МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационных систем и технологий

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«15» 12 2017 г.

УПРАВЛЕНИЕ ДАННЫМИ

Методические указания по выполнению практических заданий для студентов направлений 09.03.02 Информационные системы и технологии и 09.03.03 Прикладная информатика

Курск 2017

УДК 004.6

Составитель Д.О. Бобынцев

Рецензент: к.т.н., доцент Ватутин Э.И.

Управление данными: методические указания к выполнению практических заданий / Юго-Зап. гос. ун-т; сост.: Д.О. Бобынцев. Курск, 2017. 37 с.: ил. 7. Библиогр.: с. 37.

Содержит методические указания к выполнению практических и лабораторных работ по дисциплине «Управление данными». Указывается порядок выполнения работ, контрольные вопросы. Предназначен для студентов направлений подготовки «Информационные системы и технологии» и «Прикладная информатика».

Текст печатается в авторской редакции

Подписано в печать 15.12.2017. Формат 60х84 1/16.
Усл.печ. л. 2,15. Уч.-изд. л. 1,94. Тираж 100 экз. Заказ. Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Содержание

Анализ предметной области. Инфологическое проектирование

Цель работы: освоить первые этапы проектирования базы данных – анализ предметной области и построение инфологической модели.

Теоретический материал

Процесс конструирования базы данных (ее проектирования и реализации) состоит из последовательности преобразований модели данных одного уровня в модель данных другого уровня.

Последовательности преобразований модели данных:

- систематизация объектов реального мира;
- создание информационных структур, описывающих систему объектов реального мира;
- создание структуры данных, которая используется для представления информационных структур в базе данных и прикладных программах;
- представление структуры памяти, используемой для хранения структур данных.

Процесс проектирования базы данных включает три этапа. Первый этап **-** анализ предметной области или *этап концептуального проектирования.* На этапе концептуального проектирования осуществляется сбор, анализ и редактирование требований к данным**.**

Первым этапом проектирования БД любого типа является анализ предметной области, который заканчивается построением информационной структуры (концептуальной схемы). На данном этапе анализируются запросы пользователей, выбираются информационные объекты и их характеристики, которые предопределяют содержание проектируемой БД. На основе проведенного анализа структурируется предметная область. Анализ предметной области не зависит от программной и технической сред, в которых будет реализовываться БД.

Анализ предметной области целесообразно разбить на три фазы:

1) анализ требований и информационных потребностей;

2) выявление информационных объектов и связей между ними;

3) построение модели предметной области и проектирование схемы БД.

Рассмотрим каждую фазу данного этапа проектирования подробно.

На этапе анализа концептуальных требований и информационных потребностей необходимо выполнить**:**

1) анализ требований пользователей к базе данных (концептуальных требований);

2) выявление имеющихся задач по обработке информации, которая должна быть представлена в базе данных (анализ приложений),

3) выявление перспективных задач (перспективных приложений);

4) документирование результатов анализа

Требования пользователей к разрабатываемой БД представляют собой список запросов с указанием их интенсивности и объемов данных. Эти сведения разработчики БД получают в диалоге с ее будущими пользователями. Здесь же выясняются требования к вводу, обновлению и корректировке информации Требования пользователей уточняются и дополняются при анализе имеющихся и перспективных задач.

Рассмотрим примерный состав вопросника, требований к базе данных при анализе различных предметных областей.

Пример 1. Пусть предлагается разработать систему вопросов к БД «Сессия студентов колледжа»:

1. Сколько студентов учится в колледже?

2. Сколько отделений в данном колледже?

3. Как распределены студенты по отделениям отделений и курсам?

4. Сколько дисциплин читается на каждом курсе по каждой специальности?

5. Как часто обновляется информация в базе данных?

6. Сколько преподавателей?

7. Сколько иногородних студентов живет в общежитии, на частных квартирах?

8. Какая преемственность существует между читаемыми курсами?

9. Сколько лекционных аудиторий и аудиторий для проведения практических занятий, лабораторий?

10. Как информация, представленная в п.п. 1-9, используется в настоящее время (расписание занятий, экзаменов, зачетов и т.д.) и как ее собираются использовать?

11. Сколько раз в день, сколько человек и кто пользуются БД?

Пример 1 (продолжение). Выполним *анализ требований* к БД «Сессия студентов». *Вопрос 1.* Для каких типов задач (приложений) проектируется БД? *Ответ.* Для трех типов задач: Задача 1. Информация о студентах. Задача 2. Информация о преподавателях. Задача 3. Информация об успеваемости студентов. Задача 4. Информация о предметах.

Вопрос 2. Какими информационными объектами характеризуются эти задачи? *Ответ.* Задача 1 характеризуется информационным объектом: *личные дела студентов.* Задача 2 характеризуется информационным объектом: *личные дела преподавателей.* Задача 3 характеризуется одним информационным объектом - *сессия.* Задача 4 характеризуется одним информационным объектом - *предметы.*

Вопрос 3. Каким текущим запросам должны удовлетворять данные информационные объекты?

Вопрос 4. Каким перспективным запросам должны удовлетворять информационные объекты в БД «Сессия студентов»?

Пример 2. Пусть требуется разработать требования к локальной БД «Аэропорт»**.**

Вопрос 1. Для каких типов задач (приложений) проектируется БД? *Ответ.* Для трех типов задач: Задача 1. Информация об обслуживающем персонале. Задача 2. Информация о полетных средствах Задача 3. Информация о графике движения самолетов.

Вопрос 2. Какими информационными объектами характеризуются эти задачи? *Ответ.* Задача 1 характеризуется тремя информационными объектами: *летный состав, диспетчеры, технический персонал.* Задача 2 характеризуется двумя информа-

ционными объектами: *самолет, взлетное поле.* Задача 3 характеризуется одним информационным объектом - *рейсы.*

Вопрос 3. Каким текущим запросам должны удовлетворять данные информационные объекты? *Ответ.*

1. ФИО, звание, должность членов экипажа самолета.
2. Списочный состав диспетчеров.
3. Состав смены технического персонала.
4. Тип самолета, который может обслуживать тот или иной пилот.
5. Номер самолета, который обслуживает данный пилот, данная смена диспетчеров и технического персонала.
6. Номер личного дела сотрудника аэропорта.
7. Номер смены диспетчеров и технического персонала, обслуживающего аэропорт в заданном интервале времени.
8. Готовность самолета с номером № к полету.
9. Количество часов налета самолета с № ...
10. Готовность данной взлетной полосы в настоящее время.
11. Длина данной полосы.
12. Номер (номера) рейса до данного пункта назначения.
13. Какие промежуточные посадки совершает рейс № …?
14. Время вылета и расчетное время прибытия рейса № …
15. Время и место регистрации рейса №
16. Время посадки на рейс № …
17. До какого времени задерживается рейс № …?
18. Какие типы самолетов обслуживают рейс №. …?
19. Какой номер самолета обслуживает рейс *№ …*?

