ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ИРКУТСКОЙ ОБЛАСТИ «ЧЕРЕМХОВСКИЙ ГОРНОТЕХНИЧЕСКИЙ КОЛЛЕДЖ ИМ. М.И. ЩАДОВА»

РАССМОТРЕНО
на заседании ЦК
«Информатики и ВТ»
«04» февраля 2025 г.
«Протокол № 6
Председатель: Коровина Н.С.

УТВЕРЖДАЮ
Зам. директора
О.В. Папанова
« 26» мая 2025 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по практическим занятиям студентов

учебной дисциплины

ОП. 08 Основы проектирования баз данных

09.02.07 Информационные системы и программирование

Разработал:
Коровина Н.С.

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ

Сертификат: 54CE6FF6F2E941EC6571CFEA2EAFB5E4
Владелец: Сычев Сергей Николаевич
Действителен: с 23.08.2024 до 16.11.2025

2025 г.

СОДЕРЖАНИЕ

1.ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Методические указания по практическим занятиям учебной дисциплины **«Основы проектирования баз данных»** составлены в соответствии с учебным планом и рабочей программы дисциплины по специальности ***09.02.07 Информационные системы и программирование***

Цель проведения практических занятий: формирование практических умений, необходимых в последующей профессиональной и учебной деятельности.

Методические указания практических занятий являются частью учебно-методического комплекса по учебной дисциплине и содержат:

- тему занятия (согласно тематическому плану учебной дисциплины);
- цель;
- оборудование (материалы, программное обеспечение, оснащение, раздаточный материал и др.);
- методические указания (изучить краткий теоретический материал по теме практического занятия);
- ход выполнения;
- форму отчета.

В результате выполнения полного объема заданий практических занятий студент должен **уметь:**

- проектировать реляционную базу данных;
- использовать язык запросов для программного извлечения сведений из баз данных;

При проведении практических работ применяются следующие технологии и методы обучения: информационные технологии, ментальные карты и т.д.

Оценка выполнения практических занятий

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

В соответствии с учебным планом и рабочей программы дисциплины **«Основы проектирования баз данных»** на практические (лабораторные) занятия отводится **30 часов.**

2. ПЕРЕЧЕНЬ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

№ п/п	Тема практических занятий	Количество часов
1.	Операции с отношениями (реляционная алгебра).	2
2.	Нормализация реляционной БД, освоение принципов проектирования БД.	2
3.	Построение модели «сущность-связь.	2
4.	Преобразование реляционной БД в сущности и связи.	2
5.	Проектирование реляционной БД. Нормализация таблиц.	2
6.	Создание основных объектов БД. Задание ключей.	2
7.	Задание значений и ограничений поля.	2
8.	Задание значений и ограничений поля.	2
9.	Работа с записями базы данных. Импорт данных в таблицы.	2
10.	Создание ключевых полей. Задание индексов. Установление и удаление связей между таблицами.	2
11.	Проведение сортировки и фильтрации данных.	2
12.	Поиск данных по одному и нескольким полям. Поиск данных в таблице.	2
13.	Задание значений и ограничений поля. Проверка введенного в поле значения. Отображение данных числового типа и типа дата.	2
14.	Соединения таблиц и подзапросы. Ограничения и представления.	2
15.	Обработка транзакций. Использование функций защиты для БД.	2

3. СОДЕРЖАНИЕ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

Практическое занятие № 1

Тема: Операции с отношениями (реляционная алгебра).

Цель: закрепить теоретические знания по основным операциям реляционной алгебры.

Оборудование: тетрадь, ручка, ПК

Методические указания: выполните задания, ответьте на контрольные вопросы

Ход выполнения:

Задание 1. Произвести операции РА над исходными отношениями:

1.1.Показать результат операции Выборки : σВозраст ≥ 34(Персоны)

1.2.Показать результат операции Выборки: σВозраст = Вес(Персоны)

1.3.Показать результат операции Проекции: πВозраст,Вес(Персоны)

1.4.Показать результат операции Объединения: R(Персоны Union Персонажи)

1.5.Показать результат операции Пересечение : R (Персоны INTERSECT Персонажи1) 1.6.Показать результат операции Разность: R (Персоны MINUS Персонажи1)

1.7.Показать результат операции Произведение: R1 TIMES R2 13

Исходные данные:

Заданы три отношения: Персоны, Персонажи и Персонажи1

Персоны

Имя	Возраст	Вес
Harry	34	80
Sally	28	64
George	29	70
Helena	54	54
Peter	34	80

Персонажи

Имя	Возраст	Вес
Daffy	24	19
Donald	25	23
Scrooge	81	27

Персонажи1 (для п.5,6)

Имя	Возраст	Вес
Daffy	24	19
George	29	70
Donald	25	23
Scrooge	81	27
Sally	28	64

Отношение R1

П#	Имя	Город
S1	Сергей	Москва
S2	Николай	Москва

Отношение R2

P#	название	тип
P1	Гайка	Каленый
P2	Угол	Твердый

Задание 2. Ответить на вопросы:

2.1. В чем заключается замкнутость РА?

2.2. Что такое унарное и бинарное операции.

2.3. Закончить предложение: «При выполнении бинарной операции участвующие в операциях отношения …….»

2.4. Нарисовать схему классификации операций РА Кодда.

2.5. Кто такой Кодд?

Контрольные вопросы:

1. Понятие РА.
2. Основные операции РА над отношениями.

Содержание отчета:

1. Тему и цель практической работы.
2. Исходные данные к практической работе.
3. Полученные таблицы и операции, выполненные над отношениями (см.1).
4. Вывод: ответить на вопросы, указанные в п.2

Форма отчета: тетрадь, защита работы.

Практическое занятие № 2

Тема: Нормализация реляционной БД, освоение принципов проектирования БД.

Цель: получить навыки по приведению заданных таблиц к третьей нормальной форме.

Оборудование: тетрадь, ручка, ПК

Методические указания: выполните задания

Ход выполнения:

Задание 1. Привести исходную таблицу к 1-3NF с краткой характеристикой действий по нормализации исходной таблицы.

Исходные данные:

Задана исходная не нормализованная (то есть не являющаяся правильным представлением некоторого отношения) таблица.

ФИО	Данные
Иванов Иван Иванович	ПП-119 АСУ Муж. 19.01.1990
Петров Петр Петрович	Э-119 Электронщики Муж. 1991
Васильева Катерина Ильинична	Прикладная Информатика 1990 Жен ПК-129

Задание 2. Ответить на вопросы:

3.1. Определение и назначение нормализации данных.

3.2. Определение нормальной формы.

Контрольные вопросы:

1.Нормализация отношений.

2.Краткая характеристика всех нормальных форм отношений.

Форма отчета: документ, защита работы.

Практическое занятие № 3

Тема: Построение модели «сущность-связь».

Цель: освоение технологии построения информационной модели логического и физического уровней в нотации IDEF1X с использованием пакета Microsoft Office Visio.

Оборудование: тетрадь, ручка, ПК

Методические указания: выполните задания

Ход выполнения:

Задание 1. Рассмотреть теоретический материал.

Методология IDEF1X – язык для семантического моделирования данных, основанный на концепции «сущность-связь».

Различают два уровня информационной модели: логический и физический.

Логическая модель позволяет понять суть проектируемой системы, отражая логические взаимосвязи между сущностями.

Различают три подуровня логического уровня модели данных,

отличающиеся по глубине представления информации о данных:

- диаграмма сущность-связь (Entity-Relationship Diagram (ERD));
- модель данных, основанная на ключах (Key Based Model (KB));
- полная атрибутивная модель (Fully Attributed Model (FA)).

Физическая модель отражает физические свойства проектируемой базы данных (типы данных, размер полей, индексы). Параметры физической информационной модели зависят от выбранной

системы управления базами данных (СУБД).

Сущности и атрибуты

Сущность – это множество реальных или абстрактных объектов (людей, предметов, документов и т.п.), обладающих общими

атрибутами или характеристиками. Любой объект системы может

быть представлен только одной сущностью, которая должна быть

уникально идентифицирована. Именование сущности осуществляется

с помощью существительного в единственном числе. При этом имя сущности должно отражать тип или класс объекта, а не его конкретный экземпляр (например, Студент, а не Петров) (рис. 1).

Студенты

Рис. 1. Графическое представление сущности «Студент» в MS Office Visio

Любая сущность характеризуется набором атрибутов (свойств).

Атрибут сущности – характеристика сущности, то есть свойство реального объекта.

Например, на рис. 1 атрибутами сущности «Студент» являются «ID студента», «Фамилия», «Имя», «Отчество», «Дата поступления» и «Номер билета».

В свою очередь, атрибуты сущности делятся на 2 вида: собственные и наследуемые. Собственные атрибуты являются уникальными в рамках модели. Наследуемые атрибуты передаются от сущности-родителя при установке связи с другими сущностями.

Первичный ключ (Primary Key, PK). Каждая сущность должна обладать атрибутом или комбинацией атрибутов, чьи значения

однозначно определяют каждый экземпляр сущности. Эти атрибуты

образуют первичный ключ сущности.

Внешний ключ (Foreign Key, FK). Если между двумя сущностями имеется специфическое отношение связи или категоризации, то атрибуты, входящие в первичный ключ родительской или общей сущности, наследуются в качестве атрибутов сущностью потомком или категориальной сущностью соответственно. Эти атрибуты и называются внешними ключами.

Отношения в IDEF1X-модели.

При построении информационной модели различают следующие типы отношений между сущностями: идентифицирующее, не идентифицирующее, не специфическое (многие-ко-многим) и отношения категоризации.

Мощность отношения служит для обозначения отношения числа экземпляров родительской сущности к числу экземпляров дочерней.

Нормализация данных.

Нормализация – это процесс проверки и реорганизации сущностей и атрибутов с целью удовлетворения требований к реляционной модели данных. Процесс нормализации сводится к последовательному приведению структур данных к нормальным формам – формализованным требованиям к организации данных.

Первая нормальная форма (1НФ). Сущность находится в первой нормальной форме тогда и только тогда, когда все атрибуты содержат атомарные значения. Среди атрибутов не должно встречаться повторяющихся групп, т.е. несколько значений для каждого экземпляра.

Вторая нормальная форма (2НФ). Сущность находится во второй нормальной форме, если она находится в первой нормальной форме, и каждый не ключевой атрибут полностью зависит от первичного ключа (не может быть зависимости от части ключа).

Третья нормальная форма (3 НФ). Сущность находится в третьей нормальной форме, если она находится во второй нормальной форме и никакой не ключевой атрибут не зависит от другого не ключевого атрибута (не должно быть зависимости между не ключевыми атрибутами).

Задание 2. Построение логической информационной модели уровня «сущность-связь»

Составление пула – списка потенциальных сущностей Информационная модель может быть построена на основе функциональной модели (в нотации IDEF0). В этом случае названия всех интерфейсных дуг IDEF0-модели заносятся в пул – список потенциальных сущностей.

Список потенциальных сущностей для рассматриваемого примера будет представлен таблицей вида (рис. 2):

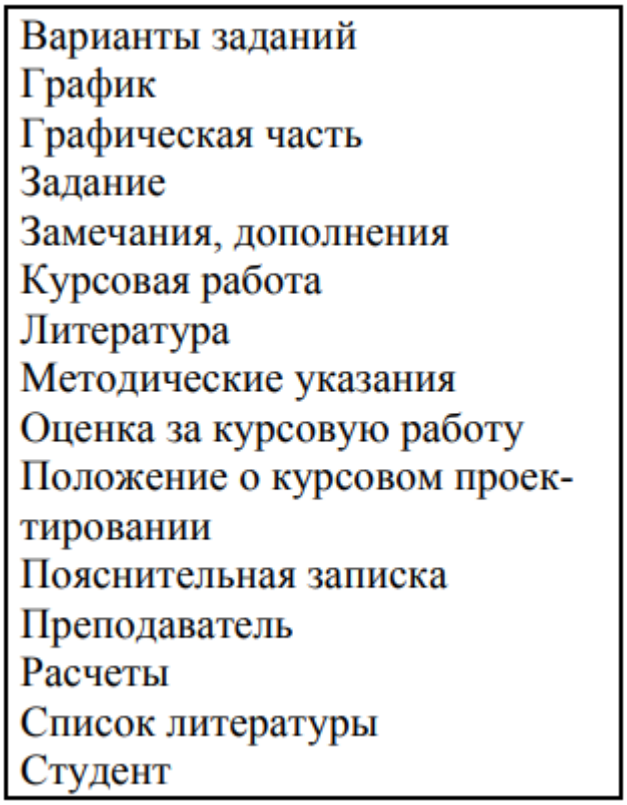


Рис. 2. Пул – список потенциальных сущностей

Теперь из этого списка необходимо выделить сущности, остальные интерфейсные дуги будут преобразованы в атрибуты сущностей.

В качестве сущностей выделим следующие:

1) задание;
2) пояснительная записка;
3) курсовая работа;
4) положение о курсовом проектировании;
5) студент;
6) преподаватель;
7) график;
8) методические указания.

Задание 2. Создание логической модели «сущность-связь».

1. Запустите MS Office Visio.

2. На закладке выбора шаблона выберите категорию Программное обеспечение и базы данных и в ней элемент Схема модели базы данных. Нажмите кнопку Создать в правой части экрана.

3. Установите необходимые параметры страницы (масштаб, ориентация страницы).

4. MS Office Visio поддерживает различные нотации моделей баз данных. Для того чтобы задать нотацию IDEF1X, необходимо выбрать пункты меню База данных → Параметры → Документ. В открывшемся окне на вкладке Общие установить переключатель в меню Набор символов на IDEF1X. Меню Имена, видимые на схеме позволяет указать, какие имена атрибутов сущности будут отображены на диаграмме

(концептуальные, физические или оба варианта одновременно). В данном случае для логического представления информационной модели необходимо выбрать пункт Концептуальные имена (рис. 3).

В закладке Отношение окна Параметры документа базы данных в меню Показывать нужно отметить галочкой пункт Мощность, в меню Отображение вида выбрать пункт Показывать вербальную фразу, снять галочку в пункте Обратный текст (рис. 4). Данные настройки позволят отобразить имя и мощность связи в модели.

5. Для того чтобы создать сущность, необходимо перетащить  элемент на рабочее поле. Переход в режим редактирования сущности осуществляется двойным щелчком по сущности или по нажатию правой кнопки мыши и выбор пункта меню Свойства базы данных.

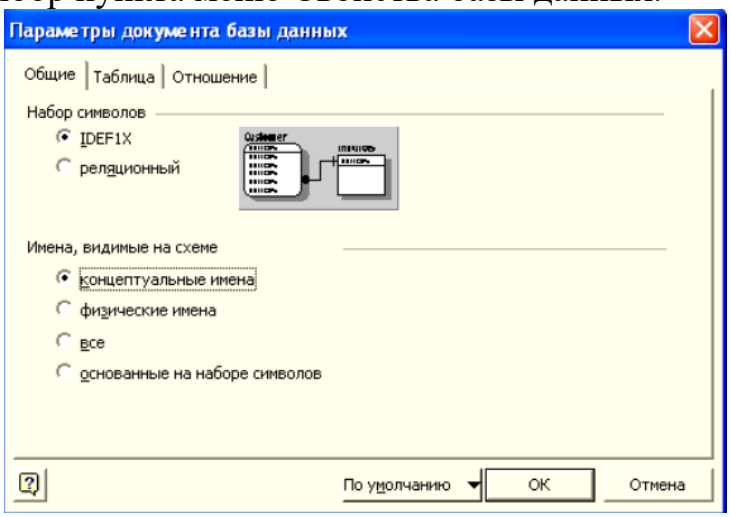


Рисунок 3 – Настройка параметров модели

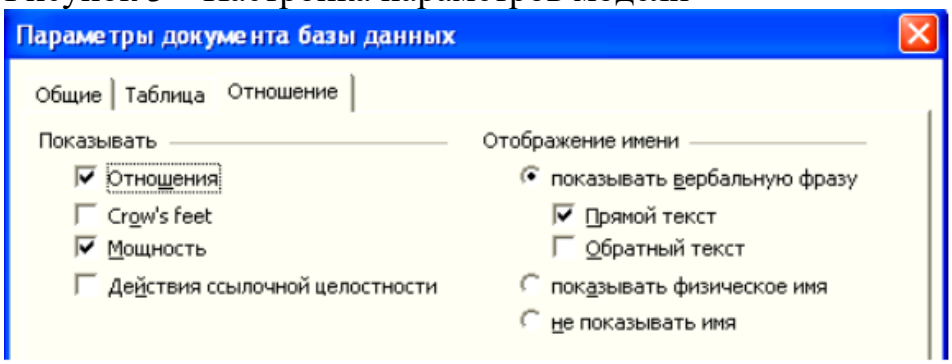


Рисунок 4 – Настройка вида отношений информационной модели

Чтобы задать имя сущности, в окне Свойства базы данных нужно выбрать категорию Определение, снять галочку в пункте Синхронизация имен при вводе (в противном случае, физическое и логическое имя сущности будут совпадать, что по практическим соображениям не всегда удобно) и задать концептуальное имя сущности. Руководствуясь данным алгоритмом, создадим 8 сущностей, определенных в задание 2 (см. рис. 5).

