УДК 681.3.07

А.Г. ЧУХРАЙ[1], Е.С. ВАГИН[1], З.В. ТОМЧЕНКО[1], П. АНЦЕНБЕРГЕР[1,2]

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

ПРИМЕНЕНИЕ МЕТОДА K-MEANS ДЛЯ ФОРМИРОВАНИЯ ВНЕШНЕГО ЦИКЛА ОБУЧЕНИЯ В КОМПЬЮТЕРНЫХ ПРОГРАММАХ, ОБУЧАЮЩИХ SQL

Предложен метод кластеризации заданий в компьютерных обучающих системах для решения задачи организации внешнего цикла обучения при изучении SQL. Решение рассматриваемой задачи относится к кластерному анализу, в частности – применению алгоритма k-means (k-средних). Рассмотрен один из подходов к кластеризации обучающих заданий, основанный на компонентах компетенции. Предложен метод упорядочивания кластеров и заданий по их сложности. Затронуты вопросы, связанные с разработкой архитектуры редакторов интеллектуальных компьютерных систем обучения, возникшие при разработке редактора SQLTOR. Применение предложенных решений в SQLTOR обеспечивает снижение трудовых и временных затрат специалиста в предметной области (преподавателя) при составлении компьютерного обучающего курса SQL. Сформулированы основные перспективные пути развития редактора SQLTOR.

Ключевые слова: *редактор интеллектуальных систем обучения, обучение SQL, кластеризация, k-средних, расстояние Хемминга, компоненты компетенции в обучении.*

Введение

Разработка компьютерных обучающих программ (КОП) является одним из наиболее приоритетных направлений развития средств обучения. Это обусловлено рядом преимуществ КОП перед классическим подходом к преподаванию: адаптацией под конкретного обучаемого, возможностью наглядного визуального моделирования объектов и явлений, снижением трудовых и временных затрат в процессе составления и проверки курсов обучения, возможностью осуществлять дистанционное обучение и т.д. Наиболее развитые из КОП часто называют интеллектуальными компьютерными системами обучения (ИКСО). ИКСО характеризуются наличием внешнего и внутреннего обучающего циклов, обратной связью, обеспечивающей подсказки и руководства к действиям, нелинейным сценарием прохождения обучающего курса, динамической, настраиваемой базой знаний, способностью самообучаться [1,2]. Как правило, в состав ИКСО входят такие элементы: модель предметной области, модель студента и модель обучения [2,3]. Также выделяют модель взаимодействия с пользователем [3]. Таким образом, развитые ИКСО требуют больших трудозатрат при их разработке, знаний не только в предметной области, но и в программировании. Преодолеть эту сложность позволяют специальные программные средства, предоставляющие удобные и понятные инструменты для создания элементов КОП и ИКСО. Такие средства получили название «Редакторы ИКСО» (англ. – ITS Authoring Tools [4]) или РИКСО. Примеры редакторов – DIAG, RIDES, SIMQUEST, XAIDA, Demonstr8, D3, TRAINER, ASPIRE, GTE, REDEEM, Eon, Interbook, MetaLinks, CALAT, CTAT, [3, 4] «Универсальная среда разработки и трансляции обучающих программ» [5] и другие. Авторский инструментарий, как правило, требует большого труда программистов, но, будучи один раз созданным, может в значительной степени облегчить работу экспертов в предметной области по созданию КОП. Постепенно РИКСО совершенствуются, в них добавляются новые возможности. Вместе с тем, по-прежнему существует ряд нерешенных задач, связанных с их разработкой. Одной из таких задач является поиск эффективного метода автоматической интеллектуальной группировки обучающих заданий в ИКСО. Группировка заданий может автоматизировать процесс формирования внешнего обучающего цикла, предоставить дополнительные возможности для принятия решения о том, какое задание увеличит прирост знаний учащегося, если будет предложено ему следующим. Ручная группировка в значительной степени увеличивает время, затрачиваемое специалистом в предметной области на подготовку обучающего курса ИКСО. Для решения таких проблем предлагается метод автоматической кластеризации, как один из инструментов РИКСО языку SQL.

Постановка задачи исследования

РИКСО должны поддерживать множество функций, облегчающих автору процесс проектиро-

вания ИКСО. В работе [3] выделяются структурные элементы РИКСО, аналогичные элементам ИКСО, а также способ представления знаний [3]. При создании РИКСО по языку SQL «SQLTOR» была использована модель знаний, представляющая собой совокупность компонентов компетенции (КК), которые ставятся в соответствие SQL-запросам. Каждый из SQL-запросов представляет собой решение задачи в обучающем курсе, а сами задачи формируют модель предметной области. Модель стратегии обучения представлена в SQLTOR внутренним обучающим циклом, ответственным за формирование обратных связей в пределах одной задачи, а также внешним обучающим циклом, ответственным за принятие решения о том, какую задачу предлагать студенту следующей, основываясь на успешности решения текущей задачи. Для того, чтобы обеспечить работу внешнего обучающего цикла, было принято решение группировать подобные задачи в кластеры. Подобие задач, может быть оценено по КК, поставленным в соответствие SQL-запросам, являющимся решениями кластеризуемых задач. Вариант ожидаемой структуры ИКСО после ее создания показан на рис. 1.

