

Государственное бюджетное профессиональное образовательное учреждение
«Кунгурский автотранспортный колледж»

**МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ
ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ**

по МДК 11.01

Разработка, администрирование и защита баз данных

по специальности

09.02.04 Информационные системы и программирование

Одобрено на заседании
цикловой комиссии
информационно-
математических дисциплин.
Протокол № ____ от «__» _____ 20__ г.
Председатель комиссии
_____ Е.А.Наговицына

УТВЕРЖДАЮ
Зам. директора
_____ М.Г.Целищева
«__» _____ 20__ г.

Организация-разработчик: ГБПОУ КАТК

Составитель: Е.А. Наговицына

СОДЕРЖАНИЕ

- Практическая работа №1-3. Сбор и анализ информации
- Практическая работа №4-6. Приведение БД к нормальной форме 3НФ
- Практическая работа №7-10. Проектирование реляционной схемы базы данных в среде СУБД
- Практическая работа №11-13. Создание базы данных в среде разработки
- Практическая работа №14-15. Организация локальной сети. Настройка локальной сети
- Практическая работа №16-17. Установка и настройка SQL-сервера
- Практическая работа №18. Экспорт данных базы в документы пользователя
- Практическая работа №19. Импорт данных пользователя в базу данных
- Практическая работа №20-21. Выполнение настроек для автоматизации обслуживания базы данных
- Практическая работа №23. Выполнение резервного копирования
- Практическая работа № 24. Восстановление базы данных из резервной копии
- Практическая работа №25-26. Реализация доступа пользователей к базе данных
- Практическая работа №27-28. Мониторинг безопасности работы с базами данных
- Практическая работа №29-30. Установка приоритетов
- Практическая работа №31. Развертывание контроллеров домена
- Практическая работа №32. Мониторинг сетевого трафика

Пояснительная записка

Методические указания по выполнению практических работ по ПМ.11 РАЗРАБОТКА, АДМИНИСТРИРОВАНИЕ И ЗАЩИТА БАЗ ДАННЫХ разработаны в соответствии с рабочей программой и предназначены для приобретения необходимых практических навыков и закрепления теоретических знаний, полученных обучающимися при изучении ПМ, обобщения и систематизации знаний перед экзаменом.

Методические указания предназначены для обучающихся специальности 09.02.07 Информационные системы и программирование.

Освоение содержания ПМ.11 РАЗРАБОТКА, АДМИНИСТРИРОВАНИЕ И ЗАЩИТА БАЗ ДАННЫХ во время выполнения практических работ обеспечивает достижение обучающимися следующих **результатов**:

Код ПК, ОК	Умения	Знания
ОК 1-11	работать с современными case-средствами проектирования баз данных; проектировать логическую и физическую схемы базы данных; создавать хранимые процедуры и триггеры на базах данных; применять стандартные методы для защиты объектов базы данных; выполнять стандартные процедуры резервного копирования и мониторинга выполнения этой процедуры; выполнять процедуру восстановления базы данных и вести мониторинг выполнения этой процедуры; обеспечивать информационную безопасность на уровне базы данных	основные положения теории баз данных, хранилищ данных, баз знаний; основные принципы структуризации и нормализации базы данных; основные принципы построения концептуальной, логической и физической модели данных; методы описания схем баз данных в современных системах управления базами данных; структуры данных систем управления базами данных, общий подход к организации представлений, таблиц, индексов и кластеров; методы организации целостности данных; способы контроля доступа к данным и управления привилегиями; основные методы и средства защиты данных в базах данных
ПК 11.1-11.6		

Порядок выполнения практической работы

- записать название работы, ее цель в тетрадь;
- выполнить основные задания в соответствии с ходом работы;
- выполнить индивидуальные задания.

Рекомендации по оформлению практической работы

Задания выполняются обучающимися по шагам. Необходимо строго придерживаться порядка действий, описанного в практической работе

Результаты выполнения практических заданий необходимо сохранять в своей папке на компьютере или USB – накопителе.

В случае пропуска занятий обучающийся осваивает материал самостоятельно в свободное от занятий время и сдает практическую работу с пояснениями о выполнении.

Критерии оценки практической работы

- наличие цели выполняемой работы, выполнение более половины основных заданий (удовлетворительно);
- наличие цели выполняемой работы, выполнение всех основных и более

половины дополнительных заданий (хорошо);

- наличие цели выполняемой работы, выполнение всех основных и индивидуальных заданий (отлично).

Практическая работа №1 Сбор и анализ информации

Цель практической работы

Получить теоретические знания и практические навыки реализации баз данных (БД). Осуществить анализ предметной области. Освоить концептуальное проектирование и научиться определять сущности и атрибуты БД. Научиться разрабатывать инфологическую модель БД в виде ER-диаграмм. Получить теоретические знания и практические навыки при физическом проектировании баз данных (БД). Научиться создавать даталогическую модель БД.

I. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ БАЗЫ ДАННЫХ

Понятие БД и СУБД

Информационная система - система, реализующая автоматизированный сбор, обработку и манипулирование данными и включающая технические средства обработки данных, программное обеспечение и соответствующий персонал.

Цель любой информационной системы - обработка данных об объектах реального мира. Основой информационной системы является **база данных**. В широком смысле слова **база данных** - это совокупность сведений о конкретных объектах реального мира в какой-либо предметной области.

Под **предметной областью** принято понимать часть реального мира, подлежащего изучению для организации управления и в конечном счете автоматизации, например, предприятие, вуз и т.д.

Создавая базу данных, пользователь стремится упорядочить информацию по различным признакам и быстро производить выборку с произвольным сочетанием признаков. Большое значение при этом приобретает структурирование данных.

Структурирование данных - это введение соглашений о способах представления данных.

Неструктурированными называют данные, записанные, например, в текстовом файле.

Ниже приведен пример неструктурированных и структурированных данных, содержащих сведения о студентах (номер личного дела, фамилию, имя, отчество и год рождения).

Неструктурированные данные:

Личное дело № 16493. Сергеев Петр Михайлович, дата рождения 1 января 1976 г.; Л/д № 16593, Петрова Анна Владимировна, дата рожд. 15 марта 1975 г.; № личн.дела 16693, д.р. 14.04.76, Анохин Андрей Борисович

Легко убедиться, что сложно организовать поиск необходимых данных, хранящихся в неструктурированном виде.

Структурированные данные:

№ личного	Фамилия	Имя	Отчество	Дата рождения
16493	Сергеев	Петр	Михайлович	01.01.76
16593	Петрова	Анна	Владимировна	15.03.75
16693	Анохин	Андрей	Борисович	14.04.76

В современной технологии баз данных предполагается, что создание базы данных, ее поддержка и обеспечение доступа пользователей к ней осуществляются централизованно с помощью специального программного инструментария - **системы управления базами данных (СУБД)**.

База данных (БД) - это поименованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области.

Объектом называется элемент предметной области, информацию о котором мы сохраняем.

Объект может быть **реальным** (например, человек, изделие; или населенный пункт) и **абстрактным** (например, событие, счет покупателя или изучаемый студентами курс).

Так, в области продажи автомобилей примерами **объектов** могут служить **МОДЕЛЬ АВТОМОБИЛЯ, КЛИЕНТ и СЧЕТ**. На товарном складе - это **ПОСТАВЩИК, ТОВАР, ОТПРАВЛЕНИЕ** и т. д.

Понятие базы данных тесно связано с такими понятиями структурных элементов, как **поле, запись, файл (таблица)** (рис.1).

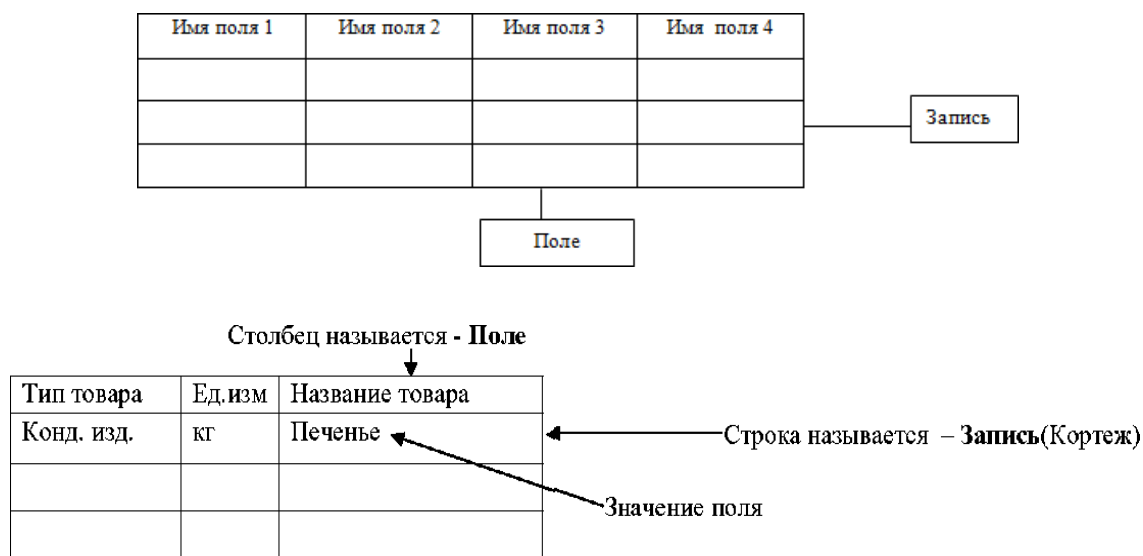


Рис.1 Основные структурные элементы БД

Структурные элементы базы данных

Поле - элементарная единица логической организации данных, которая соответствует неделимой единице информации - реквизиту. Для описания поля используются следующие характеристики:

- **имя**, например, **Фамилия, Имя, Отчество, Дата рождения;**
- **тип**, например, символьный, числовой, денежный;
- **длина**, например, 15 байт, причем будет определяться максимально возможным количеством символов;
- **точность** для числовых данных, например два десятичных знака для отображения дробной части числа,

Запись - совокупность логически связанных полей.

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

Файл (таблица) - совокупность экземпляров записей одной структуры.

Описание логической структуры записи файла содержит последовательность

расположения полей записи и их основные характеристики, как это показано на рис. 2 из3.

Имя файла					
Поле		Признак ключа	Формат поля		
Имя (обозначение)	Полное наименование		Тип	Длина	Точность (для чисел)
имя 1					
имя N					

Рис. 2. Описание логической структуры записи файла

В структуре записи файла указываются поля, значения которых являются ключами:

первичными (ПК) и внешними (ВК),

Первичный ключ (ПК) - 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 (ВК) - 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.

Имя файла: СТУДЕНТ					
Поле		Признак ключа	Формат поля		
Обозначение	Наименование		Тип	Длина	Точность
Номер	№ личного дела	•	Симв	5	
Фамилия	Фамилия студента		Симв	15	
Имя	Имя студента		Симв	10	
Отчество	Отчество студента		Симв	15	
Дата	Дата рождения		Дата	8	

Рис. 3. Описание логической структуры записи файла СТУДЕНТ

Понятие модели данных

Для того, чтобы спроектировать структуру базы данных, необходима исходная информация о предметной области. Желательно, чтобы эта информация была представлена в формализованном виде.

Такое формализованное описание предметной области будем называть **инфологической (infological) моделью предметной области (ИЛМ) или концептуальной моделью (КМ).**

Ядром любой базы данных является модель данных. **Модель данных** представляет собой множество структур данных, ограничений целостности и операций манипулирования данными. С помощью модели данных могут быть представлены объекты предметной области и взаимосвязи между ними.

Модель данных - совокупность структур данных и операций их обработки.

СУБД основывается на использовании **иерархической, сетевой или реляционной модели**, на комбинации этих моделей или на некотором их подмножестве.

Самой распространенной моделью данных является - **реляционная**.

Иерархическая модель данных

Иерархическая модель организует данные в виде древовидной структуры

К основным понятиям иерархической структуры относятся: **уровень**, **элемент (узел)**, **связь**. Дерево представляет собой иерархию элементов, называемых узлами. **Узел** - это совокупность атрибутов данных, описывающих некоторый объект. На самом верхнем уровне иерархии имеется один и только один узел - **корень**. Каждый узел, кроме корня, связан с одним узлом на более высоком уровне, называемым **исходным** для данного узла. Ни один элемент не имеет более одного исходного. Каждый элемент может быть связан с одним или несколькими элементами на более низком уровне. Они называются **порожденными** (рис. 4).

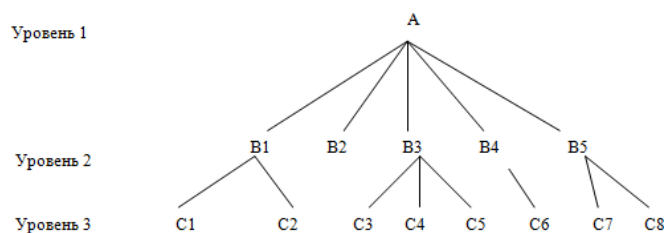


Рис. 4. Графическое изображение иерархической структуры БД

К каждой записи базы данных существует только один (иерархический) путь от корневой записи. Например, как видно из рис. 4, для записи C4 путь проходит через записи A и B3.

Сетевая модель данных

Сетевая модель организует данные в виде сетевой структуры.

Структура называется сетевой, если в отношениях между данными порожденный элемент имеет более одного исходного.

В сетевой структуре при тех же основных понятиях (уровень, узел, связь) каждый элемент может быть связан с любым другим элементом.

На рис. 5 изображена сетевая структура базы данных в виде графа.

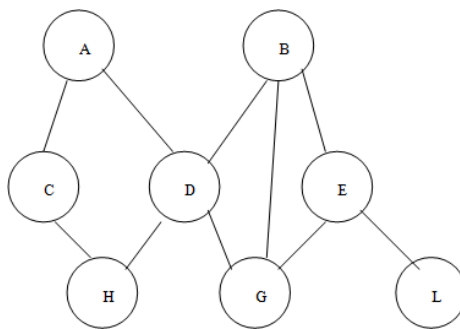


Рис. 5. Графическое изображение сетевой структуры

Реляционная модель данных

Реляционная модель данных является совокупностью взаимосвязанных двумерных таблиц объектов модели.

Например, реляционной таблицей можно представить информацию о студентах, обучающихся в вузе (рис. 6).

№ личного дела	Фамилия	Имя	Отчество	Дата рождения	Группа
16493	Сергеев	Петр	Михайлович	01.01.76	111
16593	Петрова	Анна	Владимировна	15.03.75	112
16693	Анохин	Андрей	Борисович	14.04.76	111

Рис. 6. Пример реляционной таблицы

Связи между двумя логически связанными таблицами в реляционной модели устанавливаются по равенству значений одинаковых атрибутов этих таблиц.

Каждая реляционная таблица представляет собой двумерный массив и обладает следующими **свойствами**:

- - **каждый элемент таблицы - один элемент данных;**
- - **все столбцы в таблице однородные, т.е. все элементы в столбце имеют одинаковый тип (числовой, символьный и т.д.) и длину;**
- - **каждый столбец имеет уникальное имя;**
- - **одинаковые строки в таблице отсутствуют;**
- - **порядок следования строк и столбцов может быть произвольным.**

При описании реляционной модели часто используют следующие термины:

отношение, кортеж, домен.

Отношения представлены в виде таблиц, строки которых соответствуют **записям (кортежам)**, а столбцы полям, атрибутам отношений (**доменам**).

Поле, каждое значение которого однозначно определяет соответствующую запись, называется простым ключом (ключевым полем). Если записи однозначно определяются значениями нескольких полей, то такая таблица базы данных имеет **составной ключ**.

В примере, показанном на рис.6, ключевым полем таблицы является "№ личного дела".

Между двумя реляционными таблицами могут быть сформированы **связи**. Различные таблицы, могут быть **связаны между собой через общее поле данных**.

На рис. 7 показан пример реляционной модели, построенной на основе отношений:

СТУДЕНТ, СЕССИЯ, СТИПЕНДИЯ.

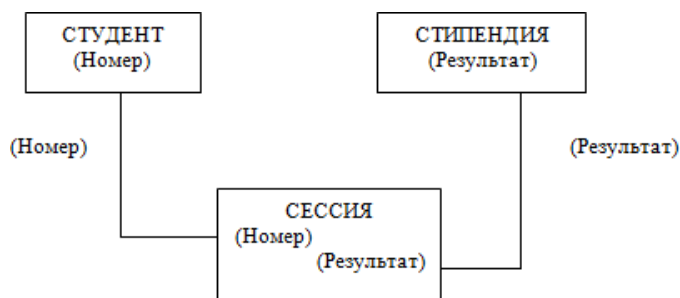


Рис.7. Пример реляционной модели

Таблица **СТУДЕНТ** имеет поля: Номер, Фамилия, Имя, Отчество, Дата рождения, Группа;
СЕССИЯ - Номер, Оценка 1, Оценка 2, Оценка 3, Оценка 4, Результат;
СТИПЕНДИЯ - Результат, Процент

Таблицы **СТУДЕНТ** и **СЕССИЯ** имеют совпадающие ключи (**Номер**), что дает возможность легко организовать связь между ними.

Таблица **СЕССИЯ** имеет **первичный ключ Номер** и содержит **внешний ключ Результат**, который обеспечивает ее связь с таблицей **СТИПЕНДИЯ**.

Благодаря имеющимся связям достигаются следующие преимущества:

1. Удастся избежать дублирования информации. Все необходимые данные можно хранить только в одной таблице. Так, например, нет необходимости в таблице **СЕССИЯ** хранить номер группы каждого студента, сдающего экзамены, достаточно задать связь с таблицей **СТУДЕНТ**.

2. В реляционных базах данных легко производить изменения. Если в таблице

СЕССИЯ изменить какие-нибудь значения, то правильная информация автоматически будет связана с другими таблицами, ссылающимися на первую (например, таблица **СТИПЕНДИЯ**).

3. В нереляционных базах данных сложно передать все имеющиеся зависимости, т.е. связать друг с другом данные из различных таблиц. Реляционная база данных выполняет все эти действия автоматически.

4. В реляционных базах данных удастся легко избежать установления ошибочных связей между различными таблицами данных, а необходимый объем памяти сокращен до минимума.

2. ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

Анализ и описание предметной области БД

Предметной областью называется фрагмент реальности, который описывается или моделируется с помощью БД и ее приложений. В предметной области выделяются информационные объекты – идентифицируемые объекты реального мира, процессы, системы, понятия и т.д., сведения о которых хранятся в БД.

Анализ предметной области выполняется в следующем порядке: название предприятия, цель деятельности предприятия, структура предприятия, информационные потребности пользователей. Для описания ПО следует привести основные решаемые задачи БД. Дать словесное описание процесса функционирования ПО, проанализировать основные хозяйственные операции, которые совершаются в ПО. Привести анализ структуры предприятия, перечислить задачи, решаемые отдельными подразделениями.

Анализ предметной области целесообразно разбить на три фазы:

1. анализ концептуальных требований и информационных потребностей;
2. выявление информационных объектов и связей между ними;
3. построение концептуальной модели предметной области и проектирование концептуальной схемы БД.

Анализ концептуальных требований и информационных потребностей

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

Рассмотрим примерный состав вопросника при анализе различных предметных областей.

Пример. Предлагается разработать БД для учета студентов вуза. Анализ предметной области:

1. Сколько студентов учится в вузе?
2. Сколько факультетов и отделений в вузе?
3. Как распределены студенты по факультетам отделений и курсам?
4. Сколько дисциплин читается на каждом курсе по каждой специальности?
5. Как часто обновляется информация в БД?
6. Сколько преподавателей в вузе?
7. Сколько иногородних студентов живет в общежитии, на частных квартирах?
8. Сколько лекционных аудиторий и аудиторий для проведения практических занятий, лабораторий?
9. Какая преемственность существует между читаемыми курсами?
10. Как информация, представленная в п.п. 1-9, используется в настоящее время (расписание занятий, экзаменов, зачетов и т.д.) и как ее собираются использовать?
11. Сколько раз в день, сколько человек и кто пользуются БД?

Выявление информационных объектов и связей между ними

Вторая фаза анализа предметной области состоит в выборе информационных объектов, задании необходимых свойств для каждого объекта, выявлении связей между объектами, определении ограничений, накладываемых на информационные объекты, типы связей между ними, характеристики информационных объектов. Проанализируем предметную область на примере БД "Видеомагнитофоны".

При выборе информационных объектов постараемся ответить на ряд вопросов:

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

Пример. БД "Видеомагнитофоны", рассчитанной на пользователей, которые хотят приобрести данный вид техники.

После беседы с различными пользователями и просмотра каталогов было выяснено, что интерес представляют три информационных объекта: видеомагнитофон, видеоплеер, видеокассета. Рассмотрим наиболее существенные характеристики каждого информационного объекта.

Объект - ВИДЕОМАГНИТОФОН.

Атрибуты - страна-изготовитель, фирма-изготовитель, № модели, телевизионные системы, число кассетных гнезд, ресурс непрерывной работы, система автопоиска, напряжение в сети, наличие таймера, число программ, габаритные размеры, масса, цена в долларах, год выпуска.

Объект - ВИДЕОПЛЕЙЕР,

Атрибуты - страна-изготовитель, фирма-изготовитель, № модели, телевизионные системы, число воспроизводящих головок, ресурс непрерывной работы, напряжение в сети, наличие таймера, габаритные размеры, масса, цена в долларах, год выпуска.

Объект - ВИДЕОКАССЕТА.

Атрибуты - наименование, страна-изготовитель, фирма-изготовитель, тип кассеты, время проигрывания, цена в долларах.

Далее выделим связи между информационными объектами. В ходе этого процесса постараемся ответить на следующие вопросы:

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

Попытаемся задать ограничения на объекты и их характеристики.

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

Введем следующие ограничения:

1. Значение атрибута "число кассетных гнезд" изменяется от 1 до 2.
2. Значение атрибута "ресурс непрерывной работы" изменяется от 4 до 24.
3. Значение атрибута "напряжение в сети" изменяется от 110 до 240 В.
4. Значение атрибута "число программ" изменяется от 1 до 20 и т.д.

Связи между различными классами объектов.

Помимо классов объектов в ИЛМ отображают связи между различными классами объектов. Такие связи моделируют отношения между объектами различных видов в реальном мире. При отборе связей помещаемых в ИЛМ следует руководствоваться информационными потребностями пользователей базы данных.

Каждая связь характеризуется именем, типом, классом принадлежности и направлением. Имя связи должно быть глагольным оборотом, например

«Принадлежит», «Закреплены за», «Входит в» и т.д.

Все информационные объекты предметной области связаны между собой. Соответствия, отношения, возникающие между объектами предметной области, называются связями. Различаются связи нескольких типов, для которых введены следующие обозначения:

Различают четыре типа связи:

- **«один к одному» (1:1);**
- **«один ко многим» (1:M);**
- **«многие к одному» (M:1)**
- **«многие ко многим» (M:M).**

Рассмотрим эти типы связей на примерах.

Пример. Дана совокупность информационных объектов, отражающих учебный процесс в вузе:
СТУДЕНТ (Номер, Фамилия, Имя, Отчество, Пол, Дата рождения, Группа) СЕССИЯ
(Номер, Оценка 1, Оценка 2, Оценка 3, Оценка 4, Результат) СТИПЕНДИЯ (Результат,
Процент)
ПРЕПОДАВАТЕЛЬ (Код преподавателя, Фамилия, Имя, Отчество)

Связь один к одному (1:1) предполагает, что в каждый момент времени одному экземпляру информационного объекта А соответствует не более одного экземпляра информационного объекта В и наоборот. Рис. 8 иллюстрирует указанный тип отношений.

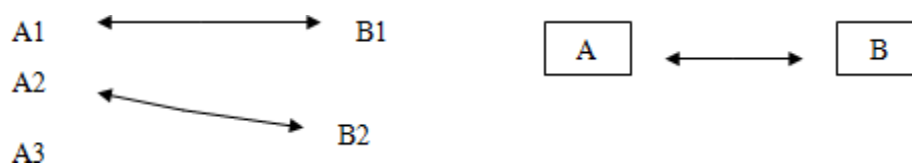


Рис. 8. Графическое изображение реального отношения 1:1

Примером связи 1:1 может служить связь между информационными объектами СТУДЕНТ и СЕССИЯ:

СТУДЕНТ <-> СЕССИЯ

Каждый студент имеет определенный набор экзаменационных оценок в сессию. При связи **один ко многим (1 : M)** одному экземпляру информационного объекта А соответствует 0, 1 или более экземпляров объекта В, но каждый экземпляр объекта В связан не более чем с 1 экземпляром объекта А. Графически данное соответствие имеет вид, представленный на рис. 9.

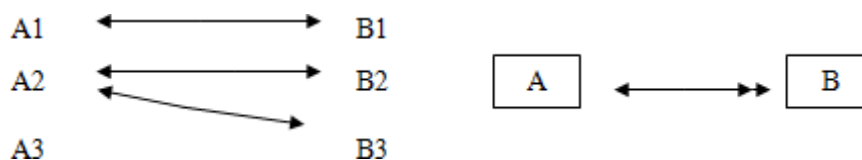


Рис. 9. Графическое изображение реального отношения 1:M

Примером связи 1: M служит связь между информационными объектами СТИПЕНДИЯ и СЕССИЯ:

СТИПЕНДИЯ <--> СЕССИЯ

Установленный размер стипендии по результатам сдачи сессии может повторяться многократно для различных студентов.

Связь **многие ко многим** (M:M) предполагает, что в каждый момент времени одному экземпляру информационного объекта A соответствует 0, 1 или более экземпляров объекта B и наоборот. На рис. 10 графически представлено указанное соответствие.

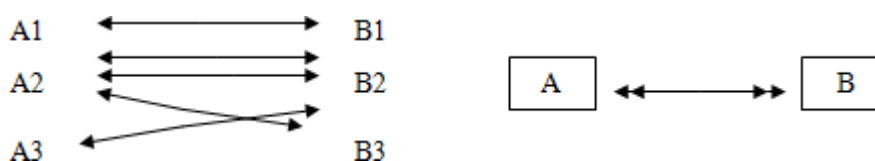


Рис. 10. Графическое изображение реального отношения M : M

Примером данного отношения служит связь между информационными объектами СТУДЕНТ и ПРЕПОДАВАТЕЛЬ:

СТУДЕНТ <<-->ПРЕПОДАВАТЕЛЬ

Один студент обучается у многих преподавателей, один преподаватель обучает многих студентов.

Связь «многие к одному» при создании БД физически обычно организуется путем введения дополнительного поля в таблицу со стороны «много». Это поле называется **внешний ключ**. На рис. 10. код группы - внешний ключ.

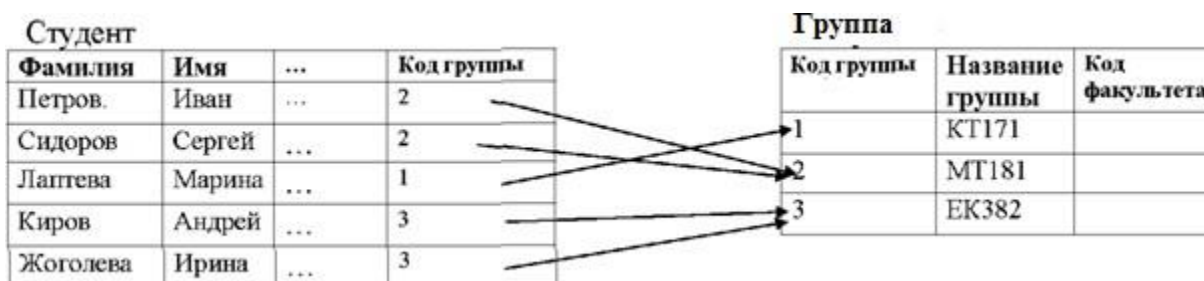


Рис. 10. Введение внешнего ключа в БД

Построение инфологической (концептуальной модели) предметной области

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

Концептуальная модель включает описания объектов и их взаимосвязей, представляющих интерес в рассматриваемой предметной области (ПО) и выявляемых в результате анализа данных.

Концептуальная модель применяется для структурирования предметной области с учетом информационных интересов пользователей системы. Она дает возможность систематизировать информационное содержание предметной области, позволяет как бы

"подняться вверх" над ПО и увидеть ее отдельные элементы. При этом уровень детализации зависит от выбранной модели.

Концептуальная модель является представлением точки зрения пользователя на предметную область и не зависит ни от программного обеспечения СУБД, ни от технических решений.

Концептуальная модель должна быть стабильной. Могут меняться прикладные программы, обрабатывающие данные, может меняться организация их физического хранения, концептуальная модель остается неизменной или увеличивается с целью включения дополнительных данных.

Одной из распространенных моделей концептуальной схемы является модель «**сущность - связь**» (**ER-моделей** (или ER-диаграмм)). Основными конструкциями данной модели являются сущности и связи.

Под **сущностью** понимают основное содержание объекта ПО, о котором собирают информацию. В качестве сущности могут выступать место, вещь, личность, явление.

Экземпляр сущности - конкретный объект.

Например:

сущность (объект) - служащий экземпляр сущности - Иванов А.В.; сущность (объект) - институт экземпляр сущности - МГУ.
--

Сущность принято определять **атрибутами** - поименованными характеристиками.

Например:

сущность - служащий атрибуты: ФИО, год рождения, адрес, образование и т.д.

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

Связь определяет отношения между сущностями. **Типы связей: один к одному, один ко многим, многие ко многим.**

При построении модели «сущность - связь» используют **графические диаграммы**. При этом обозначают: сущности - прямоугольниками, атрибуты - овалами, связи - ромбами, см. рис.11.

На практике приходится строить несколько вариантов моделей, из которых выбирается одна, наиболее полно отображающая предметную область.

Классом объектов называют совокупность объектов, обладающих одинаковым набором свойств. Например, для объектов класса «СТУДЕНТ» таким набором свойств являются: «ГОД_РОЖДЕНИЯ», «ПОЛ» и др.

Объекты могут быть реальными, как названный выше объект «СТУДЕНТ», и абстрактными, как, например, «ПРЕДМЕТЫ», которые изучают студенты.

Пример. Спроектировать БД "Сессия". База данных должна выдавать оперативную информацию об успеваемости студентов на факультетах в семестре. Результатами сессии считать только экзамены.

По сути дела и БД исходя из формулировки задания можно выделить лишь одно приложение. Речь идет об успеваемости студентов разных факультетов по тем или иным дисциплинам. Более конкретно речь идет о выдаче справок по результатам сессии каждого студента, учебной группы, курса, факультета, а также об автоматизированном составлении ведомости

Выберем следующие сущности:

**ИНСТИТУТ, ФАКУЛЬТЕТ, СТУДЕНТ, ПРЕПОДАВАТЕЛЬ,
ДИСЦИПЛИНА.**

В данном примере можно выделить сущность ЭКЗАМЕН или ВЕДОМОСТЬ, но можно не выделять, а сформировать ведомость из имеющихся данных посредством связей.

Зададим каждую сущность набором атрибутов:

ИНСТИТУТ (название, подчиненность, адрес, телефон, ФИО ректора)

ФАКУЛЬТЕТ (название, код специальности, данные о кафедрах, число выпускников, декан).

СТУДЕНТ (ФИО, группа, курс, номер текущего семестра, пол).

ПРЕПОДАВАТЕЛЬ (ФИО, должность, звание, кафедра, стаж).

ДИСЦИПЛИНА (название, число часов, код дисциплины, виды занятий, число читаемых семестров, номера текущих семестров, на каких курсах преподается)

В каждом наборе атрибутов, характеризующих сущность, необходимо выбрать ключевые атрибуты, т.е. атрибуты, делающие сущность уникальной. При задании атрибутов ключевые атрибуты подчеркивались.

Определим связи между сущностями.

Название связи **Связи между сущностями**

учится	студент, факультет
изучает	студент, дисциплина
имеет	институт, факультет
работает	преподаватель, факультет
преподает	преподаватель, дисциплина
экзамен	студент, дисциплина, преподаватель

После выбора сущностей, задания атрибутов и анализа связей можно перейти к проектированию информационной (концептуальной) схемы БД.

Концептуальная схема БД "Успеваемость» представлена на рис.11 (атрибуты сущностей на диаграмме не показаны).

Рассмотрим некоторые **ограничения** в рассматриваемом примере:

1. Значение атрибута "телефон" (сущность - ИНСТИТУТ) задается целым положительным шестизначным числом.
2. Значение атрибута "код факультета" (сущность - ФАКУЛЬТЕТ) лежит в интервале 1-10.
3. Значение атрибута "курс" (сущность - СТУДЕНТ) лежит в интервале 1 - 6
4. Значение атрибута "семестр" (сущность - СТУДЕНТ, ДИСЦИПЛИНА) лежит в интервале 1-12.
5. Значение атрибута "число часов" (сущность - ДИСЦИПЛИНА) лежит в интервале 1-300.
6. Одному студенту может быть приписана только одна группа.
7. Один студент может учиться только на одном факультете.
8. Один студент в семестре сдает от 3 до 5 дисциплин
9. Один студент изучает в семестре от 6 до 12 дисциплин.
10. Одному преподавателю приписывается только одна кафедра.
11. Один студент может передавать одну дисциплину не более трех раз.
12. Ключи: название института, название факультета, ФИО и группа студента, ФИО и кафедра преподавателя, название дисциплины.

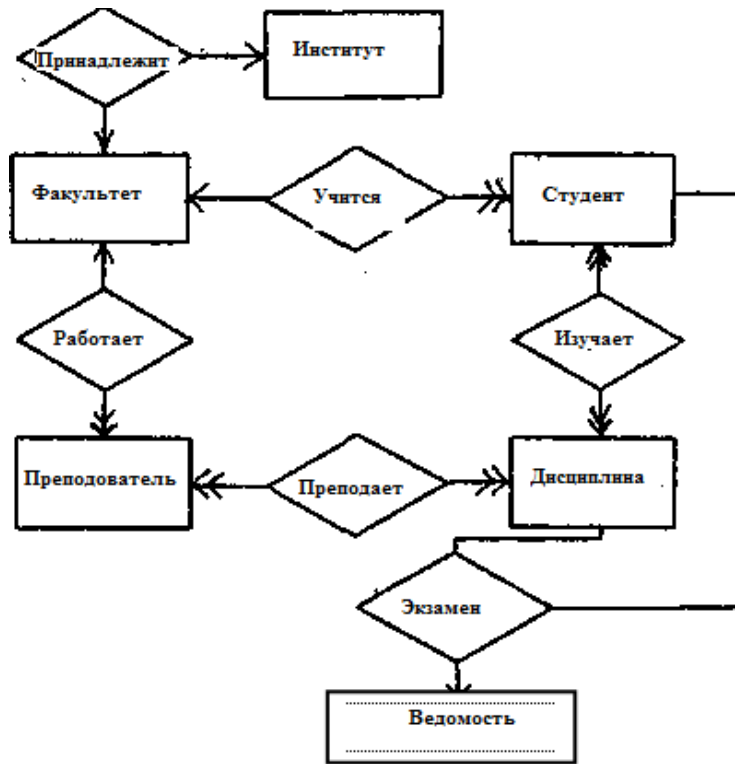


Рис 11. Концептуальная схема БД «Успеваемость»

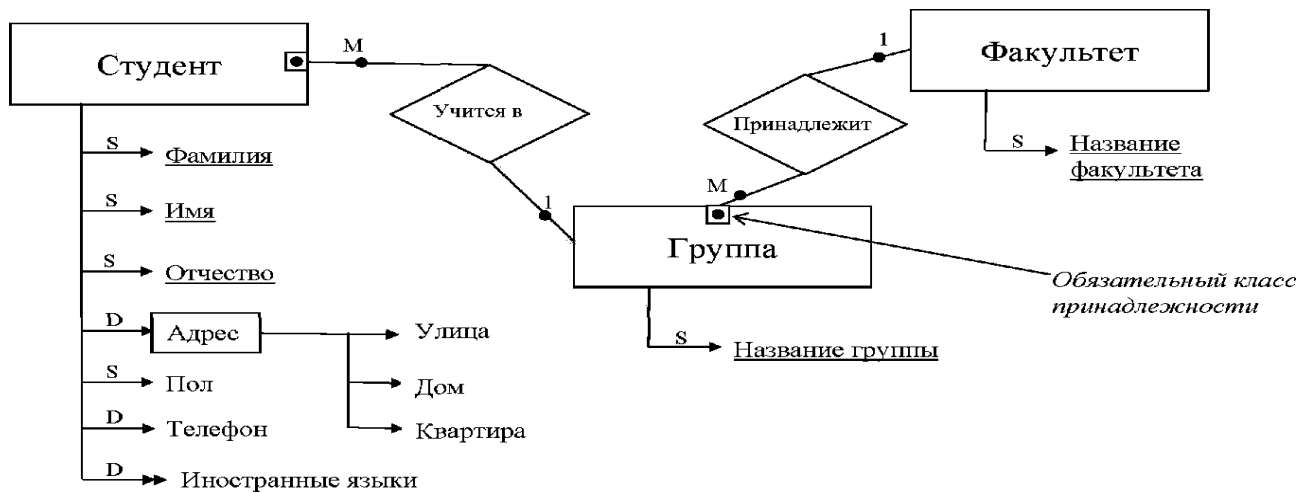


Рис. 12. Пример построения инфологической модели данных.

3. ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

Даталогическое проектирование

Инфологическая модель является исходной для построения *дательгической* модели БД и служит промежуточной моделью для специалистов предметной области (для которой создается банк данных (БнД)) и администратора БД в процессе проектирования разработки конкретной БД.

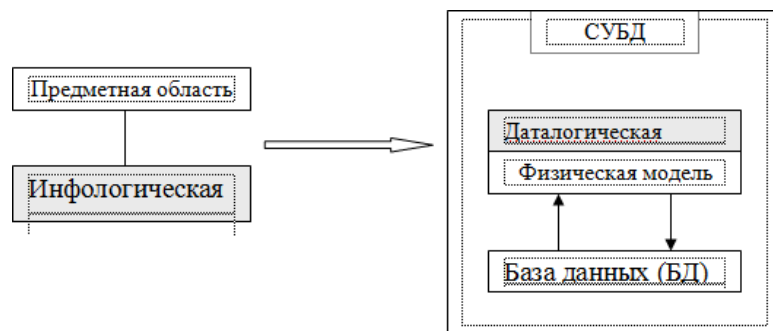


Рис. 13. Структура проектирования БД

Под *дatalogической* понимается модель, отражающая логические взаимосвязи между элементами данных безотносительно их содержания и физической организации. При этом даталогическая модель разрабатывается с учетом конкретной реализации СУБД, также с учетом специфики конкретной предметной области на основе ее инфологической модели.

Инфологическая модель предметной области строится первой. Предварительная инфологическая модель строится еще на предпроектной стадии и затем уточняется на более поздних стадиях проектирования баз данных. Затем на ее основе строятся концептуальная (логическая), внутренняя (физическая) и внешняя модели.

Конечным результатом даталогического проектирования является описание логической структуры базы данных на языке программирования. Однако если проектирование выполняется «вручную», то для большей наглядности сначала строится схематическое графическое изображение структуры базы данных. При этом должно быть обеспечено однозначное соответствие между конструкциями языка описания данных и графическими обозначениями информационных единиц и связей между ними.

Графическое представление используется и при автоматизированном проектировании структуры базы данных как интерфейсное средство общения с проектировщиком, и при документировании проекта.

Спроектировать логическую структуру базы данных означает определить все информационные единицы и связи между ними, задать их имена; если для информационных единиц возможно использование разных типов, то определить их тип. Следует также задать некоторые количественные характеристики, например, длину поля.

Описание даталогической модели.

Даталогическая модель представляет собой описание базы данных, выполненное в терминах используемой СУБД. Наиболее часто при разработках баз данных применяют реляционные СУБД. Для СУБД этого типа даталогическая удобно представить в виде набора таблиц специальной формы (табл. 1.4.).

Такая таблица составляется для каждого отношения, используемого в базе данных. Отношения в базе соответствуют классам объектов из инфологической модели. Кроме того, отношения могут представлять некоторые связи предметной области.

Каждой таблице нужно поставить в соответствие ее **ключи**. Схема ключа представляет собой перечисление атрибутов отношения, составляющих ключ.

Различают простые и составные ключи. **Простой ключ** строится на основе одного атрибута. **Составной ключ** строится на базе использования нескольких атрибутов.

Ключи принято разделять на **первичные, внешние и вспомогательные**.

Первичный индекс должен быть только один для каждой таблицы. Значения атрибутов, используемых для формирования первичного ключа, должны быть

уникальными для каждой записи в таблице. Значения первичного ключа уникально идентифицируют каждую запись. Не может быть двух записей в таблице с одинаковым значением первичного ключа. Например, в качестве первичного ключа для отношения «Сотрудники» можно выбрать атрибут «Табельный номер», значение которого является уникальным для каждой записи о сотруднике.

Внешние ключи используются для реализации связей типа **1:M** между отношениями. Внешний ключ строится для отношения, находящегося на стороне «**много**» связи 1:M. Для каждого такого отношения на даталогической модели должен быть показан внешний ключ. Внешний ключ всегда должен иметь соответствующий ему первичный ключ отношения, находящегося на стороне «**один**» связи типа 1:M.

Для связанных ключа «внешний - первичный» соединяются на схеме даталогической модели ломаной линией. Например, если в таблице «Сотрудники» имеется атрибут «Шифр категории», то этот атрибут можно использовать в качестве внешнего ключа для связи с таблицей «Категории». В последней таблице должен быть первичный ключ по полю «Шифр категории». Внешних ключей может быть несколько для одной таблицы.

Следует отметить, что первичные и внешние ключи строятся как правило на основе целочисленных атрибутов, а не атрибутов, имеющих строковый или вещественный тип.

Кроме первичных и внешних ключей часто используют **вспомогательные индексы**. Они применяются для реализации связей, получения нужного упорядочения при выводе на экран и создании отчетов и т.д. В каждом случае использования вспомогательного индекса, его необходимость должна быть обоснована. Применение большого количества индексов замедляет работу СУБД, т.к. операции над записями отношения требуют корректировки всех индексов.

II. ЗАДАНИЕ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКОЙ РАБОТЫ

Практическая работа №1 выполняется письменно и в конце занятия сдается на проверку. После проверки будет выставлена оценка.

Выбор задания

Выбрать из таблицы «**Варианты заданий для лаб. работы №1.doc**» вариант задания, соответствующий **номеру студента в списке учебной группы**. Для всех последующих практических работ вариант остается неизменным. Каждому студенту предоставляется свой вариант предметной области (ПО), который он будет использовать в процессе выполнения всех практических работ.

Анализ предметной области.

На основании выбранного варианта привести: название предприятия, цель деятельности предприятия, структура предприятия, информационные потребности пользователей (кратко).

Описание основных сущностей ПО.

Здесь следует привести описание основных сущностей (объектов) ПО. Отбор сущностей производится на основе анализа информационных потребностей. Необходимо привести таблицы описания сущностей (сущностей должно быть не менее 3-х)

Таблица 1.1. Список сущностей предметной области.

№ п.п.	Наименование сущности	Краткое описание

Здесь же приводится отбор атрибутов (не менее 5-ти) для каждого экземпляра сущности. Отбираются только те атрибуты сущностей, которые необходимы для формирования ответов на регламентированные и непредусмотренные запросы. Для каждого объекта следует привести таблицы его атрибутов.

Таблица 1.2. Список атрибутов.

№ п.п.	Наименование атрибута	Краткое описание

На основе анализа информационных запросов следует выявить связи между сущностями. Для выявленных связей также нужно заполнить таблицу 1.3.

Таблица 1.3. Список связей ПО.

№ п.п.	Наименование связи	Сущности, участвующие в связи	Краткое описание

Построение инфологической модели.

На основании ранее выбранного варианта и таблиц 1.1-1.3:

- описать классы объектов (сущностей) и их свойства,
- расставить существующие связи между ними,
- на основании табл. 1.3. в письменной форме обосновать типы связей (1:1, 1:M и т.д.).

При графическом построении ИЛМ следует придерживаться единого масштаба для всей схемы. Все прямоугольники, обозначающие классы объектов, должны быть одного размера. Аналогично, все ромбы с именами связей также должны иметь одинаковый размер.

Построение даталогической модели.

На основании ранее выбранного варианта и таблиц 1.1-1.3, инфологической модели и нормализации БД необходимо:

- провести соответствие ключей для каждой таблицы 1.1-1.3,
- заполнить для каждой таблицы БД форму, согласно табл. 1.4.

Таблица 1.4. Структура таблицы для даталогической модели.

№ п.п.	Наименование реквизита	Идентификатор	Тип	Длина	Формат изображения	Ограничения и комментарий

III. СОДЕРЖАНИЕ ОТЧЕТА

1. Название и цель работы.
2. Словесный и схематический анализ предметной области (ПО), включая схему структуры предприятия.
3. Заполненные таблицы 1.1 - 1.3. с описанием основных сущностей ПО.
4. Инфологическая модель БД, согласно варианту.
5. Обоснование типов связи в инфологической модели данных.
6. Даталогическая модель БД (табл. 1.4.).

IV. ПРИМЕР ОФОРМЛЕНИЯ

Пример. Разработать базу данных «Учеба студентов».

Решение.

Шаг первый. Анализ предметной области.

Студенты учатся на одном из факультетов, возглавляемом деканатом, в функции

которого входит контроль за учебным процессом. В учебном процессе участвуют преподаватели кафедр, административно относящиеся к одному из факультетов. Каждому факультету могут принадлежать несколько кафедр. Студенты кафедр организованные в группы.

Преподаватели кафедр характеризуются фамилией именем и отчеством, должностью, научным званием, ставкой и стажем работы, адресом проживания, возрастом.

Каждая кафедра читает определенный набор закрепленных за ней дисциплин. Каждая дисциплина характеризуется своим полным названием, указанием общего количества часов и формы контроля (зачет, экзамен).

