Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Дисциплина «Структуры и базы данных»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  магистр техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.О. Боровская |
|  | \_\_\_.\_\_\_\_.2020 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«База данных для поддержки работы кафедрЫ»**

БГУИР КР1-39 03 02 005 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 813801  Шамрей Андрей Станиславович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_\_.\_\_\_\_.2020  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2020

**РЕФЕРАТ**

БГУИР КП 1-39 03 02 005 ПЗ

**Шамрей, А.С.** База данных для поддержки работы кафедры: пояснительная записка к курсовому проекту / А.С. Шамрей. – Минск : БГУИР, 2020. – 45 с.

Пояснительная записка 45 с., 23 рис., 2 табл., 53 источника, 5 приложений

Цель проектирования: проектирование базы данных, используемой для поддержки работы кафедры.

Методология проведения работы: в процессе решения поставленных задач использованы принципы системного подхода, теория проектирования реляционных баз данных, методы проектирования баз данных с использованием *CASE*-средств.

Результаты работы: выполнен анализ предметной области   
и информационных потребностей пользователей, спроектирована реляционная база данных для работы кафедры, разработано веб‑приложение для работы с базой данных.

Курсовой проект состоит из расчётно-пояснительной записки, графической и программной части. В расчётно-пояснительной записке приводится общая характеристика базы данных, раскрываются задачи и требования базы данных.

Приложение обеспечивает управление данными, необходимыми для работы кафедры: личной информацией и расписанием преподавателей. Также приложение обеспечивает корректный просмотр необходимых данных сотрудниками кафедры: расписание преподавателей, объявления о консультациях, документы кафедры.

**СОДЕРЖАНИЕ**

Введение5

1 Анализ предметной области и ее формализация для проектирования базы данных7

1.1 Описание предметной области7

1.2 Анализ информационных потребностей пользователей и предварительное описание запросов 8

1.3 Определение требований и ограничений к базе данных с точки зрения предметной области9

1.4 Постановка решаемой задачи9

2 Проектирование базы данных для основного вида деятельности рассматриваемой предметной области 11

2.1 Разработка инфологической модели предметной области базы данных11

2.2 Выбор и обоснование используемых типов, данных и ограничений (доменов)13

2.3 Проектирование запросов к базе данных15

2.4 Программная реализация и документирование базы данных16

2.5 Проектирование разрабатываемого приложения17

Заключение18

Список использованных источников19

Приложение А (обязательное) Скрипт генерации базы данных (к пункту 2.4)20

**ВВЕДЕНИЕ**

**1** **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ЕЕ**

**ФОРМАЛИЗАЦИЯ ДЛЯ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ**

Предметная область – это целенаправленная первичная трансформация картины внешнего мира в некоторую умозрительную картину, определенная часть которой фиксируется в информационной системе в качестве алгоритмической модели фрагмента действительности.

Анализ предметной области предполагает составление описания предметной области, которое подразумевает формулирование и анализ требований, предъявляемых к содержанию и процессу обработки данных всеми известными и потенциальными пользователями базы данных. На этапе системного анализа необходимо провести подробное словесное описание информационных объектов предметной области и реальных связей, которые присутствуют между описываемыми объектами. Системный анализ является наиболее трудным и длительным этапом процесса проектирования.

В результате системного анализа должны быть сформулированы: подробное описание информации об объектах предметной области и информационных процессов; конкретные задачи, которые будут решаться базой данных с кратким описанием алгоритма решения; описание документов, которые должны генерироваться в системе; описание документов, которые служат основанием для заполнения базы данных.

**1.1 Описание предметной области**

Задача проекта – создание информационной поддержки учебного процесса и организационной деятельности на кафедре университета. Различают две группы сотрудников:

– профессорско-преподавательский состав;

– учебно-вспомогательных состав.

База данных должна содержать штатное расписание преподавателей кафедры с указанием места и времени проведения занятия, а также включать архив сведений о сотрудниках.

База данных должна предоставлять сотруднику кафедры возможность добавления информации о проводимых дополнительных занятиях, а также о возможных объявлениях и изменениях в расписании.

