illustrate the power of certain techniques or methods, to give students practise in applying these methods in diverse settings, and to increase the complexity of tasks slowly, so the component skills and models can be integrated". Cognitive apprenticeship moves traditional apprenticeship into the classroom and the cognitive domain, and is suitable for tasks that are not so 'hands-on' or practical, as it aims to "make thinking visible" [Collins et al., 1991]. A major principle of cognitive apprenticeship is collaboration and conversation with an expert [Winnips, 2003]. Through this, the learner develops the ability to solve problems as the expert would solve them. Further to this, cognitive apprenticeship uses the idea of situated learning, whereby the learner is placed in a 'real world' environment. Therefore, students learn active participation in a task in an authentic or realistic setting in close collaboration with a master [Murray, Ryan, and Pahl, 2003].

Like traditional apprenticeship, cognitive apprenticeship uses the modelling, coaching and scaffolding methods. In addition, it incorporates:

- Articulation – encouraging the student to verbalise knowledge and thinking
- Reflection – encouraging students to self-review and critique their performance
- Exploration – students create and solve their own questions,

and aims to externalise processes that are usually carried out internally [Kunz, 2004].

2.6.2.3 Virtual Apprenticeship

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, a realistic setting, and the "possibility to produce and manipulate the 'real thing'" are vital for success in this model [Murray et al., 2003].

2.6.3 Authentic / Situated learning

In an authentic learning environment, students are required to actively engage in real-world problem solving that reflects both the context and complexity of the practical situations in which the need for learning was created [Herrington and Oliver, 2000]. For example, it is widely viewed that learning a foreign language is easier by immersion is the language and culture rather than learning from textbooks and vocabulary lists. An authentic setting is regarded in many camps as being a central success factor in knowledge and skills training [Pahl et al., 2004].

An authentic activity should reflect the natural complexity of real-world environments [Murphy, 2003]. By its very nature, this setting lets the learner directly manipulate course-relevant artefacts [Murray et al., 2003], and thus it reflects the way the knowledge and skills being taught will eventually be used.

Theory / design

The following is a list of characteristics of authentic activities, based on a range of activities described by a variety of researchers [Herrington, Oliver, and Reeves, 2003].

- "Authentic activities have real-world relevance".

  They are as similar as possible to the actual tasks a professional performs in the real world.
- "Authentic activities are ill-defined, requiring students to define the tasks and sub-tasks need to complete the activity".

  The activities can be interpreted in multiple ways, and so the learner first needs to identify and define the tasks and subtasks he will attempt to complete.

- "Authentic activities can be integrated and applied across different subject areas and lead beyond domain-specific outcomes".
  Authentic activities encourage inter-disciplinary perspectives. Learners' knowledge is robust rather than limited.
- "Authentic activities are seamlessly integrated with assessment".
  Assessment that is removed from the task may seem artificial, and does not reflect the nature of the real-world assessment.
- "Authentic activities allow competing solutions and diversity of outcome".
  Activities lend themselves to multiple solutions, rather than a single correct outcome based on the application of strict rules.

Situated learning is a slight variation authentic learning. Collins [Collins, 1988] defines situated learning as "the notion of learning knowledge and skills in contexts that reflect the way the knowledge will be useful in real life". It is critical that the learner performs tasks and activities in an environment that will imitate the multiple ways in which their skills will be used in the future.

Theory / design

Oliver et al. [Oliver, Herrington, Herrington, and Sparrow, 1996] state that a situated learning environment must include variants of the following features:

- An authentic context for learning that reflects the way in which the knowledge and skills will be used in the real world
- Learning derived from authentic activities
- Access to expert performance and process modelling
- Coaching and scaffolding
- Promote articulation to allow tacit knowledge to be made explicit
- Integrated assessment

2.6.4 Levels of Teaching and Learning

Bloom [Bloom, 1956] and Biggs [Biggs, 1999] have both detailed elements of successful teaching and learning skills.

Biggs describes a good teaching system as being one that is based on constructive alignment. This method incorporates the twin principles of constructivism in learning

and teaching and assessment alignment in teaching. All aspects of the system should then work together to support appropriate student learning. Part of this alignment concept is that graded practices are integrated with the teaching and learning activities, and should encourage students to take a deep learning approach rather than enforce a surface one.

Biggs also describes how students relate to a teaching and learning environment in a surface or deep manner. Surface learning is when students remember a list of disjointed facts, but they understand little of the point that the author is trying to make. Deep learning happens when the student understands and interprets the author's meaning as well as being aware of the accompanying details and facts. The student is, in a sense, aware if the 'big picture'. According to Biggs, the surface approach arises "from an intention to get the task out of the way with minimum trouble, while appearing to meet the requirements", while the deep approach arises when the student feels a need "to engage the task appropriately and meaningfully".

Theory / design

Bloom's Taxonomy categorises the level of abstraction of questions that would occur in an educational setting. These categories are knowledge, comprehension, application, analysis, synthesis and evaluation, and are outlined in Table 2.1

Knowledge observation and recall of information knowledge of major ideas
Comprehension understanding information interpret, compare and contrasts facts translate knowledge into new meaning
Application use information use methods, concepts and theories in new situations solve problems using required skills or knowledge
Analysis seeing patterns

organisation of parts identification of components
Synthesis use old ideas to create new ones generalise from given facts predict and draw conclusions
Evaluation compare and discriminate between ideas make choices based on reasoned argument recognise subjectivity

Table 2.1. Bloom's Taxonomy

Bloom [Bloom, 1984] states that one-on-one human tutoring is a more effective method teaching then traditional classroom-based instruction. Generally, the greater that engagement of the student the higher to level achieved in the taxonomy.

2.7 Chapter Summary

In this chapter we have looked at various online learning techniques. We began by looking at general computer aided learning, which is a multi-faceted area. We then described Intelligent Tutoring Systems, followed by an overview of Human-Computer Interaction issues. Finally, we discussed a number of pedagogical methods that are used in a classroom-based environment, with a view to applying them to our online tutoring system.

Chapter 3
Related Work

3.1 Introduction

Our project draws on ideas in the Intelligent Tutoring System and the Computer-Aided Learning domains, specifically in the area of SQL tutoring. For this reason it is important to be aware of similar systems.

In this chapter we will first look at the workings of two general ITSs. While both systems aim to teach the user, their domains are very diverse. The first aims to teach users a very practical skill. The second deals with teaching a university curriculum subject. By studying both systems we can see the subject range over which an ITS can spread. We will then move onto a non-intelligent SQL tutor to see how SQL can be taught at its most basic level. Finally, we will describe and discuss a further three SQL select statement tutors. The first of these is described as an interactive SQL tutor. The second and third systems are modelled on an ITS architecture and so have ITS properties. Our system will be related to these last three high-level tutors.

3.2 Intelligent Tutoring Systems (ITSs)

Let us first look at two ITSs, the TAO ITS and Andes. Both systems are examples of ITSs that deal with very different subject areas. The first deals with teaching a slightly unusual and very practical based subject. The second shows how a conventional school or university subject can be taught by a computer. By studying these examples of ITSs we will learn about the range and ability of such a system when used for different teaching purposes.

3.2.1 TAO ITS

TAO ITS [Stottler and LCDR Vinkavich, 2000] is an intelligent training program designed to train U.S. Navy officers in high-level tactical skills. Its main aim is to

increase the active training given to officers, thus improving their ability to apply the tactical conceptual knowledge they possess.

There are three parts to the system. The first is a scenario generator, used by instructors and programmers to create any amount of simulated scenarios. An element within this (for example, an airplane) can be given its own characteristics and behaviours, thus allowing it to react in a realistic manner. The second part if the system is the actual ITS. This allows the user to practice various tactical concepts on certain simulated scenarios. The user can choose the scenario, or the system can select them based on tactical concepts the user has recently failed or has yet to practise, determined using pattern matching. Feedback is given here through an evaluation summary, which lists the situations in which the student has demonstrated understanding of principles by applying them correctly. The final section of the system is the instructor interface tool, with which the instructor can review the user's work.

Surveys show that users have had a favourable reaction to the TAO ITS.

3.2.2 Andes

Andes [Gertner and VanLehn, 2000] is an ITS designed to teach introductory physics at college. The key aims of the system are to:

- "Encourage the student to construct new knowledge by providing hints that require them to derive most of the solution on their own,
- Facilitate transfer from the system by making the interface as much like a piece of paper as possible,
- Give immediate feedback after each action to maximise the opportunities for learning and minimize the amount of time spent going down wrong paths, and
- Give the student flexibility in the order in which actions are performed, and allow then to skip steps when appropriate".

The system uses coached problem solving. With this, the system takes a background role as long as the student is proceeding along a correct solution. Should the student begin to encounter difficulty the system provides hints.

Students may use the system to perform a qualitative analysis of a given problem as well as calculating the quantitative solution. They may use a drawing tool draw, for example, force vectors. They are then asked to define the complete object that they

have drawn. This allows the system to provide feedback on what the student has drawn as well as on what they meant by the drawing. When the student writes a formula they must also define elements of it in the same way, and for the same reason. Andes uses Bayesian networks to store information about the user in the student model. This allows uncertain information to be stored correctly about his beliefs, goals, and the level of knowledge he has about the domain. For example, Andes cannot presume that the student does not know how to perform a certain action just because he has not attempted it.

The system returns feedback to the user about whether his action was correct or incorrect, along with hints and help messages. Feedback is immediate, so when an answer is correct its colour is changed to green, or changed to red when it is incorrect. Hints are given when the student asks for help, and are generated based on the student's model. An English text string is created, based on the topic the student needs help with, by using a hint template.

3.3 SQLcourse.com

SQLcourse [SQLcourse] is an introductory SQL course available on the Internet, free to all to use. It aims to teach students the basics of SQL statements.

The system's select statement tutorial is comprised of a number of pages with text about various types of statements, along with code snippets and tables. At the bottom of most of these pages is a section where the student can answer a set of questions based on a pre-defined table. Answers, in the form of SQL select statements, are typed into a text box, and are then submitted. Once submitted, a page is displayed with the resulting table. From here the user may press the back button to try submitting again or to continue with the tutorial.

The system is not intelligent and does not correct the submitted queries. However, there are links to the correct answers (i.e. the correct query text), allowing the user to check the accuracy of his attempt himself by comparing the code of each query. There is no help given to the user if he makes a mistake, nor is there any degree of personalisation.

3.4 SQLator

SQLator [Sadiq, Orlowska, Sadiq, and Lin, 2004] is described by its developers as a "web-based interactive tool for learning SQL", rather than an ITS. It is presented to the user as an online learning workbench. With it, the user can create his own learning space in a secure environment. The main features of the workbench include [Sadiq et. al, 2004]:

- A short multimedia tutorial explaining basic concepts of SQL
- A collection of tables and accompanying practise queries
- The opportunity for the student to create and execute a query based on the aforementioned tables
- An evaluation of said practise queries and student executions (this is deemed by the authors to be the key function of the system, and will be described in more detail later)
- Users may makes notes for each attempt
- Users may see individual and group status reports
- Users and staff may interact with each other using the tool.

As with our and similar systems, SQLator looks at the select statement rather than any other type of SQL statement.

The main idea behind the tool is that user solves a set of questions that cover various aspects of select statement theory. This will be the case with our system also. These student solutions are corrected by an "intelligent engine" [Sadiq et. al, 2004]. A form of feedback is then provided. The authors note that a number of variations of a select statement can all produce the correct answer. They state that their tool allows for this and maintains consistency.

As mentioned previously, the key function of the system is to evaluate queries submitted by the user. To do this, the system's Equivalence Engine judges if the student's SQL solution correctly corresponds to the given English question. The query is not actually executed by an actual database server at this stage – this is done separately if the student wishes. The system alerts the user as to whether his solution is correct or not, and a record of correct answers is stored. The user may also view the system's correct answer if he wishes. Delayed feedback is present in the form of

emails or posts by staff members when they wish. The authors say that in the future they would like to provide a hint system.

From the evaluation presented in [Sadiq et. al, 2004], this system seems to provide benefit to a large class of novice SQL users. As this system is not available for general use we could not evaluate it ourselves.

3.5 Acharya

Acharya [Bhagat, Bhagat, Kavalan, and Sasikumar, 2002] is a system that teaches its users how to program in SQL. Used in a Database Management System course, it aims to let users attempt questions posed by the system and receive "qualitative feedback" on their submitted solutions.

3.5.1 Architecture and Workings of System

The authors describe Acharya as an Intelligent Tutoring System, and later it is deemed to be a Web-based learning environment. The architecture is similar to both of these. It is composed of an interface, a pedagogical module, a database containing the set problems and solutions, a student module and student records.

The system's interface is made up of three areas. The top area shows the schema of the database the student will query, along with the question in text from. The middle area allows the user to type in his answer. This section is divided up so that the user types the select clause in one part, the from clause in another part, and so on for all six SQL clauses (i.e. select, from, where, group by, having, order by). Finally, the bottom section of the interface contains the result of the user's query.

The student, having chosen a particular question to attempt, submits his proposed solution via this interface. This is then corrected by comparing it to a predefined expert solution. This comparison is further split into six parts, based on the six SQL clauses. The comparison process is made up of three stages – pre-processing, atom processing, and truth table processing. The student's submission is not executed in the database.

Pre-processing: Here, the query to be corrected is checked for certain keywords (such as "between", "in", "exist"). These keywords are replaced with equivalent words or

symbols. For example, a clause with 'between' will be reconstructed to include appropriate greater-than-or-equal-to and less-than-or-equal-to signs. By doing this, 'AND' or 'OR' can be used to evaluate conditions, which will be needed later in the comparison process.

Atom processing: At this stage, every individual condition in the where clause is referred to as an atom. This is needed in the next stage for two main reasons:

1. To find negated conditions used by the student

For instance, if the expert solution contains a condition "age > 25" and the user's solution includes "age < 25" instead, the expert condition can be called 1 and the user's condition can be called not 1. This will be important in the next stage of the comparison.

2. To identify some kinds of mistakes that might occur in a query.

For example, if the student types "age = 25" instead of the correct "age > 25".

Truth table processing: The system uses truth tables to test if the expert solution and the user's solution are equivalent. The developers work under the assumption that if two Boolean expressions with the same set of variables have the same truth table then the Boolean expressions are equivalent. The 'where' clause of both the expert and user solution is analysed as outlined above to form their Boolean expressions. The truth table of these expressions is then generated. This is done, firstly, by counting the number of variables present and then finding all combinations of 0 and /or 1 for the set of variables. The system then looks at the output of the Boolean expression for every combination calculated. Then, to evaluate if two expressions are equivalent, the relevant truth tables are constructed and compared the same as outlined above.

It is unclear what types of structures are required to answer the questions posed by this system. For example, we cannot tell, from the literature available, if the student needs to use nesting structures, union structures, etc. in his answers.

3.5.2 Adaptivity / Personalisation

Adaptivity in the Acharya system is achieved in two main ways. Firstly, the student module contains general information about the student, along with his knowledge level of all concepts and the number of hints he has been given. It also

stores the certainty factor of concepts he has learned; that is, how much the system thinks the student has learned about the concept. The course material is broken into a set of units, each of which corresponds to a particular SQL concept. A unit may or may not have pre-requisites. The system recommends units or concepts that the student should study based on the pedagogical module and information in the student module. The student is also free to follow his own path through the system if he wishes.

Secondly, the pedagogical module decides what feedback actions are to be taken if the student answers a question incorrectly (discussed in the next subsection).

3.5.3 Learner Support / Scaffolding

As previously stated, students enter their query in a maximum of six parts based on the six select statement clauses. This, it is said, is to limit the cognitive load on the student, so that he can concentrate on the "higher-level query formulation issues" rather than on certain syntax issues.

Feedback is displayed if the student submits an incorrect answer. Specifically, this feedback is text indicating where an error was made and a link to appropriate course material is displayed. If the answer submitted is correct the result set formed by the query is displayed. Should the student make a number of mistakes in one question, the system chooses the most basic one and displays hints for this only.

The overall learning-by-doing method follows a constructivist path. The system is not stand alone in that, unlike with SQLcourse, its primary emphasis is on SQL practice alone rather than on SQL theory and content. However, a set of tutorials has been designed for instances when the student gets a question wrong and may need to see some theory.

3.5.4 Human-Computer Interaction / Learner Control

Acharya recommends questions for the student to attempt, but he can choose to ignore this recommendation and pick the next question by himself. It is thus a variation of a recommender system, as it suggests questions based on the tasks previously completed by the student.

As with SQLator, Acharya deals with select statements only. In the future, the developers hope to enhance the student module so that it will contain a representation

of the student's reasoning process, although we are not told exactly how this will be used to improve the system. The current system selects from a set of pre-written stored questions. The developers state that, in the future, they would like to automatically generate problems for the user to attempt. They point out, however, that some issues lie with this. For example, Acharya aims to give students problems of an appropriate complexity. With automatic question generation, the complexity of the question will have to be correctly set. They hope to develop an automatic question generation module that will set and correct questions according to a various templates. These templates will be predefined to suit the various select statement topics within the system.

Java servlets form the bulk of the system. Acharya is not available for open use on the Internet.

3.6 SQL-Tutor

SQLTutor [Mitrovic, 1997] [Mitrovic and Martin, 2000] is a learning-by-doing Intelligent Tutoring System developed in the second part of the 1990s by Antonija Mitrovic and her team in New Zealand. It aims to provide an environment for upper-level undergraduate students to practise and receive feedback on the SQL select statement. It is meant to be used as a complement for classroom-based teaching and learning, as opposed to acting as a replacement for it. For this reason, the developers work under the assumption that users have already been exposed, probably through lectures, to the theory behind database management systems. This assumption will also apply to our system.

SQL-Tutor is not available for general use on the Internet.

3.6.1 Architecture

The underlying principle behind SQL-Tutor is Constraint-Based Modelling, a method used to describe what the student does not know, rather than what the student does know. Originally proposed by Ohlsson [Ohlsson, 1992], Constraint-Based Modelling uses abstraction to "overcome the overspecificity problem" [Mitrovic and Ohlsson, 1999]. Knowledge about a domain is represented by constraints on correct solutions in that domain. Basically, each constraint represents a set of incorrect

solutions. Every constraint is made up of two main units: the relevance condition and the satisfaction condition. Mitrovic and Ohlsson explain that the relevance condition identifies the class of problem states for which the constraint is relevant, while the satisfaction state identifies the class in which the constraint is satisfied. Checking if a problem violates a constraint is then a two part process. First, the problem is checked against the relevance condition of the constraint. Then, all constraints that are deemed to be relevant are checked to see if the satisfaction constraint is met. If a relevant constraint is not satisfied, then that constraint has been violated.

A simple constraint, specifying that the select clause cannot be empty, is as follows [Mitrovic, 1997]:

```
(p 2
"The SELECT clause is a mandatory one.
Specify the attributes/expressions to retrieve from the database."
t
(not (null (clot-value ss 'select)))
"SELECT")
```

Constraint-based modeling is moved to an intelligent tutoring environment by creating a set of constraints for the educational domain and telling the student about any constraint violations that he may make while he progresses through a tutorial. If enough time is spent in making the constraints "psychologically appropriate" (such as creating useful feedback messages) then they should correct the student's answers as an expert in the domain would. Ohlsson [Ohlsson, 1992] explains that constraints can be designed so that they are triggered only when there is some pedagogically useful information about the student; if a particular type of information is not useful then the constraint can be written so as that information is always ignored by the relevance condition. According to Mitrovic and Ohlsson [Mitrovic and Ohlsson, 1999] a useful set of constraints does not necessarily have to be a complete set. Feedback on common errors can prove to be valuable, even if uncommon errors are missed. However, writing any set of constraints can be time-consuming [Martin, 2003] in terms of designing psychologically suitable messages, and choosing and coding a constraint to cover every possible error type.

The architecture of a constraint-based ITS differs slightly from the architecture of a typical intelligent tutoring system in that the student module is replaced with a

pattern matcher. A stored set of constraints acts as an input into this, and into the pedagogical module.

The architecture of the SQL-Tutor [Mitrovic, 1997] is a variation of this constraint-based ITS. It is composed of stored student models, constraints, the system's problems and solutions, and the accompanying databases. Along with this, there is a constraint-based modeller, a pedagogical module, and an interface.

The system uses a store of ideal solutions as there is no domain module present, and so the system cannot solve problems by itself. The developers' reason for this is that select statements are written in natural language, so using a predefined set of solutions will be easier to design and easier for the system to use. This will also apply to our system. SQL-Tutor is not connected to an actual DBMS, which removes some element of realism.

The system's interface is similar in many ways to that of Acharya (section 3.5), although it seems this system was developed before Acharya was. The text of the question is displayed in the upper part of the interface. The middle section of the interface window has space for the user to type in his query. This is divided according to the six SQL clauses – select, from, where, group by, having and order by. The user types each clause into the appropriate section.

Constraints are used here, coded in Lisp, to compare the user's solution to the stored ideal solution. The constraints used are divided into two categories; those that deal with syntax, as there is no connection to an actual DBMS, and those that deal with semantic issues. The constraints for this particular system were designed based on the domain knowledge and by analysing differences between possible correct and incorrect solutions. A constraint class has been defined in the first version [Mitrovic, 1997] of the system, and has persisted in more recent versions. It is composed of:

- The unique identification number of the constraint
- The relevance and satisfaction patterns mentioned previously
- A description of the constraint
- An error message that will be displayed to the user (hint) should the user violate the constraint in question
- The SQL clause the constraint refers to which is used as feedback (error-flagging).

The patterns described in point two can be a combination of Lisp functions and conditions that match the user's solution to the ideal solution. Examples of actual constraints are given in [Mitrovic, 1997].

A PhD thesis by Brent Martin [Martin, 2003] describes extensions to the SQL-Tutor. Constraints are used in the same way as before. Martin explains that some constraints (e.g. HAVING constraints) are practically identical to others (e.g. WHERE constraints). An extra category of constraint, the tidying constraint, has been added to the already present syntactic and semantic constraints. It's used to discourage undesirable properties in the user's solution. In Martin's version of the system, tidying constraints are used for over-qualification only, to avoid superfluous elements.

Martin also deals with the issues of improving the partial solution feature and with automatically generating problems. In the original version of SQL-Tutor partial solutions displayed are fragments of the ideal solution based on one of the clauses in which an error has occurred. While this fragment is correct when applied to the overall ideal solution, it may not be correct in the context of the student's proposed solution. Therefore, displaying a pre-written fragment of SQL may be of no help to the user. Indeed, it may serve to confuse the user even more. To overcome this, Martin develops a constraint representation that makes information about that constraint more accessible so that it can be satisfied by a partial solution generated by an algorithm. Specifically, this information is about how to generate a valid partial solution from a group of inputs that are relevant to the constraint in question. The algorithm needed reverses the logic of the constraint evaluator. So, when given a satisfaction condition it generates a partial solution that satisfies the constraint with respect to both the ideal solution and the student's solution. While this is a good idea, its implementation would also be very time-consuming.

Martin's extended SQL-Tutor also attempts to generate problems automatically. He works under the key assumption that a SQL statement can be produced from a suitable starting point. An individual constraint is chosen, relevant to a student's information in the student model, and an algorithm is applied to it. This algorithm produces the ideal solution of a problem involving the particular constraint. This same algorithm can be applied to the full constraint set, hence producing problems that cover the entire domain.

3.6.2 Adaptivity / Personalisation

The student model contains personal information about the student (name and stereotype), a history of the problems that the student has solved, and an idea of what the student 'knows', that is a measure of how often a relevant constraint was successfully used by the student. These are updated using a simple increment function.

The pedagogical model is an important part of the overall architecture and adaptivity of the system as it selects problems to be given to the student and generates actions based on the student's answers. Through this, it aims to provide individualised instruction. Problems are selected based on the contents of the student model, by choosing the constraints it appears the student is having problems with. This is determined by selecting the constraint that has been used the most and has been violated the most (i.e. maximum (used – correct)). Alternatively, a constraint that has not been used at all may be selected. The student may choose his own problem to attempt.

3.6.3 Learner Support / Scaffolding

Using the interface students enter their query in six parts, based on the six SQL clauses. The developers explain that this is to minimise the cognitive load on the student; students do not need to remember keywords or the correct ordering of clauses. The lower part of the interface contains a visualization of the database schema the student is working with. Primary keys are underlined. The student can ask for a description of any select statement element, for example functions, expressions and predicates. The same issues regarding fading of scaffolding that apply to Acharya also apply to this one.

The pedagogical model also deals with feedback to be returned to the student. There are five levels of feedback [Mitrovic, 1997]:

1. Positive or negative feedback – tells the student if his proposed solution is correct or incorrect
2. Error-flag – indicates in which clause an error has occurred. If an error occurs in more than one clause only one will be selected for display. This is selected based on maximum (used – correct).
3. Hint – this is taken directly from the constraint that has been violated. Again, only one constraint is selected if more than one has been violated.

4. Partial solution – the ideal solution is displayed, for the violated constraint only
5. Complete solution – the entire ideal solution is displayed. Only one violated constraint is displayed to avoid puzzling the student.
To recap, scaffolding exists in the form of the individual clause sections in the interface, and feedback provided. Scaffolding ideally should be faded, but these are not.

3.7 Conclusion

Here we have looked at various systems that exist, some that are intelligent, some that teach SQL, and some that are a combination of both. First we looked at two ITSs that teach naval strategies and college level physics. We then saw a non-intelligent system that teaches SQL. Here it is up to the user to determine if they have answered the question correctly and how to rectify their mistakes, if they have made any. We then described three systems that combined 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. By studying them we have a useful insight into the direction our system should take.

Table 3.1 is a summary of the various features offered by the four SQL related systems. This is based on information that is made explicit in the literature referenced throughout this chapter.