В конце каждого семестра составляется экзаменационно-зачетные ведомости, в которых указываются дисциплины и для каких групп проводится форма контроля, фамилия преподавателя и учебный год и семестр. В каждой такой ведомости составляется список студентов и выставляется оценка.

Шаг второй. Описание основных сущностей ПО.

В результате проведенного анализа предметной области базы данных «Учеба студентов» легко перечислить основные сущности этой БД. Так как на физическом уровне сущности соответствует таблица, то просто перечислим основные таблицы БД.

В реляционную модель проектированной БД будут входить следующие таблицы (сущности): Факультет, Кафедра, Преподаватели, Группы, Студенты, Дисциплины, Ведомости.

Список сущностей.

№	Название	Назначение
1	Факультет	Описание факультета и его деканата
2	Кафедра	Описание кафедры
3	Преподаватели	Описание состава сотрудников кафедр
4	Группы	Перечень групп, закрепленных за каждой кафедрой
5	Студенты	Перечень студентов каждой группы
6	Дисциплины	Перечень дисциплин, закрепленных за каждой кафедрой
7	Ведомости	Экзаменационно-зачетные ведомости с перечнем студентов и их оценками
8	Подчиненная ведомость	Это таблица внутри таблицы ведомости. Отражает связь один-ко-многим. Так как каждая ведомость выписывается каждой конкретной группе, а студентов в ней много.

Для каждой таблицы (сущности) приведем описание ее атрибутов. Атрибут на физическом уровне – это колонки таблицы и выражает определенное свойство объекта.

Список атрибутов таблицы «Факультеты»

Ключевое поле	Название	Назначение
ПК (первичный ключ)	Код факультета	Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждому факультету. Это целое число. Т.е. для идентификации каждого факультета будет применяться не названия самих факультетов, а определенный номер. Этот номер может быть случайным целым числом или счетчик по порядку.
	Название факультета	

	ФИО декана	
	Номер комнаты деканата	
	Телефон деканата	

Список атрибутов таблицы «Кафедра»

Ключевое поле	Название	Назначение
ПК (первичный ключ)	Код кафедры	Ключевое поле. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждой кафедре. Однако для идентификации каждой кафедры первичного ключа недостаточно, так как каждая кафедра принадлежит определенному факультету. Для этого будем использовать внешний ключ.
ВК (внешний ключ)	Код факультета	Внешний ключ – это атрибут отношения, который является первичным ключом другого отношения. В нашем случае это атрибут таблицы факультета. С помощью внешнего ключа будет определено к какому факультету принадлежит каждая кафедра.
	Название кафедры	
	ФИО заведующего	
	Номер комнаты кафедры	
	Телефон кафедры	

Список атрибутов таблицы «Преподаватели»

Ключевое поле	Название	Назначение
ПК (первичный ключ)	Код преподавателя	Ключевое поле. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждому преподавателю. Это например, может быть его табельный номер. Однако для идентификации каждого преподавателя первичного ключа недостаточно, так как каждый сотрудник принадлежит определенной кафедре. Для этого будем использовать внешний ключ.
ВК (внешний ключ)	Код кафедры	С помощью данного внешнего ключа будет определено к какой кафедре принадлежит каждый преподаватель.
	ФИО	
	должность	Ассистент, доцент, профессор, ст. преподаватель
	научное звание	К.т.н., проф., магистр, ст.н.с., м.н.с.
	ставка	
	стаж работы,	
	адрес проживания	
	возраст	

Список атрибутов таблицы «Группы»

Ключевое	Название	Назначение
----------	----------	------------

поле		
ПК (первичный ключ)	Код группы	Ключевое поле. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждой группе. Однако для идентификации каждой группы первичного ключа недостаточно, так как каждая группа принадлежит определенной кафедре. Для этого будем использовать внешний ключ.
ВК (внешний ключ)	Код кафедры	С помощью данного внешнего ключа будет определено к какой кафедре принадлежит каждая группа.
	Номер группы	
	Год поступления	
	Курс обучения	

Список атрибутов таблицы «Студенты»

Ключевое поле	Название	Назначение
ПК (первичный ключ)	Код студента	Ключевое поле. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждому студенту. Однако для идентификации каждого студента первичного ключа недостаточно, так как каждый студент принадлежит определенной группе. Для этого будем использовать внешний ключ.
ВК (внешний ключ)	Код группы	С помощью данного внешнего ключа будет определено к какой группе принадлежит каждый студент.
	ФИО	
	Год рождения	
	Адрес проживания	

Список атрибутов таблицы «Дисциплины»

Ключевое поле	Название	Назначение
ПК (первичный ключ)	Код дисциплины	Ключевое поле. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждой дисциплине. Однако для идентификации каждой дисциплины первичного ключа недостаточно, так как каждая дисциплина принадлежит определенной кафедре. Для этого будем использовать внешний ключ.
ВК (внешний ключ)	Код кафедры	С помощью данного внешнего ключа будет определено к какой кафедре принадлежит каждая дисциплина.
	Название дисциплины	
	Расчетка	
	Форма контроля	

Список атрибутов таблицы «Ведомости»

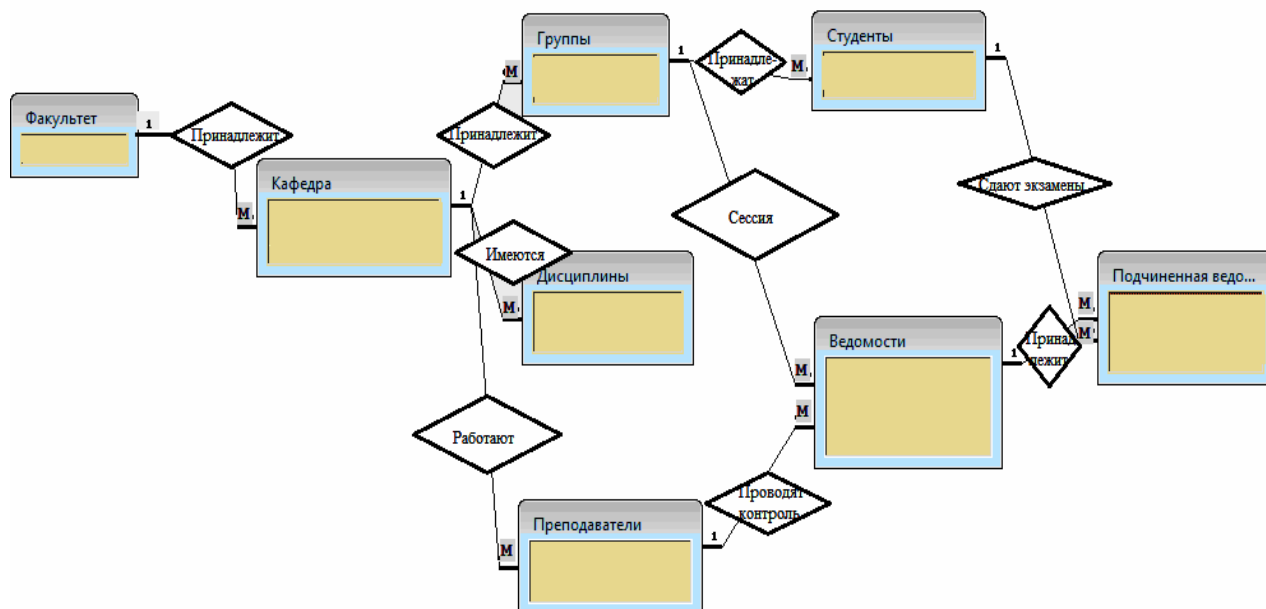
Ключевое поле	Название	Назначение
ПК (первичный ключ)	Код ведомости	Ключевое поле. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждой учебной ведомости. Однако для идентификации каждой ведомости первичного ключа недостаточно, так как каждая ведомость выписывается для определенной учебной группы по определенной дисциплине и преподавателя. Для этого будем использовать внешние ключи.
ВК (внешний ключ)	Код группы	С помощью данного внешнего ключа будет определено для какой группы выписывается ведомость.
ВК (внешний ключ)	Код дисциплины	С помощью данного внешнего ключа будет определено для какой дисциплины выписывается ведомость.
ВК (внешний ключ)	Код преподавателя	С помощью данного внешнего ключа будет определено какому преподавателю выписывается ведомость.
	Учебный год	
	Семестр	

Список атрибутов таблицы «Подчиненная таблица Ведомости»

Ключевое поле	Название	Назначение
ПК (первичный ключ)	Код под_ведомости	Ключевое поле. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждой подведомости. Однако для идентификации каждой подведомости первичного ключа недостаточно, так как каждая подведомость принадлежит определенной ведомости. Для этого будем использовать внешний ключ.
ВК (внешний ключ)	Код ведомости	С помощью данного внешнего ключа будет осуществлена связь с таблицей ведомости.
ВК (внешний ключ)	Код студента	С помощью данного внешнего ключа будет определен студент
	Оценка	

Шаг третий. Построение инфологической модели.

Инфологическую модель лучше представить графически, где будут изображены всетаблицы и связи между ними. В нашем случае схема связей представлена на рисунке.



Для выявленных связей заполним таблицу

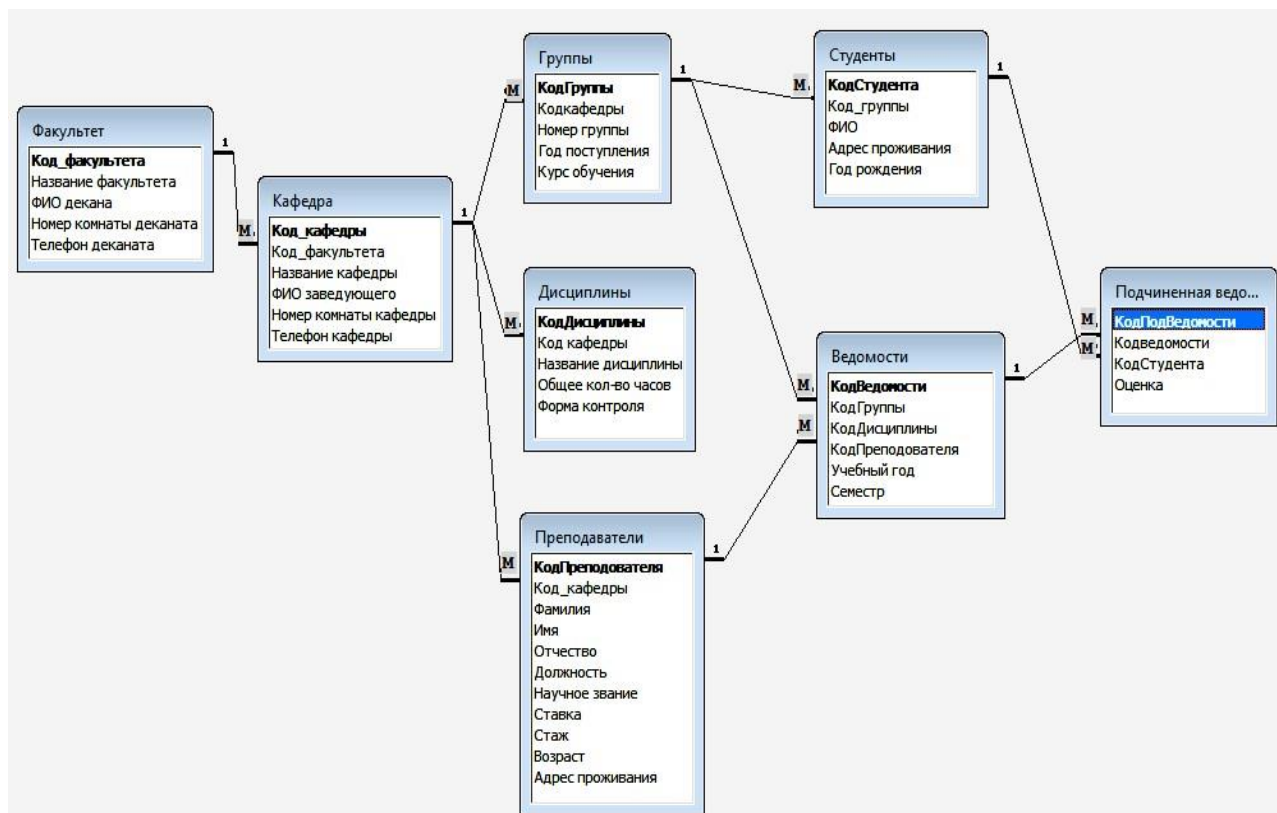
Список связей.

№	Название связи	Сущности, участвующие в связи	Назначение
1	1:M	Факультет-Кафедра	Одному факультету могут принадлежать несколько кафедр
2	1:M	Кафедра - Группа	Одной кафедре может принадлежать несколько групп
3	1:M	Кафедра - Дисциплины	Одной кафедре могут принадлежать несколько читаемых дисциплин
4	1:M	Кафедра - Преподаватели	На одной кафедре работает более одного преподавателя
5	1:M	Группа-Студенты	В каждой группе учится множество студентов
6	1:M	Группа - Ведомость	Каждой группе выписывают несколько ведомостей
7	1:M	Дисциплины - Ведомость	Ведомость выписывается из множества дисциплин
8	1:M	Преподаватели - Ведомость	Ведомость выписывается конкретному преподавателю
9	1:M	Ведомость-Подчиненная ведомость	Подчиненная ведомость принадлежит одной конкретной ведомости
10	1:M	Студенты-Подчиненная ведомость	В подчиненной ведомости перечислены все студенты группы

Шаг четвертый. Построение даталогической модели БД.

Даталогическая модель отражается графически в виде схемы базы данных, где указываются имена сущностей, их атрибуты и связи между сущностями.

В нашем случае схема связей представлена на рисунке.



Даталогическая модель БД представляется в виде набора таблиц специальной формы, в которых указываются наименование атрибута, идентификатор, тип, длина, формат, ограничения.

Таблица «Факультеты»

№	Название	Идентификатор	Тип	Не пусто	Ограничение
1	Код факультета	Kod_fakulteta	Числовой	Да	ПК (первичный ключ)
2	Название факультета	Name_fakulteta	Текстовый	Нет	
3	ФИО декана	FIO	Текстовый	нет	
4	Номер комнаты деканата	N_komnatu_dekanata	Текстовый	Нет	Например, 123/a
5	Телефон деканата	Telefon_dekanata	Текстовый	Нет	Например, 41-69-99

Список атрибутов таблицы «Кафедра»

№	Название	Идентификатор	Тип	Не пусто	Ограничение
1	Код кафедры	Kod_kafedru	Числовой	Да	ПК (первичный ключ)
2	Код факультета	Kod_fakulteta	Числовой	Да	ВК (внешний ключ)
3	Название кафедры	Name_kafedru	Текстовый		
4	ФИО	FIO	Текстовый	нет	

	заведующего				
5	Номер комнаты кафедры	N_komnatu_kafedru	Текстовый	Нет	Например, 123/а
6	Телефон кафедры	Telefon_kafedru	Текстовый	Нет	Например, 41-69-99

Список атрибутов таблицы «Преподаватели»

№	Название	Идентификатор	Тип	Не пусто	Ограничение
1	Код преподавателя	Kod_prepodavately	Числовой	Да	ПК (первичный ключ)
2	Код кафедры	Kod_kafedru	Числовой	Да	ВК (внешний ключ)
3	ФИО	FIO	Текстовый	Нет	
4	должность	Dolgnost	Текстовый	Нет	
5	научное звание	Zvanie	Текстовый	Нет	
6	ставка	Stavka	Числовой	Нет	Вещественное число Например, 0.5, 0.75, 1
7	стаж работы,	Stag	Числовой	Нет	Вещественное число
8	адрес проживания	Address	Текстовый	Нет	
9	возраст	Vozrast	Числовой	нет	

Список атрибутов таблицы «Группы»

№	Название	Идентификатор	Тип	Не пусто	Ограничение
1	Код группы	Kod_grupu	Числовой	Да	ПК (первичный ключ)
2	Код кафедры	Kod_kafedru	Числовой	Да	ВК (внешний ключ)
3	Номер группы	N_grupu	Текстовый	Нет	Например, МТ-461
4	Год поступления	God_post	Числовой	нет	
5	Курс обучения	Kurs	Числовой	Нет	Вычисляемое поле, как разность между текущей датой и годом поступления

Список атрибутов таблицы «Студенты»

№	Название	Идентификатор	Тип	Не пусто	Ограничение
1	Код студента	Kod_studenta	Числовой	Да	ПК (первичный ключ)
2	Код группы	Kod_grupu	Числовой	Да	ВК (внешний ключ)
3	ФИО	FIO	Текстовый	Нет	
4	Год рождения	God_rogdeniya	Числовой	нет	
5	Адрес проживания	Address	Текстовый	Нет	

Список атрибутов таблицы «Дисциплины»

№	Название	Идентификатор	Тип	Не пусто	Ограничение
1	Код дисциплины	Kod_disciplinu	Числовой	Да	ПК (первичный ключ)
2	Код кафедры	Kod_kafedru	Числовой	Да	ВК (внешний ключ)
3	Название дисциплины	Name_dis	Текстовый	Нет	
4	Расчасовка	Raschasovka	Числовой	нет	
5	Форма контроля	Kontrol	Текстовый	Нет	Два значения – экзамен или зачет

Список атрибутов таблицы «Ведомости»

№	Название	Идентификатор	Тип	Не пусто	Ограничение
1	Код ведомости	Kod_vedomosti	Числовой	Да	ПК (первичный ключ)
2	Код группы	Kod_grupu	Числовой	Да	ВК (внешний ключ)
3	Код дисциплины	Kod_disciplinu	Числовой	Да	ВК (внешний ключ)
4	Код преподавателя	Kod_prepodavately	Числовой	Да	ВК (внешний ключ)
5	Учебный год	God	Числовой	Нет	
6	Семестр	Semester	Числовой	Нет	Диапазон от 1-10

Список атрибутов таблицы «Подчиненная таблица Ведомости»

№	Название	Идентификатор	Тип	Не пусто	Ограничение
1	Код под_ведомости	Kod_pod_vedomosti	Числовой	Да	ПК (первичный ключ)
2	Код ведомости	Kod_edomosti	Числовой	Да	ВК (внешний ключ)
3	Код студента	Kod_studenta	Числовой	Да	ВК (внешний ключ)
4	Оценка	Osenka	Числовой	Нет	Диапазон от 0-12

Таблица 1. Варианты заданий для практической работы №1

№ варианта	Условие
Вариант №1	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – успеваемость студентов ВУЗА. БД состоит из следующих таблиц: факультеты, кафедры, учебные группы, студенты, ведомости успеваемости.</p>

	<p>Таблица факультеты имеет следующие атрибуты: название факультета, ФИО декана, номер комнаты, номер корпуса, телефон.</p> <p>Таблица кафедры имеет следующие атрибуты: название кафедры, факультет, ФИО заведующего, номер комнаты, номер корпуса, телефон, кол-во преподавателей.</p> <p>Таблица учебные группы имеет следующие атрибуты: название группы, год поступления, курс обучения, кол-во студентов в группе.</p> <p>Таблица студенты имеет следующие атрибуты: студента, фамилия, имя, отчество, группа, год рождения, пол, адрес, город, телефон.</p> <p>Таблица ведомости успеваемости имеет следующие атрибуты: группа, студент, предмет, оценка.</p>
<p>Вариант №2</p>	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – информационная система супермаркета. БД состоит из следующих таблиц: отделы, сотрудники, товары, продажа товаров, должности.</p> <p>Таблица отделы имеет следующие атрибуты: название отдела, кол-во прилавков, кол-во продавцов, номер зала.</p> <p>Таблица сотрудники имеет следующие атрибуты: фамилия, имя, отчество, отдел, год рождения, год поступления на работу, стаж, должность, пол, адрес, город, телефон.</p> <p>Таблица должности имеет следующие атрибуты: название должности, сумма ставки.</p> <p>Таблица товары имеет следующие атрибуты: название товара, отдел, страна производитель, условия хранения, сроки хранения .</p> <p>Таблица продажа товаров имеет следующие атрибуты: сотрудника являющегося продавцом, товара дата, время, кол-во, цена, сумма.</p>
<p>Вариант №3</p>	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – информационная система военного округа. БД состоит из следующих таблиц: места дислокации, вид войск, части, роты, личный состав.</p> <p>Таблица вид войск имеет следующие атрибуты: название.</p> <p>Таблица места дислокации имеет следующие атрибуты: страна, город, адрес, занимаемая площадь.</p> <p>Таблица части имеет следующие атрибуты: номер части, место</p>

	<p>дислокации, вид войск, кол-во рот. Таблица роты имеет следующие атрибуты: название роты, кол-во служащих. Таблица личный состав имеет следующие атрибуты: фамилия, рота, должность, год рождения, год поступления на службу, выслуга лет, награды, участие в военных мероприятиях.</p>
<p>Вариант №4</p>	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – информационная система библиотеки. БД состоит из следующих таблиц: библиотеки, фонд библиотеки, тип литературы, сотрудники, пополнение фонда. Таблица библиотеки имеет следующие атрибуты: название, адрес, город. Таблица фонд библиотеки имеет следующие атрибуты: название фонда, библиотека, кол-во книг, кол-во журналов, кол-во газет, кол-во сборников, кол-во диссертаций, кол-во рефератов. Таблица тип литературы имеет следующие атрибуты: название типа. Таблица сотрудники имеет следующие атрибуты: фамилия сотрудника, библиотека, должность, год рождения, год поступления на работу, образование, зарплата. Таблица пополнение фонда имеет следующие атрибуты: фонд, сотрудник, дата, название источника литературы, тип литературы, издательство, дата издания, кол-во экземпляров.</p>
<p>Вариант №5</p>	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – информационная система туристического агентства. БД состоит из следующих таблиц: пансионаты, туры, клиенты, путевки, вид жилья. Таблица пансионаты имеет следующие атрибуты: название пансионата, адрес, город, страна, телефон, описание территории, кол-во комнат, наличие бассейна, наличие медицинских услуг, наличие спа-салона, уровень пансионата, расстояние до моря. Таблица вид жилья имеет следующие атрибуты: название (дом, бунгало, квартира, 1-я комната, 2-я комната и т.д.), категория жилья (люкс, полулюкс, и т.д.), пансионат, описание условий проживания, цена за номер в сутки. Таблица туры имеет следующие атрибуты: название тура (Европа, средняя Азия, тибет и т.д.), вид транспорта, категория жилья на ночь</p>

	<p>(гостиница, отель, палатка и т.д.), вид питания (одноразовое, двухразовое, трехразовое, завтраки), цена тура в сутки.</p> <p>Таблица клиенты имеет следующие атрибуты: фамилия, имя, отчество, паспортные данные, дата рождения, адрес, город, телефон.</p> <p>Таблица путевки имеет следующие атрибуты: клиент, пансионата, вид жилья, дата заезда, дата отъезда, наличие детей, наличие мед. страховки, кол-во человек, цена, сумма.</p>
<p>Вариант №6</p>	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – информационная система автопредприятия города. БД состоит из следующих таблиц: автотранспорт, водители, маршруты, обслуживающий персонал, гаражное хозяйство.</p> <p>Таблица автотранспорт имеет следующие атрибуты: название транспорта (автобусы, такси, маршрутные такси, прочий легковой транспорт, грузовой транспорт и т.д.), кол-во наработки, пробег, кол-во ремонтов, характеристика.</p> <p>Таблица маршруты имеет следующие атрибуты: название маршрута, транспорт, водитель, график работы.</p> <p>Таблица водители имеет следующие атрибуты: фамилия, имя, отчество, год рождения, год поступления на работу, стаж, должность, пол, адрес, город, телефон.</p> <p>Таблица обслуживающий персонал имеет следующие атрибуты: должность (техники, сварщики, слесари, сборщики и др.), фамилия, имя, отчество, год рождения, год поступления на работу, стаж, пол, адрес, город, телефон.</p> <p>Таблица гаражное хозяйство имеет следующие атрибуты: название гаража, транспорт на ремонте, вид ремонта, дата поступления, дата выдачи после ремонта, результат ремонта, персонал, производящего ремонт.</p>
<p>Вариант №7</p>	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – информационная система поликлиники. БД состоит из следующих таблиц: врачи, пациенты, история болезней, отделения, обслуживающий персонал.</p> <p>Таблица отделения имеет следующие атрибуты: название отделения (хирургия, терапия, неврология и т.д.), этаж, номера комнат, ФИО заведующего.</p> <p>Таблица врачи имеет следующие атрибуты: фамилия, имя, отчество,</p>

	<p>должность, стаж работы, научное звание, адрес, номер отделения, в котором он работает.</p> <p>Таблица пациенты имеет следующие атрибуты: фамилия, имя, отчество, адрес, город, возраст, пол.</p> <p>Таблица диагнозы имеет следующие атрибуты: название диагноза, признаки болезни, период лечения, назначения.</p> <p>Таблица история болезни имеет следующие атрибуты: пациент, врач, диагноз, лечение, дата заболевания, дата вылечивания, вид лечения (амбулаторное, стационарное).</p>
<p>Вариант №8</p>	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – информационная система больницы. БД состоит из следующих таблиц: врачи, пациенты, история болезней, операции, лист лечения.</p> <p>Таблица врачи имеет следующие атрибуты: фамилия, имя, отчество, должность, стаж работы, научное звание, адрес.</p> <p>Таблица пациенты имеет следующие атрибуты: фамилия, имя, отчество, адрес, город, возраст, пол.</p> <p>Таблица история болезни имеет следующие атрибуты: пациента врач, диагноз, дата заболевания, дата вылечивания, вид лечения (амбулаторное, стационарное), код операции.</p> <p>Таблица лист лечения имеет следующие атрибуты: дата лечения, история болезни, лекарства, температура, давление, состояние больного (тяжелое, среднее, и т.д.).</p> <p>Таблица операции имеет следующие атрибуты: описание операции (удаление аппендицита, пластическая операция и т.д.), врач, дата операции, пациент, результат операции.</p>
<p>Вариант №9</p>	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – информационная система библиотек города. БД состоит из следующих таблиц: библиотеки, читальные залы, литература, читатели, выдача лит-ры.</p> <p>Таблица библиотеки имеет следующие атрибуты: название, адрес, город.</p> <p>Таблица читальные залы имеет следующие атрибуты: название читального зала, библиотека, кол-во единиц лит-ры, кол-во посадочных мест, время работы, этаж, кол-во сотрудников.</p> <p>Таблица читатели имеет следующие атрибуты: фамилия, имя, отчество, категория читателя, место работы или обучения, возраст, дата регистрации в библиотеке.</p>

	<p>Таблица литература имеет следующие атрибуты: название, категория литературы, авторы, издательство, год издательства, кол-во страниц, читальный зал.</p> <p>Таблица выдача литературы имеет следующие атрибуты: читатель, литература, дата выдачи, срок выдачи, вид выдачи, наличие залога.</p>
Вариант №10	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – информационная система автосалона. БД состоит из следующих таблиц: автомобили, марка автомобиля, сотрудники, продажа автомобилей, покупатель.</p> <p>Таблица марка автомобиля имеет следующие атрибуты: название марки, страна производитель, завод производитель, адрес.</p> <p>Таблица автомобиля имеет следующие атрибуты: название автомобиля, марка, год производства, цвет, категория, цена.</p> <p>Таблица покупателя имеет следующие атрибуты: фамилия, имя, отчество, паспортные данные, адрес, город, возраст, пол.</p> <p>Таблица сотрудника имеет следующие атрибуты: фамилия, имя, отчество, стаж, зарплата.</p> <p>Таблица продажа автомобилей имеет следующие атрибуты: дата, сотрудник, автомобиль, покупатель.</p>
Вариант №11	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – успеваемость студентов кафедры. БД состоит из следующих таблиц: кафедры, дисциплины, преподаватели, студенты, ведомости успеваемости.</p> <p>Таблица кафедра имеет следующие атрибуты: название кафедры, факультет, ФИО заведующего, номер комнаты, номер корпуса, телефон, кол-во преподавателей.</p> <p>Таблица преподаватели имеет следующие атрибуты: фамилия, имя, отчество, кафедра, год рождения, год поступления на работу, стаж, должность, пол, адрес, город, телефон.</p> <p>Таблица студенты имеет следующие атрибуты: фамилия, имя, отчество, кафедра, год рождения, пол, адрес, город, телефон.</p> <p>Таблица дисциплины имеет следующие атрибуты: название дисциплины, кафедра, читаемой эту дисциплину, кол-во часов, вид итогового контроля.</p>

	Таблица ведомости успеваемости имеет следующие атрибуты: преподаватель, дисциплина, студент, оценка.
Вариант №12	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – торговая организация. БД состоит из следующих таблиц: торговая организация, торговая точка, продавцы, поставщики, заказы поставщикам.</p> <p>Таблица торговая организация имеет следующие атрибуты: название торговой организации, адрес, ФИО директора, налоговый номер.</p> <p>Таблица торговая точка имеет следующие атрибуты: название торговой точки, тип (универмаги, магазины, киоски, лотки и т.д.), торговая организация, адрес, ФИО заведующего.</p> <p>Таблица продавцы имеет следующие атрибуты: фамилия, имя, отчество, торговая точка, должность, год рождения, пол, адрес проживания, город.</p> <p>Таблица поставщики имеет следующие атрибуты: название поставщика, тип деятельности, страна, город, адрес.</p> <p>Таблица заказы поставщикам имеет следующие атрибуты: дата заказа, торговая точка, поставщик, название товара, кол-во, цена.</p>
Вариант №13	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – проектная организация. БД состоит из следующих таблиц: отделы, сотрудники, организации, договора, проектные работы.</p> <p>Таблица отделы имеет следующие атрибуты: название отдела, этаж, телефон, начальник отдела.</p> <p>Таблица сотрудники имеет следующие атрибуты: ФИО, должность (конструкторы, инженеры, техники, лаборанты, прочий обслуживающий персонал), номер отдела, в котором работает, пол, адрес, дата рождения.</p> <p>Таблица организации имеет следующие атрибуты: название организации, тип деятельности, страна, город, адрес, ФИО директора.</p> <p>Таблица договора имеет следующие атрибуты: номер договора, дата заключения договора, организация, стоимость договора.</p> <p>Таблица проектные работы имеет следующие атрибуты: дата начала проектной работы, дата завершения проектной работы, номер договора, отдел, осуществляющий разработку.</p>
Вариант	На основании выбранного варианта выполнить следующее:

<p>№14</p>	<ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – информационная система военно-морского флота. БД состоит из следующих таблиц: базы, части, личный состав, корабли, учения.</p> <p>Базы военно-морского флота имеет следующие атрибуты: название базы, географическое расположение, кол-во частей.</p> <p>Таблица части имеет следующие атрибуты: номер части, база флота, место базирования, вид войск (морская авиация, морская пехота и т.д.).</p> <p>Таблица личный состав имеет следующие атрибуты: фамилия, часть, должность, год рождения, год поступления на службу, выслуга лет, награды,</p> <p>Таблица корабли имеет следующие атрибуты: идентификационный номер корабля, название корабля, тип корабля, дата создания, наработка, кол-во посадочных мест, устройство двигателя (парусное, гребное, пароход, теплоход, турбоход, и т.д.), тип привода (самоходное, несамоходное), размещение корпуса (подводная лодка, ныряющее, полупогружное, и т.д.)</p> <p>Таблица учения: часть, корабль, дата учения, место проведения, оценка.</p>
<p>Вариант №15</p>	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – туристическая фирма. БД состоит из следующих таблиц: туристы, туристическая группа, состав групп, гостиницы, ведомости продаж.</p> <p>Таблица туристы имеет следующие атрибуты: ФИО, паспортные данные, пол, возраст, дети.</p> <p>Таблица туры имеет следующие атрибуты: название, страна, города, тип передвижения, тип питания, цена тура, тип проживания.</p> <p>Таблица туристическая группа имеет следующие атрибуты: название, дата отправления, дата прибытия, тур, кол-во туристов.</p> <p>Таблица состав групп имеет следующие атрибуты: дата продажи, турист, группа, цена билета.</p> <p>Таблица гостиницы имеет следующие атрибуты: название гостиницы, страна, город, адрес, кол-во мест, тип гостиницы.</p> <p>Таблица ведомость продаж имеет следующие атрибуты: дата, туристическая группа, гостиница, общая стоимость.</p>
<p>Вариант №16</p>	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации;

	<p>2. Описать основные сущности предметной области;</p> <p>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;</p> <p>4. Построить инфологическую модель базы данных организации;</p> <p>5. Построить даталогическую модель базы данных организации.</p> <p>БД – цирк. БД состоит из следующих таблиц: работники цирка, представления, расписание гастролей, труппа цирка, программа цирка.</p> <p>Таблица работники цирка имеет следующие атрибуты: фамилия, имя, отчество, год рождения, год поступления на работу, стаж, должность (акробат, клоун, гимнаст, музыкант, постановщик, служащий и т.д.), пол, адрес, город, телефон.</p> <p>Таблица представления имеет следующие атрибуты: название, режиссер-постановщик, художник-постановщик, дирижер-постановщик, автор, жанр, тип.</p> <p>Таблица расписание гастролей имеет следующие атрибуты: представление, дата начала, дата окончания, места проведения гастрولي.</p> <p>Таблица труппа представления цирка имеет следующие атрибуты: представление, актер цирка, название роли.</p> <p>Таблица программа цирка имеет следующие атрибуты: представление, дата премьеры, период проведения, дни и время, цена билета.</p>
<p>Вариант №17</p>	<p>На основании выбранного варианта выполнить следующее:</p> <p>1. Выполнить анализ предметной области исследуемой организации;</p> <p>2. Описать основные сущности предметной области;</p> <p>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;</p> <p>4. Построить инфологическую модель базы данных организации;</p> <p>5. Построить даталогическую модель базы данных организации.</p> <p>БД – аптека. БД состоит из следующих таблиц: лекарства, покупатели, продавцы, рецепты, продажа лекарств.</p> <p>Таблица лекарства имеет следующие атрибуты: название, тип (готовое, изготавливаемое), вид (таблетки, мази, настойки), цена.</p> <p>Таблица покупатели имеет следующие атрибуты: фамилия, имя, отчество, адрес, город, телефон.</p> <p>Таблица продавцы имеет следующие атрибуты: фамилия, имя, отчество, дата поступления, дата рождения, образование.</p> <p>Таблица рецепты имеет следующие атрибуты: номер рецепта, дата выдачи, ФИО больного (покупатель), ФИО врача, диагноз пациента.</p> <p>Таблица продажа лекарств имеет следующие атрибуты: дата, лекарство, кол-во, рецепт, продавец.</p>
<p>Вариант №18</p>	<p>На основании выбранного варианта выполнить следующее:</p> <p>1. Выполнить анализ предметной области исследуемой организации;</p> <p>2. Описать основные сущности предметной области;</p> <p>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние</p>

	<p>ключи между сущностями;</p> <p>4. Построить инфологическую модель базы данных организации;</p> <p>5. Построить даталогическую модель базы данных организации.</p> <p>БД – городская телефонная сеть. БД состоит из следующих таблиц: АТС, абонент, ведомость звонков, прайс АТС, ведомость абонентской платы.</p> <p>Таблица АТС имеет следующие атрибуты: название АТС, вид (городские, ведомственные и учрежденческие), адрес, город, кол-во абонентов.</p> <p>Таблица абоненты имеет следующие атрибуты: фамилия, имя, отчество, вид телефона (основной, параллельный или спаренный), номер телефона, межгород (открыт/закрыт), льгота (да/нет), адрес: индекс, район, улица, дом, квартира.</p> <p>Таблица ведомость звонков имеет следующие атрибуты: абонент, дата звонка, время начала, время окончания, межгород (да/нет).</p> <p>Таблица прайс АТС имеет следующие атрибуты: АТС, цена на городские, цена на межгород.</p> <p>Таблица ведомость абонентской платы имеет следующие атрибуты: абонент, месяц, год, кол-во минут на городские, кол-во минут на межгород, стоимость, сумма льготы, общая стоимость.</p>
<p>Вариант №19</p>	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – аэропорт. БД состоит из следующих таблиц: работники аэропорта, расписание вылетов, самолеты, бригады самолетов, ведомость продаж билетов.</p> <p>Таблица работники аэропорта имеет следующие атрибуты: фамилия, имя, отчество, год рождения, год поступления на работу, стаж, должность (пилотов, диспетчеров, техников, кассиров, работников службы безопасности, справочной службы и других,), пол, адрес, город, телефон.</p> <p>Таблица расписание вылетов имеет следующие атрибуты: самолет, дата вылета, время вылета, место выбытия, место прибытия, маршрут (начальный и конечный пункты назначения, пункт пересадки), стоимость билета.</p> <p>Таблица самолеты имеет следующие атрибуты: номер, год выпуска, кол-во посадочных место, грузоподъемность.</p> <p>Таблица бригады самолетов имеет следующие атрибуты: номер бригады, самолет, работник аэропорта (пилоты, техники и обслуживающий персонал)ю</p> <p>Таблица ведомость продажи билетов имеет следующие атрибуты: дата и время продажи, ФИО пассажира, паспортные данные, номер рейса, кол-во билетов, наличие льгот (пенсионеры, дети-сироты и т.д.), багаж (да/нет), стоимость.</p>

<p>Вариант №20</p>	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – театр. БД состоит из следующих таблиц: работники театра, спектакли, расписание гастролей, труппа спектакля, репертуар театра.</p> <p>Таблица работники театра имеет следующие атрибуты: фамилия, имя, отчество, год рождения, год поступления на работу, стаж, должность (актеров, музыкантов, постановщиков и служащих), пол, адрес, город, телефон.</p> <p>Таблица спектакли имеет следующие атрибуты: название, режиссер-постановщик, художник-постановщик, дирижер-постановщик, автор, жанр (музыкальная комедия, трагедия, оперетта и пр), тип (детские, молодежные и пр.).</p> <p>Таблица расписание гастролей имеет следующие атрибуты: название, дата начала, дата окончания, места проведения гастрولي, спектакль.</p> <p>Таблица труппа спектакля имеет следующие атрибуты: спектакль, актер, название роли.</p> <p>Таблица репертуар театра имеет следующие атрибуты: спектакль, дата премьеры, период проведения, дни и время, цена билета.</p>
<p>Вариант №21</p>	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – железнодорожный вокзал. БД состоит из следующих таблиц: работники ж.д.вокзала, расписание движения поездов, поезда, бригады поездов, ведомость продаж билетов.</p> <p>Таблица работники ж.д.вокзала имеет следующие атрибуты: фамилия, имя, отчество, год рождения, год поступления на работу, стаж, должность (машинист, диспетчеров, проводник, ремонтников подвижного состава, путей, кассиров, работников службы подготовки составов, справочной службы и других,), пол, адрес, город, телефон.</p> <p>Таблица расписание движения поездов имеет следующие атрибуты: поезд, дата отправления, время отправления, место отправления, дата прибытия, время прибытия, место прибытия, маршрут ((начальный и конечный пункты назначения, основные узловые станции), стоимость билета.</p> <p>Таблица поезда имеет следующие атрибуты: номер, год выпуска, кол-во вагонов, тип поезда (общий, скоростной, высокоскоростной).</p>

	<p>Таблица бригады поездов имеет следующие атрибуты: номер бригады, поезд, работник ж.д.вокзала (машинисты, техники, проводники и обслуживающий персонал).</p> <p>Таблица ведомость продажи билетов имеет следующие атрибуты: дата и время продажи, ФИО пассажира, паспортные данные, номер рейса, кол-во билетов, наличие льгот (пенсионеры, дети-сироты и т.д.), стоимость.</p>
Вариант №22	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – информационная система ВУЗА. БД состоит из следующих таблиц: факультеты, кафедры, преподаватели, дисциплины, учебная нагрузка.</p> <p>Таблица факультеты имеет следующие атрибуты: название факультета, ФИО декана, номер комнаты, номер корпуса, телефон.</p> <p>Таблица кафедра имеет следующие атрибуты: название кафедры, ФИО заведующего, номер комнаты, номер корпуса, телефон, кол-во преподавателей.</p> <p>Таблица дисциплины имеет следующие атрибуты: название дисциплины, кол-во часов, цикл дисциплин.</p> <p>Таблица преподаватели имеет следующие атрибуты: фамилия, имя, отчество, кафедра, год рождения, год поступления на работу, стаж, должность, пол, город.</p> <p>Таблица учебная нагрузка имеет следующие атрибуты: преподаватель, дисциплина, учебный год, семестр, группы, кол-во студентов, вид итогового контроля.</p>
Вариант №23	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – информационная система военного округа. БД состоит из следующих таблиц: места дислокации, вид войск, части, роты, личный состав.</p> <p>Таблица вид войск имеет следующие атрибуты: название вида войск.</p> <p>Таблица места дислокации имеет следующие атрибуты: страна, город, адрес, занимаемая площадь, кол-во сооружений.</p> <p>Таблица части имеет следующие атрибуты: номер части, место дислокации, вид войск, кол-во рот, кол-во техники, кол-во вооружений.</p> <p>Таблица техника имеет следующие атрибуты: название техники, часть, характеристики.</p> <p>Таблица вооружения имеет следующие атрибуты: название вооружения,</p>

	часть, характеристики.
Вариант №24	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – информационная система супермаркета. БД состоит из следующих таблиц: отделы, клиенты, товары, продажа товаров, поставщики.</p> <p>Таблица отделы имеет следующие атрибуты: название отдела, кол-во прилавков, кол-во продавцов, номер зала.</p> <p>Таблица клиенты имеет следующие атрибуты: название клиента, адрес, вид оплаты.</p> <p>Таблица поставщики имеет следующие атрибуты: название поставщика, адрес, страна, вид транспорта, вид оплаты.</p> <p>Таблица товары имеет следующие атрибуты: название товара, отдел, поставщик, условия хранения, сроки хранения .</p> <p>Таблица продажа товаров имеет следующие атрибуты: клиент, товар, дата, время, кол-во, цена, сумма.</p>
Вариант №25	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> 1. Выполнить анализ предметной области исследуемой организации; 2. Описать основные сущности предметной области; 3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями; 4. Построить инфологическую модель базы данных организации; 5. Построить даталогическую модель базы данных организации. <p>БД – информационная система больницы. БД состоит из следующих таблиц: врачи, пациенты, история болезней, отделения, лист лечения.</p> <p>Таблица отделения имеет следующие атрибуты: название отделения (хирургия, терапия, неврология и т.д.), этаж, номера комнат, ФИО заведующего.</p> <p>Таблица врачи имеет следующие атрибуты: фамилия, имя, отчество, должность, стаж работы, научное звание, адрес.</p> <p>Таблица пациенты имеет следующие атрибуты: фамилия, имя, отчество, адрес, город, возраст, пол.</p> <p>Таблица история болезни имеет следующие атрибуты: пациент, врач, диагноз, дата заболевания, дата вылечивания, вид лечения (амбулаторное, стационарное).</p> <p>Таблица лист лечения имеет следующие атрибуты: дата лечения, история болезни, лекарства, температура, давление, состояние больного (тяжелое, среднее, и т.д.).</p>

Практическая работа № 2 Сбор и анализ информации

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 В таблицы функционально зависит от поля А той же таблицы в том и только в том случае, когда в любой заданный момент времени для каждого из различных значений поля А обязательно существует только одно из различных значений поля В. Отметим, что здесь допускается, что поля А и В 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 так:

$R.X \rightarrow R.Y$

$R(X \rightarrow Y)$.

$\Phi Z(X \rightarrow Y)$

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 В 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 ЗНФ, указать первичные и внешние ключи полученных отношений, построить схему "Таблица-Связь".

$R = \{ \text{НаименованиеЭмитента}, \text{ТипЦБ}, \text{ДатаЭмиссии}, \text{НоминальнаяСтоимость} \}$.

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 привилегированная	20.06.1999	500 руб.
ТОО -Искра	акция привилегированная	23.06.1999	500 руб.

Решение

1. Функциональные зависимости:

$\langle \text{ДатаЭмиссии}, \text{НоминальнаяСтоимость} \rangle \rightarrow \langle \text{НаименованиеЭмитента}, \text{ТипЦБ} \rangle$

$\text{ТипЦБ} \rightarrow \text{НаименованиеЭмитента}$

Первичный ключ отношения R состоит из двух атрибутов:

$\langle \text{ДатаЭмиссии}, \text{НоминальнаяСтоимость} \rangle$.

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

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

$R1 = \{ \text{НаименованиеЭмитента}, \text{ТипЦБ} \}$, где

Функциональные зависимости:

$\text{ТипЦБ} \rightarrow \text{НаименованиеЭмитента}$,

первичный ключ – атрибут ТипЦБ,

$R2 = \{ \text{ТипЦБ}, \text{ДатаЭмиссии}, \text{НоминальнаяСтоимость} \}$,

Функциональные зависимости:

$\langle \text{ДатаЭмиссии}, \text{НоминальнаяСтоимость} \rangle \rightarrow \text{ТипЦБ}$,

- составной первичный ключ:
 <ДатаЭмиссии, НоминальнаяСтоимость>,
 внешний ключ:
 ТипЦБ.
 3. Схема "Таблица-Связь":

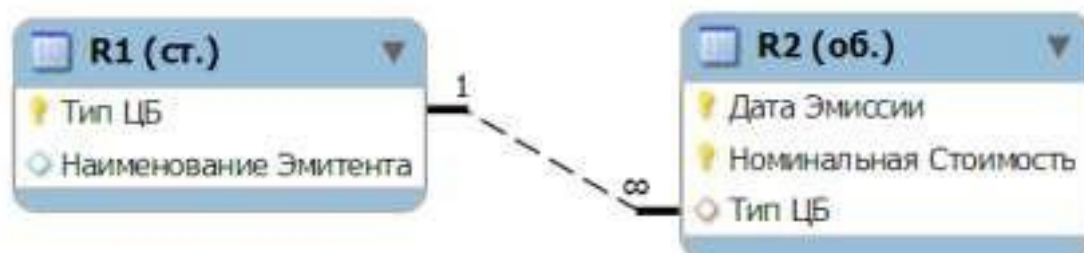


Рисунок 2.1. Схема «Таблица – связь»

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 ₂ N	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	Торг

КГТУ	ЭконФак	Арбуз
СибГТУ	МатФак	Торт
НГУ	ЭконФак	Бананы

Вариант 6.