Эта база данных должна содержать сведения о сотрудниках университета: личных данных сотрудников (фамилия, имя, отчество, дата рождения, мобильный телефон, электронная почта и так далее), их трудовой деятельности (должности, научные степени, области научных исследований, читаемые курсы и так далее). Кроме того база данных кафедры должна хранить информацию группах студентов и специальностях, на которых преподают сотрудники кафедры.

**1.2 Анализ информационных потребностей пользователей**

**и предварительное описание запросов**

Рассмотрим университет, подготавливающий специалистов разного профиля. На кафедре университета различаются две группы сотрудников: профессорско-преподавательский состав и учебно-вспомогательных состав.

Необходимость разработки базы данных обусловлена стремлением структурировать информацию о сотрудниках и облегчить получения сотрудниками информации о расписании проводимых занятий.

Сайт кафедры предоставит возможность структурировать расписание преподавателей, а также даст возможность беспрепятственно получать информацию о расписании занятий преподавателей. Также будет предоставлен доступ к хранилищу документов и файлов кафедры.

Данной базой данных будут пользоваться сотрудники кафедры университета и администратор. Сотрудники кафедры будут иметь возможность просматривать личные данные и расписание преподавателей кафедры, редактировать свои личные данные, просматривать документы кафедры, а также размещать дополнительную информацию на личной страничке. Администратор будет иметь полный доступ ко всем сущностям системы: редактирование, удаление, обновление, добавление.

Для визуального отображения информационных потребностей используется диаграмма вариантов использования, показанная на рисунке 1.1, которая позволяет наглядно отобразить информационные потребности в проектируемой системе.

