МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал) СКФУ

# Методические указания

ПО ОРГАНИЗАЦИИ И ПРОВЕДЕНИЮ УЧЕБНОЙ ПРАКТИКИ
Технологическая (проектно-технологическая) практика
для направления подготовки **09.03.02 Информационные системы и технологии**
направленность (профиль) **Информационные системы и технологии обработки цифрового контента**

**Пятигорск**
**2024**

**Цели и задачи практики**

Проектно-технологическая практика является учебной и нацелена на ознакомление и изучение процесса создания и применения конкретных информационных технологий и систем информационного обеспечения для решения реальных задач организационной, управленческой или научной деятельности в условиях, приближенных к конкретным производствам и организациям.

**Требования к результатам освоения практики**

В результате прохождения практики обучающийся должен:

Знать:

- основные этапы проектирования информационных систем (программных комплексов);
- основные методы проектирования информационных систем;
- программные средства для проектирования информационных систем;
- методологию исследования предметной

области. Уметь:

- анализировать проблемные ситуации предметной области с целью выработки методов их решения;
- оформлять нормативную документацию, справочную и техническую документацию для информационных систем;
- анализировать и оценивать перспективы развития информационных систем с учетом их жизненного цикла;
- использовать инструментальные средства разработки программных средств информационных систем;
- осуществлять и контролировать выполнение требований по охране труда и технике безопасности в конкретной сфере деятельности;
- применять отечественные и зарубежные стандарты в области проектирования, разработки, отладки и тестирования программных комплексов на всех стадиях жизненного цикла информационных систем.

Владеть:

- методами количественного анализа процессов обработки, поиска и передачи информации;
- методикой разработки многомодульных программных комплексов;
- методикой выявления и анализа, потенциально существующих проблемных ситуаций информационных систем;
- методами проектирования информационных систем с использованием современных программных средств;
- методами управления разработкой программного

обеспечения. Задачами проектно-технологической практики

являются:

- методы системного анализа и моделирования информационной системы, анализ уровня аппаратного и программного обеспечения информационных систем и применяемых технологий;
- анализ аппаратных и программных средств, используемые при проектировании и эксплуатации информационных систем и их компонентов;
- порядок внедрения организациями новых аппаратных и программных средств, информационных систем и технологий;
- изучение всех этапов жизненного цикла информационных систем;
- умение формулировать ТЗ на проектируемые функциональные модули информационных систем;

- развитие практических навыков работы в предметно ориентированных информационных системах (Программы 1С, БЭСТ, Парус и т.д.);
- развитие практических навыков при проектировании информационных систем с использованием СУБД (SQL Server, MySQL, Oracle, Visual Fox Pro);
- развитие практических навыков построения распределенных систем по технологии NET, JAVA на базе технологий ODBC, OLE DB, ADO, JDBC;
- закрепление навыков администрирования серверных операционных систем (Windows, Linux);
- закрепление навыков сетевого программирования на базе сетевых протоколов;
- умение правильно рассчитать и подобрать необходимое оборудование для ЛВС в зависимости от назначения конкретной сети;
- создание Web- информационных систем на основе технологий NET (Java) и при использовании языков программирования PHP и Perl;
- закрепление навыков создания мультимедийных приложений.

Перечень осваиваемых компетенций

| Код компетенции | Формулировки |
|---|---|
| УК-2 | Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений |
| УК-3 | Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде |
| УК-4 | Способен осуществлять деловую коммуникацию в устной и письменной формах на государственном языке Российской Федерации и иностранном(ых) языке(ах) |
| ОПК-5 | Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем |
| ОПК-6 | Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий |
| ОПК-7 | Способен осуществлять выбор платформ и инструментальных программно- аппаратных средств для реализации информационных систем |
| ОПК-8 | Способен применять математические модели, методы и средства проектирования информационных и автоматизированных систем |

Обязанности студента-практиканта

Студент при прохождении практики обязан:

– выполнять задания, предусмотренные программой практики;

– вести дневник, форма и содержание которого представлена в методических рекомендациях по организации проведению практики студентов в университете, где фиксируются все виды работ, выполняемых в течение рабочего дня;

– по окончании практики отчитываться о проделанной работе и представить индивидуальный или групповой отчет и дневник руководителю.

Обязанности руководителя практики от университета

- разрабатывать и каждый год актуализировать программу практики,

- составлять календарный план практики;
- разрабатывать тематику индивидуальных заданий студентам;
- осуществлять контроль за соблюдением сроков практики и ее содержанием;
- оказывать методическую помощь студентам при выполнении ими индивидуальных заданий;
- оценивать результаты выполнения студентами программы практики;
- сдать студенческие отчеты и дневники практики для хранения с соответствующей записью в кафедральном журнале учета отчетов практик;
- по результатам практики подготовить письменный отчет руководителя практики.

Структура и содержание учебной практики

| №п/п | Разделы (этапы) практики | Виды учебной работы на практике, включая самостоятельную работу студентов и трудоемкость (в часах) | Формы текущего контроля |
|---|---|---|---|
| 1 | Подготовительный этап: Содержательная формулировка задач для решения в ходе практики, вида и объема результатов, которые должны быть получены. Изучение специальной литературы и другой научно-технической информации о достижениях отечественной и зарубежной науки и техники в соответствующей области. | 52 | Предоставление обзорно-аналитического раздела, отчета по практике |
| 2 | Основной этап: Постановка задачи. Выбор методов решения. Сбор и предварительная обработка исходных данных. Исследование предметной области, формализация задания, разработка приложения, | 52 | Предоставление систематизированного фактического и литературного материала основного раздела, отчета по практике |

| | | | |
|---|---|---|---|
| | документирование приложения | | |
| 3 | Заключительный<br><br>этап- подготовка и защита отчета по практике. | 4 | Защита отчета |
| | Итого | 108 | Зачет с оценкой |

Задания и порядок их выполнения

Задание на учебную практику включает проработку теоретического вопроса и разработку приложения, включаемого в отчет по практике.

Предлагаемые задания на практику

Варианты заданий:

1. . Информационно-справочная система магазина товаров.
2. База сотрудников ВУЗа.
3. Учет сотрудников предприятия (АО).
4. Справочник ВУЗов КМВ.
5. Справочник «Банки Ставропольского края».
6. Отдел кадров предприятия.
7. Система учета легкового автотранспорта.
8. Система учета библиотечного фонда.
9. Учет компьютерной и оргтехники предприятия.
10. Учет расходных материалов.
11. Касса аэрофлота (электронный билет).
12. Касса автовокзала (расписание, цена билета, количество мест).
13. Касса железнодорожного вокзала (электронный билет).
14. Система учета успеваемости студентов по кафедре СУиИТ.
15. Система Кинотеатр (количество мест, стоимость билетов, время сеансов).
16. Справочник магазинов по бытовой технике КМВ.
17. Регистратура студенческой поликлиники
18. Справочник санаториев КМВ (стоимость, количество мест).
19. Система учета автомобильного парка.
20. Телефонный справочник предприятий и организаций КМВ.
21. Адресный справочник памятных мест, организаций КМВ.
22. Расписание электропоездов по КМВ (Минводы, Пятигорск, Ессентуки, Кисловодск…).
23. Система учета товаров в книжном магазине.
24. Система учета вакансий и безработных мест в центре занятости г. Пятигорска.
25. Система заказов на товары в журнальном киоске.
26. Сотрудники кафедры СУиИТ (учебная нагрузка, расписание и т.д.).
27. Справочник по стройматериалам г. Пятигорска.
28. Справочник компьютерной техники и комплектующих г. Пятигорска.
29. Выдача заданий студентам заочного отделения.
30. Выдача и регистрация тем курсовых работ студентам очного и заочного отделения.
31. Электронный магазин.
32. Информационная система о врачах МУЗ Городская больница г. Пятигорск (данные врачей, время приема и т.д.).
33. Информационная система банков КМВ.
34. Подсистема авторизации, регистрации и учета пользователей в системе дистанционного образования.
35. Информационная система учёта клиентов фирмы.

7.1 Примерная структура отчета

Отчет по практике должен
содержать:

- титульный лист (Приложение 2);
- задание на практику (Приложение 3);
- содержание;
- введение;
- основная часть:

Раздел «1. Проектирование функциональной части информационной системы»:

- исходные данные;

- цели автоматизированной системы и автоматизированные функции;
- характеристика функциональной структуры;
- разработка информационно-логической структуры базы данных;

Раздел «2. Проектирование и разработка распределенной базы и клиентского приложения»

- предлагаемые типовые решения и обоснование выбора технологии реализации;
- разработка WEB-приложения пользователя;
- заключение;
- список использованных источников;
- приложения.

7.2. Основные разделы проектирования

7.2.1. Введение

Этот раздел должен содержать описание актуальности поставленной задачи. Определить основные направления и структуру проекта.

7.2.2. Проектирование функциональной части информационной системы

Материал по данному разделу подбирается студентом самостоятельно и должен содержать следующие подразделы:

- исходные данные;
- цели автоматизированной системы и автоматизированные функции;
- характеристика функциональной структуры;
- разработка информационно-логической структуры базы данных;

Результаты выполненной работы в виде аргументированных предложений использовать при написании подразделов отчета, перечисленных в задании на практику.

Исходные данные

В данном подразделе необходимо привести перечень документов предметной области, которые используются в проектируемой распределенной базе данных. Провести анализ информации в этих документах, систематизировать данные для их представления в базе данных. В целях наиболее полного учета факторов, влияющих на проектирование функциональной части объекта автоматизации, руководствуясь «Правилами проведения работ при создании автоматизированных систем», а также были проанализированы:

технические задания на создание систем, которые можно рассматривать прообразами проектируемой системы;

государственные стандарты, руководящие документы и материалы, идентификационные номера, справочники, классификаторы, законодательство Российской федерации, международные и зарубежные стандарты, спецификация форматов, интерфейсов, протоколов;

требования нормативных документов, определяющих современные уровни требований к параметрам и характеристикам проектируемых систем, а также регламентирующих процесс выполнения разработок.

При разработке справочников и классификаторов, нужно руководствоваться нормативными документами (общероссийскими классификаторами).

Цели автоматизированной системы и автоматизируемые функции

Подраздел «Цели автоматизированной системы и автоматизируемые функции» включаются, как правило, текстовые формулировки, возможно, с включением табличного материала (при необходимости). Формулировки должны соответствовать видению конечных целей автоматизации и перечня тех функций, которые следует автоматизировать для достижения этих целей. Следует обратить внимание на то, что:

«цели» и «функции» должны соответствовать друг другу;

при формулировке «цели» и «функций» целесообразно учитывать опыт

«автоматизаторов», работавших до Вас (обычно этот опыт воплощается в форме принципов, опубликованного опыта, ГОСТов и пр.).

Например: Цели автоматизированной системы и автоматизируемые функции. Целью создания системы является повышение эффективности решения задачи, за счет автоматизации на современном уровне функций основных должностных лиц (учреждения), участвующих в процессе. Для достижения цели необходима автоматизация следующих функций (таблица 7.1).