Table Legend
Web-based – The system is presented for use on the Web rather than being distributed
Correction – The system corrects statements submitted by the user and returns an answer of correct / incorrect accordingly
System driven – The system is in control of the curriculum i.e. it chooses the questions that the user will attempt
Learner driven – The learner is in control of the curriculum i.e. he controls the questions that he will attempt
Feedback – The system provides pedagogical feedback, such as hints or partial solutions

Guidance – The system offers personalised guidance based on the user's previous results

Adaptivity – The system is adaptive or personalised

SYSTEM	SQLCOURSE	SQLATOR	ACHARYA	SQL-TUTOR
Web-based	✓	✓	✓	✓
Correction		✓	✓	✓
System driven			✓	✓
Learner driven	✓	✓	✓	✓
Feedback		✓	✓	✓
Guidance				
Adaptivity		✓	✓	✓

Table 3.1. Summary of System Features

We are particularly interested in the correction and scaffolding methods employed by the SQL systems. All but one (SQLcourse.com) of the four SQL systems described above include some form of correction and feedback. We will discuss these three systems' correction and feedback strategies in the following sections.

3.7.1 Correction Techniques

All but one (SQLcourse.com) of the four SQL systems described above include some form of student submission correction.

SQLator 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. Compared to other SQL tutors studied, this is quite a novel approach that appears to work for the authors. However, this type of correcting could be quite trivial, and may not be the ideal solution for correcting all types of select statement errors. Also, the query is not actually executed at this correction stage. We feel this could compromise the system's realism. Furthermore, this means that result set comparisons with other tables are not possible.

Acharya 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 the two expressions are the same. The process will fail if one of the expressions is made up of more atoms then 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. The student can, for example, select an incorrect number of attributes in the select clause but still be marked correct overall. As long as his where clause is correct he will be deemed to have submitted a completely suitable statement, even if the result set is incorrect. We can conclude that Acharya's method of correction has major accuracy issues.

SQL-Tutor uses constraint-based modelling 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 then checks each submitted query for relevant constraints that have been violated. This method of correction appears to yield a high level of accuracy. However, 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. Spending such time and man-power on one component of a system is not always a feasible option for project leaders and software developers.

3.7.2 Feedback and Guidance

The three systems discussed in section 3.7.1 offer some form of scaffolding to the student.

SQLator's automatic feedback consists of an error flag telling the student if his answer has been marked as correct or incorrect. There does not appear to be a more complex type of automatic feedback on offer. Asynchronous feedback is offered by allowing staff members to email or post messages when they want to. With our system, we would prefer feedback to be automatically generated and synchronous, that is available immediately. We feel this would be better for the student's overall learning experience. The authors seem to agree with this; they say that in the future they would like to provide a hint system.

Acharya provides feedback in the form of error flagging and hints. Hints are comprised of text and links, which will be of double benefit to the student. One hint only is displayed in cases where the student has made more than one error. We would

prefer to allow the user to have an option here – they may view one hint, or they may view all hints that apply. This system does not offer optional text guidance based on the student's previous actions.

SQL-Tutor is quite advanced where feedback is concerned as it offers both hints and partial solutions. There are five levels of feedback ranging from a simple indication of whether or not the student's query is correct to offering the complete solution. Feedback is in text form only however, and relevant links are not offered. Links to relevant topics are often useful to students and put his errors into context. There is no optional textual guidance based on the student's previous actions.

3.7.3 Positive Aspects and Limitations

SQLator appears to be an SQL tool rather than a tutor as many of its features are static and non-automatic. It evaluates students' queries by comparing it to the English question. This allows it to indicate to the student if his answer has been marked as correct or incorrect. This indication, if accurate, is of benefit to the student. The student must separately submit his query for actual execution. This, we feel, is a limitation. We see the system's lack of automatic and synchronous feedback as a further limitation. The student must wait for the teacher to communicate with him. As this is the case with classroom-based teaching this web-based system offers no improvement in this respect.

Achraya corrects students' submissions and offers error-specific feedback – a positive aspect. In addition, feedback is comprised of hyperlinks along with textual hints, thus allowing the student to benefit from a two-dimensional feedback message.

We see limitations to the system's correction method. It appears to correct accurately only when certain conditions apply, such as when the student's where clause is of a certain length. We will develop a more robust method of correction.

The division of the interface to suit the six SQL clauses is an idea that we will not use for two reasons:

- The division does not fit in with a realistic environment. In a real-world SQL execution environment this form of scaffolding is not available.
- Scaffolding should be faded. This type of scaffolding might be problematic to fade in a guided discovery environment, and consistency would be lost. Overall, we aim for an uncluttered interface in a realistic setting.

The same interface issues apply to SQL-Tutor. The cost of developing a vast amount of constraints is a further limitation to the system. SQL-Tutor has positive aspects however. Despite the need for a large number of individual constraints, the system appears to be considerably accurate. Other automated SQL tutoring systems could, permission permitting, reuse SQL-Tutors correction component. The tutor also provides varying levels of feedback.

Both Acharya and SQL-Tutor suggest the next question for the student to attempt. The student may accept this recommendation, or he may choose the next question himself. The reasons for this system recommendation are not made explicit to the student – a limitation; he is not told the areas of SQL he is having difficulty with, he is not told how the recommendation has come about, and he is not given a choice of recommended questions to select.

Chapter 4

Objectives and System Requirements

4.1 Introduction

Objectives are formed by identifying the various stages of the development of a project. Once identified, each objective needs to be met in order for the project to progress and be completed correctly.

A requirement describes a characteristic of the system that needs to be considered or included as part of one or more objectives. We can say that a requirement is of lesser importance than an objective, but it is still a part of the overall project.

4.2 Objectives

Our research question is a set of key objectives, previously outlined in section 1.4. These objectives guide our project from its design, to its implementation, and ending with its evaluation.

4.2.1 To design and build an automated tutoring system for teaching the SQL select statement.

Three objectives combined will be used to design and build an automated SQL select statement tutoring system. An appropriate architecture will first be developed (4.2.1.1). This architecture will allow for the implementation of the adaptivity techniques we have chosen. The system will then be built using Java servlet technologies and an Oracle 8i database. A methodology for correcting a select statement based on the analysis of its structure (4.2.1.2) will be included in this implementation. The system will include domain specific feedback and personalised guidance features (4.2.1.3). The chief features of the system will be:

- Provide a set of questions for the student to practise
- Allow the student to submit proposed solutions for correction
- Display the results of any correction
- Display, based on student preferences, feedback specific to a single submission

- Display, based on student preferences, individualised guidance based on the entire collection of submissions by the student

4.2.1.1 To develop an architecture for an automated tutoring environment.

The most fundamental reason for developing an architecture is to assist with the implementation process. Creating an architecture will indicate the components that need to be included in our system and how they will interact with one another.

We also want to develop an architecture using ideas obtained from our educational computing technology research, which is detailed in chapter two of this work. More specifically, we will develop an architecture that will merge both Intelligent Tutoring System and Computer-Aided Learning concepts. Doing this will harness the positive aspects of both of these online learning structures. The Learning Technology Systems Architecture (section 2.2.1) will also be helpful.

Lastly, the design of our architecture can aid in the separation of concepts such as intelligence, adaptivity and personalisation. This can lead to the reusability of system components in the future.

ADL SCORM [Advanced Distributed Learning] is a set of standards for learning resources launched by United States Department of Defence (DOD) and the White House Office of Science and Technology Policy (OSTP). On a high level it hopes to encourage [Conlan, Dagger, and Wade, 2002]:

- Reusability – content persistence of different learning environments
- Accessibility – globally accessible content repositories
- Durability – system and content will persist over time
- Interoperability – system and learning resources are independent of operating system and platform

This standard and similar standards are agreements that can limit flexibility within an architecture. However, its aims (listed above) are useful suggestions for the design of architecture. The key aims for us are reusability and durability. The architecture, and hence the final system, should be able to handle changes in curriculum. In particular we want the system to be modular. This will allow various modules to be updated if necessary; on an abstract level replacement modules will be "plugged in" to the overall system.

4.2.1.2 To develop techniques to analyse the SQL select statement.

We will analyse the SQL select statement in order to assess aspects of the select statement a typical student might have difficulty with while creating SQL select statements. The course instructor's experience will be an additional input into this process. We then want to place any error made by the student into a definite error category. Doing this will allow us to alert the student to an error in a very specific manner. This will minimise confusion for the student, leading to a quicker, less frustrating, and therefore richer learning experience.

To do this we will devise and implement a multi-level error categorisation scheme, a scheme absent in many other SQL tutoring systems. In our opinion a single-level error categorisation scheme is not broad enough to help the novice student in a specific manner. For example, telling the student that he has made a semantic error is often not enough to help him rectify a problem; the student may not understand what this is or where in the statement it has occurred. Similarly, a novice student may not benefit from a double-level error indication. For instance, where clauses are often quite long and contain many various elements. Indicating that a semantic error has occurred in the where clause may be of little benefit to the student – often if he has made an error it is due to a genuine misconception that needs to be explicitly pointed out to him.

4.2.1.3 To determine adaptivity techniques for use in a knowledge-based feedback and guidance system.

An adaptive learning environment allows us to adapt the student's learning experience based on specific preferences chosen by the student as well as the context in which the student is based. Adaptivity techniques include changing both content and navigation options. We will use both of these techniques to determine the type of feedback (local feedback) and guidance (a variation of global feedback) the student will receive based on information about his previous actions. Content and navigation will be adapted by the feedback and the guidance features.

4.2.2 To evaluate the SQL tutoring system.

When implementation is complete the SQL tutoring system will be evaluated. We aim to use four separate evaluation methods to carry out this evaluation, three of which are student based. They are:

- Study student performance using exam results

  Each year students taking the CA218 Introduction to Databases module sit a mid-semester SQL examination. This year the extended automated tutoring system was introduced while all other factors remained constant. By comparing this year's exam results to those of previous years we may be able to discover if the system has been effective in helping students to learn how to use the SQL select statement.

- Discover students' opinion of the system

  The opinions of students that used the system can help us to evaluate the usability of the system. We will therefore ask students questions that will help us grade the tutoring system in terms of ease of use and navigation, course content, and other features relevant to the tutor.

- Verify the above results using web usage mining

  We can evaluate the system by studying how student's used the system – when, how often, for how long, etc. We will obtain the necessary information through a variant of web mining called web usage mining. This information will be used to verify our student performance and student opinion results.

- Software Evaluation

  In this chapter we outline a set of objectives we want to fulfil and requirements that need to be considered. We will evaluate the system based on these objectives and requirements to ascertain if they have been met.

Together these methods will allow us to assess the overall effectiveness and usability of the tutoring system.

An evaluation is of key importance as it will yield information regarding the current systems performance, along with identifying areas that could benefit from improvements in the future.

4.2.3 . To fulfil these objectives in accordance with educational principles and pedagogical considerations.

The design and development of any learning system should be based on sound pedagogic strategies. Teaching and learning theories are often neglected when developing an online tutoring system. We will therefore design and implement our tutoring system in accordance with educational principles and pedagogical

considerations, to ensure the student has access to a complete and pedagogically sound learning experience.

4.3 Requirements

In order to fulfil the objectives discussed in the previous section we need to consider a number of requirements at various stages of the project. Some of these requirements are standard features of an SQL tutor (4.3.2, 4.3.4 and 4.3.5), some are derived from teaching and learning best practises (4.3.7), and the remaining are requirements we have defined based on research into teaching and learning online (4.3.1, 4.3.3, 4.3.6), detailed in chapter two.

4.3.1 Realistic setting

In 'real life', select statements are written and submitted to a DBMS for execution. A result set is then returned to the user. We wish to replicate this real-world situation by having students write their statements in a realistic setting. Thus, students will write statements from a blank canvas in a simple text area, and submit these to an actual Oracle database server. They will be returned with the actual result set of their statement. This element of realism is missing in some other SQL tutoring systems, where statements are evaluated by the system instead of actually being executed, and where the student enters the statement clause by clause.

4.3.2 Learning-by-doing / active learning

From the literature review we can see that active learning is a very successful way of learning practical tasks. Creating select statements is a practical task, therefore we can use the theory of learning-by-doing in our system. This way students can actively create statements, as opposed to simply reading about how it is done. This form of procedural knowledge generation should lead to a richer and more successful learning experience.

4.3.3 Guided discovery

We will incorporate a guided discovery navigation strategy. With this, the primary emphasis is on the student and his ability to seek information, rather than the system overwhelming him with data. This strategy allows the student to control pace and

timing, management of learning activities, and responses to exercises. Thus, the student should have the opportunity to decide the questions he wants to attempt, the ordering of these questions, and choose the level of feedback and guidance that he will receive. Consequently, the student will have a considerable amount of control over his learning experience. This should be of benefit to him through making him feel empowered and gain a sense of responsibility when learning.

The system should support this guided discovery strategy by ordering tasks. The student may follow this ordering if he desires.

4.3.4 Correction of submitted statements

It is important, in a formal language tutoring system, to indicate to the student if he has answered a question correctly. The results of the correction method employed will also influence the scaffolding that will be displayed to the student. Errors made should also be contextualised; the system's correction component should analyse a submitted SQL statement based on the syntax and semantics of the language, but it should also consider typical errors as experienced by database instructors. Accordingly, we want the system to correct student submissions, and for the resulting information to be categorised in an appropriate manner.

There will be significantly more detail about this in the next chapter.

4.3.5 Feedback specific to the submitted statement

We wish to return a certain level of feedback for every submitted solution, and a greater amount of feedback for incorrect solutions. Consequently, the system needs to include a multi-level feedback approach, in order to:

- Gradually increase and fade the feedback given
- Allow the student to decide of alternate levels of feedback displayed.

The student may choose the level of hint they want. Alternatively, he can choose the system default. This will start at a low level of hints, increase the level gradually, and then begin to fade the feedback after a time.

Hints will be optional, in keeping with the guided tutorial format.

4.3.6 Guidance based on personalised assessment

Guidance based on a personalised assessment will be offered to the student. This global feedback will consist of text explicitly telling him if he is having difficulties in any error category, as well as if these difficulties are low, moderate, or high. Further to this, they will be offered a menu of further questions to practise. This again will be based on the type and amount of errors they have made.

As with hints, the viewing of guidance will be optional.

4.3.7 Functional interface

The interface of the original SQL tutor was considerably functional and was based around simple colours and ordering. This functional interface will be used in our extension of the system because:

- We want the new system to be a natural continuation of the old system, as outlined above
- A functional interface allows for a more realistic database programming environment.

For these reasons, we will not divide the input areas based on the six SQL clauses, like two other SQL tutors we described in the previous chapter (Acharya and SQL-Tutor). Also, this division is a form of scaffolding that would need to be faded, which would result in a change of the interface and lack of consistency.

Our SQL tutor is an extension of an execution environment that already exists. We wish to continue with the theme of this existing system. We therefore need to keep the 'look and feel' of our new system the same as the old one. Also, the system will be part of a larger database course presented online, and so questions and solutions need to match those taught in the theory.

Chapter 5

System Design

5.1 Introduction

The implementation of our objectives is a vital part of our work. As such, we followed a simple design model (Figure 5.1). The starting point in this model is to identify needs and establish requirements, which we looked at in the previous chapter. The second and third tasks involve designing the system and then building an interactive version. This is what we will be looking at in this chapter. The system should then be evaluated, as we will see in the next chapter. This entire process can go through a number of iterations. This will be discussed in more detail in future work, later.

We will begin by defining our SQL tutoring system architecture. We will then give an overview of the information modelling required for the implementation of the system. We will then detail each of the system's components, beginning with the interface component and arriving at the pedagogical component via the correction component and student component. We will also, prior to the correction component description, explain our SQL select statement solution and correction design and define the error classification scheme that the system will use. Lastly, we will outline the student's interaction with the tutoring system, and conclude with a chapter summary.

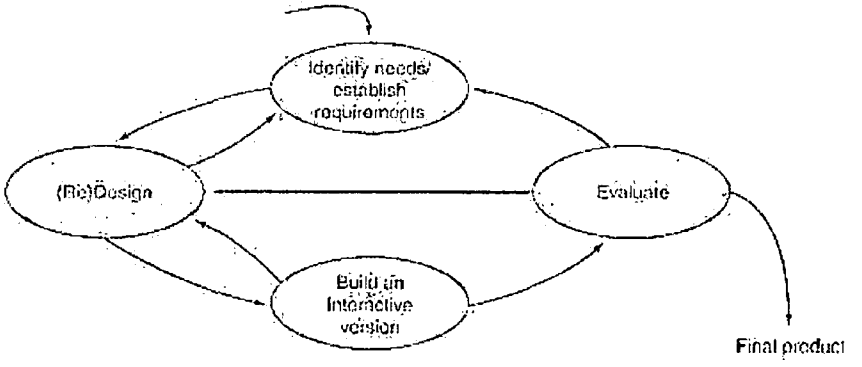

Figure 5.1. Design Model [Preece et al., 2002]

5.2 System Architecture

5.2.1 Requirements

The system to be developed is, fundamentally, a teaching and learning coach for students studying SQL for the first time. It needs to include a certain degree of personalisation (see section 2.2.3.1) based on input from the student, which will either be a single type of input or will be based on student input that has been stored over a period of time. For these reasons we can draw on typical architectures that exist for other teaching and learning systems, namely the IEEE LTSA reference architecture for learning technologies (Figure 2.2) and Intelligent Tutoring Systems (Figure 2.3).

The LTSA offers a valuable example of an architecture for information technology-supported learning. The components of this architecture are conceptual; individual components do not need to be identifiable in an actual implementation. Our system will include some, or variations, of these components. Specifically, we will include a learner entity, a means of delivery, learning resources and learner records. Also, we will include a variation of a coach and an evaluation feature.

We also choose to use components that are similar to those that form the ITS architecture. A typical ITS has an interface, an expert model, a student model and a pedagogical model. We will of course need an interface through which the student can interact with the system. The system will be intelligent to a degree (discussed shortly) but we will not use an expert model. However, the answers submitted by the student will need to be assessed in some way. Therefore we will have a correction component. A student component will be needed, as in an ITS, to store details of what the student does and does not know. This is vital for making decisions based on the student's long-term performance. Finally we will need to use a pedagogical component, although this will make less restrictive choices than such a component would make in an ITS. The pedagogical component will be used to make decisions regarding teaching strategies but will not intervene without the student asking it to.

A person's ability to behave with intelligence can be described in terms of his knowledge [Barr and Feigenbaum, 1981]. Barr and Feigenbaum [ibid.] explain that, in the field of artificial intelligence, a program can behave in a "knowledgeable" manner if it knows about objects in its domain, about events that have taken place, or about how to perform specific tasks. Our automated tutoring system will have knowledge about the ideal solutions to each question, about the types of errors that could be made

and the advice that can be given to rectify these errors, on a local or global level. Through this the system will be able to act as a human tutor would. While the system is not intelligent in the manner an ITS is, it does act in a knowledgeable way. This is primarily manifested through scaffolding (section 5.9).

5.2.2 Definition

We combine the elements discussed above to form our own architecture (Figure 5.2).

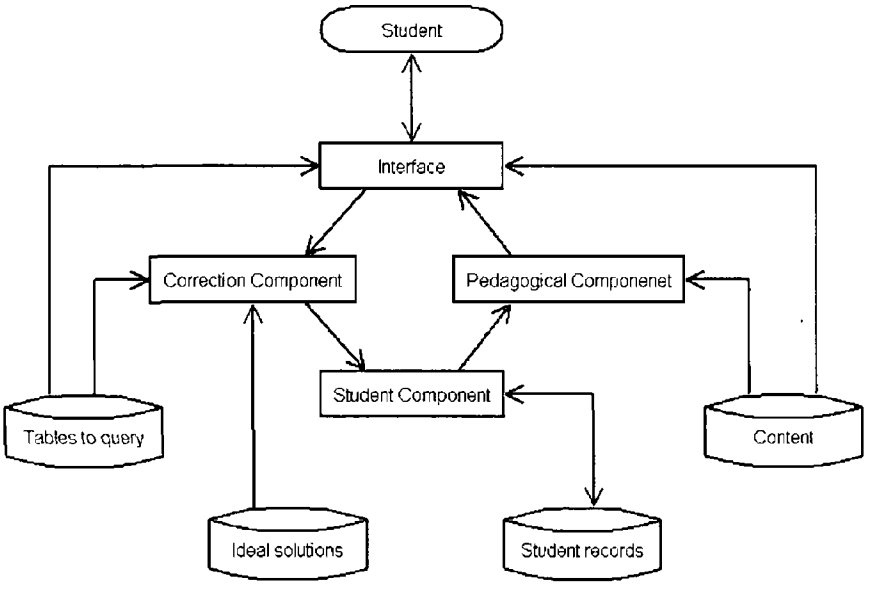

Figure 5.2. Architecture of SQL Tutoring System

The main components of the system will be an interface, a correction component, a student component and a pedagogical component, supported by information objects, which will store:

(1) The tables that the student should query in the various test questions

(2) A collection of the ideal solutions that will be used by the correction component

(3) Student records to store general and topic specific information about the student and

(4) Courseware that the student may be directed towards by the pedagogical component.

A component is a group of classes that work to together to perform one function. The aim is to achieve high internal coherence. Within a component there may be a

model of the same name. This is a single class that has the main functionality of the component. For example, in the correction component the main work is done by the correction model, with other classes in the component acting in a support role.

Our final architecture should be designed in such a manner as to allow reuse and promote modularity. Components can only communicate with certain others to promote low coupling (Figure 5.3). As we are developing a prototype of the extended SQL tutoring system, to address our research questions, the architecture should allow 'plugability' – components should be easily removed and replaced in the future, if desired. Java servlets will be the chief technologies used to implement this architecture.

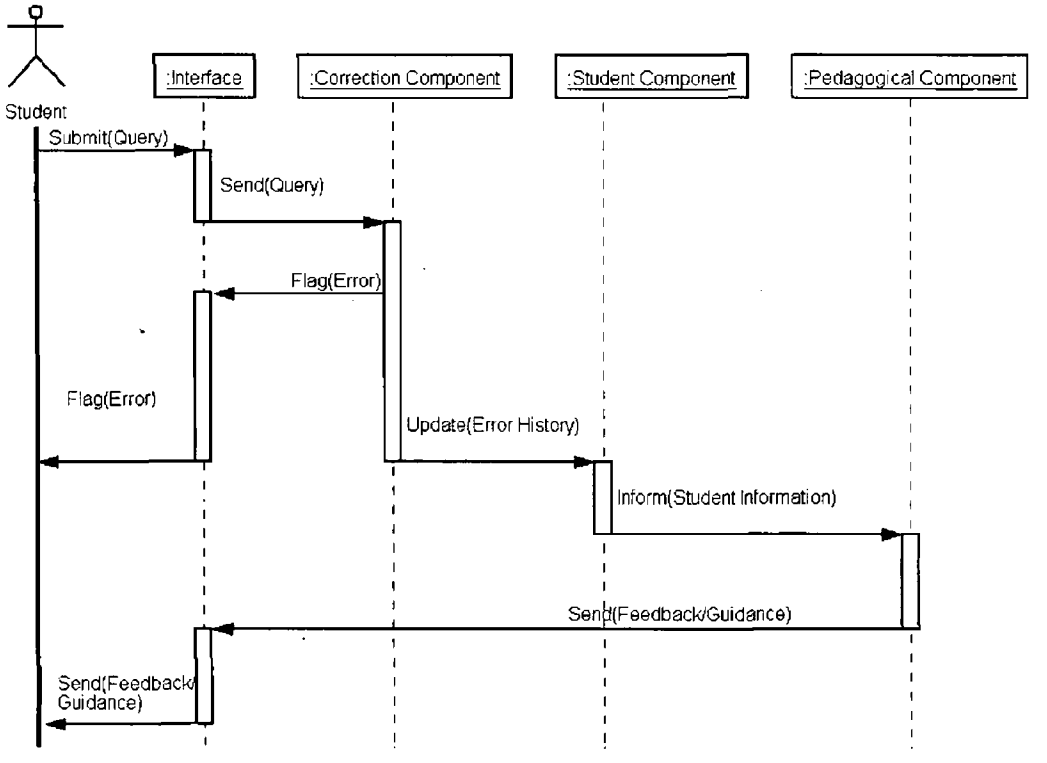

Figure 5.3. System Sequence Diagram

5.3 Information Modelling for the SQL Tutoring System

Information modelling facilitates the system's information storage and information processing. There are four types of information objects, as can be seen in Figure 5.2:

- Tables to query

- Ideal solutions
- Student records
- Courseware

Static information objects are created in an Oracle database. Their relevance to the system components are shown in Figure 5.4 and Figure 5.5 – Figure 5.4 shows the information objects' interaction with the system's components while Figure 5.5 outlines the dependencies between the various storage structures.

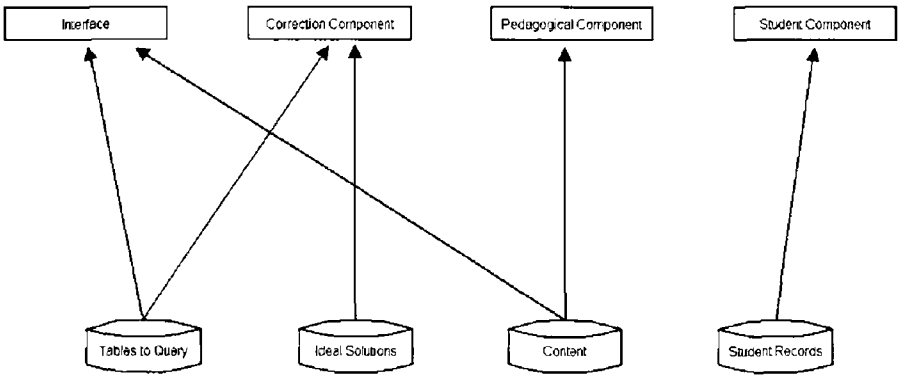

Figure 5.4. Information Objects and Component Interactions

5.3.1 Tables to Query

The original version of the system allowed the student to query actual tables stored in the database. We wish to retain this idea as it allows the student to learn in a realistic environment. These tables are called s, sp and p (standing for supplier, shipment and parts) and will be outlined in more detail later when we discuss the index of questions for the student to attempt.