Вопрос 4. Каким перспективным запросам должны удовлетворять информационные объекты в БД «Аэропорт»?

Ответ.

1. С какого года используется самолет с № в аэропорту, тип самолета?
2. Какое количество часов полета у члена экипажа, ФИО?

Расчетное время отпуска члена экипажа, диспетчера, технического работника.

Выявление информационных объектов и связей между ними

Вторая фаза анализа предметной области состоит в выборе информационных объектов, задании необходимых свойств для

каждого объекта, выявлении связей между объектами, определении ограничений, накладываемых на информационные объекты, типы связей между ними, характеристики информационных объектов.

При выборе информационных объектов следует ответить на следующие вопросы:

1. На какие классы можно разбить данные, подлежащие хранению в БД?
2. Какое имя можно присвоить каждому классу данных?
3. Какие наиболее интересные характеристики (с точки зрения пользователя) каждого класса данных можно выделить?
4. Какие имена можно присвоить выбранным наборам характеристик?

В ходе выявления связей между информационными объектами следует ответить на следующие вопросы:

1. Какие типы связей между информационными объектами?
2. Какое имя можно присвоить каждому типу связей?
3. Каковы возможные типы связей, которые могут быть использованы впоследствии?
4. Имеют ли смысл какие-нибудь комбинации типов связей?

Далее проектировщик пытается задать ограничения на объекты и их характеристики. Под ограничением целостности обычно понимают логические ограничения, накладываемые на данные. *Ограничение целостности* - это такое свойство, которое проектировщик задает для некоторого информационного объекта или его характеристики, и которое должно сохраняться для каждого их состояния.

При выявлении условий ограничения целостности проектировщик пытается ответить на следующие вопросы:

1. Какова область значений для числовых характеристик?
2. Каковы функциональные зависимости между характеристиками одного информационного объекта?
3. Какой тип отображения соответствует каждому типу связей?

Пример 1 (продолжение). Для БД «Сессия студентов» выберем следующие сущности: *институт, факультет, студент, преподаватель, дисциплина, ведомость.* Каждую сущность зададим набором атрибутов (ключевые атрибуты подчеркнем):

институт (сокращение, название, подчиненность, адрес, телефон, ФИО ректора).

факультет (код Факультета, название, код специальности, декан). ***кафедры факультета*** (код кафедры, название, код факультета, зав кафедрой). *студент* (номер зачетной книжки, ФИО, группа, пол, дата рождения, домашний адрес, телефон).

преподаватель {№ страхового свидетельства. ФИО, дата рождения, домашний адрес, телефон, должность, ученое звание, ученая степень, код кафедры, стаж).

дисциплина (шифр дисциплины, название, число часов, виды занятий, число читаемых семестров, на каких курсах преподается).

ведомость (№ п/п, номер зачетной книжки студента, код дисциплины, семестр, форма сдачи, дата сдачи, отметка, преподаватель). Определим связи между сущностями.

Имя связи учится изучает принадлежит

Связи между объектами

студент, факультет студент, дисциплина институт, факультет

учится

изучает

принадлежит

работает преподает экзамен

преподаватель, факультет

преподаватель, дисциплина

студент, дисциплина

преподаватель, студент

Рассмотрим некоторые ограничения на характеристики объектов:

1. Значение атрибута "телефон" (сущность - *институт)* задается целым положительным шестизначным числом, задавать значение будем по маске __-__-__ .

2. Значение атрибута "код факультета" (сущность *факультет)* лежит в интервале 0-10.

3. Значение атрибута "курс" (сущность - *студент)* лежит в интервале 1-6 и хранится первая цифра номера группы.

4. Значение атрибута "семестр" (сущность - *студент, дисциплина)* лежит в интервале 1-12.

5. Значение атрибута "число часов" (сущность - *дисциплина)* лежит в интервале 1-300.

6. Одному студенту может быть приписана только одна группа.

7. Один студент может учиться только на одном факультете.

8. Один студент в семестре сдает от 3 до 10 дисциплин.

9. Один студент изучает в семестре от 6 до 12 дисциплин.

10.Одному преподавателю приписывается только одна кафедра.

11.Один студент может пересдавать одну дисциплину не более трех раз.

Ключи: сокращение (названия института), код факультета, номер зачетной книжки, № страхового свидетельства преподавателя, шифр дисциплины, № п/п.

Построение концептуальной (инфологической) модели предметной области

Заключительная фаза анализа предметной области состоит в проектировании ее информационной структуры или концептуальной модели.

Концептуальная модель включает описания объектов и их взаимосвязей, представляющих интерес в, рассматриваемой предметной области (ПО) и выявляемых в результате анализа данных. Концептуальная модель применяется для структурирования предметной области с учетом информационных интересов пользователей системы. Она дает возможность систематизировать информационное содержание предметной области, позволяет как бы "подняться вверх" над ПО и увидеть ее отдельные элементы.

При этом, уровень детализации зависит от выбранной модели. Концептуальная модель является представлением точки зрения пользователя на предметную область и не зависит ни от программного обеспечения СУБД, ни от технических решений. Концептуальная модель должна быть стабильной. Могут меняться прикладные программы, обрабатывающие данные, может меняться организация их физического хранения, концептуальная модель

остается неизменной или увеличивается с целью включения дополнительных данных.

Одной из распространенных моделей концептуальной схемы является модель «сущность-связь». Остановимся на наиболее известной модели данного типа, названной по фамилии автора, - модели П. Чена, или ER-модели. Основными конструкциями данной модели являются сущности и связи.

Под *сущностью* понимают основное содержание объекта ПО, о котором собирают информацию. В качестве сущности могут выступать место, вещь, личность, явление. *Экземпляр* сущности - конкретный объект. Например, сущность (объект) - студент, экземпляр сущности - Иванов А. В.; сущность (объект) - институт, экземпляр сущности - КГУ.

Сущность принято определять *атрибутами* - поименованными характеристиками. Например, сущность - студент, атрибуты - ФИО, год рождения, адрес, номер группы и т.д.

Чтобы задать атрибут в модели, ему надо присвоить имя и определить область допустимых значений. Одно из назначений атрибута - идентифицировать сущность.

Задание: составьте описание предметной области в соответствии с персональным заданием и постройте модель «сущность-связь».

Контрольные вопросы

1. Из каких фаз состоит анализ предметной области?
2. Как выбираются информационные объекты?
3. Как выявляются связи между объектами?
4. Что такое ключ?
5. Что включает концептуальная модель?
6. Что понимается под сущностью?
7. Что такое атрибут?
8. Что такое экземпляр сущности?

Логическое проектирование базы данных. Нормализация отношений

Цель работы: выполнить этап логического проектирования реляционной базы данных. Создать спроектированную базу данных средствами имеющейся СУБД.

