Министерство просвещения Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Набережночелнинский государственный педагогический университет»

Индустриально-педагогический колледж

Кафедра профессиональных дисциплин

Методические указания для подготовки к демонстрационному экзамену

по специальности
09.02.07 Информационные системы и программирование

Квалификация
Программист

Набережные Челны, 2024

Составитель:
Сахибулина О.Н., преподаватель кафедры профессиональных дисциплин

Сахибулина О.Н. Методические указания по подготовке к демонстрационному экзамену по специальности 09.02.07 Информационные системы и программирование / О.Н. Сахибулина. – Набережные Челны, 2024. – 21 с.

Методические рекомендации по подготовке к демонстрационному экзамену по специальности 09.02.07 Информационные системы и программирование адресовано обучающимся по образовательной программе среднего профессионального образования, реализуемой согласно ФГОС СПО по специальности 4909.02.07 Информационные системы и программирование, квалификация Программист, а также лицам, осуществляющим руководство и подготовку к ГИА. В методических рекомендациях раскрыты организационные и методические вопросы подготовки обучающихся к демонстрационному экзамену по КОД 09.02.07-2-2024.

Содержание

Пояснительная записка

Демонстрационный экзамен является одним из видов аттестационных испытаний выпускников, завершающих обучение, по образовательной программе среднего

профессионального образования, реализуемой согласно ФГОС СПО по специальности 09.02.07 Информационные системы и программирование, квалификация Программист в ФГБОУ ВО «Набережночелнинский государственный педагогический университет». Демонстрационный экзамен проводится с целью определения сформированности общих и профессиональных компетенций, качества освоения видов деятельности по образовательной программе среднего профессионального образования и подготовки обучающихся в соответствии с требованиями федерального государственного стандарта среднего профессионального образования. Демонстрационный экзамен предусматривает моделирование реальных производственных условий для решения обучающимися и выпускниками практических задач профессиональной деятельности.

Методические указания по подготовке к демонстрационному экзамену составлены в соответствии с требованиями КОД 09.02.07-2-2024.и образовательной программы, реализуемой согласно ФГОС СПО по специальности 09.02.07 Информационные системы и программирование в Федеральном государственном образовательном учреждении высшего образования «Набережночелнинский государственный педагогический университет».

Методические рекомендации состоят их двух частей. Первая часть включает общие организационные вопросы проведения демонстрационного экзамена. Вторая часть содержит рекомендации по подготовке обучающихся к демонстрационному экзамену.

Методические указания к демонстрационному экзамену предназначены для обучающихся по образовательной программе среднего профессионального образования по специальности 09.02.07 Информационные системы и программирование, а также лицам, осуществляющим подготовку к государственной итоговой аттестации.

Глава 1 Организационные вопросы проведения демонстрационного экзамена

1.1 Общая характеристика демонстрационного экзамена

Демонстрационный экзамен базового уровня - это экзамен, проводимый в

индивидуальной форме. Задание демонстрационного экзамена включает комплексную практическую задачу, моделирующую профессиональную деятельность и выполняемую в режиме реального времени. Участники разрабатывают содержание деятельности программиста, решая при выполнении экзаменационных заданий по модулям различные задачи.

Продолжительность Демонстрационного экзамена по КОД 09.02.07-2-2024 не должна быть более 2 часов 30 минут. Оценка знаний участника проводится исключительно через практическое выполнение Экзаменационного задания.

Задание демонстрационного экзамена включает комплексную практическую задачу, моделирующую профессиональную деятельность и выполняемую в режиме реального времени.

Демонстрационный экзамен организован по модульному принципу. Содержанием экзамена являются виды деятельности программиста. Участники экзамена получают алгоритм выполнения задания с описанием цели выполнения модуля и планируемыми результатами представления задания.

Условия проведения демонстрационного экзамена

Университет самостоятельно определяет площадку для проведения демонстрационного экзамена, которая может располагаться как в самом Университете, так и в другой организации на основании договора о сетевом взаимодействии. Ответственность сторон, финансовые и иные обязательства определяются договором о сетевом взаимодействии.

Демонстрационный экзамен проводится в центре проведения демонстрационного экзамена, представляющем собой площадку, оборудованную и оснащенную в соответствии с комплектом оценочной документации.

Перечень оборудования и оснащения, расходных материалов, средств обучения и воспитания

Количество рабочих мест: 10					
Перечень оборудования и оснащения, расходных материалов, средств обучения и воспитания					
№	Наименование	Минимальные (рамочные) технические характеристики	Кол-во на 1 рабочее место	Единица измерения	Кол-во на общее число рабочих мест
Перечень оборудования					
1	Персональный компьютер в сборе	ЦПУ: минимальная базовая тактовая частота 2.0 ГГц, количество физических ядер не менее 2, количество потоков не менее 4 ОЗУ: объем не менее 8Гб ПЗУ: SSD объемом не менее 256 Гб, либо SSHD/HDD объемом не менее 500 Гб Сетевой адаптер: технология Ethernet стандарта 100BASE T и/или 1000BASE-T Графический адаптер: стандарт не ниже	1	шт.	10

		WXGA, возможность подключения 2-х мониторов			
2	Монитор с комплектом кабелей, совместим с комплектом ЭВМ	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
3	Клавиатура	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
4	Компьютерная мышь	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
5	Интерфейсный кабель для подключения монитора	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
6	Кабель питания	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
7	Сетевой фильтр	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
8	Рабочий стол	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
9	Рабочий стул	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
10	ПО операционная система	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
11	ПО для просмотра документов в формате PDF	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
12	ПО для архивации	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
13	ПО для офисной работы	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10