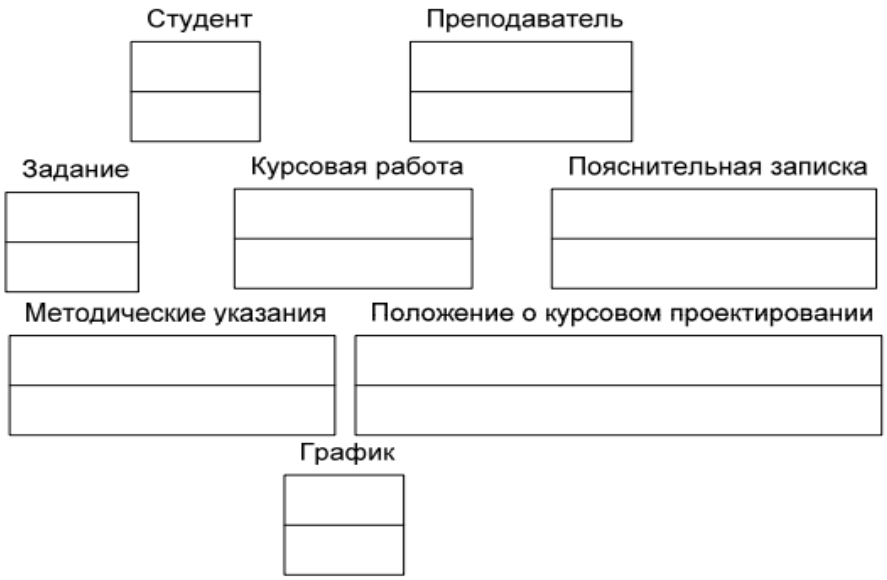


Рисунок 5. – Сущности информационной модели логического уровня

6. Далее необходимо установить связи между сущностями.

Сначала составим описание предметной области на естественном языке.

Любой студент должен выполнить одну или несколько курсовых работ.

Каждая курсовая работа должна выполняться одним студентом (в идеале).

Каждая курсовая работа выполняется в соответствии с методическими указаниями и положением о курсовом проектировании.

Курсовая работа сдается по графику.

Курсовая работа оформляется в виде пояснительной записки.

Преподаватель проводит консультации, проверяет и ставит оценку за курсовую работу.

Таким образом, сформулируем имена связей:

СТУДЕНТ выполняет КУРСОВУЮ РАБОТУ.

ПРЕПОДАВАТЕЛЬ проверяет КУРСОВУЮ РАБОТУ.

КУРСОВАЯ РАБОТА выполняется в соответствии с ЗАДАНИЕМ.

КУРСОВАЯ РАБОТА оформляется в виде ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ определяют требования к КУРСОВОЙ РАБОТЕ.

КУРСОВАЯ РАБОТА организуется согласно ПОЛОЖЕНИЮ ПО КУРСОВОМУ ПРОЕКТИРОВАНИЮ.

КУРСОВАЯ РАБОТА сдается по ГРАФИКУ.

Во всех случаях сущность Курсовая работа является дочерней, за исключением связи с сущностью Пояснительная записка. Определим типы связей и построим модель (см. рис. 8). В дальнейшем можно будет подкорректировать связи между сущностями.

Чтобы установить связи между сущностями, необходимо перетащить на рабочую область элемент  , поднести один конец стрелки к родительской сущности, другой – к дочерней.

Примечание. При правильном связывании каждая сущность будет подсвечена красным цветом.

В MS Office Visio по умолчанию используется не идентифицирующее отношение. Чтобы изменить тип связи, необходимо двойным щелчком по связи открыть окно Свойства базы данных и в категории в категории Прочее указать тип отношения (идентифицирующее, не идентифицирующее). В этой же категории указывается мощность связи (см. рис. 6).

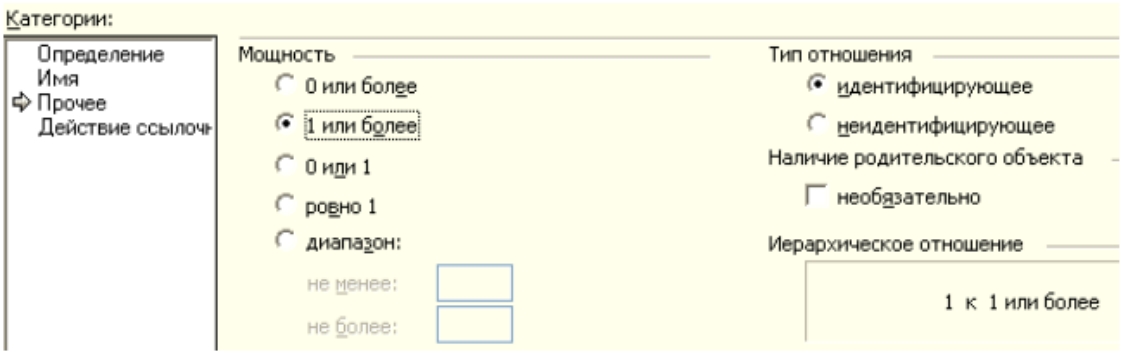


Рисунок 6. – Определение типа связи и мощности

Примечание. Кроме того, при не идентифицирующем отношении нужно указать, является ли наличие родительской сущности обязательным (т.е. может ли существовать экземпляр дочерней сущности, если не существует экземпляра родительской). Если наличие родительского объекта является необязательным, графически это отобразится в виде не закрашенного ромба со стороны родительской сущности.

Следующий шаг – в категории Имя в поле Вербальная фраза нужно указать имя отношения (рис. 7). Также можно указать имя

связи в поле Обратная фраза для спецификации отношения потомо к родителю (в нашем случае обратная фраза отображаться не будет).

Примечание. Все изменения при закрытии окна свойств сохраняются автоматически.

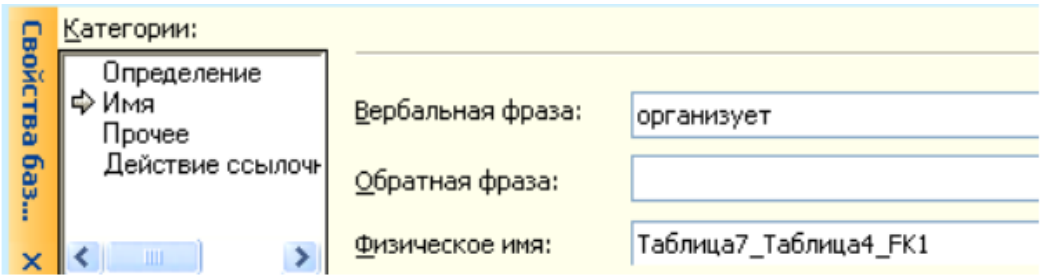


Рисунок 7 – Определение имени отношения

После определения имен, типов связей и задания мощностей получим информационную модель, представленную на рис. 8.

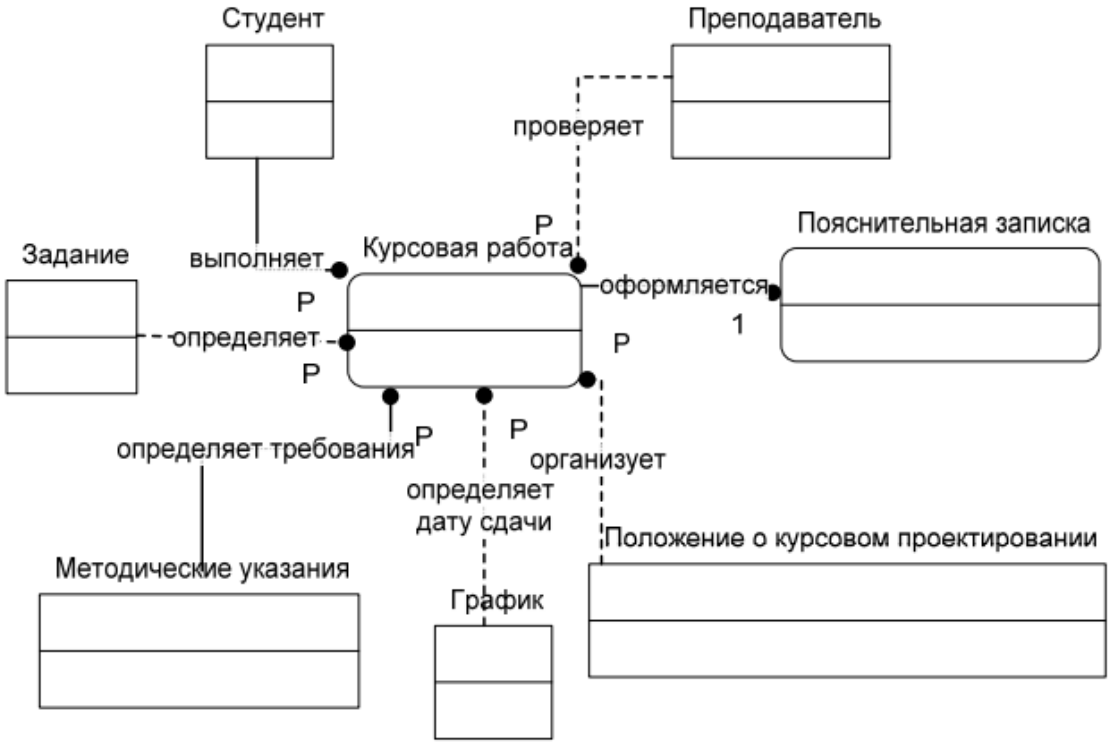


Рисунок 8. – Информационная модель уровня «сущность-связь»

Задание 3. Разработка логической модели данных, основанной на ключах

1. Необходимо определить ключевые атрибуты для каждой сущности, обращая внимание на то, что дочерние сущности наследуют ключевые атрибуты от родительских (см. рис. 10).

Для этого двойным щелчком мыши по сущности откроем окно редактирования ее свойств, перейдем в категорию Столбцы, по нажатию кнопки Добавить введем имя поля (например, для сущности Задание ключевым атрибутом будет являться Вариант задания). Чтобы сделать атрибут ключевым, необходимо отметить галочкой пункт РК (рис. 9). Данное поле становится обязательным автоматически.

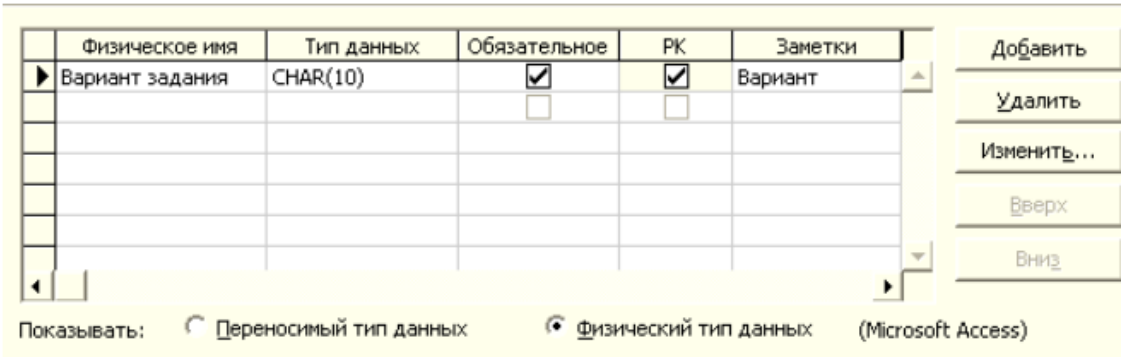


Рисунок 9. – Определение ключевого атрибута

Аналогичным образом зададим ключевые атрибуты для всех сущностей информационной модели. Результат представлен на рис.10.

Как видно из рис. 10 по сравнению с информационной моделью уровня «сущность-связь», был изменен тип связи между сущностями Методические указания и Курсовая работа, поскольку ключевые атрибуты сущности

Методические указания для сущности Курсовая работа будут являться избыточными (зная номер зачетной книжки, можно узнать специальность и курс, на котором учится студент).

2. Кроме того, отметим, что три сущности (Задание, График, Методические указания) содержат одинаковые атрибуты Дисциплина.

Это является некорректным. Чтобы устранить данную ошибку, выделим одноименную сущность и свяжем ее идентифицирующими связями с вышеуказанными сущностями (рис. 11, лабораторная работа № 4).

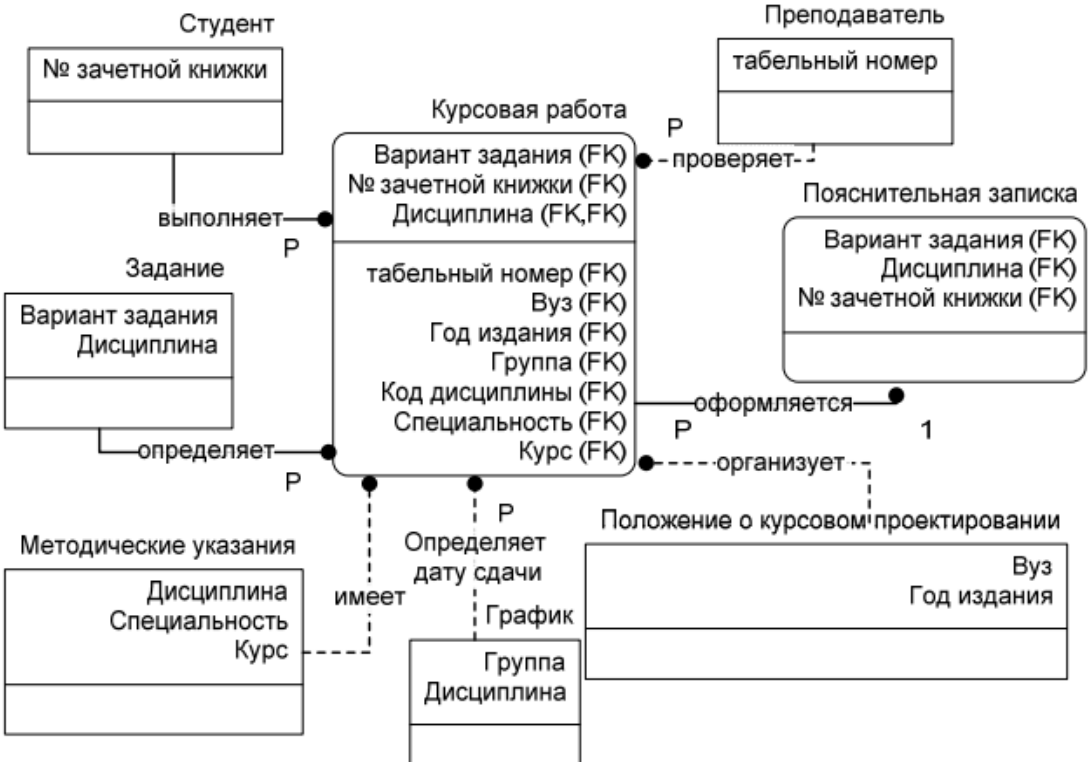


Рисунок 10 – Информационная модель с ключевыми атрибутами.

Форма отчета: отчет, защита работы.

Практическое занятие № 4

Тема: Преобразование реляционной БД в сущности и связи

Цель: изучить Преобразование реляционной БД, в сущности, и связи.

Оборудование: тетрадь, ручка, ПК

Методические указания: выполните задания

Ход выполнения:

Задание 1. Создание полной атрибутивной модели

Для того чтобы получить полную атрибутивную модель, необходимо дополнить сущности не ключевыми атрибутами. Дополненная модель представлена на рис. 12.

Примечание. Если атрибут не является обязательным, нужно убедиться, что в окне Свойства базы данных в категории Столбцы в пункте Обязательное не стоит галочка. Не обязательные к заполнению атрибуты справа от имени имеют пометку (О).

