Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Гаджибутаева Султанага Рамазановна
Должность: Директор
Дата подписания: 09.06.2024 12:40:33
Уникальный программный ключ:
2b71376f78d52b66ab183b5be5a3b5fe443c04a8

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РЕСПУБЛИКИ ДАГЕСТАН

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

УТВЕРЖДАЮ
Зам. директора по учебно-
методической работе
_____ / Кадрышева Ж.А
«_03_» июля 2023 г.

Комплект контрольно-оценочных средств

по учебной дисциплине

ОП.13 ПРОГРАММНЫЕ РЕШЕНИЯ ДЛЯ БИЗНЕСА

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

Год набора: 2023

Кизляр
2023г.

ОДОБРЕН
на заседании цикловой методической комиссии общепрофессиональных дисциплин и профессиональных модулей по специальности 09.02.07 Информационные системы и программирование
Протокол № 10 от «28» июня 2023 г.

Председатель ЦМК _____ /
Кадрышева Ж.А.

Составлен в соответствии с требованиями федерального государственного образовательного стандарта по специальности 09.02.07 Информационные системы и программирование и рабочей программы по дисциплине ОП.13 Программные решения для бизнеса

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

Разработчик(и):

Магомедова Мадина Нурмагомедовна, преподаватель
Ф.И.О., ученая степень, звание, должность

СОДЕРЖАНИЕ

1. ПАСПОРТ
комплекта КОС по учебной дисциплине ОП.13 Программные решения для бизнеса
1.1. Общие положения

Контрольно-оценочные средства (КОС) предназначены для контроля и оценки образовательных достижений обучающихся, освоивших программу учебной дисциплины ОП.13 Программные решения для бизнеса.

КОС включает контрольные материалы для проведения текущего контроля и промежуточной аттестации в форме дифференцированного зачета;

КОС разработаны в соответствии с:

- образовательной программой СПО по специальности 09.02.07 Информационные системы и программирование;
- программой учебной дисциплины ОП.13 Программные решения для бизнеса.

1.2. Результаты освоения дисциплины, подлежащие проверке

Результаты обучения (освоенные умения, усвоенные знания)	**Наименование элемента умений/знаний**
У1	Применять методики анализа деятельности пользователей;
У2	Владеть приемами проектирования архитектуры информационной системы;
У3	Владеть подходами к описанию и демонстрации результатов своей работы;
З1	Методики анализа деятельности пользователя;
З2	Приемы проектирования архитектуры информационных систем;
З3	Подходы к описанию и демонстрации результатов разработки информационных систем;
ОК 01.	Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.
ОК 02.	Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности.
ОК 03.	Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по финансовой грамотности в различных жизненных ситуациях.
ОК 04.	Эффективно взаимодействовать и работать в коллективе и команде.
ОК 05.	Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.
ОК 09.	Пользоваться профессиональной документацией на государственном и иностранном языках.
ПК 4.1.	Осуществлять инсталляцию, настройку и обслуживание программного обеспечения компьютерных систем.
ПК 11.1.	Осуществлять сбор, обработку и анализ информации для проектирования баз данных.
ПК 11.4.	Реализовывать базу данных в конкретной системе управления базами данных.
ПК 11.5.	Администрировать базы данных.
ПК 11.6.	Защищать информацию в базе данных с использованием

Результаты обучения (освоенные умения, усвоенные знания)	Наименование элемента умений/знаний
	технологии защиты информации.

1.3. Распределение оценивания результатов обучения по видам контроля

Код и наименование элемента умений или знаний	Виды аттестации	
	Текущий контроль	*Промежуточная аттестация*
У1 Применять методики анализа деятельности пользователей;	Практическая работа, тестирование	Дифференцированный зачет
У2 Владеть приемами проектирования архитектуры информационной системы;	Практическая работа, тестирование	Дифференцированный зачет
У3 Владеть подходами к описанию и демонстрации результатов своей работы;	Практическая работа, тестирование	Дифференцированный зачет
З1 Методики анализа деятельности пользователя;	Практическая работа, тестирование	Дифференцированный зачет
З2 Приемы проектирования архитектуры информационных систем;	Практическая работа, тестирование	Дифференцированный зачет
З3 Подходы к описанию и демонстрации результатов разработки информационных систем;	Практическая работа, тестирование	Дифференцированный зачет

1.4 Распределение типов оценочных средств по элементам знаний и умений текущего контроля

Содержание учебного материала по программе УД	Тип контрольного задания					
	У1	**У2**	**У3**	**З1**	**З2**	**З3**
Тема 1.1 Структурный подход в моделировании предметной области	15	15		17		
Тема 1.2 Объектно-ориентированное моделирование системы		15			17	
Тема 1.3 Спецификации языка C#. Технология .NET		15				17
Тема 1.4 Система управления базами данных MySQL и MS SQL Server.		15			17	
Тема 1.5 Тестирование и отладка программных решений	15		15		17	
Тема 1.6 Документирование программных решений	15	15	15	17		17

1.5 Распределение типов оценочных средств по элементам знаний и умений, контролируемых на промежуточной аттестации

Содержание учебного материала по программе УД	Тип контрольного задания					
	У1	**У2**	**У3**	**З1**	**З2**	**З3**
Тема 1.1 Структурный подход в моделировании предметной области	25			25		
Тема 1.2 Объектно-ориентированное моделирование системы		25				
Тема 1.3 Спецификации языка C#. Технология .NET		25			25	
Тема 1.4 Система управления базами данных MySQL и MS SQL Server.		25			25	
Тема 1.5 Тестирование и отладка программных решений			25			25
Тема 1.6 Документирование программных решений			25			25

2. СПЕЦИФИКАЦИИ И ВАРИАНТЫ ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ

2.1. Назначение

Спецификацией устанавливаются требования к содержанию и оформлению вариантов оценочного средства практическая работа и тестирование.

Практическая работа и тестирование предназначены для текущего контроля и оценки знаний и умений студентов по программе учебной дисциплины «Программные решения для бизнеса» основной профессиональной образовательной программы 09.02.07 Информационные системы и программирование

2.2. Контингент аттестуемых: студенты 1 курса

2.3. Форма и условия аттестации: Текущий контроль проходит по темам учебной дисциплины.

2.4. Время выполнения:

На выполнение текущего контроля отводится:

1)Практическая работа:
подготовка 10 мин;
выполнение 1 час 15 мин;
оформление и сдача 5 мин;
всего 1 час 30 мин.

2)Тестирование:
подготовка 5 минут;
выполнение 30 минут;
оформление и сдача 10 минут;
всего 45 минут.

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

Библиографическое описание издания (автор, заглавие, вид, место и год издания, кол. стр.)	**Основная/ дополнительная литература**	**Книгообеспеченность**	
		Кол-во. экз. в библ.	**Электронные ресурсы**
Лаврищева, Е. М. Программная инженерия и технологии программирования сложных систем : учебник для вузов / Е. М. Лаврищева. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2023. — 432 с.	Основная	-	https://urait.ru/bcode/513067
Тузовский, А. Ф. Объектно-ориентированное программирование : учебное пособие для вузов / А. Ф. Тузовский. — Москва : Издательство Юрайт, 2023. — 213 с.	Основная	-	https://urait.ru/bcode/530800
Кудрина, Е. В. Основы алгоритмизации и программирования на языке C# : учебное пособие для среднего профессионального образования / Е. В. Кудрина, М. В. Огнева. — Москва : Издательство Юрайт, 2023. — 322 с.	Основная	-	https://urait.ru/bcode/517324
Маркин, А. В. Программирование на SQL :		-	https://urait.ru/

учебное пособие для среднего профессионального образования / А. В. Маркин. — Москва : Издательство Юрайт, 2023. — 435 с.	Основная		bcode/518166
Проектирование информационных систем : учебник и практикум для среднего профессионального образования / Д. В. Чистов, П. П. Мельников, А. В. Золотарюк, Н. Б. Ничепорук. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2023. — 293 с.	Дополнительная	-	https://urait.ru/bcode/530635
Стружкин, Н. П. Базы данных: проектирование : учебник для среднего профессионального образования / Н. П. Стружкин, В. В. Годин. — Москва : Издательство Юрайт, 2023. — 477 с.	Дополнительная	-	https://urait.ru/bcode/518499
Гниденко, И. Г. Технология разработки программного обеспечения : учебное пособие для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — Москва : Издательство Юрайт, 2023. — 235 с.	Дополнительная		https://urait.ru/bcode/514591

2.6. Перечень материалов, оборудования и информационных источников.

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

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

Лаборатория № 2.2 программного обеспечения и сопровождения компьютерных систем. Компьютерный класс (для проведения практических занятий, с применением вычислительной техники) оборудован мультимедийным комплексом. Специализированная мебель и оборудование:

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

SQLServerManagementStudio, MicrosoftSQLServerJavaConnector, AndroidStudio, IntelliJIDEA.

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

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

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

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

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

Перечень современных профессиональных баз данных (СПБД)

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

Перечень информационных справочных систем (ИСС)

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

2.7. Варианты оценочных средств

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РЕСПУБЛИКИ ДАГЕСТАН
Частное профессиональное образовательное учреждение
«РЕСПУБЛИКАНСКИЙ ПОЛИПРОФЕССИОНАЛЬНЫЙ КОЛЛЕДЖ»
(ЧПОУ «Республиканский полипрофессиональный колледж»)

Лабораторная работа

по дисциплине Программные решения для бизнеса

Лабораторная работа №1 Разработка диаграмм потоков данных для программного решения

Цель работы: приобретение навыков построения диаграмм потоков данных (DFD).

Пример построения диаграммы DFD.

Как уже говорилось ранее, диаграммы потоков данных являются основным средством моделирования функциональных требований к проектируемой системе.

Функциональные требования к системе определяют, действия системы, которые она должна выполнять.

В качестве примера процесса, для которого будет строиться диаграмма DFD, как и в случае с IDEF0, будет взят процесс написания курсовой работы (КР).

Пусть системой будут пользоваться студент и преподаватель.

Тогда можно выделись следующие функциональные требования, связанные с написанием курсовой работы:

1. Система должна хранить переработанные фрагменты информации, а также ссылки на литературу, на основе которой они были сделаны.

2. Система должна хранить замечания преподавателя по КР.

Далее нужно рассмотреть требования к источникам данных, связанные с функциональными требованиями, то есть определить, какие действия над базой данных нужно делать для выполнения требований, какие процессы изменяют базу данных и взаимодействуют с внешними объектами.

Для именования внешних сущностей и хранилищ были созданы следующие классификаторы:

1. студент,
2. преподаватель,
3. литература,
4. переработанный материал,
5. замечания преподавателя.

Диаграмма DFD строилась «с нуля» с использованием декомпозиции. На контекстной диаграмме отображен основной процесс «Работать над КР», а также все внешние сущности и хранилища. Обмен информацией происходит по требованиям к источникам данных, но в обобщенном виде.

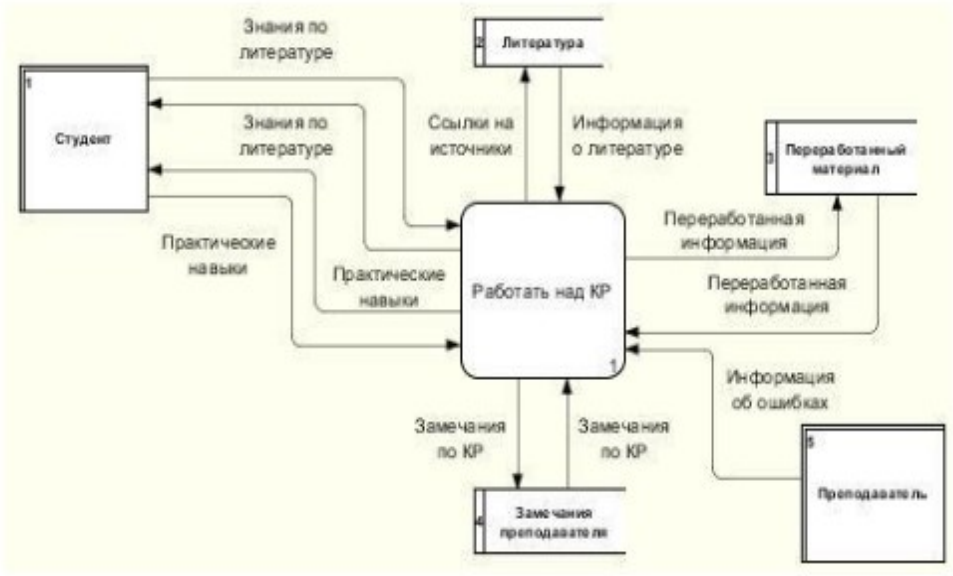


Рисунок 7 – Контекстная диаграмма процесса «Работать над КР»

Далее процесс «Работать над КР» был детализирован на три подпроцесса, которые фигурируют в требованиях к источникам данных: проанализировать литературу, написать КР, проверить КР.

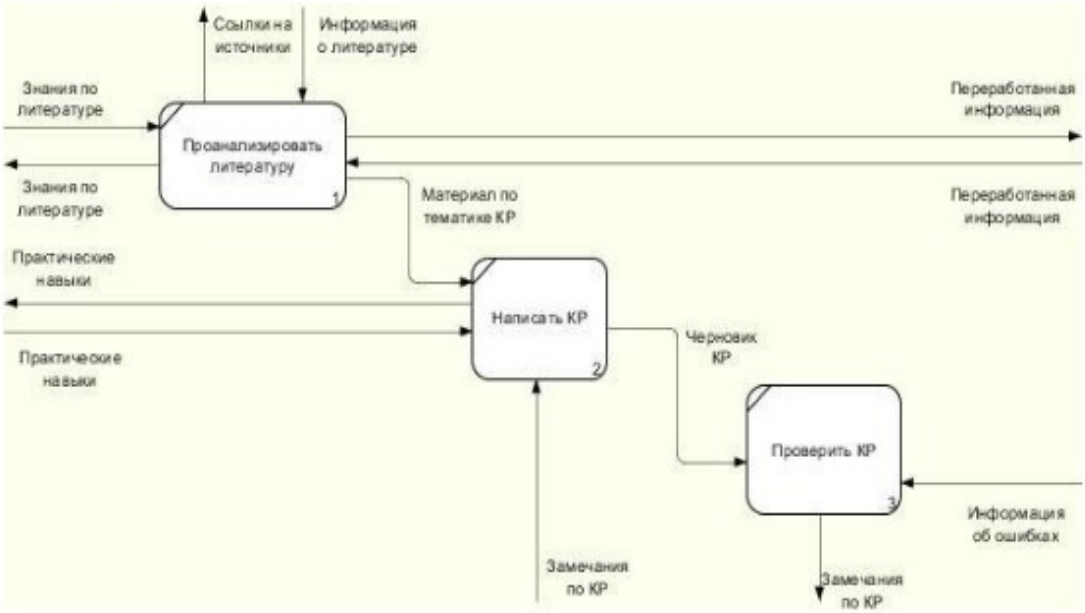


Рисунок 8 – Детализация процесса «Работать над КР»

Обмен информацией на схеме происходит по требованиям к источником данных, но с дополнениями: поток данных «Материал по тематике КР» является результатом анализа литературы и входными данными для процесса «Написать КР», а поток данных «Черновик КР» является результатом написания КР и входными данными для процесса проверки КР.

Контрольные вопросы

1. Назначение методологии DFD.
2. Какие компоненты входят в состав диаграмм?
3. Правила построения диаграмм потоков данных.

Лабораторная работа №2 Разработка диаграммы «сущность связь» для программного решения.

Цель работы: ознакомиться с работой программы Erwin, на примере создания логической модели данных.

Создание логической модели данных.

Различают три уровня логической модели, отличающихся 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).

Диаграмма сущность-связь представляет собой модель данных верхнего уровня. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Такая диаграмма не слишком детализирована, в нее включаются основные сущности и связи между ними, которые удовлетворяют основным требованиям, предъявляемым к информационным системам (ИС).Диаграмма ERD может включать связи многие-ко-многим и не включать описание ключей. Как правило, ERD используется для презентаций и обсуждения структуры данных с экспертами предметной области.

Модель данных, основанная на ключах, - более подробное представление данных. Она включает описание всех сущностей и первичных ключей и предназначена для представления структуры данных и ключей, которые соответствуют предметной области.

Полная атрибутивная модель – наиболее детальное представление структуры данных: представляет данные в третьей нормальной форме и включает все сущности, атрибуты и связи.

Сущности и атрибуты

Основные компоненты диаграммы ERWin – это сущности, атрибуты и связи. Сущность можно определить как объект, событие или концепцию, информация о которой должна сохраняться. Сущности должны иметь наименование с четким смысловым значением, фактически это имя ее экземпляра. Например, сущность Заказчик с атрибутами Номер заказчика, Фамилия заказчика, Адрес заказчика.

