МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное Государственное Бюджетное Образовательное учреждение высшего профессионального образования

«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

(ФГБОУ ВПО «КубГУ»)

Кафедра математического моделирования

КУРСОВАЯ РАБОТА

Прототип интерфейса пользователя для системы трансляции заданий с естественного языка в SQL

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бабичев М.А.

(подпись, дата)

Факультет\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_курс\_\_\_\_\_\_\_\_\_\_\_\_\_

Специальность/направление\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Научный руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бессарабов Н.В.

учен. степень, должность, Подпись Дата

Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Рубцов С. Е.

учен. степень, должность, Подпись Дата

Краснодар 2015

РЕФЕРАТ

Курсовая работа 30 с., 5 рис., 4 источников, 8 приложения.

ORM МОДЕЛЬ, TOMITA-ПАРСЕР, PHP, JAVASCRIPT, MYSQL, SQL, HTML, CSS, BOOTSTRAP, FONT-AWESOME, PSR-0, PSR-1, PSR-2, CSP, PHPMYADMIN.

Объектом исследования является Tomita-парсер и создание взаимодействия с ним через веб-интерфейс.

Цель работы:

* построение сервера взаимодействия с Tomita-парсером;
* разработка базы данных в СУБД MySQL;
* реализовать ORM модели для взаимодействия с объектами;
* реализация Web-интерфейса доступа к данным.

Исследования проводились с помощью стандартов PSR-0, PSR-1, PSR-2, технологии CSP в СУБД MySQL и технологий: PHP 5.3, JavaScript, HTML5, CSS3, Bootstrap 3, Font-awesome 4, PHPixie2, apache2, jQuery.

С использованием указанных стандартов и технологии:

* построен сервер взаимодействия с Томита-парсером;
* создана база данных в СУБД MySQL;
* созданы модели ORM для взаимодействия с объектами;
* реализован Web-интерфейс доступа к данным.

Результаты работы могут быть использованы для автоматизации обучения студентов языку SQL.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc428799914)

[1 Постановка задачи 5](#_Toc428799915)

[2 Взаимодействие с Томита-парсером 7](#_Toc428799916)

[3 jVision фреймворк и взаимодействие с PHP 9](#_Toc428799917)

[4 Обработчик построенный на phpMorphy 10](#_Toc428799918)

[ЗАКЛЮЧЕНИЕ 11](#_Toc428799919)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 12](#_Toc428799920)

[ПРИЛОЖЕНИЕ A 13](#_Toc428799921)

[ПРИЛОЖЕНИЕ Б 15](#_Toc428799922)

[ПРИЛОЖЕНИЕ В 22](#_Toc428799923)

[ПРИЛОЖЕНИЕ Г 24](#_Toc428799924)

# ВВЕДЕНИЕ

Задача трансляции инструкции с естественного (русского) языка на язык SQL.

Часть необходимых преобразований выполняется тривиально. Это перевод некоторых слов или словосочетаний естественного языка в термины SQL или термины схемы базы. Понятно, что эти действия осуществимы при наличии словаря SQL и словаря схем базы данных.

Проблема трансляции с ЕЯ в SQL разрабатывалась давно, но даже в таких серьезных работах, как [1] рассматривается упрощённый формат фразы естественного языка.

Существует подход, в котором формируется так называемое управляемое подмножество ЕЯ ограничивает допустимую структуру фраз языка. В настоящей работе этот подход не рассматривается.

Основные проблемы возникают из-за не определеннозначности естественного языка. В распространённых формулировках заданий, как минимум используются эллипсис и анафора.

Предлагаемая работа ориентирована на создание графического интерфейса программы для трансляции задания с ЕЯ в язык SQL, в которой предоставляет интерфейс к Томита-парсеру. В частности, решена задача повышения быстродействия системы интерфейс Томита-парсер.

# Постановка задачи

Создается сервис для преобразования заданий с естественного языка (ЕЯ) в специализированный декларативный язык SQL.

Существуют словари, SQL схемы c их помощью осуществима работа тривиальных преобразований. Трансляции более сложных заданий, использующих: эллипсис, анафора, должны выполняться выполнятся при помощи парсера, использующего грамматики, или наборов правил схем.

Желательно получить время реагирования меньше секунды для тривиальных преобразований. Для запросов с подзапросами до второго уровня время реагирования должно составлять не более секунды, для второго и третьего не более секунды.

Предполагается что разработанная программа будет использована как для обучения языку SQL, так и в качестве рабочего инструмента. Интерфейс должен обеспечить удобную работу пользователя (UX). Так же в интерфейсе должна быть возможность работать с подзапросами. Подзапросы должны поддерживать три уровня вложенности.