Наименование Вуза	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.

Наименование Вуза	Команда	Приз
КрасГУ	МатФак	Торт
КрасГУ	ЭконФак	Торт
НГУ	ФилФак	Арбуз
НГУ	МатФак	Бананы

Вариант 8.

Наименование Эмитента	Тип ЦБ	Дата Эмиссии
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.

Наименование Эмитента	Тип ЦБ	Дата Эмиссии
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

Вариант 11.

P1	P2	Дата
A	D	2000
A	E	2000
B	D	2005
C	D	2000

Вариант 12.

A	B	C
A1	B1	C1
A2	B2	C2
A3	B1	C1

A4	B2	C3
----	----	----

Вариант 13.

A	B	C
A1	B1	C1
A1	B1	C2
A2	B2	C2
A3	B2	C3
A4	B3	C1
A4	B3	C3

Вариант 14.

A	B	C
A1	B1	C1
A1	B2	C1
A2	B3	C2
A2	B1	C3

Вариант 15.

A	B	C
A1	B1	C1
A2	B2	C2
A3	B1	C3
A4	B2	C2

Вариант 16.

A	B	C
A1	B1	C1
A1	B1	C2
A2	B2	C3
A3	B2	C4
A4	B3	C5
A4	B3	C6

Вариант 17.

A	B	C
A1	B1	C1
A1	B2	C2
A2	B1	C1
A2	B1	C3

Вариант 18.

P	F	Q
41	21	11
42	22	12
43	21	11
44	22	13

Вариант 19.

C	D	E
71	61	51
71	61	52
72	62	52
73	62	53
74	63	51

74	63	53
----	----	----

Вариант 20.

F	G	H
App	Beer8	Call7
App	Beer2	Call7
Opp	Beer3	Call2
Opp	Beer8	Call3

Вариант 21

НаименованиеВуза	Адрес	Команда
КрасГУ	Красноярск, пр-т Свободный, 79	Первая
КрасГУ	Красноярск, пр-т Свободный, 79	Вторая
КГТУ	Красноярск, ул. Киренского, 26	Первая
КГТУ	Красноярск, ул. Киренского, 26	Вторая
НГУ	Новосибирск, ул. Пирогова, 2	Первая
НГУ	Новосибирск, ул. Пирогова, 2	Вторая

Вариант 22.

H ₂ O	Виртуальность	C ₂ N	Масса
Yes	P	27	45
Yes	P	27	91
No	D	26	NULL
No	D	27	NULL

Вариант 23.

Количество	Свойство	Артикул	Наименование
752	NP	2007	HDR
752	PN	2007	HDR
345	NP	2007	HDR
345	PN	2005	HDR

Вариант 24.

Тип	Признак	Месяц	Век
Второй	Да	10	19
Второй	Нет	10	19
Первый	Да	10	20
Первый	Нет	10	19

Вариант 25.

НаименованиеЭмитента	ТипЦБ	ДатаЭмиссии
ОАО -КамАЗ	акция привилегированная	20.06.2010
ОАО -КрАЗ	акция обыкновенная	23.06.2010
ЗАО -Агат	акция привилегированная	20.06.2010
ТОО -Искра	акция привилегированная	20.06.2010

Вариант 26.

НаименованиеВуза	Команда	Приз
ТГУ	ФилФак	Сист. блок
ТПУ	ЭконФак	Ноутбук
ТГПУ	МатФак	Монитор
ТУСУР	ЭконФак	Ноутбук

Вариант 27.

НаименованиеВуза	Адрес	Команда
ТГУ	Томск, пр. Ленина, 36	МатФак

ТГУ	Томск, пр. Ленина, 36	ГРФак
ТПУ	Томск, пр. Ленина, 2	ГРФак
ТПУ ФТИ	Томск, пр. Ленина, 2	ФизФак
ТУСУР	Томск, пр. Ленина, 40	ФизФак
ТУСУР	Томск, пр. Ленина, 40	МатФак

Вариант 28.

НаименованиеВуза	Команда	Приз
ТГУ	ЭконФак	Торт
ТГУ	МатФак	Торт
ТПУ	МатФак	Бананы
ТПУ	ФилФак	Арбуз

Вариант 29.

НаименованиеЭмитента	ТипЦБ	ДатаЭмиссии
ЗАО -Агат	акция привилегированная	13.05.2009
ЗАО -Сибириада	акция обыкновенная	14.05.2009
ТОО -Искра	акция привилегированная	12.05.2009
ОАО -КрАЗ	акция обыкновенная	14.05.2009

Вариант 30.

Организация	Адрес	Статус
ТРЕНД	Москва, Кирова, 79	РОА
ТРЕНД	Москва, Кирова, 79	ОУП
ЮРАС, первый	Томск, Кирова, 26	ООО
ЮРАС	Томск, Кирова, 26	ОАО
БЕППО	Юрга, Кирова, 2	БПФ
БЕППО	Юрга, Кирова, 2	ПРС

Вариант 31.

Город	Событие	Дата
Калуга	пожар	11.09.1812
Калуга	наводнение	14.09.1813
Вятка	пожар	11.09.1812
Вятка	пожар	15.09.1825

Вариант 32.

В1	В2	В3
post	cool	stop
post	hot	stop
put	cool	stop
get	cool	nonstop

Вариант 33.

А	В	С
14	362	154
509	412	678
648	362	154
3	412	4

Вариант 34.

Риск	Фреш	Серия
Р1	Ф1	С2
Р1	Ф1	С1
Р2	Ф2	С2

P3	Φ2	C3
P4	Φ3	C3
P4	Φ3	C1

Вариант 35.

Тип	Вес	Цена
HDR	512	13500
HDR	432	13500
GTS	256	14000
GTS	512	16000

Вариант 36.

A	B	C
A11	B7	C2
A21	B7	C1
A17	B9	C4
A3	B9	C4

Вариант 37.

A	B	C
111	31	82
101	31	3
45	77	2
45	77	7
92	91	54
92	91	115

Вариант 38.

Red	Blue	Green
F101	567	706
F101	A812	535
C255	567	706
C255	567	536

Вариант 39.

D	F	T
284	17	78
968	17	89
274	33	56
904	33	56

Вариант 40.

AA	YY	ZZ
72	62	52
71	61	52
73	62	53
74	63	53
74	63	51
71	61	51

Вариант 41.

Наличие	Тип	Число	Масса
Yes	FULL	45	90
No	DEPT	54	130
No	DEPT	45	130

Yes	FULL	45	60
-----	------	----	----

Вариант 42.

Вероятность*1000	Время, с	Мощность, Вт	Тип
694	15	2015	HDR
694	17	2015	HDR
604	15	3500	HPQ
604	17	2015	HDR

Вариант 43.

Тип	Наличие	Артикул
Второй	Да	17
Второй	Нет	17
Первый	Да	17
Первый	Нет	17
Второй	Неизвестно	18
Первый	Неизвестно	18

Вариант 44.

A	B	C
17	49	17
35	53	35
24	49	24
19	53	35

Практическая работа №3 Сбор и анализ информации

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 интересоваться 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.

Практическая работа № 4 Сбор и анализ информации

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 её решения должны быть определены содержание БД, эффективный для всех её 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 зависящих 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 создаётся рассматриваемая БД.

2. Предметный подход к проектированию БД

Предметный подход к проектированию БД применяется в тех случаях, когда у разработчиков есть чёткое 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 её отображению в БД с учётом 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 её 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 подчинёнными (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 включают 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 "сущность-сущность", "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 порождённый 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 нём 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 создаётся 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 отчёты. В соответствии с предметной областью система строится с учётом следующих особенностей:

- каждая книга издаётся 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 создаётся 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.

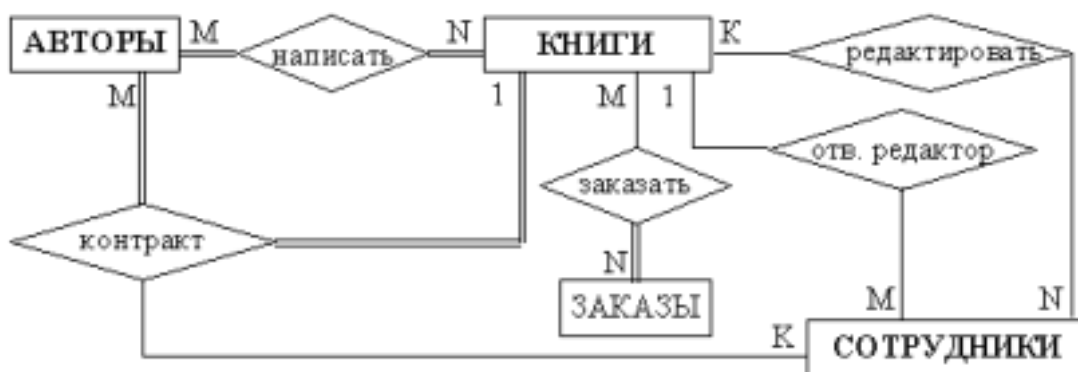


Рисунок 2. ER–диаграмма издательской компании

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;
- 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 отчётности, число часов.
- Преподаватели: паспортные данные, адрес, телефон, фотография, кафедра, должность, учёная 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, счёт/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.

Вариант 3.

Отдел кадров некоторой компании.

- Сотрудники: ФИО, паспортные данные, фотография, дом. и моб. телефоны, отдел, комната, раб. телефоны (в т.ч. местный), подчинённые 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, расчётный счёт в банке, телефон, факс, поставляемое оборудование (материалы), даты поставок (по договорам и реальные), метод и стоимость доставки.
- Сырьё: тип, марка, минимальный запас на складе, время задержки, цена, продукты, при производстве которых используется, потребляемые объёмы (необходимый, реальный, на единицу продукции).

Вариант 5.

Пункт проката видеозаписей (внутренний учёт).

- 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, актёры, 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, широкоэкранный, 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;
- ведение списка клиентов;
- ведение архива законченных дел.
- Необходимо предусмотреть:
- получение списка текущих клиентов для конкретного адвоката;
- определение эффективности защиты (максимальный срок минус полученный срок) с учётом оправданий, условных сроков и штрафов;
- определение неэффективности защиты (полученный срок минус минимальный срок);
- подсчёт суммы гонораров (по отдельным 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;
- учёт 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;
- учёт проводимых соревнований (с ведением их архива);
- учёт 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 каждой оценки .

Практическая работа № 5 Приведение БД к нормальной форме 3НФ

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).

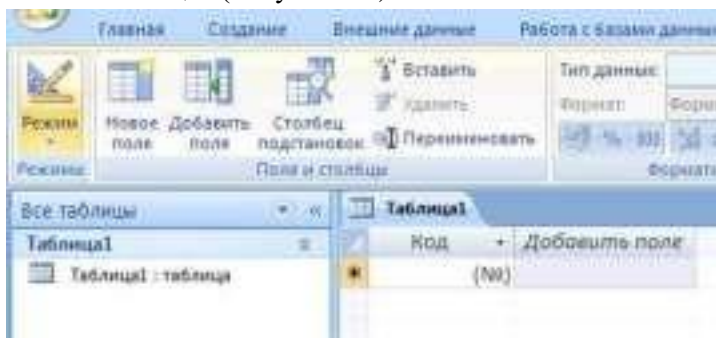





Рисунок 5.1. Окно пустой базы данных

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. 



Имя поля	Тип данных
Код	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Математика	Числовой
Менеджмент	Числовой
Сервисная деятельность	Числовой
Информационные технологии	Числовой
Стандартизация	Числовой
Гостиничная индустрия	Числовой
Пропуски по неуважительной	Числовой
Пропуски по уважительной п	Числовой



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 щелкните 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.

Практическая работа № 6 Приведение БД к нормальной форме 3НФ

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 введите новое.
 - Переименуйте аналогично <Поле 2> на <Преподаватель>.
- 3) Сохраните таблицу с именем <Преподаватели>, щелкнув по кнопке <Сохранить> (дискетка  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*.


предмет	преподаватель
Математика	Бекетова Н.И.
Менеджмент	Казумова Н.С.
Сервисная деятельность	Бессарабова Т.В.
Информационные технологии	Бабич О.А.
Стандартизация	Казарян Г.Г.
Гостиничная индустрия	Казарян Г.Г.

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  в выпадающем 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 полей» и «Каскадное удаление связанных полей». Щелкните по кнопке <Создать>. Появится 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>. Появится связь «один-к-одному».
- 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.

Практическая работа № 7 Приведение БД к нормальной форме 3НФ

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 <ОК>.
- 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 её 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.

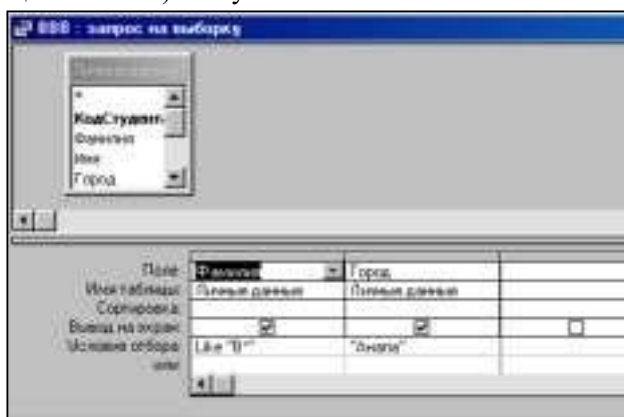


Рисунок 7.1. Запрос на выборку

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 третьей таблицы в третий столбец строки <Поле> - <Предмет> (Рисунок 7.2).

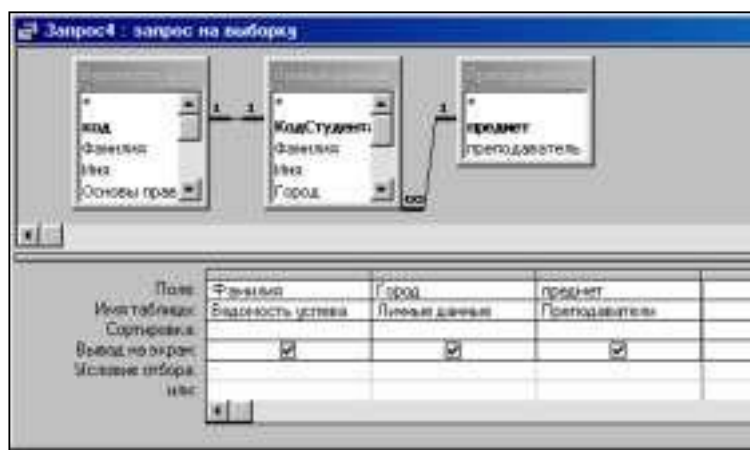


Рисунок 7.2. Запрос на выборку

- В поле <Условия отбора> в столбце <Город> введите город «Новороссийск», в столбце <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 <ОК>.
 - В окне <Создание перекрестных запросов> выделите таблицу <Ведомость успеваемости> и щелкните <Далее>.
 - Выберите поля, значения которого будут использоваться в качестве заголовков 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.
- 8) Завершите работу с Access.

Практическая работа № 8 Проектирование реляционной схемы базы данных в среде СУБД

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>. Появится 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 Aa – <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. Появится 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> (Рисунок 8.1) и щелкните по кнопке <Далее>.

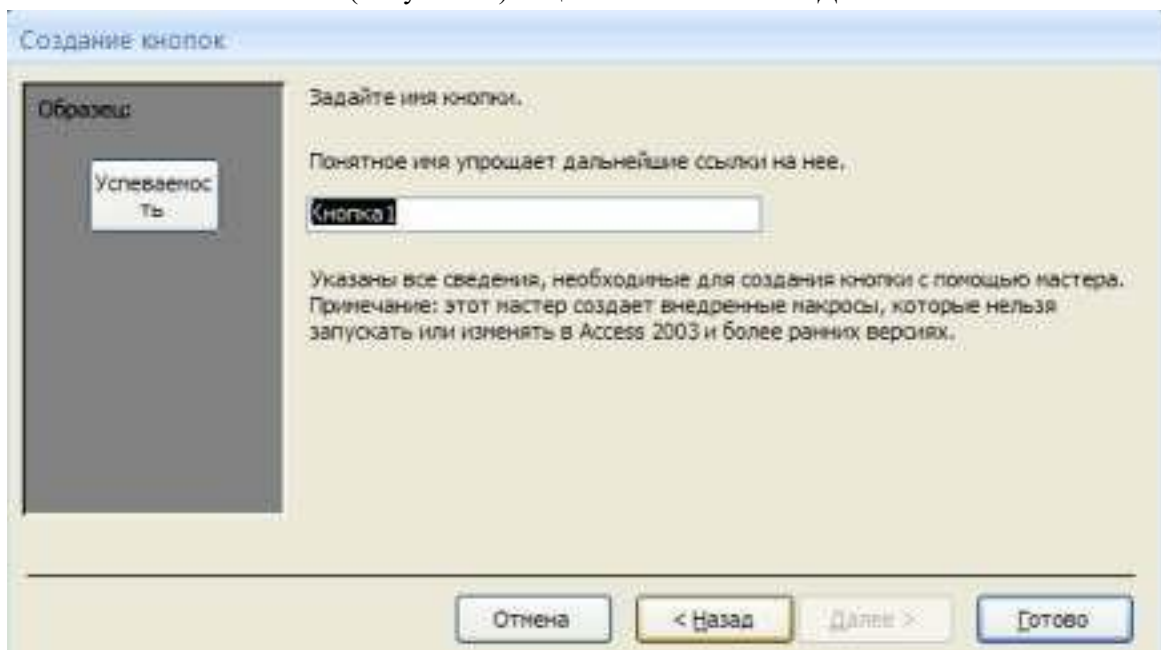


Рисунок 8.1. Создание кнопок

- 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>.

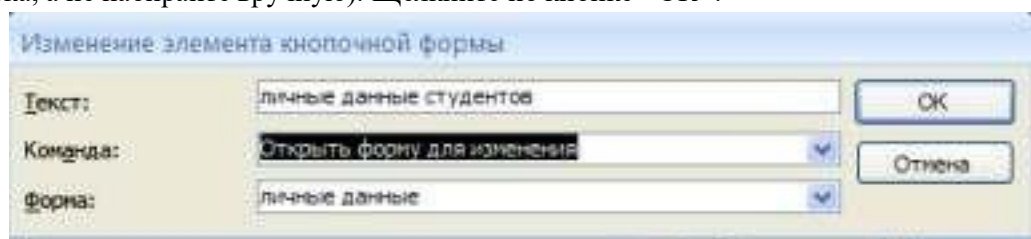


Рисунок 8.4. Изменение элементов кнопочной формы

- 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).

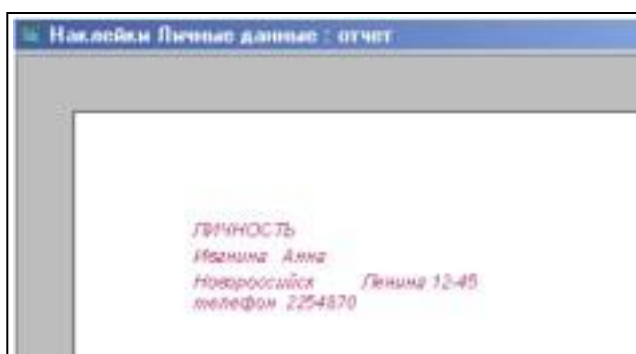


Рисунок 8.5. Отчет

Практическая работа № 9 Проектирование реляционной схемы базы данных в среде СУБД

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 её 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 9.1.

Таблица 9.1. 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 9.2. 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	Открытие БД
CLOSE	закрытие элементов БД (базы, таблиц, отчётов, запросов и т.д.)
QUIT	Выход из Visual FoxPro

Задание для практической работы

Создать в служебной папке Мои документы новую папку и присвоить ей имя, пример: 3курс4группаfoxlab (указывать свой курс и группу)

Запустить программу Microsoft Visual FoxPro:

Пуск/программы/VisualFoxPro

Ознакомиться с элементами рабочего окна программы

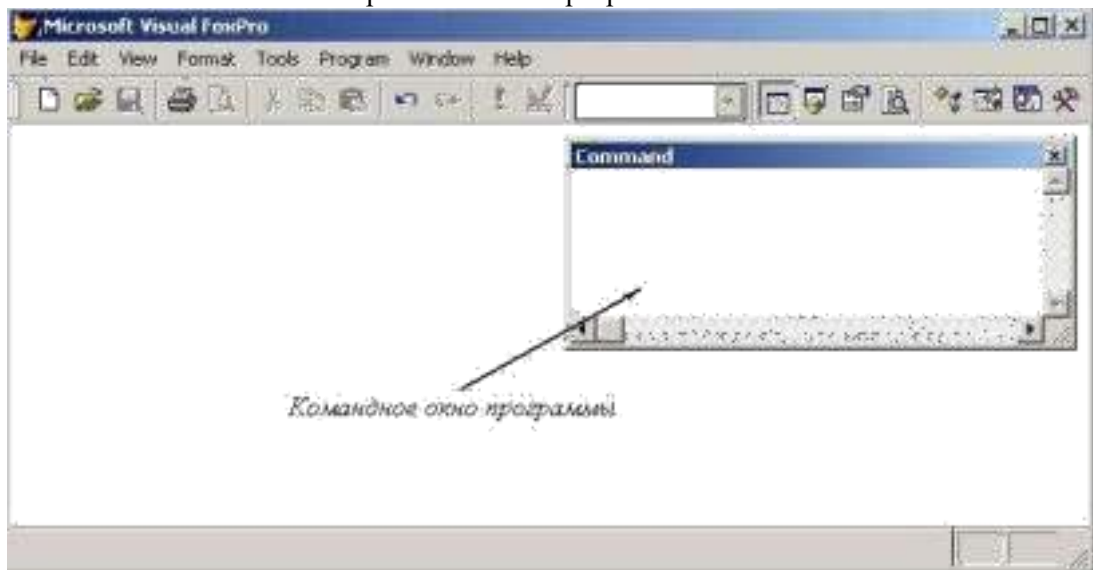


Рисунок 9.1. Главное окно VisualFoxPro 8.0

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 создаётся аналогично: File/New/Database/Newfile, присвоить имя Штат и сохранить. Структура проекта и его элементы отражаются в окне программы ProjectManager (Менеджер проекта)



Рисунок 9.2. Окно Project Manager

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

Создать таблицу в БД штат: в окне ProjectManager (см. рис.9.2.) щёлкнуть 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.

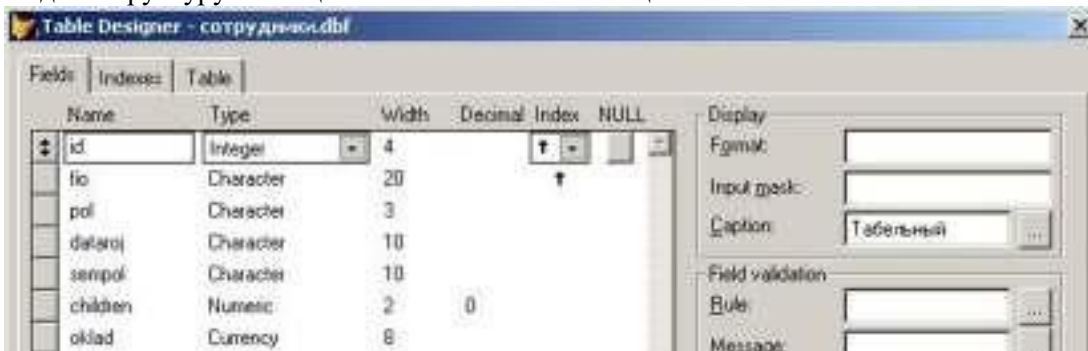


Рисунок 9.3. Окно конструктора таблицы TableDesigner вкладка Fields

Таблица 9.3.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 9.4. 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.9.1.) 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.

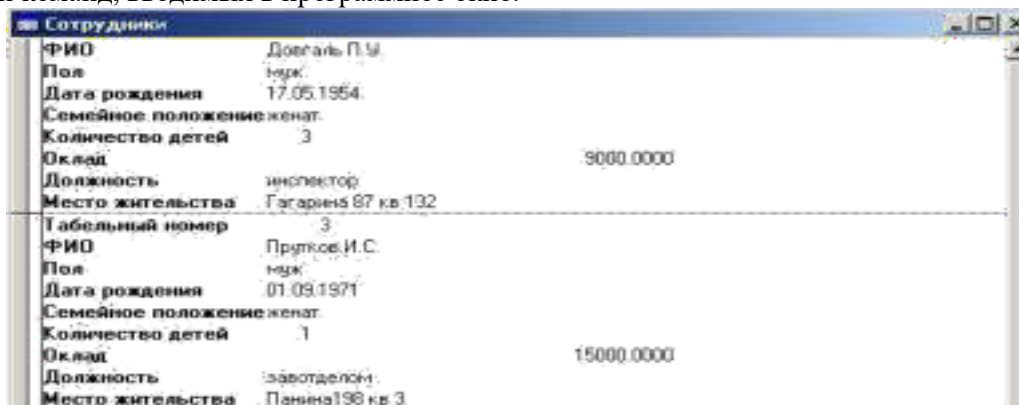


Рисунок 9.4. Просмотр и редактирование содержимого таблицы в режиме APPEND

После ввода данных (см. таблицу 9.4.) в командном окне программы введите команду BROWSE и нажмите клавишу Enter. Результат должен соответствовать рис.9.5.

ФИО	Пол	Дата рождения	Семейное положение	Количество детей	Оклад	Должность
Гусков Г.Г.	муж	22.02.1975	холост	0	7000.0000	экономист
Довгале П.М.	муж	17.05.1954	женат	3	9000.0000	инспектор
Прутков И.С.	муж	01.09.1971	женат	1	15000.0000	заводской
Косыгин И.И.	муж	28.03.1979	холост	0	7000.0000	экономист
Патрakov Е.П.	муж	14.06.1980	холост	0	4500.0000	инженер
Белкина Е.Ю.	жен	24.11.1959	замужем	4	12000.0000	бухгалтер
Петренко Е.П.	жен	29.09.1981	незамужем	0	7000.0000	экономист
Петросова Г.Н.	жен	19.06.1976	замужем	2	4500.0000	инженер
Орлова К.Д.	жен	31.12.1977	незамужем	0	7000.0000	экономист
Косыгина А.В.	жен	01.10.1969	замужем	1	8000.0000	инженер

Рисунок 9.5. Просмотр таблицы в режиме BROWSE

Таблица 9.4. Содержимое таблицы Сотрудники

Лин. 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	Липнева 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	Паньина 198 кв 3
4	Косыгина Н.И.	муж	28.03.1979	холост	0	7000	экономист	Николаева 30 кв 77
5	Петренко в.Е.П.	муж	14.08.1980	холост	0	4500	консультант	Нуритдилова 18г кв 84
6	Белькина Б.Ю.	жен	24.11.1959	замужем	4	12000	бухгалтер	Луначевского 64 кв 9
7	Петренко о.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	Паньина 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 её сначала необходимо пометить на удаление. Для этого щелкнуть клавишей мыши в ячейке слева от удаляемой записи в области узкого, непоименованного столбца, ячейка станет помеченной черным 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 9.10. Описание типов индекса

Тип индекса	Описание
Regular (Обычный)	Создается индекс, в котором для каждой записи таблицы хранится значение индексного выражения. Если несколько записей имеют одинаковое значение индексного выражения, то каждое значение хранится отдельно и содержит ссылку на связанную с ней запись
Unique (Уникальный)	Создается 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)	Создается 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)	Создается 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, отчетов и запросов используются несколько таблиц, между которыми установлены постоянные отношения. Такие таблицы называются связанными. Из двух связанных таблиц одна является главной (родительской), а другая подчиненной (дочерней). При создании индексов для родительской таблицы должен быть определен ключ типа Primary (Первичный) или типа Candidate (Кандидат), а для дочерней таблицы – индекс для связи с родительской таблицей типа Regular (Обычный).

Таблица 9.11. 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 распространенный 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.9.3).

Таблица 9.12. 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 9.13. 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 удалить 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 удалить записи в родительской таблице независимо от существования связанных записей в дочерней таблице. Целостность данных при этом не поддерживается

Таблица 9.14. 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

Индивидуальные задания к практической работе

Создать многотабличную БД и присвоить имя Библиотека. Создать следующие таблицы:

1. Таблица Книги

(наименование поля)	(тип поля)	(ширина поля)	(индексное поле)	(надпись, определяет заголовок поля)
Name	Type	Width	Index	Caption
Kodknigi	Integer	4		Код книги
nazvanie	Character	40		Название
razdel	Character	15		Раздел
izdat	Character	20		Издательство
godizdan	Numeric	4		Год издания
mestohran	Character	5		Местонахождение

Определение полей таблицы Книги в окне конструктора таблицы TableDesigner
Содержимое таблицы Книги

Код	Название	Раздел	Издательство	Год	Место
книги				издания	хранения
1	Практический курс программирования	Информатика	Наука	1983	6-11
2	TURBOPASCAL для школьников	Информатика	Финансы и статистика	1999	6-22
3	Занимательная математика	Математика	Тригон	1998	3-14
4	HTML в действии	Информатика	Питер	1997	5-4
5	Национальное счетоводство	Экономика	Финансы и статистика	1998	4-11
6	Самоучитель VisualFoxPro 8.0	Информатика	БХВ-Петербург	2003	5-34
7	Язык телодвижений	Психология	Наука	1984	2-17
8	Теория машин	Техника	Машиностроение	1957	3-15
9	Теория гипноза	Психология	Наука	1999	2-31
10	Карьера менеджера	Экономика	Парадокс	1998	1-212

2. Таблица Разделы

(наименование поля)	(тип поля)	(ширина поля)	(индексное поле)	(надпись, определяет заголовок поля)
Name	Type	Width	Index	Caption
razdel	Character	15		Раздел

Определение полей таблицы Разделы в окне конструктора таблицы TableDesigner
Содержимое таблицы Разделы

Раздел
Экономика
Информатика
Психология
Математика
Техника

3. Таблица *Издательство*

Определение полей таблицы *Издательство* в окне конструктора таблицы **TableDesigner**

(назначение поля)	(тип поля)	(ширина поля)	(индексное поле)	(название, определяет заголовок поля)
Name	Type	Width	Index	Caption
izdat	Character	20		Издательство
gorod	Character	15		Город

Содержимое таблицы *Издательство*

Издательство	Город
Финансы и статистика	Москва
Тригон	Санкт-Петербург
Питер	Санкт-Петербург
Наука	Москва
Машиностроение	Москва
Парадиз	Минск
БХВ-Петербург	Санкт-Петербург

4. Таблица *Автор_книги*

Определение полей таблицы *Автор_книги* в окне конструктора таблицы **TableDesigner**

(назначение поля)	(тип поля)	(ширина поля)	(индексное поле)	(название, определяет заголовок поля)
Name	Type	Width	Index	Caption
kodevknigi	Integer	4		Код автора книги
kodeknigi	Integer	4		Код книги
kodeavtora	Integer	4		Код автора

Содержимое таблицы *Автор_книги*

Код автора книги	Код книги	Код автора
1	1	1
2	1	2
3	2	3
4	3	4
5	4	5
6	5	6
7	6	7
8	7	8
9	8	9
10	9	10
11	10	11

5. Таблица *Авторы*

Определение полей таблицы *Авторы* в окне конструктора таблицы **TableDesigner**

(наименование поля)	(тип поля)	(ширина поля)	(индексное поле)	(надпись, определяет заголовок поля)
Name	Type	Width	Index	Caption
kodeavtora	Integer	4		Код автора
famaliya	Character	20		Фамилия
imiya	Character	15		Имя

Содержимое таблицы *Авторы*

Код автора	Фамилия	Имя
1	Фролов	Геннадий
2	Илюхин	Виктор
3	Попов	Владимир
4	Ахмедова	Светлана
5	Морис	Брюс
6	Кулагина	Галина
7	Омельченко	Людмила
8	Алан	Пиз
9	Мальцев	Анатолий
10	Горин	Дмитрий
11	Яковла	Ли

6. Открыть БД штат: в окне ProjectManager (Менеджер проекта) вкладка Data/Databases/Библиотека/Open

7. Необходимо модифицировать таблицы БД Библиотека, создать индексированные поля, а также первичные ключи для осуществления связей между таблицами. Для модификации (изменения структуры) таблицы в окне проекта ProjectManager (Менеджер проекта) установить курсор на модифицируемую таблицу Книги и нажать кнопку Modify (Модифицировать) или в окне конструктора БД DatabaseDesigner (Конструктор базы данных) установить курсор в таблицу Книги, вызвать контекстное меню и выбрать команду Modify (Модифицировать). На экране откроется диалоговое окно TableDesigner (Конструктор таблицы)

8. В окне TableDesigner (Конструктор таблицы) перейти на вкладку Indexes (Индексы) и ввести значения в соответствии с таблицей 6., нажать кнопку ОК, система попросит подтвердить сохранение изменений, нажать Yes

Таблица 9.15. Определение индексов таблицы Книги на вкладке Indexes (Индексы) окна конструктора таблицы TableDesigner

Order	Name	Type	Expression	Filter	Collate
1	kodeknigi	Primary	kodeknigi		Machine
2	razdel	Regular	razdel		Machine
3	izdat	Regular	izdat		Machine



Рисунок 9.5. Вкладка Indexes (Индексы) окна TableDesigner таблицы Книги
 Индекс, с присвоенным типом Primary является первичным ключом таблицы
 9. Аналогично внести изменения в таблицы БД Библиотека: Разделы,
 Издательство, Автор_книги, Авторы в соответствии с таблицами 9.16,9.17, 9.18, 9.19
 Таблица 9.16.Определение индексов таблицы Разделы на вкладке Indexes (Индексы) окна
 конструктора таблицы TableDesigner

Order	Name	Type	Expression	Filter	Collate
1	razdel	Primary	razdel		Machine

Таблица 9.17.Определение индексов таблицы Издательство на вкладке Indexes (Индексы)
 окна конструктора таблицы TableDesigner

Order	Name	Type	Expression	Filter	Collate
1	izdat	Primary	izdat		Machine

Таблица 9.18.Определение индексов таблицы Автор_книги на вкладке Indexes (Индексы)
 окна конструктора таблицы TableDesigner

Order	Name	Type	Expression	Filter	Collate
1	kodavknigi	Primary	kodavknigi		Machine
2	kodavtora	Regular	kodavtora		
3	kodknigi	Regular	kodknigi		

Таблица 10. Определение индексов таблицы Авторы на вкладке Indexes (Индексы) окна
 конструктора таблицы TableDesigner

Order	Name	Type	Expression	Filter	Collate
1	kodavtora	Primary	kodavtora		Machine

10. В окне DatabaseDesigner (Конструктор базы данных) выбрать родительскую таблицу Разделы. Таблицы в конструкторе БД обозначаются прямоугольниками, в нижней части которых после надписи Indexes (Индексы) расположен список индексов, созданных для данной таблицы. Первичный ключ в этом списке выделяется значком ключа, расположенным с левой стороны от наименования индекса. Установить курсор мыши на первичный ключ razdel таблицы Разделы. Нажать кнопку мыши и, не отпуская ее, переместить курсор мыши на индекс razdel дочерней таблицы Книги. Отпустить кнопку мыши. Созданные отношения между таблицами отображаются в виде линий. Аналогичным образом, создать связь между таблицами Издательство и Книги (Таблица Издательство – родительская, таблица Книги – дочерняя, первичный ключ – izdat)

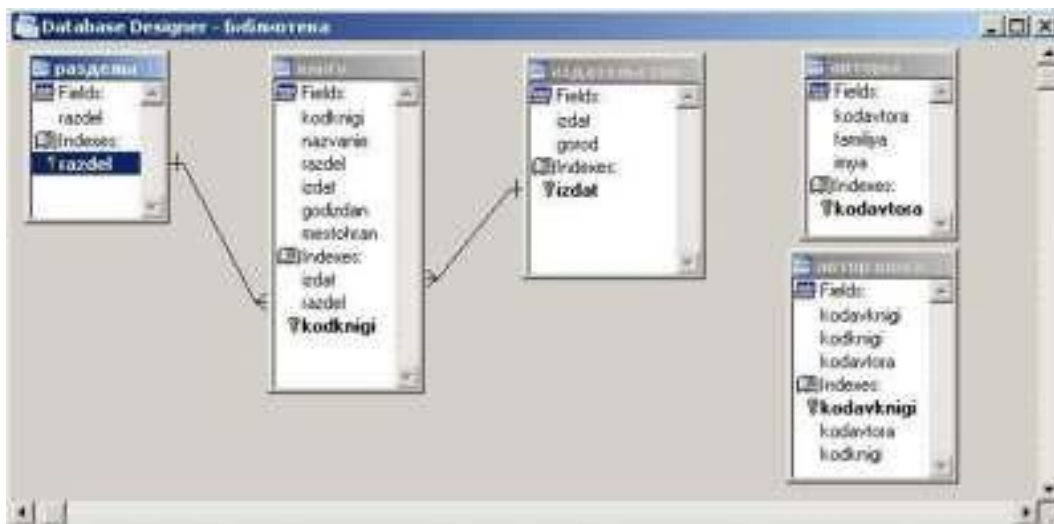


Рисунок 9.6. Отображение связей между таблицами в окне DatabaseDesigner (Конструктор базы данных)

11. Для редактирования отношений между связанными таблицами дважды щелкнуть левой кнопкой мыши на линии, появится диалоговое окно EditRelationship(Редактирование отношения), в котором слева приведено наименование родительской таблицы и расположен раскрывающийся список индексов таблицы, справа размещена аналогичная информация о дочерней таблице. В этом диалоговом окне указан также тип установленного отношения между таблицами.

9. В окне EditRelationship(Редактирование отношения) нажать кнопку ReferentialIntegrity(Целостность данных), в появившемся диалоговом окне ReferentialIntegrityBuilder (Построитель целостности данных) выбрать отношение издательство-книги. В полях Update (Изменить), Delete (Удалить), Insert (Заменить) установить тип действий Restrict (Запрет изменения). Провести аналогичные действия для отношения разделы-книги. Результат описанных действий, которые необходимы для обеспечения целостности данных, представлен на рис.9.7.

10. Для сохранения выполненных действий нажать кнопку ОК, система потребует подтверждение сохранения, нажать кнопку Да

11. Нажать кнопку ОК для закрытия диалогового окна EditRelationship



Рисунок 9.7. Диалоговое окно ReferentialIntegrityBuilder(Построитель целостности данных)

12. Создать связь между таблицами Книги и Автор_книги (kodknigi)

13. Создать связь между таблицами Авторы и Автор_книги (kodavtora)

14. Установить целостность данных в созданных отношениях: в полях Update (Изменить), Delete (Удалить), Insert (Заменить) установить тип действий Restrict (Запрет изменения)

Практическая работа №10. Проектирование реляционной схемы базы данных в среде СУБД

Цель работы: овладение практическими навыками по созданию форм

Формы в VisualFoxPro

В VisualFoxPro для просмотра, ввода и редактирования данных, хранящихся в таблицах, используются формы, являющиеся более наглядным средством представления информации. Важным преимуществом форм является, то, что они позволяют работать не с одной, а с несколькими связанными таблицами, что, в свою очередь, также увеличивает наглядность.

При создании форм в VisualFoxPro разработчик может использовать следующие средства: FormWizard (мастер форм), FormBuilder (построитель формы), Builder (построитель объектов формы), AutoFormatBuilder (построитель автоформата), FormDesigner (конструктор форм).

Чтобы создать форму для одной или связанных таблиц с возможностью задания отображаемых в форме полей, стиля их отображения и указания типа кнопок управления, можно использовать FormWizard (мастер создания форм).

Для самостоятельной разработки формы с заданными свойствами или изменения формы, созданной с помощью мастера, вам необходимо использовать FormDesigner (конструктор форм).

Для облегчения размещения в конструкторе форм полей и надписей, оформленных в соответствии с выбранным стилем, можно использовать FormBuilder (Построитель формы).

Создание формы с помощью конструктора форм

Любая форма в VisualFoxPro состоит из объектов, каждый из которых имеет характерные свойства. Для любого объекта вы можете указать действия, выполняемые написанной разработчиком программой при наступлении определённых событий. Процесс создания формы в конструкторе форм состоит в размещении в форме объектов и определении свойств, а также связанных с ними событий и выполняемых действий.

Процесс создания формы включает следующие действия:

- настройка параметров формы
- определение среды окружения, то есть выбор используемых в форме таблиц и установка связей между ними
- размещение в форме объектов: текста, полей различных типов, линий, рисунков, кнопок управления.

Задание для практической работы

Задание 1. Создание формы средствами мастера форм

Запустить программу Microsoft VisualFoxPro

Открыть созданный проект: File/Open/Информационная система

Открыть БД штат: в окне ProjectManager (Менеджер проекта) вкладка Data/Databases/штат/Open

Создать форму Сотрудники, щелкнув клавишей мыши на вкладке

Documents/Forms/New/FormWizard/FormWizard, нажать клавишу Ok



Рисунок 10.1. Диалоговое окно WizardSelection

В появившемся окне FormWizard в области Databasesandtables (Базы данных и таблицы) выбрать необходимую БД штат и указать таблицу Сотрудники, для которой создается форма. Из области

Availablefields (Имеющиеся поля) выбрать поля, которые будут размещены в форме, в соответствии с рис.10.2, используя кнопки расположенные между списками для перехода к следующему шагу нажать кнопку Next(Далее)



Рисунок 10.2. Диалоговое окно выбора полей для отображения.

В появившемся диалоговом окне в области Style установить стиль отображения объектов и в области Button type типы кнопок управления в соответствии с рис.10.3, нажать кнопку Next

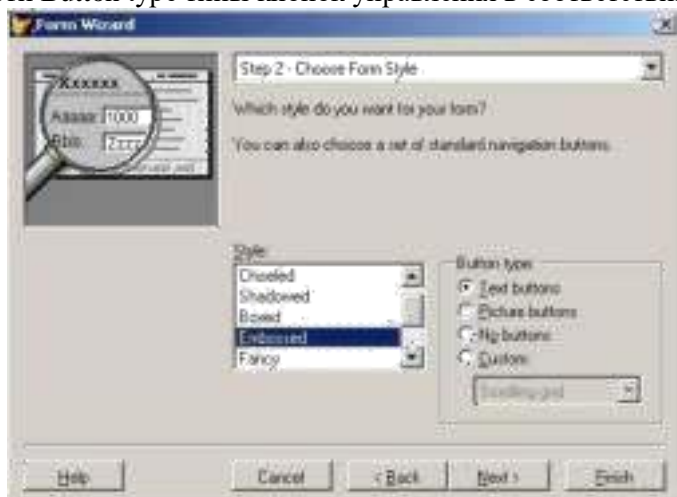


Рисунок 10.3. Окно выбора стиля отображения полей и управляющих кнопок.

В появившемся диалоговом окне задать критерий сортировки данных, указав поля по которым будет осуществляться упорядочивание в соответствии с рис.10.4. нажать кнопку Next



Рисунок 10.4. Установка критерия упорядочения данных

На заключительном шаге создания форм в области Typeatitleforyourform(Тип заголовка формы) указать имя формы

Сотрудники. Воспользовавшись кнопкой Preview (Просмотр) просмотрите, как будет выглядеть создаваемая форма, после просмотра нажмите кнопку ReturntoWizard.

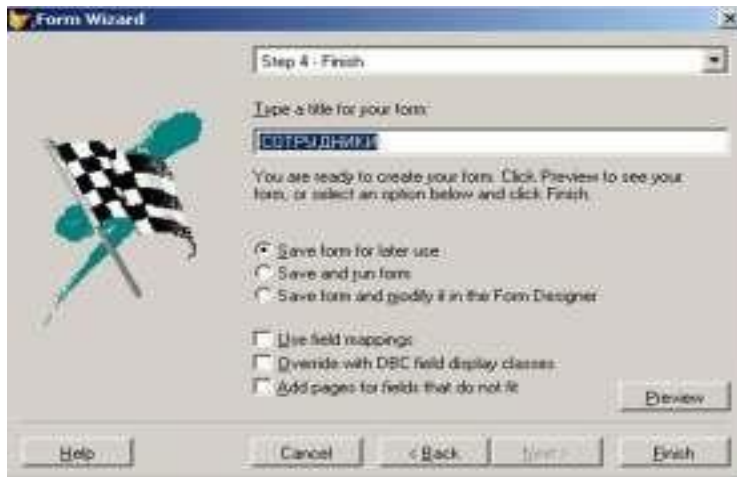


Рисунок 10.5. Окно задания заголовка формы.

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 размещена форма Сотрудники

Запустить в окне ProjectManager форму Сотрудники:
Documents/Forms/Сотрудники/Run



Рисунок 10.6. Окно ProjectManager

Или с помощью меню программы VisualFoxPro: Program/Дозадать имя формы Сотрудники и тип файла Form, нажать кнопку Do. С помощью кнопок (см. таблицу 10.1.), находящихся в нижней части формы можно вводить или редактировать записи в таблице сотрудники.