Распределение запросов по кластерам (кластеризация задач) может оказаться достаточно трудоемким процессом, если в системе зарегистрировано большое количество заданий. В связи с этим возникла задача автоматизировать процесс кластеризации. РИКСО должна предоставить возможность изменять содержимое кластеров после автоматического распределения запросов, обеспечивая тем самым максимальную гибкость настройки обучающего курса. При кластеризации также следует учитывать тот факт, что кластеризованы могут быть только те запросы (а в результате и задачи), с которыми ассоциирован хотя бы один КК. Общий алгоритм подготовки учебного курса должен включать в себя этапы создания КК, задач, запросов, ассоциирование запросов с КК и задачами, создание обучающего курса, выбор задач для него и возможность их кластеризации (сначала автоматической, а затем ручной группировки). На рис. 2 показаны этапы разработки ИКСО при помощи SQLTOR. В прямоугольниках показаны возможные действия, стрелками обозначена последовательность этих действий, а в прямоугольниках с закругленными углами – ограничения, накладываемые на переходы.

Модель данных РИКСО показана в виде диаграммы сущностей-связей на рис.3. Кроме сущностей курсов, задач, запросов и КК, она включает сущность «пользователь», которая введена в схему для хранения информации о пользователях. Отметим, что каждый из запросов имеет своего автора в БД, что позволяет затем определить того, кто пер-