5.3.2 Ideal Solutions

The system is not intelligent per se as it uses a set of ideal solutions to correct statements, rather than an expert model carrying out this correction. Instead it uses a pre-defined set of ideal solutions that it matches against the solution submitted by the student. This is explained in more detail when we discuss the correction component (section 5.7).

5.3.3 Student Records

General as well as error specific information needs to be stored about the student. There are four information objects associated with these. General information, i.e. username, password and last login time, are stored in the first information object. The second information object stores the student's preferences regarding feedback and guidance. The third and fourth information objects store information of the number of times the student has attempted each question and the amount of times he has made an error in each of the six clauses and common SQL elements.

5.3.4 Content

Information on what to teach is necessary for the system. A information object stores information of each of the questions the student should attempt – question number and title, text and details about the skills needed for each question. Prewritten text for feedback and guidance is stored in two further information objects. Please see section 5.9 for more information on this.

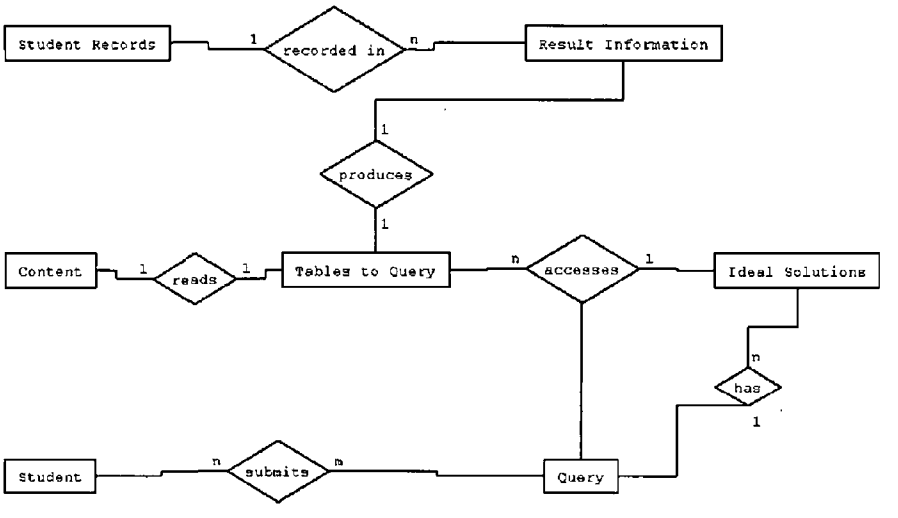

Figure 5.5. Storage Structure Interactions

5.4 Interface Component

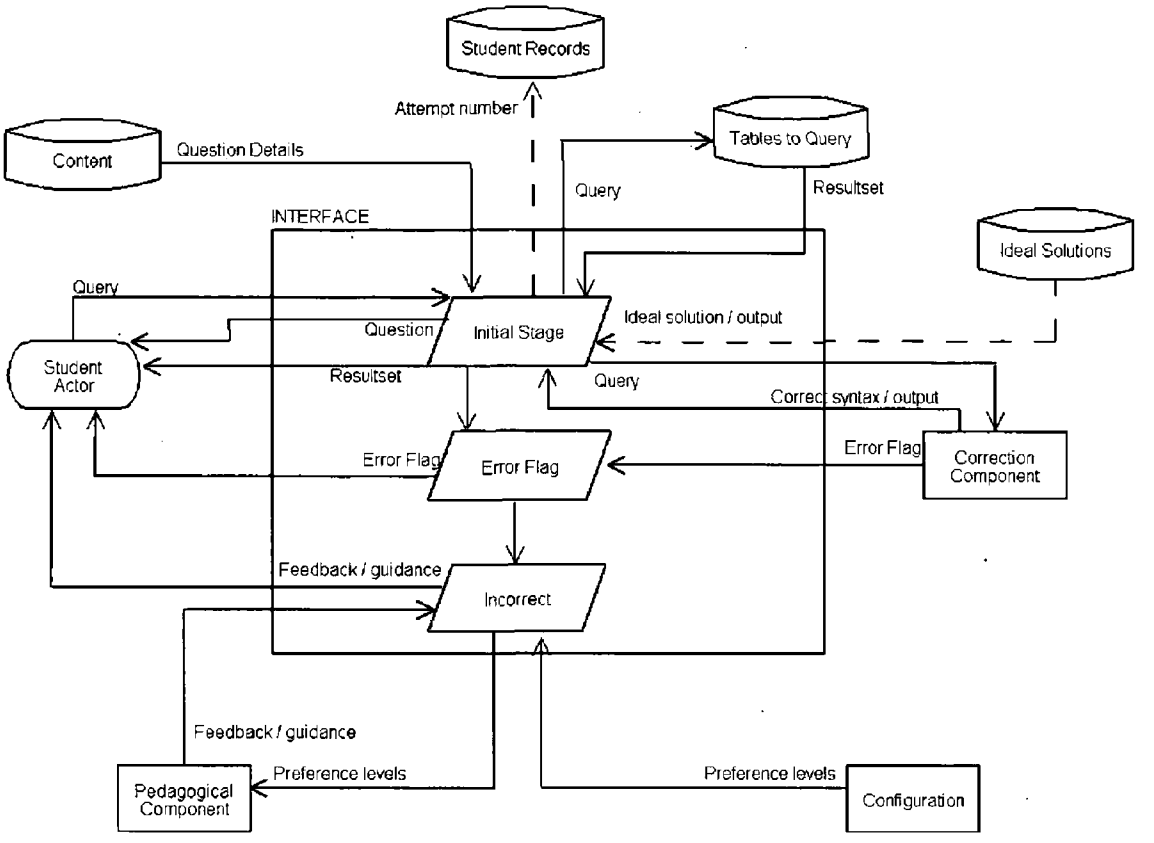

Figure 5.6. Interface Component Zoom

Direct interaction

Indirect interaction via a component not shown on this diagram

5.4.1 Preferences Setting

The navigation around the system is based on a virtual apprenticeship format and guided discovery (section 2.5.2.3 and 2.2.2 respectively). An apprenticeship begins with the master demonstrating a task to the apprentice. The online CA218 module achieves this through a series of animated tutorials. Our tutoring system accommodates the second and third stages of the apprenticeship model by allowing students to practise the task with and without help from the system. Virtual apprenticeship dictates that the student's learning should be supported by some means of scaffolding. Within the guided discovery paradigm the student assumes control of certain learning experience factors. Part of this control involves responses to

exercises. The student should therefore be able to view feedback and guidance at the level he feels suits him best. The reasons for this are twofold. Firstly, it allows the student to be supported only when he feels he needs it. Secondly, it allows scaffolding to take place. Both of these are in keeping with the apprenticeship model. Our tutoring system allows him to choose from one of five levels of feedback and one of four levels of guidance. Any combination of these levels is permitted. Alternatively, he may let the system decide the level of feedback and guidance he is to receive. With this choice, the levels of feedback and guidance are changed based on the number of times the student attempts the same question. This is done by the pedagogical model, and will be described in more detail in section 5.9.

5.4.2 Index of Queries

We implement a guided discovery learning environment (section 2.2.2). With this the student controls his navigation through the tutor – control of pace and timing, choice of content and responses to exercises. To facilitate choice of content, and pace and timing, the student should be able to choose the questions he wishes to attempt in the order he wants to attempt them. The system is based around twelve key questions, with a further twelve to be used by the guidance feature (described later). The index allows the student to select any one of the twelve queries to try. Having an index such as this serves a purpose for the both the student and the system, by facilitating guided discovery and user tracking respectively. The student's choice is supported implicitly as the questions are ordered and titled according to the system's recommended learning path.

5.4.3 Interface

One of our main requirements is to design our system in a manner that allows it to be a natural extension of the SQL execution environment that originally existed. The original interface posed no problems regarding layout, colour and ease of use. Therefore, we will reuse the original colour scheme and the original layout of text areas, buttons etc. In the original system there was a separate interface for each of the twelve queries. We replace this with a single interface, allowing for easier reusability and modularity. Let us first look at what happens (Figure 5.6), along with the corresponding reasons, in the interface.

When the student selects a question that he wishes to try he is brought to the interface that displays the question to attempt. This interface is the same for every question he may choose to attempt, but is adapted for the task in question. This is done by taking the question number, title and text from the database and displaying it in the relevant sections of the interface. This dynamic display element allows the questions the student can attempt to be changed in the database without affecting the interface in any way.

The student should type his proposed solution (query) into the text area provided and then press a button to submit the solution to be corrected. The number of times that student has attempted that particular question is updated in the student records. The student's query is then sent to the correction model for correction. The correction model sends a message back to the interface telling it if the syntax of the student's query is correct. If it is, the query is then sent to the database tables to be queried and a result set is returned. Shortly after this the correction model returns an error flag (i.e. "correct" or "incorrect") to the interface. The result of the student's query (the result set) is displayed to the student in the form of a table. It is useful for the student to physically see the result of their solution, particularly for those who prefer visual-based learning. This feature, and the accompanying code, was used in the original version of the system. The student is also explicitly told, using the error flag, if his proposed solution was deemed to be correct or incorrect.

Should the student have answered the question correctly he may move onto the next question of his choice without further output from the tutoring system. If, on the other hand, he has answered the question incorrectly he may receive scaffolding in the form of feedback or guidance. This is determined, based on the student's preferences, by the pedagogical model and is displayed in the lower section of the interface. A further form of scaffolding and personalisation is offered by the interface through form button hiding and displaying. Initially the student may only submit their proposed solution. After a number of unsuccessful attempts the interface displays a button allowing the student to view the correct output to the question. This is shown in the form of a table. After a further number of attempts another button is shown, this time allowing the student to view the ideal solution in text form as well as in table form. This is only given as an option after a certain amount of time to encourage the student to really attempt the question before they look for the correct answer, and so they will actually have to practise SQL queries.

5.5 Solution and Correction Technique Design

The student will have the opportunity to attempt any or all of twenty-four questions offered by the tutoring system. The questions are split into two sets of twelve, the first of which are the base questions with the second set used as further questions to practise offered by the guidance feature (section 5.9.2). The base questions are those that were used in the original version of the system. They were previously devised in accordance with the manner in which SQL select statements are presented and taught in the CA218 courseware. This set of questions will be used in the new system as they have been tried and tested over the last number of years, and so we are sure they are suitable for the system's educational needs. The second ('further') set of questions was devised based on the original set of twelve.

Each question posed by the system addresses a specific SQL problem or SQL construct. Furthermore, each questions is based on an individual combination of possible SQL errors from each of the SQL error category schemes discussed in section 5.6. Two of the base questions will be used as examples here. All questions, which are based on the three tables below, are included in Appendix A. Table 5.3 to Table 5.5 shows the tables on which the system's questions are based.

SNO	SNAME	STATUS	CITY
S1	Smith	20	Paris
S2	Jones	10	Paris
S3	Blake	30	Rome

Table 5.1. s (supplier)

SNO	PNO	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S3	P2	200

Table 5.2. sp (shipment)

PNO	PNAME	COLOUR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	27	Rome
P4	Screw	Red	14	London

Table 5.3. p (parts)

An SQL select statement is used to extract information from a database. It can be made up of six clauses – select, from, where, group by, having and order by. The select and from clauses are compulsory. The remaining four clauses can be added if needs be. A typical statement will consist of the necessary select and from along with where. These map to input elements (from), a computation (usually based on a condition – where), and an output element (select).

In more detail [Ramakrishnan and Gehrke, 2003]:

- "The **from-list** in the FROM clause in a list of table names. A table name can be followed by a **range variable**; a range variable is particularly useful when the same table name appears more than once in the from-list.
- The **select-list** is a list of (expressions involving) column names of tables named in the from-list. Column names can be prefixed by a range variable.
- The **qualification** in the WHERE clause is a Boolean combination (i.e., an expression using the logical connectives AND, OR, and NOT) of conditions of the form *expression* op *expression*, where op is one of the comparison operators {<, <=, =, <>, >=, >}. An *expression* is a *column* name, a *constant*, or an (arithmetic or string) expression.
- Every column that appears in ~the select list of column~ must also appear in **grouping-list**. The reason is that each row in the result of the query corresponds to one *group*, which is a collection of rows that agree on the values of columns in grouping-list. In general, if a column appears in ~the list of column names~, but not in **grouping-list**, there can be multiple rows within a group that have different values in this column, and it not clear what value should be assigned to this column in an answer row. If GROUP BY is omitted, the entire table is regarded as a single group.

- The expressions appearing in the **group-qualification** in the HAVING clause must have a single value per group. The intuition is that the HAVING clause determines whether an answer row is to be generated for a given group.
- The **order-item-list** is a list of **order-items**; an order-item is a column name. Every column mentioned in the ORDER BY clause must also appear in the **select-list** of the query; otherwise it is not clear what columns we should sort on. This clause is used to specify a sort order."

5.5.1 Select Statement Variations

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. For example, if a question requires attribute *A* to be part of the answer the ideal solution must include a single instance of attribute *A* in the select clause. In addition, the title of each question suggests the SQL concept the student should use when approaching the questions. For example it will tell students to use nesting instead of outer joins and when to use unions etc. These factors allow us to create and use an ideal solution for each question. The ideal solution is syntactically and semantically correct, and uses a minimal amount of constructs and tokens. Some variations between the student solution and the ideal solution are accepted. These variations are semantically equivalent.

Possible variations include:

- Re-ordering of attributes and table names, where sense is maintained, within the one clause e.g. both *select A, B from C, D* and *select B, A from D, C*
- Syntactical equivalence e.g. the student may use <> or *!=*
- Syntax variations may be syntactically equivalent e.g. *max(age)* is equivalent to *max (age)*
- In some cases, such as where there is no ambiguity with attribute names in separate tables, the use of prefixes is semantically equivalent to the absence of prefixes
- Redundant elements, not part of the select clause, do not affect output
- Using tables aliases results in the same output as their non-use

- *union* and *intersect* can be equivalent to *and*
- *any* and *in* are equivalent in some situations
- *not(city == 'Paris')*, for example, is equivalent to *city <> 'Paris'*
- Outer joins can yield the same output as inner joins and nesting
- Views can be created and used when querying tables

The system currently accepts the first six variations. Allowing for all semantic variations, however, would yield an extensive and lengthy implementation – similar in complexity to other SQL correction systems such as SQL-Tutor [Mitrovic, 1997]. We can view our implementation version as a means of prototyping feedback within our more comprehensive objective of integrating feedback and guidance. In terms of our overall objectives and requirements, these limitations are acceptable. The impact on the student is low as the accepted variations are consistent with the module courseware and are sufficient to allow the novice SQL student attempt a range of questions. Accepted constructs are consistent with the CA218 course material i.e. they are similar to examples presented in the CA218 online lectures.

Our error classification requires the correction model to be able to identify the type of every element in the ideal solution. For this reason an abstract notation was developed to allow the system to identify element types in the ideal solution. This notation is described in more detail in section 5.5.2.

5.5.2 Abstract Notation for Ideal Solutions

We chose pattern matching (section 2.4.1.2) as the system's central correction technique. To facilitate this process an abstract notation was created (Figure 5.9). This is used to classify any errors the student has made by matching the student's solution with the abstract notation.

For example:

Question number = 1

Question title = One table

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 (i.e. the elements the query is composed of) = select misc att att from table where att comp misc and att comp misc

Ideal solution schema legend:

- table = the corresponding element is the name of a table
- att = the corresponding element is an attribute name
- prefix = the corresponding element is a table name prefix
- comp = the corresponding element is an SQL comparison operator
- agg = the corresponding element refers to an aggregate function
- misc = the corresponding element is a miscellaneous item

The proposed ideal solution is statement that contains the minimum amount of tokens and constructs needed to correctly answer the question. We refer to keywords in the ideal solution schema legend (e.g att, table) as elements. Each relates to a token in the ideal solution. Every element in our abstract notation has a particular purpose within the query, and is derived from the grammar of SQL. We can therefore directly equate our SQL elements with an SQL grammar, such as that set out by Microsoft [MSDN SQL Elements]. For instance:

table → table-name → user-defined-name *i.e. our "table" element is equivalent to SQL grammar's "table-name" which is a "user-defined-name"*

att → column-name → user-defined-name

prefix → table-name. → user-defined-name

comp → comparison-operator → < | > | <= | >= | = | <>

agg → search-condition

misc → character-string-literal → "{character}..."

→ digit → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

→ search-condition

Throughout this work we will refer to these elements as we did in the legend i.e. using "table", "att" and so on as opposed to "table-name", "column-name" etc.

We can see from the proposed ideal solution in the above example (Question 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. Through this 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 above example the system would know that:

- *p* is the name of a table
- *colour* and *city* are attributes
- <> is a comparison operator
- *distinct*, *'Paris'* and *10* are miscellaneous items that cannot be classed as any of the previous types

5.6 An Error Classification Scheme for SQL Select Statements

We have chosen to focus on the coaching of the select statement only. The select statement is one of the most fundamental of SQL statements, used to extract information from a database. It is a declarative programming language, thus students define the result they want, not the precise methods needed to obtain this result. It can be quite simple but it also has the capacity to become quite complex. For this reason, it suits a large group of students with a wide range of ability. This will be a trait of our target audience, second year students.

Multiple concepts covered by select statements can be applied to other SQL statements, so learning how to create select statements is a good foundation for learning other SQL constructs.

The general form of a select query with all six clauses is

```
SELECT [DISTINCT] select-list
FROM from-list
WHERE qualification
GROUP BY grouping-list
HAVING group-qualification
ORDER BY order-item-list
```

Each clause performs a particular function and so addresses a different semantic issue. For example:

- The select clause specifies the output of the query
- The from clause deals with the query input
- The where clause performs a particular condition-based computation
- The group by, having and order by clauses further refine the query output.

It is important to base error categories on the difficulties the student may encounter. We have devised a multi-level error categorisation scheme (incorporating formal language properties, SQL clause types and errors with SQL common elements) to locate exactly 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. Doing this will help the tutoring system to give the student precise feedback about errors made. Such an educational error message will typically include more detail than a general system one would.

SQL is a formal language and, as with any formal language, properties of it may be syntactical, semantic, or pragmatic. Any errors made by the student can then be categorised into these three categories.

- Syntax errors are those caused by problems with the actual syntax and grammar of the language. This could be, for example, misspelling a keyword or selecting an attribute that is not present in the table in question.
- Semantic errors occur when the student creates a statement that is syntactically correct but which does not reflect his intentions correctly. By this we mean the statement can be executed but the result set is not what the student expects.
- Pragmatic errors are considerably more generic and are concerned with rules and conventions within the language.

Consider a simple table called employees that holds the name and age of three people (Table 5.2).

NAME	AGE
Jim	45
John	26
Jane	31

Table 5.4 Employees

Example 1: select **nane** from employees were age > 30

Example 2: select name from employees where age < 30

Suppose the student wishes to select the names of employees who are over the age of thirty. To do this they should submit "select name from employees". In example 1 above we can see that the student has misspelled the word "name"; he has instead typed "nane". This statement cannot be executed, as there is a syntax error due to the misspelling of a keyword. In the second example the student has made no syntax errors, so he will be returned with a result set. However, instead of using the greater-than symbol (>) he has used the less-than symbol (<). He will be returned with the names of employees under the age of thirty instead of over. This is an example of a semantic error.

An error can occur in any of the six common SQL clauses, listed above. It is for this reason that we also categorise any errors made by the student based on these six SQL select statement clauses to allow us to identify the semantic difficulty the student is having.

A select clause will typically consist of table names and attributes. It is usual for comparison operators, prefixes, and miscellaneous items such as numbers to also be included. Introductory SQL courses often include queries that use aggregate functions. We refer to these as elements of a select query. Syntax errors will apply to these, as before, so we will examine how a semantic error may occur with the aforementioned elements.

- 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 set.
- Attributes: Students may select incorrect attributes, again particularly when the attribute will not appear in the result set, an example of which can be seen in question something.
- Prefixes: Prefixes may be needed to included in a statement where two or more attributes with the same name exist in different tables.
- Comparison operators: A student may make a semantic error with a comparison operator. We saw an example of this previously when the student used the less-than symbol instead of the greater-than symbol..

- Aggregate functions: The student may use an incorrect aggregate function.
- Miscellaneous items: This category is being included for specific instances that do not adhere to the above categories and yet are not common enough to form their own separate grouping. "The optional DISTINCT keyword is one such miscellaneous item. This keyword indicates that the table computed as an answer to a particular query should not contain *duplicates*, that is, two copies of the same row. Other examples of miscellaneous items are proper nouns and numbers." [Ramakrishnan et al., 2003].

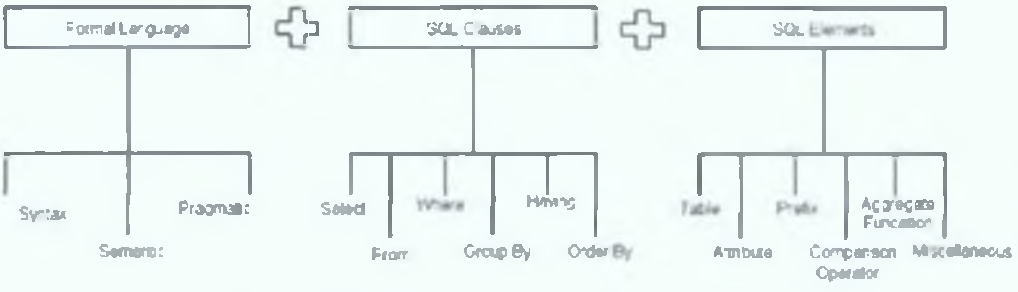

Figure 5.7. Multi-level Error Classification Scheme

Our system will implement a multi-level error categorisation scheme (illustrated in Figure 5.7). This scheme is based on a combination of the three error category schemes discussed – formal language properties, SQL clauses, and SQL common elements. The correction model will use the compound error categorisation scheme to determine the type of mistake made by the student. Let us illustrate this with some concrete examples from an actual table the students will query during a typical session.

Consider two tables, s and sp (Table 5.3 and Table 5.4), that relate to details about suppliers and shipments of parts. Suppose the student answers three questions incorrectly, all based on a different set of factors. The system will be able to identify these errors in three ways and offer specific feedback accordingly.

Question 1: Get the numbers and names of all suppliers

Student Answer 1: `select sno, name from s`

System Diagnosis 1: This statement is incorrect because the student has tried to select the attribute "name". The database server will not be able to execute this statement as "name" is not an attribute in the database – the correct attribute is "sname". This then

is the misspelling of a keyword, and is classed as a syntax error. It can also be identified as an error in the select clause (i.e. the output of the statement) with an attribute that was selected.

Question 2: Get the names of suppliers who ship a quantity equal to 200

Student Answer 2: `select sname from s, sp where s.sno = sp.sno and qty <> 200`

System Diagnosis 2: Here the student has created a syntactically correct statement, but one that will not return a correct answer. This then is a semantic error. The student has not used the correct symbol in the where clause. The system will therefore identify as a semantic one caused by a comparison operator in the where clause.

Question 3: Get the maximum status of suppliers in Paris

Student Answer 3: `select min(status) from s where city = 'Rome'`

System Diagnosis 3: As in the previous question the student has made a semantic error. In this case it is because he has selected the minimum status instead of the maximum status and has included the wrong city. Here the system will identify a semantic error due to the misuse of an aggregate function in the select clause.

The system will also identify the category of errors when the student has made more that one more mistake in his statement. The number and ordering of displayed errors depends on the student's scaffolding preferences (sections 5.9.1 and 5.9.2).

The student is advised on the categories of his statement error(s) without having to be made aware of the error classification methods within the correction component, which may serve to be confusing for him. He will be informed explicitly if there has been a syntax error, and implicitly for semantic errors. He will then be informed of the clause(s) and element(s) that the system marks as incorrect.