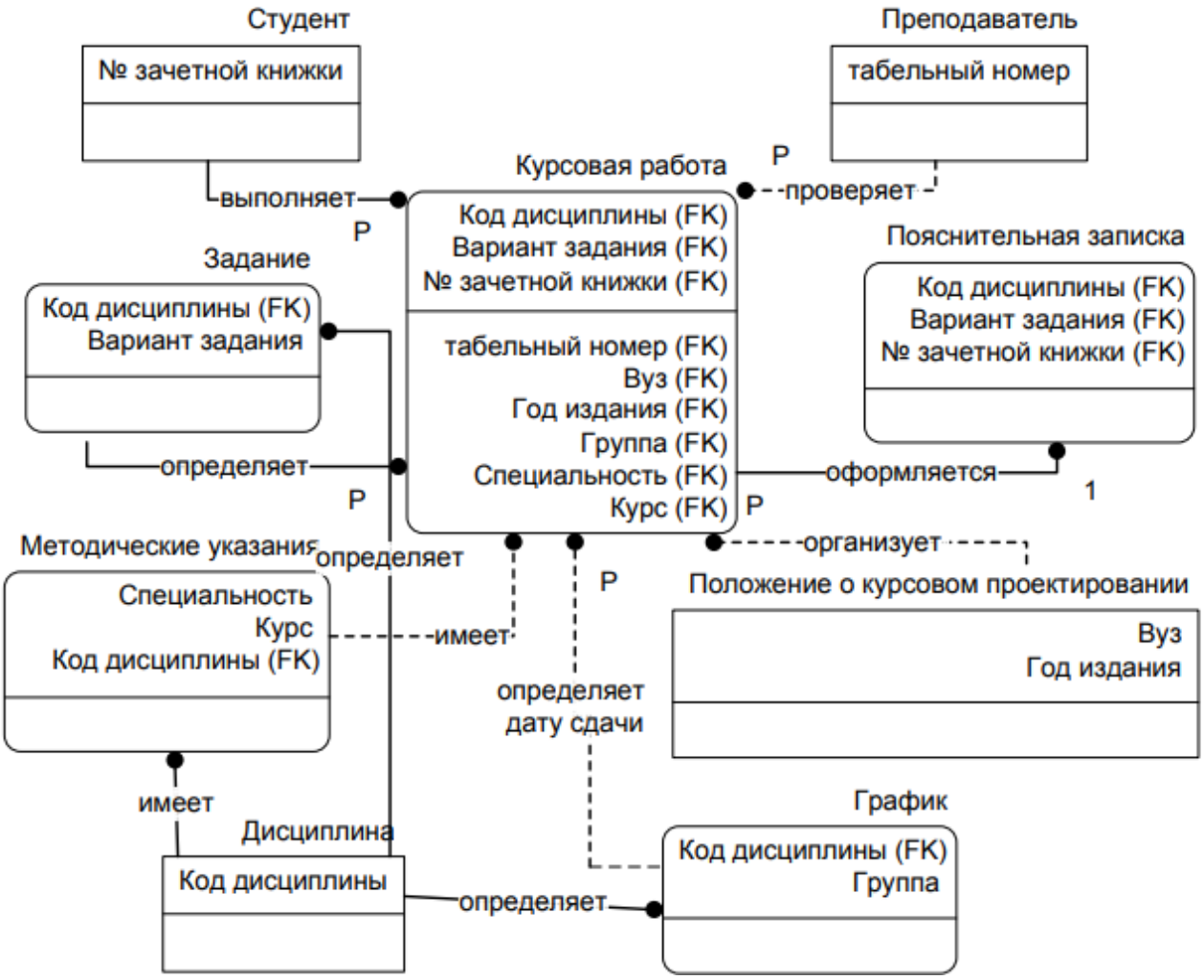


Рисунок 11. – Скорректированная информационная модель, основанная на ключах.

Задание 2. Нормализация полной атрибутивной модели

1. Проверим, все ли атрибуты имеют атомарные значения, т.е. среди атрибутов не должно встречаться повторяющихся групп, нескольких значений для каждого экземпляра (например, номер телефона_1, номер телефона_2). Видим, что атрибут Авторы, в сущности, Методические указания не удовлетворяет требованиям 1 НФ (у методических указаний может быть несколько авторов). Необходимо выделить сущность, которая будет содержать сведения об авторах методических указаний. Поскольку авторами всегда являются преподаватели вузов, новую сущность выделять не имеет смысла, свяжем сущности Методические указания и Преподаватель, предварительно удалив атрибут Авторы. Остальные атрибуты соответствуют 1 НФ.

Атрибутивная модель, приведенная к 1 НФ, представлена на рис. 13.

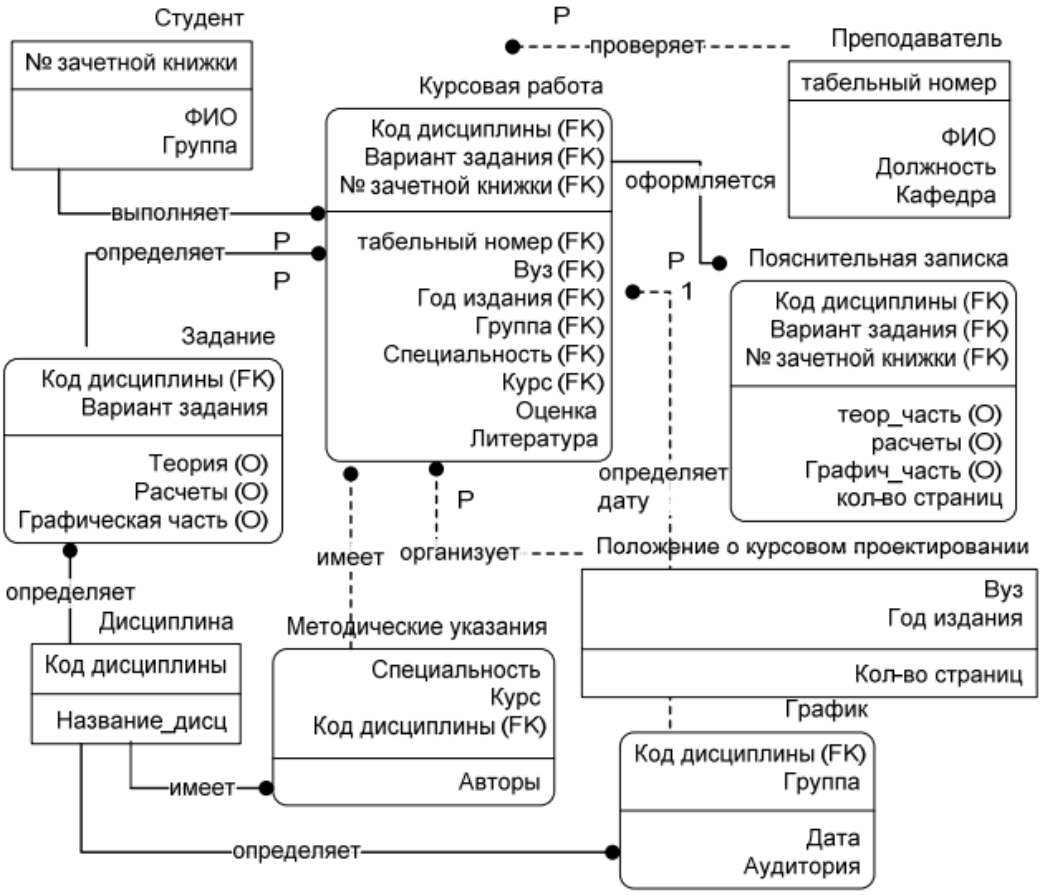


Рисунок 12 – Полная атрибутивная модель

2. Приведем модель ко 2 НФ. Проверим, все ли атрибуты зависят от составного ключа, а не от его части. Проверка показала, что все не ключевые атрибуты сущностей полностью зависят от составного ключа. Значит, модель удовлетворяет требованиям 2 НФ.

3. Проверим, есть ли транзитивная зависимость между не ключевыми атрибутами. Проверка показала, что взаимозависимости между не ключевыми атрибутами нет. Таким образом, модель, представленная на рисунке 5.12, приведена к 3 НФ.

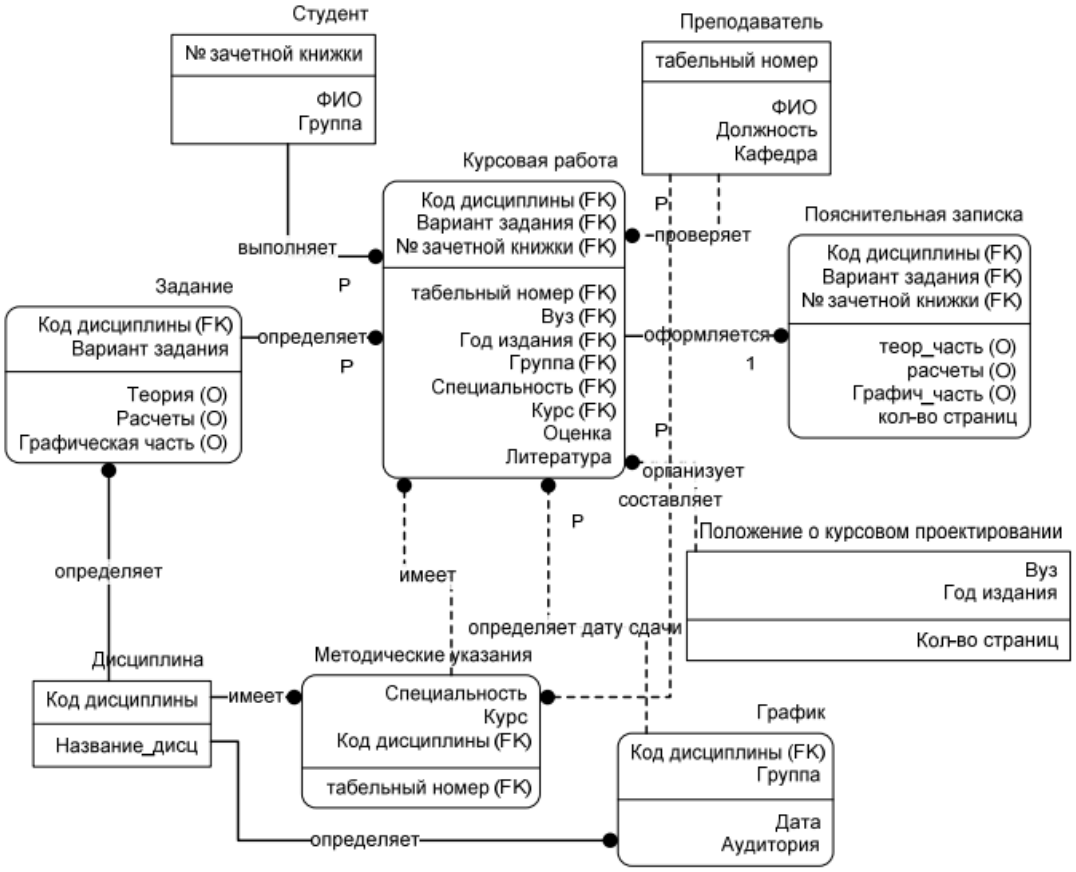


Рисунок 13 – Информационная модель, приведенная к 1 НФ.

Форма отчета: отчет, защита работы.

Практическое занятие № 5

Тема: Проектирование реляционной БД. Нормализация таблиц

Цель: изучить проектирование реляционной БД и нормализацию таблиц.

Оборудование: тетрадь, ручка, ПК

Методические указания: выполните задания

Ход выполнения:

Задание 1. Создание физической модели

1. Необходимо переключиться на физический уровень представления информационной модели. Для этого нужно выбрать пункты меню База данных → Параметры → Документ. В открывшемся окне на вкладке Общие установить переключатель в меню Имена, видимые на схеме. В данном случае для физического представления информационной модели необходимо выбрать пункт Физические имена (рис. 14).

2. В закладке Таблица окна Параметры документа базы данных в меню Отображать выбрать пункт Вертикальные линии, в меню Типы данных – Показывать физические и в меню Порядок – Физический порядок (рис. 15).

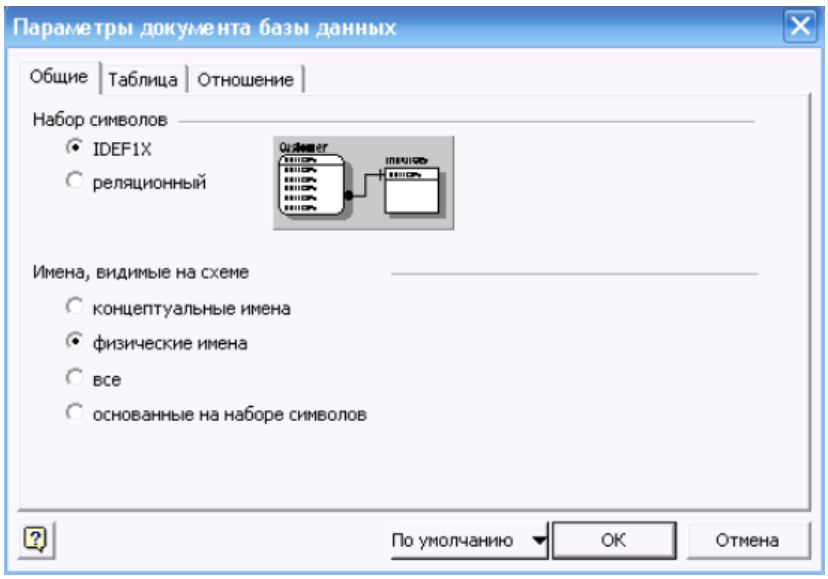


Рисунок 14 – Настройка параметров модели

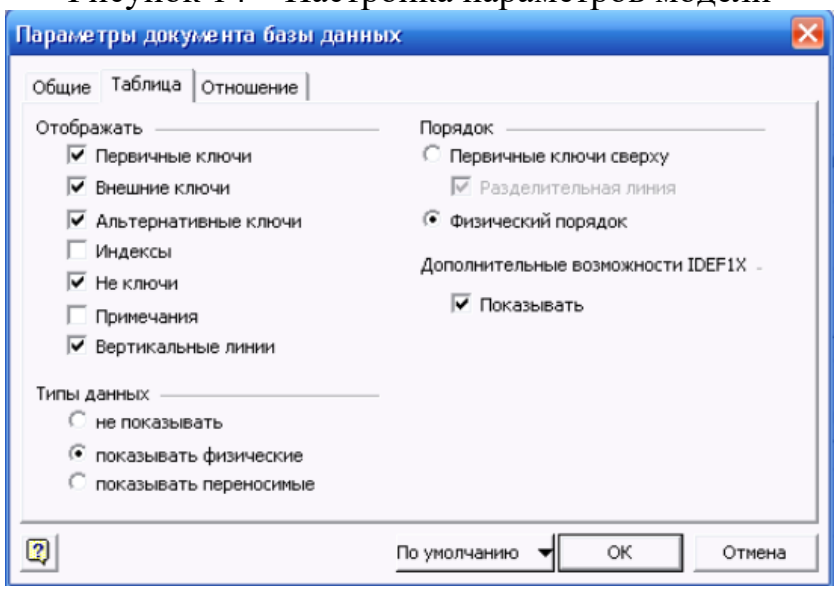


Рисунок 15 – Настройка параметров отображения сущности

3. В закладке Отношение окна Параметры документа базы данных в меню Отображение вида выбрать пункт Показывать физическое имя (рис. 16).

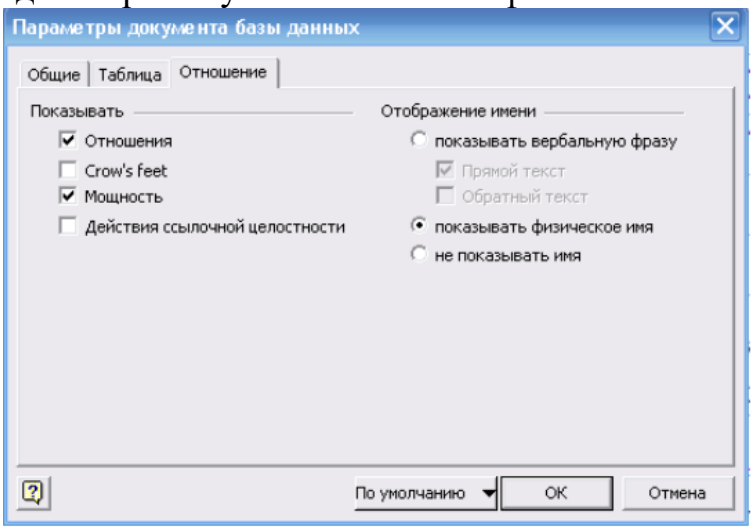


Рисунок 16 – Настройка вида отношений информационной модели

По окончании настройки документа информационная модель будет выглядеть, как представленная на рис. 17.

4. Для каждого атрибута (поля) необходимо определить тип данных.

Примечание (Выбор между переносимыми и физическими типами данных).

Переносимые типы данных — это обобщенные типы данных, соответствующие в разных системах баз данных простым, совместимым между собой физическим типам.

Физические типы данных — это типы данных, поддерживаемые целевой базой данных.

Щелкните сущность, содержащую атрибуты, для которых требуется установить типы данных.

В окне Свойства базы данных в списке Категории выберите вариант Столбцы.

Под списком столбцов установите переключатель в положение Физический тип данных.