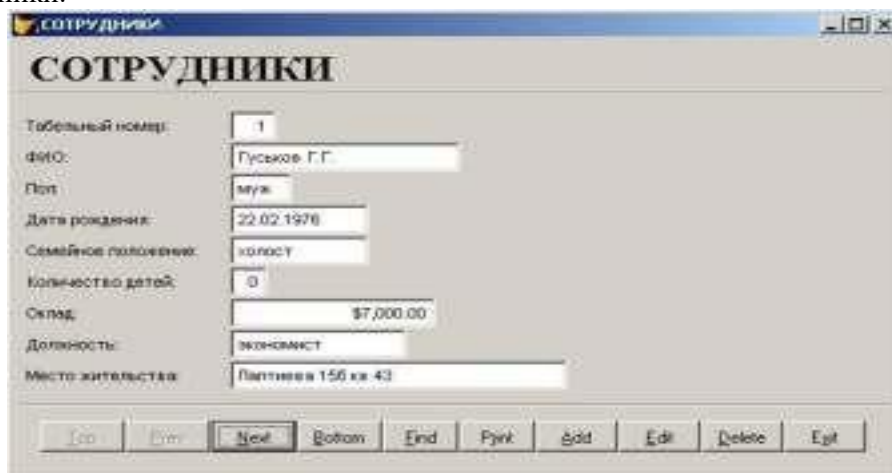


Рисунок 10.7. Форма Сотрудники

Ввести в таблицу Сотрудники дополнительно шесть записей, одну удалить, две отредактировать

Таблица 10.1. Кнопки управления и редактирования данных

Top	Prev	Next	Bottom	Find	Save
Первая запись	Предыдущая запись	Следующая запись	Последняя запись	Найти	Сохранить
Print	Add	Edit	Delete	Exit	Revert
Вывести на печать	Добавить запись	Редактировать запись	Удалить запись	Выход	Отменить

Индивидуальные задания к практической работе

Создать форму для таблицы Книги БД Библиотека

1. В диалоговом окне выбора полей для отображения выбрать все поля таблицы Книги
2. В окне выбора стиля отображения полей и управляющих кнопок (см. рис.3.) в области Style (стиль) выбрать стиль Embossed, а в области Buttontype (Типы кнопок управления) выбрать пункт NoButtons
3. Установить критерий упорядочения данных по полю kodknigi
4. В окне задания заголовка формы в области Typeatitleforyourform(Тип заголовка формы) указать имя формы Книги. Просмотреть вид создаваемой формы нажатием кнопки Preview (Просмотр).
5. После сохранения запустить форму Книги, она должна иметь вид как на рис.10.8.

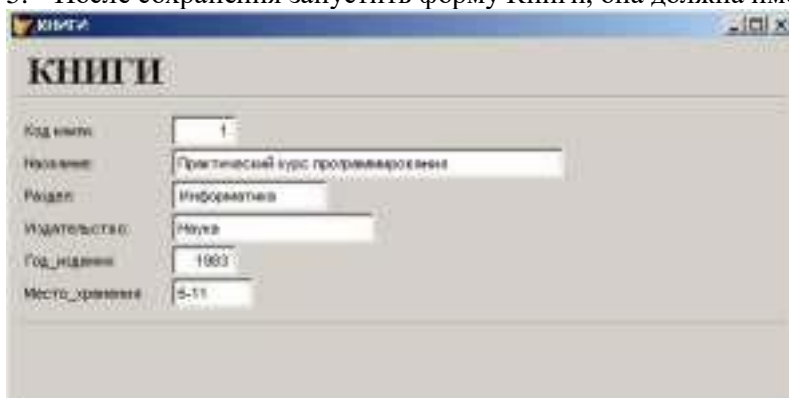


Рисунок 10.8. Форма Книги

Задание 2. Создание формы средствами конструктора форм

Запустить программу Microsoft VisualFoxPro

Открыть проект Информационная система

Открыть БД штат Выбрать в окне Project Manager вкладку Documents/Forms/New/New Form. Откроется окно FormDesigner(Конструктор форм) для создания новой формы

Открыть окно окружения формы DataEnvironment (Среда окружения): пункт меню View/ DataEnvironment. Для размещения таблицы в среде окружения выбрать команду Add(Добавить) из меню DataEnvironment (Среда окружения) или вызвать контекстное меню на окне окружения формы DataEnvironmentи выбрать пункт Add. В появившемся диалоговом окне AddTableorView (Добавить таблицу или представление данных), выбрать из списка таблиц открытой базы штат таблицу Сотрудники, для которой создаётся форма, и нажать кнопку ОК

Открыть окно свойств таблицы, размещённой в окне окруженияDataEnvironment. Для этого установить на таблицу Сотрудники курсор, вызвать контекстное меню правой клавишей мыши и выбрать команду Properties (Свойства)

В окне Properties (свойства) выделить свойство Order (Упорядочение). Для упорядочения данных в форме в поле коррекции свойства нажмите кнопку раскрытия списка и из списка индексов таблицы выберите индекс (id), по которому хотите упорядочить данные.

Закройте окно определения среды окружения DataEnvironment.

Для задания свойств формы установить курсор в форму, вызвать контекстное меню, выбрать команду Properties (Свойства). Откроется окно Properties(Свойства). В его верхнем списке, указывающем название объекта, для которого осуществляется настройка свойств содержится текст Form1 (Форма1)

В окне Properties (Свойства) скорректировать свойство Caption (Надпись), введя в текстовое поле заголовок формыСотрудники тест

Свойство формы AutoCenter (Автоцентр) должно иметь значение True(Истина), чтобы форма располагалась в центре экрана

Изменить свойства FontName (Наименование шрифта), указав шрифт TimesNewRoman и FontSize (Размер шрифта) указав размер шрифта 12 После того как вы определили параметры формы, разместили в окружении используемые таблицы, можно приступить к размещению объектов в форме. Осуществим размещение полей таблицы Сотрудники и надписей к ним, используя FormBuilder (построитель формы)

Для запуска построителя форм установить курсор в форму, вызвать контекстное меню, выбрать команду Builder (Построитель), откроется диалоговое окноFormBuilder (построитель формы), содержащее две вкладки: FieldSelection (Выбор поля) и Styles (Стиль). В вкладке FieldSelection (Выбор поля) из области Databasesandtables (Базы данных и таблицы) выбрать БД штат и таблицу Сотрудники, затем из списка Availablefields (Имеющиеся поля) перенести в Selectedfields(Выбранные поля) все необходимые поля используя кнопки расположенные между списками. Перейти на вкладку Styles (Стиль), выбрать из списка стилей стиль Embossed и нажать кнопку ОК. На форме будут размещены объекты формы (см. рис. 10.9).

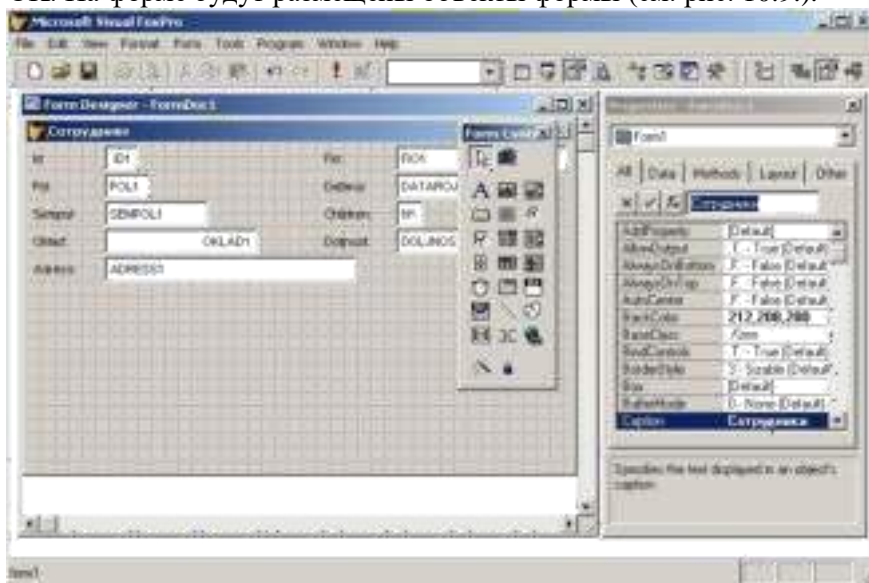


Рисунок 10.9. Окно FormDesigner(Конструктор форм)

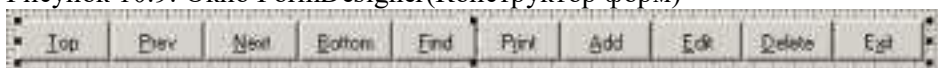


Рисунок 10.10. Объект - блок кнопок управления

Для размещения на создаваемой форме кнопок управления воспользуемся ранее созданной формой Сотрудники. Открыть форму Сотрудники в режиме редактирования: окно программы ProjectManager/Documents/Forms/сотрудники/Modify. В появившемся окне FormDesigner формы Сотрудники нажатием левой клавиши мыши выделить объект BUTTONSET1 (блок кнопок управления, см. рис.10.10.) и нажать комбинацию клавиш Ctrl+Sна клавиатуре или установить курсор в форму, вызвать контекстное меню, выбрать команду Copy. Закрыть окно FormDesigner формы Сотрудники.

В окне FormDesigner формы Сотрудники тест установить курсор в форму, вызвать контекстное меню, выбрать команду Paste или нажать комбинацию клавиш Ctrl+V. Блок кнопок управления записями таблицы (см. рис.10.10) в создаваемой форме будет скопирован.

Задать цвет фона формы. Выделить в окнеProperties (Свойства) свойство формы BackColor (цвет фона), нажать расположенную с правой стороны поля редактирования свойства кнопку и в открывшемся диалоговом окне Цвет выберите цвет, который вы хотите использовать для фона

Расположить объекты формы в соответствии с рис.10.11. Для перемещения объектов формы можно использовать метод перетаскивания мышью или кнопками управления курсором на клавиатуре, предварительно выделив необходимый объект, а также комбинацией клавиш Ctrl+[кнопки управления курсором], использовать все выше перечисленные способы, выяснить их различия. Для изменения размеров формы или объектов формы необходимо выделить объект (форму), подвести курсор к углу объекта (формы), когда курсор примет вид разнонаправленной стрелки нажать левую клавишу мыши и удерживая её, изменить размеры объекта (формы)

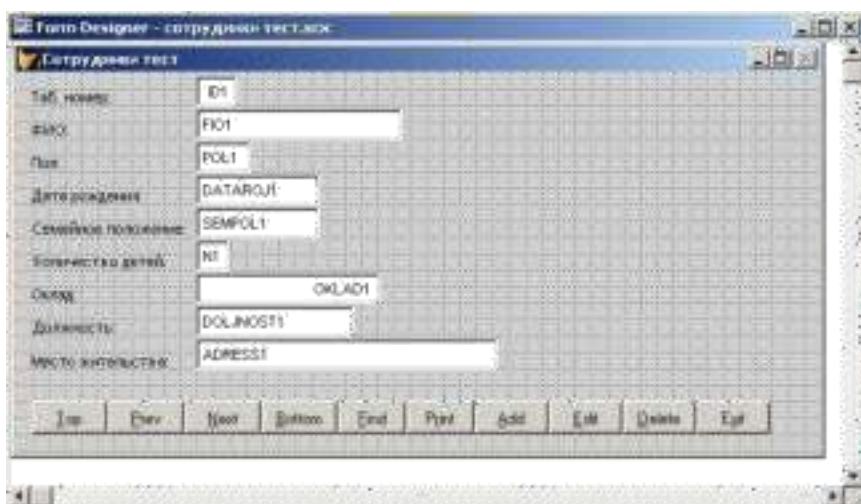



Рисунок 10.11 Вид отредактированной формы

Запустить отредактированную форму: установить курсор в форму, вызвать контекстное меню, выбрать команду RunForm или нажать кнопку [] на панели инструментов. Система попросит сохранить созданную форму, нажать кнопку Yes, в появившемся окне указать папку и имя файла (Сотрудники тест), сохранить.

Индивидуальные задания к практической работе

Создать кнопки навигации в форме Книги

Открыть форму Книги в окне FormDesigner (Конструктор форм)

Нажать кнопку CommandGroup (Группа кнопок)  на панели инструментов FormControls (элементы управления формы) и щёлкнуть в месте их предполагаемого размещения в форме

В Окне Properties (свойства) размещённого объекта выделить свойство ButtonCount (Количество кнопок) и указать необходимое количество размещаемых в форме кнопок, указав число 5

Увеличить с помощью мыши размеры рамки, окружающей данный объект. Перейти в режим редактирования: установить на объект курсор, вызвать контекстное меню и выбрать команду Edit (Редактировать). Выделяя поочередно кнопки, переместить их, расположив горизонтально, в одну линию

Выйти из режима редактирования, щёлкнув вне области объекта

CommandGroup (Группа кнопок)

Скорректировать размер рамки, окружающей объект: в свойстве AutoSize (Авторамер) установить значение True (Истина)

Открыть окно свойств объекта CommandGroup (Группа кнопок), нажать кнопку раскрытия списка в верхней части данного окна, поочередно выбрать из списка элементы Command1, Command2, Command3, Command4, Command5 и, используя свойство

Caption(Надпись) задать названия кнопок соответственно: Первая, Следующая, Предыдущая, Последняя, Выход (см. рис.10.12.)

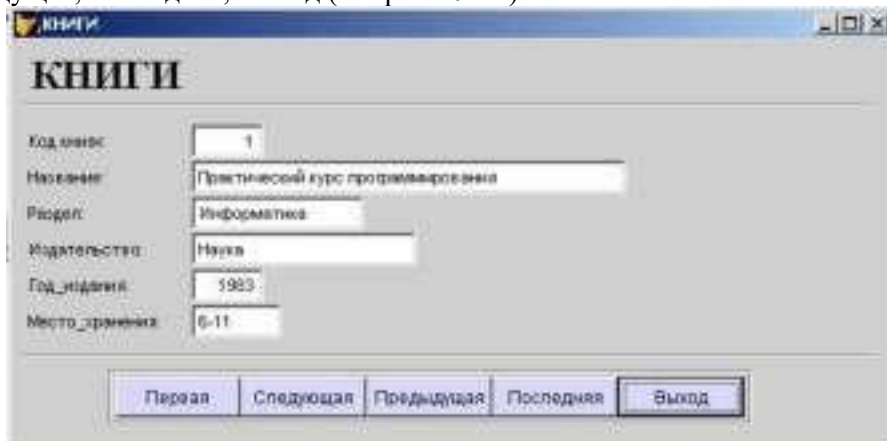


Рисунок 10.12. Форма Книги

Определить команды, которые будут выполняться при нажатии на созданные кнопки. Установить на объект CommandGroup (Группа кнопок) курсор, вызвать контекстное меню и выбрать команду Edit (Редактировать), выделить кнопку Первая, вызвать контекстное меню выбрать команду Code (Код программы), в появившемся окне ввести следующие команды:

Для кнопки Первая:

Переходим на первую запись и обновляем информацию в форме

```
IF !BOF()
```

```
GO TOP
```

```
ENDIF
```

```
_screen.ActiveForm.Refresh()
```

После ввода кода программы закрыть окно.

Ввести аналогично команды для остальных кнопок:

Для кнопки Следующая:

Переходим на следующую запись и обновляем информацию в форме

```
IF !EOF()
```

```
SKIP
```

```
ENDIF
```

```
_screen.ActiveForm.Refresh()
```

Для кнопки Предыдущая:

Переходим на предыдущую запись и обновляем информацию в форме

```
IF !BOF()
```

```
SKIP - 1
```

```
ENDIF
```

```
_screen.ActiveForm.Refresh()
```

Для кнопки Последняя:

Переходим на последнюю запись и обновляем информацию в форме

```
IF !BOF()
```

```
GO BOTTOM
```

```
ENDIF
```

```
_screen.ActiveForm.Refresh()
```

Для кнопки Выход:

Запрашиваем и выходим, если Да

```
IFMESSAGEBOX("Выходить из формы?", 4+32+256, "Выход")=6
```

```
_screen.ActiveForm.Release()
```

```
ELSE
```

```
_screen.ActiveForm.Refresh()
```

```
ENDIF
```

После ввода команд закрыть окно процедур

Запустить форму на выполнение. Проверить результат проделанной работы

Практическая работа № 11 Создание базы данных в среде разработки

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 языка Visual FoxPro

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

Для создания запросов можно использовать QueryWizard (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

Таблица 1. Назначение вкладок окна QueryDesigner (Конструктор запросов)

Вкладка	Назначение
Fields (Поля)	Позволяет указать поля исходных таблиц, выбираемые в результирующий запрос
Join (Объединение)	Позволяет задать условия объединения таблиц
Filter (Фильтр)	Позволяет определить фильтры, накладываемые для выбора записей
OrderBy (Упорядочение)	Позволяет задать критерии упорядочения данных
GroupBy (Группировка)	Позволяет задать условия группировки данных
Miscellaneous (Разное)	Позволяет задать дополнительные условия, такие как признак выборки повторяющихся значений, количество или процент выбора данных

Таблица 2. Назначение команд меню Query и кнопок панели инструментов Query Designer окна Query Designer (Конструктор запросов)

Команда меню	Кнопка	Назначение
AddTable (Добавить таблицу)		Добавляет в запрос новую таблицу
RemoveTable (Удалить таблицу)		Удаляет выбранную таблицу из запроса
RemoveJoinCondition (Удалить условие объединения)		Удаляет условие объединения таблицы
OutputFields (Результирующие поля)		Открывает вкладку Fields для выбора полей результирующей таблицы
Join (Объединение)		Открывает вкладку Join для создания условия объединения таблицы
Filter (Фильтр)		Открывает вкладку Filter для задания фильтра
OrderBy (Упорядочение)		Открывает вкладку OrderBy для определения критерия упорядочения данных
GroupBy (Группировка)		Открывает вкладку GroupBy для определения условия группировки данных
Miscellaneous (Разное)		Открывает вкладку Miscellaneous для задания дополнительных параметров запроса
Query destination (Результат запроса)		Открывает диалоговое окно Querydestination , в котором указывается куда выводить результат запроса
ViewSQL (Показать SQL)		Открывает диалоговое окно, в котором отображается SQL-оператор, соответствующий данному запросу
Maximizethetableview (Максимизировать панель отображения)		Раскрывает панель отображения используемых в запросе таблиц на весь экран. Повторное нажатие возвращает панели первоначальный размер
AddJoin (Добавить условие объединения)		Открывает диалоговое окно JoinCondition для задания условия объединения таблицы
Comments (Комментарии)		Открывает диалоговое окно, в котором вы можете ввести краткое описание создаваемого запроса
RunQuery (Выполнить запрос)		Запускает запрос на выполнение

Задание для практической работы
 Запустить программу Microsoft VisualFoxPro
 Открыть проект Информационная система
 Открыть БД Штат проекта

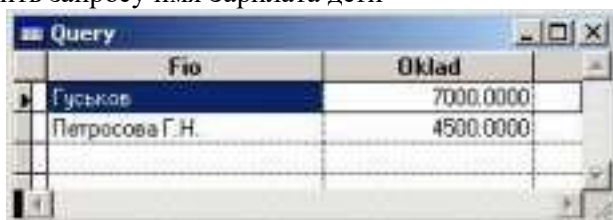
Создать запрос о сотрудниках, имеющих более одного ребёнка и получающих зарплату менее 9000: в окне ProjectManager щёлкнуть клавишей мыши на вкладке Data/Queries/New/QueryWizard, в появившемся диалоговом окне WizardSelection (Выбор мастера) выбрать пункт QueryWizard (Мастер запросов) и нажать кнопку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

В появившемся диалоговом окне выбора исходной таблицы и полей в области Databasesandtables (Базы данных и таблицы) выбрать необходимую БД штат и указать 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



Fio	Oklad
Гуськов	7000.0000
Петророва Г.Н.	4500.0000

Рисунок 11.1. Запрос Зарплата дети

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

Создать новый запрос, показывающий количество сотрудников организации, получающих определённую зарплату: в окне ProjectManager щёлкнуть 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 TablesinDatabase таблицу 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 (Закреть) 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 перенесётся 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 области SelectedFields (Выбранные поля) в область Orderingcriteria (Критерий упорядочения) перенести поле Сотрудники.oklad

Выбрать вкладку GroupBy (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 определённую 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

Зарплата	Количество_сотрудников
4500.0000	2
6000.0000	1
7000.0000	4
9000.0000	1
12000.0000	1
15000.0000	1

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 окно QueryDesigner (конструктор запросов), сохранить запрос, присвоив ему имя Книги по информатике



Название_книги	Раздел	Год_издания	Автор
Практический курс программирования	Информатика	1983	Фролов Геннадий
Практический курс программирования	Информатика	1983	Ильин Виктор
TURBO PASCAL для школьников	Информатика	1999	Попов Владимир
HTML в действии	Информатика	1997	Морис Брюс
Самоучитель Visual FoxPro 8.0	Информатика	2003	Омельченко Людмила

Рисунок 11.5. Запрос Книги по информатике

Практическая работа № 16-17 Установка и настройка SQL-сервера

Настройка локальной сети

Цель практической работы

Познакомиться с основными принципами создания базы данных в MS SQL Server. Изучить операции, проводимые с базами данных в целом. Получить навыки использования программы "SQL Server Management Studio" для создания, удаления, регистрации, подключения, извлечения метаданных, резервного копирования и восстановления базы данных. Изучить SQL-операторы для создания, подключения и удаления базы данных. Познакомиться с основными принципами управления учетными записями и ролями.

Исходные данные

Студент получает индивидуальный вариант исходных данных с кратким описанием предметной области, который используется при выполнении всех описанных в данном пособии практических работ. При этом каждая очередная Практическая работа является продолжением выполненной ранее и поэтому они должны обязательно выполняться последовательно.

Используемые программы

1. Работающий на компьютере сервер "MS SQL Server 2008 R2".
2. Установленная платформа .NET Framework 2.0, 3.0, 3.5 или 4.0.
3. Операционная система Microsoft Windows 2000/XP/2003/Vista/Windows 7/Windows 8.
4. Приложение "SQL Server Management Studio 2008 rus", установленное на локальном компьютере.

Теоретические сведения

На сегодняшний день известно более двух десятков серверных СУБД, из которых наиболее популярными являются Oracle, Microsoft SQL Server, Informix, DB2, Sybase, InterBase, MySQL.

Для выполнения практических работ будет использоваться сервер "Microsoft SQL Server 2008", установленный на сервере кафедры компьютерных систем и сетей (компьютер pi_srv).

Microsoft® SQL Server™ — это система анализа и управления реляционными базами данных в решениях электронной коммерции, производственных отраслей и хранилищ данных.

Microsoft SQL Server — система управления реляционными базами данных (СУБД), разработанная корпорацией Microsoft. Основным используемым языком запросов — **Transact-SQL**, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта



ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

В SQL Server 2008 имеется большой набор интегрированных служб, расширяющих возможности использования данных: вы можете составлять запросы, выполнять поиск, проводить синхронизацию, делать отчеты, анализировать данные. Все данные хранятся на основных серверах, входящих в состав центра обработки данных. К ним осуществляется доступ с настольных компьютеров и мобильных устройств. Таким образом, вы полностью контролируете данные независимо от того, где вы их сохранили.

Система SQL Server 2008 позволяет обращаться к данным из любого приложения, разработанного с применением технологий Microsoft .NET и Visual Studio, а также в пределах сервисно-ориентированной архитектуры и бизнес-процессов — через Microsoft BizTalk Server. Сотрудники, отвечающие за сбор и анализ информации, могут работать с данными, не покидая привычных приложений, которыми они пользуются каждый день, например приложений выпуска 2007 системы Microsoft Office.

В Microsoft SQL базы данных хранятся в виде обычных файлов на диске. Как минимум на одну БД приходится таких **файлов 2: *.mdf и *.ldf**. В первом хранятся сами данные, таблицы, индексы и пр., а во втором находится т.н. transaction log, в котором находится информация необходимая для восстановления БД.

Файл с базой данных представляет собой набор страниц одинакового размера. Размер страницы задается при создании базы данных и может быть изменен только при ее восстановлении из резервной копии. Чтение и запись данных в базе данных осуществляется постранично.

Все операции с базой данных должны производиться только посредством команд к SQL-серверу. Для клиентских приложений эти файлы абсолютно бесполезны и при правильной организации доступа пользователей к файлам в сети, вообще не должны быть доступны.

Сервер СУБД не имеет интерфейса пользователя и для выполнения операций с базой данных ему необходимо посылать команды либо с помощью командной строки или с помощью какой-либо прикладной программы.

Для выполнения операций с базой данных при проведении практических работ предлагается использовать программу "**SQL Server Management Studio 2008 Rus**" (рис. 1), представляющую собой наиболее распространенное и удобное средство администрирования баз данных под управлением MS SQL Server (Среда ManagementStudio Express доступна для свободной загрузки из центра загрузки Майкрософт - http://download.microsoft.com/download/5/C/0/5C0C5CE4-10EB-4623-A63E-8D850D55D8EF/SQLEXPRESS_x86_RUS.exe).

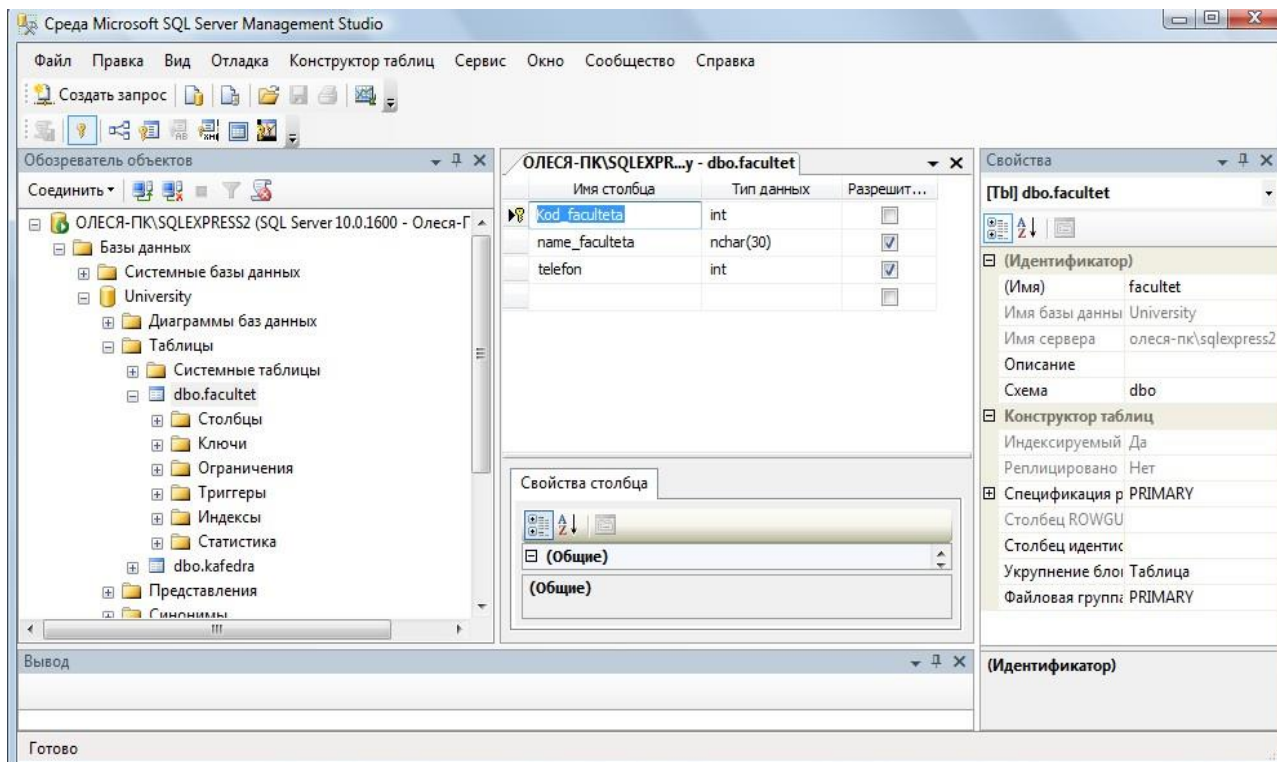


Рис. 1. Программа SQL Server Management Studio

Среда SQL Server Management Studio — это интегрированная среда для доступа, настройки, управления, администрирования и разработки всех компонентов SQL Server. Среда SQL Server Management Studio объединяет большое число графических средств с набором полнофункциональных редакторов сценариев для доступа к SQL Server разработчиков и администраторов с любым опытом работы.

Среда SQL Server Management Studio обеспечивает следующие основные возможности:

- поддерживает большинство административных задач для SQL Server;
- единая интегрированная среда для управления SQL Server Database Engine и разработки;
- новые управляющие диалоговые окна для управления объектами в компоненте SQL Server Database Engine, службах Analysis Services, Reporting Services, Notification Services и выпуске SQL Server Compact 3.5 с пакетом обновления 1 (SP1), позволяющие выполнять действия немедленно, направлять их в редактор кода или включать эти действия в сценарий для последующего выполнения;
- экспорт и импорт регистрации сервера среды SQL Server Management Studio из одной среды Management Studio в другую;
- сохранение и печать XML-файлов плана выполнения и взаимоблокировок, созданных приложением SQL Server Profiler, просмотр их в любое время и отправка для анализа администратору;
- новые окна сообщений об ошибках и информационных сообщений, предоставляющие гораздо больше сведений и позволяющие отправлять в Майкрософт комментарии о сообщениях, копировать сообщения в буфер обмена и отправлять их по электронной почте в службу поддержки;
- встроенный веб-обозреватель для быстрого обращения к библиотеке MSDN или получения интерактивной справки;
- встроенная справка от сообществ в Интернете и т.д.

Большинство действий с базой данной MS SQL Server в среде Среда SQL Server Management Studio может быть осуществлено двумя способами: **либо выполнением**

операторов языка SQL в окнах "Script Execute" (подключение к базе данных не обязательно) и "SQL Editor" (требуется подключение к базе данных), либо с использованием меню и диалоговых окон. В последнем случае операторы SQL, которые требуются для выполнения данного действия, будут сгенерированы и выполнены средой SQL Server Management Studio автоматически.

Задание

Практическую работу следует выполнять в следующем порядке:

1. Создать на сервере `pi_srv` (или на локальном компьютере, если нет сервера) рабочую папку для хранения файлов, получаемых при выполнении практической работы. Эта папка должна располагаться в папке `\Базы данных\Группа\Студент` и соответствовать номеру выполняемой практической работы.

2. На основании индивидуального задания выбрать имя файла создаваемой базы данных. Для имени лучше всего выбрать одно или несколько английских слов, соответствующих наименованию предметной области. Использование для имени русских слов, записанных латинскими буквами, не допускается.

3. Открыть приложение "Среда SQL Server Management Studio". Для этого можно либо воспользоваться меню Пуск (**Пуск/Программы/ Microsoft SQL Server 2008 / Среда SQL Server Management Studio**).

4. Создать соединение с локальным или удаленным сервером.

5. Создать базу данных для своей предметной области с помощью диалога, выбрав сервер "`pi_srv`" или локальный сервер "**Имя_компьютера\SQLEXPRESS**"

6. Создать базу данных и указать в качестве имени файла "`\Базы данных\Группа\ФИО_студента\Название_БД`".

7. Извлечь метаданные для автоматической генерации команды создания базы данных.

8. Удалить базу данных, выполнив команду "**Database/Drop Database**" (База данных/Удалить базу данных).

9. Создать базу данных вторым способом, выполнив в окне "**Script Executive**" операторы, полученные при извлечении метаданных перед предыдущим удалением.

10. Создать резервную копию базы данных.

11. Удалить базу данных.

12. Восстановить базу данных из резервной копии.

13. Сохранить файл сценария на сервере в папке "Студент", дав ему имя «лаб.№1» и стандартное расширение "***.sql**".

Ход работы

Создание соединения с сервером

Выполните следующие инструкции:

Работа с приложением **SQL Server Management Studio** начинается с создания соединения с установленным сервером. Убедитесь вначале, что сервер Microsoft SQL Server (2008) на локальной машине или на сервере компьютерного класса установлен и работает.

Откройте приложение "SQL Server Management Studio". Для этого можно либо воспользоваться меню Пуск (**Пуск/Программы/ Microsoft SQL Server 2008 / Среда SQL Server Management Studio**).

В диалогом окне **Соединение с сервером** подтвердите заданные по умолчанию параметры и нажмите кнопку **Соединить**, см. рис.2.

Для соединения необходимо, чтобы поле **Имя сервера** содержало имя компьютера,

на котором установлен SQL Server.

Если компонент **Database Engine** является именованным экземпляром, то поле **Имя сервера** должно также содержать имя экземпляра в формате **<имя_компьютера>\<имя_экземпляра>**.

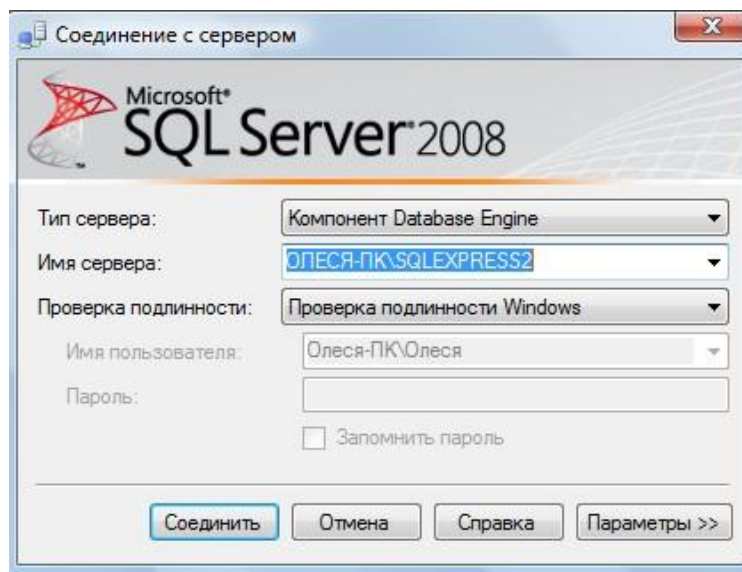


Рис. 2. Создание соединения с сервером В

параметрах указываем:

Тип сервера – Компонент Database Engine.

Имя сервера. Подключение может быть *локальным* или *удаленным*. Представляет собой название компьютера в сети, на котором установлен сервер СУБД. Если сервер установлен на том же компьютере, где сейчас работает пользователь, то в качестве имени используется имя компьютера и идентификатор сервера;

проверка подлинности – Windows (по умолчанию),

имя пользователя – имя пользователя по умолчанию, зарегистрированного на сервере MS SQL Server (задается при установке сервера),

пароль – пусто или пароль для пользователя, заданного для сервера MS SQL Server;

Нажмите кнопку **Соединить**. Если соединение будет совершенно успешно, то на экране появятся данные сервера.

Среда Management Studio представляет данные в виде окон, выделенных для отдельных типов данных. Сведения о базе данных отображаются в обозревателе объектов и окнах документов.

Обозреватель объектов является представлением в виде дерева, в котором отображаются все объекты базы данных на сервере. Он может содержать базы данных компонента SQL Server Database Engine, служб Analysis Services, служб Reporting Services, служб Integration Services и SQL Server Compact 3.5 с пакетом обновления 1 (SP1).

Обозреватель объектов включает сведения по всем серверам, к которым он подключен. При открытии среды Management Studio пользователю предлагается применить при подключении обозревателя объектов параметры, которые использовались в прошлый раз. Чтобы подключиться к любому из серверов, следует дважды щелкнуть его в компоненте «**Зарегистрированные серверы**», однако регистрировать его не обязательно, см. рис.1.

Окно документов представляет собой наиболее крупную часть среды Management Studio. В окнах документов могут размещаться редакторы запросов и окна обзора. По

умолчанию отображается страница «Сводка», подключенная к экземпляру компонента Database Engine на текущем компьютере.

Общие сведения о базах данных MS SQL Server

Кроме четырех системных баз, SQL Server может обрабатывать до **32 734** баз данных, определяемых пользователем.

База данных представляет собой:

- набор взаимосвязанных таблиц;
- связанный набор страниц, выделенных для хранения данных MS SQL Server;
- совокупность данных при архивации;
- два и более файла;
- важную совокупность данных для целей защиты и управления.

Файлы базы данных

База данных состоит из двух и более файлов, каждый из которых может использоваться лишь одной базой.

У файлов существуют два имени: **логическое** и **физическое**. **Логическое имя** подчиняется стандартным правилам выбора имен объектов SQL Server. **Физическое имя** представляет собой полное имя любого локального или сетевого файла. Максимальное число файлов в базе данных — 32 768. **Файлы делятся на три типа:**

- **Первичные файлы.** Используются для хранения данных и информации, определяющих начальные действия с базой. База данных содержит лишь один первичный файл. Стандартное расширение — **.mdf**.

- **Вторичные файлы.** Одна или несколько вспомогательных областей для хранения данных. Могут использоваться для распределения операций чтения/записи по нескольким дискам. Стандартное расширение — **.ndf**.

- **Файлы журналов.** Содержат журналы транзакций базы данных. База данных содержит по крайней мере один файл журнала. Стандартное расширение — **.ldf**. Перед непосредственной записью транзакций в файл данных все вносимые изменения записываются в журнал.

Группы файлов

Группы файлов предназначены для объединения нескольких файлов. Каждый файл может входить не более чем в одну группу. Файлы журналов не могут принадлежать никаким группам. Группы файлов используются для распределения операций чтения/записи по нескольким дискам. Если группа содержит более одного файла, операции записи распределяются между файлами группы. Базы данных могут содержать до 32 768 групп файлов.

У каждой базы данных имеется **первичная группа файлов**. Она содержит первичный файл данных и все файлы, которые не были явно назначены в другую группу файлов. Имя первичной группы файлов — **PRIMARY**.

Создание и регистрация базы данных

Для создания базы данных можно использовать один из **двух способов**:

Первый способ создания БД. Выполнить команду "База данных/Создать базу данных..." в программе SQL Server Management Studio, ввести параметры создаваемой базы данных в диалоговом окне "Создание базы данных" (рис. 3) и нажать кнопку [OK].

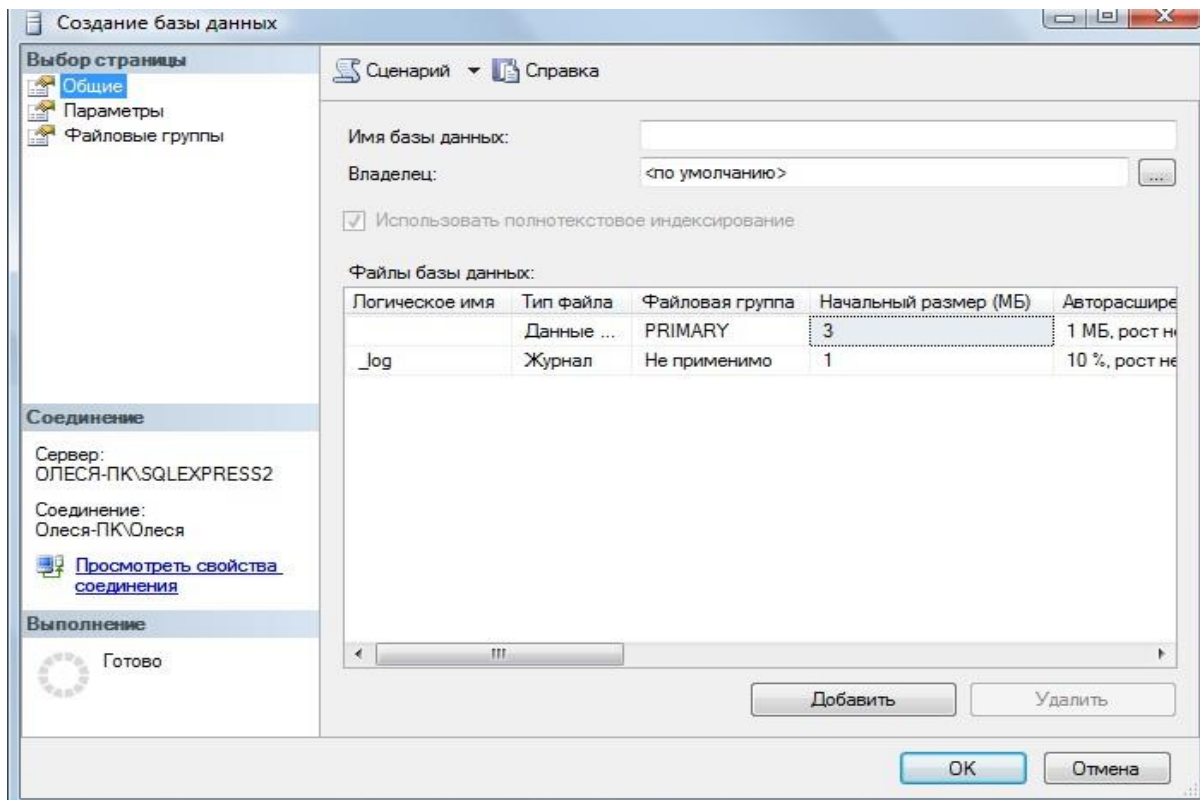


Рис. 3. Диалоговое окно создания базы данных

В поле **Имя базы данных** введите имя нашей будущей базы данных, например – **University**.

Поле **Владелец** - задан по умолчанию, в зависимости от настройки сервера.

Папка с базой данных будет создана по умолчанию на диске **C:\Program Files\Microsoft SQL Server\MSSQL10.SQLEXPRESS2\MSSQL\DATA **.

Прежде чем нажать кнопку **Добавить**, просмотрите **Параметры** и **Файловые группы** для создаваемой базы данных.

После нажатия на кнопку **[OK]** программа " SQL Server Management Studio " создаст базу данных, имя которой вы увидите в обозревателе объектов, а также сгенерирует необходимый SQL-код для создания базы данных с теми свойствами, которые указаны в этом диалоговом окне и передаст его серверу СУБД для выполнения.

Пример этих операторов приведен на рис. 4. (нажмите на имени базы данных **University** правой клавишей и из контекстного меню выберите **Создать скрипт как.. CREATE**). Если параметры введены правильно, база данных будет создана.

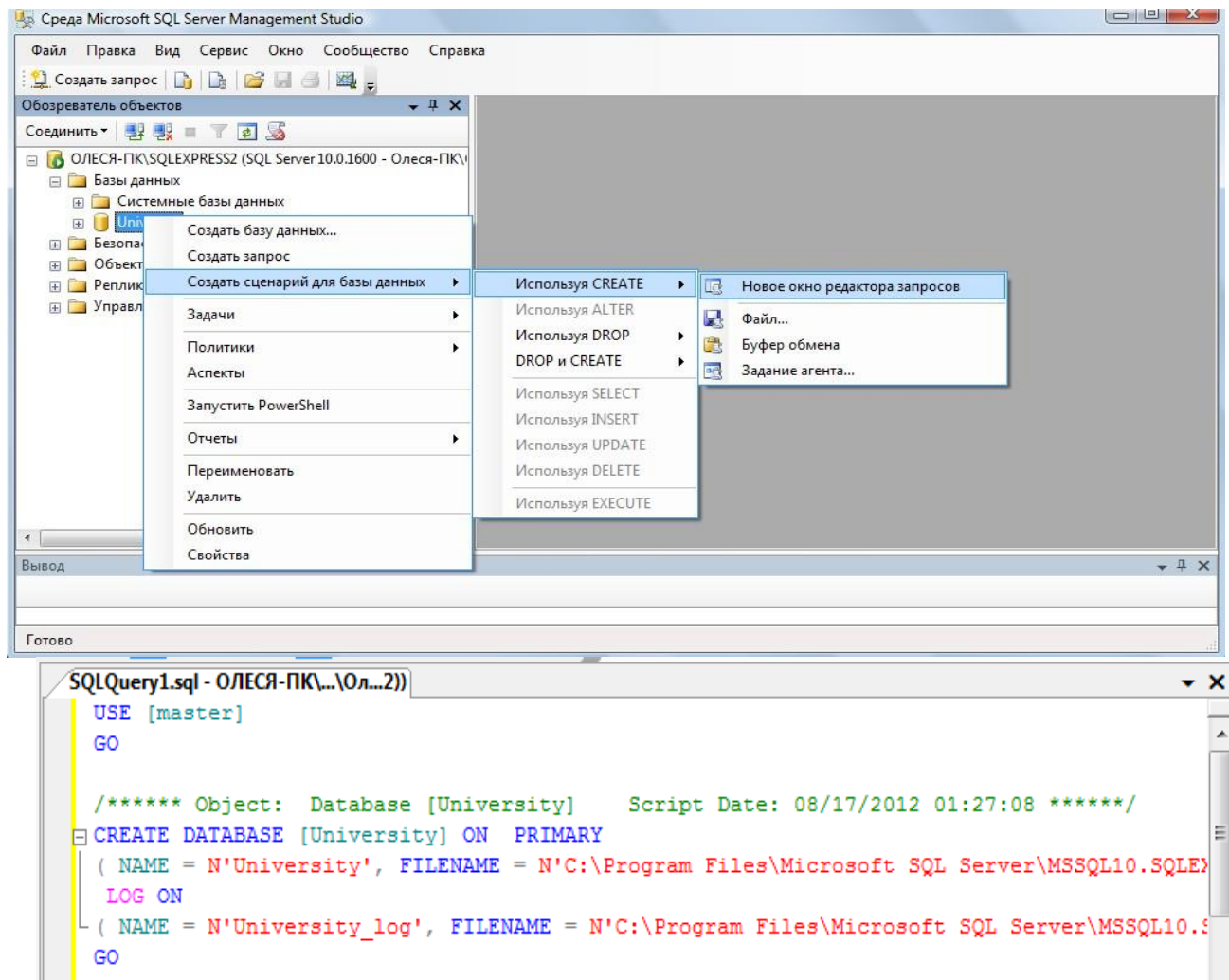


Рис. 4. Сгенерированный sql-код созданной базы данных

Содержащиеся в сценарии операторы отделяются друг от друга символом ";". Сценарий может содержать поясняющие комментарии двух видов: многострочный комментарий (начинается символами "/*" и заканчивается символами "*/") и однострочный комментарий, который начинается символами "--" и продолжается до конца строки.

При создании базы данных возможны следующие типичные ошибки:

1. На целевом компьютере не запущен или не установлен сервер СУБД – т.е. выполнять команду создания базы данных просто некому.
2. На целевом компьютере нет каталога, в котором предполагается создать базу данных.
3. Файл, в котором должна будет находиться база данных на сервере, уже существует.

После создания базы данных вся введенная о базе данных информация запоминается программой SQL Server Management Studio и в окно редактора в дерево на вкладке "Проводник" добавляется узел с зарегистрированной базой данных (рис. 5).