Таблица 7.1. - Функции, подлежащие автоматизации

| Наименование группы функций | Функции |
|---|---|
| 1 | 2 |
| Информационная поддержка процесса обслуживания заказов | Ведение БД в объеме, необходимом для совместного функционирования всех должностных лиц фирмы, оснащенных средствами автоматизации, включая первоначальный ввод информации, внесение изменений и дополнений (корректуру) в данные, учтенные ранее; формирование отчетов, предусмотренных регламентом (ежедневных, еженедельных, месячных, квартальных, годовых), а также получение справочных данных произвольного содержания в объеме предоставленных полномочий; автоматизацию подсчетов при формировании отчетов и представление их в виде, обеспечивающем максимальную наглядность (в частности, сопоставимость результатов с аналогичными периодами предшествующей деятельности, возможность выдачи данных не только в форме таблиц, но и в виде диаграмм и графиков); автоматическое информирование пользователя о появлении продукции, подлежащей доставке на почту;- автоматизированную распечатку наклеек для скомплектованных наборов, подлежащих отправке |
| Информационная поддержка бухгалтерского учета | ввод и обработка бухгалтерских записей; печать первичных документов и отчетности. |
| Информационная поддержка правовой стороны бизнеса | получение справок по действующему законодательству; поддержание правовой базы данных на уровне актуальности без участия пользователя; возможность получения последних форм документов, предусмотренных законодательством для оформления договоров, отчетов, контрактов и пр. |
| Планирование и организация служебной деятельности | ведение ежедневника (календаря) с заданием событий (работ, задач) и плановых сроков (интервалов) выполнения;<br>ведение записной книжки с адресами, телефонами, характеристикой деловых партнеров и пр.; напоминание о делах (событиях, встречах) за несколько: минут, дней, недель, месяцев, ежегодно;<br>автоматизацию телефонного общения (планирование, напоминание, автоматизированный набор №, автодозвон, фиксация времени, даты и длительности, заметок о разговоре путем набора с клавиатуры); планирование производственных дел (длительных событий: день и более) с учетом взаимоувязки с планами деловых партнеров (при планировании совместных мероприятий предварительно запрашиваются данные о свободных ресурсах);<br>возможность планирования личных (защищенных от других) и |

| | |
|---|---|
| | коллективных мероприятий. |
| Обеспечение электронных | поддержание режима удаленной работы руководителя организации с целью получения справок, отчетов и др. данных; поддержка коммуникаций с |

| | |
|---|---|
| коммуникаций | вышестоящей организацией по факсу и телефону; поддержка коммуникаций с взаимодействующими организациями по факсу, телефону и электронной почте; поддержка коммуникаций с заказчиками организациями по факсу, телефону и электронной почте. |

Характеристика функциональной структуры

В раздел "Характеристика функциональной структуры" включают:

перечень подсистем АС (автоматизированной системы) с указанием функций и (или) задач, реализуемых в каждой подсистеме;

описание процесса выполнения функций (при необходимости);

необходимые пояснения к разделению автоматизированных функций на действия (операции), выполняемые техническими средствами и человеком;

требования к временному регламенту реализации автоматизируемых функций и решения задач;

схему функциональной структуры.

При работе над данным разделом целесообразно учесть следующие рекомендации:

1) Разделение выполняемых функций на:

«автоматизированные» - реализуемые с использованием средств автоматизации, и

«ручные» - реализуемые без средств автоматизации, рекомендуется осуществлять исходя из возможностей, закладываемых в пакеты прикладных программ, которые изучались в соответствующих дисциплинах;

2) требования к временному регламенту реализации функций чаще всего представляются в табличном виде. В этой таблице отражается организация документооборота, предлагаемая к использованию после внедрения системы. Обычно эта организация базируется на уже сложившемся (привычном) порядке обмена данными, но с учетом возможных изменений, обусловленных особенностями функционирования с использованием средств автоматизации;

3) схема функциональной структуры системы должна показывать:

элементы функциональной структуры автоматизированной системы (АС);

связи между ними и внешней средой с кратким указанием содержания сообщений и (или) сигналов, передаваемых по связям.

При этом:

описание должно раскрывать информационные взаимосвязи между автоматизированными и неавтоматизированными функциями;

в зависимости от степени детализации описания системы элементами функциональной структуры могут быть:

подсистемы АС;

автоматизируемые функции и (или) задачи;

совокупности действий (операций), выполняемых при реализации автоматизируемых функций только техническими средствами (автоматически) или только человеком;

на схеме указывают информационные связи между элементами и, при необходимости, связи других типов (входимости, подчинения и т.п.);

кроме функциональной структуры АС в целом, при необходимости, разрабатываются детализированные схемы частей функциональной структуры.

Например, характеристика функциональной структуры информационной системы.

Анализ исходных данных, перечисленных выше, и выявленных функций, подлежащих автоматизации, а также имеющихся сведений о возможностях современных пакетов прикладных программ позволили предложить следующее обобщенное описание функциональной структуры проектируемой системы. На схеме показан состав элементов

функциональной структуры и связи между ними и внешней средой.

Взаимодействие элементов обеспечивается:

электронными средствами обмена данными - коммуникационной подсистемой (на схеме связи показаны сплошными линиями) и

традиционными средствами (курьерской связью, почтовой связью и пр., на схеме взаимодействие элементов показано пунктиром).

Предлагаемый регламент обмена данными представлен в табличной форме. Порядковые номера сообщений (документов, запросов и пр.), представлены на схеме рядом с соответствующей связью по правилам изображения комментариев.

Реализацию перечисленных подсистем предлагается осуществлять, на основе типовых решений и, путем использования технологии оригинального проектирования - Mysql, Microsoft SQL Sever, Visual FoxPro, Microsoft Access.

Разработка информационно-логической структуры базы данных (ИЛМ)

В разделе «Разработка информационно-логической структуры БД» необходимо представить ИЛМ базы данных для заданной предметной области. ИЛМ отображает данные предметной области в виде совокупности информационных объектов и связей между ними. Эта модель представляет данные, подлежащие хранению в базе данных.

Информационный объект - это описание некоторой сущности предметно области - реального объекта, процесса, явления или события. Информационный объект образуется совокупностью логически взаимосвязанных реквизитов, представляющих качественные и количественные характеристик сущности. Примерами информационных объектов могут быть: ТОВАР, ПОСТАВЩИК, ЗАКАЗЧИК, ПОСТАВКА, ОТГРУЗКА, СОТРУДНИК, ОТДЕЛ, СТУДЕНТ, ПРЕПОДАВАТЕЛЬ, КАФЕДРА и т. п.

Информационные объекты могут быть выделены на основе описания предметной области путем определения функциональных зависимостей между реквизитами. Совокупность реквизитов информационного объекта должна отвечать требованиям нормализации.

Каждому информационному объекту нужно присвоить уникальное имя, например, СТУДЕНТ, ПРЕДМЕТ, ПРЕПОДАВАТЕЛЬ, КАФЕДРА.

Информационный объект имеет множество реализаций или экземпляров. Например, каждый экземпляр объекта СТУДЕНТ представляет конкретного студента. Экземпляр объекта образуется совокупностью конкретных значений реквизитов и должен однозначно определяться, т.е. идентифицироваться значением ключа информационного объекта, который состоит из одного или нескольких ключевых реквизитов. Таким образом, реквизиты подразделяются на ключевые и описательные, которые являются функционально зависимыми от ключа.

Функциональная полная зависимость реквизитов имеет место в том случае, если одному значению ключа соответствует только одно значение описательного (зависимого) реквизита.

Заметим, что при выявлении функциональных зависимостей реквизитов не рассматриваются арифметические зависимости (например, стоимость от количества), поскольку устанавливается только функциональная зависимость, определяющая связи описательных и ключевых реквизитов, на основе которой и выявляется реквизитный состав каждого информационного объекта.

При графическом изображении модели данных каждый информационный объект представляется прямоугольником с обозначением его имени и идентификатора-ключа. Пример такого изображения для информационных объектов ТОВАР и ПОСТАВКА показан на рис. 7.1. Здесь KODT (код товара) – простой ключ объекта ТОВАР, а KODT + KPOST (код поставщика) - составной ключ объекта ПОСТАВКА.

| Товар | | Поставка |
|---|---|---|
| KODT | | KODT+KPOST |

Рисунок 7.1. - Пример графического изображения информационных объектов с простым и составным ключом

Требования нормализации. Реквизиты каждого информационного объекта должны отвечать требованиям нормализации: информационный объект должен содержать уникальный идентификатор - ключ:

Ключ является простым, если он состоит из одного реквизита, или составным, если из нескольких.

Все описательные реквизиты должны быть взаимонезависимы, т.е. между ними не должно быть функциональных зависимостей.

Все реквизиты, входящие в составной ключ, должны быть также взаимонезависимы.

Каждый описательный реквизит должен функционально полно зависеть от ключа, т.е. каждому значению ключа должно соответствовать только одно значение описательного реквизита.

При составном ключе описательные реквизиты должны зависеть целиком от всей совокупности реквизитов, образующих ключ.

Каждый описательный реквизит не должен зависеть от ключа транзитивно, т. е. через другой промежуточный реквизит.

В случае транзитивной зависимости между реквизитами можно выполнить расщепление совокупности реквизитов с образованием двух информационных объектов вместо одного.

Выполнение требований нормализации обеспечивает построение реляционной базы данных без дублирования данных и возможность поддержания целостности при внесении изменений.

Выделение информационных объектов предметной области

Процесс выделения информационных объектов предметной области, отвечающих требованиям нормализации, может производиться на основе интуитивного или формального подхода.

При интуитивном подходе легко могут быть выявлены информационные объекты, соответствующие реальным объектам. Однако получаемая при этом информационно-логическая модель, как правило, требует дальнейших преобразований, в частности, преобразования много -многозначных связей между объектами. При таком подходе возможны существенные ошибки, если отсутствует достаточный опыт. Последующая проверка выполнения требований нормализации обычно показывает необходимость уточнения информационных объектов.

Порядок выделения информационных объектов. Рассмотрим формальные правила, которые могут быть использованы для выделения информационных объектов, отвечающих требованиям нормализации:

На основе описания предметной области выявить документы и их реквизиты, подлежащие хранению в базе данных.

Определить функциональные зависимости между реквизитами.

Функциональную зависимость реквизитов можно изобразить графически в виде линий со стрелками, идущих от ключевого реквизита к описательному (зависимому). Эти зависимости целесообразно отразить непосредственно в таблице, где представлен состав реквизитов, сгруппированных по документам.

Выбрать все зависимые реквизиты и указать для каждого все его ключевые реквизиты, то есть те, от которых он зависит (один или несколько).

В случае транзитивной зависимости некоторые реквизиты являются одновременно

зависимыми и ключевыми и соответственно будут представлены в группе зависимых и ключевых.

Сгруппировать реквизиты, одинаково зависимые от ключевых реквизитов. Полученные группы зависимых реквизитов вместе с их ключевыми реквизитами образуют информационные объекты.

Заметим, что при использовании приведенных правил не требуется отдельно преобразовывать транзитивные зависимости реквизитов.

После выделения информационных объектов надо дать их окончательное описание. В таком описании кроме состава реквизитов и указания ключа может быть представлена также семантика информационных объектов, то есть их смысловое определение. Затем можно осуществить контрольную проверку выполнения требований нормализации. Как правило, при использовании приведенных правил сразу оказываются выделенными объекты, выполняющие роль связки между объектами, находящимися в отношении многие ко многим (M:N) и, соответственно, в модели могут рассматриваться только одно-многозначные связи. Совокупность выделенных в результате информационных объектов позволяет получить информационно-логическую модель, не требующую дальнейших преобразований для создания реляционной базы данных, отвечающей требованиям нормализации.

Примеры выделения информационных объектов

Рассмотрим выделение информационных объектов на примере предметной области «Учебный процесс».

Описание предметной области.

Пусть необходимо построить базу данных, содержащую информацию об учебном процессе текущего семестра:

Списки студентов групп.

Перечень изучаемых предметов.

Преподавательский состав кафедр, обеспечивающих учебный процесс.

Сведения о лекционных и практических занятиях в каждой из групп.

Результаты сдачи экзаменов (зачетов) по каждому из проведенных занятий.

В результате анализа предметной области выявляются документы, являющиеся источниками данных для создания базы данных.

Особо отметим, что документы предметной области не только дают возможность выявить структуру данных, но также являются основой для разработки форм ввода-вывода.

