СЕМЕНОВ Геннадий Николаевич -
кандидат технических наук, доцент кафедры информационных компьютерных технологий Российского химико-технологического университета (РХТУ) им. Д.И. Менделеева
Адрес: 125047, г. Москва, Миусская пл., 9
e-mail: sem1237@yandex.ru

САРГСЯН Сергей Арташесович -
студент факультета информационных технологий и управления РХТУ им. Д.И. Менделеева
Адрес: 125047, г. Москва, Миусская пл., 9

НАУМЕНКО Сергей Анатольевич -
ассистент кафедры информационных компьютерных технологий РХТУ им. Д.И. Менделеева
Адрес: 125047, г. Москва, Миусская пл., 9
e-mail: sergey.naumenko@yahoo.com

SQL-тренажер по дисциплине «Управление данными»

Дисциплина «Управление данными» является базовой дисциплиной профессионального цикла направления ВПО 230400 «Информационные системы и технологии», при изучении которой студенты овладевают технологией баз данных, одной из наиболее важных информационных технологий.

Дистанционное обучение занимает все большую роль в модернизации образования. Согласно приказу № 137 Министерства образования и науки РФ от 06.05.2005 «Об использовании дистанционных образовательных технологий» итоговый контроль при обучении с помощью дистанционных образовательных технологий можно проводить как очно, так и дистанционно.

Образование в области информационных технологий, прежде всего, предполагает формирование устойчивых практических навыков. В курсе «Управление данными» основные темы связаны с проектированием и администрированием баз данных, а также разработкой приложений, использующих базы данных. Базовым навыком здесь является владение языком структурированных запросов (SQL).

В интернете существует ресурс [1], позволяющий приобрести или повысить свои навыки в написании операторов манипуляции данными языка SQL. Данный проект, по нашему мнению, предназначен для профессионалов в области баз данных с целью повышения квалификации. Однако для дистанционного обучения его невозможно применить, поскольку для преподавателя здесь нет возможности контроля обучения и модификации заданий.

Цель данного проекта - разработка электронного тренажера для манипулирования данными в базах данных (SQL-тренажер) для обучения (тренинга) и контроля знаний студента и внедрение тренажера в учебный процесс.

Концептуальная модель SQL-тренажера

Тренажер состоит из трех основных компонентов: модуля для обучения и контроля знаний студента, модуля преподавателя для создания и редактирования задач и модуля помощи.

Особенностью данного SQL-тренажера является то, что обучение языку запросов ведется на реально существующих базах данных, размещенных на сервере, посредством web-клиента, что позволяет студенту и преподавателю работать с тренажером на любом компьютере, подключенном к сети Интернет.

Обучение состоит в том, что студент, выполняя задание, пишет оператор на языке SQL (SQL-запрос), который должен извлечь или изменить данные, требуемые заданием. В случае неправильного ответа студент может узнать, какие данные возвращает правильный запрос, а также увидеть, что вернул его сформированный SQL-запрос. Задания разбиты на категории по темам и по уровню сложности. Манипуляции с данными могут проводиться над несколькими учебными базами данных в зависимости от варианта.

Модуль преподавателя дает преподавателю возможность контролировать успеваемость, модифицировать задания и варианты, что позволяет обеспечить дистанционное и очное обучение по стандарту языка SQL в виде лабораторного практикума.

Одним из требований к данной системе является возможность интеграции с модульной объектно-ориентированной динамической учебной средой Moodle, с целью комплексного контроля успеваемости студентов по другим разделам изучаемой дисциплины.

Разработанные требования к электронному тренажеру и разработанная архитектура обучающей системы позволяет наращивать новые функции и модули.