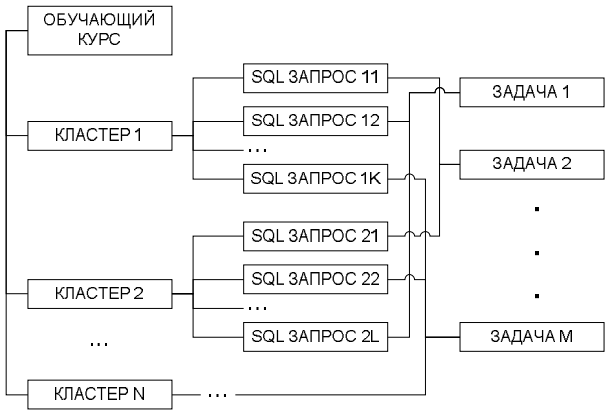

Рис. 1. Структура обучающего курса в SQLTOR

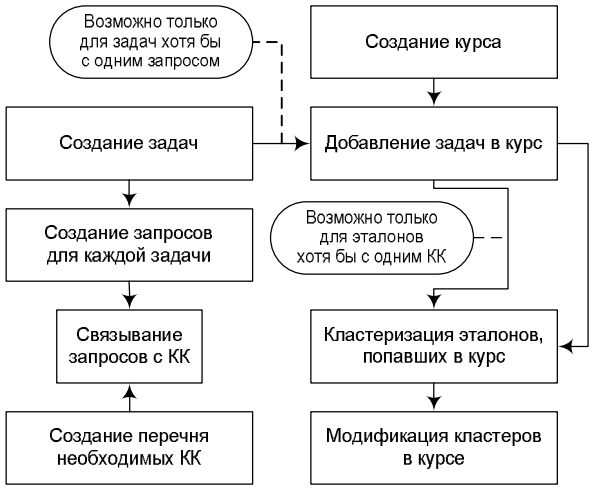

Рис. 2. Основные этапы разработки ИКСО SQL в РИКСО SQLTOR

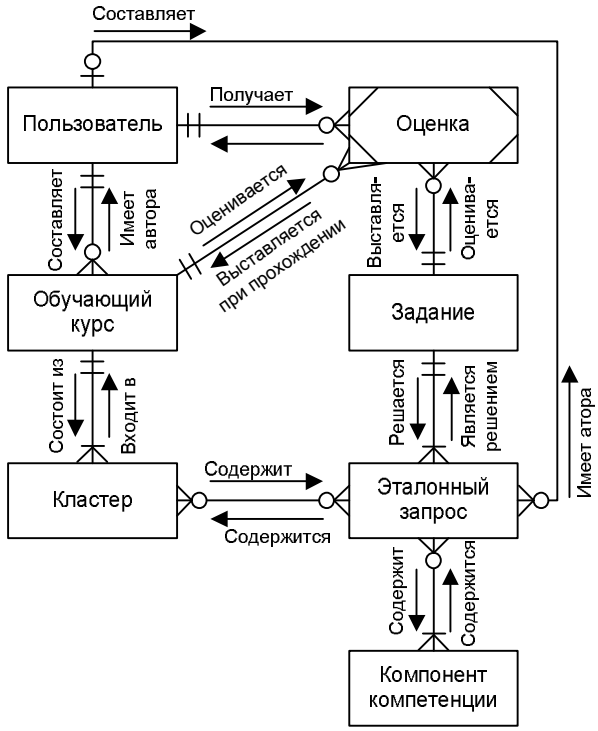

Рис. 3. ER-диаграмма базы данных

вым из пользователей добавил запрос в систему. Это удобно, когда новый запрос попадает в базу данных на этапе прохождения учебного курса, чтобы узнать, какому студенту удалось найти новый запрос, являющийся решением задачи, но еще не известный системе. Сущность «оценка» предназначена для хранения оценок, полученных студентами во время прохождения обучающего курса в ИКСО. Курс должен быть сформирован из набора кластеров, содержащих задания. Задания должны распределяться по кластерам в зависимости от того, в каких кластерах находятся запросы, ассоциированные с задачами. Нужно отметить, что это требование подразумевает возможность того, что одна и та же задача может попасть в разные кластеры. Запросы должны распределяться по кластерам в зависимости от содержащихся в них КК. Запросы с одинаковыми КК должны попасть в один кластер, отличающиеся одним КК – попасть в один кластер с большой вероятностью, подобные только одним компонентом – с малой вероятностью попасть в один и тот же кластер, не содержащие одинаковых КК – должны попасть в разные кластеры и т.п. Задачи в кластере должны быть отсортированы в нем по сложности, а сами кластеры – отсортированы в пределах курса (по средней сложности задач, входящих в них). Задача, вытекающая из такой постановки – предложить метод кластеризации запросов, отличающихся наборами КК, удовлетворяющий всем указанным требованиям.

Выбор и описание метода кластеризации

Для группировки задач в кластеры сначала воспользуемся кластеризацией эталонов (эталонных запросов), входящих в задания. В качестве средства решения был выбран алгоритм k-means (k-средних) [6,7]. Выбор этого метода обусловлен его относительной легкостью реализации, быстротой и возможностью использовать различные способы вычисления расстояния между кластеризуемыми объектами и шаблонами. В k-means группировка осуществляется при помощи сравнения данных с некоторыми шаблонами, которые ставят в соответствие каждой группе (кластеру). Шаблоны принято называть центроидами, так как обычно они вычисляются как центр тяжести всех точек, попавших в кластер. Будем использовать в дальнейшем термин «центроид» вместо «шаблон».

K-means обладает такими недостатками: неоднозначность начального выбора центроидов и необходимость явного указания количества ожидаемых кластеров [6,7]. Выбор количества кластеров предоставляется автору – его необходимо специфицировать в соответствующем поле ввода до начала процедуры кластеризации. В случае, если это количество окажется слишком большим, пустые кластеры (не содержащие запросов) не будут учтены. Если количество окажется слишком малым – все запросы будут распределены в этом количестве кластеров. При уменьшении количества ожидаемых кластеров возрастает вероятность получить в них более разнородные задания (эталонные запросы этих заданий будут сильно отличаться по наборам КК). Чтобы предотвратить такое распределение запросов, можно постепенно увеличивать количество ожидаемых кластеров до получения удовлетворительного распределения или вручную перераспределить запросы в кластерах, распределенных неверно из-за возникновения ошибок первого и второго рода.

Обозначим расстояния между запросами в кластере и его центроидом D_i^j. i – номер запроса; j – номер кластера. Как центроид так и запрос характеризуются набором КК. Тогда для характеристики каждого из запросов будем использовать кортеж булевых переменных $Q_i = \langle x_{ic} \rangle, c = \overline{1,n}$, где c – номер КК; n – число всех возможных КК в курсе. Булева переменная x_{ic} принимает истинное значение если c-тый КК ассоциирован с соответствующим i-тым запросом, в противном случае – принимает ложное значение. Для каждого центроида определим кортеж $C_j = \langle y_{jc} \rangle, c = \overline{1,n}$. Булевы переменные y_{jc} будут использоваться при определении расстояния между кластером и запросом. Расстояние (дистанцию) между i-тым запросом и j-тым центроидом запишем в виде функции $D_i^j = f_D\left(Q_i, C_j\right)$. Основываясь на значениях D_i^j, запросы помещаются в кластера по определенному правилу. Если обозначить j-тый кластер как множество Θ_j, то это правило можно записать так: $\left\{ Q_i \in \Theta_j \text{ если } \forall j' \neq j : D_i^j \leq D_i^{j'} \right\}$. На каждой итерации работы алгоритма требуется перезадавать центроиды C_j в соответствии с Θ_j. Введем функцию, при помощи которой вычисляется кортеж, характеризующий перезаданный центроид для каждого кластера: $C_j^* = f_C\left(\Theta_j\right)$.

