ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ГОРОДА МОСКВЫ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ГОРОДА МОСКВЫ

«КОЛЛЕДЖ СВЯЗИ № 54» ИМЕНИ П.М.ВОСТРУХИНА

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ**

**ПО ПРОВЕДЕНИЮ ПРАКТИЧЕСКИХ ЗАНЯТИЙ**

по ОП. 05. Основы программирования и баз данных

для специальности: 230111 Компьютерные сети

Москва,2015

|  |  |
| --- | --- |
| Рассмотрено:на заседании цикловой комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Протокол № \_\_\_\_ от «\_\_\_\_» \_\_\_ 2015г | УтверждаюЗам. директора по УМР  \_\_\_\_\_\_\_\_\_\_\_ И.Г.Бозрова«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2015г. |
| Председатель ПЦК\_\_\_\_\_\_\_\_\_\_\_\_ С.Н. Хохлов  |  |

Разработчик: Казиханов Ф.И., преподаватель ГБПОУ КС №54

Предисловие

*В результате освоения дисциплины обучающийся должен уметь:*

- использовать языки программирования высокого уровня;

- строить логически правильные и эффективные программы;

- использовать язык SQL для программного извлечения сведений из баз данных.

*В результате освоения дисциплины обучающийся должен знать:*

- общие принципы построения алгоритмов;

- основные алгоритмические конструкции;

- системы программирования;

- технологии структурного и объектно – ориентированного программирования

- основы теории баз данных;

- модели баз данных;

- основы реляционной алгебры

- принципы проектирования баз данных;

- средства проектирования структур баз данных;

- язык запросов SQL.

Правила проведения практических занятий

* 1. *Обучающийся должен подготовить ответы на теоретические вопросы к ПЗ;*
	2. *Перед началом каждогозанятия проверяется готовность обучающегося к ПЗ;*
	3. *После завершения ПЗ обучающийся должен сдатьпреподавателю отчет о проделанной работе, выполненной в тетради;*
	4. *Обучающийся, пропустивший ПЗ по уважительной или неуважительной причинам, обязан выполнить задания ПЗ в дополнительно назначенное время.*
	5. *Оценка за выполненные задания обучающемуся выставляется с учетом предварительной подготовки к занятию, доли самостоятельности при выполненииработы, точности и грамотности оформления отчета.*

ПРАКТИЧЕСКИЕ ЗАНЯТИЯ

Практическое занятие №1

«Составление алгоритма работы программы»

Цель: формирование знаний и умений по работе с блок-схемами алгоритмов для составления программ.

Пояснения.

Понятие алгоритма в программировании является фундаментальным. Для алгоритма важен не только набор определенных действий, но и то, как они организованы, т.е. в каком порядке они выполняются. Это одно из общих свойств алгоритма. Другое общее свойство алгоритма состоит в том, что каждое последующее действие выполняется лишь после завершения предшествующего.

### Свойства алгоритма

При составлении и записи алгоритма необходимо обеспечить, чтобы он обладал рядом свойств. Рассмотрим эти свойства на следующем примере: пусть требуется вычислить сумму двух целых чисел и вывести на экран результат.

***Однозначность алгоритма,*** под которой понимается единственность толкования исполнителем правил выполнения действий и порядка их выполнения. Чтобы алгоритм обладал этим свойством, он должен быть записан командами из системы команд исполнителя.

Для нашего примера исполнитель алгоритма должен понимать такую запись действий, как сложить числа А и В.

***Конечность алгоритма*** *-* обязательность завершения каждого из действий, составляющих алгоритм, и завершимость выполнения алгоритма в целом, т.е. алгоритм должен заканчиваться после конечного числа шагов.

***Результативность алгоритма,*** предполагающая, что выполнение алгоритма должно завершиться получением определенных результатов. Алгоритм в нашем примере обладает этим свойством, так как для целых чисел А и В всегда будет вычислена сумма.

***Массовость****,* т. е. возможность применения данного алгоритма для решения целого класса задач, отвечающих общей постановке задачи. Для того чтобы алгоритм обладал свойством массовости, следует составлять алгоритм, используя обозначения величин и избегая конкретных значений.

***Правильность алгоритма,*** под которой понимается способность алгоритма давать правильные результаты решения поставленных задач. Представленный в примере алгоритм обладает свойством правильности, так как в нем использована правильная формула сложения целых чисел, и для любой пары целых чисел результат выполнения алгоритма будет равен их сумме.

***Определенность алгоритма****.* Каждый шаг алгоритма должен быть определен**.**

***Входные данные алгоритма.***Алгоритм должен иметь некоторое (может быть равное 0) число входных данных.

***Выходные данные алгоритма*.** Результатом выполнения алгоритма должна быть одна или несколько выходных величин, зависящих от исходных данных.

***Эффективность алгоритма****.* Алгоритм должен быть эффективным, т.е. результат должен быть получен наименьшим числом наиболее простых операций.

Основные блоки, используемые при составлении алгоритмов

|  |  |  |
| --- | --- | --- |
| Название | Обозначение | Назначение |
| Пуск, Останов | C:\БТБиП\Языки программирования\Все для методички для ЯП\рисунки к методичке\1.bmp | Начало-конец алгоритма |
| Процесс | C:\БТБиП\Языки программирования\Все для методички для ЯП\рисунки к методичке\2.bmp | Любое вычислительное действие |
| Решение | C:\БТБиП\Языки программирования\Все для методички для ЯП\рисунки к методичке\3.bmp | Проверка условия |
| Модификатор | C:\БТБиП\Языки программирования\Все для методички для ЯП\рисунки к методичке\4.bmp | Цикл |
| Ввод-вывод | C:\БТБиП\Языки программирования\Все для методички для ЯП\рисунки к методичке\5.bmp | Ввод-вывод данных |
| Документ | C:\БТБиП\Языки программирования\Все для методички для ЯП\рисунки к методичке\6.bmp | Вывод на печатающее устройство |
| Соединитель | C:\БТБиП\Языки программирования\Все для методички для ЯП\рисунки к методичке\7.bmp | Используется на линиях разрыва |
| Комментарий | C:\БТБиП\Языки программирования\Все для методички для ЯП\рисунки к методичке\8.bmp | Комментарий |

*Пример.* Составить схему алгоритма работы программы вычисления значения выражения: 

 Для начала для построения блок –схемы алгоритма опишем последовательность действий, необходимых для решения данной задачи:

* начало
* ввод чисел a,b
* вычисление х
* вычисление z
* вывод результата
* конец

Исходя из этого составляем блок-схему алгоритма согласно ГОСТ, используя соответствующие блоки.



Необходимые принадлежности.

1. Персональный компьютер с программным обеспечением.

2. Справочная литература.

Работа в аудитории.

1. Изучить теоретические сведения по теме ”Построение блок-схем алгоритмов”.
2. Решить с использованием персонального компьютера и программного обеспечения задания для практической работы.
3. Ответить письменно на контрольные вопросы.

Задание.

1) Составить программу для определения суммы цифр заданного трехзначного числа.

2) Составить программу, в которой для заданного q вычисляется один из корней уравнения Ln(ctgx-1)=q.

3) Вычислить периметр, площадь и углы прямоугольного треугольника по заданным длинам катетов.

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Алгоритм программы;
6. Заключение

Контрольные вопросы.

1. Свойства алгоритма.
2. Блок схемы. Понятие и правила построения.
3. Примеры построения блок-схем алгоритмов.

Практическое занятие №2

«Составление алгоритма работы программы с использованием ветвления»

Цель: формирование знаний и умений по работе с блок-схемами алгоритмов для составления программ с использованием ветвления.

Пояснения.

*Линейные алгоритмы* - это алгоритм, в котором все его действия выполняются одно за другим, т.е. последовательно.

Однако в большинстве вычислительных процессов мы сталкиваемся с тем, что выбор хода дальнейших действий определяется результатом предыдущих. Такие алгоритмы называются разветвляющимися.

*Разветвляющиеся алгоритмы* - это алгоритмы, в которых в зависимости от выполнения или не выполнения некоторого условия совершается одна или другая последовательность действий.



Пример.Составить схему алгоритма вычисления значения: x=a+b при a>b, x=a\*b, при a<=b.

****

Необходимые принадлежности.

1. Персональный компьютер с программным обеспечением.

2. Справочная литература.

Работа в аудитории.

1. Изучить теоретические сведения по теме ”Построение блок-схем алгоритмов”.
2. Решить с использованием персонального компьютера и программного обеспечения задания для практической работы.
3. Ответить письменно на контрольные вопросы.

Задание.

Составить алгоритм работы программы решения уравнения ax2+b=0 (неравенства ax2+b > 0) 3 способами: словарным описанием, блок-схемой и на языке псевдокода. При отсутствии решения или бесчисленном множестве решений должен быть напечатан соответствующий текст.

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Алгоритм программы;
6. Заключение

Контрольные вопросы.

1. Определение линейного алгоритма.
2. Определение разветвляющегося алгоритма.
3. Понятие псевдокода. Объяснить достоинства и недостатки псевдокода.

Практическое занятие №3

«Составление алгоритма работы программы с использованием цикла»

Цель: формирование знаний и умений по работе с блок-схемами алгоритмов для составления программ с использованием цикла.

Пояснения.

*Циклические алгоритмы*- алгоритмы, в которых одна и та же последовательность действий совершается несколько раз до тех пор, пока выполняются некоторые условия.На рисунке представлено графическое представление вычислительных процессов



Составить схему алгоритма вычисления значения:



Для начала для построения блок –схемы алгоритма опишем последовательность действий, необходимых для решения данной задачи:



Исходя из этого составляем блок-схему алгоритма согласно ГОСТ, используя соответствующие блоки.



Необходимые принадлежности.

1. Персональный компьютер с программным обеспечением.

2. Справочная литература.

Работа в аудитории.

1. Изучить теоретические сведения по теме ”Построение блок-схем алгоритмов”.
2. Решить с использованием персонального компьютера и программного обеспечения задания для практической работы.
3. Ответить письменно на контрольные вопросы.

Задание.

Составить алгоритм работы программы нахождения делителей заданного числа N 3 способами: словарным описанием, блок-схемой и на языке псевдокода.

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Алгоритм программы;
6. Заключение

Контрольные вопросы.

1. Определение циклических алгоритмов.

2. Типы циклических алгоритмов.

3. В чём состоят сложности при написании псевдокода циклических алгоритмов.

Практическое занятие №4

«Изучение интегрированной среды программирования»

Цель: Освоение интегрированной среды программирования, использование для разработки и отладки программ.

Пояснения.

Можно создавать файлы с исходным кодом на С# с помощью обычного текстового редактора, например, Блокнота, и компилировать их в управляемые модули с помощью компилятора командной строки, который является составной частью .NET Framework. Однако наиболее удобно для этих целей использовать VS, потому что:

1. VS автоматически выполняет все шаги, необходимые для компиляции исходного кода.
2. Текстовый редактор VS настроен для работы с теми языками, которые поддерживаются VS, например С#, поэтому он может интеллектуально обнаруживать ошибки и подсказывать в процессе ввода, какой именно код необходим.
3. В состав VS входят программы, позволяющие создавать Windows- и Web-приложения путем простого перетаскивания мышью элементов пользовательского интерфейса.
4. Многие типы проектов, создание которых возможно на С#, могут разрабатываться на основе "каркасного" кода, заранее включаемого в программу. Вместо того чтобы каждый раз начинать с нуля, VS позволяет использовать уже имеющиеся файлы с исходным кодом, что уменьшает временные затраты на создание проекта.

**Создание первого проекта**

Для создания проекта следует запустить VS, а затем в главном меню VS выбрать команду ***File – New - Project*** . После чего откроется диалоговое меню ***NewProject***(см. рис.1).



Рис.1.

Вполе***Project types***следуетвыбрать*Visual C#*, вполе***Templates*** – *Console Application*.

В строчке ***Name***введите имя приложения *Hello*. Обратите внимание на то, что это же имя появится в строчке ***SolutionName***. Уберите галочку в поле ***CreatedirectoryforApplication*** (пока мы создаем простое приложение, и нам нет необходимости усложнять его структуру).

В строке ***Location*** определите положение на диске, куда нужно сохранять ваш проект. И нажмите кнопку ***OK***. Примерный вид экрана изображен на рис 2.



Рис. 2.

В правой верхней части располагается окно управления проектом ***SolutionExplorer***. Если оно закрыто, то его можно включить командой ***View - SolutionExplorer***. В этом окне перечислены все ресурсы, входящие в проект:

1) AssemblyInfo.cs – информация о *сборке*.

Компилятор в качестве результата своего выполнения создает так называемую *сборку* – файл с расширением exe или dll, который содержит IL-код и метаданные.