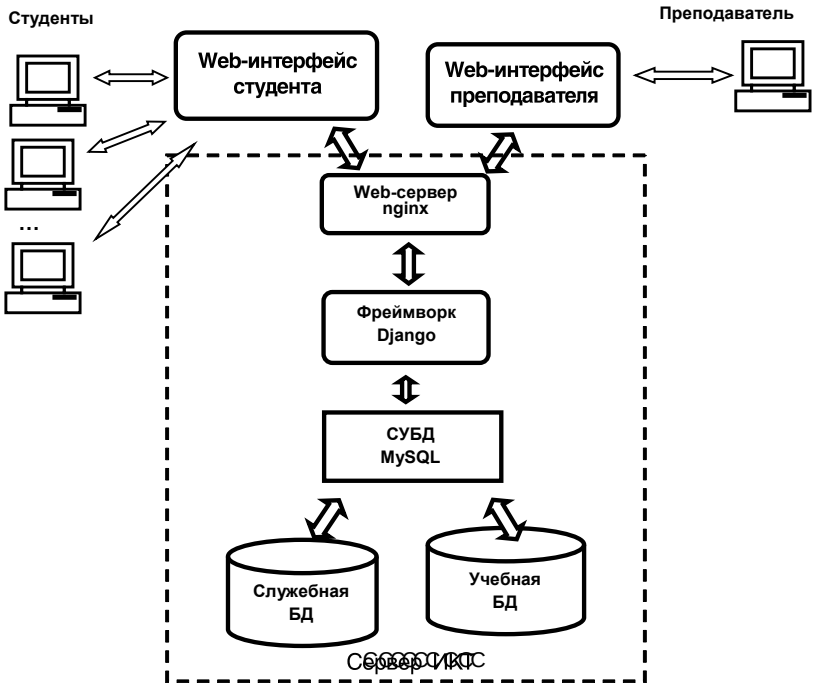


Рис. 1. *Архитектура системы «SQL-тренажер»*

Архитектура системы «SQL-тренажер»

Доступ к данным осуществляется по клиент-серверной архитектуре (**рис.1**). Клиентами являются браузеры студентов и преподавателя, серверная часть состоит из нескольких компонентов и размещена на выделенном сервере с постоянным IP-адресом (сервер ИКТ).

Сервер состоит из компонентов:

- Web-сервер (nginx) - осуществляет доступ клиентов к html-странице web-интерфейса.

- Фреймворк Django - свободный фреймворк для web-приложений на языке Python. Такая технология программирования связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных», в которой модель данных описывается классами объектно-ориентированного языка Python, и по ней генерируется схема базы данных.

- Система управления базами данных (СУБД) работает с данными из служебной и учебной баз данных (БД).

Фреймворк является каркасом для создания web-приложений, он уже содержит множество готовых функций. Выбор пал на кроссплатформенный фреймворк Django 1.3. Основные причины такого решения: высокая скорость разработки приложения и высокое быстродействие работы системы.

Существует множество диалектов SQL, но все они реализуют стандартное подмножество ANSI SQL. В качестве СУБД была выбрана MySQL. Эта СУБД достаточно популярна в web-разработках, поскольку является бесплатной, обладает высокой скоростью выполнения запросов и работает с множеством языков программирования.

При разработке web-приложений в настоящее время критически важными являются быстродействие системы, минимизация трафика, обновление данных без перезагрузки страницы.

При разработке web-приложения был использован не простой POST-запрос (метод POST применяется для передачи пользовательских данных заданному ресурсу), а AJAX-POST-запрос (технология AJAX заключается в фоновом обмене данными браузера с web-сервером, в результате чего, при обновлении данных, web-страница не перезагружается полностью, и web-приложения становятся более быстрыми и удобными). При использовании обычного POST-запроса страница будет перегружаться, что в нашем случае может привести к потере данных. AJAX-POST-запрос асинхронно подгружает нужную информацию без перезагрузки страницы. Это позволяет сохранить код SQL-запроса, написанного с ошибкой: в случае отправки такого кода на проверку исходный вариант сохраняется и может быть исправлен. Использование AJAX-запросов также позволяет уменьшить нагрузку на сервер.

В качестве web-сервера был использован сервер nginx, работающий на Unix-подобных операционных системах (web-сервер работает под операционной системой Linux), поскольку данный web-сервер прост в использовании, кроме того web-сервер nginx удобен для запуска на сервере Django-приложений.