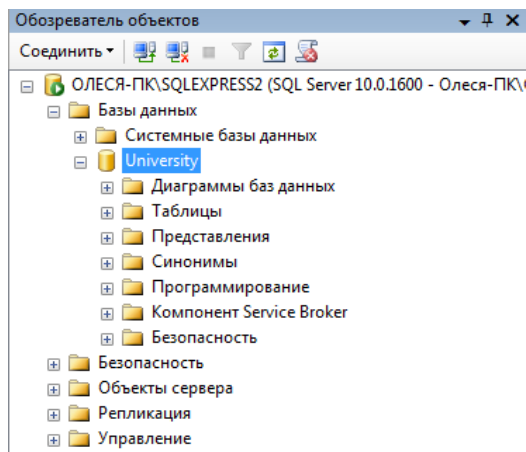




Рис. 5. Перечень зарегистрированных баз данных в SQL Server Management Studio

Второй способ создания БД. Выполнить в программе SQL Server Management Studio команду "Создать запрос"  Создать запрос на панели инструментов, затем ввести команду, создающую базу данных в окне "Script Execute" (рис. 3) и нажать кнопку  Выполнить.

Команда CREATE DATABASE - Создание базы данных MS SQL Server

Базы данных создаются командой **CREATE DATABASE**. Создание баз данных разрешается любому пользователю с ролью системного администратора или всем, кому системный администратор предоставил такое право. Команда **CREATE DATABASE** имеет следующий синтаксис:

```

01. CREATE DATABASE имя_базы
02. [ ON [PRIMARY] ([ NAME = логическое_имя_файла, ]
03. FILENAME = 'имя_файла_ОС'
04. [, SIZE = размер]
05. [, MAXSIZE = { максимальный_размер | UNLIMITED } ]
06. [, FILEGROWTH = приращение] )
07. | {FILEGROUP имя_группы_файлов FILEDEFINITIONS}
08. [,...n] ]
09. [LOG ON {[ NAME = логическое_имя_файла, ]
10. [FILENAME = 'имя_файла_ОС'
11. [, SIZE = размер]
12. [, MAXSIZE = { максимальный_размер | UNLIMITED } ]
13. [, FILEGROWTH = приращение] } [,...n]
14. [FOR LOAD | FOR ATTACH]

```

Если при создании базы не указан первичный файл данных и/или файл журнала, то отсутствующий файл (или файлы) создается с именем по умолчанию.

Физические файлы будут находиться в стандартном каталоге.

Первичному файлу присваивается имя **имя_базы.mdf**, а файлу журнала — **имя_базы_log.ldf**.

Если размер файлов не задан, то при создании размер первичного файла совпадает с размером первичного устройства базы **model**, а размер файла журнала и вторичных файлов данных равен 1 Мбайт. Он может быть и больше, если размер первичного файла базы данных **model** превышает 1 Мбайт. Хотя имена и размеры файлов указывать не обязательно, на практике это всегда следует делать. SQL Server создает базу данных за два этапа. На первом этапе база **model** копируется в новую базу данных, а на втором этапе инициализируется все неиспользуемое пространство.

Команда **CREATE DATABASE** имеет следующие параметры:

- **PRIMARY** — файл определяется как первичное устройство.

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


Пользователь, создавший базу данных, является ее владельцем. Все параметры конфигурации базы копируются из базы model, если только при создании базы не был указан параметр **FOR ATTACH**. В этом случае параметры конфигурации читаются из существующей базы данных. Рассмотрим некоторые примеры команды **CREATE DATABASE**:

/* База данных со стандартным размером и именами файлов */

```

01. CREATE DATABASE test1
02. /* Данные – 2 Мбайт, файл журнала – по умолчанию */
03. CREATE DATABASE test2
04. ON (FILENAME = 'c:\d1.mdf', SIZE = 2, NAME = 'd1')
05. /* Первичный файл – 10 Мбайт, одна группа файлов
06. g1 и журнал размером 10 Мбайт */
07. CREATE DATABASE test3
08. ON PRIMARY (FILENAME = 'c:\test3.mdf',
09. SIZE = 10 , NAME = 'd1'),
10. FILEGROUP g1 (FILENAME = 'c:\g1.mdf',
11. SIZE = 10 , NAME = 'g1')
12. LOG ON (FILENAME = 'c:\test3.ldf',
13. SIZE = 10, NAME = 'log1')

```

Задача 1. Создайте sql-скрипт создания новой базы данных под именем **Educator** на "D:\Базы данных\Группа\ФИО_студента\Название_БД.mdf, с первичным устройством, с исходным размером файла в 10 Мбайт и запустите на выполнение скрипт (кнопка  на панели инструментов). Выполните в окне обозревателя объектов **Обновление**. Сохраните созданный скрипт в текущую папку под именем **1.sql**.

После успешного выполнения и обновления проводника у вас должна появиться новая база данных.

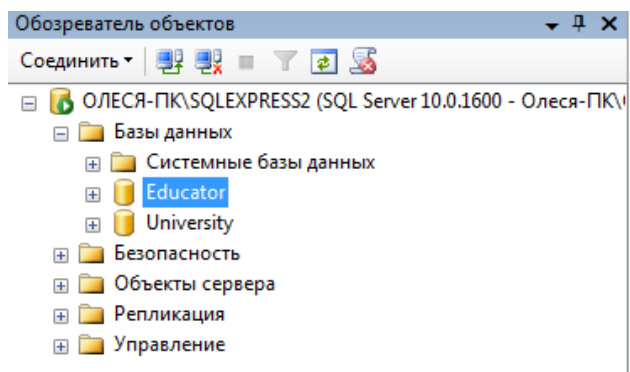


Рис. 6. Окно проводника после выполнения сценария создания базы данных

Подключение к базе данных

Чтобы подключиться к зарегистрированной базе данных, надо выбрать нужную базу данных в списке (рис. 5) и сделать двойной щелчок мышкой на выбранной базе

данных.

Если все параметры подключения были введены правильно, то произойдет подключение к базе данных, название подключенной базы данных в окне "Обозревателя объектов" будет выделено жирным шрифтом, а также появятся вложенные узлы с объектами, содержащимися в подключенной базе данных (рис. 7).

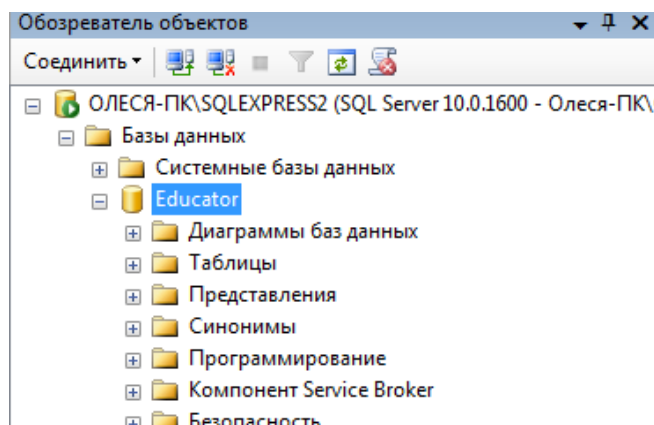


Рис. 7. Зарегистрированные базы данных в SQL Server Management Studio

После подключения к базе данных можно просматривать имеющиеся объекты, создавать новые, вносить и просматривать данные, а также проводить операции с имеющимися объектами.

После создания БД в окне **Обозревателя объектов** (его можно вызвать по <F8>) выбираем **DataBases (Базы данных)** и откроется список БД, в котором откроем созданную БД (если она не появилась, то в окне **Object Explorer** нажать <F5> для обновления списков), которая состоит из восьми вложенных разделов (некоторые содержат еще дополнительные разделы), соответствующих объектам СУБД SQL Server:

Database Diagrams (Диаграммы БД)	Views (Представления)	Programmability (Объекты программирования)
Tables (Таблицы)	Synonyms (Синонимы)	Security (Безопасность)
Service Broker	Storage	

На начальном этапе раздел созданной БД пуст, за исключением некоторых объектов, которые создаются по умолчанию, например в разделе **Security/ Users** создаются пользователи, которые имеют право на доступ к объектам БД, их можно изменить.

Удаление базы данных

Для удаления базы данных можно использовать один из трех способов:

1. Выполнить в программе " SQL Server Management Studio " команду контекстного меню "Удалить" , выбрав перед этим в списке базу данных, а затем подтвердить свое желание в диалоговом окне.
2. Выполнить оператор **DROP DATABASE** в SQL-редакторе.
3. Удалить файл с базой данных.

Синтаксис оператора **DROP DATABASE**:

DROP DATABASE database_name;

Резервное копирование и восстановление

Резервное копирование (backup) базы данных и *восстановление* из резервной копии (*restore*) – два важнейших и наиболее частых процесса, осуществляемых

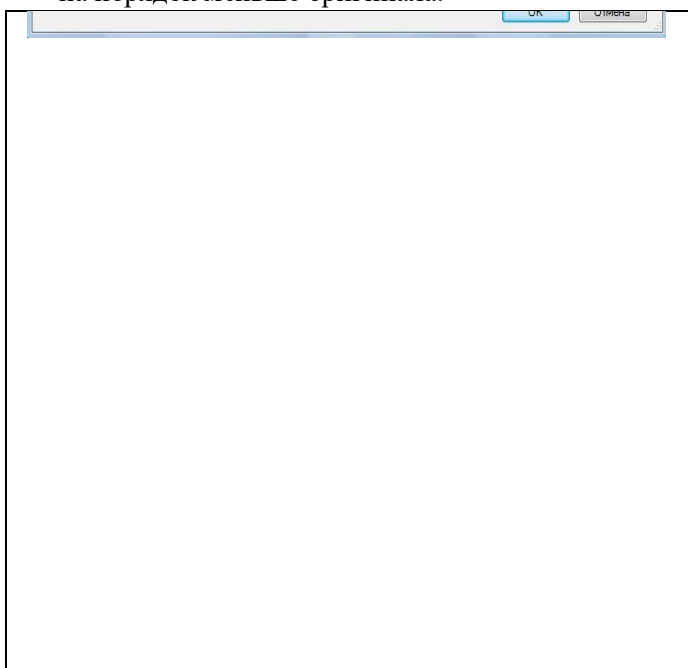
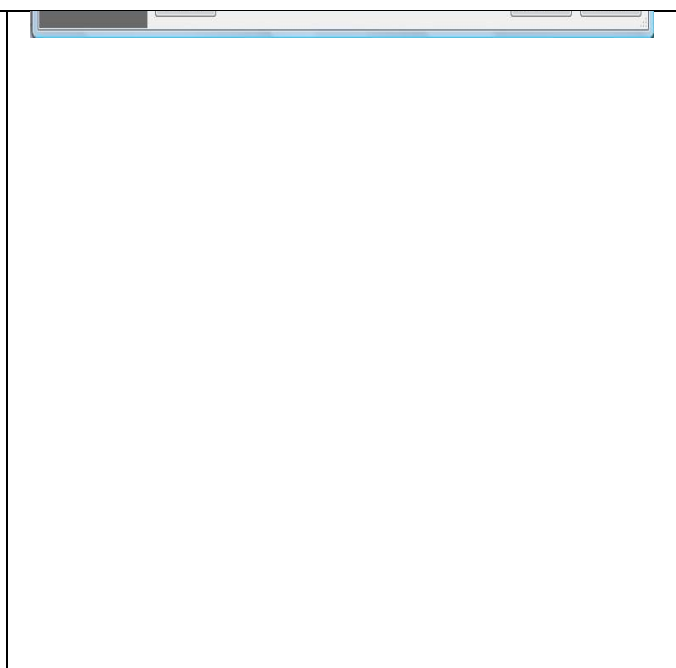
администраторами баз данных.

Резервное копирование базы данных – единственный надежный способ предохранить данные от потери в результате поломки диска, сбоев электропитания, действий злоумышленников и ошибок в программах. В процессе резервного копирования создается независимый от платформы "снимок" базы данных, с помощью которого можно перенести данные на другую операционную систему или даже другую платформу. **Полный цикл:** резервное копирование и восстановление из резервной копии приводит к корректировке статистической информации, является средством от излишнего "разбухания" базы данных и необходимой операцией обслуживания баз данных. Кроме того, миграция от одной версии сервера к другой также происходит при помощи процесса **backup/restore**.

Для создания резервной копии базы данных с помощью программы "SQL Server Management Studio" необходимо подключиться к базе данных, выбрать из контекстного меню базы данных **Задачи/ Создать резервную копию**. В открывшемся диалоговом окне "**Мастер резервного копирования**" задать несколько параметров и нажать кнопку [**Выполнить**], см. рис.8.

После выбора пути и файла для резервной копии в окне **Back Up Database** нажатием на ОК запускаем процесс создания резервной копии. В случае успешной работы появится сообщение.

В результате будет создан файл с резервной копией. Стандартным расширением таких файлов для "SQL Server Management Studio" является **"*.bak"**. Файл с резервной копией базы данных обычно на порядок меньше оригинала.

	
Рис.8. Создание резервной копии базы данных	Рис.9. Восстановление базы данных

Для восстановления базы данных из резервной копии используется команда **"База данных/ Восстановление базы данных"**. В результате откроется диалоговое окно "**Мастер восстановления баз данных**", в котором надо выбрать имя БД куда будет восстанавливаться база данных, в которую будет помещен результат, способ восстановления, файл, из которого будет восстанавливаться база данных, отмечаем выбранную резервную копию, и нажать кнопку [**Восстановить**], см.рис.9. Запускаем процесс восстановления. В случае успешного выполнения получим сообщение.

Резервное копирование и восстановление базы данных, наряду с процессом

извлечения метаданных и последующего выполнения полученного сценария, можно использовать при переносе разрабатываемой базы данных между различными компьютерами для обеспечения самостоятельной работы студентов над практическими работами и курсовым проектом.

Самостоятельно Выполните вначале резервирование, а затем восстановление базы данных.

Удалите базу данных **Educator** с помощью скрипта сохраните sql-запрос.

Копирование и перенос на другой сервер БД

Для просмотра, запуска, остановки служб MS SQL Server необходимо запустить утилиту **SQL Server Configuration Manager** (рис. 10).

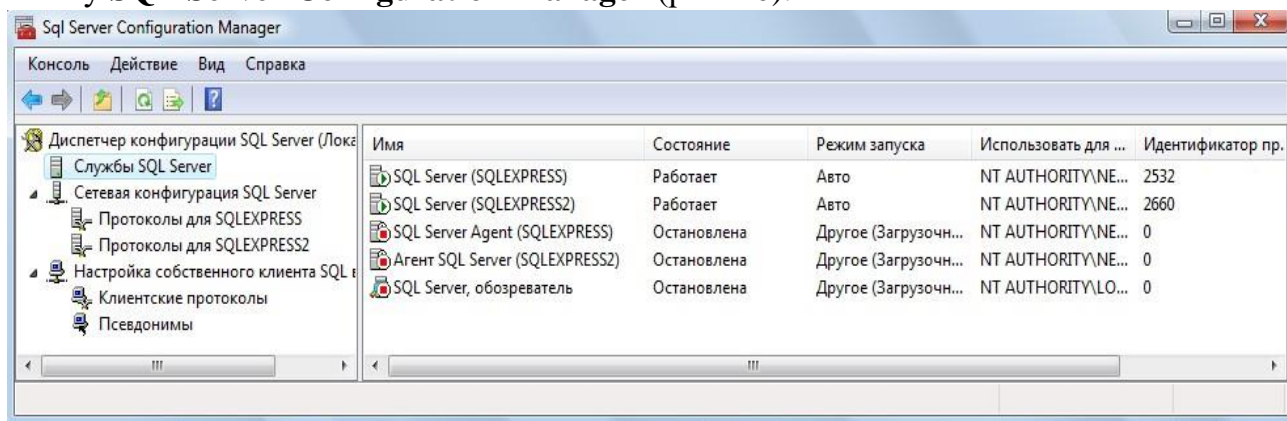


Рис.10. Список служб сервера БД

Для того **чтобы скопировать БД** необходимо остановить службу **SQL Server** (в ее контекстном меню выбрать **Stop**). Далее в подпапке **... \MSSQL.1\MSSQL\Data** скопировать файлы с вашим названием БД (по умолчанию их два). Не забудьте потом снова запустить службу **SQL Server** (в ее контекстном меню выбрать **Start**).

Для того **чтобы подключить** скопированную БД на другом сервере, нужно предварительно скопировать ваши файлы в папку **... \MSSQL.1\MSSQL\Data** соответствующего сервера. Далее запустить утилиту **SQL Server Management Studio**. В появившемся окне с названием **Object Explorer** Проводник объектов (его можно вызвать по <F8>) выбираем **DataBases (Базы данных)** и по <правой кнопке мыши> в контекстном меню (рис. 5) выбираем **Attach... (Присоединить...)**. В появившемся окне **Attach DataBases (Присоединение базы данных)** нажать <Add> и выбрать ваш файл БД с расширением **.mdf**.

Системные базы данных

Системные базы данных сервера, создаваемые при установке, и их файлы представлены в таблице 1.

Название	Назначение	Размещение
Master	Хранит всю информацию сервера, включая учетные записи и параметры, сведения о всех базах и нахождении их первичных файлов с данными об инициализации баз данных пользователя.	Master.mdb – файл данных (75.mb) Mastlog.ldf – журнал транзакций (1 mb)
TempDB	Хранит все временные системные и пользовательские объекты: таблицы, переменные, хранимые процедуры, курсоры и т.п.	Tempdb.mdf – файл данных (8 mb) Templog.ldf – журнал транзакций (0.5 mb)
Model	Является шаблоном, задаваемых администратором и используемым для создания любых пользовательских баз данных. Содержит параметры по умолчанию, которые можно переопределять при создании соответствующей базы данных пользователя.	Model.mdf – файл данных (0.75 mb) Model.ldf – журнал транзакций (0.75 mb)
MSDB	Хранит информацию, относящуюся к автоматизации администрирования и управления сервером.	Msdbdata – файл данных (3.5 mb) Msdblog – журнал транзакций (0.75 mb)
		Всего – 22.75 mb

Все системные и пользовательские базы данных содержат в обязательном порядке 18 системных таблиц, которые хранят информацию, определяющие структуру и организацию соответствующей базы данных.

MSSQL Server поддерживает два основных класса приложений клиентского типа

:

1. приложения реляционных баз данных, использующие команды Transact - SQL с расширениями ODBC и набор стандартных функций и объектно-ориентированных методов;
2. web - приложения , использующие команды Transact - SQL или запросы на языке Xpath и документы XML.

Оба класса приложений используют API интерфейс баз данных типа OLE DB или ODBC.

Основные принципы управления учетными записями и ролями в MS SQL Server

Список системных процедур и команд, которые позволяют реализовать политику разделения прав между пользователя БД.

Название встроенной процедуры	Описание
sp_grantlogin	– позволяет использовать пользователей или группы ОС для соединения с Microsoft SQL Server™ , используя Windows Authentication . Этот пример позволяет пользователю Windows NT Corporate\BobJ соединиться с SQL Server. Например, EXEC sp_grantlogin 'Corporate\BobJ'

sp_defaultdb	Изменяет для пользователя БД по умолчанию Этот пример устанавливает БД по умолчанию pubs для пользователя Victoria. Например, EXEC sp_defaultdb 'Victoria', 'pubs'
sp_grantdbaccess	Добавляет учетную запись из раздела security в текущую БД, для учетных записей Microsoft Windows также дает разрешение на доступ к текущей БД. Синтаксис: EXEC sp_grantdbaccess [@loginame =] 'login' [, [@name_in_db =] 'name_in_db' [OUTPUT]] Этот пример добавляет учетную запись Corporate\GeorgeW в текущую БД и присваивает псевдоним внутри БД Georgie. Например, EXEC sp_grantdbaccess 'Corporate\GeorgeW', 'Georgie'
sp_revokedbaccess	Удаляет информацию об учетной записи из текущей БД. Синтаксис: EXEC sp_revokedbaccess [@name_in_db =] 'name' Этот пример удаляет учетную запись Corporate\GeorgeW из текущей БД. EXEC sp_revokedbaccess 'Corporate\GeorgeW'
sp_addrole	Создает новую роль в текущей БД. Этот пример создает новую роль в текущей БД с названием Managers. EXEC sp_addrole 'Managers'
sp_addrolemember	В текущей БД назначает роль конкретному пользователю. Пример А. Этот пример добавляет учетную запись Corporate\JeffL из Windows NT в БД Sales как пользователя Jeff. Jeff затем получает роль Sales_Managers в БД Sales. USE Sales --сделать текущей БД Sales GO --выполнить команду, а потом запустить следующую EXEC sp_grantdbaccess 'Corporate\JeffL', 'Jeff' GO EXEC sp_addrolemember 'Sales_Managers', 'Jeff' Пример В. Этот пример добавляет пользователя SQL Server с именем Michael к роли Engineering в текущей БД. EXEC sp_addrolemember 'Engineering', 'Michael'
sp_helpprotect	Показывает список привилегий, ассоциированных с ролью.
sp_helprolemember	Показывает список пользователей БД, входящих в указанную роль
sp_addsrvrolemember	Присвоение встроенной серверной роли для существующей учетной записи sp_addsrvrolemember [@loginame =] 'login' , [@rolename =

	<p>] 'role'</p> <p>Например:</p> <p>sp_addsrvrolemember 'Admin_DB', 'sysadmin'</p>
sp_dropsrvrolemember	<p>Удаление встроенной серверной роли для учетной записи или группы</p> <p>sp_dropsrvrolemember [@loginame =] 'login' , [@rolename =] 'role'</p> <p>Например:</p> <p>sp_dropsrvrolemember 'Admin_DB' , 'sysadmin'</p>
sp_helpsrvrole	<p>Описание только встроенных ролей в SQL Server</p> <p>sp_helpsrvrole [[@srvrolename =] 'role']</p> <p>Например:</p> <p>sp_helpsrvrole 'sysadmin'</p>
sp_helpsrvrolemember	<p>Возвращает список ролей и учетных записей, которым присвоены эти роли</p> <p>sp_helpsrvrolemember [[@srvrolename =] 'role']</p> <p>Например:</p> <p>sp_helpsrvrolemember 'sysadmin'</p>
sp_srvrolepermission	<p>Возвращает список ролей и разрешений, которые присвоены этим ролям</p> <p>sp_srvrolepermission [[@srvrolename =] 'role']</p> <p>Например:</p> <p>sp_srvrolepermission 'sysadmin'</p>
sp_addlogin sp_adduser	<p>Создание новой учетной записи в SQL Server в разделе Security:</p> <p>sp_addlogin [@loginame =] 'login' [, [@passwd =] 'password'] [, [@defdb =] 'database'] [, [@deflanguage =] 'language'] [, [@sid =] sid] [, [@encryptopt =] 'encryption_option']</p> <p>Например:</p> <p>sp_addlogin 'login1',sysname, 'DB_Books'</p> <p>Создает пользователя в SQL Server без PUBLIC в БД 'DB_Books'.</p> <p>Нужно еще использовать</p> <p>sp_adduser [@loginame =] 'login' [, [@name_in_db =] 'user'] [, [@grpname =] 'group'],</p> <p>Пример:</p> <p>Создана база данных DB_Books. В ней создан пользователь Admin_DB с серверной ролью sysadmin, с ролью в БД db_owner.</p> <p>Создать в QueryAnalyzer нового пользователя с именем Public_</p>

	<p>и паролем Public_1 (пароль не должен совпадать с именем пользователя) с помощью следующих команд (не забудьте нажать F5 для запуска команд на выполнение):</p> <pre>EXEC sp_addlogin 'Public_', 'Public_1', 'DB_Books'</pre> <pre>use DB_Books</pre> <pre>EXEC sp_adduser 'Public_', 'Public_'</pre> <p>В БД DB_Books создан пользователь Public_ с ролью в БД DB_Books public.</p>
Deny (отрицание)	<p>Этот пример запрещает несколько системных привилегий для нескольких пользователей.</p> <p>Пользователи не могут использовать системные привилегии CREATE DATABASE or CREATE TABLE, если они не наделены ими через команду GRANT.</p> <p>Пример:</p> <pre>DENY CREATE DATABASE, CREATE TABLE</pre> <pre>TO Mary, John, [Corporate\BobJ]</pre> <pre>DENY SELECT, INSERT, UPDATE, DELETE ON authors TO</pre> <pre>Mary, John, Tom</pre>
Grant (предоставлять)	<p>This example grants multiple statement permissions to the users Mary and John, and the Corporate\BobJ Windows NT group.</p> <pre>GRANT CREATE DATABASE, CREATE TABLE</pre> <pre>TO Mary, John, [Corporate\BobJ]</pre> <p>Назначение разрешения на выборку (SELECT) для роли PUBLIC в таблице Authors:</p> <pre>GRANT SELECT ON Authors TO public</pre>
Revoke (отменять)	<p>This example revokes multiple statement permissions from multiple users.</p> <pre>REVOKE CREATE TABLE, CREATE DEFAULT</pre> <pre>FROM Mary, John</pre> <p>This example removes the denied permission from Mary and, through the SELECT permissions applied to the Budget role, allows Mary to use the SELECT statement on the table.</p> <pre>REVOKE SELECT ON Budget_Data TO Mary</pre>

Создание пользователей для доступа к серверу через утилиту Microsoft SQL Server Management Studio

Создадим новую учетную запись для нашей базы данных **University**. Для этого выберите в Обозревателе объектов раздел **Безопасность/Имена входа**. Добавьте новое имя входа – **Proba**, установите опцию **Проверка подлинности SQL Server**, присвойте свой пароль, примените к выбранной базе данных, установите язык по умолчанию – русский.

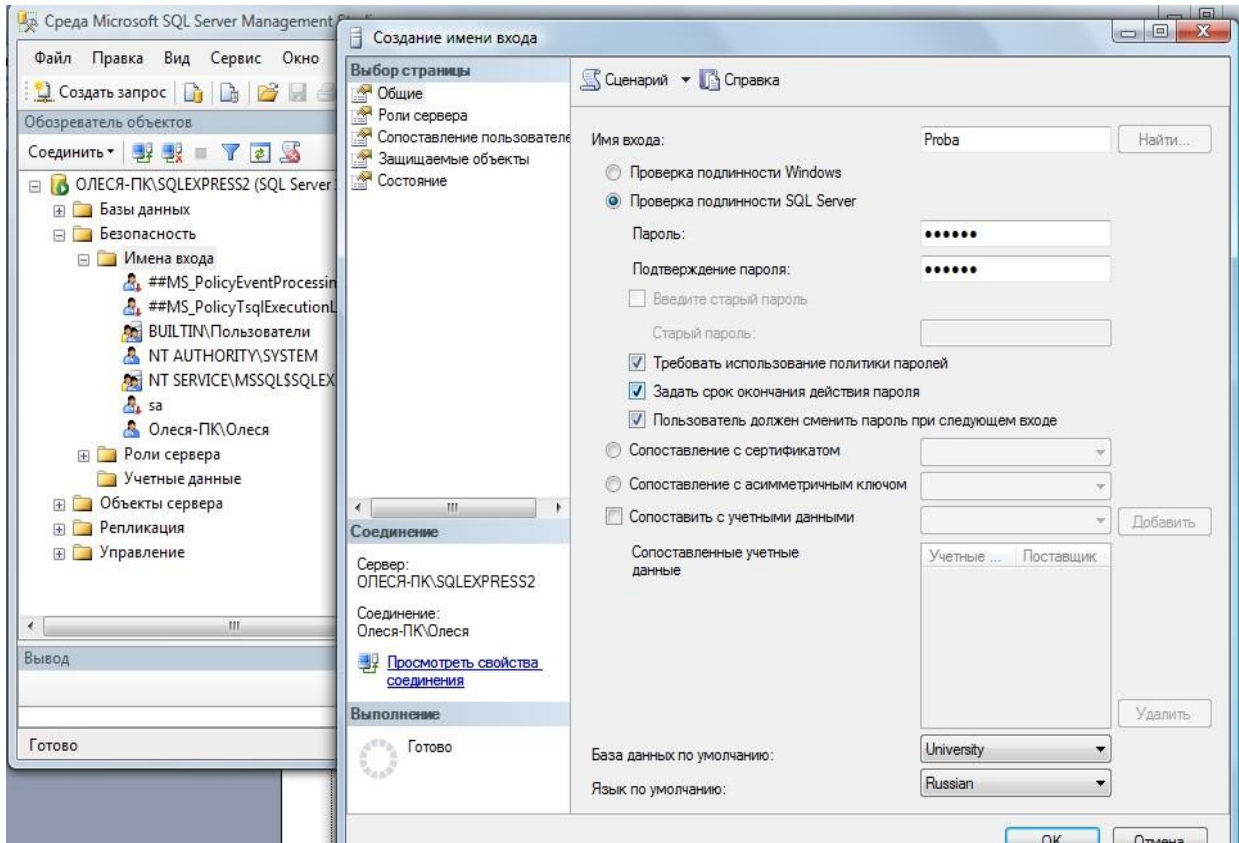


Рис. 2.1. Раздел Безопасность (Security) для работы с пользователями и создание нового пользователя (при SQL Server аутентификации нужно снять галочки с **Enforce password policy**)

Прежде чем добавлять нового пользователя просмотрите его назначенные серверные роли. Для этого в этом же окне выберите раздел **Роли сервера**. Установите для пользователя **Proba** роль **sysadmin**.

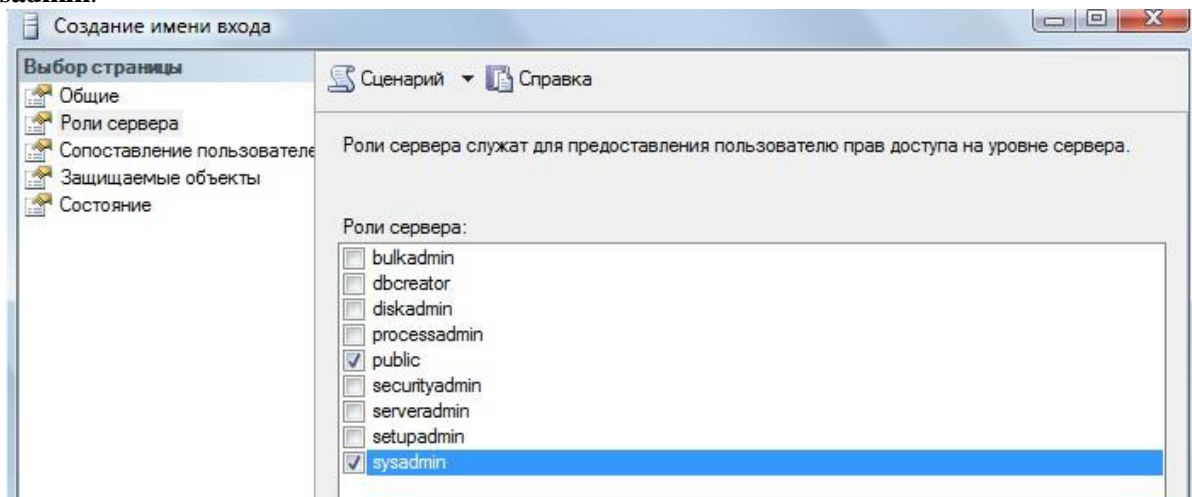


Рис. 2.2. Настройка серверной роли для нового пользователя (весь список серверных ролей с их привилегиями в конце работы)

Далее просмотрите раздел **Сопоставление пользователя**. Установите для базы данных **University** у пользователя **Proba** права доступа **Db_owner**, означающие, что пользователь может выполнять **любые действия с БД**. Ниже перечислены все возможные варианты прав доступа.

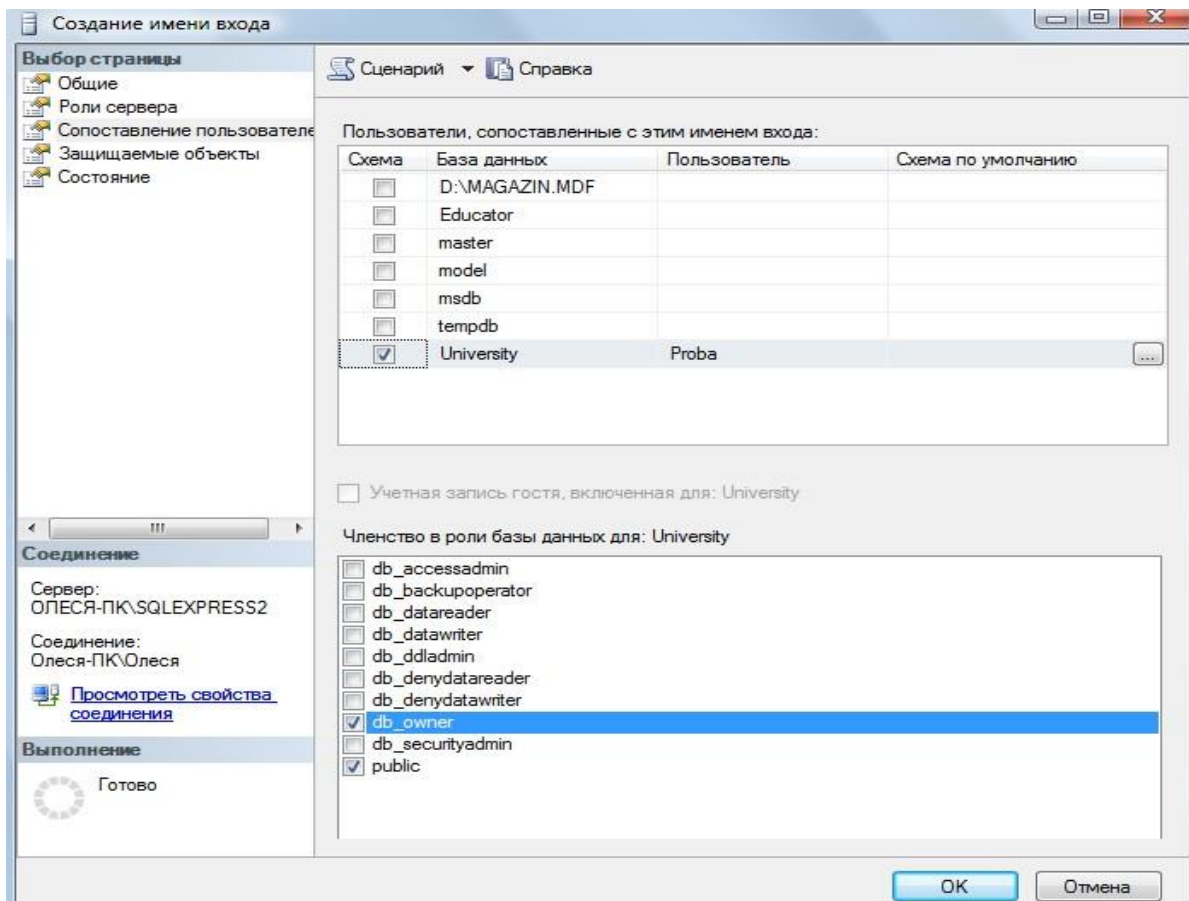


Рис. 2.3. Настройка роли базы данных для нового пользователя (весь список ролей базданных с их привилегиями ниже)

Перечень ролей БД:

Public – минимальные права доступа к БД (на просмотр)

Db_owner – может выполнять любые действия с БД

Db_accessadmin – добавляет и удаляет пользователей БД

Db_securityadmin – управляет ролями в БД и разрешениями на запуск команд и работу с объектами БД

Db_ddladmin – добавляет, изменяет и удаляет объекты БД

Db_backupoperator – осуществляет резервное копирования БД

Db_dataSTUDENT – может просматривать все данные в каждой таблице в БД

Db_datawriter - может добавлять, удалять и изменять данные в каждой таблице в

БД

Db_denydataSTUDENT – запрет на просмотр всех данных в каждой таблице в БД

Db_denydatawriter - запрет на добавление, удаление и изменение всех данных в каждой таблице в БД

Далее перейдите на раздел **Состояние**. Установите опции **Разрешение к подключению к ядру СУБД** – предоставить и **имя входа** включить.

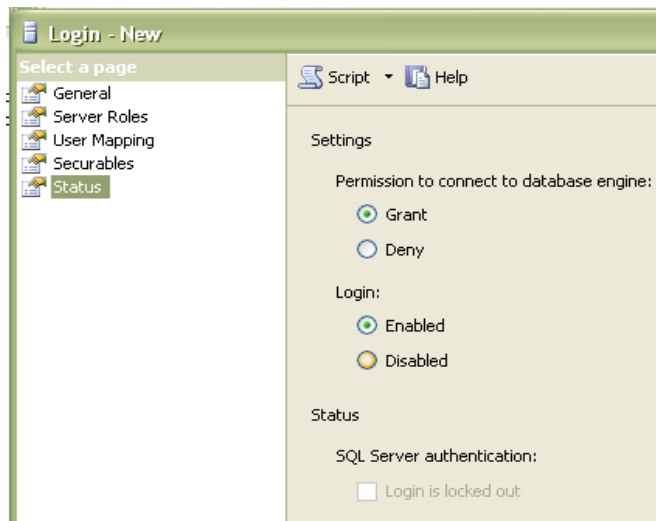


Рис. 4.4. Разблокирование создаваемой учетной записи

После нажатия на <ОК> в БД появится пользователь **Proba** с правами собственника БД, который может выполнять все манипуляции с БД **University**.

Откройте в окне обозревателя объектов БД **University** и перейдите на вкладку **Безопасность**, там вы найдете только что созданного пользователя.


Создание ролей программно

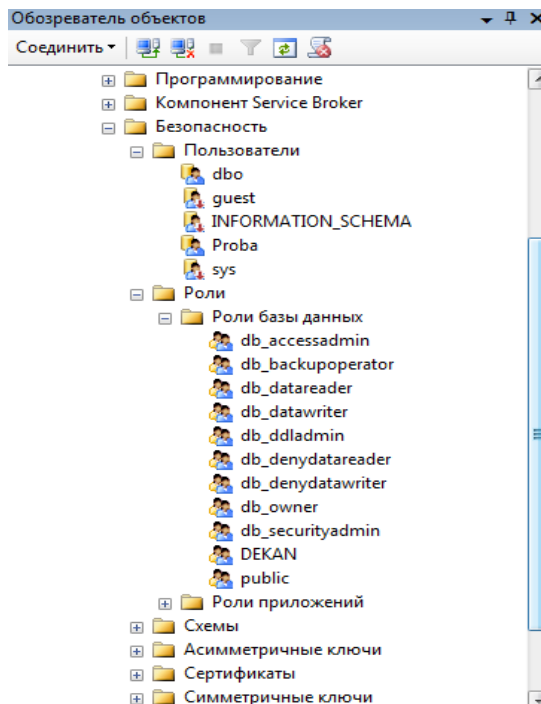
Для упрощения управления правами доступа в системе создаются **роли**, которые затем можно назначать **группе пользователей**.

Создадим для нашего примера роли декана (**ДЕКАН**) и студента (**STUDENT**).

Пример создания роли декана:

USE University --сделать текущей БД University
EXEC sp_addrole 'ДЕКАН'

Эти операторы набрать на странице, вызванной нажатием кнопки <Создатьзапрос>. Для запуска команд на выполнение нажать  **Выполнить**. Сохраните запрос.



Повторный запуск тех же команд сгенерирует ошибки типа «В БД уже существует роль DEKAN».

Чтобы посмотреть, что роль добавлена, откройте вкладку **Безопасность/Роли/Роли базы данных**.

Пример создания роли студента:

```
USE University --сделать текущей БД university
EXEC sp_addrole 'STUDENT'
```

Декан должен обладать правами на **чтение, удаление, изменение, добавление во все таблицы БД University**, а также должен иметь возможность запускать на исполнение процедуры и функции БД University. Поэтому роли декана из системных привилегий назначаем **EXECUTE**, а из привилегий доступа к объектам назначаем **DELETE, INSERT, UPDATE, SELECT**.

Студент должен обладать правами **на чтение из таблиц**. Поэтому роли читателя из привилегий доступа к объектам назначаем **SELECT**.

Оператор представления привилегий

Синтаксис:

```
GRANT <привилегия>, ...
ON <объект >, ...
TO <имя>
[WITH grant option];
```

Атрибут **WITH GRANT OPTION** дает право пользователю самому раздавать права, которые он получил.

С помощью оператора **GRANT** для каждого пользователя формируется список привилегий, привилегии управляют работой сервера данных с точки зрения защиты данных. Выполнению каждой транзакции предшествует проверка привилегий пользователя, сеанс которого породил транзакцию.

Например (не выполнять):

GRANT select, update (Sales, num) ON Sales_data TO user1 WITH GRANT OPTION

Пользователь, предоставивший привилегию другому, называется **грантор** (grantor — предоставитель). Привилегия является предоставляемой, если право на нее можно предоставить другим пользователям.

PUBLIC — имя роли, которую получает пользователь при добавлении в список пользователей конкретной БД, включает в себя минимальный набор прав на чтение данных из таблиц и представлений в БД.

Для примера (немного забегаая вперед) создадим таблицу **Disciplinu**. Без объяснения синтаксиса выполните следующий sql-запрос:

```
USE University --сделать текущей БД university  
create table Disciplinu (  
    Kod_Disciplinu int NOT NULL primary key,  
    name_Disciplinu nchar(30) NULL,  
    kol_chasov int NULL  
);
```

Выполните код и обновите вкладку **Таблицы**. Вы должны увидеть созданную таблицу для сохранения данных о всех дисциплинах. Эта таблица пока пустая с тремя столбцами **Kod_Disciplinu, name_Disciplinu, kol_chasov**.

Роль декана названа **DEKAN**. Операторы назначения прав доступа для этой роли представлены ниже:

```
GRANT DELETE, INSERT, UPDATE, SELECT ON Disciplinu TO DEKAN  
GRANT EXECUTE TO DEKAN
```

Роль студента названа **STUDENT**. Операторы назначения прав доступа для этой роли представлены ниже:

```
GRANT SELECT ON Disciplinu TO STUDENT
```

Примените роли декана и студента к созданной таблице.

Создание пользователей с определенной ролью

Пример создания декана Ivanov_Dek и присвоения ему роли:

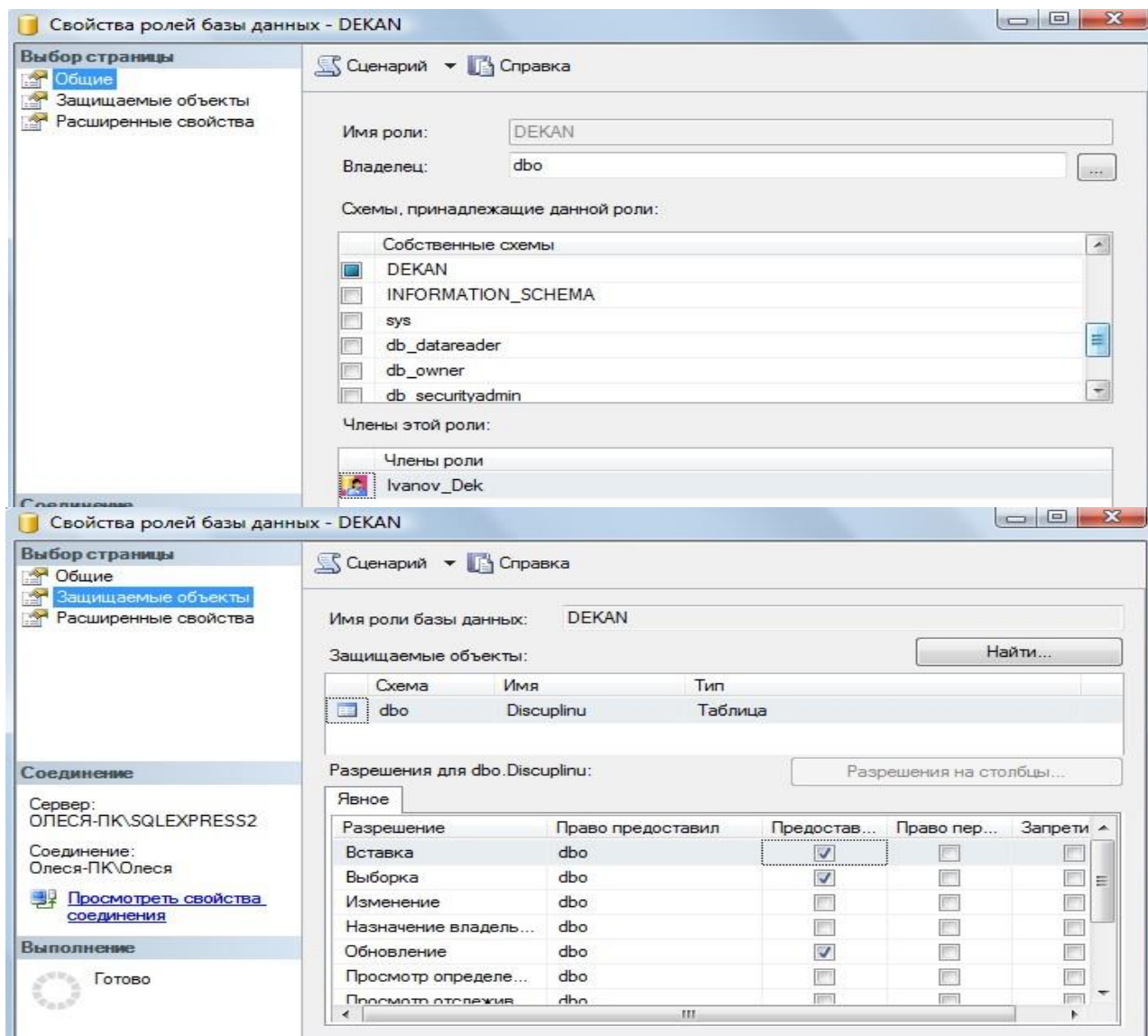
```
EXEC sp_addlogin 'Ivanov_Dek','Ivanov', 'University'  
use University  
EXEC sp_adduser 'Ivanov_Dek','Ivanov_Dek'  
EXEC sp_addrolemember 'DEKAN', 'Ivanov_Dek'
```

Пример создания студент Petrov_Stud и присвоения роли:

```
EXEC sp_addlogin 'Petrov_Stud','Petrov', 'University'  
use University  
EXEC sp_adduser 'Petrov_Stud','Petrov_Stud'  
EXEC sp_addrolemember 'STUDENT', 'Petrov_Stud'
```

Выполните команды. Перейдите в окне Обозреватель объектов на **Роли/Роли базы данных/Dekan** и просмотрите его свойства. Просмотрите назначенные общие свойства, защищаемые объекты и расширенные свойства.