1. System, System.Data, System.Xml – ссылки на стандартные библиотеки.
2. Program.cs - текст программы на языке C#.

***Замечание***. В других версиях VS сюда же включается файл с расширением ico, отвечающий за вид ярлыка приложения.

В правой нижней части экрана располагается окно свойств ***Properties***. Если оно закрыто, то его можно включить командой ***View - Properties***. В этом окне отображаются важнейшие характеристики выделенного элемента.

Основное пространство экрана занимает окно редактора, в котором располагается текст программы, созданный средой автоматически. Текст представляет собой каркас, в который программист будет добавлять нужный код. При этом зарезервированные слова отображаются синим цветом, комментарии – зеленым, основной текст – черным.

Текст структурирован. Щелкнув на знак минус, мы скроем блок кода, щелкнув на знаке плюс – откроем.

Откроем папку, содержащую проект, и рассмотрим ее структуру (см. рис.3). Файлы, выделенные жирным шрифтом, появятся только после компиляции.

Hello

bin

obj

Properties

Hello.csproj

Debug

**Hello.exe**

Hello.vshost.exe

**Hello.pdb**

Debug

TempPE

**Hello.csprojFileList.txt**

**Hello.exe**

**Hello.pdb**

AssemblyInfo.cs

Ва

Hello.sln

Hello.suo

Program.cs

Рис. 3.

На данном этапе особый интерес для нас будут представлять следующие файлы:

1. *Hello.sln*– основной файл, отвечающий за весь проект. Если необходимо открыть проект для редактирования, то нужно выбрать именно этот файл. Остальные файлы откроются автоматически.
2. *Program.cs* – файл, в котором содержится исходный код - код, написанный на языке С#. Именно с этим файлом мы и будем непосредственно работать.
3. *Hello.exe* – файл, в котором содержатся сгенерированный IL-код и метаданные проекта. Другими словами, этот файл и есть готовое приложение, которое может выполняться на любом компьютере, на котором установлена платформа .Net.

Теперь рассмотрим сам текст программы.

***usingSystem*** – это директива, которая разрешает использовать имена стандартных классов из пространства имен ***System*** непосредственно без указания имени пространства, в котором они были определены.

Ключевое слово ***namespace*** создает для проекта свое собственное пространство имен, которое по умолчанию называется именем проекта. В нашем случае пространство имен называется Hello. Однако программист вправе указать другое имя. Пространство имен ограничивает область применения имен, делая его осмысленным только в рамках данного пространства. Это сделано для того, чтобы можно было давать имена программным объектам, не заботясь о том, что они совпадут с именами в других приложениях. Таким образом, пространства имен позволяют избегать конфликта имен программных объектов, что особенно важно при взаимодействии приложений.

С# - объектно-ориентированный язык, поэтому написанная на нем программа будет представлять собой совокупность взаимодействующих между собой классов. Автоматически был создан класс с именем ***Program*** (в других версиях среды может создаваться класс с именем ***Class1***).

Данный класс содержит только один метод - метод ***Main()***. Метод Main() является точкой входа в программу, т.е. именно с данного метода начнется выполнение приложения. Каждая программа на языке С# должна иметь метод Main ().

***Замечание****.* Технически возможно иметь несколько методов Main() в одной программе, в этом случае потребуется с помощью параметра командной строки сообщить компилятору С#, какой именно метод Main() является точкой входа в программу.

Метод Main() имеет одну важную особенность. Перед объявлением типа возвращаемого значения ***void*** (который означает, что метод не возвращает значение) стоит ключевое слово ***static***, которое означает что метод Main() можно вызывать, не создавая объект типа ***Program***.

***Замечание****.* В некоторых версиях требуется, чтобы перед словом ***static*** стояло слово ***public***.

Добавим в метод следующий код: ***Console.WriteLine("Hello!");***

Здесь *Console* имя стандартного класса из пространства имен System. Его метод *WriteLine* выводит на экран текст, заданный в кавычках

Для запуска программы следует нажать клавишу F5 или выполнить команду
***Debug-StartDebugging.*** Если программа выполнена без ошибок, то сообщение выведется в консольное окно, которое мелькнет и быстро закроется. Чтобы просмотреть сообщение в нормальном режиме нужно нажать клавиши Ctrl+F5  или выполнить команду ***Debug-StartWithoutDebugging.*** В нашем случае откроется следующее консольное окно:

******

Если код программы будет содержать ошибки, например, пропущена точка с запятой после команды вывода, то после нажатия клавиши F5 откроется диалоговое окно, в котором выведется сообщение о том, что обнаружена ошибка, и вопрос, продолжать ли работу дальше. Если вы ответите ***Yes***, то будет выполнена предыдущая удачно скомпилированная версия программы. Иначе процесс будет остановлен и управление передано окну списка ошибок ***ErrorList***.

Необходимые принадлежности

1. Персональный компьютер с программным обеспечением.

2. Справочная литература

Работа в аудитории.

1. Запустить интегральную среду разработки.

2. Выполнить задание.

3. Продемонстрировать работоспособное приложение преподавателю.

Задание

1. Ознакомиться с меню и интерфейсом интегрированной среды программирования.

2. Создать консольный проект с именем MyFirstProgram.

3. Выполнить реализацию приложения согласно образцу:



Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Исходный код приложения;
6. Заключение.

Контрольные вопросы

1. Назначение ИСП.
2. В чём удобство использования ИСП.
3. Какие типы проектов существуют в ИСП.
4. Структура проекта. Назначение основных файлов.
5. Запуск отладки приложения.

Практическое занятие №5

«Составление требований к программному продукту»

Цель: научатся применять основные шаги и техники в процессе разработки требований от дополнительных и взаимозаменяемых, а так же отличать обязательные требования отнеобязательных.

Пояснения.

Требования к программному обеспечению — совокупность утверждений относительно атрибутов, свойств или качеств программной системы, подлежащей реализации. Создаются в процессе разработки требований к программному обеспечению, в результате анализа требований.

Требования могут выражаться в виде текстовых утверждений и графических моделей.

В классическом техническом подходе совокупность требований используется на стадии проектирования ПО. Требования также используются в процессе проверки ПО, так как тесты основываются на определённых требованиях.

Этапу разработки требований, возможно, предшествовало технико-экономическое обоснование, или концептуальная фаза анализа проекта. Фаза разработки требований может быть разбита на выявление требований (сбор, понимание, рассмотрение и выяснение потребностей заинтересованных лиц), анализ (проверка целостности и законченности), спецификация (документирование требований) и проверка правильности.

Функциональный характер — требования к поведению системы

* Бизнес-требования
* Пользовательские требования
* Функциональные требования

Нефункциональный характер — требования к характеру поведения системы

* Бизнес-правила — определяют ограничения, проистекающие из предметной области и свойств автоматизируемого объекта (предприятия)
* Системные требования и ограничения — определения элементарных операций, которые должна иметь система, а также различных условий, которым она может удовлетворять. К системным ограничениям относятся ограничения на программные интерфейсы, требования к атрибутам качества, требования к применяемому оборудованию и ПО.
* Атрибуты качества
* Внешние системы и интерфейсы
* Ограничения

**Проверка требований**

Все требования должны быть поддающимися проверке. Наиболее общепринятая методика проверки — тесты. Если проверка тестами невозможна, тогда должен использоваться другой метод проверки (анализ, демонстрация, осмотр или обзор дизайна).

Определённые требования, по своей сути, не являются поддающимися проверке. Они включают требования, которые говорят, что система *никогда* не должна или *всегда* должна показывать специфическое свойство. Надлежащее тестирование этих требований потребовало бы бесконечного цикла тестирования. Такие требования должны быть переопределены так, чтобы они стали поддающимися проверке. Как указано выше все требования должны быть поддающимися проверке.

Нефункциональные требования, которые являются неподдающимися проверке на программном уровне, все равно должны быть сохранены как документация намерений клиента; Такие требования к продукту могут быть преобразованы в требования к процессу. Например, нефункциональное требование, чтобы ПО не содержало «потайных ходов», может быть удовлетворено заменой на требование использовать парное программирование.

Необходимые принадлежности

1. Персональный компьютер с программным обеспечением.

2. Справочная литература

Работа в аудитории.

1. На основании выбранного программного продукта разработать:

* Требования функционального характера
* Требования нефункционального характера.

2.Составить письменно отчёт и продемонстрировать преподавателю.

Задание.

Разработать требования к программному продукту. Подробно расписать требования функционально и нефункционального характера.

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Требования к программному продукту;
6. Заключение

Контрольные вопросы.

1. Дать определение, что такое требования к программному обеспечению.
2. В каком виде могут выражаться требования?
3. На какой стадии используются совокупность требований?
4. Перечислите требования, носящие функциональный характер.
5. Перечислите требования, носящие нефункциональный характер.

Практическое занятие №7

«Проектирование программы с использованием операций языка»

Цель: научить проектировать программу с использованием стандартных операций языка программирования.

Пояснения.

Операции языка С# приведены в порядке убывания приоритетов. Операции с разными приоритетами разделены чертой.

|  |  |
| --- | --- |
| Операция | Описание  |
| . | Доступ к элементу |
| x() | Вызов метода или делегата |
| x[] | Доступ к элементу |
| x++ | Постфиксный инкремент |
| x-- | Постфиксный декремент |
| new | Выделение памяти |
| typeof | Получение типа |
| checked | Проверяемый код |
| unchecked | Непроверяемый код |
| + | Унарный плюс |
| - | Арифметическое отрицание |
| ! | Логическое отрицание |
| ~ | Поразрядное отрицание |
| ++x | Префиксный инкремент |
| --x | Префиксный декремент |
| (тип)x | Преобразование типа |
| \* | Умножение |
| / | Деление |
| % | Остаток от деления |
| << | Сдвигвлево |
| >> | Сдвиг вправо |
| < | Меньше |
| > | Больше |
| <= | Меньшеили равно |
| >= | Больше или равно |
| is | Проверка принадлежности типу |
| as | Приведение типа |
| == | Равно |
| != | Не равно |

|  |  |
| --- | --- |
| & | Поразрядное И |
| ^ | Поразрядное исключающее ИЛИ |
| | | Поразрядное ИЛИ |
| && | Логическое И |
| || | Логическое ИЛИ |
| ? : | Условная операция |
| = | Простое присваивание |
| \*= | Умножение с присваиванием |
| /= | Деление с присваиванием |
| %= | Остаток от деления с присваиванием |
| += | Сложение с присваиванием |
| -= | Вычитание с присваиванием |
| <<= | Сдвиг влево с присваиванием |
| >>= | Сдвиг вправо с присваиванием  |
| &= | Поразрядное И с присваиванием |
| ^= | Поразрядное исключающее ИЛИ с присваиванием |
| |= | Поразрядное ИЛИ с присваиванием |

Пример разработанной программы.

Периметр квадрата, площадь которого равна а;

using System;

namespace Example

{

class Program

 {

static void Main()

 {

Console.Write("s= ");

float s = float.Parse(Console.ReadLine());

double p = 4 \* Math.Sqrt(s);

Console.WriteLine("p=" + p);

 }

 }

}

Необходимые принадлежности

1. Персональный компьютер с программным обеспечением.

2. Справочная литература

Работа в аудитории.

1. Спроектировать программу согласно заданию.

2. Составить письменно отчёт и продемонстрировать преподавателю.

Задание.

Спроектировать программу для решения:

1. площадь прямоугольного треугольника по двум катетам a, b.
2. радиус окружности, длина которой равна l;
3. среднее арифметическое кубов двух данных чисел;

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Проект программы;
6. Заключение

Контрольные вопросы.

1. Допустимы ли следующие способы записи ++(++i), (i--)--, ++(i--) и т.д. И почему.

2. Допустимы ли следующие способы записи !(-i), -(!a). И почему.

3. Выясните, чему равен результат данного выражения: 1) i<j<k 2) true<false

 И объясните, как получился данный результат.

4. Объясните, какое значение примет переменная t в данном фрагменте программы:

int a=10, b=3;

bool t=(a>=b && a!=2\*b || a<0);

5. Объясните, какие значения примут переменные t и b после выполнения данного фрагмента программы:

int a=10, b=3;

int t=(a++)-b;

int b+=t\*a;

Практическое занятие №8

«Проектирование программы с использованием операторов языка»

Цель: научиться проектировать программы с использованием операторов языка программирования.

Пояснения.

Программа на языке С# состоит из последовательности операторов, каждый из которых определяет законченное описание некоторого действия и заканчивается точкой с запятой. Все операторы можно разделить на 4 группы: операторы следования, операторы ветвления, операторы цикла и операторы передачи управления.

Операторы следования выполняются компилятором в естественном порядке: начиная с первого до последнего. К операторам следования относятся: выражение и составной оператор.

Операторы ветвления позволяют изменить порядок выполнения операторов в программе. К операторам ветвления относятся условный оператор *if* и оператор выбора *switch.*

Операторы цикла используются для организации многократно повторяющихся вычислений. К операторам цикла относятся: цикл с предусловием while, цикл с постусловием dowhile, цикл с параметром for и цикл перебора foreach.

