УДК 681.3.07

А.С. КУЛИК[1], П. АНЦЕНБЕРГЕР[1,2], А.Г. ЧУХРАЙ[1], В.В. КАЛИНИЧЕНКО[1]

[1]*Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Украина*
[2]*Университет прикладных наук, Вельс, Австрия*

ОБ ОДНОМ ПОДХОДЕ К ИНТЕЛЛЕКТУАЛЬНОМУ КОМПЬЮТЕРНОМУ ОБУЧЕНИЮ SQL

Предложен новый подход к интеллектуальному компьютерному обучению языку структурированных запросов к реляционным базам данных SQL. Представлены метод анализа студенческих SQL-запросов, модель классификации запросов, способ сравнения логических выражений, а также метод синтаксического разбора SQL-запросов и построения их синтаксических деревьев с целью выработки педагогически правильных обратных связей посредством автоматического сопоставления синтаксических деревьев эталонных и студенческого SQL-запросов.

Ключевые слова: *интеллектуальное компьютерное обучение, SQL, анализ студенческих запросов.*

Введение

На протяжении последних десятилетий информационные технологии (ИТ) существенно трансформировали среду жизнедеятельности человека. В настоящее время они становятся основным орудием труда, направленным на переработку информации и знаний.

В таких условиях необходима качественная подготовка специалистов, способных создавать функциональные и надежные ИТ-системы различного, в том числе критического, применения.

Неотъемлемой частью большинства сложных ИТ-систем являются реляционные базы данных (РБД) и алгоритмы доступа к данным на основе языка структурированных запросов SQL.

В то же время традиционное обучение языку SQL, впрочем, как и многим другим сложным темам в условиях массового производства, как правило, не дает требуемого качества подготовки каждого студента.

Причина тому – сложность индивидуального и адаптивного подхода к каждому студенту, обусловленная, в том числе, магическим числом Миллера [1], демонстрирующим ограниченность кратковременной памяти человека.

По мнению многих исследователей [2 – 11], выход следует искать в создании и внедрении компьютерных обучающих программ, обладающих практически неограниченной памятью и развитыми интеллектуальными способностями.

Настоящая статья посвящена новому подходу к интеллектуальному компьютерному обучению (ИКО) SQL.

1. Постановка задач

В известных работах [2 – 4] описаны различные подходы к ИКО SQL. Анализ этих работ был проведен авторами в работе [5].

Основной проблемой предложенных подходов является их недостаточная гибкость при появлении новых требований. Вследствие этого возникает задача создания нового подхода к ИКО SQL, который объединял бы в себе достоинства уже существующих подходов, и, насколько это возможно, устранял их недостатки.

Задачами настоящей работы являются:

1) разработка модели классификации студенческих SQL-запросов.

2) разработка метода анализа студенческих SQL-запросов;

3) разработка способа сравнения логических выражений;

4) разработка метода синтаксического разбора SQL-запросов и построения их синтаксических деревьев.

2. Модель классификации студенческих SQL-запросов

Поскольку для языка SQL характерна избыточность форм представления запросов, то для любой задачи может существует более одного рационального SQL-запроса.

Отсюда можно сделать предположение о том, что необходимо проверять соответствие студенческого SQL-запроса некоторому набору эталонов, выявленных для данной задачи.

В случае совпадения с одним из эталонных решений система идентифицирует ответ как известный системе. Каждый эталон, хранимый в системе, характеризуется набором параметров, определяющих его качество (и, следовательно, знания и умения обучаемого). Такими параметрами, к примеру, могут пространственно-временные затраты, количество символов в запросе (обуславливает удобство чтения и понимания запроса другим человеком).

В случае, когда система не идентифицирует ответ как известный, она может попробовать выполнить его на реальной БД на нескольких наборах данных. Поскольку выборка некоторого множества кортежей является основным заданием работы SQL-запроса (в рамках данной работы рассматриваются только запросы выборки), то все правильные ответы к данному заданию должны выбирать одни и те же множества данных на любом начальном множестве данных. Отсюда с большой долей вероятности можно сделать вывод, что если запрос не известен системе, но на разных начальных наборах данных дает на выходе одно и то же множество кортежей, что и эталонный (правильный по умолчанию), то этот запрос можно классифицировать как верный ответ.