Содержание работы

Цель логического этапа проектирования - организация данных, выделенных на этапе инфологического проектирования в форму, принятую в выбранной СУБД. Задачей логического этапа проектирования является отображение объектов предметной области в объекты используемой модели данных, чтобы это отображение не противоречило семантике предметной области и было по возможности наилучшим (эффективным, удобным и т.д.). С точки зрения выбранной СУБД задача логического проектирования реляционной базы данных состоит в обоснованном принятии решений о том:

- из каких отношений должна состоять база данных;
- какие атрибуты должны быть у этих отношений;
- какие ограничения должны быть наложены на атрибуты и отношения базы данных, чтобы обеспечить ее целостность.

Требования к выбранному набору отношений и составу их атрибутов должны удовлетворять следующим условиям:

- отношения должны отличаться минимальной избыточностью атрибутов;
- выбранные для отношения первичные ключи должны быть минимальными;
- между атрибутами не должно быть нежелательных функциональных зависимостей;
- выбор отношений и атрибутов должен обеспечивать минимальное дублирование данных;
- не должно быть трудностей при выполнении операций включения, удаления и модификации данных;
- время выполнения запросов на выборку данных (см. описание запросов из варианта задания учебного пособия "Введение в проектирование реляционных баз данных") должно удовлетворять предъявляемым требованиям;

- перестройка набора отношений при введении новых типов должна быть минимальной.

Удовлетворение отмеченных требований обеспечивается аппаратом нормализации отношений. Нормализация отношений - это пошаговый обратимый процесс композиции или декомпозиции исходных отношений в отношения, обладающие лучшими свойствами при включении, изменении и удалении данных, назначение им ключей по определенным правилам нормализации и выявление всех возможных функциональных зависимостей.

Процесс получения реляционной схемы базы данных из ER-диаграммы включает следующие шаги:

1. Каждая простая сущность превращается в отношение. Простая сущность - сущность, не являющаяся подтипом и не имеющая подтипов. Имя сущности становится именем отношения.

2. Каждый атрибут становится возможным столбцом с тем же именем; может выбираться более точный формат исходя из возможностей СУБД. Столбцы, соответствующие необязательным атрибутам, могут содержать неопределенные значения; столбцы, соответствующие обязательным атрибутам, - не могут.

3. Компоненты уникального идентификатора сущности превращаются в первичный ключ отношения. Если имеется несколько возможных уникальных идентификатора, выбирается наиболее используемый.

4. Связи M:1 (и 1:1) становятся внешними ключами. Для этого делается копия уникального идентификатора с конца связи "один" и соответствующие столбцы составляют внешний ключ. Необязательные связи соответствуют столбцам, допускающим неопределенные значения; обязательные связи - столбцам, не допускающим неопределенные значения.

5. В таблицах, построенных на основе ассоциаций, внешние ключи используются для идентификации участников ассоциации, а в таблицах, построенных на основе характеристик и обозначений, использовать внешние ключи используются для идентификации сущностей, описываемых этими характеристиками и обозначениями. Специфицировать ограничения, связанные с каждым из этих внешних ключей.

6. Если в концептуальной схеме присутствовали подтипы, то возможны два способа:

а) все подтипы размещаются в одной таблице;

б) для каждого подтипа строится отдельная таблица.

При применении способа (а) таблица создается для наиболее внешнего супертипа. В таблицу добавляется по крайней мере один столбец, содержащий код ТИПА, и он становится частью первичного ключа. Для работы с подтипами могут создаваться представления. При использовании метода (б) супертип воссоздается с помощью конструкции *UNION*.

При обработке данных необходима гарантия сохранения целостности данных в базе, поэтому важным этапом проектирования реляционной базы данных является обеспечение целостности базы данных.

Выделяют три группы правил целостности:

- целостность по сущностям;
- целостность по ссылкам;
- целостность, определяемая пользователем.

Обеспечение целостности базы данных обеспечивается заданием ограничений целостности. Ограничение целостности - это некоторое утверждение, которое может быть истинным или ложным в зависимости от состояния базы данных.

По способам реализации ограничения целостности делятся на:

- декларативные, выполняемые средствами языка SQL;
- процедурные, выполняемые посредством триггеров и хранимых процедур.

При выполнении этой лабораторной работы в процессе построения реляционной модели данных должны быть обеспечены декларативные ограничения целостности. Декларативные ограничения целостности должны обеспечивать:

- задание первичных ключей для обеспечения целостности по сущностям;
- определение необходимых внешних ключей для обеспечения целостности по ссылкам;
- контроль функциональных ограничений на значения атрибутов, определяемых требованиями предметной области;

- задание неопределенных значений и значений по умолчанию;
- задание условий каскадного удаления и пр.

Последовательность выполнения лабораторной работы:

1. Изучить вопросы теории нормализации, условия нахождения отношения в той или иной нормальной форме

2. Выполнить процедуру построения реляционной модели данных из ER-модели, построив необходимый набор отношений. Определить состав атрибутов отношений.

Определить первичные и внешние ключи отношений.

Выполнить шаги по нормализации полученных отношений, приведя модель к третьей нормальной форме.

3. Задать необходимые декларативные ограничения целостности исходя из специфики предметной области.

4. Представить связи между первичными и внешними ключами в виде вертикальной диаграммы.

5. Средствами имеющейся СУБД создать спроектированную базу данных, ее таблицы, задать необходимые ограничения целостности.

6. На языке SQL записать выражения для указанных в варианте задания запросов на выборку данных из созданной базы данных. Проверить работоспособность написанных запросов в интерактивном режиме.

7. Оформить следующие разделы отчета:

- "Логическое проектирование реляционной модели базы данных", включив в него информацию из пп. 2 - 6;
- "Типовые запросы на выборку данных", включив в него тексты запросов на языке SQL на выборку данных из созданной базы данных.

Контрольные вопросы

1. Каковы задачи, решаемые на этапе логического проектирования?
2. Каковы базовые свойства реляционной модели данных?
3. В чем состоят требования структурной части реляционной модели данных?

4. В чем состоят требования манипуляционной части реляционной модели данных?
5. В чем состоят требования целостной части реляционной модели данных?
6. Каковы общие свойства нормальных форм?
7. Что такое функциональная, функционально полная зависимость?
8. Каковы условия нахождения отношений в первой нормальной форме?
9. Каковы условия нахождения отношений во второй нормальной форме?
10. Каковы условия нахождения отношений в третьей нормальной форме?
11. Каковы условия нахождения отношений в третьей усиленной нормальной форме?
12. Что понимается под многозначной зависимостью?
13. Каковы условия нахождения отношений в четвертой нормальной форме?
14. Что понимается под понятием "проецирование без потерь"?
15. Каковы условия нахождения отношений в пятой нормальной форме?
16. В чем состоят общие требования обеспечения ограничений целостности?
17. Каковы средства задания ограничений целостности в языке SQL?

Физическое проектирование базы данных. SQL-запросы, триггеры и хранимые процедуры

Цель работы: изучить основные средства языка SQL.