Связи информационных объектов

Следующим шагом проектирования после выявления информационных объектов является определение связей между ними. Связь устанавливается между двумя информационными объектами. Наличие связи, как правило, определяется природой реальных объектов, процессов или явлений, отображаемых этими информационными объектами. Связи между объектами существуют, если логически взаимосвязаны экземпляры из этих информационных объектов. Например, связи имеются между такими парами объектов, как поставщик-товар, склад - готовая продукция, кафедра-преподаватель, группа-студент и т. п.

Тип связи информационных объектов.

Связи информационных объектов могут быть разного

типа: Одно-однозначные (1:1).

Одно-многозначные (1:M).

Много-многозначные (M:N).

Одно-однозначные связи имеют место, когда каждому экземпляру первого объекта (A) соответствует только один экземпляр второго объекта (B) и наоборот, каждому экземпляру второго объекта (B) соответствует только один экземпляр первого объекта (A). Следует заметить, что такие объекты легко могут быть объединены в один, структура которого образуется объединением реквизитов обоих исходных объектов, а в качестве ключевого

реквизита может быть выбран любой из альтернативных ключей, т.е. ключей исходных объектов. Графическое изображение одно-однозначной связи приведено на рис. 7.2.

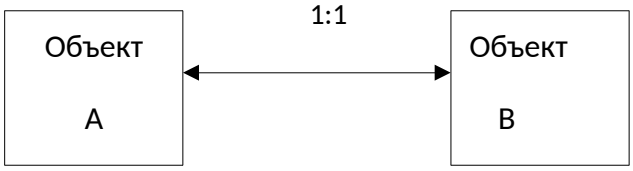


Рисунок 7.2 - Графическое изображение одно-однозначных отношений объектов.

Одно-многозначные связи (1:M) характеризуются тем, что каждому экземпляру одного объекта (A) может соответствовать несколько экземпляров другого объекта (B), а каждому экземпляру второго объекта (B) может соответствовать только один экземпляр первого объекта (A). Графическое изображение одно-многозначной связи приведено на рис. 7.3.

В такой связи объект A является главным, а объект B - подчиненным, то есть имеет место иерархическая подчиненность объекта B объекту A. Примерами одно-многозначных связей являются: подразделения-сотрудники, специализированный склад-готовая продукция и т. п.

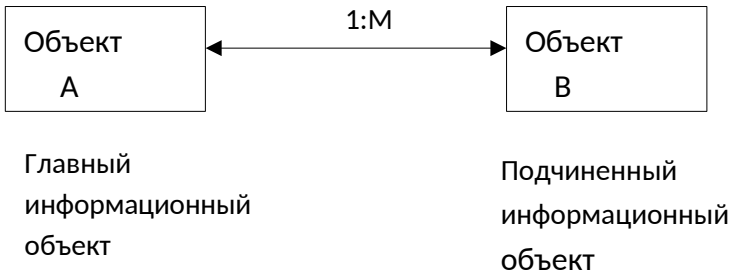


Рисунок 7.3 - Графическое изображение одно-многозначных отношений объектов

Много-многозначные связи (M:N). Каждому экземпляру одного объекта (A) могут соответствовать несколько экземпляров второго объекта (B) и наоборот, каждому экземпляру второго объекта (B) могут соответствовать тоже несколько экземпляров первого объекта (A). Графическое изображение связи типа M:N показано на рис. 7.4.

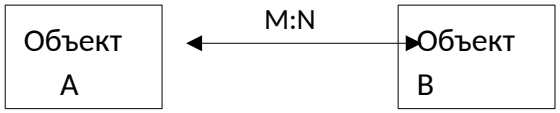


Рисунок 7.4 - Графическое изображение связи типа M:N

Много-многозначные связи не могут непосредственно реализовываться в реляционной базе данных. Поэтому, если такие связи выявлены, может понадобиться их преобразование путем введения дополнительного объекта «связка». Исходные объекты будут связаны с этим объектом одно-многозначным связями. Таким образом, объект-связка является подчиненным в одно-многозначных связях по отношению к каждому из исходных объектов.

На рис. 7.5 показано преобразование связи типа M:N через объект, выполняющий роль «связки». Объект-связка должен иметь идентификатор, образованный из идентификаторов исходных объектов, например, Ка и Кв.

При рассмотренном выше подходе к выделению информационных объектов объект-связка, как правило, выявляется в результате анализа функциональных зависимостей реквизитов. Много-многозначные связи в этом случае не требуют специальной реализации, т. к. осуществляются через объект, выполняющий роль объекта-связки.

Примером много-многозначных связей может быть связь поставщики-товары, если один поставщик поставляет разные наименования товаров, а товар одного наименования поставляется несколькими поставщиками.

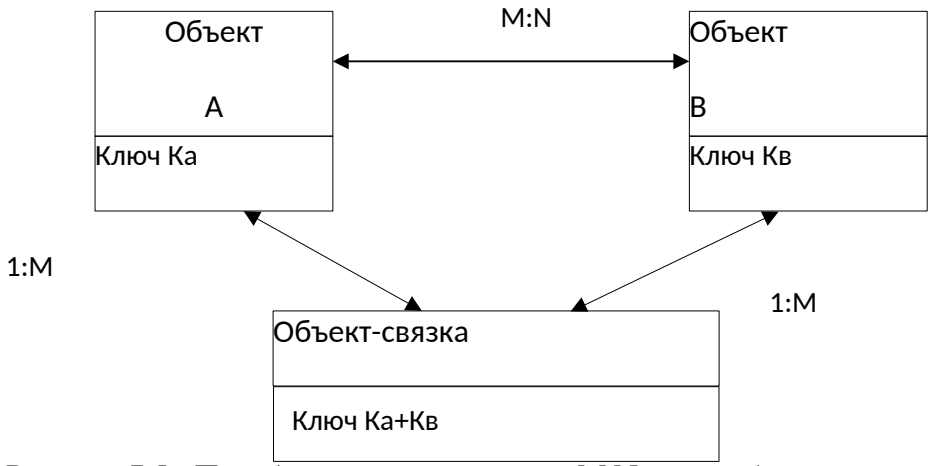


Рисунок 7.5 - Преобразование связи типа M:N через объект-связку.

Пример информационно-инфологической модели предметной области «учебный процесс».

На рис. 7.6 представлена информационно-логическая модель предметной области «учебный процесс», построенная в соответствии с выявленными информационными объектами и связями между ними. Информационно-логическая модель приведена в каноническом виде, так как объекты размещены по уровням. На нулевом уровне размещаются объекты, не подчиненные никаким другим объектам. Уровень остальных объектов определяется наиболее длинным путем к объекту от нулевого уровня.

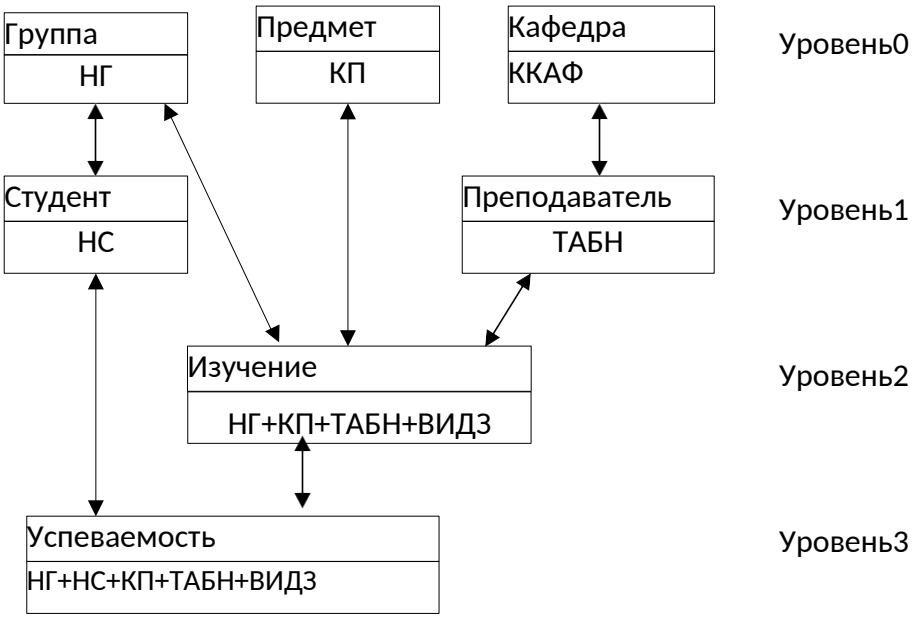


Рисунок 7.6 - Информационно-логическая модель предметной области "Учебный

процесс"

Такое размещение объектов дает представление об их иерархической подчиненности, делает модель более наглядной и облегчает понимание одно-многозначных отношений между объектами.

Логическая структура реляционной базы данных

Логическая структура реляционной базы данных является адекватным отображением полученной информационно-логической модели, не требующим дополнительных преобразований. Каждый информационный объект модели данных отображается соответствующей реляционной таблицей. Структура реляционной таблицы определяется реквизитным составом соответствующего информационного объекта, где каждый столбец (поле) соответствует одному из реквизитов объекта. Ключевые реквизиты объекта образуют уникальный ключ реляционной таблицы. Для каждого столбца задается формат и размер данных. Строки (записи) таблицы соответствуют экземплярам объекта и формируются при загрузке таблицы.

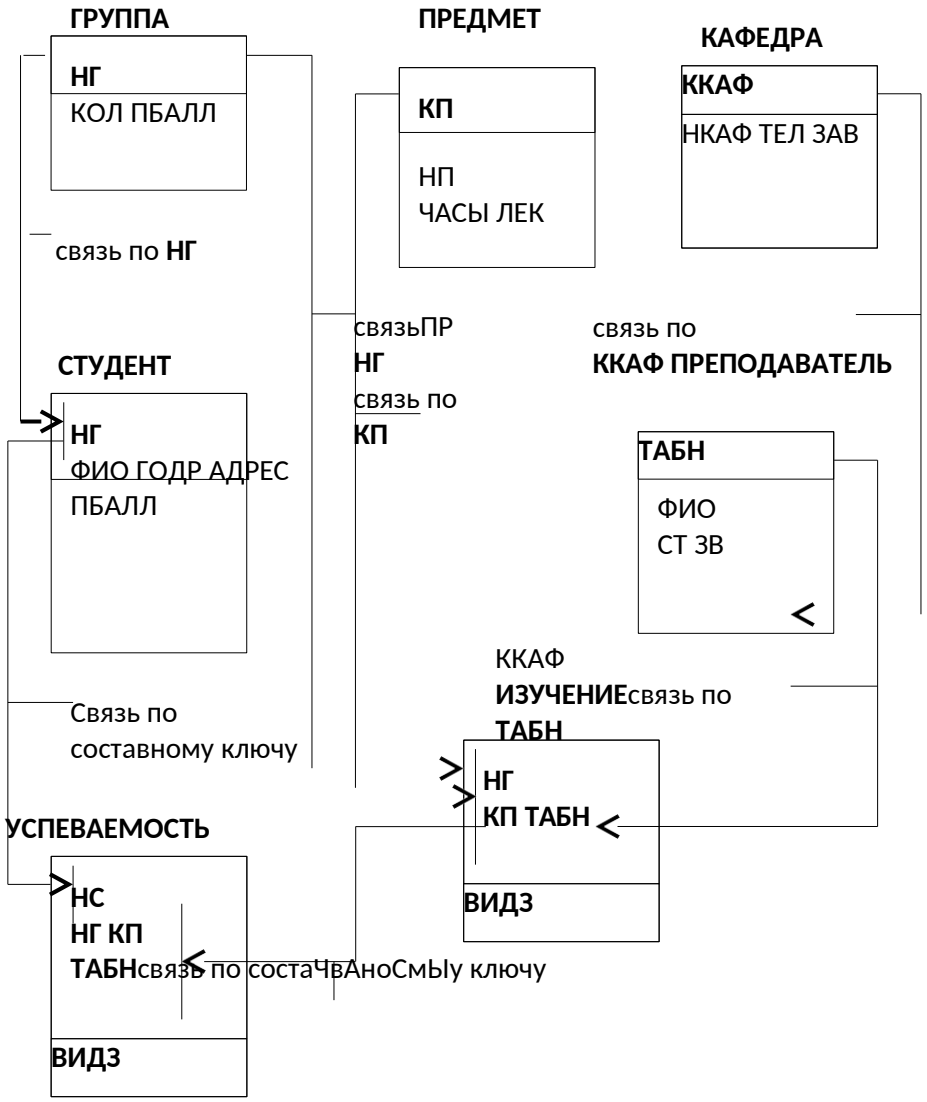


Рисунок 7.7 - Логическая структура реляционной базы данных предметной области Учебный процесс

Связи между объектами модели данных реализуются одинаковыми реквизитами - ключами связи в соответствующих таблицах. При этом ключом связи всегда является
уникальный ключ главной таблицы. Ключом связи в подчиненной таблице являются либо некоторая часть уникального ключа в ней, либо поле, не входящее в состав первичного ключа. Ключ связи в подчиненной таблице называют внешним ключом.