Самостоятельно просмотрите свойства роли базы данных **Student**. Просмотрите назначенные общие свойства, защищаемые объекты и расширенные свойства.



Оператор отмены привилегий

Синтаксис отмены привилегий:

REVOKE [with grant option]

< привилегии >,...

ON < объект >,...

FROM <имя_пользователя>;

Предложение **with grant option** сохраняет за пользователем перечисленные привилегии, но отменяет его право передавать их кому-либо другому.

Пример:

REVOKE SELECT ON Disciplinu FROM STUDENT

Выполните команду.

Оператор изымания роли у пользователя

Revoke <список ролей> from <список пользователей>.

Пример:

use University

EXEC sp_droprolemember 'STUDENT', 'Petrov_Stud'

Выполните команду и просмотрите результат.

Задание для практической работы №16-17

- Создать файл базы данных, согласно номеру варианта, выданного в практической работе №1 с помощью sql-команды.
- Создать резервную копию базы данных.
- Определить **2-3** должностных лица, которые смогут работать с таблицами БД. Для каждого должностного лица определить набор привилегий, которыми он может пользоваться.
- В утилите **SQL Server Management Studio** создать под каждое должностное лицо соответствующую роль, наделить эту роль определенными привилегиями. Далее создать по одному пользователю на каждую должность и присвоить им соответствующие роли.
- Сохранить последовательно SQL-операторы с указанием заданий в файле с названием **ФамилияСтудента_Лаб_2**.
- Создать текстовый отчет, в котором отобразить sql-команды разработанных запросов и скриншоты результатов работы из СУБД **SQL Server Management Studio**.

Практическая работа №18. Экспорт данных базы в документы пользователя

Цель работы

Изучить используемый в реляционных СУБД оператор извлечения данных из таблиц SELECT и выполнение группировки и сортировки данных. Изучить синтаксис языка модификации данных. Научится использовать встроенные функции в запросах.

Исходные данные

Исходными данными является индивидуальное задание и результат предыдущих практических работ.

Используемые программы

Программа SQL Server Managment Studio.

Задание

Практическую работу следует выполнять в следующем порядке:

1. Изучить синтаксис создания запросов с использованием функций, группировки и сортировки данных, язык манипулирования данными на примерах запросов, использовать встроенные функции к учебной базе данных "University.mdf".
2. Выполнить в окне "SQL Editor" 41 запросов к базе данных "University.mdf", согласно приведенным в практической работе образцам выполнения запросов и сохранить их в файле "Lab7.sql" в своей рабочей папке.

Тема 1. Создание запросов с использованием функций

Функции SQL подобны любым другим операторам языка в том смысле, что они производят действия с данными и возвращают результат в качестве своего значения. Функции имеют тип, который определяется типом возвращаемого значения, поэтому можно говорить о числовых, строковых, временных функциях и т. д. От обычных операторов функции отличаются форматом представления:

имя_функции[(аргумент[, аргумент]...)]

Этот формат допускает, что функции могут иметь ноль, один или более аргументов, причем при отсутствии аргументов круглые скобки не используются.

Имеется два основных класса функций SQL: встроенные и определяемые пользователем.

Встроенными являются функции, предопределенные в SQL. Ко второму классу относятся функции, которые пишутся пользователями на специальном языке, обеспечивающем использование всех возможностей SQL. Каждая СУБД использует для этого свой собственный язык.

SQL Server содержит множество встроенных функций, а также поддерживает создание определяемых пользователем функций.

В SQL определено множество встроенных функций различных категорий. На этом уроке мы рассмотрим:

– агрегатные (или групповые) функции, оперирующие значениями столбцов множества строк и возвращающие одно значение;

– функции одной строки, использующие в качестве аргументов значения столбцов одной строки и возвращающие одно значение.

Встроенные функции (Transact-SQL)

SQL Server содержит множество встроенных функций, а также поддерживает создание определяемых пользователем функций. Категории встроенных функций перечислены на этой странице.

Типы функций

Функция	Описание
Функции, возвращающие наборы строк.	Возвращают объект, который можно использовать так же, как табличные ссылки в SQL-инструкции.
Агрегатные функции	Обрабатывают коллекцию значений и возвращают одно результирующее значение.
Ранжирующие функции	Возвращают ранжирующее значение для каждой строки в секции.
Скалярная функция (описывается далее)	Обрабатывают и возвращают одиночное значение. Скалярные функции можно применять везде, где выражение допустимо.

Скалярные функции

Категория функции	Описание
Функции конфигурации	Возвращают сведения о текущей конфигурации.
Функции преобразования	Поддержка приведения и преобразования типов данных.
Функции работы с курсорами	Возвращают сведения о курсорах.
Функции и типы данных даты и времени	Выполняют операции над исходными значениями даты и времени, возвращают строковые и числовые значения, а также значения даты и времени.
Логические функции	Выполнение логических операций.
Математические функции	Выполняют вычисления, основанные на числовых значениях, переданных функции в виде аргументов, и возвращают числовые значения.
Функции метаданных	Возвращают сведения о базах данных и объектах баз данных.
Функции безопасности	Возвращают данные о пользователях и ролях.
Строковые функции	Выполняют операции со строковым (char или varchar) исходным значением и возвращают строковое или числовое значение.
Системные функции	Выполняют операции над значениями, объектами и параметрами экземпляра SQL Server и возвращают сведения о них.
Системные статистические функции	Возвращают статистические сведения о системе.
Функции обработки текста и изображений	Выполняют операции над текстовыми или графическими исходными значениями или столбцами и возвращают сведения о

Агрегатные функции

Аргументами агрегатных функций могут быть как столбцы таблиц, так и результаты выражений над ними. Агрегатные функции и сами могут включаться в другие арифметические выражения. В стандарте SQL определены следующие виды

агрегатных функций: унарные, бинарные, инверсного распределения, гипотетические функции множеств.

Мы будем рассматривать только определенные в стандарте SQL унарные агрегатные функции. Их перечень представлен в табл. 1.1. Конкретные СУБД расширяют этот список.

AVG - среднее

MIN - минимум

CHECKSUM_AGG - Возвращает контрольную сумму значений в группе.

Значения NULL не учитываются.

SUM - сумма

COUNT - количество

STDEV – среднее квадратическое отклонение

COUNT_BIG - Возвращает количество элементов в группе.

STDEVP - Возвращает статистическое стандартное отклонение всех значений в указанном выражении.

GROUPING - Указывает, является ли указанное выражение столбца в списке GROUP BY статистическим или нет. В результирующем наборе функция GROUPING возвращает 1 (статистическое выражение) или ноль (нестатистическое выражение).

VAR - дисперсия

GROUPING_ID - Представляет собой функцию, которая вычисляет уровень группирования.

VARP - Возвращает статистическую дисперсию для заполнения всех значений в указанном выражении.

MAX - максимум

Общий формат унарной агрегатной функции следующий:

имя_функции([ALL | DISTINCT] выражение) [FILTER (WHERE условие)]

где **DISTINCT** указывает, что функция должна рассматривать только различные значения аргумента, а **ALL** — все значения, включая повторяющиеся (этот вариант используется по умолчанию). Фраза **FILTER** позволяет дополнительно отобрать строки таблицы, столбец которой используется в качестве аргумента функции.

Агрегатные функции применяются во фразах **SELECT** и **HAVING**. Здесь мы рассмотрим их использование во фразе **SELECT**. В этом случае выражение в аргументе функции применяется ко всем строкам входной таблицы фразы **SELECT**. Кроме того, во фразе **SELECT** нельзя использовать и агрегатные функции, и столбцы таблицы (или выражения с ними) при отсутствии фразы **GROUP BY**, которую мы рассмотрим в теме 2.

Функция COUNT

Функция **COUNT** имеет два формата. В первом случае возвращается количество строк входной таблицы, во втором случае — количество значений аргумента во входной таблице:

COUNT(*)

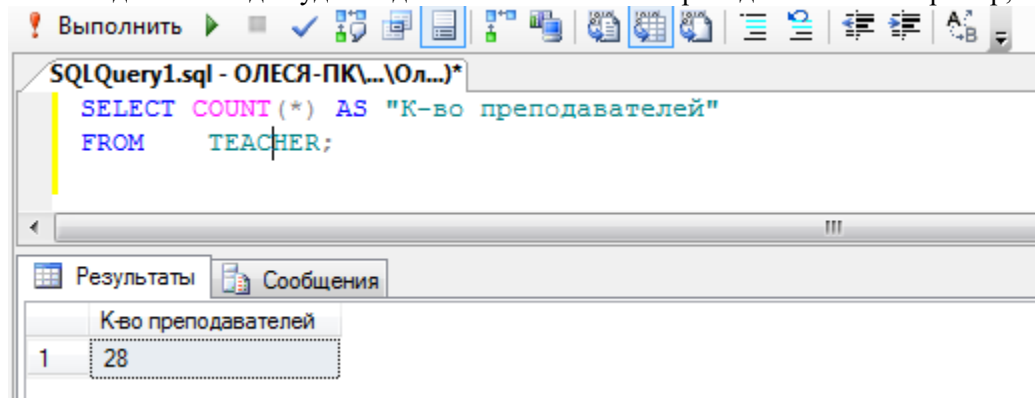
COUNT([DISTINCT | ALL] выражение)

Простейший способ использования этой функции - подсчет количества строк в таблице (всех или удовлетворяющих указанному условию). Для этого используется первый вариант синтаксиса.

Создайте новый запрос, введите sql-запрос, выполните его, сохраните его в рабочую папку ЛАБ7_SQL под именем 1.sql.

**Запрос 1. Информация о скольких преподавателях имеется в базе данных?
SELECT COUNT(*) AS "К-во преподавателей"
FROM TEACHER;**

Чтобы выполнить sql-команду нажмите на панели редактора кнопку Выполнить. В результате выполнения данного кода будет подсчитано кол-во всех преподавателей. Например,



Самостоятельно создать запрос 2. Сколько ассистентов не имеют телефонов?

Запросы с агрегатными функциями можно строить и с использованием нескольких таблиц, так как входная таблица и в этом случае будет только одна.

Самостоятельно создать запрос 3. Сколько кафедр на факультете математики и информатики?

Во втором варианте синтаксиса функции COUNT в качестве аргумента может быть использовано имя отдельного столбца. В этом случае подсчитывается количество либо всех значений в этом столбце входной таблицы, либо только неповторяющихся (при использовании ключевого слова DISTINCT).

Запрос 4. На скольких различных должностях работают преподаватели кафедры «Компьютерные системы и сети»?

**SELECT COUNT(DISTINCT DOLGNOST)
FROM KAFEDRA d, TEACHER t
WHERE d.KOD_KAFEDRU = t.KOD_KAFEDRU AND
LOWER(d.NAME_KAFEDRU) = 'Компьютерные системы и сети';**

Функция SUM

Эта агрегатная функция подсчитывает сумму значений аргумента для всех строк входной таблицы. Аргумент должен иметь числовой тип или быть временным промежутком. В качестве аргумента может выступать имя столбца или выражение над столбцами входной таблицы. В этой функции также допускается использовать ключевые слова DISTINCT и ALL. Приведем примеры.

Запрос 5. Какая суммарная ставка всех ассистентов?

**SELECT SUM(Salary)
FROM TEACHER
WHERE LOWER(DOLGNOST) = 'ассистент';**

Функция AVG

Агрегатная функция AVG подсчитывает среднее значение аргумента для всех строк входной таблицы. Аргумент должен иметь числовой тип или быть временным промежутком. В качестве аргумента может выступать имя столбца или выражение над столбцами входной таблицы. Допускается использовать ключевые слова DISTINCT и ALL. Приведем ряд примеров.

Самостоятельно создать запрос 6. Какая средняя ставка среди всех преподавателей?

Самостоятельно создать запрос 7. Какое среднее значение ставки в вузе?

В данном запросе используйте ключевое слово DISTINCT, чтобы применить AVG не ко всем имеющимся в таблице TEACHER ставкам, а только к различным значениям ставки.

Функции MIN и MAX

Эти функции позволяют находить максимальное (MAX) и минимальное (MIN) значения аргумента для всех строк входной таблицы. Хотя и в этом допускается использование ключевых слов DISTINCT и ALL, они не оказывают влияния на результат. Аргумент этих функций может быть любого типа, для которого определено упорядочение, то есть числовой, строковый и временной.

Запрос 8. Какова максимальная зарплата преподавателя ?

```
SELECT MAX(Salary + Rise)  
FROM TEACHER;
```

Самостоятельно создать запрос 9. Когда в последний раз (максимальная дата приема на работу) принимали на работу преподавателя на кафедре информатики?

Выражения с использованием агрегатных функций

Агрегатные функции не только могут иметь выражение в своем аргументе, но сами могут использоваться в выражениях.

Запрос 10. Вывести процентное соотношение суммарной ставки к суммарной зарплате и наоборот.

```
SELECT SUM(Salary)*100/SUM(Rise) AS "Процент зарплаты к зарплате",  
SUM(Rise)*100/SUM(Salary) AS "Процент зарплаты к ставке"  
FROM TEACHER;
```

Однорочные функции

Напомним, что эти функции используют в качестве аргумента одно значение (одного столбца одной строки таблицы) и возвращают в качестве своего результата также единственное значение. Мы рассмотрим эти функции по типам их аргументов.

Строковые функции

Эти функции используют в качестве аргумента строку символов и в качестве результата возвращают также символьную строку. Стандарт SQL предлагает варианты таких функций и для двоичных строк.

Функции UPPER, LOWER

Эти функции мы уже рассматривали и многократно использовали. Они имеют следующий формат:

```
UPPER(строка)
```


LOWER(строка)

Приведем для них примеры .

**Запрос 11. Вывести фамилии всех преподавателей прописными буквами.
SELECT UPPER(NAME_TEACHER) AS "Все прописные"
FROM TEACHER;**

Аналогично можно вывести все фамилии преподавателей строчными буквами.

Числовые функции над числами

Эти функции возвращают числовые значения на основании заданных в аргументе значений того же типа. Числовые функции используются для обработки данных, а также в условиях их поиска. Стандарт SQL предлагает ряд числовых функций с очевидной семантикой. Часть функций перечислены ниже:

ABS абсолютное значение

DEGREES Возвращает для значения угла в радианах соответствующее значение в градусах.

RAND – Возвращает псевдослучайное значение типа float от 0 до 1. EXP экспонента

ROUND - Возвращает числовое значение, округленное до указанной длины или точности.

FLOOR Возвращает наибольшее целое число, меньшее или равное указанному числовому выражению.

LOG логорифм

SIN - синус

LOG10 десятичный логорифм

SQRT – корень квадратный

PI число 3.14 SQUARE –

квадрат числа

POWER Возвращает значение указанного выражения, возведенное в заданную степень.

TAN - тангенс

Особой функцией является WIDTH_BUCKET, с помощью которой можно легко строить гистограммы:

WIDTH_BUCKET(число, минимум, максимум, количество)

Некоторые СУБД расширяют приведенный выше набор функций, включая другие числовые функции, например вычисления обычных и гиперболических тригонометрических функций.

Временные функции

Эти функции используют в качестве аргумента типы даты, времени, временной отметки или временного промежутка. Тип возвращаемого значения не всегда соответствует типу аргумента.

Функции даты и времени в Transact-SQL делятся на Функции,

получающие значения системной даты и времени Функции,

получающие компоненты даты и времени

Функции, получающие значения даты и времени из их компонентов Функции,

получающие разность даты и времени

Функции, изменяющие значения даты и времени

Функции, устанавливающие или получающие формат сеанса
Функции, проверяющие значения даты и времени.

Функции, получающие компоненты даты и времени.

Функция извлекает из операнда указанный компонент и возвращает его в виде
чис

ла. **DATENAME (datepart , date)**

Здесь date - это выражение временного типа, а datepart - временная единица, которая может иметь одно из следующих значений: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND и т.д.

DATEPART (datepart,date) - Возвращает целое число, представляющее указанный компонент datepart указанной даты date.

DAY (date) - Возвращает целое число, представляющее день указанной даты date.

MONTH (date) - Возвращает целое число, представляющее месяц указанной даты date.

YEAR (date) - Возвращает целое число, представляющее год указанной даты date.

Рассмотрим пример.

Запрос 12. Вывести фамилии всех преподавателей родившихся в 1979 году.

```
SELECT Name_teacher, BIRTHDAY
```

```
FROM TEACHER
```

```
WHERE DATENAME(YEAR, BIRTHDAY)=1979;
```

Функции, получающие значения системной даты и времени

Функция CURRENT_TIMESTAMP - Возвращает значение типа datetime2(7), которое содержит дату и время компьютера, на котором запущен экземпляр SQL Server. Смещение часового пояса не включается.

Эта функция возвращает текущую дату. Аргументов она не имеет.

Функция GETDATE () Возвращает значение типа datetime2(7), которое содержит дату и время компьютера, на котором запущен экземпляр SQL Server. Смещение часового пояса не включается.

Функция GETUTCDATE () Возвращает значение типа datetime2(7), которое содержит дату и время компьютера, на котором запущен экземпляр SQL Server. Возвращаемые дата и время отображаются в формате UTC.

Многие СУБД существенно расширяют список функций, оперирующих датой и временем. Далее мы приведем некоторые из важных функций этого типа, которые используются в Oracle. Напоминаем, что тип DATE в Oracle содержит в себе как дату, так и время.

Функции, получающие значения даты и времени из их компонентов

Функция DATEADD (datepart, number , date) Возвращает новое значение datetime, добавляя интервал к указанной части datepart заданной даты date.

Добавляет к дате, указанной в первом аргументе, количество месяцев второго аргумента.

Dateadd (компонент, кол-во , дата)

Здесь кол-во - это количество прибавляемых лет, месяцев, дней и т.д., а компонент - временная единица, которая может иметь одно из следующих значений: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND.

Например, DATEADD(month, 1, '2006-08-30')

Запрос 13. Осуществить пересчет даты приема на работу преподавателя на фамилию начинающуюся на букву С в сторону увеличения на 3 месяца.

```
SELECT NAME_TEACHER, DATA_HIRE AS ' Дата приема ',  
DATEADD(month, 3, DATA_HIRE) AS ' Плюс 3 месяца '  
FROM TEACHER  
WHERE (NAME_TEACHER) LIKE 'С%';
```

Функция EOMONTH

EOMONTH (start_date [, month_to_add])

Возвращает дату последнего дня того месяца, который указан в аргументе. Обычно используется для определения, сколько дней осталось до конца месяца.

LAST_DAY(дата)

Функция DATEDIFF

DATEDIFF (datepart , startdate , enddate)

Возвращает количество пересеченных границ (целое число со знаком), указанных аргументом datepart, за период времени, указанный аргументами startdate и enddate.

Запрос 14. Например, если вы хотите узнать, сколько месяцев уже проработал Статывка, можно выполнить такой запрос:

```
SELECT 'Статывка проработал ' ||  
ROUND(DATEDIFF(month,GETDATE(), DATA_HIRE),1) ||  
' месяцев' AS "Стаж Статывки"  
FROM TEACHER  
WHERE NAME_TEACHER LIKE 'Статыв%';
```

Функция NEXT_DAY

Возвращает ближайшую к первому параметру дату, в которой название дня недели совпадает с указанным во втором параметре.

NEXT_DAY(дата, день_недели)

Функции преобразования

Стандарт SQL предлагает единственную функцию преобразования данных из одного типа в другой — это функция CAST.

Функция CAST и CONVERT

Производит преобразование выражения, заданного первым аргументом, в тип, заданный вторым аргументом. Преобразование допускается только для определенных пар типов данных.

CAST (expression AS data_type [(length)])

CONVERT (data_type [(length)] , expression [, style])

Например

CAST(10.3496847 AS money)

CAST(10.6496 AS int)

Тема 2. Группировка и сортировка

В этом уроке мы рассмотрим еще три фразы предложения SELECT, а именно:

HAVING, GROUP BY и ORDER BY.

Первая из них позволяет группировать строки таблицы и применять к созданным группам агрегатные функции.

Рассмотрим простейшие варианты группировки. Фраза HAVING используется вместе с фразой GROUP BY и позволяет формулировать условия на группах строк для дополнительного отбора.

Наконец, фраза ORDER BY позволяет сортировать строки результирующей таблицы.

Запросы с группировкой строк

Часто при создании отчетов появляется необходимость в формировании промежуточных итоговых значений, то есть относящихся к данным не всей таблицы, а ее частей.

Именно для этого предназначена фраза GROUP BY. Она позволяет все множество строк таблицы разделить на группы по признаку равенства значений одного или нескольких столбцов (и выражений над ними).

Фраза GROUP BY должна располагаться вслед за фразой WHERE (если она отсутствует, то за фразой FROM).

Общий синтаксис фразы GROUP BY следующий:

GROUP BY выражение[, выражение]...

При наличии фразы GROUP BY фраза SELECT применяется к каждой группе, сформированной фразой группировки. В этом случае и действие агрегатных функций, указанных во фразе SELECT, будет распространяться не на всю результирующую таблицу, а только на строки в пределах каждой группы. Каждое выражение в списке фразы SELECT должно принимать единственное значение для группы, то есть оно может быть:

- константой;
- агрегатной функцией, которая оперирует всеми значениями аргумента в пределах группы и агрегирует их в одно значение (например, в сумму);
- выражением, идентичным стоящему во фразе GROUP BY;
- выражением, объединяющим приведенные выше варианты.

Рассмотрим возможности фразы GROUP BY, переходя от простых вариантов ее использования к более сложным.

Группировка по одному столбцу

Группировка по значениям одного столбца является самым простым вариантом использования фразы GROUP BY. Приведем примеры.

Запрос 15. Для каждого корпуса подсчитать количество находящихся в нем кафедр.

```
SELECT NUM_KORPUSA AS "Корпус",  
COUNT(*) AS "К-во кафедр"  
FROM KAFEDRA  
GROUP BY NUM_KORPUSA ;
```

Самостоятельно создать запрос 16. Для каждой из должностей указать суммарный фонд заработной платы.

Если в запросе используются фразы и WHERE, и GROUP BY, строки, не удовлетворяющие условию фразы WHERE, исключаются до выполнения группировки. Вследствие этого группировка производится только по тем строкам, которые удовлетворяют условию.

В случае многотабличных запросов сначала производится соединение таблиц, а затем их группировка. Приведем примеры.

Самостоятельно создать запрос 17. Для каждого факультета, расположенного в корпусе 1, вывести количество групп и общее количество студентов по каждой кафедре.

Группировка по нескольким столбцам

SQL позволяет группировать строки таблицы и по нескольким столбцам. В этом случае имена столбцов перечисляются во фразе GROUP BY через запятую.

Запрос 18. Для каждого факультета, расположенного в корпусе 1, вывести сколько учится студентов по каждой группе.

```
SELECT f.Name_faculteta,  
s."GROUP", count(s."GROUP") AS "Кол-во студентов в группе"  
FROM FACULTET f, KAFEDRA d, STUDENT s  
WHERE f.KOD_FACULTETA = d.KOD_FACULTETA AND  
d.KOD_kafedru = s.KOD_kafedru AND  
d.NUM_KORPUSA = '1'  
GROUP BY f.Name_faculteta,s."GROUP";
```

Самостоятельно создать запрос 19. Для каждой кафедры и должности вывести суммарную и среднюю зарплату преподавателей.

Даже при группировке по двум и более столбцам этот вариант фразы GROUP BY обеспечивает только один уровень группировки. Так, приведенный выше запрос обеспечивает только одну итоговую строку для пары значений кафедра-должность.

Использование выражений

Хотя стандарт SQL не допускает группировку по выражениям над столбцами, некоторые СУБД такую возможность предоставляют. В этом случае во фразе SELECT также можно использовать выражение группировки, однако нельзя выводить по отдельности столбцы, участвующие в этом выражении.

Запрос 20. Для каждого значения зарплаты, не превышающего 1500, вывести это значение и количество преподавателей, такую зарплату получающих.

```
SELECT Salary + Rise, COUNT(*)  
FROM TEACHER  
WHERE Salary + Rise <= 1500  
GROUP BY Salary + Rise;
```

Вложение агрегатных функций

Если фраза GROUP BY в запросе отсутствует, то во фразе SELECT нельзя вкладывать агрегатные функции друг в друга. Например, следующий запрос приведет к ошибке:

```
SELECT AVG(MIN(Salary))  
FROM TEACHER;
```

Однако при наличии фразы GROUP BY такое вложение допускается. Оно интерпретируется следующим образом: сначала для каждой группы выполняется вложенная агрегатная функция, затем к полученной таким образом промежуточной таблице применяется внешняя агрегатная функция. Двойное вложение, например MAX(AVG(MIN(Salary))), недопустимо. Приведем пример.

Запрос 21. Вывести среднее значение среди минимальных и максимальных ставок для каждой группы преподавателей, занимающих одну должность, а также минимальное и максимальное значения среди средних ставок.

```
SELECT AVG(MIN(Salary)) AS AVG_MIN,
AVG(MAX(Salary)) AS AVG_MAX,
MIN(AVG(Salary)) AS MIN_AVG,
MAX(AVG(Salary)) AS MAX_AVG
FROM TEACHER
GROUP BY Dolgnost ;
```

Условие отбора групп

Предположим, что нужно вывести номера кафедр, у которых суммарное количество работающих профессоров более 1. Приведенная ниже формулировка запроса является неверной:

```
SELECT KOD_kafedru
FROM TEACHER
WHERE count(dolgnost) > 1 and dolgnost='профессор'
GROUP BY KOD_kafedru;
```

```
-----
WHERE count(dolgnost) > 3 and dolgnost='профессор';
```

*

Ошибка в строке 3;

CRA-00934: групповая функция здесь не разрешена

Дело в том, что фраза WHERE проверяет на соответствие условию строки исходных таблиц, а мы указали в ней агрегатную функцию. Для отбора строк среди полученных групп следует применять фразу HAVING. Она играет такую же роль для групп, что и фраза WHERE для исходных таблиц, и может использоваться лишь при наличии фразы GROUP BY. В предложении SELECT фразы WHERE, GROUP BY и HAVING обрабатываются в следующем порядке.

Фразой WHERE отбираются строки, удовлетворяющие указанному в ней условию. Фраза GROUP BY группирует отобранные строки.

Фразой HAVING отбираются группы, удовлетворяющие указанному в ней условию. В связи с вышесказанным, предыдущий запрос необходимо записать так.

Перепишем тогда запрос так:

Запрос 22. Вывести номера кафедр, у которых суммарное количество работающих профессоров более 1.

```
SELECT KOD_kafedru as "Номер кафедры" ,Count(*) as "Кол-во профессоров
на кафедре"
FROM TEACHER
WHERE dolgnost='профессор'
GROUP BY KOD_kafedru
having count(dolgnost) > 1 ;
```

Использование столбцов группировки во фразе HAVING

Рассмотрим использование во фразе HAVING условий отбора, заданных для группируемых столбцов (или выражений над ними). Для этого усложним предыдущий запрос.

Запрос 23. Вывести названия кафедр факультета математики и информатики, на которых работают один и более профессоров. Указать также количество профессоров и их суммарную зарплату.

```
SELECT d.Name_kafedru, Count(*), SUM(t.salary + t.Rise)
FROM FACULTET f, KAFEDRA d, TEACHER t
WHERE f.KOD_FACULTETA = d.KOD_FACULTETA AND
d.KOD_kafedru = t.KOD_kafedru AND
LOWER(f.Name_faculteta) = 'математики и информатики' AND
LOWER(t.Dolgnost ) = 'профессор'
GROUP BY d.Name_kafedru
HAVING COUNT(*) > 0;
```

Фраза HAVING без фразы GROUP BY

Выше мы указали, что фраза HAVING может использоваться лишь при наличии фразы GROUP BY. Из этого правила синтаксис SQL допускает только одно исключение: когда вся таблица интерпретируется как одна группа. В этом случае в списке фразы SELECT можно использовать только константы, агрегатные функции и выражения над ними. Приведем примеры.

Запрос 24. Если суммарная зарплата всех преподавателей превышает 15 000, вывести их минимальную ставку, максимальную надбавку и суммарную зарплату.

```
SELECT MIN(Salary), MAX(Rise), SUM(Salary + Rise)
FROM TEACHER
HAVING SUM(Salary + Rise) > 15000;
```

При наличии фразы WHERE сначала производится отбор строк согласно ее условию, и только после этого применяется условие фразы HAVING.

Запрос 25. Если суммарная зарплата всех ассистентов превышает 2500, вывести их среднюю ставку, среднюю надбавку и суммарную зарплату.

```
SELECT AVG(Salary), AVG(Rise), SUM(Salary + Rise)
FROM TEACHER
WHERE LOWER(Dolgnost ) = 'ассистент'
HAVING SUM(Salary + Rise) > 2500;
```

На практике фраза HAVING очень редко используется без фразы GROUP BY, из-за чего такая возможность предоставляется не во всех СУБД.

Сортировка результирующих строк

Как мы уже отмечали, строки в таблицах базы данных неупорядочены. Также неупорядочены и строки результирующей таблицы запроса, однако для их упорядочения в предложении SELECT можно воспользоваться фразой ORDER BY. Она сортирует по значению указанных в ней столбцов (и выражений над столбцами) строки результирующей таблицы запроса. Синтаксис этой фразы следующий:

ORDER BY спецификация_сортировки[. спецификация_сортировки]...

где спецификация_сортировки имеет такой синтаксис:

выражение_сортировки [направление_сортировки] [положение_NULL]

Сортировать можно по столбцам (выражениям) тех типов, для которых определены операции сравнения. Это относится, в частности, к символьным строкам, числам и временным значениям. Можно указывать направление сортировки и месторасположения строк, имеющих значение NULL для выражений сортировки.

Далее в этом уроке мы рассмотрим общие способы упорядочения результирующих строк.

Сортировка по столбцу или выражению

Сортировать строки результирующей таблицы запроса можно по отдельным столбцам, совокупности столбцов, а также по одному или нескольким выражениям над столбцами. Ниже рассматриваются все эти варианты.

Сортировка по столбцу

Простейший вариант сортировки - это сортировка по одному из столбцов результирующей таблицы.

Запрос 26. Вывести алфавитный список фамилий профессоров и доцентов.

```
SELECT NAME_TEACHER  
FROM TEACHER  
WHERE LOWER(Dolgnost) = 'профессор' OR  
LOWER(Dolgnost) = 'доцент'  
ORDER BY NAME_TEACHER;
```

Сортировка по выражению над столбцами

Упорядочивать строки можно не только по значению столбца, но и по значению выражения над столбцами.

Запрос 27. Вывести фамилии ассистентов и их зарплату по ее возрастанию.

```
SELECT Name_teacher, Salary + Rise  
FROM TEACHER  
WHERE LOWER(Dolgnost) = 'ассистент'  
ORDER BY Salary + Rise;
```

Направление сортировки

Во всех до сих пор приводимых примерах сортировка производилась в порядке возрастания значений. В SQL такой порядок определен по умолчанию. Однако есть возможность и явно указать направление сортировки с помощью ключевых слов ASC (по возрастанию) и DESC (по убыванию), которые следует располагать после имени сортируемого столбца (выражается).

Запрос 28. Вывести фамилии ассистентов и дату их приема на работу по возрастанию даты.

```
SELECT Name_teacher, Data_hire  
FROM TEACHER  
WHERE LOWER(Dolgnost) = 'ассистент'  
ORDER BY Data_hire ASC;
```

Самостоятельно создать запрос 29. Вывести фамилии доцентов в обратном алфавитном порядке и их зарплату.

Тема 3. Внесение изменений в базу данных

Добавление новых данных

Новые данные добавляются оператором INSERT. Наименьшей единицей информации, которую можно добавить в реляционную базу данных, является одна строка таблицы.

Немного упрощенный синтаксис оператора INSERT имеет вид:

```
INSERT INTO Имя_Таблицы [(Колонка [, Колонка ...])]
{VALUES(<величина> [, <величина> ...]} | <оператор SELECT>;
<величина> = {:Переменная | <константа> | <выражение>
| <функция> | udf([<величина> [, <величина> ...]])
| NULL | USER}
```

<константа> = Число | 'Строка'

```
<функция> = CAST(<величина> AS <тип данных>)
UPPER(<величина>)
GEN_ID(Имя_Генератора, <величина>)
```

<выражение> = SQL выражение, возвращающее единичное значение

В этом описании можно выделить два варианта оператора:

1. Вставка одной строки. Для этого после ключевого слова VALUES в круглых скобках указывают вставляемые величины.

2. Вставка в таблицу нескольких строк, выбранных с помощью оператора SELECT *.

В этой практической работе рассматривается только первый вариант оператора INSERT.

Пример, когда в качестве вставляемых величин применены константы:

```
INSERT INTO Person(Pr_ID, Pr_LastName, Pr_FirstName)
VALUES(150, 'Иванов', 'Петр');
```

Удаление существующих данных

Для удаления строк из таблицы используется оператор DELETE. Вот его упрощенный синтаксис:

```
DELETE FROM Имя_Таблицы
[WHERE <условие поиска>];
```

<условие поиска> = как в операторе SELECT

Если не использовать предложение WHERE, то будут удалены все строки в таблице.

-- Удаление всех служащих:

```
DELETE FROM Employee;
```

-- Удаление всех людей с номерами 150 и больше:

```
DELETE FROM Person WHERE Pr_ID >= 150;
```

Отбирать строки для удаления не обязательно только на основании содержимого этих строк. Можно составить условие для удаляемых строк, опираясь на данные из других таблиц. Для составления таких условий необходимо сначала изучить оператор SELECT.

Обновление существующих данных

Оператор UPDATE обновляет значения одного или нескольких столбцов в выбранных строках одной таблицы. Строки для обновления указываются в предложении WHERE. Если пропустить предложение WHERE, то изменятся все строки таблицы.

```
UPDATE Имя_Таблицы
SET Колонка = <величина>[,
Колонка = <величина>...]
[WHERE <условие поиска>]
<величина> = { Колонка | :Переменная | <константа>
| <выражение> | <функция>
| udf((<величина> [, <величина> ...])) | NULL | USER}
<выражение> = SQL выражение, возвращающее единичное значение
<условие поиска> = как в операторе SELECT
```

Примеры:

```
-- Увеличить зарплату всем служащим на 10%:
```

```
UPDATE Employee
SET Salary = 1.1*Salary;
```

```
/* Увеличить зарплату всем служащим, которые имеют зарплату
меньше 10000 на 15%: */
```

```
UPDATE Employee
SET Salary = 1.15*Salary;
WHERE Salary <= 10000;
```

Отбирать строки для изменения, как и для удаления, можно с использованием подчиненного запроса SELECT, который позволит учитывать в условии поиска изменяемых строк данные из других таблиц.

Например, можно выполнить такой запрос: увеличить зарплату на 10% всем служащим, работающим в отделе продаж, которые обслужили за последний месяц клиентов больше чем в полтора раза, чем в среднем по их отделу.

Задание

Добавление новых строк

Предложение INSERT вставляет строки в таблицу базы данных. Есть три разновидности этой команды:

```
INSERT VALUES
INSERT SELECT
INSERT DESALT VALUES
```

Первая из них производит вставку в таблицу явно заданной строки, вторая разновидность – вставку группы строк, выбранных в результате выполнения запроса, а третья — вставку значений по умолчанию.

Вставка отдельных строк

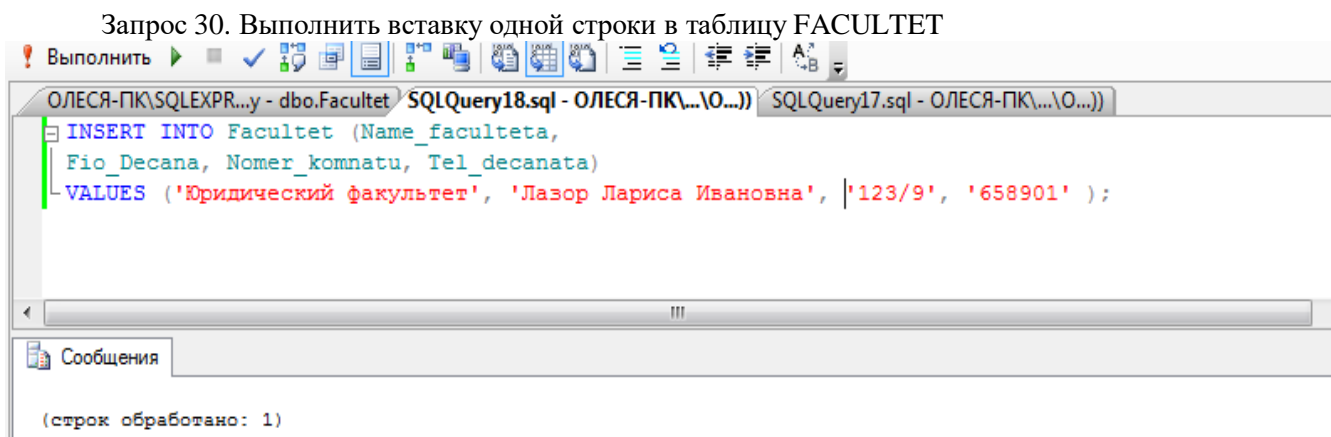
Предложение INSERT... VALUES выполняет вставку в таблицу одной строки. Его удобно использовать для небольших операций, когда в таблицу нужно вставить несколько строк. Синтаксис этого предложения следующий:

```
INSERT INTO имя_таблицы [<имя_столбца[, имя_столбца]...])  
VALUES (значение[, значение]...);
```



Указание вставляемых столбцов

Этот формат предполагает указания имени таблицы, в которую производится вставка, списка имен столбцов, в которые будут вставляться значения, и списка собственно вставляемых в строку значений. При этом следует придерживаться следующих правил:

- вставляемые данные должны согласовываться с типами данных указанных столбцов;
- размеры данных должны соответствовать размерам столбцов;
- порядок данных во фразе VALUES должен соответствовать порядку перечисления столбцов.



В таблицу вставляется строка со значениями, указанными в списке фразы VALUES, причем расположение значений в списке соответствует расположению соответствующих столбцов в списке столбцов таблицы.

Примечание. Чтобы данные были добавлены, не забудьте нажать на кнопки , а затем  для подтверждения внесенных изменений в таблицу. Просмотрите внесенные изменения в таблицу FACULTET.

В этом примере мы перечислили столбцы в том порядке, в каком они были определены при создании таблицы, однако это не обязательно. При желании порядок перечисления имен столбцов в команде INSERT можно изменить.

Список имен столбцов может быть не полным. Можно указывать только те из них, значения которых известны для вставляемой строки. Столбцы, отсутствующие в списке, будут принимать значения NULL для вводимой строки.

Самостоятельно создайте запрос 31. Выполнить вставку одной строки в таблицу KAFEDRA для столбцов name_kafedru, fio_zavkaf, kod_faculteta с данными 'Психологии', 'Иванова', 5.

В этом примере в вводимой строке отсутствуют значения столбцов Nomer_komnatu, Num_korpusa и Tel_kafedru. В базе данных они примут значение NULL.

Поддержка ограничений целостности

Помните, некоторые из столбцов или наборов столбцов могут иметь ограничения целостности PRIMARY KEY и NOT NULL. Такие столбцы не могут принимать значения NULL.

Приведенные выше рассуждения относятся ко всем ограничениям целостности, определенным для таблиц. При попытке ввода данных (как, впрочем, и при обновлении и удалении) СУБД проверяет возможное нарушение объявленных ограничений целостности. И если это так, команда будет отклонена с выдачей соответствующего уведомления.

Использование выражений

В качестве вставляемых значений могут использоваться выражения.

Самостоятельно создайте запрос 32. Ввести в таблицу TEACHER данные (50, 10, ' Капуста Леонид Владимирович', 1271, 1271/3, 'доцент', GETDATE()-1)

Здесь мы указали, что надбавка равна третьей части ставки (1271 / 3), а дата приема на работу на один день меньше текущей даты (CURRENT_DATE -1).

Результат запроса в качестве вставляемого значения

Вместо вставляемого значения можно использовать запрос. Это оказывается очень удобным в том случае, когда вставляемое значение присутствует в базе данных.

Запрос 33. Например, в следующем предложении в качестве фамилии заведующего вновь вставляемой кафедры выбирается фамилия декана факультета «Компьютерных наук и технологий» .

```
INSERT INTO KAFEDRA (name_kafedru, kod_faculteta, fio_zavkaf)  
VALUES ( 'Философии', 5, (SELECT fio_decana FROM FACULTET  
WHERE LOWER(Name_faculteta) = 'международный'));
```

Обновление существующих данных

Целью предложения обновления является изменение значений отдельных столбцов всех или удовлетворяющих указанному условию строк таблицы. Упрощенный синтаксис предложения следующий:

```
UPDATE имя_таблицы [[AS] синоним]  
SET имя_столбца = выражение[, имя_столбца = выражение]...  
[WHERE условие];
```

Его элементы означают следующее:

имя_таблицы — имя обновляемой таблицы;

синоним — синоним обновляемой таблицы для ссылки на нее в подзапросе; имя_столбца

- имя обновляемого столбца;

выражение - допустимое в SQL выражение соответствующего типа, значение которого присваивается обновляемому столбцу;

условие - допустимое в SQL выражение условия, которое используется для отбора обновляемых строк.

По одному предложению UPDATE обновлению подвергаются строки только одной базовой таблицы.

Обновление всех строк

Как видно из определения синтаксиса команды UPDATE, фраза WHERE является факультативной. При ее отсутствии все строки таблицы подвергаются обновлению согласно фразе SET.



Во фразе SET можно одновременно изменять значения нескольких столбцов таблицы.

Запрос 34. Например, в следующем примере всем преподавателям увеличивается ставка на 12 % и надбавка на 7 %:

UPDATE TEACHER

SET Salary = Salary + Salary * 0.12, Rise = Rise + Rise * 0.08;

Во фразе SET в правой части оператора присваивания может использоваться любое допустимое в SQL выражение того же типа, что и столбец, имя которого приведено слева от оператора присваивания. Если в этом выражении используется имя столбца целевой таблицы, для вычисления выражения применяется значение этого столбца в текущей строке, которое было перед обновлением.

Примечание. Чтобы данные были добавлены, не забудьте нажать на кнопки , а затем  для подтверждения внесенных изменений в таблицу. Просмотрите внесенные изменения в таблицу TEACHER.

Обновление по условию

Данный вариант использует фразу WHERE. В этом случае обновляются столбцы только тех строк таблицы, на которых выполняется условие фразы WHERE. Рассмотрим несколько примеров.

Запрос 35. Увеличить всем ассистентам зарплату и надбавку на 10 %:

UPDATE TEACHER

SET Salary = Salary * 1.1, Rise = Rise * 1.1

WHERE LOWER(Dolgnost) = 'ассистент';

Самостоятельно создать запрос 36. Установить, что деканат юридического факультета переместился в комнату 232 8 корпуса.

Подзапросы во фразе WHERE

Во фразе WHERE можно использовать подзапросы, как мы это делали в предложении SELECT. Это дает возможность отбирать строки для обновления на основе информации из других таблиц.

Запрос 37. Например, увеличить ставку всех преподавателей кафедры прикладной математики факультета математики и информатики в полтора раза:

UPDATE TEACHER

SET Salary = Salary * 1.5

WHERE KOD_kafedru = (SELECT KOD_kafedru FROM KAFEDRA

WHERE LOWER(Name_kafedru) = 'прикладная математика');

Подзапросы во фразе SET

До сих пор новые значения представляли собой константы или выражения с использованием значений обновляемой строки. Однако если такие значения присутствуют в других строках обновляемой таблицы или вообще в других таблицах, можно воспользоваться подзапросом. В этом случае допускается использовать две формы фразы SET:

SET {имя_столбца | (список_имен_столбцов)} = (подзапрос)

В обоих вариантах подзапрос должен возвращать одну строку. В первом случае он также должен возвращать значение одного столбца, а во втором возвращаемая строка должна содержать столько значений, сколько столбцов приведено в списке имен столбцов. При этом производится присвоение значений строки подзапроса соответствующим столбцам из списка слева.

Запрос 38. Установить всем ассистентам надбавку, равную 70 % текущей средней надбавки по вузу.

UPDATE TEACHER

SET Rise = (SELECT SUM(Rise) * 0.7 / COUNT(*) FROM TEACHER)
WHERE LOWER(Dolgnost) = 'ассистент';

Удаление существующих строк

Удалять строки из таблицы можно с помощью предложения DELETE. Оно удаляет только строки целиком, а не индивидуальные значения столбцов. Синтаксис команды следующий:

**DELETE FROM имя_таблицы [[AS] синоним]
[WHERE условие];**

При использовании предложения DELETE вы прежде всего обнаружите, что предупреждающая подсказка, как правило, не выдается. Обычно, когда пользователь удаляет какой-либо объект операционной среды, он получает сообщение типа «Вы уверены (Д/Н)?». В системах, поддерживающих SQL, строки удаляются без такого сообщения. Поэтому будьте внимательны.

В зависимости от применения фразы WHERE предложение DELETE позволяет удалить отдельную строку, несколько или все строки таблицы. Строки могут быть и не удалены. При использовании предложения DELETE помните о следующем:

нельзя удалить значение отдельного столбца (используйте для этого предложение UPDATE);

как и предложения INSERT и UPDATE, удаление строк может нарушить ограничения целостности;

сама таблица не удаляется (используйте для этого предложение DROP TABLE).