Entity Editor в контекстном меню для сущности позволяет определить имя, описание, комментарии, иконку. Для описания атрибутов сущности выбирается пункт Attribute Editor. Здесь можно указать имя нового атрибута и домен, который будет использоваться при определении типа колонки на уровне физической модели. Атрибуты должны именоваться в единственном числе и иметь четкое смысловое значение. Каждый атрибут должен быть определен (закладка Definition), при этом следует избегать циклических определений и производных атрибутов. Для атрибутов первичного ключа (это атрибут или группа атрибутов, идентифицирующая сущность) необходимо сделать пометку в окне выбора Primary Key.

Связи.

Связь является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой (Relationship Verb Phrases). Имя связи облегчает чтение диаграммы, например:

По умолчанию имя связи на диаграмме не показывается. Для отображения имени следует в контекстном меню для свободного места диаграммы выбрать пункт Display Option/relationship и включить опцию Verb Phrase.

На логическом уровне можно установить идентифицирующую связь один-ко-многим, связь многие-ко-многим и неидентифицирующую связь один-ко-многим (кнопки в палитре инструментов). Тип сущности определяется ее связью с другими сущностями. Различают зависимые и независимые сущности. Идентифицирующая связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. Когда рисуется идентифицирующая связь, ERWin автоматически преобразует дочернюю сущность в зависимую. Зависимая сущность изображается прямоугольником со скругленными углами (в предыдущем примере сущность Заказ). Информация о заказе не может быть внесена и не имеет смысла без информации о клиенте, который ее размещает. При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности и помечается в дочерней сущности как внешний ключ (FK). Эта операция называется миграцией атрибутов. В дальнейшем, при генерации схемы БД, атрибуты первичного ключа получат признак NOT NULL, что означает невозможность внесения записи в таблицу заказов без информации о номере клиента.

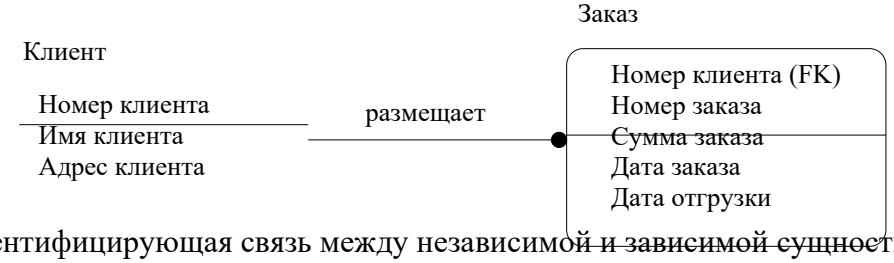


Рис. 5. Идентифицирующая связь между независимой и зависимой сущностью.

При установлении неидентифицирующей связи дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов родительской сущности. Неидентифицирующая связь служит для связывания независимых сущностей. Экземпляр сущности Сотрудник может существовать безотносительно к какому-либо экземпляру сущности Отдел, т. е. Сотрудник может работать в организации, не числясь в каком-либо отделе.

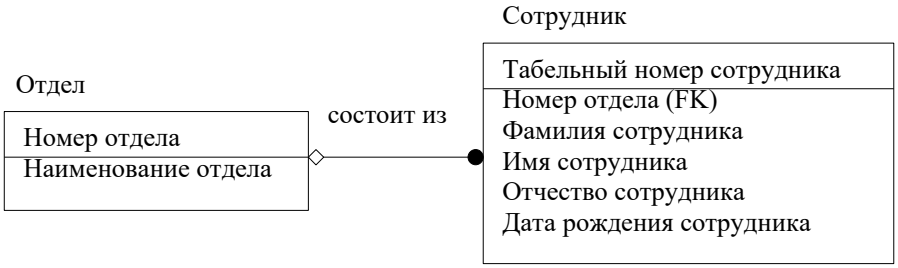


Рис. 6. Неидентифицирующая связь.

Во вкладке *General* меню *Relationship Editor* можно задать мощность, имя и тип связи.

Мощность связи (Cardinality) – служит для обозначения отношения числа экземпляров родительской сущности к числу экземпляров дочерней.

Можно использовать одну из четырех типов мощности:

- общий случай, когда одному экземпляру родительской сущности соответствует 0, 1 или много экземпляров дочерней сущности (не помечается каким-либо символом);
- одному экземпляру родительской сущности соответствует 1 или много экземпляров дочерней сущности (помечается символом P);
- одному экземпляру родительской сущности соответствует 0 или 1 экземпляр дочерней сущности (помечается символом Z);
- одному экземпляру родительской сущности соответствует заранее заданное число экземпляров дочерней сущности (помечается цифрой точного соответствия).

По умолчанию символ, обозначающий мощность связи, не показывается на диаграмме. Для отображения имени следует в контекстном меню для диаграммы (в месте не занятом объектами модели) выбрать пункт Display Options/Relationship и затем включить опцию Cardinality.

Имя связи (Verb Phrase) – фраза, характеризующая отношение между родительской и дочерней сущностями. Для связи один-ко-многим идентифицирующей или неидентифицирующей достаточно указать имя, характеризующее отношение от родительской к дочерней сущности (Parent-to-Child). Для связи многие-ко-многим следует указывать имена как Parent-to Child так и Child-to-Parent.

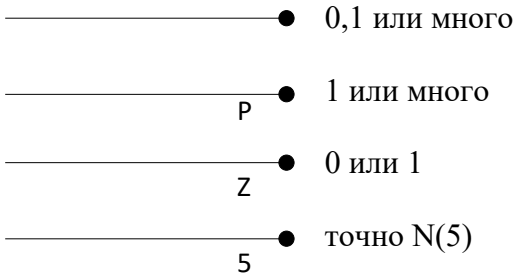


Рис.2.3. Обозначения мощности.

Связь многие-ко-многим возможна только на логическом уровне. При переходе к физическому уровню Erwin автоматически преобразует связь многие-ко-многим, добавляя новую таблицу и устанавливая две новые связи один-ко-многим от старых к новой таблице. Имя новой таблице присваивается автоматически как «Имя1_Имя2».

Контрольные вопросы

1. Основные этапы проектирования базы данных
2. Принципы построения логической модели данных.
3. Какие типы связей используются при построении модели «сущность-связь»?
4. Привести примеры идентифицирующих и неидентифицирующих связей?
5. Что такое мощность связи?

Лабораторная работа №3 Проектирование системы с использованием UML диаграмм

Цель работы:

Ознакомление с основными элементами определения, представления, проектирования и моделирования программных систем с помощью языка UML.

Общие сведения об объектном моделировании ИС

Существует множество технологий и инструментальных средств, с помощью которых можно реализовать в некотором смысле оптимальный проект ИС, начиная с этапа анализа и заканчивая созданием программного кода системы. В большинстве случаев эти технологии предъявляют весьма жесткие требования к процессу разработки и используемым ресурсам, а попытки трансформировать их под конкретные проекты оказываются безуспешными. Эти технологии представлены CASE-средствами верхнего уровня или CASE-средствами полного жизненного цикла (upper CASE tools или full life-cycle CASE tools). Они не позволяют оптимизировать деятельность на уровне отдельных элементов проекта, и, как следствие, многие разработчики перешли на так называемые CASE-средства нижнего уровня (lower CASE tools). Однако они столкнулись с новой проблемой — проблемой организации взаимодействия между различными командами, реализующими проект.

Диаграммы вариантов использования

Понятие варианта использования (use case) впервые ввел Ивар Якобсон и придал ему такую значимость, что в настоящее время вариант использования превратился в основной элемент разработки и планирования проекта.

Вариант использования представляет собой последовательность действий (транзакций), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). Вариант использования описывает типичное взаимодействие между пользователем и системой. В простейшем случае вариант использования определяется в процессе обсуждения с пользователем тех функций, которые он хотел бы реализовать. На языке UML вариант использования изображают следующим образом:

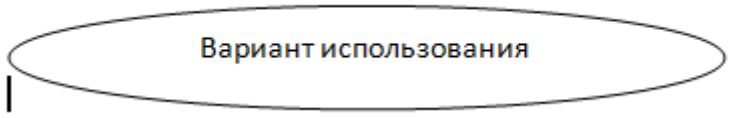


Рис.2. Вариант использования

Действующее лицо (actor) – это роль, которую пользователь играет по отношению к системе. Действующие лица представляют собой роли, а не конкретных людей или наименования работ. Несмотря на то, что на диаграммах вариантов использования они изображаются в виде стилизованных человеческих фигурок, действующее лицо может также быть внешней системой, которой необходима некоторая информация от данной системы. Показывать на диаграмме действующих лиц следует только в том случае, когда им действительно необходимы некоторые варианты использования. На языке UML действующие лица представляют в виде фигур:

Рис.3. Действующее лицо (актер)

Действующие лица делятся на три основных типа:

- пользователи;
- системы;
- другие системы, взаимодействующие с данной;
- время.

Время становится действующим лицом, если от него зависит запуск каких-либо событий в системе.

Связи между вариантами использования и действующими лицами

В языке UML на диаграммах вариантов использования поддерживается несколько типов связей между элементами диаграммы. Это связи коммуникации (communication), включения (include), расширения (extend) и обобщения (generalization).

Связь коммуникации – это связь между вариантом использования и действующим лицом. На языке UML связи коммуникации показывают с помощью однонаправленной ассоциации (сплошной линии).

Рис.4. Пример связи коммуникации

Связь включения применяется в тех ситуациях, когда имеется какой-либо фрагмент поведения системы, который повторяется более чем в одном варианте использования. С помощью таких связей обычно моделируют многократно используемую функциональность.

Связь расширения применяется при описании изменений в нормальном поведении системы. Она позволяет варианту использования только при необходимости использовать функциональные возможности другого.

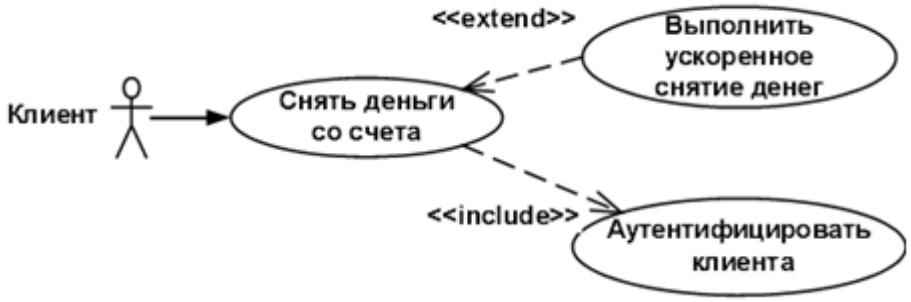


Рис.5. Пример связи включения и расширения

С помощью связи обобщения показывают, что у нескольких действующих лиц имеются общие черты.

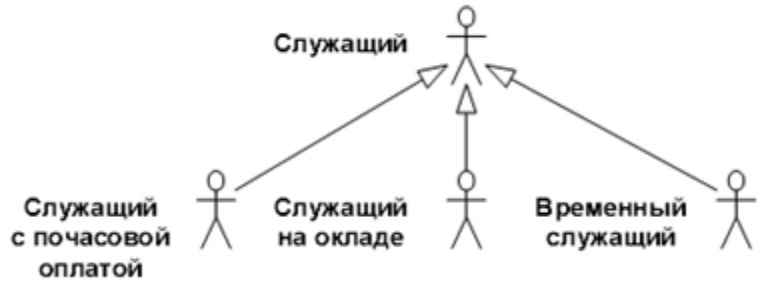


Рис.6. Пример связи обобщения

Диаграммы взаимодействия (interaction diagrams)

Диаграммы взаимодействия (interaction diagrams) описывают поведение взаимодействующих групп объектов. Как правило, диаграмма взаимодействия охватывает поведение объектов

в рамках только одного варианта использования. На такой диаграмме отображается ряд объектов и те сообщения, которыми они обмениваются между собой.

Сообщение (message) – это средство, с помощью которого объект-отправитель запрашивает у объекта получателя выполнение одной из его операций.

Информационное (informative) сообщение – это сообщение, снабжающее объект-получатель некоторой информацией для обновления его состояния.

Сообщение-запрос (interrogative) – это сообщение, запрашивающее выдачу некоторой информации об объекте-получателе.

Императивное (imperative) сообщение – это сообщение, запрашивающее у объекта-получателя выполнение некоторых действий.

Существует два вида диаграмм взаимодействия: диаграммы последовательности (sequence diagrams) и кооперативные диаграммы (collaboration diagrams).

Диаграмма последовательности (sequence diagrams)

Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования.

Все действующие лица показаны в верхней части диаграммы. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций.

На диаграмме последовательности объект изображается в виде прямоугольника, от которого вниз проведена пунктирная вертикальная линия. Эта линия называется линией жизни (lifeline) объекта. Она представляет собой фрагмент жизненного цикла объекта в процессе взаимодействия.

Каждое сообщение представляется в виде стрелки между линиями жизни двух объектов. Сообщения появляются в том порядке, как они показаны на странице сверху вниз. Каждое сообщение помечается как минимум именем сообщения. При желании можно добавить также аргументы и некоторую управляющую информацию. Можно показать самоделегирование (self-delegation) – сообщение, которое объект посылает самому себе, при этом стрелка сообщения указывает на ту же самую линию жизни.

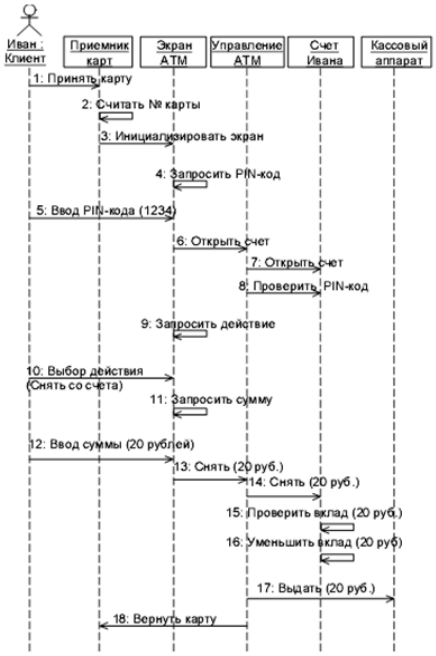


Рис. 7. Пример диаграммы последовательности

Диаграмма кооперации (collaboration diagram)

Диаграммы кооперации отображают поток событий через конкретный сценарий варианта использования, упорядочены по времени, а кооперативные диаграммы больше внимания заостряют на связях между объектами.

На диаграмме кооперации представлена вся та информация, которая есть и на диаграмме последовательности, но кооперативная диаграмма по-другому описывает поток событий. Из нее легче понять связи между объектами, однако, труднее уяснить последовательность событий.

На кооперативной диаграмме так же, как и на диаграмме последовательности, стрелки обозначают сообщения, обмен которыми осуществляется в рамках данного варианта использования. Их временная последовательность указывается путем нумерации сообщений.

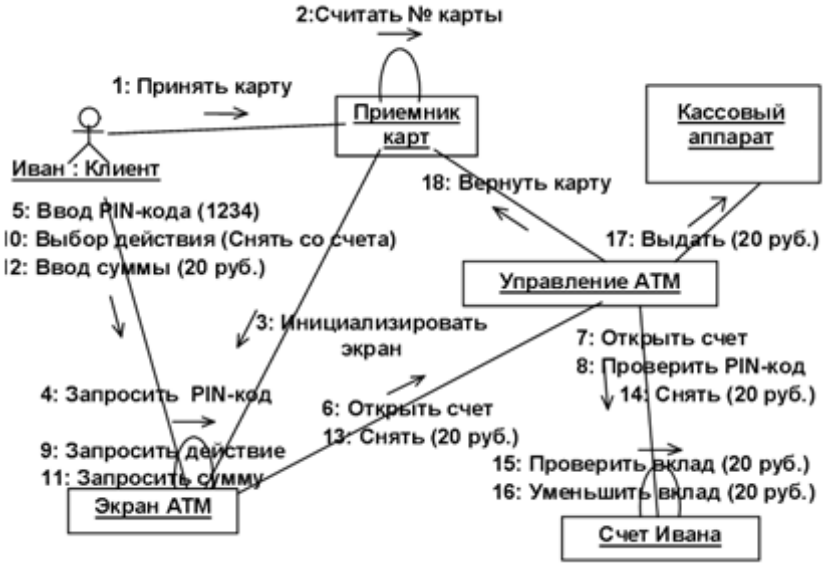


Рис. 8. Пример диаграммы кооперации

Диаграммы классов

Общие сведения

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.

Диаграмма классов UML - это граф, узлами которого являются элементы статической структуры проекта (классы, интерфейсы), а дугами - отношения между узлами (ассоциации, наследование, зависимости).