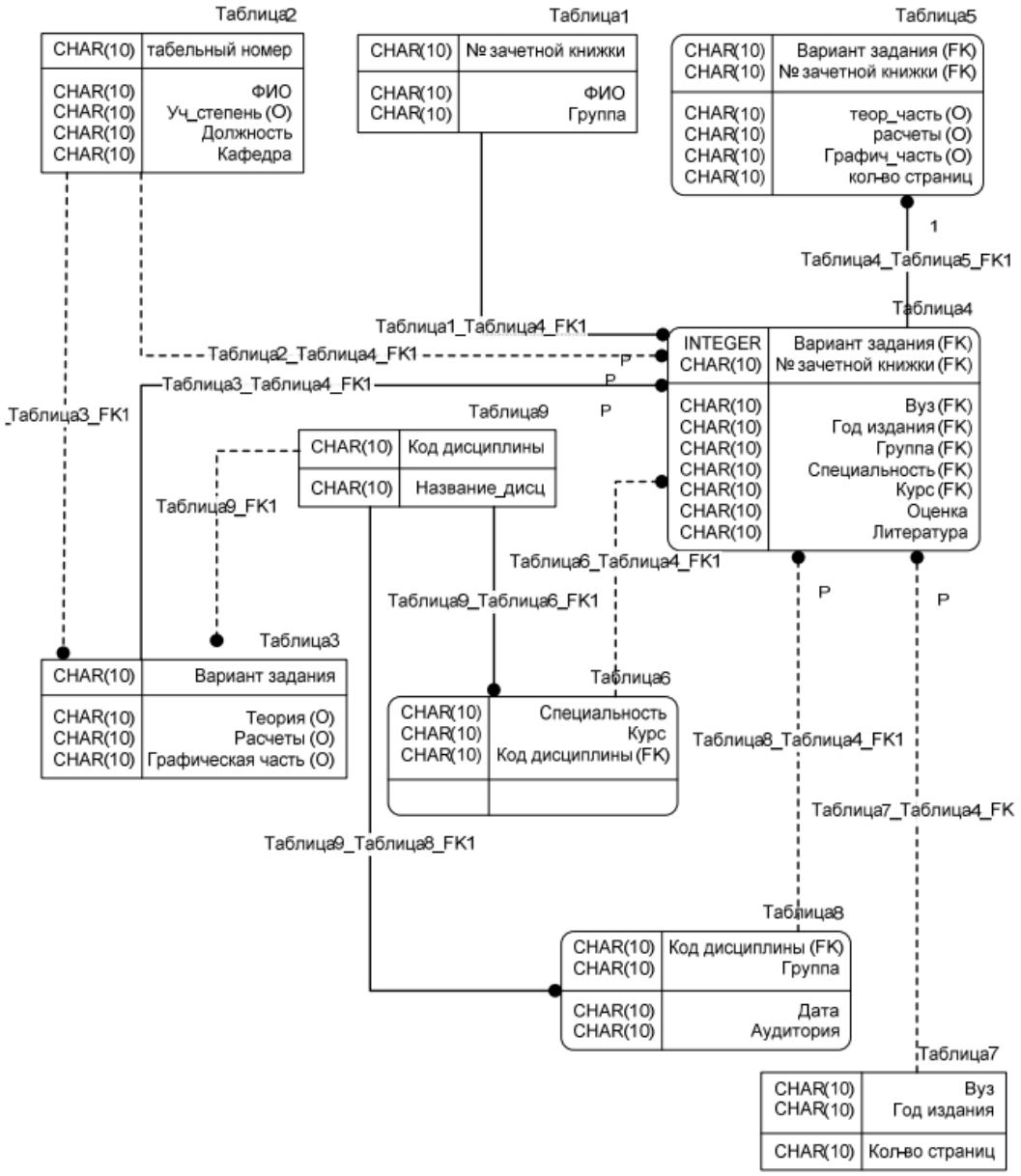


Рисунок 17 – Вид физической модели

В группе Тип данных для каждого атрибута выберите необходимый вариант из множества альтернатив (рис. 18). Описание типов данных приведено в Приложении Б.

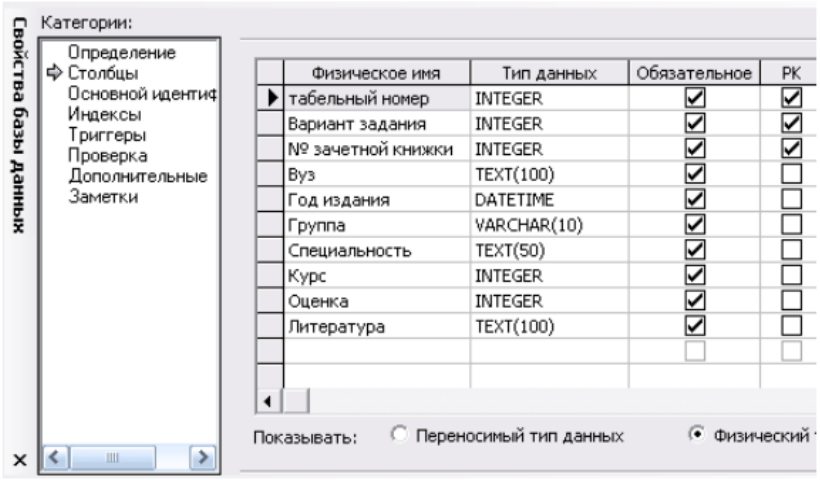


Рисунок 5.17 – Определение типа данных атрибутов сущности

После того, как будут выполнены все действия, физическая модель будет выглядеть, как показано на рис. 19.

Таким образом, проделав все вышеперечисленные действия, получим информационную модель физического уровня, на основе которой может быть сгенерирована схема БД (в нашем случае в MS Office Access).

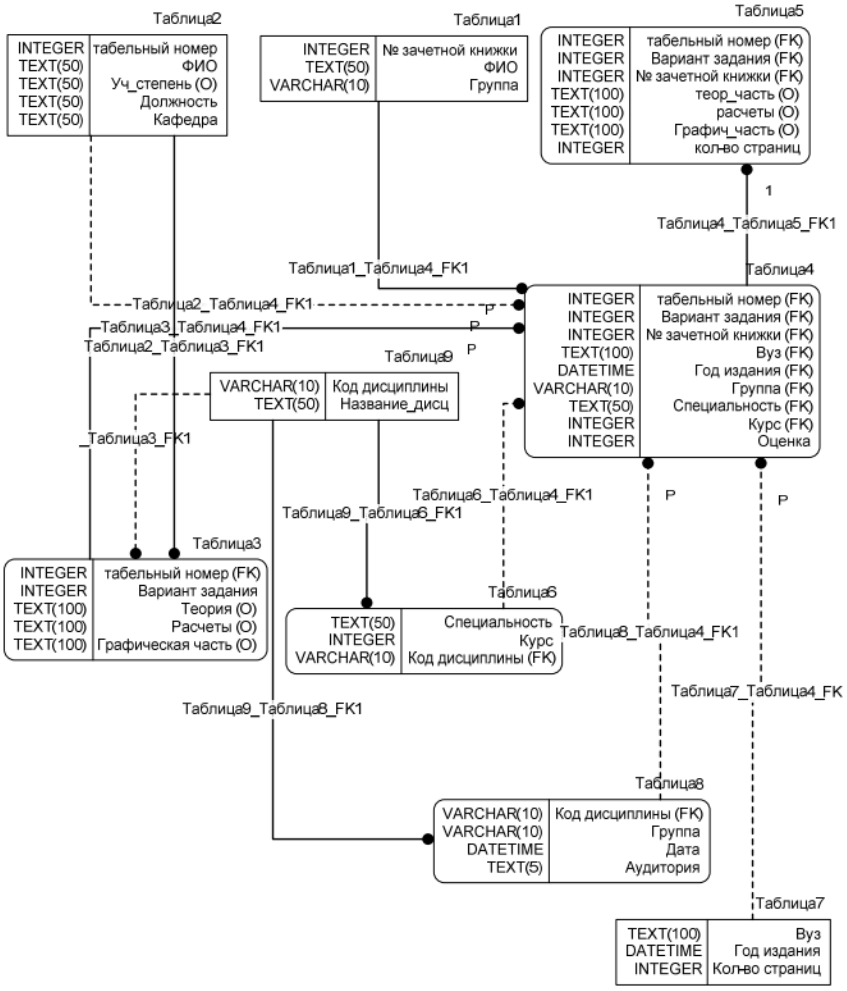


Рисунок 19 – Физическая модель базы данных

Требования к содержанию и оформлению отчета

Отчет по лабораторной работе должен быть оформлен в форме документа MS Office Word и содержать:

1) титульный лист;
2) название лабораторной работы, цель;
3) пул – список потенциальных сущностей;
3) нормализованную информационную модель логического уровня;
4) информационную модель физического уровня;
5) выводы по проделанной работе.

Титульный лист должен содержать следующие сведения: название и порядковый номер лабораторной работы, ФИО студентов, группу, ФИО преподавателя и т.п. По усмотрению преподавателя отчет может быть представлен в бумажном или электронном виде.

Контрольные вопросы

1. Для чего предназначена диаграмма «сущность-связь»?

2. Чем отличается полная атрибутивная модель от диаграммы «сущность-связь»?
3. Какие виды отношений существуют и чем они отличаются?
4. Приведите пример идентифицирующего отношения.
5. Приведите пример отношения полной категоризации.
6. Чем отличаются отношения полной и неполной категоризации?
7. Что представляет собой нормализация?
8. В чем разница между логическим уровнем модели данных и физическим?

Форма отчета: отчет, защита работы.

Практическое занятие № 6

Тема: Создание основных объектов БД. Задание ключей.

Цель: 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.

Оборудование: тетрадь, ручка, ПК

Методические указания: выполните задания

Ход выполнения:

Задание 1. Создать базу данных на сервере.

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.№6» и стандартное расширение "*.sql".

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 диалог окне Соединение с сервером подтвердите заданные по умолчанию

параметры и нажмите кнопку Соединить. 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>.

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.

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.

Задание 2. 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. 1) и нажать кнопку [OK].

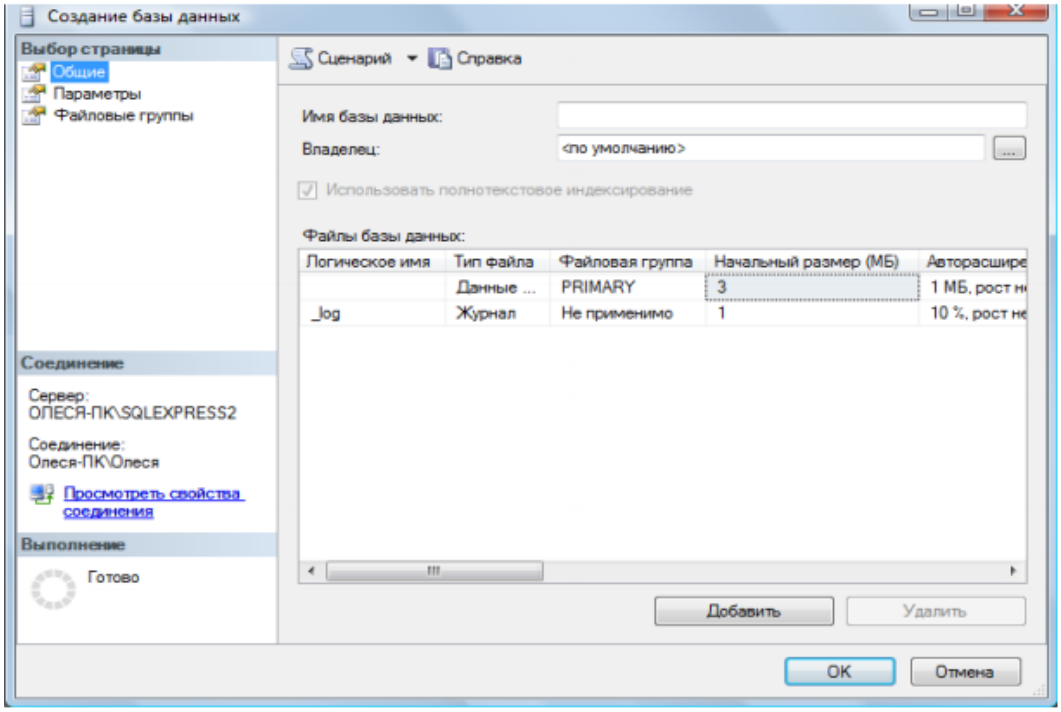


Рис. 1. Диалоговое окно создания базы данных

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. 2. (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.

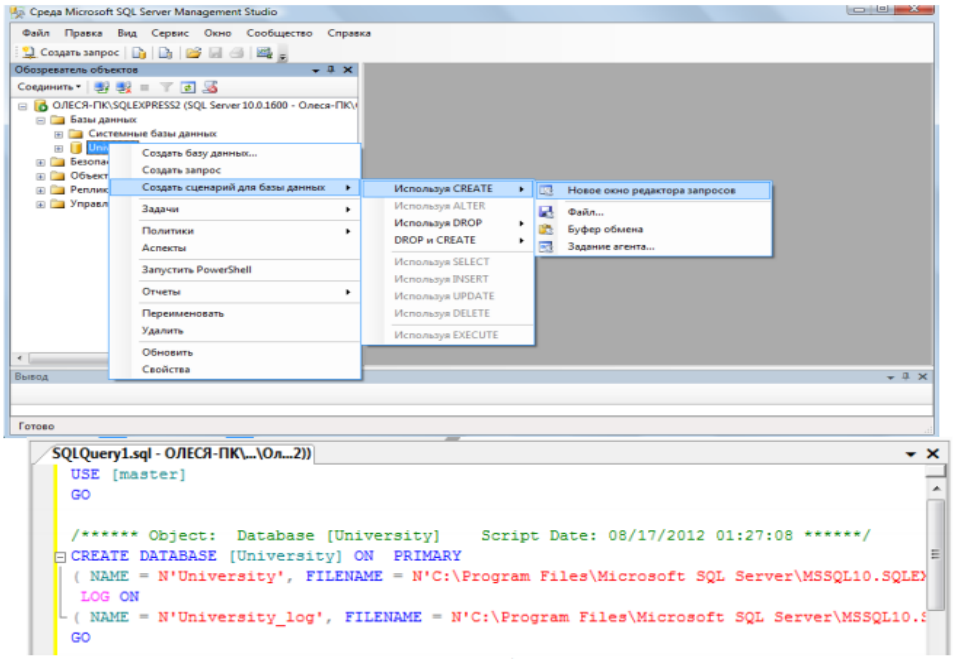


Рис. 2. Сгенерированный sql-код созданной базы данных

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. 3).

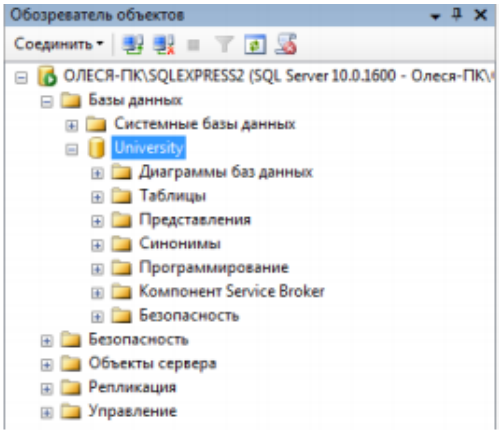


Рис. 3. Перечень зарегистрированных баз данных в SQL Server Management Studio

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. 1) 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 появится новая база данных.

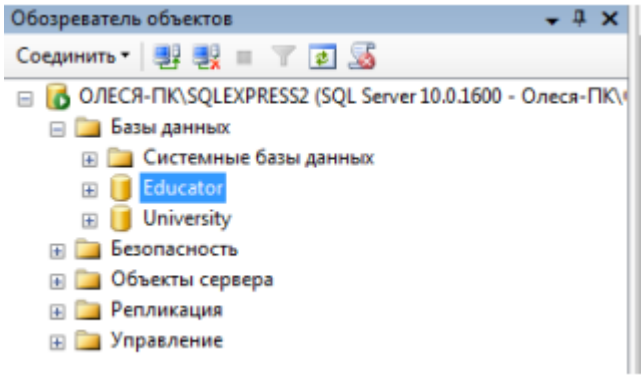


Рис. 5. Зарегистрированные базы данных в SQL Server Management Studio

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 базы 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.6.

После выбора пути и файла для резервной копии в окне Back Up Database нажатием на OK 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.

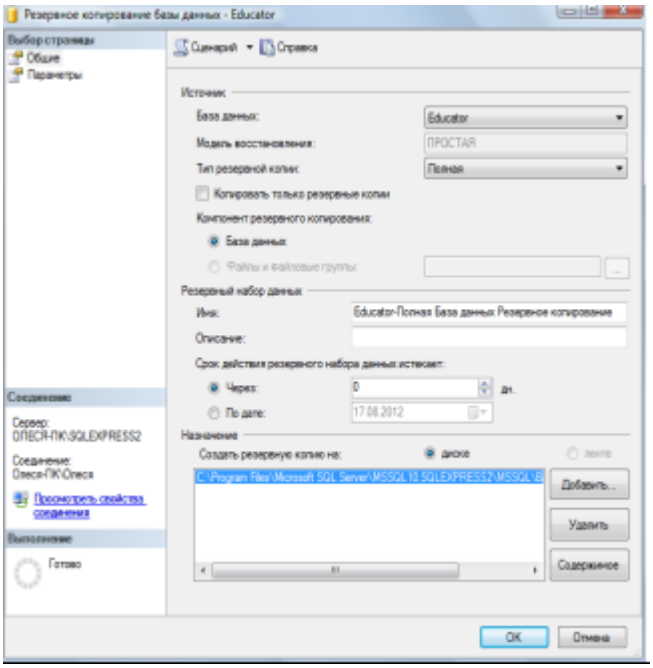


Рис.8. Создание резервной копии базы данных

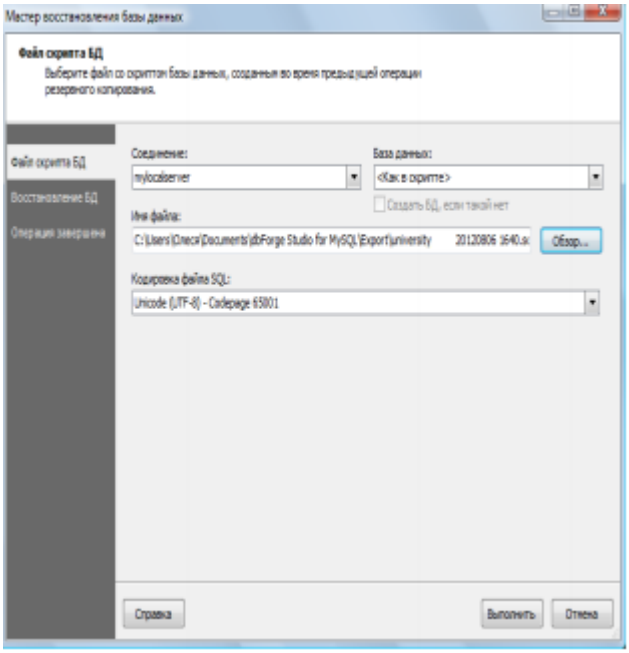


Рис.7. Восстановление базы данных

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.7. 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.