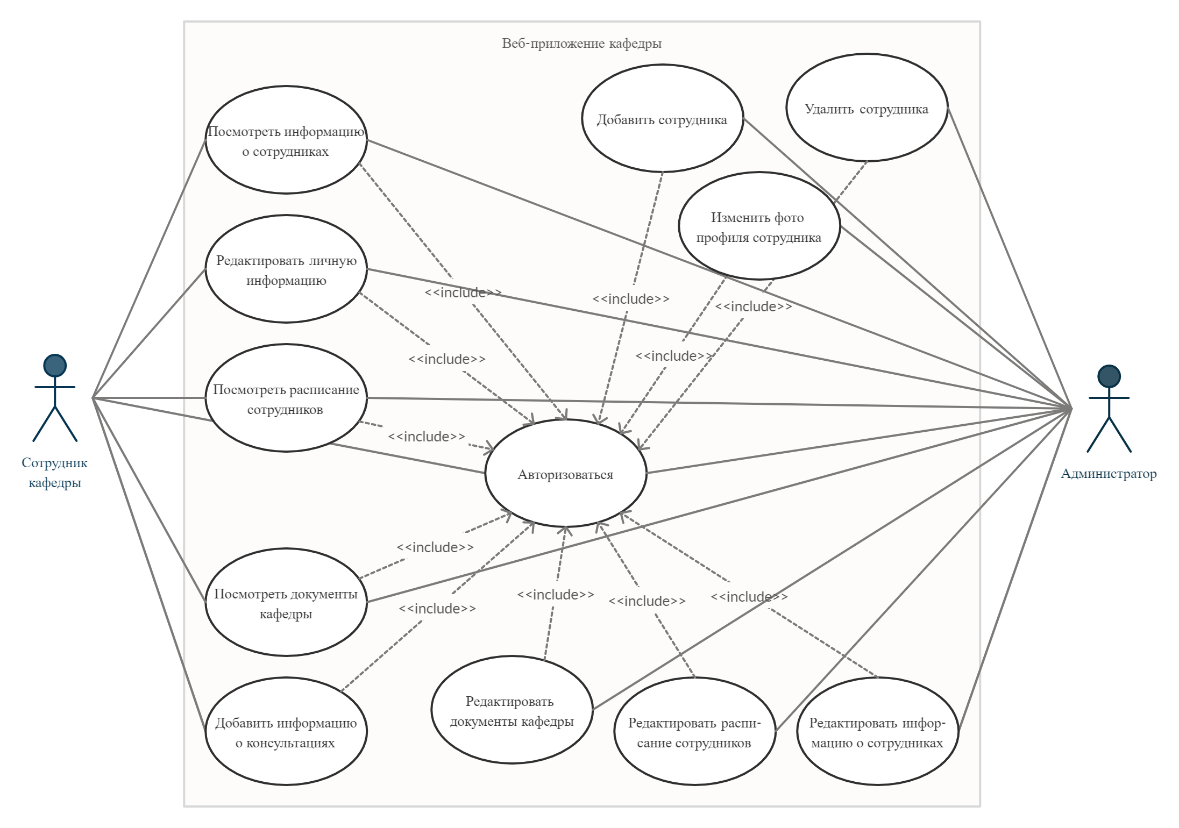


Рисунок 1.1 – Диаграмма вариантов использования веб-приложения кафедры

**1.3 Определение требований и ограничений к базе данных с точки зрения предметной области**

Информационная система, как и любой другой инструмент, должна иметь свои характеристики и требования, в соответствии с которыми можно было бы определить ее функциональность и эффективность. Ниже приведены основные требования и ограничения, предъявляемые к проектируемой базе данных.

Требование уникальности записей распространяется на следующие атрибуты: уникальный номер пользователя user\_id для аутентификации; role\_id для определения типа пользователя; group\_number для указания номера группы; speciality\_name для указания названия специальности. Атрибуты user\_id, role\_id и group\_number должны включать только цифры.

Обязательность указания имеют следующие атрибуты: first\_name, last\_name и father\_name для указания ФИО; login и pass для указания логина и пароля пользователя. Атрибуты first\_name, last\_name и father\_name должны включать только буквы.

Данные, хранимые в атрибуте subject\_time\_start, предназначенном для указания старта занятия, не могут быть равны или больше данных, указанных в атрибуте subject\_time\_end.

**1.4 Постановка решаемой задачи**

В данном курсовом проекте требуется изучить особенности и структуру реляционных баз данных. Реляционные базы данных основаны на теоретико-множественной реляционной даталогической модели. Все данные представляются в виде связанных между собой таблиц, разбитых на строки и столбцы. Каждая таблица должна иметь первичный ключ – поле или набор полей, содержимое которых однозначно определяет запись в таблице и отличает ее от других. Связь между двумя таблицами обычно образуется при добавлении в первую таблицу поля, содержащего значение первичного ключа второй таблицы.

Для создания базы данных необходимо проанализировать имеющуюся информацию о уже существующих, создать на ее основе необходимые таблицы, установить определенные связи между ними. Поэтому задачи, которые нужно будет решить в ходе курсового проекта, заключаются в разработке связей между таблицами базы данных, устранении избыточности в таблицах, их обработке и редактировании.

Эффективность работы любой организации во многом зависит от комфорта и условий работы сотрудников. Работники должны иметь быстрый доступ к расписанию сотрудников кафедры, а также к документам и публикациям кафедры.

Работа кафедры связана с накоплением большого количества информации о личных данных сотрудников. Традиционно информация храниться на бумажных носителях. При этом трудно осуществить быстрый отбор нужных данных.

Создание веб-приложения с взаимодействием с базой данных позволит сократить время на обработку информации, произойдет сокращение затрат на обработку информации, уменьшатся затраты времени на поиск необходимой информации, улучшится качество контроля и учета обрабатываемой информации, повысится эффективность работы всего состава кафедры.

Необходимость проектирования базы данных обусловлена стремлением структурировать информацию о сотрудниках и облегчить получения сотрудниками информации о расписании проводимых занятий

Целью данного проекта является проектирование базы данных для удобства хранения и использования информационной базы о сотрудниках кафедры, автоматизации получения и анализа данных о занятости сотрудников кафедры. Необходимо не только упорядочить информацию, но и упростить процессы ее анализа и принятия необходимых управленческих решений.

Целью всех выполняемых действий являются:

– автоматизировать процесс получения информации о сотрудниках;

– сократить время поиска необходимых данных;

– оптимизировать хранение информации, необходимой для деятельности кафедры.

Задачами данного проекта являются:

– выявить необходимость автоматизации информационных потоков процессов деятельности кафедры;

– проектирование базы данных MySQL;

– создание веб-приложения для проверки работоспособности спроектированной базы данных.