На диаграмме классов изображаются следующие элементы:

- Пакет (package) - набор элементов модели, логически связанных между собой;
- Класс (class) - описание общих свойств группы сходных объектов;
- Интерфейс (interface) - абстрактный класс, задающий набор операций, которые объект произвольного класса, связанного с данным интерфейсом, предоставляет другим объектам.

Класс

Класс - это группа сущностей (объектов), обладающих сходными свойствами, а именно, данными и поведением. Отдельный представитель некоторого класса называется объектом класса или просто объектом.

Под поведением объекта в UML понимаются любые правила взаимодействия объекта с внешним миром и с данными самого объекта.

На диаграммах класс изображается в виде прямоугольника со сплошной границей, разделенного горизонтальными линиями на 3 секции:

- Верхняя секция (секция имени) содержит имя класса и другие общие свойства (в частности, стереотип).
- В средней секции содержится список атрибутов

- В нижней - список операций класса, отражающих его поведение (действия, выполняемые классом).

Любая из секций атрибутов и операций может не изображаться (а также обе сразу). Для отсутствующей секции не нужно рисовать разделительную линию и как-либо указывать на наличие или отсутствие элементов в ней.

На усмотрение конкретной реализации могут быть введены дополнительные секции, например, исключения (Exceptions).

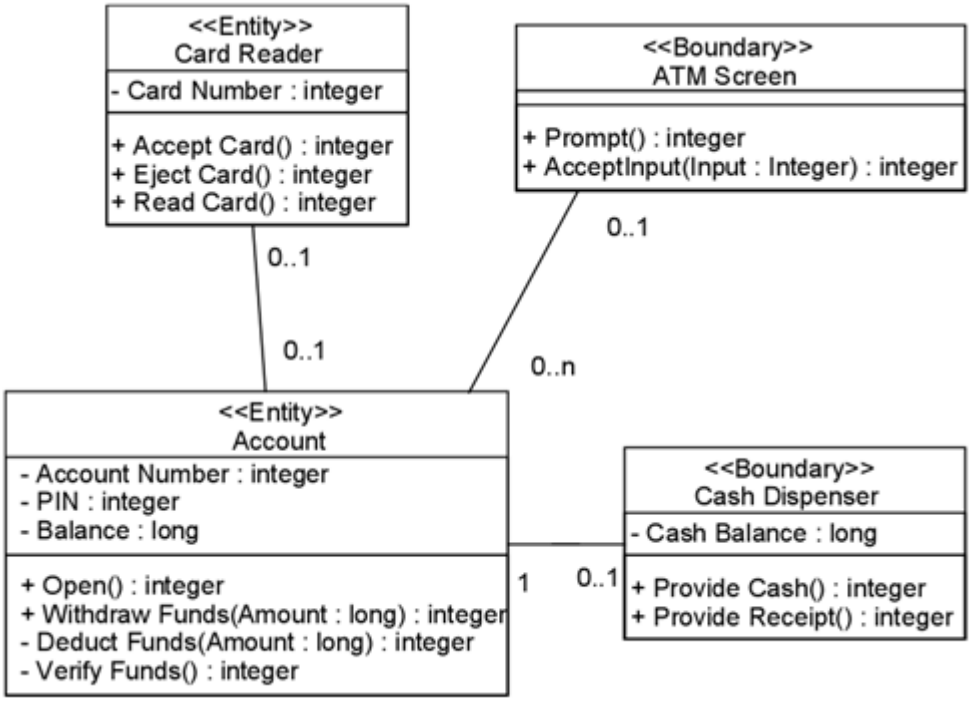


Рис. 9. Пример диаграммы классов

Стереотипы классов

Стереотипы классов – это механизм, позволяющий разделять классы на категории.

В языке UML определены три основных стереотипа классов:

- Boundary (граница);
- Entity (сущность);
- Control (управление).

Граничные классы

Граничными классами (boundary classes) называются такие классы, которые расположены на границе системы и всей окружающей среды. Это экранные формы, отчеты, интерфейсы с аппаратурой (такой как принтеры или сканеры) и интерфейсы с другими системами.

Чтобы найти граничные классы, надо исследовать диаграммы вариантов использования. Каждому взаимодействию между действующим лицом и вариантом использования должен соответствовать, по крайней мере, один граничный класс. Именно такой класс позволяет действующему лицу взаимодействовать с системой.

Классы-сущности

Классы-сущности (entity classes) содержат хранимую информацию. Они имеют наибольшее значение для пользователя, и потому в их названиях часто используют термины из предметной области. Обычно для каждого класса-сущности создают таблицу в базе данных.

Управляющие классы

Управляющие классы (control classes) отвечают за координацию действий других классов. Обычно у каждого варианта использования имеется один управляющий класс, контролирующий последовательность событий этого варианта использования.

Управляющий класс отвечает за координацию, но сам не несет в себе никакой функциональности, так как остальные классы не посылают ему большого количества сообщений. Вместо этого он сам посылает множество сообщений. Управляющий класс просто делегирует ответственность другим классам, по этой причине его часто называют классом-менеджером.

В системе могут быть и другие управляющие классы, общие для нескольких вариантов использования. Например, может быть класс SecurityManager (менеджер безопасности), отвечающий за контроль событий, связанных с безопасностью. Класс TransactionManager (менеджер транзакций) занимается координацией сообщений, относящихся к транзакциям с базой данных. Могут быть и другие менеджеры для работы с другими элементами функционирования системы, такими как разделение ресурсов, распределенная обработка данных или обработка ошибок.

Помимо упомянутых выше стереотипов можно создавать и свои собственные.

Атрибуты

Атрибут – это элемент информации, связанный с классом. Атрибуты хранят инкапсулированные данные класса.

Так как атрибуты содержатся внутри класса, они скрыты от других классов. В связи с этим может понадобиться указать, какие классы имеют право читать и изменять атрибуты. Это свойство называется видимостью атрибута (attribute visibility).

У атрибута можно определить четыре возможных значения этого параметра:

- Public (общий, открытый). Это значение видимости предполагает, что атрибут будет виден всеми остальными классами. Любой класс может просмотреть или изменить значение атрибута. В соответствии с нотацией UML общему атрибуту предшествует знак « + ».
- Private (закрытый, секретный). Соответствующий атрибут не виден никаким другим классом. Закрытый атрибут обозначается знаком « – » в соответствии с нотацией UML.
- Protected (защищенный). Такой атрибут доступен только самому классу и его потомкам. Нотация UML для защищенного атрибута – это знак « # ».
- Package or Implementation (пакетный). Предполагает, что данный атрибут является общим, но только в пределах его пакета. Этот тип видимости не обозначается никаким специальным значком.

В общем случае, атрибуты рекомендуется делать закрытыми или защищенными. Это позволяет лучше контролировать сам атрибут и код.

С помощью закрытости или защищенности удается избежать ситуации, когда значение атрибута изменяется всеми классами системы. Вместо этого логика изменения атрибута будет заключена в том же классе, что и сам этот атрибут. Задаваемые параметры видимости повлияют на генерируемый код.

Операции

Операции реализуют связанное с классом поведение. Операция включает три части – имя, параметры и тип возвращаемого значения.

Параметры – это аргументы, получаемые операцией «на входе». Тип возвращаемого значения относится к результату действия операции.

На диаграмме классов можно показывать как имена операций, так и имена операций вместе с их параметрами и типом возвращаемого значения. Чтобы уменьшить загруженность диаграммы, полезно бывает на некоторых из них показывать только имена операций, а на других их полную сигнатуру.

В языке UML операции имеют следующую нотацию:

Имя Операции (аргумент: тип данных аргумента, аргумент2:тип данных аргумента2,...): тип возвращаемого значения

Следует рассмотреть четыре различных типа операций:

- Операции реализации;

- Операции управления;
- Операции доступа;
- Вспомогательные операции.

Операции реализации

Операции реализации (implementor operations) реализуют некоторые бизнес-функции. Такие операции можно найти, исследуя диаграммы взаимодействия. Диаграммы этого типа фокусируются на бизнес-функциях, и каждое сообщение диаграммы, скорее всего, можно соотнести с операцией реализации.

Каждая операция реализации должна быть легко прослеживаема до соответствующего требования. Это достигается на различных этапах моделирования. Операция выводится из сообщения на диаграмме взаимодействия, сообщения исходят из подробного описания потока событий, который создается на основе варианта использования, а последний – на основе требований. Возможность проследить всю эту цепочку позволяет гарантировать, что каждое требование будет реализовано в коде, а каждый фрагмент кода реализует какое-то требование.

Операции управления

Операции управления (manager operations) управляют созданием и уничтожением объектов. В эту категорию попадают конструкторы и деструкторы классов.

Операции доступа

Атрибуты обычно бывают закрытыми или защищенными. Тем не менее, другие классы иногда должны просматривать или изменять их значения. Для этого существуют операции доступа (access operations). Такой подход дает возможность безопасно инкапсулировать атрибуты внутри класса, защитив их от других классов, но все же позволяет осуществить к ним контролируемый доступ. Создание операций Get и Set (получения и изменения значения) для каждого атрибута класса является стандартом.

Вспомогательные операции

Вспомогательными (helper operations) называются такие операции класса, которые необходимы ему для выполнения его ответственностей, но о которых другие классы не должны ничего знать. Это закрытые и защищенные операции класса.

Чтобы идентифицировать операции, выполните следующие действия:

1. Изучите диаграммы последовательности и кооперативные диаграммы. Большая часть сообщений на этих диаграммах является операциями реализации. Рефлексивные сообщения будут вспомогательными операциями.
2. Рассмотрите управляющие операции. Может потребоваться добавить конструкторы и деструкторы.
3. Рассмотрите операции доступа. Для каждого атрибута класса, с которым должны будут работать другие классы, надо создать операции Get и Set.

Связи

Связь представляет собой семантическую взаимосвязь между классами. Она дает классу возможность узнавать об атрибутах, операциях и связях другого класса. Иными словами, чтобы один класс мог послать сообщение другому на диаграмме последовательности или кооперативной диаграмме, между ними должна существовать связь.

Существуют четыре типа связей, которые могут быть установлены между классами: ассоциации, зависимости, агрегации и обобщения.

Ассоциации

Ассоциация (association) – это семантическая связь между классами. Их рисуют на диаграмме классов в виде обыкновенной линии.

Рис. 10. Связь ассоциация

Ассоциации могут быть двунаправленными, как в примере, или однонаправленными. На языке UML двунаправленные ассоциации рисуют в виде простой линии без стрелок или со стрелками с обеих ее сторон. На однонаправленной ассоциации изображают только одну стрелку, показывающую ее направление.

Направление ассоциации можно определить, изучая диаграммы последовательности и кооперативные диаграммы. Если все сообщения на них отправляются только одним классом и принимаются только другим классом, но не наоборот, между этими классами имеет место однонаправленная связь. Если хотя бы одно сообщение отправляется в обратную сторону, ассоциация должна быть двунаправленной.

Ассоциации могут быть рефлексивными. Рефлексивная ассоциация предполагает, что один экземпляр класса взаимодействует с другими экземплярами этого же класса.

Зависимости

Связи зависимости (dependency) также отражают связь между классами, но они всегда однонаправлены и показывают, что один класс зависит от определений, сделанных в другом. Например, класс А использует методы класса В. Тогда при изменении класса В необходимо произвести соответствующие изменения в классе А.

Зависимость изображается пунктирной линией, проведенной между двумя элементами диаграммы, и считается, что элемент, привязанный к концу стрелки, зависит от элемента, привязанного к началу этой стрелки.

Рис. 11. Связь зависимость

При генерации кода для этих классов к ним не будут добавляться новые атрибуты. Однако, будут созданы специфические для языка операторы, необходимые для поддержки связи.

Агрегации

Агрегации (aggregations) представляют собой более тесную форму ассоциации. Агрегация – это связь между целым и его частью. Например, у вас может быть класс Автомобиль, а также классы Двигатель, Покрышки и классы для других частей автомобиля. В результате объект класса Автомобиль будет состоять из объекта класса Двигатель, четырех объектов Покрышек и т. д. Агрегации визуализируют в виде линии с ромбиком у класса, являющегося целым:

Рис. 11. Связь агрегация

В дополнение к простой агрегации UML вводит более сильную разновидность агрегации, называемую композицией. Согласно композиции, объект-часть может принадлежать только единственному целому, и, кроме того, как правило, жизненный

цикл частей совпадает с циклом целого: они живут и умирают вместе с ним. Любое удаление целого распространяется на его части.

Такое каскадное удаление нередко рассматривается как часть определения агрегации, однако оно всегда подразумевается в том случае, когда множественность роли составляет 1..1; например, если необходимо удалить Клиента, то это удаление должно распространиться и на Заказы (и, в свою очередь, на Строки заказа).

Обобщения (Наследование)

Обобщение (наследование) - это отношение типа общее-частное между элементами модели. С помощью обобщений (generalization) показывают связи наследования между двумя классами. Большинство объектно-ориентированных языков непосредственно поддерживают концепцию наследования. Она позволяет одному классу наследовать все атрибуты, операции и связи другого. Наследование пакетов означает, что в пакете-наследнике все сущности пакета-предка будут видны под своими собственными именами (т.е. пространства имен объединяются). Наследование показывается сплошной линией, идущей от класса-потомка к классу-предку (в терминологии ООП - от потомка к предку, от сына к отцу, или от подкласса к суперклассу). Со стороны более общего элемента рисуется большой полый треугольник.

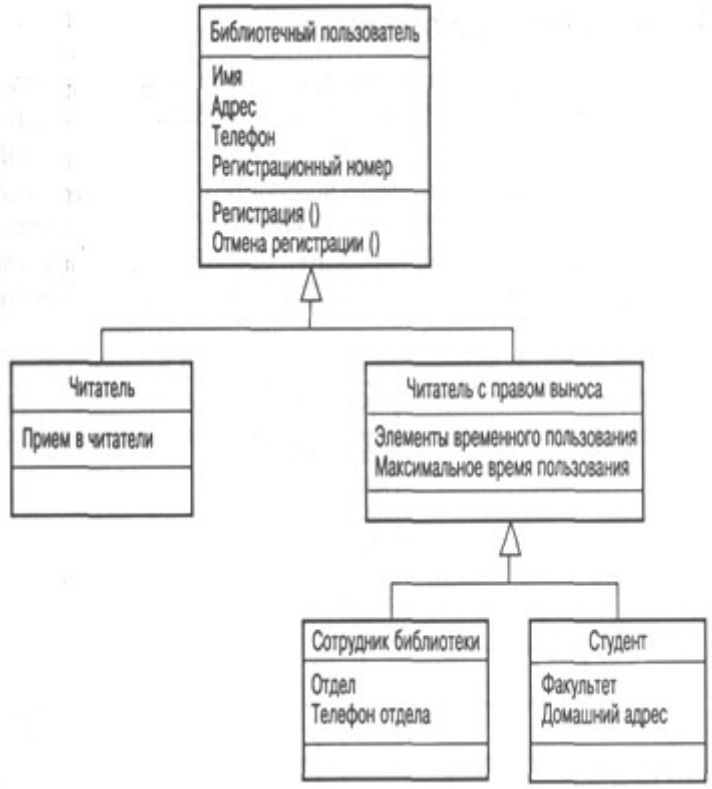


Рис. 12. Пример связи наследование

Помимо наследуемых, каждый подкласс имеет свои собственные уникальные атрибуты, операции и связи.

Множественность

Множественность (multiplicity) показывает, сколько экземпляров одного класса взаимодействуют с помощью этой связи с одним экземпляром другого класса в данный момент времени.

Например, при разработке системы регистрации курсов в университете можно определить классы Course (курс) и Student (студент). Между ними установлена связь: у курсов могут быть студенты, а у студентов – курсы. Вопросы, на который должен ответить параметр множественности: «Сколько курсов студент может посещать в данный момент? Сколько студентов может за раз посещать один курс?»

