В. А. Дьяченко

Разработка интерфейса доступа к истории информационной системы УГП

Аннотация. В статье описан ход работы над информационной системой УГП: разработка и применение модуля History.pm для работы с историей базы данных информационной системы edu.botik.ru.

Ключевые слова и фразы: История, модуль, база данных, Perl, History.pm.

1. Введение

Информационная система УГП edu.botik.ru хранит множество данных: информация о студентах, студенческих группах, программ курсов и многое другое. Все внесённые данные в систему хранятся. Это означает, что база данных содержит историю изменений.

Работа с историей является средством обеспечения информационной безопасности, так как позволяет узнать кто, когда и какие изменения вносил, а так же просмотреть или восстановить предыдущие данные. Для этого необходимо наладить работу с историей изменений базы данных УГП. Решением проблемы стало создание подключаемого модуля History.pm, созданного на языке программирования Perl, средствами которого возможны все необходимые взаимодействия с историей.[1]

2. Разработка

2.1. Строение базы данных

Для работы с историей БД необходимо понимать строение базы данных. База данных УГП edu.botik.ru это не реляционная БД типа ключ-значение (key-value store). Каждая запись в базе состоит из ключа записи и её значения, и то и другое являются строками произвольной длины, где ключ представляет собой конкатенацию логического ключа, времени его появления, указания автора записи

либо специфической информации о её происхождении и другой информации. По названию логического ключа можно понимать какую информацию мы собираемся прочитать из базы благодаря контексту данных. Каждый кусочек контекста разделяется символом '/' и несет особую смысловую нагрузку.[3]

2.2. Начало разработки

Язык программирования Perl был выбран в следствии того, что большая часть системы написана на этом языке и вся функциональность подключаемого модуля потребуется в скриптах написанных на Perl.[2] Во время разработки подключаемого модуля History.pm первостепенное внимание уделялось надёжности и скорости работы. History.pm проектировался как базовое средство работы с историей, на основе которого возможно решить всевозможные задачи этого направления.

History.pm создавался с использованием модуля Exporter, так как он реализует стандартный метод import, которым пользуется большинство других модулей. Пример использования модуля Exporter:

```
use Exporter();
our @ISA = "Exporter";
our @EXPORT = qw (
   &get_history
   &get_history_val
   &get_context_history
);
our @EXPORT_OK;
```

Массивы @EXPORT и @EXPORT_OK содержат списки символов, которые мы экспортируем во внешние пространства имен по умолчанию и по запросу соответственно. Что позволяет использовать &get_history и другие функции из History.pm в скриптах без прямого обращения к модулю.

2.3. Основные функции

Главные функции History.pm обеспечивающие базовую работу по извлечению и обработку с историей это:

```
&get_history($key, $primary_time, $secondary_time);
&get_history_val($key, $primary_time, $secondary_time);
&get_context_history($context, $primary_time, $secondary_time);
```

Где:

- get_history возвращает историю изменений одного ключа базы данных.
- get_history_val возвращает историю изменений значений одного ключа базы данных.
- get_context_history возвращает историю изменений множества ключей базы данных по указанному контексту.

Под историей изменений ключа БД понимается список всех версий этого ключа. get_history и get_history_val работают с одним конкретным ключом, а get_context_history с контекстом логического ключа собирая информацию сразу о группе ключей с общим смыслом. Обязательным параметром для этих функций является только первый, который передаёт контекст или логический ключ. Остальные ключи опциональные, каждый последующий параметр позволяет более детально работать с информацией. $primary_time, $secondary_time указывают временные рамки. В базе данных информационной системы УГП время есть количество секунд с начала эпохи. Что позволяет сравнивать время логическим условием больше или меньше. При указании, дополнительно к контексту, временных рамок будут обработаны записи из базы данных время создания которых не превышает $primary_time и (если указанно три параметра) меньше $secondary_time. Это позволяет провести первичный отбор нужных ключей из базы данных.

В информационной системе УГП в качестве СУБД применяется Tokyo Cabinet. Tokyo Cabinet - подключаемая библиотека для управления базами данных ключ-значение. Сопоставимымая в функциональности с SQLite (но без фактического внедрения SQL). Вся работа с базой данных осуществляется с помощью Tokyo Cabinet. Быстрой работы с базой данных можно добиться использованием класса TokyoCabinet::BDBCUR реализующий работу с курсором.[4] Курсор - это механизм, для получения доступа к каждому отчету базы данных в порядке возрастания или порядке по убывания.

В TokyoCabinet::BDBCUR есть множество методов для работы с курсором это: first, jump, key, new, next, prev, val и другие. Метод jump позволяет инициализировать объект cursor в базе данных и переместить курсор на место в БД соответствующее ключу. Методы next и prev позволяют переместите курсор в следующий или предыдущей записи соответственно. key и val получают ключ или значение в БД той записи, где курсор находиться в момент обращения.

Смысл алгоритма работы &get_context_history заключается в следующем:

- Инициализации обьекта cursor в базе данных и перемещение курсора на место в БД соответствующее контексту с помощью метода jump.
- Установлении курсора на самую верхнюю запись в БД перемещением методами next и prev.
- Последовательное чтение данных из базы используя и сдвиг курсора на одну запись ниже до тех пор пока записи соответствуют контексту применяя для этого методы key и next.