В СУБД может быть создана схема данных, наглядно отображающая логическую структуру базы данных. Определение одно-многозначных связей в этой схеме должно осуществляться в соответствии с построенной моделью данных. Внешний вид схемы данных практически совпадает с графическим представлением информационно-логической модели. Для модели данных, построенной в рассмотренном примере, логическая структура базы данных в виде схемы данных приведена на рис. 7.7. На этой схеме прямоугольники отображают таблицы БД с полным списком их полей, а связи показывают, по каким полям осуществляется взаимосвязь таблиц. Имена ключевых полей для наглядности выделены и находятся в верхней части полного списка полей каждой таблицы.

В заключение отметим, что рассмотренные выше этапы проектирования базы данных,

основанные на построении модели данных предметной области, позволяют легко получить

логическую структуру реляционной базы данных.

7.2.3. Проектирование и разработка распределенной базы и клиентского приложения

Раздел "Проектирование и разработка распределенной базы и клиентского приложения" должен содержать следующие подразделы:

предлагаемые типовые решения и обоснование выбора технологии реализации; разработка приложения пользователя;

Предлагаемые типовые решения и обоснование выбора технологии реализации

В данном подразделе необходимо описать выбранный инструментарий, распределенную базу данных, описать основные объекты и их характеристики. Обосновать выбор технологии решения задачи.

Проектирование базы данных

База данных - это файл или набор файлов, хранящий структурированную определенным образом информацию. Для обработки этой информации используются особые программы, называемые системами управления базами данных, или СУБД. Примером такой СУБД будем рассматривать MySQL.

Реализация таких аспектов как разработка информационной системы рассмотрим в WEB-технологии в среде PHP и MySQL.

MySQL - это сервер данных. Особая программа, позволяющая хранить упорядоченные данные в особых структурах, называемых базами данных. Например, в такую базу можно записать инвентарную книгу, список работников, экзаменационные ведомости за несколько лет, список книг, хранящихся в библиотеке, и многое другое. И не просто записать, и сохранить, а еще и получать эти данные по первому запросу. И не просто получать, а быстро, без особого труда и только те данные, которые были запрошены.

PHP - это платформа для создания серверных Web-страниц. Грубо говоря, с ее помощью можно писать программы, которые по запросу посетителя сайта создают Web-страницы и отправляют их ему. Такие Web-страницы могут содержать данные, хранящиеся в базе данных, в том числе, и MySQL.

Например, серверная программа, написанная на PHP и являющаяся частью сайта интернет-магазина, может извлекать из базы данных MySQL список товаров и формировать из него красивую Web-страницу. А другая серверная программа, другая часть того же интернет-магазина, спрашивает посетителя, какой товар он хочет заказать и по какому адресу этот товар нужно выслать, и сама записывает в ту же базу данных сведения о новом заказе.

PHP (рекурсивный акроним для "PHP: Hypertext Preprocessor") это широко распространённый Открытый ресурс - язык скриптинга (сценариев) общего назначения, который создан специально для Web и который можно внедрять в HTML.

Пример:

```
<html>
  <head>
    <title>Пример</title>
  </head>
  <body>

    <?php
    echo "Hi, I'm a PHP script!";
    ?>

  </body>
</html>
```

Заметьте, как это отличается от скриптов, написанных на языках Perl или C - вместо

написания программы с большим количеством команд для вывода HTML, вы пишете HTML-скрипт с некоторым количеством встроенного кода для выполнения каких-либо действий (в данном случае - для вывода некоторого текста). Код PHP заключён в специальные начальный и конечный тэги, что позволяет вам входить в и выходить из «режима PHP».

PHP отличается от других подобных языков, типа клиентского JavaScript, тем, что код выполняется на сервере. Если вы имеете скрипт, аналогичный вышеприведённому на сервере, то клиент получит результат работы этого скрипта, не имея возможности определить, каков был исходный код. Вы также можете сконфигурировать ваш web-сервер таким образом, чтобы он обрабатывал все ваши HTML-файлы с помощью PHP, и реально пользователь не будет иметь способа определить, что у вас "в рукаве".

Наилучшим качеством PHP является то, что он предельно прост для новичка в программировании, но предлагает много продвинутых возможностей для программиста-профессионала.

Когда PHP разбирает файл, он просто передаёт текст файла, пока не обнаружит один из специальных тэгов, который говорит о необходимости начать интерпретацию текста как кода PHP. Разборщик выполняет весь найденный код до закрывающего тэга PHP, который говорит разборщику, что нужно снова начать просто передавать текст. Этот механизм позволяет внедрять PHP-код в HTML: всё за пределами тэгов PHP остаётся без изменений, а внутри тэгов - разбирается как код.

Имеются четыре набора тэгов, которые используются для обозначения блоков кода PHP. Только два из них (<?php. . .?> и <script language="php">. . .</script>) всегда доступны; другие можно включать и отключать из файла конфигурации php.ini. Хотя сокращённые тэги и тэги в стиле ASP могут быть удобны, они не так переносимы, как их длинные версии. Также, если вы предполагаете внедрять PHP-код в XML или XHTML, нужно использовать форму <?php. . .?> для соответствия XML.

Тэги, поддерживаемые PHP:

Пример:

1. <?php echo("если вы хотите работать с документами XHTML или XML, делайте так\n"); ?>
2. <? echo ("это простейшая SGML-инструкция процессинга\n"); ?>
   <?= выражение ?> Это аббревиатура для "<? echo выражение ?>"
3. <script language="php">
   echo ("некоторые редакторы (вроде FrontPage) не любят инструкции процессинга");
   </script>
4. <% echo ("Вы можете по выбору использовать тэги в стиле ASP"); %>
   <%= $variable; # Это аббревиатура для "<% echo . . ." %>

Первый способ, <?php. . .?>, это предпочтительный метод, так как он позволяет использовать PHP в коде, соответствующем правилам XML, таком как XHTML.

Второй способ вообще невозможен. Сокращённые тэги доступны только тогда, когда они подключены. Это можно сделать функцией short_tags() (только в PHP 3), включив установку конфигурации short_open_tag в PHP config-файле, или скомпилировав PHP с опцией --enable-short-tags в configure. Даже если вы по умолчанию включили в php.ini-dist, использование сокращённых тэгов не рекомендуется.

Четвёртый способ доступен, только если тэги в стиле ASP включены с использованием установки конфигурации asp_tags.

Примечание: использование сокращённых тэгов должно быть исключено при разработке приложений или библиотек, предназначенных для распространения или при публикации на PHP-серверы, которые вами не контролируются, поскольку сокращённые тэги могут не поддерживаться на целевом сервере. Для обеспечения переносимости и распространения кода не используйте сокращённые тэги.

Закрывающий тэг блока будет иметь следом за собой ведомый символ newline, если он имеется. Также закрывающий тэг автоматически подразумевает точку с запятой; вам не нужно также вводить символ "точка с запятой" в конце последней строки PHP-блока.

PHP позволяет использовать структуры такого вида:

Пример:

```
<?php
if ($expression) {
   ?>
   <strong>Это правильно.</strong>
   <?php
} else {
   ?>
   <strong>Это неправильно.</strong>
   <?php
}
?>
```

Этот код работает так, как ожидается, поскольку, когда PHP встречает закрывающие тэги ?>, он просто начинает выводить всё, что обнаруживает после них, до обнаружения другого открывающего тэга. Этот пример, конечно, надуманный, но при выводе больших блоков текста выход из режима разбора PHP обычно более эффективен, чем отправка всего текста через echo() или print() или что-нибудь похожее.

Некоторые функции mySQL

Эти функции дают доступ к серверам баз данных mySQL. Чтобы иметь возможность работать с этими функциями, вы обязаны скомпилировать PHP с поддержкой mysql, используя опцию --with-msql[=dir].

С помощью опции конфигурации --with-mysql вы включаете доступ PHP к БД MySQL. Если вы используете эту опцию без специфицирования пути к MySQL, PHP будет использовать встроенные клиентские библиотеки MySQL.

mysql_connect - открывает соединение с MySQL-сервером

mysql_create_db - создаёт БД MySQL

mysql_data_seek - перемещает внутренний результирующий указатель

mysql_db_name - получает результирующие данные

mysql_db_query - отправляет MySQL query

mysql_drop_db - удаляет БД MySQL

mysql_errno - возвращает числовое значение сообщения об ошибке из предыдущей MySQL-операции

mysql_error - возвращает текст сообщения об ошибке из предыдущей MySQL-операции

mysql_query - отправляет MySQL query

mysql_select_db - выбирает БД MySQL

Переменные в PHP представлены знаком dollar ($) с последующим именем переменной. Имя переменной чувствительно к регистру символов.

Имена переменных следуют тем же правилам, что и другие метки в PHP. Правильное имя переменной начинается с буквы или символа подчёркивания, с последующими (в любом количестве) буквами, числами или символами подчёркивания. Это можно выразить в виде регулярного выражения:

'[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'

Примечание: для наших целей здесь - буквы это a-z, A-Z и ASCII-символы от 127 до 255 (0x7f-0xff).

$var = "Bob";

```
$Var = "Joe";
echo "$var, $Var";      // выводит "Bob, Joe"

$4site = 'not yet';     // неправильно; начинается с числа
$_4site = 'not yet';    // правильно; начинается с символа подчёркивания/underscore
$täyte = 'mansikka';    // правильно; 'ä' это ASCII 228.
```

Для присвоения по ссылке просто присоедините амперсанд (&) к началу имени переменной (исходной переменной). Например, следующий фрагмент кода выводит 'My name is Bob' дважды:

```
<?php
$foo = 'Bob';              // присваивается 'Bob' переменной $foo
$bar = &$foo;              // ссылка на $foo через $bar
$bar = "My name is $bar"; // изменение $bar
echo $bar;
echo $foo;                 // $foo также изменилась
?>
```

Важно отметить, что по ссылке можно присвоить только именованные переменные.

```
<?php
$foo = 25;
$bar = &$foo;      // правильное присвоение
$bar = &(24 * 7);  // неверно; ссылка на неименованное выражение

function test()
{
  return 25;
}
$bar = &test();    // неправильно
?>
```

Любой PHP-скрипт состоит из серии операторов. Это может быть присвоение, вызов функции, цикл, условный оператор или даже оператор, который ничего не делает (пустой оператор). Оператор обычно завершается точкой с запятой. Кроме того, операторы можно группировать с помощью фигурных скобок {}. Группа операторов сама также является оператором.

Конструкция if является одной из ключевых во многих языках, в том числе и в PHP. Она позволяет выполнять фрагменты кода при выполнении условия. PHP предлагает структуру if, которая аналогична такой же структуре языка C: if (expr) statement

Expr вычисляется в булево значение. Если expr вычисляется в TRUE, PHP выполнит statement, а если вычисляется в FALSE - оператор игнорируется.