Таким образом, можно выделить следующие классы студенческих SQL-запросов:

1. Наилучшее решение, известное системе;
2. Приемлемое решение, но существует лучше;
3. Решение, неизвестное системе:
   3.1. Неправильное;
   3.2. Похожее на правильное.

Графическая модель классификации студенческих ответов представлена на рис. 1.

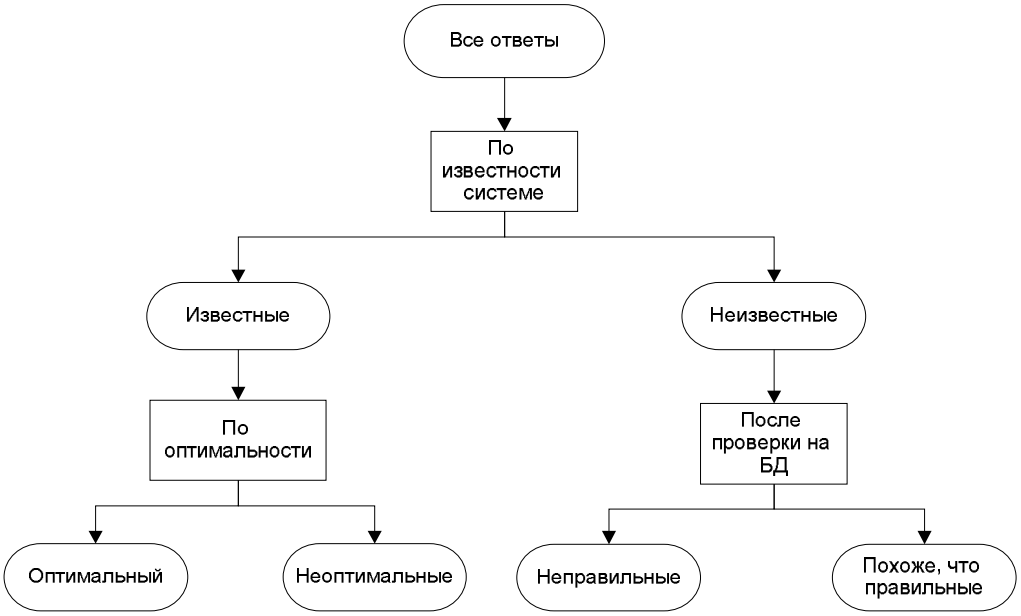

Рис. 1. Графическая модель классификации студенческих ответов

3. Метод анализа студенческих SQL-запросов

Каждое задание, которое средство ИКО SQL предоставляет студенту, характеризуется двумя типами параметров: скрытыми и открытыми. К скрытым параметрам можно отнести те параметры, которые студенту необходимо отразить в написанном SQL-запросе, например, имена таблиц, полей, значения полей, условия и т.д.; эти параметры необходимы системе для генерации эталонных решений конкретного задания и не должны показываться студенту. Открытыми параметрами являются условие задания, схема таблиц и их связей в БД; они показываются студенту и на их основе он решает поставленную задачу.

После генерации нового задания (его скрытых и открытых параметров) студенту на рабочую форму программы выводится текст (условие) задания, схема БД и поле для ввода собственного SQL-запроса. После нажатия кнопки подтверждения ответа система считывает SQL-запрос из текстового поля, и далее приступает к его анализу в соответствии со следующим сценарием:

1. Происходит лексический анализ SQL-запроса (разбиение непрерывной строки на лексемы, и их классификация как терминалов и нетерминалов);

2. Выполняется синтаксический анализ SQL-запроса в соответствии с описанным набором правил грамматики данного языка;

3. Дерево синтаксического разбора студенческого запроса сравнивается с деревьями заранее

сгенерированного набора эталонов. При этом каждый эталон описывается набором параметров, характеризующих его качество по ряду аспектов: пространственно-временные затраты, удобство восприятия другими пользователями (читабельность) и т.д. (набор параметров можно расширить в условиях изменившихся требований);

При этом:

3.1. В случае, если студенческий SQL-запрос совпадает с эталоном, который характеризуется оптимальными параметрами качества, студенту выводится сообщение об абсолютно верном решении;

3.2. Если студенческий SQL-запрос совпадает с одним из эталонов, который характеризуется не наилучшими показателями, студенту выводится предупреждение о правильном, но не оптимальном ответе, и предлагается подумать еще;

3.3. Если студенческий SQL-запрос в большой степени совпадает с одним из эталонов, но не является правильным, то студент информируется о месте допущенной ошибки, и ему предлагается решить данную задачу еще раз и исправить ошибку;

3.4. Если SQL-запрос студента не соответствует ни одному из эталонов, система переходит к следующему пункту;

4. Запрос выполняется на реальной БД на разных наборах данных, и полученное множество кортежей сравнивается с множеством кортежей, полученных при выполнении эталонного запроса, характеризующегося наилучшими показателями качества. Сравнение множеств может производиться путем расчета хеш-функций обеих множеств, и сравнения их значений;

4.1. В случае совпадения выбранных множеств студенту выводится сообщение о правильном ответе. Поскольку SQL-запрос верен, но не известен программе, то далее в системе регистрируется новый эталонный запрос для данного задания;

4.2. В случае, когда показатели качества студенческого SQL-запроса выше, чем в наилучшем эталонном запросе, известном системе, этот новый запрос регистрируется как новый наилучший эталон. Студенту в таком случае могут быть начислены дополнительные баллы;

4.3. В случае несовпадения множеств ответ студенту считается неверным. Ему предлагается другое задание на такую же тему;

5. В случае правильного ответа студенту начисляются баллы за это задание, и система переходит к новому заданию на другую тему.

Представленный метод позволяет проводить анализ студенческого SQL-запроса по многим критериям.

Например, если студенческий ответ является правильным, но не оптимальным решением, система предлагает студенту найти оптимальный ответ, вынуждая студента более глубоко вникнуть в данную тему.

Если же ответ неверен, то студенту указывается место его ошибки. Для анализа неизвестных системе ответов существует их проверка на реальной БД, которая отображает способность системы к самообучению.

Графическое представление метода анализа студенческих SQL-запросов представлено на рис. 2.

4. Способ сравнения логических выражений

Логические выражения, которые для многих запросов содержатся в секции WHERE, имеют значительную избыточность эквивалентных форм представления. Например, выражение $A \vee B \wedge \overline{C}$ может быть также записано как $A \vee \overline{C} \wedge B$ или $B \wedge \overline{C} \vee A$. В связи с этим возникает задача проверки логических выражений на эквивалентность.

Два выражения могут считаться эквивалентными, если на любых наборах аргументов они дают одинаковые результаты.

В нашем случае, два логических выражения эквивалентны тогда и только тогда, когда их таблицы истинности (ТИ) совпадают.

ТИ для выражения из N пропозициональных слов будет иметь N столбцов и 2^N строк аргументов.

Для вычисления значений выражений на заданном наборе аргументов удобно использовать обратную польскую запись (ОПЗ), так как в ней операции не имеют приоритетов, и отсутствуют скобки.

Алгоритм расчета ТИ:

1. Выражение переводится в ОПЗ;
2. Составляется матрица аргументов логической функции размерностью $[2^N, N]$;
3. Для очередной строки ТИ производится подстановка аргументов, содержащихся в данной строке, в выражение, записанное как ОПЗ;
4. Рассчитанное значение функции записывается в массив результатов.

В то же время, если пропозициональные слова в двух выражениях будут записаны в разном порядке, то и ТИ будут иметь разный вид.

Для удобства их сравнения можно отсортировать столбцы ТИ по именам пропозициональных слов, либо посчитать для каждой ТИ контрольную сумму.

Предполагается в дальнейшем использовать второй подход.