ВС# есть несколько операторов, изменяющих естественный порядок выполнения команд: оператор безусловного перехода *goto*, оператор выхода *break*, оператор перехода к следующей итерации цикла *continue*, оператор возврата из метода *return*и оператор генерации исключения *throw.*

Необходимые принадлежности

1. Персональный компьютер с программным обеспечением.

2. Справочная литература

Работа в аудитории.

1. Спроектировать программу согласно заданию.

2. Составить письменно отчёт и продемонстрировать преподавателю.

Задание.

Дана точка на плоскости с координатами (х, у). Составить программу, которая выдает одно из сообщений «Да», «Нет», «На границе» в зависимости от того, лежит ли точка внутри заштрихованной области, вне заштрихованной области или на ее границе.

|  |  |  |
| --- | --- | --- |
| 1.  | 2.  | 3.  |

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Проект программы;
6. Заключение

Контрольные вопросы.

1. Какие группы операторов существуют в языке программирования.

2. Приведите пример операторов следования и их применение.

3. Приведите пример операторов ветвления и их применение.

4. Приведите пример операторов цикла и их применение.

5. Приведите пример операторов передачи управления и их применение.

Практическое занятие №18

«Проектирование программы с использованием классов и методов»

Цель:научится проектировать приложения с использованием классов и методов.

Пояснения.

Класс - это шаблон, который определяет форму объекта. Класс должен определять только одну логическую сущность.

Ключевые слова C#, указывающие уровень доступности - классификаторы доступности :

|  |  |
| --- | --- |
| **Классификатор доступности C#** | **Описание** |
| public | Помечает метод, как доступный из объектной переменной, а также из всех производных классов |
| private | Помечает метод, как доступный только из класса, определяющего этот метод. В C# любой член по умолчанию определяется, как private |
| protected | Помечает метод, как доступный для определяющего класса, а так же для любого производного класса. Однако защищенные методы не доступны из объектной переменной |
| internal | Определяет метод, как доступный для любого типа только внутри данного компоновочного блока, но не снаружи |
| protectedinternal | Определяет метод, доступ к которому ограничивается рамками текущего компоновочного блока или типами, созданными из определяющего класса в данном компоновочном блоке |

Классификатор доступности необязателен и если он не указан, то по умолчанию подразумевается, что член закрыт (private) в рамках класса, где он определен.

Наследуемый класс не может иметь более открытый классификатор доступности, чем его предок.

Методы - это процедуры(подпрограммы), которые манипулируют данными, определенными в классе, и во многих случаях обеспечивают доступ к этим данным.

В качестве имени метода можно использовать любой допустимый идентификатор, отличный от тех, которые уже использованы для других элементов программы в пределах текущей видимости. В качестве имен методов нельзя использовать ключевые слова C#.

Элемент **доступ** означает классификатор доступа.

Элемент **тип\_возврата** указывает тип значения, возвращаемого методом. Это может быть любой допустимый тип, включая типы классов, создаваемые программистом. Если метод не возвращает значения, необходимо указать тип void.

Элемент **список\_параметров** представляет собой последовательность пар, состоящих из типа данных и идентификатора, разделенных запятыми. Параметры - это переменные, которые получают значения аргументов, передаваемых методу при вызове. Если метод не имеет параметров, **список\_параметров** остается пустым.

***Пример***.

using System;

namespace Hello

{

class Program

 {

static double min(double a, double b)

 {

return (a < b) ? a : b;

}

static void Main(string[] args)

 {

Console.Write("x=");

double x = double.Parse(Console.ReadLine());

Console.Write("y=");

double y = double.Parse(Console.ReadLine());

double z = min(3 \* x, 2 \* y) + min(x - y, x + y);

Console.WriteLine("z=" + z);

 }

 }

}

Необходимые принадлежности

1. Персональный компьютер с программным обеспечением.

2. Справочная литература

Работа в аудитории.

1. Спроектировать программу согласно заданию.

2. Составить письменно отчёт и продемонстрировать преподавателю.

Задание.

1. Разработать метод f(x), который вычисляет значение по следующей формуле: f(x)=x3-sinx. Определить, в какой из точек а или b, функция принимает наибольшее значение

2.Разработать метод f(x), который вычисляет значение по следующей формуле: f(x)=cos(2x)+sin(x-3). Определить, в какой из точек а или b, функция принимает наименьшее значение.

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Проект программы;
6. Заключение

Контрольные вопросы.

1. Приведите запись сигнатуры метода.

2. Дать определение понятию конструктор.

3. Дать определение понятие формального и фактического параметра. Объяснить их различия.

4. Какой оператор отвечает за возврат значения из метода

5.Какой оператор показывает, что возвращаемое значение отсутствует.

Практическое занятие №№19 - 20

«Проектирование программы для работы с массивами и строками»

Цель:научится проектировать программы с использованием массивов и строк.

Пояснения.

Массивы являются коллекциями объектов одного типа. Поскольку длина массивов практически не ограничена, они могут использоваться для хранения тысяч или даже миллионов объектов, но размер массива должен быть указан при его создании. Каждый элемент массива доступен по числовому индексу, указывающему позицию или ячейку, в которой объект хранится в массиве. Массивы могут хранить как ссылочные типы, так и типы значений.

Строка C# представляет собой группу одного или нескольких знаков, объявленных с помощью ключевого слова string, которое является ускоренным методом языка C# для класса System.String. В отличие от массивов знаков в C или C++, строки в C# гораздо проще в использовании и менее подвержены ошибкам программирования.

Строковые объекты являются неизменяемыми: после создания их нельзя изменить. Методы, работающие со строками, возвращают новые строковые объекты. Поэтому в целях повышения производительности большие объемы работы по объединению строк или другие операции следует выполнять в классе StringBuilder, как показано в далее в примерах кода.

***Пример***

using System;

usingSystem.Text;

namespaceConsoleApplication

{

 classClass

 {

 static void Main()

 {

 Console.WriteLine("Введитестроку: ");

 StringBuilder a = new StringBuilder(Console.ReadLine());

 Console.WriteLine("Исходнаястрока: "+a);

 Console.WriteLine("Введитесимвол x: ");

 char x=char.Parse(Console.ReadLine());

 Console.WriteLine("Введитесимвол y: ");

 char y=char.Parse(Console.ReadLine());

 for (inti=0; i<a.Length; ++i)

 if (a[i]==x){a.Insert(i+1,y); ++i;}

 Console.WriteLine("Измененная строка: "+a);

 }

 }

}

Необходимые принадлежности

1. Персональный компьютер с программным обеспечением.

2. Справочная литература

Работа в аудитории.

1. Спроектировать программу согласно заданию.

2. Составить письменно отчёт и продемонстрировать преподавателю.

Задание.

1. Разработать программу, которая для заданной строки s определяет, какой из двух заданных символов встречается чаще в строке.

2. Разработать программу, которая для заданной строки s определяет, сколько различных символов встречается в строке.

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Проект программы;
6. Заключение

Контрольные вопросы.

1. В какой кодировке хранится строка?

2. Какие методы отвечают за преобразование символов в верхний и нижний регистр.

3. За что отвечает экземплярный метод Split?

4. Приведите примеры способов инициализации строки.

5. В чём отличия класса String от StringBuilder?

Практическое занятие №28

«Использование средств манипуляции реляционной алгебры при работе с БД»

Цель: используя средства манипуляции реляционной алгебры добиться научиться эффективно, работать с БД.

Пояснения.

Объединением двух совместимых по объединению отношений R1 и R2 является отношение R3, содержащее множество всех кортежей, принадлежащих или R1, или R2, или обоим вместе. Другой вариант завершения этого определения «…, принадлежащих R1 и тех кортежей R2, которые не принадлежат R1». Последнее определение подчеркивает, что в результирующем отношении не должно быть совпадающих кортежей (дублей). Учитывая, что в отношении по определению не может быть одинаковых кортежей, далее мы не будем подчеркивать это требование к результирующему отношению. Естественно, что R3 также совместимо по объединению с R1 и R2.

Пересечением двух совместимых по объединению отношений R1 и R2 является отношение R3, содержащее кортежи, принадлежащие как R1, так и R2.

Разностью двух совместимых по объединению отношений R1 и R2 является отношение R3, кортежи которого принадлежат R1, но не принадлежат R2 (т.е. кортежи из R1, которых нет в R2).

Необходимые принадлежности

1. Персональный компьютер с программным обеспечением.

2. Справочная литература

Работа в аудитории.

1. Разработать БД согласно заданию.

2. Составить письменно отчёт и продемонстрировать преподавателю.

Задание.

1.Заполнить произвольными данными 10 строк таблицы. Выполнить объединение таблиц БД по правилам реляционной алгебры.

**ПРЕПОДАВАТЕЛЬ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Личный номер преподавателя** |  Ф.И.О.  преподавателя |  Дата рождения |  Пол |  Адрес |

**НАУЧНЫЙ РАБОТНИК**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Личный номер работника** | Ф.И.О. научного работника |  Дата рождения |  Пол |  Адрес |

2. Заполнить произвольными данными 10 строк таблицы.Выполнить пересечение таблиц БД по правилам реляционной алгебры.

**СТУДЕНТ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ф.И.О. студента | **Серия****и номер паспорта** | Пол |  Дата  рождения | Семейное положение |

**ЖИЛЕЦ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ф.И.О. жильца | **Серия****и номер паспорта** | Пол | Датарождения | Семейное положение |

3. Выполнить операцию разности чтобы в результате вычитания отношения СТУДЕНТ из отношения ЖИЛЕЦ получили отношение НЕСТУДЕНТ, содержащее сведения о всех проживающих в общежитии, но не являющихся студентами.

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Проект БД;
6. Заключение

Контрольные вопросы.

1. Дать определение понятию кортеж.

2. Дать определение понятию объединение отношений.

3. Дать определение понятию пересечение отношений.

4.Дать определение понятию разность отношений.

Практическое занятие №29

«Использование реляционного исчисления при работе с БД»

Цель: научиться использовать средства реляционного исчисления при работе с БД.

Пояснения.

Декартовым произведением отношения А со схемой и отношения В со схемой является отношение С, со схемой равной объединению схем отношений А и В, кортежи которого получены путем конкатенации (присоединения) каждого кортежа отношения В с каждым кортежем отношения А.

Операция деления одного отношения (делимого) на другое отношение (делитель) может быть выполнена, если все множество атрибутов схемы делителя является подмножеством атрибутов схемы делимого.

Результирующее отношение содержит только те атрибуты делимого, которых нет в делителе.В него включаются только те кортежи, декартово произведение которых с делителем содержится в делимом (является подмножеством делимого).

Выбор (селекция) – операция получения нового отношения с той же схемой, что и исходное отношение, но, кортежи которого являются подмножеством кортежей исходного. В него включаются только те кортежи исходного отношения, значения определенных атрибутов которых удовлетворяют заданным ограничениям.

Ограничения могут быть заданы в виде логического выражения, элементами которого являются простейшие условия типа «Имя атрибута – знак сравнения – значение атрибута» (в общем случаев «Выражение – знак сравнения – выражение»), которые могут соединяться булевыми операторами И (AND), ИЛИ (OR), иметь знак отрицания (NOT), а также использовать круглые скобки для изменения старшинства выполнения булевых операций по правилам математической логики.

Необходимые принадлежности

1. Персональный компьютер с программным обеспечением.

2. Справочная литература

Работа в аудитории.

1. Разработать БД согласно заданию.

2. Составить письменно отчёт и продемонстрировать преподавателю.

Задание.

1. Пусть имеем отношение **КС,** содержащее список участников команды факультета **Компьютерные сети** по шахматам и отношение **ИБ,** содержащее аналогичный список команды факультета **Информационная безопасность**. Вычислить декартово произведение **КС × ИБ** имеющее название **ВСТРЕЧИ,** содержащее список пар участников, которые должны играть друг с другом. Полученный результат письменно внести в отчёт.

**КС ИБ**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  Ф.И.О.  игрока  КС |  Разряд |  Дата  рождения |  |  Ф.И.О.  игрока  ИБ |  Разряд |  Дата  рождения |

2. Выполнить операцию деления согласно представленным таблицам. Полученный результат письменно внести в отчёт.

ДЕЛИМОЕ ДЕЛИТЕЛЬ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  **Ф.И.О.** | **Иност-ый язык** |  **Степень** **владения языком** | **.****.** | **Иностранный язык** | **Степень** **владения** **языком** |
| Иванов |  английский |  свободно |  |  английский |  со словарем |
| Сидоров |  английский |  свободно |  |  турецкий |  свободно |
| Сидоров |  японский | разговорный |  |  |  |
| Сидоров |  турецкий |  со словарем |  |  |  |
| Кузнецова |  английский |  со словарем |  |  |  |
| Кузнецова |  турецкий |  свободно |  |  |  |
| Иванова |  немецкий | разговорный |  |  |  |
| Иванова |  турецкий |  свободно |  |  |  |
| Иванова |  английский | со словарем |  |  |  |

3. Требуется отобрать кортежи, относящиеся к первому году учебы, причем для студентов, не получающих стипендию, но имеющих достаточно высокий рейтинг – от 800 до 900 баллов. Полученный результат письменно внести в отчёт.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  Код  студента | Ф.И.О. студента |  Номер  семестра |  Тип стипендии  в семестре |  Рейтинг  за семестр |

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Проект БД;
6. Заключение

Контрольные вопросы.