Программный код &get_context_history:

```
1  sub get_context_history {
2     my ($context, $primary_time, $secondary_time) = @_;
3
4     return undef unless defined $context or $context;
5
6     $primary_time   = Apache2::DBI::db_time() unless defined
       $primary_time   or $primary_time;
7     $secondary_time = 0                       unless defined
       $secondary_time or $secondary_time;
8
9     Apache2::DBI::open_db('r');
10
11    my $cursor = Apache2::DBI::cursor();
12    my @result;
13
14    $cursor->jump($context);
15
16    if($cursor->key =~ /^[^_]+_\d+_/){
17       $cursor->prev while  $cursor->key =~ /^$context[^_]*_
       */;
18       $cursor->next unless $cursor->key =~ /^$context[^_]*_
       */;
19
20       while($cursor->key =~ /^$context[^_]*_(\d+)_/){
21          my $keyTime = $1;
22
23          push @result, $cursor->key if $keyTime >=
       $secondary_time and $keyTime <= $primary_time;
24
25          $cursor->next;
```

```
26          }
27      }else{
28          warn "Data::History::get_context_history:ERROR!: Bad
        cursor key.";
29      }
30
31      Apache2::DBI::close_db();
32
33      return @result;
34  }
```

Алгоритмы работы &get_context_history и &get_history схожий, в связи с этим и во избежание дублирования программного кода исполняющей функцией является только &get_context_history, т.е. если в качестве контекста функции передать конкретный ключ базы данных то и обработка будет ограничена только лишь одним ключом. Однако &get_history остается полезным инструментом в задачу которого входит задание строгого, определённой формы, запроса к &get_context_history ограничивающего обработку в рамках одного ключа БД и избавляющего от возможной ошибки, когда искомый ключ совпадает с контекстом других ключей.

Алгоритм &get_history_val отличается от предыдущих функций способом обращения к базе данных.

3. Результаты

На основе базовых функций по работе с историей (get_history, get_history_val, get_context_history) в момент написании статьи реализованы следующие функции:

```
&program_course_history
&students_list
&student_groups_info
&last_student_group
```

Где:

- &program_course_history Возвращает историю изменений программы курса.
- &students_list Возвращает хеш связывающий id студентов и их полные имена.
- &student_groups_info Возвращает список всех групп студента по его id.

- &last_student_group Возвращает последнюю группу студента по его id.

Примеры использования:

```
my $last_group = &last_student_group($studentID);
my @student_groups = &student_groups_info($studentID);
my %students = students_list = &students_list();
```

Список функций не полный, будет расширяться и дальше. В статье указаны только те функции которые созданы для работы с интерфейсами.

Для использования функций History.pm в других модулях, пакетах, скриптах необходимо только написать в коде следующую строку и все основные функции будут доступны:

```
use Data::History.pm
```

На основе модуля History.pm, М.В.Шустовой разработан интерфейс, который позволяет просматривать старые версии программы учебной дисциплины.

Список литературы

[1] С. М. Абрамов, Н. С. Живчикова, С. В. Знаменский, Е. С. Иванов, А. В. Котомин, Д. Н. Степанов, Е. В. Титова, В. Н. Юмагужина. *Архитектура системы для разработки технологий организации сложной совместной деятельности* // Прикладная информатика, 2010, № 2(26), с 31-41. ↑ 169.

[2] *Документация языка программирования Perl*, URL http://perldoc.perl.org. ↑ 170.

[3] *Страница проекта информационной системы для УГП*, URL http://wiki.botik.ru/IS4UGP/. ↑ 170.

[4] *Официальный сайт проекта Tokyo Cabinet*, URL http://fallabs.com/tokyocabinet. ↑ 171.

Специфика статьи: *Управление организационными структурами*, Развитие информационно-вычислительных технологий, *Подпрограмма или библиотека программ*, *Интерактивное приложение или его часть*, Информационный ресурс, *Языки программирования*, Системы управления базами данных.

Научный руководитель:

д.ф.-м.н. С. В. Знаменский

Об авторе:

Владислав Андреевич Дьяченко
УГП имени А. К. Айламазяна, 3М21
e-mail: dyachenko.vlad_76@mail.ru

Пример ссылки на эту публикацию:

В. А. Дьяченко. «Разработка интерфейса доступа к истории информационной системы УГП ». *Наукоёмкие информационные технологии: Труды XIX Молодежной научно-практической конференции* SIT-2015. УГП имени А. К. Айламазяна. — Переславль-Залесский: Изд-во «Университет города Переславля», 2015 с. 169–176.
URL https://edu.botik.ru/proceedings/sit2015.pdf

Vladislav Dyachenko. *Development of the UGP information system interface of access to history.*.

Abstract. Ihis article describes the progress of work on the information system of Pereslavl University, edu.botik.ru: development and use of the History.pm module for access and work with history of the information system database of edu.botik.ru.

Key Words and Phrases: History, module, database, Perl, History.pm.

Sample citation of this publication:

Vladislav Dyachenko. "Development of the UGP information system interface of access to history.". *Science-intensive information techlnologies: Proceedings of XIX Junior R&D conference SIT-2015*. Ailamazyan Pereslavl University. — Pereslavl-Zalesskiy: Pereslavl University Publishing, 2015 pp. 169–176. (*In Russian.*)

URL https://edu.botik.ru/proceedings/sit2015.pdf