Форма отчета: отчет, защита работы.

Практическое занятие № 7

Тема: Задание значений и ограничений поля.

Цель: изучить способы создания, изменения и удаления таблиц. Получить навыки использования приложения " SQL Server Management Studio " для создания, удаления и изменения структуры таблиц. Изучить SQL-операторы для работы с таблицами и индексами. Изучить используемые в SQL Server типы ограничений. Получить навыки использования программы " SQL Server Management Studio " для создания, изменения и удаления ограничений. Изучить SQL-операторы для работы с ограничениями.

Оборудование: тетрадь, ручка, ПК
Методические указания: выполните задания
Ход выполнения:
Задание 1. Выполнить операций создания таблиц и индексов в диалоговом режиме.

Откройте среду SQL Server Management Studio, выполните соединение с сервером, откройте созданную базу данных University в лаб.раб. №6.

Для создания таблицы в диалоговом режиме, нажмите в окне "Обозревателя

объектов" правую клавишу мыши на узле "Tables" (Таблицы) или на одной из

имеющихся таблиц и в открывшемся меню выберите команду "New Table…(Создать

таблицу)" (рис. 1). В результате откроется окно создания таблицы (рис. 2).

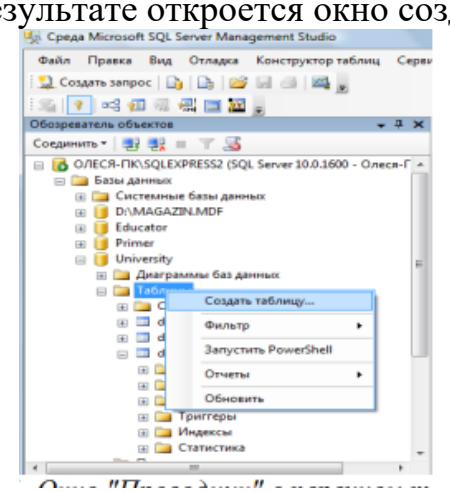


Рис. 1. Окно "Проводник" с перечнем таблиц

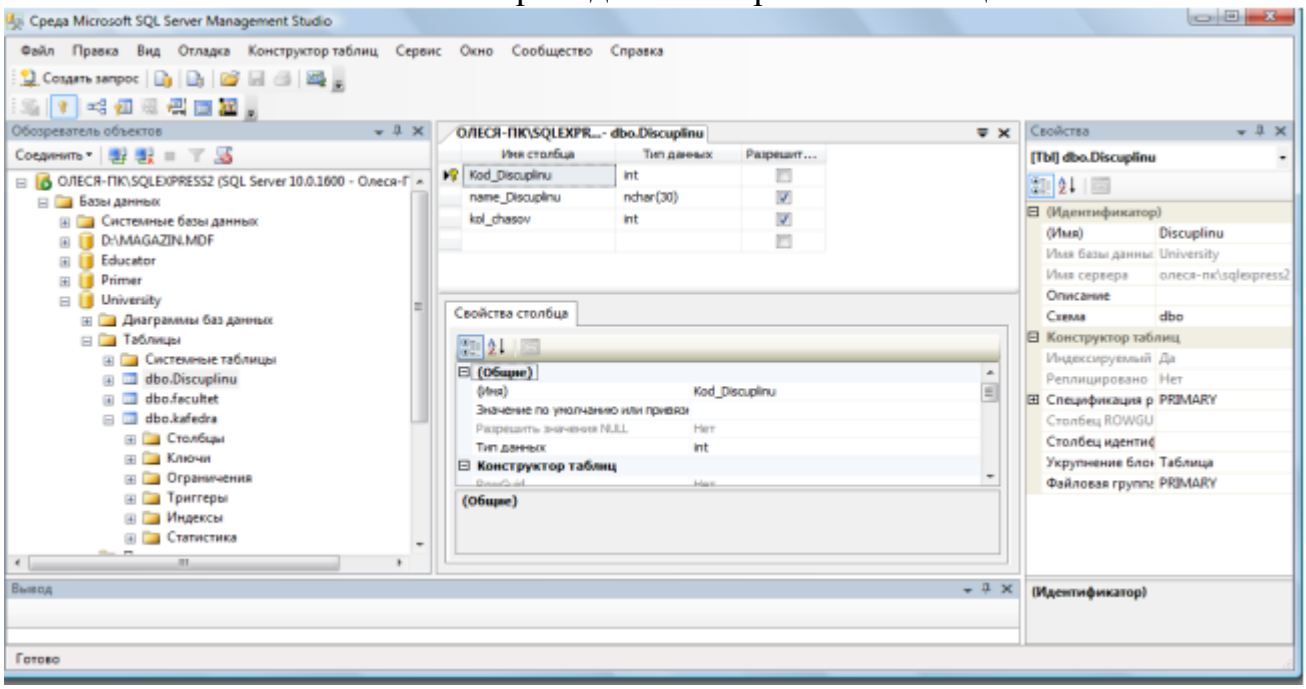


Рис. 2. Окно формирования таблицы в диалоговом режиме

Сетка в средней части окна содержит сведения о полях таблицы.

Чтобы добавить поле в таблицу, следует нажать правую клавишу и выбрать из контекстного меню [Вставить столбец].

В колонке "Имя столбца" вводится имя создаваемого поля, в колонке "Тип данных" выбирается тип данных. Чтобы задать полю ограничение "NOT NULL" достаточно установить флажок в колонке "Разрешить значения NULL (пустые значения)".

Чтобы присвоить полю статус первичного ключа необходимо в колонке ПК щелкнуть правой клавишей мыши и выбрать из контекстного меню Создать первичный ключ.

Дополнительные свойства можно настраивать с помощью окна Свойства столбца, которое находится внизу рабочей области.

Самостоятельно

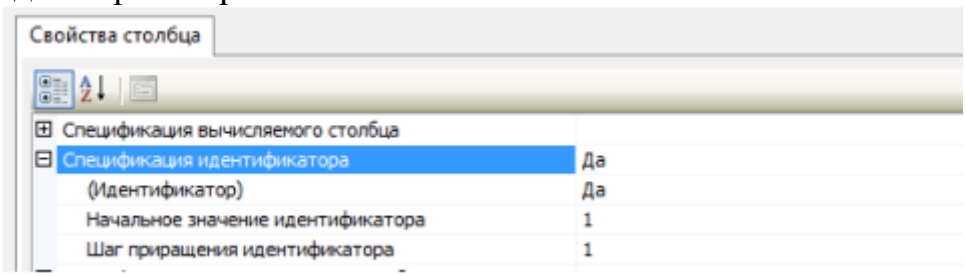
Создайте новую таблицу, хранящую информацию о всех специальностей в колледже.

Присвойте ей имя – Facultet.

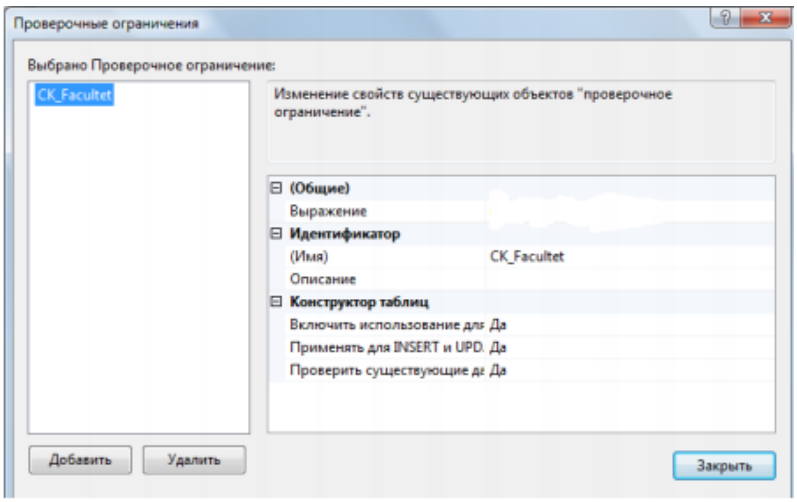
Добавьте поля Kod_faculteta (уникальный номер факультета, тип – целый int, первичный ключ, автоинкремент), Name_faculteta (название ЦК, тип текстовый - varchar, длина 255 символов, не допускается пустых значений), Fio_Decana (ФИО декана факультета, тип – текстовый varchar), Nomer_komnatu (номер комнаты деканата, тип – символьный varchar(допускается запись 134-2, где 134 – номер комнаты, а 2 – номер корпуса)), Tel_decanata (телефон деканата, тип – длинное целое число bigint с точностью в 10 символов, значение по умолчанию ‘999999’,ограничение на значение меньше ‘1 000 000’).

Примечание.

1). Для того, чтобы сделать автоприращение ключевого поля kod_faculteta измените в дополнительных свойствах столбца свойство – Спецификация идентификатора:



2). Для того, чтобы добавить проверочное ограничение на столбец Tel_decanata, выделите его и правой клавишей из меню выберите Проверочные ограничения. В появившемся окне нажмите кнопку Добавить. Будет создано ограничение под именем по умолчанию, для которого необходимо создать выражение вычисления ограничения на вводимые значения выбранного поля. В нашем случае ограничение на значение меньше ‘1 000 000’ будет иметь вид Tel_decanata< 1 000 000.



После введения данных о всех полях таблицы следует нажать кнопку [Сохранить] на панели инструментов.

Создадим для данной таблицы индексы (для других таблиц создавать индексы не нужно!!).

Для этого выберите из контекстного меню "Индексы и ключи" (рис. 4).

В нашем случае для первичного ключа автоматически присваивается уникальный индекс по возрастанию, см. рис.4.

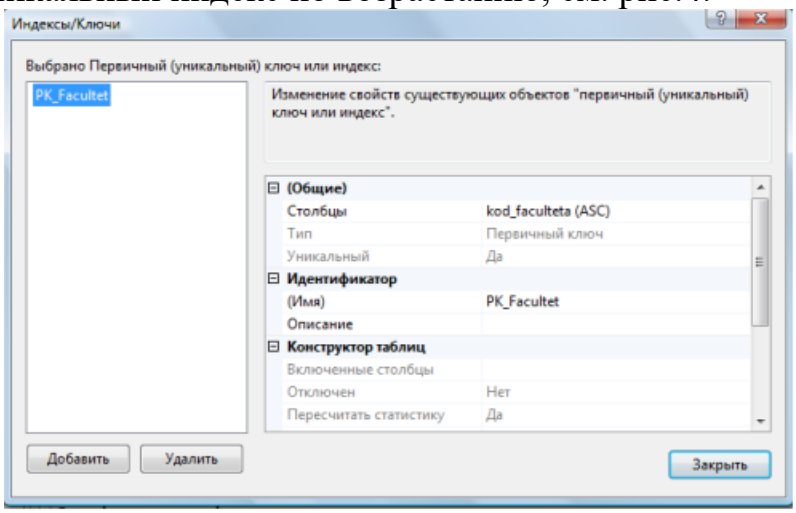


Рис. 4. Окно просмотра и редактирования индексов

Мастер позволяет просматривать, редактировать, создавать и удалять индексы.

Для создания индекса выполните следующие действия:

1. Нажмите в этой сетке клавишу [Добавить]. В результате будет вставлена новая строка.

2. Задайте в колонке "имя" имя индекса – My_Index_Facultet.

3. Нажмите кнопку в колонке "Столбцы". В результате откроется окно с для выбора столбцов и порядка сортировки. В левом списке "Имя столбца" будут находиться поля, которые можно добавить к индексу, в правом списке "Сортировка" находиться список сортировки. Для формирования перечня полей, которые будут входить в индекс, добавьте нужные поля.

4. Если создается уникальный индекс, то изменяется свойство в колонке "Уникальный" .

5. Чтобы создать индекс нажмите кнопку сохранить.

После создания индекса его можно в любой момент изменить, если изменить параметры индекса и снова нажать кнопку Сохранить.

Для заполнения таблиц данными вначале выделите таблицу в окне проводника и из контекстного меню выберите Изменить первые 200 строк. Вид окна представлен на рис. 5.

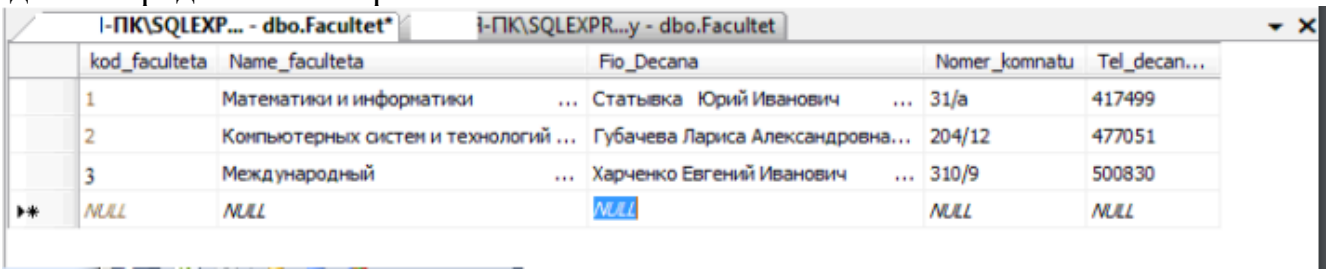


Рис. 5. Диалог создания записей в таблице

Введите самостоятельно три записи. Для добавления новой записи воспользуйтесь кнопкой перехода .

После ввода данных нажмите на панели кнопку о подтверждении изменения данных Выполнить SQL – код.

Примечание.

При вводе данных в поле kod_faculteta заполнять числами не нужно, они будут добавлены автоматически согласно формуле инкремента.

При вводе данных в поле tel_decanata значение можно оставить не заполненным. В результате выполнения sql-кода поле будет заполнено значением по-умолчанию.

При вводе данных в поле tel_decanata значение более 1 000 000 будет отображаться ошибка ввода связанной с проверочным ограничением.

Самостоятельно создайте новую таблицу, присвойте ей имя Kafedra.

Добавьте следующие поля в таблицу: Kod_kafedru (уникальный номер кафедры,

тип – целый, первичный ключ, автоинкремент), kod_faculteta (номер факультета, к которому принадлежит кафедра, тип – целый, не допускается пустых значений), Name_kafedru (название кафедры, тип текстовый, длина 50 символов, не допускается пустых значений), Fio_zavkaf (ФИО заведующего кафедрой, тип – текстовый), Nomer_komnatu (номер комнаты кафедры, тип – числовой, num_korpusa (номер корпуса, тип – числовой, с ограничением – номер корпуса может быть, только от 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12), Tel_kafedru (телефон кафедры, тип – текстовый).

Примечание.

Для создания ограничений для столбца num_korpusa используйте в выражении

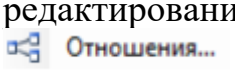
функцию

IN (‘знач1’,’знач2’, ..,’значn’)

Создадим связь между двумя таблицами Факультет и Кафедра.

Между данными таблицами можно установить связь «один-ко-многим», так как на одном факультете может быть несколько кафедр. Например, к факультету Математики и информатики относятся кафедры Информатики,

Компьютерные сети и системы, Прикладной математики и Математического анализа.

Чтобы создать связь «один-ко-многим» необходимо вызвать в окне редактирования таблицы Кафедра каскадное меню и выбрать Отношения 

.

Внешний ключ будет создан к полю kod_faculteta и оно будет связано с полем

первичным ключом kod_faculteta внешней таблицы Факультет.

Для этого в появившемся окне Связи по внешнему (см. рис. 6) перейдите на поле Спецификация таблиц и столбцов и нажмите кнопку напротив  для его изменения.

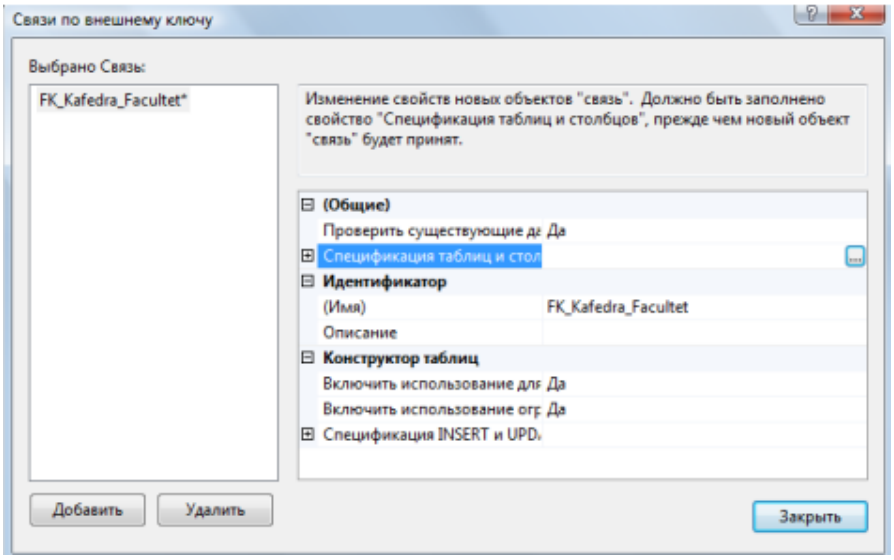


Рис. 6. Диалог создания внешних ключей

Вам будет открыто диалоговое окно создания Внешних ключей (рис. 7), где имя связи по умолчанию присвоенное внешнему ключу можете оставить без изменения.