1. Разъяснить суть операции декартово произведение при работе с БД.

2. Разъяснить суть операции деление при работе с БД.

3. Разъяснить суть операции селекции при работе с БД.

4. Разъяснить суть операции ограничения при работе с БД.

Практическое занятие №30

«Проектирование концептуальной модели базы данных»

Цель:научиться проектировать концептуальную модель БД.

Пояснения.

Концептуальная (содержательная) модель — это абстрактная модель, определяющая структуру моделируемой системы, свойства её элементов и причинно-следственные связи, присущие системе и существенные для достижения цели моделирования.

Рассмотрим проектирование пошагово концептуальной модели.

1. **Выделить основные абстракции (сущность, атрибут, связь) в предметной области и определить их параметры.**

 Определим следующие сущности: СТУДЕНТ, ЭКЗАМЕН, ОЦЕНКА

Определим атрибуты сущностей. Пусть для упрощения сущность СТУДЕНТ характеризуется только фамилией. Фамилию мы и возьмем в качестве атрибута. Так как фамилия может неоднозначно идентифицировать объект, введем дополнительный атрибут Код студента, уникальный для каждого студента. Таким образом, сущность СТУДЕНТ характеризуется двумя атрибутами код студента, фамилия.

Аналогично определим сущность ЭКЗАМЕН с атрибутами код экзамена, предмет, дата экзамена и сущность ОЦЕНКА с атрибутом значение оценки (оценка). Между этими сущностями существуют следующие связи: студент сдавал экзамен, студент получил оценку, по экзамену получены следующие оценки.

 2.**Сформировать максимально полный перечень возможных запросов к базе данных на основе анализа предметной области.**

 По смыслу задачи к базе данных возможны следующие запросы:

* Какие оценки получил студент с заданной фамилией (кодом);
* Какие студенты получили заданное значение оценки;
* Какие экзамены сдал студент с заданной фамилией (кодом);
* Какую оценку по конкретному предмету получил студент с заданной фамилией (кодом).

В данном примере остановимся на этих запросах.

 **3.Построить концептуальную модель.**



По этой диаграмме можно ответить на все вопросы, кроме последнего. Для реализации и последнего запроса в перспективе введем новую агрегированную сущность. Определим эту сущность как ЭКЗАМЕНАЦИОННАЯ ВЕДОМОСТЬ с атрибутами код студента, фамилия, код экзамена, предмет, дата экзамена, оценка.

**4. Описать домены (допустимые множества значений, которые могут принимать атрибуты), указывая типы соответствующих данных и их характеристики.**

*Код студента* принимает значения из множества целых чисел.

*Фамилия* принимает символьное значение, максимальная длина 20 символов.

*Код экзамена* принимает значения из множества целых чисел.

*Предмет* принимает символьное значение, максимальная длина 20 символов.

*Дата экзамена* принимает значение дата в формате 00.00.00.

*Оценка* принимает целое значение от 2 до 5.

Необходимые принадлежности

1. Персональный компьютер с программным обеспечением.

2. Справочная литература

Работа в аудитории.

1. Разработать БД согласно заданию.

2. Составить письменно отчёт и продемонстрировать преподавателю.

Задание.

Спроектировать концептуальную модель базы данных страховой компании.

*Описание предметной области*

Вы работаете в страховой компании. Вашей задачей является отслеживание ее финансовой деятельности.Компания имеет различные филиалы по всей стране. Каждый филиал характеризуется названием, адресом и телефоном. Деятельность компании организована следующимобразом: к вам обращаются различные лица с целью заключения договора о страховании.В зависимости от принимаемых на страхование объектов и страхуемых рисков договорзаключается по определенному виду страхования (например, страхование автотранспортаот угона, страхование домашнего имущества, добровольное медицинское страхование).При заключении договора вы фиксируете дату заключения, страховую сумму, вид страхования, тарифную ставку и филиал, в котором заключался договор.

*Возможный набор сущностей*

Договоры (Номер договора, Дата заключения, Страховая сумма, Тарифная ставка,

Код филиала, Код вида страхования).

Вид страхования (Код вида страхования, Наименование).

Филиал (Код филиала, Наименование филиала, Адрес, Телефон).

Контрольные вопросы.

1.Дать определение понятию концептуальная модель.

2.Какие действия включает первый шаг построения концептуальной модели?

3.Какие действия включает второй шаг построения концептуальной модели?

4.Какие действия включает третий шаг построения концептуальной модели?

5.Какие действия включает четвёртый шаг построения концептуальной модели?

Практическое занятие №31

«Проектирование реляционной модели базы данных»

Цель: научиться проектировать реляционную модель баз данных.

Пояснения.

Реляционная база данных — база данных, основанная на реляционной модели данных. Слово «реляционный» происходит от англ. relation (отношение). Для работы с реляционными БД применяют реляционные СУБД.

Рассмотрим построение реляционной базы данных на примере.

Табличная база данных "Комплектующие компьютера и поставщики" содержит информацию о различных комплектующих и имеет поля:"Счетчик", "Наименование", "Описание", "Название фирмы", "Адрес", "Цена" (в рублях) - табл. 1.

|  |
| --- |
| Таблица 1. Комплектующие компьютера и поставщики |
|

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Счетчик | Наименование | Описание | Название фирмы | Адрес | Цена |
| 1 | Системный блок | Pentium | Фирма 1 | Адрес 1 | 10000 |
| 2 | Системный блок | Pentium | Фирма 2 | Адрес 2 | 9000 |
| 3 | Монитор | 15" | Фирма 1 | Адрес 1 | 5000 |
| 4 | Монитор | 15" | Фирма 2 | Адрес 2 | 6000 |
| 5 | Клавиатура | 104 кл. | Фирма 1 | Адрес 1 | 250 |
| 6 | Клавиатура | 104 кл. | Фирма 2 | Адрес 2 | 300 |
| 7 | Мышь | Зкн | Фирма 1 | Адрес 1 | 100 |
| 8 | Мышь | 3 кн | Фирма 2 | Адрес 2 | 150 |

 |

Мы видим, что почти половину объема таблицы составляет избыточная, дублированная информация.

Проанализируем причину дублирования. Комплектующие компьютера имеют два неотъемлемых свойства: "Наименование" и "Описание". "Название фирмы", "Адрес" и "Цена" не являются свойствами комплектующих компьютера, они являются свойствами поставщика.

Естественно разделить исходную таблицу на две: "Комплектующие" (табл. 2) и "Поставщики" (табл. 3).

Каждая таблица должна содержать, по крайней мере, одно *ключевое поле*, содержимое которого уникально для каждой записи в этой таблице. В таблицу "Комплектующие" введем поле "Код комплектующих". Именно это поле будет ключевым в данной таблице.

|  |
| --- |
| Таблица 2. Комплектующие |
|

|  |  |  |
| --- | --- | --- |
| Код комплектующих | Наименование | Описание |
| К1 | Системный блок | Pentium |
| К2 | Монитор | 15" |
| КЗ | Клавиатура | 104 кл. |
| К4 | Мышь | Зкн. |

 |

В таблицу "Поставщики" введем дополнительное поле "Код поставщика". Именно это поле будет ключевым в данной таблице.

|  |
| --- |
| Таблица 3. Поставщики |
|

|  |  |  |
| --- | --- | --- |
| Код поставщика | Название фирмы | Адрес |
| П1 | Фирма1 | Адрес 1 |
| П2 | Фирма2 | Адрес2 |

 |

**Связывание таблиц**

После создания различных таблиц, содержащих данные, относящиеся к различным аспектам базы данных, необходимо обеспечить целостность базы данных. Для этого надо *связать* таблицы между собой.

При связи "один-ко-многим" каждой записи в одной (главной) таблице могут соответствовать несколько записей в другой (подчиненной) таблице, а запись в подчиненной таблице не может иметь более одной соответствующей ей записи в главной таблице.

Если одной записи в первой таблице могут соответствовать несколько записей во второй таблице и, наоборот, одной записи во второй таблице - несколько записей в первой таблице, то реализуется связь "многие-ко-многим".

В нашем случае реализуется именно такая связь. Одной записи в таблице "Комплектующие" соответствуют две записи в таблице "Поставщики", так как устройства одного типа продаются двумя фирмами. Одной же записи таблицы "Поставщики" соответствуют четыре записи таблицы "Комплектующие", так как одна фирма продает устройства четырех типов.

Две таблицы, находящиеся в отношении "многие-ко-многим", могут быть связаны только с помощью третьей (связующей) таблицы. Таблицы "Комплектующие" и "Поставщики" можно связать в отношении "многие-ко-многим" путем создания двух связей "один-ко-многим" по отношению к таблице "Цена".

Таблицы "Комплектующие" и "Поставщики" будут являться главными по отношению к таблице "Цена".

Связь между таблицами устанавливает отношения между совпадающими значениями в полях с одинаковыми именами. С ключевым полем главной таблицы (*первичный ключ*) связывается одноименное поле подчиненной таблицы (*внешний ключ*).

В главной таблице "Комплектующие" поле "Код комплектующих" является первичным ключом, соответственно в подчиненной таблице "Цена" должно существовать одноименное поле, которое является внешним ключом.

Таблица "Поставщики" также является главной по отношению к таблице "Цена". Ее поле "Код поставщика" является первичным ключом, соответственно в подчиненной таблице "Цена" должно существовать одноименное поле, которое является внешним ключом.

Таким образом, таблица "Цена" должна содержать следующие поля (табл. 4):

* "Счетчик" (ключевое поле);
* "Код комплектующих" (поле внешнего ключа для таблицы "Комплектующие");
* "Код поставщика" (поле внешнего ключа для таблицы "Поставщики");
* "Цена" (числовое поле).

|  |
| --- |
| Таблица 4. Цена |
|

|  |  |  |  |
| --- | --- | --- | --- |
| Код комплектующих | Код комплектующих | Код поставщика | Цена |
| 1 | К1 | П1 | 9000 |
| 2 | К1 | П2 | 10000 |
| 3 | К2 | П1 | 5000 |
| 4 | К2 | П2 | 6000 |
| 5 | КЗ | П1 | 250 |
| 6 | КЗ | П2 | 300 |
| 7 | К4 | П1 | 100 |
| 8 | К4 | П2 | 150 |

 |

Межтабличная связь *обеспечивает целостность данных*. Связанные таблицы представляют собой единую базу данных, в которой можно создавать новые таблицы, а также запросы и отчеты, содержащие данные из связанных таблиц.

*Базы данных, состоящие из связанных двумерных таблиц, принято называть* ***реляционными****.*

Прежде чем приступить к созданию реляционной базы данных, необходимо продумать ее *проект*. Проект представляет собой модель будущей БД, состоящей из объектов и их связей, необходимых для выполнения поставленных задач.

Процесс проектирования включает, прежде всего, определение перечня необходимых таблиц и задание их структуры, а также установление типа связей между этими таблицами.

Необходимые принадлежности

1. Персональный компьютер с программным обеспечением.

2. Справочная литература

Работа в аудитории.

1. Разработать БД согласно заданию.

2. Составить письменно отчёт и продемонстрировать преподавателю.

Задание.

Спроектрировать проект реляционной базы данных "Коллекция аудиозаписей", которая бы содержала главную таблицу "Список аудио-CD" и подчиненную таблицу "Содержание аудио-CD".

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Проект БД;
6. Заключение

Контрольные вопросы.

1. Дать определение понятию реляционная модель данных.

2. Почему в некоторых случаях целесообразно использовать многотабличные, а не однотабличные базы данных?

2. Какие типы связей между таблицами возможны в реляционных базах данных?

4.Дать определение понятию ключевое поле.

Практическое занятие №33

«Построение баз данных с использованием СУБД MSAccess»

Цель: научится разрабатывать базы данных с помощью системы управления базами данных MS Access.

Пояснения.

MicrosoftOfficeAccessилипростоMicrosoftAccess — реляционнаяСУБДкорпорацииMicrosoft.Имеет широкий спектр функций, включая связанные запросы, связь с внешними таблицами и базами данных. Благодаря встроенному языку VBA, в самомAccess можно писать приложения, работающие с базами данных.

Основные компоненты MS Access:

* построитель таблиц;
* построитель экранных форм;
* построитель SQL-запросов (язык SQL в MS Access не соответствует стандарту ANSI);
* построитель отчётов, выводимых на печать.