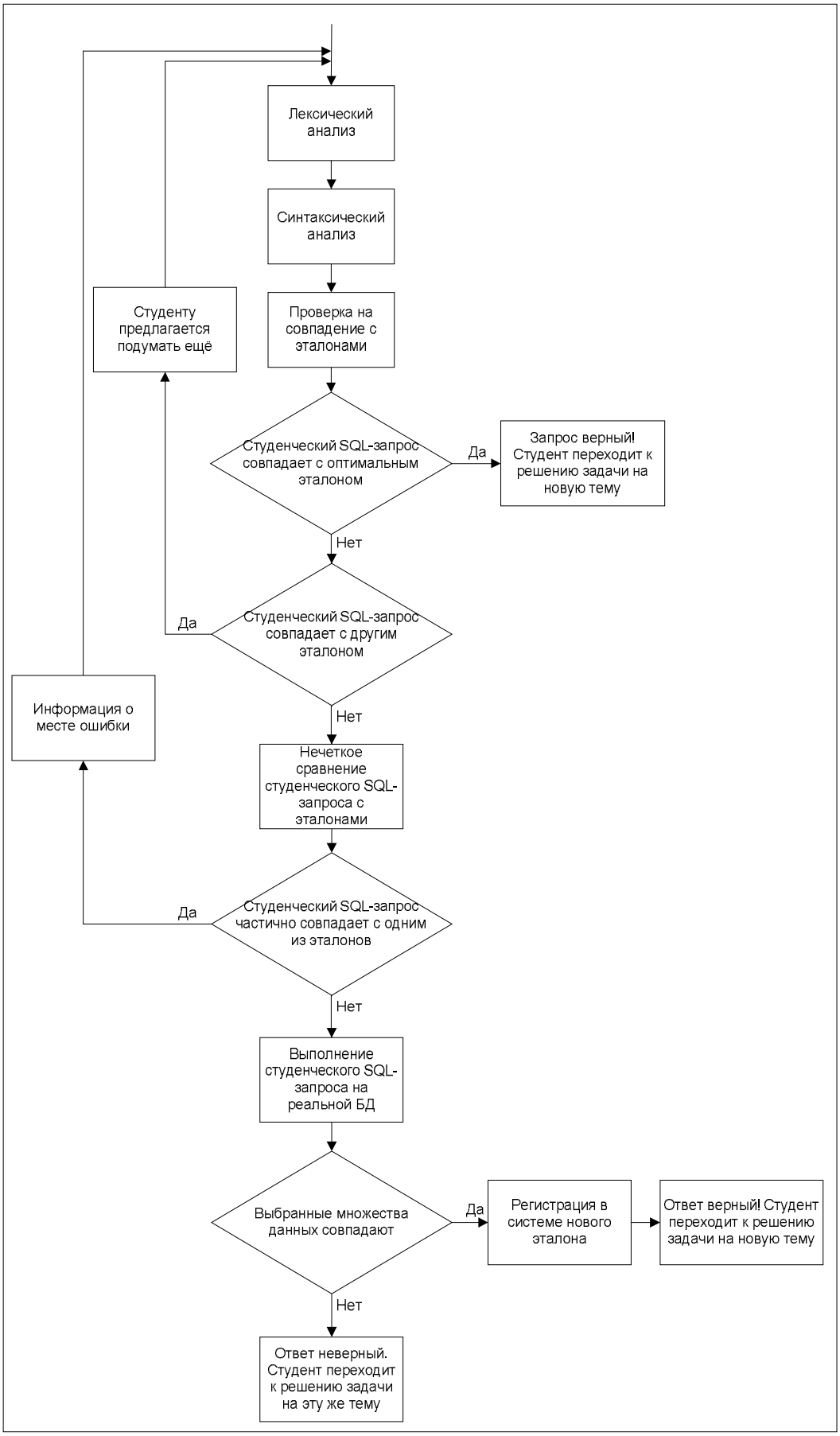

Рис. 2. Графическое представление метода анализа студенческих SQL-запросов

5. Метод синтаксического разбора SQL-запросов и построения их синтаксических деревьев

SQL – язык, который может быть описан в форме Бэкуса-Наура (БНФ). Его грамматика является контекстно-свободной (КС-грамматика).

Для создания компилятора SQL в данной работе использовалась система LEX/YACC [12], которая позволяет, на базе грамматики языка, описанного в БНФ, создать собственный компилятор этого языка.