Принцип работы интерфейса студента построен на GET-запросах (метод GET используется для запроса содержимого указанного ресурса) и AJAX-POST-запросах **(рис. 2)**. При запуске страницы студента автоматически подгружается из служебной базы список задач. При выборе задачи AJAX-POST-запросом посылаются данные на сервер, сервер обрабатывает и отдает условие задачи обратно. По условию задачи пишется SQL-код, при нажатии кнопки «выполнить» посылается AJAX-POST-запросом на сервер SQL-код, сервер отправляет код в СУБД MySQL, которая взаимодействует с учебной БД, обратно посылаются данные о выполнении SQL-кода; в web-приложении формируется таблица результата запроса (если SQL-код правильный) и отсылается на страницу студента.

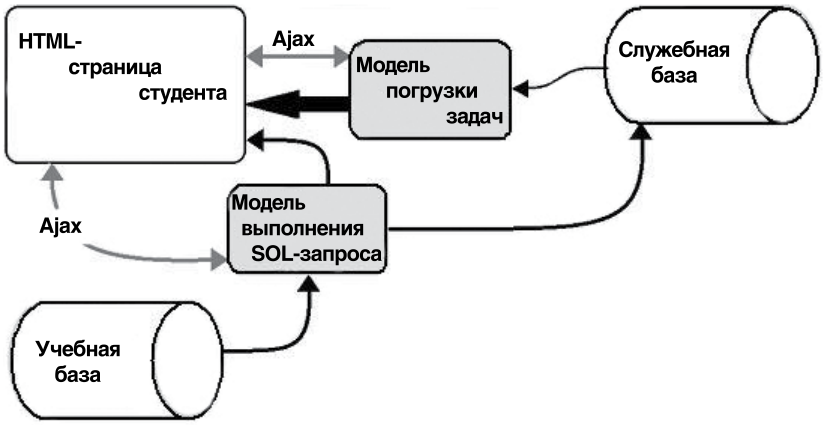


Рис. 2. *Схема работы интерфейса студента*

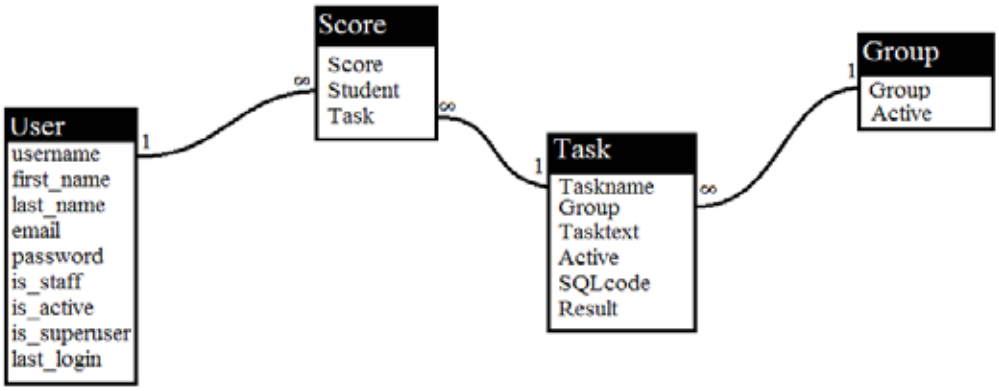
Базы данных

Служебная база Django

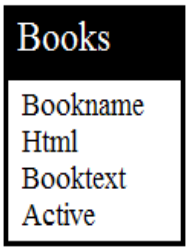
Служебная база данных состоит из: базы данных заданий (**рис. 3а**), базы данных учебного материала (**рис. 3б**).

Тестовая база данных состоит из 4-х связанных между собой таблиц (рис. 3а):

- Group(группы)
  - Group - себе название групп вопросов;



а



б

***Рис. 3** Инфологическая модель служебной базы данных Django: а - база данных заданий, б - база данных учебного материала*

o Active - активность данной группы, переменная булевского типа, принимающая два значения: 1 или 0.

- Task(задачи)

o Taskname - название задачи, например: Задача 1;

o Group - название группы, к которой принадлежит данная задача, связана с таблицей Group по внешнему ключу;

o Tasktext - условие данной задачи;

o Active - активность данной задачи, переменная булевского типа, принимающая два значения: 1 или 0;

o SQLcode - в данном поле хранится SQL-запрос;

o Result - в данном поле хранится ответ данной задачи.