Они могут вызывать скрипты на языке VBA, поэтому MS Access позволяет разрабатывать приложения и БД практически «с нуля» или написать оболочку для внешней БД.

MicrosoftJetDatabaseEngine (англ.), которая используется в качестве движка базы данных MS Access является файл-серверной СУБД и потому применима лишь к приложениям, работающим с небольшими объемами данных и при небольшом числе пользователей, одновременно работающих с этим данными. Непосредственно в Access отсутствует ряд механизмов, необходимых в многопользовательских БД, таких, например, как триггеры.

Access, при работе с базой данных, иначе взаимодействует с жёстким (или гибким) диском, нежели другие программы.

В других программах, файл-документ, при открытии, полностью загружается в оперативную память, и новая редакция этого файла (изменённый файл) целиком записывается на диск только при нажатии кнопки «сохранить».

В Access новая редакция содержимого изменённой ячейки таблицы записывается на диск (сохраняется) сразу, как только курсор клавиатуры будет помещён в другую ячейку (или новая редакция изменённой записи записывается на диск сразу, как только курсор клавиатуры будет поставлен в другую запись (в другую строку)?). Таким образом, если внезапно отключат электричество, то пропадёт только изменение той записи, которую не успели покинуть.

Целостность данных в Access обеспечивается также за счет механизма транзакций.

Кнопка «Сохранить» в Access тоже есть, но в Access в режиме просмотра данных она нужна, в первую очередь, для сохранения изменённого режима показа таблицы или другого объекта — то есть, для сохранения таких изменений, как:

* изменение ширины столбцов и высоты строк,
* перестановка столбцов в режиме просмотра данных, «закрепление» столбцов и освобождение закреплённых столбцов,
* изменение сортировки,
* применение нового фильтра,
* изменение шрифта; цвета текста, сетки и фона
* и т. п.

Кроме того, в Access эта кнопка нужна в режиме «Конструктор» для сохранения изменений структуры объекта базы данных, сделанных в этом режиме.

Необходимые принадлежности

1. Персональный компьютер с программным обеспечением.

2. Справочная литература

Работа в аудитории.

1. Разработать БД согласно заданию.

2. Составить письменно отчёт и продемонстрировать преподавателю.

Задание.

1. Откройте **Access**, создайте **новую** базу данных, сохраните в своей папке с именем **Student.** Изучите **окно базы данных**, найдите вкладки объектов, ознакомьтесь с формой представления данных и панелью инструментов.
2. **В режиме конструктора** создайте макет таблицы, содержащей следующие поля:

Таблица 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Код студента | Фамилия | Имя | Группа | Адрес | Телефон | Фото | Примечания |
|  |  |  |  |  |  |  |  |

Имена полей задайте такие же, как в таблице 1.

Свойства полей:

**Код студента:** *тип* ***счетчик****, подпись поля - №, индексированное, без повторений.*

**Фамилия:** *тип* ***текстовый***, *размер поля 50, обязательное, без пустых строк, индексированное.*

**Имя:** *тип* ***текстовый***, *размер поля 25, обязательное, без пустых строк.*

**Группа:** *тип****текстовый****, размер поля 7, подпись № Группы, маска ввода КОМ-999, обязательное, без пустых строк, индексированное.*

**Адрес:** *тип* ***текстовый***, *размер поля 200, не обязательное, не индексированное.*

**Телефон:** *тип* ***текстовый***, *размер поля 8, маска ввода 99-99-99, не обязательное, не индексированное.*

**Фото:** *тип****OLE – объект.***

**Примечания:** *тип* ***memo.***

Закройте таблицу, присвойте имя **Студенты**.

1. Для ввода данных в таблицу создайте **автоформув столбец** с именем **Формуляр**. Она отображает все поля таблицы. Обратите внимание, как используются подписи полей. Откройте форму и введите 10-12 записей, проверяя правильность описания полей. В данных должны быть повторения фамилий, номеров групп (используйте три номера группы). Для ввода фото используйте вставку объекта из **ClipGallery.** Если необходимо сделать исправления, откройте **таблицу в режиме конструктора** и внесите необходимые изменения в макет таблицы.
2. Войдите в **режим конструктора форм**. Отредактируйте формуляр: текстовые поля расположены в столбце слева один под другим, фотографию и примечания поместите справа один под другим. Уменьшите размер полей фотографии и примечаний, а также измените в макете формы свойство рамки объекта «Установка размеров» на значение *«По размеру рамки»*. Окно свойства вызывается нажатием правой кнопки мыши на активном элементе формы.
3. Для просмотра записей таблицы создайте с использованием **мастера форм** ленточную форму, не включая в нее фотографии и примечания. Задайте имя **Ленточная форма.**
4. Выполните в этой форме операции, характерные для баз данных.

А). **Сортировка записей** выполняется по значению одного поля, активного в данный момент.

Выполните сортировку записей:

* по фамилиям,
* по номерам групп,
* по номерам телефонов.

Б). **Отбор записей** выполняется с использованием **фильтров**. Чтобы задать условия отбора, используется кнопка . Чтобы выполнить отбор или отменить условия отбора, используется кнопка . В бланке отбора можно задать условия, выполняемые одновременно (задаются в одной строке бланка) или условия, выполняемые порознь (задаются через закладку «**Или**»). Условия отбора могут быть выбраны из списка или записаны в виде отношений или логических выражений.

Выполните последовательно отбор:

* Всех студентов, учащихся в одной группе (например, 364).
* Всех студентов, учащихся в двух каких-нибудь группах (например, 364**Или** 365).
* Студента, например, Петрова, учащегося в указанной группе.
* Всех студентов, фамилии которых начинаются на букву «А» или «Б».Используется маска \*,например А\* **Или** Б\*.

В). Пункт меню **Записи – Расширенный фильтр** позволяет выполнить отбор записей с использованием **бланка фильтра** (запроса). В бланке фильтра будет добавлена таблица Студенты. Для фильтра можно задать поля, по которым выполняется отбор, сортировки по значениям полей, и условия отбора. Для отбора данных поле таблицы двойным щелчком переносится в нижнюю часть бланка. В строке «Сортировка» выбирается способ сортировки по указанному полю. В строке «Условие отбора» накладывается произвольное условие на значения поля. Если условия записаны в одной строке бланка, они накладываются друг на друга, то есть должны быть выполнены одновременно. Для записи условий, выполняемых врозь, используется строка бланка «Или». Для записи условий используется построитель выражений, вызываемый правой кнопкой мыши в области условия . Вызовите построитель выражений и ознакомьтесь с его окном. Для сравнения текстовых строк используется операция **Like**, например, LikeПетров, или Like П\*.

Выполните последовательно отбор данных, как в пункте Б). Каждый результат отбора сохраните (меню **Файл - Сохранить как**) в **форме отчета** с именами, соответственно, **Фильтр1 - Фильтр4.**

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Проект БД;
6. Заключение

Контрольные вопросы.

1. Дать определение понятию система управления базами данных (СУБД)?

2. Перечислите состав программы MicrosoftAccess?

3. Какие языки программирования используются в MicrosoftAccess?

4. Как происходит сохранение БД в MicrosoftAccess?

Практическое занятие №34

«Построение баз данных с использованием СУБД MS SQL-Server»

Цель: научится разрабатывать базы данных с помощью системы управления базами данных MS SQL Server.

Пояснения.

Microsoft SQL Server — система управления реляционными базами данных (СУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Сервер баз данных Microsoft SQL Server в качестве языка запросов использует версию языка SQL, получившую название Transact-SQL (сокращённо T-SQL). Язык T-SQL является реализацией SQL-92 (стандарт ISO для языка SQL) с множественными расширениями. T-SQL позволяет использовать дополнительный синтаксис для хранимых процедур и обеспечивает поддержку транзакций (взаимодействие базы данных с управляющим приложением).

При взаимодействии с сетью Microsoft SQL Server и Sybase ASE используют протокол уровня приложения под названием TabularDataStream (TDS, протокол передачи табличных данных). Протокол TDS также был реализован в проекте FreeTDS с целью обеспечить различным приложениям возможность взаимодействия с базами данных Microsoft SQL Server и Sybase.

Для обеспечения доступа к данным Microsoft SQL Server поддерживает OpenDatabaseConnectivity (ODBC) — интерфейс взаимодействия приложений с СУБД. Версия SQL Server 2005 обеспечивает возможность подключения пользователей через веб-сервисы, использующие протокол SOAP. Это позволяет клиентским программам, не предназначенным для Windows, кроссплатформенно соединяться с SQL Server. Компания Microsoft также выпустила сертифицированный драйвер JDBC, позволяющий приложениям под управлением Java (таким как BEA и IBM WebSphere) соединяться с Microsoft SQL Server 2000 и 2005.

Также SQL Server поддерживает зеркалирование и кластеризацию баз данных. Кластер сервера SQL — это совокупность одинаково конфигурированных серверов; такая схема помогает распределить рабочую нагрузку между несколькими серверами. Все сервера имеют одно виртуальное имя, и данные распределяются по IP-адресам машин кластера в течение рабочего цикла. Также в случае отказа или сбоя на одном из серверов кластера доступен автоматический перенос нагрузки на другой сервер.

SQL Server поддерживает избыточное дублирование данных по трем сценариям:

 Снимок: Производится «снимок» базы данных, который сервер отправляет получателям.

 История изменений: Все изменения базы данных непрерывно передаются пользователям.

 Синхронизация с другими серверами: Базы данных нескольких серверов синхронизируются между собой. Изменения всех баз данных происходят независимо друг от друга на каждом сервере, а при синхронизации происходит сверка данных. Данный тип дублирования предусматривает возможность разрешения противоречий между БД.

В SQL Server 2005 встроена поддержка .NET Framework. Благодаря этому хранимые процедуры БД могут быть написаны на любом языке платформы .NET, используя полный набор библиотек, доступных для .NET Framework, включая CommonTypeSystem (система обращения с типами данных в Microsoft .NET Framework). Однако, в отличие от других процессов, .NET Framework, будучи базисной системой для SQL Server 2005, выделяет дополнительную память и выстраивает средства управления SQL Server вместо того, чтобы использовать встроенные средства Windows. Это повышает производительность в сравнении с общими алгоритмами Windows, так как алгоритмы распределения ресурсов специально настроены для использования в структурах SQL Server.

Для построения базы данных с использованием MS SQL Server необходимо использовать команду и задать ряд параметров :

• PRIMARY — файл определяется как первичное устройство.
• NAME — логическое имя; по умолчанию совпадает с именем файла.
• FILENAME — полное имя файла на диске.
• SIZE — исходный размер файла. Минимальный размер файла журнала равен 512 Кбайт.
• MAXSIZE — максимальный размер файла.
• UNLIMITED — размер файла не ограничивается.
• FILEGROWTH — приращение размера в мегабайтах (MB), килобайтах (KB) или процентах (%). По умолчанию приращение равно 10%.
• FOR LOAD — обеспечивает обратную совместимость со сценариями SQL, написанными для предыдущих версий SQL Server.
• FOR ATTACH — указывает, что файлы базы данных уже существуют.

Рассмотрим построение БД в MS SQL на примере.

Построить базу данных для бухгалтерской программы.

Первый способ. Создание БД по умолчанию.

CREATE DATABASE test1

/\* Данные — 2 Мбайт, файл журнала — по умолчанию \*/

Второй способ. Создание БД с характеристиками: Данные — 2 Мбайт, файл журнала — по умолчанию.

CREATE DATABASE test2

ON (FILENAME = 'c:\d1.mdf', SIZE = 2, NAME = 'd1')

Третий способ. Создание БД с характеристиками: Первичный файл — 10 Мбайт, одна группа файлов g1 и журнал размером 10 Мбайт.

CREATE DATABASE test3

ON PRIMARY (FILENAME ='c:\test3.mdf',

SIZE = 10 , NAME = 'd1'),

FILEGROUP g1 (FILENAME — 'c:\g1.mdf',

SIZE = 10 , NAME = 'g1')

LOG ON (FILENAME ='c:\test3.ldf',

SIZE = 10, NAME = 'log1')

Необходимые принадлежности

1. Персональный компьютер с программным обеспечением.

2. Справочная литература

Работа в аудитории.

1. Разработать БД согласно заданию.

2. Составить письменно отчёт и продемонстрировать преподавателю.

Задание.

Построить БД для работы складской программы подбирая оптимальные характеристики.

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Проект БД;
6. Заключение

Контрольные вопросы.

1. Какой способ построения БД по Вашему мнению самый быстрый и почему?

2. Какой способ построения БД по Вашему мнению самый эффективный и почему?

3. Какой способ построения БД по Вашему мнению самый самый оптимальный и почему?

Практическое занятие №37

«Использование принципов нормализации при проектировании базы данных»

Цель: научиться проектировать БД с использованием принципов нормализации.

Пояснения.