Выберите в качестве таблицы первичного ключа – Facultet и поле kod_faculteta.

Выберите в качестве таблицы внешнего ключа – Kafedra и поле kod_faculteta.

Нажмите Ок и вернитесь в предыдущее окно.

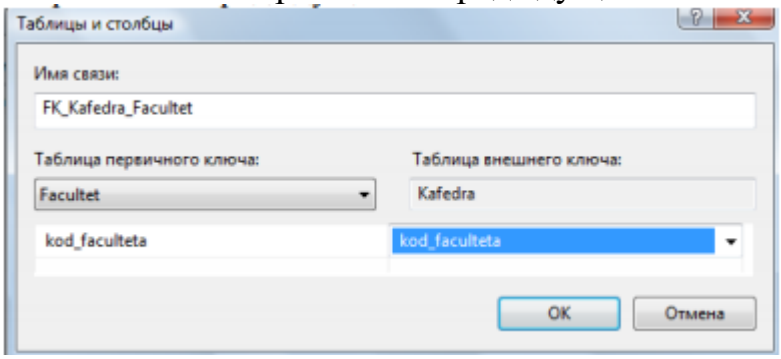


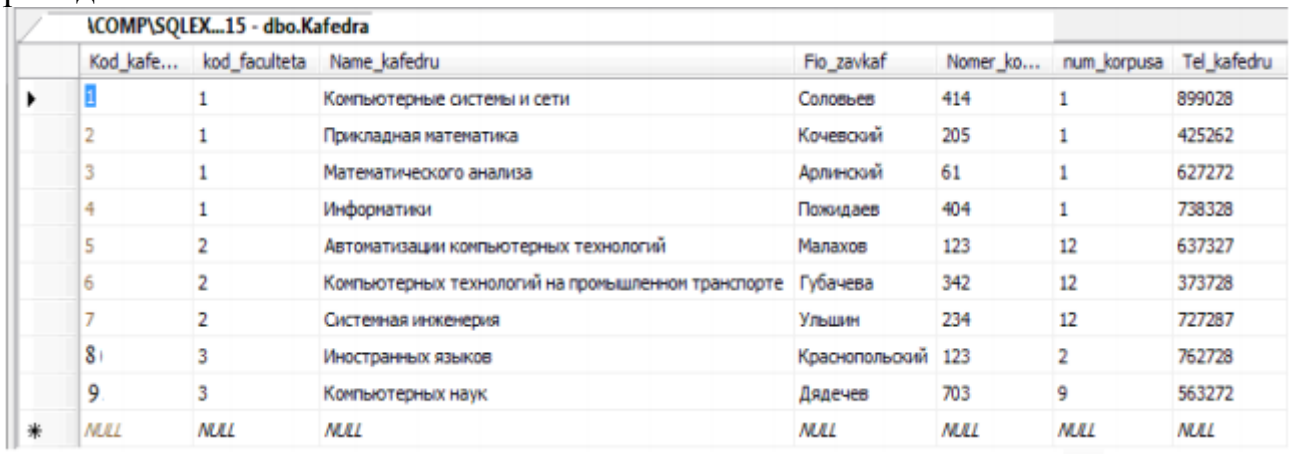
Рис. 7. Создание Внешних ключей

Установите правила удаления и обновления данных. Для этого перейдите на поле Спецификации INSERT и UPDATE и выберите правило каскадного удаления и обновления.

Сохраните таблицу.

После подтверждения операции у вас в таблице Кафедра появится внешний ключ по полю kod_faculteta. Для этого в окне проводника выберите в таблице Кафедра элемент Ключи и вы увидите первичный ключ и внешний ключ.

Введите самостоятельно название кафедр и их реквизиты на все созданные факультеты. Для добавления новой записи воспользуйтесь кнопкой перехода .

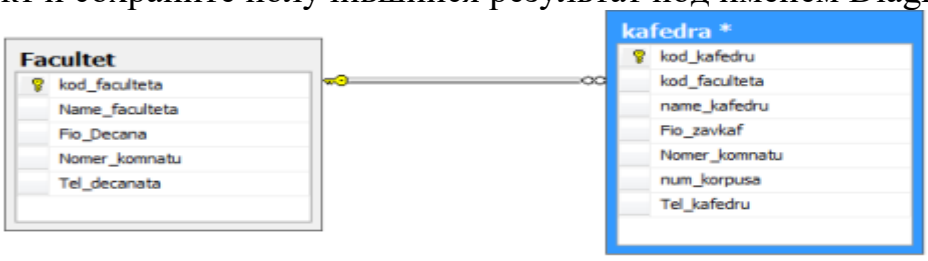


\COMP\SQLEX...15 - dbo.Kafedra

Kod_kafe...	kod_faculteta	Name_kafedru	Fio_zavkaf	Nomer_ko...	num_korpusa	Tel_kafedru
1	1	Компьютерные системы и сети	Соловьев	414	1	899028
2	1	Прикладная математика	Кочевский	205	1	425262
3	1	Математического анализа	Арлинский	61	1	627272
4	1	Информатики	Пожидаев	404	1	738328
5	2	Автоматизации компьютерных технологий	Малахов	123	12	637327
6	2	Компьютерных технологий на промышленном транспорте	Губачева	342	12	373728
7	2	Системная инженерия	Ульшин	234	12	727287
8	3	Иностранных языков	Краснопольский	123	2	762728
9	3	Компьютерных наук	Дядечев	703	9	563272
NULL	NULL	NULL	NULL	NULL	NULL	NULL

После ввода данных нажмите на панели кнопку о подтверждении .

Для визуального отображения связей между таблицами воспользуйтесь средством Диаграммы базы данных, добавьте две созданных вами таблицы в проект и сохраните получившийся результат под именем Diagram_BD.



Форма отчета: отчет, защита работы.

Практическое занятие № 8

Тема: Создание проекта БД. Создание БД. Редактирование и модификация таблиц.

Цель: Изучение структурированного языка запросов Transact - SQL, являющегося
основой системы программирования SQL Server, и приобретение навыков применение инструментальных средств разработки и программирования объектов создаваемых баз данных. Изучить SQL-операторы для работы с

таблицами и индексами. Изучить sqlкоманды для создания, изменения и удаления таблиц. Изучить используемые в SQL Server типы ограничений. Изучить SQL-операторы для работы с ограничениями.

Оборудование: тетрадь, ручка, ПК

Методические указания: выполните задания

Ход выполнения:

Задание.

Самостоятельно, используя команды языка SQL, в базе данных Университет создать:

1). Новую таблицу под именем STUDENT (Студент) с помощью sql-операторов с полями:

STUDENT_ID – целого типа для уникальной идентификации записей в таблице

первичный ключ тип счетчик,

SUTNAME – текстового типа для обозначения имени студента,

SUTFNAME - текстового типа для обозначения фамилии,

STIPEND – действительного типа для обозначения стипендии. При этом на это

поле наложено ограничение числом – величина размера стипендии должна быть меньше 500.

KURS - целого типа для обозначения курса. При этом на это поле наложено

ограничение – курс на котором может учиться студент может принимать значение от 1 до 5,

CITY - текстового типа для обозначения города,

BIRTDAY –типа даты/времени для обозначения день рождения,

GROUP - текстового типа для обозначения студенческой группы,

KOD_KAFEDRU – целого типа для обозначения названия кафедры, на которой учится студент. Поле KOD_KAFEDRU из таблицы STUDENT и поле KOD_KAFEDRU из таблицы KAFEDRA связаны тем, что описывают одни и те же данные, т.е. содержат идентификаторы кафедр, информация о которых содержит база данных. Более того, значение идентификаторов кафедр, которые допустимы в таблице STUDENT, должны выбираться только из списка значений поля KOD_KAFEDRU, т.е. принадлежащих реально описанных в базе данных кафедрам. Т.е. между этими полями имеется прямая связь. Т.о. поле KOD_KAFEDRU из таблицы STUDENT будет являться внешним ключом.

Кроме того, при определении таблицы STUDENT запрещено использовать значение NULL – значений для столбцов STUDENT_ID, SUTNAME, SUTFNAME.

В качестве первичного ключа принято значение столбца STUDENT_ID.

Выполните sql-код. Обновите базу данных и просмотрите созданную таблицу.

Сохраните sql-запрос под именем Студент.sql в папке ФИО_студента/Лаб8.

2). Новую таблицу под именем TEACHER (Преподаватели) с помощью sqlоператоров. Эта таблица содержит информацию о преподавателях вуза. Каждый преподаватель может работать на одной кафедре, иметь множество лекционных занятий и быть куратором более чем одной группы.

Описание столбцов таблицы TEACHER

KOD_TEACHER Уникальный идентификатор преподавателя. Является первичным ключом

KOD_KAFEDRU Уникальный идентификатор кафедры, на которой работает преподаватель. Является внешним ключом

NAME_TEACHER Фамилия преподавателя

INDEF_KOD Идентификационный код. Является уникальным для преподавателя

DOLGNOST Должность, может принимать только определенные значения из списка, такие как 'профессор', 'доцент', 'старший преподаватель', 'ассистент'. Значение по умолчанию 'ассистент'.

ZVANIE Научное звание, может принимать только определенные значения из списка, такие как 'к.т.н', 'к.г.у', 'к.с.н', 'к.ф.м.н.', 'д.т.н', 'д.г.у', 'д.с.н', 'д.ф.м.н', 'нет'. Значение по умолчанию 'нет'.

SALARY ставка зарплаты . Значение по умолчанию 1000. Зарплата должна быть больше нуля.

RISE надбавка к зарплате. Ее значение по умолчанию =0 и не может быть отрицательным числом.

DATA_HIRE дата приема на работу. По умолчанию текущая дата.

BIRTHDAY день рождения

POL пол, может принимать только определенные значения из списка, 'ж', 'Ж', 'м', 'М'

TEL_TEACHER Телефон. Может принимать значения только в виде '[1-9][0-9]- [0-9][0-9]-[0-9][0-9]'.

В качестве первичного ключа принято значение столбца KOD_TEACHER.

Поле KOD_KAFEDRU из таблицы TEACHER и поле KOD_KAFEDRU из таблицы KAFEDRA связаны тем, что описывают одни и те же данные, т.е. содержат идентификаторы кафедр, информация о которых содержит база данных. Более того, значение идентификаторов кафедр, которые допустимы в таблице TEACHER, должны выбираться только из списка значений поля KOD_KAFEDRU, т.е. принадлежащих реально описанных в базе данных кафедрам. Т.е. между этими полями имеется прямая связь. Т.о. поле KOD_KAFEDRU из таблицы TEACHER будет являться внешним ключом.

Выполните sql-код. Обновите базу данных и просмотрите созданную таблицу.

Сохраните sql-запрос под именем Преподаватель.sql в папке ФИО_студента/Лаб8.

САМОСТОЯТЕЛЬНО используя команды языка SQL!!!:

Создать на языке Transact-SQL файл базы данных согласно номеру варианта (присвоить ей новое имя).

Создать программно на языке SQL все таблицы, с указанием первичных и

внешних ключей и ограничения целостности.

Все программные инструкции команд SQL сохранять в файлах с расширением *.sql в папке ФИО_студента/Лаб4.

Заполнить таблицы данными по 5 записей в каждой.

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

Самостоятельно заполните вручную данными таблицы Студент и Преподаватель согласно рисункам, приведенным ниже.

Также ранее должны были введены следующие данные:

I-ПК\SQLEXP... - dbo.Facultet* | I-ПК\SQLEXPR...y - dbo.Facultet

kod_faculteta	Name_faculteta	Fio_Decana	Nomer_komnatu	Tel_decan...
1	Математики и информатики	Статывка Юрий Иванович	31/a	417499
2	Компьютерных систем и технологий	Губачева Лариса Александровна	204/12	477051
3	Международный	Харченко Евгений Иванович	310/9	500830
NULL	NULL	NULL	NULL	NULL

\COMP\SQLEX...15 - dbo.Kafedra

Kod_kafe...	kod_faculteta	Name_kafedru	Fio_zavkaf	Nomer_ko...	num_korpusa	Tel_kafedru
1	1	Компьютерные системы и сети	Соловьев	414	1	899028
2	1	Прикладная математика	Кочевский	205	1	425262
3	1	Математического анализа	Арлинский	61	1	627272
4	1	Информатики	Пожидаев	404	1	738328
5	2	Автоматизации компьютерных технологий	Малахов	123	12	637327
6	2	Компьютерных технологий на промышленном транспорте	Губачева	342	12	373728
7	2	Системная инженерия	Ульшин	234	12	727287
8	3	Иностранных языков	Краснопольский	123	2	762728
9	3	Компьютерных наук	Дядечев	703	9	563272
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Новые данные ввести вручную на всю группу

-ПК\SQLEXP...ty - dbo.kafedra | ПК\SQLEX...y - dbo.STUDENT | добавление.sql - (| I-ПК\....\...)* | -ПК\SQLEX...y - dbo.

STUDEN...	SUTNAME	SUTFNAME	STIPEND	KURS	CITY	BIRTHDAY	KOD_KAFEDRU	GROUP
2	Анна	Грешкина	450	2	Ростов-на-Дону	1989-01-01	1	МТ-401
3	Борис	Котовский	400	2	Ростов-на-Дону	1989-05-27	1	МТ-401
4	Петр	Комаров	300	2	Ростов-на-Дону	1989-11-18	1	МТ-401
5	Марина	Погребняк	353	2	Ростов-на-Дону	1989-10-21	1	МТ-402
6	Иванна	Смирнова	400	2	Ростов-на-Дону	1989-03-11	1	МТ-402
7	Роман	Сергеенко	0	2	Ростов-на-Дону	1989-07-04	1	МТ-402
8	Владимир	Невечеров	100	3	Ростов-на-Дону	1989-03-24	2	МТ-231
9	Мая	Соломка	450	3	Ростов-на-Дону	1988-01-12	2	МТ-231

Форма отчета: отчет, защита работы.

Практическое занятие № 9

Тема: Работа с записями базы данных. Импорт данных в таблицы**.**

Цель: изучить используемый в реляционных СУБД оператор извлечения данных из таблиц. Получить навыки работы с оператором SELECT в программе ‘SQL Server Managmant Studio’.

Оборудование: тетрадь, ручка, ПК
Методические указания: выполните задания
Ход выполнения:
Задание 1. Изучить синтаксис оператора SELECT и примеры запросов к учебной базе данных ‘University.mdf’.

Примечание. У вас должны быть перед выполнением этой практической работы созданы все таблицы базы данных университета, созданы ключи, а также заполнены данными.

Выполнение sql-запросов

Для выполнения запросов SELECT в программе ‘SQL Server Managmant Studio’

необходимо выполнить следующие действия:

1. Подключиться к базе данных и выполнить команду ‘Создать запрос’. В результате откроется окно ‘Конструктора запросов’ (рис. 1).

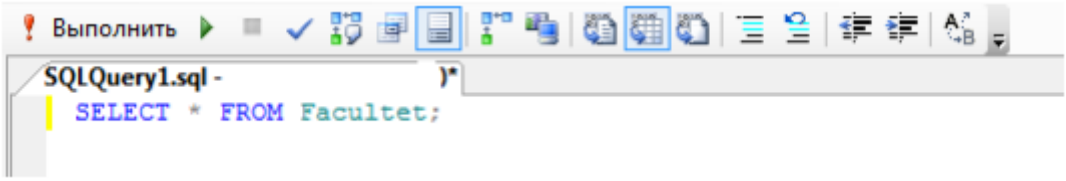


Рис. 1. Окно выполнения запросов

2. Ввести текст запроса согласно рис.1.

3. Нажать на панели инструментов кнопку  [Выполнить] .

4. Если запрос правильный, то в результате произойдет его выполнение и результат будет отображен на вкладке ‘Результаты’ (рис. 2).