Следующий пример выведет a is bigger than b, если $a больше $b:

```
if ($a > $b)
   print "a is bigger than b";
```

Часто необходимо выполнить по условию не один, а несколько операторов. Разумеется, нет необходимости создавать для каждого оператора конструкцию if. Вместо этого вы можете сгруппировать несколько операторов в блок. Например, этот код выведет a is bigger than b, если $a больше $b, а затем присвоит значение переменной $a переменной $b:

```
if ($a > $b) {
   print "a is bigger than b";
   $b = $a;
}
```

Операторы if могут вкладываться друг в друга, что даёт вам полную свободу при условном выполнении различных частей программы.

Синтаксис PHP довольно простой. Программы на PHP встраиваются в текст web-страницы так же, как и сценарии на JavaScript, VBScript, при помощи окаймляющих угловых скобок с вопросительными знаками и указанием языка:

<?php

...

текст программы

...

?>

Команды PHP обязательно разделяются символом точки с запятой - ";" (символ конца абзаца или конца строки не учитывается никак), после последней в программе команды его можно не ставить. Также символ ";" не ставится после условных операторов (if, switch) и операторов цикла (for, while и других). Пример программы на PHP - на рис. 7.8.

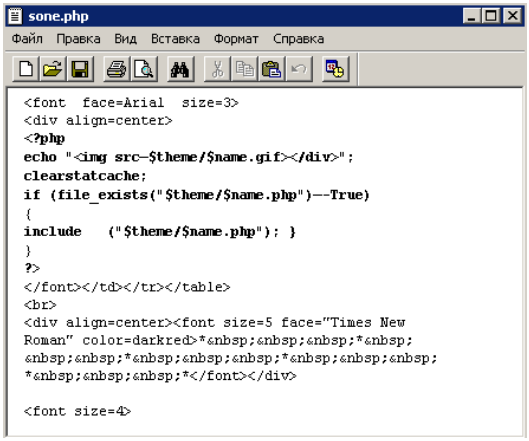


Рисунок 7.8 - Программа на PHP (выделена полужирным) в тексте web-страницы

Как и во всех языках программирования, в PHP есть возможность работать с переменными - некими объектами, имеющими имя и могущими принимать различные значения. Однако работа с переменными в PHP, пожалуй, самая легкая из всех возможных. Переменные не надо заранее объявлять (если не знаете, что это такое - то пока и не надо), разве что за исключением использования их в функциях. Для введения новой переменной достаточно просто присвоить ей какое-либо значение, а для помещения значения переменной в строку текста, имени файла или параметра команды, нужно просто написать ее имя в том месте, где должно быть ее значение. Чтобы PHP мог отличать переменные от строк или команд, имя переменной должно начинаться со знака доллара - ("$") и не должно содержать пробелов, знаков апострофа и некоторых других символов. При анализе программного кода интерпретатор PHP считает именем переменной все, что содержится между $ и ближайшим к нему символом, недопустимым в имени переменной.

В качестве имен переменных можно также использовать другие переменные - для этого другую переменную следует просто указать на месте имени первой: $$a.

Переменные в PHP могут быть четырех типов - число (целое и дробное), строка текста, массив и объект. Интерпретатор PHP автоматически определяет тип переменной на основании анализа ее содержимого. Подробнее о типах данных вы можете узнать в руководстве по PHP (например, в том, что доступно с сайта "PHP по-русски"), там же рассказано и об особенностях работы с массивами (и вообще объяснено, что это такое, если вы еще не знаете).

Для включения числовой или строковой переменной в строку текста достаточно

просто поместить переменную туда, где она должна стоять в этой строке. Например, в итоге выполнения кода $a="птица певчая"; $b="Дятел - $a"; переменной $b будет присвоено значение "Дятел - птица певчая".

Помните, что в PHP нельзя для сложения строковых переменных использовать символ "+" - он пригоден лишь для числовых выражений. Поэтому необходимо помещать переменные в строки или использовать команду конкатенации (точку): $c = $a. $b.

Есть еще два типа переменных - PDF-документ и PDF-инфо, но они применяются только при работе с файлами PDF (и при установленном модуле поддержки PDF).

Довольно часто используются сокращенные обозначения арифметических действий над переменными и действий по присваиванию им каких-либо значений. Например, команда $a+=3 означает, что переменную $a надо увеличить на 3, что и будет сделано, если она имеет числовой формат. Точно так же команда $a- = 3 уменьшает переменную $ a на 3, а команды $a*=2 и $a/=2 соответственно умножают и делят на два переменную $a. Команда $a.=" строка" эквивалентна команде $a="$a строка".

В PHP применяются также операции инкремента и декремента, т.е. изменения значения переменной на 1. Так, команды $a++ и $a-- соответственно увеличивают и уменьшают значение переменной $a на единицу. То же самое делают и команды ++$a, --$a, однако, в том случае, если подобная команда используется в выражении, они, в отличие от предыдущих, сначала изменяют значение переменной, а потом выдают его в выражение. Иными словами, если переменная $a равна 2, то после выполнения команды $b=$a++; ее значение достигнет 3, а $b будет установлена в 2. В то же время команда $b= + + $a; установит обе переменные в 3.

Массив - это совокупность под одним именем перенумерованных переменных. Имя каждой переменной в массиве состоит из имени этого массива и индекса переменной - нечто вроде номера переменной в массиве или ее имени в нем. Индекс переменной может быть цифровым или символьным - т. е. представлять собой либо номер переменной в массиве, либо ее имя в нем.

Например, вот массив с числовыми индексами (нумерация индексов начинается с нуля, а не единицы!):

$a[0]=100; $a[1]=101; $a[2]=102;

а вот с символьными:

$a['first']=100; $a['second']=101; $a['third']=102;

Массив с числовыми индексами называется еще "скалярным", а с символьными - "ассоциативным".

Зачем нужны массивы? А для того чтобы можно было к ним обращаться как к чему-то целому, тем самым получая возможность совершать автоматические действия со всеми элементами массива или с частью этих элементов, не указывая имени каждого их элемента.

Иными словами - скажем, в какие-то переменные записали имена клиентов и теперь желаем вывести их. Как это сделать? Естественно, только перебрав все эти переменные, для чего нам понадобятся имена этих переменных, которые придется жестко задать в программе. А если заранее неизвестно, сколько будет клиентов, как тогда быть? Если же имена клиентов поместить в массив, то все их можно перебрать специальной командой, добавить же новое имя тоже нетрудно.

В PHP добавлять элементы в массив можно как явно указывая индекс элемента (например, $a [100] = "Андрей";), так и просто упоминая, в какой массив этот элемент добавляется - $a [ ] ="Андрей" ;. В последнем случае добавляемый элемент становится последним в массиве.

Ниже перечислены некоторые основные команды PHP, которых вполне хватит для реализации несложных проектов. Для более полного ознакомления с командами PHP можно изучить Руководство по этому языку, доступное, например, с адреса http://php.spb.ru, или

другие публикации.

include "имя файла" - команда для включения содержимого одного файла в другой. Содержимое файла, имя которого указывается в команде, целиком и полностью вставляется на то место, где располагается эта команда, при этом все коды PHP, содержащиеся во вставляемом файле, исполняются так же, как если бы они были на месте этой команды. (Помните, что файл именно вставляется - т. е., например, пути к картинкам, которые должны присутствовать во вставляемом файле, следует указывать от местонахождения того файла, в котором находилась команда include.)

Если файл, включаемый в страницу при помощи команды include, отсутствует, то вместо него размещается уведомление об этом, а программа на PHP выполняется дальше. (При необходимости завершения обработки и выдачи web-страницы в случае отсутствия включаемого файла, вместо команды include следует использовать команду require.)

mail ("Кому", "Тема", "Текст сообщения", "Дополнительные заголовки") - отправка почтового сообщения. При выполнении данной команды на сервере в соответствии с указанными параметрами формируется электронное письмо и отправляется с помощью установленной на сервере почтовой программы. В качестве параметра "Кому" может выступать набор адресов, разделенных запятыми. "Дополнительные заголовки" могут быть любые (естественно, допустимые почтовыми протоколами!), разделяться они должны должны комбинацией символов /п, которая в PHP означает перевод строки. (Если среди "Дополнительных заголовков" не указано поле From, то оно заполняется по умолчанию почтовой программой web-сервера, например, именем "Unprivileged User".)

echo ("текст") - вывод на web-страницу какого-либо текста. Чтобы вывести на web-страницу значение какой-либо переменной, достаточно просто написать ее имя внутри выводимой строки: команда echo "это цифра $a" выведет в web-страницу текст "это цифра 1", если ранее переменной $a было присвоено значение, равное единице. В случае необходимости использовать в выводимой строке кавычки или иные специальные символы перед этими символами следует ставить символ " \".

if (условие) {...команды, которые должны выполняться, если условие верно...;} else {...команды, которые должны выполняться, если условие неверно...} -команда, позволяющая выполнить то или иное действие в зависимости от истинности верности или ложности того или иного условия. В фигурных скобках может располагаться несколько команд, разделенных точкой с запятой. В качестве условия может быть оператор сравнения "равно" -("==") (именно два-знака равенства!), "больше" -(">"), "меньше" - ("<") и их комбинации, скажем, "< = " - ("меньше или равно"). Можно использовать и несколько условий, взяв каждое из них, а также все вместе в скобки и разделяя знаками "&&" - ("и") или "| |" -("или").

Для того чтобы выполнять различные команды в зависимости от условия, которое может принимать три или больше значений, следует использовать оператор switch (описание смотрите ниже) - аналог оператора case в VBA и некоторых других языках.

for (начальное значение счетчика, условие продолжения цикла, изменение счетчика на каждом цикле) { . . . команды. . . ;} - цикл, т. е. повторение указанных в нем команд столько раз, сколько позволит условие изменения счетчика цикла (т. с. переменной, специально выделенной для подсчета числа выполнений команд цикла). К примеру цикл for ($i = 1; $i <= 10; $i + +) {echo $i;} выводит в web-страницу числа с 1 до 10, так как в нем изначально устанавливается значение счетчика в 1 - ($i = l), каждый цикл его значение увеличивается на 1 - ($i ++), а продолжаться он будет до тех пор, пока значение счетчика не превысит 10 (т. е. пока $i< = 10).

while (условие) { . . .команды. . . } - цикл с условием. Команды в фигурных скобках выполняются до тех пор, пока выполняется условие в заголовке цикла. Для того чтобы цикл прервался, нужно, чтобы условие выполняться перестало - поэтому внутри цикла необходимо предусмотреть возможность влиять на это условие. Скажем, цикл while ($i<=10)

{ . . .команды. . . ; $i++; } будет выполняться до тех пор, пока значение переменной $i не превысит 10 -если изначально оно было равно 1, то цикл выполнится 10 раз.

Цикл do {. . .команды. . . } while (условие) работает так же, однако команды, указанные в фигурных скобках, будут выполнены по меньшей мере один раз - даже если условие выполняться не будет.

Прервать выполнение любого цикла можно оператором break -дальнейшее выполнение программы пойдет с команды, следующей после закрывающей фигурной скобки. Оператор же continue прерывает текущую стадию выполнения цикла, т. е. после этого оператора дальнейшее выполнение программы начнется с очередной проверки условия заголовка цикла.

switch (выражение) {case значение: ... команды...; break; case другое значение: ... команды...; break;} - оператор выбора. При его работе содержимое, заключённое в фигурные скобки, просматривается сверху вниз. Как только будет найден оператор case со значением, совпадающим со значением выражения, PHP начнёт выполнять весь код, следующий за этим оператором case до последней фигурной скобки оператора switch или до первого оператора break, в зависимости от того, что появится раньше. (Обратите внимание, что если команду break не указать в конце кода, относящегося к одному варианту значения выражения в заголовке оператора switch, PHP будет выполнять код дальше - т. е. тот, который принадлежит уже следующему оператору case! Это - одно из отличий данного оператора от аналогичных в других языках программирования.)