5.7 Correction Component

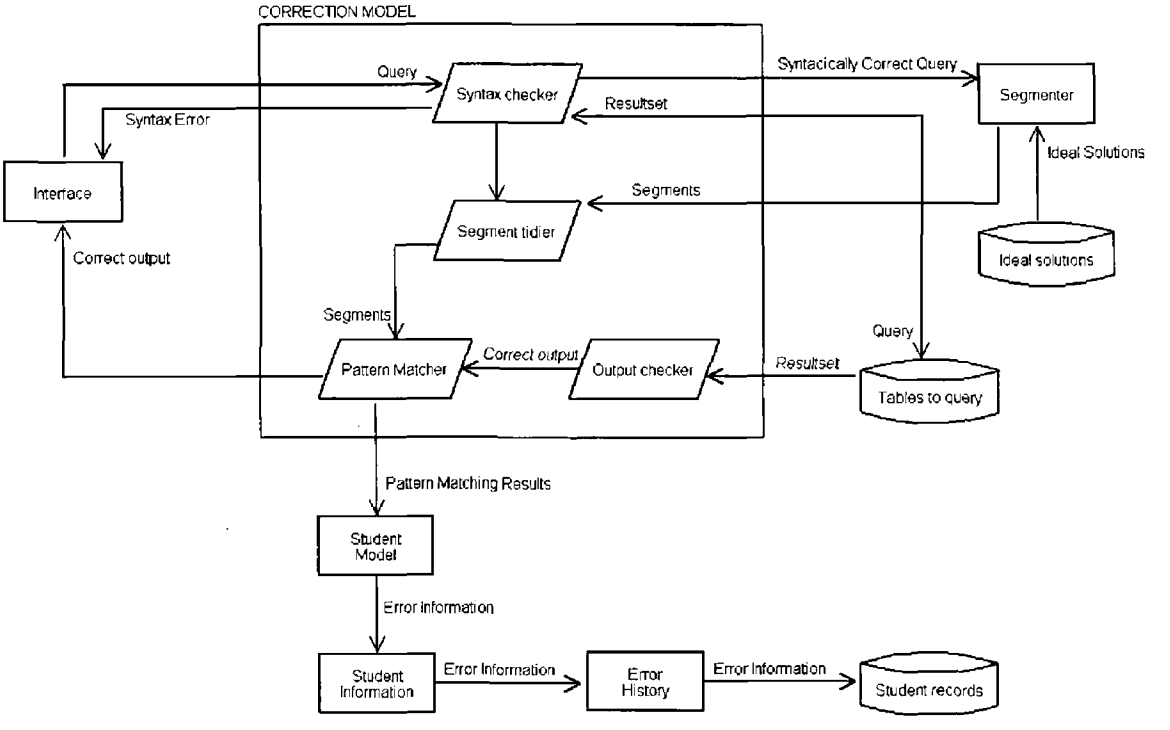

Figure 5.8. Correction Component Zoom

The workings of the correction component (Figure 5.8), which is described in detail in this section, uses the information outlined in section 5.5 and section 5.6. This component is a very important part of the overall system as it corrects the solutions submitted by the student, the result of which can have knock on effects for other parts of the system and the overall learning experience of the student.

Developing an accurate means of correction is a difficult task so it is important to follow a clearly defined methodology. We also should keep our aim of modularity in mind, so that the correction model can be easily replaced with another version in the future, if desired.

We have chosen a pattern matching method of correction (see section 2.4.1.2). This method permits us to create a flexible correction component without having to invest the time that, say, a constraint-based correction method would require. Also, as SQL is a relatively small-scale formal language, pattern matching techniques are sufficient to correct a range of query variations. Pre-defined ideal solutions will be

dynamically matched against the solution submitted by the student. Any inconsistencies here will be identified using the abstract notation outlined in section 5.5. This is a relatively simple yet powerful correction method that can be implemented within our time constraint, and will lend itself to modification in the future if necessary.

Let us now look at the data flows within the correction component (Figure 5.9) in chronological order, where pattern matching will be expanded upon.

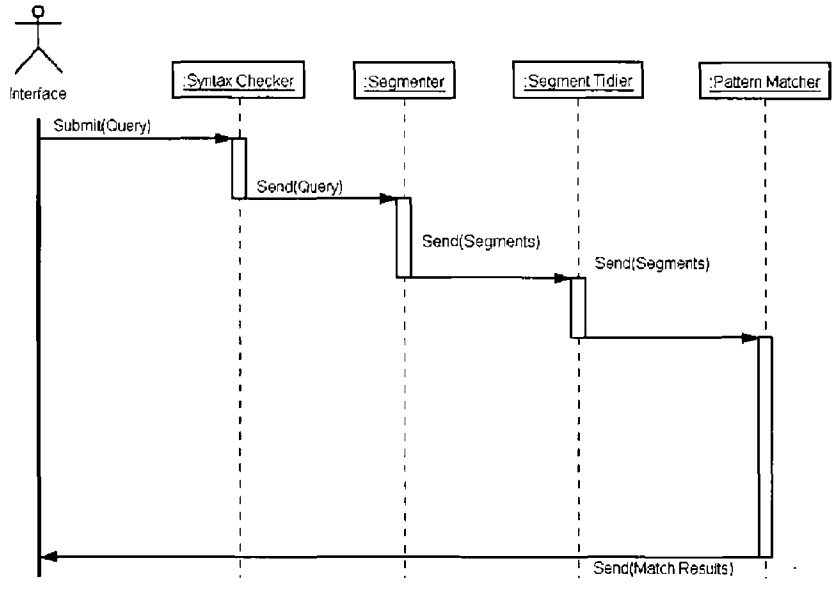

Figure 5.9. Correction Model Sequence Diagram

The student's query is sent to the correction model via the interface. The first of the formal language error types, the syntax error, is checked for immediately by sending the query to actual Oracle database. If the query causes a syntax error the exact text is returned to the interface and displayed to the student. Doing this allows the student to view and work with actual database server error messages. This promotes learning in a realistic setting, which is one of our system requirements (section 4.2.1). There are also allowances to annotate the Oracle syntax error message with suggestions on how to rectify the problem. The query is not corrected any further at this point as we know that the student has made a fundamental error that will need to be rectified.

A syntactically correct query will be matched against an ideal solution for possible semantic errors. Our multi-level error classification scheme uses the notion that the

select statement can be composed of six different clauses. Because of this the correction model sends the student's query to the Segment class. This class splits the query into six strings based on the six clauses. To illustrate this with a trivial example, let us suppose that the student's query is "select section A from section B where section C group by section D having section E order by section F". The segmenter will split this into six separate strings:

1 = select section A

2 = from section B

3 = where section C

4 = group by section D

5 = having section E

6 = order by section F.

The segmenter will also take the ideal solution for particular question and segment it in the same way. All segments are returned to the correction model. There will be at least four segments (both queries require a select and a from clause), with a maximum of twelve segments. Our solution does not treat a nested query as a separate segment. Rather, we view a nested query as an error in the condition section of the query. Errors with SQL elements, however, are treated as separate.

Having been split into segments the student's and the ideal query are tidied for easier matching. This step consists of removing an extraneous white space that may have been introduced to the segments, making sure all operations used are the same in both queries (such as changing != to <>) and standardising certain miscellaneous items (for example, changing the decimal number .9 to 0.90).

When tidied, all segments are then sent to the pattern matching element. Queries are matched segment by segment. This allows the correction model to easily identify which SQL clauses are violated. The ideal solutions have been devised to include a minimal amount of words. Each word or phrase in the ideal solution is necessary. The correction model then works under the assumption that every word in the ideal solution must appear somewhere in the student's proposed solution. For example, for a question that requires the student to find the maximum of some amount, the word "max" must appear. Any extra words that appear in the student's solution may or may not affect the resulting table. This is checked for later. At the moment we are concentrating on making sure that the student includes vital elements in his query.

As mentioned previously, pattern matching is done segment (clause) by segment. Each token in every ideal solution segment is searched for in the corresponding student's segment. If the word is found the pattern matcher moves on to the next word in the ideal solution segment. If the word is not found the segment name and the type of element missing is recorded. The type of element is determined from the solution schema stored in the database.

With this method of correction, the student's query will be marked correct if they have the correct attributes even if the ordering is different from the ideal solution. For example, if in the ideal solution an attribute named *A* is before an attribute *B*, the student will be marked correct if his solutions selects *B* before *A*.

Example:

Question 1: Get distinct colour and city for non-Paris parts with weight greater than 10

Student's solution: `select distinct pname, colour from p where city = 'Paris' and weight > 10`

Ideal solution: `select distinct colour, city from p where city <> 'Paris' and weight > 10`

Solution schema: `select misc att att from table where att comp misc and att comp misc`

These ideal solution SQL statement can also be arranged as follows. Figure 5.9 is based on the SQL grammer outline previously (section 5.5.2).

statement	:: =	select from where
select	:: =	'select' misc att att
	:: =	'select' 'distinct' colour city
from	:: =	'from' table
	:: =	'from' p
where	:: =	'where' att comp misc 'and' att comp misc
	:: =	'where' city <> "Paris" 'and' weight > '10'

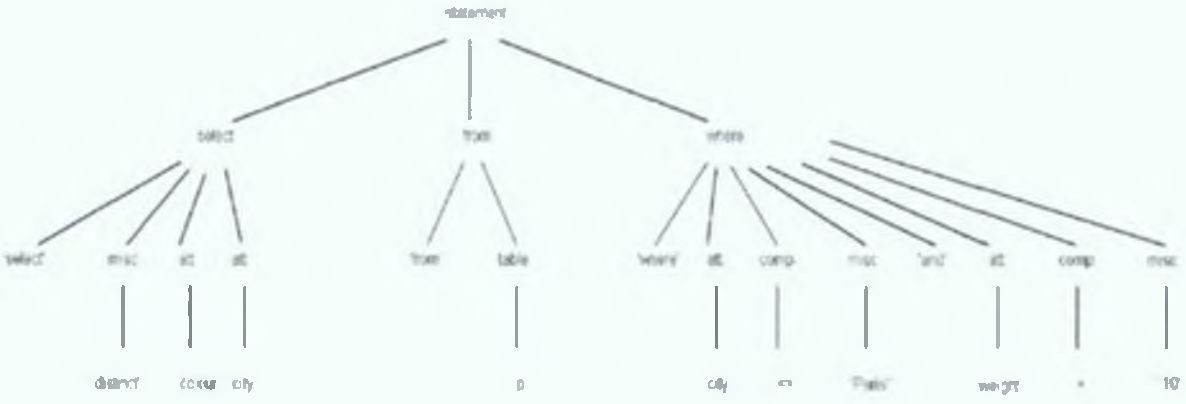

Figure 5.10. Ideal Solution Tree Structure

In this example the student has made two errors. Instead of selecting colour and city, he has made the mistake of selecting colour and pname. He has also specified Paris parts instead of non-Paris parts (i.e. he used the equals-to sign instead of the not-equals-to sign).

The correction model will first send the student's solution to the database to be checked for syntax errors. It is a syntactically correct statement, so it will be corrected further by the correction model. It will first be sent to the segmenter to be split up according to the six SQL clauses. The same will be done for the ideal solution for question one. We will therefore have two sets of three segments to match.

Student's solution:
```
select distinct pname, colour
from p
where city <> 'Paris' and weight < 10
```

Ideal solution:
```
select distinct colour, city
from p
where city <> 'Paris' and weight > 10
```

Solution schema:
```
select misc att att
from table
where att comp misc and att comp misc
```

The pattern matcher will take the student's solution, segment by segment, and match it against the ideal solution. It will first look at the two select segments. The "distinct" keyword is present in both, so the matcher will look at the next word in the ideal solution segment. This word, "colour" is present in both segments. It is in a

different position in the student's solution to the ideal solution, but this does not affect the overall meaning of the query, so the matcher will now look for the next word. This next word, "city" is not found in the student' solution segment; the matcher therefore knows there is an error in the select clause. To determine what type of element is causing the error, the matcher looks at the "solution schema" which is stored in the database for every ideal solution. The matcher notes the position of the element in the ideal solution that is missing from the student's solution. It then finds the corresponding element type in the solution schema. This type is "att", which stands for attribute. So we know that the student has made an error with an attribute in the select clause.

This algorithm is repeated for every clause. In the same way as outlined above the pattern matcher will find an error with a comparison operator in the where clause.

After this the model also checks the output of the two solutions. This acts as verifier for the pattern matcher. It is common, especially for a small number of tables with limited entries in them, for the student to accidentally produce the correct output, even though his actual solution does not meet the requirements send out by the question. For this reason the correction model compares strings as well as the queries' output. Increasing the amount of contents in tables of the same database schema could reduce the accidentally correct outputs produced by the student.

A list of all errors found will be sent to the student model. Our system deals primarily with syntax and semantic errors. It is hoped that students will learn the pragmatics of the language as they use the system.

While SQL is a relatively simple programming language, many variations of a single statement will correctly answer a mid to long length question. The method of correction used here leads to an error diagnosis that is usually accurate, but occasionally the diagnosis is incorrect (such as when the student creates a particularly obscure answer, or due to the variations mentioned in section 5.5). It is very difficult to create a perfect method of correction, an issue that is also raised by the developers of other web-based SQL tutoring systems. Having said this, the correction method could be refined in the future to further reduce the instances of error misdiagnosis.

Now to summarise the tasks carried out by the correction component, in pseudocode, from when the student's solution has been received from the interface:

Algorithm: correct student solution

```
check student solution for syntax errors
if there are syntax errors
      return message to student and exit correction
retrieve ideal solution from the database
segment student and ideal solutions based on six SQL clauses
tidy each segment
pattern match each student solution segment to the corresponding
ideal solution segment
execute student solution and ideal solution in database
compare student solution output with ideal solution output
return error flag and student solution result set to the interface
send any errors found to the student component
```

The correction component design allows, in the future, for the student's statement to be graded. This would allow the tutoring system to be used to return percentage marks to the student and potentially to mark lab exams. 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.

5.8 Student Component

The student model and its accompanying classes are 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 errors, if any, made by the student for each question he attempts. The student component arranges this information in such a manner as to be easily stored in and retrieved from the database in relevant and useful categories. Specifically, this is:

- The total number of attempts for every question. The total number of attempts for each question during the current session is also stored.
- The overall number of errors made by the student for each clause and for each element for the student's entire use of the system, which may be over a number

of days, weeks, or longer. This total is used by the guidance element of the system (section 5.9.2).

- The same type of information as above is stored on a short-term scale, which persists for a single user session. This short-term information also includes, if relevant, a list of elements that the student should have but did not include in the last question he answered incorrectly. This is used for feedback.
- Information such as username, password, and most recent login time

5.9 Pedagogical Component

The pedagogical component contains the teaching strategies of the system, and so we will address both technical and educational issues in this section. This component decides what level of feedback and guidance to return to the student by examining his preferences and the errors he has made in both recent and in all questions he has attempted. It is the personalisation of scaffolding element of the entire tutoring system, used primarily when the student has answered a question incorrectly.

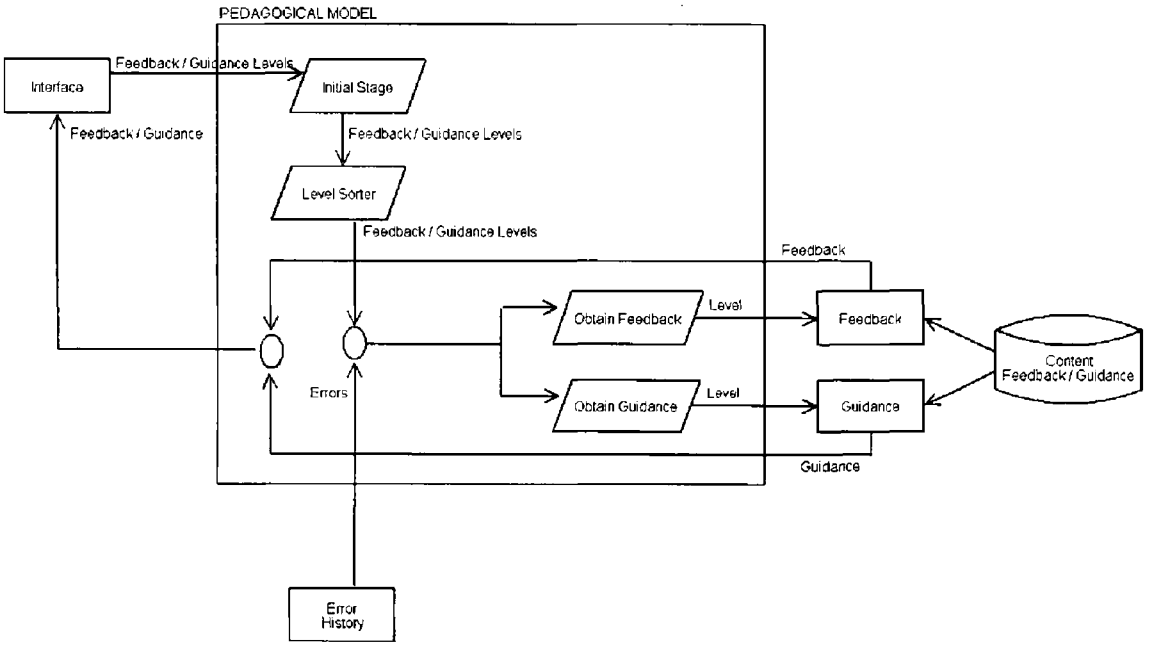

Figure 5.11. Pedagogical Component Zoom

The pedagogical process (Figure 5.11 and Figure 5.12) is initiated by the interface. If the student has made one or more semantic 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 viewing of this scaffolding is optional – at this stage of learning the apprenticeship theory, which we incorporate, allows the student to dictate when he needs high-level scaffolding. If the student has indicated that he wishes to receive neither feedback nor guidance the pedagogical model has no further action to take and the student can attempt another question. If, on the other hand, he has expressed a wish to be given feedback or guidance the pedagogical model will sort these levels to determine what further action needs to be taken. This sorting of levels is particularly important if the student has indicated that the system can decide what degree of feedback and guidance should be displayed. This is the default option. If this is the case, the system will gradually increase the levels from minimum to maximum as the student repeatedly unsuccessfully attempts a question. The scaffolding will plateau at the maximum levels possible for a further number of attempts. This increase of feedback and guidance is a form of scaffolding, and so it should be faded. Consequently, after the plateau period, both forms of scaffolding will be gradually decreased from a maximum level to a minimum level if the student continues to submit solutions to be corrected. The student can choose, via the configuration page, to receive maximum levels of feedback and guidance after the system has automatically faded this scaffolding.

Algorithm:
```
let the system decide scaffolding levels
get question attempt number from student component
if attempt number is low
      set scaffolding levels to low
get feedback and get guidance
gradually increase scaffolding levels while continuing to get
feedback and get guidance
keep scaffolding levels static for a small number of further attempts
while continuing to get feedback and get guidance
gradually decrease scaffolding levels while continuing to get
feedback and get guidance
```

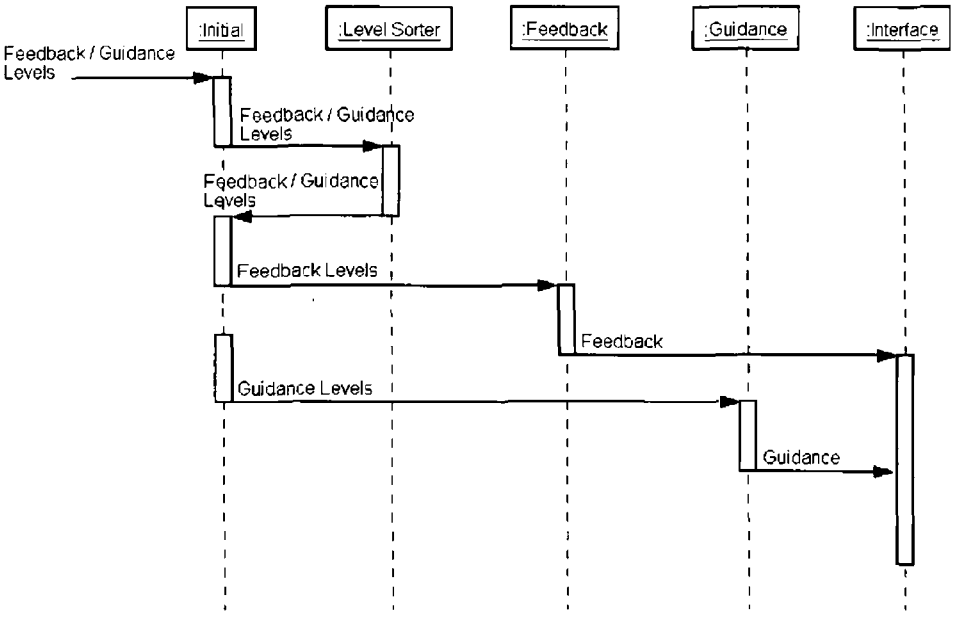

Figure 5.12. Pedagogical Model Sequence Diagram

Based on the levels chosen by the student, the student will obtain an appropriate type of feedback or guidance. These are forms of scaffolding, therefore the student can choose not to view either type. The provision of such scaffolding is in keeping with our guided discovery navigation as, through this, the student can control responses to exercises. There are a range of levels of each to allow the gradual increase and decrease of this scaffolding, in accordance with pedagogical principles. These levels have been devised in such a manner as to allow small but relevant increments and decrements of both types of scaffolding. There is one more level of feedback than guidance available as feedback needs to be quite precise whereas the nature of guidance is to be more general. Let us look at feedback and guidance in more detail.

5.9.1 Feedback

Feedback can be defined as "information given in response to a product, performance etc. used as a basis for improvement" [Soanes and Hawker, 2005]. In the educational context in question here we redefine feedback as immediate (or synchronous) advice given to a student, based on a solution he has submitted, that aims to aid him in his learning experience. In our system, it is an umbrella term for hints about mistakes the student has made along with links to relevant knowledge and

lecture material. Feedback is specific to a single question. The system offers delayed feedback, thus the student must type and submit before hints etc. are displayed.

There are five levels of feedback. The student may choose, on the configuration page, any level from zero (no feedback) to five (maximum feedback). When the student answers a question incorrectly the pedagogical component assess the level of feedback he requires, which is then obtained from the database. Hint strings and links are prewritten in such a manner as to make pedagogical sense when they are combined. They can also be easily changed as the system evolves. Each level is an extension of the previous level. The fundamental aim of feedback is that the student will examine the hint, reassess the requirements of question, and then re-formulate his solution in the context of both the question and the given hint. While the hints aim to be helpful they are not so specific as to give the complete answer to the student.

Let us use the same example as was used to illustrate the correction component; the student therefore has made two errors – the first is with selecting an attribute in the select clause, the second with using a comparison operator in the where clause. We will provide some references to visual examples of feedback (section 5.10.4) throughout the following text.

Level 1: The system retrieves a hint based on the first SQL clause that has been violated. If errors occurred in other clauses they will be ignored for the present time. This is to minimise the amount of feedback the student receives. The hint corresponding to the violated clause is taken directly from the database by the feedback class. In the above example, the student would be told he has made a mistake in the select clause. See also Figure 5.20 for a separate example taken directly from the system.

Level 2: The student is alerted to one clause and one element in that clause that is causing problems. In our example this is the select clause and an attribute. The clause hint is selected as in level one. Secondly a hint specific to the incorrect element (in this case the attribute) is taken from the database and appended to the clause hint. These hints have been written in such a manner as to form a sensible hint when added together. See Figure 5.21.

Level 3: Feedback at level three is similar to level two in that in combines hints for both the erroneous clause and the erroneous element. The previous levels dealt with one clause and one element only. This level gives the student hints about all clause and elements that are causing problems. In our example the student will be told that

there were errors in the select clause and the where clause, with selecting an attribute and with using a comparison operator. See Figure 5.22.

Level 4: A level four hint should be considerably detailed, therefore we extend the level three hint further. Specific details in the ideal solution not included in the student solution are saved, via the student model, by the correction model. This information is utilised by the feedback component. As stated before, every element in the ideal solution should appear somewhere in the relevant clauses in the student's solution. The level four hint will display all of the missing (and hence error-causing) details to the student. In our example, this means that the student will be shown the two elements, "colour" and "<>", that need to be included. It is up to the student to discover where they should be placed. This level of hint is similar in ways to a partial solution. More comprehensive partial solutions run the risk of not matching the structure of the student's solution, consequently acting as more of a hindrance than a help. This level of hint offers very specific advice to the student, but in a manner that will not confuse him. See Figure 5.23.

Level 5: This maximum level of feedback adds to the previous level by including a set of links for the student to click on. When clicked, these links open a new window composed of knowledge from the courseware. The information here is particularly relevant to the question the student is working on. The links are stored in the database, so they can easily be changed as the courseware is updated. See Figures 5.24 and 5.25.

Feedback decision procedure

(Formal language property errors + SQL clause errors + SQL element errors) made by student = error diagnosis

Error diagnosis → Feedback

↑

Student's preferred feedback level

Error diagnosis + Level 1 = One SQL clause hint

Error diagnosis + Level 2 = One SQL clause hint + One SQL element hint

Error diagnosis + Level 3 = All SQL clause hints + All SQL element hints

Error diagnosis + Level 4 = All SQL clause hints + All SQL element hints + Required elements

Error diagnosis + Level 5 = All SQL clause hints + All SQL element hints + Required elements + Relevant links

Algorithm: `get feedback`

```
retrieve student's local error history from student component
retrieve student's feedback level from student component
send local error history and feedback level to database
retrieve appropriate hint and links from database
```

5.9.2 Guidance

Guidance is "advice or information" [Soanes and Hawker, 2005]. We refine the term, to suit our educational context, to mean offering the student advice and recommendations about subject areas to concentrate on, based on a previous determination. This determination may be a single session, but typically it is data collected over a number of sessions over a period of time. Our system will use the student records to offer personalised guidance to the student based on their total actions while using the system. This is a form of knowledge-based recommendation as the student's need is matched up with a particular recommendation. Guidance such as this is absent in other online SQL tutoring systems.

The virtual apprenticeship model and the guided discovery format, both of which we are incorporating into this system, assume that the student will take a degree of responsibility for his learning experience. Many forms of scaffolding are optional and the onus is on the student to, firstly, use them when he needs to and, secondly, embrace the opportunity to learn. Guidance suggests areas that the student should relearn or pay particular attention to. It is the student's decision whether he wishes to do so or not.

The pedagogical model retrieves the information on all errors made by the student. This is then arranged in a manner that can be comprehended by the guidance feature. There are four levels of guidance. As with the feedback feature, each level builds on the information retrieved and displayed for the previous level. As before, we will provide some references to visual examples of feedback (section 5.10.5) throughout the following text.

Level 1: The pedagogical model selects the clause with the largest total number of errors. The guidance feature then retrieves the relevant pre-stored guidance string from the database for display on the interface. This string tells the student the clause with which he is having the most difficulty. See also Figure 5.26 for a separate example taken directly from the system.

Level 2: The pedagogical model informs the guidance feature about the clause and the element that are causing the most problems for the student. The relevant strings are then retrieved from the database and joined together. As with hints, they have been written in such a manner as to make them understandable when they are read together. The student is then told of both the clause and the element that he has made the most errors using. See Figure 5.27.

