МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РЕСПУБЛИКИ ДАГЕСТАН

Частное профессиональное образовательное учреждение
«РЕСПУБЛИКАНСКИЙ ПОЛИПРОФЕССИОНАЛЬНЫЙ КОЛЛЕДЖ»
(ЧПОУ «Республиканский полипрофессиональный колледж»)

УТВЕРЖДАЮ
Зам. директора по учебно-
методической работе
_____ / Кадрышева Ж.А
« 25 » октября 2022 г.

Комплект контрольно-оценочных средств
по учебной дисциплине

ОП.08 ОСНОВЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ

программы подготовки специалистов среднего звена по специальности: 09.02.07 Информационные системы и программирование

Год набора: 2021

Кизляр
2022г.

ОДОБРЕН
на заседании цикловой методической комиссии общепрофессиональных дисциплин и профессиональных модулей по специальности 09.02.07 Информационные системы и программирование
Протокол № 2 от «18» октября 2022 г.

Председатель ЦМК _____/
Кадрышева Ж.А.

Составлен в соответствии с требованиями федерального государственного образовательного стандарта по специальности 09.02.07 Информационные системы и программирование и рабочей программы по дисциплине ОП.08 Основы проектирования баз данных

Организация-разработчик: Частное профессиональное образовательное учреждение «Республиканский полипрофессиональный колледж».

Разработчик(и):

Потапов Игорь Алексеевич, преподаватель
Ф.И.О., ученая степень, звание, должность

СОДЕРЖАНИЕ

1. ПАСПОРТ КОМПЛЕКТА КОС ПО УЧЕБНОЙ ДИСЦИПЛИНЕ
ОП.08 Основы проектирования баз данных

1.1 Общие положения.

Контрольно-оценочные средства (КОС) разработаны в соответствии с требованиями образовательной программы и Федерального государственного образовательного стандарта СПО по специальности 09.02.07 Информационные системы и программирование, программы дисциплины ОП.08 Основы проектирования баз данных.

КОС включает контрольные материалы для проведения текущего контроля и промежуточной аттестации в форме:

- 2 семестр - экзамен.

КОС разработаны в соответствии с:

- образовательной программой СПО по специальности 09.02.07 Информационные системы и программирование;

- программы учебной дисциплины ОП.08 Основы проектирования баз данных.

1.2. Результаты освоения дисциплины, подлежащие проверке

Результаты обучения (освоенные умения, усвоенные знания)	**Наименование элемента умений/знаний**
У1	Проектировать реляционную базу данных;
У2	Использовать язык запросов для программного извлечения сведений из баз данных
Знания –	
З1 –	Основы теории баз данных; модели данных;
З2 –	Особенности реляционной модели и проектирование баз данных;
З3 –	Изобразительные средства, используемые в ER-моделировании;
З4 –	Основы реляционной алгебры;
З5 –	Принципы проектирования баз данных;
З6 –	Обеспечение непротиворечивости и целостности данных;
З7 –	Средства проектирования структур баз данных; язык запросов SQL

1.3. Распределение оценивания результатов обучения по видам контроля

Код и наименование элемента умений или знаний	**Виды аттестации**	
	Текущий контроль	*Промежуточная аттестация*
У1 Проектировать реляционную базу данных;	Наблюдение за выполнением практического задания. Оценка выполнения практического задания. Контроль выполнения самостоятельной работы.	Контрольная работа; Диф.зачет
У2 Использовать язык запросов для программного	Наблюдение за выполнением практического задания.	Контрольная работа;

извлечения сведений из баз данных	Оценка выполнения практического задания. Контроль выполнения самостоятельной работы.	Диф.зачет
Знания		Контрольная работа; Диф.зачет
З1 Основы теории баз данных; модели данных;	Собеседование, тестирование	Контрольная работа; Диф.зачет
З2 Особенности реляционной модели и проектирование баз данных;	Наблюдение за выполнением практического задания. Оценка выполнения практического задания. Контроль выполнения самостоятельной работы.	Контрольная работа; Диф.зачет
З3 Изобразительные средства, используемые в ER-моделировании;	Наблюдение за выполнением практического задания. Оценка выполнения практического задания. Контроль выполнения самостоятельной работы.	Контрольная работа; Диф.зачет
З4 Основы реляционной алгебры;	Наблюдение за выполнением практического задания. Оценка выполнения практического задания. Контроль выполнения самостоятельной работы.	Контрольная работа; Диф.зачет
З5 Принципы проектирования баз данных;	Наблюдение за выполнением практического задания. Оценка выполнения практического задания. Контроль выполнения самостоятельной работы.	Контрольная работа; Диф.зачет
З6 Обеспечение непротиворечивости и целостности данных;	Наблюдение за выполнением практического задания. Оценка выполнения практического задания. Контроль выполнения самостоятельной работы.	Контрольная работа; Диф.зачет
З7 Средства проектирования структур баз данных; язык запросов SQL	Наблюдение за выполнением практического задания. Оценка выполнения практического задания. Контроль выполнения самостоятельной работы.	Контрольная работа; Диф.зачет

2. СПЕЦИФИКАЦИИ И ВАРИАНТЫ ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ

2.1 СПЕЦИФИКАЦИЯ ОЦЕНОЧНЫХ СРЕДСТВ

2.1.1 Назначение

Спецификацией устанавливаются требования к содержанию и оформлению вариантов теста.

Тест входит в состав комплекса оценочных средств и предназначен для *промежуточного* контроля, оценки знаний и умений аттестуемых, соответствующих основным показателям оценки результатов подготовки по программе учебной дисциплины «Основы проектирования баз данных» основной профессиональной образовательной программы по специальности 09.02.07 «Информационные системы и программирование».

2.1.2. Контингент аттестуемых: студенты 1 курса

2.1.3. Форма и условия аттестации: Текущий контроль проходит в виде тестирования по разделам:

Раздел 1 Теория проектирования баз данных
Раздел 2 Организация баз данных
Раздел 3 Организация запросов SQL

2.1.4. Время выполнения:

подготовка 5 минут;
выполнение 80 минут;
оформление и сдача 5 минут;
всего 90 минут.

2.1.5. Рекомендуемая литература для разработки оценочных средств и подготовки, обучающихся к аттестации.

литературы

Библиографическое описание издания (автор, заглавие, вид, место и год издания, кол. стр.)	**Основная/ дополнительная литература**	**Книгообеспеченность**	
		Кол-во. экз. в библ.	**Электронные ресурсы**
Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для среднего профессионального образования / В. М. Илюшечкин. — испр. и доп. — Москва : Издательство Юрайт, 2023. — 213 с.	Основная	-	https://urait.ru/bcode/471698
Стружкин, Н. П. Базы данных: проектирование : учебник для среднего профессионального образования / Н. П. Стружкин, В. В. Годин. — Москва : Издательство Юрайт, 2023. — 477 с.	Основная	-	https://urait.ru/bcode/518499
Нестеров, С. А. Базы данных : учебник и практикум для среднего профессионального образования / С. А. Нестеров. — Москва : Издательство Юрайт, 2023. — 230 с.	Основная	-	https://urait.ru/bcode/518507
Советов, Б. Я. Базы данных : учебник для среднего профессионального образования / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. — 3-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2023. — 420 с.	Основная	-	https://urait.ru/bcode/514585

Федорова Г.Н. Основы проектирования баз данных. – Москва: Академия, 2021. – 224 с.	Основная	-	https://urait.ru/bcode/513827
Стружкин, Н. П. Базы данных: проектирование. Практикум : учебное пособие для среднего профессионального образования / Н. П. Стружкин, В. В. Годин. — Москва : Издательство Юрайт, 2023. — 291 с.	Дополнительная	-	https://urait.ru/bcode/516929

Таблица 2 – Перечень современных профессиональных баз данных (СПБД)

№	Наименование СПБД
1	Научная электронная библиотека eLIBRARRY - www.elibrary.ru
2	Научная электронная библиотека КиберЛениНка - www.cyberleninka.ru

Таблица 3 – Перечень информационных справочных систем (ИСС)

№	Наименование ИСС
1	Справочная правовая система КонсультантПлюс www.consultant.ru
2	Электронная библиотечная система ЭБС ЮРАИТ - www.urait.ru

2.1.6. Перечень материалов, оборудования и информационных источников.

Кабинет № 31информатики (для проведения занятий лекционного типа и занятий семинарского типа, курсового проектирования (выполнения курсовых работ) групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации) оборудован мультимедийным комплексом. Специализированная мебель: Учебная мебель на 39 посадочных места (столов трехместных 13 шт., скамеек 13 шт.), рабочее место преподавателя (стол 1 шт., стул 1 шт.), кафедра 1 шт. доска меловая 3х секционная 1шт. Компьютер Intel Pentium Dual CPU E2160 1,8 GHz ОЗУ- 2 Gb, HDD-500Gb, DVD RV-ROM, Клавиатура, Мышь. ОС windows 7 Максимальная. Локальный сеть с выходом в Интернет. Видеопроектор потолочный Epson EB-S82, проекционный экран Clasic Solition 266x149, акустические колонки Genius.

Наборы демонстрационного оборудования и учебно-наглядных пособий: мультимедийные приложения к лекционным курсам и практическим занятиям, интерактивные учебно-наглядные пособия.

Лаборатория № 2.2 программирования и баз данных. Компьютерный класс (для проведения практических занятий, с применением вычислительной техники) оборудован мультимедийным комплексом. Специализированная мебель и оборудование:

Учебная мебель на 16 посадочных мест, рабочее место преподавателя (стол – 1 шт., стул – 1 шт.). Компьютер Intel i5 7400/1Tb/8Gb/Philips 243V5Q 23' – 16 шт. Компьютер Intel i3 -2100 2.4 Ghz/4/500Gb/Acer V193 19» – 1 шт. Мультимедийный проектор Тип 1 Optoma x 400 – 1 шт. Консультант + (Договор поставки и сопровождения экземпляров системы № 124 от 28.08.2020), 7-Zip (freeware), Acrobat Reader DC (freeware), Adobe Acrobat Reader DC (freeware), FireFox 77.0.1 (freeware), Google Chrome 83.0.4103.97 (freeware), VLC media player (freeware), K-Lite Codec Pack Full (freeware). Программное обеспечение общего и профессионального назначения бесплатное (с открытой лицензией): EclipseIDEforJavaEEDevelopers, .NETFrameworkJDK 8, MicrosoftSQLServerExpressEdition, RAD Studio, NetBeans, ARIS Inkscape, MySQLInstallerforWindows, SQLServerManagementStudio, MicrosoftSQLServerJavaConnector, AndroidStudio, IntelliJIDEA.

Наборы демонстрационного оборудования и учебно-наглядных пособий:

мультимедийные приложения к лекционным курсам и практическим занятиям, интерактивные учебнонаглядные пособия.

2.2. СПЕЦИФИКАЦИЯ ОЦЕНОЧНЫХ СРЕДСТВ

2.2.1. Назначение

Спецификацией устанавливаются требования к содержанию и оформлению вариантов оценочного средства.

Практическое задание входит в состав комплекса оценочных средств и предназначено для *текущего* контроля и оценки знаний и умений обучающихся, соответствующих основным показателям оценки знаний и умений обучающихся, соответствующих основным показателям оценки результатов подготовки по программе учебной дисциплины «Основы проектирования баз данных» основной профессиональной образовательной программы по специальности 09.02.07 «Информационные системы и программирование».

2.2.2. Контингент аттестуемых: студенты 1 курса

2.2.3. Форма и условия аттестации: Текущий контроль проходит в виде написания рефератов по темам:

Этапы проектирования баз данных

Основные понятия языка SQL

2.2.4. Время выполнения:

подготовка 10 минут;

выполнение 60 минут;

оформление и сдача 10 минут;

всего 1 час 20 минут.

2.2.5. Рекомендуемая литература для разработки оценочных средств и подготовки, обучающихся к аттестации.

Библиографическое описание издания (автор, заглавие, вид, место и год издания, кол. стр.)	**Основная/ дополнительная литература**	**Книгообеспеченность**	
		Кол-во. экз. в библ.	**Электронные ресурсы**
Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для среднего профессионального образования / В. М. Илюшечкин. — испр. и доп. — Москва : Издательство Юрайт, 2023. — 213 с.	Основная	-	https://urait.ru/bcode/471698
Стружкин, Н. П. Базы данных: проектирование : учебник для среднего профессионального образования / Н. П. Стружкин, В. В. Годин. — Москва : Издательство Юрайт, 2023. — 477 с.	Основная	-	https://urait.ru/bcode/518499
Нестеров, С. А. Базы данных : учебник и практикум для среднего профессионального образования / С. А. Нестеров. — Москва : Издательство Юрайт, 2023. — 230 с.	Основная	-	https://urait.ru/bcode/518507
Советов, Б. Я. Базы данных : учебник для среднего профессионального образования / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. — 3-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2023. — 420 с.	Основная	-	https://urait.ru/bcode/514585
Федорова Г.Н. Основы проектирования баз данных. – Москва: Академия, 2021. – 224 с.	Основная	-	https://urait.ru/bcode/513827

Стружкин, Н. П. Базы данных: проектирование. Практикум : учебное пособие для среднего профессионального образования / Н. П. Стружкин, В. В. Годин. — Москва : Издательство Юрайт, 2023. — 291 с.	Дополнительная	-	https://urait.ru/bcode/516929

Таблица 2 – Перечень современных профессиональных баз данных (СПБД)

№	Наименование СПБД
1	Научная электронная библиотека eLIBRARRY - www.elibrary.ru
2	Научная электронная библиотека КиберЛениНка - www.cyberleninka.ru

Таблица 3 – Перечень информационных справочных систем (ИСС)

№	Наименование ИСС
1	Справочная правовая система КонсультантПлюс www.consultant.ru
2	Электронная библиотечная система ЭБС ЮРАИТ - www.urait.ru

2.2.6. Перечень материалов, оборудования и информационных источников.

Кабинет № 31информатики (для проведения занятий лекционного типа и занятий семинарского типа, курсового проектирования (выполнения курсовых работ) групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации) оборудован мультимедийным комплексом. Специализированная мебель: Учебная мебель на 39 посадочных места (столов трехместных 13 шт., скамеек 13 шт.), рабочее место преподавателя (стол 1 шт., стул 1 шт.), кафедра 1 шт. доска меловая 3х секционная 1шт. Компьютер Intel Pentium Dual CPU E2160 1,8 GHz ОЗУ- 2 Gb, HDD-500Gb, DVD RV-ROM, Клавиатура, Мышь. ОС windows 7 Максимальная. Локальный сеть с выходом в Интернет. Видеопроектор потолочный Epson EB-S82, проекционный экран Clasic Solition 266x149, акустические колонки Genius.

Наборы демонстрационного оборудования и учебно-наглядных пособий: мультимедийные приложения к лекционным курсам и практическим занятиям, интерактивные учебно-наглядные пособия.

Лаборатория № 2.2 программирования и баз данных. Компьютерный класс (для проведения практических занятий, с применением вычислительной техники) оборудован мультимедийным комплексом. Специализированная мебель и оборудование:

Учебная мебель на 16 посадочных мест, рабочее место преподавателя (стол – 1 шт., стул – 1 шт.). Компьютер Intel i5 7400/1Tb/8Gb/Philips 243V5Q 23' – 16 шт. Компьютер Intel i3 -2100 2.4 Ghz/4/500Gb/Acer V193 19» – 1 шт. Мультимедийный проектор Тип 1 Optoma x 400 – 1 шт. Консультант + (Договор поставки и сопровождения экземпляров системы № 124 от 28.08.2020), 7-Zip (freeware), Acrobat Reader DC (freeware), Adobe Acrobat Reader DC (freeware), FireFox 77.0.1 (freeware), Google Chrome 83.0.4103.97 (freeware), VLC media player (freeware), K-Lite Codec Pack Full (freeware). Программное обеспечение общего и профессионального назначения бесплатное (с открытой лицензией): EclipseIDEforJavaEEDevelopers, .NETFrameworkJDK 8, MicrosoftSQLServerExpressEdition, RAD Studio, NetBeans, ARIS Inkscape, MySQLInstallerforWindows, SQLServerManagementStudio, MicrosoftSQLServerJavaConnector, AndroidStudio, IntelliJIDEA.

Наборы демонстрационного оборудования и учебно-наглядных пособий: мультимедийные приложения к лекционным курсам и практическим занятиям, интерактивные учебнонаглядные пособия.

2.3 ВАРИАНТЫ ОЦЕНОЧНЫХ СРЕДСТВ
Тест по разделу Основы проектирования баз данных

Вариант 1

1. Определите, каких баз данных не существует:
   a. Сетевые
   b. Реляционные
   c. Табличные
   d. Иерархические

2. Определите, что входит в состав сетевой базы данных:
   a. Элемент данных
   b. Домен
   c. Агрегат данных
   d. Поле

3. Выберите правильные суждения:
   a. База данных – это поименованная совокупность хранящихся вместе данных при наличии такой минимальной избыточности, при которой возможно её использование одним или несколькими приложениями
   b. Целостность данных – это совокупность методов по обеспечению полноценного сбора и хранения данных в БД
   c. Архитектура "файл – сервер" больше всего подходит для не больших баз данных
   d. Одним из требований к СУБД является минимизация избыточностей

4. Выберите правильные суждения:
   a. К конечным пользователям базы данных относятся косвенные и прямые
   b. Одним из требований к СУБД является проектирование инфологической и даталогической модели
   c. Одним из требований к СУБД является простота эксплуатации и безопасность
   d. Одним из требований к СУБД является простота физической реорганизации

5. Выберите правильные суждения:
   a. Данные – это представление фактов и идей в формализованном виде, пригодном для передачи и переработки в некотором процессе
   b. Данные – это смысл, который придается данным при их представлении
   c. Информация – это представление фактов и идей в формализованном виде, пригодном для передачи и переработки в некотором процессе
   d. Информация – это смысл, который придается данным при их представлении

6. Определите, какая архитектура изображена на рисунке:

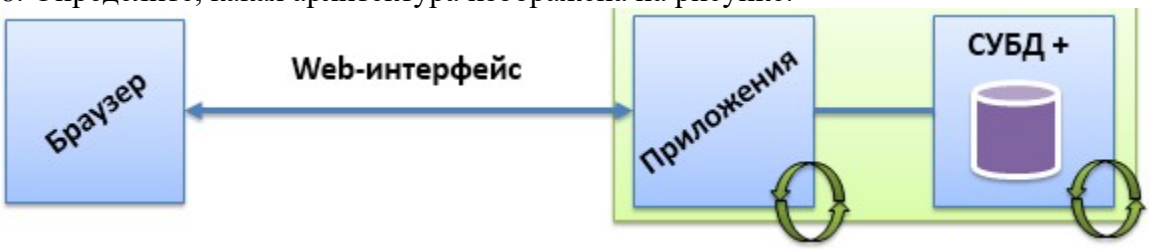
   a. Файл-сервер

b. Клиент-сервер
c. Web-сервер
d. Сервер-клиент

7. Определите, какая архитектура изображена на рисунке:

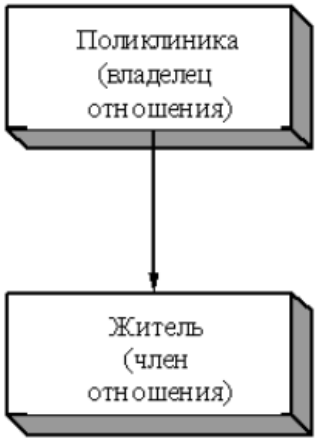
a. клиент-сервер
b. файл-сервер
c. пример веб-приложения
d. интерфейс-субд

8. Определите, какая архитектура изображена на рисунке:



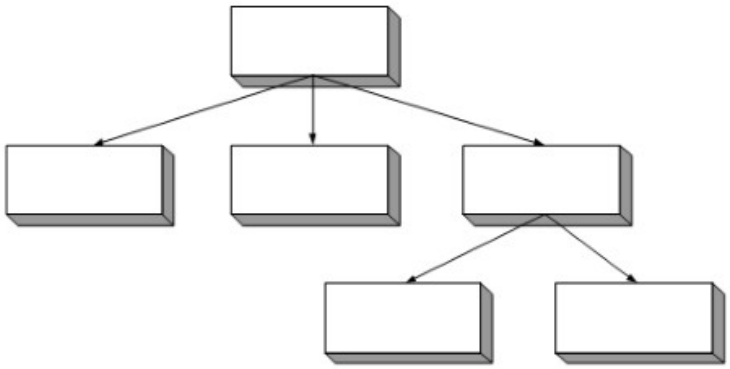
a. клиент-сервер
b. файл-сервер
c. веб приложение
d. веб-архитектура

9. Определите, что изображено на рисунке:



a. Пример сетевой БД
b. Групповое отношение
c. Пример иерархической БД
d. Диаграмма сущность связь

10.Определите модель базы данных (БД), изображенной на рисунке:



A. Пример иерархической БД
B. Пример сетевой БД
C. Пример реляционной БД
D. архитектура файл-сервер

11. Выберите правильные суждения:
    A. Структура данных как и в сетевой модели, определяется терминами: элемент данных, агрегат данных, запись, групповое отношение, база данных.
    B. Структура данных иерархической и сетевой модели отличаются агрегатом данных
    C. В структуру данных сетевой модели обязательно входит домен
    D. Структура данных иерархической модели, определяется терминами: элемент данных, агрегат данных, запись, групповое отношение, база данных.

12. Определите допустимые операции над данными в иерархической модели:
    A. позволяет занести в БД новые записи
    B. изменение значений элементов предварительно извлеченной записи, ключевые значения обновляться не должны
    C. операция служит для исключения из БД некоторой записи и всех подчиненных ей.
    D. Присвоение данных атрибута сетевой модели и иерархической

13. Выберите правильные суждения:
    A. Инфологическая модель позволяет определить какие данные будут храниться в БД
    B. Системный и прикладной программист входит в состав внутренних пользователей информационной системы
    C. Запись – это агрегат, не входящий ни в какой другой агрегат. Это основная единица обработки БД.
    D. Отношение это поле БД

14. Выберите правильные суждения
    A. Реляционная база данных состоит из отношений(таблиц), которые включают в себя атрибуты(поля)
    B. Запись в таблице именуется кортежем
    C. Запись в таблице именуется атрибутом
    D. мощность отношения - количество столбцов бд

15. Выберите правильные варианты ответа для определения «Степень отношения» :
    A. это количество доменов образующих данное отношение, как правило, степень отношения в процессе жизненного цикла не меняется.
    B. это количество кортежей отношения (количество строк в таблице). В общем случае она изменяется с течением времени
    C. есть совокупность отношений содержащих информацию о предметной области.

D. количество столбцов таблицы

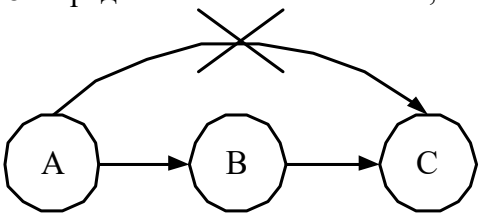
16 Выберите правильный вариант ответ для определения «Мощность отношения»:

A. это количество доменов (столбцов) образующих данное отношение, как правило, степень отношения в процессе жизненного цикла не меняется.
B. это количество кортежей отношения (количество строк в таблице). В общем случае она изменяется с течением времени
C. количество атрибутов отношения
D. отношение атрибутов и кортежей

17 Выберите правильное суждение:

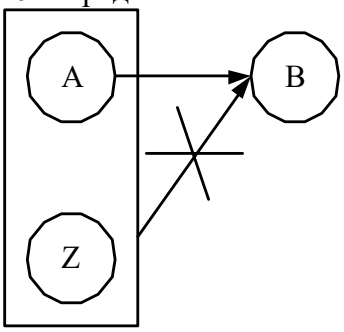
A. Говорят, что B функционально зависит от A, если для каждого значения A существует только одно связанное с ним значение B.
B. Функциональная зависимость получается приведением к получению двух отношений из одного.
C. Говорят, что B функционально зависит от A, если для каждого значения B существует только одно связанное с ним значение A.
D. Говорят, что B функционально зависит от A, если для каждого значения C существует только одно связанное с ним значение A.

18 Определите тип зависимости, изображенной на рисунке:



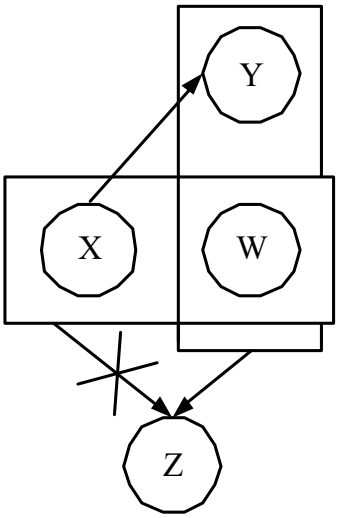
A. Транзитивная зависимость
B. Корректные, но избыточные зависимости
C. Объединение функциональных зависимостей
D. Декомпозиция функциональных зависимостей

19 Определите тип зависимости, изображенной на рисунке:



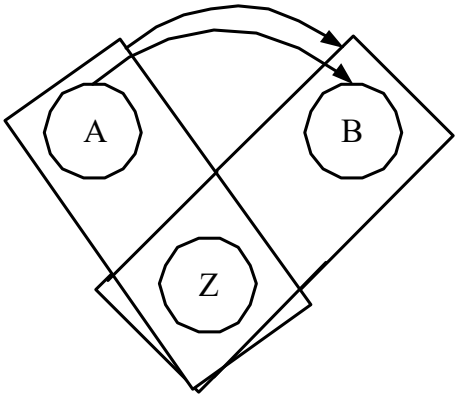
A. Корректные, но избыточные зависимости
B. Транзитивные зависимости
C. Объединение функциональных зависимостей
D. Декомпозиция функциональных зависимостей

20 Определите тип зависимости изображенной на рисунке:



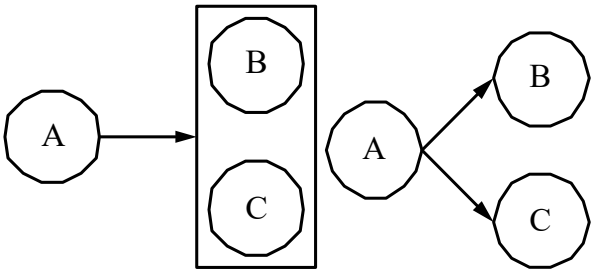
A. Псевдотранзитивность
B. Декомпозиция функциональных зависимостей
C. Объединение функциональных зависимостей
D. Корректные, но избыточные зависимости

21 Определите тип зависимости изображенной на рисунке:



A. Корректные, но избыточные зависимости
B. Транзитивные зависимости
C. Объединение функциональных зависимостей
D. Псевдотранзитивность

22 Определите тип зависимости изображенной на рисунке:

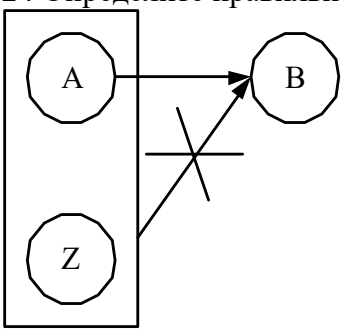


A. Декомпозиция функциональных зависимостей
B. Псевдотранзитивность
C. Объединение функциональных зависимостей
D. Корректные, но избыточные зависимости

23. Определите тип зависимости для выражения: Если A>B,C , то A>B и A>C.
A. Декомпозиция функциональных зависимостей

B. Псевдотранзитивность
C. Объединение функциональных зависимостей
D. Корректные, но избыточные зависимости

24 Определите правильное суждение для следующего изображения:



A. Если существует A> B, то зависимость A,Z>B – корректная, но избыточная.
B. Если существует A>B, то зависимость A,Z>B,Z – корректная, но избыточная
C. Если A>B и A>C, то A>B,Z
D. Если A>B и Z,A>Z то зависимость Z,B>A

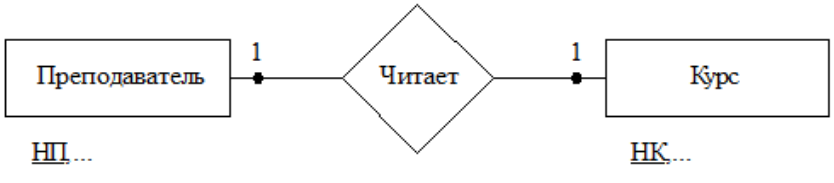
25 Выберите правильный вариант ответ для определения «Сущность»
A. определяется как некий объект, представляющий интерес для пользователей БД
B. представляет собой взаимодействие между двумя или более сущностями
C. это атрибут или набор атрибутов, значения которых однозначно определяют экземпляр сущности.
D. это атрибут или набор кортежей

26 Выберите правильный вариант ответ для определения «Связь»
A. определяется как некий объект, представляющий интерес для пользователей БД
B. представляет собой взаимодействие между двумя или более сущностями
C. есть свойство сущности
D. это атрибут или набор атрибутов, значения которых однозначно определяют экземпляр сущности.

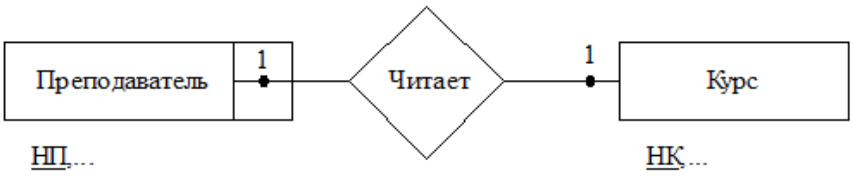
27 Выберите правильный вариант ответ для определения «Атрибут»
A. определяется как некий объект, представляющий интерес для пользователей БД
B. представляет собой взаимодействие между двумя или более сущностями
C. есть свойство сущности
D. ключевое поле

28 Выберите правильный вариант ответ для определения «Ключ сущности»
A. определяется как некий объект, представляющий интерес для пользователей БД.
B. представляет собой взаимодействие между двумя или более сущностями
C. это атрибут или набор атрибутов, значения которых однозначно определяют экземпляр сущности
D. есть свойство сущности

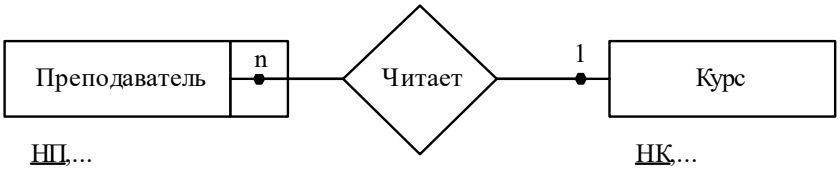
29 Выберите какой из рисунков диаграммы ER-типа соответствует связи: «Один преподаватель в обязательно читает множество дисциплин»

A. 

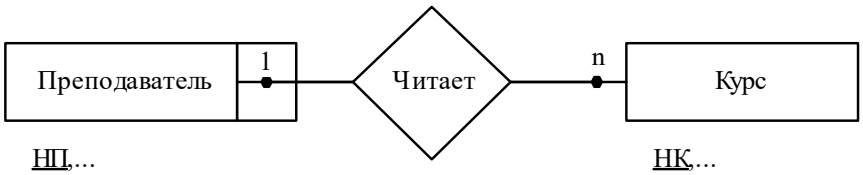
B.



C.

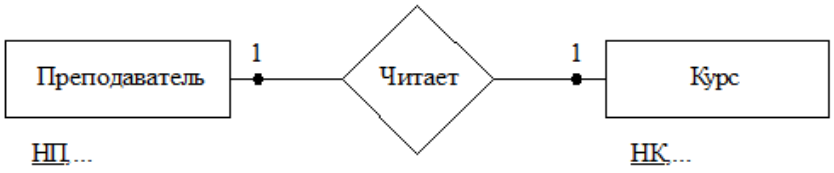


D.

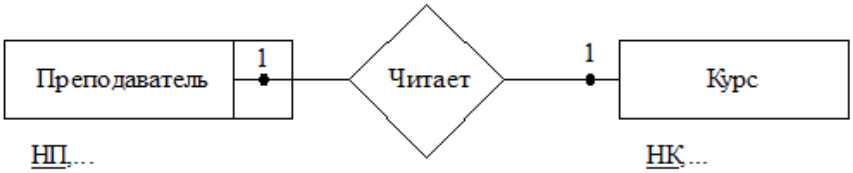


30 Выберите какой из рисунков диаграммы ER-типа соответствует связи: «один преподаватель в любом случае читает только одну дисциплину»

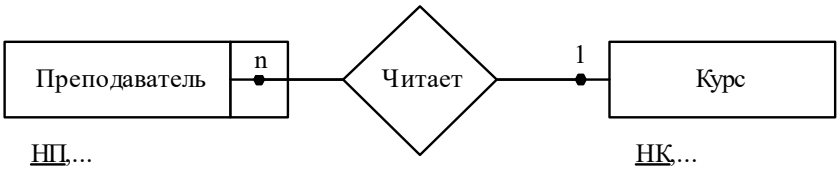
A.



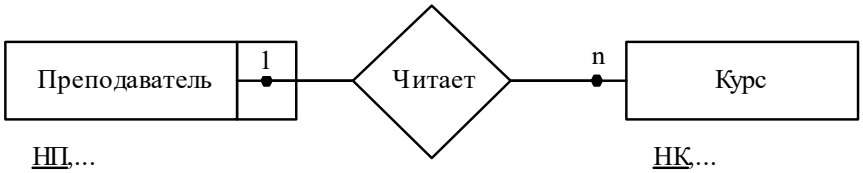
B.



C.

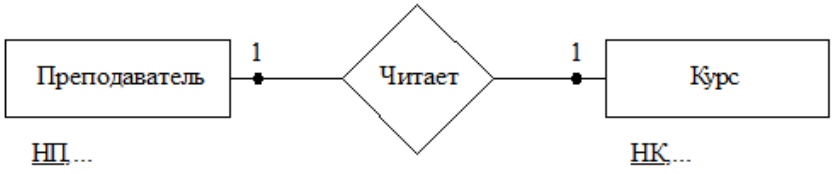


D.

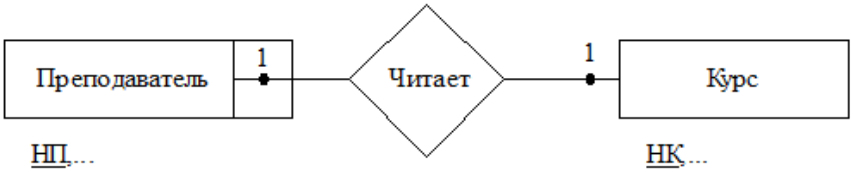


31 Выберите какой из рисунков диаграммы ER-типа соответствует связи когда: «преподаватель может не иметь читаемых дисциплин»

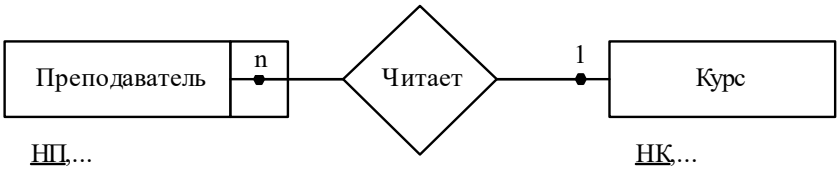
A.



B.

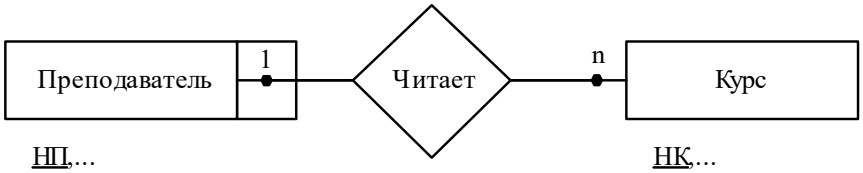


C.



НП,... НК,...

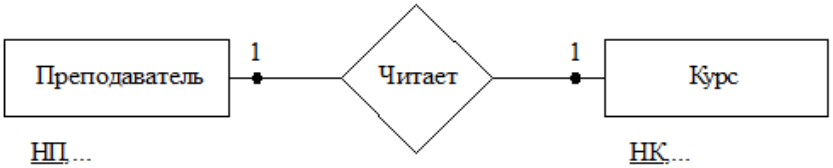
D.



НП,... НК,...

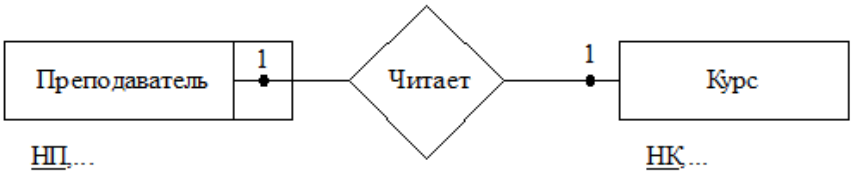
32 Выберите какой из рисунков диаграммы ER-типа соответствует связи: «преподаватель может не иметь читаемых дисциплин»

A.



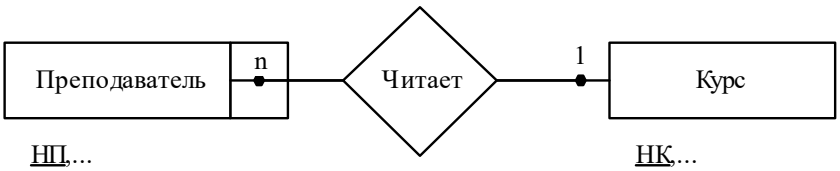
НП,... НК,...

B.



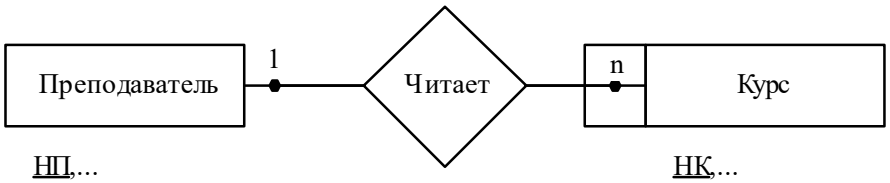
НП,... НК,...

C.



НП,... НК,...

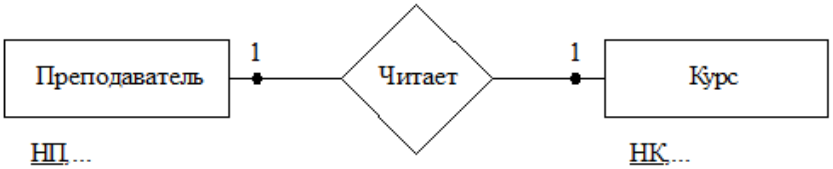
D.



НП,... НК,...

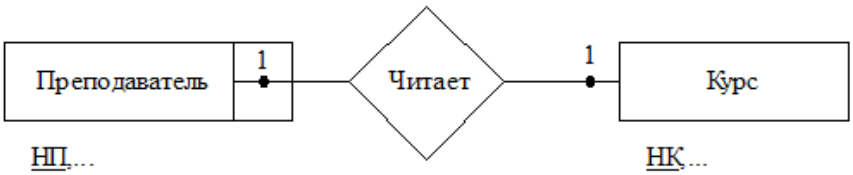
33 Выберите какие рисунки диаграммы ER-типа соответствуют связи: «курс может быть не закреплен ни за одним из преподавателей»

A.



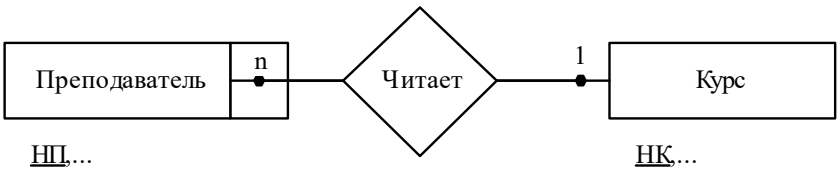
НП,... НК,...