Главная задача моделируемой системы – сохранение в базе данных всех необходимых сведений о сотрудниках кафедры и их расписании; представление данных в удобном для пользователя виде.

Требования к целостности данных: данные в базе данных в любой момент времени должны быть правильными и непротиворечивыми.

Требования к безопасности: доступ к системе должен быть осуществлён после аутентификации пользователя, на основе прав и ролей пользователей.

Кроме того, клиентское приложение должно иметь дружественный и приятный интерфейс.

**2 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ ДЛЯ**

**ОСНОВНОГО ВИДА ДЕЯТЕЛЬНОСТИ**

**РАССМАТРИВАЕМОЙ ПРЕДМЕТНОЙ ОБЛАСТИ**

Проектирование базы данных является длительным и трудоёмким процессом. Это одна из наиболее сложных и ответственных задач, связанных с созданием информационной системы. Разработка базы данных и прикладного программного обеспечения для поддержания работы кафедры университета состоит из проектирования базы данных и разработки прикладного программного обеспечения. ­

Процесс проектирования включает в себя следующие этапы: концептуальное проектирование, логическое проектирование, физическое проектирование [1].

В этом разделе будет рассмотрен способ создания полноценной базы данных.

**2.1 Разработка инфологической модели предметной области базы данных**

Инфологическая модель данных – обобщенное неформальное описание создаваемой базы данных, выполненное с использованием естественного языка, математических формул, таблиц, графиков и других средств, понятных всем людям, работающим над проектированием базы данных. Она является человеко-ориентированной моделью, которая полностью независима от физических параметров среды хранения данных [2].

Цель инфологического моделирования – обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных. Храниться данные будут в диаграмме «сущность – связь», которая будет создана с помощью системы управления базами данных.

Модель «сущность – связь» является графическим средством представления объектов рассматриваемой предметной области, их характеристик и отношений между объектами [1].

Основными элементами инфологической модели являются сущности, связи между ними и их атрибуты.

Сущность – любой конкретный или абстрактный объект в рассматриваемой предметной области. Сущности – это базовые типы информации, которые хранятся в базе данных.

Атрибут – это свойство сущности в предметной области. Его наименование должно быть уникальным для конкретного типа сущности.

Связь – это некоторая ассоциация между двумя сущностями. Связи представляют собой соединения между частями базы данных [3].

Диаграмма «сущность – связь» (*ER*-диаграмма) позволяет графически представить все элементы информационной модели согласно простым, интуитивно понятным, но строго определенным правилам – нотациям.

Прежде чем проектировать базу данных, необходимо привести данные к третьей нормальной форме и ответить на два вопроса:

– какой атрибут будет являться первичным ключом;

– какие атрибуты будут являться внешними ключами.

На этапе концептуального проектирования, имея словесное описание предметной области, предусматривается выполнение таких работ как:

– идентификация объектов предметной области, их атрибутов и первичных ключей;

– идентификация отношений между объектами и указание мощности этих связей;

– построение концептуальной схемы базы данных на основе модели «сущность – связь».

В данной работе основными информационными объектами предметной области являются: сотрудники, деятельность, консультации, авторизация, расписание, читаемые курсы, роли, группы, предметы.

Между ними можно установить следующие логические связи:

– связь сотрудники-деятельность мощностью «один-к-одному». Один сотрудник может заниматься одной деятельностью.

– связь сотрудники-консультации мощностью «один-к-одному». Один сотрудник может иметь только одно поле для ввода дополнительной информации на сайте.

– связь сотрудники-авторизация мощностью «один-ко-многим». Один сотрудник может иметь только один аккаунт в веб-приложении кафедры.

– связь авторизация-роли мощностью «один-к-одному». Один сотрудник может иметь только одну роль: администратор либо сотрудник.

– связь сотрудники-расписание мощностью «один-ко-многим». Один сотрудник может вести несколько пар в день и у разных групп.

– связь расписание-группы мощностью «один-к-одному». Одному номеру группы может соответствовать только одно название специальности.