Проектирование БД начинается с ответов на вопросы:

1. Какие товары продает фирма, какова их цена, описание, и т.д.
2. По выбранным товарам оформляются заказы. Кто (покупатели), когда и совершили заказ и какова общая сумма заказа?

Таким образом, схему организации фирмы можно представить в виде двух отношений:


Начальная схема отношений.

Далее определяется схема отношений:

ТОВАРЫ-ЗАКАЗЫ

ТОВАРЫ {товарНом, товарНазв, товарОпис, товарРис, товарЦена, колНаСкладе, типНом, типНазв}

ЗАКАЗЫ {заказНом, заказДата, ПокупНом, ФИОПокуп, ЭлпочтаПокуп, адресПокуп, товарНом, количТовЗаказа, видДост, ценаДост, налогНДС, общСумма, заказСост}

В атрибутах «типНом» и «типНазв» содержится информация о типах товаров, например, название типа — Процессоры (типНом = 1), к которому относятся конкретные товары - Intel 386, Intel 486, AMD, и т.д. Атрибут «колНаСкладе» содержит общее количество товаров с данным номером на складе.

В одном заказе может заказываться несколько товаров с разными номерами. Атрибут «количТовЗаказа» содержит количество товаров данного типа, заказанных в данном заказе.

В атрибутах «ПокупНом», «ФИОПокуп», «ЭлпочтаПокуп», «адресПокуп» содержится информация о покупателе — номер, ФИО, Электронная почта и адрес.

В атрибутах «видДост», «ценаДост» хранится информация о доставке — номер, вид и цена доставки. Атрибут «заказСост» содержит состояние заказа, который может быть иметь значение «Сделка совершена» либо «Сделка выполнена неполно».

Для вычисления общей суммы заказа применяем формулу

общСумма = ТоварЦена\*количТов + налогНДС + ценаДост.

На 1-м шаге проектирования форма 1NF приводится к виду 2NF.

На 2-м шаге проектирования форма 2NF приводится к виду 3NF.

Необходимые принадлежности

1. Персональный компьютер с программным обеспечением.

2. Справочная литература

Работа в аудитории.

1. Разработать БД согласно заданию.

2. Составить письменно отчёт и продемонстрировать преподавателю.

Задание.

Спроектировать БД с использованием принципов нормализации для работы предприятия (фирмы), занимающейся продажей компьютерной техники используя:

* первую нормальную форму (1 NF);
* вторую нормальную форму (2 NF);
* третью нормальную форму (3 NF);

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Проект БД;
6. Заключение

Контрольные вопросы.

1. Напишите последовательность нормальных форм.

2. Напишите основные свойства нормальных форм.

3. Укажите,когда можно применять нормальную форму Бойса-Кодда.

Практическое занятие №38

«Использование семантических моделей при проектировании базы данных»

Цель: научиться применять семантическое моделирование при проектировании БД.

Пояснения.

Потребности проектировщиков баз данных в более удобных и мощных средствах моделирования предметной области вызвали к жизни направление семантических моделей данных. При том, что любая развитая семантическая модель данных, как и реляционная модель, включает структурную, манипуляционную и целостную части, главным назначением семантических моделей является обеспечение возможности выражения семантики данных.

Прежде, чем мы коротко рассмотрим особенности одной из распространенных семантических моделей, остановимся на их возможных применениях.

Наиболее часто на практике семантическое моделирование используется на первой стадии проектирования базы данных. При этом в терминах семантической модели производится концептуальная схема базы данных, которая затем вручную преобразуется к реляционной (или какой-либо другой) схеме. Этот процесс выполняется под управлением методик, в которых достаточно четко оговорены все этапы такого преобразования:

1.Построение ER-диаграммы.

1.1 Выберите из описания предметной области все существительные. Продумайте, какие из них будут соответствовать сущностям, а какие атрибутам сущностей. Зарисуйте в отчет все сущности с их атрибутами согласно обозначениям, принятым в ER-диаграммах.

1.2 На рисунке подчеркиванием атрибутов обозначьте для каждой сущности уникальный идентификатор (Ключ). При необходимости добавьте сущностям атрибуты, которые помогут однозначно отличить каждый экземпляр сущности.

1.3 Определите и включите в схему связи сущностей. Подпишите названия связей и пронумеруйте связи. Для первой связи укажите тип и модальность. Для всех связей запишите их прочтение слева направо и справа налево.

1.4 Если в схеме присутствуют связи типа «много-со-многими» уберите их путем ввода дополнительной сущности. Измененную схему зарисуйте в отчет.

Необходимые принадлежности

1. Персональный компьютер с программным обеспечением.

2. Справочная литература

Работа в аудитории.

1. Разработать БД согласно заданию.

2. Составить письменно отчёт и продемонстрировать преподавателю.

Задание.

По описанию предметной области построить семантическую модель БД методом ER-диаграмм, на основании которой построить набор таблиц БД.

Описание предметной области (Грузоперевозки).

АТП имеет грузовые автомобили с гос. номерами и организует перевозки для своих заказчиков. Стоимость перевозки зависит от расстояния и грузоподъемности автомобиля, который ее выполняет. Каждый заказчик может сделать заказ нескольких перевозок. Одну перевозку выполняет один грузовик.

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Проект БД;
6. Заключение

Контрольные вопросы.

1. Каковы задачи, решаемые на этапе семантического проектирования?

2. Что включает в себя семантическая модель?

3. В чём состоит идея модели «Сущность - связь»?

Практическое занятие №42

«Использование SQL запросов при формировании выборки БД»

Цель: научится использовать язык запросов SQL для формирования выборки данных из БД.

Пояснения.

Назначение SQL команды SELECT – это выборка и отображение данных из одной или нескольких таблиц базы данных.

Синтаксис команды SELECT:

SELECT [DISTINCT | ALL] {\* | [*выражение\_столбца* [AS *новое\_имя*]] [, …]} FROM *имя\_таблицы* [*псевдоним*] [, …] [WHERE *условие*] [GROUP BY *список\_столбцов*] [HAVING *условие*] [ORDER BY *список\_столбцов*]

Где: *Выражение\_столбца* – это имя столбца или выражение из нескольких имен *Имя\_таблицы*– это имя таблицы или представления, из которой нужно

выбрать данные *Псевдоним* – это сокращенное имя таблицы DISTINCT – ключевое слово, для исключения повторяющихся строк из результата (по умолчанию подразумевается ALL).

Порядок конструкций в операторе SELECT не может быть изменен. Обязательными являются конструкции SELECT и FROM. Остальные конструкции могут быть опущены.

***Порядок обработки элементов оператора SELECT:***

1. FROM -Определяются имена используемой таблицы или нескольких таблиц.
2. WHERE – накладывается условие отбора данных.
3. GROUP BY – образуются группы строк, имеющие одинаковые значения в указанном столбце.
4. HAVING – накладывается условие на отбор сгруппированных строк.
5. SELECT – определяются столбцы, которые нужно отобразить в результате.
6. ORDER BY – отобранные данные сортируются по указанным столбцам.

***Конструкция WHERE***

Конструкция WHERE предназначена для ограничения попадания строк в выходные данные, по результату проверки на соответствие определенному условию.

Существуют пять основных условий поиска или предикатов:

|  |  |
| --- | --- |
| Условие поиска  | Ключевые слова, специальные символы  |
| Сравнение  | >,<,=,>=,<=,<>,!=  |
| Диапазон  | BETWEEN/NOT BETWEEN  |
| Принадлежность к множеству  | IN/NOT IN  |
| Соответствие шаблону  | LIKE/NOT LIKE  |
| Значение NULL  | IS NULL/IS NOT NULL  |

***Конструкция GROUP BY***

Конструкция GROUP BY предназначена для группировки строк. Строки группируются по одинаковым значениям в указанном столбце. В результате для каждой группы строк определяется одна итоговая строка. При использовании конструкции GROUP BY каждый элемент списка SELECT должен иметь единственное значение для всех строк группы. Таким образом элементами списка SELECT при использовании оператора GROUP BY могут быть:

* Имена столбцов (только если они используются так же в конструкции GROUP BY).
* Агрегатные (групповые, агрегирующие) функции (COUNT – количество значений в столбце, SUM – сумма значений в столбце, AVG – усредненное значение в столбце, MIN – минимальное значение в столбце, MAX – максимальное значение в столбце).
* Константы.
* Выражения из перечисленных выше элементов.

***Конструкция HAVING***

Конструкция HAVING предназначена для ограничения групп, помещенных в результирующую таблицу запроса, путем проверки на соответствие определенным условиям. Используется совместно с конструкцией GROUP BY. Имена столбцов, используемые в конструкции HAVING должны присутствовать в конструкции GROUP BY или применяться в агрегатных функциях.

***Примеры простых запросов***

Для примеров используется таблица shop (Магазины) (см. пункт «Задание»).

1. Выбрать всю информацию по всем магазинам:

сервер: ps-345=>пользователь: SELECT \* FROM shop;

2. Выбрать всю информацию по магазинам города Челябинска:

сервер: ps-345=>пользователь: SELECT \* FROM shopпользователь: WHERE town LIKE 'Челябинск';

3. Выбрать информацию, включающую в себя название магазина, адрес магазина, торговую площадь для всех магазинов города Челябинска. Озаглавить столбцы:

сервер: ps-345=>пользователь: SELECT nameas «название магазина», пользователь: adresas «адрес», areaas «площадь» пользователь: FROM shopпользователь: WHERE town LIKE 'Челябинск';

4. Определить максимальную торговую площадь в каждом городе. Отсортировать по убыванию торговой площади:

сервер: ps-345=>пользователь: SELECT townas «город»,MAX(area) пользователь: as «максимальная торговая площадь» пользователь: FROM shopпользователь: GROUP BY townпользователь: ORDER BY MAX(area) DESC;

5. Определить количество магазинов в каждом городе. Из результирующего списка исключить города , в которых меньше 2 магазинов. Отсортировать список городов по алфавиту:

сервер: ps-345=>пользователь: SELECT townas «город»,COUNT(shopid) пользователь: as «количество магазинов» пользователь: FROM shopпользователь: GROUP BY town

пользователь: HAVING COUNT(shopid)>=2

пользователь: ORDER BY town;

***Подзапросы***

Подзапросы это операторы SELECT, внедренные в тело другого оператора SELECT. Подзапросы могут использоваться в конструкциях WHERE и HAVING. Подзапросы бывают трех видов:

* скалярные – возвращают значение;
* строковые – возвращают строку;
* табличные – возвращают таблицу. Одновременно с подзапросами могут использоваться ключевые слова ALL,

ANY, SOME.

ALL – условие должно соблюдаться для всех строк подзапроса.

ANY, SOME – условие должно соблюдаться хотя бы для одной строки подзапроса.

Так же с подзапросами могут использоваться ключевые слова EXISTS и NOT EXISTS. Эти ключевые слова проверяют наличие строк в результирующей таблице подзапроса. Результат использования этих ключевых слов может быть TRUE или FALSE. Для ключевого слова EXISTS результат равен TRUE, если подзапрос, с которым это ключевое слово использовано возвращает хотя бы одну строку.

Для проверки нахождения некоторой строки в подзапросе используют предикат IN.

***Примеры запросов с подзапросами***

Для примеров используется таблица shop (Магазины) и таблица seller (Продавцы)

Определить магазины, в которых работают сотрудники с именем Татьяна. Указать название магазина, адрес:

сервер: ps-345=>

пользователь: SELECT nameas «магазин», adresas «адрес»

пользователь: FROM shop

пользователь: WHERE shopid IN

пользователь: (SELECT shopid

пользователь: FROM seller

пользователь: WHERE fio LIKE '%Татьяна%');

Определить магазины с максимальной торговой площадью:

сервер: ps-345=>

пользователь: SELECT nameas «магазин», adresas «адрес»

пользователь: areaas «площадь»

пользователь: FROM shop

пользователь: WHERE area=(SELECT MAX(area) FROM shop);

Определить всех сотрудников, работающих в городе Челябинске:

сервер: ps-345=>

|  |  |
| --- | --- |
| пользователь:  | SELECT fioas «ФИО»  |
| пользователь:  | FROM sellersl |
| пользователь:  | WHERE EXISTS (SELECT \* FROM shop s  |
| пользователь: | WHERE sl.shopid=s.shopid |
| пользователь: | AND town='Челябинск');  |
|  | *Многотабличные запросы* |

Многотабличные запросы -это запросы, которые используют данные нескольких таблиц.

Для того, чтобы объединить в результирующей таблице столбцы нескольких таблиц, используют соединение таблиц и ключевые слова JOIN ON. В результате операции соединения двух таблиц создается таблица, состоящая из пар связанных строк, выбранных из каждой таблицы. Каким образом связываются пары строк определяет условие, которое задается после ключевого слова ON.