B.

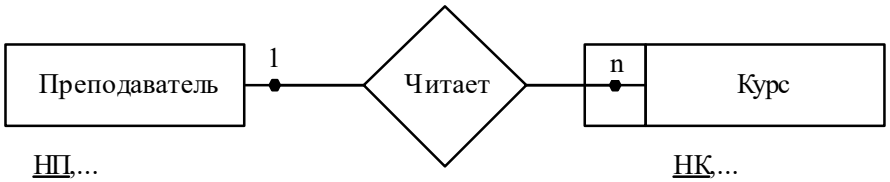


НП,... НК,...

C.

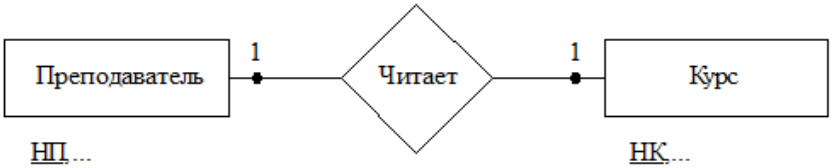


НП,... НК,...

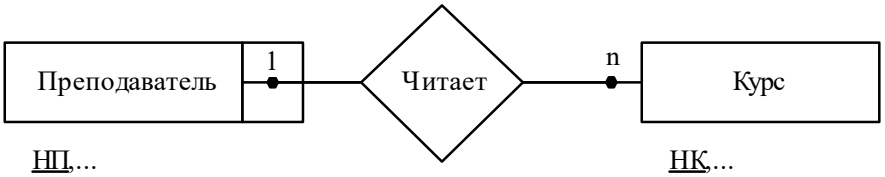


D. НП,... НК,...

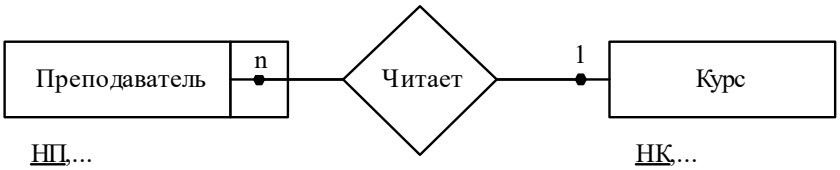
34. Выберите какой из рисунков диаграммы ER-типа соответствует связи: «курс может читаться множеством преподавателей»



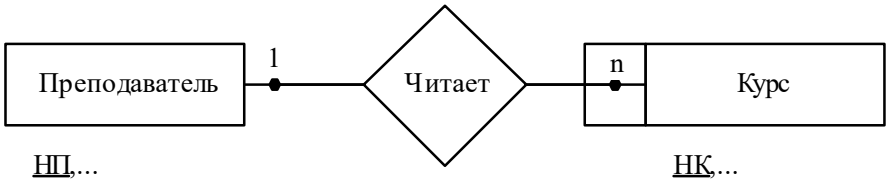
A. НП,... НК,...



B. НП,... НК,...

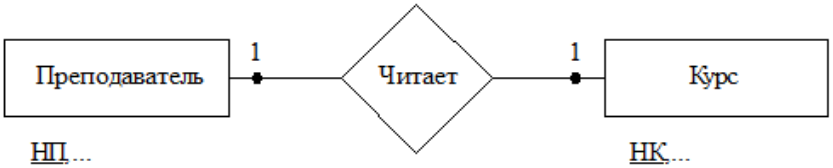


C. НП,... НК,...

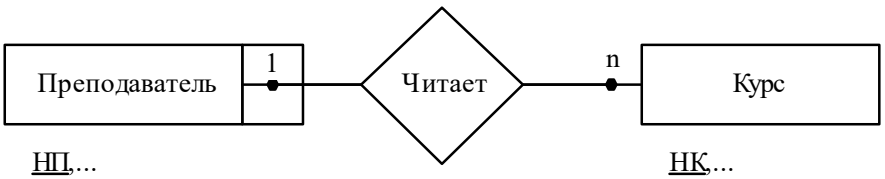


D. НП,... НК,...

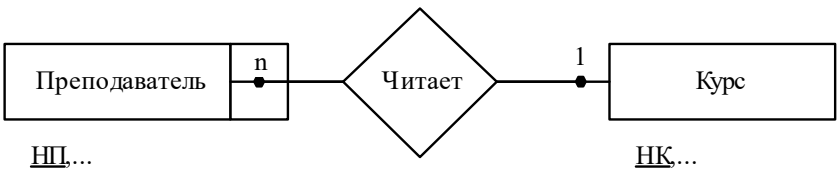
35 Выберите какой из рисунков диаграммы ER-типа соответствует связи: «курс может читаться множеством преподавателей и при этом курс может ни кем не читаться»



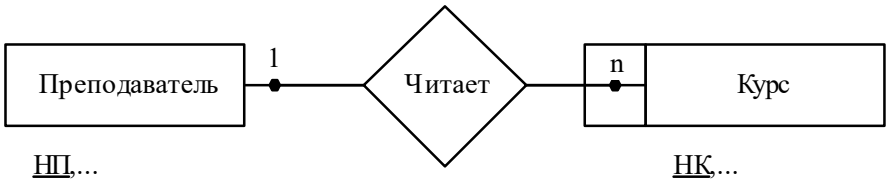
A. НП,... НК,...



B. НП,... НК,...

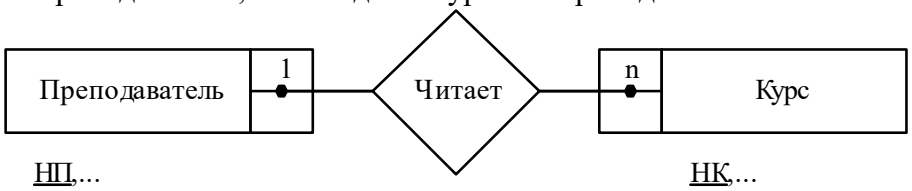


C. НП,... НК,...

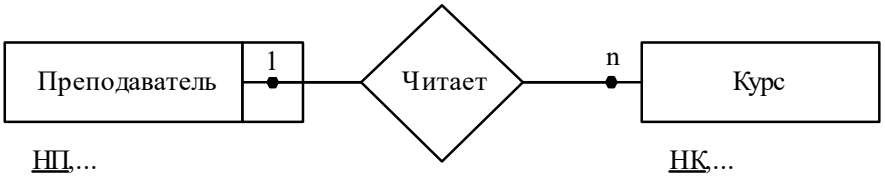


D. НП,... НК,...

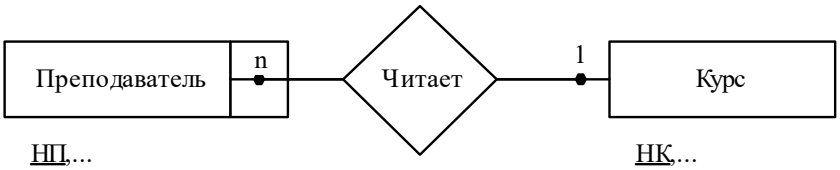
36 Выберите какие рисунки диаграммы ER-типа соответствуют связи: «каждый преподаватель читает множество курсов , но при этом каждый курс может читать только один преподаватель, а свободных курсов и преподавателей нет»



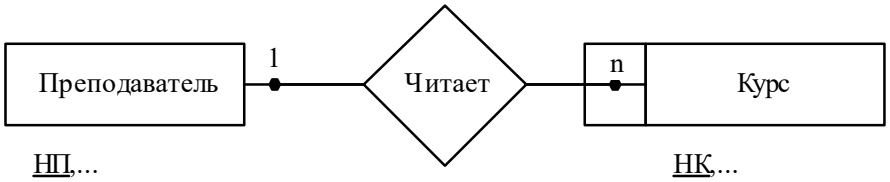
A. НП,... НК,...



B. НП,... НК,...

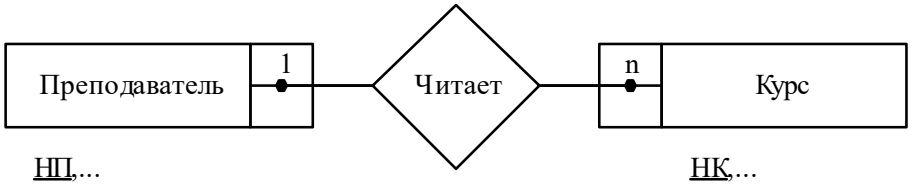


C. НП,... НК,...

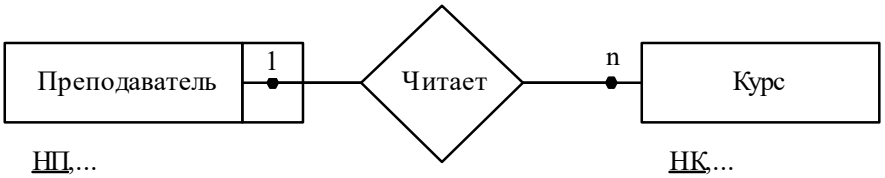


D. НП,... НК,...

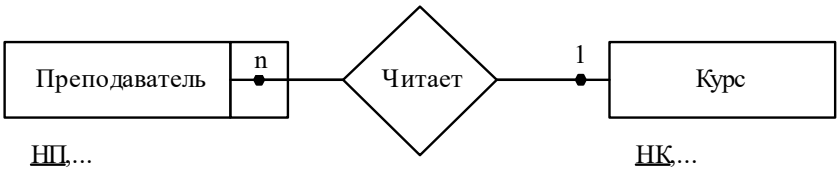
37 Выберите какие рисунки диаграммы ER-типа соответствуют связи: «курс может ни кем не читаться»



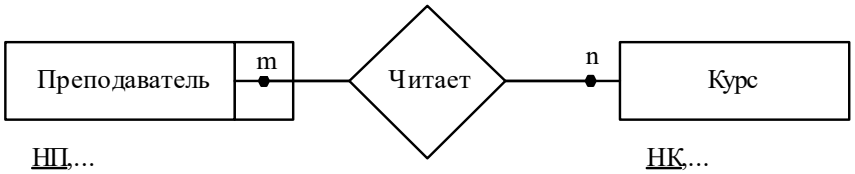
A. НП,... НК,...



B. НП,... НК,...



C. НП,... НК,...



D. НП,... НК,...

38 Выберите правильное суждение:

A. Маска ввода задается для полей таблицы и применяется для контроля вводимых данных и упрощения ввода пользователем
B. Сортировка - это упорядочивание данных в таблице по какому либо критерию
C. Сортировка - это поиск данных в таблице
D. Декомпозиция получается приведением к получению двух атрибутов из одного.

39 Определите, каких запросов не существует

A. Запрос на модификацию данных
B. Запрос на выборку(поиск) данных
C. Запрос удаление данных
D. запрос с вычисляемыми кортежами

Перечень вопросов для теста (вариант 2):

1. Определите, каких баз данных не существует:
A. Сетевых
B. Иерархических
C. Объектно-табличных
D. Реляционных

2. Определите, в какую модель базы данных входят понятия кортежа и домена:
A. Сетевых
B. Иерархических
C. Табличных
D. Реляционных

3. Выберите правильные варианты ответ для определения «Кортеж»:
A. Строка таблицы
B. Запись
C. Поле таблицы
D. Группа записей

4. Выберите правильные варианты ответ для определения «Мощность отношения»:
A. Количество столбцов
B. Количество полей
C. Количество записей
D. Количество кортежей

5. Определите, что определяется как некий объект в реляционной базе данных, представляющий интерес для пользователей.
A. Ключ
B. Атрибут
C. Связь
D. Сущность

6. Выберите, что представляет собой взаимодействие между двумя или более сущностями
A. Ключ
B. Атрибут
C. Связь
D. Сущность

7. Определите, что подразумевается под свойством сущности
A. Ключ
B. Атрибут
C. Связь
D. Сущность

8. Определите, какой элемент базы данных выступает в качестве атрибута или набора атрибутов, значения которых однозначно определяют экземпляр сущности
A. Ключ
B. Атрибут
C. Связь
D. Сущность

9. Определите варианты ответов, позволяющтих описать схемы связи таблиц
A. Диаграмма Er-типа
B. Диаграмма Er-экземпляров
C. Схема данных
D. Инфологическая модель

10. Определите какая модель позволяет определить, какие данные будут храниться в будущей таблице
A. Схема данных
B. Диаграмма er-типа
C. Инфологическая модель
D. Даталогическая модель

11. Определите каких систем взаимодействия с базой данных не существует
A. Файл-сервер
B. Топология «Звезда»
C. Web-интерфейс
D. Запись-клиент

12.Определите, какие пункты не входят в задачи администратора БД
A. Формирование сообщений об ошибках
B. Резервное копирование БД
C. Программирование клиентского приложения
D. Проектирование инфологической модели

13. Определите объект, который существует только в пределах инфраструктуры конкретной таблицы или представления
A. Ключ сущности
B. Сущность-связь
C. Кластеризованные и некластеризованные индексы
D. Атрибут

14.Определите, какие пункты не входят в задачи администратора БД
A. Формирование сообщений об ошибках
B. Резервное копирование БД
C. Программирование клиентского приложения
D. Проектирование инфологической модели

15. Определите, что представляет из себя своего рода виртуальную таблицу, которая не

содержит данных в MS SQL Server
A. Индексы
B. Файловая группа
C. Триггер
D. Представление

16.Определите, какой SQL оператор используется для создания схемы базы данных
A. Delete
B. Select
C. CREATE Table
D. CREATE SCHEMA

17. Определите, какой SQL оператор используется для удаления домена:
A. Delete
B. Delete domain
C. CREATE Domain
D. Drop domain

18.Выберите правильные утверждения:
A. Триггер является представлением SQL запроса в БД
B. Резервное копирование БД не входит в обязанности администратора базы данных
C. Оператор AND означает, что общий предикат будет истинным только тогда, когда условия, связанные по “AND”, будут истинны
D. В языке SQL приоритет операций сравнения выше приоритета логических операций.

19.Выберите правильные утверждения
A. Сущность-связь проектируется только по просьбе заказчика клиентского приложения
B. Резервное копирование БД не входит в обязанности администратора базы данных
C. Оператор NOT означает, что общий предикат будет истинным, когда условие, перед которым стоит этот оператор, будет ложным.
D. В языке SQL приоритет операций сравнения выше приоритета логических операций.

20.Определите, какие операторы позволяют манипулировать данными:
A. If
B. For
C. Insert
D. CREATE Domain

21. Определите, оператор позволяющий манипулировать данными:
A. Domain
B. And
C. Create
D. Select

22. Определите, оператор позволяющий осуществить выборку данных
A. Domain
B. And
C. Rollback
D. Select

23. Определите оператор, результатом выполнения которого является таблица
A. insert

B. Update
C. Create
D. Select

24. Определите операторы, результатом выполнения которых является таблица
A. Domain
B. And
C. Create table
D. Select

25. Определите, что используется для того, что бы упорядочить строки в результате применения запроса Select
A. Нет правильного варианта ответа
B. Where…
C. Select * from …
D. Order by…

26. Определите, что не входит в групповые функции
A. MIN
B. COUNT
C. AVG
D. Групповые отношения

27. Определите, что не входит в групповые функции
A. MAX
B. SUM
C. AVG
D. OR

28. Определите, что может задавать условие поиска для групп в операторе Select:
A. Where
B. Select
C. Order by
D. Having

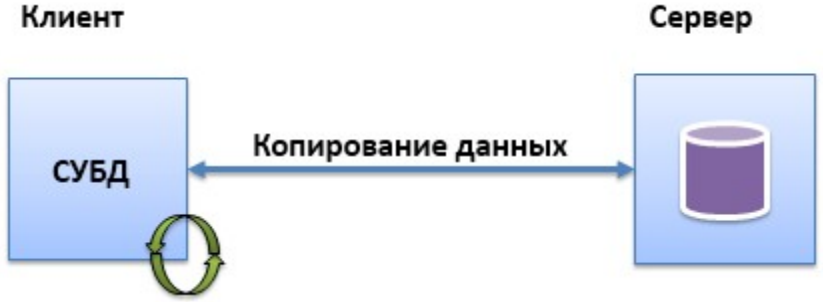
29. Определите оператор языка SQL, который является реализацией операции соединения реляционной алгебры
A. OR, And, not
B. Select
C. Delete
D. Join

30. Определите, отличительными особенностями операции соединения:
A. В схему таблицы-результата не входят столбцы обеих исходных таблиц (таблиц-операндов);
B. В резервном копировании таблицы
C. Таблицы объединяются
D. В схему таблицы-результата входят столбцы обеих исходных таблиц (таблиц-операндов);

31. Выберите правильные утверждения:
A. Select –создает представления физически

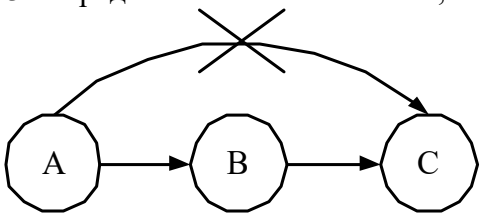
B. Select –не создает представления физически
C. Having – позволяет определять условие поиска в операторе Select
D. Оператор внутреннего соединения INNER JOIN соединяет две таблицы. Порядок не важен

32. Определите, какая архитектура изображена на рисунке:



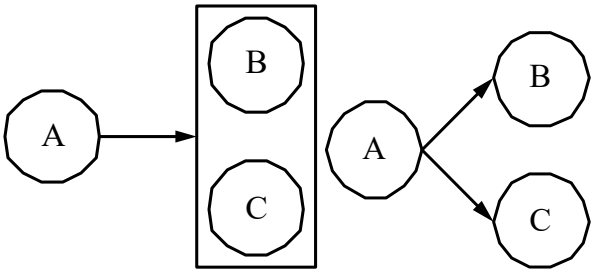
a. Файл-сервер
b. Клиент-сервер
c. Web-сервер
d. Сервер-клиент

33. Определите тип зависимости, изображенной на рисунке:



A. Транзитивная зависимость
B. Корректные, но избыточные зависимости
C. Объединение функциональных зависимостей
D. Декомпозиция функциональных зависимостей

34. Определите тип зависимости изображенной на рисунке:

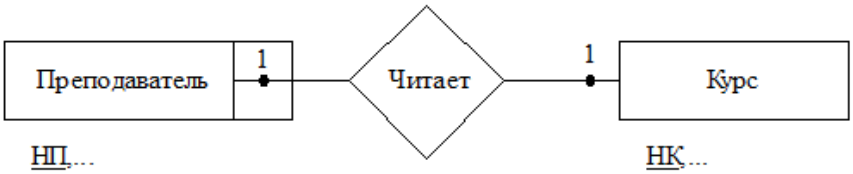


A. Декомпозиция функциональных зависимостей
B. Псевдотранзитивность
C. Объединение функциональных зависимостей
D. Корректные, но избыточные зависимости

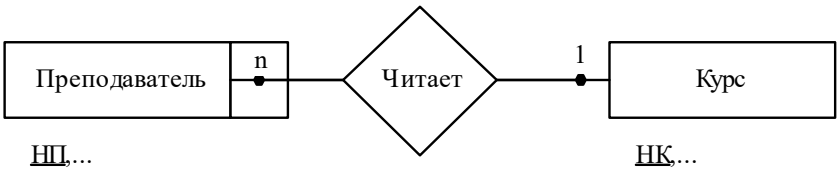
35. Выберите какой из рисунков диаграммы ER-типа соответствует связи: «Один преподаватель обязательно читает множество дисциплин»

A.

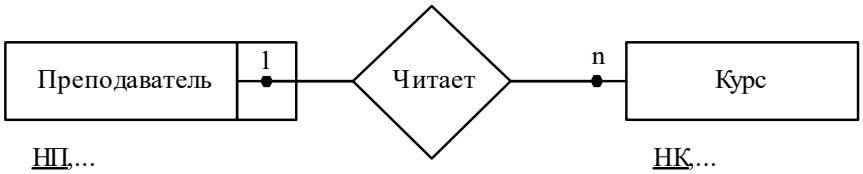
B.



C.

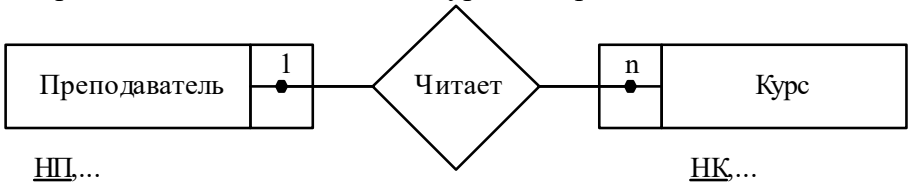


D.

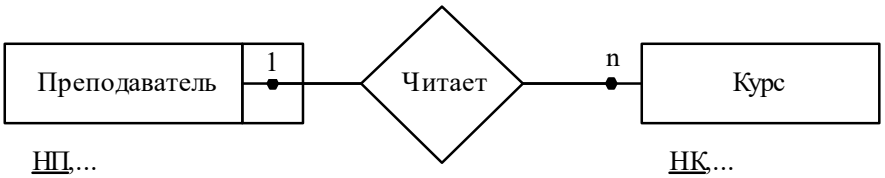


36. Выберите какой из рисунков диаграммы ER-типа соответствуют связи: «каждый преподаватель читает множество курсов , но при этом каждый курс может читать только один преподаватель, а свободных курсов и преподавателей нет»

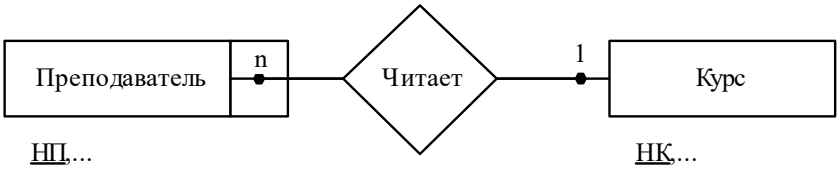
A.



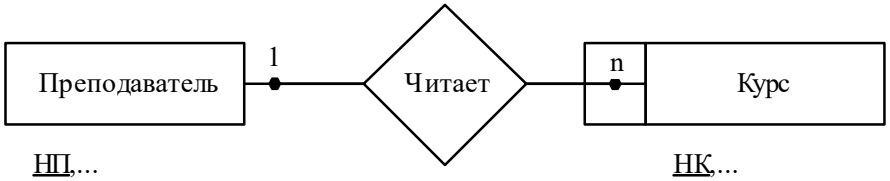
B.



C.

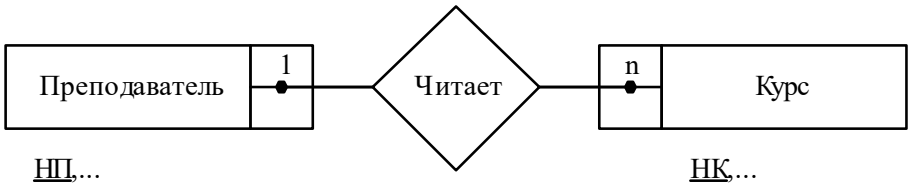


D.



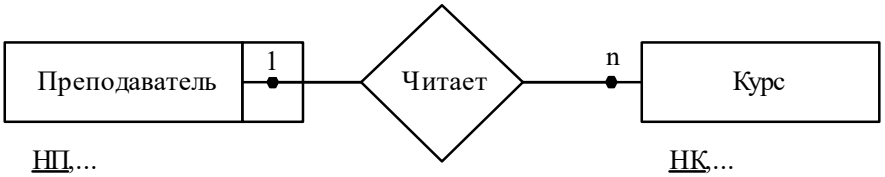
37 Выберите какие рисунки диаграммы ER-типа соответствуют связи: «курс может ни кем не читаться»

A.



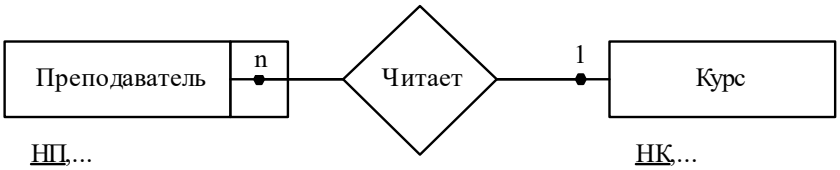
НП,... НК,...

B.



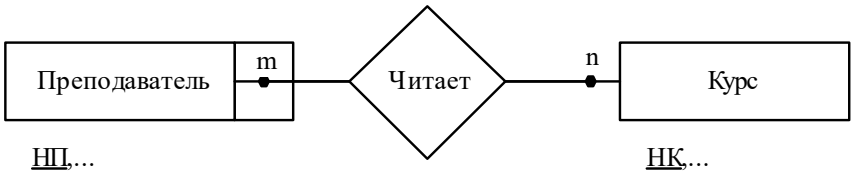
НП,... НК,...

C.



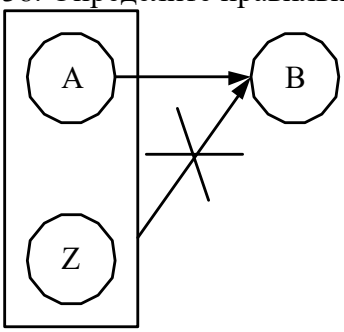
НП,... НК,...

D.



НП,... НК,...

38. Определите правильное суждение для следующего изображения:



A. Если существует A> B, то зависимость A,Z>B – корректная, но избыточная.
B. Если существует A>B, то зависимость A,Z>B,Z – корректная, но избыточная
C. Если A>B и A>C, то A>B,Z
D. Если A>B и Z,A>Z то зависимость Z,B>A

39. Определите, какая архитектура изображена на рисунке:

a. клиент-сервер
b. файл-сервер
c. веб приложение
d. веб-архитектура

Критерии оценки результатов:
5 баллов – задание выполнено правильно, без ошибок
4 балла – допущены 1-3 ошибки,
3 балла – допущены 3-5 ошибок,
2 бала – допущены 5 и более ошибок

Критерии оценки:
Оценка за контроль ключевых компетенций учащихся производится по пятибалльной системе. При выполнении заданий ставится отметка:
«3» - за 40-50% правильно выполненных заданий,
«4» - за 50-65% правильно выполненных заданий,
«5» - за правильное выполнение более 65% заданий.

Практическое занятие №1

Нормализация реляционной БД, освоение принципов проектирования БД

Цель работы: выработать практические навыки моделирования предметной области и построении ER-модели данных, закрепить технологию проектирования БД, закрепить основные понятия теории реляционных баз данных, освоить технологию построения ER-диаграмм, научиться получать реляционные БД из ER-диаграмм

Сущность-связь

Работа с базой данных начинается с построения модели. Наиболее распространенной является ER-модель (entity-relationship model) - модель "Сущность-связь". Для "ручного" построения ER-модели на практике будем использовать простую систему обозначений, предложенную Питером Ченом (обозначения, встречающиеся в разных источниках, могут отличаться от нижеприведенных):

Рисунок 1.1. Сущность - связь

Первичный ключ - атрибут или группа атрибутов, однозначно идентифицирующих объект. Первичный ключ может состоять из нескольких атрибутов, тогда подчеркивается каждый из них.

Связи между объектами могут быть 3-х типов: *Один - к одному*. Этот тип связи означает, что каждому объекту первого вида соответствует не более одного объекта второго вида, и наоборот. Например: сотрудник может руководить только одним отделом, и у каждого отдела есть только один руководитель. *Один - ко многим.* Этот тип связи означает, что каждому объекту первого вида может соответствовать более одного объекта второго вида, но каждому объекту второго вида соответствует не более одного объекта первого вида. Например: в каждом отделе может быть множество сотрудников, но каждый сотрудник работает только в одном отделе. *Многие - ко многим.* Этот тип связи означает, что каждому объекту первого вида может соответствовать более одного объекта второго

вида, и наоборот. Например: каждый счет может включать множество товаров, и каждый товар может входить в разные счета.

Реляционная структура данных

В конце 60-х годов появились работы, в которых обсуждались возможности применения различных табличных даталогических моделей данных. Наиболее значительной из них была статья сотрудника фирмы IBM д-ра Эдварда Кодда (Codd E.F., A Relational Model of Data for Large Shared Data Banks. CACM 13: 6, June 1970), где впервые был применен термин «реляционная модель данных».

Будучи математиком по образованию, Э. Кодд предложил использовать для обработки данных аппарат теории множеств (объединение, пересечение, разность, декартово произведение). Он показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как *отношение* – relation.

Наименьшая единица данных реляционной модели – это отдельное атомарное (неразложимое) для данной модели значение данных.

Так, в одной предметной области фамилия, имя и отчество могут рассматриваться как единое значение, а в другой – как три различных значения.

Доменом называется множество атомарных значений одного и того же типа. Так, на рис. домен пунктов отправления (назначения) – множество названий населенных пунктов, а домен номеров рейса – множество целых положительных чисел.

Таблица 1.1. Рейс

Номер рейса	Дни недели	Пункт отправления	Время вылета	Пункт назначения	Время прибытия	Тип самолета	Стоимость билета
138	2_4_7	Баку	21.12	Москва	0.52	ИЛ-86	115.00
57	3_6	Ереван	7.20	Киев	9.25	ТУ-154	92.00
1234	2_6	Казань	22.40	Баку	23.50	ТУ-134	73.50
242	1 по 7	Киев	14.10	Москва	16.15	ТУ-154	57.00
86	2_3_5	Минск	10.50	Сочи	13.06	ИЛ-86	78.50
137	1_3_6	Москва	15.17	Баку	18.44	ИЛ-86	115.00
241	1 по 7	Москва	9.05	Киев	11.05	ТУ-154	57.00
577	1_3_5	Рига	21.53	Таллин	22.57	АН-24	21.50
78	3_6	Сочи	18.25	Баку	20.12	ТУ-134	44.00
578	2_4_6	Таллин	6.30	Рига	7.37	АН-24	21.50

Смысл доменов состоит в следующем. Если значения двух атрибутов берутся из одного и того же домена, то, вероятно, имеют смысл сравнения, использующие эти два атрибута (например, для организации транзитного рейса можно дать запрос «Выдать рейсы, в которых время вылета из Москвы в Сочи больше времени прибытия из Архангельска в Москву»). Если же значения двух атрибутов берутся из различных доменов, то их сравнение, вероятно, лишено смысла: стоит ли сравнивать номер рейса со стоимостью билета?

Отношение на доменах D1, D2, ..., Dn состоит из заголовка и тела. На рис. 3.4 приведен пример отношения для расписания движения самолетов (таблица 1). Ai - атрибуты, Vi - значения атрибутов.

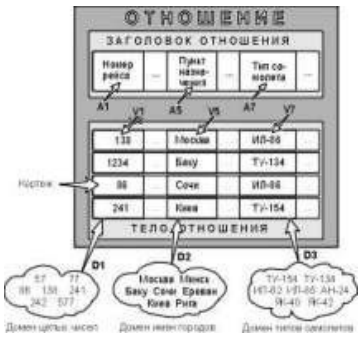


Рисунок 1.2. Отношение с математической точки зрения

Заголовок отношения состоит из такого фиксированного множества атрибутов A1, A2, ..., An, что существует взаимно однозначное соответствие между этими атрибутами Ai и определяющими их доменами Di (i=1,2,...,n).

Тело отношения состоит из меняющегося во времени множества кортежей, где каждый кортеж состоит в свою очередь из множества пар атрибут-значение (Ai:Vi), (i=1,2,...,n), по одной такой паре для каждого атрибута Ai в заголовке.

Для любой заданной пары атрибут-значение (Ai:Vi) Vi является значением из единственного домена Di, который связан с атрибутом Ai.

Степень отношения – это число его атрибутов.

Отношение степени один называют унарным, степени два – бинарным, степени три – тернарным, ..., а степени n – n-арным. Степень отношения «Рейс» (таблица 1) равна 8.

Кардинальное число или мощность отношения – это число его кортежей.

Мощность отношения «Рейс» равна 10. Кардинальное число отношения изменяется во времени в отличие от его степени.

Поскольку отношение – это множество, а множества по определению не содержат совпадающих элементов, то никакие два кортежа отношения не могут быть дубликатами друг друга в любой произвольно заданный момент времени.

Пусть R – отношение с атрибутами A1, A2, ..., An. Говорят, что множество атрибутов K=(Ai, Aj, ..., Ak) отношения R является возможным ключом R тогда и только тогда, когда удовлетворяются два независимых от времени условия:

Уникальность: в произвольный заданный момент времени никакие два различных кортежа R не имеют одного и того же значения для Ai, Aj, ..., Ak.

Минимальность: ни один из атрибутов Ai, Aj, ..., Ak не может быть исключен из K без нарушения уникальности.

Каждое отношение обладает хотя бы одним возможным ключом, поскольку по меньшей мере комбинация всех его атрибутов удовлетворяет условию уникальности. Один из возможных ключей (выбранный произвольным образом) принимается за его первичный ключ. Остальные возможные ключи, если они есть, называются альтернативными ключами.

Вышеупомянутые и некоторые другие математические понятия явились теоретической базой для создания реляционных СУБД, разработки соответствующих языковых средств и программных систем, обеспечивающих их высокую производительность, и создания основ теории проектирования баз данных.

Также на практике широко используются неформальные эквиваленты этих понятий: Отношение – Таблица, Кортеж – Строка таблицы или Запись, Атрибут – Столбец Таблицы или Поле.

При этом принимается, что «запись» означает «экземпляр записи», а «поле» означает «имя и тип поля».

Задание для практической работы

Задание 1. Проектирование реляционных Баз Данных.

Вариант 1.

Построить реляционную таблицу Базы Данных имен родственников студентов

вашей группы, содержащую данные об именах родителей, братьев и сестер студентов.

Вариант 2.

Построить реляционную таблицу Базы Данных домов, где живут студенты вашей группы, содержащую данные о районе расположения дома, количестве этажей в нем и номере этажа, где живет студент.

Вариант 3.

Построить реляционную таблицу Базы Данных дней рождения студентов вашей группы, содержащую данные о дате рождения, знаке зодиака и годе по Китайскому календарю.

Вариант 4.

Построить реляционную таблицу Базы Данных Сотовых телефонов студентов вашей группы, содержащую данные о модели телефона, типе корпуса, операторе.

Технология выполнения работы и оформление отчета

1.1. Придумайте заголовок отношения и запишите его в отчет.

1.2. Определите атрибуты отношения. Начертите сетку таблицы в отчет и занесите в нее атрибуты.

1.3. Опросите студентов вашей группы и занесите полученные данные в таблицу.

1.4. На чертеже таблицы укажите чему соответствуют понятия: Заголовок отношения, тело отношения, атрибут отношения, кортеж отношения.

1.5. Определите и запишите в отчет степень отношения и мощность отношения.

1.6. Дайте определение первичного ключа. Укажите Первичный ключ получившегося отношения

1.7. Докажите, что у вас получилась реляционная таблица, для этого укажите типы данных всех атрибутов.

Задание 2. Проектирование Баз Данных. ER-диаграммы.

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

Вариант 1.

Описание предметной области (Ресторан).

Посетители ресторана обслуживаются за столиками. За одним столом может располагаться не более 4 посетителей, каждый из которых может сделать заказ тех или иных блюд. Столики обслуживают официанты. У одного официанта в обслуживании несколько столов.

Задачи для БД:

- Есть ли свободные столы?
- Сколько посетителей обслужил официант за смену?
- Сколько каких блюд было реализовано?

Вариант 2.

Описание предметной области (Колледж).

Студенты колледжа объединены в группы. Набор дисциплин, изучаемых студентом, зависит от номера группы в которой он учится. Преподаватели читают дисциплины и выставляют зачеты студентам. Один преподаватель может читать несколько дисциплин, но каждую дисциплину ведет один преподаватель.

Задачи для БД:

- Какие дисциплины изучает студент?
- Какая оценка у студента по данной дисциплине?
- Кто выставил эту оценку?

Вариант3.

Описание предметной области (Театральная касса).

В театральной кассе продаются билеты на спектакли. Стоимость билета зависит от ряда, театра и спектакля. Каждый день в театре может идти не более одного спектакля. Спектакль характеризуется названием и автором. Каждый покупатель может купить

сколько угодно билетов на любые спектакли.

Задачи для БД:

- Какие спектакли идут в определенный день?
- Есть ли билеты на конкретный спектакль?
- Сколько стоит конкретный билет?

Вариант 4.

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.

Задачи для БД:

- Какие грузовики свободны?
- Какой заказчик сделал самый дорогой заказ?
- Какой грузовик выполнил наибольшее количество заказов? Технология выполнения работы 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.

2.Получение реляционной схемы из ER-диаграммы.

2.1. Каждая сущность превращается в таблицу. Имя сущности – имя таблицы. Набор всех таблиц – БД. Вспомните, что такое схема БД. Запишите схему вашей БД в отчет.

2.2. Зарисуйте все полученные таблицы с их заголовками и названиями столбцов. Выделите потенциальные и внешние ключи (если есть) для каждой таблицы. Укажите столбцы, допускающие неопределенные значения.

Докажите, что полученные отношения находятся в Первой нормальной форме.

Практическое занятие № 3 Преобразование реляционной БД в сущности, связи.

Цель работы: выработать практические навыки моделирования предметной области и построения различных видов модели баз данных

Нормализация, функциональные и многозначные зависимости

Нормализация – это разбиение таблицы на две или более, обладающих лучшими свойствами при включении, изменении и удалении данных. Окончательная цель нормализации сводится к получению такого проекта базы данных, в котором каждый факт появляется лишь в одном месте, т.е. исключена избыточность информации. Это делается не столько с целью экономии памяти, сколько для исключения возможной противоречивости хранимых данных.

Каждая таблица в реляционной БД удовлетворяет условию, в соответствии с которым в позиции на пересечении каждой строки и столбца таблицы всегда находится единственное атомарное значение, и никогда не может быть множества таких значений. Любая таблица, удовлетворяющая этому условию, называется нормализованной.

Фактически, ненормализованные таблицы, т.е. таблицы, содержащие повторяющиеся группы, даже не допускаются в реляционной БД.

Всякая нормализованная таблица автоматически считается таблицей в первой нормальной форме, сокращенно 1НФ. Таким образом, строго говоря, "нормализованная" и "находящаяся в 1НФ" означают одно и то же. Однако на практике термин "нормализованная" часто используется в более узком смысле – "полностью нормализованная", который означает, что в проекте не нарушаются никакие принципы нормализации.

Теперь в дополнение к 1НФ можно определить дальнейшие уровни нормализации – вторую нормальную форму (2НФ), третью нормальную форму (3НФ) и т.д.

По существу, таблица находится в 2НФ, если она находится в 1НФ и удовлетворяет, кроме того, некоторому дополнительному условию, суть которого будет рассмотрена ниже. Таблица находится в 3НФ, если она находится в 2НФ и, помимо этого, удовлетворяет еще другому дополнительному условию и т.д.

Таким образом, каждая нормальная форма является в некотором смысле более ограниченной, но и более желательной, чем предшествующая. Это связано с тем, что "(N+1)-я нормальная форма" не обладает некоторыми непривлекательными особенностями, свойственным "N-й нормальной форме". Общий смысл дополнительного условия, налагаемого на (N+1)-ю нормальную форму по отношению к N-й нормальной форме, состоит в исключении этих непривлекательных особенностей.

За время развития технологии проектирования реляционных БД были выделены следующие нормальные формы:

- первая нормальная форма (1NF); - вторая нормальная форма (2NF);
- третья нормальная форма (3NF);
- нормальная форма Бойса-Кодда (BCNF);
- четвертая нормальная форма (4NF);
- пятая нормальная форма, или нормальная форма проекции-соединения (5NF).

Обычно на практике применение находят только первые три нормальные формы.

Теория нормализации основывается на наличии той или иной зависимости между полями таблицы. Определены два вида таких зависимостей: функциональные и многозначные.