Так как множественность дает ответ на оба эти вопроса, её индикаторы устанавливаются на обоих концах линии связи. В примере регистрации курсов мы решили, что один студент может посещать от нуля до четырех курсов, а один курс могут слушать от 0 до 20 студентов.
В языке UML приняты определенные нотации для обозначения множественности.
Таблица 1 - Обозначения множественности связей в UML

Множественность	Значение
0..*	Ноль или больше
1..*	Один или больше
0..1	Ноль или один
1..1 (сокращенная запись: 1)	Ровно один

Имена связей
Связи можно уточнить с помощью имен связей или ролевых имен. Имя связи – это обычно глагол или глагольная фраза, описывающая, зачем она нужна. Например, между классом Person (человек) и классом Company (компания) может существовать ассоциация. Можно задать в связи с этим вопрос, является ли объект класса Person клиентом компании, её сотрудником или владельцем? Чтобы определить это, ассоциацию можно назвать «employs» (нанимает):

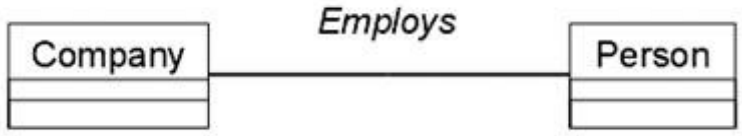


Рис. 13. Пример имен связей
Роли
Ролевые имена применяют в связях ассоциации или агрегации вместо имен для описания того, зачем эти связи нужны. Возвращаясь к примеру с классами Person и Company, можно сказать, что класс Person играет роль сотрудника класса Company. Ролевые имена – это обычно имена существительные или основанные на них фразы, их показывают на диаграмме рядом с классом, играющим соответствующую роль. Как правило, пользуются или ролевым именем, или именем связи, но не обоими сразу. Как и имена связей, ролевые имена не обязательны, их дают, только если цель связи не очевидна. Пример ролей приводится ниже:

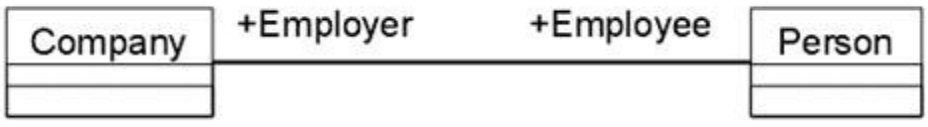


Рис. 14. Пример ролей связей
Пакет. Механизм пакетов
В контексте диаграмм классов, пакет - это вместилище для некоторого набора классов и других пакетов. Пакет является самостоятельным пространством имен.

Рис. 15. Обозначение пакета в UML

В UML нет каких-либо ограничений на правила, по которым разработчики могут или должны группировать классы в пакеты. Но есть некоторые стандартные случаи, когда такая группировка уместна, например, тесно взаимодействующие классы, или более общий случай - разбиение системы на подсистемы.

Пакет физически содержит сущности, определенные в нем (говорят, что "сущности принадлежат пакету"). Это означает, что если будет уничтожен пакет, то будут уничтожено и все его содержимое.

Существует несколько наиболее распространенных подходов к группировке.

Во-первых, можно группировать их по стереотипу. В таком случае получается один пакет с классами-сущностями, один с граничными классами, один с управляющими классами и т.д. Этот подход может быть полезен с точки зрения размещения готовой системы, поскольку все находящиеся на клиентских машинах пограничные классы уже оказываются в одном пакете.

Другой подход заключается в объединении классов по их функциональности. Например, в пакете Security (безопасность) содержатся все классы, отвечающие за безопасность приложения. В таком случае другие пакеты могут называться Employee Maintenance (Работа с сотрудниками), Reporting (Подготовка отчетов) и Error Handling (Обработка ошибок). Преимущество этого подхода заключается в возможности повторного использования.

Механизм пакетов применим к любым элементам модели, а не только к классам. Если для группировки классов не использовать некоторые эвристики, то она становится произвольной. Одна из них, которая в основном используется в UML, – это зависимость. Зависимость между двумя пакетами существует в том случае, если между любыми двумя классами в пакетах существует любая зависимость.

Таким образом, диаграмма пакетов представляет собой диаграмму, содержащую пакеты классов и зависимости между ними. Строго говоря, пакеты и зависимости являются элементами диаграммы классов, то есть диаграмма пакетов – это форма диаграммы классов.

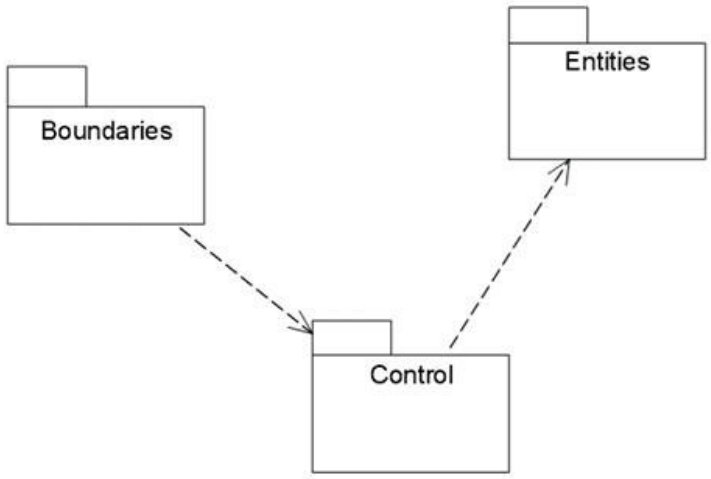


Рис. 16. Пример диаграммы пакетов

Диаграммы состояний

Диаграммы состояний определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате наступления некоторых событий.
Существует много форм диаграмм состояний, незначительно отличающихся друг от друга семантикой.
На диаграмме имеются два специальных состояния – начальное (start) и конечное (stop). Начальное состояние выделено черной точкой, оно соответствует состоянию объекта, когда он только что был создан. Конечное состояние обозначается черной точкой в белом кружке, оно соответствует состоянию объекта непосредственно перед его уничтожением. На диаграмме состояний может быть одно и только одно начальное состояние. В то же время, может быть столько конечных состояний, сколько вам нужно, или их может не быть вообще. Когда объект находится в каком-то конкретном состоянии, могут выполняться различные процессы. Процессы, происходящие, когда объект находится в определенном состоянии, называются действиями (actions).
С состоянием можно связывать данные пяти типов: деятельность, входное действие, выходное действие, событие и история состояния.

Деятельность

Деятельностью (activity) называется поведение, реализуемое объектом, пока он находится в данном состоянии. Деятельность – это прерываемое поведение. Оно может выполняться до своего завершения, пока объект находится в данном состоянии, или может быть прервано переходом объекта в другое состояние. Деятельность изображают внутри самого состояния, ей должно предшествовать слово do (делать) и двоеточие.

Входное действие

Входным действием (entry action) называется поведение, которое выполняется, когда объект переходит в данное состояние. Данное действие осуществляется не после того, как объект перешел в это состояние, а, скорее, как часть этого перехода. В отличие от деятельности, входное действие рассматривается как непрерываемое. Входное действие также показывают внутри состояния, ему предшествует слово entry (вход) и двоеточие.

Выходное действие

Выходное действие (exit action) подобно входному. Однако, оно осуществляется как составная часть процесса выхода из данного состояния. Оно является частью процесса такого перехода. Как и входное, выходное действие является непрерываемым.
Выходное действие изображают внутри состояния, ему предшествует слово exit (выход) и двоеточие.
Поведение объекта во время деятельности, при входных и выходных действиях может включать отправку события другому объекту. В этом случае описанию деятельности, входного действия или выходного действия предшествует знак « ^ ».
Соответствующая строка на диаграмме выглядит как

Do: ^Цель.Событие (Аргументы)

Здесь Цель – это объект, получающий событие, Событие – это посылаемое сообщение, а Аргументы являются параметрами посылаемого сообщения.
Деятельность может также выполняться в результате получения объектом некоторого события. При получении некоторого события выполняется определенная деятельность.
Переходом (Transition) называется перемещение из одного состояния в другое. Совокупность переходов диаграммы показывает, как объект может перемещаться между своими состояниями. На диаграмме все переходы изображают в виде стрелки, начинающейся на первоначальном состоянии и заканчивающейся последующим.
Переходы могут быть рефлексивными. Объект может перейти в то же состояние, в котором он в настоящий момент находится. Рефлексивные переходы изображают в виде стрелки, начинающейся и завершающейся на одном и том же состоянии.
У перехода существует несколько спецификаций. Они включают события, аргументы, ограждающие условия, действия и посылаемые события.

События

Событие (event) – это то, что вызывает переход из одного состояния в другое. Событие размещают на диаграмме вдоль линии перехода.

На диаграмме для отображения события можно использовать как имя операции, так и обычную фразу.

Большинство переходов должны иметь события, так как именно они, прежде всего, заставляют переход осуществиться. Тем не менее, бывают и автоматические переходы, не имеющие событий. При этом объект сам перемещается из одного состояния в другое со скоростью, позволяющей осуществиться входным действиям, деятельности и выходным действиям.

Ограждающие условия

Ограждающие условия (guard conditions) определяют, когда переход может, а когда не может осуществиться. В противном случае переход не осуществится.

Ограждающие условия изображают на диаграмме вдоль линии перехода после имени события, заключая их в квадратные скобки.

Ограждающие условия задавать необязательно. Однако если существует несколько автоматических переходов из состояния, необходимо определить для них взаимно исключающие ограждающие условия. Это поможет читателю диаграммы понять, какой путь перехода будет автоматически выбран.

Действие

Действием (action), как уже говорилось, является непрерываемое поведение, осуществляющееся как часть перехода. Входные и выходные действия показывают внутри состояний, поскольку они определяют, что происходит, когда объект входит или выходит из него. Большую часть действий, однако, изображают вдоль линии перехода, так как они не должны осуществляться при входе или выходе из состояния.

Действие рисуют вдоль линии перехода после имени события, ему предшествует косая черта.

Событие или действие могут быть поведением внутри объекта, а могут представлять собой сообщение, посылаемое другому объекту. Если событие или действие посылается другому объекту, перед ним на диаграмме помещают знак « ^ ».

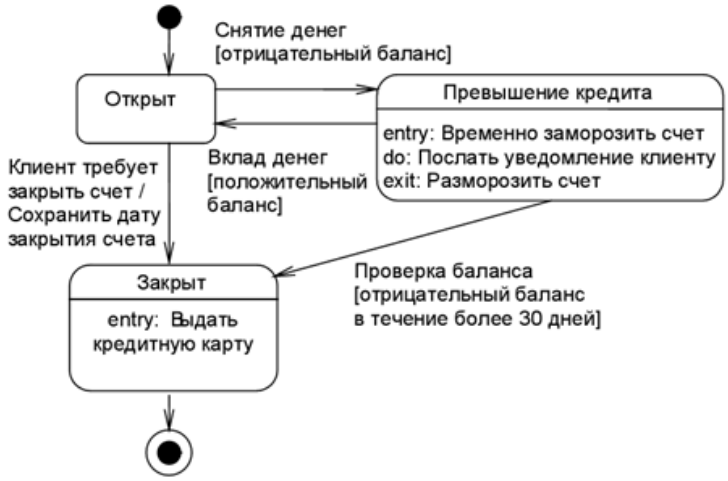


Рис. 17. Пример диаграммы состояний

Диаграммы состояний не надо создавать для каждого класса, они применяются только в сложных случаях. Если объект класса может существовать в нескольких состояниях и в каждом из них ведет себя по-разному, для него может потребоваться такая диаграмма.

Диаграммы размещения

Диаграмма размещения (deployment diagram) отражает физические взаимосвязи между программными и аппаратными компонентами системы. Она является хорошим средством для того, чтобы показать маршруты перемещения объектов и компонентов в распределенной системе.

Каждый узел на диаграмме размещения представляет собой некоторый тип вычислительного устройства – в большинстве случаев, часть аппаратуры. Эта аппаратура может быть простым устройством или датчиком, а может быть и мэйнфреймом.
Диаграмма размещения показывает физическое расположение сети и местонахождение в ней различных компонентов.

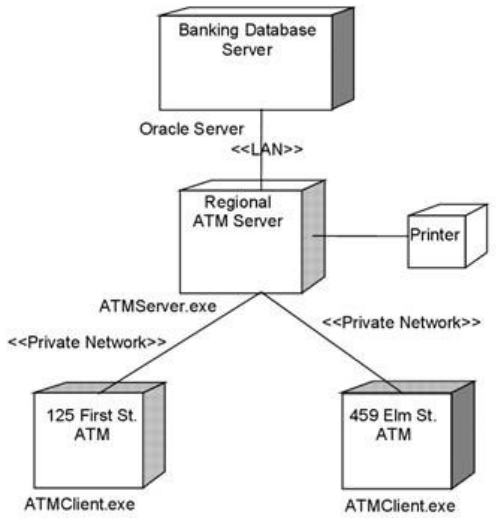


Рис. 19. Пример диаграммы размещения

Диаграмма размещения используется менеджером проекта, пользователями, архитектором системы и эксплуатационным персоналом, чтобы понять физическое размещение системы и расположение её отдельных подсистем.

Диаграммы компонентов

Диаграммы компонентов показывают, как выглядит модель на физическом уровне. На них изображены компоненты программного обеспечения и связи между ними. При этом на такой диаграмме выделяют два типа компонентов: исполняемые компоненты и библиотеки кода.

Каждый класс модели (или подсистема) преобразуется в компонент исходного кода. После создания они сразу добавляются к диаграмме компонентов. Между отдельными компонентами изображают зависимости, соответствующие зависимостям на этапе компиляции или выполнения программы.

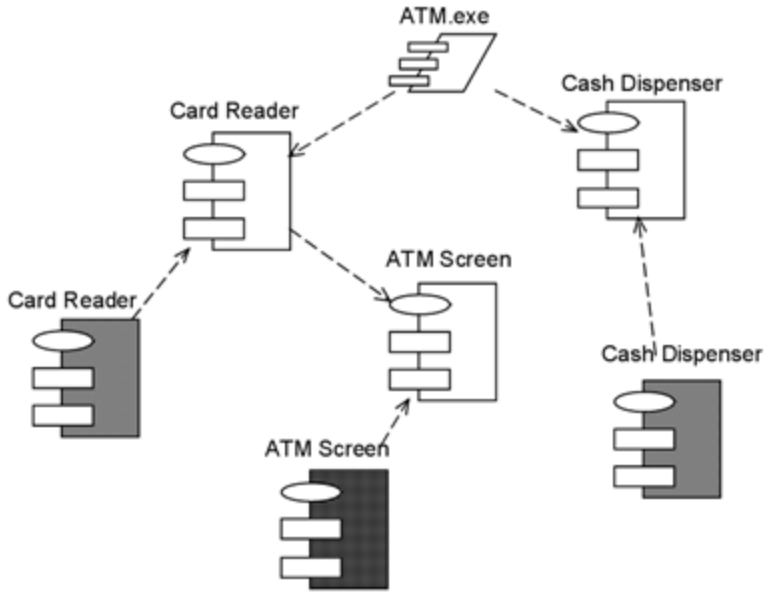


Рис. 18. Пример диаграммы компонентов

Диаграммы компонентов применяются теми участниками проекта, кто отвечает за компиляцию системы. Из нее видно, в каком порядке надо компилировать компоненты, а также какие исполняемые компоненты будут созданы системой. На такой диаграмме показано соответствие классов реализованным компонентам. Она нужна там, где начинается генерация кода.

Порядок выполнения работы

1. Изучить предлагаемый теоретический материал.
2. Постройте диаграмму вариантов использования для выбранной информационной системы.
3. Выполните реализацию вариантов использования в терминах взаимодействующих объектов и представляющую собой набор диаграмм:

- диаграмм классов, реализующих вариант использования;
- диаграмм взаимодействия (диаграмм последовательности и кооперативных диаграмм), отражающих взаимодействие объектов в процессе реализации варианта использования.

4. Разделить классы по пакетам использую один из механизм разбиения.
5. Постройте диаграмму состояний для конкретных объектов информационной системы.
6. Построить отчёт, включающий все полученные уровни модели, описание функциональных блоков, потоков данных, хранилищ и внешних объектов.

Лабораторная работа №4 Разработка многооконных и многоуровневых приложений

Цель работы: Обобщить знания по управляющим элементам ИСР Delphi; получить практические навыки работы с кнопочными компонентами, овладеть практическими навыками в организации ввода/вывода и обработки значений, получить практические навыки создания приложений

Все формы создаются на основе класса TForm. Класс TForm предоставляет возможность изменять поведение и внешний вид формы с помощью ряда свойств, методов и событий.

Таблица. События класса TForm

Название	Описание
OnActive	Возникает при активизации формы
OnClose	Вызывается перед закрытием формы. Параметр Action определяет последствия обработки события: caNone - форма не закрывается, caHide - спрятать форму, caFree - удаление из приложения и динамической памяти, caMinimize - минимизация формы
OnCloseQuery	Вызывается перед закрытием окна. Параметр CanClose определяет возможность закрытия окна
OnCreate	Появляется при создании формы, но перед ее появлением на экране
OnDeactivate	Генерируется, когда форма становится неактивной
OnDestroy	Обрабатывает событие, возникающее перед удалением формы из динамической памяти
OnHelp	Возникает при обращении к справке
OnHide	Срабатывает перед удалением формы с экрана
OnPaint	Является обработчиком события, возникающего при необходимости перерисовки формы
OnResize	Реагирует на изменение размеров формы
OnShow	Возникает, когда форма становится видимой на экране

Многостраничный блокнот PageControl.