– связь расписание-предметы мощностью «один-к-одному». Одному номеру предмета может соответствовать только одно название предмета.

– связь сотрудники-читаемые курсы мощностью «один-ко-многим». Один сотрудник может читать несколько курсов.

– связь читаемые курсы-предметы мощностью «один-к-одному». Один курс соответствует одному названию предмета.

На этапе логического проектирования разрабатывается логическая схема базы данных, обосновывается выбранная методика проектирования базы данных, описывается процесс проектирования логической схема базы данных и этапы ее нормализации.

Также на этапе логического проектирования концептуальная модель отображается в логическую с учетом выбранной реляционной модели данных по правилам преобразования.

Каждый объект концептуальной модели отображается в таблицу базы данных. Результат логического проектирования изображается на *ER*-диаграмме, показанной на рисунке 2.1.

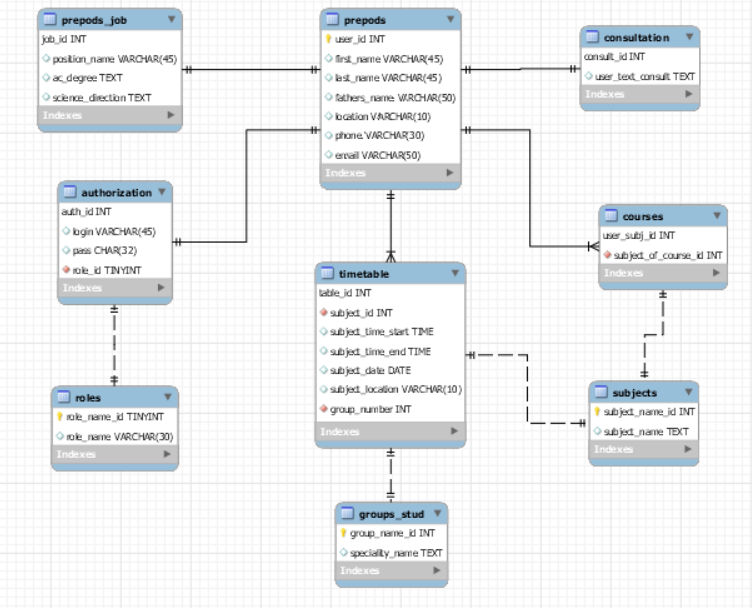


Рисунок 2.1 – Логическая схема проектируемой базы данных

Для достижения минимальной избыточности данных в базе данных необходимо произвести нормализацию таблиц (отношений) базы данных. На практике достаточно, чтобы отношение находилось в 3 нормальной форме. Отношение удовлетворяет третьей нормальной форме, тогда и только тогда, когда отношение находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых.

**2.2 Выбор и обоснование используемых типов, данных и ограничений (доменов)**

Физическое проектирование является третьим и последним этапом создания проекта базы данных, при выполнении которого принимаются решения о способах реализации разрабатываемой базы данных.

Приступая к физическому проектированию базы данных, прежде всего необходимо выбрать конкретную целевую СУБД, так как физическое проектирование неразрывно связано с конкретной СУБД.

В данной работе используется СУБД *MySQL*. *MySQL* – это популярная свободная реляционная система управления базами данных. *MySQL* базируется на языке *SQL* и поддерживает многочисленные возможности.

Важным этапом проектирования базы данных является выбор и обоснование типов данных. Любые данные, помещаемые в столбец, должны отвечать этому типу данных. Тип данных определяет, какие значения могут храниться в столбце и сколько они будут занимать места в памяти.

*MySQL* поддерживает несколько типов столбцов, которые можно разделить на три категории: числовые типы данных, типы данных для хранения даты и времени и символьные (строковые) типы данных.

При проектировании базы данных были использованы следующие типы данных.