14	ПО для построения и редактирования диаграмм (UML) и блок-схем	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
15	ПО среда разработки с библиотеками	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
16	Система управления базами данных	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
17	Среда для управления инфраструктурой SQL	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
18	ПО для развертывания локального сервера	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
19	ПО текстовый редактор	Программное обеспечение для работы с текстом. Характеристики позиции – на усмотрение образовательной организации.	1	шт.	10
20	ПО редактор кода	Программное обеспечение, способное поддерживать ряд языков программирования, подсветку синтаксиса, рефакторинг, отладку, навигацию по коду. Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
21	МФУ	Характеристики позиции – на усмотрение образовательной организации	1	шт.	1
22	Корзина для мусора	Характеристики позиции – на усмотрение образовательной организации	1	шт.	1
Перечень расходных материалов					
1	Ручка шариковая	Характеристики позиции – на усмотрение образовательной организации	1	шт.	10
2	Бумага	Характеристики позиции –	1	уп.	1

		на усмотрение образовательной организации			
3	Картридж	Характеристики позиции – на усмотрение образовательной организации	1	шт.	1
Оснащение средствами, обеспечивающими охрану труда и технику безопасности					
1	Огнетушитель	ОУ-1	1	шт.	1
2	Аптечка первой помощи	Для сотрудников	1	шт.	1

Критерии оценки выполненного задания на экзамене

Комплект оценочной документации (КОД) 09.02.07-2-2024 разработан в целях организации и проведения демонстрационного экзамена по специальности 09.02.07 Информационные системы и программирование и рассчитан на выполнение заданий продолжительностью 2 часа 30 минут.

Содержательная структура КОД представлена в таблице 1.

Таблица 1 - Требования к содержанию

Вид деятельности (вид профессиональной деятельности)	Перечень оцениваемых ОК, ПК	Перечень оцениваемых умений, навыков (практического опыта)
Разработка модулей программного обеспечения для компьютерных систем	ПК: Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием	Умение: формировать алгоритмы разработки программных модулей в соответствии с техническим заданием
		Умение: оформлять документацию на программные средства
		Практический опыт: разрабатывать алгоритм решения поставленной задачи и реализовывать его средствами автоматизированного проектирования
	ПК: Разрабатывать программные модули в соответствии с техническим заданием	Умение: создавать программу по разработанному алгоритму как отдельный модуль
		Практический опыт: разрабатывать код программного продукта на основе готовой спецификации на уровне модуля
	ПК: Выполнять отладку программных модулей с использованием специализированных программных средств	Умение выполнять отладку и тестирование программы на уровне модуля
		Практический опыт: использовать инструментальные средства на этапе отладки программного продукта
	ПК: Выполнять тестирование программных модулей	Умение: оформлять документацию на программные средства
		Практический опыт: проводить тестирование программного модуля

		по определенному сценарию
		Практический опыт: использовать инструментальные средства на этапе тестирования программного продукта
Разработка, администрирование и защита баз данных	ПК: Проектировать базу данных на основе анализа предметной области	Умение: работать с современными case-средствами проектирования баз данных. Создавать объекты баз данных в современных СУБД
	ПК: Разрабатывать объекты базы данных в соответствии с результатами анализа предметной области	Практический опыт: работать с объектами баз данных в конкретной системе управления базами данных

Общее максимально возможное количество баллов задания по всем критериям оценки составляет 50 (таблица 2).

Таблица 2 - Распределение баллов по критериям оценивания

№ п/п	Модуль задания (вид деятельности, вид профессиональной деятельности)	Критерий оценивания	Баллы
1	Разработка модулей программного обеспечения для компьютерных систем	Формирование алгоритмов разработки программных модулей в соответствии с техническим заданием	12,00
		Разработка программных модулей в соответствии с техническим заданием	10,00
		Выполнение отладки программных модулей с использованием специализированных программных средств	7,00
		Выполнение тестирования программных модулей	9,00
2	Разработка, администрирование и защита баз данных	Проектирование баз данных на основе анализа предметной области	6,00
		Разработка объектов базы данных в соответствии с результатами анализа предметной области	6,00
		Итого	**50,00**

Максимальное количество баллов, которое возможно получить за выполнение задания демонстрационного экзамена, принимается за 100%. Перевод баллов в оценку осуществляется на основе таблицы 3.

Таблица 3

Схема перевода результатов демонстрационного экзамена из стобалльной шкалы в пятибалльную

Оценка	**«2»**	**«3»**	**«4»**	**«5»**

(пятибалльная шкала)				
Оценка в баллах (стобальная шкала)	0,00 – 19,99	20,00 – 39,99	40,00 – 69,99	70,00 - 100,00

1.2 Структурные элементы демонстрационного экзамена

Демонстрационный экзамен организован по модульному принципу. Содержанием экзамена являются виды деятельности программиста. Участники экзамена получают алгоритм выполнения задания с описанием цели выполнения модуля и планируемыми результатами представления задания.

Модули задания и необходимое время

Минимальный КОД демонстрационного экзамена содержит задания по двум модулям:

Модуль 1. Разработка модулей программного обеспечения для компьютерных систем.

Модуль 2. Разработка, администрирование и защита баз данных.

Наименования модуля	Рабочее время	Время на задание
Модуль 1. Разработка модулей программного обеспечения для компьютерных систем.	09.00-10.30	1 час 30 мин.
Модуль 2. Разработка, администрирование и защита баз данных	10.30-11.30	1 час 00 мин.

Модуль 1. Разработка модулей программного обеспечения для компьютерных систем.