- User(пользователи)

o username - логин пользователя;

o first_name - имя пользователя;

o last_name - фамилия пользователя;

o email - email пользователя;

o password - пароль пользователя;

o is_staff - статус персонала, переменная булевского типа, принимающая два значения: 1 или 0;

o is_active - активность пользователя, переменная булевского типа, принимающая два значения: 1 или 0;

o is_superuser - суперпользователь, переменная булевского типа, принимающая два значения: 1 или 0;

o last_login - последний вход в систему данного пользователя.

- Score(оценки)

o Score - поле с оценкой студента;

o Student - пользователь, который получил оценку, связана внешним ключом с таблицей User;

o Task - название задачи, на которую ответил студент, связана внешним ключом с таблицей Task.

База данных учебного материала, состоит из одной таблицы (рис. 3б):

- Books

o Bookname - название учебного материала;

o Html - html-ссылка на внешние источники информации;

o Booktext - текст данного материала;

o Active - активность данного материала, переменная булевского типа, принимающая два значения: 1 или 0.

Учебная база данных

Учебная база данных моделирует учет успеваемости студентов при сдаче экзаменов и состоит из пяти таблиц, где хранятся данные о предметной области: экзаменах, группах, предметах, преподавателях и студентах (**рис. 4**).

Структура таблиц учебной базы данных (**рис. 5**):

- Grup(группы)

o Id_Group - первичный ключ для таблицы Grup;

o Name_Group - имя группы.

- Subject(дисциплины)

o Id_Sub - первичный ключ для таблицы Subject;

o Name - название дисциплины;

o Semestr - семестр, в котором преподается данная дисциплина.

- Student(студенты)

o Is_Stud - первичный ключ для таблицы Student;

o Last_Name - фамилия студента;

o First_Name - имя студента.

- Lecturer(лекторы)

o Id_Lect - первичный ключ для таблицы Lecturer;

o Last_Name - фамилия преподавателя (лектора);

o First_Name - имя преподавателя (лектора);

o Post - должность преподавателя (лектора).

- Exams(экзамены)

o Date_Exam - дата экзамена;

o Id_Group - внешний ключ;

o Id_Stud - внешний ключ;

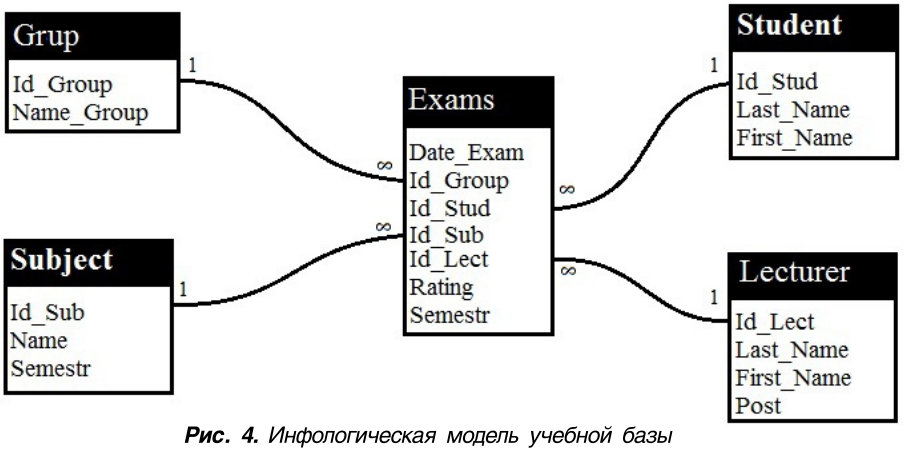


***Рис. 4.** Инфологическая модель учебной базы*

Grup

поле	тип	комментарий
Id_Group Name_Group	INT VARCHAR	первичный ключ

Student

поле	тип	комментарий
Id_Stud Last_Name First_Name	INT VARCHAR VARCHAR	первичный ключ

Subject

поле	тип	комментарий
Id_Sub Name Semestr	INT VARCHAT INT	первичный ключ

Lecturer

поле	тип	комментарий
Id_Lect Last_Name First_Name Post	INT VARCHAR VARCHAR VARCHAR	первичный ключ