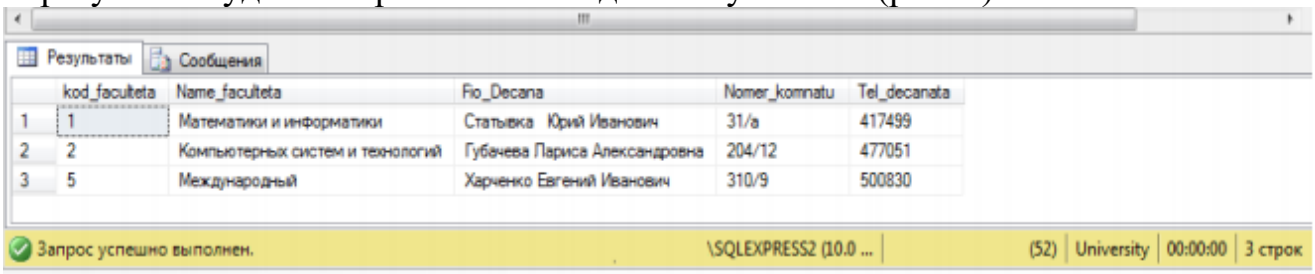


Рис. 2. Окно с результатом выполнения запроса

5. Количество извлеченных в результате выполнения запроса строк отображается над сеткой с данными справа. На рис.2 там содержится строка ‘3 строк’. В данном примере извлечено столько строк, сколько требуется, чтобы заполнить сетку (в ней помещается только 3 строки) *.

6. Чтобы узнать, сколько всего строк соответствуют выполненному оператору, надо перейти в конец отображаемого набора данных.

Чтобы выполнить другой запрос, надо вернуться на вкладку ‘Редактора’, создать новый запрос и повторить те же действия.

К тексту ранее выполнявшихся правильных запросов можно вернуться, если перейти на вкладку ‘История’.

Тема Примеры создания запросов с отбором строк по условию.

SQL дает возможность определить критерии отбора необходимых строк во фразе WHERE предложения SELECT. В этом случае строки исходных таблиц будут включены в результирующую только если строка соответствует

указанным критериям. Условие — это выражение, которое может быть истинным или ложным (логическое выражение или предикат), то есть принимать логические значения TRUE или FALSE соответственно. В результирующую таблицу включаются только те строки, для которых указанное во фразе WHERE условие равно TRUE (иными словами, которые удовлетворяют заданному условию).

В случае одной таблицы механизм работы предложения SELECT с фразой WHERE следующий.

1. Из таблицы, указанной во фразе FROM, выбирается очередная строка.
2. Она проверяется на соответствие условию во фразе WHERE.
3. Если результат равен TRUE, строка включается в результирующую таблицу и форматируется в соответствии с фразой SELECT, а если он равен FALSE, строка пропускается.

Далее будут рассмотрены основные выражения, допустимые для условия во фразе WHERE.

Использование простейших условий Простейшими считаются условия, в которых используются операторы сравнения и логические операторы.

Хотя такие условия являются простейшими в смысле семантики конструкций, они могут иметь довольно сложную структуру из многих операторов сравнений и вложенных друг в друга логических связок.

Операторы сравнения

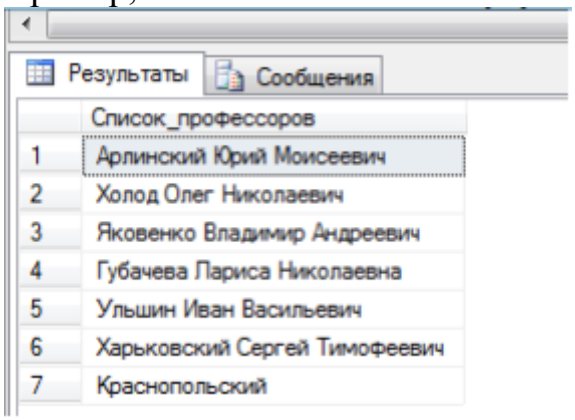
Особенностью операторов сравнения является то, что независимо от типов операндов их результатом являются логические значения. Предположим, вы хотите получить список всех профессоров.

Создайте новый запрос, введите sql-запрос, выполните его, сохраните его в рабочую папку ЛАБ6_SQL под именем 1.sql.

Запрос 1. Вывести фамилии профессоров.

```
SELECT NAME_TEACHER AS ‘Список профессоров’
FROM TEACHER
WHERE DOLGNOST = 'профессор';
```

Чтобы выполнить sql-команду нажмите на панели редактора кнопку . В результате выполнения данного кода будут выданы все профессора. Например,



	Список_профессоров
1	Арлинский Юрий Моисеевич
2	Холод Олег Николаевич
3	Яковенко Владимир Андреевич
4	Губачева Лариса Николаевна
5	Ульшин Иван Васильевич
6	Харьковский Сергей Тимофеевич
7	Краснопольский

Слово 'профессор' в запросе является строковой константой, поэтому ее следует заключить в кавычки. Обратите внимание, что мы указали фразу

SELECT без ключевого слова DISTINCT, так как тогда от нас была бы скрыта информация о существовании среди профессоров однофамильцев. Чтобы при выводе результирующий столбец имел содержательный заголовок, мы поименовали его как Список профессоров.

Это первый пример использования предиката над строковым типом данных. Здесь столбец строкового типа сравнивается со строковой константой. Запрос выполнен правильно, однако нужно всегда помнить о том, что предикаты над строками являются чувствительными к регистру букв. Например, предикат 'ИВАНОВ' = 'Иванов' будет ложным. Поэтому, если для некоторого профессора его должность была введена в таблицу TEACHER как 'Профессор', он не будет найден по условию WHERE DOLGNOST = 'профессор'.

Чтобы на предикаты над строками не влиял регистр букв, нужно использовать обычно имеющиеся в СУБД функции преобразования букв в прописные и строчные. В стандарте SQL, например, указаны функции UPPER и LOWER, выполняющие такие преобразования. Следовательно, для предыдущего запроса правильней будет записать условие фразы WHERE одним из следующих способов:

WHERE LOWER(DOLGNOST) = 'профессор'

WHERE UPPER(DOLGNOST) = 'ПРОФЕССОР'

Самостоятельно измените в исходном запросе строку условия с использованием функции изменения регистра.

Чтобы сохранить запрос нажмите правую кнопу и из контекстного меню выберите Сохранить в файл. Присвойте имя 1.sql и сохраните в папку ЛАБ5_SQL.

Запрос 2. Найти всех студентов с стипендией, превышающим 300.

В sql-редакторе создайте новый запрос и введите следующий код:

```
SELECT SUTNAME, SUTFNAME
FROM STUDENT
WHERE STIPEND > 300;
```

Выполните его.

Приведем несколько примеров использования операторов сравнения для столбцов строкового и временного типа. Обратите внимание, что сравнивать можно не только значение столбца с константой, но и значения столбцов между собой.

Запрос 3. Вывести фамилии и должности преподавателей, принятых на работу после 01.01.2012.

```
SELECT NAME_TEACHER AS 'Фамилия', DOLGNOST AS 'Должность'
FROM TEACHER
WHERE DATA_HIRE > '1/01/2002';
```

Запрос 4. Вывести фамилии и должности преподавателей, фамилии которых в алфавитном порядке располагаются после фамилии Иванова.

```
SELECT NAME_TEACHER, DOLGNOST
FROM TEACHER
WHERE UPPER(NAME_TEACHER) > 'Иванова';
```

Самостоятельно создайте запрос 5. Вывести фамилии преподавателей, у которых надбавка меньше ставки в 2,5 и более раз.

Логические операторы

Операндами и результатом логических операторов являются логические значения.

Стандартными логическими операторами являются AND, OR и NOT. Их действие показано в так называемых истинностных таблицах. Использование операторов сравнения вместе с логическими операторами предоставляет возможность формулировать составные условия для отбора строк таблиц.

Использование логического оператора AND

Логический оператор AND во многих случаях действует как связка ‘и’ в русском языке. Рассмотрим несколько примеров с использованием этого оператора.

Запрос 6. Вывести фамилии студентов, проживающих в городе Черемхово и имеющих стипендию больше 480.

```
SELECT SUTFNAME
FROM STUDENT
WHERE CITY = 'Черемхово' AND STIPEND >480;
```

Самостоятельно создайте запрос 7. Вывести фамилии преподавателей, которые

имеют высшую категорию и ставка которых превышает 6500.

Самостоятельно создайте запрос 8. Вывести фамилии студентов учащихся на кафедре под порядковым номером 2 (Прикладная математика) с стипендией в диапазоне 100-500.

Использование логического оператора OR

Логический оператор OR во многих случаях действует как связка ‘или’ в русском языке. Рассмотрим несколько примеров.

Форма отчета: отчет, защита работы.

Практическое занятие № 10

Тема: Создание ключевых полей. Задание индексов. Установление и удаление связей между таблицами.

Цель: изучить используемый в реляционных СУБД оператор извлечения данных из таблиц. Получить навыки работы с оператором SELECT в программе ‘SQL Server Managmant Studio’

Оборудование: тетрадь, ручка, ПК

Методические указания: выполните задания

Ход выполнения:

Задание 1. Рассмотреть теоретический материал.

При проектировании стремятся создавать таблицы, в каждой из которых содержалась бы информация об одном и только одном типе сущности. Это облегчает модификацию базы данных и поддержку ее целостности. Именно так мы поступили, создавая учебную базу данных. Однако сущности могут быть взаимосвязанными.

Кафедры связаны с факультетами по признаку вхождения в их состав, преподаватели работают на кафедрах, студенты учатся на кафедрах и т. д.

Связь между таблицами устанавливается за счет размещения специального столбца первичного ключа одной таблицы, которая называется родительской, в другой таблице, которая называется дочерней. Столбец (или совокупность столбцов) дочерней таблицы, определенный для связи с родительской таблицей, называется внешним ключом.

Наличие внешних ключей является основной для инициирования поиска по многим таблицам.

Одна из наиболее важных особенностей предложения SELECT — это способность использования связей между различными таблицами, а также вывода содержащейся в них информации. Операция, которая приводит к соединению из двух таблиц всех пар строк, для которых выполняется заданное условие, называется соединением таблиц. Для того чтобы указать соединяемые таблицы, их следует перечислить через запятую во фразе FROM.

Декартово произведение таблиц

Соединение таблиц — это частный случай операции декартового произведения (или просто произведения). Декартово произведение двух таблиц — это таблица, состоящая из всех возможных пар строк обеих таблиц. Это определение можно естественным образом расширить на любое количество таблиц. В SQL декартово произведение выражается указанием имен перемножаемых таблиц во фразе FROM и указанием всех их столбцов во фразе SELECT.

Так, произведение таблиц FACULTET и KAFEDRA выражается следующим образом:

```
SELECT *
FROM FACULTET, KAFEDRA
```

Так как результирующая таблица содержит много столбцов, и они не помещаются по ширине страницы, мы приведем только интересующие нас столбцы произведения этих таблиц.

Задание 2. Создать запросы.

Запрос 1. Декартово произведение таблиц.

```
SELECT FACULTET.Name_faculteta, FACULTET. Kod_faculteta,
KAFEDRA. Kod_faculteta, KAFEDRA.Name_Kafedru
FROM FACULTET, KAFEDRA;
```

Каждая строка таблицы факультетов оказалась соединенной с каждой строкой таблицы кафедр, в результате получилось 27 строк (3 факультета х 9 кафедр = 27 комбинаций).

В произведении может участвовать много таблиц. Например, произведение таблиц факультетов, кафедр и преподавателей записывается следующим образом:

```
SELECT *
FROM FACULTET, KAFEDRA, TEACHER
```

Условие соединения

Соединение таблиц может быть указано во фразе WHERE или во фразе FROM.

Сначала рассмотрим первый вариант. Большинство запросов, имеющих несколько таблиц во фразе FROM, содержат фразу WHERE, в которой указаны условия, попарно сравнивающие столбцы из различных таблиц. Такое условие называется условием соединения. В этом случае SQL предполагает сцепление только тех пар строк из разных таблиц, для которых условие соединения принимает истинное значение. Теоретически при соединении сначала выполняется декартово произведение указанных таблиц в одну, а затем из нее отбираются строки согласно условию соединения. Естественно, ни одна СУБД не работает таким образом.

Фраза WHERE помимо условия соединения может также содержать другие условия, каждое из которых ссылается на столбцы соединенной таблицы. Эти условия производят отбор строк соединенной таблицы.

Соединения можно разделить на следующие категории.

Внутренние соединения (типичные операции соединения, использующие такие операторы сравнения, как = или <>). Они включают эквивалентные соединения и естественные соединения.

Внутренние соединения используют оператор сравнения для установки соответствия строк из двух таблиц на основе значений общих столбцов в каждой таблице. Примером может быть получение всех строк, в которых идентификационный номер студента одинаковый как в таблице students, так и в таблице courses.

Внешние соединения. Внешние соединения бывают левыми, правыми и полными.

Если внешние соединения задаются в предложении FROM, они указываются с одним из следующих наборов ключевых слов.

LEFT JOIN или LEFT OUTER JOIN

Результирующий набор левого внешнего соединения включает все строки из левой таблицы, заданной в предложении LEFT OUTER, а не только те, в которых соединяемые столбцы соответствуют друг другу. Если строка в левой таблице не имеет совпадающей строки в правой таблице, результирующий набор строк содержит значения NULL для всех столбцов списка выбора из правой таблицы.

RIGHT JOIN или RIGHT OUTER JOIN

Правое внешнее соединение является обратным для левого внешнего соединения.

Возвращаются все строки правой таблицы. Для левой таблицы возвращаются значения NULL каждый раз, когда строка правой таблицы не имеет совпадающей строки в левой таблице.

FULL JOIN или FULL OUTER JOIN

Полное внешнее соединение возвращает все строки из правой и левой таблицы.

Каждый раз, когда строка не имеет соответствия в другой таблице, столбцы списка выбора другой таблицы содержат значения NULL. Если между таблицами имеется соответствие, вся строка результирующего набора содержит значения данных из базовых таблиц.

Перекрестные с соединения Перекрестное соединение возвращает все строки из левой таблицы. Каждая строка из левой таблицы соединяется со всеми строками из правой таблицы. Перекрестные соединения называются также декартовым произведением.

Таблицы или представления в предложении FROM могут указываться в любом порядке с внутренним соединением или полным внешним соединением. Однако важен порядок таблиц или представлений, заданных при использовании левого или правого внешнего соединения.

Соединение таблиц по равенству

Если таблицы соединяются по равенству значений пары столбцов (группы столбцов) из различных таблиц, такая операция называется соединением таблиц по равенству. Соединение по равенству, в отличие от декартового произведения, позволяет соединить только те пары строк, которые действительно взаимосвязаны друг с другом.

Так, например, мы можем соединить таблицы факультетов и кафедр по условию FACULTET.Kod_faculteta = KAFEDRA.Kod_faculteta. В таком варианте мы соединяем таблицы осмысленно, так как каждая строка таблицы FACULTET соединяется только со строками соответствующих кафедр. На базе таблиц FACULTET и KAFEDRA мы получаем таблицу со столбцами из обеих таблиц, имеющую строки с понятным смыслом.

Можно также сказать, что в таблицу KAFEDRA вместо столбца Kod_faculteta мы вставляем все характеристики (столбцы) соответствующего факультета из таблицы FACULTET.

Соединение таблиц используется, когда необходимо вывести значения столбцов:

– разных таблиц;

– одной таблицы, но отвечающих условию, заданному на другой таблице.

Эти два варианта, а также их комбинация, характерны для любого вида соединения, а не только по равенству. Перейдем к рассмотрению примеров.

Вывод столбцов разных таблиц

Этот вид запросов характерен тем, что фраза WHERE содержит только условие соединения, а список фразы SELECT содержит имена столбцов из различных таблиц.

Запрос 2. Вывести названия кафедр и номера их групп.

```
SELECT Name_Kafedru, [Group]
FROM KAFEDRA, STUDENT
WHERE KAFEDRA.kod_kafedru = STUDENT.kod_kafedru;
```

или

```
SELECT Name_Kafedru, student.[GROUP]
FROM KAFEDRA, STUDENT
WHERE KAFEDRA.kod_kafedru = STUDENT.kod_kafedru;
```

Мы привели два варианта запроса. В первом имена столбцов не уточняются

именами таблиц, а во втором — уточняются. В данном случае это не имеет значения, оба запроса корректны.

Уточнение имен столбцов

До тех пор, пока запрос относится к одной таблице, обращение к столбцам по их именам не вызывает проблем — в таблице все имена столбцов должны быть неповторяющимися. Однако, как только запрос соединяет несколько таблиц, может возникнуть неоднозначность при ссылках на столбцы с одинаковыми именами из разных таблиц. Для разрешения этой неоднозначности во фразах SELECT и WHERE (как и в некоторых других фразах) имена столбцов необходимо уточнять именами таблиц.

Запрос 3. Вывести названия факультетов и их кафедр.

```
SELECT FACULTET.NAME_FACULTETA, KAFEDRA.Name_Kafedru
FROM FACULTET, KAFEDRA
WHERE FACULTET.Kod_faculteta = KAFEDRA.Kod_faculteta;
```

В этом запросе мы уточнили имена столбцов во фразах SELECT и WHERE, хотя во втором случае это не обязательно, так как используются неповторяющиеся имена. Тем не менее, рекомендуем при соединении таблиц для наглядности уточнять имена столбцов.

Обратите внимание на то, что в предыдущем примере отсутствует факультет математики — на нем нет кафедр.

Вывод столбцов с условием отбора Вариант, когда отбираются строки одной таблицы, а условие задается с участием другой, используется довольно часто. Приведем примеры.