Задание модуля 1:

Проанализировать техническое задание, составить краткую спецификацию разрабатываемого модуля выделить входные и выходные данные; сформировать основной алгоритм решения учета заявок на ремонт оборудования в виде блок-схемы в соответствии с техническим заданием. Детализировать в виде алгоритма одну из функций (расчета количества выполненных заявок; расчета среднего времени выполнения заявки).

Алгоритмы представить одним из способов:

− Алгоритм в виде блок-схемы выполнить по правилам, установленным ГОСТ 19.701.

− Алгоритм в виде таблиц выполнить по правилам, установленным ГОСТ 2.105.

− Алгоритм в виде текстового описания выполнить по правилам, установленным ГОСТ 24.301.

Разработать интерфейс программного модуля по составленному алгоритму в среде разработки в соответствии технического задания. Реализовать последовательности алгоритма по этапам (выходные данные должны соответствовать алгоритму, обрабатывающему входные данные). Реализовать алгоритм с использованием всех необходимых данных. В качестве источников данных для реализации алгоритмов используйте динамические списки или массивы в вашем коде, если не реализовывается БД.

Для работы с разными сущностями используйте разные формы, где это уместно.

Все компоненты системы должны иметь единый согласованный внешний вид, соответствующий руководству по стилю, а также следующим требованиям:

− последовательный пользовательский интерфейс, позволяющий перемещаться между существующими окнами в приложении (в том числе обратно, например, с помощью кнопки «Назад»);

− соответствующий заголовок на каждом окне приложения. Выполнить исходный код модуля в соответствии гайдлайну: идентификаторы должны соответствовать

соглашению об именовании, например (CodeConvention), стилю CamelCase (для C# и Java), snake_case (для Python) и https://its.1c.ru/db/v8std#browse:13:-1:31 (для 1С). Допустимо использование не более одной команды в строке. Необходимо использовать комментарии для пояснения неочевидных фрагментов кода. Запрещено комментирование кода. Хороший код воспринимается как обычный текст. Не используйте комментарии для пояснения очевидных действий. Комментарии должны присутствовать только в местах, которые требуют дополнительного пояснения.

Реализовать программные обработки исключительных ситуаций в приложении. Уведомляйте пользователя о совершаемых им ошибках или о запрещенных в рамках задания действиях, запрашивайте подтверждение перед удалением, предупреждайте о неотвратимых операциях, информируйте об отсутствии результатов поиска и т.п. Окна сообщений соответствующих типов (например, ошибка, предупреждение, информация) должны отображаться с соответствующим заголовком и пиктограммой. Текст сообщения должен быть полезным и информативным, содержать полную информацию о совершенных ошибках пользователя и порядок действий для их исправления. Также можно использовать визуальные подсказки для пользователя при вводе данных.

Выполнить отладку модуля.

Выполнить отладку программного обеспечения с использованием инструментальных средств. Сохранить и представить результаты в скриншотах.

Определить наборы входных данных и выполнить функциональное тестирование модуля по определенному сценарию. Провести тестирование для проверки функциональности программы (хотя бы 1 тест на 1 функцию). Использовать инструментальные средства для тестирования. Представить результаты тестирования в виде протокола тестирования, в соответствии со стандартами.

Время на выполнение задания: 90 минут

Модуль 2. Разработка, администрирование и защита баз данных

Задание модуля 2:

На основе задания демонстрационного экзамена Вам необходимо спроектировать ER-диаграмму для учета заявок на ремонт оборудования. Обязательна 3 нормальная форма с обеспечением ссылочной целостности. При разработке диаграммы обратите внимание на согласованную осмысленную схему именования, создайте необходимые первичные и внешние ключи, определите ограничения внешних ключей, отражающие характер предметной области.

ER - диаграмма должна быть представлена в формате удобном для просмотра и содержать таблицы, связи между ними, атрибуты и ключи (типами данных на данном этапе можно пренебречь) проведение анализа поставленной задачи и проектирования базы данных (ERD модели) с применением case-средств;

Создайте все необходимые сущности, определите отношения, создайте ограничения на связи между сущностями (при наличии всех связей), приведите базу данных к 3НФ (при наличии всех сущностей и связей).

Создайте базу данных, используя предпочтительную платформу, на сервере баз данных, которую Вам предоставили. Создайте таблицы основных сущностей, атрибуты, отношения и необходимые ограничения.

Выполните названия таблиц и полей в едином стиле, согласно отраслевой документации.

Заказчик системы предоставил файлы с данными (с пометкой import в ресурсах) для переноса в новую систему. Заполните базу данных. Создайте запросы к базе данных и сформируйте отчеты с выводом необходимых данных в соответствии с заданием.

Выполните резервное копирование БД, сохраните полученные результаты.

Выберите принцип регистрации пользователей в системе учета заявок на ремонт оборудования в соответствии с функциональными обязанностями.

Создайте группы пользователей. Выполните реализацию уровней доступа для различных категорий пользователей.

1. привлекать других специалистов к выполнению ремонта;
2. продлевать срок выполнения заявки с согласованием клиента.

Время на выполнение задания: 60 минут

Глава 2 Методические рекомендации по подготовке к демонстрационному

экзамену

2.1 Модуль 1: Разработка модулей программного обеспечения для компьютерных систем

Техническое задание представляет собой документ, в котором формулируются основные цели разработки, требования к программному продукту, определяются сроки и этапы разработки и регламентируется процесс приемно-сдаточных испытаний. В формулировании технического задания участвуют представители как заказчика, так и исполнителя. В основе этого документа лежат исходные требования, заказчика, результаты выполнения предпроектных исследований и т. п.