Exams

поле	тип	комментарий
Date_Exam	DATE	
Id_Group	INT	вторичный ключ на Id_Group, таблица Grup
Id_Stud	INT	вторичный ключ на Id_Stud, таблица Student
Id_Sub	INT	вторичный ключ на Id_Sub, таблица Subject
Id_Lect	INT	вторичный ключ на Id_Lect, таблица Lecturer
Rating	INT	
Semestr	INT	

Рис. 5. *Даталогическая модель учебной базы*

- Id_Sub - внешний ключ;
- Id_Lect - внешний ключ;
- Rating - оценка, которую получил студент на экзамене;
- Semestr - семестр, в котором проходил экзамен.

Первая версия проекта ориентируется на стандартный ANSI SQL 1992, и задачи, связанные с извлечением данных, делятся на следующие категории [2]:

1. Однотабличные запросы.
2. Многотабличные запросы.
3. Сложные запросы, использующие: вложенные запросы, объединения.
4. Запросы с группировкой.
5. Вычисляемые запросы.

Примеры задач однотабличных запросов:

1. Задача: Выведите преподавателей, которые являются профессорами, из таблицы лекторов (Lecturer). Примечание: должность (Post) - 'Professor'.

Соответствующий SQL-код:

```
SELECT * FROM Lecturer WHERE Post = 'Professor'.
```

2. Задача: Вывести все предметы, которые преподаются после 5 семестра, из таблицы (Subject).

Соответствующий SQL-код:

```
SELECT * FROM Subject WHERE Semestr > 5.
```

3. Задача: Вывести максимальную оценку (Rating), которая была получена на экзамене в 3 семестре, (Semestr) из таблицы экзаменов (Exams).

Соответствующий SQL-код:

```
SELECT MAX (Rating) FROM Exams WHERE Semestr = '3'.
```

4. Задача: Вывести имена, фамилии и оценки всех студентов.

Соответствующий SQL-код:

```
SELECT Student.First_Name, Student.Last_Name, Exams.Rating
FROM Student, Exams
WHERE Exams.Id_Stud = Student.Id_Stud.
```

Примеры задач на многотабличные запросы:

5. Задача: Вывести имена, фамилии и оценки всех студентов (Примечание: оценки (Rating) в таблице экзаменов (Exams).

Соответствующий SQL-код:

```
SELECT Student.First_Name, Student.Last_Name, Exams.Rating
FROM Student, Exams
WHERE Exams.Id_Stud = Student.Id_Stud.
```

6. Задача: Вывести имена и фамилии преподавателей и дисциплины, по которым они принимали экзамен, исключить повторы (Exams - таблица экзаменов, Lecturer - таблица преподавателей, Subject - таблица дисциплин).

Соответствующий SQL-код:

```
SELECT DISTINCT Lecturer.Last_Name, Lecturer.First_Name, Subject.Name
FROM Exams, Lecturer, Subject
WHERE Subject.Id_Sub = Exams.Id_Sub
AND Lecturer.Id_Lect = Exams.Id_Lect.
```

Примеры задач на вложенные запросы:

7. Задача: Вывести дату экзаменов, когда лектор "Filipova" принимала экзамены.

Соответствующий SQL-код:

```
SELECT Date_Exam FROM Exams WHERE Id_lect = (
SELECT Id_Lect FROM LecturerWHERE Last_Name = 'Filipova' ).
```

В дальнейшем система может развиваться в направлении включения более широкого множества элементов языка SQL-операторов модификации данных, возможностей работы с XML-данными, процедур, триггеров, представлений и в направлении изучения особенностей диалектов MySQL и PL/SQL (Oracle).

Литература:

1. Практическое владение языком SQL. [Электронный ресурс]. - Режим доступа: http://www.sql-ex.ru. - Заглавие с экрана (Дата обращения 16 ноября 2011).

2. Дейт К. SQL и реляционная теория. Как грамотно писать код на SQL/ Пер. с англ. - М.; Символ-Плюс, 2010. - 480 с.