Алгоритм кластеризации, используемый SQLTOR, отличается от известных способом вычисления функции $f_D\left(Q_i, C_j\right)$. Она должна возвращать значение, при помощи которого возможно оценить степень подобия Q_i и C_j. Для вычисления подобия двух кортежей можно использовать расстояние

Хемминга. Расстояние Хемминга в нашем случае представляет собой число, равное количеству несовпадающих элементов в двух сравниваемых кортежах, с учетом их положения. Рассмотрим ситуацию, когда сравниваются три запроса, представленных кортежами: $Q_1 = \langle 1,0,1,0 \rangle$, $Q_2 = \langle 1,0,1,1 \rangle$, $Q_3 = \langle 1,0,0,0 \rangle$ (в этой записи мы использовали обозначения «1», если булева переменная истинна и «0» – если ложна). Обозначим расстояния Хемминга между ними как h_{12}, h_{13}, h_{23} (для расстояний между Q_1 и Q_2, Q_1 и Q_3, Q_2 и Q_3 соответственно). Тогда из определения расстояния Хемминга следует, что $h_{12} = 1$, $h_{13} = 1$, $h_{23} = 2$. Заметим, что $h_{12} = h_{13}$. В случае если необходимо получить два кластера с центроидами $C_1 = Q_2$ и $C_2 = Q_3$ возникает неопределенность при соотнесении запроса Q_1 с ними. Для преодоления этой неопределенности условимся, что будем помещать запрос в равной степени похожий на центроиды двух и более кластеров в тот кластер, в котором на текущей итерации находится меньше эталонов. Чтобы это правило было учтено, увеличим значение расстояния Хемминга на величину $|\Theta_j| / |\{Q_i\}|$, где $|\Theta_j|$ – количество запросов в j-том кластере; $|\{Q_i\}|$ – общее количество запросов. Благодаря этой поправке эталоны должны распределяться более равномерно по кластерам. В случае, если количество эталонов в рассматриваемых кластерах окажется одинаковым, будем помещать эталон в произвольный кластер, так как в такой ситуации становится безразлично куда он будет помещен. Теперь можем записать выражение для вычисления дистанции D_i^j в виде:

$$D_i^j = f_D\left(Q_i, C_j\right) = h_{ij}\left(Q_i, C_j\right) + \frac{|\Theta_j|}{|\{Q_i\}|}, \qquad (1)$$

где h_{ij} – функция для определения дистанции Хемминга между i-тым запросом и j-тым центроидом.

Вторая особенность алгоритма кластеризации в SQLTOR связана с вычислением значения функции $f_C\left(\Theta_j\right)$. Каждая переменная кортежа $C_j^* = f_C\left(\Theta_j\right)$ y_{jc} должна принимать значение «истина» с вероятностью, прямопропорциональной числу переменных x_{ic} имеющих истинное значение и содержащихся во всех запросах $Q_i \in \Theta_j$. Разделив количество переменных $x_{ic} \in Q_i \in \Theta_j$ со значениями «1» $\sum_{x_{ic} \in Q_i \in \Theta_j} x_{ic}$ на количество запросов в кластере $|\Theta_j|$ получим частоту вхождения рассматриваемого c-того КК в запросы, содержащиеся в j-том кластере:

$$F_{jc} = \frac{\sum_{x_{ic} \in Q_i \in \Theta_j} x_{ic}}{|\Theta_j|}. \qquad (2)$$

Тогда будем присваивать значение «1» переменной y_{jc}, если частота F_{jc} превышает некоторую пороговую частоту F' и значение «0», если $F_{jc} \le F'$. В результате, можем записать выражение для вычисления C_j^*:

$$C_j^* = f_C\left(\Theta_j\right) = \left\langle f_C^*\left(\{x_{i1}\}\right), ..., f_C^*\left(\{x_{in}\}\right)\right\rangle,$$
$$f_C^*\left(\{x_{ic}\}\right) = \begin{cases} 1, & \text{если } F_{jc} > F', \\ 0, & \text{если } F_{jc} \le F', \end{cases} \quad x_{ic} \in Q_i \in \Theta_j. \qquad (3)$$

Значения, формирующие кортеж C_j^* определят на следующей итерации (если выполнение алгоритма не прекратится на текущей итерации) перераспределение запросов по кластерам. При этом возможны ошибки первого и второго рода – когда в j-тый кластер попадает запрос, который не должен был в него попасть и когда запрос, который должен был попасть в j-тый кластер, не попадает в него. Суммарное количество этих ошибок должно быть минимизировано. Увеличение числа ошибок первого и второго рода связано с ситуацией неопределенности, упомянутой выше, когда запрос может быть в равной степени соотнесен с двумя или более кластерами. Такая ситуация возникает, когда на какой-либо итерации кортежи, рассчитанные для двух или более кластеров, оказываются идентичными (состоят из одних и тех же значений в одном порядке). При этом, чаще всего на последующих итерациях центроиды, ассоциированные с этими кортежами, не перезадаются, что создает условия для возникновения новых ошибок первого и второго рода.