Томита-парсер – создан для извлечения структурированных данных из текста на естественном языке. Вычленение фактов происходит при помощи контекстно-свободных грамматик и словарей ключевых слов.

Существует словарь ключевых слов и грамматика определенной базы данных. Именно для неё и будем решать задачу, используя Томита–парсер.

Взаимодействие с Томита-парсером должно осуществляться при помощи языка PHP. Предполагается что результатом взаимодействия с парсером будет файл формата JSON, для удобного взаимодействия с ним в дальнейшем.

# Взаимодействие с Томита-парсером

Взаимодействие PHP и Томита-парсера будет построено на ожидании завершения процесса. Когда создается процесс, он получает уникальный индекс в бытовых ОС, в случае с RTOS [Приложение А] будем перехватывать указатель. Возможны ситуации, когда Томита-парсер не сможет уложиться в отложенное для него время.

Например, прислали обработчику слишком много данных. Тогда у пользователя должно появиться диалоговое окошко, см. Рисунок 1, в котором предоставляется выбор пользователю. Закрыть сессию (остановить) и продолжить работать с данными, или ждать ответ.

В случае, если в течение пяти секунд ответа от сервера не поступило, автоматически соединение с пользователем закрывается, но процесс продолжает работать. Если пользователь снова отправит эти данные, он получит индекс старого процесса, если он еще не завершился.

Если процесс закончил работу, данные кэшируются и в течение суток получить эти данные можно мгновенно, без обработки и Томита-парсера. Если в течение часа преобладают одинаковые запросы, они кэшируются на сутки. Кэш рассчитан на 128 MB.

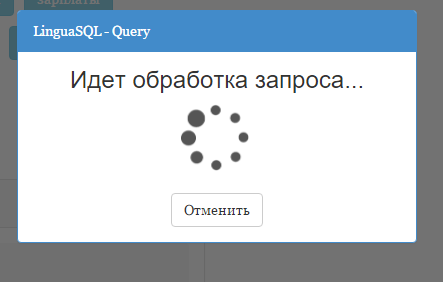


Рисунок 1 – ожидание запроса

Процесс работающий более трех минут закрывается и уже данные передаются обработчику, построенному на phpMorphy (библиотека морфологического анализа, реализованная на платформе PHP), он позволяет решать следующие задачи:

* Лемматизация (получение нормальной формы слова)
* Получение всех форм слова
* Получение грамматической информации для слова (часть речи, падеж, спряжение и т.д.)
* Изменение формы слова в соответствии с заданными грамматическими характеристиками
* Изменение формы слова по заданному образцу

Исходный код вынесен в приложение Б.

# jVision фреймворк и взаимодействие с PHP

Основным инструментом работы и динамических изменений на странице является DOM (Document Object Model) — объектная модель, используемая для XML/HTML-документов.

Согласно DOM-модели, документ является иерархией, деревом. Каждый HTML-тег образует узел дерева с типом «элемент». Вложенные в него теги становятся дочерними узлами. Для представления текста создаются узлы с типом «текст».

Интерфейс фреймворка jVision [6] вынесен в приложение В.

jVision позволяет быстро работать с DOM-деревом. С его помощью происходит обработка событий, всплывающих окошек, анимации. Так же jVision поддерживает взаимодействие с PHP при помощи ajax технологии. Прошлая версия сайта работала на jQuery, из-за большого размера объекта, сервис терял скорость. Был создан фреймворк jVision, который решил проблему отзывчивости в шестнадцать раз.

Код взаимодействия пользователя с сервером вынесен в приложение Г.

# Запросы с подзапросами

Создание запросов с подзапросами реализуется с помощью дерева. Пользователь, в ручном режиме, с помощью интерфейса (UX) собирает дерево, которое отправляет в обработчик. Используя SQL схему, обработчик, осуществляет трансляцию из ЕЯ в язык SQL.

# ЗАКЛЮЧЕНИЕ

Результатом данной работы являются прототип системы обучения. Система может быть применена как в обучении, так и в профессиональной сфере.

Изучены технологии: PHPixie, Bootstrap, jQuery, Haml, формат архитектуры приложения PSR–0.

Реализован программный интерфейс к парсеру, открывающий доступ к взаимодействию с ним по средствам технологий Hypertext Preprocessor и Ajax.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Найханова, Л.В. МЕТОДЫ И АЛГОРИТМЫ ТРАНСЛЯЦИИ ЕСТЕСТВЕННО–ЯЗЫКОВЫХ ЗАПРОСОВ К БАЗЕ ДАННЫХ В SQL / Л.В. Найханова, И.С. Евдокимова –ЗАПРОСЫ. ВСГТУ, 2004.
2. Бессарабов Н.В. Реализация графовых моделей данных и знаний в Cache / Н.В. Бессарабов, А.Г. Коблов – Экологический вестник научных центров ЧЭС, 2006.
3. Бессарабов Н.В. Гипотеза. Сепира–Уорфа и инвариантные структуры данных / Н.В. Бессарабов – Обозрение прикладной и промышленной математики, 2002.
4. Селко Д. Программирование на SQL для профессионалов / Д. Селко – М.: Издательство "Лори", 2004.
5. Документация к фреймворку PHPixie [электронный ресурс] URL: http://phpixie.com/2.x.html (дата обращения 11.03.2015).
6. Документация к фреймворку jVision [электронный ресурс] URL: http://rez1dent3.github.io/jVision/ (дата обращения 17.05.2015).