Запрос 4. Вывести названия кафедр факультета Математики и информатики.

```
SELECT KAFEDRA.Name_Kafedru AS ‘Кафедры факультета математики и
информатики’ FROM FACULTET, KAFEDRA
WHERE FACULTET. Kod_faculteta = KAFEDRA. Kod_faculteta AND
LOWER(FACULTET.NAME_FACULTETA) = 'математики и информатики';
```

Запрос 4. Вывести фамилии доцентов кафедры информатики.

```
SELECT TEACHER.NAME_TEACHER AS ‘Доценты кафедры информатики’
FROM KAFEDRA, TEACHER
WHERE KAFEDRA.kod_kafedru = TEACHER. kod_kafedru AND
LOWER(KAFEDRA.Name_Kafedru) = ‘информатики' AND
LOWER(TEACHER.DOLGNOST) = 'доцент';
```

В последнем запросе помимо условия соединения используется также отбор строк по условиям, заданным для разных таблиц.

Синонимы таблиц

Синонимы таблиц часто используются для задания более лаконичного имени таблицы, по которому можно сослаться на нее в любых других местах запроса.

Запрос 5. Вывести названия кафедр, на которых имеются студенты со стипендией >480.

```
SELECT DISTINCT k.Name_Kafedru
FROM KAFEDRA k, STUDENT s
```

WHERE k.Kod_kafedru = s. Kod_kafedru AND s.Stipend > 480;

Запросы по трем и более таблицам

SQL позволяет формулировать запросы, которые предполагают использование трех и более таблиц. При этом следует применять ту же методику соединения, что и для двух таблиц.

Запрос 6. Вывести названия тех кафедр факультета математики и информатики, на которых работают профессора.

```
SELECT DISTINCT KAFEDRA.Name_Kafedru
FROM FACULTET, KAFEDRA, TEACHER
WHERE FACULTET.Kod_faculteta = KAFEDRA.Kod_faculteta AND
KAFEDRA.Kod_kafedru = TEACHER.Kod_kafedru AND
FACULTET.Name_faculteta ='Математики и информатики' AND
TEACHER.DOLGNOST = 'профессор';
```

Для ответа на запрос необходимы три таблицы: на таблицах факультетов и преподавателей заданы условия отбора, а из таблицы кафедр следует вывести столбец названий. Поэтому три необходимые таблицы указываются во фразе FROM, а во фразе WHERE производится их соединение по условию равенства первичного и внешнего ключей:

FACULTET. Kod_faculteta = KAFEDRA. Kod_faculteta -- соединение таблиц факультетов и кафедр

KAFEDRA. Kod_kafedru = TEACHER. Kod_kafedru -- соединение таблиц кафедр и преподавателей

Таблица, образующаяся в результате соединений, будет иметь столько же строк, сколько имеется в таблице преподавателей (если все преподаватели работают на кафедрах). Выясним, почему это так, но сначала заметим, что результат соединения таблиц не зависит от порядка соединения. Поэтому рассмотрим случай, когда сначала мы соединяем таблицы кафедр и преподавателей, а затем результат соединяем с таблицей факультетов.

Так как между таблицами кафедр и преподавателей существует связь типа одинко-многим, их соединение фактически означает приписывание к строке каждого преподавателя данных о его кафедрах. Количество строк этого соединения будет равным количеству преподавателей. Связь между таблицами факультетов и кафедр также имеет тип один-ко-многим, поэтому второе соединение означает, что к каждой строке таблицы, полученной после первого соединения, приписываются данные о факультете кафедры.

Таким образом, количество строк останется равным числу преподавателей.

Вернемся к запросу. Последние два условия фразы WHERE отбирают строки из соединенной таблицы, а во фразе SELECT указан выводимый столбец. Ключевое слово

DISTINCT указано в нем потому, что названия кафедр в соединенной таблице могут повторяться.

Форма отчета: отчет, защита работы.

Практическое занятие № 11

Тема: Проведение сортировки и фильтрации данных.

Цель: 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.
Оборудование: тетрадь, ручка, ПК
Методические указания: выполните задания
Ход выполнения:
Задание 1. Изучить синтаксис создания запросов с использованием функций, группировки и сортировки данных, язык манипулирования данных на примерах запросов, использовать встроенные функции к учебной базе данных "University.mdf".

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.

Функция COUNT

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

C0UNT(*)

C0UNT([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 ЛАБ11_SQL под именем 1.sql.

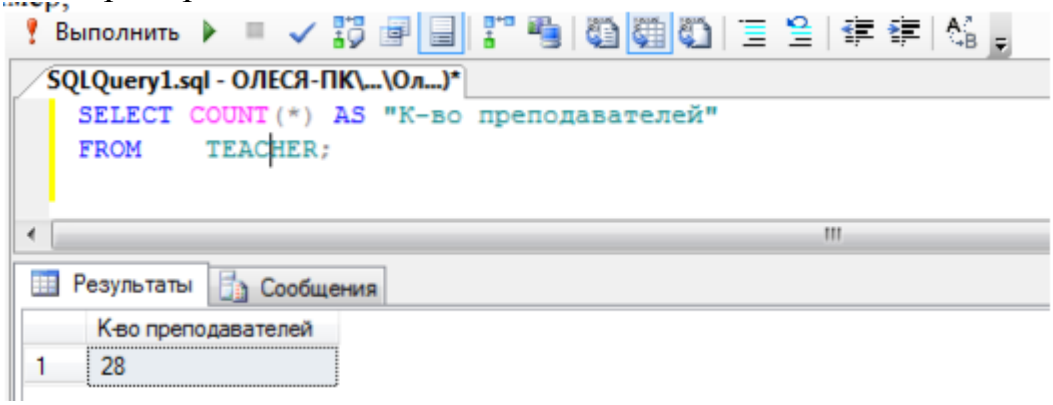
Задание 2. Выполнить в окне "SQL Editor" запросы к базе данных "University.mdf", согласно приведенным в практической работе образцам выполнения запросов и сохранить их в файле "Lab11.sql" в своей рабочей папке.

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 C0UNT(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.

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 кафедру информатики?

Выражения с использованием агрегатных функций

Агрегатные функции не только могут иметь выражение в своем аргументе, но и 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;

Форма отчета: отчет, защита работы.

Практическое занятие № 12

Тема: Поиск данных по одному и нескольким полям. Поиск данных в таблице.

Цель: изучить поиск данных по одному и нескольким полям.

Оборудование: тетрадь, ручка, ПК

Методические указания: выполните задания

Ход выполнения:

Задание 1: выполнить в окне "SQL Editor» запросы к базе данных "University.mdf", согласно приведенным в практической работе образцам выполнения запросов и сохранить их в файле "Lab12.sql" в своей рабочей папке.

Однострочные 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 1. 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, миминум, 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 деляться 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 числа.

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 компонентdatepart 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 2. 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 3. 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 'C%';
```

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 4. Например, если вы хотите узнать, сколько месяцев уже проработал

Статывка, можно выполнить такой запрос:

```
SELECT 'Статывка проработал ' ||
R0uND(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_DAУ(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)

Форма отчета: отчет, защита работы.

Практическое занятие № 13

Тема: Задание значений и ограничений поля. Проверка введенного в поле значения. Отображение данных числового типа и типа дата.

Цель: изучить задание значений и ограничений поля и проверку введенного в поле значения. Отображение данных числового типа и типа дата.

Оборудование: тетрадь, ручка, ПК

Методические указания: выполните задания

Ход выполнения:

Задание: Выполнить в окне "SQL Editor" запросы к базе данных University.mdf", согласно приведенным в практической работе образцам

выполнения запросов и сохранить их в файле "Lab13.sql" в своей рабочей папке.

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 1. 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 2. 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 3. 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 4. 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 5. Для каждого значения зарплаты, не превышающего 1500, вывести это значение и количество преподавателей, такую зарплату получающих.

```
SELECT Salary + Rise, C0UNT(*)
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;
- - - - - - - - - - - - - - - - - - - - - - - - - - - -
ORA-00978: вложенная групповая функция без GROUP BY
```

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 6. 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 7. 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 8. 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 9. 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 10. 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 место расположения 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 11. 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 12. 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 13. 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 14. Вывести фамилии доцентов в обратном алфавитном порядке и их зарплату.

Форма отчета: отчет, защита работы.

Практическое занятие № 14

Тема: Соединения таблиц и подзапросы. Ограничения и представления

Цель: Изучение назначения и типов триггеров, условий их активации, синтаксиса и семантики команд языка Transact – SQL для их создания, модификации, переименования, программирования и удаления, а также приобретение навыков их проектирования, кодирования и отладки с применением хранимых процедур для получения информации о триггерах базы данных.

Оборудование: тетрадь, ручка, ПК

Методические указания: выполните задания

Ход выполнения:

Задание 1. Создать таблицу authsmall из таблицы authors базы данных Pubs и для новой таблицы запрограммировать триггер auth_del, который будет выводить информацию о попытках удаления и количестве удаляемых строк, выполнив действия:

1. Создание таблицы authsmall с колонками au_id, au_fname, au_lname, phone и копирование в нее данных из таблицы authors:

```
SELECT au_id, au_fname, au_lname, phone
INTO authsmoll
FROM authors
PRINT ‘Содержимое таблицы authsmoll:’
SELECT * FROM authsmall
```

2. Создание и программирование триггера:

```
CREATE TRIGGER auth_del
ON authsmall
FOR DELETE
AS
PRINT ‘Попытка удаления’ + STR (@@ POWCOUNT)+  ‘строк в таблице authsmall’
```

PRINT ‘Пользователь’ + CURRENT_USER
IF CURRENT_USER <> ‘dbo’
BEGIN
PRINT ‘Удаление запрещено’
ROLLBACK TRANSACTION
END
ELSE
PRINT ‘Удаление разрешено’
3. Тестирование триггера :
DELETE FROM authsmall WHERE au_fname = ‘Johnson’
DELETE FROM authsmall WHERE 2*2=5

Задание 2. Создать триггер auth_upd для таблицы authsmall, построенный в первом задании, который будет разрешать изменение столбца au_id этой таблицы всем, кроме владельца dbo, выполнив следующие действия:

1. Создание и программирование триггера:
CREATE TRIGGER auth_upd
ON authsmall
FOR UPDATE
AS
SET NOCOUNT ON -- не сообщать о завершении команд;
PRINT ‘Попытка изменения данных в таблице authsmall’
IF (COLUMNS_UPDATE () &1)! = 0 -- 1-й столбец;
PRINT ‘Изменение столбца au_id’
IF (COLUMNS_UPDATE () &2)! = 0 -- 2-й столбец;
PRINT ‘Изменение столбца au_fname’
IF (COLUMNS_UPDATE () &4)! = 0 -- 3-й столбец;
PRINT ‘Изменение столбца au_lname’
IF UPDATE (Phone)
PRINT ‘Изменение столбца phone’
IF ((CURRENT_USER = ‘dbo’) AND
(COLUMNS_UPDATED()&1)! = 0 -- 1-ый стлбец;
BEGIN
PRINT ‘Пользователь dbo не может изменять’ + ‘идентификационный номер автора’
ROLLBACK TRANSACTION
END
2. Тестирование триггера:
UPDATED authsmall SET phone =‘415 986 - 7020’, au_fname = ‘John’
WHERE au_lname = ‘Green’
UPDATED authsmall SET phone =‘913 843 - 7302’, au_id = ‘748-126859’
WHERE au_lname = ‘Smith’

Задание 3. Создать триггер для команд INSERT и UPDATE, запрещающий производить изменения для автора Billy Geitsi, выполнив действия:

1. Создание и программирование триггера:
CREATE TRIGGER auth_ ins_upd ON authsmall

179
FOR INSERT, UPDATE
AS
IF EXISTS (SELECT * FROM authsmall -- inserted;
WHERE au_lname = ‘Geitsi’ -- фамилия;
au_fname = ‘Billy’) -- имя;
BEGIN
PRINT ‘Недопустимо написание кнги’+
‘автором Billy Geitsi’
ROLLBACK TRANSACTION
END
2. Тестирование триггера:
UPDATE authsmall SET au_lname = ‘Geitsi’,
au_fname = ‘Billy’ WHERE au_lname = ‘Smith’..

Форма отчета: отчет, защита работы.

Практическое занятие № 15

Тема: Обработка транзакций. Использование функций защиты для БД.

Цель: Изучение способов обеспечения надежной работы SQL Server с помощью механизма транзакций и контрольных точек, приобретение навыков управления локальными и распределенными транзакциями различных видов, а также ознакомление с физической и логической архитектурой журнала транзакций и способами восстановления баз данных.

Оборудование: тетрадь, ручка, ПК

Методические указания: выполните задания

Ход выполнения:

Задание 1. Проверить режимы автоматического начала транзакций и неявного начала транзакций, используя переключатель IMPLICIT_TRANSACTION и команду COMMIT**.**

Задание 2. Создать несколькими способами распределенные транзакции и убедиться в корректности их выполнения.

Задание 3. Создать вложенные транзакции, выполнив следующие команды:

CREATE TABLE #aaa (cola int) -- 0-й уровень
BEGIN TRAN -- 1-й уровень
INSERT INTO #aaaVALUES (111)
BEGIN TRAN -- 2-й уровень
INSERT INTO #aaaVALUES (222)
BEGIN TRAN -- 3-й уровень
INSERT INTO #aaaVALUES (333)
SELECT * FROM #aaa
SELECT ‘Вложенность транзкций’, @@TRANCOUNT
ROLLBACK TRAN
SELECT * FROM #aaa -- откат на 0-й уровень
SELECT ‘Вложенность транзакций’, @@TRANCOUNT
Проанализировать полученные результаты.

Задание 4. Написать пример пакета запросов с использованием команд COMMIT
и ROLLBACK для автоматических, неявных и явных транзакций.
Задание 5. Написать пример пакета команд, иллюстрирующих использование средств оптимизации при откате транзакций.
Задание 6. Используя средства MS SQL Server 2000, изучить физическую и логическую архитектуру журнала транзакций.
Задание 7. С помощью системной хранимой процедуры sp_configure изменить интервал контрольных точек для базы данных Pubs.
Задание 8. Уточнить синтаксис команд управления транзакциями и написать пример пакета с использованием всех вариантов этих команд.
Задание 9. 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 вкладе 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 задать, то сервер будет автоматически подключать к базе master.
12. Включить создаваемую учетную запись в требуемую встроенную роль сервера: Sysadmin, Serveradmin, Setupadmin, Securityadmin, Processadmin, Dbereator, 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 10. 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 “OK” этих окон.
9. Повторить задания, используя следующий набор команд:
а) Security/Server Rolees;

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 11. 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, DELET, 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.

Форма отчета: отчет, защита работы.

4. ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ

4.1 Основные печатные и (или) электронные издания:

О-1. Илюшечкин, В. М. Основы использования и проектирования баз данных: учебник для среднего профессионального образования / В. М. Илюшечкин. — испр. и доп. — Москва: Издательство Юрайт, 2024. — 213 с. — (Профессиональное образование). — ISBN 978-5-534-01283-5. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/538545 (дата обращения: 02.05.2025).

4.2 Дополнительные печатные и (или) электронные издания (электронные ресурсы):

Д-1. Карпова, Т.С. Базы данных: модели, разработка, реализация: учебное пособие/ Т.С. Карпова. – М.: Питер, 2001. – 304 с.

Д-2. Риккарди, Г., Системы баз данных. Теория и практика использования в Интернет и среде Java. - М.: Вильямс, 2001. – 480 с.

Д-3. Малыхина, М.П. Базы данных: основы, проектирование, использование/ М.П. Малыхина. – М.: БХВ-Петербург, 2004. – 512 с.

Д-4. Глушаков, С.В., Ломотько Д.В. Базы данных: основы, проектирование, использование/ С.В. Глушаков, Д.В Ломотько: учебный курс. – М.: Абрис, 2000. – 504 с.

Д-5. Хомоненко, А.Д., Цыганков В.М., Мальцев М.Г. Базы данных/ А.Д. Хомоненко, В.М. Цыганков, М.Г. Мальцев: учебник– М.: Корона, 2003. – 672 с.

Д-6. Золотова, С.И. Практикум по Assess/ С.И. Золотова: Практикум – М.: Финансы и статистика, 2000. – 144 с.

Д-7. Голицына, О.Л., Максимов Н.В., Попов И.И. Базы данных: учебник/ О.Л.Голицына, Н.В.Максимов, И.И. Попов - М.: ИД "ФОРУМ"-ИНФРА-М, 2004. – 352 с.

Д-8. Кузин, А.В., Демин В.М. Разработка баз данных в системе Mikrossoft Assess: учебник/ А.В. Кузин, В.М.Демин. - М.: ИД "ФОРУМ"-ИНФРА-М, 2005. – 224 с.

Д-9. Кузин, А.В., Демин В.М. Разработка баз данных в системе Mikrossoft Assess: учебник/ А.В. Кузин, В.М.Демин. - М.: ИД "ФОРУМ"-ИНФРА-М, 2007. – 224 с.

ЛИСТ ИЗМЕНЕНИЙ И ДОПОЛНЕНИЙ, ВНЕСЕННЫХ В МЕТОДИЧЕСКИЕ УКАЗАНИЯ

№ изменения, дата внесения, № страницы с изменением	
Было	**Стало**
Основание: **Подпись лица, внесшего изменения**	