Теоретические положения

Для получения данных из реляционной базы данных предназначен оператор SELECT, который имеет следующий формат:

SELECT [ALL | DISTINCT] (<список полей> | *)
FROM <список таблиц>
[WHERE <предикат условия>]
[GROUP BY <список полей результата>]
[HAVING <предикат условия>]
[ORDER BY <список полей>]

Здесь используются следующие обозначения:

- ALL– результирующая таблица содержит все строки, в том числе повторяющиеся;
- DISTINCT– результирующая таблица содержит только различающиеся строки;
- * – результирующая таблица содержит все столбцы, полученные из части FROM;
- FROM– предложение, в котором указывается исходная таблица или операция декартово произведения или операция соединения таблиц;
- WHERE– предложение, которое содержит предикат выборки;
- GROUPBY– предложение, в котором указываются столбцы, по которым выполняется группирование строк;
- HAVING– предложение, которое содержит предикат для фильтрации групп (а не отдельных строк);
- ORDERBY– предложение, содержащее список столбцов по которым должно выполняться упорядочивание (ASC – по возрастанию, DESC– по убыванию).

Для получения связанной информации из нескольких таблиц используют условные соединения. Предусмотрены следующие виды условных соединений:

- **INNER**– внутреннее соединение. Выбираются пары строк, для которых выполняется условие соединения, заданное предикатом в предложении ON.
- **LEFT**– левое внешнее соединение. В результат включается внутреннее соединение таблиц, к которому добавляются строки из левой таблицы, не вошедшие во внутреннее соединение. Строки из левой таблицы, не вошедшие во внутреннее соединение, дополняются значениями NULLв соответствии со схемой результирующей таблицы.
- **RIGHT**– правое внешнее соединение. В результат включается внутреннее соединение таблиц, к которому добавляются строки из правой таблицы, не вошедшие во внутреннее соединение. Строки из правой таблицы, не вошедшие во внутреннее соединение, дополняются значениями NULLв соответствии со схемой результирующей таблицы.
- **FULL**– полное открытое соединение. В результат включается внутреннее соединение таблиц, к которому добавляются строки из левой таблицы, не вошедшие во внутреннее соединение, и строки из правой таблицы, не вошедшие во внутреннее соединение, которые дополняются значениями NULL в соответствии со схемой результирующей таблицы.

В языке SQL предусмотрены следующие основные агрегатные функции:

- COUNT({[ALL|DISTINCT] <имя атрибута> | *}) – количество строк с непустыми значениями атрибута. Если *, то количество всех строк таблицы, не зависимо от содержания. Для числовых и символьных атрибутов.
- SUM([ALL|DISTINCT] <имя атрибута>) – сумма значений. Для числовых атрибутов.
- AVG([ALL|DISTINCT] <имя атрибута>) – среднее значение. Для числовых атрибутов.
- MIN([ALL|DISTINCT] <имя атрибута>) – минимальное значение. Для числовых и символьных атрибутов.
- MAX([ALL|DISTINCT] <имя атрибута>) – максимальное значение. Для числовых и символьных атрибутов.

Хранимая процедура (Stored Procedure) – это именованный набор команд языка Transact-SQL, хранящийся на сервере в

качестве самостоятельного объекта БД. Синтаксис оператора создания хранимой процедуры следующий:

CREATE PROC[EDURE] <имя процедуры>
[{@<параметр> <тип>} [= <значение по умолчанию>]
[OUTPUT]] [,…n]
AS<SQLоператор> [,…n]

Здесь <имя процедуры> – уникальное имя; @<параметр> – имя формального параметра, которое должно содержать символ @; <SQLоператор> – операторы SQL, составляющие тело процедуры. Формальные параметры могут быть входными, если для них не указывается ключевое слово OUTPUT и входными-выходными, если указывается ключевое словоOUTPUT. Если для входного параметра указывается значение по умолчанию, то при вызове хранимой процедуры значение этого параметра может не указываться.

Для вызова хранимой процедуры необходимо использовать оператор вызова:

EXEC[UTE]
[@<переменная> =] <имя процедуры>
[[@<параметр> =] {<значение> | @<переменная> [OUT[PUT]] | [DEFAULT] }] [,…n]

Если при вызове указывается имена формальных параметров, то порядок следования передаваемых параметров может быть любым. Если же имена формальных параметров не указываются, то порядок следования фактических параметров должен быть такой же, как в объявлении хранимой процедуры. Выходные параметры должны быть помечены ключевым словом OUTPUT. Значение выходного параметра может быть присвоено переменной.

Получать данные из процедуры можно следующими способами:

- Посредством выходных параметров хранимой процедуры, которые помечаются ключевым словом OUTPUT.
- Через набор строк или набор данных. Набор строк возвращается из хранимой процедуры, если в теле процедуры был выполнен SELECT-запрос. Если в теле процедуры более одного запроса, то возвращается набор данных – набор данных.

- В виде кода завершения. Процедура всегда возвращает целочисленное значение. Его можно задать параметром в операторе RETURN в теле процедуры.

Триггер– процедура, связанная с таблицей или представлением, которая автоматически выполняется при выполнении операции вставки, изменения или удаления строки этой таблицы или представления. Триггер создается командой

CREATE TRIGGER <имя триггера>
ON <имя таблицы> | <имя представления>
{FOR | AFTER | INSTEAD OF}
{ [DELETE] [,] [INSERT] [,] [UPDATE]}
AS
<SQL оператор> […n]

Классификация триггеров по типу действия:

- INSERTTRIGGER – запускаются при выполнении команды INSERT;
- UPDATETRIGGER – запускаются при выполнении команды UPDATE;
- DELETETRIGGER– запускаются при выполнении команды DELETE.

Классификация триггеров по типу поведения:

- AFTER – триггер выполняется ***после успешного выполнения*** команды;
- INSTEADOF – триггер вызывается ***вместо выполнения*** команды. Для представлений можно использовать только триггер INSTEADOF.

AFTER триггер выполняется после того, как действие команды было завершено. Поэтому, если необходимо отменить действие команды, то в AFTER триггере надо использовать конструкцию ROLLBACK TRANSACTION. В этой же ситуации в INSTEAD OF триггере не надо отменять действие, т.к. оно не выполняется (т.е. не надо использовать ROLLBACK TRANSACTION). Но для фиксации операции сам триггер должен выполнить соответствующую операцию (INSERT, DELETE, UPDATE).

В теле триггера могут использоваться две псевдо-таблицы с именами **inserted** и **deleted**. Схема этих псевдо-таблиц совпадает со

схемой той таблицы или представления, с которой связан триггер. При выполнении команды INSERT псевдо-таблица inserted содержит все вставляемые строки, deleted – пустая. При выполнении команды DELETE псевдо-таблица inserted – пустая, deleted содержит удаленные строки. При выполнении команды UPDATE псевдо-таблица inserted содержит новые значения строк, deleted – старые (заменяемые) значения.