# ПРИЛОЖЕНИЕ A

Система реального времени (СРВ) — это система, которая должна реагировать на события во внешней по отношению к системе среде или воздействовать на среду в рамках требуемых временных ограничений. Оксфордский словарь английского языка говорит об СРВ как о системе, для которой важно время получения результата. Другими словами, обработка информации системой должна производиться за определённый конечный период времени, чтобы поддерживать постоянное и своевременное взаимодействие со средой. Естественно, что масштаб времени контролирующей системы и контролируемой ей среды должен совпадать.

Под реальным временем понимается количественная характеристика, которая может быть измерена реальными физическими часами, в отличие от логического времени, определяющего лишь качественную характеристику, выражаемую относительным порядком следования событий. Говорят, что система работает в режиме реального времени, если для описания работы этой системы требуются количественные временные характеристики.

Операционная система, ОС — комплекс взаимосвязанных программ, предназначенных для управления ресурсами вычислительного устройства и организации взаимодействия с пользователем.

В логической структуре типичной вычислительной системы операционная система занимает положение между устройствами с их микроархитектурой, машинным языком и, возможно, собственными (встроенными) микропрограммами (драйверами) — с одной стороны — и прикладными программами с другой.

Разработчикам программного обеспечения операционная система позволяет абстрагироваться от деталей реализации и функционирования устройств, предоставляя минимально необходимый набор.

Операционная система реального времени, ОСРВ — тип операционной системы, основное назначение которой — предоставление необходимого и достаточного набора функций, обеспечивающих разработку программными средствами систем реального времени на конкретном аппаратном оборудовании.

# ПРИЛОЖЕНИЕ Б

protected function singular($str)

{

$regexes = array(

'/^(.\*?us)$/i' => '\\1',

'/^(.\*?[sxz])es$/i' => '\\1',

'/^(.\*?[^aeioudgkprt]h)es$/i' => '\\1',

'/^(.\*?[^aeiou])ies$/i' => '\\1y',

'/^(.\*?)s$/' => '\\1',

);

foreach ($regexes as $key => $val)

{

$str = preg\_replace($key, $val, $str, -1, $count);

if ($count)

{

return $str;

}

}

return $str;

}

protected function plural($str)

{

$regexes = array(

'/^(.\*?[sxz])$/i' => '\\1es',

'/^(.\*?[^aeioudgkprt]h)$/i' => '\\1es',

'/^(.\*?[^aeiou])y$/i' => '\\1ies',

);

foreach ($regexes as $key => $val)

{

$str = preg\_replace($key, $val, $str, -1, $count);

if ($count)

{

return $str;

}

}

return $str.'s';

}

public function weight($word)

{

$profile = [

'С' => 3, 'Г' => 3,

'П' => 2, 'Н' => 2,

'ПРЕДЛ' => 0, 'СОЮЗ' => 0,

'МЕЖД' => 0, 'ВВОДН' => 0,

'ЧАСТ' => 0, 'МС' => 0

];

$parts\_of\_speech = $this->morphy->get('ru')->getPartOfSpeech($word);

if (!$parts\_of\_speech)

return 1;

$range = [];

$index = 0;

foreach($parts\_of\_speech as $word => $speech) {

$range[$index] = 1;

if (is\_array($speech)) {

if (count($speech)) {

$temp\_speech = [current($speech)];

while (next($speech)) {

$key = current($speech);

if (isset($profile[$key]))

$temp\_speech[] = $profile[$key];

}

$range[$index] = max($temp\_speech);

}

}

else {

if (isset($profile[$speech])) {

$range[$index] = $profile[$speech];

}

}

}

return max($range);

}

public function get\_rating\_reps($words)

{

$words = array\_count\_values($words);

arsort($words);

return $words;

}

public function get\_rows($string, $tables = [])

{

$\_rows = [];

$indices = $this->get\_indices($string, $tables);

foreach($indices as $key => $row) {

$table = $this->get\_table($row->table\_id);

if ($table) {

$\_rows[$table][$key] = $row->table\_index;