5.1. Лексический анализ

Для создания лексического анализатора SQL использовалась программа GPLEX (Gardens Point LEX). Входной файл для GPLEX состоит из следующих блоков: определений, правил и кода пользователя (последняя часть необязательна)

В блоке определений описываются лексические классы анализатора. Для SQL были описаны 2 класса:

1) Буквы (включает в себя буквы английского алфавита верхнего и нижнего регистров, а также символ подчеркивания «_»);
2) Цифры (от 0 до 9).

Также в этой части указывается пространство имен, в которое будет помещен класс лексического анализатора.

Блок определений имеет следующий вид:

```
%namespace SQLTeacher

letter [A-Za-z_]
digit [0-9]
```

В блоке правил приводится классификация токенов в зависимости от последовательности символов, полученной на входе анализатора. Лексический анализатор передает синтаксическому анализатору класс токена, который, в свою очередь, объединяет их в нетерминалы.

Например, следующая запись:

```
[Ss][Ee][Ll][Ee][Cc][Tt]
        {return (int)Tokens.SELECT;}
```

определяет ключевое слово SELECT, причем буквы могут быть написаны и в верхнем, и в нижнем регистре. В класс синтаксического анализатора передается номер токена SELECT.

```
{letter}({letter}|{digit})*
        {return (int)Tokens.NAME;}
```

Данная запись определяет класс идентификатора. На них в языке SQL, как и в большинстве формальных языков, накладываются следующие ограничения: первый символ должен быть буквой, а все следующие могут быть буквой или цифрой. Символ «*» означает, что конструкция, написанная в скобках, может повторяться ноль или больше раз. При этом в класс синтаксического анализатора передается номер токена NAME.

5.2. Синтаксический анализ

Для проведения синтаксического анализа использовалась программа GPPG (Gardens Point Parser Generator).

Синтаксический анализатор, построенный с помощью GPPG, является LALR-анализатором.

Входной файл для GPPG, по аналогии с LEX, состоит из трех частей: объявлений, грамматических правил и кода пользователя. При этом код пользователя может быть вставлен как в начале, так и в конце. Часть объявления, использованная в данной системе, выглядит следующим образом:

```
%namespace SQLTeacher

%start list

%token DIGIT NAME AND OR NOT SELECT FROM WHERE IN

%left OR
%left AND
%left NOT
%left <subtok> COMPARISON /* = < > */
%left '+' '-'
%left '*' '/'
%left UMINUS
```

Первая строка описывает пространство имен, в которое будет помещен класс синтаксического анализатора. Далее следует определение корневого нетерминала (list). После команды %token приводится список всех терминалов. Также здесь указываются приоритеты логических и алгебраических операций (все операции – лево-ассоциированные, поэтому указана команда %left).

В секции правил описывается грамматика в БНФ: слева пишутся нетерминалы, начиная с корневого, а справа – их продукции. В свою очередь, продукции тоже описываются как нетерминалы. Любое правило, в конце концов, должно полностью представляться терминалами.

Фрагмент описания синтаксиса SQL в данной системе представлен далее:

```
sel_sect:
SELECT columns    { }
;
columns:
column                    { }
|   columns ',' column     { }
;
column:
NAME                      { }
;
```

Секция SELECT представляется как терминал SELECT, за которым следует список имен столбцов таблицы. Список может состоять из одного или нескольких имен столбцов. На самом нижнем уровне имя является терминалом NAME. Необходимо отметить, что данное описание приводится здесь в качестве демонстрации принципа работы. В действительности же набор правил для SQL значительно сложнее.

Также GPPG позволяет для каждого грамматического правила вставлять программный код, который будет выполнен в случае выполнения этого правила. Таким образом, если на каждом шагу записывать информацию о текущем нетерминале в массив, то после проведения анализа всего запроса, на основе этого массива можно будет построить дерево синтаксического разбора. В секции кода каждого правила вызывается процедура AddToRuleOrder, которая описывается следующим образом:

AddToRuleOrder(int ruleNum, char type, char arity), где:

ruleNum – номер текущей продукции среди всех продукций данной грамматики;

type – переменная, которая указывает на то, терминальное это правило или нетерминальное (т.е. есть ли у него потомки);

arity – переменная, указывающая на арность продукции (унарная или бинарная).