Определение. Функциональная зависимость. Поле B таблицы функционально зависит от поля A той же таблицы в том и только в том случае, когда в любой заданный момент времени для каждого из различных значений поля A обязательно существует только одно из различных значений поля B. Отметим, что здесь допускается, что поля A и B могут быть составными.

Другими словами, в отношении R атрибут Y функционально зависит от атрибута X в том и только в том случае, если каждому значению X соответствует одно значение Y.

Схематично функциональную зависимость атрибута Y от атрибута X изображают гак:

R.X -> R.Y R(X -> Y). ФЗ(X -> Y) Пример 1.

1. В таблице Блюда (б.д. Пансион) поля Блюдо и В функционально зависят от ключа БЛ.
2. Таблица Поставщики вида:

Таблица 2.1. Поставщики

Поставщик	Статус	Город	Страна	Адрес	Телефон

В таблице Поставщики поле Страна функционально зависит от составного ключа (Поставщик, Город). Однако последняя зависимость не является функционально полной, так как Страна функционально зависит и от части ключа – поля Город. Отсюда определение:

Определение. Полная функциональная зависимость. Поле B находится в полной

функциональной зависимости от составного поля А, если оно функционально зависит от А и не зависит функционально от любого подмножества поля А. Пример нормализации

Постановка задачи. Дано отношение.

1) определить первичный ключ отношения и все функциональные зависимости отношения;

2) привести отношение к 3НФ, указать первичные и внешние ключи полученных отношений, построить схему "Таблица-Связь".

R = {НаименованиеЭмитента, ТипЦБ, ДатаЭмиссии, НоминальнаяСтоимость}.

Таблица 2.2. Эмитенты

Наименование Эмитента	ТипЦБ	Дата Эмиссии	Номинальная Стоимость
ОАО ―КрАЗ‖	акция обыкновенная	23.06.1999	100 руб.
ОАО ―КрАЗ‖	акция обыкновенная	23.06.1999	200 руб.
ТОО ―Искра‖	акция привилегирванная	20.06.1999	500 руб.
ТОО ―Искра‖	акция привилегированная	23.06.1999	500 руб.

Решение

1. Функциональные зависимости:

<ДатаЭмиссии, НоминальнаяСтоимость> -> <НаименованиеЭмитента, ТипЦБ>
ТипЦБ ->НаименованиеЭмитента
Первичный ключ отношения R состоит из двух атрибутов:
<ДатаЭмиссии, НоминальнаяСтоимость>.

Таким образом, существует функциональная зависимость между неключевыми атрибутами отношения R, т.е. отношение R не находится в 3НФ.

2. Приведение отношения R к 3НФ состоит в декомпозиции (разбиении отношения R на два отношения):

R1= {НаименованиеЭмитента,
ТипЦБ}, где Функциональные
зависимости: ТипЦБ ->
НаименованиеЭмитента, первичный
ключ – атрибут ТипЦБ,
R2={ТипЦБ, ДатаЭмиссии, НоминальнаяСтоимость},
Функциональные зависимости:
<ДатаЭмиссии, НоминальнаяСтоимость> ->ТипЦБ,
составной первичный ключ:
<ДатаЭмиссии, НоминальнаяСтоимость>,
внешний ключ:
ТипЦБ.

3. Схема "Таблица-Связь":

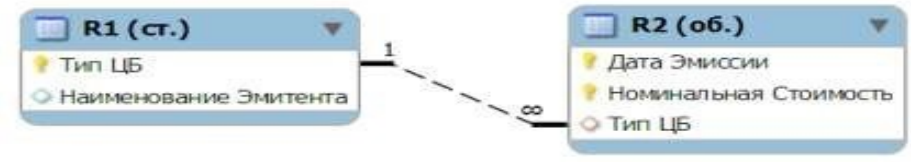


Рисунок 2.1. Схема «Таблица – связь»

Задание для практической работы

Дан фрагмент отношения (таблицы). Предполагается, что функциональные зависимости, имеющиеся во фрагменте, распространяются на все отношение (таблицу). Для вашего варианта:

1. Определить первичный ключ отношения и все функциональные зависимости отношения.
2. Привести отношение к 3НФ, указать первичные и внешние ключи полученных отношений, построить схему "Таблица - Связь".

Индивидуальные задания к практической работе

Вариант 1.

Область	Тип	C_2N	Количество
А	С	27	5
А	С	27	6
Б	Д	26	7
Б	Д	27	7

Вариант 2.

Наименование	Свойство	Артикул	Количество
А	С	27	9
А	Д	27	9
Б	С	27	9
Б	Д	25	9

Вариант 3.

Тип	Характеристика	Группа	Количество
А	С	13	100
А	Д	13	100
Б	С	13	200
Б	Д	13	100

Вариант 4.

НаименованиеЭмитента	ТипЦБ	ДатаЭмиссии
ОАО ―КрАЗ‖	акция привилегированная	20.06.1999
ОАО ―КрАЗ‖	акция обыкновенная	20.06.1999
ЗАО ―Агат‖	акция привилегированная	23.06.1999
ТОО ―Искра‖	акция привилегированная	20.06.1999

Вариант 5.

НаименованиеВуза	Команда	Приз
КрасГУ	МатФак	Торт
КГТУ	ЭконФак	Арбуз
СибГТУ	МатФак	Торт
НГУ	ЭконФак	Бананы

Вариант 6.

НаименованиеВуза	Адрес	Команда
КрасГУ	Красноярск, пр-т Свободный,79	МатФак
КрасГУ	Красноярск, пр-т Свободный,79	ЭконФак
КГТУ	Красноярск, ул.Киренского,26	ЭконФак
КГТУ эк. институт	Красноярск, ул.Киренского,26	ФизФак
НГУ	Новосибирск, ул. Пирогова, 2	МатФак
НГУ	Новосибирск, ул. Пирогова, 2	ФизФак

Вариант 7.

НаименованиеВуза	Команда	Приз
КрасГУ	МатФак	Торт
КрасГУ	ЭконФак	Торт
НГУ	ФилФак	Арбуз
НГУ	МатФак	Бананы

Вариант 8.

НаименованиеЭмитента	ТипЦБ	ДатаЭмиссии
ОАО ―КрАЗ‖	акция привилегированная	24.06.1999
ЗАО ―Сибириада‖	акция обыкновенная	25.06.1999
ТОО ―Искра‖	акция привилегированная	23.06.1999
ЗАО ―Агат‖	акция обыкновенная	25.06.1999

Вариант 9.

Наименование Вуза	Адрес	Команда
КрасГУ	Красноярск, пр-т Свободный,79	МатФак
КрасГУ	Красноярск, пр-т Свободный,79	ЭконФак
КГТУ	Красноярск, ул.Киренского,26	ФизФак
КГТУ эк. институт	Красноярск, ул.Киренского,26	ЮрФак
НГУ	Новосибирск, ул. Пирогова, 2	ФилФак
НГУ	Новосибирск, ул. Пирогова, 2	БиоХим

Вариант 10.

НаименованиеЭмитента	ТипЦБ	ДатаЭмиссии
ОАО ―КрАЗ‖	акция привилегированная	20.06.1999
ОАО ―КрАЗ‖	акция обыкновенная	23.06.1999
ЗАО ―Агат‖	акция привилегированная	20.06.1999
ЗАО ―Агат‖	акция привилегированная	24.06.1999

Практическое занятие № 4 Проектирование реляционной БД. Нормализация таблиц.

Цель работы: выработать практические навыки моделирования предметной области и построения нормальных форм баз данных

Нормализация, функциональные и многозначные зависимости

Нормализация – это разбиение таблицы на две или более, обладающих лучшими свойствами при включении, изменении и удалении данных. Окончательная цель нормализации сводится к получению такого проекта базы данных, в котором каждый факт появляется лишь в одном месте, т.е. исключена избыточность информации. Это делается не столько с целью экономии памяти, сколько для исключения возможной противоречивости хранимых данных.

Каждая таблица в реляционной БД удовлетворяет условию, в соответствии с которым в позиции на пересечении каждой строки и столбца таблицы всегда находится единственное атомарное значение, и никогда не может быть множества таких значений. Любая таблица, удовлетворяющая этому условию, называется нормализованной. Фактически, ненормализованные таблицы, т.е. таблицы, содержащие повторяющиеся группы, даже не допускаются в реляционной

БД.

Всякая нормализованная таблица автоматически считается таблицей в первой нормальной форме, сокращенно 1НФ. Таким образом, строго говоря, "нормализованная" и "находящаяся в 1НФ" означают одно и то же. Однако на практике термин "нормализованная" часто используется в более узком смысле – "полностью нормализованная", который означает, что в проекте не нарушаются никакие принципы

нормализации.

Теперь в дополнение к 1НФ можно определить дальнейшие уровни нормализации – вторую нормальную форму (2НФ), третью нормальную форму (3НФ) и т.д.

По существу, таблица находится в 2НФ, если она находится в 1НФ и удовлетворяет, кроме того, некоторому дополнительному условию, суть которого будет рассмотрена ниже. Таблица находится в 3НФ, если она находится в 2НФ и, помимо этого, удовлетворяет еще другому дополнительному условию и т.д.

Таким образом, каждая нормальная форма является в некотором смысле более ограниченной, но и более желательной, чем предшествующая. Это связано с тем, что "(N+1)-я нормальная форма" не обладает некоторыми непривлекательными особенностями, свойственным "N-й нормальной форме". Общий смысл дополнительного условия, налагаемого на (N+1)-ю нормальную форму по отношению к N-й нормальной форме, состоит в исключении этих непривлекательных особенностей.

За время развития технологии проектирования реляционных БД были выделены следующие нормальные формы: первая нормальная форма (1NF); вторая нормальная форма (2NF); третья нормальная форма (3NF); нормальная форма Бойса-Кодда (BCNF); четвертая нормальная форма (4NF); пятая нормальная форма, или нормальная форма проекции-соединения (5NF).

Обычно на практике применение находят только первые три нормальные формы.

Задание для практической работы

Задание 1. Приведение отношения к первой нормальной форме

1. Открыть файл journal.xls. Лист Ученик. Здесь собрана информация об ученике (см. лист Ученик). Анализ таблицы показывает, что в столбце «Родители» и «Место работы» указаны по 2 значения. Кроме того, в столбце адрес слишком много данных. Возможна ситуация, когда РОНО будет интересовать район проживания, при этом конкретная квартира – неинтересна. Из-за неоднозначности значений придется привести таблицу к первой нормальной форме.
2. В таблице скопировать строки и сделать так, чтобы в каждой строке был один родитель. 3. Одновременно нужно создать еще один столбец «Контекст адреса»
4. Получим отношение в первой нормальной форме.
5. Выделим в ней первичный ключ: строки отличаются друг от друга столбцами - «№ билета» и «Фамилии родителей». Зальем желтым цветом.

Задание 2. Приведение отношения ко второй нормальной форме

1. Открыть файл journal.xls. Лист Ученик.
2. Сделайте еще одну таблицу. Будет 2 таблицы. В одной будет все, что определяется только столбцом «№ билета», а в другой все, что определяется 2-мя столбцами.
3. Первую таблицу назовем «Личные данные ученика». Первичный ключ «№ билета»
4. Вторую таблицу назовем «Родители». Первичный ключ «№ билета» и «Фамилии родителей».
5. Отношение «Родители» конечно, больше преобразованиям не подлежит.

Задание 3. Приведение отношения к третьей нормальной форме

1. Открыть файл journal.xls. Лист Ученик.
2. Выделим из отношения «Личные данные ученика» таблицу «Класс».
3. Останется таблица «Личные данные». Однако в ней нужно оставить столбец «Класс».
4. В таблице «Класс» ключом является столбец «Класс».

ЗАМЕЧАНИЕ. Название специализации встречается многократно для разных классов. Со временем формулировка может измениться. Поэтому целесообразно сделать справочную таблицу «Справка» и сделать столбец «Код специализации».

Задание 4. Нормализация отношения «Преподаватель».

1. Открыть файл journal.xls. Лист «Преподаватель».

2. Отношение «Преподаватель» привести к первой нормальной форме. Для этого скопировать строки, исправить данные так, чтобы в каждой строке был только один класс и предмет.
3. Выделим первичный ключ. Это столбцы «Фамилия преподавателя», «Класс» и «Предмет».
4. Анализируем и видим, что есть атрибуты, которые зависят только от фамилии преподавателя. Значит, нужно привести отношение ко второй нормальной форме.
5. Получим две таблицы. Первую назовем «Преподаватель – класс – предмет». Она уже находится в третьей нормальной форме, т.к. здесь атрибуты ключевые и все не зависят друг от друга. Вернее здесь наблюдаются связи многие-ко-многим. Отставим пока.
6. Рассмотрим вторую таблицу «Личные данные преподавателя». Первичный ключ «Фамилия». Поскольку данные преподавателя могут меняться (замуж вышла), то целесообразнее сделать столбец «Код преподавателя» и сделать этот столбец ключевым. Но тогда в отношении «Класс» нужно вместо поля «Классный руководитель» поставить «Код классного руководителя».
7. Посмотрим на столбец «Классное руководство».
8. Такая информация у нас уже есть, дублировать ее не надо. Поэтому просто вычеркнем этот столбец.

Задание 5. Приведение отношения к четвертой нормальной форме

1. Открыть файл journal.xls. Лист «Преподаватель». Отношение «Преподаватель – класс – предмет».
2. Разделим его на 2 таблицы «Преподаватель – предмет» и «Преподаватель – класс».
3. Видим, что таблица «Преподаватель – класс» осталась «многие – ко – многим». Оставим ее в покое.

Задание 6. Конечный результат

1. Откройте файл journal_ready.xls.
2. Сравните эти таблицы с теми, которые получились у вас.

Задание 7. Постройте сетевую модель по примеру в программе ERModeler

На листе «Конечный результат» приведена реляционная модель той части задачи школьного журнала, нормализацией которой мы занимались на предыдущих страницах. Приведенную модель сложно считать наглядной и удобной для восприятия, однако именно такой вид представления наиболее удобен для проведения дальнейших этапов проектирования. Понимая это, условимся в дальнейшем проводить этап инфологического проектирования с учетом требований к реляционным моделям.

Практическое занятие № 5 Задание ключей. Создание основных объектов БД

Цель работы: приобретение практических навыков анализа предметной области, информационных задач и построения концептуальной модели базы данных.

Проектирование базы данных (БД)

Проектирование базы данных (БД) – одна из наиболее сложных и ответственных задач, связанных с созданием информационной системы (ИС). В результате еѐ решения должны быть определены содержание БД, эффективный для всех еѐ будущих пользователей способ организации данных и инструментальные средства управления данными.

Основная цель процесса проектирования БД состоит в получении такого проекта, который удовлетворяет следующим требованиям:

- корректность схемы БД, т.е. база должна быть гомоморфным образом моделируемой предметной области (ПО), где каждому объекту предметной области соответствуют данные в памяти ЭВМ, а каждому процессу – адекватные процедуры обработки данных;
- обеспечение ограничений ;

- эффективность функционирования ;
- защита данных (от аппаратных и программных сбоев и несанкционированного доступа);
- простота и удобство эксплуатации;
- гибкость, т.е. возможность развития и адаптации к изменениям предметной области и/или требований пользователей.

Внимание! Базы данных всегда проектируются под конкретное назначение системы.

Техника проектирования баз данных может измениться в целом и в деталях в зависимости от назначения системы. Например, следует различать проектирование систем складирования данных и проектирование так называемых OLTP-систем, ориентируемых на оперативную обработку транзакций. В данном учебном курсе рассматривается проектирование баз данных в основном для OLTP-систем. Именно на таких системах исторически сложилась техника проектирования баз данных.

Этапы проектирования базы данных

Процесс проектирования включает в себя следующие этапы:

Концептуальное проектирование – это процедура конструирования информационной модели, не зависящей от каких-либо физических условий реализации.

Логическое проектирование – это процесс конструирования информационной модели на основе существующих моделей данных, не зависимо от используемой СУБД и других условий физической реализации.

Физическое проектирование – это процедура создания описания конкретной реализации БД с описанием структуры хранения данных, методов доступа к данным.

Концептуальное проектирование

Основными задачами концептуального проектирования являются определение предметной области системы и формирование взгляда на ПО с позиций сообщества будущих пользователей БД, т.е. инфологической модели ПО.

Концептуальная модель ПО представляет собой описание структуры и динамики ПО, характера информационных потребностей пользователей в терминах, понятных пользователю и не зависимых от реализации БД. Это описание выражается в терминах не отдельных объектов ПО и связей между ними, а их типов, связанных с ними ограничений целостности и тех процессов, которые приводят к переходу предметной области из одного состояния в другое.

Рассмотрим основные подходы к созданию концептуальной модели предметной области.

1. Функциональный подход к проектированию БД

Этот метод реализует принцип "от задач" и применяется тогда, когда известны функции некоторой группы лиц и/или комплекса задач, для обслуживания информационных потребностей которых создаѐтся рассматриваемая БД.

2. Предметный подход к проектированию БД

Предметный подход к проектированию БД применяется в тех случаях, когда у разработчиков есть чѐткое представление о самой ПО и о том, какую именно информацию они хотели бы хранить в БД, а структура запросов не определена или определена не полностью. Тогда основное внимание уделяется исследованию ПО и наиболее адекватному еѐ отображению в БД с учѐтом самого широкого спектра информационных запросов к ней.

3. Проектирование с использованием метода "сущность-связь"

Метод "сущность–связь" (entity–relation, ER–method) является комбинацией двух предыдущих и обладает достоинствами обоих. Этап инфологического проектирования начинается с моделирования ПО. Проектировщик разбивает еѐ на ряд локальных областей, каждая из которых (в идеале) включает в себя информацию, достаточную для обеспечения запросов отдельной группы будущих пользователей или решения отдельной

задачи (подзадачи). Каждое локальное представление моделируется отдельно, затем они объединяются.

Выбор локального представления зависит от масштабов ПО. Обычно она разбивается на локальные области таким образом, чтобы каждая из них соответствовала отдельному внешнему приложению и содержала 6-7 сущностей.

Сущность – это объект, о котором в системе будет накапливаться информация. Сущности бывают как физически существующие (например, СОТРУДНИК или АВТОМОБИЛЬ), так и абстрактные (например, ЭКЗАМЕН или ДИАГНОЗ). Для сущностей различают тип сущности и экземпляр. Тип характеризуется именем и списком свойств, а экземпляр – конкретными значениями свойств. Типы сущностей можно классифицировать как сильные и слабые. Сильные сущности существуют сами по себе, а существование слабых сущностей зависит от существования сильных. Например, читатель библиотеки – сильная сущность, а абонемент этого читателя – слабая, которая зависит от наличия соответствующего читателя. Слабые сущности называют подчинѐнными (дочерними), а сильные – базовыми (основными, родительскими).

Для каждой сущности выбираются свойства (атрибуты). Различают:

- Идентифицирующие и описательные атрибуты. Идентифицирующие атрибуты имеют уникальное значение для сущностей данного типа и являются потенциальными ключами. Они позволяют однозначно распознавать экземпляры сущности. Из потенциальных ключей выбирается один первичный ключ (ПК). В качестве ПК обычно выбирается потенциальный ключ, по которому чаще происходит обращение к экземплярам записи. Кроме того, ПК должен включать в свой состав минимально необходимое для идентификации количество атрибутов. Остальные атрибуты называются описательными и заключают в себе интересующие свойства сущности.
- Составные и простые атрибуты. Простой атрибут состоит из одного компонента, его значение неделимо. Составной атрибут является комбинацией нескольких компонентов, возможно, принадлежащих разным типам данных (например, ФИО или адрес). Решение о том, использовать составной атрибут или разбивать его на компоненты, зависит от характера его обработки и формата пользовательского представления этого атрибута.
- Однозначные и многозначные атрибуты (могут иметь соответственно одно или много значений для каждого экземпляра сущности).
- Основные и производные атрибуты. Значение основного атрибута не зависит от других атрибутов. Значение производного атрибута вычисляется на основе значений других атрибутов (например, возраст студента вычисляется на основе даты его рождения и текущей даты).

Спецификация атрибута состоит из его названия, указания типа данных и описания ограничений целостности – множества значений (или домена), которые может принимать данный атрибут. Далее осуществляется спецификация связей внутри локального представления. Связи могут иметь различный содержательный смысл (семантику). Различают связи типа "сущностьсущность", "сущность-атрибут" и "атрибут-атрибут" для отношений между атрибутами, которые характеризуют одну и ту же сущность или одну и ту же связь типа "сущность-сущность". Каждая связь характеризуется именем, обязательностью, типом и степенью. Различают факультативные и обязательные связи. Если вновь порождѐнный объект одного типа оказывается по необходимости связанным с объектом другого типа, то между этими типами объектов существует обязательная связь (обозначается двойной линией). Иначе связь является факультативной. По типу различают множественные связи "один к одному" (1:1), "один ко многим" (1:N) и "многие ко многим" (M:N). Степень связи определяется количеством сущностей, которые охвачены данной связью. Пример бинарной связи – связь между отделом и сотрудниками, которые в нѐм работают. Примером тернарной связи является связь типа экзамен между сущностями ДИСЦИПЛИНА, СТУДЕНТ, ПРЕПОДАВАТЕЛЬ. Из последнего примера видно, что

связь также может иметь атрибуты (в данном случае это Дата проведения и Оценка). Пример ER–диаграммы с указанием сущностей, их атрибутов и связей приведен на рис. 1.

Рисунок 1. Пример ER–диаграммы с однозначными и многозначными атрибутами

Пример проектирования реляционной базы данных

В качестве примера возьмем базу данных компании, которая занимается издательской деятельностью. База данных создаѐтся для информационного обслуживания редакторов, менеджеров и других сотрудников компании. БД должна содержать данные о сотрудниках компании, книгах, авторах, финансовом состоянии компании и предоставлять возможность получать разнообразные отчѐты. В соответствии с предметной областью система строится с учѐтом следующих особенностей:

- каждая книга издаѐтся в рамках контракта;
- книга может быть написана несколькими авторами;
- контракт подписывается одним менеджером и всеми авторами книги;
- каждый автор может написать несколько книг (по разным контрактам);
- порядок, в котором авторы указаны на обложке, влияет на размер гонорара;
- если сотрудник является редактором, то он может работать одновременно над несколькими книгами;
- у каждой книги может быть несколько редакторов, один из них – ответственный редактор;
- каждый заказ оформляется на одного заказчика;
- в заказе на покупку может быть перечислено несколько книг.

Выделим базовые сущности этой предметной области:

1. Сотрудники компании. Атрибуты сотрудников – ФИО, табельный номер, пол, дата рождения, паспортные данные, ИНН, должность, оклад, домашний адрес и телефоны. Для редакторов необходимо хранить сведения о редактируемых книгах; для менеджеров – сведения о подписанных контрактах.
2. Авторы. Атрибуты авторов – ФИО, ИНН (индивидуальный номер налогоплательщика), паспортные данные, домашний адрес, телефоны. Для авторов необходимо хранить сведения о написанных книгах.
3. Книги. Атрибуты книги – авторы, название, тираж, дата выхода, цена одного экземпляра, общие затраты на издание, авторский гонорар.
4. Контракты будем рассматривать как связь между авторами, книгами и менеджерами. Атрибуты контракта – номер, дата подписания и участники.
5. Для отражения финансового положения компании в системе нужно учитывать заказы на книги. Для заказа необходимо хранить номер заказа, заказчика, адрес заказчика, дату поступления заказа, дату его выполнения, список заказанных книг с указанием количества экземпляров.

ER–диаграмма издательской компании приведена на рис. 2 (базовые сущности на рисунках выделены полужирным шрифтом).

Анализ информационных задач и круга пользователей системы

Система создаѐтся для обслуживания следующих групп пользователей:

- администрация (дирекция);
- менеджеры;
- редакторы;
- сотрудники компании, обслуживающие заказы.

Определим границы информационной поддержки пользователей: 1) Функциональные возможности:

- ведение БД (запись, чтение, модификация, удаление в архив);
- обеспечение логической непротиворечивости БД;
- обеспечение защиты данных от несанкционированного или случайного доступа (определение прав доступа);
- реализация наиболее часто встречающихся запросов в готовом виде;
- предоставление возможности сформировать произвольный запрос на языке манипулирования данными.

2) Готовые запросы:

- получение списка всех текущих проектов (книг, находящихся в печати и в продаже);
- получение списка редакторов, работающих над книгами;
- получение полной информации о книге (проекте);
- получение сведений о конкретном авторе (с перечнем всех книг);
- получение информации о продажах (по одному или по всем проектам);
- определение общей прибыли от продаж по текущим проектам;
- определение размера гонорара автора по конкретному проекту.

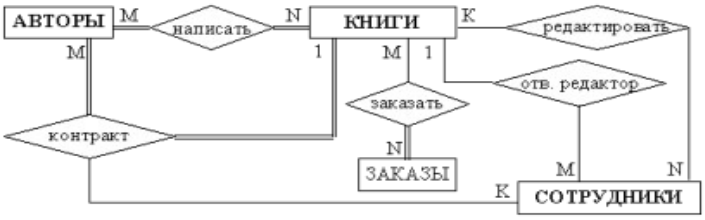


Рисунок 2. ER–диаграмма издательской компании

Задание для практической работы

По заданному описанию предметной области построить концептуальную модель базы данных

Выделите типы сущностей;

- Выделите типы связей и определите для них показатели кардинальности и степень участия сторон;
- Выделите атрибуты и свяжите их типами сущностей и связей;
- Определите потенциальные и первичные ключи сущностей; ˇ Нарисуйте ER-диаграмму.
- и проанализируйте информационные задачи и группы пользователей. **Индивидуальные задания к практической работе**

Вариант 1.

Задача – организация учебного процесса в вузе:

- Студенты: паспортные данные, адрес, дата зачисления, номер приказа, факультет, группа, является ли старостой, кафедра (специализация), изучаемые (изученные) предметы, оценки, задолженности, стипендия.
- Учебные курсы: название, факультет(ы), групп(ы), кафедра, семестр(ы), форма отчѐтности, число часов.
- Преподаватели: паспортные данные, адрес, телефон, фотография, кафедра, должность, учѐная степень, начальник (зав. кафедрой), предмет(ы), число ставок, зарплата.

Вариант 2.

Учет и выдача книг в библиотеке вуза:

- Книги: авторы, название, раздел УДК, раздел (техническая, общественно-политическая и т.п.), место и год издания, издательство, количество страниц, иллюстрированность, цена, дата покупки, номер сопроводительного документа (чек, счѐт/накладная),

вид издания (книги, учебники, брошюры, периодические издания), инвентарный номер (есть только для книг и некоторых учебников), длительность использования читателями (год, две недели, день), электронная версия книги или ее реферата (отсканированный текст).

- Читатели: номер читательского билета, ФИО, год рождения, адрес, дата записи, вид (студент, аспирант, преподаватель, сотрудник), курс, номер группы, названия взятых книг и даты их выдачи.

Вариант3.

Отдел кадров некоторой компании.

- Сотрудники: ФИО, паспортные данные, фотография, дом. и моб. телефоны, отдел, комната, раб. телефоны (в т.ч. местный), подчинѐнные сотрудники, должность, тип(ы) работы, задание(я), проект(ы), размер зарплаты, форма зарплаты (почасовая, фиксированная).
- Отделы: название, комната, телефон(ы), начальник, размер финансирования, число сотрудников.
- Проекты: название, дата начала, дата окончания, размер финансирования, тип финансирования (периодический, разовый), задачи и их исполнители, структура затрат и статьи расходов.

Вариант 4.

Отдел поставок некоторого предприятия.

- Поставщики: название компании, ФИО контактного лица, расчѐтный счѐт в банке, телефон, факс, поставляемое оборудование (материалы), даты поставок (по договорам и реальные), метод и стоимость доставки.
- Сырьѐ: тип, марка, минимальный запас на складе, время задержки, цена, продукты, при производстве которых используется, потребляемые объемы (необходимый, реальный, на единицу продукции).

Вариант5.

Пункт проката видеозаписей (внутренний учет).

- Видеокассеты: идентификационный номер видеокассеты, тип видеокассет, дата его создания, компания-поставщик, число штук данного типа (общее, в магазине, выдано в настоящее время, выдано всего, выдано в среднем за месяц), общая длительность записей; записи видеокассет: название, длительность, категория, год выпуска и производитель (оригинала).
- Клиенты: ФИО, паспортные данные, адрес, телефон; заказы, т.е. взятые видеокассеты (сейчас и в прошлом): номер, дата выдачи, дата возвращения, общая стоимость заказа.

Вариант 6.

Пункт проката видеозаписей (информация для клиентов).

- Видеокассеты: краткое описание, внешний вид (этикетка), марка (пустой) видеокассеты, цена за единицу прокатного времени (например: 1 день, 3 дня, неделя), есть ли в наличии, общая длительность записей; записи на видеокассете: название, длительность, жанр (категория), тема, год и страна выпуска (оригинала), кинокомпания, описание, актеры, режиссер.
- Заказы: идентификационные номера и названия выданных видеокассет, дата выдачи, дата возвращения (продления), общая стоимость заказа, возвращены ли кассеты заказа. Вариант 7.

Кинотеатры (информация для зрителей).

- Фильмы: название, описание, жанр (категория), длительность, популярность (рейтинг, число проданных билетов в России и в мире), показывается ли сейчас (сегодня, на текущей неделе), в каких кинотеатрах показывается, цены на билеты (в т.ч. средние).
- Кинотеатры: название, адрес, схема проезда, описание, число мест (в разных залах, если их несколько), акустическая система, широкоэкранность, фильмы и цены на

них: детские и взрослые билеты в зависимости от сеанса (дневной, вечерний и т.п.) и от категории мест (передние, задние и т.п.); сеансы показа фильмов (дата и время начала).

Вариант 8.

Ресторан (информация для посетителей).

- Меню: дневное или вечернее, список блюд по категориям.
- Блюда: цена, название, вид кухни, категории (первое, второе и т.п.; мясное, рыбное, салат и т.п.), является ли вегетарианским, компоненты блюда, время приготовления, есть ли в наличии.
- Компоненты блюд: тип (гарнир, соус, мясо и т.п.), калорийность, цена, рецепт, время приготовления, есть ли в наличии, ингредиенты (продукты) и их расходы на порцию.

Вариант 9.

Задача - информационная поддержка деятельности склада.

База данных должна содержать информацию о наименовании товара, его поставщике, количестве, цене товара, конечном сроке реализации, сроке хранения на складе. Торговый склад производит уценку хранящейся продукции. Если продукция хранится на складе дольше 10 месяцев, то она уценивается в 2 раза, а если срок хранения превысил 6 месяцев, но не достиг 10, то в 1,5 раза. Ведомость уценки товаров должна содержать информацию: наименование товара, количество товара(шт.), цена товара до уценки, срок хранения товара, цена товара после уценки, общая стоимость товаров после уценки.

Вариант10.

Задача – информационная поддержка деятельности адвокатской конторы. БД должна осуществлять:

- ведение списка адвокатов; ˇ ведение списка клиентов; ˇ ведение архива законченных дел.
- Необходимо предусмотреть:
- получение списка текущих клиентов для конкретного адвоката;
- определение эффективности защиты (максимальный срок минус полученный срок) с учѐтом оправданий, условных сроков и штрафов;
- определение неэффективности защиты (полученный срок минус минимальный срок);
- подсчѐт суммы гонораров (по отдельных делам) в текущем году;
- получение для конкретного адвоката списка текущих клиентов, которых он защищал ранее (из архива, с указанием полученных сроков и статей).

Вариант11.

Задача – информационная поддержка деятельности гостиницы.

БД должна осуществлять:

- ведение списка постояльцев;
- учѐт забронированных мест;
- ведение архива выбывших постояльцев за последний год.
- Необходимо предусмотреть:
- получение списка свободных номеров (по количеству мест и классу);
- получение списка номеров (мест), освобождающихся сегодня и завтра;
- выдачу информации по конкретному номеру;
- автоматизацию выдачи счетов на оплату номера и услуг;
- получение списка забронированных номеров;
- проверку наличия брони по имени клиента и/или названию организации Вариант12.

Описание предметной области:

- В компании несколько отделов.

- В каждом отделе есть некоторое количество сотрудников, занятых в нескольких проектах и размещающихся в нескольких офисах.
- Каждый сотрудник имеет план работы, т.е. несколько заданий, которые он должен выполнить. Для каждого такого задания существует ведомость, содержащая перечень денежных сумм, полученных сотрудником за выполнение этого задания.
- В каждом офисе установлено несколько телефонов.
- В базе данных должна храниться следующая информация.
- Для каждого отдела: номер отдела (уникальный), его бюджет и личный номер сотрудника, возглавляющего отдел (уникальный).
- Для каждого сотрудника: личный номер сотрудника (уникальный), номер текущего проекта, номер офиса, номер телефона, название выполняемого задания вместе с датой и размером выплат, проведенных в качестве оплаты за выполнение данного задания.
- Для каждого проекта: номер проекта (уникальный) и его бюджет.
- Для каждого офиса: номер офиса (уникальный), площадь в квадратных футах, номера всех установленных в нем телефонов.

Вариант13.

Задача – информационная поддержка деятельности спортивного клуба. БД должна осуществлять:

- ведение списков спортсменов и тренеров;
- учѐт проводимых соревнований (с ведением их архива);
- учѐт травм, полученных спортсменами.

Необходимо предусмотреть:

- возможность перехода спортсмена от одного тренера к другому;
- составление рейтингов спортсменов;
- составление рейтингов тренеров;
- выдачу информации по соревнованиям;
- выдачу информации по конкретному спортсмену;
- подбор возможных кандидатур на участие в соревнованиях (соответствующего уровня мастерства, возраста и без травм).

Вариант14.

Задача – информационная поддержка деятельности аптечного склада.

В аптечном складе хранятся лекарства. Сведения о лекарствах содержатся в специальной ведомости: наименование лекарственного препарата; количество (в шт.); цена; срок хранения на складе (в месяцах). Лекарства поступают на склад ежедневно от разных поставщиков, отпускаются два раза в неделю по предварительным заказам аптек. Выяснить, сколько стоит самый дорогой и самый дешевый препарат; сколько препаратов хранится на складе более 3 месяцев; сколько стоят все препараты, хранящиеся на складе, отыскать препараты, остаток которых равен нулю , ниже требуемого по заказам.

Вариант15.

—Электронный журнал посещаемости"

Предметная область представлена следующими документами:

- Список студентов
- Журнал посещаемости
- Расписание занятий
- Предусмотреть учет пропусков по уважительным, неуважительным причинам. Подсчет пропусков по каждому студенту, за неделю, месяц, заданный период, по конкретному предмету.

Вариант16.

«Итоги сессии»

База данных должна содержать информацию о двух последних сессиях студентов. Источником информации являются экзаменационные ведомости. Необходимо проводить анализ успеваемости по специальностям, формам обучения, курсам, группам, предметам,

вычислять средний балл по указанным критериям, а также число каждых оценок

Практическое занятие № 6 Создание проекта БД. Создание БД. Редактирование и модификация таблиц

Цель работы: освоить основные приемы заполнения и редактирования таблиц; познакомиться с простой сортировкой данных и с поиском записей по образцу; научиться сохранять и загружать базы данных.

Microsoft Office Access

Access является наиболее сложной программой из всех офисных приложений Microsoft Office. Чтобы начать работу с этой программой, вначале необходимо создать структуру базы данных, подробно ее описать, а затем создать различные формы.

ACCESS – это реляционная СУБД. Это означает, что с ее помощью можно работать одновременно с несколькими таблицами базы данных, эти таблицы между собой связаны. Таблицу ACCESS можно связать с данными, хранящимися на другом компьютере. Данные ACCESS очень просто комбинировать с данными EXCEL, WORD и другими программами Office.

Access во многом похож на Excel. Основное различие между таблицей БД и электронной таблицей – в системе адресации: в электронной таблице адресуется каждая ячейка, а в таблице БД – только поля текущей записи.

Таблицы – основные объекты базы данных (БД). В них хранятся данные. Реляционная база данных может иметь много взаимосвязанных таблиц. Сведения по разным вопросам следует хранить в разных таблицах.

Запрос – это средство, с помощью которого извлекается из базы данных информация, отвечающая определенным критериям. Результаты запроса представляют не все записи из таблицы, а только те, которые удовлетворяют запросу.

Формы – Обеспечивают более наглядную работу с таблицами, с помощью форм в базу вводят новые данные или просматривают имеющиеся.

Отчеты – средство представления данных таблиц. Отчеты могут быть оформлены надлежащим образом и распечатаны в том виде, в котором требуется пользователю.

Макросы – набор из одной или более макрокоманд, выполняющих определенные операции

(открытие форм, печать отчетов)

Модули - это программы, написанные на языке программирования Visual Basic.

Задание для практической работы

Заполните таблицы по образцу

1. Вызвать программу Access 2007 (Access 2016).
2. В окне системы управления базы данных щелкнуть по значку «Новая база данных». Справа в появившемся окне дать имя новой базе данных «Анкета ГС-31» и щелкнуть по значку папки, находящемуся справа от окна названия  . Откроется окно сохранения, найдите свою папку и сохраните в нее новый файл базы данных «Анкета ГС-31» (вместо ГС-31 укажите номер вашей группы). Затем нажмите на кнопку «Создать».
3. Появится окно «Таблица» (Рисунок 5.1).

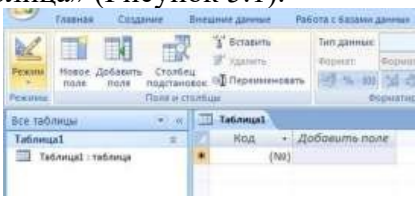


Рисунок 5.1. Окно пустой базы данных

4. В появившемся окне откройте меню команды <Режим> и выберите вариант <Конструктор>  и сохраните будущую таблицу под названием <Ведомость

успеваемости>. Появится окно Конструктора.

5. Заполните поля в Конструкторе данными из *рисунка 5.2.* Тип данных можно выбрать из меню, появившемся при нажатии на кнопку  в ячейке справа.

Обратите внимание*: ключевое поле «Счетчик» внесен в таблицу автоматически. Если напротив поля отсутствует значок ключа, то на панели инструментов щелкните по этому значку.* 

Рисунок 5.2. Создание таблицы через конструктор

6. Перейдите в режим таблицы, щелкнув по кнопке **Режим** на панели инструментов, Введите данные в этом режиме, заполняя клетки таблицы. Значение поля **Код** будет меняться автоматически.

7. Заполните базу данных значениями из *таблицы 5.1*. Напротив каждой фамилии выставьте по всем дисциплинам оценки от 2 до 5

Таблица 5.1.

Код	Фамилия	Имя	Математика	Менеджмент	Сервисная деятельность	Информационные технологии	Стандартизация	Гостиничная индустрия	Пропуски по неуважительной причине	Пропуски по уважительной причине
1	Иваникова	Анна								
2	Баранова	Ирина								
3	Корнилова	Ольга								
4	Воробьев	Алексей								
5	Воробьев	Олег								
6	Скоркин	Алекс								
7	Володина	Нина								
8	Новоселов	Алексей								
9	Петрова	Елена								
10	Чернова	Кристина								
11	Терещинка	Инна								
12	Истратов	Максим								
13	Бондарь	Ольга								
14	Ревин	Олег								
15	Шарова	Оксана								

8. Выполните редактирование ячеек:

– Замените фамилию Иванникова на Иванова.

9. Отсортируйте:

а) *фамилии* – по алфавиту (поставьте маркер на любую фамилию в столбце Фамилия и щелкнете мышкой по кнопке  на панели инструментов или произведите сортировку с помощью контекстного меню)

б) *имя* – по алфавиту

10. Сохраните текущую таблицу, щелкнув по кнопке «крестик» в правом верхнем углу окна таблицы.