Порядок выполнения работы

1. Разработать схемы таблиц для предметной области. Таблиц должно быть не менее трех, например, две таблицы сущностей с первичными ключами и связующая таблица, включающая внешние ключи к первым двум таблицам. Согласовать с преподавателем схемы таблиц.
2. Разработать запросы, полезные для работы в предметной области. Запросы должны содержать реализации следующих операций реляционной алгебры:

    1) выборка из всех трех таблиц; проекция и выборка из всех трех таблиц;
    2) внутреннее условное соединение двух таблиц; естественное соединение двух таблиц;
    3) условное соединение (внутреннее) трех таблиц;
    4) внешнее соединение двух таблиц (левое, правое, полное);
    5) объединение двух таблиц (можно использовать объединение выборок из одной и той же таблицы);
    6) вычисление всех агрегатных функций;
    7) упорядочивание данных;
    8) группирование данных.

4. Запустить консольную программу для работы с СУБД.
5. Создать базу данных.
6. Создать таблицы.
7. Заполнить таблицы значениями. В каждой таблице должно быть не менее 10 строк

8. В редакторе SQL запросов выполнить разработанные запросы. Текст SQL-скриптов с запросами сохранить в файл. Результаты запросов также сохранять в файл.
9. Создать дополнительную таблицу для хранения изменений. Назначение журнала изменений (пусть соответствующая таблица называется LOG) состоит в следующем. Если в какой-то таблице происходит изменение, то в LOG записывается характер изменения и время его выполнения.
10. Для одной из таблиц разработать по одному триггеру на каждую операцию, которые записывали бы в журнал записи об изменениях таблицы при добавлении, изменении и удалении записей.
11. Разработать триггер, выполняющий каскадное удаление записей из связных таблиц при удалении записи из родительской таблицы.
12. Разработать хранимую процедуру архивации данных одной из таблиц. Для этого создать таблицу для хранения архивных данных. В исходную таблицу добавить атрибут с индикатором, указывающим на то, что запись была занесена в архивную таблицу.

Рекомендации

1. Для создания базы данных надо выполнить команду CREATE DATABASE <имя базы данных>.
2. Для того, чтобы созданная база данных стала текущей для последующих действий, выполнить команду USE <имя базы данных>
3. Создать таблицы используя команду CREATETABLE. Например, CREATE TABLE E (fio CHAR(20), disc CHAR (30), mark INTEGER)
4. Для добавления новых значений в базу данных использовать оператор INSERT, например, INSERT INTO g (fio, gr) VALUES ('Petrov', 'IVT-200').
5. Для изменения данных использовать оператор UPDATE, а для удаления записей – оператор DELETE.

6. Для сохранения результатов запроса в файл следует в окне результирующей таблицы выполнить команду локального меню «Select All» и команду сохранения в файл.

Содержание отчета

1. Все таблицы с исходными данными.
2. Тексты всех разработанных SQL скриптов. Команды создания таблиц должны сопровождаться письменным объяснением назначения таблиц и их атрибутов.
3. Результаты выполнения запросов и объяснение полученных результатов.
4. Скрипты создания хранимых процедур и триггеров.
5. Описание результатов выполнения запросов, в которых задействованы хранимые процедуры и триггеры.
6. Выводы по работе.

Контрольные вопросы

1. Операции реляционной алгебры.
2. Оператор SELECT и реализация с его помощью операций реляционной алгебры.
3. Обобщающие функции.
4. Группирование записей.
5. Хранимые процедуры.
6. Триггеры.
7. Процедурный способ обеспечения целостности БД.
8. Декларативный способ обеспечения целостности БД.
9. Расширение языка SQL.

Разработка клиентского приложения базы данных

Цель работы: разработать программу на прикладном языке программирования для работы с базой данных.

Спецификация программы:

Программа должна подключаться ранее созданной базе данных и должна позволять пользователю выполнять предусмотренные требования к базе данных операции. Соединение с базой данной с помощью технологии ADO (от англ. ActiveX Data Objects — «объекты данных ActiveX») — интерфейс программирования приложений для доступа к данным, разработанный компанией Microsoft.

Для создания формы использовать компоненты:

Label – для подписей

Button – для инициирования действий

Edit – для вывода количества полей (колонок) и записей (строк) таблицы

ADOConnection –компонент для подключения к базе данных

ADOTable – компонент для работы со структурой и данными таблицы базы данных

DataSource – компонент для передачи данных компоненту DBGrid и DBNavigator.

DBGrid – компонент для визуализации таблицы из БД

DBNavigator – компонент для редактирования записей подключенной таблицы БД

Примерная компоновка формы программы представлена на рисунке 1.

Рекомендации для выполнения лабораторной работы:

1) сохранить ранее созданную базу данных в папке программы;

2) запустить C++ Builder. При запуске автоматически создается новый проект. Окно C++ Builder показано на рисунке 2. Для создания нового проекта, в случае если он не создался автоматически или вы его закрыли, выполнить команду меню File / New /Application;

3) сохранить проект в свою рабочую папку, выполнив команду меню File / Save Project As. Будет сохранено несколько файлов проекта;

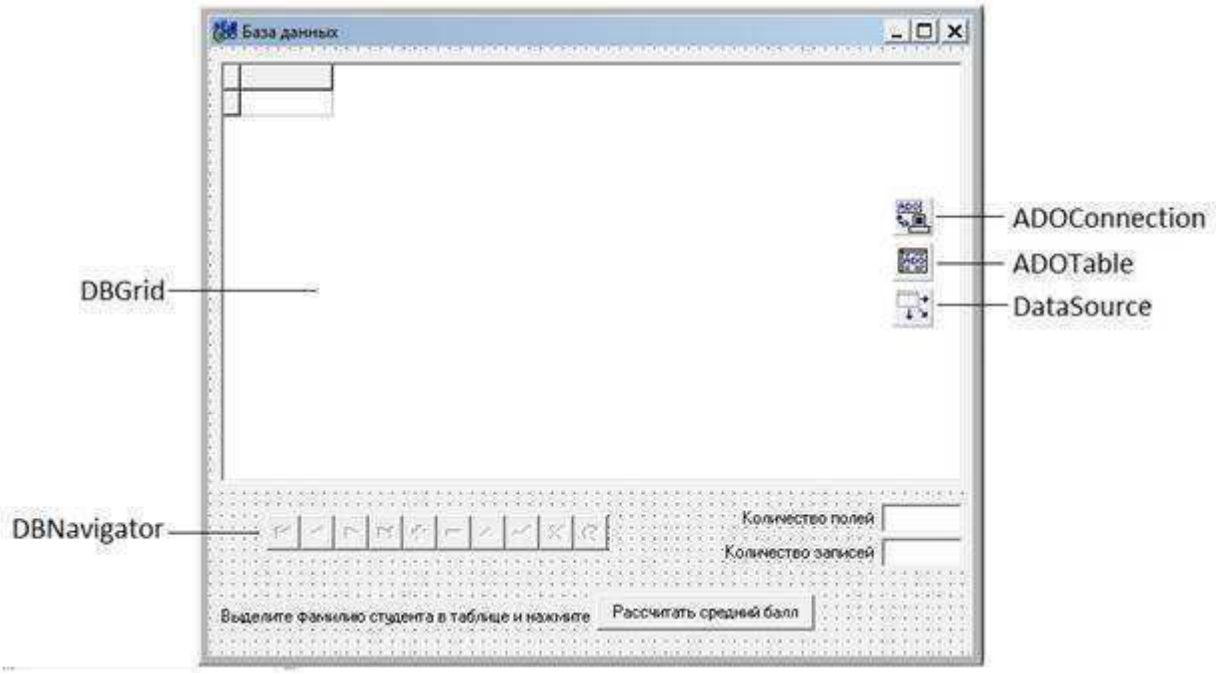

Рис. 1. Примерная компоновка формы

4) расположить на форме требуемое количество объектов;

Вкладка Standard: Label, Button, Edit.