В конце оператора switch можно указать оператор default. Код, стоящий после него, выполнится в том случае, если значение выражения в заголовке оператора не совпадет ни с одним из значений после операторов case.

foreach (переменная as массив) { . . .команды. .. ;} - поочередное считывание всех элементов массива. Foreach считывает в указанную в его параметрах переменную поочередно все элементы указанного в них же массива, выполняя каждый раз указанный в фигурных скобках код, в котором может использоваться указанная переменная. (Значения элементов массива этим оператором только считываются, их модификация при помощи команды foreach невозможна.) Оператор f oreach может быть использован только в PHP версии 4.0 и выше.

Программа на PHP может прерываться кодом web-страницы - для этого достаточно вставить закрывающий тэг до этого кода и открывающий - после. Все, что находится между ними, будет выдаваться в браузер без какой-либо обработки, рассматриваясь как выводимое с помощью команды echo. Иными словами, код

<?php if ($a==l) { ?> Переменная а равна 1 <?php> }?

> эквивалентен коду

<?php if ($a==l) {echo "<p> Переменная а равна 1</p>";}?>

однако, первый вариант меньше нагружает процессор компьютера, на котором расположен интерпретатор PHP.

Из сказанного также следует, что все программы на PHP, расположенные на одной web-странице, представляют собой одну большую программу, несмотря на то, что они разделяются блоками обычного текста страницы. Именно поэтому переменная, объявленная в расположенном в начале страницы коде, сохраняет свое значение не только до ее конца, но и во всех присоединяемых с помощью команды include файлах. Пример - на рис. 9.

В PHP можно создавать функции - подпрограммы, которые можно вызывать по своим именам, при необходимости передавая им определенную информацию. Необходимы они в том случае, когда один и тот же код нужно выполнять несколько раз для разных данных, особенно если требуемое количество выполнений заранее неизвестно. Создать функцию на PHP можно, вставив в программу инструкцию function имя (переменные, в которые записываются передаваемые параметры, и их тип) {...команды функции . . . }, а вызвать -

простым указанием имени этой функции и параметров.

Посмотрите внимательно на код и результат его отображения. Обратите внимание, что переменная a, которой присвоено значение еще в первом программном блоке левой страницы, сохранила его не только в других ее блоках, но и в программе, расположенной во включаемой с помощью команды include странице.

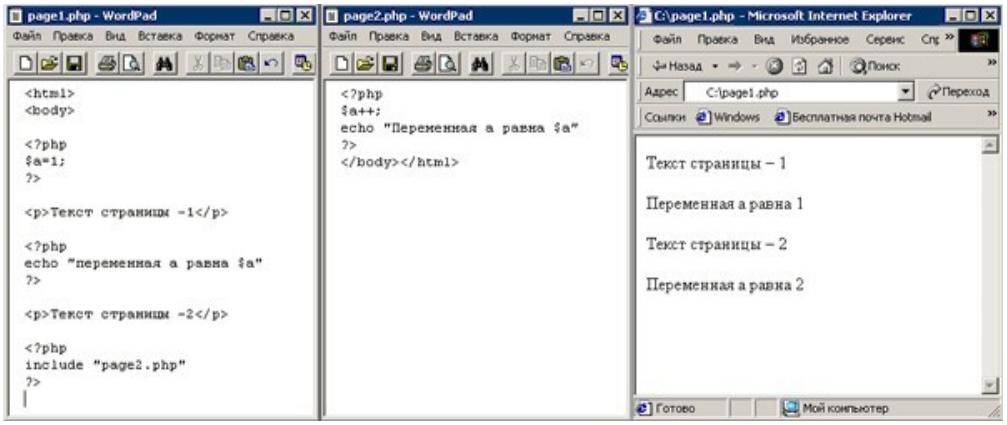


Рисунок 7.9. - Пример PHP-кода

Помните, что переменные, созданные в функции, по умолчанию имеют установленное значение только внутри функции. Кроме того, также по умолчанию переменные, объявленные вне функции, в ней самой никакого значения не имеют, а если надо, чтобы имели, то вначале функции их следует "подключить" командой global <переменная>; - и лишь после этого они станут доступными в функции. Подробнее о функциях и о переменных в них читайте в руководстве по PHP, например, с того же сайта http://php.spb.ru.

Обычно web-сервер настраивается так, что на предмет наличия программ на PHP просматриваются файлы, имеющие расширение php, .php3, .phtml, остальные же файлы передаются в браузер пользователя без поиска в них команд PHP. Делается так для более быстрой работы сервера, а также для обеспечения возможности установки на сервере разных интерпретаторов (например, SSI - Server Side Includes, технологии, в какой-то мере, предшествовавшей PHP), так как тогда каждому из интерпретаторов назначаются свои расширения для обработки соответствующих файлов.

Так как PHP-код полностью исполняется на web-сервере, то в страницах, выдаваемых браузеру, он будет отсутствовать, и если кто заинтересуется вашим опытом программирования, то вам придется отправлять ему этот код по почте, так как при просмотре сайта каким-нибудь образом узнать исходный PHP-код его страниц нельзя.

Значения переменных можно передавать между различными страницами сайта - с помощью использования форм. Формой называется конструкция, состоящая из поименованных элементов особых типов, заключенных между HTML-тэгами <form...> n</form>. В качестве элементов формы могут выступать поля ввода текста, кнопки, выпадающие меню, переключатели, квадратики для отметки галочкой - checkbox'bi, а также картинки формата jpg или gif. Каждый элемент формы может иметь свое имя.

Наиболее важным свойством формы является то, что в ее заголовке в открывающем тэге <form...> можно указать адрес какого-либо файла. В этом случае при загрузке этого файла в программный код, если он будет там присутствовать, передадутся значения всех переменных, установленных в этой форме, в частности, значения всех элементов формы, как если бы эти значения были установлены в программе, расположенной в самом загружаемом файле. Таким образом, можно передавать значения переменных между различными web-страницами, используя их в программном коде.

Во всех версиях PHP имена передаваемых переменных соответствуют тем именам, которые были даны элементам формы в их тэгах, а значения - соответственно значениям этих элементов (если в конфигурационном файле PHP - php.ini - параметр register_globals установлен в on.): для поля ввода текста - введенному тексту, для переключателя или checkbox'a - True при отмеченном и False при неотмеченном, для рисунка - координаты указателя мыши относительно верхнего левого угла изображения, для выпадающего меню (элемент <select пате="имя"><option value="textl"> text </option>...</select>) - значение параметра value выбранного пункта option.

Кроме того, переменные, передаваемые через форму, помещаются в ассоциативные массивы $HTTP_POST_VARS и $HTTP_GET_VARS (если в конфигурагцюнном файле PHP - php.ini - параметр track_vars установлен в on) с именами элементов, соответствующими именам переменных (т. е. содержимое поля ввода текста <input type=text name=qwerty size=3 0> окажется в элементе $HTTP_POST_VARS['qwerty']). SHTTPPOSTVARS содержит переменные, переданные с помощью метода POST (метод указывается в заголовке формы), а $HTTP_GET_VARS - метода GET. Различие между методами состоит в том, что при передаче данных методом GET эти данные отображаются в адресной строке браузера, а при использовании метода POST - нет.

Начиная с PHP версии 4.1, передаваемые через форму переменные помещаются еще и в массивы SPOST и SGET. Отличие этих массивов о предыдущих состоит в том, что их переменные доступны еще и во всех функциях, расположенных в программе PHP, т. е. они являются глобальными.

Этапы проектирования и создания базы данных

Перед созданием базы данных необходимо располагать описанием выбранной предметной области, которое должно охватывать реальные объекты и процессы, определить все необходимые источники информации для удовлетворения предполагаемых запросов пользователя и определить потребности в обработке данных.

На основе такого описания на этапе проектирования базы данных определяется состав и структура данных предметной области, которые должны находиться в БД и обеспечивать выполнение необходимых запросов и задач пользователя. Структура данных предметной области может отображаться информационно-логической моделью. На основе этой модели легко создается реляционная база данных.

Пример создания БД в MySQL.

Для успешной реализации веб-проекта очень важна организация сохранения и извлечения данных. Подход к сфере хранения данных базируется на применении БД, т.е. ориентирован на работу с таблицами и запросами вместо папок и файлов. PHP поддерживает широкий спектр различных серверов БД (СУБД), т.е. программ, которые занимаются обслуживанием и сопровождением БД. Особое место среди них занимает Oracle. Но мы в наших разработках будем пользоваться не менее популярным сервером Mysql. Этот сервер не такой мощный, но удобен в управлении, не велик по объему и занимаемым ресурсам, может быть бесплатно загружен из Интернета и использоваться без каких бы то ни было нарушений лицензионного законодательства.

Для наглядности создания БД и WEB-приложения приведем пример создания простой БД содержащую одну таблицу в которую будут записываться данные о студентах факультета ИСТ.

Для начала работы создадим простую БД с одной таблицей STUDENTS. Описание учебной базы данных.

1. Запустите утилиту MySQL Query Browser

2. Необходимо подключится к серверу БД (serverbd) задав в качестве username (studN, где N - номер заданный преподавателем и Shema - studN), где Server host – 192.168.5.6 или serverbd.fist.local (рисунок 6.10).

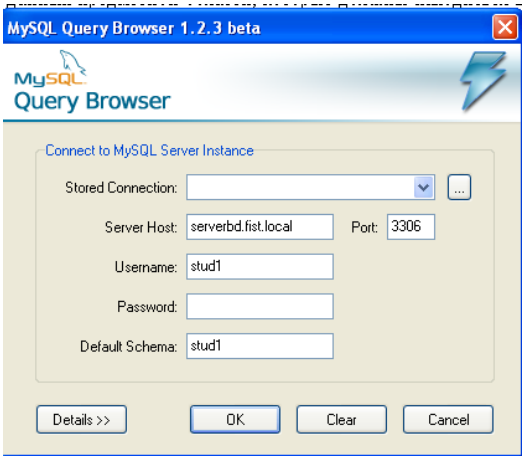


Рисунок 7.10 - Диалоговое окно MySQL Query Browser

Необходимо создать в БД с помощью SQL запросов таблицу, отражающую данные о студентах факультета ИСТ. (рисунок 7.11).

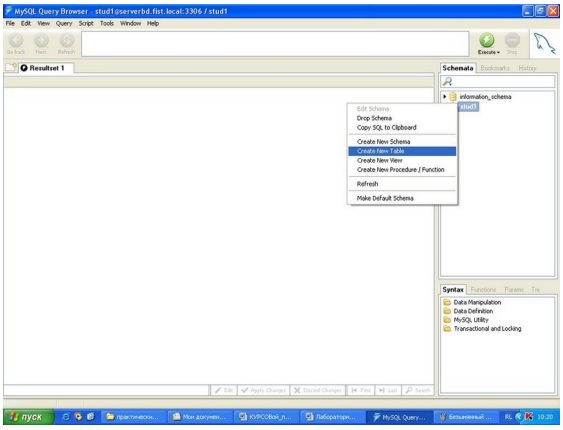

Рисунок 7.11 - Создание новой таблицы в MySQL

Описание учебной базы данных.

В таблице СТУДЕНТЫ (STUDENTS) содержится пять полей с информацией о студентах-

kod - номер студенческого билета;
fam - фамилия студента;
name - имя студента;
otch - отчество студента:
grup – номер группы.

Таблица создается командой CREATE TABLE. Эта команда создает, пустую таблицу, т.е. не содержащую записей. Очевидно, что значения в нее можно ввести, например, с помощью команды INSERT. Главное в команде CREATE TABLE - это определение имени таблицы и описания набора имен полей, которые указываются в соответствующем порядке. Кроме того, этой командой также оговариваются типы данных и размеры полей таблицы.