11. Откройте снова свою базу данных.

12. Выполните поиск записей по образцу: *найти студентку по фамилии Володина*. Для этого установите курсор в поле фамилия, щелкните на кнопке  <Бинокль> на панели инструментов меню Главная и в появившемся диалоговом окне введите в поле <Образец> фамилию *Володина* и щелкните по кнопке <Найти>.

Примечание: Если требуется найти следующую подобную запись, то щелкните мышкой по кнопке <Найти далее>. По окончании работы щелкните по кнопке <Отмена>.

13. Переименуйте поле «Математика» на «Информатика» с помощью контекстного меню.

(Верните все как было назад).

14. Скройте столбец Пр н/пр., потом отобразите его назад.

15. Войдите в режим *Конструктора* и назначьте полю Пр н/пр и Пр ув/пр. *Маску ввода* 00 «часов». Заполните эти поля данными от 0 до 99.

16. Завершите работу с Access.

Практическое занятие № 7 Редактирование, добавление и удаление записей в таблице. Применение логических условий к записям. Открытие, редактирование и пополнение табличного файла.

Цели работы: закрепить навыки по редактированию таблиц; познакомиться с основными видами запросов; научиться создавать запросы на выборку различными способами; научиться создавать сложные запросы; научиться создавать перекрестные запросы.

Запрос – это средство, с помощью которого извлекается из базы данных информация, отвечающая определенным критериям. Результаты запроса представляют не все записи из таблицы, а только те, которые удовлетворяют запросу.

Запросы состоят из ряда условий, каждое условие состоит из трех элементов:

1. поле, которое используется для сравнения;
2. оператор, описывающий тип сравнения;
3. величина, с которой должно сравниваться значение поля. Выражения и операторы, применяемые в условиях отбора.

Выражения и операторы	Описание выражений и операторов
Числа	**Вводятся без ограничений**
Текст	Должен быть заключен в кавычки
Даты	Ограничиваются с двух сторон символами # (например, #01.02.02#)
*; +; -; /; ^	Арифметические операторы, связывающие выражения
<; <=; >; >=; =; <>	Операторы сравнения
And (И); Not (Нет); Or (Или)	Логические операторы
Like	Используется для логики замены в выражениях
In	Для определения, содержится ли элемент данных в списке значений
Between… And…	Для выбора значений из определенного интервала
?	Заменяет один символ (букву или цифру)
*	Заменяет несколько символов

Запросы могут быть простые, сложные перекрестные.

Задание для практической работы Создайте запросы к вашей базе данных 1) Откройте свою учебную базу данных.

2) Создайте запрос на выборку студентов, у которых по всем предметам только хорошие оценки с помощью *Мастера запросов*.

- На панели инструментов выберите команду <Мастер запросов>.

• В появившемся диалоговом окне выберите <Простой запрос> и щелкните по кнопке <OK>.

• В следующем окне выберите таблицу, по которой строится запрос (<Ведомость успеваемости>), и те поля, которые участвуют в запросе. Перенесите их в правую часть окна с помощью кнопки , нажмите <Далее>. В следующем окне тоже нажмите <Далее>.

• В другом окне дайте название запроса «Хорошисты» и нажмите <Готово>.

• Появится таблица <Хорошисты>, в которой отражены фамилии всех студентов и изучаемые предметы.

• Откройте таблицу «Хорошисты», перейдите в режим <Конструктор>. Здесь в поле <Условия отбора> под каждым предметом поставьте условие >=4 или 4OR5.

Примечание: Галочки в каждом поле означают, что по вашему выбору можно включить или убрать любое поле на выборку.

• Перейдите в режим таблицы, ответив <Да> на вопрос о сохранении запроса. (В таблице должны остаться фамилии «хорошистов»).

3) С помощью <Конструктора запросов> создайте запрос на выборку по таблице <Личные данные>.

• Щелкните по таблице <Личные данные>, зайдите в меню <Создание>, выберите команду <Конструктор запросов >.

• Добавьте нужную таблицу в поле запроса. Выделите еѐ в списке и щелкните по кнопке <Добавить>. Закройте окно <Добавление таблицы>.

• Выберите студентов, чьи фамилии начинаются на букву «В» и которые проживают в Анапе. Для этого:

- добавьте в строку <Поле> два поля <Фамилия> и <Город>;
- в строке <Условия отбора> в первом столбце укажите значение Like ―В * ‖, а во втором столбце с названием <Город> - «Анапа»;
- закройте запрос, сохранив его под названием ―ВВВ‖ (у вас должны остаться в списке студенты, проживающие в Анапе). Рисунок 7.1.

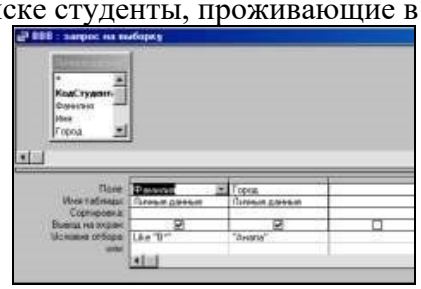


Рисунок 7.1. Запрос на выборку

Индивидуальные задания к практической работе

а) Составьте запрос с названием <Запрос 1> на базе таблицы <Ведомость успеваемости>, в котором будут указаны студенты, имеющие по первым двум предметам оценки не менее «4».

(Выполните запрос или через *Конструктор запросов*, или через *Мастер запросов*)

б) Составьте <Запрос 2> на базе таблицы <Ведомость успеваемости>, в котором будут указаны студенты, имеющие не более 30 часов пропусков по неуважительной причине. Добавьте в этот запрос поле пропуски по уважительной причине в интервале от 30 часов до 45 часов

(используйте оператор *Between… And*…)

в) Составьте <Запрос> на базе таблицы <Личные данные>. Выведите список студентов, которым на данный момент, т.е. на сегодняшнее число, исполнилось уже 17 лет (используйте оператор *Between… And…*)

Примечание: Дата записывается с использованием символа #, например,

#01.02.02.#

4) Составьте запрос на базе трех таблиц <Ведомость успеваемости>, <Личные данные> и <Преподаватель>. Выберите студентов, которые проживают в Новороссийске и у которых любимый предмет «Менеджмент». Озаглавьте <Запрос 4>. Используйте <Конструктор запросов>.

- В меню <Создание> выберите <Конструктор запросов>.
- Добавьте все три таблицы в поле запроса. Закройте окно <Добавление таблицы>.
- В первый столбец в строку <Поле> перетащите из первой таблицы с помощью мышки <Фамилия>, из второй таблицы во второй столбец <Город> и из третей таблицы в третий столбец строки <Поле> - <Предмет> (Рисунок 7.2).

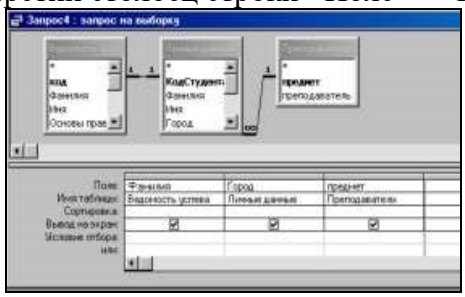


Рисунок 7.2. Запрос на выборку

- В поле <Условия отбора> в столбце <Город> введите город «Новороссийск», в столбец <Предмет> введите «Менеджмент».
- Сохраните запрос под именем <Запрос 4>.
- Откройте запрос и проверьте результат проделанной работы.

5) Выполните запрос на создание новой таблицы, в которой должны быть поля <Фамилия>, <Имя>, <Пропуски по неуважительной причине>, <Город> и <Предмет>.

- В меню <Создание> выберите <Конструктор запросов>.
- Добавьте все три таблицы из списка окна <Добавление таблицы>. Закройте это окно.
- В первую строчку <Поле> из первой таблицы перенесите в первый столбец поля <Фамилия>, во второй <Имя> и в третий <Пропуски по уважительной причине>, в четвертый столбец перетащите поле <Город> из второй таблицы и в последнем столбце будет поле <Предмет> из третьей таблицы.
- Закройте запрос, сохранив его с именем <Запрос 5>.

6) Создайте *перекрестный запрос*.

Допустим, нужно посчитать для ведомости, сколько в группе человек получили по предмету ―троек‖, ―четверок‖ и ―пятерок‖. Для этих целей используется *перекрестный запрос*.

- В меню <Создание> выберите <Мастер запросов>.
- В диалоговом окне выберите <Перекрестный запрос>, щелкните по кнопке <OK>.
- В окне <Создание перекрестных запросов> выделите таблицу <Ведомость успеваемости> и щелкните <Далее>.
- Выберите поля, значения которого будут использоваться в качестве заголовок строк – это <Фамилия> и <Имя>. Щелкните по кнопке <Далее>.
- Выберите поле, значение которого будут использоваться в качестве заголовков столбцов, например <Менеджмент>. Щелкните по кнопке <Далее>.
- Выберите функцию, по которой будут вычисляться значения ячеек на пересечении столбцов и строк (в данном случае Count – количество). Щелкните по кнопке <Далее>.
- Задайте имя запроса <Итог по менеджменту> и щелкните по кнопке <Готово>. Составьте аналогичные запросы для оценок по трем другим предметам.

7) Предъявите преподавателю все запросы своей базы данных на экране дисплея.

Завершите работу с Access.

Практическое занятие № 8 Создание ключевых полей. Задание индексов. Установление и удаление связей между таблицами.

Цели работы: научиться самостоятельно создавать ключевое поле; закрепить навыки по удалению, добавлению, заполнению и редактированию таблиц; научиться использовать фильтр в таблице. **Microsoft Office Access**

Access является наиболее сложной программой из всех офисных приложений Microsoft Office. Чтобы начать работу с этой программой, вначале необходимо создать структуру базы данных, подробно ее описать, а затем создать различные формы.

ACCESS – это реляционная СУБД. Это означает, что с ее помощью можно работать одновременно с несколькими таблицами базы данных, эти таблицы между собой связаны. Таблицу ACCESS можно связать с данными, хранящимися на другом компьютере. Данные ACCESS очень просто комбинировать с данными EXCEL, WORD и другими программами Office.

Access во многом похож на Excel. Основное различие между таблицей БД и электронной таблицей – в системе адресации: в электронной таблице адресуется каждая ячейка, а в таблице БД – только поля текущей записи.

Таблицы – основные объекты базы данных (БД). В них хранятся данные. Реляционная база данных может иметь много взаимосвязанных таблиц. Сведения по разным вопросам следует хранить в разных таблицах.

Запрос – это средство, с помощью которого извлекается из базы данных информация, отвечающая определенным критериям. Результаты запроса представляют не все записи из таблицы, а только те, которые удовлетворяют запросу.

Формы – Обеспечивают более наглядную работу с таблицами, с помощью форм в базу вводят новые данные или просматривают имеющиеся.

Отчеты – средство представления данных таблиц. Отчеты могут быть оформлены надлежащим образом и распечатаны в том виде, в котором требуется пользователю.

Макросы – набор из одной или более макрокоманд, выполняющих определенные операции

(открытие форм, печать отчетов)

Модули - это программы, написанные на языке программирования Visual Basic.

Задание для практической работы

Создайте таблицы и схему данных по заданным критериям 1) Откройте учебную базу данных <Анкета ГС-31>.

2) Создайте таблицу <Преподаватели > в *Режиме таблицы*. Для этого в меню Создание выберите кнопку Таблица. В появившейся таблице сделайте следующее:

- Добавьте два поля – Поле 1 и Поле 2, выполнив команду через контекстное меню.
- Переименуйте <Поле 1> на <Предмет>. Для этого поставьте курсор в любую ячейку столбца <Поля 1> и выполните команду *Переименовать столбец* из контекстного меню. Или щелкните два раза по имени поля, удалите старое название и впечатайте новое.
- Переименуйте аналогично <Поле 2> на <Преподаватель>.

3) Сохраните таблицу с именем <Преподаватели>, щелкнув по кнопке <Сохранить>

(дискетка  на панели инструментов).

4) Перейдите в режим <Конструктор> и удалите строку с ключевым словом Счетчик. Посмотрите как заданы поля. Сделайте поле <Предмет> ключевым, поместив

курсор на имя этого поля и щелкнув по кнопке - *Ключевое поле*. Тип данных поля задайте *текстовым*.

5) Перейдите в *Режим таблицы* и заполните таблицу <Преподаватели> записями из *Рисунок 6.1.*

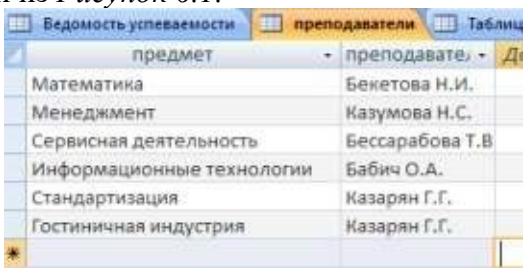


Рисунок 6.1. Таблица «Преподаватели»

6) Закройте таблицу <Преподаватели>, сохранив все изменения.

7) Используя <Шаблон таблиц>, создайте таблицу <Личные данные> студентов с ключевым полем. Для этого:

- Находясь на закладке <Создание> щелкните по кнопке <Шаблоны таблиц>, <Контакты>. Появится таблица уже с готовыми полями.
- Переименуйте предложенные поля на следующие поля: <Код студента>, <Фамилия>, <Имя>, <Город>, <Адрес>, <Телефон>, <Дата рождения>, <Фотография>, <Любимый предмет>, лишние поля удалите.
- Сохраните полученную таблицу под названием <Личные данные>. Ключевое поле задано автоматически.

8) Внесите данные в новую таблицу, заполнив поля <Фамилия>, <Имя>, <Город>, <Адрес>, <Телефон>, <Дата рождения>.

ПРИМЕЧАНИЕ. Поля <Фамилия> и <Имя> можно скопировать из таблицы <Ведомость успеваемости>. В поле <Город> внесите четыре разных города (например, Новороссийск, Геленджик, Анапа, Крымск)

9) Перейдите в режим <Конструктор> и назначьте типы данных: для поля <Телефон> - *числовой*, для поля <Дата рождения> - *дата/время*, для поля <Фотография> *– поле объекта OLE, для остальных – текстовый.*

Для поля <Любимый предмет> выполните свойство выбор предмета из списка с помощью *Мастера подстановок*. Для этого в строке <Любимый предмет> в поле *Тип данных – текстовый* щелкните по кнопке и в ниспадающем меню выберите команду <Мастер подстановок>.

- В диалоговом окне <Создание подстановки> поставьте флажок напротив способа <*Будет введен фиксированный набор значений*> и нажмите <Далее>.
- В следующем окне внесите в столбец все предметы (предметы из таблицы <Преподаватели>), нажмите <Далее>.
- В последнем окне, не изменяя имени столбца нажмите <Готово>.

10) Перейдите в режим таблицы и выберите для каждого студента с помощью кнопки из списка любимый предмет.

11) Создайте *схему данных*, т.е. установите связи между таблицами.

- Щелкните по кнопке - *Схема данных* на панели инструментов меню <Работа с базами данных>. В окне <Отобразить таблицу> выделите таблицу <Ведомость успеваемости> и щелкните по кнопке <Добавить>. Также добавьте таблицы <Преподаватели> и <Личные данные>. В окне <Схема данных> появиться условный вид этих таблиц. Закройте окно <Добавление таблицы>.
- Поставьте мышку на имя поля <Предметы> в таблице <Преподаватели> , и не отпуская кнопку мыши перетащите его на поле <Любимый предмет> таблицы <Личные данные>. Отпустите мышку. Появиться диалоговое окно <Связи>, в котором включите значки «*Обеспечение целостности данных*», «*Каскадное обновление связанных по-*

лей» и «*Каскадное удаление связанных полей»*. Щелкните по кнопке <Создать>. Появиться связь «один-ко-многим».

- Поставьте мышку на имя поля <Код студента> в таблице <Личные данные> и перетащите его, не отпуская мышки, на поле <Код> таблицы <Ведомость успеваемости>. В появившемся окне <Связи> включите значок «*Обеспечение целостности данных*» и щелкните по кнопке <Создать>. Появиться связь «один-кодному».
- Закройте схему данных, сохранив ее.

12) Произведите фильтрацию данных в таблице <Личные данные> *по выделенному*.

- Откройте таблицу в режиме таблицы.
- Выберите студентов, проживающих в Новороссийске. Для этого поставьте курсор в одну из первых записей, где есть город Новороссийск и щелкните по кнопке - *Фильтр по выделенному* на панели инструментов. Выберите команду <Равно «Новороссийск» >. Access отобразит все записи, удовлетворяющие критерию фильтрации. 
- Для отображения всех записей выполните команду <Удалить фильтр> для этого щелкните по соответствующей кнопке на панели инструментов  .

13) Закончите работу с базой данных Access.

Практическое занятие № 9 Проведение сортировки и фильтрации данных. Поиск данных по одному и нескольким полям. Поиск данных в таблице.

Цели работы: Научиться создавать формы ввода-вывода; научиться создавать кнопочные формы.

Форма – это средство, упрощающее ввод, редактирование и отображение информации, хранящейся в таблицах базы данных. Она представляет собой окно с набором элементов управления.

Форма сама по себе не хранит информацию, она просто обеспечивает удобный способ доступа к информации, хранящейся в одной или нескольких таблицах. Формы по сравнению с обработкой данных в режиме таблицы обладают следующими преимуществами:

- Форма позволяет в каждый момент сфокусировать внимание на отдельной записи;
- Элементы управления на форме можно расположить логичным образом, облегчающим чтение и работу с данными;
- Отдельные элементы управления обладают возможностями облегчить ввод и изменение отдельных данных;
- Некоторые объекты баз данных, такие как рисунки, анимации, звуки и видеоклипы, могут отображаться только в режиме формы, но не в режиме таблицы. Создание кнопочной формы.

Кнопочное меню представляет собой форму, на которой расположены элементы управления – кнопки с поясняющими надписями. Щелчок на кнопке открывает соответствующую таблицу, запрос, форму или отчет. Меню - удобный инструмент работы с базами данных, и он практически всегда присутствует в базах созданных для предприятий или фирм.

Кнопочное меню создают с помощью Диспетчера кнопочных форм.

Отчет – это гибкое и эффективное средство для организации просмотра и распечатки итоговой информации. В отчете можно получить результаты сложных расчетов, статистических сравнений, а также поместить в него рисунки и диаграммы. Пользователь имеет возможность разработать отчет самостоятельно (в режиме *Конструктора*) или создать отчет с помощью *Мастера*, т.е. полуавтоматически. **Задание для практической работы**

Задание 1. Создайте формы к базе данных 1) Откройте свою базу данных.

2) Создайте форму с помощью <Мастера форм> на базе таблицы <Ведомость успеваемости>.

- Откройте таблицу <Ведомость успеваемости>.
- Выберите закладку <Формы >, щелкните мышкой по кнопке <Другие формы>. 
- В появившемся диалоговом окне выберите <Мастер форм>.
- В поле <Таблицы/Запросы> выберите таблицу <Ведомость успеваемости>, в поле <Доступные поля> выберите поля <Фамилия>, <Имя> и перенесите их стрелкой в поле <Выбранные поля>. Также перенесите поля с названием предметов, щелкните по кнопке <Далее>.
- Выберите внешний вид формы – Табличный, щелкните по кнопке <Далее>.
- Выберите требуемый стиль (н-р, Обычная), щелкните по кнопке <Далее>.
- Задайте имя формы <Успеваемость> и щелкните по кнопке <Готово>. В результате получите форму, в которой можно менять данные и вводить новые значения. • Закройте форму.

3) Создайте форму на основе таблицы <Преподаватели>. • Откройте таблицу <Преподаватели>.

- Выберите закладку <Формы >, щелкните мышкой по кнопке <Другие формы>. 
- В появившемся диалоговом окне выберите <Мастер форм> .
- Выберите внешний вид формы - < ленточный>.
- Выберите любой стиль.
- Получите готовую форму. Сохраните ее под именем <Преподаватели>. • Закройте форму.
- Создайте форму <Личные данные> с помощью инструмента <Пустая форма>
- На вкладке Создание в группе Формы щелкните Пустая форма. 
- Access открывает пустую форму в режиме макета и отображает область Список полей.
- В области Список полей щелкните знак плюс (+) рядом с таблицей или таблицами, содержащими поля, которые нужно включить в форму.
- Чтобы добавить поле к форме, дважды щелкните его или перетащите его на форму. Чтобы добавить сразу несколько полей, щелкните их последовательно, удерживая нажатой клавишу CTRL. Затем перетащите выбранные поля на форму. • Закройте окно списка полей.
- Перейдите в режим Конструктора

Примечание 1. Размер окошка для названия поля и для его значений меняются мышкой.

Для этого выделите черный квадратик рамки (рамка станет цветной), установите курсор на границу рамки и с помощью двунаправленной стрелки измените размеры рамки.

Примечание 2. С помощью кнопок панели инструментов Шрифт меняйте соответственно цвет фона, текста, линии/границы и т.д.

- Расположите элементы удобно по полю.
- Задайте размер текста поля <Фамилия> равным 24 пт, шрифт - синего цвета.
- Увеличьте в высоту рамку поля <Фотография>.
- Сохраните форму с именем <Данные студентов>.
- Посмотрите все способы представления форм: в режиме Конструктора, режиме Макета и режиме Форм.
- Закройте форму.

4) Добавьте в таблицу <Личные данные> логическое поле <Институт> (т.е., собирается ли в дальнейшем учащийся поступать в институт). Значение этого поля <ДА> или <НЕТ>.

- Откройте таблицу <Личные данные> в режиме Конструктор. Добавьте поле с именем <Институт> и типом Логический. Закройте таблицу.
- Перейдите на закладку Формы и откройте форму <Данные студентов> в режиме Конструктор
- Щелкните по кнопке <Список полей> на панели инструментов, выделите название <Институт> и перетащите его мышкой в область данных, появиться значок и надпись <Институт>.
- Расположите новые элементы по правилам оформления формы (с помощью мыши). • Закройте <Список полей>

Примечание 3. Если флажок установлен, поле в таблице имеет значение <ДА>, если снят, то <НЕТ>.

- Перейдите в режим <Раздельная форма> и посмотрите записи. Установите флажки у восьми разных учащихся.
- Закройте форму, ответив утвердительно на вопрос о сохранении.

5) Создайте кнопочную форму <Заставка> с помощью Конструктора. • Щелкните по кнопке <Создать>.

- Выберите <Конструктор>. Появиться пустая форма. Задайте мышкой ширину формы, равную 10см, а высоту – 7см.
- Сохраните работу с именем <Заставка>.
- Откройте созданную форму <Заставка> в режиме Конструктора.
- Выберите на панели инструментов <Элементы управления> кнопку Аа – <Надпись>. Курсор мышки примет вид крестика с «приклеенной» буквой А. Щелкните мышкой по месту начала надписи и введите:

База данных

«Гостиница» группа ГС - 31

(после слов База данных нажмите одновременно комбинацию клавиш Shift+Enter.)

- Нажмите клавишу <Enter>. Выберите размер букв 18, а выравнивание - по центру. Цвет фона – голубой. Растяните мышкой надпись на ширину окна.
- Выберите на панели элементов значок - Кнопка. Щелкните мышкой по тому месту области данных, где должна быть кнопка. Появиться диалоговое окно <Создание кнопок>.
- Выберите категорию <Работа с формой>, а действие <Открыть форму>, и щелкните по кнопке <Далее>.
- Выберите форму <Успеваемость> , открываемую этой кнопкой щелкните по кнопке <Далее>. В следующем окне также щелкните по кнопке <Далее>.
- В следующем окне поставьте переключатель в положение <Текст>, наберите в поле слово <Успеваемость> (Рисунок8.1) и щелкните по кнопке <Далее>.

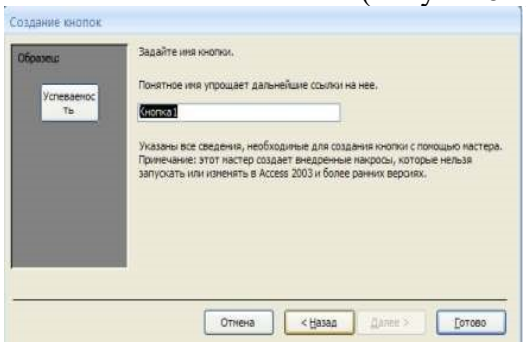


Рисунок 8.1. Создание кнопок

- Задайте имя кнопки <Успеваемость> и щелкните по кнопке <Готово>.

Примечание 4. Размер и расположение кнопок можно менять мышкой в режиме

Конструктор.

Самостоятельно создайте кнопки для форм <Личные данные> и <Преподаватели>.

• Перейдите в режим формы (Рисунок 8.2). Теперь при щелчке мышью по соответствующим кнопкам будут открываться соответствующие формы для работы. • Закройте форму.

Рисунок 8.2. Окно формы

6) Создайте кнопочную форму при помощи Диспетчера кнопочных форм.

• Откройте вкладку Работа с базами данных, команда - Диспетчер кнопочных форм. Вы получите диалоговое окно, представленное на Рисунке 8.3.

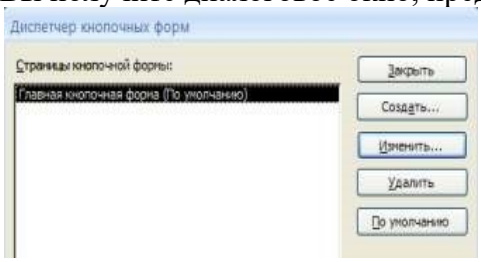


Рисунок 8.3. Диспетчер кнопок • Щелкните в этом окне по кнопке <Изменить>.

• В следующем окне щелкните по кнопке <Создать> и в появившемся окне измените содержимое полей в соответствии с Рисунком 8.4 (Команду и Форму выбирайте из списка, а не набирайте вручную). Щелкните по кнопке <ОК>.

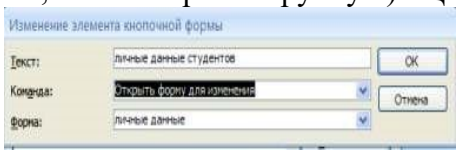


Рисунок 8.4. Изменение элементов кнопочной формы

• Аналогично создайте еще три элемента кнопочной формы: <Успеваемость>, <Преподаватели > и <Заставка>.

• Добавьте кнопку закрытия базы данных. Для этого щелкните по кнопке <Создать>, наберите в поле Текст слово <Выход>, а в поле Команда выберите <Выйти из приложения>. Закройте диалоговые окна.

• Откройте окно <Кнопочная форма> в режиме Конструктора или Макета, измените цвет надписи и название вашей базы данных на ГОСТИНИЦА, сохраните форму.

• Украсьте вашу форму рисунком. Для этого щелкните по значку Эмблема и выберите в открывшемся окне папку с рисунками, выберите понравившийся и вставьте в свою кнопочную форму.

• Перейдите в режим формы, проверьте работу всех кнопок кнопочной формы.

Завершите работу с базой данных, нажав на кнопку <Выход>.

Задание 2. Создайте отчет с помощью *Мастера отчетов.*

• Откройте вкладку *Создание,* меню *Отчеты*.

• Выберите *Мастер отчетов* и таблицу «Личные данные».

• Выберите нужные поля, которые будут участвовать в отчете, нажмите кнопку «Далее».

- В новом окне выберите поля для группировки так, чтобы сначала было указано поле «Фамилия», нажмите кнопку «Далее».
- На этом шаге отсортируйте данные по алфавиту, нажмите кнопку «Далее».
- Выберите вид макета *Ступенчатый* и щелкните по кнопке «Далее».
- Выберите стиль отчета: *Открытая* и щелкните по кнопке «Далее».
- Задайте имя отчета: «Отчет1» и щелкните по кнопке «Готово». Вы попадете в режим просмотра отчета.
- Закройте отчет согласившись с сохранением.

Самостоятельно. Составьте еще два отчета по запросам – «Запрос 3» и «Запрос 5», выбирая из разных макетов: *блок*; *структура*, выбирая из разных стилей. Сохраните отчеты под именами «Отчет 2» и «Отчет 3».

Задание 3. Создайте Пустой отчет в столбец на базе таблицы «Ведомость успеваемости» и сохраните его с именем «Успеваемость».

С помощью Конструктора измените цвет букв заголовка, их размер и шрифт.

Задание 4. Создайте почтовые наклейки.

- Откройте вкладку *Создание,* меню *Отчеты*.
- Выберите таблицу «Личные данные», команда Наклейки.
- В следующем окне щелкните по кнопке «Далее».
- В следующем окне выберите шрифт, размер шрифта, насыщенность и цвет, вновь щелкните по кнопке «Далее».
- В следующем окне создайте прототип наклейки, напечатав слово ЛИЧНОСТЬ и выбрав соответствующие поля, щелкните по кнопке «Далее».
- В следующем окне укажите поля для сортировки (Фамилия, Имя), щелкните по кнопке «Далее».
- Введите имя отчета «Наклейки» и щелкните по кнопке «Готово».
- Просмотрите Наклейки (Рисунок 8.5).

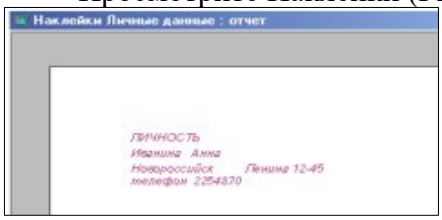


Рисунок 8.5.
Отчет

Практическое занятие № 10 Работа с переменными. Написание программного файла и работа с табличными файлами. Заполнение массива из табличного файла. Заполнение табличного файла из массива.

Цель работы: овладение навыками работы с переменными, обработка табличных файлов в среде VisualFoxPro.

VisualFoxPro

При создании приложения используется проект, который объединяет элементы приложения VisualFoxPro и группирует их по типам. База данных в VisualFoxPro – это совокупность таблиц, отношений между таблицами, индексов, триггеров, хранимых процедур.

База данных является частью проекта, поэтому еѐ целесообразно создавать в окне проекта.

Структурными элементами базы данных являются:

Поле – элементарная единица логической организации данных, которая соответствует неделимой единице информации – реквизиту. Для описания поля используются следующие характеристики: имя, длина, тип и точность (для числовых данных). В структуре записи файла указываются поля, значения которых являются ключами: первичными и внешними.

Первичный ключ - это одно или несколько полей, однозначно идентифицирующих

запись. Если первичный ключ состоит из одного поля, он называется простым, если из нескольких полей - составным ключом.

Внешний ключ - это одно или несколько полей, которые выполняют роль поисковых или группировочных признаков. В отличие от первичного, значение внешнего ключа может повторяться в нескольких записях файла, то есть он не является уникальным. Если по значению первичного ключа может быть найден один единственный экземпляр записи, то по внешнему – несколько.

Файл (таблица) – совокупность экземпляров записей одной структуры. Описание логической структуры записи файла содержит последовательность расположения полей записи и их основные характеристики,

Запись – совокупность логически связанных полей.

Экземпляр записи - отдельная реализация записи, содержащая конкретные значения ее полей.

Таблицы составляют основу вашей базы данных. В них будет храниться вся необходимая информация. В дальнейшем данные в таблице будут дополняться новыми данными, редактироваться или исключаться из таблицы. Поля таблицы предназначены для хранения в них данных. Это могут быть числа, текстовая информация, даты, графические файлы и т.д. В

VisualFoxPro допустимыми являются типы полей, указанные в таблице 10.1.
Таблица 10.1.Некоторые типы данных, используемых в СУБД FoxPro

Тип	Описание	Пример
Integer	Целые числа	9846
Numeric	Данные, с фиксированной точкой	3.1456
		-56.235
		"01/07/04"
Logical	Поля, содержащие только одно из двух возможных значений (да/нет)	True; False
Date	Дата	01/07/04
DateTime	Дата и время	01/07/04 12:30:00 pm
Memo	Очень длинный текст или комбинация текста и чисел	

Таблица 10.2. Некоторые команды СУБД FoxPro

Команда	Описание
CREATE	создание элементов БД (базы, таблиц, отчётов, запросов и т.д.)
MODIFY (STRUCTURE)	изменение элементов БД (базы, таблиц, отчётов, запросов и т.д.)
BROWSE, EDIT, CHANGE, APPEND	команды редактирования и просмотра содержимого таблиц
OPEN DATABASE	ОткрытиеБД
CLOSE	закрытие элементов БД (базы, таблиц, отчётов, запросов и т.д.)
QUIT	Выходиз Visual FoxPro

Задание для практической работы

Создать в служебной папке Мои документы новую папку и присвоить ей имя, пример:

3курс4группafoxlab (указывать свой курс и группу) Запустить программу MicrosoftVisualFoxPro:

Пуск/программы/VisualFoxPro

Ознакомиться с элементами рабочего окна программы

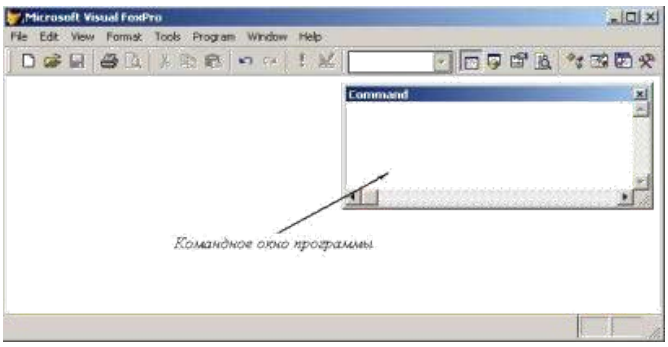


Рисунок 10.1. Главное окно VisualFoxPro 8.0

Создать новый проект: File/New/Project/NewFile, указать созданную нами ранее папку, присвоить имя проекту Информационные системы и сохранить. Все создаваемые в последующем элементы приложения будут храниться в проекте Информационные системы БД создаèтся аналогично: File/New/Database/Newfile, присвоить имя Штат исохранить. Структура проекта и его элементы отражаются в окне программы ProjectManager (Менеджер проекта)

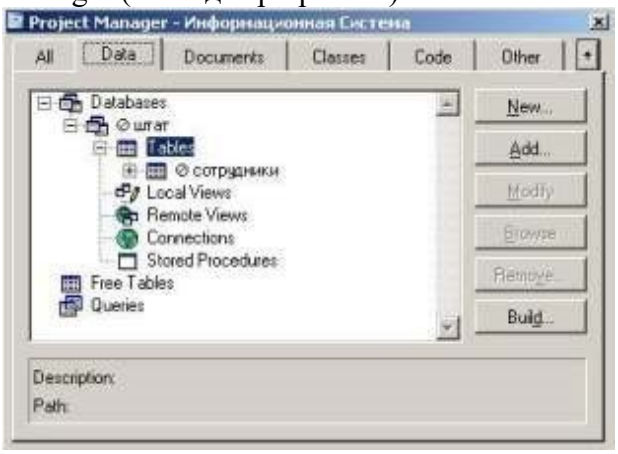


Рисунок 10.2. ОкноProject Manager

Добавить БД в проект: в окне ProjectManager щèлкнуть на вкладке Data/Databases/Add, в открывшемся диалоговом окне выбрать БД и нажать OK

Создать таблицу в БД штат: в окне ProjectManager (см. рис.9.2.) щèлкнуть клавишей мыши на вкладке Data/Databases/штат/Tables/New/Newtable, присвоить имя Сотрудники и сохранить

В появившемся окне TableDesigner(Конструктор таблиц) (см. рис.9.3.) на вкладке Fields (Поля) создать структуру таблицы в соответствии с таблицей 3.

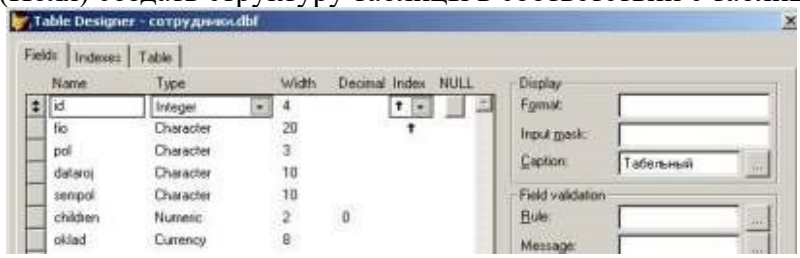


Рисунок 10.3. Окно конструктора таблицы TableDesigner вкладка Fields

Таблица 10.3.Определение полей таблицы Сотрудники в окне конструктора таблицы

TableDesigner

(наименование поля)	(тип поля)	(ширина поля)	(индексное поле)	(надпись, определяет заголовок поля)
Name	Type	Width	Index	Caption
id	Integer	4	↑Ascending	Табельный номер
fio	Character	20		ФИО
pol	Character	3		Пол
dataroj	Character	10		Датарождения
sempol	Character	10		Семейное положение
children	Numeric	2		Количестводетей
oklad	Carrency	8		Оклад
doljnost	Character	15		Должность
adress	Character	30		Место жительства

Перейти на вкладку Indexes(Индексы) окна конструктора таблицы TableDesigner и присвоить созданному индексу значения в соответствии с таблицей 10.4. Это необходимо для создания ключевого поля

Order	Name	Type	Expression	Filter	Collate
↑	id	Primary	id		Machine

Для ввода и редактирования данных в таблице *Сотрудники* в командном окне программы введите команду APPEND и нажмите клавишу Enter Командное окно программы **FoxPro** (см. рис.10.1.) предназначено для ввода команд с клавиатуры и последующего их выполнения. Любые действия и операции над элементами приложения в **СУБД FoxPro** могут осуществляться при помощи команд, вводимых в программное окно.

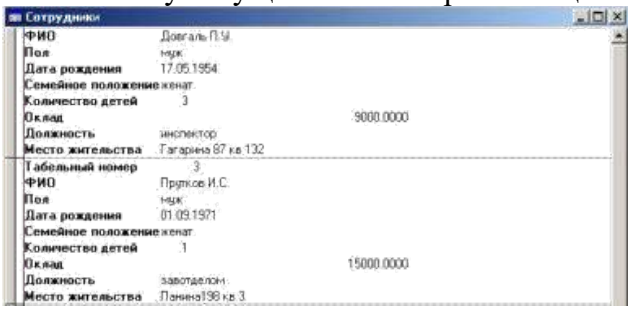


Рисунок 10.4. Просмотр и редактирование содержимого таблицы в режиме APPEND

После ввода данных (см. таблицу 10.4.) в командном окне программы введите команду BROWSE и нажмите клавишу Enter. Результат должен соответствовать рис.10.5.