Тип данных CHAR, который представляет стоку фиксированной длины, при этом, если указано меньше символов, чем задано, то строка дополняется пробелами и в итоге все равно будет занимать заданное количество символов;

Тип данных VARCHAR, который представляет строку переменной длины. Хранимая строка будет занимать именно столько места, сколько необходимо, что позволит сократить память, выделенную для этой строки, если в ней указано символов меньше, чем задано;

Тип данных TEXT, который представляет текст длиной до 65 килобайт и используется для записи в строку символьной информации, превосходящей длину в 255;

Тип данных INT, который представляет целые числа от -2147483648 до 2147483647, занимает 4 байта и используется для нумерации элементов и записи числовых данных (например, номера группы);

Тип данных TINYINT, который представляет целые числа от -127 до 128, занимает 1 байт и используется для нумерации небольшого количества элементов (например, id роли – администратор либо сотрудник);

Тип данных TIME, который хранит время от -838:59:59 до 838:59:59. По умолчанию для хранения времени применяется формат "hh:mm:ss". Занимает 3 байта и используется для задания времени (например, начало и конец занятия в расписании преподавателя);

Тип данных DATE, который хранит даты с 1 января 1000 года до 31 деабря 9999 года (c "1000-01-01" до "9999-12-31"). По умолчанию для хранения используется формат yyyy-mm-dd. Занимает 3 байта и используется для величин с информацией только о дате, без части, содержащей время (например, дата проведения занятия преподавателем).

**2.3 Проектирование запросов к базе данных**

Запрос строится на основе одной или нескольких взаимосвязанных таблиц, позволяя комбинировать содержащуюся в них информацию. Запрос позволяет выбрать необходимые данные из одной или нескольких взаимосвязанных таблиц и получить результат в виде новой таблице. Полученная таблица может использоваться в качестве источника данных в следующих запросах, формах, отчетах, страницах доступа к данным. Через запрос можно производить вычисление, изменение данных в таблицах, добавление и удаление записей и многое другое.

Для работы с проектируемой базой данных в большинстве случаев достаточно паттерна *CRUD*. *CRUD* – акроним, обозначающий четыре базовые функции, используемые при работе с базами данных: создание (create), чтение (read), модификация (update) и удаление (delete).

В *SQL* этим функциям операциям соответствуют операторы *Insert* (создание записей), *Select* (чтение записей), *Update* (редактирование записей), *Delete* (удаление записей).

Оператор INSERT INTO предназначен для добавления данных в таблицу. Синтаксис оператора: *INSERT INTO table\_name ([column\_name, ... ]) VALUES (expressions, ...)*.

Оператор UPDATE предназначен для обновления данных таблицы. Эта команда имеет следующий синтаксис: *UPDATE table\_name SET expression [WHERE condition]*.

Оператор DELETE предназначен для удаления данных из таблицы. Синтаксис оператора: *DELETE from table\_name [WHERE condition]*. Если не будут указаны дополнительные условия с помощью команды *WHERE,* то из таблицы будут удалены все значения.

Для вывода данных из таблицы используется оператор SELECT. Синтаксис оператора: *SELECT column\_list FROM table\_name [WHERE сondition GROUP BY expression HAVING condition ORDER BY expression]*.

Рассмотрим команды вывода данных на примере проектируемой базы данных. Чтобы вывести список сотрудников и их номера телефонов используем запрос, приведенный на рисунке 2.2.



Рисунок 2.2 – Пример использования оператора SELECT №1

Также можно вывести список сотрудников, id которых больше 1. Необходимый для этого запрос приведен на рисунке 2.3.



Рисунок 2.3 – Пример использования оператора SELECT №2

Чтобы вывести список сотрудников, отсортированный по имени, используется запрос, приведенный на рисунке 2.4.



Рисунок 2.4 – Пример использования оператора SELECT №3

Чтобы вывести количество сотрудников, работающих в каждом кабинете, можно использовать запрос, приведенный на рисунке 2.5.



Рисунок 2.5 – Пример использования оператора SELECT №4

Чтобы вывести только те кабинеты, в которых работает больше одного сотрудника, можно использовать запрос, приведенный на рисунке 2.6.



Рисунок 2.6 – Пример использования оператора SELECT №5

Запросы позволяют решать многие задачи, не прибегая к программированию. Например, представлять данные в агрегированном виде, производить вычисления над полями базы данных, группировать записи и находить для полей итоговые значения с помощью статистических функций: *Sum*, *Avg*, *Max*, *Min,* *Count* и другие.