Разработка технического задания выполняется в такой последовательности:

1) устанавливают набор выполняемых функций, а также перечень и характеристики исходных данных;

2) определяют перечень результатов, их характеристики и способы их представления,

3) уточняют среду функционирования программного обеспечения: конкретную комплектацию и параметры технических средств, версию используемой операционной системы и, возможно, версии и параметры другого установленного программного обеспечения, с которым предстоит взаимодействовать будущему программному продукту.

Описание предметной области

Основная цель учёта заявок на ремонт оборудования - эффективное и оперативное осуществление ремонтных работ с минимизацией простоев и удовлетворением запросов клиентов или сотрудников. Эта предметная область широко используется в различных сферах деятельности, таких как сервисные услуги, производство, информационные технологии и другие.

Предметная область учёта заявок на ремонт оборудования касается процесса подачи, обработки и учёта заявок на ремонт различного оборудования.

В данной области включены следующие основные составляющие:

1. Заявка на ремонт: это информация, предоставленная клиентом или сотрудником о неисправности оборудования, которое требует ремонта. Заявка может содержать данные о типе оборудования, его серийном номере, описании проблемы и другой важной информации.

2. Регистрация заявки: этот процесс включает приём и регистрацию заявки в системе учёта. Важными аспектами регистрации являются присвоение уникального идентификатора заявке, сохранение информации о заявке и её приоритете.

3. Обработка заявки: процесс, включающий анализ заявки, определение её приоритетности и назначение исполнителя (ремонтного специалиста) для задачи. В процессе обработки может потребоваться дополнительная информация или уточнение деталей проблемы у клиента или сотрудника.

4. Исполнение заявки: фактическое выполнение ремонта оборудования. В этом этапе назначенный исполнитель ремонтирует оборудование, вносит необходимые изменения или заменяет неисправные компоненты. Важно отметить, что на этом этапе могут возникать необходимость заказа запчастей или координации работ с другими специалистами.

5. Отчётность и информирование: важной составляющей учёта заявок на ремонт является фиксация и отчёт о выполненной работе. После завершения ремонта, исполнитель должен предоставить отчёт о проделанной работе, включая информацию о затраченных ресурсах (время, материалы, стоимость), причине неисправности и оказанной помощи.

6. Мониторинг и анализ: этот этап предполагает контроль и анализ процесса учёта заявок на ремонт. Важно отслеживать и анализировать время обработки заявок, качество

выполненных работ, расходы и прочие параметры, которые могут помочь в оптимизации и улучшении процесса

Техническое задание

1. Общие сведения
1.1. Наименование проекта: Разработка программного модуля для учета заявок на ремонт оборудования.
1.2. Заказчик: ООО "Техносервис".
1.3. Исполнитель: Компания "IT-Решения".

2. Функциональные требования
2.1. Возможность добавления заявок в базу данных с указанием следующих параметров:

- Номер заявки;
- Дата добавления;
- Оборудование, которое требует ремонта;
- Тип неисправности;
- Описание проблемы;
- Клиент, который подал заявку;
- Статус заявки (в ожидании, в работе, выполнено).

2.2. Возможность редактирования заявок:
- Изменение этапа выполнения (выполнено, в работе, не выполнено);
- Изменение описания проблемы;
- Изменение, ответственного за выполнение работ.

2.3. Возможность отслеживания статуса заявки:
- Отображение списка заявок;
- Получение уведомлений о смене статуса заявки;
- Поиск заявки по номеру или по параметрам.

2.4. Возможность назначения ответственных за выполнение работ:
- Добавление исполнителя к заявке;
- Отслеживание состояния работы и получение уведомлений о ее завершении;
- Исполнитель может добавлять комментарии на форме заявки.

2.5. Расчет статистики работы отдела обслуживания:
- Количество выполненных заявок;
- Среднее время выполнения заявки;
- Статистика по типам неисправностей.

3. Нефункциональные требования
3.1. Кроссплатформенность:
- Поддержка работы на ОС семейства Windows.

3.2. Безопасность:
- Логин и пароль для доступа к приложению;
- Доступ к данным должен быть ограничен в зависимости от роли пользователя.

3.3. Удобство использования:
- Простой и интуитивный интерфейс;
- Информативные уведомления и подсказки.

3.4. Производительность:
- Приложение должно иметь быстрый доступ к данным;
- Минимальное время отклика на запросы пользователя.

4. Требования к реализации
4.1. Язык программирования: на усмотрение разработчика
4.2. СУБД: на усмотрение разработчика

5. Требования к документации
5.1. Техническое задание на разработку программного модуля.
5.2. Руководство системному программисту.

Алгоритм – четкое описание последовательности действий, которые необходимо выполнить при решении задачи.

Разработка алгоритма решения задачи – это разбиение задачи на последовательно выполняемые этапы, причем результаты выполнения предыдущих этапов могут использоваться при выполнении последующих. При этом должны быть четко указаны как содержание каждого этапа, так и порядок выполнения этапов

Под интерфейсом в программной инженерии подразумевается связь между компонентами программной системы, целью которой является описание способа обмена информацией. Использование интерфейсов позволяет сосредоточиться на том, *что* делает определенный модуль, и абстрагироваться от особенностей того, *как* он это делает.

Интерфейс — ключевое понятие в задаче интеграции. При разработке любой программы непременно приходится использовать «чужие» компоненты, такие как встроенная библиотека типов и стандартных функций или сторонние библиотеки. Более сложные приложения могут сами состоять из многих компонентов, которые надо интегрировать между собой; зачастую эта задача усложняется тем, что взаимодействующие модули написаны на различных языках программирования и функционируют в различных операционных средах.