Компонент PageControl является управляющим элементом, включающим набор из нескольких страниц, размещаемых одна под другой. Каждая страница имеет закладку, которая является неотъемлемой частью данной страницы, в отличие от одностраничного блокнота. Страницы используются для объединения различных управляющих элементов в группы, обеспечивая их компактное размещение и простое переключение между ними.

Многостраничный блокнот является более сложным управляющим элементом, чем компонент TabControl, однако многие свойства этих двух элементов совпадают. Поэтому будут рассмотрены только свойства, специфичные для компонента PageControl.

Свойство	
ActivePage (тип TTabSheet)	определяет название текущей (выбранной) страницы компонента PageControl.
ActivePagelndex (тип Integer)	служит для определения индекса текущей страницы. С помощью свойств ActivePage и ActivePagelndex можно программно устанавливать новую активную страницу
PageCount (тип Integer)	позволяет определить количество страниц многостраничного блокнота.
Pages [Index: Integer] (тип TTabSheet)	представляет собой список всех страниц управляющего элемента PageControl. Используя данное свойство во время выполнения приложения, можно получить доступ к любой странице блокнота по ее номеру, задаваемому параметром index.

Добавление и удаление страниц, а также перемещение между страницами компонента PageControl в процессе проектирования приложения осуществляется с помощью вызова контекстного меню (нажатием правой кнопки мыши в поле компонента)и дальнейшего выбора соответствующего пункта меню. Перемещаться между страницами можно также простым нажатием на закладке необходимой страницы. Это возможно, потому что каждая страница (включая закладку) является отдельным независимым объектом.

Индивидуальное задание Задание. Разработать многооконное приложение.

1. На главную форму приложения поместить компонент TAnimate. Компонент должен реализовать просмотр пользовательской анимации, путь которому прописывается в свойстве FileName.
2. На главной форме реализовать прослушивание музыкальной композиции при помощи компонента TMediaPlayer.
3. Поместить на форму главное меню, содержащее следующие пункты:
   - задание 1
   - задание 2
   - выход

При выборе пункта меню «Задание 1», необходимо реализовать открытие формы «График» (в модальном режиме). На форме «График» разместить 2 вкладки:

- «Таблица»- содержит таблицу, содержащее протабулированное значение функции y=f(x) на отрезке [a,b]. Количество значений функций равно n. Шаг вычисляется по формуле h=(b-a)/n..

- «График» - содержит график функции y=f(x), начерченный с помощью компонента TChart.

При выборе пункта меню «Задание 2», необходимо открытие формы «Теория» (в немодальном режиме). На форме «Теория» расположить 2 компонента TreeView и WebBrowser. Узлами дерева TreeView являются темы теоретического материала. По щелчку названию темы необходимо осуществить загрузку соответствующей страницы html в компонент WebBrowser.

При выборе пункта меню «Выход», необходимо закрыть приложение.

Контрольные вопросы 1.Назначение, свойства, события и методы компонентов, используемых при разработке приложения

Компонент Form. Назначение, свойства, события и методы

Модальный и немодальный режим открытия окна. Различие, назначение.

Лабораторная работа №5 Разработка пользовательских интерфейсов

Цель работы: изучение принципов проектирования пользовательского интерфейса. Разработка пользовательского интерфейса десктопного приложения

Этапы разработки пользовательского интерфейса Полный цикл разработки интерфейса представлен на рис 1.

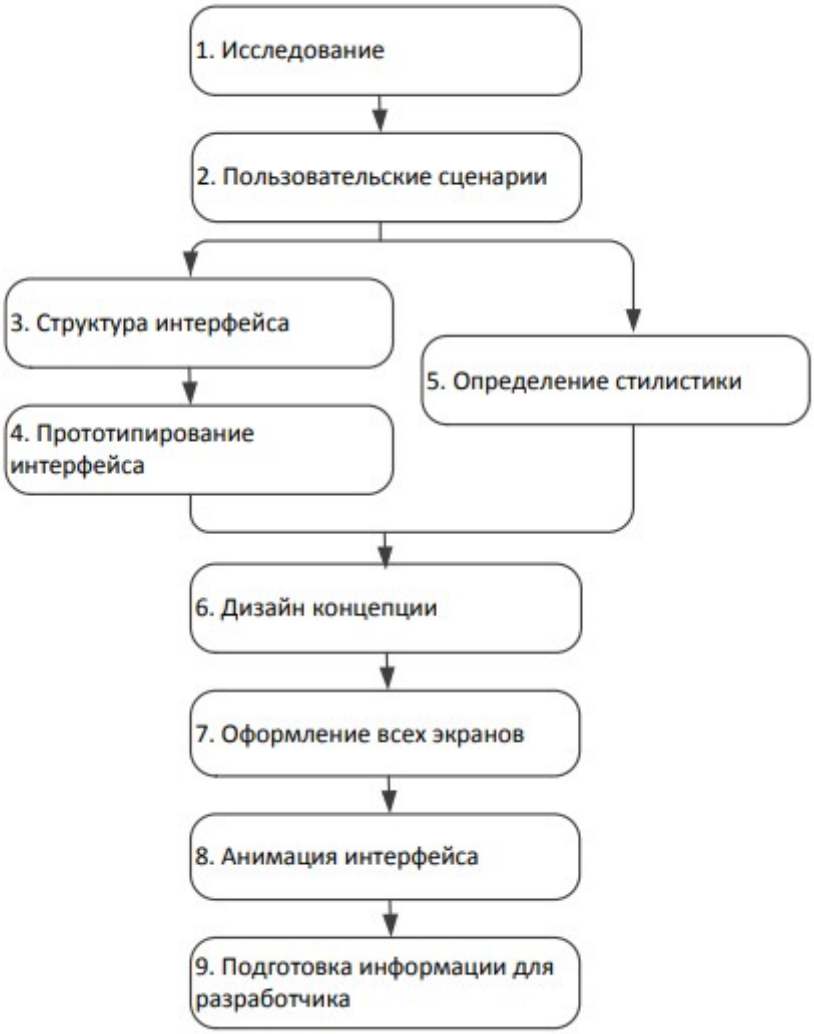


Рис. 2.1. Этапы разработки пользовательского интерфейса

Задание на лабораторную работу Разработать пользовательский интерфейс для десктопного клиента, описанного в лабораторной работе № 1. Разработку пользовательского интерфейса производить согласно п. 2. настоящего пособия.

1. Произвести анализ аналогов, выявить их достоинства и недостатки, результаты анализа отразить в отчете. 2. Разработать пользовательские сценарии и привести их часть в отчете. 3. Разработать карту экранов в части описанных пользовательских сценариев. 4. Разработать черновой прототип экранов. 5. Подобрать подходящие стилистики для приложения не менее двух. 6. На основании одной из стилистик разработать дизайн концепцию приложения
Контрольные вопросы:
1. В чем отличие понятий UI и UX? 2. Какие этапы включает разработка пользовательского интерфейса? 3. Что такое UI-кит? 4. Для чего разрабатывают дизайн-концепцию? 5. В чем отличие чернового прототипа интерфейса от финального?

Лабораторная работа №6 Разработка интерфейсов подключения и работы с данными базы данных

Цель работы: Познакомиться с языком программирования C#, средствами разработки среды Visual Studio .Net 2022 и с использованием сервера баз данных SQL Server 2022, на примере создания клиентского Windows-приложения, работающего с БД. Задача участника: Написать приложение на Windows Forms позволяющее соединяться с удалённой базой данных, просматривать её содержимое, и выполнять стандартные SQL-запросы.
В тело класса DBRequest добавьте поле типа OleDbConnection и его автоматическую инициализацию (выполняемую при создании экземпляра объекта DBRequest):
private OleDbConnection DBCon = new OleDbConnection();
Объект класса OleDbConnection – DBCon способен выполнять соединение пользовательского приложения с базой данных на SQL Server 2000.
Напишите метод ConnectTo, открывающий соединение с базой данных на сервере:

```
//************** Открывает Соединение с Базой Данных*****************
public void ConnectTo( string DataSource, string InitialCatalog )
{
DBCon.ConnectionString = "Provider=SQLOLEDB;"+
"Integrated Security=SSPI;"+
"Data Source="+DataSource+";"+
"Initial Catalog="+InitialCatalog+";"+
"User ID=sa; Password=";
try
{
DBCon.Open();
}
catch(Exception e)
{
throw e;
}
}
```

Метод формирует строку с командой соединения с БД по передаваемым в функцию полям DataSource – строке с сетевым путём к компьютеру с установленным SQL Server 2000 или MSDE,
а так же InitialCatalog – строке с названием каталога базы данных на сервере. Затем делается попытка открыть соединение с базой данных. В случае неудачи, возникающее исключение передаётся вышестоящей по стеку вызовов функции.
Напишите метод Disconnect, выполняющий разрыв открытого соединения с БД:

```
//************** Закрывает Соединение с Базой Данных *****************
public void Disconnect()
{
try
{
```

```
if (DBCon.State == ConnectionState.Open)
DBCon.Close();
}
catch
{
}
}
```

В функции делается проверка – если соединение открыто, то выполняется его закрытие. Добавьте деструктор класса, выполняющий закрытие, открытых соединений:

```
//************************ Деструктор ****************************
~DBRequest()
{
Disconnect();
}
```

Теперь напишем пару методов, реализующих основную функциональность нашего приложения. Первая из этих функций – GetTableFields, для запрашиваемой таблицы из базы (по
её имени TableName) возвращает строку с перечислением полей и их типов:

```
//************
Получение Списка Полей Некоторой Таблицы **************
public string GetTableFields( string TableName )
{
if (DBCon.State == ConnectionState.Open)
```

Если Вы ещё не знакомы с синтаксисом языка SQL, то в данной работе Вы познакомитесь с основными возможностями этого языка, прочитав эту краткую справку.

SQL – это язык структурированных запросов для реляционных баз данных. SQL включает в себя следующие языковые подмножества:

- Data Definition Language (DDL) - определения данных;
- Data Manipulation Language (DML) - манипулирования данными;
- Transaction Control Language (TCL) - управление транзакциями;
- Data Control Language (DCL) - управление привилегий;
- Cursor Control Language (CCL) - управление курсором;

В нашей лабораторной мы ограничимся командами языка DML. Язык DML содержит операторы, позволяющие выбирать, добавлять, удалять и модифицировать данные:

- SELECT – применяется для выборки данных;
- INSERT – применяется для добавления строк к таблице;
- DELETE – применяется для удаления строк из таблицы;
- UPDATE – применяется для исправления данных;

Написанная нами программа, предусматривает только выборку данных из уже существующей базы. Поэтому нам понадобится знание синтаксиса всего одной команды языка
DML, а именно, команды SELECT. Полный синтаксис этой команды записывается в структурированной форме следующим образом:

```
SELECT [ALL | DISTINCT | TOP n] { * | expr_1 [AS c_alias_1] [, ... [, expr_k [AS c_alias_k]]]}
FROM table_name_1 [t_alias_1] [, ... [, table_name_n [t_alias_n]]]
 [ WHERE condition ]
 [ ORDER BY name_of_attr_i [ASC|DESC] [, ... [, name_of_attr_j [ASC|DESC] ] ] ];
 [ GROUP BY name_of_attr_i [,... [, name_of_attr_j]] [ HAVING condition] ]
[ {UNION [ALL] | INTERSECT | EXCEPT} SELECT ...]
```

ALL | DISTINCT – выбирать все или только различные строки таблиц
TOP n [PERCENT] – выбирать только первые n [n процентов] строк

После получения инструкций от помощников, проводящих лабораторную работу,

подключите, написанную Вами программу к базе данных (CompShop) на сервере SQL Server 2000

и введите названия всех таблиц в ячейки DataGrid, отображающего поля таблиц БД.

Щёлкая мышью по строкам DataGrid, с названиями каждой из таблиц просмотрите данные,

содержащиеся в таблицах БД. После этого самостоятельно придумайте 3 демонстрационных

SQL-запроса к базе данных с использованием команды SELECT и выполните их с помощью

Вашей программы. Продемонстрируйте работу вашего приложения

Лабораторная работа №7 Разработка интерфейсов администратора

Цель работы: *сформировать навыки по определению ролей пользователей и созданию интерфейсов.*

Пояснения к работе

После того, как созданы все основные объекты конфигурации можно приступить к определению ролей пользователей и созданию интерфейсов.

До сих пор мы с вами использовали пункт меню « Операции», для того, чтобы получить доступ к тому или иному объекту конфигурации. Нам были доступны абсолютно все объекты конфигурации, и мы могли осуществлять с ними все доступные действия.

Однако при реальной работе пользователей одной из главных возможностей, которую должно обеспечивать прикладное решение, является разграничение прав доступа пользователей к той или иной информации, хранящейся в информационной базе.

Например, руководитель должен, очевидно, иметь доступ ко всей информации, которая содержится в базе данных, а вот кладовщик - напротив, должен иметь доступ только к информации, касающейся движения товаров на складах и не иметь возможности просматривать бухгалтерскую или кадровую информацию.

Кроме этого, должна существовать возможность ограничить пользователей в выполнении тех или иных действий с объектами базы данных. Например, кладовщик может создавать и изменять приходные накладные, поскольку он отвечает за учет материалов на предприятии. Мастеру может понадобиться просматривать приходные накладные для того, чтобы знать, какие материалы и когда были получены. Однако мастер не должен иметь возможности вносить какие-либо изменения в приходные накладные.

Объект конфигурации Роль

Для описания подобных разрешений используются объекты конфигурации Роль. С помощью такого объекта разработчик получает возможность описать набор прав на выполнение тех или иных действий над каждым из объектов базы данных и над всей конфигурацией в целом.

Как правило, роли создаются отдельно для каждого вида деятельности, и каждому пользователю системы ставится в соответствие одна или несколько ролей.

В случае, когда пользователю поставлено в соответствие несколько ролей, предоставление доступа будет осуществляться по следующему алгоритму:

• если хотя бы в одной роли есть разрешение, то доступ будет открыт,

• если во всех ролях разрешение отсутствует, то доступ будет закрыт.

Объект конфигурации Интерфейс

Помимо того, что для каждого пользователя необходимо определить набор его прав в системе, следует также, исходя из разрешенных действий, предоставить пользователю удобный и функциональный интерфейс, не содержащий лишних элементов. Например, кладовщик должен иметь возможность принять и выдать товар, и ему совсем не нужно видеть пункты меню которые позволяют отслеживать работу мастеров или управлять заказами.

Для создания индивидуальных пользовательских интерфейсов предназначены объекты конфигурации Интерфейс. Эти объекты позволяют создавать наборы команд главного меню и панели инструментов, с которыми будет работать пользователь.

Как правило, для каждой категории пользователей создается свой интерфейс, который ставится в соответствие конкретному пользователю. В отличие от ролей, каждому пользователю можно назначить только один интерфейс по умолчанию, однако средствами встроенного языка можно управлять видимостью других интерфейсов.

Задания

После выполнения лабораторной работы Вы должны:

- знать, для чего предназначен объект конфигурации Роль;
- знать, для чего предназначен объект конфигурации Интерфейс;
- уметь создавать роль, используя подсистемы конфигурации;
- уметь создавать интерфейс, используя подсистемы конфигурации;

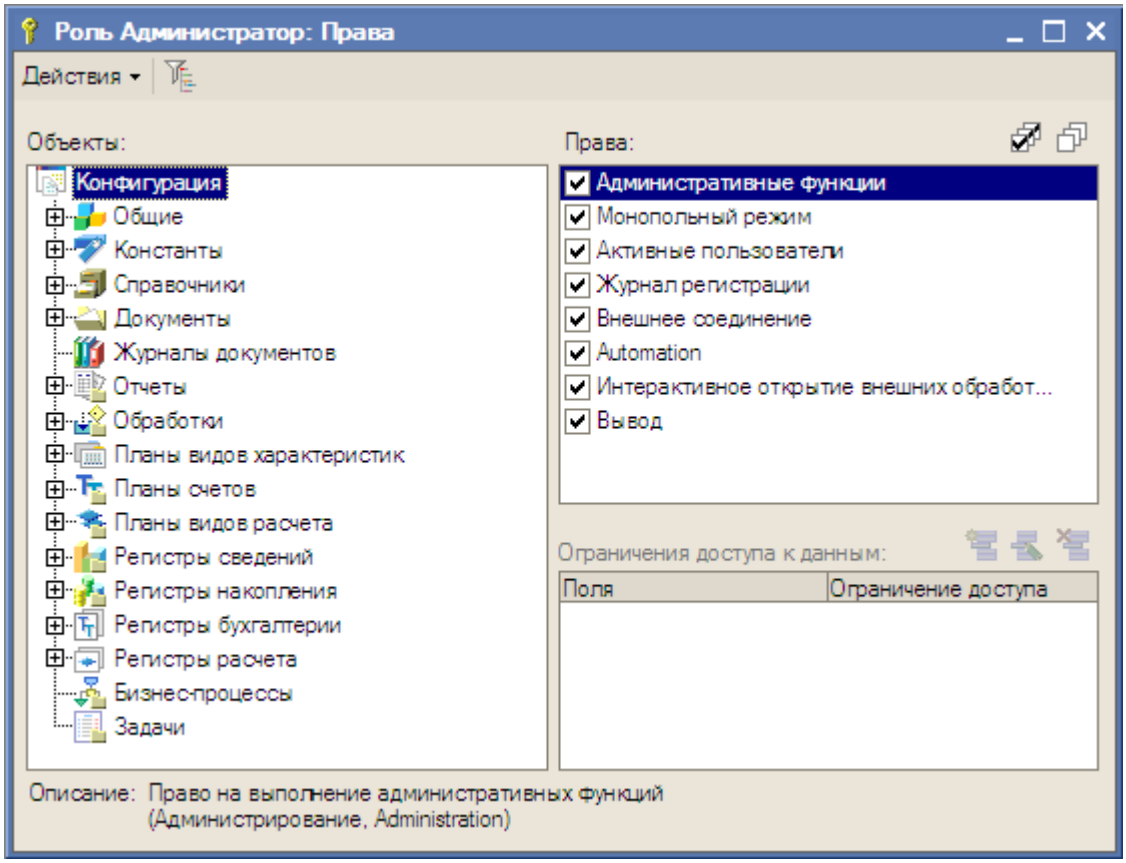
Ход выполнения лабораторной работы

Создание ролей

При создании ролей исходят, как правило, из того, какие полномочия требуются различным группам пользователей на доступ к информации. Для этого ролей мы воспользуемся подсистемами, которые значительно облегчат нашу задачу.