**2.4 Программная реализация и документирование базы данных**

Для визуального проектирования баз данных была выбрана программа *MySQL Workbench*, позволяющая интегрировать визуальное проектирование, моделирование, создание и эксплуатацию базы данных в единое бесшовное окружение для системы баз данных MySQL.

Для удобства сперва была разработана ER-диаграмма спроектированной базы данных, затем были установлены связи и столбцы в таблицах. На основе ER-диаграммы и схемы в среде MySQL Workbench, используя программные средства интерпретации визуального проектирования в автоматически генерируемый SQL-скрипт была создана база данных заданной предметной области – база данных для поддержки работы кафедры.

Чтобы подключить существующую базу данных к веб-серверу, необходимо подключить MySQL-соединение (MySQL Connection), открывающее доступ к базе данных посредством сетевых технологий. Стандартно MySQL разворачивает сервер с базой данных на локальном IP-адресе 127.0.0.1 (или localhost) и располагает оконечные точки для обмена сообщениями на порте 3306.

В качестве администратора соединения установлен пользователь с именем root и паролем root.

Для подключения к локальному серверу с базой данных необходимо ввести личные данные администратора в графическом пользовательском интерфейсе MySQL Workbench или выполнить команду mysql –u root –p в приветственной строке интерфейса командной строки (Command Line Interface, CLI).

Таким образом, спроектированная ранее визуально база данных была разработана программно и подключена к локальному серверу внутренними средствами утилиты MySQL Workbench, предоставляющей графический пользовательский интерфейс СУБД MySQL.

Программная реализация база данных на языке запросов представлена в Приложении А.

**2.5 Проектирование разрабатываемого приложения**

В рамках курсового проекта, для проверки работоспособности спроектированной базы данных было предложено разработать веб-приложение, реализующее клиент-серверное взаимодействие.

Клиент – это оконечный пользователь приложения, который видит результаты его работы непосредственно в окне браузера на экране персонального компьютера, смартфона или ноутбука. Сервер – вычислительный блок, расположенный в сети Интернет или развёрнутый локально на компьютере администратора веб-приложения, обращающийся к базе данных и обменивающийся *HTTP*-запросами с клиентской стороной, то есть браузером.

Первичное изучение подходов веб-разработки и анализ аналогичных существующих веб-систем, показал, что с архитектурной точки зрения разрабатываемая система будет состоять из нескольких слабо зависимых частей, каждая из которых будет служить определенной цели, и соответственно, иметь свой набор технологий. К таким частям относят:

**–** внешний интерфейс (*Frontend*);

**–** серверное *API* (*Backend*).

В качестве серверной стороны веб-приложения для обеспечения работоспособности кафедры университета был выбран язык программирования PHP, реализующий принципы ООП.

Само веб-приложение будет разработано на язык гипертекстовой разметки *HTML*. Для создания приятного и эргономичного пользовательского интерфейса использовался формальный язык описания внешнего вида документа *CSS*.Также был использован язык программирования, с помощью которого страницы сайтов становятся интерактивными – *JavaScript.*

**ЗАКЛЮЧЕНИЕ**

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Лукин, В. Н. Базы данных : конспект лекций / В. Н. Лукин – Москва : Московский авиационный институт, 2009. – 105 с.

[2] Studfiles. Файловый архив студентов. Инфологическая модель данных «сущность-связь» [Электронный ресурс]. – Режим доступа : https://studfiles.net/preview/720297/.

[3] Studfiles. Файловый архив студентов. Основы проектирования баз данных [Электронный ресурс]. – Режим доступа : https://studfiles.net/preview/1587882/page:2/.