SELECT [DISTINCT | ALL] {\* | *список столбцов*} FROM *имя\_таблицы1*JOIN *имя\_таблицы2* ON *условие* Аналогичное действие можно выполнить и таким образом: SELECT [DISTINCT | ALL] {\* | *список столбцов*} FROM *имя\_таблицы1, имя\_таблицы2* WHERE *условие*

Соединение таблиц, это частный случай декартова произведения таблиц. Декартово произведение двух таблиц – это таблица, состоящая из всех возможных пар строк, входящих в состав обеих таблиц. Для задания декартова произведения используются ключевые слова CROSS JOIN.

SELECT [DISTINCT | ALL] {\* | *список столбцов*} FROM *имя\_таблицы1*CROSS JOIN *имя\_таблицы2*

Аналогичное действие можно выполнить и таким образом:

SELECT [DISTINCT | ALL] {\* | *список столбцов*} FROM *имя\_таблицы1, имя\_таблицы2*

Для того чтобы объединить в результирующей таблице строки нескольких таблиц используют объединение таблиц и ключевое слово UNION. Таблицы должны быть совместимы по соединению – т.е. иметь одинаковую структуру (одинаковое количество столбцов, с одинаковыми типами данных).

SELECT [DISTINCT | ALL] {\* | *список столбцов*} FROM *имя\_таблицы1*WHERE *условие* UNION SELECT [DISTINCT | ALL] {\* | *список столбцов*} FROM *имя\_таблицы2* WHERE *условие*

***Примеры многотабличных запросов***

Определить всех сотрудников, работающих в городе Челябинске:

сервер: ps-345=>пользователь: SELECT fioas «ФИО» пользователь: FROM sellersl JOIN shop s пользователь: ON sl.shopid=s.shopidпользователь: WHERE town='Челябинск';

Определить количество продавцов в каждом магазине. В итоговой информации указать название магазина, адрес, торговую площадь, количество продавцов.

сервер: ps-345=>пользователь: SELECT nameas «магазин», adresas «адрес», пользователь: areaas «площадь», c as «количество продавцов» пользователь: FROM shop s JOIN пользователь:(SELECT shopid, COUNT(\*) as c пользователь: FROM sellerпользователь:GROUP BY shopid) n пользователь: ON s.shopid=n.shopid;

Необходимые принадлежности

1. Персональный компьютер с программным обеспечением.

2. Справочная литература

Работа в аудитории.

1. Разработать БД согласно заданию.

2. Составить письменно отчёт и продемонстрировать преподавателю.

Задание.

Используемая база данных «Сеть магазинов по продажам сумок» Состоит из 7 таблиц:

1. seller (Продавцы)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Атрибут | Ключ | Семантика | Тип данных | Ограничение целостности |
| 1 | sellerid | PK | Уникальный код продавца | serial | PRIMARY KEY |
| 2 | fio |  | Фамилия, имя, отчество продавца | varchar(40) | NOT NULL |
| 3 | shopid | FK (shop.shop.id) | Уникальныйкодмагазина | serial | NOT NULL, FOREIGN KEY |

2. shop (Магазины)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Атрибут | Ключ | Семантика | Тип данных | Ограничение целостности |
| 1 | shopid | PK | Уникальныйкодмагазина | serial | PRIMARY KEY |
| 2 | name |  | Название магазина | varchar(40) | NOT NULL |
| 3 | town |  | Город, в котором находится магазин | varchar(20) | NOT NULL, FOREIGN KEY |
| 4 | adress |  | Адрес магазина | varchar(40) | NOT NULL |
| 5 | area |  | Торговая площадь | Numeric(4.1) | NOT NULL, >0 |

3. sale (Продажи)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1  | saleid | PK  | Уникальный код продажи  | serial | PRIMARY KEY  |
| 2  | dateprod |  | Дата продажи  | date | DEFAULT NULL  |
| 3  | goodsid | FK (goods. goodsid)  | Уникальный код товара  | serial | NOT NULL, FOREIGN KEY  |
| 4  | sellerid | FK (seller. sellerid)  | Уникальный код продавца  | serial | FOREIGN KEY  |
| 5  | proceeds |  | Выручка за товар  | Numeric(6.2)  | NOT NULL  |

4. supplier (Поставщики)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1  | supid | PK  | Уникальный код поставщика  | serial | PRIMARY KEY  |
| 2  | supname |  | Название поставщика  | Varchar(40)  | NOT NULL  |
| 3  | suptown |  | Город поставщика  | Varchar(20)  | NOT NULL  |
| 4  | supadres |  | Адрес поставщика  | Varchar(40)  | NOT NULL  |
| 5  | rating |  | Рейтинг поставщика  | Integer | DEFAULT 0  |
| 6  | note |  | примечание  | text | DEFAULT NULL  |

5. goods (Товары)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1  | goodsid | PK  | Уникальный код товара  | serial | PRIMARY KEY  |
| 2  | model | FK (model. model)  | Название модели  | Varchar(40)  | NOT NULL, FOREIGN KEY  |
| 3  | cost |  | Цена закупки  | Numeric(6.2)  | NOT NULL, >0  |
| 4  | number |  | Поставлено всего  | integer | NOT NULL, >0  |
| 5  | remain |  | Количество в наличии  | integer | >0  |
| 5  | customid | FK (custom. customid)  | Уникальный код заказа  | serial | NOT NULL, FOREIGN KEY  |
| 6  | shopid | FK (shop.shopid)  | Уникальный код магазина  | serial | FOREIGN KEY  |
| 7  | costplus |  | Цена для покупателя  | Numeric(6.2)  | NOT NULL, CostPlus>Cost |

6. Model (Модели)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1  | model  | PK  | Названиемодели | Varchar(40)  | PRIMARY KEY  |
| 2  | descripti on  |  | Описание | text  |  |

7. Custom (Заказ)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1  | customid | PK  | Уникальный код заказа  | serial | PRIMARY KEY  |
| 2  | nom |  | Номер заказа  | integer | NOT NULL, >0  |
| 3  | datecust |  | Дата заказа  | date | NOT NULL  |
| 4  | supid | FK (supplier. supid)  | Уникальный код поставщика  | serial | NOT NULL, FOREIGN KEY  |
| 5  | period |  | Время выполнения заказа (дни)  | integer | NOT NULL, >=0  |
| 4  | datedel |  | Дата поставки  | date | DEFAULT NULL  |

Простые запросы.

1. Выбрать всю информацию по поставщикам. Отсортировать по названию.

2. Выбрать информацию по поставщикам, включающую название, адрес, рейтинг. Отсортировать по убыванию рейтинга.

3. Выбрать всех поставщиков из города Челябинск

4. Выбрать всех поставщиков из города Челябинск и Москва. Информация о поставщике должна включать название поставщика, рейтинг, город.

5. Выбрать всех поставщиков с рейтингом менее 0. Отсортировать по убыванию рейтинга.

Агрегатные функции

1. Определить среднюю торговую площадь в каждом городе.

2. Определить максимальную торговую площадь в каждом городе.

3. Определить минимальную торговую площадь в каждом городе.

4. Определить суммарную торговую площадь во всех городах, кроме Челябинска.

5. Определить количество магазинов в каждом городе.

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Проект БД;
6. Заключение

Контрольные вопросы.

1. Какие основные конструкции входят в оператор SELECT?

2. Можно ли менять порядок конструкций в операторе SELECT?

3. Укажите ключевое слово, предназначенное для исключения повторяющихся строк из результата запроса?

4. В чем отличие конструкций WHERE и HAVING?

5. С помощью какой конструкции выполняется сортировка результата запроса?

6. Какое ключевое слово используется для выполнения сортировки по убыванию?

7. Какие конструкции являются обязательными для оператора SELECT?

Практическое занятие №43

«Использование SQL запросов при сортировке данных в БД»

Цель: научится использовать язык запросов SQL для формирования сортировки данных в БД.

Пояснения.

Ключевое слово GROUP BY

Ключевое слово GROUP BY используется в операторе SELECT для того, чтобы объединять повторяющиеся значения в группы. Ключевое слово GROUP BY должно следовать за выражением WHERE и предшествовать ключевому слову ORDER BY.

Вот какая должна быть последовательность ключевых слов в операторе, выполняющем запрос:

SELECT FROM WHERE GROUP BY ORDER BY

Ключевое слово GROUP BY должно следовать за условиями в выражении ключевого слова WHERE и предшествовать ключевому слову ORDER BY, если последнее имеется.

SELECT столбец1, столбец2

FROM таблица1, таблица2 WHERE условия

GROUP BY столбец1, столбец2

ORDER BY столбец1, столбец2

Группирование выбранных данных

Группировать данные просто. В выражении ключевого слова GROUP BY могут использоваться только выбранные столбцы (т. е. столбцы из списка ключевого слова SELECT в операторе запроса). Если имя столбца не указано в списке ключевого слова SELECT, то имя этого столбца в выражении ключевого слова GROUP BY использовать нельзя. Это логично — как группировать в отчете данные, которых в нем нет?

Но если столбец выбран, то его имя должно быть включено в выражение ключевого слова GROUP BY. Имя столбца можно представить и его номером, о чем мы поговорим немного позже. При группировании данных порядок группирования столбцов не обязан совпадать с порядком, заданным в выражении ключевого слова SELECT.

К функциям группирования — функциям, используемым в выражении ключевого слова GROUP BY для объединения данных в группы, — относятся AVG, MAX, MIN, зим и COUNT.

 Создание групп и использование итоговых функций

При использовании в операторе SELECT ключевого слова GROUP BY должны соблюдаться некоторые правила. В частности, имена выбранных для отображения столбцов должны присутствовать и в выражении ключевого слова GROUP BY, за исключением тех, к которым применены итоговые функции. Столбцы в выражении ключевого слова GROUP BY не обязательно должны быть представлены в том же порядке, что и в выражении ключевого слова SELECT. Но если имя столбца указано в выражении ключевого слова SELECT, имя этого столбца должно присутствовать и в выражении ключевого слова GROUP BY. Вот несколько примеров использования оператора SELECT с ключевым словом GROUP BY.

Пример

SELECT EMP\_ID, CITY

FROM EMPLOYEEJTBL

GROUP BY CITY, EMP\_ID;

В этом операторе SQL из таблицы EMPLOYEE\_TBL выбираются столбцы EMP\_ID и CITY, а данные последних выводятся сгруппированными сначала по CITY, а затем по EMP\_ID.

Обратите внимание на порядок выбора столбцов и на порядок столбцов в выражении ключевого слова GROUP BY

Пример

SELECT EMP\_ID, SUM(SALARY)

FROM EMPLOYEE\_PAY\_TBL

GROUP BY SALARY, EMP\_ID;

Этот оператор SQL возвращает данные столбца EMP\_ID и сумму по группам зарплат, созданным по величине зарплаты (SALARY) и табельному номеру (EMP\_ID).

Пример

SELECT SUM(SALAPY)

FROM EMPLOYEE\_PAY\_TBL;

Здесь оператор SQL возвращает сумму всех выплат по зарплате из таблицы

ЕМ PLOYEE\_PAY\_TBL.

Пример

SELECT SUM(SALARY)

FROM EMPLOYEE\_PAY\_TBL

GROUPBYSALARY;

Здесь оператор SQL возвращает суммы по группам, созданным по всем уровням зарплаты.

Вот несколько примеров с использованием реальных данных. Сначала убедимся, что в таблице EMPLOYEE\_TBL представлены три города.

Ввод:

SELECT CITY

FROM EMPLOYEEJTBL ;

Вывод:

CITY

GREENWOOD

INDIANAPOLIS

WHITELAND

INDIANAPOLIS

INDIANAPOLIS

INDIANAPOLIS

В следующем примере подсчитывается число записей по каждому городу. Именно из-за того, что используется ключевое слово GROUP BY, вы здесь видите результаты по каждому из городов в отдельности.

Ввод:

SELECT CITY, COUNT(\*)

FROM EMPLOYEE\_TBL

GROUP BY CITY;

Вывод:

CITY COUNT(\*)

GREENWOOD 1

INDIANAPOLIS 4

WHITELAND 1

Следующий запрос осуществляет выборку из временной таблицы, созданной на основе таблиц EMPLOYEE\_TBL и EMPLOYEE\_PAY\_TBL. О том как объединить две таблицы в одном запросе, мы поговорим чуть позже.

Ввод:

SELECT \*

FROM EMP\_PAY\_TMP;

Вывод:

CITY LAST\_NAME FIRST\_NAME PAY\_RATE SALARY

GREENWOOD STEPHENS TINA 30000

INDIANAPOLIS PLEW LINDA 14.75

WHITELAND GLASS BRANDON 40000

INDIANAPOLIS GLASS JACOB 20000

INDIANAPOLIS WALLACE MARIAH 11

INDIANAPOLIS SPURGEON TIFFANY 15

В следующем примере с помощью функции AVG извлекаются средние значения для почасовой оплаты и зарплаты по каждому городу в отдельности. Для городов Гринвуд и Уайтленд среднего значения почасовой оплаты нет, поскольку работа ни одного из представленных в таблице служащих из этих городов не оплачивается почасово.