Все компоненты интерфейса системы должны иметь единый согласованный внешний вид, соответствующий руководству по стилю, а также следующим требованиям:

− последовательный пользовательский интерфейс, позволяющий перемещаться между существующими окнами в приложении (в том числе обратно, например, с помощью кнопки «Назад»);

− соответствующий заголовок на каждом окне приложения. Выполнить исходный код модуля в соответствии гайдлайну: идентификаторы должны соответствовать соглашению об именовании, например (CodeConvention), стилю CamelCase (для C# и Java), snake_case (для Python) и https://its.1c.ru/db/v8std#browse:13:-1:31 (для 1С). Допустимо использование не более одной команды в строке. Необходимо использовать комментарии для пояснения неочевидных фрагментов кода. Запрещено комментирование кода. Хороший код воспринимается как обычный текст. Не используйте комментарии для пояснения очевидных действий. Комментарии должны присутствовать только в местах, которые требуют дополнительного пояснения.

Реализация программной обработки исключительных ситуаций в приложении. Уведомляйте пользователя о совершаемых им ошибках или о запрещенных в рамках задания действиях, запрашивайте подтверждение перед удалением, предупреждайте о неотвратимых операциях, информируйте об отсутствии результатов поиска и т.п. Окна сообщений соответствующих типов (например, ошибка, предупреждение, информация) должны отображаться с соответствующим заголовком и пиктограммой. Текст сообщения должен быть полезным и информативным, содержать полную информацию о совершенных ошибках пользователя и порядок действий для их исправления. Также

можно использовать визуальные подсказки для пользователя при вводе данных.

Выполнение отладки программного обеспечения с использованием инструментальных средств может выполняться при помощи нижеперечисленного программного обеспечения:

Отладчики – это инструменты, которые позволяют программистам искать и исправлять ошибки в программном коде. Они обеспечивают возможность пошагового выполнения программы, просмотра значений переменных, а также отслеживания вызовов функций и обработки исключений. Некоторые популярные отладчики включают в себя:

-GDB – отладчик для языка программирования C и C++. Он позволяет выполнять программу пошагово, устанавливать точки останова и просматривать значения переменных.

-Visual Studio Debugger – отладчик, встроенный в среду разработки Visual Studio. Он предоставляет широкий набор функций для отладки программ на различных языках программирования.

-PyCharm Debugger – отладчик для языка программирования Python, встроенный в среду разработки PyCharm. Он позволяет отслеживать выполнение программы, просматривать значения переменных и выполнять другие отладочные операции.

Не забудьте сохранить и представить результаты в скриншотах.

Выполнение функциональное тестирование модуля по определенному сценарию возможно с использованием одного из инструментальных средств, таких как:

Автоматическое тестирование – это метод тестирования программы, при котором тестовые сценарии запускаются автоматически с использованием специальных инструментов. Некоторые популярные инструменты для автоматического тестирования включают в себя:

-Selenium – инструмент для автоматизации тестирования веб-приложений. Он позволяет записывать и воспроизводить действия пользователя на веб-страницах.

-JUnit и TestNG – фреймворки для автоматического тестирования программ на языке Java. Они предоставляют возможность создания и запуска тестовых сценариев, а также управления их выполнением и отчетностью.

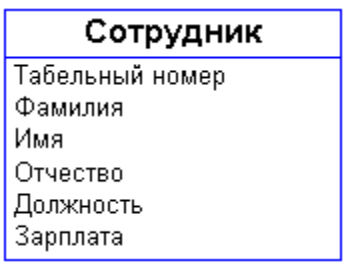
-PyTest – фреймворк для автоматического тестирования программ на языке Python. Он обеспечивает простой и удобный способ написания и запуска тестовых сценариев, а также генерацию отчетов о выполнении тестов.

Модуль 2. Разработка, администрирование и защита баз данных

ER-диаграмма (диаграмма "сущность-связь") — это графическое представление данных и их взаимосвязей, используемое в процессе проектирования баз данных. Основные компоненты ER-диаграммы включают:

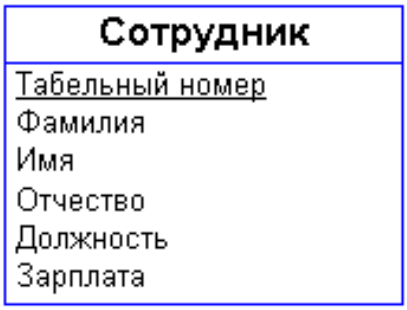
Сущности (Entities): объекты или понятия, которые существуют в системе и о которых нужно хранить информацию (например, товары, заказы, клиенты).

Атрибуты (Attributes): свойства или характеристики сущностей (например, название товара, цена, дата заказа). Атрибуты изображаются в пределах прямоугольника, определяющего сущность:



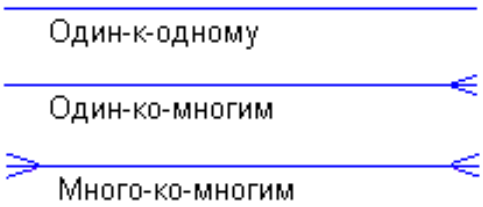
Связи (Relationships): логические ассоциации между сущностями (например, заказ включает товары, клиент делает заказ). Графически связь изображается линией, соединяющей две сущности:

Ключ сущности - это *неизбыточный* набор атрибутов, значения которых в совокупности являются *уникальными* для каждого экземпляра сущности. Неизбыточность заключается в том, что удаление любого атрибута из ключа нарушается его уникальность. Сущность может иметь несколько различных ключей. Ключевые атрибуты изображаются на диаграмме подчеркиванием:



Каждая связь имеет два конца и одно или два наименования. Наименование обычно выражается в неопределенной глагольной форме: "иметь", "принадлежать" и т.п. Каждое из наименований относится к своему концу связи. Иногда наименования не пишутся ввиду их очевидности.