5.3. Построение дерева синтаксического разбора

В работе была выдвинута гипотеза о том, что педагогически правильные обратные связи могут быть построены в результате сравнения синтаксических деревьев эталонных и студенческого SQL-запросов.

Построение синтаксического дерева происходит за несколько шагов.

На первом шаге создается массив, в который записывается список нетерминалов, обрабатываемых синтаксическим анализатором.

Далее, на основе информации, которая содержится в этом массиве, и, базируясь на том факте, что каждый нетерминал имеет не более 2-х продукций (это обусловлено описанной грамматикой), строится дерево синтаксического разбора. Поскольку данный анализатор является восходящим, то для любого правильного с точки зрения грамматики запроса корневое правило будет выполнено последним. Поэтому в теле кода этого правила вызывается процедура построения синтаксического дерева.

На рис. 3 представлен пример построения дерева для простого запроса.

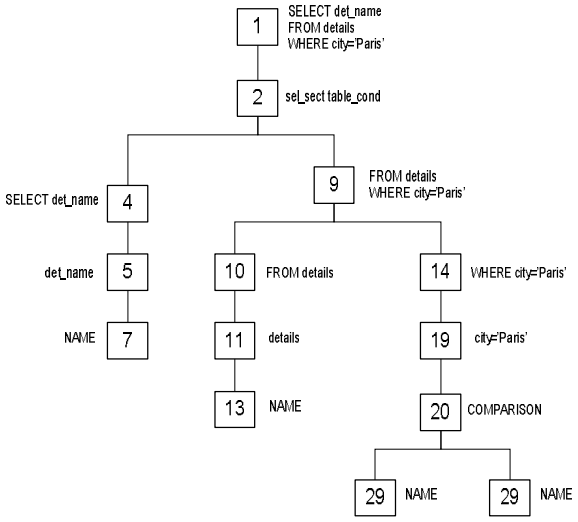

Рис. 3. Дерево синтаксического разбора

Терминалы грамматики для удобства восприятия записаны заглавными буквами. В узлах записаны номера правил, справа от узла – каким терминалом или нетерминалом является данное правило. При этом узлы правил 7, 13 и 29 являются терминальными, все остальные – нетерминальными. Узлы 2, 9 и 20 являются бинарными, все остальные – унарными. Следует отметить, что грамматика описана таким образом, чтобы дерево разбора любого запроса было бинарным, то есть каждый узел имел бы не более 2-х потомков. Это обусловлено практической нецелесообразностью написания эффективного и универсального алгоритма для деревьев арностью больше двух.

Заключение

Предложенный в работе подход нацелен на повышение эффективности подготовки студентов высших учебных заведений по дисциплинам, связанным с использованием баз данных, с помощью интеллектуального компьютерного средства обучения SQL.

Перспективной в данном направлении является разработка методов адаптивного выбора новой задачи в зависимости от текущих успехов студента, а также создание интерактивной системы подсказок обучаемому.

Литература

1. Miller G. The magical Number Seven, Plus Or Minus Two: Some Limits on Our Capacity for Information Processing / G. Miller // The Psychology Review. – March, 1956. – Vol. 63, № 2.

2. Mitrovic A.A. Comparative Analysis of Cognitive Tutoring and Constraint-Based Modeling / A. Mitrovic, K. Koedinger, B. Martin // Lecture Notes in Computer Science. – 2702. – P. 313-322.

3. Mitrovic A. An intelligent SQL tutor on the web / A. Mitrovic // International Journal of Artificial Intelligence in Education. – 2003. – 13. – P. 171-195.

4. Brass S. Semantic Errors in SQL Queries: A Quite Complete List / S. Brass, C. Goldberg // Quality Software, Fourth International Conference on (QSIC'04). – 2004. – P. 250-257.

5. Кулик А.С. Задачи разработки адаптивного компьютерного средства обучения SQL / А.С. Кулик, А.Г. Чухрай, Петер Анценбергер // Радіоелектронні комп'ютерні системи. – 2009. – № 7. – С. 19-25.

6. VanLehn K. The Behavior of Tutoring Systems / K. VanLehn // International Journal of Artificial Intelligence in Education. – 2006. – Vol. 13, Issue 3. – P. 227-265.