Первая роль, которую мы создадим, будет роль « Администратор». Она должна включать в себя полные права на работу с данными информационной базы.

Создадим новый объект конфигурации Роль с именем «Администратор». Откроется окно редактирования прав:



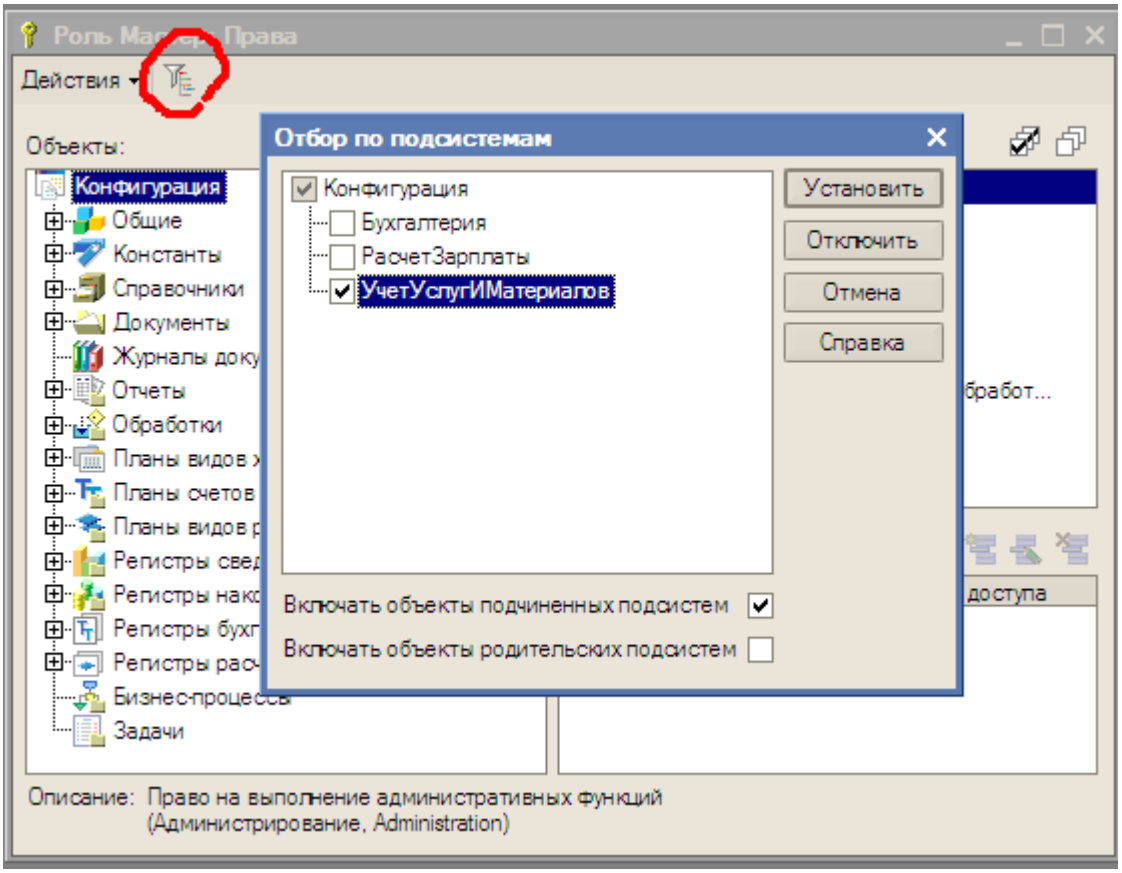
Слева, в списке объектов, перечислены все объекты и виды объектов конфигурации, а справа, в окне прав, перечислены доступные права для выбранного объекта или видов объектов конфигурации. Пробежав по списку объектов, вы обнаружите, что по умолчанию новая роль имеет полные права на все объекты и виды объектов. В данном случае нас это устраивает, поскольку администратор и должен иметь такие права. Единственное, что следует сделать - снять разрешение на интерактивное удаления для всех объектов. Это необходимо для того, чтобы администратор случайно не мог удалить какой-либо объект базы данных.

Следующей ролью, которую мы создадим, будет роль « ТолькоПросмотр». Создадим новый объект конфигурации Роль с именем « ТолькоПросмотр» и в открывшемся окне редактирования прав выполним команду **Действия | Снять все права. В** результате этого, все права на доступ ко всем объектам будут сняты, за исключением тех видов объектов конфигурации, для которых не создано ни одного объекта. Для таких видов объектов конфигурации останутся установлены полные права.

Теперь нам останется лишь пройти по видам объектов конфигурации и установить для них права « Чтение», « Просмотр» и « Использование». Вторая роль нашей конфигурации готова.

Следующая роль, которую мы создадим, будет роль « Мастер». Снова создадим новый объект конфигурации Роль с именем « Мастер» и снимем все права в окне редактирования прав. После этого, Выполним команду **Действия | Установить по подсистемам** и выберем подсистему « УчетМатериаловИУслуг». В результате будут установлены все права на объекты конфигурации, относящиеся к данной подсистеме.

Если теперь установить фильтр объектов по подсистеме « УчетМатериаловИУслуг», то можно, при необходимости, внести уточнения в установленные права. Установим фильтр по подсистеме:



В частности, для справочника « Сотрудники» мы запретим добавление, изменение и удаление. Обратите внимание, что при запрете права « Добавление» исчезла отметка и у права « Интерактивное добавление», т.к. оно является « уточнением» права « Добавление». Точно также « уточненные» права запрещаются и при отмене прав на изменение и удаление.

Кроме этого мы снова снимем разрешения на интерактивное удаление для всех объектов базы данных.

В заключение нам с вами осталось создать две роли: « Бухгалтер» и « Расчетчик». Мы разделим права по расчету зарплаты и по ведению бухгалтерского учета. Дело в том, что в ООО « На все руки мастер» есть бухгалтер и помощник бухгалтера. Помощник бухгалтера занят, в основном, расчетом зарплаты, но иногда это делает и главный бухгалтер. Поэтому ему необходимо будет назначить обе роли, в то время как помощнику - только роль « Расчетчик».

Создадим новый объект конфигурации Роль с именем « Расчетчик». В окне редактирования прав снимем все права и затем установим их по подсистеме « РасчетЗарплаты» (и не забудем запретить интерактивное удаление).

В заключение создадим объект конфигурации Роль с именем « Бухгалтер». В окне редактирования прав снимем все права и затем установим их по подсистеме « Бухгалтерия». После этого отфильтруем список объектов по этой подсистеме и для справочника « Номенклатура» запретим добавление, изменение и удаление. Также запретим интерактивное удаление для всех объектов.

Список прав для каждой роли можно получить, выполнив в окне редактирования прав команду **Действия | Вывести список.**

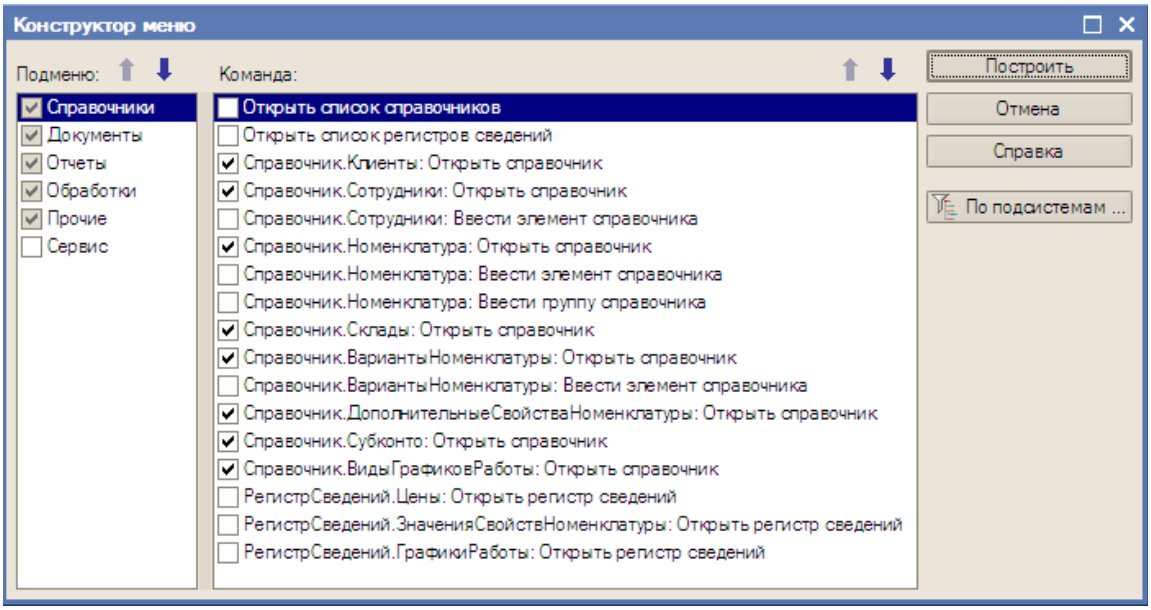
Теперь мы можем перейти к созданию интерфейсов

Создание интерфейсов

Для того, чтобы понять, какие интерфейсы нужно создать в нашей конфигурации, следует определиться с тем, какие группы пользователей собираются работать с нашим прикладным решением.

Скорее всего, это будут руководители, мастера и бухгалтеры. В соответствии с этим мы создадим три различных интерфейса: « Руководитель», « Мастер» и « Бухгалтер». Кроме этого, следует не забыть про то, что у каждой базы данных, как правило, есть администратор - специально выделенный человек, отвечающий за непрерывное функционирование базы, сохранность и достоверность данных. Поскольку администратору нужно предоставить возможность осуществлять обслуживание базы данных - для него мы тоже создадим отдельный интерфейс - « Администратор».

Создадим новый объект конфигурации Интерфейс, и на экране появится конструктор главного меню:

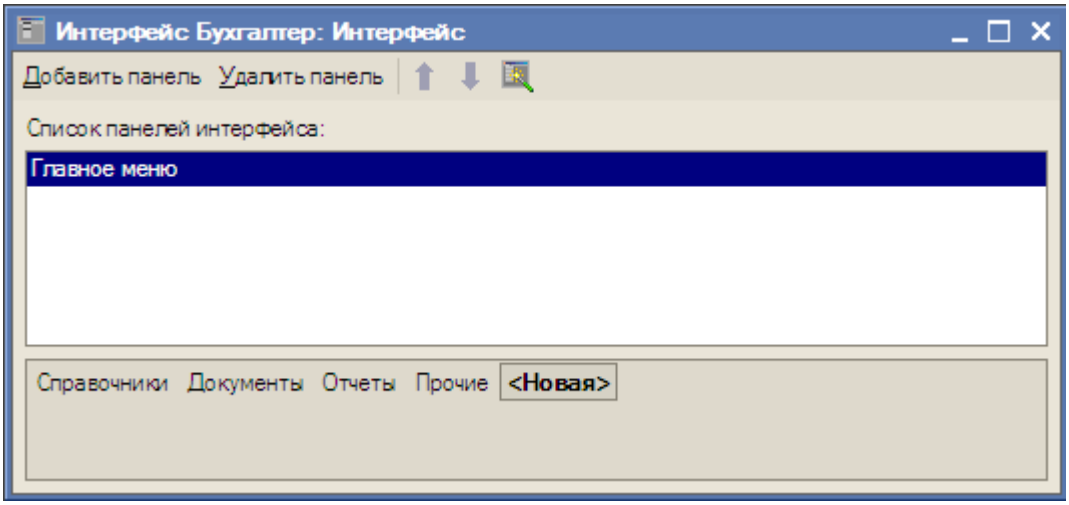


Он содержит два списка: список подменю и список команд выделенного подменю. В этих списках перечислены те пункты подменю и те команды в них, которые предлагается создать по умолчанию. Разработчик может вручную отметить или снять отметку с тех или иных подменю и команд или воспользоваться возможностью построения интерфейса на основе подсистем, существующих в конфигурации.

По умолчанию предлагается сформировать списки подменю и команд по всем подсистемам конфигурации, но, нажав кнопку « По подсистемам», можно указать только некоторые подсистемы. Тогда конструктор построит меню, основываясь на тех объектах конфигурации, которые относятся к указанным подсистемам.

Мы так и поступим. Первый интерфейс, который мы будем создавать, будет интерфейс « Бухгалтер». Поэтому выберем подсистемы « Бухгалтерия» и « РасчетЗарплаты» и нажмем « Установить». В конструкторе меню обновится список команд и используемых подменю. Нажмем « Построить» и зададим имя интерфейса - « Бухгалтер». Укажем, что этот интерфейс будет относиться к подсистемам « Бухгалтерия» и « РасчетЗарплаты».

Вместе с палитрой свойств на экране открылось окно редактст- интерфейса - остановимся на нем подробнее:



Окно редактора интерфейса состоит из трех частей: панель инструментов, список панелей интерфейса и редактора панели. У каждого интерфейса может быть всего одна панель главного меню и несколько панелей инструментов.

Сейчас наш интерфейс « Бухгалтер» содержит только панель главного меню, пункты которого отображены в редакторе панели. При нажатии на любой пункт меню открывается список подменю, содержащий команды этого пункта.

В данном случае нас все устраивает, за исключением пункта « Прочие», в котором для команды « Основной» мы дадим более понятный текст - « План счетов Основной»:

Теперь создадим интерфейсы « Мастер, « Руководитель», « Администратор» мы не будем выбирать никаких подсистем, а сразу построим меню.

Разработчик, по своему усмотрению может добавлять, изменять и удалять пункты меню. Эти действия просты и не требуют специальных описаний. И поскольку создание удобного и эргономичного меню ~ задача творческая - мы лишь показали возможность быстрого создания некоей заготовки, которую разработчик может впоследствии самостоятельно доработать под нужды конкретной группы пользователей.

Администрирование работы пользователей

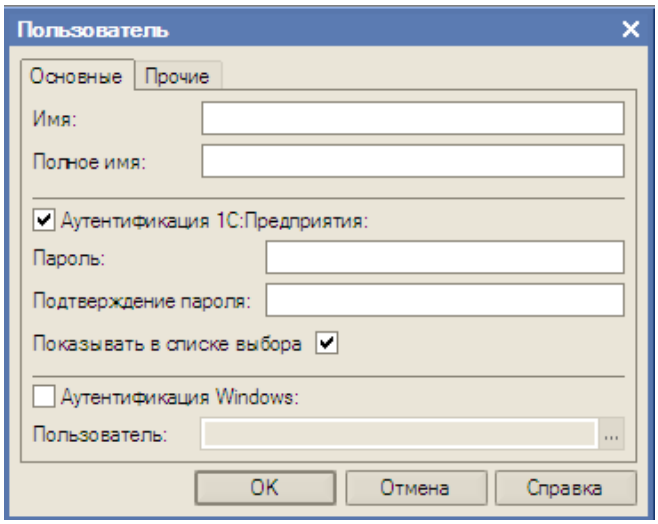
Для того, чтобы иметь возможность отличать друг от друга пользователей, работающих с информационной базой, в системе 1С:Предприятие существует режим ведения списка пользователей.