Level 3: The third level of guidance differs in ways from the first two. This level alerts the user to all clause and all elements with which he has made an error. This is graded based on the amount of errors – small, moderate and large. The student is informed of this grade. See Figure 5.28.

Level 4: The first section of level four guidance is the same as level three. A clickable button is added at this stage. When the student clicks this, a new window is opened containing an index of questions he should attempt. This index is dynamically created based on an assessment of the student's work up to this point. The number of further questions to attempt is weighted based on the number of errors the student has made in each error category. For example, if the student made a large amount of mistakes using aggregate functions he will be presented with, say, six questions to try that all require the specific use of aggregate functions. If, on the other hand, he has made a small number of errors with comps he will be given only one or two questions to attempt that require the specific use of comps. Obviously there will be some overlap here; while practising the use of aggregate functions the student could also be required to practise selecting attributes etc. These questions are a mixture of the original set and an extra set (outlined earlier). All are based on the original three tables to avoid confusing the student. See Figure 5.29.

When the student selects one of these further questions to attempt he is brought to the same interface as before. His feedback and guidance levels persist here, and so he is shown hints, links and guidance. The highest level of guidance available here is level three, as technically he is working with an "open" level four. Level three guidance will therefore be displayed in the place of level four.

Guidance decision procedure:

(Formal language property errors + SQL clause errors + SQL element errors) made by student = error diagnosis

Error diagnosis → Guidance

↑

Student's preferred guidance level

Error diagnosis + Level 1 = One SQL clause text string

Error diagnosis + Level 2 = One SQL clause text string + one SQL element string

Error diagnosis + Level 3 = All SQL clause text strings (graded) + All SQL element text strings (graded)

Error diagnosis + Level 4 = All SQL clause text strings (graded) + All SQL element text strings (graded) + Menu of relevant questions to practise

Algorithm: `get guidance`

```
retrieve student's global error history from student component
retrieve student's guidance level from student component
send global error history and guidance level to database
retrieve appropriate text and further practise questions from
database
```

5.10 Student-System Interaction

We have examined the system from the design perspective. Let us continue by looking at the content and layout of the Web pages the student will visit, and the dialogue (Figure 5.13) between these pages. We will present this from the student's perspective by outlining the steps the student can take in a typical session, which we will illustrate with screen shots.

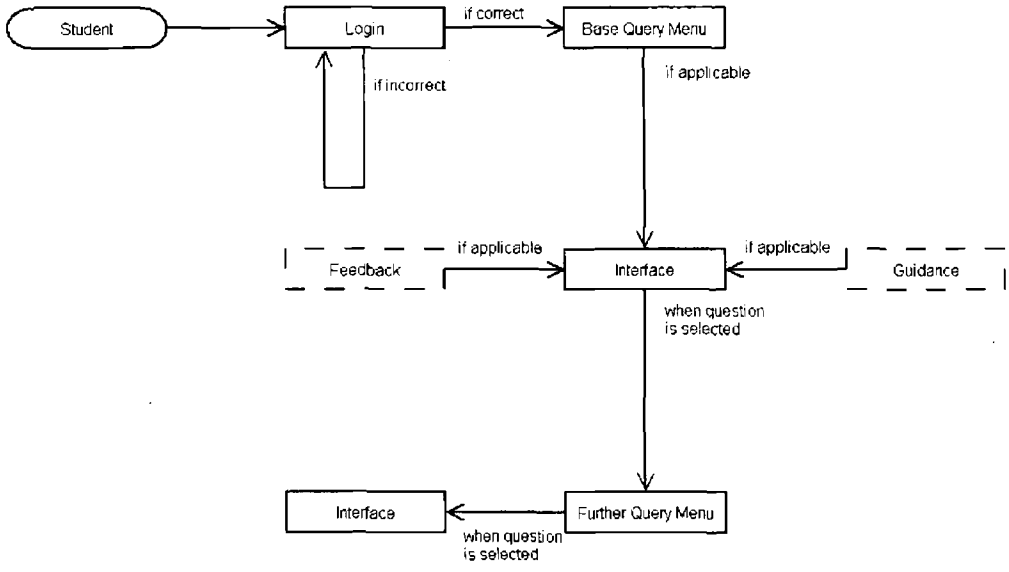

Figure 5.13. Inter-Web Page Design

Internal System Component

Web Page

Information Flow

5.10.1 Login and Query Menu

The student logs on to the system using a username and password that has been assigned to them. He is then brought to a menu of questions to attempt (Figure 5.14). This is in guided tutorial format. The student can choose the questions he wishes to attempt, and the order in which he wants to do them. He is helped by the title of the question, e.g. aggregate functions, string comparison, etc. Also the student may attempt a question any many of times. The menu format allows the student to

- Control the questions he attempts – this will give him a sense of control over his learning experience
- Repeat any questions he finds he needs to re-practise
- Challenge himself by attempting a range of questions on various select statement topics.

Figure 5.14. Section of Menu of Questions

5.10.2 Configuration Page

There are various levels of feedback and guidance (both are forms of scaffolding) available, as described below. When student accounts for the tutor are created each student is assigned a default level of feedback and guidance that lets the system decide the levels the student should receive. The student may change these settings at any time through the configuration page (Figure 5.15), which is linked to from the bottom of the query menu. They could choose from no hints to level five of hints, and from no guidance to level four of guidance. Any combination of this is permitted. They can revert back to the default at any stage.

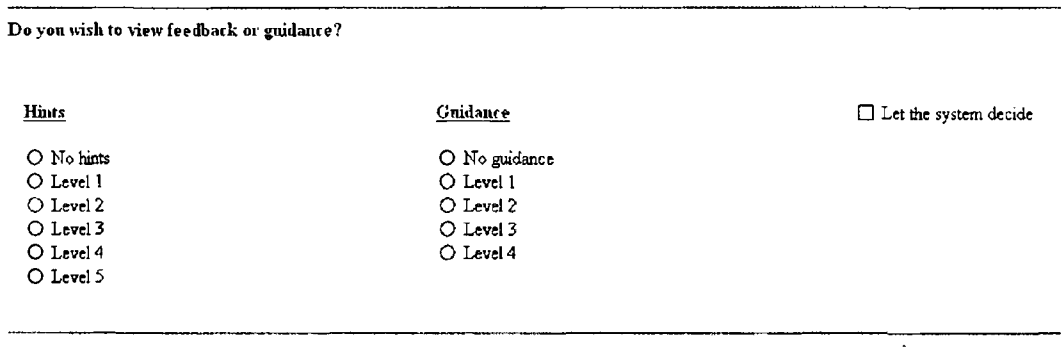

Figure 5.15. Section of Configuration Page

The default gradually increases and then decreases the feedback and guidance given for each question. Both begin at level one, and are increased to the higher levels should the student continue to answer the question incorrectly. There is a low level to start to encourage to student to think out the answer with as little prompting as possible. This is increased after every submission - if the student needs to submit a number of times it is obvious that he is having difficulty with the question. After a set number of submissions the feedback and guidance offered is gradually decreased, because scaffolding should be faded.

5.10.3 Query Interface

There is a common interface for every question (Figure 5.16).

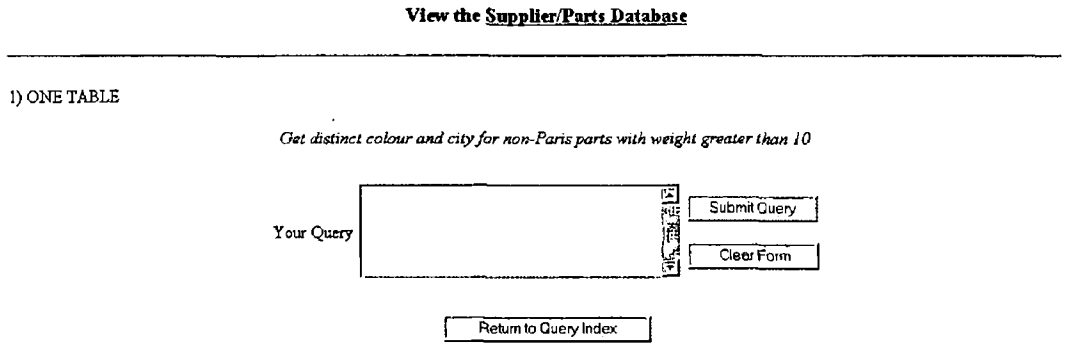

Figure 5.16. Section of Question Interface

There is a link at the top of the interface through which the student can view the tables that he should work with (Figure 5.17).

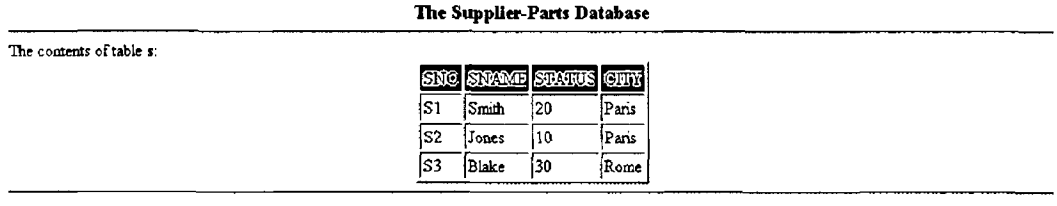

Figure 5.17. Section of Page with Tables to be Queried

The type of problem and the question, in natural English language, are clearly printed towards the top of the interface (Figure 5.18). There is a text section in which the student should type his proposed solution. He can then submit this to be corrected by pressing the appropriate button. An error flag is then returned (correct or incorrect) as well as the result of the query in table form (Figure 5.19). If a syntax error is made a genuine Oracle error message will be returned (Figure 5.20). This is compatible our realistic learning environment.

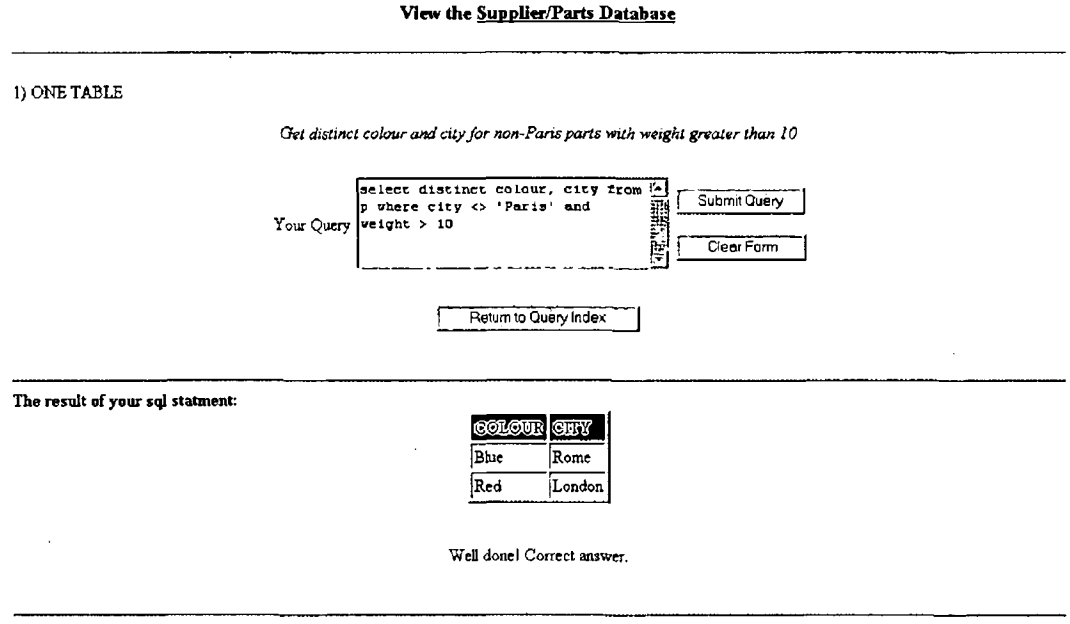

Figure 5.18. Section of Interface with Correct Answer

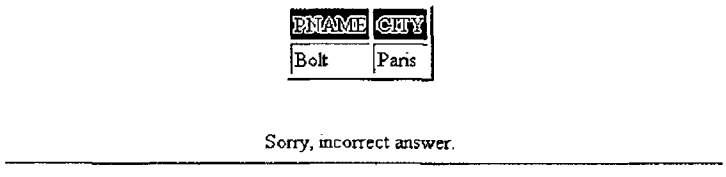

Figure 5.19. Incorrect Answer

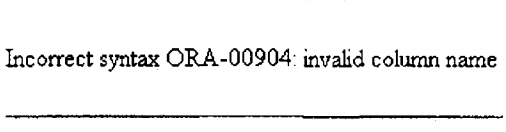

Figure 5.20. Syntax Error Message

The student can submit answers as many times as he wants. Two forms of scaffolding will be made available after a certain number of submissions (Figure 5.21); the first allows the student to see the output of the correct solution (i.e. a table), the second allows him to view the correct solution in pictorial and text format. This scaffolding is optional – at this stage of learning the apprentice is advised by the master when he needs it.

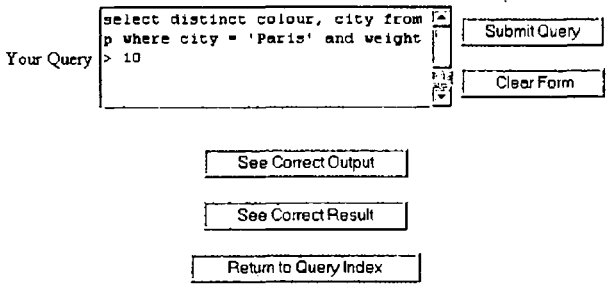

Figure 5.21. Correct Output and Correct Result Scaffolding

If the student answers the question incorrectly he may view hints, depending on his configuration choices. These hints are of varying levels.

5.10.4 Feedback

In question one, shown below, the correct answer is *select distinct colour, city from p where city <> 'Paris' and weight > 10.* This selects distinct rows of colours and cities from the table p for cities that are not Paris and where weight is greater than 10.

In the examples below, the student has made two errors. The first is in the select clause where he has selected the attribute pname instead of colour. The second is in the where clause with a comparison operator as he has used the equals-to sign instead of the non-equals-to sign.

A level one hint alerts the student to one clause that is causing a problem. In the example about the student has made a mistake in two clauses. A level one hint however, only displays on clause. In Figure 5.22 we see that only the select clause is mentioned in the hint.

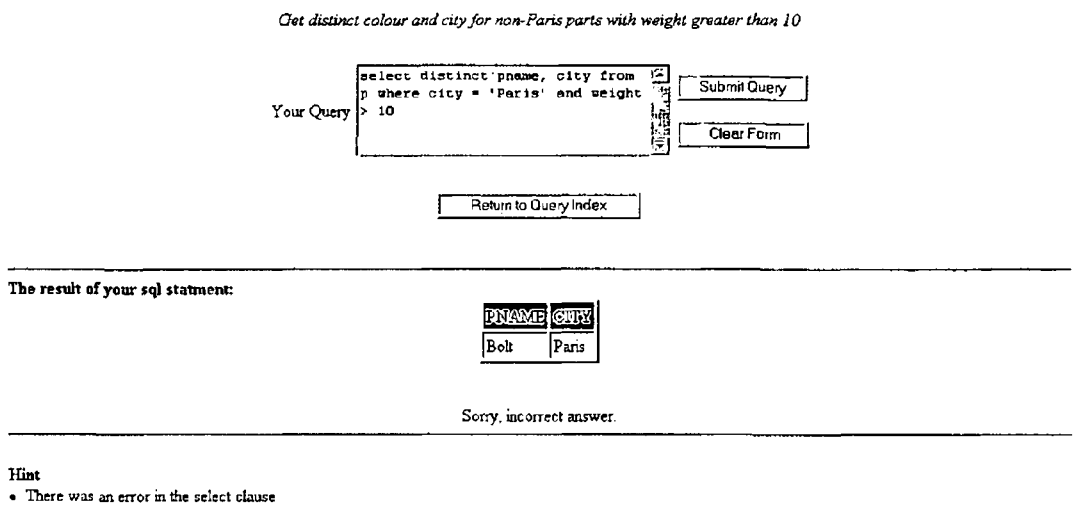

Figure 5.22. Level One Hint

A level two hint elaborates on the error mentioned by a level one hint. While the hint still concentrates on one clause, it mentions the element that is in error. In the example we are using the student has used an incorrect attribute name in the select clause. The hint in Figure 5.23 states this.

Hint
- There was an error in the select clause
- There was an error with an attribute that was selected

Figure 5.23. Level Two Hint

A level three hint is similar to the level two hint, except that this time it alerts the user to all errors they have made. In our example the student has selected an incorrect attribute in the select clause and has used the wrong symbol in the where clause. Both of these errors are indicated in a level three hint (see Figure 5.24).

Hint
- There was an error in the select clause
- There was an error in the where clause
- There was an error with an attribute that was selected
- There was an error with a symbol that was used

Figure 5.24. Level Three Hint

A hint at level four augments on the previous by explicitly stating the elements that the student needs to include. In our example he needs to include the attribute colour and the not-equals-to symbol (Figure 5.25). This hint is similar in ways to a partial solution, except it does not cause confusion by giving the student a complete clause that may not fit in with their own solution.

Hint

- There was an error in the select clause
- There was an error in the where clause
- There was an error with an attribute that was selected
- There was an error with a symbol that was used
- Are you including the following elements? colour ... <> ...

Figure 5.25. Level Four Hint

The final hint level is level five. This hint is the same as the previous with the addition of links the student might benefit from viewing (Figure 5.26). These links are directly connected to the problem the student is attempting. The link is opened in a new window so that the student can easily return to the tutorial (Figure 5.27).

Hint

- There was an error in the select clause
- There was an error in the where clause
- There was an error with an attribute that was selected
- There was an error with a symbol that was used
- Are you including the following elements? colour ... <> ...
- Try the following links - 1 - 2 - 3

Figure 5.26. Level Five Hint

Figure 5.27. Level Five Hint and Link

5.10.5 Guidance

The student can view guidance, at the level he has previously chosen, when he answers a question incorrectly. As with hints, the various levels contain a differing amount of detail. Guidance is based on the overall performance of the student – all errors for all questions are taken into account. Level one tells the student with which of the six clauses he is having the most difficulty (Figure 5.28).

Figure 5.28. Level One Guidance

Level two guidance states, along with the clause the student is having the most trouble with, the element that he has made the most amount of errors using (Figure 5.29).

Guidance

- Your greatest amount of errors involves using the where clause and using symbols

Figure 5.29. Level Two Guidance

Should the student choose level three guidance he will be told about all clauses and elements with which he is having difficulty. These difficulties are graded as small, moderate or large (Figure 5.30). The student can immediately see the SQL constructs that are causing him more trouble than others, and can use this information to revise appropriate course notes or redo relevant questions in the tutorial.

Guidance

- You are having a large amount of problems using the select clause
- You are having a large amount of problems using the where clause
- You are having a small amount of problems using symbols

Figure 5.30. Level Three Guidance

Level four, the highest level of guidance available, is based on the previous level (Figure 5.31). Now, by clicking a button, there is a menu of further questions that the student can attempt (Figure 5.32). These are based on the same tables and questions as before, but are different questions to what has gone before. This menu is dynamically created based on the student model. All questions are weighted. So, if the student has been having a large amount of problems with aggregate functions the new menu will contain a considerable amount of questions that involve the use of aggregate functions. If he is having a small number of difficulties with comparison operators there will be a small number of further questions that require their use. This extra menu is opened in a new window so that the student can easily return to the main tutorial. All further questions are corrected as before, and feedback and guidance will still be displayed according to the student's preferences.

Guidance

- You are having a large amount of problems using the select clause
- You are having a large amount of problems using the where clause
- You are having a small amount of problems using symbols
- Try the following questions

Go!

Figure 5.31. Level Four Guidance

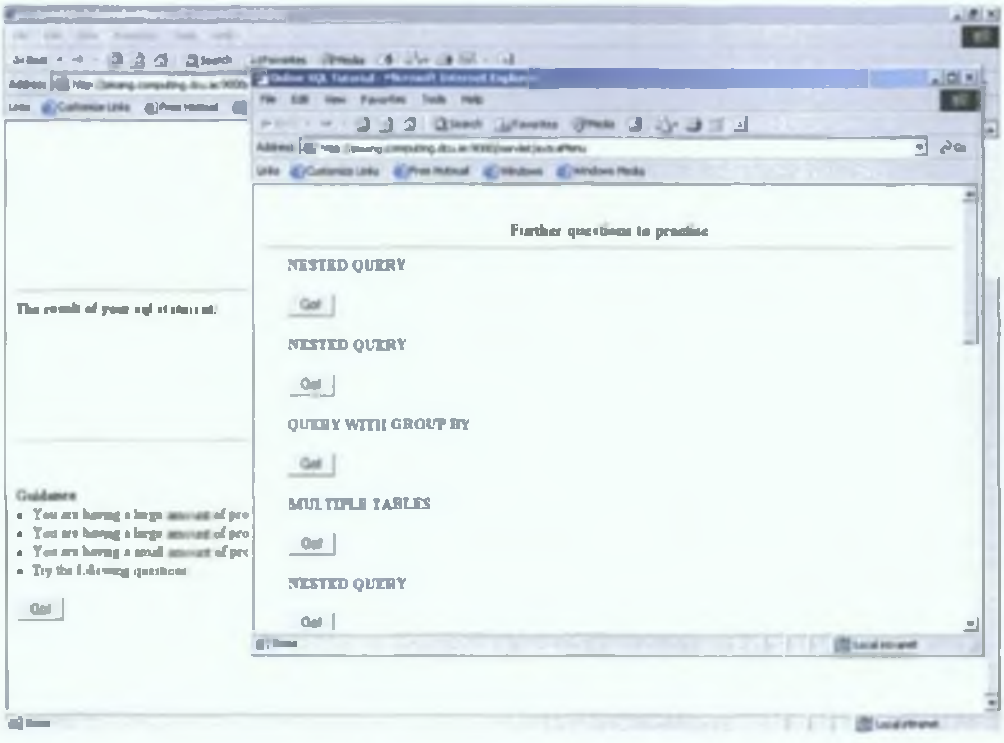

Figure 5.32. Menu of Further Questions to Practise

5.11 Chapter Summary

This chapter described the design of the automated SQL tutoring system. We first described the system's architecture. We designed our architecture using both UML diagrams and flow-charts, during which time we found the LTSA (section 2.2.1) to be a very useful starting point for the design of our online learning system. The world of computing is still evolving, so an architecture should be modular, flexible and updateable.

The result of the architecture design process was an architecture composed of four key components – interface component, correction component, student component

and pedagogical component - which are supported by backend storage structures in a database.

The interface design is similar in many ways to that of the original SQL execution environment. It acts as the chief point of interaction between the student and the system. The student reads questions, submits solutions, and views scaffolding through it without having to be aware of the inner workings of the system. Our interface allowed the student to engage in a guided discovery format by allowing him to control the pace and timing of his learning experience, the path he took through the curriculum, and the scaffolding he viewed.

The next stage of our project involved defining an error classification technique. We use a multi-level error classification scheme so as to closely pinpoint the type of error the student has made. This scheme places each error in all of three categories, which are based on:

- Formal language properties. SQL is a formal language, thus its errors may be syntactical, semantic or pragmatic
- SQL clauses. There are six main SQL clauses, each of which serves a specific purpose. Errors can occur in any of these clauses.
- SQL elements. A typical select statement will be composed of a number of elements e.g. table names, attributes, comps, etc. The student may use any of these incorrectly.

The correction component will make a compound diagnosis of an error using all three categories, as specified by our original objectives (section 4.2.1.2).

Our correction component was implemented based on a pattern matching algorithm. It first obtains both the student solution and the ideal solutions and divides them into segments that correspond to the six SQL clauses. It then matches each student solution segment to the relevant ideal solution segment. Through this the component can flag the student's solution as being correct or incorrect, which is displayed on the interface. The output of both solutions is compared as a secondary correction feature. Any errors in the student's proposed solution are grouped according to our multi-level error classification scheme before being sent to the student component. Although SQL is a relatively small-scale language, there are typically a number of variations of a select statement that correctly answer a given question. It is difficult to develop a perfectly accurate method of correction, but we

feel this can be achieved by carefully identifying the SQL statement variations that exist. Our correction method can be extended in time to allow for a wider range of such variations.

The student component stores information regarding the student's errors, which is primarily used by the pedagogical component.

The pedagogical model makes decisions about teaching strategies that the system will use. Its main function is to choose the content of scaffolding that should be displayed for the user. It does this by examining the type of errors the student has made and then adjusting the amount of scaffolding based on the student's preferences. Specifically, scaffolding is feedback and guidance. At its highest level feedback is a set of hints specific to the errors made by the student along with a selection of links relevant to these errors. Guidance is individual assistance formed based on the student's performance within the system. At the highest level it consists of helpful text and a personalised menu of questions the student might benefit from practising. Because we are using a guided discovery method of navigation, the student can control the levels of scaffolding displayed. He also has the choice of switching off the scaffolding feature or allowing the system to set the levels. If the latter of these is chosen, the system dynamically increases and decreases scaffolding based on the amount of attempts the student is making for each question.

Towards the end of this chapter we outlined the system from the student's perspective by taking a step-by-step walk through the features offered. Screen shots were included for clarity.