Приведем пример команды, которая создаст структуру таблицы STUDENTS:

```
CREATE TABLE STUD (kod INTEGER,
                   fam CHAR  (15),
                   name CHAR (10),
                   otch  CHAR  (15) ,
```

grup (6)) ;

Порядок расположения полей в таблице определяется тем, в какой последовательности они указаны в команде создания таблицы. После того, как таблица была создана, ее можно изменить

Модификация созданных таблиц: Команда ALTER TABLE является широко доступным средством для того, чтобы изменить определение существующей таблицы. Чаще всего с ее помощью добавляют поля к таблице, хотя она может удалять или изменять их размеры. Типичный синтаксис этой команды для добавления столбца к таблице, такой:

ALTER TABLE <TABLE NAME> ADD <COLUMN NAME> <DATA TYPE> <SIZE>;

Пример: для добавления к таблице STUDENTS двух полей для хранения информации о курсе и специальности студента, можно воспользоваться следующей командой:

ALTER TABLE STUDENTS ADD COURS INTEGER, SPEC CHAR (10);

С использованием этой команды имеется возможность удалять или изменять поля, причем наиболее часто изменением бывает просто увеличение его размера. Обязательно нужно убедиться, что любые вносимые изменения не противоречат существующей структурой.

Удаление таблиц: DROP TABLE <TABLE NAME>;

После выполнения этой команды, имя таблицы больше не распознается, и нет таких действий, которые могли быть выполнены с этим объектом. Перед удалением стоит убедиться в том, что эта таблица не ссылается на другую таблицу и что она не используется в каком-либо представлении. Например, для удаления таблицы STUDENTS, в которой все записи предварительно удалены, просто вводится следующее:

DROP TABLE STUDENTS;

При работе с SQL исключительно важно не только уметь выбирать данные, но и пользоваться средствами, которые управляют значениями в таблице. Значения могут быть помещены и удалены из полей тремя командами:

INSERT - вставить

UPDATE -

модифицировать DELETE

– удалить.

4. Сохранить таблицу.

5. Открыть пустой текстовый документ (блокнот) и набрать программу на PHP-HTML языках, предоставленную в следующем листинге программы, где:

«Имя файла.php» – это имя вашего текстового документа, который надо сохранить под расширением php,

«Studn» – username,

«Пароль» – устанавливаете свой, если пароль не нужен, достаточно набрать просто двойные кавычки без пробела - «""»

«Имя таблицы» – набирается имя таблицы, созданной в MySQL.

Примечание: при наборе имен таблиц, полей, тегов будьте внимательны в написании, т.к. учитывается регистр и язык набора.

Листинг программы.

```
<?PHP
IF($_POST[‘STEP’]!=1){
?>
<FORM ACTION=“имя файла.php“METHOD=”POST”NAME=”FORM1”>
<TABLE WIDTH=”80%”BORDER=”0”>
<TR>
        <TD>КОД СТУДЕНТА </TD>
        <TD><INPUT TYPE=”TEXT”NAME=”kod”></TD>
<TD>ФАМИЛИЯ</TD>
```

```
<TD>><INPUT TYPE=”TEXT”NAME=”fam”></TD>
<TD>ИМЯ</TD>
<TD>><INPUT TYPE=”TEXT”NAME=”name”></TD>
</TR>
<TR>
    <TD>ОТЧЕСТВО </TD>
    <TD><INPUT TYPE=”TEXT”NAME=”otch”></TD>
<TD>ГРУППА</TD>
<TD>><INPUT TYPE=”TEXT”NAME=”grup”></TD>
</TR>
</TABLE>
<CENTER>< INPUT NAME=”OK” TYPE=”SUBMIT” VALUE=”СОХРАНИТЬ”></CENTER>
< INPUT NAME=”STEP” TYPE=”HIDDEN” VALUE=”1”>
</FORM>
<?PHP
}
ELSE{
$kod=$_POST[‘kod’];
$fam=$_POST[‘fam’];
$name=$_POST[‘name’];
$otch=$_POST[‘otch’];
$grup=$_POST[‘grup’];
if($db=mysql_connect(“server bd.fist.local”,”studn”,”пароль”)){
$sel=mysql_select_db(“studn”,$db); If(!
$sel){echo(“БАЗА НЕ ВЫБРАНА”);}
$q=mysql_query("insert into имя таблицы values($kod,'$fam','$name','$otch','$grup');", $db)
; if (!$q){echo("ОШИБКА ЗАПРОСА");
}
}
else{
echo("ОШИБКА СОЕДИНЕНИЯ");
}
}
?>
```

6. Сохраните этот документ в папке сервера (ftp://192.168.7.25), и запустите для тестирования (http://studn.fist.local/имя файла.php).

В дальнейшем изложении будем использовать в качестве примера небольшую БД, отражающую учет успеваемости студентов ВУЗа, содержащую уже четыре таблицы.

В таблице СТУДЕНТЫ (STUDENTS) содержится пять полей с информацией о студентах:

NUM - номер студенческого билета;
SFAM - фамилия студента;
SIMA. - имя студента;
SOTCH - отчество студента:
STIP - размер получаемой студентом стипендии.

В таблице ПРЕДМЕТЫ (PREDMET), состоящей из пяти полей, содержится информация об учебных предметах. Назначение полей таблицы следующее:

PNUM - номер (код) учебного предмета;
PNAME - наименование учебного предмета;
TNUM - номер (код) преподавателя;
HOURS - продолжительность учебной дисциплины в часах:
COURS - курс, на котором ведется данный учебный предмет

В таблице ПРЕПОДАВАТЕЛИ (TEACHERS), состоящей из пяти полей, содержится информация о преподавателях. Поля таблицы следующие:

TNUM - код преподавателя;
TFAM - фамилия преподавателя;
TIMA - имя преподавателя;
TOTCH - отчество преподавателя;
TDATE - дата принятия преподавателя на работу.

В таблице УСПЕВАЕМОСТЬ (USP), состоящей из пяти полей, хранится информация об успеваемости студентов по учебным дисциплинам. Поля таблицы следующие:

UNUM - код факта сдачи учебной дисциплины;
OCENKA - оценка, полученная студентом по учебному предмету;
UDATE - дата сдачи;
SNUM - номер студенческого билета; PNUM - код учебного предмета.

Таблицы создаются командой CREATE TABLE. Эта команда создает, пустую таблицу, т.е. не содержащую записей. Очевидно, что значения в нее можно ввести, например, с помощью команды INSERT. Главное в команде CREATE TABLE - это определение имени таблицы и описания набора имен полей, которые указываются в соответствующем порядке. Кроме того, этой командой также оговариваются типы данных и размеры полей таблицы.

Приведем пример команды, которая создаст структуру таблицы STUDENTS:

```
CREATE TABLE STUD (SNUM INTEGER,
                   SFAM CHAR (20),
                   SIMA CHAR (10),
                   SOTCH CHAR (15) ,
                   STIP DECIMAL);
```

Порядок расположения полей в таблице определяется тем, в какой последовательности они указаны в команде создания таблицы. После того, как таблица была создана, ее можно изменить

Модификация созданных таблиц: команда ALTER TABLE является широко доступным средством для того, чтобы изменить определение существующей таблицы. Чаще всего с ее помощью добавляют поля к таблице, хотя она может удалять или изменять их размеры. Типичный синтаксис этой команды для добавления столбца к таблице, такой:

ALTER TABLE <TABLE NAME> ADD <COLUMN NAME> <DATA TYPE> <SIZE>;

Пример: для добавления к таблице STUDENTS двух полей для хранения информации о курсе и специальности студента, можно воспользоваться следующей командой:

```
ALTER TABLE STUDENTS ADD COURS INTEGER, SPEC CHAR (10);
```

С использованием этой команды имеется возможность удалять или изменять поля, причем наиболее часто изменением бывает просто увеличение его размера. Обязательно нужно убедиться, что любые вносимые изменения не противоречат существующей структурой.

Удаление таблиц: DROP TABLE <TABLE NAME>;

После выполнения этой команды, имя таблицы больше не распознается, и нет таких действий, которые могли быть выполнены с этим объектом. Перед удалением стоит убедиться в том, что эта таблица не ссылается на другую таблицу и что она не используется в

каком-либо представлении. Например, для удаления таблицы STUDENTS, в которой все записи предварительно удалены, просто вводится следующее:

DROP TABLE STUDENTS;

4. Создание четырех таблиц базы данных:

Приведем пример команды, которая создаст структуру таблицы STUDENTS:

```
CREATE TABLE STUDENTS
(SNUM INTEGER,
SFAM CHAR (20),
SIMA CHAR (10),
SOTH CHAR (15),
STIP DECIMAL);
```

Порядок расположения полей в таблице определяется тем, в какой последовательности они указаны в команде создания таблицы. После того, как таблица была создана, ее можно изменить. Команда ALTER TABLE является широко доступным средством для того, чтобы изменить определение существующей таблицы. Чаще всего с ее помощью добавляют поля к таблице, хотя она может или изменять их размеры. Типичный синтаксис этой команды для добавления столбца к таблице, такой:

```
ALTER TABLE <TABLE NAME>
ADD <COLUMN NAME> <DATA TIPE> <SIZE>;
```

Стоит помнить, что поле будет добавлено с NULLзначениями для всех записей таблицы. Кроме того, новое поле станет последним по порядку в таблице. Допускается добавление сразу нескольких новых полей, отделив их запятыми в одной команде.

```
ALTER TABLE STUDENTS
ADD COURS INTEGER,
SPEC CHAR (10);
```

С использованиебм этой команды имеется возможность удалять или изменять поля, причем наиболее часто изменением бывает просто увеличение его размера. Обязательно нужно убедиться, что любые вносимые изменения не противоречат существующим. Ввод данных в таблицы (табл. 7.2-7.5).

Таблица 7.2 – Студенты

| STUDENTS | | | | |
|---|---|---|---|---|
| SNUM | SFAM | SI MA | SOTCH | STIP |
| 3412 | Поляков | Анатолий | Алексеевич | 25.50 |
| 3413 | Старова | Любовь | Михайловна | 17.00 |
| 3414 | Гриценко | Владимир | Николаевич | 0.00 |
| 3415 | Котенко | Анатолий | Николаевич | 0.0 |

Таблица 7.3 – Предметы

| PREDMETY | | | | |
|---|---|---|---|---|
| 2001 | Физика | 4001 | 34 | 1 |
| 2002 | Химия | 4002 | 68 | 1 |
| 2003 | Математика | 4003 | 68 | 1 |
| 2004 | Философия | 4005 | 17 | 2 |
| 2005 | Экономика | 4004 | 17 | 3 |

Таблица 7.4 Преподаватели

| TEACHERS | | | | |
|---|---|---|---|---|
| TNUM | TFAM | TIMA | SOTCH | STIP |

| | | | | |
|---|---|---|---|---|
| 4001 | Викулина | Валентина | Ивановна | 01/04/1984 |
| 4002 | Костыркин | Олег | Владимирович | 01/09/1997 |
| 4003 | Казанко | Виталий | Владимирович | 01/09/1988 |
| 4004 | Поздняков а | Любовь | Алексеевна | 01/09/1988 |
| 4005 | Загарийчук | Игорь | Дмитриевич | 10/05/1989 |

Таблица 7.5 - Успеваемость

| USP | | | | |
|---|---|---|---|---|
| UNUM | OCENKA | UDATE | SNUM | PNUM |
| 1001 | 5 | 10/06/1999 | 3412 | 2001 |
| 1002 | 4 | 10/06/1999 | 3413 | 2003 |
| 1003 | 3 | 11/06/1999 | 3414 | 2005 |
| 1004 | 4 | 12/06/1999 | 3412 | 2003 |
| 1005 | 5 | 12/06/1999 | 3416 | 2004 |

При работе с SQL исключительно важно не только уметь выбирать данные, но и пользоваться средствами, которые управляют значениями в таблице. Значения могут быть помещены и удалены из полей тремя командами языка DML (Язык Манипулирования Данными), а именно:

INSERT – вставить;

UPDATE - модифицировать;

DELETE - удалить.