Вкладка ADO: ADOConnection, ADOTable.

Вкладка DataAccess: DataSource.

Вкладка DataControls: DBGrid, DBNavigator.

5) изменить подписи объектов Label и пользовательской формы Form1. Для этого необходимо у перечисленных объектов отредактировать свойство Caption в соответствии с рисунком 1;

6) у объектов Edit и ComboBox очистить поле свойства Text;

7) поскольку объекты Edit используются только для вывода, то необходимо присвоить свойству ReadOnly для этих объектов значение true;

8) настроить подключение к БД следующим образом.

Выделите компонент ADOConnection1. Установить значение false для свойств Connected (соединение) и LoginPromt (вход с паролем) Сформировать строку подключения ConnectionString (строка параметров подключения к базе данных), нажав на кнопку с тремя точками.

Рис. 2. Настройка свойств объекта ADOConnection1

В появившемся окне выберите пункт «Use Connection String» и нажмите на кнопку «Build»:

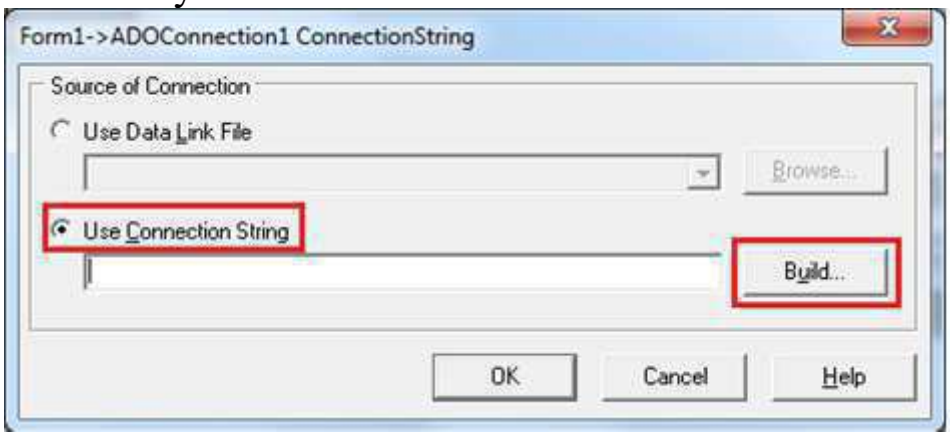

Рис. 3. Окно настройки подключения

Далее необходимо выбрать поставщика данных и нажать на кнопку далее:

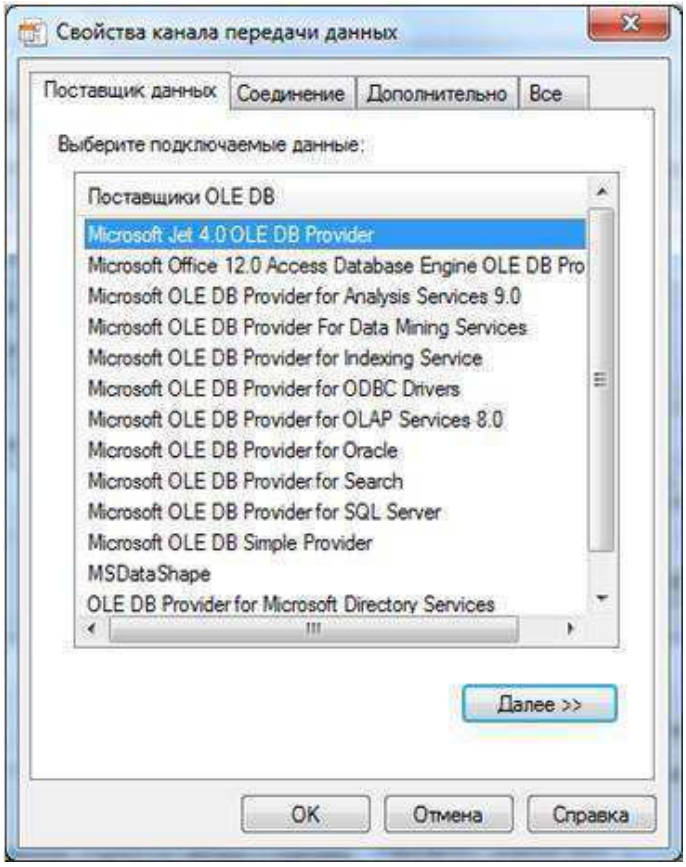

Рис. 4. Выбор поставщика данных

Указать путь к базе данных и проверить соединение.

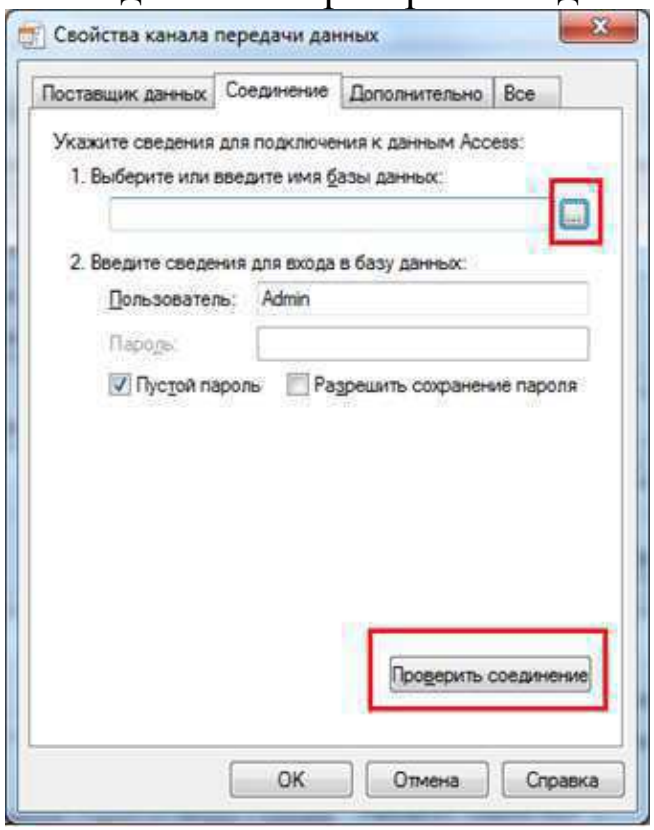

Рис. 5. Выбор файла базы данных

Примените все изменения и поменять значение свойства Connected на true.