Удаление всех строк таблицы

Чтобы удалить все содержимое таблицы, не нужно использовать фразу WHERE. Помните, что вы удаляете не саму таблицу, а только все ее строки.

Запрос 39. Удалить содержимое таблицы Сотрудники базы данных Educator.
use Educator
DELETE FROM Сотрудники;

Удаление по условию

Обычно нужно удалять только некоторые строки из таблицы. Чтобы определить, какие строки будут удалены, нужно использовать условие во фразе WHERE. Приведем несколько примеров.

Самостоятельно создать запрос 40. Удалить сведения об ассистентах, которые были приняты на работу до 01.01.1986.

Удаление одной строки

Чтобы удалить одну конкретную строку, нужно сформулировать условие таким образом, чтобы оно идентифицировало эту единственную строку. Обычно для этого в условии используются первичный ключ таблицы или уникальный набор столбцов.

Самостоятельно создать запрос 41. Удалить всех преподавателей под фамилией Швец.

Задание для практической работы №18

Для созданной базы данных, согласно номеру варианта, самостоятельно создать на языке **Transact-SQL 14** многотабличных запросов:

- 1 запрос с использованием функции **COUNT**;
- 1 запрос с использованием функции **SUM**;
- 1 запрос с использованием функций **UPPER, LOWER**;
- 1 запрос с использованием временных функций;
- 1 запрос с использованием группировки по одному столбцу;
- 1 запрос на использование группировки по нескольким столбцам;
- 1 запрос с использованием условия отбора групп **HAVING**;
- 1 запрос с использованием фразы **HAVING** без фразы **GROUP BY**;
- 1 запрос с использованием сортировки по столбцу;
- 1 запрос на добавление новых данных в таблицу;
- 1 запрос на добавление новых данных по результатам запроса в качестве вставляемого значения;
- 1 запрос на обновление существующих данных в таблице;
- 1 запрос на обновление существующих данных по результатам подзапроса во фразе **WHERE**;
- 1 запрос на удаление существующих данных.

Все программные инструкции команд SQL сохранять в файлах с расширением ***.sql** в папке **ФИО_студента/Лаб7**.

Для каждого запроса сформулировать текстовое задание, которое должно быть выполнено к базе данных.

Создать текстовый отчет, в котором отобразить sql-команды разработанных запросов и скриншоты результатов работы из СУБД **SQL Server Management Studio**.

Практическая работа №19 Импорт данных пользователя в базу данных

Цель работы: овладение практическими навыками обработки табличных данных

Вставка, удаление и обновление данных

После создания БД и таблиц перед разработчиком встает задача заполнения таблиц данными. В реляционных БД традиционно применяют три подхода:

- однострочный оператор *INSERT* – добавляет в таблицу новую запись;
- многострочный оператор *INSERT* – добавляет в таблицу несколько записей;
- пакетная загрузка *LOAD DATA INFILE* – добавление данных из файла.

Вставка данных с помощью оператора *INSERT*. Однострочный оператор *INSERT* может использоваться в нескольких формах. Упрощенный синтаксис первой формы:

```
INSERT [IGNORE] [INTO] имя_таблицы [(имя_столбца, ... )]  
VALUES (выражение, ... );
```

Оператор вставляет новую запись в таблицу *имя_таблицы*. Значения полей записи перечисляются в списке (*выражение*, ...). Порядок следования столбцов задается списком (*имя_столбца*, ...). Список столбцов (*имя_столбца*, ...) позволяет менять порядок следования столбцов при добавлении.

Первичный ключ таблицы является уникальным, и попытка добавить уже существующее значение приведет к ошибке. Чтобы новые записи с дублирующим ключом отбрасывались без генерации ошибки, следует добавить после оператора *INSERT* ключевое слово *IGNORE*.

Другая форма оператора *INSERT* предполагает использование слова *SET*:

```
INSERT [IGNORE] [INTO] имя_таблицы  
SET имя_столбца1 = выражение1, имя_столбца2 = выражение2, ... ;
```

Оператор заносит в таблицу *имя_таблицы* новую запись, столбец *имя_столбца* в которой получает значение *выражение*.

Многострочный оператор *INSERT* совпадает по форме с однострочным оператором, но после ключевого слова *VALUES* добавляется через запятую несколько списков (*выражение*, ...).

Практические примеры использования оператора *INSERT* для заполнения учебной БД *book* см. ниже, в пункте «Пример выполнения работы».

Удаление данных. Для удаления записей из таблиц предусмотрены:

- оператор *DELETE*;
- оператор *TRUNCATE TABLE*.

Оператор *DELETE* имеет следующий синтаксис:

```
DELETE FROM имя_таблицы  
[WHERE условие]  
[ORDER BY имя_поля]  
[LIMIT число_строк];
```

Оператор удаляет из таблицы *имя_таблицы* записи, удовлетворяющие условию. В следующем примере из таблицы *catalogs* удаляются записи, имеющие значение первичного ключа *catalog_id* больше двух.

```
mysql> DELETE FROM catalogs WHERE cat_ID>2;  
Query OK, 3 rows affected (0.05 sec)
```

```
mysql> SELECT * FROM catalogs;  
+-----+-----+  
| cat_ID | cat_name |  
+-----+-----+  
|      1 | Программирование |  
|      2 | Интернет |  
+-----+-----+  
2 rows in set (0.00 sec)
```

Если в операторе отсутствует условие *WHERE*, удаляются все записи таблицы.

```
mysql> DELETE FROM catalogs;  
Query OK, 2 rows affected (0.03 sec)
```

```
mysql> SELECT * FROM catalogs;  
Empty set (0.00 sec)
```

Ограничение *LIMIT* позволяет задать максимальное число записей, которые могут быть удалены. Следующий запрос удаляет все записи таблицы *orders*, но не более 3 записей.


```
mysql> DELETE FROM orders LIMIT 3;
Query OK, 3 rows affected (0.01 sec)

mysql> SELECT * FROM orders;
+----+-----+-----+-----+-----+
| order_ID | o_user_ID | o_book_ID | o_time          | o_number |
+----+-----+-----+-----+-----+
| 4 | 4 | 20 | 2009-03-10 18:20:00 | 1 |
| 5 | 3 | 20 | 2009-03-17 19:15:36 | 1 |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Конструкция *ORDER BY* обычно применяется вместе с ключевым словом *LIMIT*. Например, если необходимо удалить 20 первых записей таблицы, то производится сортировка по полю типа *DATETIME* – тогда в первую очередь будут удалены самые старые записи.

Оператор *TRUNCATE TABLE* полностью очищает таблицу и не допускает условного удаления. Он аналогичен оператору *DELETE* без условия *WHERE* и ограничения *LIMIT*. Удаление происходит гораздо быстрее, т. к. осуществляется не перебор записей, а полное очищение таблицы.

```
mysql> TRUNCATE TABLE orders;
Query OK, 5 rows affected (0.03 sec)

mysql> SELECT * FROM orders;
Empty set (0.00 sec)
```

Обновление данных. Обновление данных (изменение значений полей в существующих записях) обеспечивают:

- оператор *UPDATE*;
- оператор *REPLACE*.

Оператор *UPDATE* позволяет обновлять отдельные поля в существующих записях. Имеет следующий синтаксис

```
UPDATE [IGNORE] имя_таблицы
SET имя_столбца1=выражение1 [, имя_столбца2 = выражение2 ... ]
[WHERE условие]
[ORDER BY имя_поля ]
[LIMIT число_строк] ;
```

После ключевого слова *UPDATE* указывается таблица, которая изменяется. В предложении *SET* указывается, какие столбцы обновляются и устанавливаются их новые значения. Необязательное условие *WHERE* позволяет задать критерий отбора строк (обновляться будут только строки, удовлетворяющие условию).

Если указывается необязательное ключевое слово *IGNORE*, то команда обновления не будет прервана, даже если при обновлении возникнет ошибка дублирования ключей. Строки, породившие конфликтные ситуации, обновлены не будут.

Запрос, изменяющий в таблице *catalogs* «Сети» на «Компьютерные сети».

```
mysql> UPDATE catalogs SET cat_name='Компьютерные сети'
-> WHERE cat_name='Сети';
Query OK, 1 row affected (0.03 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

```
mysql> SELECT * FROM catalogs;
+----+-----+
| cat_ID | cat_name          |
+----+-----+
| 1 | Программирование |
| 2 | Интернет         |
| 3 | Базы данных      |
| 4 | Компьютерные сети |
| 5 | Мультимедиа     |
+----+-----+
5 rows in set (0.00 sec)
```

Обновлять можно всю таблицу. Пусть требуется уменьшить на 5 % цену на все книги. Для этого следует старую цену в рублях умножить на 0,95.

```
mysql> UPDATE books SET b_price=b_price*0.95;
Query OK, 30 rows affected (0.03 sec)
Rows matched: 30 Changed: 30 Warnings: 0

mysql> SELECT book_ID, b_name, b_price FROM books;
```

book_ID	b_name	b_price
1	JavaScript в кармане	39.90
2	Visual FoxPro 9.0	627.00
3	C++ Как он есть	207.10
4	Создание приложений с помощью С#	160.55
5	Delphi. Народные советы	230.85
6	Delphi. Полное руководство	475.00
7	Профессиональное программирование на PHP	293.55
8	Совершенный код	732.45
9	Практика программирования	203.30
10	Принципы маршрутизации в Internet	406.60
11	Поиск в Internet	101.65
12	Web-конструирование	168.15
13	Самоучитель Интернет	114.95
14	Популярные интернет-браузеры	77.90
15	Общение в Интернете	80.75
16	Базы данных	309.70
17	Базы данных. Разработка приложений	179.55
18	Раскрытие тайн SQL	190.00
19	Практикум по Access	82.65
20	Компьютерные сети	598.50
21	Сети. Поиск неисправностей	412.30
22	Безопасность сетей	438.90
23	Анализ и диагностика компьютерных сетей	326.80
24	Локальные вычислительные сети	77.90
25	Цифровая фотография	141.55
26	Музыкальный компьютер для гитариста	206.15
27	Видео на ПК	219.45
28	Мультимедиа во Flash	200.45
29	Запись CD и DVD	158.65
30	Запись и обработка звука на компьютере	48.45

```
30 rows in set (0.00 sec)
```

Инструкции *LIMIT* и *ORDER BY* позволяют ограничить число изменяемых записей. При этом за один запрос можно обновить несколько столбцов таблицы. Например, необходимо в таблице *books* для десяти самых дешевых товарных позиций уменьшить количество книг на складе на единицу, а цену – на 5 %.

```
mysql> UPDATE books SET b_price=b_price*0.95,b_count=b_count-1
-> ORDER BY b_price LIMIT 10;
Query OK, 10 rows affected (0.05 sec)
Rows matched: 10 Changed: 10 Warnings: 0

mysql> SELECT book_ID, b_name, b_price, b_count FROM books;
```

book_ID	b_name	b_price	b_count
1	JavaScript в кармане	39.90	9
2	Visual FoxPro 9.0	660.00	2
3	C++ Как он есть	218.00	4
4	Создание приложений с помощью С#	169.00	1
5	Delphi. Народные советы	243.00	6
6	Delphi. Полное руководство	500.00	6
7	Профессиональное программирование на PHP	309.00	5
8	Совершенный код	771.00	1
9	Практика программирования	214.00	12
10	Принципы маршрутизации в Internet	428.00	4
11	Поиск в Internet	101.65	1
12	Web-конструирование	177.00	6
13	Самоучитель Интернет	114.95	3
14	Популярные интернет-браузеры	77.90	5
15	Общение в Интернете	80.75	4
16	Базы данных	326.00	2
17	Базы данных. Разработка приложений	189.00	6
18	Раскрытие тайн SQL	200.00	3
19	Практикум по Access	82.65	5
20	Компьютерные сети	630.00	6
21	Сети. Поиск неисправностей	434.00	4
22	Безопасность сетей	462.00	5
23	Анализ и диагностика компьютерных сетей	344.00	3
24	Локальные вычислительные сети	77.90	7
25	Цифровая фотография	141.55	19
26	Музыкальный компьютер для гитариста	217.00	15
27	Видео на ПК	231.00	10
28	Мультимедиа во Flash	211.00	20
29	Запись CD и DVD	158.65	11
30	Запись и обработка звука на компьютере	48.45	7

```
30 rows in set (0.00 sec)
```

Оператор *REPLACE* работает как оператор *INSERT*, за исключением того, что старая запись с тем же значением индекса *UNIQUE* или *PRIMARY KEY* перед внесением новой будет удалена. Если не используются индексы *UNIQUE* или *PRIMARY KEY*, то применение оператора *REPLACE* не имеет смысла.

Синтаксис оператора *REPLACE* аналогичен синтаксису оператора *INSERT*:
REPLACE [INTO] имя_таблицы [(имя_столбца, ...)]
VALUES (выражение, ...)

В таблицу вставляются значения, определяемые в списке после ключевого слова *VALUES*. Задать порядок столбцов можно при помощи необязательного списка, следующего за именем таблицы. Как и оператор *INSERT*, оператор *REPLACE* допускает многострочный формат.

Задание для практической работы

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

- заполнить согласованными данными таблицы БД;
- при необходимости исправить введенную информацию;
- составить отчет по практической работе.

Операторы заполнения БД *book* имеют следующий вид.

USE book;

SET CHARACTER SET cp1251;

DELETE FROM catalogs;

INSERT INTO catalogs VALUES (1, 'Программирование');

INSERT INTO catalogs VALUES (2, 'Интернет');

INSERT INTO catalogs VALUES (3, 'Базы данных');

INSERT INTO catalogs VALUES (4, 'Сети');

INSERT INTO catalogs VALUES (5, 'Мультимедиа');

DELETE FROM books;

INSERT INTO books VALUES (1, 'JavaScript в кармане', 'Рева О.Н.', 2008, 42.00, 10, 1);

INSERT INTO books VALUES (2, 'Visual FoxPro 9.0', 'Клепинин В.Б.', 2007, 660.00, 2, 1);

INSERT INTO books VALUES (3, 'C++ Как он есть', 'Тимофеев В.В.', 2009, 218.00, 4, 1);

INSERT INTO books VALUES (4, 'Создание приложений с помощью C#', 'Фаронов В.В.', 2008, 169.00, 1, 1);

INSERT INTO books VALUES (5, 'Delphi. Народные советы', 'Шкрыль А.А.', 2007, 243.00, 6, 1);

INSERT INTO books VALUES (6, 'Delphi. Полное руководство', 'Сухарев М.', 2008, 500.00, 6, 1);

INSERT INTO books VALUES (7, 'Профессиональное программирование на PHP',

'Шлоснейгл Дж.', 2006, 309.00, 5, 1);

INSERT INTO books VALUES (8, 'Совершенный код', 'Макконнелл С.', 2007, 771.00, 1, 1);

INSERT INTO books VALUES (9, 'Практика программирования', 'Керниган Б.', 2004, 214.00, 12, 1);

INSERT INTO books VALUES (10, 'Принципы маршрутизации в Internet', 'Хелеби С.', 2001, 428.00, 4, 2);

INSERT INTO books VALUES (11, 'Поиск в Internet', 'Гусев В.С.', 2004, 107.00, 2, 2);

INSERT INTO books VALUES (12, 'Web-конструирование', 'Дуванов А.А.', 2003, 177.00, 6, 2);

INSERT INTO books VALUES (13, 'Самоучитель Интернет', 'Константинов Ю.П.', 2009, 121.00, 4, 2);

INSERT INTO books VALUES (14, 'Популярные интернет-браузеры', 'Маринин С.А.', 2007, 82.00, 6, 2);

INSERT INTO books VALUES (15, 'Общение в Интернете', 'Экслер А.', 2006, 85.00, 5, 2);

INSERT INTO books VALUES (16, 'Базы данных', 'Малыхина М.П.', 2006, 326.00, 2, 3);

INSERT INTO books VALUES (17, 'Базы данных. Разработка приложений', 'Рудикова Л.В.', 2006, 189.00, 6, 3);

INSERT INTO books VALUES (18, 'Раскрытие тайн SQL', 'Оппель Э.', 2007, 200.00, 3, 3);

INSERT INTO books VALUES (19, 'Практикум по Access', 'Золотова С.И.', 2007, 87.00, 6, 3);

INSERT INTO books VALUES (20, 'Компьютерные сети', 'Танненбаум Э.', 2007, 630.00, 6, 4);

INSERT INTO books VALUES (21, 'Сети. Поиск неисправностей', 'Бигеллоу С.', 2005, 434.00, 4, 4);

INSERT INTO books VALUES (22, 'Безопасность сетей', 'Брегг Р.', 2006, 462.00, 5, 4);

INSERT INTO books VALUES (23, 'Анализ и диагностика компьютерных сетей', 'Хогдал Дж.', 2001, 344.00, 3, 4);

INSERT INTO books VALUES (24, 'Локальные вычислительные сети', 'Епанешников А.', 2005, 82.00, 8, 4);

INSERT INTO books VALUES (25, 'Цифровая фотография', 'Надеждин Н.', 2004, 149.00, 20, 5);

INSERT INTO books VALUES (26, 'Музыкальный компьютер для гитариста', 'Петелин Р.Ю.', 2004, 217.00, 15, 5);

```

INSERT INTO books VALUES (27,'Видео на ПК','Федорова А.',2003,231.00,10,5);
INSERT INTO books VALUES (28,'Мультипликация во Flash','Киркпатрик Г.', 2006, 211.00,
20, 5);
INSERT INTO books VALUES (29,'Запись CD и DVD','Гультияев А.К.', 2003, 167.00, 12, 5);
INSERT INTO books VALUES (30,'Запись и обработка звука на компьютере', 'Лоянич А.А.',
2008, 51.00, 8, 5);
DELETE FROM users;
INSERT INTO users VALUES (1,'Александр','Валерьевич','Иванов','58-98-78',
'ivanov@email.ru', 'active');
INSERT INTO users VALUES (2,'Сергей','Иванович','Лосев','90-57-77', 'losev@email.ru',
'passive');
INSERT INTO users VALUES (3,'Игорь','Николаевич','Симонов','95-66-61',
'simonov@email.ru', 'active');
INSERT INTO users VALUES (4,'Максим','Петрович','Кузнецов',NULL, 'kuznetsov@email.ru',
'active');
INSERT INTO users VALUES (5,'Анатолий','Юрьевич','Петров', NULL, NULL, 'lock');
INSERT INTO users VALUES (6,'Александр','Александрович','Корнеев','89-78-36',
'korneev@email.ru', 'gold');
DELETE FROM orders;
INSERT INTO orders VALUES (1,3,8,'2009-01-04 10:39:38',1);
INSERT INTO orders VALUES (2,6,10,'2009-02-10 09:40:29',2);
INSERT INTO orders VALUES (3,1,20,'2009-02-18 13:41:05',4);
INSERT INTO orders VALUES (4,4,20,'2009-03-10 18:20:00',1);
INSERT INTO orders VALUES (5,3,20,'2009-03-17 19:15:36',1);

```

Практическая работа №20-21. Выполнение настроек для автоматизации обслуживания базы данных

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
<i>AVG</i> ([<i>DISTINCT</i>] <i>expr</i>)	Возвращает среднее значение аргумента <i>expr</i> . В качестве аргумента обычно выступает имя столбца. Необязательное слово <i>DISTINCT</i> позволяет обрабатывать только уникальные значения столбца <i>expr</i>
<i>COUNT</i> ()	Подсчитывает число записей и имеет несколько форм. Форма <i>COUNT</i> (<i>выражение</i>) возвращает число записей в таблице, поле <i>выражение</i> для которых не равно <i>NULL</i> . Форма <i>COUNT</i> (*) возвращает общее число строк в таблице независимо от того, принимает какое-либо поле значение <i>NULL</i> или нет. Форма <i>COUNT</i> (<i>DISTINCT</i> <i>выражение1</i> , <i>выражение2</i> , ...) позволяет использовать ключевое слово <i>DISTINCT</i> , которое позволяет подсчитать только уникальные значения столбца
<i>MIN</i> ([<i>DISTINCT</i>] <i>expr</i>)	Возвращает минимальное значение среди всех непустых значений выбранных строк в столбце <i>expr</i> . Необязательное слово <i>DISTINCT</i> позволяет обрабатывать только уникальные значения столбца <i>expr</i>
<i>MAX</i> ([<i>DISTINCT</i>] <i>expr</i>)	Возвращает максимальное значение среди всех непустых значений выбранных строк в столбце <i>expr</i> . Необязательное слово <i>DISTINCT</i> позволяет обрабатывать только уникальные значения столбца <i>expr</i>
<i>STD</i> (<i>expr</i>)	Возвращает стандартное среднееквадратичное отклонение в аргументе <i>expr</i>
<i>STDDEV_SAMP</i> (<i>expr</i>)	Возвращает выборочное среднееквадратичное отклонение в аргументе <i>expr</i>
<i>SUM</i> ([<i>DISTINCT</i>] <i>expr</i>)	Возвращает сумму величин в столбце <i>expr</i> . Необязательное слово <i>DISTINCT</i> позволяет обрабатывать только уникальные значения столбца <i>expr</i>

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). <i>NULL</i> 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

Таблица 16.3. Логические операторы

Оператор	Пример	Значение
AND	n AND m	Логическое И: истина AND истина = истина, ложь AND любое = ложь. Все остальные выражения оцениваются как NULL
OR	n OR m	Логическое ИЛИ: истина OR любое = истина, NULL OR ложь = NULL, NULL OR NULL = NULL, ложь OR ложь = ложь
NOT	NOT n	Логическое НЕ: NOT истина = ложь, NOT ложь = истина. NOT NULL = NULL
XOR	n XOR m	Логическое исключающее ИЛИ: истина XOR истина = ложь, истина XOR ложь = истина, ложь XOR истина = истина, ложь XOR ложь = ложь, NULL XOR любое = NULL, любое XOR NULL = NULL

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	Ильевич	0
Посев	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 С#	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	Самостоятель. Internet	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 по 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)

Практическая работа №22. Мониторинг работы сервера

Цель практической работы

Изучение общих правил разграничения и предоставление прав доступа пользователям баз данных, архитектуры и компонент системы безопасности SQL Server и режимов аутентификации пользователей, а также приобретение навыков администрирования системы безопасности: создания и управления учетными записями, управления пользователями, ролями и группами.

Методические рекомендации для выполнения практической работы

Система хранения информации должна быть максимально защищена как от случайного, так и от злонамеренного повреждения, искажения или утечки информации. С этой целью, прежде всего, надо определить круг пользователей, которые будут иметь доступ к базам данных. Далее, для этих пользователей необходимо создать учетные записи в домене Windows, а также в соответствующем экземпляре сервера SQL Server, чтобы разрешить им обращаться к этому серверу. Разрешение доступа к серверу не дает автоматически доступа к его базам данных и их объектам.

Второй этап планирования безопасности использования баз данных сервера заключается в определении действий, который может выполнить конкретный пользователь. Полный доступ к базам данных и всем их объектам имеет администратор – ему позволено все. Второе лицо после администратора – это владелец объекта. При создании любого объекта в любой базе данных ему назначается владелец, который может устанавливать права доступа к этому объекту, модифицировать и удалять его. Все остальные пользователи, составляющую третью, основную группу, имеют права доступа, выданные администратором или владельцем объекта. Эти права должны быть тщательно спланированы в соответствии с занимаемой должностью и необходимостью выполнения конкретных действий.

Для работы с базами данных пользователи любой категории проходят следующие два этапа проверки системой безопасности. На первом этапе пользователь идентифицируется по имени учетной записи и паролю, т. е. проходит аутентификацию. Если данные введены правильно, то пользователь подключается к требуемому серверу, выбрав его из списка серверов и исполнив команду подключения либо с помощью Enterprise Manager, либо исполнив команды Transact –SQL в Query Analyzer. Пользователю при этом будут предоставлены те права доступа к серверу, которые имеют роль сервера, содержащая данного пользователя. Подключение к выбранному серверу, или регистрация, не дает автоматического доступа к базам данных.

На втором этапе по регистрационному имени находится имя пользователя базы данных, которое было создано администратором, и пользователь получает права доступа к выбираемой базе данных в соответствии с той ролью базы данных, в которую был включен этот пользователь администратором базы на этапе конфигурирования системы без опасности. В разных базах данных один и тот же пользователь может иметь одинаковые или разные имена пользователя базы данных с разными правами доступа, как правило. Таким образом, пользователь имеет одно имя, которое он задает при входе в систему, и, возможно, несколько имен для доступа к базам данных и их объектам.

Для доступа приложений к базам данных им также понадобятся права. Чаще всего приложениям выдаются те же права, которые предоставлены пользователям, запускающим эти приложения. Однако для работы некоторых приложений необходимо

иметь фиксированный набор прав доступа, не зависящих от прав доступа пользователя. Это обеспечивается использованием специальных ролей приложения.

Итак, компонентами системы безопасности SQL Server 2000 на уровне сервера являются: система аутентификации средствами Windows и средствами SQL Server, учетные записи пользователей и встроенные роли сервера. На уровне базы данных компонентами системы безопасности являются: идентификация пользователей баз данных, фиксированные и пользовательские роли баз данных, а также роли приложений.

Фиксированными ролями сервера являются:

Sysadmin – для выполнения любых действий в сервере; Seleradmin– для конфигурирования и выключения сервера;

Setupadmin– для управления связанными серверами и процедурами, автоматически запускающимися при старте сервера;

Securityadmin– для управления учетными записями и правами на создание базы данных, а также для контроля журнала ошибок;

Processadmin - для управления процессами, запущенными на сервере_____

Dbcreator – для создания и модификации баз данных;

Diskadmin– для управления файлами сервера; Bulcadmin– для массивного копирования баз данных.

Фиксированную роль сервера нельзя удалить или модифицировать. Нельзя также создать новую фиксированную роль. Предоставить права доступа к серверу можно только путем включения пользователя в требуемую роль сервера. Таким образом, роли сервера позволяют объединять пользователей, выполняющих одинаковые функции, для упрощения администрирования системы безопасности SQL Server. В предыдущих версиях SQL Server можно было использовать только учетную запись sa, которая предоставляла все права доступа к серверу.

При создании базы данных сервер автоматически создает для нее фиксированные роли, которые, как и фиксированные роли сервера, нельзя удалить или модифицировать:

Db_owner – для выполнения любых действий в базе данных;

Db_accessadmin – для добавления и удаления пользователей;

Db_securityadmin – для управления всеми разрешениями, объектами, ролями и именами ролей;

Db_ddladmin – для выполнения любых команд DDL, кроме GRANT, DENY и REVOKE;

Db_backupoperator– для выполнения команд DBCC, CHECK, POINT и BACKUP; Db_datareader – для контроля данных во всех таблицах базы данных и их чтения; Db_datawriter – для модификации данных в любых таблицах базы данных; Db_denydatareader – для запрета просмотра данных в любой таблице базы данных;

Db_denydatawriter – для запрета модификации данных во всех таблицах базы данных.

Кроме этих фиксированных ролей любой базы данных есть еще одна роль public, членами которой автоматически становятся все пользователи, имеющие тот или иной доступ к базе данных. Эта роль имеет специальное назначение и обеспечивает минимальные права доступа к базе данных тем пользователям, для которых их права не определены явно. Эта роль имеется во всех базах данных, включая системные master, tempdb, msdb и не может быть удалена.

Если в базе данных разрешен пользователь quest, то установленный для public доступ будут иметь все пользователи, получившие доступ к SQL Server.

В отличие от сервера базы данных могут иметь пользовательские роли и роли приложения, которые создает администратор с помощью Enterprise Manager или Transact_SQL индивидуально для групп пользователей и групп приложений, наделяя их необходимыми правами доступа к конкретной базе данных.

В любую роль базы данных можно включать:

- а) пользователей сервера;б)
- роли сервера;
- в) пользователей Windows;
- г) группы пользователей Windows.

Средствами Enterprise Manager можно включать только пользователей сервера.Процедура SQL Server `sp_addrolemember 'role', 'security_account'` позволяет включать как роли сервера, так и пользователей Windows, в том числе и групп пользователей с помощью задания их учетной записи в SQL Server или Windows `'security_account'` и указания требуемой роли `'role'`.

Работа с данными и выполнение хранимых процедур требуют наличия класса доступа, называемого правами на доступ к объектам баз данных: таблицам и ее столбцам, представлениям и хранимым процедурам.

Таковыми правами являются:

SELECT, INSERT, UPDETE, DELETE, RFERENCES – для таблиц и представлений, а **SELECT** и для столбца (тоже и для **UPDETE**), **SELECT** и **UPDETE** – для столбца таблицы или представления;

EXECUTE – для хранимых процедур и функций.

Здесь право RFERENCES разрешает создавать внешние ключи и представления для таблиц.

Командой GRANT можно разрешать пользователям определенные права доступа к объектам, командой **DENI** – запрещать их.

Помимо прав доступа к объектам имеются еще и права, на создание объектов базы данных и самой базы данных:

CREATE DATABASE – на создание базы данных ;

CREATE TABLE – на создание таблиц;

CREATE VIEW – на создание представлений;

CREATE DEFAULT – на создание умолчаний;

CREATE RULE – на создание правил;

CREATE PROCEDURE – на создание хранимых процедур;

BACKUP DATABASE – на резервное копирование баз данных;

BACKUP LOG – на резервное копирование журнала транзакций;

ALL – на создание любых объектов.

При установке SQL Server имеется возможность выбрать один из двух режимов аутентификации:

а) средствами Windows ;

б) средствами Windows и/или средствами SQL Server.

В первом случае после успешной аутентификации с помощью Windows SQL Server автоматически обеспечивает доступ пользователя к требуемому экземпляру сервера и к нужной базе данных. Этот метод подключения называется методом установления доверительного подключения. В этом случае член стандартной роли `sysadmin` или `securityadmin` должен указать серверу, какие группы или пользователи Windows имеют

доступ к серверу. Во время подключения пользователя его имя и пароль запрашиваются только один раз при входе в систему Windows.

Во втором режиме системным администратором, входящим в роль sysadmin или securityadmin, должна быть создана и сконфигурирована учетная запись в SQL Server для каждого пользователя. Эта запись будет содержать собственное имя, имя экземпляра сервера и пароль. Две такие записи с именем BUILTIN\Administrators и sa создаются автоматически при установке сервера. Обе эти записи автоматически включаются также во встроенную роль сервера sysadmin, в результате системные администраторы получают полный доступ ко всем базам данных с именем пользователя dbo (DataBase Owner). Если функции системного администратора и администратора баз данных выполняют разные люди, следует исключить учетную запись BUILTIN\Administrators. Запись sa не следует использовать, так как она предназначена для совместимости со старыми версиями SQL Server и для входа в сервер, если администратор баз данных забыл пароль. Как правило, для администратора баз данных создается отдельная учетная запись с ролью сервера sysadmin.

После того как пользователь прошел аутентификацию и получил идентификатор учетной записи (LoginID), он считается зарегистрированным и ему предоставляется доступ к серверу.

Учетная запись при создании была связана с конкретной базой данных, а пользователь – с конкретным именем пользователя базы данных. Именно пользователи баз данных являются специальными объектами, которым предоставляются права доступа к данным. Если учетная запись не связывается с конкретным пользователем, то такому пользователю предоставляется неявный доступ с использованием гостевого имени quest, которому даются минимальные права владельцами баз данных. Все учетные записи связаны с quest.

Если в базе данных разрешен пользователь quest, то установленный для роли public доступ будут иметь все пользователи, получившие доступ к SQL Server.

Для управления системой безопасности сервера SQL Server можно использовать следующие хранимые процедуры и команды языка Transact_SQL:

- sp_addapprole – создать роль для приложения ; sp_addlogin – создать новую учетную запись сервера; sp_addrole – создать новую роль в базе данных; sp_addrolemember – добавить члена в роль базы данных;
- sp_addsrvrolemember – добавить члена в фиксированную роль сервера;
- sp_approlepassword – изменить пароль для роли приложения; sp_defaultldb – изменить базу данных по умолчанию для учетной записи; sp_defaultlanguage – изменить язык по умолчанию для учетной записи; p_denylogin – запретить доступ пользователю или группе Windows; sp_dropapprole – удалить роль приложения;
- sp_droplinkedsvlogin – удалить отображения учетной записи с другого сервера; sp_droplogin – удалить учетную запись сервера;
- sp_droprole – удалить роль базы данных;
- sp_droprolemember – удалить пользователя из роли базы данных;
- sp_dropsvrolemember – удалить члена из роли сервера;
- sp_grantdbaccess – разрешить доступ к базе данных учетной записи сервера, пользователям и группам пользователей Windows;
- sp_grantlogin – разрешить доступ к серверу;
- sp_helpdbfixedrole – выдать список фиксированных ролей в базе данных; sp_helplogins – посмотреть учетную запись;
- sp_helpntgraer – посмотреть группы NT в сервере;

sp_helprole – посмотреть роли, определенные в базе данных; sp_helpsrvrole – выдать список фиксированных ролей сервера; sp_helpsrvrolemember – выдать информацию о члене роли сервера; sp_helpuser – посмотреть информацию о пользователе; sp_password – изменить пароль учетной записи сервера; sp_setapprole – инициализировать роль приложения; GRANT – предоставить доступ; DENY – запретить доступ; REVOKE – неявно отключить доступ.

Задание к практической работе №22

Задание 1. Создать учетную запись SQL сервера, используя графическую утилиту Enterprise Manager, выполнив следующие действия:

1. Выбрать нужный сервер.
2. Открыть папку Security этого сервера.
3. Выбрать объект Logins, щелкнув по соответствующему значку.
4. В правом окне просмотреть список учетных записей данного сервера:

Name – имя учетной записи сервера;

Type – происхождение учетной записи:

User W – пользователь Windows;

Group W – группа пользователей Windows;

Standard – пользователь SQL сервера; Server

Access – доступ к серверу SQL;

Permit – разрешен;

Deny – запрещен;

Default Database – база данных по умолчанию, к которой подключен пользователь (обязательный параметр)

User – имя пользователя базы данных;

Default Language – язык по умолчанию для данной учетной записи.

5. Для создания новой учетной записи сервера открыть контекстное меню объекта Logins, щелкнув по нему правой клавишей мыши или по значку на панели инструментов левой клавишей мыши.

6. В появившемся диалоговом окне на вкладке General (общие) ввести имя учетной записи в поле Name.

7. Выбрать тип аутентификации: Windows Authentication или SQL Server Authentication.

8. Если выбрана аутентификация Windows, то задать в поле Domain имя домена, в котором учтен пользователь или группа Windows. Имя заданного домена добавиться впереди имени пользователя также и в поле Name (для выбора домена использовать кнопку "...").

9. В группе Security Access (безопасный доступ) установить переключатель Grant Access (доступ разрешен). Установка переключателя Deny Access навсегда запретит регистрацию пользователя или домена Windows.

10. Если выбрана аутентификация SQL Server, то задать только пароль для учетной записи.

11. Задав параметры аутентификации Windows или SQL Server, указать в группе Defaults (умолчания) имя базы данных в поле Database, к которой пользователь будет

подключаться автоматически, и язык Language (Russian). Если имя базы данных не задано, то сервер будет автоматически подключаться к базе master.

12. Включить создаваемую учетную запись в требуемую встроенную роль сервера: Sysadmin, Serveradmin, Setupadmin, Securityadmin, Processadmin, Dbcreator, Diskadmin, Bulkadmin, установив флажок против этой роли на вкладке Server Role.

13. На вкладке Database Access выбрать требуемую базу данных, установив флажок слева от ее имени, и задать имя пользователя, в которое будет отображаться рассматриваемая учетная запись, а в нижней части вкладки с помощью флажка включить пользователя в ту или иную роль в зависимости от его работы с базой данных.

14. Щелкнув по кнопке Properties (свойства) и просмотреть список пользователей, включенных в выбранную роль рассматриваемой базы данных.

15. Щелкнув по кнопке Permissions (права) и просмотреть список прав, предоставленных выбранной роли базы данных.

16. Закрыть все окна.

17. Вновь открыть список учетных записей сервера, дважды щелкнуть по вновь созданной записи и проверить правильность введенных параметров.

18. Закрыть все окна.

19. Приступить к работе с базами данных, используя новую учетную запись.

Задание 2. Включить учетную запись пользователя или группы пользователей Windows в фиксированную роль сервера SQL с помощью Enterprise Manager, выполнив следующие действия:

1. Выбрать требуемый сервер в левом окне Tree.
2. Открыть объекты сервера, щелкнув по его кнопке “+”.
3. Открыть объекты Security этого сервера, щелкнув по кнопке “+” для Security.
4. Выбрать объект Logins (записи) и щелкнуть по нему, при этом в правом окне откроется список учетных записей сервера.

5. Дважды щелкнуть по требуемой учетной записи сервера.

6. В открывшемся окне на Server Login Properties открыть вкладку Server role.

7. Установить флажки возле тех ролей сервера, в которые требуется включить конфигурируемую запись.

8. Закрыть все открытые окна, щелкая по кнопкам “ОК” этих окон.

9. Повторить задания, используя следующий набор команд:

а) Security/Server Roles;

б) Щелкнуть левой клавишей;

в) В правом окне выбрать требуемую фиксированную роль; г) Два раза щелкнуть по выбранной роли;

д) В открывшемся окне Server Role Properties щелкнуть по кнопке Add вкладки General;

е) Добавить учетные записи в заданную роль; ж)

Закрыть окно со списком учетных записей;

з) На вкладке Permission окна Server Role Properties просмотреть предоставляемые права для рассматриваемой роли.

10. Закрыть все открытые диалоговые окна, щелкая по кнопкам ОК.

11. Проверить правильность выполненных действий.

Задание 3. Создать нового пользователя базы данных для учетной записи Windows с помощью Enterprise Manager, выполнив следующие действия:

1. Выбрать требуемый сервер и требуемую базу данных в левом окне Tree.
2. Открыть объекты выбранной базы данных, щелкнув по значку "+" этой базы.
3. Выбрать в раскрывшемся списке объектов рассматриваемой базы данных объект Users (пользователи).
4. Щелкнуть правой клавишей мыши и открыть контекстное меню объекта Users (пользователи).
5. В контекстном меню исполнить команду New Database User (новый пользователь базы данных).
6. В открывшемся диалоговом окне ввести:
 - а) в поле Login Name – имя учетной записи пользователя или группы пользователей Windows;
 - б) в поле User Name – имя нового пользователя рассматриваемой базы данных.
7. Включить нового пользователя в необходимые роли базы данных: public, db – owner, db – denydatareader и т.д. Для этого требуемые роли надо отметить флажками в списке фиксированных ролей базы данных, расположенном в правой части окна.
8. Щелкнуть по кнопке Properties и, просмотрев список всех пользователей базы данных, убедиться, что новый пользователь включен этот список.
9. Щелкнуть по кнопке Permission и задать права доступа пользователя к объектам базы данных: SELECT, INSERT, UPDATE, DELETE, EXEC, DRI. В окне находится полный список объектов базы данных.
10. Щелкнуть по кнопке Columns (столбцы) для выбранной базы данных и задать права доступа к конкретным столбцам таблицы: SELECT и/или UPDATE.
11. Закрыть все открытые диалоговые окна, щелкая по кнопкам ОК.
12. Проверить работу нового пользователя с рассматриваемой базой данных и его права.

Задание 4. Создать учетную запись SQL сервера, используя мастер Create Login Wizard, выполнив следующие действия:

1. Выполнить команду в Enterprise Manager Run an Wizard/Create Login Wizard.
2. В открывшемся окне мастера ознакомиться с этапами создания учетной записи сервера:
 - а) Select an authentication mode – выбрать режим аутентификации;
 - б) Grant access to security roles – представить доступ секретным ролям; в) Grant access to databases – предоставить доступ к базам данных.
3. Щелкнуть по кнопке Next.
4. Выбрать режим аутентификации: Windows или SQL Server.
5. Щелкнуть по кнопке Next.
6. Если выбран режим аутентификации Windows, то в открывшемся окне в поле Windows account задать имя учетной записи или группы Windows и домена и указать тип доступа: Grant access to the server (предоставить доступ к серверу) или Deny access to the server (запретить доступ к серверу).
7. Если выбран режим аутентификации SQL Server, то помимо имени учетной записи, задаваемой в поле Login I указать пароль в поле Password (пароль) и в поле Confirm Password (подтвердить пароль). Этот пароль пользователь будет использовать при подключении к SQL серверу.
8. Щелкнуть по кнопке Next в том или в другом случае.

9. Включить учетную запись в требуемые фиксированные роли сервера,устанавливая против роли флажок.
10. Щелкнуть по кнопке Next.
11. Разрешить для учетной записи доступ к базам данных, отмечая их флажком.
12. Щелкнуть по кнопке Next.
13. Мастер автоматически создаст имена пользователей баз данных.
14. Проверить сделанные установки.

Задание5. Создать новую пользовательскую роль баз данных с помощьюEnterprise Manager, выполнив следующие действия:

1. Выбрать требуемую базу данных.
 2. Открыть объекты выбранной базы данных, щелкнув по значку “+” этой базы.
 3. Выбрать в открывшемся списке объект Role (роль).
 4. Открыть контекстное меню объекта Role (роль).
 5. Исполнить команду меню New Database Role (новая роль базы данных).
 6. В открывшемся диалоговом окне ввести имя роли в поле Name, которое должно быть уникальным в пределах базы данных.
 7. Выбрать тип роли: стандартная роль Standart role или роль приложения Application Role.
 8. Если выбрана стандартная роль, то с помощью кнопки “добавить ” Add включить в нее нужных пользователей.
 9. Если выбрана роль приложения, то ввести в поле Password пароль, который будет использоваться для инициализации данной роли приложения. Нельзя добавлять пользователей в роль приложения.
 10. Для созданной стандартной роли или роли приложения (для пользовательской роли) задать права доступа к объектам базы данных на вкладке Permissions (права),выполнив действия:
 - а) выделить очередной объект базы данных;
 - б) для каждого права: SELECT, INSERT, UPDATE, DELETE, EXEC и DRI –установить одно из трех состояний доступа:
V – GRANT – предоставить,
X – DENI – запретить,
– REVOKE – неявное отклонение (т.е. может иметь доступ через членство роли).Закрывать все окна, щелкая по кнопке “ОК” каждого окна.
- Проверьте правильность выполненных действий.

Практическая работа №27-28 Мониторинг безопасности работы с базами данных

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     | Перископия |
+----+-----+
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 IP-адресом 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 SQL-запросы.