В этом режиме администратор базы имеет возможность создавать и удалять пользователей системы, назначать им интерфейсы и роли и т.д.

Прежде чем мы приступим к созданию списка пользователей, необходимо выполнить обновление конфигурации базы данных, поскольку пользователю можно поставить в соответствие только тот интерфейс, и те роли, которые существуют в конфигурации базы данных.

После того, как обновление произведено, выполним команду **Администрирование | Пользователи.** Откроется список пользователей системы.

Пока что он пуст, поэтому добавим нового пользователя **(Действия | Добавить):**



Имя пользователя - это идентификатор, который будет появляться в окне выбора пользователей при запуске системы в режиме 1С:Предприятие.

Полное имя - строка, которая может быть использована внутри конфигурации при выводе различной справочной информации Хорошим стилем администрирования считается указание в качестве полного имени - фамилии, имени и отчества пользователя (без сокращений).

Следующие две области окна посвящены способам аутентификации пользователя.

Аутентификация средствами 1С:Предприятия подразумевает, что после запуска системы пользователю будет предложено выбрать имя одного из пользователей системы и ввести пароль. Если введенный пароль соответствует тому, который сохранен в системе для этого идентификатора пользователя, система открывается с правами и интерфейсом, которые указаны для этого пользователя. *Аутентификация Windows* подразумевает, что при запуске системы 1С:Предприятие от пользователя не требуется никакой дополнительной информации. Система 1С:Предприятие определяет под каким пользователем запущена операционная система Windows (имеет смысл использовать для NT-подобных операционных систем: NT, 2000, XP), и затем обращается к своему списку пользователей. Если она находит в нем пользователя, которому поставлен в соответствие текущий пользователь Windows, система открывается с правами и интерфейсом, которые указаны для этого пользователя.

Зададим имя пользователя « Администратор», полное имя тоже « Администратор». Перейдем на закладку « Прочие».

Отметим роль « Администратор», основным интерфейсом укажем « Администратор» и язык конфигурации выберем « Русский».

После этого создадим остальных пользователей системы. Для всех них мы будем использовать аутентификацию Windows и русский язык:

Имя	**Полное имя**	**Роли**	**Основной интерфейс**
Администратор	Администратор	Администратор	Администратор
Деловой	Деловой Иван Сергеевич	Только просмотр	Руководитель
Гусаков	Гусаков Николай Дмитриевич	Мастер	Мастер
Симонов	Симонов Валерий Михайлович	Мастер	Мастер
Назарова	Назарова Инна Семеновна	Расчетчик Бухгалтер	Бухгалтер

Смирнова	Смирнова Эльвира Денисовна	Расчетчик	Бухгалтер

Обратите внимание, что главному бухгалтеру Назаровой поставлены в соответствие две роли: « Расчетчик» и « Бухгалтер», поскольку она должна иметь возможность не только вести бухгалтерский учет, но и рассчитывать зарплату.

Список пользователей, зарегистрированных в системе, можно получить, выполнив команду **Действия | Вывести список.**

Теперь вы можете зайти в нашу информационную базу под различными пользователями и посмотреть, чем отличаются внешний вид интерфейса и возможности различных пользователей.

Контрольные вопросы

1. Опишите, как создать список пользователей системы и определить их права.
2. Чем аутентификация средствами 1С: Предприятия отличается от аутентификации Windows.
3. Какой командой вызывается на экран список пользователей?.

Лабораторная работа №8 Разработка классов.

Цель работы Получение навыков в разработке программ с использованием классов.

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Требуется создать небольшую иерархию классов, описывающих основные графические примитивы: эллипс, окружность, прямоугольник, квадрат.

Библиотека должна включать следующий **минимальный** набор классов:

- корневой класс фигур;
- дочерний класс эллипсов, наследующий классу фигур (первый уровень наследования);
- дочерний класс прямоугольников, наследующий классу фигур (первый уровень наследования);
- дочерний класс окружностей, наследующий классу эллипсов (второй уровень наследования);
- дочерний класс квадратов, наследующий классу прямоугольников (второй уровень наследования).

Корневой класс фигур должен определять общие свойства и поведение всех объектов-примитивов:

1. координаты базовой точки примитива;
2. конструктор;
3. методы доступа;
4. абстрактные метод прорисовки **Draw**;
5. абстрактный метод перемещения **MoveTo.**

В каждом классе необходимо реализовать:

- конструктор;
- методы прорисовки фигуры;
- метод удаления выбранной фигуры;
- метод перемещения выбранной фигуры.

При реализации метода перемещении необходимо предусмотреть проверку невозможности выхода фигуры за границы области рисования.

Кроме того, классы должны содержать методы, уникальные только для соответствующего поддерева:

- изменение радиуса окружности;
- изменение линейных размеров прямоугольника.

Вся библиотека оформляется в виде одного или нескольких модулей, которые подключаются к основной программе для демонстрации возможностей этой библиотеки.

Добавить в созданную библиотеку классов для графических примитивов следующий набор классов:

- дочерний класс многоугольников, наследующий классу фигур (первый уровень наследования),
- дочерний класс треугольников, наследующий классу многоугольников (второй уровень наследования).

Реализовать класс сложной фигуры, состоящей из простых фигур из иерархии классов. Вид сложной фигуры выбирается согласно индивидуальному варианту, определенного преподавателем.

В каждом классе необходимо реализовать:

- конструктор;
- методы прорисовки фигуры;
- метод удаления выбранной фигуры;
- метод перемещения выбранной фигуры.

Выполнить модификацию созданной ранее библиотеки классов для графических примитивов на основе использования механизма виртуальных методов. Цель – устранение ситуации повторения в каждом классе одинаковых методов перемещения и тем самым реализация универсального метода для перемещения любых графических объектов.

Контрольные вопросы

1 Дайте определение терминам «класс» и «объект». Как соотносятся эти понятия между собой?

2 Приведите синтаксис создания объекта в общем виде. Проиллюстрируйте его фрагментом программы.

3 Какие члены класса содержат код?

4 Какие члены класса содержат данные?

5 Перечислите модификаторы доступа к членам класса.

6 Объясните принцип инкапсуляции и его применение к классам.

Лабораторная работа №9 Обработка исключений.

Цель работы – ознакомиться с понятием исключительных ситуаций и способами их обработки, изучить методы создания классов исключений и их объектов, способов формирования исключений в программе.

Чтобы определить блок кода, подлежащий выполнению по выходу из try/catch-блока, включите в конец try/catch-последовательности блок finally.

Общая форма записи последовательности try/catch-блоков, содержащей блок finally, выглядит следующим образом.

```
try {
// Блок кода, предназначенный для обработки ошибок.
}
catch (ExcepTypel exOb) {
// Обработчик для исключения типа ExcepTypel.
}
catch (ExcepType2 exOb) {
// Обработчик для исключения типа ExcepType2.
finally {
// Код завершения обработки исключений.
}
```

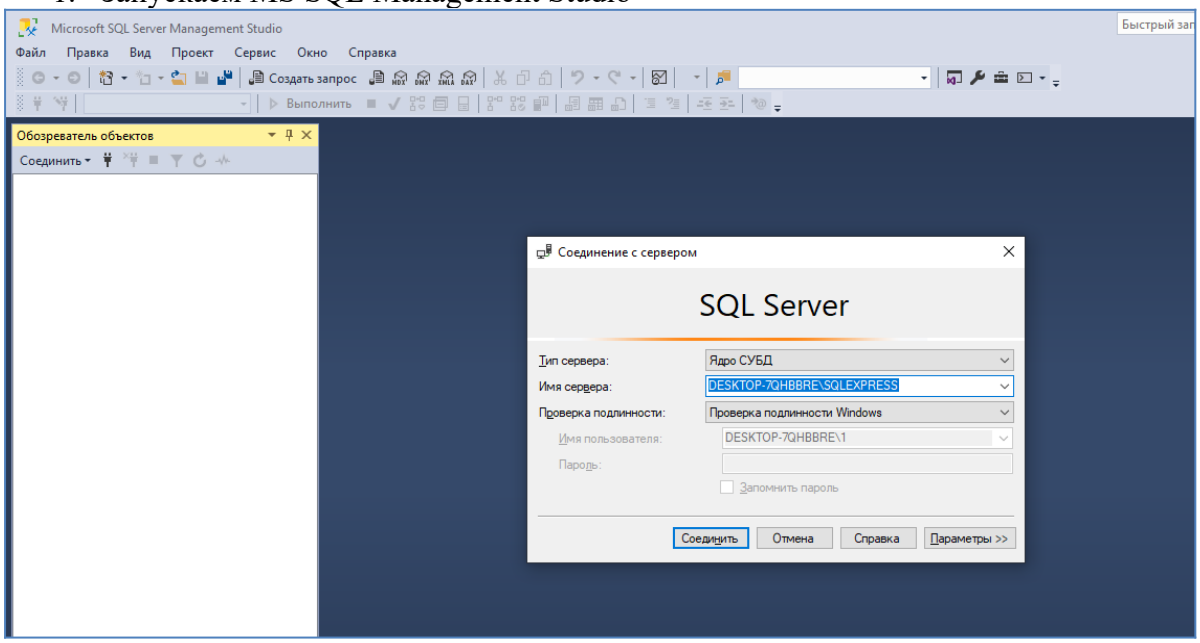
Блок finally будет выполнен после выхода из try/catch-блока, независимо от условий его выполнения. Другими словами, при нормальном завершении try-блока или в условиях возникновения исключения содержимое finallyблока будет безусловно отработано. Блок finally выполнится и в том случае,

если любой код внутри try-блока или любая из его catch-инструкций определены внутри метода

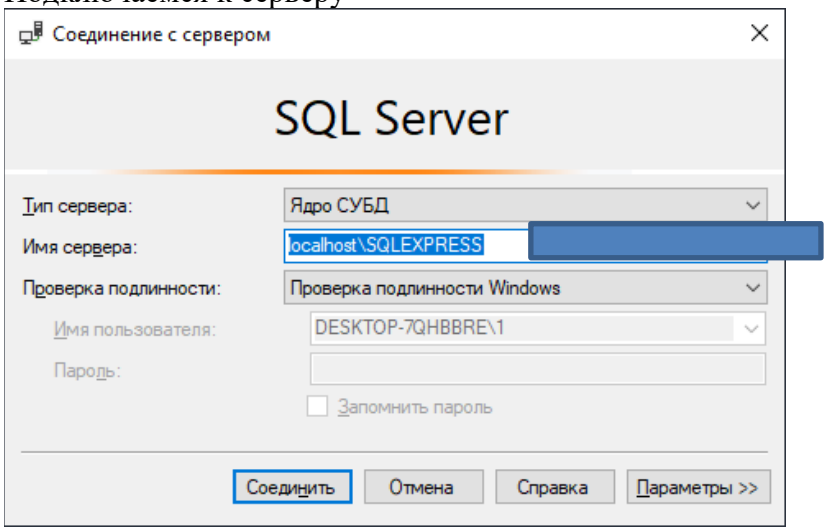
Контрольные вопросы
1. Что такое исключительная ситуация? Приведите примеры программных исключений.
2. Как выделить в программе контролируемый блок?
3. С помощью какого ключевого слова осуществляется генерация исключительных ситуаций? Укажите существующие формы этого оператора.
4. Перечислите известные Вам формы описания обработчиков исключений.
5. Каким образом осуществляется обработка исключений во вложенных контролируемых блоках?

Лабораторная работа №10 Создание базы данных.
Цель работы: научиться создавать базы данных с использованием Microsoft SQL Server.
Задание: Создать базу данных «Моя библиотека», которая содержит информацию о книгах в библиотеке.
Ход работы:

1. Запускаем MS SQL Management Studio

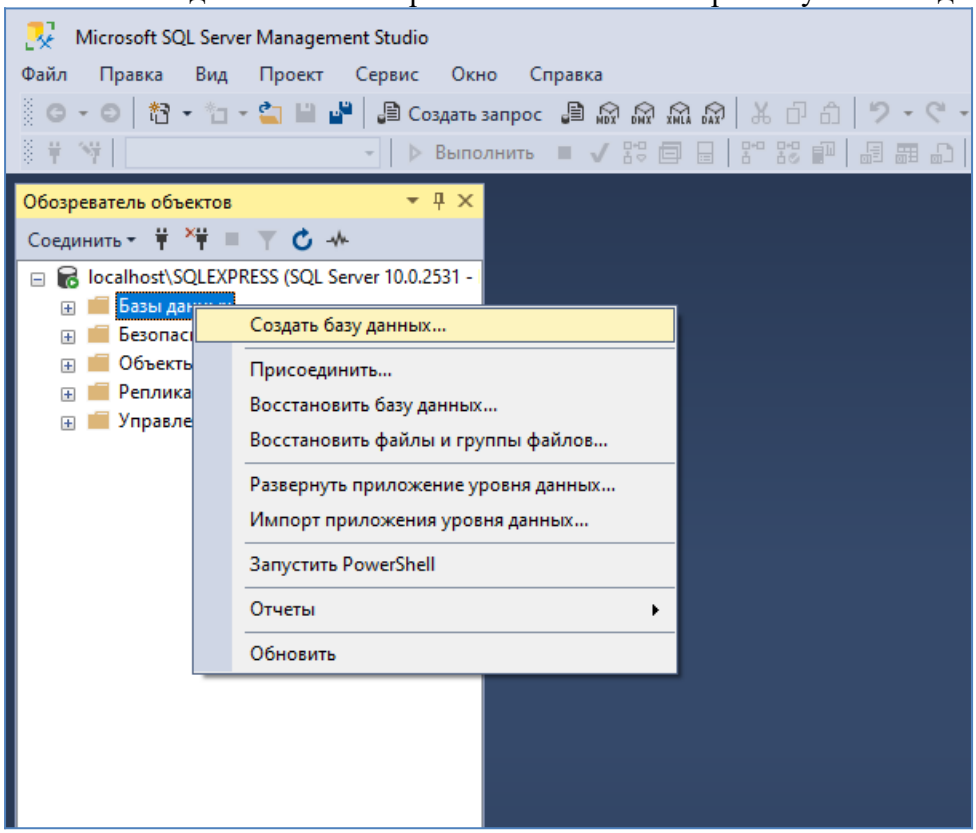


Подключаемся к серверу

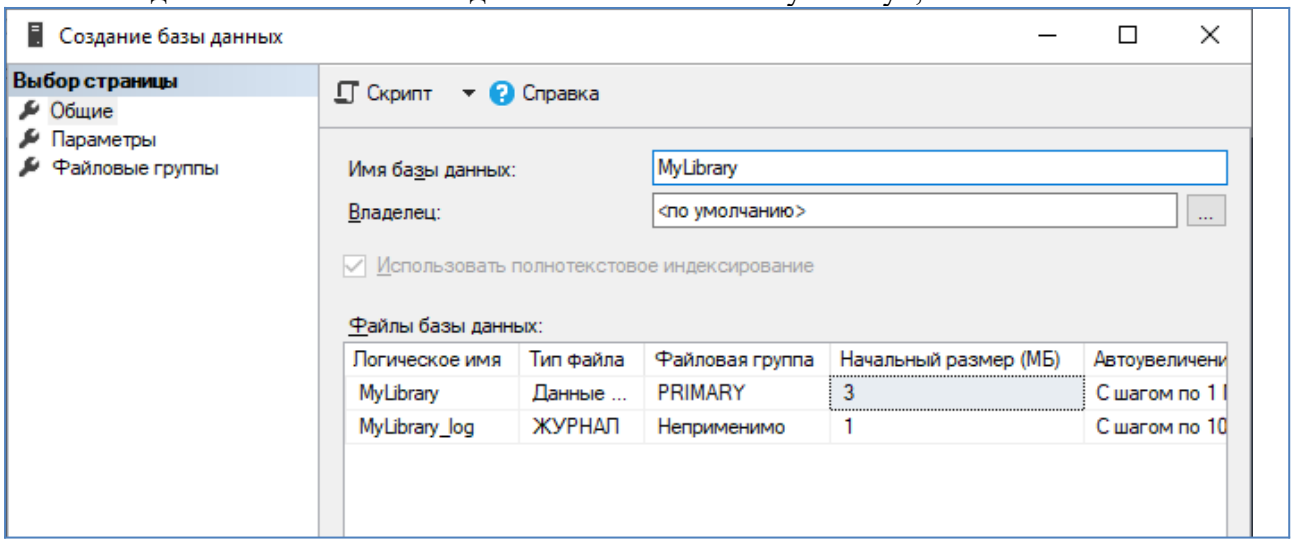


Жмем кнопку «Соединить» (Connect). Открывается список баз данных сервера.