Важное примечание: Строка подключения представляет собой обычную строку, в которой перечислены параметры подключения программы к базе данных. Строка подключения содержит путь к базе данных, который при необходимости можно заменять программно. Это необходимо для подключения различных баз данных одного типа к программе.

На этом настройка компонента ADOConnection1 закончена.

Выделите объект ADOTable и в окне «Object Inspector» в поле Connection выберите объект ADOConnection1, а в поле TableName выбрать таблицу из базы данных. Если при выборе таблицы возникает, то соединение с базой данных не было установлено. В этом случае следует проверить строку подключения в объекте ADOConnection1. В самую последнюю очередь установить переключатель Active в положение true.

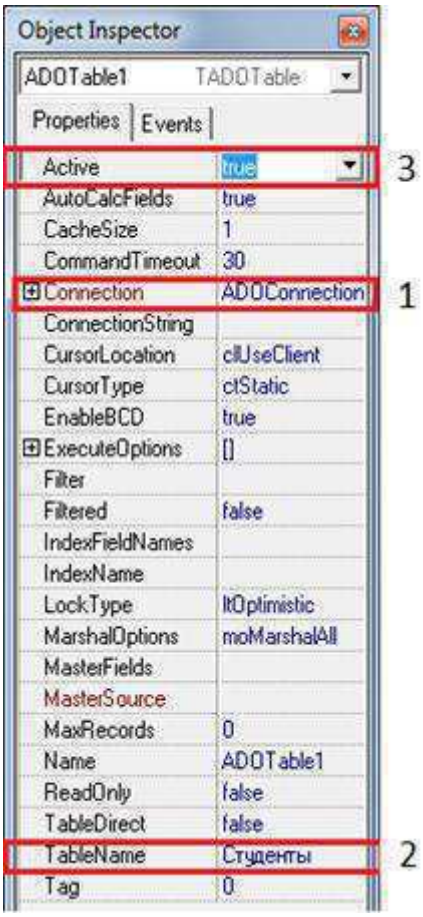

Рис. 6. Настройка объекта ADOTable1

Выделите объект DataSource1 и в списке свойств в поле DataSet выберите объект ADOTable1.

Рис. 7. Настройка объекта DataSource1

Для объектов DBGrid и DBNavigator в поле свойства DataSource выберите объект DataSource1

При правильном выполнении всех вышеперечисленных операций в объект DBGrid должна быть загружена таблица из базы данных. Настройка подключения таблицы базы данных к программе завершена. Далее необходимо написать код для работы с базой.

Контрольные вопросы

1. Операции реляционной алгебры.
2. Оператор SELECT и реализация с его помощью операций реляционной алгебры.
3. Обобщающие функции.
4. Группирование записей.
5. Хранимые процедуры.
6. Триггеры.
7. Процедурный способ обеспечения целостности БД.
8. Декларативный способ обеспечения целостности БД.

Разработка веб-приложения базы данных на сервере MySQL с помощью php

***Цель работы*:** изучить настройки сервера БД MySQL, возможности подключения к БД с помощью PHP

Краткие теоретические сведения

MySQL – это реляционная системауправления базами данных. То есть данные в ее базах хранятся в виде логически связанных между собой таблиц, доступ к которым осуществляется с помощью языка запросов SQL. MySQL – свободно распространяемая система Кроме того, это достаточнобыстрая, надежная и простая в использовании СУБД, вполне подходящая для не слишком глобальных проектов.

Работать с MySQL можно в текстовом режиме и в графическом. Существует очень популярный визуальный интерфейс (написанный на PHP) для работы с этой СУБД – называется PhpMyAdmin. Этот интерфейс позволяет значительно упростить работу с базами данных в MySQL.

В текстовом режиме работа с базой данных выглядит просто как ввод команд в командную строку, а результаты выборок возвращаются в виде своеобразных таблиц, поля в которых налезают друг на друга, если данные не помещаются на экран.

PhpMyAdmin позволяет пользоваться всеми достоинствами браузера, включая прокрутку изображения, если оно не умещается на экран. Многие из базовых SQL-функций работы с данными в PhpMyAdmin сведены к интуитивно понятным интерфейсам и действиям, напоминающим переход по ссылкам в Internet.

Сервер MySQL очень распространен и часто используется в Веб–технологиях. На этом сервере базируются большинство сайтов и решений в Интернете.

По настройке сервера MySQL можно посмотреть здесь: http://www.netbeans.org/kb/docs/ide/install-and-configure-mysql-server_ru.html

Пример:

1 Для создания новой БД на сервере (http://localhost/denwer/) выбрать ссылку: phpMyAdmin - администрирование СУБД MySQL:

2 Создать таблицу в БД, например, для хранения информации о кофе (info_coffee)

3. Создать набор полей, указать типы данных, размер текстовых полей. Если необходимо указать «auto_increment» для ключевого поле (предварительно его создав – primary key). Нажать «Сохранить»

4 Для добавления данных нажать «Вставить»:

5. Для просмотра всех записей нажать «Обзор»:

Взаимодействие php и MySql

Чтобы построить интерфейс для добавления информации в эту таблицу, нужно ее структуру (т.е. набор ее полей) отобразить в html-форму.

Разобьем эту задачу на следующие подзадачи:

- установка соединения с БД;
- выбор рабочей БД;
- получение списка полей таблицы;
- отображение полей в html-форму.

После этого данные, введенные в форму, нужно записать в базу данных. Рассмотрим все эти задачи по порядку.

Итак, первое, что нужно сделать, - это установить соединение с базой данных. Воспользуемся функцией mysql_connect:

ресурс mysql_connect ([строка server

[, строка username [, строка password

[, логическое new_link

[, целое client_flags]]]]])

Данная функция устанавливает соединение с сервером MySQL и возвращает указатель на это соединение или FALSE в случае неудачи. Для отсутствующих параметров устанавливаются следующие значения по умолчанию:

server = 'localhost:3553' (если по умолчанию используется порт 80, то ничего указывать не нужно)

username = имя пользователя владельца процесса сервера (по умолчанию root)

password = пустой пароль

```
<?
$conn = mysql_connect("localhost", "root","")
```

```
or die("Невозможно установить соединение: ". mysql_error());
echo "Соединение установлено";
mysql_close($conn);
?>
```

Соединение с сервером закрывается при завершении исполнения скрипта, если оно до этого не было закрыто с помощью функции mysql_close().

После установки соединения нужно выбрать базу данных, с которой будем работать. Наши данные хранятся в базе данных coffee.В PHP для этого существует функция mysql_select_db.

Синтаксис mysql_select_db:

логическое mysql_select_db (строка database_name[, ресурс link_identifier])

Эта функция возвращает TRUE в случае успешного выбора базы данных и FALSE - в противном случае.

```
mysql_select_db("coffee");
```

Как получить список полей таблицы? В PHP есть своя команда - mysql_list_fields.