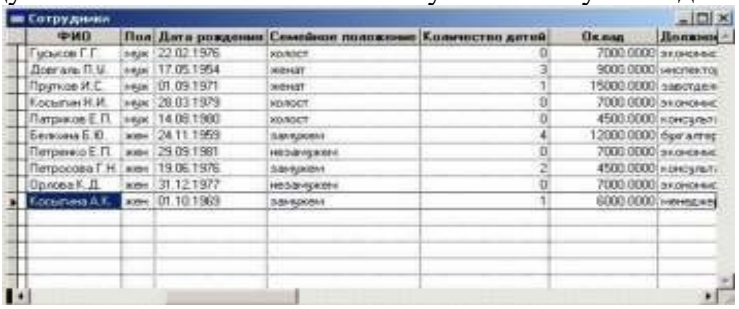


Рисунок 10.5. Просмотр таблицы в режиме BROWSE

Таблица 10.4. Содержимое таблицы Сотрудники

Таб. номер	ФИО	Пол	Дата рождения	Семейное положение	Количество детей	Оклад	Должность	Место жительства
1	Гуськов Г.Г.	муж.	22.02.1976	холост	0	7000	экономист	Лаптиева 156 кв 43
2	Довгаль П.У.	муж.	17.05.1954	женат	3	9000	инспектор	Гагарина 87 кв 132
3	Прутков И.С.	муж.	01.09.1971	женат	1	15000	завотделом	Панина198 кв 3
4	Косыгин Н.И.	муж.	28.03.1979	холост	0	7000	экономист	Николаева 30 кв 77
5	Патрикова Е.П.	муж.	14.08.1980	холост	0	4500	консультант	Нурадилова 18г кв 84
6	Бежина Б.Ю.	жен	24.11.1959	замужем	4	12000	бухгалтер	Луначевского 64 кв 9
7	Петренко Е.П.	жен	29.09.1981	незамужем	0	7000	экономист	Ленина 101 кв 15
8	Петросова Г.Н.	жен	19.06.1976	замужем	2	4500	консультант	Мира пр. 120 кв 2
9	Орлова К.Д.	жен	31.12.1977	незамужем	0	7000	экономист	Седова 18 кв 112
10	Косыгина А.К.	жен	01.10.1969	замужем	1	6000	менеджер	Панина 198 кв 3

В окне BROWSE установите курсор в нижней части таблицы и нажмите комбинацию клавиш Ctrl + Y на клавиатуре. В таблицу добавится новая строка

Прежде чем удалить строку в VisualFoxPro еѐ сначала необходимо пометить на удаление. Для этого щелкнуть клавишей мыши в ячейке слева от удаляемой записи в области узкого, непоименованного столбца, ячейка станет помеченной чѐрным цветом, или выделить необходимую строку и выбрать команду ToggleDeletionMark (Установка метки на удаление) пункта Table (Таблица) системного меню VisualFoxPro, запись будет помечена на удаление. Удалить запись, выбрав команду RemoveDeletedRecords (Удалить запись) пункта Table (Таблица) системного меню VisualFoxPro. Запись будет удалена

Для закрытия всех элементов приложения в командном окне программы введите команду

CLOSE ALL

Для завершения работы с программой **VisualFoxPro** в командном окне программы введите команду QUIT.

Создание отношений между таблицами в многотабличной БД Библиотека

Среди требований, предъявляемых к СУБД, основное место занимает возможность быстрого поиска необходимой информации. Средством, позволяющим решить эту проблему, является применение индексов. В VisualFoxPro для создания первичных ключей, определяющих отношения между таблицами и условия целостности данных также предназначены индексы. В этом случае индексы должны быть уникальными, то есть значения индексированного поля должны быть неповторяющимися (уникальными).

Для создания индекса таблицы используется вкладка Indexes (Индексы) окна конструктора таблиц TableDesigner. Все индексы в VisualFoxPro имеют имена, задаваемые в поле Name (Имя).

Для задания типа создаваемого индекса используется список Type (Тип).

Таблица 10.10.Описание типов индекса

Тип индекса	Описание
Regular (Обычный)	Создаётся индекс, в котором для каждой записи таблицы хранится значение индексного выражения. Если нескольо записей имеют одинаковое значение индексного выражения, то каждое значение хранится отдельно и содержит ссылку на связанную с ней запись
Unique(Уникальный)	Создаётся индекс, в котором хранятся только неповторяющиеся значения индексного выражения. Если две или более записей содержат одинаковое значение индексного выражения, то будет храниться только одно значение и ссылка на первую из записей с одинаковым значением индексного выражения. Таблица может иметь несколько уникальных индексов
Candidate (Кандидат)	Создаётся уникальный индекс, который не содержит полей с пустыми значениями. Этот индекс обладает всеми качествами первичного ключа и не является им только по той причине, что таблица не может содержать более одного первичного ключа
Primary (Первичный)	Создаётся уникальный индекс, который используется для связывания таблиц и определения условий целостности данных. Поля, входящие в первичный ключ, не должны допускать ввода пустых значений. В отличие от уникального индекса, таблица может иметь только один первичный ключ

Обычно, в VisualFoxPro при создании форм, отчѐтов и запросов используются несколько таблиц, между которыми установлены постоянные отношения. Такие таблицы называются связанными. Из двух связанных таблиц одна является главной (родительской), а другая подчинѐнной (дочерней). При создании индексов для родительской таблицы должен быть определѐн ключ типа Primary (Первичный) или типа Candidate (Кандидат), а для дочерней таблицы – индекс для связи с родительской таблицей типа Regular (Обычный).

Таблица 10.11. Типы отношений между таблицами

Типы отношений	Описание
Отношение "один-к-одному"	Каждая запись в одной таблице соответствует только одной записи в другой таблице
Отношение "один-ко-многим"	Наиболее распространѐнный тип отношений, каждой записи в одной таблице может соответствовать несколько записей в другой таблице
Отношение "много-к-одному"	Отношение "много-к-одному" можно сравнить с отношением "один-ко-многим", рассматриваемое с другой точки зрения
Отношение "много-ко-многим"	Нескольким записям одной таблицы может соответствовать несколько записей в другой таблице

Одним из самых важных требований, предъявляемых к базам данных, является целостность данных, которую определяют установленные между таблицами отношения. Для определения целостности данных в VisualFoxPro используется окно построителя условий целостности данных ReferentialIntegrityBuilder (Построитель целостности данных), которое содержит перечень всех установленных отношений между таблицами (см. рис.10.3.).

Таблица 10.12.Описание действий VisualFoxPro, в зависимости от выбранной

опции, при изменении значения первичного ключа или ключа типа

Наименование опции	Описание
Cascade (Каскадное изменение)	При изменении значений полей первичного ключа или ключа-кандидата в родительской таблице автоматически осуществляется каскадное изменение всех соответствующих значений в дочерней таблице
Restrict (Запрет изменения)	Не позволяет изменять значения полей первичного ключа или ключа-кандидата в родительской таблице, если в дочерней таблице имеется хотя бы одна запись, содержащая ссылку на изменяемую запись
Ignore(Игнорировать)	Позволяет изменять значения полей первичного ключа или ключа-кандидата в родительской таблице независимо от существования связанных записей в дочерней таблице. Целостность данных при этом не поддерживается

Таблица 10.13.Действия VisualFoxPro, в зависимости от выбранной опции, при удалении записи из родительской таблицы

Наименование опции	Описание
Cascade (Каскад)	При удалении записи из родительской таблицы автоматически осуществляется каскадное удаление всех записей из дочерней таблицы, связанных с удаляемой записью
Restrict (Запрет)	Не позволяет удалять записи в родительской таблице, если в дочерней таблице имеется хотя бы одна запись, содержащая ссылку на удаляемую запись. При попытке удаления записи возникает ошибка, которую вы можете обработать программно
Ignore(Игнорировать)	Позволяет удалять записи в родительской таблице независимо от существования связанных записей в дочерней таблице. Целостность данных при этом не поддерживается

Таблица 10.14.Действия **VisualFoxPro**, в зависимости от выбранной опции, при добавлении новой записи в родительскую таблицу

Наименование опции	Описание
Restrict (Запрет)	Не позволяет вводить запись, если значение индексного выражения дочерней таблицы не соответствует одной из записей в родительской
	таблице
Ignore(Игнорировать)	При вводе данных в дочернюю таблицу не анализируется значение индексного выражения. Целостность данных при этом не поддерживается

Практическое занятие № 11 Добавление записей в табличный файл из двумерного массива. Работа с командами ввода-вывода. Использование функций для работы с массивами

Цель работы: овладение навыками изменения табличных файлов при помощи запросов и функций

Обработка запросов в VisualFoxPro

Одним из основных назначений разработанного приложения является быстрый

поиск информации в БД и получение ответов на разнообразные вопросы. Для этих целей в VisualFoxPro используются средства, называемые запросами.

При создании запросов в VisualFoxPro разработчик может использовать следующие средства:

- QueryWizard (Мастер запросов)
- QueryDesigner (Конструктор запросов)
- Команда SELECT языкаVisual FoxPro

Результатом запроса является таблица, которую можно сохранить в массиве, создаваемой новой таблице, отобразить на экране в режиме Browse (Просмотр) или вывести в виде отчѐта.

Для создания запросов можно использоватьQueryWizard (Мастер запросов), который последовательно запрашивает наименования таблиц, используемых в запросе, перечень полей таблиц, критерий упорядочения и условия фильтрации данных. С помощью QueryDesigner (Конструктор запросов) можно формировать различной сложности критерии для выбора записей из одной или нескольких таблиц, над полями, выбираемыми из таблиц с помощью запросов, можно выполнять различные вычисления.

В верхней части окна QueryDesigner (Конструктор запросов) расположена панель, на которой отображаются используемые в запросе таблицы. Ниже находятся вкладки, предназначенные для выбора полей запроса и формирования условий выборки. Назначение этих вкладок приведено в таблице 1.. Открывая в окне QueryDesigner (Конструктор запросов) необходимые вкладки, можно выполнять следующие действия:

- Выбирать поля результирующей таблицы запроса
- Формировать вычисляемые поля
- Задавать критерии для выборки, группировки и упорядочения данных
- Указывать, куда выводить результаты выборки

Задание для практической работы

Запустить программу MicrosoftVisualFoxPro

Открыть проект Информационная система

Открыть БД Штат проекта

Создать запрос о сотрудниках, имеющих более одного ребѐнка и получающих зарплату менее 9000: в окне ProjectManager щѐлкнуть клавишей мыши на вкладке Data/Queries/New/QueryWizard, в появившемся диалоговом окне WizardSelection (Выбор мастера) выбрать пункт QueryWizard (Мастер запросов) и нажать кнопку OK

В появившемся диалоговом окне выбора исходной таблицы и полей в области Databasesandtables (Базы данных и таблицы) выбрать необходимую БДштат иуказать таблицу Сотрудники. Из области Availablefields (Имеющиеся поля) выбрать поля Fio и Oklad. Для перехода к следующему шагу нажать кнопку Next(Далее)

В появившемся диалоговом окне выбора условий выборки в области Field(Поле) верхней строки выбрать поле Children (Количество детей) таблицы Сотрудники, в области Operator (Оператор Условия) выбрать условие morethen, в области Value (Выражение) установить значение 1, в нижней строке внести следующие данные соответственно: в области Field(Поле) – поле Oklad (Зарплата), в области Operator (Оператор Условия) – lessthan, в области Value (Выражение) – 9000. Нажать кнопку Next(Далее)

В появившемся диалоговом окне выбора условия сортировки данных из области Availablefields (Имеющиеся поля) выбрать поле Fio. Нажать кнопку Next(Далее)

Нажать кнопку Preview (Просмотр) и просмотреть результат запроса (запрос будет выглядеть в виде простой таблицы см. рис.11.1.), закрыть таблицу запроса. Нажать кнопку Finish (Готово), в появившемся диалоговом окне SaveAs (Сохранить как) указать необходимую папку и присвоить запросу имя Зарплата дети

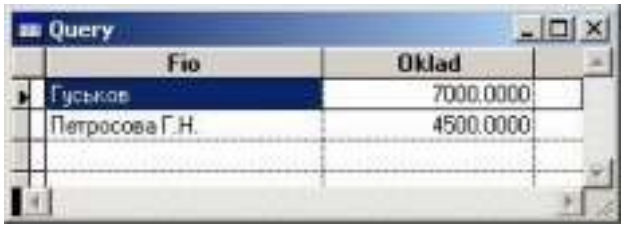


Рисунок 11.1. Запрос Зарплата дети

Запустить запрос: в окне ProjectManager щѐлкнуть клавишей мыши на вкладке Data/Queries/Зарплата дети/Run 10. Закрыть запрос Зарплата дети

Создать новый запрос, показывающий количество сотрудников организации, получающих определѐнную зарплату: в окне ProjectManager щѐлкнуть клавишей мыши на вкладке Data/Queries/New/NewQuery. В появившемся диалоговом окне AddTableorView (Добавить таблицу или представление данных) в области Database (База данных) выбрать БД Штат, в области TablesinDatabaseтаблицу Сотрудники. Нажать кнопку Add, при этом в верхней области окна
QueryDesigner (конструктор запросов) (см. рис.11.2.), находящегося позади окна AddTableorView (Добавить таблицу или представление данных) появится выбранная таблица. Нажать кнопку Close (Закрыть) для закрытия окна AddTableorView.

Рисунок 11.2. Окно QueryDesigner (конструктор запросов)

На вкладке Fields (Поля) окна QueryDesigner (конструктор запросов) в области Availablefields (Имеющиеся поля) выделить поле Сотрудники.oklad, оно автоматически перенесѐтся в область Functionandexpressions (Функции и выражения), нажать расположенную справа от поля кнопку вызова построителя выражения, откроется диалоговое окно
ExpressionBuilder (Построитель выражения) (см. рис.11.3.)

Рисунок 11.3. Диалоговое окно ExpressionBuilder (Построитель выражения)

В поле ввода Expression (Выражение) диалогового окна ExpressionBuilder (Построитель выражения) используя поля таблиц, расположенные в списке Fields (Поля), функции области Functions (Функции), и ввод данных с клавиатуры сформировать следующее выражение:Сотрудники.oklad AS Зарплата

Для проверки правильности набранного выражения нажать кнопку Verify (Проверить), нажать кнопку OK

Для переноса созданного выражения из области Functionandexpressions (Функции и выражения) в область SelectedFields (Выбранные поля) нажать находящуюся между данными областями кнопку Add (Добавить)

Руководствуясь предыдущими пунктами 12-15 практической части создать следующее выражение:

COUNT(Сотрудники.fio) AS Количество_сотрудников

Выбрать вкладку OrderBy (Упорядочение) окна QueryDesigner (конструктор запросов) и из областиSelectedFields (Выбранные поля) в область Orderingcriteria (Критерий упорядочения) перенести поле Сотрудники.oklad

Выбрать вкладкуGroupBy (Группировка) окна QueryDesigner (конструктор запросов) и из области Availablefields (Имеющиеся поля) в область GroupedFields (Поля группировки) перенести поле Сотрудники.oklad

Выбрать пункт меню Query (Запрос) окна программы VisualFoxPro, выбрать команду Comments (Комментарии) (см. таблицу 2.). В появившемся окне Comment(Комментарий) в поле ввода области AddComment (Добавить комментарий) ввести строку: Запрос, показывающий количество сотрудников организации, получающих определѐнную зарплату. Закрыть окно Comment(Комментарий)

Запустить созданный запрос: установить курсор в окне QueryDesigner (конструктор запросов), вызвать контекстное меню, выбрать команду RunQuery, или нажать кнопку [ ] на панели инструментов. Закрыть запрос

Рисунок 11.4. Запрос Зарплата

Просмотреть SQL-оператор, соответствующий запросу: установить курсор в окне QueryDesigner (конструктор запросов), вызвать контекстное меню, выбрать команду ViewSQL (Показать SQL), откроется диалоговое окно, в котором отображается SQL-оператор, соответствующий данному запросу. Закрыть окно QueryDesigner (конструктор запросов), система предложит сохранить запрос, нажать кнопку Yes (Да), в появившемся диалоговом окне SaveAs
(Сохранить как) указать необходимую папку и присвоить запросу имя Зарплата

Индивидуальные задания к практической работе

Создать многотабличный запрос, выводящий в таблицу названия, год издания и автора книг по информатике

Создать новый запрос, добавить таблицы Авторы, Автор_книги и Книги из БД Библиотека в окно QueryDesigner (конструктор запросов)

Из области Availablefields (Имеющиеся поля) вкладки Fields (Поля) внести в область SelectedFields (Выбранные поля) используя построитель выражения ExpressionBuilder следующие выражения:

Книги.nazvanie AS название_книги

Книги.razdel AS раздел

Книги.godizdan AS год_издания

ALLTRIM(Авторы.familiya)+" "+ALLTRIM(Авторы.imya) AS автор

Необходимо задать условие выбора записей из таблицы (Книги только по информатике) по полю razdel (Раздел). Перейти на вкладку Filter (Фильтр), в области FieldName (Имя поля) нажать кнопку раскрытия списка и выбрать поле Книги.razdel, в области Criteria (Критерии) выбрать значение ==, в области Example (Образец) ввести значение ― Информатика‖

Запустить запрос и посмотреть результат (см. рис.11.5.), закрыть запрос, закрыть окноQueryDesigner (конструктор запросов), сохранить запрос, присвоив ему имя Книги по информатике

Рисунок 11.5. Запрос Книги по информатике

Практическое занятие № 12 Создание меню различных видов. Модификация и управление меню.

Цель работы: овладение навыками создания и модификации структуры меню

Создание меню в VisualFoxPro

В соответствии со стандартами Windows в любом приложении рекомендуется иметь строку меню, которая в VisualFoxPro содержит команды, предназначенные для вызова форм, формирования отчѐтов, запросов и т.д.

Строкой меню называется горизонтальное меню, располагаемое в верхней части экрана. Примером строки меню является основное меню VisualFoxPro, а также меню программ, работающих в среде Windows. Созданное в конструкторе MenuDesigner (Конструктор меню) меню может замещать основное меню VisualFoxPro или добавляться к нему.

Для создания меню необходимо выполнить следующие действия:

- Открыть окно конструктора меню MenuDesigner (Конструктор меню)
- Описать вид меню, текст, пункты меню и его атрибуты
- Сгенерировать меню. При этом создаѐтся программа, которая в результате запускается на выполнение

VisualFoxPro имеется возможность создания меню в виде строки Menu, или всплывающего меню Shortcut, в котором основные пункты расположены по вертикали.

Таблица 12.1.Назначение кнопок конструктора меню MenuDesigner (Конструктор меню)

Кнопка	Назначение
Insert (Вставить)	Добавляет в меню новый пункт
InsertBar (Вставить команды системного меню)	Открывает диалоговое окно **InsertSystemMenuBar**, содержащее команды системного меню **VisualFoxPro**, позволяя разместить их в создаваемом пользовательском меню
Delete (Удалить)	Удаляет текущий пункт меню
MoveItem (Переместить элемент)	Открывает одноимённое диалоговое окно, позволяющее указать пункт меню, в который переносится текущий подпункт
Preview (Просмотр)	Размещает создаваемое меню на экране, позволяя просмотреть его внешний вид

Таблица 12.2. Типы меню

Тип меню	Назначение
Command(Команда)	При выборе пункта меню данного типа будет выполняться связанная с ним команда
PadName (Наименование строки меню)	При выборе пункта меню никаких действий выполняться не будет. Как правило, используется в качестве дополнительного пояснения к меню
Submenu(Подменю)	При выборе пункта меню раскрывается связанное с данным пунктом ниспадающее меню
Procedure (Процедура)	При выборе пункта меню вызывается процедура, определённая для данного пункта меню

Таблица 12.3. Типы пункта меню

Тип пункта меню	Действие
Submenu(Подменю)	Раскрывается связанное с данным пунктом меню ниспадающее меню
Procedure (Процедура)	Выполняется процедура, определенная в конструкторе меню
Command(Команда)	Выполняется команда, расположенная в поле рядом с типом пункта меню

Задание для практической работы

Запустить программу MicrosoftVisualFoxPro

Открыть проект Информационная система

Открыть окно MenuDesigner (Конструктор меню): в окне ProjectManager щѐлкнуть клавишей мыши на вкладке Other/Menus/New, в появившемся диалоговом окне NewMenu (Новое меню) выбрать пункт Menu(Меню), появится окно MenuDesigner (Конструктор меню) (см.
рис.12.1.)

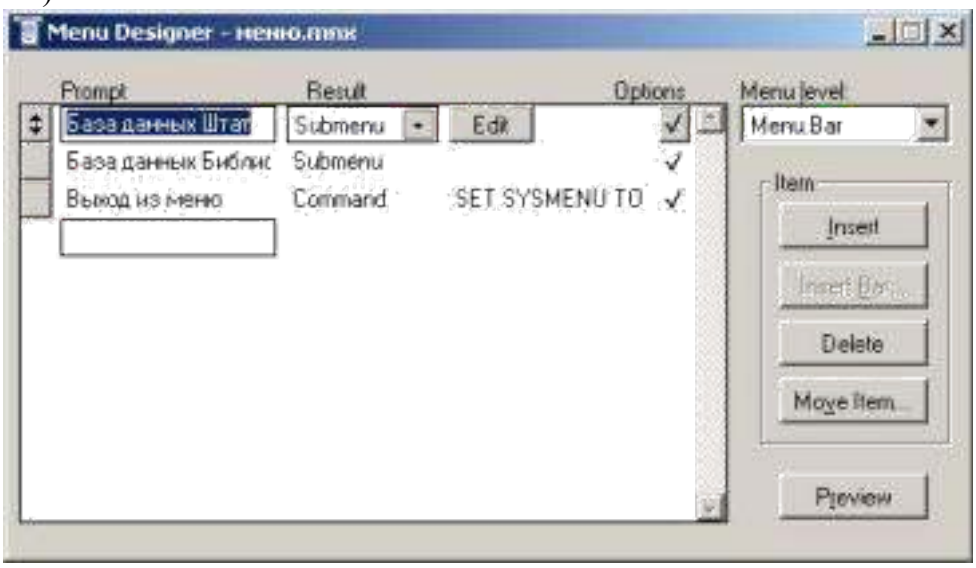


Рисунок 12.1. Окно MenuDesigner (Конструктор меню)

Область конструктора меню, над которой расположены надписи Prompt(Приглашение), Result (Результат) и Options(Опции), предназначена для формирования меню. В поле Prompt(Приглашение) вводят наименования пунктов меню, раскрывающийся список Result (Результат) используется для указания типа пункта меню, а кнопка Options(Опции) открывает диалоговое окно PromptOptions (Опции элемента меню), в котором можно определить дополнительные параметры данного элемента меню. В списке Menulevel (Уровень меню) указывается уровень текущего меню. Слева в конструкторе меню размещены кнопки (см. таблицу
12.1.)

Рисунок 12.2. Схема создаваемого меню Меню

В поле Prompt(Приглашение) ввести наименование первого пункта меню – База данных Штат, нажать для перехода на следующее поле клавишу Enter или Tab на клавиатуре, курсор окажется в списке Result (Результат) (см. рис.1.) Указать тип пункта меню Submenu(Подменю), нажать кнопку Options(Опции), откроется диалоговое окно

PromptOptions (Опции элемента меню), в котором, в области Shortcut(Всплывающее меню) установить курсор в поле KeyLabel (Метка) и нажать комбинацию клавиш Alt + F1 на клавиатуре. Мы определили комбинацию клавиш быстрого вызова пункта меню База данных Штат. Нажать кнопку OK

Перейти на следующую строку и ввести наименование второго пункта – База данных

Библиотека, тип пункта меню Submenu(Подменю), определить комбинацию клавиш Alt + F2 быстрого вызова пункта меню База данных Библиотека (см. пункт 4. лабораторной работы)

Перейти на следующую строку и ввести наименование третьего пункта – Выход из меню, в списке Result (Результат) указать тип пункта меню Command(Команда), в появившейся справа строке ввода ввести команду – SET SYSMENU TO DEFAULT. Определить комбинацию клавиш Alt + F4 быстрого вызова действия пункта меню Выход из меню

Пункты меню База данных Библиотека и База данных Штат имеют свои подменю. Для создания подменю перейти на строку База данных Библиотека, нажать кнопку Create (Создать), находящуюся справа от списка Result (Результат), на экране появится пустое окно конструктора меню. Ввести в поле Prompt(Приглашение) наименование подпункта – Таблицы базы данных, в списке Result (Результат) указать тип пункта меню Submenu(Подменю), нажать кнопку Create (Создать), так как он тоже имеет подменю (см.рис.2.)

В появившемся пустом окне конструктора меню ввести в поле Prompt(Приглашение) наименование подпункта – Автор книги, в списке Result (Результат) указать тип пункта меню Procedure (Процедура), нажать кнопку Create (Создать) и в открывшемся окне редактирования процедуры ввести следующие команды:

```
OPEN DATABASE \<Имя вашей папки>\библиотека.dbc
USE библиотека!автор_книги
SELECTавтор_книги
BROWSE CLOSE
DATABASES
```

Закрыть окно редактирования процедуры, перейти на следующую строку и ввести наименование следующего пункта – Авторы, в списке Result (Результат) указать тип пункта меню Procedure (Процедура), нажать кнопку Create (Создать) и в открывшемся окне редактирования процедуры ввести следующие команды:

```
OPEN DATABASE \<Имя вашей папки>\библиотека.dbc
USE библиотека!авторы
SELECT авторы
BROWSE CLOSE
DATABASES
```

Закрыть окно редактирования процедуры, перейти на следующую строку и ввести наименование следующего пункта – Издательство, в списке Result (Результат) указать тип пункта меню Procedure (Процедура), нажать кнопку Create (Создать) и в открывшемся окне редактирования процедуры ввести следующие команды:

```
OPEN DATABASE \<Имя вашей папки>\библиотека.dbc
USE библиотека!издательство
SELECT издательство
BROWSE CLOSE
DATABASES
```

Закрыть окно редактирования процедуры, перейти на следующую строку и ввести наименование следующего пункта – Книги, в списке Result (Результат) указать тип

пункта меню Procedure (Процедура), нажать кнопку Create (Создать) и в открывшемся окне редактирования процедуры ввести следующие команды:

OPEN DATABASE \<Имя вашей папки>\библиотека.dbc
USE библиотека!книги
SELECT книги
BROWSE CLOSE
DATABASES

Закрыть окно редактирования процедуры, перейти на следующую строку и ввести наименование следующего пункта – Разделы, в списке Result (Результат) указать тип пункта меню Procedure (Процедура), нажать кнопку Create (Создать) и в открывшемся окне редактирования процедуры ввести следующие команды:

OPEN DATABASE \<Имя вашей папки>\библиотека.dbc
USE библиотека!разделы
SELECT разделы
BROWSE CLOSE
DATABASES

Закрыть окно редактирования процедуры. Для перехода в меню верхнего уровня использовать список MenuLevel (Уровень меню): выбрать из выпадающего списка MenuLevel (Уровень меню) пункт – Базаданных

Под строкой Таблицы базы данных ввести наименование следующего пункта – Форма Книги, в списке Result (Результат) указать тип пункта меню Command(Команда), в появившейся справа строке ввода ввести команду – DO FORM \< Имя вашей папки>\ книги.scx

Перейти на следующую строку и ввести наименование следующего пункта – Отчѐт Книги, в списке Result (Результат) указать тип пункта меню Command(Команда), в появившейся справа строке ввода ввести команду – REPORT FORM \< Имя вашей папки>\книги.frx PREVIEW

Перейти на следующую строку и ввести наименование следующего пункта – Запрос, выводящий на экран названия, год издания и авторов книг по информатике, в списке Result
(Результат) указать тип пункта менюProcedure (Процедура), нажать кнопку Create (Создать) и в открывшемся окне редактирования процедуры ввести следующие команды:

OPEN DATABASE \<Имя вашей папки>\библиотека.dbc
DO "\<Имя вашей папки>\книги по информатике.qpr"
CLOSE DATABASES

Закрыть окно редактирования процедуры. Для просмотра создаваемых пунктов меню, нажать кнопку Preview (Просмотр), при этом основное меню VisualFoxPro будет заменено создаваемым меню. После просмотра нажать кнопку OKдиалового окна Preview (Просмотр)

Для улучшения внешнего вида, а также для объединения в группы схожих по смыслу команд, в меню можно использовать разделительные линии. Разделительные линии представляют собой пункт меню, в котором в поле ввода Prompt(Приглашение) вместо наименования пункта вводятся символы \-. Также VisualFoxPro позволяет справа от команд пользовательскогоменю расположить графическое изображение, для этого необходимо выполнить следующие действия: нажать кнопку Options(Опции) для необходимого пункта меню, откроется диалоговое окно PromptOptions (Опции элемента меню), в котором, в области Picture(Изображение) указать опциюResource (Ресурс), нажать кнопку, располагаемую справа от поля, находящегося под опцией. Откроется диалоговое окно InsertSystemMenuBar (Вставить из системного меню). Из списка графических изображений, используемых VisualFoxPro в системном меню выбрать наиболее соответствующее пункту меню значение и нажать кнопку OK. Выбранное

значение переносится в область просмотра области Picture(Изображение). Нажать кнопку OK для закрытия окна PromptOptions (Опции элемента меню)

Для перехода в меню верхнего уровня выбрать из выпадающего списка MenuLevel
(Уровень меню) пункт – MenuBar

Указать месторасположение создаваемого меню: пункт меню VisualFoxProView/GeneralOptions, откроется диалоговое окно GeneralOptions (Основные параметры), в области Location (Размещение) выбрать пункт Append (Добавить), для добавления создаваемого меню в основное меню VisualFoxPro

Практическое занятие № 13 Создание рабочих и системных окон. Добавление элементов управления рабочим окном

Цель работы: овладение практическими навыками работы с системными окнами, создание отчетов по представленным данным

Отчеты в VisualFoxPro

Отчет – форматированное представление данных, выводимое на экран, принтер или в файл. Отчет, создаваемый в VisualFoxPro, может быть представлен в табличном виде или в свободной форме. Табличные отчеты используются для печати данных, представленных в виде списка. При создании отчета в VisualFoxProразработчикможет использовать следующие средства:

- ReportWizard – Мастер отчета. Позволяет быстро создать отчѐт, применяя сортировку, группировку данных и заданный разработчиком стиль оформления;
- ReportDesigner – Конструктор отчета. Позволяет разрабатывать собственные отчѐты или модифицировать отчѐты, созданные с помощью мастера;
- QuickReport – Быстрый отчет. Данное средство предназначено для размещения в конструкторе отчѐта полей и задания среды окружения.

Задание для практической работы

Запустить программу MicrosoftVisualFoxPro

Открыть проект Информационная система

Открыть БД проекта. Для этого необходимо на вкладке Data (Данные) установить курсор на еѐ название и нажать кнопку Open (Открыть) окна проекта. При этом на стандартной панели инструментов в списке Databases (Базы данных) появится название открытой БД Перейти на вкладку Documents/Reports/New/New Report /Report Wizard

В появившемся диалоговом окне WizardSelection указать тип создаваемого отчета ReportWizard(Мастер отчетов) и нажать кнопку OK

Появляется первое диалоговое окно мастера отчѐта (см. рис.13.1.). Выбрать из верхнего списка области Databasesandtables (Базы данных и таблицы) базу данных штат, а из нижнего – таблицу сотрудники

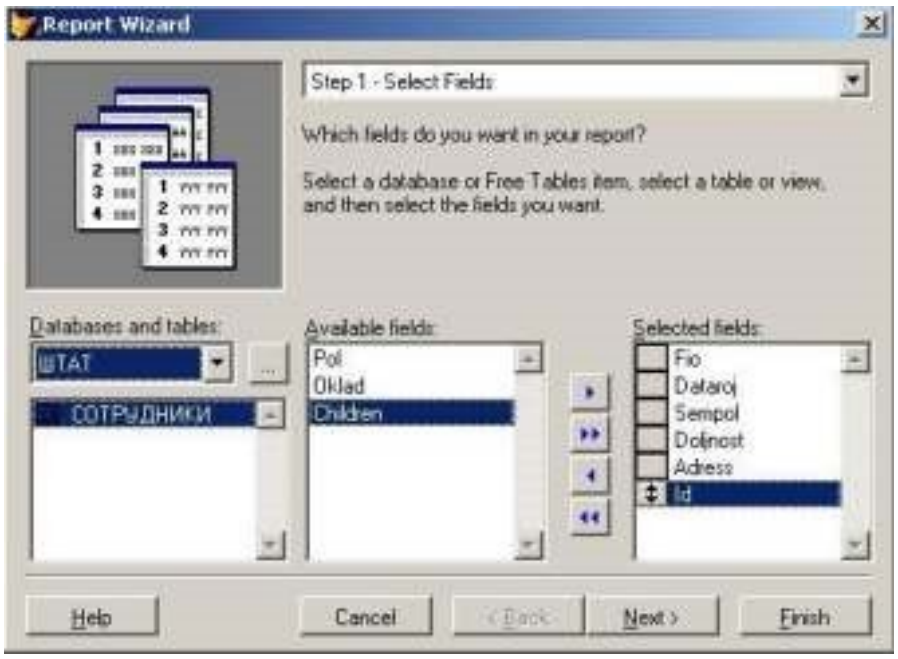


Рисунок 13.1. Диалоговое окно создания однотабличного отчета с помощью мастера. Перенести из списка Availablefields (Имеющиеся поля) в список Selectedfields (Выбранные поля) поля, которые вы хотите разместить в создаваемом отчете (в соответствии с рис.13.1.). После формирования списка отображаемых в отчете полей нажать кнопку Next (Далее) для перехода к следующему шагу в создании отчета

В следующем окне мастера создания отчета указать поля, по которым будут осуществляться группировка данных в отчете (рис.13.2.)

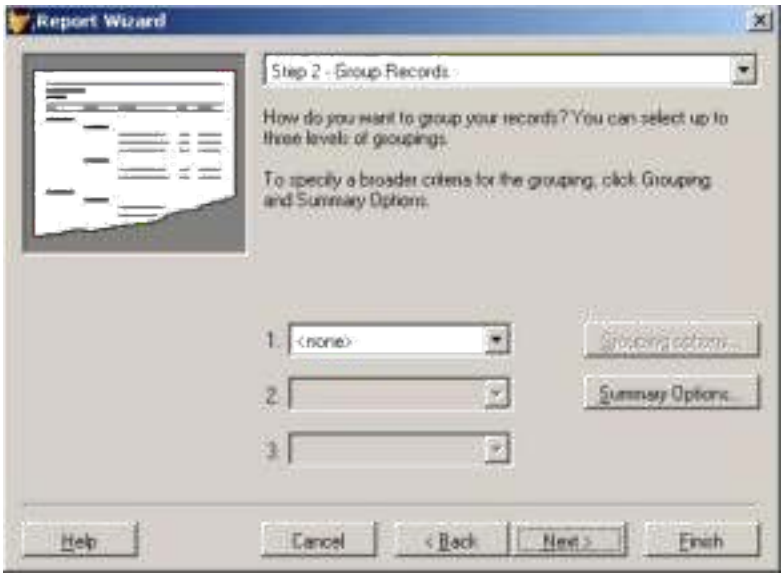


Рисунок 13.2. Диалоговое окно определения полей для группировки данных в отчете.

В центре диалогового окна расположены три раскрывающихся списка, позволяющих задать до трех группировок данных в отчете. Эти списки содержат все поля таблицы. Для осуществления группировки данных в отчете выбрать нужное поле из раскрывающегося списка 1. При создании второй и третьей группировки используются, соответственно, списки 2 и 3. В нашем примере группировка данных по полям не осуществляется, это диалоговое окно мастера отчѐтов необходимо пропустить и нажать кнопку Next

В следующем диалоговом окне мастера задается стиль отображения объектов в отчете (рис.13.3.). Список Style (Стиль) содержит несколько вариантов отображения объектов

(полей, линий, заголовок и т.д.) в отчете. С помощью области просмотра в верхнем левом углу диалогового окна можно просмотреть тот стиль, который мы выбрали. Выбрав стиль, нажать кнопку

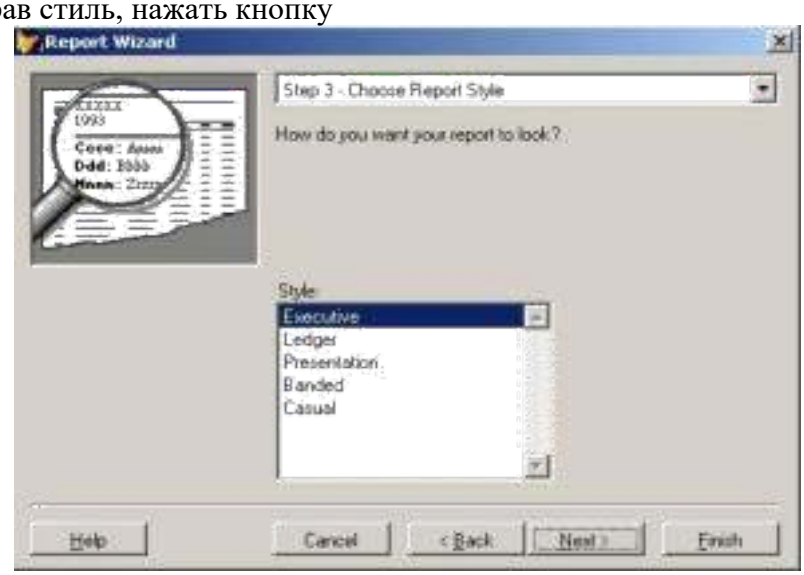


Рисунок 13.3. Диалоговое окно выбора стиля отображения объектов.

Указать порядок размещения объектов в отчете (рис.13.4.) и ориентацию страницы отчета. В области NumberofColumns (Количество колонок) указать 1, в области Orientation (Ориентация) Указать Portrait (Книжная), в области FieldLayout (Расположение полей) указать
Rows (Строка). После установки требуемых опций, нажать кнопку Next

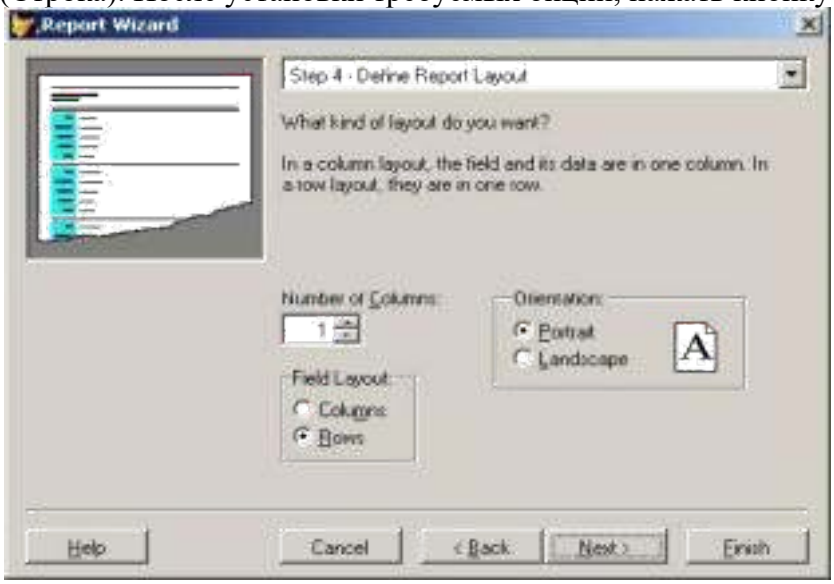


Рисунок 13.4. Диалоговое окно установки порядка размещения объектов.

Задать поле, по которому требуется упорядочение данных в отчете в соответствии с рис.13.5. Из списка Availablefieldsorindextag (Выбранные поля и индексы) перенести в список Selectedfields (Выбранные поля) поле Fio (упорядочивание данных по фамилии). Для переноса полей используйте кнопку Add (Добавить). Сформировав список полей, нажать кнопку Next (Далее);

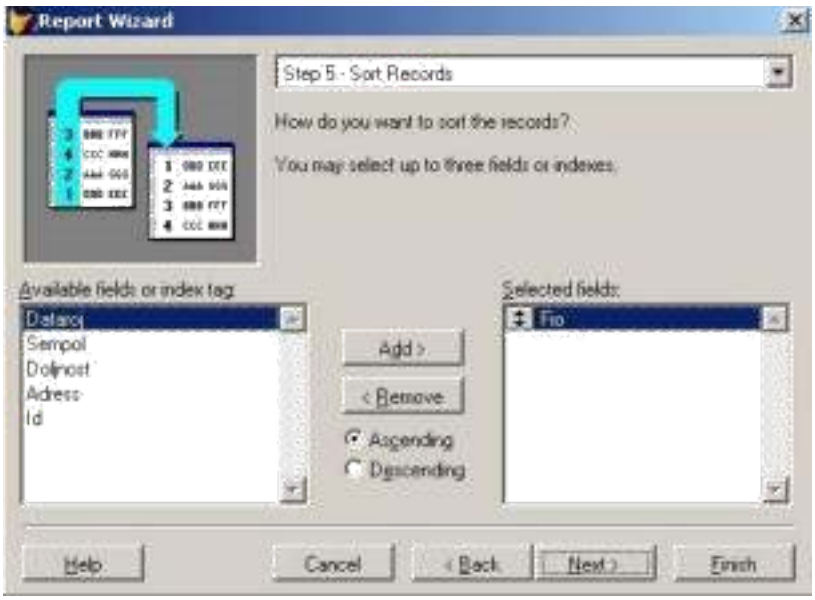


Рисунок 13.5. Диалоговое окно установки критерия упорядочения данных.