Каждая связь может иметь один из следующих ***типов связи***:

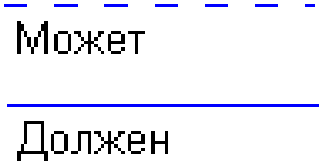


язь типа ***один-к-одному*** означает, что один экземпляр первой сущности (левой) связан с одним экземпляром второй сущности (правой). Связь один-к-одному чаще всего свидетельствует о том, что на самом деле мы имеем всего одну сущность, неправильно разделенную на две.

Связь типа ***один-ко-многим*** означает, что один экземпляр первой сущности (левой) связан с несколькими экземплярами второй сущности (правой). Это наиболее часто используемый тип связи. Левая сущность (со стороны "один") называется ***родительской***, правая (со стороны "много") - ***дочерней***. Характерный пример такой связи приведен на Рис. 4.

Связь типа ***много-ко-многим*** означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и каждый экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности. Тип связи много-ко-многим является *временным* типом связи, допустимым на ранних этапах разработки модели. В дальнейшем этот тип связи должен быть заменен двумя связями типа один-ко-многим путем создания промежуточной сущности.

Каждая связь может иметь одну из двух ***модальностей связи***:



При разработке ER-моделей мы должны получить следующую информацию о предметной области:

1. Список сущностей предметной области.
2. Список атрибутов сущностей.
3. Описание взаимосвязей между сущностями.

Нормализация представляет собой процесс реорганизации данных путем ликвидации повторяющихся групп и иных противоречий в хранении данных с целью приведения таблиц к виду, позволяющему осуществлять непротиворечивое и корректное редактирование данных.

Теория нормализации основана на концепции нормальных форм. Говорят, что таблица находится в данной нормальной форме, если она удовлетворяет определенному набору требований. Теоретически существует пять нормальных форм, но на практике обычно используются только первые три. Более того, первые две нормальные формы являются по существу промежуточными шагами для приведения базы данных к третьей нормальной форме.

Первая нормальная форма

Чтобы таблица соответствовала первой нормальной форме, все значения ее полей должны быть атомарными, и все записи — уникальными. Поэтому любая реляционная таблица, в том числе и таблица OrderedProducts, по определению, уже находится в первой нормальной форме.

Вторая нормальная форма

Говорят, что реляционная таблица находится во второй нормальной форме, если она находится в первой нормальной форме и ее неключевые поля полностью зависят от всего первичного ключа.

Третья нормальная форма

Говорят, что реляционная таблица находится в третьей нормальной форме, если она находится во второй нормальной форме и все ее неключевые поля зависят только от первичного ключа.

Чтобы перейти от второй нормальной формы к третьей, нужно выполнить следующие шаги:

Определить все поля (или группы полей), от которых зависят другие поля.

Создать новую таблицу для каждого такого поля (или группы полей) и группы зависящих от него полей и переместить их в эту таблицу. Поле (или группа полей), от которого зависят все остальные перемещенные поля, станет при этом первичным ключом новой таблицы.

Удалить перемещенные поля из исходной таблицы, оставив лишь те из них, которые станут внешними ключами.

ER - диаграмма должна быть представлена в формате удобном для просмотра и содержать таблицы, связи между ними, атрибуты и ключи (типами данных на данном этапе можно пренебречь) проведение анализа поставленной задачи и проектирования базы данных (ERD модели) с применением case-средств;

Использование среды SQL Server Management Studio

Создание базы данных

В **обозревателе объектов**подключитесь к экземпляру компонента SQL Server

Database Engine и разверните его.

Щелкните правой кнопкой мыши узел **Базы данных** и выберите команду **Создать базу данных**.

В поле **Новая база данных**введите имя базы данных.

Чтобы создать базу данных, приняв все значения по умолчанию, нажмите кнопку **ОК**; в противном случае продолжайте выполнять указанные ниже дополнительные действия.

Чтобы изменить имя владельца, нажмите (**…**) и выберите другого владельца.

Чтобы изменить значения первичных данных по умолчанию и файлы журнала транзакций, выберите соответствующую ячейку в сетке **Файлы базы данных** и введите новое значение. Дополнительные сведения см. в статье АДобавление файлов данных или журналов в базу данных.

Чтобы изменить параметры сортировки базы данных, выберите страницу **Параметры** и выберите из списка желаемые параметры сортировки.

Чтобы изменить модель восстановления, выберите страницу **Параметры** и модель восстановления из списка.

Чтобы изменить параметры базы данных, выберите страницу **Параметры** и измените параметры базы данных. Описание каждого параметра см. в разделе ALTER DATABASE SET Options (Transact-SQL).

Чтобы добавить новую файловую группу, перейдите на страницу **Файловые группы**. Нажмите кнопку **Добавить** и введите значения для файловой группы.

Чтобы добавить расширенное свойство в базу данных, выберите страницу **Расширенные свойства** .

Основные этапы копирования базы данных, используя функции резервного копирования и восстановления

При использовании резервного копирования и восстановления для копирования базы данных на другой экземпляр SQL Server компьютер-источник и целевой компьютер могут быть любой платформой, на которой запускается SQL Server.