}

}

$rows = [];

foreach ($\_rows as $table\_name => $\_row) {

$ids = array\_values($\_row);

foreach ($ids as &$id)

$id = "'$id'";

$ids = implode(',', $ids);

$fields = $this->pixie->db->expr("FIELD(id, " . $ids . " )");

$ids = $this->pixie->db->expr('(' . $ids . ')');

$\_\_rows = $this->pixie->orm->get($table\_name)

->where('id', 'IN', $ids)

->order\_by($fields)

->find\_all()

->as\_array(true);

$keys = array\_keys($\_row);

foreach ($\_\_rows as $row) {

$key = current($keys);

next($keys);

$row->\_\_table = $table\_name;

$rows[$key] = $row;

}

}

ksort($rows);

return $rows;

}

public function get\_words($string)

{

$string = htmlspecialchars\_decode($string);

// FIXME: Дополнительный пробел "перед" тегом, для <p>hello</p>world

$string = preg\_replace('/([<].\*?[>])/', ' $1 ', $string);

$string = strip\_tags($string);

$string = preg\_replace('/[^а-я\w-\s]/iu', ' ', $string);

$string = preg\_replace('/\_/iu', ' ', $string);

$string = preg\_replace('/[\s\t\n\r]-[\s\t\n\r]/', ' ', $string);

$string = preg\_replace('/ё/iu', 'Е', $string);

$string = preg\_replace('/[\s\t\n\r]+/', ' ', $string);

$string = preg\_replace('/[-]{2,}/', '-', $string);

$string = mb\_strtoupper($string);

$string = trim($string);

if (empty($string)) {

return [];

}

$array = explode(' ', $string);

$count = count($array);

for ($key = 0; $key < $count; $key++)

{

$word = $array[$key];

if (is\_numeric($word))

continue;

if ($word == '-') {

unset($array[$key]);

continue;

}

$\_word = $this->morphy->get('ru')->getBaseForm($word);

if ($\_word) {

$array[$key] = current($\_word);

}

else {

if (preg\_match('/\d+/', $word)) {

unset($array[$key]);

continue;

}

$word = explode('-', $word);

if (count($word) > 2) {

$array[$key] = current($word);

unset($word[0]);

foreach ($word as $\_\_key => $\_\_word) {

if (empty($word))

unset($word[$\_\_key]);

}

$array = array\_merge($array, $word);

$count = count($array);

$key--;

}

}

}

return $array;

}

# ПРИЛОЖЕНИЕ В

var jVision = window.jVision = new (function () {

this.maintenance = function ();

this.major = function ();

this.minor = function ();

this.build = function ();

this.version = function ();

this.call = function (type, callback);

this.ready = function (callback);

this.get = function (selector);  
});

window.requestAnimationFrame = function(callback, element);

Array.prototype.shuffle = function ();

String.prototype.trim = function ();

String.prototype.toDom = function ();

var jVObject = function (selector) {

this.treeDom = [];

this.pointerTreeDom = undefined;;

this.push = function (newDom);

this.setTreeDom = function (value);

this.reset = function ();

this.count = function ();

this.currentNode = function ();

this.cloneCurrentNode = function ();

this.get = function (index);

this.eq = function (index);

this.first = function ();

this.last = function ();

this.each = function (callback);

this.call = function (callback);

this.text = function (text);

this.html = function (html);

this.parent = function ();

this.prev = function ();

this.next = function ();

this.remove = function ();

this.append = function (html);

this.outerHtml = function ();

this.attr = function (type, value);

this.css = function (type, value);

this.width = function ();

this.height = function ();

this.toggleClass = function (className);

this.hasClass = function (className);

this.addClass = function (className);

this.removeClass = function (className);

this.find = function (selector);

this.animate = function (event, duration, complite);

this.hide = function (timer);

this.show = function (timer);

};

window.$ = function (selector) {

switch (typeof selector) {

case 'function':

return jVision.ready(selector);

default:

return jVision.get(selector);

}

};

# ПРИЛОЖЕНИЕ Г

function getXmlHttp(){

var xmlhttp;

try {

xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");

}

catch (e) {

try {

xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

}

catch (E) {

xmlhttp = false;

}

}

if (!xmlhttp && typeof XMLHttpRequest!='undefined') {

xmlhttp = new XMLHttpRequest();

}

return xmlhttp;

}

jVision.prototype.ajax = function(options) {

var req = getXmlHttp()

var statusElem = this.currentNode();

req.onreadystatechange = function() {

if (req.readyState == 4) {

statusElem.innerHTML = req.statusText

if(req.status == 200) {

return req.responseText;

}

Else {

Throw new jVisionExaption(“Нет данных”);

}

}

}

req.open(options.method, options.url, options.lowS);

req.send(options.data);

this.showLoading(‘Идет обработка запроса…’);

}