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 | root | |
| production.mysql.com | root | |
+-----+-----+-----+
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
<i>ALL</i> [<i>PRIVILEGES</i>]	Комбинация всех привилегий, за исключением привилегии <i>GRANT OPTION</i> , которая задается отдельно
<i>ALTER</i>	Позволяет редактировать таблицу с помощью оператора <i>ALTER TABLE</i>
<i>ALTER ROUTINE</i>	Позволяет редактировать или удалять хранимую процедуру
<i>CREATE</i>	Позволяет создавать таблицу при помощи оператора <i>CREATE TABLE</i>
<i>CREATE ROUTINE</i>	Позволяет создавать хранимую процедуру
<i>CREATE TEMPORARY TABLES</i>	Позволяет создавать временные таблицы
<i>CREATE USER</i>	Позволяет работать с учетными записями с помощью <i>CREATE USER</i> , <i>DROP USER</i> , <i>RENAME USER</i> и <i>REVOKE ALL PRIVILEGES</i>
<i>CREATE VIEW</i>	Позволяет создавать представление с помощью оператора <i>CREATE VIEW</i>
<i>DELETE</i>	Позволяет удалять записи при помощи оператора <i>DELETE</i>
<i>DROP</i>	Позволяет удалять таблицы при помощи оператора <i>DROP TABLE</i>
<i>EXECUTE</i>	Позволяет выполнять хранимые процедуры
<i>INDEX</i>	Позволяет работать с индексами, в частности, использовать операторы <i>CREATE INDEX</i> и <i>DROP INDEX</i>
<i>INSERT</i>	Позволяет добавлять в таблицу новые записи оператором <i>INSERT</i>
<i>LOCK TABLES</i>	Позволяет осуществлять блокировки таблиц при помощи операторов <i>LOCK TABLES</i> и <i>UNLOCK TABLES</i> . Для вступления в действие этой привилегии должна быть установлена привилегия <i>SELECT</i>
<i>SELECT</i>	Позволяет осуществлять выборки таблиц оператором <i>SELECT</i>
<i>SHOW DATABASES</i>	Позволяет просматривать список всех таблиц на сервере при помощи оператора <i>SHOW DATABASES</i>

<i>SHOW VIEW</i>	Позволяет использовать оператор <i>SHOW CREATE VIEW</i>
<i>UPDATE</i>	Позволяет обновлять содержимое таблиц оператором <i>UPDATE</i>
<i>USAGE</i>	Синоним для статуса «отсутствуют привилегии»
<i>GRANT OPTION</i>	Позволяет управлять привилегиями других пользователей, без данной привилегии невозможно выполнить операторы <i>GRANT</i> и <i>REVOKE</i>

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 <i>db</i>
ON db.tbl	Уровень таблицы – привилегии распространяются на таблицу <i>tbl</i> базы данных <i>db</i>
ON db.tbl	Уровень столбца – привилегии касаются отдельных столбцов в таблице <i>tbl</i> базы данных <i>db</i> . Список столбцов указывается в скобках через запятую после ключевых слов <i>SELECT</i> , <i>INSERT</i> , <i>UPDATE</i>

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)
```

Варианты заданий к практическим работам по MySQL

1. *Страховая компания.* Страховая компания имеет филиалы, которые характеризуются наименованием, адресом и телефоном. В филиалы обращаются клиенты с целью заключения договора о страховании. В зависимости от принимаемых на страхование объектов и страхуемых рисков договор заключается по определенному виду страхования (страхование автотранспорта от угона, страхование домашнего имущества, добровольное медицинское страхование). При заключении договора фиксируются: дата заключения, страховая сумма, вид страхования, тарифная ставка и филиал, в котором заключался договор. Договоры заключают страховые агенты. Помимо информации об агентах (фамилия, имя, отчество, адрес, телефон) нужно хранить филиал, в котором они работают. Необходимо иметь возможность рассчитывать заработную плату агентам. Зарплата составляет некоторый процент от страхового платежа (платеж – страховая сумма, умноженная на тарифную ставку). Процент зависит от вида страхования, по которому заключен договор.

2. *Гостиница.* Гостиница предоставляет номера клиентам. Каждый номер характеризуется вместимостью, комфортностью (люкс, полулюкс, обычный) и ценой. О клиентах собирается

определенная информация (фамилия, имя, отчество, паспортные данные, адрес жительства и некоторый комментарий). Сдача номера клиенту производится при наличии свободных мест в номерах, подходящих клиенту по указанным выше параметрам. При заселении фиксируется дата заселения. При выезде из гостиницы для каждого места запоминается дата освобождения. Необходимо также осуществлять бронирование номеров. Для постоянных клиентов, а также для определенных категорий клиентов предусмотрена система скидок. Скидки могут суммироваться.

3. *Ломбард*. В ломбард обращаются различные лица с целью получения денежных средств под залог товаров. Клиент сообщает фамилию, имя, отчество и другие паспортные данные. После оценивания стоимости принесенного в качестве залога товара работник ломбарда определяет сумму, которую готов выдать на руки клиенту, а также свои комиссионные. Кроме того определяется срок возврата денег. Если клиент согласен, то договоренности фиксируются в виде документа, деньги выдаются клиенту, а товар остается в ломбарде. Если в указанный срок не происходит возврата денег, товар переходит в собственность ломбарда. После перехода прав собственности на товар, ломбард может продавать товары по цене, меньшей или большей, чем была заявлена при сдаче. Цена может меняться несколько раз, в зависимости от ситуации на рынке (например, владелец ломбарда может устроить распродажу зимних вещей в конце зимы). Помимо текущей цены нужно хранить все возможные значения цены для данного товара.

4. *Оптово-розничная продажа товаров*. Компания торгует товарами из определенного спектра. Каждый товар характеризуется наименованием, оптовой ценой, розничной ценой и справочной информацией. В компанию обращаются покупатели, для каждого из которых в базе данных фиксируются стандартные данные (наименование, адрес, телефон, контактное лицо). По каждой сделке составляется документ, в котором наряду с покупателем фиксируются количество купленного им товара и дата покупки. Обычно покупатели в рамках одной сделки покупают не один товар, а сразу несколько. Также компания решила предоставлять скидки в зависимости от количества закупленных товаров и их общей стоимости.

5. *Ведение заказов*. Компания занимается оптовой продажей различных товаров. Каждый из товаров характеризуется ценой, справочной информацией и признаком наличия или отсутствия доставки. В компанию обращаются заказчики. Для каждого из них в базе данных запоминаются стандартные данные (наименование, адрес, телефон, контактное лицо). По каждому заказу составляется документ, в котором наряду с заказчиком фиксируются количество купленного им товара и дата покупки. Доставка товаров может производиться способами, различными по цене и скорости. Нужно хранить информацию о том, какими способами может осуществляться доставка каждого товара, и информацию о том, какой вид доставки (и какую стоимость доставки) выбрал клиент при заключении сделки.

6. *Бюро по трудоустройству*. Бюро готово искать работников для различных работодателей и вакансии для ищущих работу специалистов различного профиля. При обращении в бюро работодателя его стандартные данные (название, вид деятельности, адрес, телефон) фиксируются в базе данных. При обращении в бюро соискателя его стандартные данные (фамилия, имя, отчество, квалификация, профессия, иные данные) также фиксируются в базе данных. По каждому факту удовлетворения интересов обеих сторон составляется документ. В документе указываются соискатель, работодатель, должность и комиссионные (доход бюро). В базе должна фиксироваться не только сделка, но и информация по открытым вакансиям. Кроме того для автоматического поиска вариантов необходимо вести справочник «Виды деятельности».

7. *Нотариальная контора*. Нотариальная контора готова предоставить клиенту определенный комплекс услуг. Услуги формализованы, т. е. составлен их список с описанием каждой услуги. При обращении клиента его стандартные данные (название, вид деятельности, адрес, телефон) фиксируются в базе данных. По каждому факту оказания услуги клиенту составляется документ, в котором указываются дата, услуга, сумма сделки, комиссионные (доход конторы), описание сделки. В рамках одной сделки клиенту может быть оказано несколько услуг. Стоимость каждой услуги фиксирована. Кроме того компания предоставляет в рамках одной сделки различные виды скидок. Скидки могут суммироваться.

8. *Фирма по продаже запчастей*. Фирма продает запасные части для автомобилей. Фирма имеет определенный набор поставщиков, по которым известны название, адрес и телефон. У поставщиков приобретаются детали. Каждая деталь характеризуется названием, артикулом и ценой. Некоторые из поставщиков могут поставлять одинаковые детали (один артикул). Каждый факт покупки запчастей у поставщика фиксируется в базе данных, причем обязательными для запоминания являются дата покупки и количество приобретенных деталей. Цена детали может

меняться от поставки к поставке. Поставщики заранее ставят фирму в известность о дате изменения цены и ее новом значении. Нужно хранить не только текущее значение цены, но и всю историю изменения цен.

9. *Курсы по повышению квалификации.* В учебном заведении организованы курсы повышения квалификации. Группы слушателей формируются в зависимости от специальности и отделения. В каждую из них включено определенное количество слушателей. Проведение занятий обеспечивает штат преподавателей, для каждого из которых в базе данных зарегистрированы стандартные анкетные данные (фамилия, имя, отчество, телефон) и стаж работы. В результате распределения нагрузки получена информация о том, сколько часов занятий проводит каждый преподаватель с соответствующими группами. Хранятся также сведения о виде занятий (лекция, практика), дисциплине и оплате за 1 час. Размер почасовой оплаты зависит от предмета и типа занятия. Кроме того каждый преподаватель может вести не все предметы, а только некоторые.

10. *Определение факультативов для студентов.* Преподаватели кафедры в высшем учебном заведении обеспечивают проведение факультативных занятий по некоторым предметам. Имеются сведения о студентах, включающие стандартные анкетные данные (фамилия, имя, отчество, группа, адрес, телефон). По каждому факультативу существует определенное количество часов и вид проводимых занятий (лекции, практика, лабораторные работы). В результате работы со студентами появляется информация о том, кто из них записался на какие факультативы. Существует некоторый минимальный объем факультативных предметов, которые должен прослушать каждый студент. По окончании семестра в базу данных заносится информация об оценках, полученных студентами на экзаменах. Некоторые из факультативов могут длиться более одного семестра. В каждом семестре для предмета устанавливается объем лекций, практик и лабораторных работ в часах. В качестве итоговой оценки за предмет берется последняя оценка, полученная студентом.

11. *Распределение учебной нагрузки.* Необходимо распределять нагрузку между преподавателями кафедры. Имеются сведения о преподавателях, включающие наряду с анкетными данными сведения об их ученой степени, занимаемой должности и стаже работы. Преподаватели кафедры должны обеспечить проведение занятий по некоторым дисциплинам. По каждой из них существует определенное количество часов. В результате распределения нагрузки необходимо получить информацию следующего рода: «Такой-то преподаватель проводит занятия по такой-то дисциплине с такой-то группой». Все проводимые занятия делятся на лекционные и практические. По каждому виду занятий устанавливается свое количество часов. Кроме того данные по нагрузке нужно хранить несколько лет.

12. *Распределение дополнительных обязанностей.* Кафедра вуза имеет штат сотрудников, каждый из которых получает определенный оклад. Каждый месяц возникает потребность в выполнении некоторой дополнительной работы, не входящей в круг основных обязанностей сотрудников. Для наведения порядка в этой сфере классифицированы все виды дополнительных работ и определена сумма оплаты по факту их выполнения. При возникновении дополнительной работы назначается ответственный, фиксируется дата начала. По факту окончания фиксируется дата и выплачивается дополнительная сумма к зарплате с учетом классификации. Необходимо учесть разделение сотрудников на преподавателей и учебно-вспомогательный персонал. Для первых нужно хранить сведения об ученой степени и ученом звании, для вторых – о должности. Некоторые работы являются трудоемкими и срочными, что требует привлечения к их выполнению нескольких сотрудников. Длительность работ различна. Нужно заранее планировать длительность работы и количество сотрудников, занятых для выполнения работы.

13. *Техническое обслуживание станков.* Компания занимается ремонтом станков и другого оборудования. Клиентами компании являются промышленные предприятия. Ремонтные работы организованы следующим образом: все станки классифицированы по типам, странам-производителям, годам выпуска и маркам. Все виды ремонта отличаются названием, продолжительностью в днях, стоимостью. Исходя из этих данных, по каждому факту ремонта фиксируется вид станка, дата начала и дата окончания ремонта. Анализ показал, что нужно не просто подразделять станки по типам, а иметь информацию о том, сколько раз ремонтировался тот или иной станок.

14. *Туристическая фирма.* Фирма продает путевки клиентам. У каждого клиента запрашиваются стандартные данные – фамилия, имя, отчество, адрес, телефон. После этого сотрудники компании выясняют у клиента, куда он хотел бы поехать отдыхать. Ему демонстрируются различные варианты, включающие страну проживания, особенности климата,

отель. Обсуждается длительность пребывания и стоимость путевки. Если удалось найти приемлемый вариант, регистрируется факт продажи путевки (или путевок, если клиент покупает сразу несколько), фиксируется дата отправления. Иногда клиенту предоставляется скидка (скидки фиксированы и могут суммироваться). Фирма работает с несколькими отелями (название, категория, адрес) в нескольких странах. Путевки продаются на одну, две или четыре недели. Стоимость путевки зависит от длительности тура и отеля.

15. *Грузовые перевозки.* Компания осуществляет перевозки грузов по различным маршрутам. Необходимо отслеживать стоимость перевозок с учетом заработной платы водителей. Для каждого маршрута определено название, вычислено примерное расстояние и установлена некоторая оплата для водителя. Информация о водителях включает фамилию, имя, отчество и стаж. Для проведения расчетов хранится полная информация о перевозках (маршрут, водитель, даты отправки и прибытия). По факту некоторых перевозок водителям выплачивается премия. Фирма решила ввести гибкую систему оплаты. Оплата водителям должна зависеть не только от маршрута, но и от стажа водителя. Кроме того, нужно учесть, что перевозку могут осуществлять два водителя.

16. *Учет телефонных переговоров.* Телефонная компания предоставляет абонентам телефонные линии для междугородних переговоров. Абонентами компании являются юридические лица, имеющие телефонную точку, ИНН, расчетный счет в банке. Стоимость переговоров зависит от города, в который осуществляется звонок, и времени суток (день, ночь). Каждый звонок абонента автоматически фиксируется в базе данных. При этом запоминаются город, дата, длительность разговора и время суток. Компания решила ввести гибкую систему скидок. Так, стоимость минуты теперь уменьшается в зависимости от длительности разговора. Размер скидки для каждого города разный.

17. *Учет внутриофисных расходов.* Сотрудники частной фирмы могут осуществлять мелкие покупки для нужд фирмы, предоставляя в бухгалтерию товарный чек. Бухгалтерия отслеживает внутриофисные расходы. Фирма состоит из отделов, каждый из которых имеет название. В каждом отделе работает определенное количество сотрудников. Сотрудники могут осуществлять покупки в соответствии с видами расходов. Каждый вид расходов имеет название, некоторое описание и предельную сумму средств, которые могут быть потрачены по данному виду расходов в месяц. При каждой покупке сотрудник оформляет документ, где указывает вид расхода, дату, сумму и отдел. Нужно хранить данные о расходах не только в целом по отделу, но и по отдельным сотрудникам. Нормативы по расходованию средств устанавливаются не в целом, а по каждому отделу за каждый месяц. Неиспользованные в текущем месяце деньги могут быть использованы позже.

18. *Библиотека.* Библиотека решила зарабатывать деньги, выдавая напрокат книги, имеющиеся в небольшом количестве экземпляров. У каждой книги, выдаваемой в прокат, есть название, автор, жанр. В зависимости от ценности книги для каждой из них определена залоговая стоимость (сумма, вносимая клиентом при взятии книги напрокат) и стоимость проката (сумма, которую клиент платит при возврате книги, получая назад залог). Читатели регистрируются в картотеке, которая содержит стандартные анкетные данные (фамилия, имя, отчество, адрес, телефон). Каждый читатель может обращаться в библиотеку несколько раз. Все обращения читателей фиксируются, при этом по каждому факту выдачи книги запоминаются дата выдачи и ожидаемая дата возврата. Стоимость проката книги должна зависеть не только от самой книги, но и от срока ее проката. Кроме того, необходимо добавить систему штрафов за вред, нанесенный книге, и систему скидок для некоторых категорий читателей.

19. *Прокат автомобилей.* Фирма, занимающаяся прокатом автомобилей, имеет автопарк, содержащий некоторое количество автомобилей различных марок, стоимостей и типов. Каждый автомобиль имеет свою стоимость проката. В пункт проката обращаются клиенты. Клиенты проходят обязательную регистрацию, в ходе которой о них собирается стандартная информация (фамилия, имя, отчество, адрес, телефон). Каждый клиент может обращаться в пункт проката несколько раз. Обращения клиентов фиксируются, при этом по каждой сделке запоминаются дата выдачи и ожидаемая дата возврата. Стоимость проката автомобиля должна зависеть не только от самого автомобиля, но и от срока его проката, а также от года выпуска. Также нужно ввести систему штрафов за возвращение автомобиля в ненадлежащем виде и систему скидок для постоянных клиентов.

20. *Выдача банком кредитов.* Коммерческий банк выдает кредиты юридическим лицам. В зависимости от условий получения кредита, процентной ставки и срока возврата все кредитные

операции делятся на несколько основных видов. Каждый из этих видов имеет свое название. Кредит может получить юридическое лицо (клиент), при регистрации предоставивший следующие сведения: название, вид собственности, адрес, телефон, контактное лицо. Каждый факт выдачи кредита регистрируется банком, при этом фиксируются сумма кредита, клиент и дата выдачи. Чтобы отслеживать динамику возврата кредитов принято решение учитывать в системе еще и дату фактического возврата денег. Нужно еще учесть, что кредит может гаситься частями, и за задержку возврата кредита начисляются штрафы.

21. *Инвестиционная компания.* Компания занимается вложением денежных средств в ценные бумаги, которые характеризуются рейтингом, доходностью за прошлый год, минимальной суммой сделки и некоторой дополнительной информацией. Клиентами компании являются предприятия, которые доверяют ей управлять их свободными денежными средствами на определенный период. Необходимо выбрать вид ценных бумаг, которые позволят получить прибыль и компании и клиенту. При работе с клиентом существенной является информация о предприятии – название, вид собственности, адрес и телефон. Каждая инвестиция характеризуется информацией о клиенте, информацией о ценной бумаге, котировкой бумаги, датой ее покупки и датой ее продажи. Необходимо хранить историю котировок каждой ценной бумаги. Кроме того, помимо вложений в ценные бумаги существует возможность вкладывать деньги в банковские депозиты.

22. *Занятость актеров театра.* Коммерческий директор театра организует привлечение актеров и заключение контрактов. Каждый год театр осуществляет постановку различных спектаклей. Каждый спектакль имеет определенный бюджет. Для участия в конкретных постановках в определенных ролях привлекаются актеры. С каждым из актеров заключается персональный контракт на определенную сумму. Каждый актер имеет стаж работы, некоторые из них удостоены различных званий. В рамках одного спектакля на одну и ту же роль привлекается несколько актеров. Контракт определяет базовую зарплату актера, а по итогам реально отыгранных спектаклей актеру назначается премия. В базе данных нужно хранить информацию за несколько лет.

23. *Платная поликлиника.* В поликлинике работают врачи различных специальностей, имеющие разную квалификацию. Каждый день в поликлинику обращаются пациенты. Все пациенты проходят обязательную регистрацию, при которой в базу данных заносятся стандартные анкетные данные (фамилия, имя, отчество, год рождения, адрес). При обращении в поликлинику пациент обследуется и проходит лечение у разных специалистов. Каждый пациент может обращаться в поликлинику несколько раз, нуждаясь в различной медицинской помощи. Все обращения пациентов фиксируются, при этом устанавливается диагноз, определяется стоимость лечения, запоминается дата обращения. Общая стоимость лечения зависит от стоимости консультаций и процедур, назначенных пациенту. Для определенных категорий граждан предусмотрены скидки.

24. *Анализ динамики показателей финансовой отчетности предприятий.* Информационно-аналитический центр крупного холдинга отслеживает динамику показателей предприятий холдинга. В структуру холдинга входят несколько предприятий. Каждое предприятие имеет стандартные характеристики (название, реквизиты, телефон, контактное лицо). Работа предприятия может быть оценена следующим образом: в начале каждого отчетного периода на основе финансовой отчетности вычисляется определенный набор показателей. Важность показателей характеризуется некоторыми числовыми константами. Значение каждого показателя измеряется в некоторой системе единиц. Некоторые показатели считаются в рублях, некоторые в долларах, некоторые в евро. Для удобства работы с показателями нужно хранить изменения курсов валют относительно друг друга.

25. *Учет телекомпанией стоимости прошедшей в эфире рекламы.* Работа коммерческой службы телевизионной компании построена следующим образом: заказчики просят поместить свою рекламу в определенной передаче в определенный день. Каждый рекламный ролик имеет определенную продолжительность. Для каждой организации-заказчика известны банковские реквизиты, телефон и контактное лицо для проведения переговоров. Передачи имеют определенный рейтинг. Стоимость минуты рекламы в каждой передаче определяется, исходя из рейтинга передачи и прочих соображений. Необходимо хранить информацию об агентах, заключивших договоры на рекламу. Зарплата рекламных агентов составляет некоторый процент от общей стоимости рекламы, прошедшей в эфире.

26. *IT-компания.* Компания оказывает IT-услуги организациям и предприятиям. В компании работают сотрудники, о которых должна сохраняться стандартная информация и данные о квалификации (владение языками и системами программирования, знание СУБД, операционных систем). В компанию обращаются клиенты, о которых собираются стандартные данные (наименование и адрес организации, телефон, адрес электронной почты, фамилия, имя и отчество контактного лица для связи). Задание для клиента выполняет определенный сотрудник, при этом фиксируется дата выдачи задания и трудоемкость выполнения (в часах). При повторном обращении клиент переходит в категорию постоянных и получает скидку. С ростом компании возникла необходимость разделения ее на отделы. Увеличились масштабы проектов, и теперь задание клиента поручается отделу. В рамках одного договора может выполняться несколько заданий разными отделами компании.

27. *Ювелирная мастерская.* Ювелирная мастерская осуществляет изготовление ювелирных изделий для частных лиц на заказ. Мастерская работает с определенными материалами (платина, золото, серебро, драгоценные камни). При обращении потенциального клиента выясняется, какое именно изделие ему необходимо. Все изготавливаемые изделия принадлежат к некоторому типу (серьги, кольца, броши, браслеты), выполняются из определенного материала, имеют некоторый вес и цену (включающую стоимость материалов и работы). Ювелирное изделие может состоять из нескольких материалов. Кроме того, постоянным клиентам мастерская предоставляет скидки.

28. *Парикмахерская.* Парикмахерская стрижет клиентов в соответствии с их пожеланиями и некоторым каталогом различных видов стрижки. Для каждой стрижки определены название, категория (мужская, женская, детская), стоимость работы. Для наведения порядка составляется база данных клиентов, где запоминаются их анкетные данные (фамилия, имя, отчество). Начиная с 5-ой стрижки, клиент переходит в категорию постоянных и получает скидку в 3 % при каждой последующей стрижке. После того, как закончена очередная работа, в БД фиксируются стрижка, клиент и дата производства работ. Кроме того, у парикмахерской появился филиал и необходима раздельная статистика по филиалам. Стоимость стрижки может меняться с течением времени. Нужно хранить не только последнюю цену, но и все данные по изменению цены стрижки.

29. *Химчистка.* Химчистка осуществляет прием у населения вещей для выведения пятен. Для наведения порядка составляется база данных клиентов, в которой запоминаются их анкетные данные (фамилия, имя, отчество, адрес, телефон). Начиная с 3-го обращения, клиент переходит в категорию постоянных клиентов и получает скидку в 3 % при чистке каждой последующей вещи. Все оказываемые услуги подразделяются на виды, имеющие название, тип и стоимость, зависящую от сложности работ. Работа с клиентом первоначально состоит в определении объема работ, вида услуги и, соответственно, ее стоимости. Если клиент согласен, он оставляет вещь (при этом фиксируется услуга, клиент и дата приема) и забирает ее после обработки (при этом фиксируется дата возврата). Химчистка заключает с клиентом договор. Клиент может одновременно сдавать в чистку несколько вещей. У химчистки появились филиалы, и необходима раздельная статистика по филиалам. Введены надбавки за срочность и сложность.

30. *Сдача в аренду торговых площадей.* Торговый центр сдает в аренду коммерсантам свои торговые площади. В результате планирования определено некоторое количество торговых точек в пределах здания, которые могут сдаваться в аренду. Для каждой из торговых точек важными данными являются этаж, площадь, наличие кондиционера и стоимость аренды в день. С потенциальных клиентов собираются стандартные данные (название, адрес, телефон, реквизиты, контактное лицо). При появлении потенциального клиента ему показывают имеющиеся свободные площади. При достижении соглашения оформляется договор и в базе данных фиксируется торговая точка, клиент, период (срок) аренды. Некоторые клиенты в рамках одного договора арендуют сразу несколько торговых точек, причем для каждой точки возможен свой срок аренды. Дата заключения договора может не совпадать с датой начала аренды. Необходимо собирать информацию об ежемесячных платежах, поступающих от арендаторов.

Практическая работа №29-30. Установка приоритетов

15.1 Цель практической работы

Приобретение навыков регистрации удаленных серверов с помощью утилиты Enterprise Manager, мастера Register Server Wizard, а также команд Transact-SQL и системной хранимой процедуры sp-addserver. Приобретение навыков управления основной службой MSSQLServer и вспомогательными службами сервера MS SQL Server: задание режима автоматического запуска службы, ручной запуск службы, запуск сервера в однопользовательском режиме, с минимальными требованиями и нестандартной конфигурации, приостановка службы и остановка служб и сервера.

Методические рекомендации для выполнения практической работы

Регистрация удаленных серверов

Перед использованием локального или удаленного сервера в среде Enterprise Manager его необходимо зарегистрировать.

При запуске Enterprise Manager первый раз регистрация локального сервера происходит автоматически. После установки дополнительных копий они также регистрируются в Enterprise Manager автоматически. И только при работе с удаленными серверами их необходимо регистрировать, используя среду Enterprise Manager для выполнения команд меню, запуска Register Server Wizard или интерпретации команд Transact-SQL и системной хранимой процедуры sp-addserver.

При регистрации сервера необходимо указать следующую информацию:

1. Имя сервера.
2. Тип безопасности, используемый для входа на сервер.
3. Имя учетной записи и пароль, используемые для входа на сервер.
4. Имя группы в иерархии групп, в которой необходимо зарегистрировать сервер.

Утилита Enterprise Manager представляет собой всего лишь графический интерфейс для выполнения системных хранимых процедур SQL Server 2000. Таким образом, она является клиентским средством, устанавливающим соединение с SQL Server и выполняющим те или иные процедуры. Поэтому, прежде чем эта программа сможет выполнить какие-либо операции с сервером, она должна получить соответствующие права доступа, т.е. пройти аутентификацию.

15.2.2 Запуск, остановка и приостановка служб сервера

До выполнения каких-либо работ по администрированию сервера MS SQL Server или баз данных, а также манипулированию данными необходимо запустить сервер. Точнее говоря, запустить его основную службу MSSQLServer. Только после запуска этой службы и проверки прав доступа пользователя, пользователь сможет выполнять функции, определенные его правами и разрешениями. Остальные службы являются вспомогательными, и их работа строится на фундаменте, обеспечиваемом службой MSSQLServer. Например, служба SQLServerAgent запускается лишь тогда, когда требуется автоматическое администрирование и управление системой на базе SQL Server. Служба MSSearch используется для работы с электронными документами, обеспечивает полнотекстовый поиск информации и, как правило, используется автономно. Служба MSDTC позволяет организовать доступ к распределенным источникам информации и управлять распределенными транзакциями.

Дополнительные службы запускаются отдельно и устанавливают соединение с сервером, подобно обычным клиентам. Каждая такая служба самостоятельно подключается к основной службе MSSQLServer, используя определенные учетные записи с соответствующими правами доступа.

Для сетевого варианта установки управлять службами можно как локально, так и удаленно даже средствами операционной системы. Для операционной системы Windows 98 можно запустить только один экземпляр сервера в качестве приложения, так как в Windows 98 нет служб, и управлять этим приложением локально. Запускать, останавливать и приостанавливать сервер можно также при отсутствии сети.

Задание к практической работе 15

Задание 1. Произвести регистрацию удаленного сервера с помощью окна параметров регистрации сервера Register SQL Server Properties утилиты Enterprise Manager, выполнив действия:

1. На дереве объектов консоли выбрать одну из групп серверов, где будет зарегистрирован удаленный сервер.

2. Открыть контекстное меню группы серверов и выполнить команду New SQL Server Registration.

3. В открывшемся окне Register SQL Server Properties задать следующие параметры:

а) Имя удаленного сервера в виде следующей записи: сетевое имя NetBios соответствующего компьютера, косая черта «\», имя копии сервера (для сервера по умолчанию это имя копии можно не задавать);

б) Учетную запись, которая будет использоваться для установления соединения с соответствующим сервером: либо учетная запись домена Windows NT и ее набор прав в SQL Server, либо учетная запись сервера, созданная на регистрируемом сервере и включающая входное имя пользователя Login Name и его пароль Password, с указанием режима подключения с вводом пароля при каждом соединении или без ввода пароля;

в) Имя группы серверов из числа имеющихся или имя новой группы, которую можно создать, щелкнув по кнопке с многоточием в том же окне в области Options;

г) Установить с2k, если необходимо, следующие переключатели: Display SQL Server state in console – показывать состояние сервера в окне объектов Enterprise Manager; Automatically start SQL Server when connecting – автоматически запускать сервер при соединении; Show system database and system table – отображать системные базы данных и таблиц.

Задание 2. Произвести регистрацию удаленного сервера с помощью мастера Register Server Wizard, выполнив следующие действия:

1. Щелкнуть на кнопке Run a Wizard панели инструментов Enterprise Manager.

2. В открывшемся окне, вид которого зависит от левого окна Enterprise Manager (выбран или не выбран сервер или папка группы серверов), выбрать мастер регистрации сервера Register Server Wizard.

3. Щелкнуть по кнопке ОК.

4. В первом окне мастера предлагается следующий порядок работы:

а) выбрать SQL сервер;

б) выбрать режим аутентификации;

в) определить группу SQL серверов.

5. В этом же окне сбросить флажок From now on, I want to perform this task without using a wizard (теперь я хотел бы выполнить задачу без использования мастера), иначе мастер закончит свою работу, открыв окно Register SQL Server Properties для ручной регистрации сервера

6. Щелкнуть по кнопке Next.

7. Во втором окне выбрать или ввести имя регистрируемого сервера в левой части окна.

8. Щелкнуть по кнопке добавить Add. В случае ошибки использовать кнопку удалить Remove. Если регистрируется сразу несколько серверов, то они будут включены в одну и ту же группу с одинаковыми параметрами и с одной и той же учетной записью для установления соединения.

9. Щелкнуть по кнопке Next.

10. В третьем окне необходимо выбрать режим аутентификации при подключении к регистрируемому серверу.

11. Если использовать аутентификацию Windows NT, то возможность подключения к серверу будет зависеть от учетной записи, под которой работает SQL Server.

12. Если выбрать аутентификацию SQL Server, то для установления соединения потребуется имя и пароль учетной записи, предварительно созданной на регистрируемом сервере SQL Server. В этом случае открывается окно, в котором надо сделать выбор режим подключения к регистрируемому серверу:

а) с использованием сохраняемой одной и той же учетной записи, для которой надо в этом же окне ввести имя и пароль;

б) с использованием учетной записи, имя и пароль который надо вводить каждый раз при подключении к удаленному серверу.

13. Щелкнуть по кнопке Next и перейти к следующему окну мастера.

14. Выбрать или создать новую группу, в которую включить регистрируемый сервер.

15. Щелкнуть по кнопке Next и перейти в последнее окно мастера со списком регистрируемых серверов.

16. Щелкнуть по кнопке Finish.

17. Если регистрируемый сервер найден, то произойдет подключение к нему.

18. Если регистрируемый сервер не найден. То Enterprise Manager выдаст запрос на повторную регистрацию сервера.

19. Возможные ошибки при регистрации:

а) регистрируемый сервер был остановлен;

б) на компьютере, с которого выполняется регистрация используются иные сетевые библиотеки и протоколы, чем на регистрируемом сервера;

с) если сервер зарегистрирован с использованием аутентификации Windows NT, а для пользователей не создано соответствующей учетной записи на SQL Server (Login failed);

д) если используется аутентификация SQL Server и имя пароль заданы неверно.

Задание 3. Произвести регистрацию удаленного сервера, выполняя команду:EXEC sp.addserver @server = 'server', @local = 'local', @duplicate.ok = 'duplicate.ok'

Здесь параметры имеют следующее назначение:

Server – имя регистрируемого сервера, которое состоит из сетевого имени NetBios

соответствующего компьютера и имени копии сервера, между которыми ставится разделитель ”\”, при этом для копии сервера по умолчанию второе имя задавать не требуется;

Local – сервер для регистрации является локальным, иначе – сервер удаленный; duplicate.ok – разрешает дублирование имен серверов, что приводит к тому, что информация о новом сервере будет записана поверх старой информации.

Задание 4. Произвести настройку конфигурации утилиты Enterprise Manager, выполнив действия:

- исполнить команду Tools/Options утилиты;
- в открывшемся окне SQL Server Enterprise Manager Properties выбрать вкладку General.

- в группе Server state pooling (опрос состояния сервера) выбрать службу Service и задать количество секунд, через которое будет проводиться опрос состояния соответствующей службы, и отображаться это состояние в виде соответствующего значка.

- для конфигурирования одного из серверов в качестве центрального хранилища информации необходимо снять флажок Read/Store User Independent (независимое считывание/хранение пользователей), а на локальном сервере установить переключатель Read from remote (считывать с удаленного сервера) и указать имя удаленного сервера, с которого будет считываться информация о конфигурации.

- убедиться, что установленный флажок Read/Store User Independent означает коллективное использование информации о конфигурации, а сброшенный – личное, когда информация для каждого пользователя сохраняется отдельно.

Задание 5. Установить режим автоматического запуска служб SQL Server 2000, который производится автоматически операционной системой при ее запуске, выполнив следующие действия:

1. При установке сервера MS SQL Server 2000 задать режим автоматического запуска служб сервера. В этом случае сразу же после установки и каждый раз при запуске операционной системы все установленные на компьютере службы сервера будут запускаться автоматически.

2. Если режим автоматического запуска не был задан при установке или по каким-либо причинам был отключен в дальнейшем, то его можно задать следующими действиями (три варианта):

Войти в Enterprise Manager и выполнить команды:

- a) в его левом окне выбрать требуемый сервер, так как для каждого экземпляра, или копии сервера имеются отдельные экземпляры, или копии служб MSSQLServer, SQLServerAgent и MSDTC;

- b) щелкнуть правой клавишей мыши, чтобы открылось контекстное меню сервера;

- c) щелкнуть левой клавишей по элементу Properties (свойства);

- d) в открывшемся окне SQL Server Properties (свойства SQL Server) на вкладке General (общие) установить флажок для требуемых служб:

Autostart SQL Server; Autostart

SQL Server Agent; Autostart

MSDTC;

- e) щелкнуть по кнопке ОК;

- f) перезагрузить операционную систему и убедиться, что нужные службы запущены.

Войти в утилиту Services (Службы) операционной системы Windows NT или Windows, исполнив команду Пуск/Настройка/Панель управления/Службы (Start/.../Control panel/Services) и выполнить команды:

- a) в открывшемся окне служб Services выбрать требуемую службу;
- b) дважды щелкнуть по выбранной службе;
- c) в открывшемся окне свойств выбранной службы Properties на вкладке General (общие) раскрыть список Start type (тип запуска);
- d) в списке выбрать режим Automatic и щелкнуть по нему;
- e) щелкнуть по кнопке ОК;
- f) закрыть все окна операционной системы;
- g) перезагрузить операционную систему и убедиться, что все нужные службы запущены.

Войти в утилиту SQL Server Services Manager и в открывшемся окне с таким же названием выполнить команды:

- a) раскрыть список Server (сервер);
- b) щелкнуть по требуемому серверу;
- c) раскрыть список Services (службы) для этого сервера;
- d) щелкнуть по рассматриваемой службе;
- e) в открывшемся окне установить флажок Autostart service when OS start (автоматический старт при запуске операционной системы);
- f) закрыть окна утилиты Services Manager;
- g) перезагрузить операционную систему и убедиться, что все нужные службы запущены.

Задание 6. Произвести ручной запуск службы SQL Server 2000 одним из следующих четырёх способов:

1. Войти в Enterprise Manager и выполнить действия:
 - a) выбрать требуемый сервер;
 - b) открыть его контекстное меню;
 - c) щелкнуть по команде Start для запуска службы MSSQLServer;
 - d) для запуска службы SQLServerAgent надо открыть папу Management сервера и щелкнуть по команде Start;
 - e) для запуска служб MSDTC и SQLMail надо открыть папку Support Services и щелкнуть по команде Start для соответствующей службы.
2. Войти в утилиту SQL Server Service Manager, выбрать требуемый сервер и службу и щелкнуть по кнопке Start.
3. В командной строке запустить утилиту командной строки net start, указав в качестве параметра имя требуемой службы или экземпляра сервера:

```
net start mssqlserver net
start sqlserveragent
net start MSSQL$TRELON net
start SQLAgent$TRELON
net start
```

 для выдачи списка запущенных в ОС служб
4. Установить режим работы операционной системы сеанс DOS и в командной строке исполнить команду sqlserver для запуска сервера, как отдельного приложения операционной системы. В этом случае все средства администрирования системы SQL Server такие, как Service Manager, Enterprise Manager, Service (для панели управления) будут показывать, что сервер остановлен, и все системные сообщения будут появляться в консольном окне, в котором выполнена команда sqlserver. Сервер будет запущен под

учетной записью пользователя, и если необходимо завершить сеанс работы ОС, то сначала надо завершить работу SQL Server.

Задание 7. Запустить SQL Server в однопользовательском режиме, выполнив действия:

1. Убедиться, что все службы рассматриваемого сервера остановлены.
2. В командной строке исполнить команду: `sqlserver.exe -m`.
3. Приступить к конфигурированию характеристик сервера или восстановлению поврежденной системной базы, учитывая, что:
 - a) модифицированные страницы сразу записываются на диск, а не остаются, как обычно в кэш-памяти;
 - b) разрешен прямой доступ к системным таблицам с помощью команд INSERT< DELETE и UPDATE.

Задание 8. Произвести аварийный запуск сервера с минимальными требованиями для проведения восстановительных работ из-за неправильного конфигурирования:

1. Для запуска SQL Server 2000 как службы с минимальными требованиями исполнить команду в командной строке: `sqlserver.exe -f`.
2. Для запуска SQL Server 2000 как приложения с минимальными требованиями исполнить команду в командной строке: `sqlserver.exe -f -c`.
3. Для первого случая убедиться, что:
 - a) количество открытых баз данных, таблиц, открытых объектов, размер КЭШа процедур минимальны;
 - b) запрещено исполнение хранимых процедур;
 - c) установлен однопользовательский режим;
 - d) удаленный доступ запрещен;
 - e) разрешен прямой доступ к таблицам.

Задание 9. Приостановите, а затем и остановите работу служб сервера. Запустите их вновь.

Просмотрите параметры запуска в реестре по адресу

HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\MSSQLSERVER
\PARAMETERS

Практическая работа №31. Развертывание контроллеров домена

Цель практической работы

Ознакомление с основными концепциями и технологиями, лежащими в основе функционирования сервера, и реализующими и их компонентами: средствами администрирования, сетевыми библиотеками, службами, интерфейсами для создания клиентских приложений

Методические рекомендации для выполнения практической работы

SQL Server имеет множество инструментов для импорта и экспорта данных. Лучшим является служба преобразования данных Data Transformation Services (DTS), которая предоставляет набор инструментальных средств. Она также позволяет извлекать, преобразовывать и объединять данные из источников данных разной природы, расположенных как в одном, так и в разных местах. Можно управлять данными, используя инструментальные средства DTS, для графического построения пакетов DTS или создавая объектно-ориентированные пакеты DTS.

Пакет DTS – это объект, в котором хранится описание выполняемых в ходе импорта, экспорта и трансформации данных действий. Каждый пакет реализует один или большее количество шагов, которые выполняются последовательно или параллельно. С помощью пакета может выполняться копирование и преобразование данных и объектов баз данных. Пакеты можно редактировать, защищать паролем, конфигурировать для автоматического выполнения по расписанию.

Задание к практической работе №31

Задание1. Осуществить передачу данных с помощью мастера Data Transformation Services(DTS), используя способ Copy table(s) and view(s) from the source database(копировать таблицу(таблицы) и представление(представления) из источника),выполнив следующие действия:

1. Запустить мастер: Пуск \ Программы \ Microsoft SQL Server \ Import and Export Data.
2. В первом открывшемся окне, которое содержит общую информацию о работе мастера, щёлкнуть по кнопке Next.
3. Во втором окне в раскрывающемся списке Source(источник) необходимо выбрать тип источника данных; в списке Server(сервер) выбрать сервера-источника; указать список аутентификации; в списке Database выбрать базу данных, в которую будет осуществляться взаимодействие. После этого щёлкнуть по кнопке Next.
4. Для редактирования, по необходимости, параметров конфигурации щёлкнуть на кнопке Advanced(дополнительно).
5. В третьем окне сконфигурировать получатель: в раскрывающемся списке Database(база данных) выбрать пункт New(создать) и создать новую базу данных.
6. В четвёртом окне DTS Wizard выбрать способ передачи данных Copy table(s) and view(s) from the source database.
7. В пятом окне в столбце Source Table(таблица источник) выбрать одну или более таблиц или представлений для копирования.
8. Для того, чтобы увидеть содержание исходной таблицы, щёлкните на кнопке Preview(просмотр).
9. В столбце Destination(получатель) указать имя таблицы-получателя.
10. Если необходимо выполнить преобразование данных, то в столбце Transform(преобразование) для соответствующей таблицы щёлкните на кнопке с многоточием. В открывшемся окне можно настроить процесс трансформации не только самих данных, но и их типов.
11. Следующее окно мастера DTS Wizard (рис. 24.22) будет общим для всех способов переноса. В этом окне для созданного пакета DTS указать способ его сохранения.
12. Если выбрали вариант SQL Server, то необходимо установить параметры:
 - в поле Name(имя) указывается имя, под которым пакет DTS будет сохранен в системной базе данных msdb;
 - в поле Description (описание) можно ввести описание объекта в произвольной форме;
 - в поле Owner Password (пароль владельца), чтобы скрыть информацию, указанную при создании пакета, от просмотра неавторизованными пользователями, можно установить пароль владельца;
 - установив в поле User Password (пароль пользователя) пароль пользователя, можно запретить выполнение пакета пользователями, которые не имеют на это права. Только те пользователи, которые знают пароль, смогут выполнить пакет DTS.
 - в списке Server name (имя сервера) выбирается имя сервера, на котором будет сохранен пакет DTS.
13. При выборе режима хранения SQL Server Meta Data Services (службы метаданных SQL Server) мастер выведет окно, во многом напоминающее окно при режиме хранения SQL Server. Добавлена лишь кнопка Scanning (сканирование), с помощью которой можно установить взаимосвязи между объектами в источнике и получателе данных, сохраняемые в хранилище(первичный и внешний ключи, индексы, столбцы, типы данных и т. д.).

14. При выборе двух оставшихся режимов хранения мастер выведет окно, в котором помимо имени, описания, пароля пользователя и пароля владельца необходимо указать только имя файла, в который будет записан пакет.

На этом работа с мастером DTS Wizard по созданию пакетов для импорта экспорта данных заканчивается. В последнем окне (рис. 24.28) приведена сводная информация о созданном пакете. После щелчка на кнопке Finish(готово) будет создан сам пакет.

15. Если при создании пакета было задано его незамедлительное выполнение, то мастер откроет окно Executing Package (выполнение пакета), позволяющее следить за процессом выполнения пакета.

Задание 2. Осуществить передачу данных с помощью мастера Data Transformation Services(DTS), используя способ Use a query to specify the data to

transfer(использовать запрос для выборки данных), выполнив следующие действия:

1. Выполнить с первого по пятый пункты задания1.
2. В четвёртом окне мастера DTS Wizard установить переключатель Use a query to specify the data to transfer.
3. В открывшемся окне ввести SQL-код запроса; если имеется готовый код, сохранённый на диске, его можно подключить, воспользовавшись кнопкой Browse(обзор).
4. Если необходимо написать сложный запрос с перечислением множества таблиц и столбцов и при этом гарантировать, что указаны правильные имена объектов, можно

воспользоваться встроенным в мастер конструктором запросов. Для вызова конструктора запроса щёлкнуть на кнопке Query Builder (конструктор запросов).

Откроется окно, в котором нужно выбрать, какие столбцы, из каких таблиц будут включены в запрос.

5. Щёлкнуть по кнопке Next.

6. В открывшемся окне, перенося имена столбцов из левой части окна в правую, задать порядок сортировки, которая ведётся по столбцам, указанным в самом верху списка.

7. Щёлкнуть по кнопке Next.

8. В следующем окне указать критерии для выборки данных: установить указатель Only Rows meeting criteria(только строки, соответствующие критерию).

9. Если необходимости в фильтрации нет, установите переключатель All rows(все строки). Нажмите на кнопку ОК.

10. После того как редактирование запроса закончено, мастер откроет окно, в котором можно настроить трансформацию данных. Работа с этим окном практически ничем не отличается от работы по настройке трансформации данных при копировании между таблицами, описанной в задании 1.

11. После настройки трансформации данных необходимо сохранить пакет DTS одним из способов, которые указаны в предыдущем задании.

Задание 3. Осуществить передачу данных с помощью мастера Data Transformation Services(DTS), используя способ Copy objects and data between SQL Server databases(копировать объекты и данные между базами данных SQL Server), выполнив следующие действия:

1. Выполнить с первого по пятый пункты задания 1.

2. В четвёртом окне мастера DTS Wizard установить переключатель Copy objects and data between SQL Server databases.

3. Щёлкнуть на кнопке Next.

4. В открывшемся окне указать, какие объекты и данные будут копироваться: установка флажка Create destination objects – создание переносимых объектов; установка флажка Drop destination objects first – удаление всех одноимённых объектов из конечной базы данных; установка флажка Include all dependent objects – включение всех зависимых объектов; установка флажка Copy data – копирование только структуры объектов.

5. Установив флажок Copy all objects, выполняется копирование всех объектов.

6. Если необходимо скопировать только часть объектов, сбросьте флажок Copy all objects и выберите нужные объекты, щёлкнув на кнопке Select Objects(выбор объектов).

7. Чтобы выбрать только некоторые из них, в окне мастера сбросьте флажок Use default options(использовать параметры по умолчанию). После щёлчка на кнопке Options(параметры) в открывшемся окне укажите объекты, которые необходимо скопировать.

8. После указания объектов необходимо сохранить пакет DTS одним из способов, которые указаны в первом задании. После будет создан сам пакет.

Контрольные вопросы:

1. На каких критериях следует основываться при выборе метода импорта или экспорта данных.

2. Что собой представляет служба преобразования данных Data Transformation Services(DTS).

3. Какие способы передачи данных можно выделить, используя мастер Data Transformation Services (DTS) Import and Export Wizard.
4. Какими способами можно осуществить хранение пакета DTS.
5. Что собой представляет внутренняя структура пакета DTS.

Список литературы

Электронные издания (электронные ресурсы)

1. Голицына, О.Л. Основы проектирования баз данных [Электронный ресурс]: учеб. пособие / О.Л. Голицына, Т.Л. Партыка, И.И. Попов. – 2-е изд., перераб. и доп. – Москва: ФОРУМ: ИНФРА-М, 2019. – 416 с. - Режим доступа: <http://znanium.com/catalog/product/1018906> (ЭБС Znanium)
2. Шустова, Л.И. Базы данных [Электронный ресурс]: учебник / Л.И. Шустова, О.В. Тараканов. — М. : ИНФРА-М, 2019. – 304 с. + Доп. материалы .- Режим доступа: <http://znanium.com/catalog/product/967755> (ЭБС Znanium)
3. Стружкин, Н. П. Базы данных: проектирование. Практикум : учеб. пособие для СПО / Н. П. Стружкин, В. В. Годин. — Москва: Издательство Юрайт, 2019. – 291 с. - Текст : электронный // ЭБС Юрайт [сайт]. – Режим доступа: <https://biblio-online.ru/bcode/442343>
4. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для СПО / В. М. Илюшечкин. – испр. и доп. – Москва : Издательство Юрайт, 2019. – 213 с. – Текст: электронный // ЭБС Юрайт [сайт]. – Режим доступа: <https://biblio-online.ru/bcode/437670>

Методические издания

1. Игнатенко, Е.С. Методические указания по выполнению практических работ по ОП.08 Основы проектирования баз данных .– Нефтеюганск: НИК(филиал) ФГБОУ ВО «ЮГУ», 2019 [Электронный ресурс] Режим доступа: локальная сеть филиала

Периодические издания

1. Программные продукты и системы [Электронный ресурс]: журнал. - Тверь: – Научно-исследовательский институт «Центрпрограммсистем» Электронно-библиотечная система «Лань»: [сайт]. – Режим доступа: <https://e.lanbook.com/journal/2276>