Основные этапы.

Создайте резервную копию исходной базы данных, которая может находиться в экземпляре SQL Server 2005 (9.x) или более поздней версии. Компьютер, на котором выполняется этот экземпляр SQL Server, является исходным компьютером.

На компьютере, куда нужно скопировать базу данных (**целевой компьютер**), подключите экземпляр SQL Server, на котором будет восстановлена база данных. При необходимости создайте те же устройства резервного копирования на **целевом** экземпляре сервера, что использовались для резервного копирования баз данных- **источников** .

Восстановите резервную копию базы данных- **источника** на **целевом** компьютере. При восстановлении базы данных автоматически создаются все ее файлы.

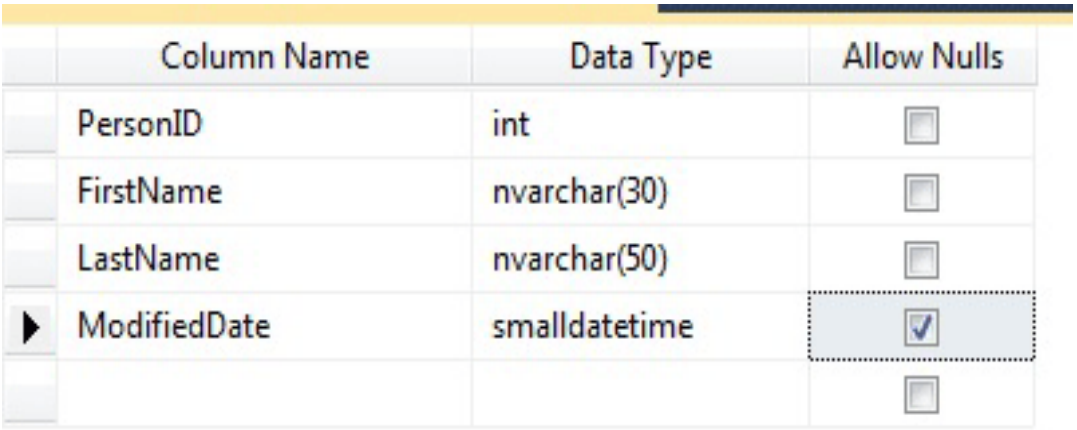
Использование конструктора таблиц в SQL Server Management Studio

В SSMS в **обозревателе объектов**подключитесь к экземпляру Компонент Database Engine , который содержит изменяемую базу данных.

В **обозревателе объектов**разверните узел **Базы данных** , а затем базу данных, в которой будет размещена новая таблица.

В обозреватель объектов щелкните правой кнопкой мыши узел **Таблицы** базы данных и выберите **Создать таблицу**.

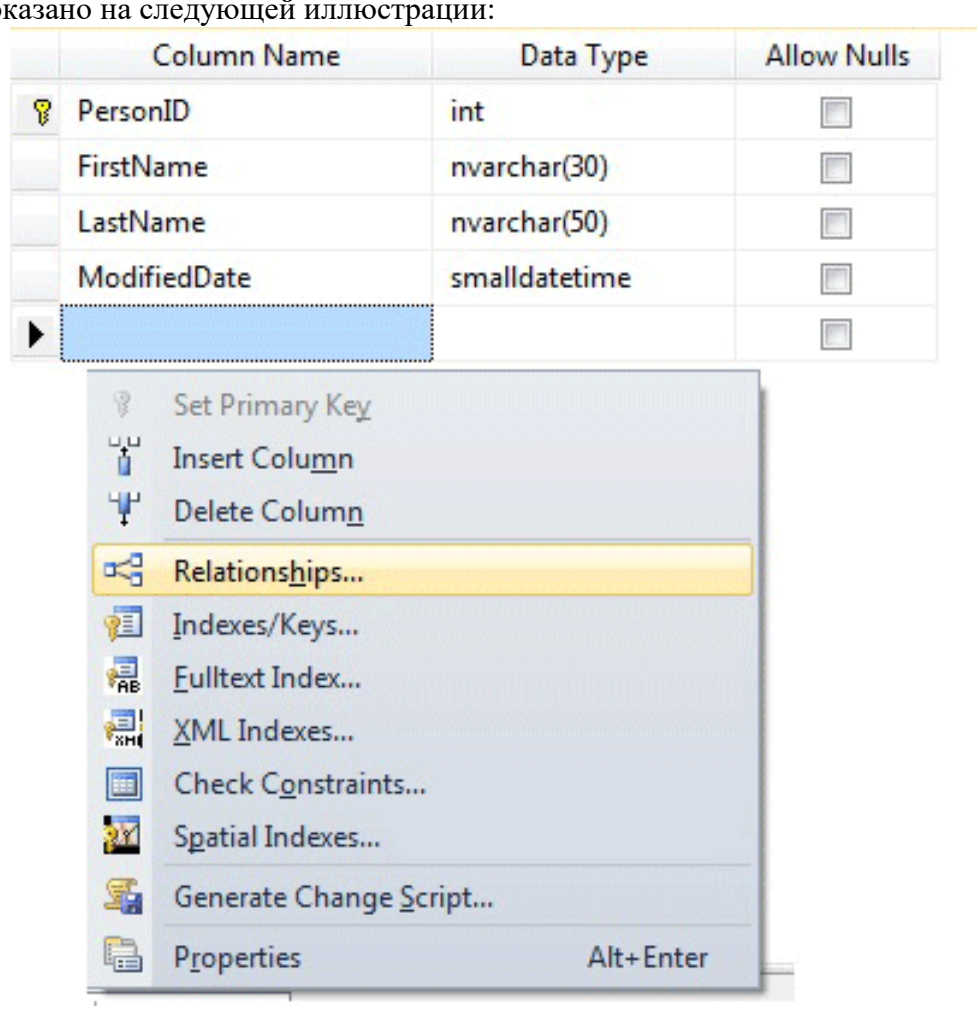
Введите имена столбцов, выберите типы данных и определите для каждого столбца, могут ли в нем присутствовать значения NULL, как показано на следующей иллюстрации:



Чтобы указать дополнительные свойства столбца, например идентификатор или вычисляемые значения столбца, выберите столбец и на вкладке свойства столбца выберите соответствующие свойства. Дополнительные сведения о свойствах столбца см. в разделе Свойства столбца таблицы (SQL Server Management Studio).

Чтобы указать, что столбец является столбцом первичного ключа, щелкните его правой кнопкой мыши и выберите **Задать первичный ключ**. Дополнительные сведения см. в статье Create Primary Keys.

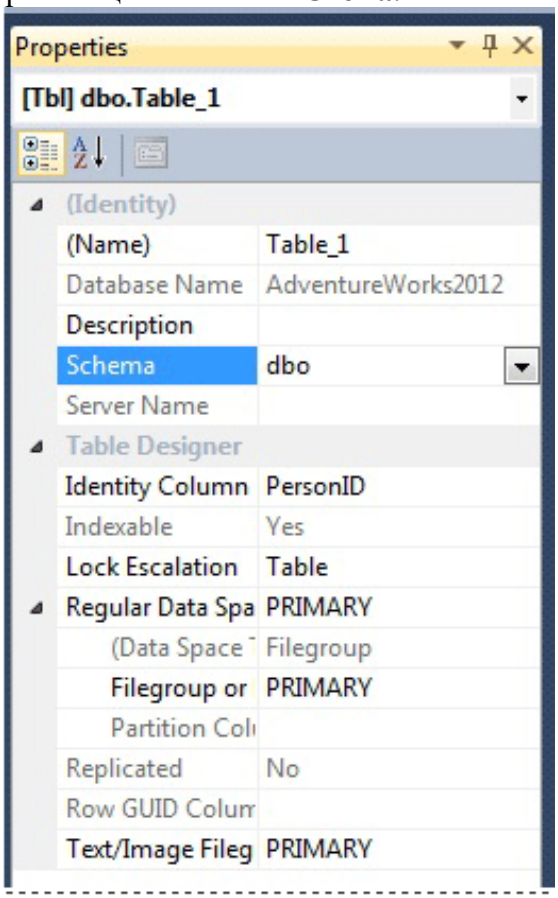
Чтобы создать связи по внешнему ключу, проверочные ограничения или индексы, щелкните правой кнопкой мыши панель конструктора таблиц и выберите в списке объект, как показано на следующей иллюстрации:



Дополнительные сведения об этих объектах см. в разделах Create Foreign Key Relationships, Create Check Constraints и Indexes.

По умолчанию таблица содержится в схеме **dbo** . Чтобы указать другую схему для

таблицы, щелкните правой кнопкой мыши панель конструктора таблиц и выберите **Свойства** , как показано на следующей иллюстрации. Выберите нужную схему из раскрывающегося списка **Схема**.



Дополнительные сведения о схемах см. в разделе Create a Database Schema.

В меню **Файл** выберите **Сохранить***имя таблицы*.

В диалоговом окне **Выбор имени** введите имя таблицы и нажмите **кнопку ОК**.

Чтобы просмотреть новую таблицу, в **обозревателе объектов**разверните узел **Таблицы**, а затем нажмите клавишу **F5** , чтобы обновить список объектов. Новая таблица будет отображена в списке таблиц.

Операции в реляционной базе данных выполняются над множеством строк. Например, набор строк, возвращаемый инструкцией SELECT, содержит все строки, которые удовлетворяют условиям, указанным в предложении WHERE инструкции. Этот полный набор строк, возвращаемых инструкцией, называется *результирующий набор*.

Внутренние соединения можно задавать в предложениях FROM и WHERE. **Внешние соединения** и **перекрестные соединения** можно задавать только в предложении FROM. Условия соединения сочетаются с условиями поиска WHERE и HAVING для управления строками, выбранными из базовых таблиц, на которые ссылается предложение FROM.

Чтобы легко управлять разрешениями в базах данных, SQL Server предоставляет несколько ролей, которые являются субъектами безопасности, которые группируют другие субъекты. Они похожи на **группы** в операционной системе Microsoft Windows. Разрешения ролей уровня базы данных распространяются на всю базу данных.

Чтобы добавлять и удалять пользователей в роли базы данных, используйте параметры ADD MEMBER и DROP MEMBER инструкции ALTER ROLE . Система платформы аналитики (PDW) и Azure Synapse Analytics не поддерживают использование ALTER ROLE. Используйте вместо этого более старые

процедуры sp_addrolemember и sp_droprolemember .

Существует два типа ролей уровня базы данных: *предопределенные роли базы данных* , являющиеся стандартными для базы данных, и *пользовательские роли базы данных* , которые можно создавать.

Предопределенные роли базы данных задаются на уровне базы данных и предусмотрены в каждой базе данных. Члены ролей базы данных **db_owner** могут управлять членством в предопределенных ролях базы данных. Кроме того, в базе данных msdb имеются специальные роли базы данных.

Вы можете добавить любую учетную запись базы данных и другие роли SQL Server в роли уровня базы данных.