Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: “Методы и системы искусственного интеллекта”

Лабораторная работа №4

“ЕЯ ДОСТУП К БАЗЕ ДАННЫХ НА ОСНОВЕ АЛГОРИТМА

СОПОСТАВЛЕНИЯ С ОБРАЗЦОМ”

Выполнил:

ст.гр. ИС/б-17-2

Волобуев Ю.С.

Проверила:

Сметанина Т.И.

Севастополь

2020

1 ЦЕЛЬ РАБОТЫ

Исследование алгоритма сопоставления с образцом и особенностей его применения для формирования запросов к базам данных, а также для организации доступа к базам данных на ограниченном подмножестве естественного языка.

2 ПОСТАНОВКА ЗАДАЧИ

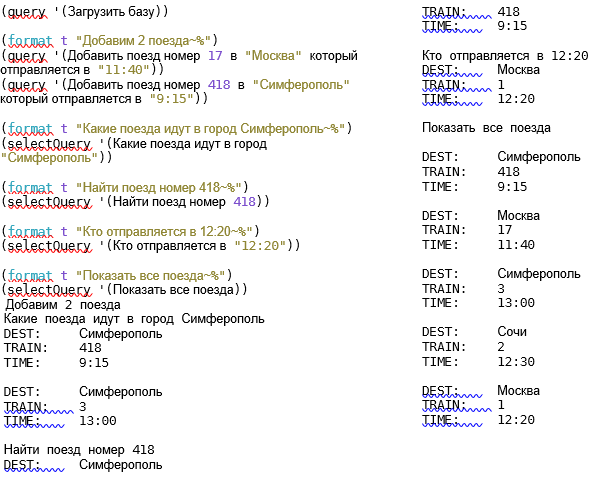
Для базы данных, созданной в лабораторной работе 3, необходимо написать на языке Лисп интерфейс, который позволяет выполнять ЕЯ-запросы с помощью алгоритма сопоставления с образцом. Кроме запроса, заданного по варианту задания, предусмотреть 5-6 различных дополнительных запросов.

3 ХОД РАБОТЫ

Код функций:

|  |
| --- |
| (defvar \*db\* nil) |
|  |
| (*defun* insert (dest train time) |
| (push (list :dest dest :train train :time time) \*db\*) |
| ) |
|  |
| (*defun* savef (filename) |
| (with-open-file (out filename :direction :output :if-exists :supersede) |
| (with-standard-io-syntax |
| (print \*db\* out) |
| ) |
| ) |
| ) |
|  |
| (*defun* loadf (filename) |
| (with-open-file (in filename) |
| (with-standard-io-syntax |
| (setf \*db\* (read in)) |
| ) |
| ) |
| ) |
|  |
| (*defun* select\* () |
| (format t "~%") |
| (format t "~{~{~a:~10t~a~%~}~%~}" \*db\*) |
| ) |
|  |
| (*defun* where(&key dest train time) |
| #'(lambda (row) |
| (and |
| (if dest  (equal (getf row :dest)  dest)  t) |
| (if train (equal (getf row :train) train) t) |
| (if time  (equal (getf row :time)  time)  t) |
| ) |
| ) |
| ) |
|  |
| (*defun* update (where-func &key dest train time) |
| (setf \*db\* |
| (mapcar |
| #'(lambda (row) |
| (when (funcall where-func row) |
| (if dest  (setf (getf row :dest)  dest)) |
| (if train (setf (getf row :train) train)) |
| (if time  (setf (getf row :time)  time)) |
| ) |
| row |
| ) |
| \*db\* |
| ) |
| ) |
| ) |
|  |
| (*defun* selectByDest (\_dest) |
| (if (eq nil (setf rows (remove-if-not #'(lambda (row) (equal (getf row :dest) \_dest)) \*db\*))) |
| (print "Поезда туда не ездят") |
| (print rows) |
| ) |
| ) |
|  |
| (*defun* match (p d) |
| (cond |
| ;;правило 1 |
| ((and (null p) (null d)) t) |
| ;;правило 2 |
| ((and (null d) |
| (eq (car p) '$) |
| (null (cdr p))) t) |
| ;;один из списков исчерпан |
| ((or (null p) (null d)) nil) |
| ;;правило 3 и правило 4 |
| ((or (equal (car p) '?) |
| (equal (car p) (car d))) |
| (match (cdr p) (cdr d))) |
| ;;правило 5 и 6 |
| ((eq (car p) '$) |
| (cond ((match (cdr p) d) t) |
| ((match p (cdr d)) t))) |
| ;;правило 7 - сопоставление списков,включающих подсписки |
| ((and (not (atom (car p))) |
| (not (atom (car d))) |
| (match (car p) (car d))) |
| (match (cdr p) (cdr d)) ) |
| ;;правило 8 – подстановка значения в переменную |
|  |
| ((and (atom (car p)) |
| (eq (car-letter (car p)) #\?) |
| (match (cdr p)(cdr d))) |
| (set (cdr-name (car p)) (car d)) t) |
| ;;правило 9 - подстановка сегмента значений в переменную |
|  |
| ((and (atom (car p)) |
| (eq (car-letter (car p)) #\$)) |
| (cond ((match (cdr p)(cdr d)) |
| (set (cdr-name (car p)) (list (car d))) |
| t) |
| ((match p (cdr d)) |
| (set (cdr-name (car p)) |
| (cons (car d)(eval (cdr-name (car p))))) |
| t))) |
| ;; правило 10 - обработка пакета ограничений, если в пакете есть «?» |
| ((and (not(atom (car p))) |
| (eq (caar p) 'restrict) |
| (eq (cadar p) '?) |
| (and-to-list |
| (mapcar #'(lambda (pred) |
| (funcall pred (car d))) (cddar p)))) |
| (match (cdr p)(cdr d))) |
| ;; правило 11 - обработка пакета ограничений, если в пакете есть «?V» |
| ;; например: (match '((restrict ?V integerp evenp) b c) '(36 b c)) |
| ((and (not (atom (car p))) |
| (not (atom d)) |
| (eq (caar p) 'restrict) |
| (eq (car-letter (cadar p)) #\?) |
| (and-to-list |
| (mapcar #'(lambda (pred) |
| (funcall pred (car d))) (cddar p))) |
| (match (cdr p)(cdr d))) |
| (set (cdr-name (cadar p)) (car d)) |
| t) |
| )) |
| ;;;;<2. Вспомогательные функции |
| ;;; выделение первой литеры из имени |
| (*defun* car-letter (x) (if (not (numberp x)) (car (coerce (string x) 'list)))) |
| ;;; возвращает имя без первой |
| (*defun* cdr-name (x) |
| (intern (coerce (cdr (coerce (string x) 'list)) 'string))) |
| ;;; проверяет, все ли элементы списка lis имеют значение T |
| (*defun* and-to-list ( lis ) |
| ;lis - список логических значений |
| (let ((res t)) |
| (dolist (temp lis res) |
| (setq res (and res temp))))) |
|  |
| (*defun* get-matches (p database) |
| (remove-if-not #'(lambda (record) (match p record)) database) |
| ) |
|  |
| (*defun* query (q) |
| (cond |
| ((match `($ загрузить $) q) |
| (loadf "test.txt")) |
| ((match `($ сохранить $) q) |
| (savef "test.txt")) |
| ((match `(Добавить $ номер ?train в $ ?dest $ отправляется $ ?time) q) |
| (insert dest train time)) |
| ((match `($ город $ ?dest) q) |
| (setf temp (get-matches `($ :dest ,dest $) \*db\*)) |
| (if (null temp) "Поезда туда не едут" temp)) |
| ((match `($ номер $ ?train) q) |
| (setf temp (get-matches `($ :train ,train $) \*db\*)) |
| (if (null temp) "Поездов с таким номером нет" temp)) |
| ((match `($ отправляется $ ?time) q) |
| (setf temp (get-matches `($ :time ,time $) \*db\*)) |
| (if (null temp) "В такое время никто не едет" temp)) |
| ((match `($ все $) q) |
| (select\*)) |
| ) |
| ) |
|  |
| (*defun* selectQuery (q) |
| (setf temp (query q)) |
| (if (listp temp) |
| (format t "~{~{~a:~10t~a~%~}~%~}" temp) |
| (format t "~a~%" temp) |
| ) |
| ) |

Результат работы:



ВЫВОД

В ходе работы был исследован алгоритм сопоставления с образцом и особенности его применения для формирования запросов к базам данных, организован доступ к базе данных с применением ограниченного подмножества естественного языка.