Синтаксис mysql_list_fields

ресурс mysql_list_fields (строка database_name, строка table_name

[, ресурс link_identifier])

Эта функция возвращает список полей в таблице table_name в базе данных database_name. Результат работы этой функции – переменная типа ресурс. Это ссылка, которую можно использовать для получения информации о полях таблицы, включая их названия, типы и флаги.

Функция mysql_field_name возвращает имя поля, полученного в результате выполнения запроса. Функция mysql_field_len возвращает длину поля. Функция mysql_field_type возвращает тип поля, а функция mysql_field_flags возвращает список флагов поля, записанных через пробел. Типы поля могут быть int, real, string, blob и т.д. Флаги могут быть not_null, primary_key, unique_key, blob, auto_increment и т.д.

Синтаксис у всех этих команд одинаков:

строка mysql_field_name (ресурс result, целое field_offset)

строка mysql_field_type (ресурс result, целое field_offset)

строка mysql_field_flags (ресурс result, целое field_offset)

строка mysql_field_len (ресурс result, целое field_offset)

Здесь result – это идентификатор результата запроса (например, запроса, отправленного функциями mysql_list_fields или mysql_query, а field_offset - порядковый номер поля в результате.

То, что возвращают функции типа mysql_list_fields или mysql_query, представляет собой таблицу, а точнее, указатель на нее. Чтобы получить из этой таблицы конкретные значения, нужно задействовать специальные функции, которые построчно читают эту таблицу. К таким функциям и относятся mysql_field_name и т.п. Чтобы перебрать все строки в таблице результата выполнения запроса, нужно знать число строк в этой таблице. Команда mysql_num_rows(ресурс result) возвращает число строк во множестве результатов result.

А теперь попробуем получить список полей таблицы info_coffee (коллекция экспонатов).

```
<?
$conn = mysql_connect("localhost","root","")
or die("Невозможно установить соединение: ". mysql_error());
echo "Соединение установлено";
mysql_select_db("info_coffee");
$list_f = mysql_list_fields ("coffee","info_coffee",$conn);
$n = mysql_num_fields($list_f);
for($i=0;$i<$n; $i++){
$type = mysql_field_type($list_f, $i);
$name_f = mysql_field_name($list_f,$i);
$len = mysql_field_len($list_f, $i);
$flags_str = mysql_field_flags (
$list_f, $i);
echo "<br>Имя поля: ". $name_f;
echo "<br>Тип поля: ". $type;
echo "<br>Длина поля: ". $len;
echo "<br>Строка флагов поля: ".
$flags_str . "<hr>";
}
?>
```

Создадим форму для наполнения БД:

```
<form action=insert.php method=post>
введите название <input type=text name=name_c> <br>
введите цену <input type=text name=price_c> <br>
выберите фирму <select name=firma_c>
<option> nescafe
<option> jacobs
<option> tchibo
</select>
<br><input type=submit name=ok value=insert_c>
</form>
```

Теперь нужно сделать самое главное – отправить данные из этой формы в нашу базу данных. Как вы уже знаете, для того чтобы записать данные в таблицу, используется команда INSERT языка SQL. Например:

```
mysql> INSERT INTO info_coffee
SET name='Черная карта арабика';
```

Возникает вопрос, как можно воспользоваться такой командой (или любой другой командой SQL) в PHP скрипте. Для этого существует функция mysql_query().

Синтаксис mysql_query:

ресурс mysql_query (строка query [, ресурс link_identifier])

mysql_query() посылает SQL-запрос активной базе данных MySQL сервера, который определяется с помощью указателя link_identifier (это ссылка на соединение с сервером MySQL). Если параметр link_identifier опущен, используется последнее открытое соединение. Если открытые соединения отсутствуют, функция пытается соединиться с СУБД, аналогично функции mysql_connect() без параметров. Результат запроса буферизируется.

Insert.Php

```
<?php
$name=$_REQUEST["name_c"];
$price=$_REQUEST["price_c"];
$firma=$_REQUEST["firma_c"];
$db=mysql_connect(localhost,'root','');
mysql_select_db("Coffee");
if ($but=insert_c)
```

```
{$sql1="INSERT INTO name_coffee
VALUES('','$name','$price','$firma')";
mysql_query($sql1);
}
print "Кофе $name $firma добавлено";
?>
```

poisk.php

```
$a=$_REQUEST["price_c"];
{print " кофе по цене < $a грн: <br>"; }
$conn=mysql_connect(localhost,'root','');
mysql_select_db("Coffee");
$sql="SELECT * FROM info_coffee where price=".$a;
$result=mysql_query($sql);
for ($i=1; $i<=mysql_num_rows($result);$i++)
{$row=mysql_fetch_object($result);
print "Кофе $row->name $row->firma стоит $row->price грн.
<br>";
}
```

Предварительно была создана форма для ввода цены кофе, которое необходимо найти:

```
<form action=poisk.php method=post>
введите цену <input type=text name=price_c> <br>
<input type=submit name=ok value=search>
</form>
```

Примечание. БД и таблицы хранятся на сервере по следующему пути:

Webserver\usr\local\mysql5\data\coffee

Задание к работе:

1. Ознакомиться с теоретическим материалом.
2. Используя существующую базу данных из предыдущих работ создать скрипт на php для просмотра записей таблиц, поиска по критерию, добавления информации, удаления записей и изменения информации
3. Оформить отчет согласно требованиям.

Отчет должен содержать:

1. Название и цель работы.

2. Ход работы с детальным описанием выполненных действий с рисунками, листингом кода.
3. Экранные формы браузера с загруженными страницами.
4. Выводы о проделанной работе.

Контрольные вопросы:

1. Взаимодействие PHP и СУБД MySql.
2. Как происходит установка соединения с базой данных?
3. Функции отправки запросов и обработке ответов.
4. Обработка ошибок при подключении к БД.

Список литературы

1. Громов, Ю.Ю. Управление данными [Электронный ресурс] : учебник / Ю.Ю. Громов и др. - Тамбов : Изд-во ФГБОУ ВПО "ТГТУ", 2015. - 192 с. - Режим доступа / http://biblioclub.ru/index.php?page=book_view_red&book_id=444642.
2. Цехановский, В.В. Управление данными [Текст] : учебник / В. В. Цехановский, В. Д. Чертовской. - Санкт-Петербург : Лань, 2015. - 432 с.
3. Васюков, О.Г. Управление данными [Электронный ресурс] : учебно-методическое пособие / О.Г. Васюков. - Самара : СГАСУ, 2014. - 161 с. - Режим доступа / http://biblioclub.ru/index.php?page=book_view_red&book_id=438334.
4. Кузовкин, А.В. Управление данными [Текст]: учебник / А.В. Кузовкин, А.А. Цыганов, Б.А. Щукин. – М.: Академия, 2010. - 256 с.