Добавление информации в базу данных

Все записи в SQL вводятся с использованием команды модификации INSERT. В самой простой форме эта команда имеет следующий синтаксис:

INSERT INTO <table name>

VALUES «value>, <value> . . .);

Так, например, для добавления записи в таблицу преподавателей TEACHERS, можно воспользоваться следующим выражением:

INSERT INTO TEACHERS

VALUES (4006, 'Федченко', 'Светлана', 'Геннадиевна', 01/09/1999) ;

Команда INSERT не производит никакого вывода, но желательно, чтобы СУБД давала некоторое подтверждение того, что данные были успешно внесены. Кроме того, следует помнить, что имя таблицы, в которую производится вставка, должно быть предварительно определено, а каждое значение в списке вставляемых данных должно совпадать с типом данных столбца, в который оно вставляется. Значения в этом списке вводятся в таблицу в том порядке, в котором они записаны в команде, поэтому первое значение автоматически попадает в первый столбец, второе - во второй столбец и т. д.

Если требуется ввести в таблицу NULL значение, то оно вводится точно так же, как и обычное. Например, следующая команда, вставляющая запись с неизвестным значением кода преподавателя, вполне допустима:

INSERT INTO TEACHERS

VALUES (NULL, 'Федченко',‘Светлана','Геннадиевна', 01/09/1999);

Так как значение NULL - специальное служебное слово, то заключать его в одиночные кавычки не требуется.

Также допускается указывать столбцы, куда необходимо осуществить вставку значения, что позволяет делать это в любом порядке. Например, команда:

INSERT INTO TEACHERS (TDATE, TFAM, TIMA) VALUES (01/09/1999, 'Федченко', 'Светлана');

позволяет вставить значения в поля таблицы в порядке TDATE, TFAM, TIMA, причем столбцы TNUM и TOTCH отсутствуют. Это означает, что для этих полей автоматически

устанавливается значение по умолчанию. Значение по умолчанию может быть введено заранее или. в противном случае, это будет NULL, значение. Если ограничение запрещает использование значения NULL в данном поле, то обязательно надо позаботиться об обеспечении столбца содержательным значением для любой команды INSERT.

Можно использовать команду INSERT для того, чтобы получать или выбирать значения из одной таблицы и помещать их в другую вместе с запросом. Для этого предложение VALUES заменяется на соответствующий запрос:

```
INSERT INTO EiXCELLENT
SELECT *
FROM USP
WHERE OCENKA =5;
SEIECT SFAM, SIMA, SOTCH, 'y.e.', STIP*2
FROM STUDENTS;
```

Вывод этого запроса будет следующий:

| SFAM | SIMA | SOTCH | | |
|---|---|---|---|---|
| Поляков | Анатолий | Алексеевич | у.е. | 51.00 |
| Стасова | Любовь | Михайловна | у.е. | 34.00 |
| Гриценко | Владимир | Николаевич | у.е. | 0.00 |
| Котенко | Анатолий | Николаевич | у-е. | 0.00 |
| Нагорный | Евгений | Васильевич | у-е. | 51.00 |

Стоит иметь в виду, что все символы, в т. ч. пробелы, в строке текста также вставляются в вывод, поэтому предложенный способ можно использовать для маркирования вывода вместе со вставляемыми комментариями, однако необходимо помнить, что этот же самый комментарий будет напечатан в каждой строке вывода, а не просто один раз для всей таблицы. Предположим, что необходим отчет о количестве студентов, получающих ту или иную стипендию, тогда можно предложить следующий запрос:

```
SELECT COUNT (DISTINCT SNUM),
'студ. получают стипендию',
STIP, 'y. e.'
FROM STUDENTS
GROUP BY STIP;
```

Построение многотабличных запросов:

Пример1: SELECT TEACHERS.TFAM, PREDMET.PNAME FROM TEACHERS, PREDMET WHERE TEACHERS.TFAM <PREDMET.PNAME AND TEACHERS.TFAM BETWEEN ‘K' AND 'C’;

Пример 2: Допускается также создавать запросы, объединяющие более двух таблиц. Например, необходимо вывести список оценок, выставленных тем или иным преподавателем. Тогда запрос будет объединять сразу три таблицы:

```
SELECT TEACHERS.TFAM, USP.0CENKA
FROM TEACHERS, PREDMET, USP
WHERE TEACHERS.TNUM = PREDMET.THUM AMD PREDMET.PNUM = USP.PNUM;
```

В результате будет получено:

| TFAM | OCENKA |
|---|---|

| | |
|---|---|
| Никулина | 5 |
| Казанко | 4 |
| Казанко | 4 |
| Позднякова | 3 |
| Загарийчук | 5 |

Форма представления отчета по практике

Основной формой аттестации учебной практики является составление и защита индивидуального отчета студентов. В итоге по результатам прохождения практики выставляется дифференцированный зачет. Программа практики студента носит индивидуальный характер и разрабатывается

совместно с его руководителем.

Оформление, структура и содержание отчета по практике. Отчет - итоговый документ, на основании которого и после его защиты студент получает зачет по практике.

Объем отчета вместе с приложениями – 15-25 страниц формата А4. Он должен быть изложен грамотно, аккуратно оформлен, напечатан с помощью компьютера.

Структурно отчет содержит следующие элементы: титульный лист, введение, основная часть (перечень разделов), заключение, список использованных источников, приложения.

Защита студентами отчетов по практике осуществляется в комиссии в установленные кафедрой и институтом сроки. По итогам аттестации (защиты отчета) выставляется оценка (отлично, хорошо, удовлетворительно, неудовлетворительно). Студенты, не выполнившие программу практик без уважительной причины или получившие отрицательную оценку, могут быть отчислены из университета как имеющие академическую задолженность в порядке, предусмотренном Уставом вуза.

Критерии выставления оценок

Критериями положительного решения кафедры об успешном прохождении практики студентом являются: степень выполнения программы практики, соответствие проведенных мероприятий целям и задачам планируемой практической работы, положительный отзыв руководителя практики, сроки и качество отчета студента о проделанной работе на практике, дисциплина и исполнительность студента за время практики.

Оценка по практике выставляется по 4-х бальной системе в соответствии со следующими требованиями.

Оценка «отлично» выставляется за выполнение на высоком уровне всех требований программы практики; ~ современное представление итоговой документации и успешное собеседование с руководителем практики; ~ выраженное стремление к приобретению и совершенствованию профессиональных знаний, умений и навыков; ~ активное участие в выполнении программы практики; ~ умение правильно планировать и эффективно осуществлять установленные программой практики виды и формы деятельности; ~ самостоятельность, творческий подход в процессе практики.

Оценка «хорошо» выставляется в случае, когда студент выполнил все требования программы практики, но при этом не проявил стремления к совершенствованию знаний, умений и навыков; не отличался инициативностью, высокой активностью, творческим подходом и самостоятельностью в выполнении заданий.

Оценка «удовлетворительно» выставляется за: наличие поверхностных знаний, неустойчивых умений в области учебной деятельности; отсутствие активности работе; ошибки в планировании, организации и осуществлении установленных программой форм и видов деятельности; слабое владение приемами учебной деятельности;

Оценка «неудовлетворительно» ставится в случае невыполнения требований учебной практики.

Учебно-методическое и информационное обеспечение учебной практики


1. Корпоративные информационные системы управлениия : учебник / [Н.М. Абдикеев, Н.Б. Завьялова, А.Д. Киселев и др.] ; под ред. Н.М. Абдикеева, О.В. Китовой. - М. : ИНФРА-М, 2011. - 464 с. : ил. - (Учебники для МВА). - Библиогр. в конце глав. - ISBN 978-5-16-004373-9
2. Росс, К. Компьютерные сети / К. Росс, Дж. Куроуз. - М.: СПб: Питер; Издание 2-е, 2017. - 768 с.
3. Варфоломеева, А.О. Информационные системы предприятия: Учебное пособие / А.О. Варфоломеева, А.В. Коряковский, В.П. Романов. - М.: НИЦ ИНФРА-М, 2013. - 283 с.
4. Олейник, П.П. Корпоротивные информационные системы.Учебник для вузов. / П.П. Олейник, С.П. Олейник. - СПб.: Питер, 2012. - 176 с..
5. Рыжко, А.Л. Информационные системы управления учебной компанией: Учебник для академического бакалавриата / А.Л. Рыжко, А.И. Рыбников, Н.А. Рыжко. - Люберцы: Юрайт, 2019. - 354 с..
6. Шелухин, О. И. Моделирование информационных систем / О.И. Шелухин, А.М. Тенякшев, А.В. Осин. - М.: Радиотехника, 2019. - 368 с
7. Барский А.Б. Параллельные информационные технологии [Электронный ресурс] : учебное пособие / А.Б. Барский. — Электрон. текстовые данные. — Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 503 c. — 978-5-4487-0087-3. — Режим доступа: http://www.iprbookshop.ru/67379.html
8. Журавлева Т.Ю. Информационные технологии [Электронный ресурс] : учебное пособие / Т.Ю. Журавлева. — Электрон. текстовые данные. — Саратов: Вузовское образование, 2018. — 72 c. — 978-5-4487-0218-1. — Режим доступа: http://www.iprbookshop.ru/74552.html
9. Основы информационных технологий [Электронный ресурс] / С.В. Назаров [и др.]. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 530 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/52159.html
10. Основы информационных технологий [Электронный ресурс] : учебное пособие / Г.И. Киреева [и др.]. — Электрон. текстовые данные. — Саратов: Профобразование, 2017. — 272 c. — 978-5-4488-0108-2. — Режим доступа: http://www.iprbookshop.ru/63942.html
11. Антонов, В.Ф. Методы и средства проектирования информационных систем : учебное пособие / В.Ф. Антонов, А.А. Москвитин ; Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Северо-Кавказский федеральный университет». - Ставрополь : СКФУ, 2016. - 342 с. : ил. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php? page=book&id=458663
12. Методы и средства проектирования информационных систем и технологий : учебное пособие / Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Северо-Кавказский федеральный университет» ; авт.-сост. Е.В. Крахоткина. – Ставрополь : СКФУ, 2015. – 152 с. : ил. – Библиогр. В кн. ; То же [Электронный ресурс]. – URL: http://biblioclub.ru/index.php? page=book&id=458082


Интернет-ресурсы:

1 http://www.biblioclub.ru Университетская библиотека online
2 http://www.iprbookshop.ru ЭБС «IPRbooks»
3 http://catalog.ncstu.ru/ Электронная библиотека СКФУ.

Программное обеспечение:

- 1С: Предприятие 8. Комплект для обучения в высших и средних учебных заведениях (рег. номер 9334708), AutoCAD 2015 (бесплатный для вузов), Embarcadero rad studio - Г/к 445/01 от 30 июля 2010 г., IBM Rational Rose modeler (бесплатно по программе IBM Academic Initiative), Mathcad Education - University Edition (50 pack) -договор № 24-эа/15 от 19 августа 2015г., Microsoft Office - №61541869, Cisco Packet Tracer - договор № 23-с от 27 июня 2012 г., Microsoft Windows 7 Профессиональная - №61541869, Visual Studio IDE – AzureDev ID: a6c2b0d7-162e-479f-8a58-384701f33665, Microsoft Visual Basic – AzureDev ID: a6c2b0d7-162e-479f-8a58-384701f33665, Microsoft SQL Server – AzureDev ID: a6c2b0d7-162e-479f-8a58-384701f33665, PascalABC.NET (бесплатный), Oracle VM VirtualBox (бесплатный)