Экспериментальным путем было определено, что наименьшее количество ошибок первого и второго рода при кластеризации в SQLTOR достигается при значении $F' = 0,5$. При достаточно большом количестве КК, зарегистрированных в системе, алгоритм не давал идентичных центроидов для кластеров на этапе их перезадания. Принимая это во внимание, с учетом (3), можем записать:

$$f_C^*\left(\{x_{ic}\}\right) = \begin{cases} 1, & \text{если } F_{jc} > 0,5, \\ 0, & \text{если } F_{jc} \le 0,5, \end{cases} \quad x_{ic} \in \Theta_j. \qquad (4)$$

Неопределенность при инициализации центроидов относится к недостаткам алгоритма k-means, так как произвольный выбор начальных

центроидов, предлагаемый в основной формулировке алгоритма, приводит к различным результатам для каждого варианта этого выбора. Часто, для того чтобы снизить влияние этого недостатка, предлагают применить алгоритм для нескольких вариантов начального выбора центроидов, а затем усреднить результаты. Также, перед тем как использовать алгоритм k-means, может быть использован другой вид кластеризации (например, иерархическая кластеризация), а начальные центроиды выбраны на основании полученных результатов [7].

При кластеризации запросов наилучшим выбором начальных центроидов было бы такое их распределение, чтобы расстояние Хемминга между любыми двумя центроидами было максимальным.

На рис. 4 показан результат работы одной итерации предложенного алгоритма кластеризации, в случае если имеется набор начальных центроидов C_1, C_2, C_3, C_4, C_5 и набор запросов: $Q_1 = \langle 0,0,0,1 \rangle$, $Q_2 = \langle 0,0,1,0 \rangle$, $Q_3 = \langle 0,0,1,1 \rangle$, $Q_4 = \langle 0,1,0,0 \rangle$, $Q_5 = \langle 0,1,0,1 \rangle$, $Q_6 = \langle 0,1,1,0 \rangle$, $Q_7 = \langle 0,1,1,1 \rangle$, $Q_8 = \langle 1,0,0,0 \rangle$, $Q_9 = \langle 1,0,0,1 \rangle$, $Q_{10} = \langle 1,0,1,0 \rangle$, $Q_{11} = \langle 1,0,1,1 \rangle$, $Q_{12} = \langle 1,1,0,0 \rangle$, $Q_{13} = \langle 1,1,0,1 \rangle$, $Q_{14} = \langle 1,1,1,0 \rangle$, $Q_{15} = \langle 1,1,1,1 \rangle$. Необходимо заметить, что после автоматической кластеризации РИКСО «SQLTOR» позволяет перенастроить конфигурацию кластеров вручную. Задачи распределятся по кластерам автоматически в соответствии с привязанными к ним эталонами. Пример демонстрирует, что запросы, близкие по содержащимся в них КК, группируются в один и тот же кластер. Во второй итерации изменится лишь центроид C_3. Он примет вид $C_3^* = \langle 1,1,0,0 \rangle$. Распределение запросов останется неизменным. На третьей итерации изменений центроидов не произойдет, и, следовательно, алгоритм завершит свою работу.

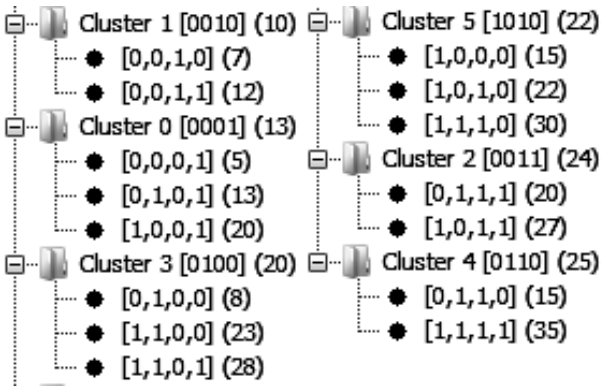

Рис. 4. Результаты работы k-means после первой итерации

Заметим, что при достаточно большом количестве КК, зарегистрированных в системе, становится возможным описать каждый запрос более точно (детализировать КК). В случае такой детализации кластеризация будет давать лучшие результаты, так как увеличится количество критериев (атрибутов) для сравнения близости (похожести) запросов, уменьшится вероятность получения центроидов, характеризующихся одинаковыми кортежами, а, следовательно, снизится суммарное количество ошибок первого и второго рода.

Сортировка кластеров и запросов

Для того чтобы обеспечить правильный порядок прохождения внешнего обучающего цикла, необходимо чтобы задания предлагались учащемуся в порядке возрастания их сложности при успешном выполнении и в порядке убывания сложности при ошибках. Другими словами, каждое последующее задание должно находиться в зоне ближайшего развития, предложенной Л.С. Выготским [8]. Очевидно, что указание сложности для каждого задания увеличит время составления обучающего курса. Поэтому было решено требовать у пользователя ввода сложностей только для КК (рис.5), так как их количество значительно меньше количества заданий.

Для запросов, задач и кластеров сложности рассчитываются автоматически. В случае необходимости автор может изменить значение автоматически рассчитанной сложности для кластера и запроса, нажав на соответствующую кнопку-переключатель (рис. 6).