Ввод:

SELECT CITY, AVG(PAY\_RATE), AVG(SALARY)

FROM EMP\_PAY\_TMP

GROUP BY CITY;

Вывод:

CITY AVG(PAY\_RATE) AVG(SALARY}

GREENWOOD 30000

INDIANAPOLIS 13.5833333 20000

WHITELAND 40000

В следующем примере для группирования данных комбинируется использование нескольких компонентов запроса. Необходимо получить опять же средние значения для почасовой оплаты и зарплаты, но только для городов Индианаполис и Уайтленд. Для этого данные группируются по полю CITY — другого выбора здесь нет, поскольку иначе из выбранных столбцов используется итоговая функция. Наконец, отчет упорядочивается сначала по столбцу 2, а затем по столбцу 3, т. е. по средней почасовой оплате и средней зарплате. Попытайтесь до конца разобраться в показанном ниже операторе и выведенных данных.

Ввод:

SELECT CITY, AVG(PAY\_RATE), AVG(SALARY)

FROM EMP\_PAY\_TMP

WHERE CITY IN ('INDIANAPOLIS','WHITELAND')

GROUP BY CITY

ORDER BY 2,3;

Вывод:

CITY AVG(PAY\_RATE) AVG(SALARY)

INDIANAPOLIS 13.5833333 20000

WHITELAND 40000

Значения сортируются так, что значения NULL оказываются в конце. Поэтому запись для города Индианаполис представлена первой. Город Гринвуд не был выбран, но если бы был, то соответствующая ему запись была бы представлена перед записью для Уайтленда, поскольку для Гринвуда средняя зарплата (SALARY) равна 30000, а средняя зарплата является вторым параметром сортировки в выражении ключевого слова ORDER BY.

В завершение раздела рассмотрим использование в выражении ключевого слова ORDER BY итоговых функций МАХ и MIN.

Ввод:

SELECT CITY, MAX(PAY\_RATE), MIN(SALARY)

FROM EMP\_PAY\_TMP

GROUP BY CITY;

Вывод:

CITY MAX(PAY\_RATE) MIN(SALARY)

GREENWOOD 30000

INDIANAPOLIS 15 20000

WHITELAND 40000

Представление имен столбцов числами

В отличие от выражения ключевого слова ORDER BY, в выражении ключевого слова GROUP BY указать порядок столбцов с помощью их номеров нельзя — за исключением того случая, когда используется ключевое слово UNION и имена всех столбцов разные. Вот пример использования номеров вместо имен столбцов.

SELECT EMP\_ID, SUM(SALARY)

FROM EMPLOYEE\_PAY\_TBL

UNION

SELECT EMP\_ID, SUM(PAY\_RATE)

FROM EMPLOYEE\_PAY\_TBL

GROUP BY 2, 1;

Этот оператор SQL возвращает табельный номер служащего (EMP\_ID) и группирует суммы по значениям зарплаты. При использовании ключевого слова UNION результаты двух операторов SELECT объединяются. Группирование выполняется сначала по столбцу 2Б, представляющем зарплату (SALARY), а затем по столбцу 1, представляющем табельный номер служащего (EMP\_ID).

 GROUP BY И ORDER BY

Обратите внимание на то, что GROUP BY и ORDER BY работают одинаково в том смысле, что оба эти ключевые слова задают сортировку данных. В выражении ключевого слова ORDER BY задается сортировка данных запроса, а в выражении ключевого слова GROUP BY — сортировка этих данных по группам. Поэтому ключевое слово GROUP BY можно использовать для сортировки точно так же, как и ORDER BY.

Вот несколько особенностей использования ключевого слова GROUP BY для сортировки.

• Все выбранные столбцы, к которым не применяются итоговые функции, должны быть указаны в списке ключевого слова GROUP BY.

• В отличие от выражения ключевого слова ORDER BY, в выражении ключевого слова GROUP BY имена столбцов нельзя заменить числами.

• Использовать ключевое слово GROUP BY вообще нет необходимости, если не используются итоговые функции.

Вот пример использования для сортировки данных ключевого слова GROUP BY вместо ключевого слова ORDER BY:

Ввод:

SELECT LAST\_NAME, FIRST\_NAME, CITY

FROM EMPLOYEE\_TBL

GROUP BY LAST\_NAME;

Вывод:

SELECT LAST\_NAME, FIRST\_NAME, CITY

 \*

ERROR at line 1:

ORA-00979: not a GROUP BY expression

В этом примере сервер базы данных сообщает об ошибке из-за того, что имя столбца FIRST\_NAME не указано в выражении ключевого слова GROUP BY. Помните о том, что все столбцы из списка ключевого слова SELECT должны быть указаны в выражении ключевого слова GROUP BY, за исключением тех столбцов, к которым применяются итоговые функции.

В следующем примере проблема предыдущего оператора решена путем добавления в список ключевого слова GROUP BY недостающих имен из списка ключевого слова SELECT.

Ввод:

SELECT IAST\_NAME, FIRST\_NAME, CITY

FROM EMPLOYEE\_TBL

GROUP BY LAST\_NAME, FIRST\_NAME, CITY;

Вывод:

LAST\_NAME FIRST\_NAME CITY

GLASS BRANDON WHITELAND

GLASS JACOB INDIANAPOLIS

PLEW LINDA INDIANAPOLIS

SPURGEON TIFFANY INDIANAPOLIS

STEPHENS TINA GREENWOOD

WALLACE MARIAH INDIANAPOLIS

В этом примере выбираются те же данные из той же таблицы, но уже все столбцы перечислены в выражении ключевого слова GROUP BY в том же порядке, в каком они указаны в списке ключевого слова SELECT. В результате данные показаны отсортированными сначала по столбцу LAST\_NAME, затем по столбцу FIRST\_NAME и наконец, по столбцу CITY. С помощью ключевого слова ORDER BY то же самое получить легче, но для правильного использования ключевого слова GROUP BY, наверное, будет полезно разобраться, как с его помощью сортируются данные, перед тем, как выполнить группирование результатов.

В следующем примере получается выборка из таблицы EMPLOYEE\_TBL и используется ключевое слово GROUP BY для упорядочения данных по значениям столбца CITY.

Ввод:

SELECT CITY, LAST\_HAME

FROM EMPLOYEEJTBL

GROUP BY CITY, LAST\_NAME;

Вывод:

CITY LAST\_NAME

GREENWOOD STEPHENS

INDIANAPOLIS GLASS

INDIANAPOLIS PLEW

INDIANAPOLIS SPURGEON

INDIANAPOLISWALLACE

WHITELANDGLASS

Сравните порядок представленных здесь данных с порядком в предыдущих примерах. Здесь учтены все записи таблицы EMPLOYEE\_TBL, результаты группируются по значениям столбца CITY, но упорядочены сначала по числу записей для каждого города.

Ввод:

SELECT CITY, COUNT(\*)

FROM EMPLOYEEJTBL

GROUP BY CITY

ORDER BY 2,1;

Вывод:

CITY COUNT(\*)

GREENWOOD 1

WHITELAND 1

INDIANAPOLIS 4

Обратите внимание на порядок представления полученных данных. Они сначала отсортированы по числу записей для каждого города и только потом по названиям городов. Для первых двух городов число записей равно 1. Поскольку числа совпадают, порядок представления записей определяется вторым параметром сортировки, а именно, названием города. Поэтому Гринвуд идет перед Уайтлендом.

Хотя ключевые слова GROUP BY и ORDER BY и функционируют подобным образом, между ними имеется существенное отличие. Ключевое слово GROUP BY предназначено для группирования одинаковых значений, а задачей ORDER BY является представление данных просто в определенном порядке. Ключевые слова GROUP BY и ORDER BY можно использовать в одном операторе SELECT, но каждое из них должно выполнять свою задачу. В одном операторе SELECT ключевое слово GROUP BY должно предшествовать ключевому слову ORDER BY.

Ключевое слово GROUP BY можно использовать для сортировки данных в операторе CREATE VIEW, а вот ключевое слово ORDER BY использовать в операторе CREATE VIEW нельзя.

Необходимые принадлежности

1. Персональный компьютер с программным обеспечением.

2. Справочная литература

Работа в аудитории.

1. Разработать БД согласно заданию.

2. Составить письменно отчёт и продемонстрировать преподавателю.

Задание.

1. Открыть БД «Сеть магазинов по продаже сумок».

2. Выполнить сортировку данных по ФИО продавца.

3. Выполнить сортировку данных по названию магазина.

4. Выполнить сортировку по выручке за товар.

5. Выполнить сортировку по рейтингу поставщика.

6. Выполнить сортировку по количеству товара в магазине.

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Проект БД;
6. Заключение

Контрольные вопросы.

1. Какие способы сортировки данных существуют?

2. Для чего используется ключевое слово GroupBy.

3. С какими итоговыми функциями SQL можно использовать ключевое слово GroupBy.

4. В чём отличия и сходства между ключевыми словами GroupBy и OrderBy?

5. Для чего используется ключевое слово Union?

Практическое занятие №44

«Использование триггеров в БД»

Цель: научится использовать триггеры для формирования БД.

Пояснения.

Триггер – это хранимая процедура особого типа, вызываемая на выполнение в ответ на определённые события. Триггеры подразделяются на два основных типа: триггеры языка определения данных ( DDL) и триггеры языка манипулирования данными ( DML)

Триггеры DDL активизируются в ответ на внесение каких-либо изменений в структуру б.д пользователями ( создание, удаление, изменение структуры таблиц,).

Триггеры DML представляют собой фрагменты кода, которые закрепляются за конкретной таблицей.

В отличие от хранимых процедур, при использовании которых необходимо явно вызывать на выполнение определённый код, триггеры вызываются на выполнение автоматически при обнаружении события, связанного с ним. Триггеры не получают параметров и не возвращают значений.

Конструкция FOR ( или AFTER) позволяет указать какое действие приводит к запуску триггера.

Триггер Insert этот триггер вызывается на выполнение каждый раз, когда вставляется новая строка в таблицу, за которой закреплён триггер. Для каждой новой строки создаётся её копия и вставляется в таблицу INSERTED. Эта таблица сохраняется до тех пор пока действует триггер ( с момента запуска до его завершения).

Триггер Delete вызывается на выполнение каждый раз, при удалении записи. Копия удаляемой записи помещается во временную таблицу Deleted.

Триггер Update вызывается на выполнение при обновлении записи таблицы, для которой он создан.операция модификации строки трактуется как удаление старой версии строки ( помещение её в Deleted) и добавление новой строки ( помещение её в INSERTED). При этом триггеры на удаление и добавление не запускаются.

*Условный оператор для триггера в языке SQL-Transact*

|  |  |
| --- | --- |
| Один оператор | Несколько операторов |
| IF условиеОператор1ElseОператор2 | **IF** условие*begin*Оператор1Оператор2*end***Else***begin*Оператор3Оператор4*end* |

Примертриггера:

//Определениетриггера

CREATE TRIGGER Add\_copies BEFORE

INSERT ORDER 1

ON Stepanov.Books

REFERENCING NEW AS New\_books

FOR EACH ROW

/\* Предваряющий строчный триггер, активизирующийся при наступлении события "Добавление новой записи" в таблицу Stepanov.Books. В этой таблице имеются три поля, для двух из которых определены значения по умолчанию. Поэтому воператоре INSERT достаточно ввести значения только одного поля - Code\_book \*/

BEGIN

 DECLARE Kol SMALLINT;

 SET Kol=New\_books.number;

 /\* MESSAGE 'Code\_book=',

New\_books.Code\_book; \*/

 WHILE Kol>0 LOOP

 INSERT INTO Stepanov.Copies(

Code\_book)

VALUES(

New\_books.Code\_book);

 SET Kol=Kol-1

 END LOOP

END

Необходимые принадлежности

1. Персональный компьютер с программным обеспечением.

2. Справочная литература

Работа в аудитории.

1. Разработать БД согласно заданию.

2. Составить письменно отчёт и продемонстрировать преподавателю.

Задание.

Создайте триггер на изменение зарплаты сотрудника, проверяющий, находится ли она в заданных должностных рамках. Рамки зарплаты берутся из таблицы JOBS. Если назначенная зарплата меньше минимально возможной, то установить сотруднику минимально возможную зарплату (min\_salary). И наоборот, если назначенная зарплата больше максимально возможной, то установить сотруднику максимально возможную зарплату (max\_salary).

Содержание отчета

1. Номер и название работы;
2. Цель работы;
3. Задание с исходными данными;
4. Необходимые принадлежности;
5. Проект БД;
6. Заключение

Контрольные вопросы.

1. Дать определение понятию триггер.

2. Какие виды триггеров существуют?

3. Чем триггеры отличаются от процедур?

4. Перечислите часто используемые триггеры.