# **ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Скрипт генерации базы данных**

**(к пункту 2.4)**

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema new\_test

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `new\_test` DEFAULT CHARACTER SET utf8 ;

-- -----------------------------------------------------

-- Schema my\_project

-- -----------------------------------------------------

USE `new\_test` ;

-- -----------------------------------------------------

-- Table `new\_test`.`prepods`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `new\_test`.`prepods` (

`user\_id` INT NOT NULL,

`first\_name` VARCHAR(45) NULL,

`last\_name` VARCHAR(45) NULL,

`fathers\_name` VARCHAR(50) NULL,

`location` VARCHAR(10) NULL,

`phone` VARCHAR(30) NULL,

`email` VARCHAR(50) NULL,

PRIMARY KEY (`user\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `new\_test`.`roles`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `new\_test`.`roles` (

`role\_name\_id` TINYINT NOT NULL,

`role\_name` VARCHAR(30) NULL,

PRIMARY KEY (`role\_name\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `new\_test`.`authorization`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `new\_test`.`authorization` (

`auth\_id` INT NOT NULL,

`login` VARCHAR(45) NULL,

`pass` CHAR(32) NULL,

`role\_id` TINYINT NOT NULL,

PRIMARY KEY (`auth\_id`),

INDEX `role\_id\_idx` (`role\_id` ASC) VISIBLE,

CONSTRAINT `auth\_id`

FOREIGN KEY (`auth\_id`)

REFERENCES `new\_test`.`prepods` (`user\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `role\_id`

FOREIGN KEY (`role\_id`)

REFERENCES `new\_test`.`roles` (`role\_name\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `new\_test`.`consultation`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `new\_test`.`consultation` (

`consult\_id` INT NOT NULL,

`user\_text\_consult` TEXT NULL,

PRIMARY KEY (`consult\_id`),

UNIQUE INDEX `user\_id\_UNIQUE` (`consult\_id` ASC) VISIBLE,

CONSTRAINT `consult\_id`

FOREIGN KEY (`consult\_id`)

REFERENCES `new\_test`.`prepods` (`user\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `new\_test`.`subjects`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `new\_test`.`subjects` (

`subject\_name\_id` INT NOT NULL,

`subject\_name` TEXT NULL,

PRIMARY KEY (`subject\_name\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `new\_test`.`courses`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `new\_test`.`courses` (

`user\_subj\_id` INT NOT NULL,

`subject\_of\_course\_id` INT NOT NULL,

PRIMARY KEY (`user\_subj\_id`),

INDEX `subject\_of\_course\_id\_idx` (`subject\_of\_course\_id` ASC) VISIBLE,

CONSTRAINT `user\_subj\_id`

FOREIGN KEY (`user\_subj\_id`)

REFERENCES `new\_test`.`prepods` (`user\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `subject\_of\_course\_id`

FOREIGN KEY (`subject\_of\_course\_id`)

REFERENCES `new\_test`.`subjects` (`subject\_name\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `new\_test`.`groups\_stud`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `new\_test`.`groups\_stud` (

`group\_name\_id` INT NOT NULL,

`speciality\_name` TEXT NULL,

PRIMARY KEY (`group\_name\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `new\_test`.`prepods\_job`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `new\_test`.`prepods\_job` (

`job\_id` INT NOT NULL,

`position\_name` VARCHAR(45) NULL,

`ac\_degree` TEXT NULL,

`science\_direction` TEXT NULL,

PRIMARY KEY (`job\_id`),

UNIQUE INDEX `user\_id\_UNIQUE` (`job\_id` ASC) VISIBLE,

CONSTRAINT `job\_id`

FOREIGN KEY (`job\_id`)

REFERENCES `new\_test`.`prepods` (`user\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `new\_test`.`timetable`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `new\_test`.`timetable` (

`table\_id` INT NOT NULL,

`subject\_id` INT NOT NULL,

`subject\_time\_start` TIME NULL,

`subject\_time\_end` TIME NULL,

`subject\_date` DATE NULL,

`subject\_location` VARCHAR(10) NULL,

`group\_number` INT NOT NULL,

PRIMARY KEY (`table\_id`),

INDEX `group\_number\_idx` (`group\_number` ASC) VISIBLE,

INDEX `subject\_id\_idx` (`subject\_id` ASC) VISIBLE,

CONSTRAINT `table\_id`

FOREIGN KEY (`table\_id`)

REFERENCES `new\_test`.`prepods` (`user\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `group\_number`

FOREIGN KEY (`group\_number`)

REFERENCES `new\_test`.`groups\_stud` (`group\_name\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `subject\_id`

FOREIGN KEY (`subject\_id`)

REFERENCES `new\_test`.`subjects` (`subject\_name\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;