Chapter 6
Evaluation

6.1 Introduction and Motivation

Any design model, including the one we are using (Figure 5.1), includes an evaluation stage. Our evaluation will be guided by the DECIDE generic evaluation framework [Preece et al., 2004]:

- Determine the overall *goals* that the evaluation addresses
- Explore the specific *questions* to be answered
- Choose the *evaluation paradigm* and *techniques* to answer the questions
- Identify the *practical issues* that must be addressed, such as selecting participants
- Decide how to deal with *ethical issues*
- Evaluate, interpret, and present the *data*

Determine the overall goals that the evaluation addresses

We wish to establish if SQL tutoring system has been successful in aiding the student as they learn how to create SQL select statements. Success is based on three factors, outlined in section 6.1.2. We also want to determine if the system has met the requirements we set down in the fourth chapter. This information can be used in the system to improve the system.

Explore the specific questions to be answered

Specifically, we want to assess the academic benefit of the system to the student, that is, are the students more able to use the select statement having used the system. Parallel to this, we want to see if the students enjoyed their learning experience: if they personally found the system to be of benefit and if they enjoyed using it.

Finally, we want to determine if the system fulfils the implementation requirements we laid out.

Choose the evaluation paradigm and techniques to answer the questions

We will use a summative form of evaluation; one that is carried out when the project has been completed. We aim to evaluate the system in three ways:

(1) By studying student performance and behaviour, using exam results and data mining

(2) By discovering students' opinion of the system

(3) By using our own evaluation goals to assess the system

The first method will allow us to assess the effectiveness of the system, with the second we will evaluate the system's usability, and the third method will allow us to determine the levels of effectiveness and usability.

Identify the practical issues that must be addressed, such as selecting participants

All students were given a username and password for accessing the system. Had we used a test group, say half the class, we might have clearly seen if the tutoring system had an impact on the students' performance. This, however, would have been somewhat unethical – all students have to take the same examinations. Students may feel unhappy if certain tools were restricted to them, or indeed imposed on them. Thus, we will be assessing the performance of all students and the behaviour of a small sample.

To assess student opinion, questionnaires will be distributed to the students in a lecture within two weeks of the mid-semester SQL examination. The completion of the questionnaire will be optional.

Lastly, we will examine our original system requirements and the final implementation of the tool to determine if we have met said goals.

Decide how to deal with ethical issues

As mentioned in 6.1.4, use of the system will be available to all students, so as to avoid any type of discrimination. Because the topic will be examined, use of the system will be optional – students can then study in the manner that suits them best. Finally, students will not be penalised for not completing the questionnaire as it will be handed out in a lecture that all students may not attend.

Evaluate, interpret, and present the data

The remainder of this chapter presents the results of our evaluation.

6.2 Evaluation Methods

As previously mentioned, the automated tutoring system will be evaluated in four ways, three of which are student-based:

6.2.1 Student Performance

Student performance is a substantial part of any degree programme. One of the hopes for the SQL tutoring system was that it helped students to learn how to create SQL select statements, which would then be reflected in exam results. We will therefore compare the results obtained by students this year compared to those of previous years to ascertain if there has been an improvement in student grades.

6.2.2 Student Opinion

Survey methods can be used to address student motivation and acceptance [Pahl, 2003b]. By combining open and closed questions we can establish a multi-dimensional view the students have of the system.

6.2.3 Student Behaviour

Student behaviour can be obtained by examining web access logs, which store details regarding the student's usage of the system. We will use this information to validate the student performance results and to complement our survey data.

6.2.4 Fulfilment of Objectives and Requirements

In chapter four we set out a list of objectives to meet within a set of requirements. We will carry out a separate evaluation of the system based on how many and to what extent these objectives and requirements were met.

6.3 Evaluation Results

6.3.1 Student Performance

The system was made available to students on a voluntary basis in the weeks preceding an in class SQL select statement exam. This exam, marked out of twenty five, forms 12.5% of the overall module assessment. The standard of the exam has remained constant over the last number of years.

Over the course of the previous four years there has been a 2% increase per annum in the marks obtained by students in SQL exams [Pahl et al., 2004]. The original SQL execution environment had been introduced and gradually improved during this time; other factors have remained constant. Our system is a considerable extension of what existed previously. The original system, for example, did not have hints for semantic errors, or guidance.

The average mid-semester exam result obtained by last year's class was 56.9%. The average mark this year was 62.3%. This is an increase in one year of 5.4%, a jump of over 3% of that which was expected.

6.3.2 Student Opinion

We devised a comprehensive questionnaire of open and closed questions to gauge the students' thoughts on the SQL tutoring system. It uses various surveying techniques [Fink, 1995] to determine the students' experience of using system. Points in a general checklist [Preece et al., 2002] were followed:

- Provide clear instructions on how to complete the questionnaire
- Make questions clear and specific
- Ask closed questions with a list of option, when possible and appropriate
- Consider a "don't know" option for relevant questions
- When using scales, check the range is appropriate and does not overlap

For example, the questionnaire included asking students if they agreed or disagreed with a series of statements relating to the usability of the tutoring system (6.3.2.2), the tutoring system's course content and material (6.3.2.3), and their learning experience (6.3.2.4) (see Table 6.4 for all results). They were prompted to tick one box for each

question. The possible answers were strongly agree, agree, undecided, disagree, or strongly disagree. This type of scale is called a Likert scale, and is commonly used in questionnaires. Other questions involved students rating various given options, choosing a reply from a gradual scale, and answering questions in their own words.
The questionnaire was distributed to students that were present for a CA218 lecture, shortly after the mid-semester SQL exam had taken place.

6.3.2.1 Demographic Information

Thirty six students completed the optional questionnaire. The majority of respondents were male and from the CASE (Computer Applications Software Engineering) stream. Approximately a quarter were in the CAIS (Computer Applications Information Systems) stream and close to ten percent in total were female. This demographic information (Table 6.1 and Table 6.2) is representative of the make-up of the entire class.

Programme

NAME	PERCENTAGE
CASE	68.6
CAIS	25.7
Other	5.7
TOTAL	100

Table 6.1. Demographic Information

Gender

GENDER	PERCENTAGE
Male	91.7
Female	8.3
TOTAL	100

Table 6.2. Demographic Information

The majority (47.2%) of students had no experience (i.e. level 1) of using an online tutoring system, apart from those that are part of other modules in this degree course. No student claimed to have the highest level of experience with such a system. See Table 6.3 for the full list of results for this question. The total percentage for this table is 100%.

LEVEL	PERCENTAGE
Level 1	47.2

Level 2	13.8
Level 3	5.6
Level 4	8.3
Level 5	5.6
Level 6	5.6
Level 7	11.1
Level 8	2.8
Level 9	0
Level 10	0

Table 6.3. Online Tutoring Experience

The vast majority (94.4%) of those surveyed stated that they did not use any other SQL tutoring system over the course of the semester.

All results are rounded to the first decimal place.

QUESTION NUMBER	STRONGLY AGREE	AGREE	UNDECIDED	DISAGREE	STRONGLY DISAGREE
1	13.9	61.1	16.7	8.3	0
2	13.9	50	22.2	11.1	2.8
3	8.6	31.4	40	20	0
4	0	44.4	47.3	8.3	0
5	2.8	69.4	25	2.8	0
6	8.3	75	13.9	2.8	0
7	2.8	47.2	25	22.2	2.8
8	11.1	52.8	16.7	19.4	0
9	25	61.1	8.3	5.6	0
10	5.7	25.7	31.5	31.5	5.6
11	17.2	74.3	5.7	0	2.8
12	20	42.9	28.6	8.5	0

Table 6.4. Combined Results of Likert Scale Questions

6.3.2.2 Usability of the SQL Tutoring System

Question 1: The system is easy to use (Figure 6.1)

A large majority of students, 75%, agreed or strongly agreed that the system is easy to use. Close to 17% of those surveyed were undecided, which could also mean they neither agreed nor disagreed with the statement. A small percentage were of the opinion that they system is not easy to use.

Question 2: The system is easy to navigate (index of queries, configuration page, guidance links, etc.) (Figure 6.2)

When surveyed, 63.9% of students agreed or strongly agreed that the system is easy to navigate, which is a definite majority. Approximately 22% were undecided, while the remainder felt that the system was not easy to navigate.

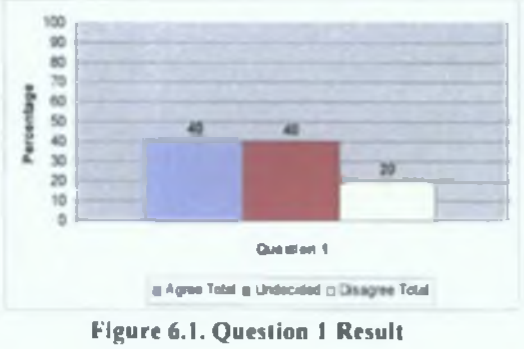

Figure 6.1. Question 1 Result

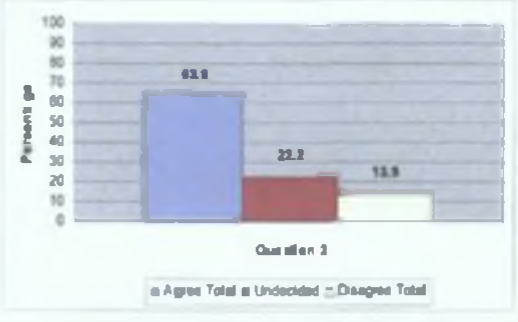

Figure 6.2. Question 2 Result

Question 3: The feedback element (hints) is useful (Figure 6.3)

The feedback feature is an important part of the system. The survey results regarding its usefulness vary. 40% agreed (31.4%) or strongly agreed (8.6%) that it is useful. The same amount was undecided. 20% did not think it was useful, although no one strongly disagreed with the statement. While a significant proportion of those surveyed were neutral towards the feedback component, the percentage that found it useful was double that who did not. This result is of particular importance to our work, and is discussed in section 6.4.1.2.1.

Question 4: The guidance element is useful (Figure 6.4)

A personalised guidance feature based on a non-short-term assessment is a unique part of our online tutoring system. Over two fifths of students surveyed agreed that this personalised guidance is useful. Slightly more than this amount was undecided. Approximately 8% did not think this feature was useful. As with question 3, above, this result has particular importance for our work, and is discussed in section 6.4.1.2.1.

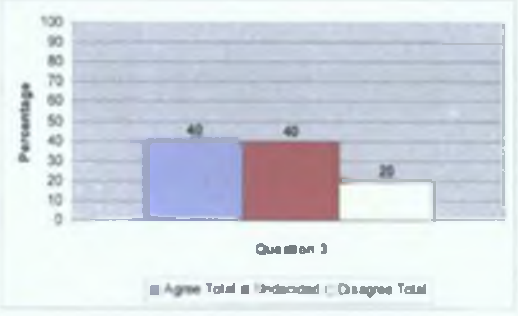

Figure 6.3. Question 3 Result

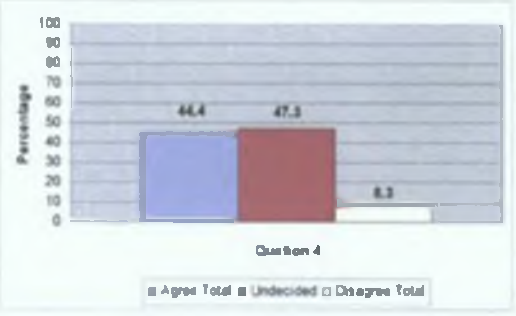

Figure 6.4. Question 4 Result

Question 5: Having various levels of feedback and guidance is useful (Figure 6.5)

The students were given the freedom to choose the levels of feedback and guidance they were given. Alternatively they could let the system decide the levels, which would be adapted according to the number of attempts the student has made for a particular question. A considerable majority of 72.2% agreed or strongly agreed that various levels of feedback and guidance are useful. A quarter of those surveyed were undecided, while one person disagreed, but not strongly.

6.3.2.3 Course Content and Material

Question 6: The system sufficiently gives me a broad overview of SQL select statements (Figure 6.6)

The questions around which the tutoring system is based is small but covers a wide range of SQL select statement features. Of those surveyed, in excess of four fifths agreed or strongly agreed that the system offered a broad overview of select statements. Almost 3% surveyed disagreed; none disagreed strongly.

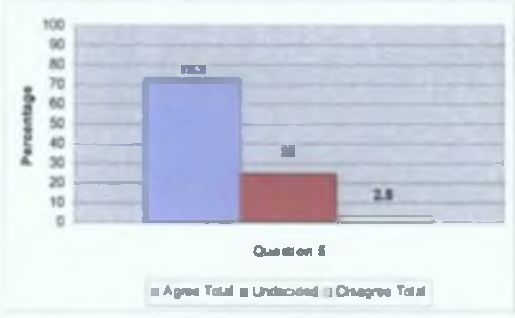

Figure 6.5. Question 5 Result

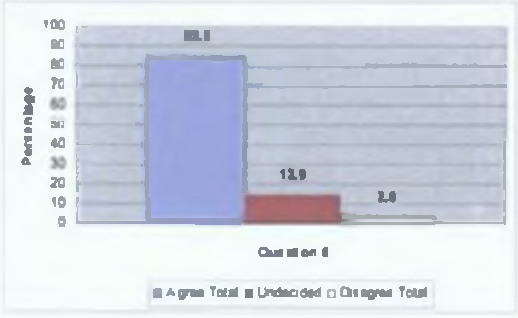

Figure 6.6. Question 6 Result

Question 7: The system is sufficient enough to allow the student to feel competent using SQL select statements (Figure 6.7)

We asked the surveyed students if they felt the system was sufficient enough to allow them to feel competent using SQL select statements. Opinions varied. 50% agreed or strongly agreed that the system was sufficient. A quarter was undecided, while a quarter disagreed in some manner.

Question 8: The questions provided by the system are of varying levels of difficulty (Figure 6.8)

Almost 64% agreed in some manner that there was a range of difficulty in the questions posed by the system. 19.4% of those surveyed disagreed, while the remainder were undecided.

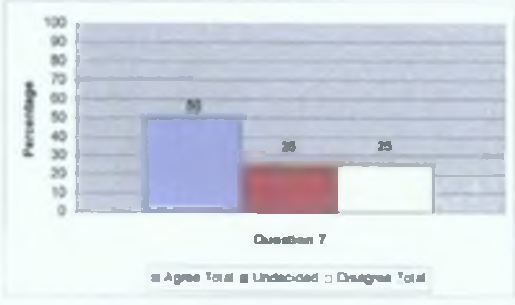

Figure 6.7. Question 7 Result

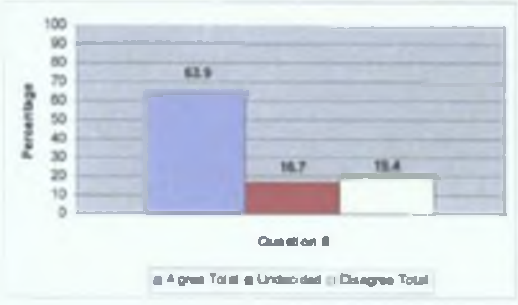

Figure 6.8. Question 8 Result

6.3.2.4 Learning Experience

Question 9: Using the SQL tutor has improved my understanding of SQL select statements (Figure 6.9)

A quarter of those surveyed strongly agreed with the statement that the tutor has improved their understanding of SQL select statements. A further 61% agreed. This is a significantly large total of 86%.

Question 10: Using the SQL tutor is more useful than attending lectures, tutorials and labs in preparation for continuous assessment and exams (Figure 6.10)

Opinion was divided quite evenly as to whether the online tutoring system was more useful than attending lectures, tutorials and labs in preparation for the aforementioned assessments. Just over 30% agreed or strongly agreed with the above statement, practically the same percentage were undecided, while the same again disagreed. A further 5.6% disagreed strongly.

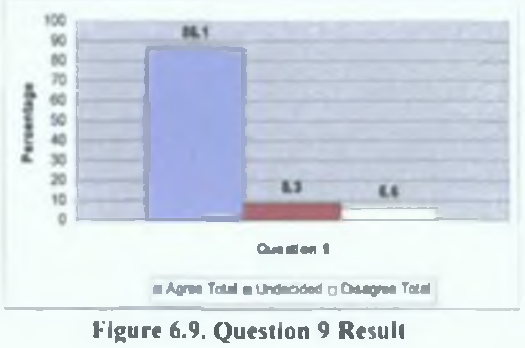

Figure 6.9. Question 9 Result

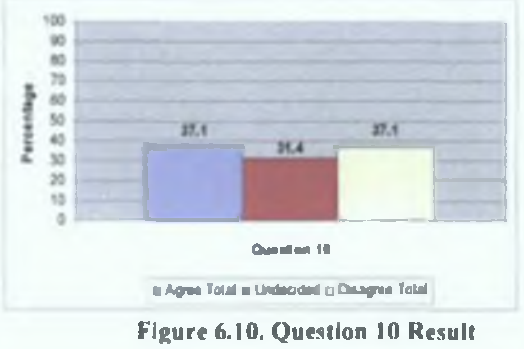

Figure 6.10. Question 10 Result

Question 11: The SQL tutor is a useful teaching and learning tool (Figure 11)

The above question asked students if they felt the tutor was more useful than traditional methods of teaching and learning. Over 90% of students agreed, some strongly, that this was the case. This is a very significant majority.

Question 12: I enjoy learning SQL select statements through practising them (Figure 12)

Almost 63% of those surveyed expressed their enjoyment in learning select statements by practising them. This is a definite majority. 28.6% were undecided. Just 8.5% disagreed.

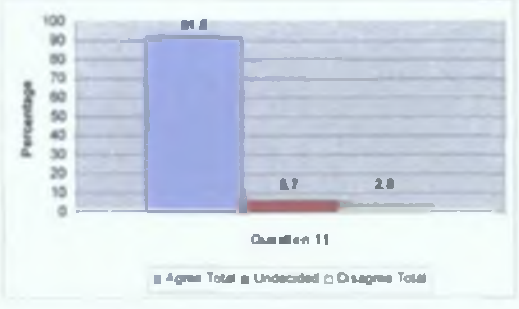

Figure 6.11. Question 11 Result

Figure 6.12. Question 12 Result

Next we wanted to identify elements of the system that were most important to the student (Table 6.5, Figure 6.13). The students were given a list of five options, and were asked to rate them from one (most important) to five (least important). Students could also add their own option, but none did.

- Always available

  Approximately 25% of those surveyed rated this advantage as being the most important of all options to them, with the same number rating it as the second most important option. Close to a quarter (22.6%) of all surveyed rated it as being the third most important to them. The remainder was split between the last two ratings.

- Self-paced learning

  A rating of number two received the highest number of ticks (38.7%) from those surveyed, followed by a number three rating at just over 25%. The remaining votes were split almost evenly between the remaining rating places.

- Easy to use

  6.5% of those surveyed rated the system's ease of use as being most important. Double this rated it as being least important. 22.5%, 22.6% and 35.5% rated it as number two, three and four, respectively.

- A new and enjoyable way to learn

The highest ratings for this option had it at least important (45.2%) and second least important (29%). 12.9% rated is as being most important.

- Practical rather than lecture based

  A majority of almost 42% rated this option as the most important. Interestingly, this option also received the second highest rating (19.3%) overall of "least important".

	RATE 1 %	RATE 2 %	RATE 3 %	RATE 4 %	RATE 5 %
Always available	25.8	25.8	22.6	12.9	12.9
Self-paced learning	12.9	38.7	25.7	12.9	9.7
Easy to use	6.5	22.5	22.6	35.5	12.9
New and enjoyable way to learn	12.9	6.5	6.5	29	45.2
Practical rather than lecture based	41.9	6.5	22.6	9.7	19.3
Other	0	0	0	0	0
TOTAL	100	100	100	100	100

Table 6.5. System Elements Rating

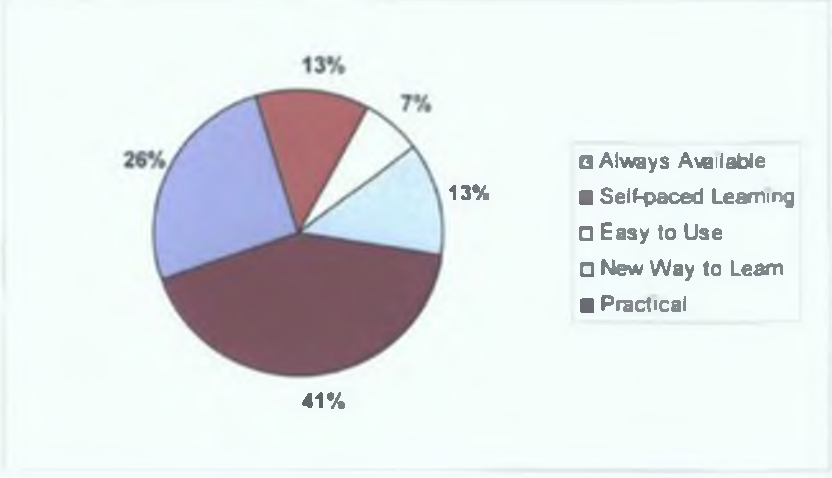

Figure 6.13. System Elements Rating

Most students that used the tutoring system used it on more than one occasion. We asked the students the reason for this, giving a list of options and the opportunity to add their own reasons (Table 6.6, Figure 6.14). The students could tick as many options as applied. The main motivation for using the system was to prepare for the up-coming lab exam – over 45% indicated this was a reason for using it. Close to 30% indicated that they found it easier to use the tutoring system to learn SQL. Over 22.6% of students used it because they attended the supervised lab sessions. Finally, 3.8% used it because they liked it.

RESPONSE	PERCENTAGE
I knew there would be an exam on SQL	45.3
I attended lab sessions and so I used it there	22.6
I enjoyed using it	3.8
I find it easier to learn SQL using this tutoring system	28.3
Other	0
TOTAL	100

Table 6.6. Reasons for Use

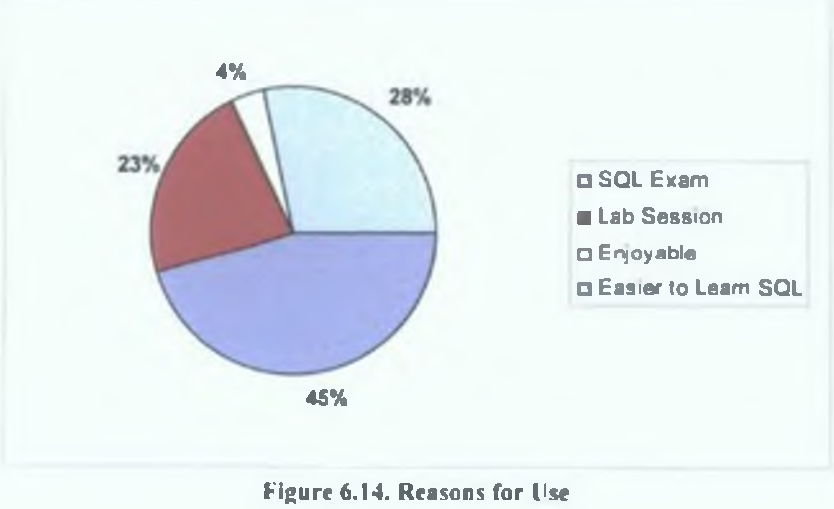

Figure 6.14. Reasons for Use

As previously mentioned, the tutoring system was used in conjunction with lectures and supervised lab sessions. We asked those surveyed to indicate, on a scale of one (yes)

to ten (no), if they thought the tutoring system could act as a complete replacement for human lab tutors (Table 6.7). The average result was number seven on the scale.

LEVEL	PERCENTAGE
Level 1	0
Level 2	2.78
Level 3	13.89
Level 4	8.33
Level 5	5.56
Level 6	0
Level 7	8.33
Level 8	27.78
Level 9	0
Level 10	33.33
TOTAL	100

Table 6.7. Human Tutor Replacement Rating

A significant number of students indicated that they used the tutoring system to prepare for an up-coming exam. Did they feel that their exam results would be better because they used the system? When asked to grade their thoughts on this, on a scale of one (yes) to ten (no), a quarter of those surveyed answered with a definite "yes" (number one on the scale). The average result was number four on the scale.

LEVEL	PERCENTAGE
Level 1	25
Level 2	5.56
Level 3	13.88
Level 4	22.22
Level 5	8.33
Level 6	5.56

Level 7	11.11
Level 8	2.78
Level 9	2.78
Level 10	2.78
TOTAL	100

Table 6.8. Improved Exam Results Perception

Finally, we asked the students to list strengths and weaknesses/suggestions of the system – these are summarised below. Percentage weightings are given in brackets.

Strengths:

- Easy to use (16.67)
- Always available, can be used at home (16.67)
- Visualisation of answers (8.33)
- Feedback is helpful (12.5)
- Interactive (8.33)
- Variety of topics, enough material to cover the course (12.5)
- Practical, activity based, easier to learn SQL then with theory based learning (12.5)
- Self-paced (8.33)
- Fun and enjoyable (4.17)

Students also suggested areas of the system that could be improved. These are discussed in section 6.4.1.2.4.

- Answers not provided for the later questions (8.7)
- Compare the student submission output to the ideal solution output (4.35)
- Include moving pictures (4.35)
- Make the questions harder, to meet the standard of the SQL exam (4.35)
- Downloadable version of an SQL compiler (4.35)
- Accuracy issues (17.38)
- Have less issues about spacing etc. (13.04)
- Better feedback (21.74)

- Poor navigation (4.35)
- More questions to practise, even randomly generated (13.04)

6.3.3 Student Behaviour

We studied the behaviour of students from when the system was first made available until they took the mid-semester SQL exam, and then from the completion of this exam until the middle of June when the end of semester exams took place. 96.6% of students taking the CA218 module logged into the system on at least one occasion. A little over 81% of the total usage of the system took place before the mid-semester SQL exam, while the remainder occurred after this exam.