Сложность запроса рассчитывается как суммарная сложность КК, входящих в него:

$$P_{Q_i} = \sum_{c=1}^{n} P_{x_{ic}}, \quad x_{ic} \in Q_i, \tag{5}$$

где P_{Q_i} – искомая сложность запроса Q_i;

$P_{x_{ic}}$ – сложность c-того компонента компетенции (если c-й КК не ассоциирован с запросом Q_i, то $P_{x_{ic}} = 0$).

Сложность задачи выбирается как максимальная сложность запроса из ассоциированных с ней:

$$P_T = \max_{Q_i \in T} P_{Q_i}, \tag{6}$$

где P_T – сложность задачи T; P_{Q_i} – сложность запроса Q_i, являющегося решением задачи T.

После расчета сложностей запросов они располагаются в порядке ее возрастания в пределах кластера. Благодаря этому стало возможным организовать внешний обучающий цикл следующим образом. При переходе обучающей последовательности

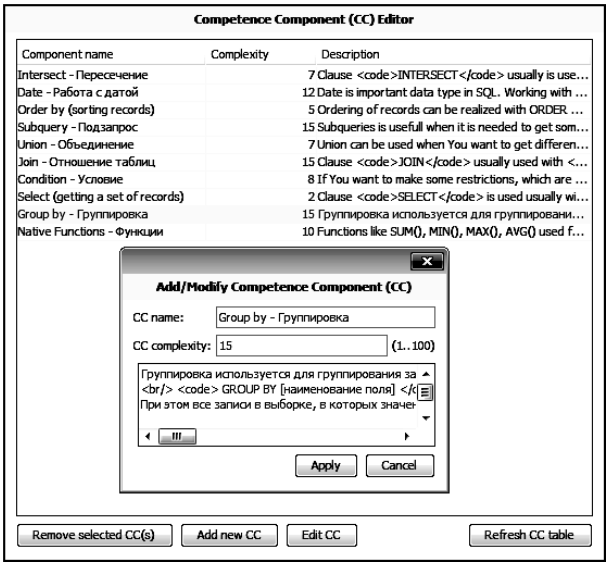

Рис. 5. Графический интерфейс редактора КК в SQLTOR

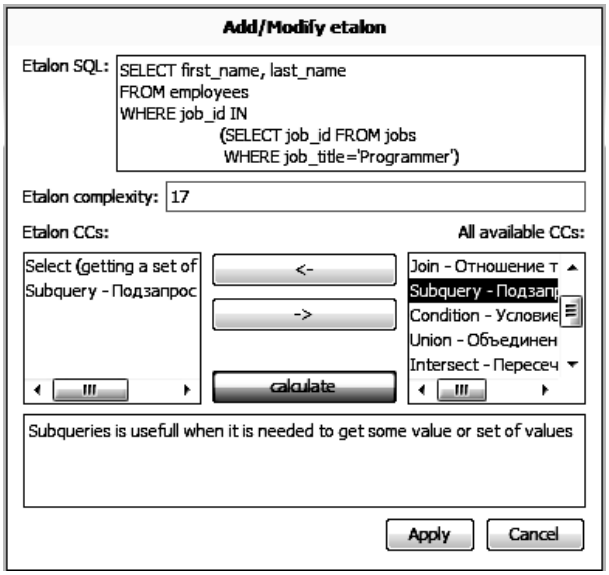

Рис. 6. Графический интерфейс диалога добавления/редактирования запроса в SQLTOR

[3] в очередной кластер, студенту будет предложена задача со средней сложностью. При успешном ее решении учащемуся будут предложены задачи из того же кластера с большей сложностью. Если же учащийся не справился с задачей, получив все возможные подсказки, то ему предлагается выполнить более простую задачу из того же кластера. После успешного решения самой сложной задачи в кластере, обучающая последовательность перейдет в следующий по порядку кластер. Если же учащийся дошел до самой простой задачи в кластере (не смог решить большинство из предложенных ему задач), тогда ему предоставляется пример, в котором разбирается задача, подобная задачам в кластере.

В качестве разбираемой задачи система автоматически устанавливает самую простую задачу в кластере (совокупность условия задачи и одного из запросов, зарегистрированных в системе как ее решение), однако автор может в любой момент изменить пример задачи для любого кластера.

Сложность кластеров должна быть рассчитана для того, чтобы расположить их в порядке ее возрастания. Это позволит при обучении переходить от групп более простых к группам более сложных задач. Сложность кластера рассчитывается как средняя сложность запросов, входящих в него:

$$P_{C_j} = \sum_i P_{Q_i} \Big/ N_j, \quad Q_i \in \Theta_j. \qquad (7)$$

Сложность кластера – показатель, на основании которого кластеру присваивается порядковый номер, указывающий на то, когда именно задачи из него будут предложены учащемуся. Порядковый номер кластера может быть изменен автором вручную. Задачи из кластера с меньшим порядковым номером будут предложены учащемуся раньше, а с большим – позже. Фактически, в SQLTOR сложность кластера учитывается только при определении его порядкового номера и никак не учитывается при оценивании студенческого решения или при переходах в рамках внешнего цикла в явном виде. Более того, на этапе создания обучающего курса, автор может изменить порядок кластеров без учета их сложностей. В связи с этим, расчет сложности для кластера следует рассматривать как дополнительную возможность системы. Графический интерфейс редактора обучающего курса представлен на рис. 7.