7. Katz S. Sherlock 2: An intelligent tutoring system built on the LRDC framework. Facilitating the development and use of interactive learning environments / S. Katz, A. Lesgold, E. Hughes, D. Peters, G. Eggan, M. Gordin. – NJ: Erlbaum, 1998. – P. 227-258.

8. VanLehn K. The Andes physics tutoring system: Lessons learned. / K. VanLehn, C. Lynch, K. Schultz, J.A. Shapiro, R.H. Shelby, L. Taylor et al. // International Journal of Artificial Intelligence in Education. – 2005. – 15(3). – P. 147-204.

9. Anderson J.R. Cognitive Tutors: Lessons Learned / J.R. Anderson, A.T. Corbett, K.R. Koedinger, R. Pelletier // Journal of the Learning Sciences. – 1995. – 4(2). – P. 167-207.

10. Collins A.. Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics / A. Collins, J.S. Brown, S.E. Newman. – In L. B. Resnick (Ed.) Knowing, learning and instruction: Essays in honor of Robert Glaser. – Hillsdale, NJ: Lawrence Erlbaum Associates. – P. 543-494.

11. Chukhray A. The approach to student queries classification for adaptive computer teaching SQL / A. Chukhray, A. Kulik, P. Anzenberger, V. Kalinichenko, M. Chukhray // Proc. of Interactive Computer Aided Learning Conference 2009, Villach, Austria.

12. Levine J.R.. LEX & YACC / J.R. Levine, T. Mason, D. Brown. – O'Reilly & Associates, Inc., 1992 – 387 p.

Поступила в редакцию 28.01.2010

Рецензент: д-р педагог. наук, проф. С.А. Раков, зам. директора, Украинский центр оценивания качества образования, Украина.

ПРО ОДИН ПІДХІД ДО ІНТЕЛЕКТУАЛЬНОГО КОМП'ЮТЕРНОГО НАВЧАННЯ SQL

А.С Кулік, П. Анценбергер, А.Г. Чухрай, В.В. Калініченко

Запропонований новий підхід до інтелектуального комп'ютерного навчання мові структурованих запитів до реляційних баз даних SQL. Представлені метод аналізу студентських SQL-запитів, модель класифікації запитів, спосіб порівняння логічних виразів, а також метод синтаксичного розбору SQL-запитів і побудови їх синтаксичних дерев с метою створення педагогічно правильних зворотних зв'язків за допомогою автоматичного порівняння синтаксичних дерев еталонних і студентського SQL-запитів/

Ключові слова: інтелектуальне комп'ютерне навчання, SQL, аналіз студентських запитів.

ABOUT ONE APPROACH TO INTELLIGENT COMPUTER TUTORING OF SQL

A.S. Kulik, P. Anzenberger, A.G. Chukhray, V.V. Kalinichenko

A new approach of intelligent computer tutoring to the structured query language to relational database SQL is offered. Method of students' SQL-queries analysis, model of query classification, way of logical expressions comparison and method of SQL-queries parsing and its syntax trees building aimed for creation pedagogically right feedbacks by automatically comparison etalon's and students' SQL-queries are presented.

Keywords: intelligent computer tutoring, SQL, students' queries analysis.

Кулик Анатолий Степанович – д-р техн. наук, проф., декан факультета систем управления летательных аппаратов, зав. кафедрой систем управления летательных аппаратов, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков, Украина, e-mail: kulik@d3.khai.edu.

Анценбергер Петер – аспирант кафедры систем управления летательных аппаратов, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», препод., Университет прикладных наук, Вельс, Австрия, e-mail: peter@peter-partner.com.

Чухрай Андрей Григорьевич – канд. техн. наук, доц., зам. декана факультета систем управления летательных аппаратов по НИР, доц. кафедры систем управления летательных аппаратов, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков, Украина, e-mail: chukhray@d3.khai.edu.

Калиниченко Вячеслав Вячеславович – студент факультета систем управления летательных аппаратов, кафедры систем управления летательных аппаратов, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков, Украина, e-mail: slavakpss89@ukr.net.