22.78% of usage before the mid-semester exam occurred during the scheduled lab sessions, which took place once a week for two hours at a time. Human lab assistant were available for consultation during these sessions. 77.22% of the pre-exam usage took place outside of these sessions, some on college computers and some from elsewhere (presumably from the student's home). When we break this result down further, we can see that 27.84% of the total pre-exam usage occurred after six o' clock in the evening. Weekend usage accounted for 4.44% of this period of usage. Of those that used the system, an excess of 70% used it during the twenty four hours preceding the mid-semester exam.

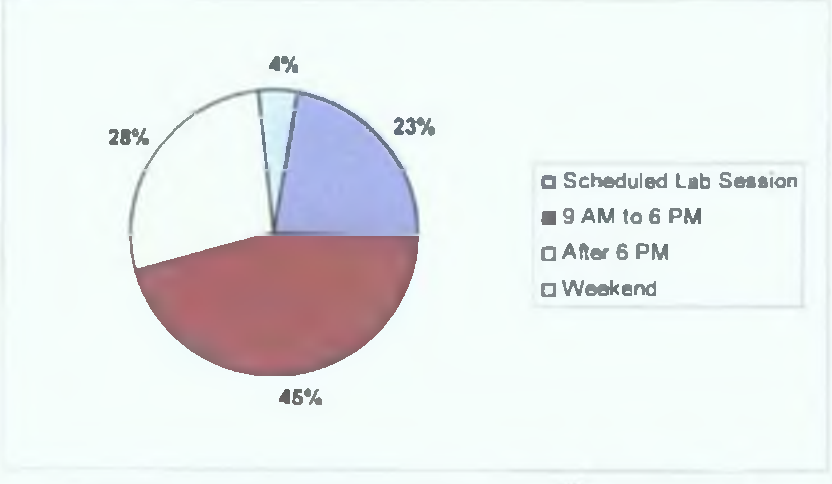

Figure 6.15. Pre Mid-Semester Examination Usage Breakdown

Post mid-semester exam, almost 11% of usage took place during the scheduled lab sessions. The remainder took occurred on week days outside of scheduled lab sessions, one third of which was after six o' clock in the evening.

6.4 Discussion

6.4.1 Student Related Evaluation

6.4.1.1 Student Performance

The trend over the last number of years was for the average mid-semester exam mark to increase by 2% on the year previously. Our extended tutoring system was introduced this year. This was the only change in the module, online and in the traditional classroom-based setting; all other factors, including the difficulty of the mid-semester examination, remained constant. The average mid-semester exam result has increased by 5.4% on last year's average mark.

On first impressions, it would seem that, due to other factors remaining the same as previously, the introduction of our system is responsible for the increase in marks. It is important, however, to be aware of external factors that may have had an influence. Perhaps this year's class are "brighter" than last year's? When we examined the CAO secondary school Leaving Certificate points we found that the average number of points obtained by last year's class was 396, whereas this year's class got an average of 368 – a 28 point difference. So, the previous year's class had "better" academic results that this year's. In addition, this year's overall CA218 result is 1.8% lower than last year. It appears that this year's class are of a lower academic ability, both generally and in terms of the CA218 module, yet they performed better in the mid-semester SQL exam than the preceding CA218 class.

It should also be noted that, due to the fall of a public holiday, there were less scheduled lab sessions this year than last year.

We can conclude, therefore, that the introduction of the new SQL tutoring system has improved student performance in the mid-semester SQL exam.

6.4.1.2 Student Opinion

We have previously presented the results of the student survey. The questionnaire was presented to all students that were present for a routine CA218 lecture. Students had not being previously informed off the questionnaire; the students asked to complete the questionnaire therefore were randomly selected. Thirty six students completed the questionnaire. This number represents over 40% of the number of students that sat the mid-semester exam, which is a sufficient sample of the CA218 student population. Let us now discuss them to see if the results of the questionnaire give a positive or negative evaluation of our automated tutoring tool.

6.4.1.2.1 Usability of the SQL Tutoring System

An over-complicated user interface and user experience is generally not a good trait in an online learning system as it could discourage the student from using it. We hoped to have developed a system that gave an overall impression of ease of use. We also wanted students to feel the scaffolding provided was helpful to them.

The majority of students agreed or strongly agreed that the system is easy to use in general, which is a positive result for us. No one believed strongly that system was not easy to use, again a good result.

A guided discovery learning environment was a key consideration in the design of this SQL tutoring system. By this we mean that we wished for students to be able to choose the direction of their own learning by choosing the combination of questions they wanted to try. This type of environment could lead to some confusion on the students' part; a flaw we wished to avoid. In short, the system should be easy to navigate. A definite majority agreed in some manner that the system was indeed easy to navigate. Some students felt it was not easy to navigate, while others were undecided – they neither agreed nor disagreed. This implies that the navigation of the system, or possibly the system's introduction and instructions page, could benefit from some more work. All in all, however, the navigation received positive feedback.

The feedback and personalised guidance features are an important part of our tutoring system. Indeed, a personalised guidance feature based on a mid-to-long term assessment

is a unique part of our online tutoring system. The survey results regarding the usefulness of these features vary. While a significant proportion of those surveyed were neutral towards the feedback component, the percentage that found it useful was double that who did not. Overall, these results imply that the idea of providing feedback has its merits. Direct face-to-face feedback from CA218 students leads us to believe that these opinions could stem from the occasional inaccuracy of the Correction Component. Obviously, the actual feedback implementation will have to be altered, even slightly, to improve its overall usefulness, along with improving the reliability of the Correction Component.

A considerable amount of students were undecided about the usefulness of the guidance feature. On a positive note however, only a small proportion thought that it was not particularly useful. Again, from this we can conclude that providing guidance is useful, but the actual implementation of both the guidance and query correction features could benefit from being reviewed.

Any apprenticeship model contains a stage where the student is given advice and help when and if he needs it. In keeping with this we provided a method of allowing the student to tailor the levels of feedback or guidance he received. He may keep these levels static, or he can allow the system to change them dynamically based on his actions within the system. A considerable majority agreed or strongly agreed that it was useful to have varying levels of this scaffolding.

6.4.1.2.2 Course Content and Material

The questions around which the tutoring system is based is small but covers a wide range of SQL select statement features. We wished to determine if those using the system felt they were given a sufficiently broad overview of SQL select statements. Of those surveyed, in excess of four fifths agreed or strongly agreed. We can conclude that, as far as presenting a broad overview of the select statement, the questions posed by the automated tutor are of a suitable range.

Varying levels of difficulty in a set of questions will allow the student to feel a sense of achievement while also posing a certain amount of challenge. We wanted to see if the students surveyed did indeed feel that the questions provided were of varying levels of

difficulty. The majority of those surveyed agreed in some manner that there was a range of difficulty – a positive result for the system. However, a number did disagree. This could be attributed to a problem with the questions asked by the system. Alternatively, these students may have either a high or low ability for writing select statements, and may have found all questions easy or difficult, respectively.

We then asked the surveyed students if they felt the system was sufficient enough to allow them to feel competent using SQL select statements. Opinions varied. We can conclude from our results that three quarters of those questioned did not give an explicitly negative answer. Perhaps when the delivery of feedback and guidance are enhanced there will be a greater overall positive reaction to given statement.

6.4.1.2.3 Learning Experience

The primary aim of this tutoring system was to help students to learn how to create SQL select statements; that is, to improve their understanding of them. Very positive results were found in the completed questionnaires with a significant majority indicating that the tutor has improved their understanding of SQL select statements.

Activity-based learning is a key part of this tutor. This means that students learn by practising the select statement rather than learning about various methods of creating a statement from a book. Ideally, for the use of the system to be a well-rounded learning experience, the students should enjoy learning how to write select statements by practising them. A definite majority expressed their enjoyment of learning select statements by practising them.

Online tutoring systems are often developed to replace the traditional classroom-based teaching and learning environment, in which a human tutor would be present. Our tutoring system was used voluntarily in conjunction with traditional lectures and timetabled lab sessions, in which lab tutors were made available. This then is a blended method of learning, due to the combination of computer-based and human-presented learning. All students were working towards a mid-semester exam on SQL select statements, along with a final exam and project that would include sections in SQL. Opinion was split amongst those surveyed as to whether the online tutoring system was

more useful than attending lectures, tutorials and labs in preparation for the aforementioned assessments. Votes were divided evenly between those who agreed in some manner, those who were undecided, and those who disagreed. A further small amount disagreed strongly. While the "no voters" were in majority, we must remember that the question asked if the tutoring could act as a complete replacement. Perhaps the proportions would have been different if the students had been asked if they would accept it as a partial replacement.

While opinions varied on the ability of the system to replace a human tutor, over 90% agreed, some strongly, that the tutoring system was a useful teaching and learning tool in its own right. This is a very significant majority. From this we can see that students see the benefit of it, and so there would be a definite purpose to improving certain features if necessary and making it available to students again in the future.

6.4.1.2.4 Further Opinion about the System

Students were asked to rate five statements about the system in order of most to least importance. Students indicated that the most important aspect of the system is that it is practical rather than lecture based. This is encouraging as one of the systems requirements was that it is based on active learning. Students have obviously embraced this concept. The second most important aspect is that the tutoring system is always available. This is a key feature of the type of computer-based learning environment that we aimed to design and implement. The lowest rating statement was that the system being a new and enjoyable way to learn. The system was part of a module that was subject to continuous assessment and an end-of-semester exam. Generally, when continuous assessment is part of a number of modules over the course of a semester, students are under pressure to cover topics and complete projects in a set period of time. Possibly, the enjoyment of any system would be seen as least important, as learning under a strict time schedule could be stressful rather than enjoyable.

A significant number of students indicated that they used the tutoring system to prepare for an up-coming exam. Did they feel that their exam results would be better because they used the system? When asked to grade their thoughts on this, three out of

four students gave a positive response, i.e. that they felt using the system would result in better results for them in the mid-semester SQL exam. This is a very good result for the system.

Finally, we asked the students to list, in their own words, strengths and weaknesses of the system. A considerable proportion listed strengths that we had originally aimed for; a positive result for this project. For instance, we wanted to develop a tutoring system that is easy to use, is available at times that suit the student's learning patterns, and offers constructive feedback to the student. These elements were among the strengths listed the greatest number of times by students.

Students also suggested areas of the system that could be improved. Some of these suggestions may stem from a lack of information or use of the system, some are not feasible to examine currently, and others will be definite recommendations for future work. Let us comment briefly on the most commonly occurring suggestions.

- Better feedback

  Some students suggested that the hints displayed could be improved. We recommend this be examined in the future.

- Accuracy issues

  It is very difficult to achieve a "perfect" correction method – one that is infallible. The system's accuracy can be improved in the future. For example, SQL constructs not accommodated in the current implementation (outlined in section 5.5) could be dealt with in future versions.

- Have fewer issues about spacing etc.

  This issue is closely related to that of accuracy, and is recommended for examination in the future.

- More questions to practise, even randomly generated

  The system currently poses a pre-defined set of questions. These have been developed to test the student on both fundamental and common aspects of SQL. Some students suggested that the system provide a greater number of questions. Work has begun on the automatic generation of such questions, which is discussed in section 7.3.

- Answers not provided for the later questions

  This is not actually the case – answers are provided for all questions after a set number of incorrect attempts by the student.

6.4.1.3 Student Behaviour

The information found through web usage mining can tell us a lot about the behaviour of students. We use this to validate and complement the results of both the student performance evaluation and student opinion evaluation. We examined usage of the system before and after the mid-semester SQL lab exam took place. Over four fifths of the total usage happened before the mid-semester SQL exam. This was probably because SQL was covered in the lectures in the weeks preceding the mid semester exam, and because students were preparing for this exam.

We can see from student opinion (6.3.2) that some students stated that they used the system during the scheduled lab sessions. This is verified by our web usage mining results – almost 23% of the pre-exam usage of the system was during these sessions. Students also indicated that they used the system to prepare for the mid-semester SQL exam. We can see from our web access logs that a number of students used the system the day before the SQL exam, some in the evening. A significant quantity of students also used the system in the morning before the 9AM exam. Generally a human tutor would not have been available late in the evening or so early in the morning. This is an example of "just-in-time" learning. However, we can tell from web mining that this is also complemented by long-term and pre-emptive use. The majority of usage before the mid-semester exam occurred outside of the scheduled lab sessions. We can see from the logs that most students used the system at college, but some did use it at home. Over a quarter of the total pre-exam system usage took place after six o' clock in the evening. This validates the aim of our, and all, CAL system to allow the user to access the system at anytime and from anywhere. 4.44% pre-exam usage was on weekends, again which would not generally be possible if the presence of a human tutor was required.

Of those that used the system after the mid-semester exam, the majority used it outside of scheduled lab sessions, mainly before six o' clock in the evening. At this stage of the semester students were completing a project and preparing for the end-of-semester exam.

Both of these involved a relatively small degree of SQL. Students may have been using the system to revise what they had learned previously in order to prepare for the assessments just mentioned.

6.4.2 Software Evaluation

The previous evaluations tested the effectiveness and student usability of the system. These results are important for a number of stakeholders, namely the learner, the instructor and the instructional designer.

We can further evaluate the system based on the achievement of our own goals and the general usability of the system. This evaluation, for the benefit of the developer, will test the system on various software engineering and human-computer interaction goals, such as maintainability and usability.

6.4.2.1 Achievement of Goals

Our first goal was to design and implement an automated tutoring system for the SQL select statement. This was to be done using three sub-goals – to develop an architecture for an automated tutoring environment, to analyse the select statement to determine the errors a novice SQL student may encounter, and to discover a set of adaptivity techniques that could be implemented on the system. The first step was to develop the architecture. After giving relevant consideration to the architectures of similar systems, we developed an appropriate architecture which we specified using UML. It was designed in such a manner as to allow various modules to be replaced, or "plugged in" and "plugged out". Modularity was an important feature. We then analysed the select statement to determine errors that could be made. This analysis was used to create a multi-level error categorisation scheme, as described in chapter five. Thirdly, we decided on a set of adaptivity techniques that could be used to help the student learn. These are chiefly manifested in the feedback and guidance features, which were planned in detail before implementation began. The system was then implemented. Throughout this implementation stage pedagogical issues were considered so that we could include

educational principles such as active learning, scaffolding and fading of this scaffolding, and a realistic environment.

Our second goal was to evaluate the system, which is the topic of the current chapter. In order to evaluation to be done the system needed to be tested and used. The system was first tested locally by ourselves, the developers, and by other MSc. and PhD. students in the School of Computing in Dublin City University. Based on this testing phase changes were made to the automated tutor, primarily with the display and with small correction model bugs that presented themselves. The tutoring system was then made available for students taking the CA218 module to use.

The system was the subject to a two-stage evaluation – after the mid-semester SQL exam, and then on the completion of the CA218 end-of-semester exam – as detailed in this chapter.

6.4.2.2 Usability

The ISO set down principles about usability [International Standards Organisation], which Nielsen used [Nielsen, 1994] to create a set of heuristics. Slightly modified [Karoulis, 2003] and reorganised, they are:

- Simple and natural dialogue and aesthetic and minimalist design
- Speak the user's language: match between the system and the real world
- Minimise the users' cognitive load: recognition rather than recall
- Consistency and standards
- Flexibility and efficiency of use – provide shortcuts
- Support users' control and freedom
- Help and documentation

As part of our own evaluation of the system, we need to determine if the system meets the usability principles stated above.

A simple functional interface is one of our own requirements (chapter four). For this reason we retained the "look and feel" of the original interface, which we feel meets this principle.

The system developed is a tutoring tool. As such, it aims to communicate with the student in a language he will understand. While the language is specific at times to errors made in the select statement, it is not overloaded with technical words and phrases.

In chapter three we described a number of systems related to our work. Those closest to ours were Acharya and SQL-Tutor. Both of these systems have an interface through which the student must enter his statement in six separate parts. The developers of these systems claim that this is to minimise the student's cognitive load. We feel, however, that this is in conflict with our aim to provide a realistic setting – in genuine SQL servers the statement is entered as one sentence. We provide one space for the statement to be entered.

Each page that the student visits is based on the same general design to achieve a degree of consistency.

The user of any system needs to be able to navigate with minimum obstruction and time. The same principle should apply to our system. For this reason, there are links on every page to allow the user to return to the main index of queries. The main interface, which the student sees most often, also allows the student to return to the CA218 module homepage and to view the relevant tables. Extra menus, offered by the guidance feature, and the courseware tables are opened in separate windows to lessen confusion.

The principle of supporting users' control and freedom is very similar to ours of providing a guided discovery learning environment. The student, therefore, is free to select the problem he wishes to next attempt, and to view feedback and guidance at a level of his choosing, if any. This allows the student to control the system to his liking. The questions have relevant titles and the system can control the scaffolding provided if the student wishes, thus giving him a degree of support.

There is an information page available through the login area of the system. This page contains details on how to use the system and what to expect from it. The student may benefit from further help and documentation however, for example a short animated tutorial on how to use the system, or pop-up help messages on the main interface.

6.5 Conclusion

We can conclude, overall, that the results we obtained were positive. The average mark obtained in the mid-semester SQL select statement exam has risen by over 5% on last year's result. Previously the marks obtained have increased annually by 2%. The only change in the teaching of SQL select statements this year was the introduction of our automated tutor, so it seems this may have been the reason for the increase in marks.

In general, there was a positive response to the system by students. The majority of those surveyed agreed that system was easy to use and easy to navigate. They liked having the opportunity to view various levels of scaffolding. They felt that the system not only gave them a broad view of select statements, but that it improved their overall understanding of this type of statement. A large majority of 90% agreed that the system is a useful teaching and learning tool, which is a very encouraging response for us. Opinion was slightly more neutral towards the feedback and guidance features. On balance, we can conclude that this means that these features are a good idea but possibly their implementation needs to be reviewed. Those surveyed appear to be quite traditional in that they feel that the system in its current state could not act as a complete replacement for a human tutor.

As discussed in the fifth chapter, there are some issues regarding the system's correction accuracy that need to be resolved. This was compounded when some students suggested they experienced some problems with the correction of their submissions. It is difficult, in the context of this work, to develop and implement a completely accurate correction component. We identified, in the previous chapter, areas of the correction component that could be improved, and we recommend that these be addressed in he future.

We examined student behaviour using web mining and the system's student records. The majority of system usage took place outside of scheduled lab sessions. From this we can conclude that students felt they could use the system independent of help from human lab tutors. We chose a small sample of students to track their individual performance, opinion and behaviour. We found that students who liked the system got high mid-

semester exam results, whereas the student with one of the lowest exam results had not used the system at all.

Finally, we evaluated the system based on our own original objectives and on ISO usability principles. Our system is based on the virtual apprenticeship model, which is characterised by a web context, scaffolding, and activity based learning. While it is not an ITS, it is similar in its architecture. Also, interaction within an ITS is typically learner-content rather than learner-learner; our tutoring system is similar to an ITS in that respect [Kenny and Pahl, 2005]. Overall, we found that we have met our original objectives and requirements within out time constraint; a positive result. We also comply in some manner with the ISO usability principles. We identified some areas of the system that could be improved; this is discussed in the next chapter.

Chapter 7
Conclusion

7.1 Summary of Work

7.1.1 Focus of Work

Our work was based in the areas of university education and Internet technologies. Specifically, our work dealt with developing an automated tutor to teach Structured Query Language (SQL), a database systems language, to undergraduate computer applications students. It was important that the student could work at his own pace, and without a human tutor being present. Within this tutor we focused primarily on the fundamental SQL select statement. This statement can range from being quite simple to being considerably complex. Thus, the teaching and learning of the SQL select statement suits a large group of students with a wide range of ability, as is the case with CA218 class groups. While our implementation was specific to SQL, the key ideas behind the project are applicable to many other areas. Most other formal languages could have taken the place of SQL. Indeed, many of the issues raised can also be explored in a system designed to teach any of a broad range of topics.

7.1.2 Context of Work

Our project mixes designs and theories from Intelligent Tutoring Systems (ITS) and Computer Aided Learning (CAL), and incorporates pedagogical considerations. Research relating to ITSs was examined. We were particularly interested in the ITS's main components – the expert model, the student model, the pedagogical model and the interface. Research into CAL systems and methodologies pointed to the benefits of incorporating personalisation and adaptive hypermedia into such a system. The fundamental aim of any CAL system is to aid the learning process. It is a common mistake to design an online learning without giving due consideration to best teaching and learning practices. It is important, therefore, to design and implementation CAL systems in accordance with pedagogical theories and methodologies. As part of our

research into the theories and work related to our project we looked at online learning systems that dealt specifically with SQL. Four of the five systems reviewed include a various degrees of adaptivity and offer some form of feedback. Unlike those reviewed, our system offers the student optional and personalised guidance based on a previous assessment.

7.1.3 Objectives

The fundamental aim of our project was to develop an actively engaging automated tutoring system. It was important that this system offered various forms of feedback to the student to aid him through his learning path. We chose SQL select statements as the topic to be taught.

Our project then was guided by six objectives. The first objective was to extend the SQL execution environment currently in use as part of the CA218 module. Specifically, we needed to design and then implement an automated tutoring system to teach the SQL select statement. This system was to include domain specific feedback and personalised guidance features. Before beginning the first objective we needed to fulfil three others – our second, third and fourth objectives. The second objective was to create an architecture for an automated tutoring environment. This architecture was based on research into the architecture of various successful online learning systems. Following this, as our third objective, we needed to analyse the SQL select statement in order to identify problems a typical novice student might come have while creating these statements. These errors were categorised according to a multi-level error classification scheme that we have defined. This enabled us to present different views of the error to the student i.e. to display a constructive error in the appropriate context. This aims to minimise student confusion and lead to a quicker and less frustrating learning experience for him. Our fourth objective then aimed to determine adaptivity techniques for use in a knowledge-based feedback and guidance system. These were to be used to develop a system that allows for the adaptation of content and navigation by feedback and guidance features. Following the design, implementation, and testing of the system it was made available for use as part of the CA218 module. Our fifth objective was to evaluate the system, which began after the mid-semester SQL exam. This stage of the project

involved evaluating the results, behaviour, and opinion of the current CA218 students, along with assessing the system based on the fulfilment of our objectives and general software engineering goals. Our final objective was to carry out all of the tasks outlined above based on sound educational principles and pedagogical considerations. This was to ensure that the student has access to a complete and pedagogically sound learning experience.

7.1.4 Implementation

Our implementation was carried out in accordance with our objectives, and with consideration to the requirements set out in chapter four. We first defined our architecture, which was designed to be modular and updateable.

We then analysed the SQL select statement in order to be fully aware of the mistakes a novice SQL student might make. These were then categorised and combined to form a multi-level error classification scheme. There are three levels in total:

- Formal language errors – syntax, semantic, or pragmatic
- SQL clauses – a mistake can be made in any of the six SQL clauses (select, from, where, group by, having, order by)
- SQL elements – common elements present in select statements e.g. tables, attributes, aggregate functions, etc.

Following this the practical implementation of the system began. Firstly, created a number of information objects. These were necessary for storing student information, the questions to be asked, ideal solutions, and such.

We then used Java Servlet technologies to implement the system components:

Interface Component

The student interacts with the system through a standard interface. Here he can view the question, submit his solutions, see the results of his query, and view feedback and guidance if he wishes. This module also contains other ways for the student to communicate with the system – he can choose his scaffolding preferences, view all questions to practice, see a pictorial representation of the tables to query, and link to further questions to practice.

Correction Component

The tutoring system corrects proposed solutions submitted by the student. This is done by comparing submissions to a set of ideal solutions as well as comparing the ideal output to the student query output. If the student makes a mistake it is classified using an abstract error classification notation. Information from this is sent to the interface and to the student module.

Student Component

The student module stores generic information on the student, such as username, password, scaffolding preferences, etc. This module also sorts and stores information of the errors made by the student, both on a short-term and long-term scale.

Pedagogical Component

The pedagogical module plays and important part in the selection of scaffolding to be displayed to the student. Based on information stored by the student module, the pedagogical module selects appropriate feedback and guidance for the student to be shown if he has answered a question incorrectly. The pedagogical therefore is responsible for adding personalisation to the system. Feedback is a form of local scaffolding that is specific to the solution just submitted by the student. It is comprised of hints and links to relevant information, all focussed specifically on errors the student has made. Guidance is scaffolding on a more global scale. This is advice given to the student based on errors he has made over a course of time. Guidance also consists of a menu of further questions for the student to attempt. This is dynamically created based on the student's weak areas. This form of guidance is not available with other web-based SQL tutoring systems that we have researched.

7.1.5 Evaluation

The lifecycle of any project includes an evaluation stage. Much can be learned from an evaluation, and information taken from it can be used to modify and improve the system. We evaluated the system along four lines: student performance, student opinion, student behaviour, and an evaluation based on the achievement of our objectives.

CA218 students must take a mid-semester SQL exam as part of their continuous assessment requirements. It has been observed that the average mark for this exam has increased, on average, by 2% per annum due to the completion of gradual and small scale improvements. This year the average increased by 5.4%, over 3% more than what was expected. The only change made to the module this year was the introduction of our extended automated SQL tutoring system. Other influencing factors were examined and then ruled out, therefore we can conclude that the introduction of the new SQL tutoring system has improved student performance in exams.

Students must enjoy using, primarily, and have a degree of trust in an online learning system. They must also feel that such a system helps them to learn in order to feel motivated to use it. Secondary to this, students should find the system easy and enjoyable to use. The CA218 students were asked to complete a detailed questionnaire about their experiences of using the SQL tutoring system. The students expressed many positive opinions. Students felt that the questions provided by the system were of varying levels of difficulty and gave them a broad overview of SQL select statements. Furthermore, the majority of students surveyed agreed that the system had improved their understanding of select statements and that they thought that their exam results would be higher than if they had not used the system. Certain features of the system were subject to mixed opinions. For example, opinion varied on the usefulness of the feedback and guidance features. Generally, however, there was not a strong negative reaction towards these features. From this we can conclude that these features are good idea, but may appear more useful after some improvement. The number that expressed the opinion that the system could act as a replacement for a human tutor was lower than those who felt it could not. Perhaps a majority would accept it as a partial replacement. More than 90% indicated that they felt the system was a useful teaching and learning tool. Additionally, students particularly liked that the system was practical based and that it was constantly available for use. A considerable majority found the system easy to use, and many agreed that it was easy to navigate.