На заключительном шаге создания отчета в области Typeatitleforyourreport (Введите имя заголовка отчета) указать имя отчѐта Сотрудники. Воспользовавшись кнопкой Preview (Просмотр) просмотреть, как будет выглядеть создаваемый отчет, после просмотра нажать кнопку ClosePreviewна панелиPrintPreview(см. рис.13.6.),если что-то не так, вернитесь к предыдущим шагам, воспользовавшись кнопкой Back. Нажать кнопку Finish (Готово) и в появившемся диалоговом окне SaveAs (Сохранить как) указать папку, в которой будет размещен отчет Сотрудники.

Рисунок 13.6. Панель просмотра отчѐтаPrintPreview

Просмотреть созданный отчѐт: в окне ProjectManager выбрать вкладку Documents/Reports/Сотрудники/Preview или команду Preview (Просмотр) из меню программы View (Вид)

Открыть отчѐт Сотрудники в окне ReportDesigner (конструктор отчѐта): в окне ProjectManager выбрать вкладку Documents/Reports/Сотрудники/Modify

Таблица 13.1. Типы полос отчѐта

Title (Титул)	В этой полосе размещается информация, появляющаяся перед основным отчётом (имя отчёта, сопроводительное письмо и т. д.) и называется титульной
PageHeader (Верхний колонтитул)	Данные, помещённые в эту полосу, печатаются в начале каждой страницы (название отчёта, текущая дата и т. д.)
Detail (Детали)	Эта полоса содержит данные полей из таблицы или результат вычислений над ними
PageFooter (Нижний колонтитул)	В нижнем колонтитуле печатается название отчёта, дата, номер страницы, итоговые значения по данным текущей страницы

Выделить все объекты в полосе Title (Титул) и перенести их полосу PageHeader (Верхний колонтитул) используя метод перетаскивания мышью или кнопками управления курсором на клавиатуре, предварительно выделив необходимый объект, или

комбинацией клавиш Ctrl+[кнопки управления курсором]

В полосе Title (Титул) напечатать следующий текст: — Отчѐт - форматированное представление данных, выводимое на экран, принтер или в файл. Табличный отчѐт используется для печати данных, представленных в виде списка‖

Поместить набранный текст в прямоугольник со скругленными краями: нажать на панели инструментов ReportControls кнопку  поместить данный объект в отчѐт. Отредактировать его расположение в соответствии с рис.13.7.

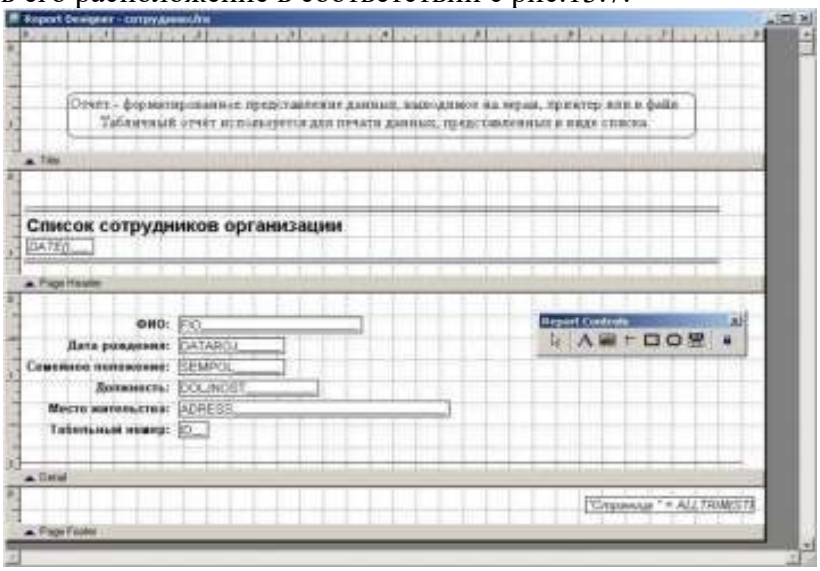
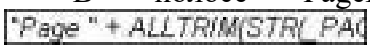


Рисунок 13.7. Окно ReportDesigner (Конструктор отчѐтов)

В процессе работы просматривайте результаты редактирования, используя пункт меню View/Preview

В полосе Detail (Детали), ниже поля Табельный номер провести линию, используя кнопку  панели инструментов ReportControls и отредактировать еѐ расположение в отчѐте в
соответствии с рис.7.

В полосе PageFooter (Нижний колонтитул) переместить элемент  в правый нижний угол в соответствии с рис.137.

Вызвать свойства элемента  отчѐта двойным щелчком мыши или через контекстное меню команда Properties и в появившемся диалоговом окне ReportExpression (Выражение отчѐта) в поле ввода Expression заменить словоPage на слово Страница, нажать OK

Просмотреть результат выполненной работы: пункт меню View/Preview

Индивидуальные задания к практической работе

Создать отчет для таблицы Книги БД Библиотека

В диалоговом окне ReportWizard(Мастер отчетов) (см. рис.1.) выбрать следующие поля таблицы Книги: Nazvanie, Razdel, Izdat, Godizdan

В диалоговом окне выбора стиля отображения объектов (см. рис.3.) в поле области Style
(Стиль) выбрать стиль Executive

В диалоговом окне установки порядка размещения объектов (см. рис.4.) в поле областиNumberofColumns (Количество колонок) указать 1, в области FieldLayout (Расположение полей) указать Rows (Строка)

В диалоговом окне установки критерия упорядочения данных (см. рис.5.) из списка

Availablefieldsorindextag (Выбранные поля и индексы) перенести в список Selectedfields (Выбранные поля) поле Nazvanie

Просмотреть вид создаваемого отчета нажатием кнопки Preview (Просмотр).

После сохранения запустить отчет Книги, она должна иметь вид как на рис.13.8.

Рисунок 13.8. Отчѐт Книги

Практическое занятие № 14 Создание файла проекта базы данных. Создание интерфейса входной формы. Использование исполняемого файла проекта БД, приемы создания и управления.

Цель работы: овладеть навыками проектирования предметной области, создания интерфейсов и применения исполняемого файла БД

Проектирование базы данных с использованием ER-технологии

Для заданной предметной области должен быть определен состав реляционных таблиц и логические связи между таблицами. Для каждого атрибута должны быть заданы тип и размер данных, ограничения целостности. Для каждой таблицы – первичный ключ, потенциальные ключи и внешние ключи.

Разработка логической модели методом «сущность-связь» (ER-методом) предусматривает выполнение следующих шагов, детально описанных в работе:

1. построение ER-диаграммы, включающей все сущности и связи, важные с точки зрения интересов предметной области;
2. анализ связей и определение их характеристик – степени связи, мощности и класса принадлежности;
3. построение набора предварительных отношений с указанием предполагаемого первичного ключа для каждого отношения;
4. подготовка списка всех неключевых атрибутов и назначение каждого из этих атрибутов одному из предварительных отношений;
5. проверка нахождения всех полученных отношений в нормальной форме Бойса-Кодда; 6. построение модели данных.

Создание и связывание таблиц базы данных в среде MySQL

Рассмотрим следующие вопросы:

- создание и выбор базы данных;
- создание таблиц;
- столбцы и типы данных в MySQL;
- создание индексов;
- удаление таблиц, индексов и баз данных; • изменение структуры таблиц.

Базы данных, таблицы и индексы легко создаются в рамках графического интерфейса MySQL, но мы будем использовать монитор MySQL (клиент командной строки), чтобы лучше понять структуру БД, таблиц и индексов.

Чувствительность к регистру и идентификаторы.

- Имена БД подчиняются тем же правилам зависимости от регистра символов, каким следуют каталоги операционной системы. Имена таблиц следуют тем же правилам, что и имена файлов. Все остальное не зависит от регистра.
- Все идентификаторы, кроме имен псевдонимов, могут содержать до 64 символов. Имена псевдонимов могут иметь до 255 символов.
- Идентификаторы могут содержать любые допустимые символы, но имена баз данных не могут содержать символы /, \ и . , а имена таблиц – символы . и /.
- Зарезервированные слова можно использовать для идентификаторов, если заключить их в кавычки.

Комментарий в SQL. Начинается с двух дефисов (--), за которыми должен следовать пробел. Кроме того, MySQL содержит ряд собственных комментариев. Shell-комментарий # действует аналогично – все, что расположено правее его, является текстом комментария. Скомментарий /* */ является многострочным – комментарий начинается с /* и заканчивается, когда встретится завершение */.

Создание и выбор базы данных. Осуществляется с помощью оператора

CREATE DATABASE имя_базы_данных;

Убедиться в том, что оператор выполнил задачу, можно с помощью оператора

SHOW DATABASES;

Теперь имеется пустая БД, ожидающая создания таблиц. Прежде чем работать с БД, необходимо выбрать эту БД с помощью оператора

USE имя_базы_данных;

Теперь все действия по умолчанию будут применяться именно к этой БД.

Создание таблиц. Используется оператор *CREATE TABLE*, который в общем виде выглядит следующим образом:

CREATE [TEMPORARY] TABLE [IF NOT EXISTS]
имя_таблицы (определение таблицы)
[TYPE=тип_таблицы];

Ключевое слово *TEMPORARY* используется для создания таблиц, которые будут существовать только в текущем сеансе работы с БД и будут автоматически удалены, когда сеанс завершится.

При использовании выражения *IF NOT EXISTS* таблица будет создана только в том случае, если еще нет таблицы с указанным именем.

Создать таблицу с такой же схемой, как у существующей, позволяет
команда *CREATE [TEMPORARY] TABLE [IF NOT EXISTS]*
имя_таблицы LIKE имя_старой_таблицы;

После имени таблицы в скобках объявляются имена столбцов, их типы и другая информация. В определение столбца можно добавить следующие описания.

- Объявить для любого столбца *NOT NULL* или *NULL* (столбцу запрещено или не запрещено содержать значения *NULL*). По умолчанию – *NULL*.
- Объявить для столбца значение по умолчанию, используя ключевое слово *DEFAULT*, за которым должно следовать значение по умолчанию.
- Использовать ключевое слово *AUTO_INCREMENT*, чтобы генерировать порядковый номер. Автоматически генерируемое значение будет на единицу большим, чем наибольшее значение в таблице. Первая введенная строка будет иметь порядковый номер 1. В таблице можно иметь не более одного столбца *AUTO_INCREMENT*, и он должен индексироваться.
- Объявить столбец первичным ключом таблицы с помощью выражения *PRIMARY KEY*.

- Объявить столбец внешним ключом, используя выражение *FOREIGN KEY*, с ссылкой на соответствующую таблицу с помощью выражения *REFERENCES*.
- Индексировать столбец с помощью слов *INDEX* или *KEY* (синонимы). Такие столбцы не обязательно должны содержать уникальные значения.
- Индексировать столбец с помощью слова *UNIQUE,* которое используется для указания того, что столбец должен содержать уникальные значения.
- Создать полнотекстовые индексы на основе столбцов типа *TEXT, CHAR* или *VARCHAR* с помощью слова *FULLTEXT* (только с таблицами *MyISAM*). После закрывающей скобки можно указать тип таблицы:
- *MyISAM* – таблицы этого типа являются «родными» для MySQL, работают очень быстро и поддерживают полнотекстовую индексацию;
- *InnoDB* – ACID-совместимый механизм хранения, поддерживающий транзакции, внешние ключи, каскадное удаление и блокировки на уровне строк;
- *BDB (Berkeley DB)* – является механизмом хранения, который обеспечивает поддержку транзакций и блокировки на уровне страниц;
- *MEMORY (HEAP)* – таблицы целиком хранятся в оперативной памяти и никогда не записываются на диск, поэтому работают очень быстро, но ограничены в размерах и не допускают возможности восстановления в случае отказа системы;
- *MERGE* – тип позволяет объединить несколько таблиц *MyISAM* с одной структурой, чтобы к ним можно было направлять запросы как к одной таблице;
- *NDB Cluster* – тип предназначен для организации кластеров MySQL, когда таблицы распределены между несколькими компьютерами, объединенными в сеть;
- *ARCHIVE* – тип введен для хранения большого объема данных в сжатом формате; таблицы поддерживают только два SQL-оператора: *INSERT* и *SELECT,* причем оператор *SELECT* выполняется по методу полного сканирования таблицы;
- *CSV* – формат представляет собой обычный текстовый фал, записи в котором хранятся в строках, а поля разделены точкой с запятой (широко распространен в компьютерном мире, любая программа, поддерживающая CSV-формат, может открыть такой файл);
- *FEDERATED* – тип позволяет хранить данные в таблицах на другой машине сети (при создании таблицы в локальной директории создается только файл определения структуры таблицы, а все данные хранятся на удаленной машине).

MySQL поддерживает следующие типы данных, допустимые для столбцов:

- числовые;
- строковые;
- календарные;
- *NULL* – специальный тип, обозначающий отсутствие информации.

Числовые типы используются для хранения чисел и представляют два подтипа: • точные числовые типы;

- приближенные числовые типы.

К точным числовым типам (табл. 1) относятся целый тип *INTEGER* и его вариации, а также вещественный тип *DECIMAL* (синонимы *NUMERIC* и *DEC*). Последний используется для представления денежных данных.

Числовые типы могут характеризоваться максимальной длиной *M*. Для типа *DECIMAL* параметр *M* задает число символов для отображения всего числа, а *D* – для его

дробной части. Например: *b_price DECIMAL (5, 2)*. Цифра 5 определяет общее число символов под число, а цифра 2 – количество знаков после запятой (интервал величин от –99.99 до 99.99). Можно не использовать параметры вообще, указать только общую длину или указать длину и число десятичных разрядов.

Объявления точных числовых типов можно завершать ключевыми словами *UNSIGNED* и (или) *ZEROFILL*. Ключевое слово *UNSIGNED* указывает, что столбец содержит только положительные числа или нули. Ключевое слово *ZEROFILL* означает, что число будет отображаться с ведущими нулями.

Таблица 14.1. Числовые типы данных

Тип	**Объем памяти**	**Диапазон**
TINYINT (M) *TINYINT UNSIGNED*	1 байт	от -128 до 127 (от -2^7 до 2^7-1) от 0 до 255 (от 0 до 2^8-1)
SMALLINT (M) *SMALLINT UNSIGNED*	2 байта	от -32 768 до 32 767 (от -2^{15} до $2^{15}-1$) от 0 до 65 535 (от 0 до $2^{16}-1$)
MEDIUMINT (M) *MEDIUMINT UNSIGNED*	3 байта	от -8 388 608 до 8 388 607 (от -2^{23} до $2^{23}-1$) от 0 до 16 777 215 (от 0 до $2^{24}-1$)
INT (INTEGER) (M) *INT UNSIGNED*	4 байта	от -2 147 683 648 до 2 147 683 647 (от -2^{31} до $2^{31}-1$) от 0 до 4 294 967 295 (от 0 до $2^{32}-1$)
BIGINT (M) *BIGINT UNSIGNED*	8 байт	(от-2^{63} до$2^{63}-1$) (от 0 до $2^{64}-1$)
BIT (M)	(*M*+7)/8 байт	От 1 до 64 битов, в зависимости от значения *M*
BOOL, BOOLEAN	1 байт	0 (*false*) либо 1 (*true*)
DECIMAL (M, D), NUMERUC (M, D)	*M* + 2 байта	Повышенная точность, зависит от параметров *M* и *D*

К приближенным числовым типам (табл. 14.2) относятся:

- *FLOAT* – представление чисел с плавающей запятой с обычной точностью;
- *DOUBLE* – представление чисел с плавающей запятой с двойной точностью.

,D)

Числовые типы с плавающей точкой также могут иметь параметр *UNSIGNED*. Атрибут предотвращает хранение в столбце отрицательных величин, но максимальный интервал величин столбца остается прежним.

Приближенные числовые данные могут задаваться в обычной форме (например, 45.67) и в форме с плавающей точкой (например, 5.456E-02 или 4.674E+04).

Текстовые типы и строки (табл. 3):

- *CHAR* – хранение строк фиксированной длины;
- *VARCHAR* – хранение строк переменной длины;
- *TEXT, BLOB* и их вариации – хранение больших фрагментов текста;
- *ENUM* и *SET* – хранение значений из заданного списка.

Таблица 14.3. Тестовые типы и строки

Тип	**Объем памяти**	**Максимальный размер**
CHAR(M)	*M* символов	*M* символов
VARCHAR(M)	*L*+1 символов	*M* символов
TINYBLOB, TINYTEXT	*L*+1 символов	2^8-1 символов
BLOB, TEXT	*L*+2 символов	$2^{16}-1$ символов
MEDIUMBLOB,	*L*+3 символов	$2^{24}-1$ символов

MEDIUMTEXT		
LONGBLOB, LONGTEXT	*L*+4 символов	2^{32}-1 символов
ENUM('value 1', 'value2', ...)	1 или 2 байта	65 535 элементов
SET('value 1', 'value2', ...)	1, 2, 3, 4 или 8 байт	64 элемента

Здесь *L* – длина хранимой в ячейке строки, а приплюсованные к *L* байты – накладные расходы для хранения длины строки.

Для строк *VARCHAR* требуется количество символов, равное длине строки плюс 1 байт, тогда как тип *CHAR(M)*, независимо от длины строки, использует для ее хранения все *M* символов. Тип *CHAR* обрабатывается эффективнее переменных типов. Нельзя смешивать в таблице столбцы *CHAR* и *VARCHAR*. Если есть столбец переменной длины, все столбцы типа *CHAR* будут приведены к типу *VARCHAR*.

Типы *BLOB* и *TEXT* аналогичны и отличаются в деталях. При выполнении операций над столбцами типа *TEXT* учитывается кодировка, а типа *BLOB* – нет. Тип *TEXT* используется для хранения больших объемов текста, тип *BLOB* – для больших двоичных объектов (электронные документы, изображения, звук). Основное отличие *TEXT* от *CHAR* и *VARCHAR* – поддержка полнотекстового поиска.

Строки типов данных *ENUM* и *SET* принимают значения из заданного списка. Значение типа *ENUM* должно содержать точно одно значение из указанного множества, тогда как столбцы *SET* могут содержать любой или все элементы заданного множества одновременно. Для типа *SET* (как и для *ENUM*) при объявлении задается список возможных значений, но ячейка может принимать любое значение из списка, а пустая строка означает, что ни один из элементов списка не выбран.

Типы *ENUM* и *SET* задаются списком строк, но во внутреннем представлении элементы множеств сохраняются в виде чисел. Элементы типа *ENUM* нумеруются последовательно, начиная с 1. Под столбец может отводиться 1 байт (до 256 элементов в списке) или 2 байта (от 257 до 65536 элементов в списке). Элементы типа *SET* обрабатываются как биты, размер типа определяется числом элементов в списке: 1 байт (от 1 до 8 элементов), 2 байта (от 9 до 16 элементов), 3 байта (от 17 до 24 элементов), 4 байта (от 25 до 32 элементов) и 8 байт (от 33 до 64 элементов).

Календарные типы данных (табл. 14.4):

- *DATE* – для хранения даты (формат *YYYY-MM-DD* для дат вида 2009-10-15 и формат *YY-MM-DD* для дат вида 09-10-15);
- *TIME* – для хранения времени суток (формат *HH:MM:SS*, где *HH* – часы, *MM* – минуты, *SS* – секунды, например, 10:48:56);
- *DATETIME*– для представления и даты, и времени суток;
- *TIMESTAMP* – если в соответствующем столбце строки не указать конкретное значение или *NULL*, там будет записано время, когда соответствующая строка была создана или в последний раз изменена (в формате *DATETIME*); • *YEAR* – позволяет хранить только год.

Таблица 14.4. Календарные типы данных

Тип	Объем памяти	Диапазон
DATE	3 байта	от '1000-01-01' до '9999-12-31'
TIME	3 байта	от '-828:59:59' до '828:59:59'
DATATIME	8 байт	от '1000-01-01 00:00:00' до '9999-12-31 00:00:00'
TIMESTAMP (M)	4 байта	от '1970-01-01 00:00:00' до '2038-12-31 59:59:59'
YEAR(2)	1 байт	формат *YY,* диапазон – от 1970 до 2069 формат *YYYY*, диапазон – от 1901 до 2155

YEAR(4)		

Дни, месяцы, часы, минуты и секунды можно записывать как с ведущим нулем, так и без него. Например, все следующие записи идентичны:

'2009-04-06 02:04:08' '2009-4-06 02:04:08' '2009-4-6 02:04:08'
'2009-4-6 2:04:08' '2009-4-6 2:4:08' '2009-4-6 2:4:8'

В качестве разделителя между годами, месяцами, днями, часами, минутами, секундами может выступать любой символ, отличный от цифры. Так, следующие значения идентичны:

'09-12-31 11:30:45' '09.12.31 11+30+45' '09/12/31 11*30*45'

При указании времени после секунд через точку можно указать микросекунды, т. е.
использовать расширенный формат вида *HH:MM:SS.FFFFFF*, например '10:25:14.000001'. Кроме того, можно использовать краткие форматы *HH:MM* и *HH* (вместо пропущенных величин будут подставлены нулевые значения).

Если время задается в недопустимом формате, то в поле записывается нулевое значение. Нулевое значение присваивается полям временного типа по умолчанию, когда им не присваивается инициирующее значение (табл. 14.5).

Таблица 14.5. Значения времен полей временного типа

Тип	Нулевое значение
DATE	'0000-00-00'
TIME	'00:00:00'
DATATIME	'0000-00-00 00:00:00'
TIMESTAMP	00000000000000
YEAR	0000

Формат *TIMESTAMP* совпадает с *DATETIME*, но во внутреннем представлении дата хранится как число секунд, прошедших с полуночи 1 января 1970 г. (такое исчисление принято в операционной системе UNIX, а дата 01.01.1970 считается началом эпохи UNIX и днем рождения операционной системы).

Если в таблице несколько столбцов *TIMESTAMP*, при модификации записи текущее время будет записываться только в один из них (первый). Можно явно указать столбец, которому необходимо назначать текущую дату при создании или изменении записи. Чтобы поля принимали текущую дату при создании записи, следует после определения столбца добавить *DEFAULT CURRENT_TIMESTAMP*. Если текущее время должно выставляться при модификации записи, при использовании оператора *UPDATE* следует добавить *ON UPDATE CURRENT_TIMESTAMP*.

Тип данных *NULL* используется, когда информации недостаточно и для части данных нельзя определить, какое значение они примут. Для указания того, что поле может принимать неопределенное значение, в определении столбца после типа данных следует указать ключевое слово *NULL*. Если поле не должно принимать значение *NULL*, следует указать ключевое слово
NOT NULL.

Рекомендации по выбору типа данных.

- Обработка числовых данных происходит быстрее строковых. Так как типы *ENUM* и *SET* имеют внутреннее числовое представление, им следует отдавать предпочтение перед другими видами строковых данных, если это возможно.
- Производительность можно увеличить за счет представления строк в виде чисел. Пример – преобразование IP-адреса из строки в *BIGINT*. Это позволит уменьшить размер таблицы и значительно увеличить скорость при сортировке и выборке данных, но потребует дополнительных преобразований.

- Базы данных хранятся на жестком диске, и чем меньше места они занимают, тем быстрее происходит поиск и извлечение. Если есть возможность, следует выбирать типы данных, занимающие меньше места.
- Типы фиксированной длины обрабатываются быстрее типов переменной длины, т. к. в последнем случае при частых удалениях и модификациях таблицы происходит ее фрагментация.
- Если применение столбцов с данными переменной длины неизбежно, для дефрагментации таблицы следует применять команду *OPTIMIZE TABLE.*

Обеспечение ссылочной целостности. Задается конструкцией:

FOREIGN KEY [name_key] (col1, ...) REFERENCES tbl (tbl_col, ...)
[ON DELETE {CASCADE | SET NULL | NO ACTION | RESTRICT | SET DEFAULT}]
[ON
UPDATE {CASCADE | SET NULL | NO ACTION | RESTRICT | SET DEFAULT}]

Конструкция позволяет задать внешний ключ с необязательным именем *name_key* на столбцах, которые задаются в круглых скобках (один или несколько). Ключевое слово *REFERENCES* указывает таблицу *tbl*, на которую ссылается внешний ключ, в круглых скобках указываются имена столбцов. Необязательные конструкции *ON DELETE* и *ON UPDATE* позволяют задать поведение СУБД при удалении и обновлении строк из таблицы-предка. Параметры, следующие за этими ключевыми словами, имеют следующие значения:

- *CASCADE* – при удалении или обновлении записи в таблице-предке, содержащей первичный ключ, записи со ссылками на это значение в таблице-потомке удаляются или обновляются автоматически;
- *SET NULL* – при удалении или обновлении записи в таблице-предке, содержащей первичный ключ, в таблице-потомке значения внешнего ключа, ссылающегося на таблицу-предка, устанавливаются в *NULL*;
- *NO ACTION* – при удалении или обновлении записей, содержащих первичный ключ, с таблицей-потомком никаких действий не производится;
- *RESTRICT* – если в таблице-потомке имеются записи, ссылающиеся на первичный ключ таблицы-предка, при удалении или обновлении записей с таким первичным ключом возвращается ошибка;
- *SET DEFAULT* – согласно стандарту *SQL*, при удалении или обновлении первичного ключа в таблице-потомке для ссылающихся на него записей в поле внешнего ключа должно устанавливаться значение по умолчанию (в MySQL это ключевое слово зарезервировано, но не обрабатывается).

Создание индексов. Индексы играют большую роль в БД, т. к. это основной способ ускорения их работы. Записи в таблице располагаются хаотически. Чтобы найти нужную запись, необходимо сканировать всю таблицу, на что уходит много времени. Идея индексов состоит в том, чтобы создать для столбца копию, которая постоянно будет поддерживаться в отсортированном состоянии. Это позволяет быстро осуществлять поиск по такому столбцу.

Все необходимые индексы формируются при создании таблицы. Индексированы будут все столбцы, объявленные как *PRIMARY KEY, KEY, UNIQUE* или *INDEX*. Индекс также можно добавить с помощью оператора *CREATE INDEX*. Перед выполнением оператор преобразуется в оператор *ALTER TABLE*. Например, создание индекса с именем *name* на основе поля *u_name* из таблицы *users*:

CREATE INDEX name ON users (u_name);

Перед ключевым словом *INDEX* может присутствовать *UNIQUE*, требующее уникальности ограничения.

Корректность таблиц в БД можно проверить с помощью оператора

SHOW TABLES;

Более подробную информацию о структуре таблицы дает команда
DESCRIBE имя_таблицы;

Переименование БД. Специального оператора переименования БД нет, но можно переименовать каталог БД в системном каталоге (*...\DATA*).

Удаление БД. Удалить всю БД вместе с ее содержимым можно командой:
DROP DATABASE [IF EXISTS] имя_базы_данных;

Удаление таблиц и индексов. Удалить таблицу можно с помощью оператора:
DROP TABLE [IF EXISTS] имя_таблицы;

Удалить индекс можно с помощью
оператора:
DROP INDEX имя_индекса ON имя_таблицы;

Изменение структуры таблиц. Изменить структуру существующей таблицы можно с помощью оператора *ALTER TABLE*. Например, можно создать индекс *name* для таблицы *users* следующим образом: *ALTER TABLE users ADD INDEX name (u_name);*

Оператор *ALTER TABLE* является исключительно гибким, поэтому он имеет огромное множество дополнительных ключевых слов.

Задание для практической работы

При выполнении практической работы необходимо для заданной предметной области средствами MySQL:

- создать базу данных;
- создать таблицы, определить поля таблиц, индексы;
- определить связи между таблицами и ограничения целостности; • составить отчет по практической работе.

Создайте базу данных *book* Интернет-магазина, торгующего компьютерной литературой. В базе данных должна поддерживаться следующая информация:

- тематические каталоги, по которым сгруппированы книги;
- предлагаемые книги (название, автор, год издания, цена, имеющееся на складе количество);
- зарегистрированные покупатели (имя, отчество, фамилия, телефон, адрес электронной почты, статус – авторизованный, неавторизованный, заблокированный, активный с хорошей кредитной историей);
- покупки, совершенные в магазине (время совершения покупки, число приобретенных экземпляров книги).

Логическая модель данных предметной области в стандарте IDEF1X представлена на рис. 14.1. Выделены сущности *КАТАЛОГ, КНИГА, КЛИЕНТ, ЗАКАЗ*, между которыми установлены не идентифицирующие связи мощностью один-ко-многим, определенные спецификой предметной области.

Рисунок 14.1. Логическая модель данных предметной области

Физическая модель данных предметной области в стандарте IDEF1X для целевой СУБД MySQL представлена на рис. 14.2.

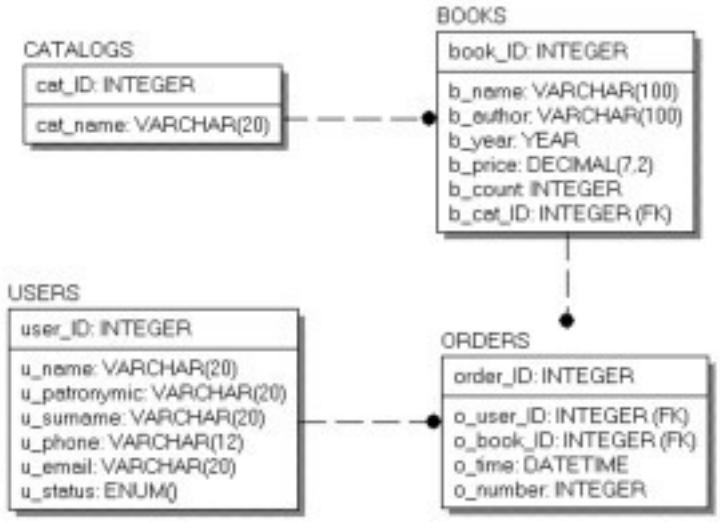


Рисунок 14.2. Физическая модель предметной области

База данных *book* состоит из четырех таблиц:

- *catalogs* – список торговых каталогов;
- *books* – список предлагаемых книг;
- *users* – список зарегистрированных пользователей магазина; • *orders* – список заказов (осуществленных сделок). Таблица *catalogs* состоит из двух полей: • *cat_ID* – уникальный код каталога;
- *cat_name* – имя каталога.

Оба поля должны быть снабжены атрибутом *NOT NULL*, поскольку неопределенное значение для них недопустимо.

Таблица *books* состоит из семи полей:

- *book_ID* – уникальный код книги;
- *b_name* – название книги;
- *b_author* – автор книги;
- *b_year* – год издания;
- *b_price* – цена книги;
- *b_count* – количество книг на складе;

- *b_cat_ID* – код каталога из таблицы *catalogs*.

Цена книги *b_price* и количество экземпляров на складе *b_count* могут иметь атрибут *NULL*. На момент доставки часто неизвестны количество товара и его цена, но отразить факт наличия товара в прайс-листе необходимо.

Поле *b_cat_ID* устанавливает связь между таблицами *catalogs* и *books*. Это поле должно быть объявлено как внешний ключ (FK) с правилом каскадного удаления и обновления. Обновление таблицы *catalogs* вызовет автоматическое обновление таблицы *books*. Удаление каталога в таблице *catalogs* приведет к автоматическому удалению всех записей в таблице *books*, соответствующих каталогу.

Таблица *users* состоит из семи полей:

- *user_ID* – уникальный код покупателя;
- *u_name* – имя покупателя;
- *u_patronymic* – отчество покупателя;
- *u_surname* – фамилия покупателя;
- *u_phone* – телефон покупателя (если имеется); • *u_email* – e-mail покупателя (если имеется);
- *u_status* – статус покупателя.

Статус покупателя представлен полем типа *ENUM*, которое может принимать одно из четырех значений:

- *active* – авторизованный покупатель, который может осуществлять покупки через Интернет;
- *passive* – неавторизованный покупатель (значение по умолчанию), который осуществил процедуру регистрации, но не подтвердил ее и пока не может осуществлять покупки через Интернет, однако ему доступны каталоги для просмотра;
- *lock* – заблокированный покупатель, не может осуществлять покупки и просматривать каталоги магазина;
- *gold* – активный покупатель с хорошей кредитной историей, которому предоставляется скидка при следующих покупках в магазине.

Поля *u_phone* и *u_email* могут быть снабжены атрибутом *NULL*. Остальные поля должны получить атрибут *NOT NULL*.

Таблица *orders* включает пять полей:

- *order_ID* – уникальный номер сделки;
- *o_user_ID* – номер пользователя из таблицы *users*;
- *o_book_ID* – номер товарной позиции из таблицы *books*; • *o_time* – время совершения сделки; • *o_number* – число приобретенных товаров.

Поля таблицы *orders* должны быть снабжены атрибутом *NOT NULL*, т. к. при совершении покупки вся информация должна быть занесена в таблицу.

В таблице *orders* устанавливается связь с таблицами *users* (за счет поля *o_user_ID*) и *books* (за счет поля *o_book_ID*). Эти поля объявлены как внешние ключи (FK) с правилом каскадного удаления и обновления. Обновление таблиц *users* и *books* приведет к автоматическому обновлению таблицы *orders*. Удаление любого пользователя в таблице *users* приведет к автоматическому удалению всех записей в таблице *orders*, соответствующих этому пользователю.

Операторы создания БД *book* имеют следующий вид (целесообразно создать в *Блокноте* текстовый файл и записать туда эти операторы).

```
DROP DATABASE IF EXISTS book;
CREATE DATABASE book;
USE book;
CREATE TABLE catalogs (
```

```
        cat_ID int(6) NOT NULL AUTO_INCREMENT,
    cat_name varchar(20) NOT NULL,
        PRIMARY KEY  (cat_ID)
    ) TYPE=InnoDB;
    CREATE TABLE books (
        book_ID int(6) NOT NULL
    AUTO_INCREMENT,          b_name
    varchar(100) NOT NULL,   b_author
    varchar(100) NOT NULL,   b_year year NOT
    NULL,
        b_price decimal(7,2) NULL default
    '0.00',  b_count int(6) NULL default '0',
    b_cat_ID int(6) NOT NULL default '0',
        PRIMARY KEY  (book_ID),
        FOREIGN KEY (b_cat_ID) REFERENCES catalogs(cat_ID)
    ON DELETE CASCADE ON UPDATE CASCADE ) TYPE=InnoDB;
    CREATE TABLE users (
        user_ID int(6) NOT NULL
    AUTO_INCREMENT,          u_name varchar(20)
    NOT NULL,   u_patronymic varchar(20) NOT
    NULL,          u_surname varchar(20) NOT
    NULL,          u_phone varchar(12) NULL,
        u_email varchar(20) NULL,
      u_status ENUM ('active','passive','lock','gold') default 'passive',
        PRIMARY KEY  (user_ID)
    ) TYPE=InnoDB;
    CREATE TABLE orders (
        order_ID int(6) NOT NULL AUTO_INCREMENT,
    o_user_ID int NOT NULL,
        o_book_ID int NOT NULL,
      o_time datetime NOT NULL default '0000-00-00 00:00:00',
       o_number int(6) NOT NULL default '0',
        PRIMARY KEY  (order_ID),
        FOREIGN KEY (o_book_ID) REFERENCES books(book_ID)
    ON DELETE CASCADE ON UPDATE CASCADE,
     FOREIGN KEY (o_user_ID) REFERENCES users(user_ID) ON DELETE CASCADE
ON UPDATE CASCADE
    )TYPE=InnoDB;
```

Практическое занятие № 15 Создание формы. Управление внешним видом формы.

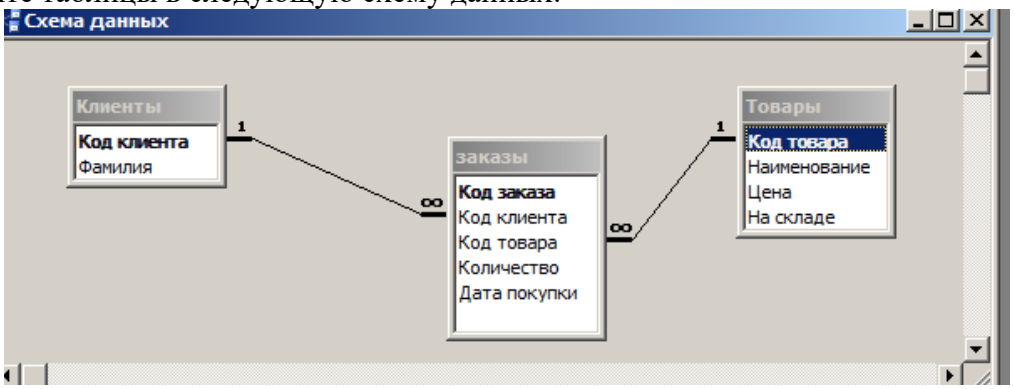
Постановка задачи. Построить базу данных, содержащую информацию о товарах, имеющихся на складе, о клиентах, выполняющих заказы и покупки, о сотрудниках фирмы. Необходимо предусмотреть возможность выборки информации по различным критериям, создать формы для ввода информации и отчеты для подготовки печатных документов.

Порядок выполнения задачи.

1. Создание базы данных. В меню *Файл* выберите команду *Создать* и из списка шаблонов на закладке *Общие* выберите шаблон «База данных». Сохраните создаваемую БД под именем *Торговая организация.mdb*.

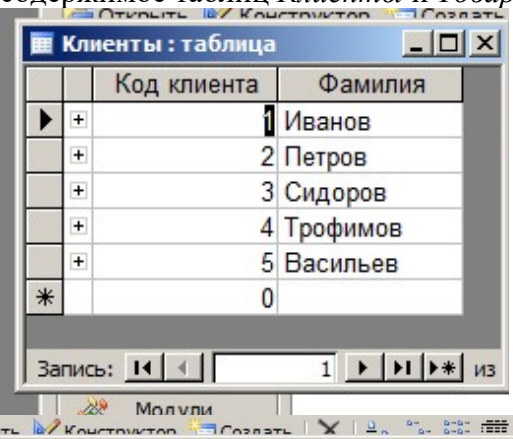
2. С помощью конструктора создайте следующие таблицы: *Клиенты*, содержащую информацию о клиентах (поля: Код клиента, Фамилия); *Товары*, содержащую информацию по товарам (поля: Код товара, Наименование, Цена, На складе) и *Заказы* (поля: Код заказа, Код клиента, Код товара, Количество, Дата покупки).

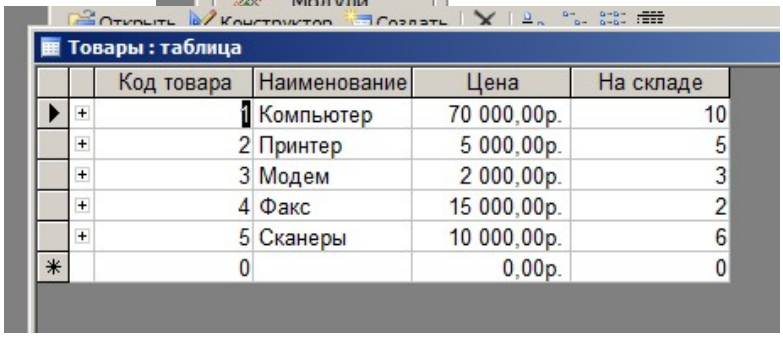
3. Свяжите таблицы в следующую схему данных:



4. Выполните подстановку поля *Код клиента* из таблицы *Клиенты* в поле *Код клиента* таблицы *Заказы* и поля *Код товара* таблицы *Товары* в поле *Код товара* таблицы *Заказы*.