Заключение

Разработанная авторами РИКСО «SQLTOR» предназначена для облегчения работы преподавателей при подготовке и проверке обучающих курсов по языку запросов к базам данных SQL. В статье описана одна из функциональных особенностей системы, позволяющая распределить задачи в курсе по кластерам таким образом, чтобы подобные задачи попали в одну группу (кластер). В процессе кластеризации задачи также сортируются по возрастанию их сложности внутри кластера, в который попали. Для решения задачи кластеризации был использован алгоритм k-means. В качестве функции вычисления дистанции, при помощи которой оценивалось подобие SQL-запросов, использовалось модифицированное расстояние Хемминга (1). Такой подход обеспечил снижение трудовых и временных затрат на подготовку обучающего курса.

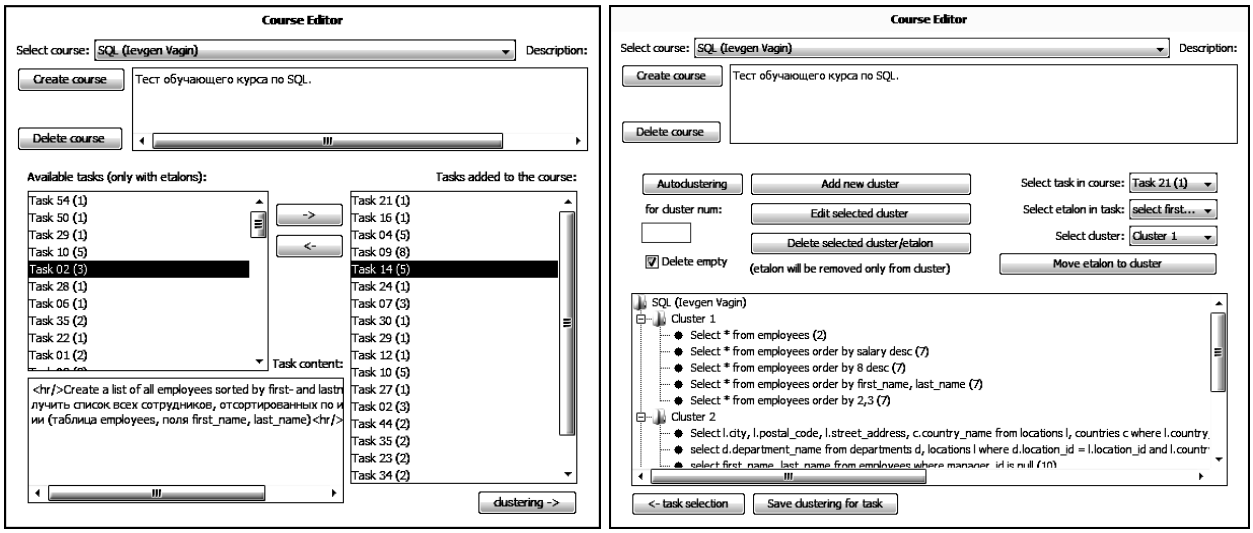

Рис. 7. Графический интерфейс редактора обучающего курса в SQLTOR

Пример кластеризации, приведенный в статье, показывает, что в результате кластеризации запросов они группируются по принципу схожести КК. Также был предложен способ выбора начальных центроидов, позволяющий получить равномерное распределение запросов по кластерам. Были затронуты некоторые вопросы, касающиеся архитектуры ИКСО, создаваемые в SQLTOR. В частности – принцип работы внешнего цикла при прохождении обучающего курса. Были предложены методы оценки сложности задач по запросам и сортировки кластеров по мере возрастания сложности, входящих в них задач. Перспективной задачей является усовершенствование РИКСО SQLTOR, в том числе, добавление возможности кластеризовать модели учащихся на основании результатов прохождения входного теста знаний (уже предусмотрен в SQLTOR). Это даст возможность выбирать тот уровень сложности, который бы подошел каждому отдельному студенту. Планируется разработать алгоритмы автоматического определения КК для конкретных предметных областей, структуру данных для хранения модели студента, что позволит более обоснованно принимать решения в рамках внутреннего и внешнего обучающего циклов.

В настоящее время ведутся работы по созданию web-версии РИКСО SQLTOR.

Литература

1. VanLehn, K. The behavior of tutoring Systems [Текст]/ K. VanLehn // International Journal Of Artificial Intelligence In Education. – Athens (Greece). – 2006. – №16(3). – P. 227-265.

2. Self, J. The defining characteristics of intelligent tutoring systems research: ITSs care, precisely [Текст] / J. Self // International Journal Of Artificial Intelligence in Education. – 1999. – Vol. 10. – P. 350-364.

3. Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning [Текст]. – Beverly Park Woolf. Department of Computer Science, University of Massachusetts. – Amherst (USA). – 2008. – P. 480.