Student behaviour can be determined by carrying out web usage mining. We studied web access logs to validate and complement the student performance and student opinion evaluation results.

In chapter four we set down a set of objectives and requirements that we wanted to accomplish. Each of these objectives and requirements was considered at each relevant stage of this project, and we can conclude that we fulfilled all the objectives we set down, and included all the requirements that we had listed.

We should give some thought to the applicability and limitations of our automated tutoring system. It was developed specifically for the teaching of SQL. The correction and student components are therefore designed to correct and store SQL errors. A certain degree of coding would be required, along with changing the tutor questions and solutions, in order for the system to be used to teach a non-SQL topic. The system's schemes can be applied directly to other computing subjects however. SQL is a formal language, and we have defined one level of the error classification scheme based on formal language error types. Naturally, these error types can be applied to other formal languages. The architecture is also general enough to be applied to the online teaching and learning of other computing topics. Furthermore, the system implements many pedagogical theories, such as guided discovery and scaffolding, which can be used for other fields of learning, even those that are non-computing related.

There are certain limitations to the system. Firstly, it is part of a blended learning course where most of the theory of SQL is taught through lectures by a human tutor. To benefit from the system the student must know the basics of the SQL select statement. It would not stand alone then as a means of distance learning. In addition, the development of the project, from initial research to evaluation, was considerably costly.

7.2 Conclusions

Automated tutoring systems have become an acceptable method of instruction in universities and institutions across the globe. The implementation of such systems must be coupled with sound teaching and learning methods.

We see, from our research on related work and literature, that pedagogical issues should be addressed throughout the entire project. Learning can be divided into two

categories – knowledge acquisition and skills training. While each type is quite different to the other, we have learned that, regardless of the category, students reach a higher level of understanding when they are being actively engaged in an appropriate environment. For example, students often travel to a relevant country when learning a foreign language (knowledge acquisition), and trades are studied through repeated activity and practise (skills training).

The apprenticeship theory, situated within the constructivism paradigm, 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, and synchronous or asynchronous, each of which has individual qualities. We feel that, with regards to feedback, any automated tutoring system should act as a human tutor would if he was sitting beside the student. That is, it should offer synchronous local feedback when errors have been made, as well as providing global feedback after a period of time based on the student's overall performance. Indeed, in our opinion, automated feedback is a prerequisite for an online active learning environment. In addition, there should be a high level of interaction between the student and the system. By engaging the student repeatedly over a period of time the student should move from shallow learning to a deeper, and so more beneficial, learning level.

The student should also have an amount of control over his learning experience (student autonomy). The reasons for this are two-fold: firstly, it will encourage a sense of responsibility, and secondly, this empowerment should increase his motivation levels. This school of thought has led to the move from restrictive tutoring systems (such as an Intelligent Tutoring System) to facilitative or individually tailored learning environments. While these newer system may be challenging to develop, their benefits could well outweigh the 'old-school' systems.

We agree that a realistic setting is a central success factor in online learning [Pahl et al., 2004] that has long-term benefits. By teaching students a topic in an environment similar to the real one, students will learn from the very beginning how to apply skills in the proper context. A typical database development environment includes editors, and programming, modelling and optimisation facilities. The CA218 online module allows

for each of these functions. Specifically, the automated tutoring system described in this work applies to the programming part of a database development environment.

Formal languages lend themselves to online tutoring – their structure makes them relatively easy to analyse. The student can therefore make submissions to the system and receive results based on a correction. Designing and implementing a flawless correction method is no easy task. It is important, however, to endeavour to achieve the highest level of accuracy possible. Student's trust in the system will increase as correction errors are lessened; they will be confident that their errors are accurately diagnosed and the scaffolding provided is relevant.

We have learned that it is important to carefully define an appropriate and flexible architecture before at the early stages of a system development. Ideally, the architecture will be modular, maintainable and upgradeable. Components and inter-communications should be planned to ensure the system will benefit the student, the instructor, the instructional designer and the software developer.

In conclusion, the automated tutoring system has great potential in the world of education. It is time and location independent (providing there is access to a computer), meaning the student can learn when and where suits him. It can be developed to mimic the action of a typical human tutor – correcting, providing feedback, presenting an assortment of questions, etc. – in a consistent manner. In addition, today's tutoring system is moving towards an adaptive flexible tutor rather than imposing a strict learning path on the student. We conclude that scaffolding, particularly feedback and guidance, are key features of a successful online tutoring system. The system's accuracy and the student's trust level are also important for success.

A number of limitations exist, however. An automated tutoring system requires some development time, and further time may be required to change the topic being taught. For certain topics it may be difficult to create a perfect method of correction. This leads to either a certain fallibility within the system, or an extended development time. It can also be argued that an automated system can never fully replace human tutoring. This may be partly due to a level of inertia among students i.e. an unwillingness to fully embrace an

automated learning experience. This can be alleviated by providing an interesting and, importantly, useful system for them to use. A challenging task perhaps, but ultimately a rewarding one.

7.3 Recommendations for Future Work

There are a number of issues not addressed by this work that could be looked at in the future. Firstly, the correction component is based on pattern matching. Its error diagnosis is usually accurate, but occasionally it produces an incorrect result. Developing a perfect method of correction is a very difficult task, an issue that has been raised by other SQL tutoring systems. While we may not achieve a faultless correction method, we can further refine our correction methods to reduce the instances of misdiagnosis. Other web-based SQL tutoring systems incorporate alternative methods of correction, truth table processing (Acharya, section 3.5) and constraint-based modeling (SQL-Tutor, section 3.6). These or other methods may be also used in our system in the future to provide a more robust method of correction. The architecture has been designed so as to facilitate the replacement of any component.

The current version of our system uses questions and ideal solutions that have been pre-written and stored in the database. In the future the system could generate it own questions and ideal solutions with which the student can work. The generated questions should be sensible and relevant to the automated tutor topics. Work has already begun on this by Holohan [Holohan, 2005], whereby tables are represented in an ontology. Concepts are then used to represent relationships, property values, etc. This requires two models – one of the database concepts (metalevel), and one of the domain. Select statements can then be generated along with the accompanying English text question.

Finally, as part of our evaluation we asked students for their opinion of various aspects of the tutor. Some highlighted aspects that could be improved in the second version of the system. For instance, some students were neutral when asked about the usefulness of the feedback and guidance features. We believe that they are useful features but they could benefit from some change, such as changing the text of the hints and guidance to provide

more accuracy, or making the features more visual. There was some negative opinion expressed when students were asked if the system was sufficient enough to allow students to feel competent when using select statements. The suggested feedback and guidance features may help this. Alternatively, more questions could be added to the system's catalogue. Student's also expressed opinion about the accuracy of the correction model and regarding issues of spacing and formatting. Replacing or upgrading the correction model, as discussed above, could address these issues.

References

Abdelraheem, A.Y. (2003). Computerized learning environments: Problems, design challenges and future promises. *Journal of Interactive Online Learning*, 2 (2).

Advanced Distributed Learning. (2004). *Sharable Content Object Reference Model, SCORM 2004*. 2nd edition. (Online). Available from: <http://www.adlnet.org> (Accessed June 2005).

Bangert-Drowns, R.L., Kulik, C.-L. C., Kulik, J. A. and Morgan, M. (1991). The instructional effect of feedback in test-like events. *Review of Educational Research*, 61, pp213-238.

Barr, A., and Feigenbaum, E.A. (1981). *The handbook of Artificial Intelligence*, 1. William Kaufmann Inc.

Batista, P. and Silva, M.J. (2002). Mining Web access logs of an on-line newspaper. *IN: 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems, RpeC'02 Workshop on Recommendation and Personalization in eCommerce.*

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).

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

Biggs, J. (1999). *Teaching for quality learning at university*. SRHE and Open University Press.

Bloom, B.S. (ed) (1956). *Taxonomy of educational objectives: Book 1 cognitive domain.* London: Longman Group Limited.

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

Brush, T. and Saye, J. (2002). A summary of research exploring hard and soft scaffolding for teachers and students using a multimedia supported learning environment. *Journal of Interactive Online Learning*, 1 (2).

Brusilovsky, P. (2000). Adaptive Hypermedia: From Intelligent Tutoring Systems to Web-based education. *IN: Proceedings of 5th International Conference on Intelligent Tutoring Systems, ITS 2000*, June 2000. *IN:* G. Gauthier, C. Frasson and K. VanLehn (eds.) Intelligent Tutoring Systems, lecture notes in computer science, 1839. Springer Verlag. pp1-7.

Brusilovsky, P. and Peylo, C. (2003). Adaptive and Intelligent Web-based educational systems. *International Journal of Artificial Intelligence in Education*, 13, pp156–169.

Chang, G., Healey, M.J., McHugh, J.A.H. and Wang, J.T.L. (2001). *Mining the world wide web.* Kluwer Academic Publishers.

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), pp255-26.

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

Collins, A., Brown, J.S. and Newman, S.E. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics *IN:* Resnick, L.B. (ed). *Knowing,*

learning, and instruction: Essays in honor of Robert Glaser. Hillsdale, NJ : Lawrence Erlbaum Associates. pp453-494.

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

Conlan, O., Dagger, D. and Wade, V. (2002). Towards a standards-based approach to e-Learning personalization using reusable learning objects. *IN: Proceedings of World Conference on E-Learning, E-Learn 2002*, October 2002.

Connolly, T. and Begg, C. (2002). *Database systems*. Addison Wesley.

Corbett, A.T. and Anderson, J.R. (2001). Locus of feedback control in computer-based tutoring: Impact on learning rate, Achievement and Attitudes. *Proceedings of ACM CHI'2001 Conference on Human Factors in Computing Systems*. New York: ACM Press. pp245-252.

Dagger, D., Wade, V. and Conlan, O. (2004). Developing adaptive pedagogy with the Adaptive Course Construction Toolkit (ACCT). *IN: Workshop proceedings of AH2004 conference (online)*, August 2004.

Dagger, D., Wade, V. and Conlan, O. (2005). Personalisation for All: Making adaptive course composition easy. *IEEE IFETS, Special Edition of the Educational Technology and Science Journal, 2005.*

Daly, C. (1999). RoboProf and an introductory computer programming course. *IN: Proceedings of ITiCSE'99*, June 1999. ACM Press. pp151-158.

De Bra, P. and Stash, N. (2004). Multimedia adaptation using AHA! *IN: Proceedings of ED-MEDIA 2004 World Conference on Educational Multimedia, Hypermedia & Telecommunications*, June 2004. pp563-570.

Dennis, A., Wixom, B.H. and Tegarden, D. (2005). *System analysis and design with UML: Version 2.0.* John Wiley & Sons, Inc.

Dix, A., Finlay, J., Abowd, G. and Beale, R. (1998). *Human-Computer Interaction.* 2nd edition. Prentice Hall Europe.

Feng-Kwei, W. and Bonk, C.J. (2001). A design framework for electronic Cognitive Apprenticeship. *Journal of Asynchronous Learning Networks (JALN)*, 5(2), pp131-139.

Fink, A. (1995). *How to analyze survey data.* SAGE Publications, Inc.

FOLDOC: Free Online Dictionary of Computing. (Online). Available from : <http://wombat.doc.ic.ac.uk/foldoc/> (Accessed 20 March 2005).

Gertner, A.S. and VanLehn, K. (2000). Andes: A coached problem solving environment for physics. *IN: Proceedings of 5th International Conference ITS 2000.* June 2000.

Greenfield, P.S. (1984). A theory of the teacher in the learning activities of everyday life *IN:* Rogoff, B. and Lave, J. (eds). *Everyday Cognition: Its development in social context.* Havard University Press.

Hadjerrouit, S. (2004). Implementing Web-based learning in higher education: An evolutionary software engineering approach. *IN: Proceedings of ED-MEDIA 2004* World Conference on Educational Multimedia, Hypermedia & *Telecommunications.* June 2004. pp483-490.

Heaney, D. and Daly, C. (2004). Mass production of individual feedback. *IN: Proceedings of ITiCSE'04*, June 2004. ACM Press. pp117-121.

Hedberg, J.G. (2002). Ensuring high quality thinking and scaffolding Learning in an online world. *IN: Proceedings of ASCILITE 2002 (Australasian Society for Computers In Learning In Tertiary Education.* December 2002..

Herrington, J. and Oliver, R. (2000). An instructional design framework for authentic learning environments. *Educational Technology Research and Development*, 48 (3), pp23-48.

Herrington, J., Oliver, R. and Stoney, S. (2000). Engaging learners in complex, authentic contexts: Instructional design for the web. *IN: Proceedings of the Moving On Line Conference.*

Herrington, J., Oliver, R. and Reeves, T.C. (2003). Patterns of engagement in authentic online learning environments. *Australian Journal of Educational Technology*, 19 (1), pp59-71.

Holohan, E. (2005). *Dynamic e-Learning problem generation.* Dublin City University Personal Communication, June 2005.

IEEE P1484.1/D9, 2001-11-30. (2001). *Draft standard for Learning Technology – Learning Technology Systems Architecture (LTSA).* (Online). (Accessed June 2005).

ISO – International Organisation for Standardisation. (1998). ISO 9241: Ergonomic requirements for office work with visual display terminals (VDTs). Part 10, Dialogue Principles.

Karoulis, A. and Pombortsis, A. (2003). Heuristic evaluation of Web-based ODL programs *IN:* Ghaoui, C. (ed). *Usability Evaluation of Online Learning Programs.* Information Science Publishing. pp88-109.

Kelly, D. and Tangney, B. (2002). Incorporating learning Characteristics into and intelligent tutor. *IN: Proceedings of Intelligent Tutoring Systems, 6th International Conference, ITS 2002*. June 2002. pp729-738.

Kenny, C. and Pahl, C. (2005). Automated tutoring for a database skills training environment. *IN: Proceedings of ACM SIGCSE Symposium 2005*, February, 2005. ACM Press. pp58-62.

Kenny, C., McMullen, D., Melia, M. and Pahl, C. (2005). Interactive multimedia-enabled learning and training. *ERCIM News, European Research Consortium for Informatics and Mathematics*, 62, pp25-26.

Kunz, P. (2004). The next generation of learning management system (LMS): Requirements from a constructivist perspective. *IN: Proceedings of ED-MEDIA 2004 World Conference on Educational Multimedia, Hypermedia & Telecommunications*, June 2004. pp300-307.

Lajoie, S.P. and Derry, S.J. (eds) (1993). *Computers as cognitive tools*. New Jersey : Lawence Erlbaum Associates Ltd.

Liegle, J.O. and Woo, H.-G. (2000). Developing adaptive Intelligent Tutoring Systems: A general framework and its implementations. *IN: Proceedings of ISECON 2000 (Information Systems Educators Conference)*, November 2000. Volume 17.

Manouselis, N. and Sampson, D. (2004). A multi-criteria model to support automatic recommendation of e-Learning quality approaches. *IN: Proceedings of ED-MEDIA 2004 World Conference on Educational Multimedia, Hypermedia & Telecommunications*, June 2004. pp518-526.

Martin, B.I. (2003). *Intelligent Tutoring Systems: The practical implementation of Constraint-Based modelling.* PhD. University of Canterbury Computer Science and Software Engineering Department, Christchurch, New Zealand.

Masthoff, J. (2002). Design and evaluation of a navigation agent with a mixed locus of control. *IN: Proceedings of ITS 2002. IN:* Cerri, S.A., Gouardères, G. and Paraguaçu, F. (ed)s. *Intelligent Tutoring Systems.* Verlag, pp982-991.

Mayo, M. and Mitrovic, A. (2001). Optimising ITS behaviour with Bayesian Networks and Decision Theory. *International Journal of Artificial Intelligence in Education,* 12, pp124-153.

McArthur, D., Lewis, M. and Bishay, M. (1993). The roles of Artificial Intelligence in education: Current progress and future prospects. *RAND Corporation DRU-472-NSF.*

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

Mitrovic, A. (1997). SQL-Tutor: A preliminary report. Technical Report TR-COSC 08/97. Computer Science Department, University of Canterbury, New Zealand.

Mitrovic, A. and Ohlsson, S. (1999). Evaluation of a Constraint-Based tutor for a database language. *International Journal of Artificial Intelligence in Education,* 10, pp238-256.

Mitrovic, A. and Martin, B. (2000). Evaluating effectiveness of feedback in SQL-Tutor. *IN: Proceedings of Advanced Learning Technologies IWALT 2000,* December 2000. pp143-144.

Mitrovic, A., Martin, B. and Mayo, M. (2000). Using evaluation to shape ITS design: Results and experiences with SQL-Tutor. *User Modeling and User-Adapted Interaction*, 12 (2/3), pp243-279.

MSDN SQL Elements. (2004) MSDN Elements Used in SQL Statements. (Online). Available from: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/odbcsql_grammar.asp> (Accessed July 2005).

Murphy, E. (2003). Moving from theory to practice in the design of Web-based learning from the perspective of constructivism. *Journal of Interactive Online Learning*, 1 (4).

Murray, T. (1999). Authoring Intelligent Tutoring Systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, 10, pp98-129.

Murray, S., Ryan, J. and Pahl, C. (2003). A tool-mediated Cognitive Apprenticeship approach for a computer engineering course. *IN: Proceedings of International Conference on Advanced Learning Technologies ICALT2003*, July 2003. IEEE Press. pp2-6.

Nielsen, J. (1994). Heuristic evaluation *IN:* Nielsen J. and Mack R.L. (eds). *Usability inspection methods*. John Wiley & Sons.

Ohlsson, S. (1992). Constraint-Based student modelling. *Journal of Artificial Intelligence in Education*, 3 (4), pp429-447.

Oliver, R., Herrington, J., Herrington, T. and Sparrow, L. (1996). Using situated learning in the design of interactive multimedia-based learning environments. *IN: Proceedings of the 12th Conference of the Japan Society for Educational Technology*. Japan : JET. pp32.

Pahl, C. (2003a). Managing evolution and change in Web-based teaching and learning environments. *Computers and Education*. 40, pp99-114.

Pahl, C. (2003b). Analysing learning behaviour in Web-mediated interactive learning environments. INVITE Report 02/03. Dublin City University, Ireland. June 2003.

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

Pahl, C. and Kenny, C. (2005). An active learning and training environment for database programming. *IN: Proceedings of International Conference on Educational Hyper- and Multimedia EdMedia'05*, July 2005. AACE.

Patel, A., Russell, D. and Kinshuk (1999). Assessment in a Cognitive Apprenticeship based learning environment – potential and pitfalls *IN:* Brown, S., Race, P. and J. Bull (eds). *Computer-Assisted Assessment in Higher Education.* London : Kogan Page. pp139-147.

Preece, J., Rogers, Y. and Sharp, H. (2002). *Interaction design.* John Wiley & Sons, Inc.

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), pp49-56.

Sadiq, S., Orlowska, M., Sadiq, W. and Lin, J. (2004). SQLator – an online SQL learning workbench. IN: *Proceedings of ITiCSE'04*, June 2004. pp223-227. ACM Press.

Sadler, D.R. (1989). Formative assessment and the design of instructional systems. *Instructional Science*, 18 (2), pp119-145.

Soanes, C. and Hawker, S. (2005). Compact Oxford English Dictionary. 3rd edition. Oxford University Press.

SQLcourse (Online). Available from: <http://www.sqlcourse.com> (Accessed June 2005).

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

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

Urban-Lurain, M. (1996). Intelligent Tutoring Systems: An historic review in the context of the development of Artificial Intelligence and educational psychology. (Online). Available from: <http://www.cse.msu.edu/rgroups/cse101/ITS/its.htm> (Accessed October 2004).

Wikipedia (Online). Available from : <http://www.wikipedia.com> (Accessed 12 April 2005).

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

Zaïane, O.R. (2001). Web usage mining for a better Web-based learning environment. *IN: Proceedings of Conference on Advanced Technology for Education*, June 2001. pp60-64.

Zaïane, O.R. (2002). Building a recommender agent for e-Learning systems. *IN: Proceedings of the 7th International Conference on Computers in Education (ICCE 2002)*, December 2002. pp55-59.

Appendix A

Questions Offered by Tutoring System

1) ONE TABLE

Get distinct colour and city for non-Paris parts with weight greater than 10

select distinct colour, city from p where city <> 'Paris' and weight > 10

select misc att att from table where att symbol misc and att symbol misc

2) MULTIPLE TABLES

Get the name of suppliers and the name of the parts they supply

select sname, pname from s, sp, p where s.sno = sp.sno and sp.pno = p.pno

select att att from table table table where prefix att symbol prefix att and pre fix att symbol prefix att

3) NESTED QUERY

Get supplier names for suppliers who supply part P2

select sname from s where sno in (select sno from sp where pno = 'P2')

select att from table where att in (select att from table where att symbol misc)

4) NESTED QUERY

Get supplier names for suppliers who supply at least one red part

select sname from s where sno in (select sno from sp where pno in (select pno from p where colour = 'Red'))

select att from table where att in (select att from table where att in (select att from table where att symbol misc))

5) QUERY WITH AGGREGATE FUNCTIONS

Get max and min quantity of shipments for part P2

select max(qty), min(qty) from sp where pno = 'P2'

select agg att agg att from table where att symbol misc

6) QUERY WITH GROUP BY

For each part supplied, get the part number and total shipment quantity

select pno, sum(qty) from sp group by pno

select att agg att from table group by att

7) QUERY WITH UNION

Get supplier and part numbers where shipments are greater than 300 or where the supplier who supplies a part has a status not equal to 10

select sno, pno from sp where qty > 300 union select sp.sno, sp.pno from sp,s where s.sno = sp.sno and status <> 10

select att att from table where att symbol misc union select prefix att prefix att from table table where prefix att symbol prefix att and att symbol symbol misc

8) QUERY WITH STRING COMPARISON

Retrieve supplier names which contain the letter 'e'

select sname from s where sname like '%e%'

select att from table where att misc misc

9) QUERY WITH ARITHMETIC OPERATORS

For each shipment of part P1 print out the supplier number (SNO) and 90% of the quantity value

select sno, qty * 0.90 from sp where pno = 'P1'

select att att misc misc from table where att symbol misc

10) QUERY WITH UNIQUE

Get unique colour and city for non-Paris parts with weight greater than 10

select unique colour, city from p where city <> 'Paris' and weight > 10

select misc att att from table where att symbol symbol misc and att symbol misc

11) QUERY WITH ORDERED OUTPUT

Get unique colour and city for non-Paris parts with weight greater than 10, order by colour

select unique colour, city from p where city <> 'Paris' and weight > 10 order by colour

select misc att att from table where att symbol symbol misc and att symbol misc order by att

12) QUERY WITH ALIASES

Get the name of suppliers and the name of the parts they supply - use aliases for table names (supplier, shipment, part)

select supplier.sname, part.pname from s supplier, sp shipment, p part where supplier.sno = shipment.sno and shipment.pno = part.pno

select prefix att prefix att from table misc table misc table misc where prefix att symbol prefix att and prefix att symbol prefix att

13) ONE TABLE

Get distinct supplier names and their cities where status is more than 15

select distinct sname, city from s where status > 15

select misc att att from table where att symbol misc

14) MULTIPLE TABLES

Get the name of suppliers and the city of the corresponding parts

select sname, p.city from s, sp, p where s.sno = sp.sno and sp.pno = p.pno

select att prefix att from table table table where prefix att symbol prefix att and prefix att symbol prefix att

15) NESTED QUERY

Get the names of suppliers who supply a quantity of 200 of any part

select sname from s where sno in (select sno from sp where qty = 200)

select att from table where att in (select att from table where att symbol misc)

16) NESTED QUERY

Get the weights of parts supplied by Jones

select weight from p where pno in (select pno from sp where sno in (select sno from s where sname = 'Jones'))

select att from table where att in (select att from table where att in (select att from table where att symbol misc))

17) QUERY WITH UNION

Get part numbers of red parts or where the supplier of the part is situated in Rome

select pno from p where colour = 'Red' union select pno from sp, s where sp.sno = s.sno and city = 'Rome'

select att from table where att symbol misc union select att from table table where prefix att symbol prefix att and att symbol misc

18) QUERY WITH AGGREGATE FUNCTIONS

Get min and max part weights

select min(weight), max(weight) from p

select agg att agg att from table

19) QUERY WITH GROUP BY

For each supplier, get the total shipment quantity

select sum(qty) from sp group by sno

select agg att from table group by att

20) QUERY WITH STRING COMPARISON

For all parts find cities beginning with 'R'

select city from p where city like 'R%'

select att from table where att misc misc

21) QUERY WITH ARITHMETIC OPERATORS

Print out half of the weight for every London part

select city, weight * 0.50 from p where city = 'London'

select att att misc misc from table where att symbol misc

22) QUERY WITH UNIQUE

Get unique supplier names and their cities where status is more than 15

select unique sname, city from s where status > 15

select misc att att from table where att symbol misc

23) QUERY WITH ORDERED OUTPUT

Get unique supplier names and their cities where status is more than 15, order by city

select unique sname, city from s where status > 15 order by city

select misc att att from table where att symbol misc order by att

24) QUERY WITH ALIASES

Get the name of suppliers and the city of the corresponding parts - use aliases for table names

select supplier.sname, part.city from s supplier, sp shipment, p part where supplier.sno = shipment.sno and shipment.pno = part.pno

select prefix att prefix att from table misc table misc table misc where prefix att symbol prefix att and prefix att symbol prefix att