5. Заполните таблицы данными, введя в каждую из них не менее пяти строк. Примерное содержимое таблиц *Клиенты* и *Товары*:



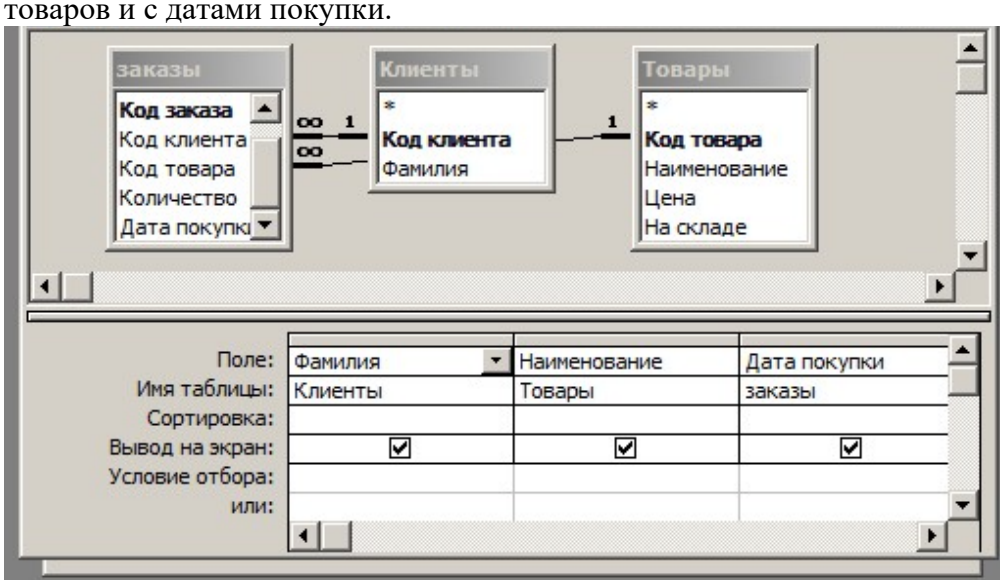


Примерное содержимое таблицы *Заказы*:

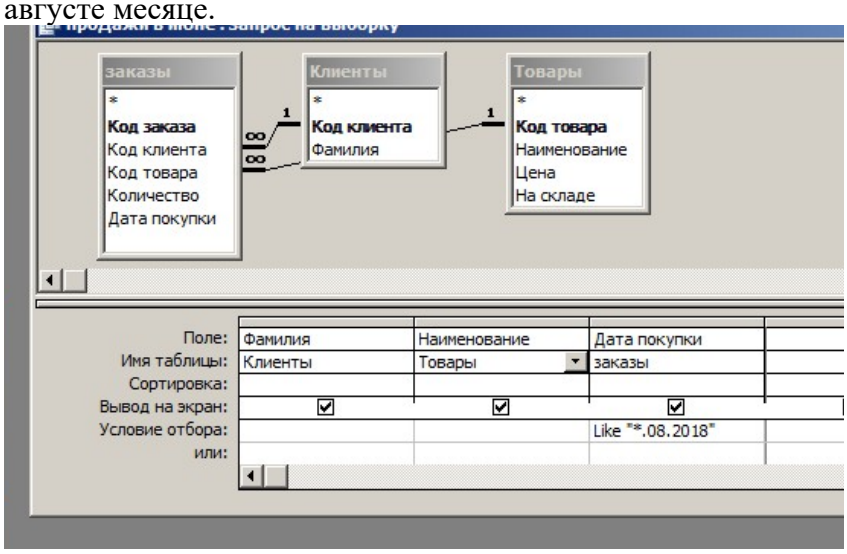
заказы : таблица

Код заказа	Код клиента	Код товара	Количество	Дата покупки
1	1	1	2	20.01.2018
2	2	1	3	02.03.2018
3	3	4	3	10.07.2018
4	1	3	2	15.08.2018
5	5	1	1	17.08.2018
0	0	0	0	

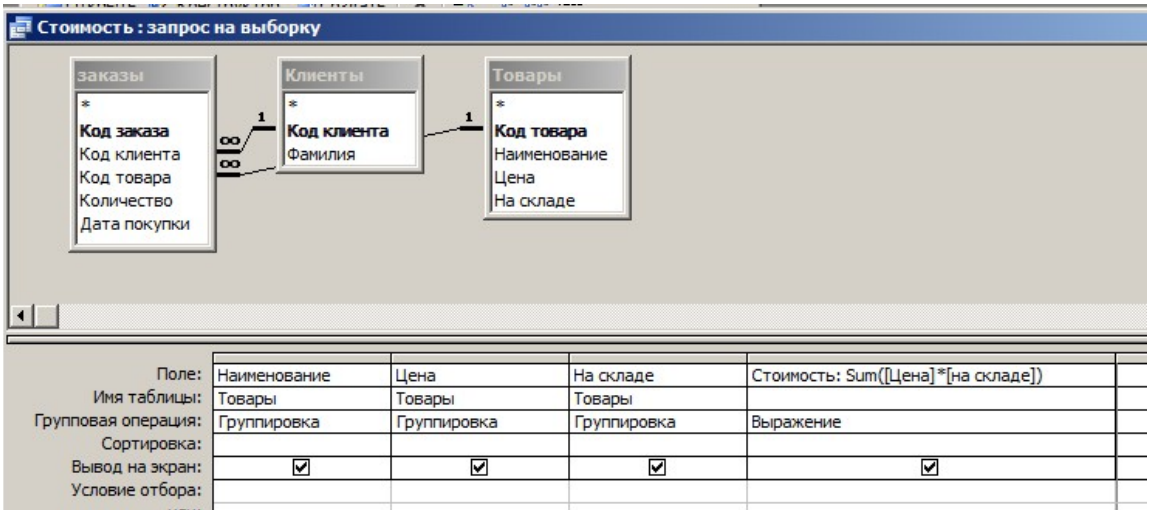
6. Создайте запрос *Заказы*, который выводит список клиентов с названиями заказанных товаров и с датами покупки.



7. Создайте запрос *Покупки в августе*, выводящий список клиентов, сделавших покупки в августе месяце.



8. Создайте запрос *Стоимость* для просмотра общей стоимости каждого товара на складе (цена* на складе).



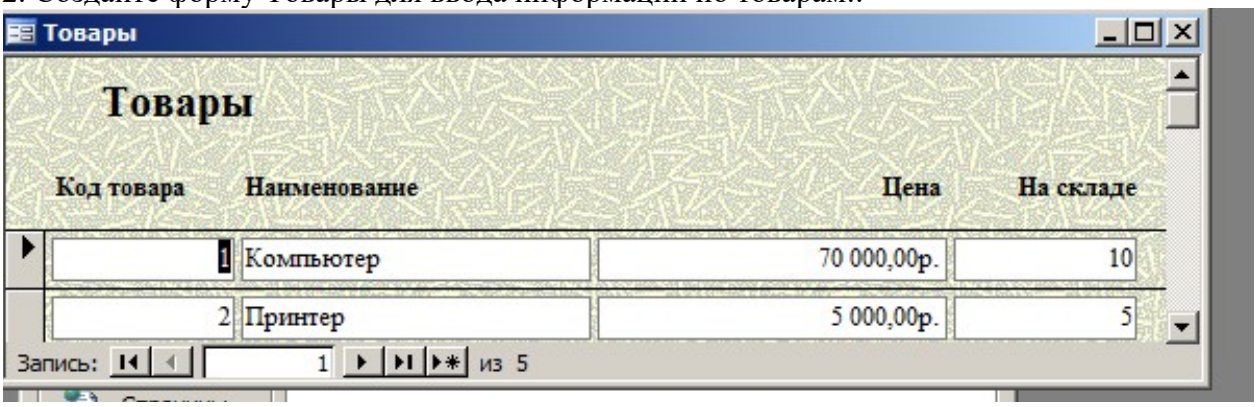
9. Создайте запрос *Покупки*, результатом которого стала бы таблица *Покупки*, содержащая информацию о фамилии клиента, наименовании, цене и количестве купленного им товара, а также дате приобретения и общей заплаченной сумме.

10. На основе запроса *Покупки* создайте перекрестный запрос Количество товаров, который выводил бы информацию следующего вида:

Фамилия	Итоговое значе	Компьютер	Модем	Факс
Васильев	1	1		
Иванов	4	2	2	
Петров	3	3		
Сидоров	3			3

11. Создайте запрос на удаление из таблицы *Заказы* записей, относящихся к 1 кварталу 2018 года.

12. Создайте форму Товары для ввода информации по товарам:.



13. Создайте форму с заголовком *Заказы*, в которой выводилась бы фамилия клиента, количество заказанных товаров и дата покупки.

14. Создайте ленточный отчет с заголовком *Товары на складе* об имеющихся на складе товарах, их цене, количестве и общей стоимости.

15. Создайте макросы для запуска форм и отчетов и закрепите их за кнопками главной кнопочной формы следующего вида:

Практическое занятие № 16 Задание значений и ограничений поля. Проверка введенного в поле значения. Отображение данных числового типа и типа дата

Цель работы

1. Изучить основные типы данных.
2. Изучить встроенные функции для работы со строками.
3. Изучить встроенные функции для работы с числами.
4. Изучить встроенные функции для работы с датами и временем.
5. Изучить встроенные функции преобразования данных.
6. Изучить CASE и IIF.

Теоретическая часть

Типы данных

На языке Transact-SQL используется множество различных *типов данных*. Всех их можно разделить на следующие группы:

Числовые типы данных: BIT (значение 0 или 1), TINYINT (от 0 до 255), SMALLINT (от

-32768 до 32767), INT (от -2147483648 до 2147483647), BIGINT (от -9223372036854775808 до 9223372036854775807), DECIMAL (числа с фиксированной точностью), NUMERIC: (аналогичен типу DECIMAL), SMALLMONEY (дробные значения от -214748.3648 до 214748.3647), MONEY (дробные значения от -922337203685477.5808 до 922337203685477.5807), FLOAT (от -1.79E+308 до 1.79E+308), REAL (числа от –340E+38 до 3.40E+38);

Типы данных, представляющие дату и время: DATE (дата от 01/01/0001 до 31/12/9999), TIME (время в диапазоне от 00:00:00.0000000 до 23:59:59.9999999), DATETIME (дата и время от 01/01/1753 до 31/12/9999), DATETIME2 (дата и время от 01/01/0001 00:00:00.0000000 до 31/12/9999 23:59:59.9999999), SMALLDATETIME (дата и время от 01/01/1900 до 06/06/2079), DATETIMEOFFSET (дата и время от 01/01/0001 до 31/12/9999);

Строковые типы данных: CHAR (фиксированная строка длиной от 1 до 8000 символов), VARCHAR (переменная строка длиной от 1 до 8000 символов), NCHAR (Unicode – фиксированная строка длиной от 1 до 4000 символов), NVARCHAR (Unicode – переменная строка длиной от 1 до 4000 символов), TEXT и NTEXT (устаревшие, не рекомендуется использовать);

Бинарные типы данных: BINARY (фиксированные бинарные данные от 1 до 8000 байт), VARBINARY (переменные бинарные данные от 1 до 8000 байт), IMAGE (устаревшая, не рекомендуется использовать);

Другие типы данных: UNIQUEIDENTIFIER (уникальный идентификатор GUID), TIMESTAMP (номер версии строки в таблице), CURSOR (набор строк таблицы), HIERARCHYID (позиция в иерархии), SQL_VARIANT (данные любого типа), XML (документы или фрагменты XML), TABLE (таблица), GEOGRAPHY (географические данные, такие как широта и долгота), GEOMETRY (координаты на плоскости).

Встроенные функции Transact-SQL

Функции SQL производят действия с данными и возвращают результат. *Встроенные функции* делятся на три основные группы:

– *скалярные функции* – обрабатывают одиночное значение и возвращают одно значение. Их можно использовать везде, где допускается применение выражений.

– *агрегатные функции* – используются для получения обобщающих значений. Они, в отличие от скалярных функций, оперируют значениями столбцов множества строк; – *функции для списка значений*.

Скалярные функции бывают следующих категорий:

– *строковые функции* – выполняют определенные действия над строками и возвращают строковые или числовые значения;

– *числовые функции* – возвращают числовые значения на основании заданных в аргументе значений того же типа;

– *функции времени и даты* – выполняют различные действия над входными значениями времени и даты и возвращают строковое, числовое значение или значение в формате даты и времени;

– *функции преобразования типа*.

Список часто используемых *строковых функций*:

LEN(строка)	возвращает количество символов в заданной строке
TRIM(строка) TRIM([символ FROM] строка)	удаляет символ пробела или другие заданные символы в начале и в конце строки.
LTRIM(строка)	удаляет начальные пробелы из заданной строки
RTRIM(строка)	удаляет конечные пробелы из заданной строки
CHARINDEX(подстрока, строка) CHARINDEX(подстрока, строка, начальная позиция)	возвращает индекс, по которому находится первое вхождение подстроки в строке.
PATINDEX('%шаблон%', строка)	возвращает индекс, по которому находится первое вхождение определенного шаблона в строке
LEFT(строка, число)	возвращает с начала строки определенное количество символов
RIGHT(строка, число)	возвращает с конца строки определенное количество символов
SUBSTRING(строка, начальная позиция, длина)	возвращает подстроку заданной длиной, начиная с данной позиции
REPLACE(строка, подстрока, замена)	заменяет одну подстроку другой
REVERSE(строка)	переворачивает строку наоборот
CONCAT(строка1, строка2 [, строкаN])	объединяет заданные строки в одну
LOWER(строка)	переводит строку в нижний регистр
UPPER (строка)	переводит строку в верхний регистр
SPACE(число)	возвращает заданное количество пробелов
REPLICATE(строка, число)	повторяет значение строки указанное число раз
STUFF(строка, начальная позиция, количество, замена)	удаляет указанное количество символов первой строки в начальной позиции и вставляет на их место замену.

Список часто используемых *числовых функций*:

ABS(число)	возвращает абсолютное значение числа

CEILING(число)	возвращает наименьшее целое, большее или равное заданного числа.
FLOOR(число)	возвращает наибольшее целое число, меньшее или равное заданного числа
POWER(число, степень)	возвращает значение указанного выражения, возведенное в заданную степень
RAND([начальное значение])	возвращает псевдослучайное значение от 0 до 1
ROUND(число, точность)	возвращает число, округленное до указанной точности
SIGN(число)	возвращает положительное (+1), нулевое (0) или отрицательное (-1) значение, обозначающее знак заданного выражения
SQRT(число)	возвращает квадратный корень данного числа
SQUARE(число)	возвращает квадрат указанного числа
PI()	возвращает константное значение π
ACOS(число)	возвращает угол в радианах, косинус которого задан – арккосинус.
ASIN(число)	возвращает угол в радианах, синус которого задан – арксинус.
ATAN(число)	возвращает угол в радианах, тангенс которого задан – арктангенс.
COS(число)	возвращает косинус указанного угла в радианах.
SIN(число)	возвращает синус указанного угла в радианах.
TAN(число)	возвращает тангенс указанного угла в радианах.
COT(число)	возвращает котангенс указанного угла в радианах
DEGREES(число)	возвращает для значения угла в радианах соответствующее значение в градусах.
RADIANS(число)	возвращает для значения угла в градусах соответствующее значение в радианах
EXP(число)	возвращает экспонент заданного числа
LOG(число)	возвращает натуральный логарифм указанного числа
LOG(число, основа)	возвращает логарифм указанного числа
LOG10(число)	возвращает десятичный логарифм указанного числа

Список часто используемых *функций времени и даты*:

GETDATE()	возвращает текущую дату и время
CURRENT_TIMEZONE()	возвращает имя часового пояса
GETUTCDATE()	возвращает текущую дату и время по Гринвичу (UTC/GMT)
DAY(дата)	возвращает день месяца указанной даты
MONTH(дата)	возвращает номер месяца указанной даты
YEAR(дата)	возвращает год указанной даты
DATEPART(часть, дата)	возвращает целое число, представляющее указанную часть заданной даты

DATENAME(часть, дата)	возвращает строку символов, представляющую указанную часть заданной даты
DATEADD(часть, число, дата)	добавляет указанное целое число со знаком к части входного значения даты, а затем возвращает это измененное значение
DATEDIFF(часть, начальная дата, конечная дата)	возвращает разницу как целое число со знаком между частями заданных дат
EOMONTH(дата)	возвращает последний день месяца, заданной даты

Для функций времени и даты используются следующие аргументы как часть даты и времени:

Часть даты и времени	***Сокращения***
year	yy, yyyy
quarter	qq, q
month	mm, m
dayofyear	dy, y
day	dd, d
week	wk, ww
weekday	dw
hour	hh
minute	mi, n
second	ss, s
millisecond	ms
microsecond	mcs
nanosecond	ns
tzoffset	tz
iso_week	isowk, isoww

Список часто используемых *функций преобразования*:

CAST(выражение AS тип)	преобразуют выражение в заданный тип
CONVERT(тип, выражение [, стиль])	
ASCII(строка)	возвращает код ASCII первого символа указанного символьного выражения
UNICODE(строка)	возвращает код Юникод первого символа указанного символьного выражения
CHAR(число)	возвращает символ ASCII с указанным кодом
NCHAR(число)	возвращает символ Юникода с указанным кодом
STR(число)	возвращает символьные данные, преобразованные из числовых данных

Список часто используемых *функций проверки значений*:

ISDATE(выражение)	возвращает 1, если выражение имеет допустимое значение типа даты и времени, иначе возвращает значение 0
ISNUMERIC(выражение)	возвращает 1, если выражение имеет допустимое значение числовой тип данных, иначе возвращает 0
ISNULL(выражение, замена)	заменяет значение NULL указанным замещающим значением

COALESCE(выражение[,...n])	вычисляет аргументы по порядку и возвращает текущее значение первого выражения, изначально не вычисленного как NULL.

Особое место среди встроенных скалярных функций языка SQL занимают функции вывода, которые являются разновидностью CASE-выражений. Функция CASE проверяет значение некоторого выражения, и в зависимости от результата проверки может возвращать тот или иной результат.

Выражение CASE имеет два формата:

– простое выражение CASE для определения результата сравнивает выражение с набором простых выражений;

– поисковое выражение CASE для определения результата вычисляет набор логических выражений.

Оба формата поддерживают дополнительный аргумент ELSE.

Функция IIF(условие, выражение_если_истина, выражение_если_ложь) – возвращает одно из двух значений в зависимости от того, принимает логическое выражение значение true или false.

Практическая часть

Дана таблица ***Академики:***

ФИО	**Дата_рождения**	**Специализация**	**Год_присвоения_звания**
Аничков Николай Николаевич	1885-11-03	медицина	1939
Бартольд Василий Владимирович	1869-11-15	историк	1913
Белопольский Аристарх Апол-лонович	1854-07-13	астрофизик	1903
Бородин Иван Парфеньевич	1847-01-30	ботаник	1902
Вальден Павел Иванович	1863-07-26	химик-технолог	1910
Вернадский Владимир Иванович	1863-03-12	геохимик	1908
Виноградов Павел Гаврилович	1854-11-30	историк	1914
Ипатьев Владимир Николаевич	1867-11-21	химик	1916
Истрин Василий Михайлович	1865-02-22	филолог	1907
Карпинский Александр Петрович	1847-01-07	геолог	1889
Коковцов Павел Константинович	1861-07-01	историк	1906
Курнаков Николай Семёнович	1860-12-06	химик	1913
Марр Николай Яковлевич	1865-01-06	лингвист	1912
Насонов Николай Викторович	1855-02-26	зоолог	1906
Ольденбург Сергей Фёдорович	1863-09-26	историк	1903
Павлов Иван Петрович	1849-09-26	физиолог	1907
Перетц Владимир Николаевич	1870-01-31	филолог	1914
Соболевский Алексей Иванович	1857-01-07	лингвист	1900
Стеклов Владимир Андреевич	1864-01-09	математик	1912

Пример 1: Вывести ФИО академиков и длину ФИО: SELECT

ФИО

,LEN(ФИО) AS Количество_символов

FROM

Академики

Пример 2: Вывести список академиков, убрать лишние пробелы в ФИО:

SELECT

TRIM(ФИО) AS ФИО

,Дата_рождения

,Специализация

,Год_присвоения_звания

FROM

Академики

Пример 3: Найти позиции буквы «о» в ФИО каждого академика. Вывести ФИО и позицию:

```
SELECT
        ФИО
    ,CHARINDEX('о',ФИО) AS Позиция_о    FROM
        Академики
```

Пример 4: Вывести ФИО и первые три буквы специализации каждого академика: SELECT

```
        ФИО
    ,LEFT(Специализация, 3) AS Спец_3    FROM
        Академики
```

Пример 5: Вывести ФИО и от второй до пятой буквы специализации каждого академика:

```
SELECT
        ФИО
    ,SUBSTRING(Специализация, 2, 4) AS Спец_2_5        FROM
        Академики
```

Пример 6: Вывести список академиков, заменить специализацию «лингвист» на «языковед»:

```
SELECT
        ФИО
            ,Дата_рождения
                    ,REPLACE(Специализация, 'лингвист', 'языковед') AS Спец
            ,Год_присвоения_звания
FROM
        Академики
```

Пример 7: Вывести список академиков, специализацию на верхнем регистре: SELECT

```
        ФИО
            ,Дата_рождения
                ,UPPER(Специализация) AS Спец
            ,Год_присвоения_звания
FROM
        Академики
```

Пример 8: Вывести ФИО академиков в правильном и обратном виде: SELECT

```
        ФИО
                ,REVERSE(ФИО) AS ФИО_Обр
FROM
        Академики
        Название
```

Пример 9: Вывести каждую специализацию 4 раза в одной строке. Убрать дубликаты:

```
SELECT DISTINCT
                    REPLICATE(Специализация, 4) AS Спец_4
FROM
Академики
```

Пример 10: Вывести абсолютное значение тригонометрических функций на точке π:

```
SELECT
                ABS(COS(PI()))   AS Косинус_Пи
                ,ABS(SIN(PI()))  AS Синус_Пи
                ,ABS(TAN(PI()))  AS Тангенс_Пи
                ,ABS(COT(PI()))  AS КоТангенс_Пи
```

Пример 11: Вывести число 132.456, округленное с точностью от 3 до -3:

```
SELECT
```

```
ROUND(123.456, 3)   AS Окр3
,ROUND(123.456, 2)   AS Окр2
,ROUND(123.456, 1)   AS Окр1
,ROUND(123.456, 0)   AS Окр0
,ROUND(123.456, -1)   AS Окр_1
,ROUND(123.456, -2)   AS Окр_2
,ROUND(123.456, -3)   AS Окр_3
```

Пример 12: Вывести наименьшее целое число, которое больше или равно 123.456, и наибольшее целое число, которое меньше или равно 123.456:

```
SELECT
CEILING(123.456)  AS Больше
,FLOOR(123.456)    AS Меньше
```

Пример 13: Вывести квадратный корень, квадрат и куб числа 25:

```
SELECT
SQRT(25)     AS Корень
,SQUARE(25)   AS Квадрат
,POWER(25, 3) AS Куб
```

Пример 14: Вывести текущую дату и время:

```
SELECT
GETDATE() AS Сейчас
```

Пример 15: Вывести день, месяц, год, час, минуту, секунду, номер квартала, номер недели, день года, день недели для текущей даты и времени:

```
SELECT
DAY(GETDATE()) AS День
,MONTH(GETDATE()) AS Месяц
,YEAR(GETDATE()) AS Год
,DATEPART(HOUR, GETDATE()) AS Час
,DATEPART(MINUTE, GETDATE()) AS Минута
,DATEPART(SECOND, GETDATE()) AS Секунда
,DATEPART(QUARTER, GETDATE()) AS Квартал
,DATEPART(WEEK, GETDATE()) AS Неделя
,DATEPART(DAYOFYEAR, GETDATE()) AS День_года
,DATEPART(WEEKDAY, GETDATE()) AS День_недели
```

Пример 16: Вывести дату 100 дней назад от текущей:

```
SELECT
DATEADD(DAY, -100, GETDATE()) AS День_100_Назад
```

Пример 17: Академик Игорь Евгеньевич Тамм родился 8 июля 1895 года. И.Е. Тамм скончался 12 апреля 1971 года. Вывести количество прожитых дней:

```
SELECT
DATEDIFF(DAY, '18950708', '19710412') AS Количество_прожитых_дней
```

Пример 18: Вывести ФИО и время года рождения каждого академика:

```
SELECT
ФИО
, CASE MONTH(Дата_рождения)  WHEN 3 THEN 'Весна'
WHEN 4 THEN 'Весна'
WHEN 5 THEN 'Весна'
WHEN 6 THEN 'Лето'
WHEN 7 THEN 'Лето'
WHEN 8 THEN 'Лето'
```

```
                    WHEN 9 THEN 'Осень'
                    WHEN 10 THEN 'Осень'
                    WHEN 11 THEN 'Осень'
                ELSE 'Зима'
                    END AS Времени_года
    FROM Академики
```

Пример 19: Вывести ФИО, дату рождения и знак зодиака каждого академика:

```
SELECT
            ФИО
                , Дата_рождения
            , CASE

WHEN (MONTH(Дата_рождения)=3  AND DAY(Дата_рождения) >= 21)
    OR (MONTH(Дата_рождения)=4  AND DAY(Дата_рождения) <= 20) THEN 'Овен'

WHEN (MONTH(Дата_рождения)=4  AND DAY(Дата_рождения) >= 21)
    OR (MONTH(Дата_рождения)=5  AND DAY(Дата_рождения) <= 21) THEN 'Телец'

WHEN (MONTH(Дата_рождения)=5  AND DAY(Дата_рождения) >= 22)
    OR (MONTH(Дата_рождения)=6  AND DAY(Дата_рождения) <= 21) THEN 'Близнецы'

WHEN (MONTH(Дата_рождения)=6  AND DAY(Дата_рождения) >= 22)
    OR (MONTH(Дата_рождения)=7  AND DAY(Дата_рождения) <= 22) THEN 'Рак'

WHEN (MONTH(Дата_рождения)=7  AND DAY(Дата_рождения) >= 23)
    OR (MONTH(Дата_рождения)=8  AND DAY(Дата_рождения) <= 21) THEN 'Лев'

WHEN (MONTH(Дата_рождения)=8  AND DAY(Дата_рождения) >= 22)
    OR (MONTH(Дата_рождения)=9  AND DAY(Дата_рождения) <= 23) THEN 'Дева'

WHEN (MONTH(Дата_рождения)=9  AND DAY(Дата_рождения) >= 24)
    OR (MONTH(Дата_рождения)=10 AND DAY(Дата_рождения) <= 23) THEN 'Весы'

WHEN (MONTH(Дата_рождения)=10 AND DAY(Дата_рождения) >= 24)
    OR (MONTH(Дата_рождения)=11 AND DAY(Дата_рождения) <= 22) THEN
'Скорпион'

WHEN (MONTH(Дата_рождения)=11 AND DAY(Дата_рождения) >= 23)
    OR (MONTH(Дата_рождения)=12 AND DAY(Дата_рождения) <= 22) THEN 'Стрелец'

WHEN (MONTH(Дата_рождения)=12 AND DAY(Дата_рождения) >= 23)
    OR (MONTH(Дата_рождения)=1  AND DAY(Дата_рождения) <= 20) THEN 'Козерог'

WHEN (MONTH(Дата_рождения)=1  AND DAY(Дата_рождения) >= 21)
    OR (MONTH(Дата_рождения)=2  AND DAY(Дата_рождения) <= 19) THEN 'Водолей'

WHEN (MONTH(Дата_рождения)=2  AND DAY(Дата_рождения) >= 20)
    OR (MONTH(Дата_рождения)=3  AND DAY(Дата_рождения) <= 20) THEN 'Рыбы'
                    END AS Знак_зодиака
    FROM Академики
```

Пример 20: Вывести список академиков. Для каждого академика, в зависимости от возраста, при присвоении звания вывести «молодой» или «старый» в дополнительном столбце:

```
SELECT
            ФИО
```

```
,Дата_рождения
,Специализация
,Год_присвоения_звания

,IIF(Год_присвоения_звания - Year(Дата_рождения) <= 45, 'Молодой','Старый')
AS Возраст_при_присвоении
FROM Академики
```

Задание

1. Вывести список академиков, отсортированный по количеству символов в ФИО.
2. Вывести список академиков, убрать лишние пробелы в ФИО.
3. Найти позиции «ов» в ФИО каждого академика. Вывести ФИО и номер позиции.
4. Вывести ФИО и последние две буквы специализации для каждого академика.
5. Вывести список академиков, ФИО в формате Фамилия и Инициалы.
6. Вывести список специализаций в правильном и обратном виде. Убрать дубликаты.
7. Вывести свою фамилию в одной строке столько раз, сколько вам лет.
8. Вывести абсолютное значение функций $sin^2\left(\frac{\pi}{2}\right) - cos\left(\frac{3\pi}{2}\right)$ с точностью два знака после десятичной запятой.
9. Вывести количество дней до конца семестра.
10. Вывести количество месяцев от вашего рождения.
11. Вывести ФИО и високосность года рождения каждого академика.
12. Вывести список специализаций без повторений. Для каждой специализации вывести «длинный» или «короткий», в зависимости от количества символов.

Выборка всех столбцов и всех строк одной таблицы Locations 1. (Рисунок 52).

Практическое занятие № 17 Создание и модификация таблиц БД. Выборка данных из БД. Модификация содержимого БД

Цель работы: овладение практическими навыками по модификации таблиц БД, создание запросов к БД

Создание простых запросов на выборку

Для выполнения запросов (извлечения строк из одной или нескольких таблиц БД) используется оператор *SELECT*. Результатом запроса всегда является таблица. Результаты запроса могут быть использованы для создания новой таблицы. Таблица, полученная в результате запроса, может стать предметом дальнейших запросов. Общая форма оператора *SELECT*:

SELECT столбцы FROM таблицы
[WHERE условия]
[GROUP BY группа [HAVING групповые_условия]]
[ORDER BY имя_поля]
[LIMIT пределы];

Оператор *SELECT* имеет много опций. Их можно использовать или не использовать, но они должны указываться в том порядке, в каком они приведены. Если требуется вывести все столбцы таблицы, необязательно перечислять их после ключевого слова *SELECT*, достаточно заменить этот список символом *.

```
mysql> SELECT * FROM orders;
+----------+-----------+-----------+---------------------+----------+
| order_ID | o_user_ID | o_book_ID | o_time              | o_number |
+----------+-----------+-----------+---------------------+----------+
|        1 |         3 |         8 | 2009-01-04 10:39:38 |        1 |
|        2 |         6 |        10 | 2009-02-10 09:40:29 |        2 |
|        3 |         1 |        20 | 2009-02-18 13:41:05 |        4 |
|        4 |         4 |        20 | 2009-03-10 18:20:00 |        1 |
|        5 |         3 |        20 | 2009-03-17 19:15:36 |        1 |
+----------+-----------+-----------+---------------------+----------+
5 rows in set (0.02 sec)
```

Список столбцов в операторе *SELECT* используют, если нужно изменить порядок следования столбцов в результирующей таблице или выбрать часть столбцов.

```
mysql> SELECT cat_name, cat_ID FROM catalogs;
+--------------------+--------+
| cat_name           | cat_ID |
+--------------------+--------+
| Программирование   |      1 |
| Интернет           |      2 |
| Базы данных        |      3 |
| Сети               |      4 |
| Мультимедиа        |      5 |
+--------------------+--------+
5 rows in set (0.01 sec)
```

Условия выборки. Гораздо чаще встречается ситуация, когда необходимо изменить количество выводимых строк. Для выбора записей, удовлетворяющих определенным критериям поиска, можно использовать конструкцию *WHERE*.

```
mysql> SELECT user_ID, u_surname FROM users
    -> WHERE u_status='active';
+---------+-----------+
| user_ID | u_surname |
+---------+-----------+
|       1 | Иванов    |
|       3 | Симонов   |
|       4 | Кузнецов  |
+---------+-----------+
3 rows in set (0.03 sec)
```

В запросе можно использовать ключевое слово *DISTINCT*, чтобы результат не содержал повторений уже имеющихся значений, например:

```
mysql> SELECT DISTINCT u_status FROM users;
+----------+
| u_status |
+----------+
| active   |
| passive  |
| lock     |
| gold     |
+----------+
4 rows in set (0.01 sec)
```

Сортировка. Результат выборки – записи, расположенные в том порядке, в котором они хранятся в БД. Чтобы отсортировать значения по одному из столбцов, необходимо после конструкции *ORDER BY* указать этот столбец, например:

```
mysql> SELECT * FROM orders ORDER BY o_user_ID;
+----------+-----------+-----------+---------------------+----------+
| order_ID | o_user_ID | o_book_ID | o_time              | o_number |
+----------+-----------+-----------+---------------------+----------+
|        3 |         1 |        20 | 2009-02-18 13:41:05 |        4 |
|        1 |         3 |         8 | 2009-01-04 10:39:38 |        1 |
|        5 |         3 |        20 | 2009-03-17 19:15:36 |        1 |
|        4 |         4 |        20 | 2009-03-10 18:20:00 |        1 |
|        2 |         6 |        10 | 2009-02-10 09:40:29 |        2 |
+----------+-----------+-----------+---------------------+----------+
5 rows in set (0.00 sec)
```

Сортировку записей можно производить по нескольким столбцам (их следует указать после слов *ORDER BY* через запятую). Число столбцов, указываемых в конструкции *ORDER BY*, не ограничено.

По умолчанию сортировка производится в прямом порядке (записи располагаются от наименьшего значения поля сортировки до наибольшего). Обратный порядок сортировки реализуется с помощью ключевого слова *DESC*:

```
mysql> SELECT o_time FROM orders ORDER BY o_time DESC;
+---------------------+
| o_time              |
+---------------------+
| 2009-03-17 19:15:36 |
| 2009-03-10 18:20:00 |
| 2009-02-18 13:41:05 |
| 2009-02-10 09:40:29 |
| 2009-01-04 10:39:38 |
+---------------------+
5 rows in set (0.27 sec)
```

Для прямой сортировки существует ключевое слово *ASC*, но так как записи

сортируются в прямом порядке по умолчанию, данное ключевое слово опускают.

Ограничение выборки. Результат выборки может содержать тысячи записей, вывод и обработка которых занимают значительное время. Поэтому информацию часто разбивают на страницы и предоставляют ее пользователю частями. Постраничная навигация используется при помощи ключевого слова *LIMIT*, за которым следует число выводимых записей. Следующий запрос извлекает первые 5 записей, при этом осуществляется обратная сортировка по полю *b_count*:

```
mysql> SELECT book_ID, b_count FROM books
    -> ORDER BY b_count DESC
    -> LIMIT 5;
+---------+---------+
| book_ID | b_count |
+---------+---------+
|      28 |      20 |
|      25 |      20 |
|      26 |      15 |
|      29 |      12 |
|       9 |      12 |
+---------+---------+
5 rows in set (0.03 sec)
```

Для извлечения следующих пяти записей используется ключевое слово *LIMIT* с двумя цифрами. Первая указывает позицию, начиная с которой необходимо вернуть результат, вторая цифра – число извлекаемых записей, например:

```
mysql> SELECT book_ID, b_count FROM books
    -> ORDER BY b_count DESC
    -> LIMIT 5,5;
+---------+---------+
| book_ID | b_count |
+---------+---------+
|       1 |      10 |
|      27 |      10 |
|      24 |       8 |
|      30 |       8 |
|      20 |       6 |
+---------+---------+
5 rows in set (0.00 sec)
```

При определении смещения нумерация строк начинается с нуля (поэтому в последнем примере для шестой строки указано смещение 5).

Группировка записей. Конструкция *GROUP BY* позволяет группировать извлекаемые строки. Она полезна в комбинации с функциями, применяемыми к группам строк. Эти функции (табл. 16.1) называются агрегатами (суммирующими функциями) и вычисляют одно значение для каждой группы, создаваемой конструкцией *GROUP BY*. Функции позволяют узнать число строк в группе, подсчитать среднее значение, получить сумму значений столбцов. Результирующее значение рассчитывается для значений, не равных *NULL* (исключение – функция *COUNT(*)*). Допустимо использование этих функций в запросах без группировки (вся выборка – одна группа).

Пример использования функции *COUNT( )*, которая возвращает число строк в таблице, значения указанного столбца для которых отличны от *NULL*:

```
mysql> SELECT COUNT(book_ID) FROM books;
+----------------+
| COUNT(book_ID) |
+----------------+
|             30 |
+----------------+
1 row in set (0.16 sec)
```

Таблица 16.1. Агрегатные функции

Обозначение	Описание
AVG ([DISTINCT] expr)	Возвращает среднее значение аргумента *expr*. В качестве аргумента обычно выступает имя столбца. Необязательное слово *DISTINCT* позволяет обрабатывать только уникальные значения столбца *expr*
COUNT ( )	Подсчитывает число записей и имеет несколько форм. Форма *COUNT (выражение)* возвращает число записей в таблице, поле *выражение* для которых не равно *NULL*. Форма *COUNT(*)* возвращает общее число строк в таблице независимо от того, принимает какое-либо поле значение *NULL* или нет. Форма *COUNT (DISTINCT выражение1, выражение2, ...)* позволяет использовать ключевое слово *DISTINCT*,

	которое позволяет подсчитать только уникальные значения столбца
MIN ([DISTINCT] expr)	Возвращает минимальное значение среди всех непустых значений выбранных строк в столбце *expr*. Необязательное слово *DISTINCT* позволяет обрабатывать только уникальные значения столбца *expr*
MAX ([DISTINCT] expr)	Возвращает максимальное значение среди всех непустых значений выбранных строк в столбце *expr*. Необязательное слово *DISTINCT* позволяет обрабатывать только уникальные значения столбца *expr*
STD (expr)	Возвращает стандартное среднеквадратичное отклонение в аргументе *expr*
STDDEV_SAMP (expr)	Возвращает выборочное среднеквадратичное отклонение в аргументе *expr*
SUM ([DISTINCT] expr)	Возвращает сумму величин в столбце *expr*. Необязательное слово *DISTINCT* позволяет обрабатывать только уникальные значения столбца *expr*

Использование ключевого слова *DISTINCT* с функцией *COUNT( )* позволяет вернуть число уникальных значений *b_cat_ID* в таблице *books*, например:

```
mysql> SELECT COUNT(DISTINCT b_cat_ID) FROM books;
+----------------------------+
| COUNT(DISTINCT b_cat_ID) |
+----------------------------+
|                          5 |
+----------------------------+
1 row in set (0.02 sec)
```

В *SELECT*-запросе столбцу можно назначить новое имя с помощью оператора *AS*. Например, результату функции *COUNT( )* присваивается псевдоним *total*:

```
mysql> SELECT COUNT(order_ID) AS total FROM orders;
+--------+
| total |
+--------+
|     5 |
+--------+
1 row in set (0.05 sec)
```

Использование функций в конструкции *WHERE* приведет к ошибке. В следующем примере показана попытка извлечения из таблицы *catalogs* записи с максимальным значением поля *cat_ID*:

```
mysql> SELECT * FROM catalogs WHERE cat_ID=MAX(cat_ID);
ERROR 1111 (HY000): Invalid use of group function
```

Решение задачи следует искать в использовании конструкции *ORDER BY*:

```
mysql> SELECT * FROM catalogs ORDER BY cat_ID DESC LIMIT 1;
+--------+--------------+
| cat_ID | cat_name     |
+--------+--------------+
|      5 | Мультимедиа |
+--------+--------------+
1 row in set (0.00 sec)
```

Для извлечения уникальных записей используют конструкцию *GROUP BY* с именем столбца, по которому группируется результат:

```
mysql> SELECT b_cat_ID FROM books
    -> GROUP BY b_cat_ID ORDER BY b_cat_ID;
+----------+
| b_cat_ID |
+----------+
|        1 |
|        2 |
|        3 |
|        4 |
|        5 |
+----------+
5 rows in set (0.03 sec)
```

При использовании *GROUP BY* возможно использование условия *WHERE*:

```
mysql> SELECT b_cat_ID, COUNT(b_cat_ID) FROM books
    -> WHERE b_cat_ID > 2
    -> GROUP BY b_cat_ID
    -> ORDER BY b_cat_ID;
+-----------+-----------------+
| b_cat_ID | COUNT(b_cat_ID) |
+-----------+-----------------+
|         3 |               4 |
|         4 |               5 |
|         5 |               6 |
+-----------+-----------------+
3 rows in set (0.02 sec)
```

Часто при задании условий требуется ограничить выборку по результату функции (например, выбрать каталоги, где число товарных позиций больше 5). Использование для этих целей конструкции *WHERE* приводит к ошибке. Для решения этой проблемы вместо ключевого слова *WHERE* используется ключевое слово *HAVING*, располагающееся за конструкцией *GROUP BY*:

```
mysql> SELECT b_cat_ID, COUNT(b_cat_ID) AS total FROM books
    -> GROUP BY b_cat_ID
    -> HAVING total > 5
    -> ORDER BY b_cat_ID;
+-----------+-------+
| b_cat_ID | total |
+-----------+-------+
|         1 |     9 |
|         2 |     6 |
|         5 |     6 |
+-----------+-------+
3 rows in set (0.00 sec)
```

Запрос, извлекающий уникальные значения столбца *b_cat_ID*, большие двух:

```
mysql> SELECT b_cat_ID, COUNT(b_cat_ID) FROM books
    -> GROUP BY b_cat_ID
    -> HAVING b_cat_ID > 2
    -> ORDER BY b_cat_ID;
+-----------+-----------------+
| b_cat_ID | COUNT(b_cat_ID) |
+-----------+-----------------+
|         3 |               4 |
|         4 |               5 |
|         5 |               6 |
+-----------+-----------------+
3 rows in set (0.00 sec)
```

При этом в случае использования ключевого слова *WHERE* сначала производится выборка из таблицы с применением условия и лишь затем группировка результата, а в случае использования ключевого слова *HAVING* сначала происходит группировка таблицы и лишь затем выборка с применением условия. Допускается использование условия *HAVING* без группировки
GROUP BY.