4. Murray, T. An overview of intelligent tutoring system authoring tools: Updated Analysis of the State of the Art [Текст] / T. Murray, S. Blessing, S. Ainsworth // In Authoring Tools for Advanced Technology Learning Environments. Kluwer Academic Publishers. – Dordrecht (Netherlands) – Dec. 31, 2003. – P. 491-497.

5. Универсальная среда создания и трансляции интеллектуальных обучающих программ [Текст] / А.С. Кулик, А.Г. Чухрай, С.И. Педан, П. Анценбергер // Интеллектуальные системы принятия решений и проблемы вычислительного интеллекта: Материалы международной научной конференции. – Т. 1 – Херсон: ХНТУ, 2009. – 189-192 с.

6. Jain, A.K. Data clustering: A review [Текст]/ A.K. Jain, M.N. Murty, P.J. Flynn // ACM Computing Surveys. –1999. – №31(3). – P. 264-323.

7. Tan, Pang-Ning Introduction to data mining [Текст]/ Pang-Ning Tan, M. Steinbach, V. Kumar. – Addison-Wesley Longman Publishing Co., Inc. – Boston (USA). – 2005. – P. 769.

8. Выготский, Л.С. Психология развития человека [Текст] / Л.С. Выготский. – М.: Смысл; ЭКСМО, 2005. – 1136 с.

Поступила в редакцию 20.05.2011

Рецензент: д-р техн. наук, зав. каф. информатики А.Ю. Соколов, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.

ЗАСТОСУВАННЯ МЕТОДУ K-MEANS ДЛЯ ФОРМУВАННЯ ЗОВНІШНЬОГО ЦИКЛУ НАВЧАННЯ КОМП'ЮТЕРНИХ НАВЧАЛЬНИХ ПРОГРАМ З SQL

А.Г. Чухрай, Є.С. Вагін, З.В. Томченко, П. Анценбергер

Запропонований метод кластеризації завдань в комп'ютерних системах навчання для вирішення проблеми організації зовнішнього учбового циклу при вивченні SQL. Вирішення розглянутої задачі відноситься до кластерного аналізу, зокрема – до використання алгоритму k-means (k-середніх). Розглянуто один з підходів до кластеризації навчальних завдань, що базується на компонентах компетенції. Запропоновано метод ранжування кластерів і завдань за їх складністю. Підіймаються питання, що пов'язані з розробкою архітектури редакторів інтелектуальних комп'ютерних систем навчання, що виникли під час розробки редактора SQLTOR. Використання запропонованих рішень в SQLTOR забезпечує зниження трудових витрат та витрат часу спеціаліста з предметної області (викладача) під час складання комп'ютерного навчального курсу SQL. Сформульовані основні перспективні шляхи розвитку редактору SQLTOR.

Ключові слова: редактор інтелектуальних комп'ютерних систем навчання, навчання SQL, кластеризація, k-середніх, відстань Хемінга, компоненти компетенції в навчанні.

K-MEANS METHOD OF OUTER TUTORING LOOP GENERATION APPLICATION FOR COMPUTER TUTORING PROGRAMS ON SQL

A.G. Chukhray, Ie.S. Vagin, Z.V. Tomchenko, P. Anzenberger

A method of tasks clustering in computer SQL tutoring systems for outer tutoring loop organizing purposes is proposed. Solution of the problem relates to the cluster analysis, in particular – to the k-means algorithm application. One of the approaches based on competence components sets difference for the clustering of tutoring tasks is considered. A method of clusters and tasks sorting based on their complexity comparing is proposed. Some problems of authoring tools for intelligent tutoring systems architecture development are analized by SQLTOR development process example. Usage of solutions implemented in SQLTOR reduces manual and time efforts of a domain specialist (a teacher) during SQL tutoring course compiling. Main directions of future SQLTOR development are given.

Key words: Authoring tool for ITS, SQL learning, clustering, k-means, Hamming distance, competence components in tutoring.

Чухрай Андрей Григорьевич – канд. техн. наук, доцент, докторант кафедры систем управления летательных аппаратов, Национальный аэрокосмический университет им. Н.Е. Жуковского «Харьковский авиационный институт», Харьков, Украина, e-mail: achukhray@gmail.com.

Вагин Евгений Сергеевич – аспирант кафедры систем управления летательных аппаратов, Национальный аэрокосмический университет им. Н.Е. Жуковского «Харьковский авиационный институт», Харьков, Украина, e-mail: TzipTwork@gmail.com.

Томченко Захар Владимирович – инженер кафедры систем управления летательных аппаратов, Национальный аэрокосмический университет им. Н.Е. Жуковского «Харьковский авиационный институт», Харьков, Украина, e-mail: Rigveda@mail.ru.

Анценбергер Петер – аспирант кафедры систем управления летательных аппаратов, Национальный аэрокосмический университет им. Н.Е. Жуковского «Харьковский авиационный институт», Харьков, Украина; преподаватель Университета прикладных наук Верхней Австрии, Штайер, Австрия.