Использование функций. Для решения специфических задач при выборке удобны встроенные функции MySQL. Большинство функций предназначено для использования в выражениях *SELECT* и *WHERE*. Существуют также специальные функции группировки для использования в выражении *GROUP BY* (см. выше).

Каждая функция имеет уникальное имя и может иметь несколько аргументов (перечисляются через запятую в круглых скобках). Если аргументы отсутствуют, круглые скобки все равно следует указывать. Пробелы между именем функции и круглыми скобками недопустимы.

Число доступных для использования функций велико, в приложениях приведены наиболее полезные из них.

Пример использования функции, возвращающей версию сервера MySQL:

```
mysql> SELECT VERSION();
+-----------------------+
| VERSION()             |
+-----------------------+
| 5.0.51b-community-nt |
+-----------------------+
1 row in set (0.00 sec)
```

Отметим также возможность использования оператора *SELECT* без таблиц вообще. В такой форме *SELECT* можно использовать как калькулятор:

```
mysql> SELECT 2+3;
+-----+
| 2+3 |
+-----+
|   5 |
+-----+
1 row in set (0.00 sec)
```

Можно вычислить любое выражение без указания таблиц, получив доступ ко всему разнообразию математических и других операторов и функций. Возможность выполнять математические расчеты на уровне *SELECT* позволяет проводить финансовый анализ значений таблиц и отображать полученные результаты в отчетах. Во всех выражениях MySQL (как в любом языке программирования) можно использовать скобки, чтобы контролировать порядок вычислений.

Операторы. Под операторами подразумеваются конструкции языка, которые производят преобразование данных. Данные, над которыми совершается операция, называются операндами.

В MySQL используются три типа операторов:

- арифметические операторы;
- операторы сравнения;
- логические операторы.

Арифметические операции. В MySQL используются обычные арифметические операции:

сложение (+), вычитание (–), умножение (*), деление (/) и целочисленное деление *DIV* (деление и отсечение дробной части). Деление на 0 дает безопасный результат *NULL*.

Операторы сравнения. При работе с операторами сравнения необходимо помнить о том, что, за исключением нескольких особо оговариваемых случаев, сравнение чего-либо со значением *NULL* дает в результате *NULL*. Это касается и сравнения значения *NULL* со значением *NULL*:

```
mysql> SELECT NULL=NULL;
+-----------+
| NULL=NULL |
+-----------+
|      NULL |
+-----------+
1 row in set (0.02 sec)
```

Корректнее использовать следующий запрос:

```
mysql> SELECT NULL IS NULL;
+--------------+
| NULL IS NULL |
+--------------+
|            1 |
+--------------+
1 row in set (0.00 sec)
```

Поэтому следует быть предельно внимательными при работе с операторами сравнения, если операнды могут принимать значения *NULL*.

Наиболее часто используемые операторы сравнения приведены в табл. 16.2.

Логические операторы. MySQL поддерживает все обычные логические операции, которые можно использовать в выражениях. Логические выражения в MySQL могут принимать значения 1 (истина), 0 (ложь) или *NULL*.

Кроме того, следует учитывать, что MySQL интерпретирует любое ненулевое значение, отличное от *NULL*, как значение «истина». Основные логические операторы приведены в табл. 8.

Таблица 16.2. Наиболее часто используемые операторы

Оператор	Значение
=	Оператор равенства. Возвращает 1 (истина), если операнды равны, и 0 (ложь), если не равны
<=>	Оператор эквивалентности. Аналогичен обычному равенству, но возвращает только два значения: 1 (истина) и 0 (ложь). *NULL* не возвращает
<>	Оператор неравенства. Возвращает 1 (истина), если операнды не равны, и 0 (ложь), если равны
<	Оператор «меньше». Возвращает 1 (истина), если левый операнд меньше правого, и 0 (ложь) – в противном случае
<=	Оператор «меньше или равно». Возвращает 1 (истина), если левый операнд меньше правого или они равны, и 0 (ложь) – в противном случае

>	Оператор «больше». Возвращает 1 (истина), если левый операнд больше правого, и 0 (ложь) – в противном случае
>=	Оператор «больше или равно». Возвращает 1 (истина), если левый операнд больше правого или они равны, и 0 (ложь) – в противном случае
n BETWEEN min AND max	Проверка диапазона. Возвращает 1 (истина), если проверяемое значение *n* находится между *min* и *max*, и 0 (ложь) – в противном случае
IS NULL и *IS NOT NULL*	Позволяют проверить, является ли значение значением *NULL* или нет
n IN (множество)	Принадлежность к множеству. Возвращает 1 (истина), если проверяемое значение *n* входит в список, и 0 (ложь) – в противном случае. В качестве множества может использоваться список литеральных значений или выражений или подзапрос

Переменные SQL и временные таблицы. Часто результаты запроса необходимо использовать в последующих запросах. Для этого полученные данные необходимо сохранить во временных структурах. Эту задачу решают переменные SQL и временные таблицы. Объявление переменной начинается с символа @, за которым следует имя переменной. Значения переменным присваиваются посредством оператора *SELECT* С использованием оператора присваивания := . Например:

```
mysql> SELECT @total := COUNT(*) FROM books;
+----------------------+
| @total := COUNT(*) |
+----------------------+
|                   30 |
+----------------------+
1 row in set (0.42 sec)

mysql> SELECT @total;
+---------+
| @total |
+---------+
| 30     |
+---------+
1 row in set (0.02 sec)
```

Объявляется переменная *@total*, которой присваивается число записей в таблице *books*. Затем в рамках текущего сеанса в последующих запросах появляется возможность использования данной переменной. Переменная действует только в рамках одного сеанса соединения с сервером MySQL и прекращает свое существование после разрыва соединения.

Переменные также могут объявляться при помощи оператора *SET*:

```
mysql> SET @last=CURDATE()-INTERVAL 7 DAY;
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT CURDATE(), @last;
+------------+------------+
| CURDATE()  | @last      |
+------------+------------+
| 2009-12-22 | 2009-12-15 |
+------------+------------+
1 row in set (0.00 sec)
```

При использовании оператора *SET* в качестве оператора присваивания может выступать обычный знак равенства =. Оператор *SET* удобен тем, что он не возвращает результирующую таблицу. Не рекомендуется одновременно присваивать переменной некоторое значение и использовать эту переменную в одном запросе.

Переменная SQL позволяет сохранить одно промежуточное значение. Когда необходимо сохранить результирующую таблицу, прибегают к временным таблицам. Создание временных таблиц осуществляется при помощи оператора *CREATE TEMPORARY TABLE,* синтаксис которого ничем не отличается от синтаксиса оператора *CREATE TABLE*.

Временная таблица автоматически удаляется по завершении соединения с сервером, а ее имя действительно только в течение данного соединения. Это означает, что два разных клиента могут использовать временные таблицы с одинаковыми именами без конфликта

друг с другом или с существующей таблицей с тем же именем.

Задание для практической работы

1. Создайте простой запрос на выборку к таблице *books*, который выводит максимальную и минимальную цены товарных позиций, присваивая им соответственно псевдонимы *maximum* и *minimum*:

```
mysql> SELECT MAX(b_price) AS maximum, MIN(b_price) AS minimum
    -> FROM books;
+---------+---------+
| maximum | minimum |
+---------+---------+
|  771.00 |   42.00 |
+---------+---------+
1 row in set (0.00 sec)
```

2. Создавайте простой запрос на выборку к таблице *books*, который выводит количество записей, соответствующих каждому из уникальных значений *b_cat_ID.* Для этого используем функцию *COUNT( )* вместе с выражением *GROUP BY*:

```
mysql> SELECT b_cat_ID, COUNT(b_cat_ID) FROM books
    -> GROUP BY b_cat_ID ORDER BY b_cat_ID;
+----------+----------------+
| b_cat_ID | COUNT(b_cat_ID) |
+----------+----------------+
|        1 |              9 |
|        2 |              6 |
|        3 |              4 |
|        4 |              5 |
|        5 |              6 |
+----------+----------------+
5 rows in set (0.00 sec)
```

3. Создавайте многотабличный запрос на выборку, который выводит фамилии, имена и отчества покупателей магазина, сделавших менее двух покупок:

```
mysql> SELECT users.u_surname,users.u_name,users.u_patronymic,
    -> COUNT(orders.order_ID) AS total
    -> FROM users LEFT JOIN orders ON users.user_ID=orders.o_user_ID
    -> GROUP BY users.user_ID
    -> HAVING total<2
    -> ORDER BY total DESC;
+-----------+----------+--------------+-------+
| u_surname | u_name   | u_patronymic | total |
+-----------+----------+--------------+-------+
| Иванов    | Александр | Валерьевич  |     1 |
| Корнеев   | Александр | Александрович |   1 |
| Кузнецов  | Максим   | Петрович     |     1 |
| Петров    | Анатолий | Юрьевич      |     0 |
| Лосев     | Сергей   | Иванович     |     0 |
+-----------+----------+--------------+-------+
5 rows in set (0.02 sec)
```

4. Создайте запрос на выборку с вложенным запросом, выводящим перечень книг, которые не заказывались покупателями:

```
mysql> SELECT book_ID, b_name, b_price FROM books
    -> WHERE NOT EXISTS (SELECT * FROM orders
    -> WHERE orders.o_book_ID=books.book_ID);
+---------+-----------------------------------------------+---------+
| book_ID | b_name                                        | b_price |
+---------+-----------------------------------------------+---------+
|       1 | JavaScript в кармане                          |   42.00 |
|       2 | Visual FoxPro 9.0                             |  660.00 |
|       3 | C++ Как он есть                               |  218.00 |
|       4 | Создание приложений с помощью C#              |  169.00 |
|       5 | Delphi. Народные советы                       |  243.00 |
|       6 | Delphi. Полное руководство                    |  500.00 |
|       7 | Профессиональное программирование на PHP      |  309.00 |
|       9 | Практика программирования                     |  214.00 |
|      11 | Поиск в Internet                              |  107.00 |
|      12 | Web-конструирование                           |  177.00 |
|      13 | Самоучитель Интернет                          |  121.00 |
|      14 | Популярные интернет-браузеры                  |   82.00 |
|      15 | Общение в Интернете                           |   85.00 |
|      16 | Базы данных                                   |  326.00 |
|      17 | Базы данных. Разработка приложений            |  189.00 |
|      18 | Раскрытие тайн SQL                            |  200.00 |
|      19 | Практикум по Access                           |   87.00 |
|      21 | Сети. Поиск неисправностей                    |  434.00 |
|      22 | Безопасность сетей                            |  462.00 |
|      23 | Анализ и диагностика компьютерных сетей       |  344.00 |
|      24 | Локальные вычислительные сети                 |   82.00 |
|      25 | Цифровая фотография                           |  149.00 |
|      26 | Музыкальный компьютер для гитариста           |  217.00 |
|      27 | Видео на ПК                                   |  231.00 |
|      28 | Мультипликация во Flash                       |  211.00 |
|      29 | Запись CD и DVD                               |  167.00 |
|      30 | Запись и обработка звука на компьютере        |   51.00 |
+---------+-----------------------------------------------+---------+
27 rows in set (0.03 sec)
```

Практическое занятие № 18 Обработка транзакций. Использование функций защиты для БД

Цель работы: получение практических навыков обработки транзакций и защиты баз

данных при помощи функций.

Обработка транзакций

Изменения БД часто требуют выполнения нескольких запросов, например при покупке в электронном магазине требуется добавить запись в таблицу заказов и уменьшить число товарных позиций на складе. В промышленных БД одно событие может затрагивать большее число таблиц и требовать многочисленных запросов.

Если на этапе выполнения одного из запросов происходит сбой, это может нарушить целостность БД (товар может быть продан, а число товарных позиций на складе не обновлено). Чтобы сохранить целостность БД, все изменения должны выполняться как единое целое. Либо все изменения успешно выполняются, либо, в случае сбоя, БД принимает состояние, которое было до начала изменений. Это обеспечивается средствами обработки транзакций.

Транзакция – последовательность операторов SQL, выполняющихся как единая операция, которая не прерывается другими клиентами. Пока происходит работа с записями таблицы (обновление или удаление), никто другой не может получить доступ к этим записям, т. к. MySQL автоматически блокирует доступ к ним.

Таблицы *ISAM, MyISAM* и *HEAP* не поддерживают транзакции. В настоящий момент их поддержка осуществляется только в таблицах *BDB* и *InnoDB*.

Транзакции позволяют объединять операторы в группу и гарантировать, что все операторы группы будут выполнены успешно. Если часть транзакции выполняется со сбоем, результаты выполнения всех операторов транзакции до места сбоя отменяются, приводя БД к виду, в котором она была до выполнения транзакции.

По умолчанию MySQL работает в режиме автоматического завершения транзакций, т. е. как только выполняется оператор обновления данных, который модифицирует таблицу, изменения тут же сохраняются на диске. Чтобы объединить операторы в транзакцию, следует отключить этот режим: *SET AUTOCOMMIT=0;*

После отключения режима для завершения транзакции необходимо ввести оператор *COMMIT*, для отката – *ROLLBACK*.

Включить режим автоматического завершения транзакций для отдельной последовательности операторов можно оператором *START TRANSACTION*.

Для таблиц *InnoDB* есть операторы *SAVEPOINT* и *ROLLBACK TO SAVEPOINT*, которые позволяют работать с именованными точками начала транзакции.

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Периферия');
Query OK, 1 row affected (0.00 sec)

mysql> SAVEPOINT point1;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Разное');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM catalogs;
+--------+-------------------+
| cat_ID | cat_name          |
+--------+-------------------+
|      1 | Программирование  |
|      2 | Интернет          |
|      3 | Базы данных       |
|      4 | Сети              |
|      5 | Мультимедиа       |
|     12 | Периферия         |
|     13 | Разное            |
+--------+-------------------+
7 rows in set (0.00 sec)

mysql> ROLLBACK TO SAVEPOINT point1;
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> SELECT * FROM catalogs;
+--------+--------------------+
| cat_ID | cat_name           |
+--------+--------------------+
|      1 | Программирование   |
|      2 | Интернет           |
|      3 | Базы данных        |
|      4 | Сети               |
|      5 | Мультимедиа        |
|     12 | Периферия          |
+--------+--------------------+
6 rows in set (0.00 sec)
```

Оператор *SAVEPOINT* устанавливает именованную точку начала транзакции с именем *point1*. Оператор *ROLLBACK TO SAVEPOINT point1* откатывает транзакцию к состоянию, в котором находилась БД на момент установки именованной точки. Все точки сохранения транзакций удаляются, если выполняются операторы *COMMIT* или *ROLLBACK* без указания имени точки сохранения.

Управление правами пользователей

СУБД MySQL является многопользовательской средой, поэтому для доступа к таблицам БД могут быть созданы различные учетные записи с разным уровнем привилегий. Учетной записи редактора можно предоставить привилегии на просмотр таблицы, добавление новых записей и обновление уже существующих. Администратору БД можно предоставить более широкие полномочия (возможность создания таблиц, редактирования и удаления уже существующих). Для пользователя БД достаточно лишь просмотра таблиц.

Рассмотрим следующие вопросы:

- создание, редактирование и удаление учетных записей пользователей;
- назначение и отмена привилегий.

Учетная запись является составной и принимает форму *'username' @ 'host'*, где *username* – имя пользователя, а *host* – наименование хоста, с которого пользователь может обращаться к серверу. Например, записи *'root' @ '127.0.0.1'* и *'wet' @ '62.78.56.34'* означают, что пользователь с именем *root* может обращаться с хоста, на котором расположен сервер, а *wet* – только с хоста с IPадресом 62.78.56.34.

IP-адрес 127.0.0.1 всегда относится к локальному хосту. Если сервер и клиент установлены на одном хосте, то сервер слушает соединения по этому адресу, а клиент отправляет на него SQLзапросы.

IP-адрес 127.0.0.1 имеет псевдоним *localhost*, поэтому учетные записи вида *'root' @ '127.0.0.1'* можно записывать в виде *'root' @ 'localhost'*.

Число адресов, с которых необходимо обеспечить доступ пользователю, может быть значительным. Для задания диапазона в имени хоста используется специальный символ "%". Так, учетная запись *'wet' @ '%'* позволяет пользователю *wet* обращаться к серверу MySQL с любых компьютеров сети.

Все учетные записи хранятся в таблице *user* системной базы данных с именем *mysql*. После первой инсталляции содержимое таблицы *user* выглядит так, как показано в листинге.

```
mysql> SELECT Host,User,Password FROM mysql.user;
+-----------------------+------+----------+
| Host                  | User | Password |
+-----------------------+------+----------+
| localhost             | root |          |
| production.mysql.com  | root |          |
| 127.0.0.1             | root |          |
| localhost             |      |          |
| production.mysql.com  |      |          |
+-----------------------+------+----------+
5 rows in set (0.27 sec)
```

Создание новой учетной записи. Создать учетную запись позволяет оператор

CREATE USER 'username' @ 'host' [IDENTIFIED BY [PASSWORD] 'пароль'];

Оператор создает новую учетную запись с необязательным паролем. Если пароль не указан, в его качестве выступает пустая строка. Разумно хранить пароль в виде хэш-кода, полученного в результате необратимого шифрования. Чтобы воспользоваться этим механизмом авторизации, необходимо поместить между ключевым словом *IDENTIFIED BY*

и паролем ключевое слово *PASSWORD*.

Удаление учетной записи. Удалить учетную запись позволяет оператор

DROP USER 'username' @ 'host';

Изменение имени пользователя в учетной записи. Осуществляется с помощью оператора

RENAME USER старое_имя TO новое_имя;

Назначение привилегий. Рассмотренные выше операторы позволяют создавать, удалять и редактировать учетные записи, но они не позволяют изменять привилегии пользователя – сообщать MySQL, какой пользователь имеет право только на чтение информации, какой на чтение и редактирование, а кому предоставлены права изменять структуру БД и создавать учетные записи.

Для решения этих задач предназначены операторы *GRANT* (назначает привилегии) и *REVOKE* (удаляет привилегии). Если учетной записи, которая показана в операторе *GRANT*, не существует, то она автоматически создается. Удаление всех привилегий с помощью оператора *REVOKE* не приводит к автоматическому уничтожению учетной записи. Для удаления пользователя необходимо воспользоваться оператором *DROP USER*.

В простейшем случае оператор *GRANT* выглядит следующим образом:

```
mysql> GRANT ALL ON *.* TO 'wet'@'localhost' IDENTIFIED BY 'pass';
Query OK, 0 rows affected (0.17 sec)
```

Данный запрос создает пользователя с именем *wet* и паролем *pass*, который может обращаться к серверу с локального хоста (*localhost*) и имеет все права (*ALL*) для всех баз данных (*.*). Если такой пользователь существует, то его привилегии будут изменены на *ALL*.

Вместо ключевого слова *ALL* можно использовать любое из ключевых слов, представленных в табл. 17.1. Ключевое слово *ON* в операторе *GRANT* задает уровень привилегий, которые могут быть заданы на одном из четырех уровней, представленных в табл. 10. Для таблиц можно установить только следующие типы привилегий: *SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, GRANT OPTION, INDEX* и *ALTER*. Это следует учитывать при использовании конструкции *GRANT ALL*, которая назначает привилегии на текущем уровне. Так, запрос уровня базы данных *GRANT ALL ON db.** не предоставляет никаких глобальных привилегий.

Отмена привилегий. Для отмены привилегий используется оператор *REVOKE*:

```
mysql> REVOKE DELETE, UPDATE ON *.* FROM 'wet'@'localhost';
Query OK, 0 rows affected (0.02 sec)
```

Оператор *REVOKE* отменяет привилегии, но не удаляет учетные записи, для их удаления необходимо воспользоваться оператором *DROP USER*.

Таблица 17.1. Вместо ключевого слова ALL

Привилегия	**Операция, разрешенная привилегией**
ALL [PRIVILEGES]	Комбинация всех привилегий, за исключением привилегии *GRANT OPTION*, которая задается отдельно
ALTER	Позволяет редактировать таблицу с помощью оператора *ALTER TABLE*
ALTER ROUTINE	Позволяет редактировать или удалять хранимую процедуру
CREATE	Позволяет создавать таблицу при помощи оператора *CREATE TABLE*
CREATE ROUTINE	Позволяет создавать хранимую процедуру
CREATE TEMPORARY TABLES	Позволяет создавать временные таблицы
CREATE USER	Позволяет работать с учетными записями с помощью *CREATE*

	USER, *DROP USER, RENAME USER* и *REVOKE ALL PRIVILEGES*
CREATE VIEW	Позволяет создавать представление с помощью оператора *CREATE* *VIEW*
DELETE	Позволяет удалять записи при помощи оператора *DELETE*
DROP	Позволяет удалять таблицы при помощи оператора *DROP TABLE*
EXECUTE	Позволяет выполнять хранимые процедуры
INDEX	Позволяет работать с индексами, в частности, использовать операторы *CREATE INDEX* и *DROP INDEX*
INSERT	Позволяет добавлять в таблицу новые записи оператором *INSERT*
LOCK TABLES	Позволяет осуществлять блокировки таблиц при помощи операторов *LOCK TABLES* и *UNLOCK TABLES*. Для вступления в действие этой привилегии должна быть установлена привилегия *SELECT*
SELECT	Позволяет осуществлять выборки таблиц оператором *SELECT*
SHOW DATABASES	Позволяет просматривать список всех таблиц на сервере при помощи оператора *SHOW DATABASES*
SHOW VIEW	Позволяет использовать оператор *SHOW CREATE VIEW*
UPDATE	Позволяет обновлять содержимое таблиц оператором *UPDATE*
USAGE	Синоним для статуса «отсутствуют привилегии»
GRANT OPTION	Позволяет управлять привилегиями других пользователей, без данной привилегии невозможно выполнить операторы *GRANT* И *REVOKE*

Таблица 17.2. Вместо ключевого слова ON

Ключевое слово *ON*	**Уровень**
*ON *.**	Глобальный уровень – пользователь с полномочиями на глобальном уровне может обращаться ко всем БД и таблицам, входящим в их состав
*ON db.**	Уровень базы данных – привилегии распространяются на таблицы базы данных *db*
ON db.tbl	Уровень таблицы – привилегии распространяются на таблицу *tbl* базы данных *db*
ON db.tbl	Уровень столбца – привилегии касаются отдельных столбцов в таблице *tbl* базы данных *db*. Список столбцов указывается в скобках через запятую после ключевых слов *SELECT, INSERT, UPDATE*

Задание для практической работы

При выполнении практической работы необходимо:

- создать транзакцию, произвести ее откат и фиксацию;
- составить отчет по практической работе.

Задание 1. Обработка транзакций

Для выполнения задания объединим несколько операций по добавлению в таблицу *catalogs* новых каталогов, а затем произведем откат транзакции, т. е. отмену произведенных действий. Отключаем режим автоматического завершения, добавляем новые записи и проверяем, добавились записи или нет.

```
mysql> SET AUTOCOMMIT=0;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Аппаратура');
Query OK, 1 row affected (0.06 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Безопасность');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM catalogs;
+--------+-------------------+
| cat_ID | cat_name          |
+--------+-------------------+
|      1 | Программирование  |
|      2 | Интернет          |
|      3 | Базы данных       |
|      4 | Сети              |
|      5 | Мультимедиа       |
|      6 | Аппаратура        |
|      7 | Безопасность      |
+--------+-------------------+
7 rows in set (0.02 sec)
```

Откатываем транзакцию оператором *ROLLBACK* (изменения не сохранились).

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT * FROM catalogs;
+--------+-------------------+
| cat_ID | cat_name          |
+--------+-------------------+
|      1 | Программирование  |
|      2 | Интернет          |
|      3 | Базы данных       |
|      4 | Сети              |
|      5 | Мультимедиа       |
+--------+-------------------+
5 rows in set (0.00 sec)
```

Воспроизведем транзакцию и сохраним действия оператором *COMMIT*.

```
mysql> INSERT INTO catalogs VALUES(NULL,'Аппаратура');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Безопасность');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM catalogs;
+--------+-------------------+
| cat_ID | cat_name          |
+--------+-------------------+
|      1 | Программирование  |
|      2 | Интернет          |
|      3 | Базы данных       |
|      4 | Сети              |
|      5 | Мультимедиа       |
|      8 | Аппаратура        |
|      9 | Безопасность      |
+--------+-------------------+
7 rows in set (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM catalogs;
+--------+-------------------+
| cat_ID | cat_name          |
+--------+-------------------+
|      1 | Программирование  |
|      2 | Интернет          |
|      3 | Базы данных       |
|      4 | Сети              |
|      5 | Мультимедиа       |
|      8 | Аппаратура        |
|      9 | Безопасность      |
+--------+-------------------+
7 rows in set (0.00 sec)
```

Задание 2. Защита данных

Для работы выберем два компьютера, подключенных к локальной сети. На одном необходимо развернуть сервер MySQL, на другой – скопировать клиент командной строки *mysql.exe*. Определим IP-адрес сервера:

```
C:\Documents and Settings\admin>ipconfig

Настройка протокола IP для Windows


Подключение по локальной сети - Ethernet адаптер:

        DNS-суффикс этого подключения . . :
        IP-адрес  . . . . . . . . . . . . : 192.168.67.6
        Маска подсети . . . . . . . . . . : 255.255.255.0
        Основной шлюз . . . . . . . . . . : 192.168.67.254
```

Создадим новую учетную запись, позволив пользователю *user1* обращаться к серверу MySQL с любых компьютеров сети:

```
mysql> CREATE USER 'user1'@'%' IDENTIFIED BY '123';
Query OK, 0 rows affected (0.01 sec)
```

Назначим этому пользователю привилегии глобального уровня:

```
mysql> GRANT ALL ON *.* TO 'user1'@'%' IDENTIFIED BY '123';
Query OK, 0 rows affected (0.00 sec)
```

На клиентском компьютере в командной строке (например, с помощью FAR), запустим клиент командной строки в следующем формате:

```
The FAR manager, version 1.70 beta 5 (build 1634)
Copyright (C) 1996-2000 Eugene Roshal, Copyright (C) 2000-2003 FAR Group
Зарегистрирован: xUSSR регистрация
C:\>mysql -u user1 -p123 -h 192.168.67.6
```

Наблюдаем отклик удаленного сервера и работаем с ним как обычно:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.0.51b-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| book               |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.00 sec)
```

Критерии оценки выполнения практических заданий:

«5» – все задания выполнены правильно;
«4» – наблюдались неточности при выполнении работы;
«3» – наблюдались ошибки при выполнении работы;
«2» – работа выполнена менее 50 %.

Преподаватель _____ И.А. Потапов

(подпись)

3. СПЕЦИФИКАЦИИ И ВАРИАНТЫ ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

3.1 СПЕЦИФИКАЦИИ И ВАРИАНТЫ ОЦЕНОЧНЫХ СРЕДСТВ

3.1.1 Назначение

Спецификацией устанавливаются требования к содержанию и оформлению вариантов оценочного средства Экзамен.

Экзамен предназначен для промежуточной аттестации и оценки знаний и умений студентов по программе учебной дисциплины «Основы проектирования баз данных» основной профессиональной образовательной программы 09.02.07 Информационные системы и программирование

3.1.2. Контингент аттестуемых: студенты 1 курса

3.1.3. Форма и условия аттестации:

Аттестация проводится в форме экзамена по завершению освоения учебного материала учебной дисциплины и при положительных результатах текущего контроля.

Итоговый контроль проходит в виде письменного выполнения заданий экзаменационного билета и устного собеседования

Экзаменационный билет состоит из двух частей:

1. Теоретическая часть, которая включает вопросы разных видов из разных тем (1-2 вопроса), взятых из фонда вопросов к экзамену для промежуточного контроля.

2. Практическая часть. Практическая часть экзаменационного билета состоит из задачи, взятой из фонда типовых расчетных задач.

3.1.4. Время выполнения:

выполнение 30 минут;

собеседование 15 минут;

всего 45 минут.

3.1.5. Рекомендуемая литература для разработки оценочных средств и подготовки, обучающихся к аттестации.

Библиографическое описание издания (автор, заглавие, вид, место и год издания, кол. стр.)	**Основная/ дополнительная литература**	**Книгообеспеченность**	
		Кол-во. экз. в библ.	**Электронные ресурсы**
Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для среднего профессионального образования / В. М. Илюшечкин. — испр. и доп. — Москва : Издательство Юрайт, 2023. — 213 с.	Основная	-	https://urait.ru/bcode/471698
Стружкин, Н. П. Базы данных: проектирование : учебник для среднего профессионального образования / Н. П. Стружкин, В. В. Годин. — Москва : Издательство Юрайт, 2023. — 477 с.	Основная	-	https://urait.ru/bcode/518499
Нестеров, С. А. Базы данных : учебник и практикум для среднего профессионального образования / С. А. Нестеров. — Москва : Издательство Юрайт, 2023. — 230 с.	Основная	-	https://urait.ru/bcode/518507
Советов, Б. Я. Базы данных : учебник для среднего профессионального образования / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. — 3-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2023. — 420 с.	Основная	-	https://urait.ru/bcode/514585
Федорова Г.Н. Основы проектирования баз данных. – Москва: Академия, 2021. – 224 с.	Основная	-	https://urait.ru/bcode/513827
Стружкин, Н. П. Базы данных: проектирование. Практикум : учебное пособие для среднего профессионального образования / Н. П. Стружкин, В. В. Годин. — Москва : Издательство Юрайт, 2023. — 291 с.	Дополнительная	-	https://urait.ru/bcode/516929

Таблица 2 – Перечень современных профессиональных баз данных (СПБД)

№	Наименование СПБД
1	Научная электронная библиотека eLIBRARRY - www.elibrary.ru
2	Научная электронная библиотека КиберЛениНка - www.cyberleninka.ru

Таблица 3 – Перечень информационных справочных систем (ИСС)

№	Наименование ИСС
1	Справочная правовая система КонсультантПлюс www.consultant.ru
2	Электронная библиотечная система ЭБС ЮРАИТ - www.urait.ru

3.1.6. Перечень материалов, оборудования и информационных источников.

Кабинет № 31информатики (для проведения занятий лекционного типа и занятий семинарского типа, курсового проектирования (выполнения курсовых работ) групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации) оборудован мультимедийным комплексом. Специализированная мебель: Учебная мебель на 39 посадочных места (столов трехместных 13 шт., скамеек 13 шт.), рабочее место преподавателя (стол 1 шт., стул 1 шт.), кафедра 1 шт. доска меловая 3х секционная 1шт. Компьютер Intel Pentium Dual CPU E2160 1,8 GHz ОЗУ- 2 Gb, HDD-500Gb, DVD RV-ROM, Клавиатура, Мышь. ОС windows 7 Максимальная. Локальный сеть с выходом в Интернет. Видеопроектор потолочный Epson EB-S82, проекционный экран Clasic Solition 266x149, акустические колонки Genius.

Наборы демонстрационного оборудования и учебно-наглядных пособий: мультимедийные приложения к лекционным курсам и практическим занятиям, интерактивные учебно-наглядные пособия.

Лаборатория № 2.2 программирования и баз данных. Компьютерный класс (для проведения практических занятий, с применением вычислительной техники) оборудован мультимедийным комплексом. Специализированная мебель и оборудование:

Учебная мебель на 16 посадочных мест, рабочее место преподавателя (стол – 1 шт., стул – 1 шт.). Компьютер Intel i5 7400/1Tb/8Gb/Philips 243V5Q 23' – 16 шт. Компьютер Intel i3 -2100 2.4 Ghz/4/500Gb/Acer V193 19» – 1 шт. Мультимедийный проектор Тип 1 Optoma x 400 – 1 шт. Консультант + (Договор поставки и сопровождения экземпляров системы № 124 от 28.08.2020), 7-Zip (freeware), Acrobat Reader DC (freeware), Adobe Acrobat Reader DC (freeware), FireFox 77.0.1 (freeware), Google Chrome 83.0.4103.97 (freeware), VLC media player (freeware), K-Lite Codec Pack Full (freeware). Программное обеспечение общего и профессионального назначения бесплатное (с открытой лицензией): EclipseIDEforJavaEEDevelopers, .NETFrameworkJDK 8, MicrosoftSQLServerExpressEdition, RAD Studio, NetBeans, ARIS Inkscape, MySQLInstallerforWindows, SQLServerManagementStudio, MicrosoftSQLServerJavaConnector, AndroidStudio, IntelliJIDEA.

Наборы демонстрационного оборудования и учебно-наглядных пособий: мультимедийные приложения к лекционным курсам и практическим занятиям, интерактивные учебнонаглядные пособия.

Вопросы к дифференцированному зачету
по дисциплине
Основы проектирования БД

1. База данных, определение, основные понятия. Типы организации базы данных.
2. Модели данных: понятие, основные компоненты и классификация
3. Типы и виды запросов пользователей.
4. СУБД: основные функции, типы. Свойства и сравнительные характеристики СУБД.
5. Обеспечение функционирования БД. Транзакции: понятия, модели завершения, свойства. Управление транзакциями
6. Модели «клиент-сервер» в технологии БД. Схема, основные функции клиента, понятие сервера и клиента.

7. Физическое проектирование. Особенности, влияющие на организацию внешней памяти. Технологии хранения данных.

8. Модели данных: понятие, основные компоненты и классификация

9. Объектно-реляционная модель данных. Объектно-ориентированная модель данных.

10. Основы реляционной алгебры. Операции над отношениями.

11. Трехуровневая архитектура описания базы данных. Режимы работы с базой данных.

12. Типы и виды запросов пользователей.

13. Отличие SQL от процедурных языков программирования. Интерактивный и встроенный SQL. Составные части SQL.

14. Нормализация отношений. Преобразование ER-модели в схему реляционной базы данных.

15. Создание запросов, отчетов.

16. Охарактеризуйте таблицу как основной компонент базы данных.

17. Сетевая модель данных. Достоинства и недостатки

18. Защита данных. Управление доступом к данным.

19. История развития баз данных

20. Нормализация отношений. Преобразование ER-модели в схему реляционной базы данных

21. Обеспечение функционирования БД. Транзакции: понятия, модели завершения, свойства. Управление транзакциями

22. Модели «клиент-сервер» в технологии БД. Схема, основные функции клиента, понятие сервера и клиента.

23. База данных, определение, основные понятия. Типы организации базы данных.

24. Модели данных: понятие, основные компоненты и классификация

25. Модели данных. Классификация моделей данных.

26. Языки баз данных. Работа с базами данных

27. Трехуровневая архитектура описания базы данных. Режимы работы с базой данных.

28. Типы и виды запросов пользователей.

29. Язык SQL. Функции и достоинства языка

30. Охарактеризуйте таблицу как основной компонент базы данных

31. Классификация моделей данных. Иерархическая модель данных

32. Развитие систем обработки данных

33. Функции СУБД.

34. Логическое проектирование базы данных.

35. Функциональные зависимости и ключи.

36. Архитектура клиент-сервер.

37. База данных, определение, основные понятия. Типы организации базы данных.

38. Модели данных: понятие, основные компоненты и классификация

39. Классификация моделей данных. Иерархическая модель данных

40. Развитие систем обработки данных

41. Функции СУБД.

42. Логическое проектирование базы данных.

43. Физическое проектирование. Особенности, влияющие на организацию внешней памяти. Технологии хранения данных.

44. Модели данных: понятие, основные компоненты и классификация

45. Функциональные зависимости и ключи.

46. Архитектура клиент-сервер.

47. Обеспечение функционирования БД. Транзакции: понятия, модели завершения, свойства. Управление транзакциями

48. Модели «клиент-сервер» в технологии БД. Схема, основные функции клиента, понятие сервера и клиента.

49. Язык SQL. Функции и достоинства языка

50. Охарактеризуйте таблицу как основной компонент базы данных

Критерии оценки:

Оценка экзамена	**Требования к знаниям** *(дописать оценку в соответствии с компетенциями, привязать к дисциплине)*
«отлично»	Оценка «отлично» выставляется студенту, если он глубоко и прочно усвоил программный материал, исчерпывающе, последовательно, четко и логически стройно его излагает, умеет тесно увязывать теорию с практикой, свободно справляется с задачами, вопросами и другими видами применения знаний, причем не затрудняется с ответом при видоизменении заданий, использует в ответе материал монографической литературы, правильно обосновывает принятое решение, владеет разносторонними навыками и приемами выполнения практических задач.
«хорошо»	Оценка «хорошо» выставляется студенту, если он твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопрос, правильно применяет теоретические положения при решении практических вопросов и задач, владеет необходимыми навыками и приемами их выполнения.
«удовлетворительно»	Оценка «удовлетворительно» выставляется студенту, если он имеет знания только основного материала, но не усвоил его деталей, допускает неточности, недостаточно правильные формулировки, нарушения логической последовательности в изложении программного материала, испытывает затруднения при выполнении практических работ.
«неудовлетворительно»	Оценка «неудовлетворительно» выставляется студенту, который не знает значительной части программного материала, допускает существенные ошибки, неуверенно, с большими затруднениями выполняет практические работы. Как правило, оценка «неудовлетворительно» ставится студентам, которые не могут продолжить обучение без дополнительных занятий по соответствующей дисциплине.