2. Создаем новую базу данных.
   Для этого в окне «Обозреватель объектов» жмем правой кнопкой мыши на объект «Базы данных» и в открывшемся меню выбираем пункт «Создать базу данных…»



Вводим в поле «Имя базы данных» название «MyLibrary», нажимаем ОК



3. Определяем основные сущности базы данных и создаем таблицы. (смотри всю информацию о таблицах БД в папке **Ресурсы: файл ERD и файл Словарь данных**)

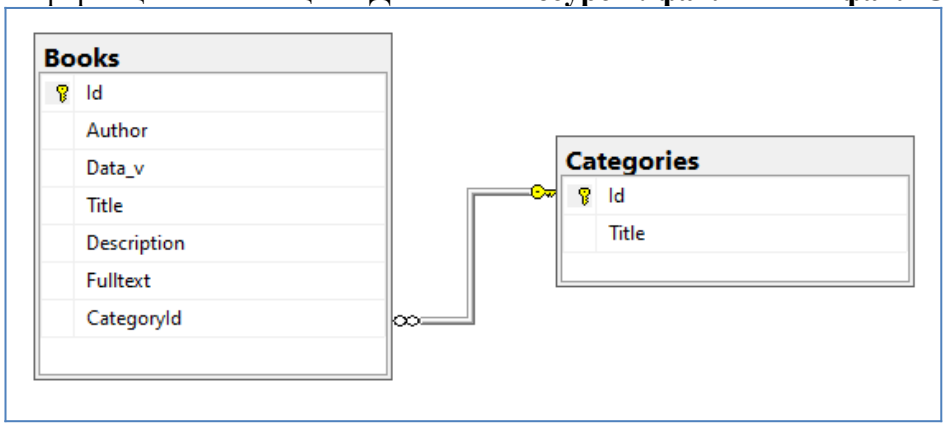
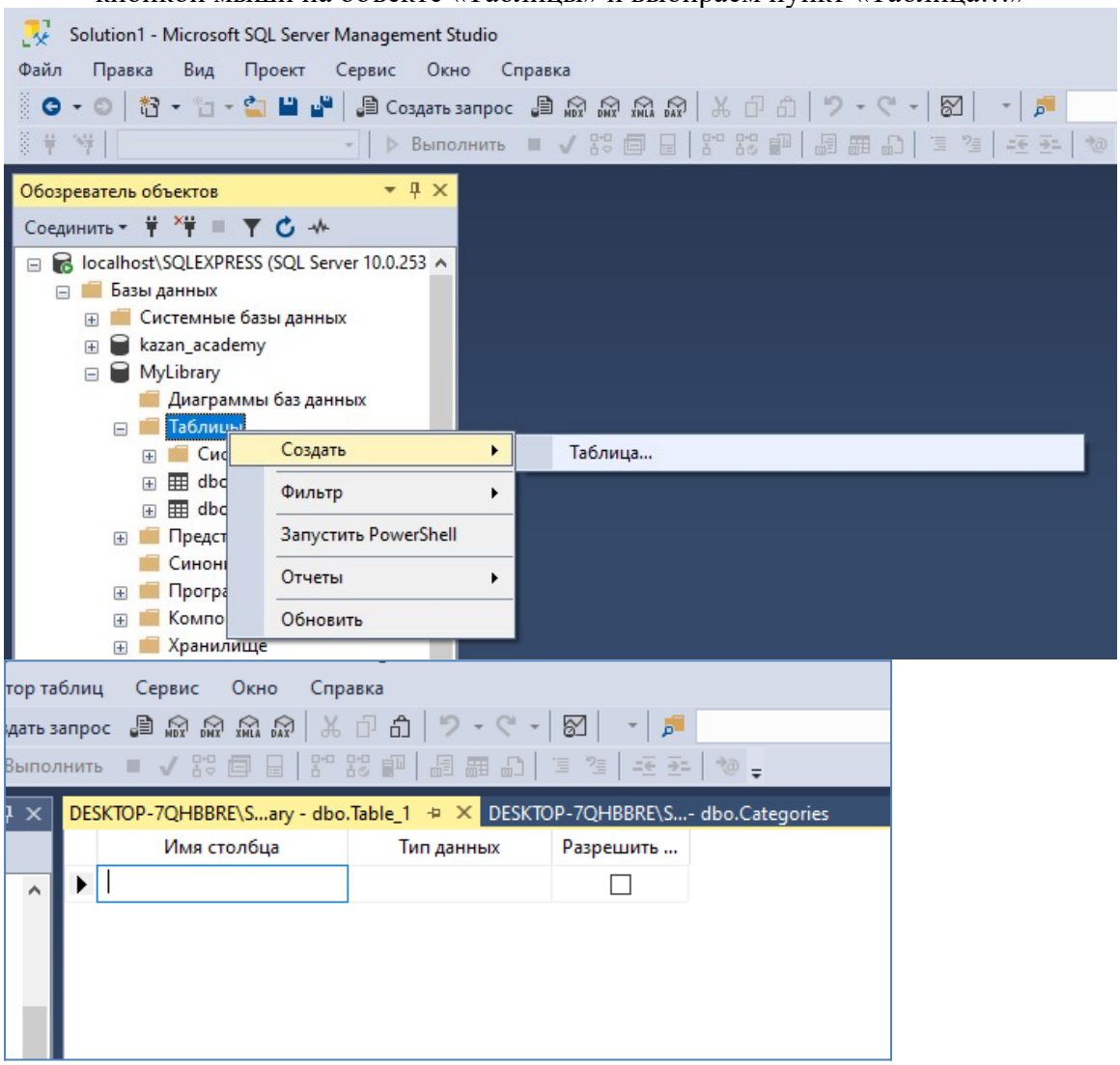


Рисунок 1. ERD-диаграмма

Создаем таблицы.

Для этого в обозревателе объектов в созданной базе MyLibrary нажимаем правой кнопкой мыши на объекте «Таблицы» и выбираем пункт «Таблица…»



4. Создаем столбцы и устанавливаем типы данных как на скриншоте. В третьем столбце есть маркер, отвечающий за обязательность поля. В случае, если мы отметим его галочкой, поле будет необязательным при заполнении в таблицу

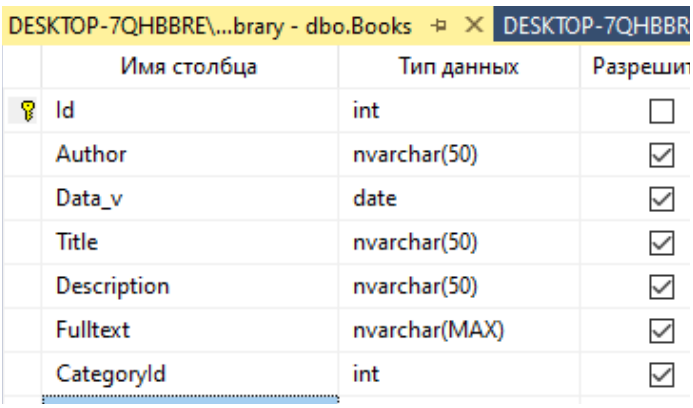
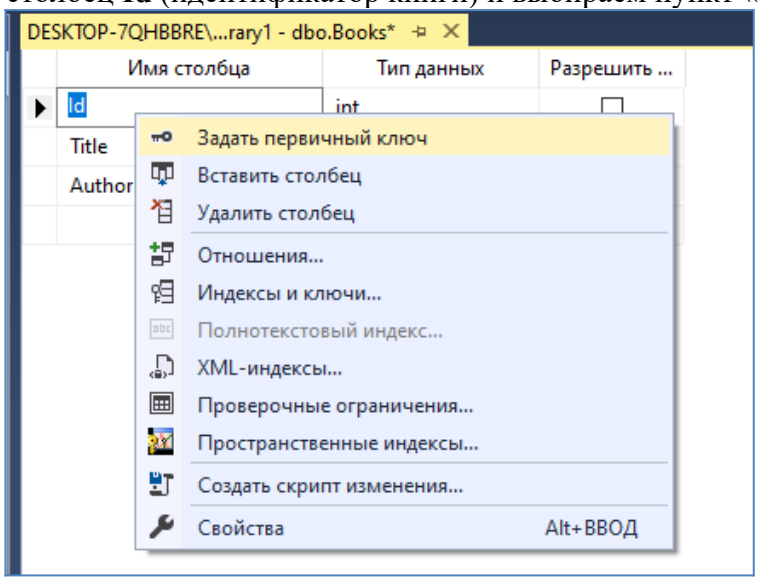


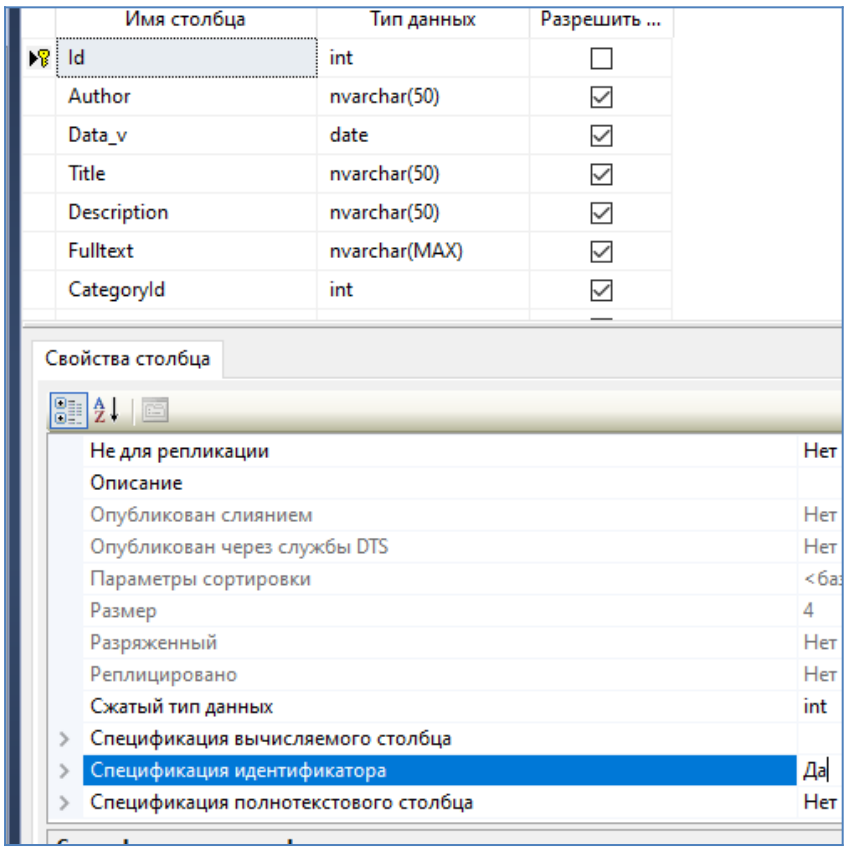
Рисунок 2. Конструктор таблиц

5. Определим первичный ключ для столбца. Нажимаем правой кнопкой мыши на столбец **Id** (идентификатор книги) и выбираем пункт «Задать первичный ключ»

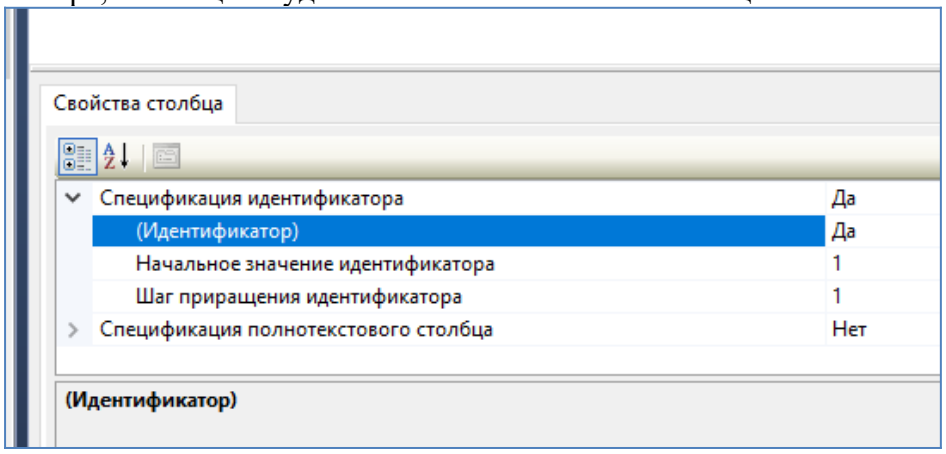


Важно! Первичный ключ — поле, которое уникально характеризует запись (строку) в таблице

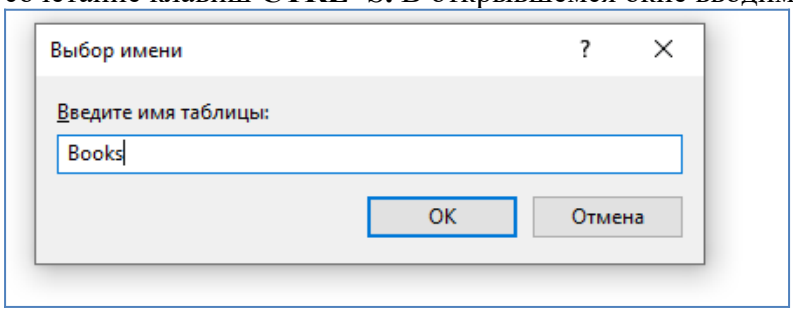
6. Определяем свойства столбца. Для ключевого столбца **Id** в нижней вкладке «**Свойства столбца**» выберем свойство «**Спецификация идентификатора».**



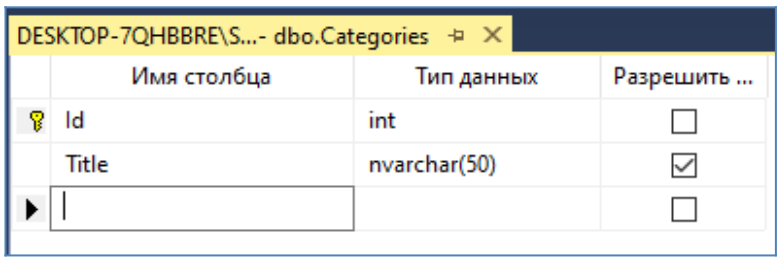
По двойному щелчку раскрываем свойство и в поле «Идентификатор» выбираем «Да». Теперь, столбец Id будет заполняться автоматически целыми числами (1,2,3,4…).



Сохраняем таблицу. Выбираем меню «Файл-> Сохранить таблицу…» или нажимаем сочетание клавиш **CTRL+S.** В открывшемся окне вводим название таблицы “**Books”**



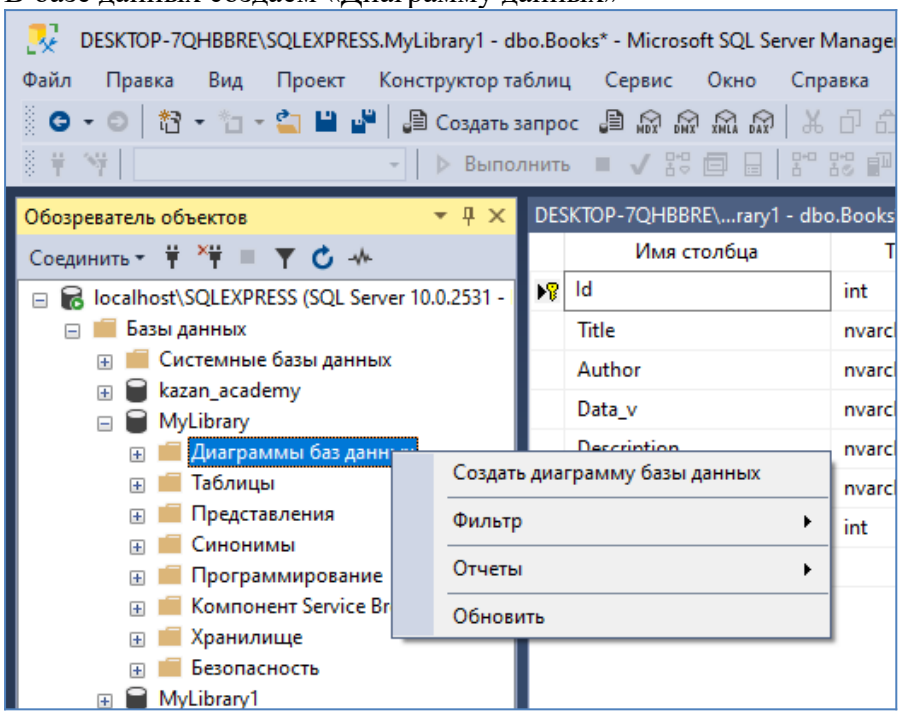
7. Аналогично создайте таблицу **Categories.** Не забудьте определить первичный ключ и спецификацию идентификатора для столбца **Id.**



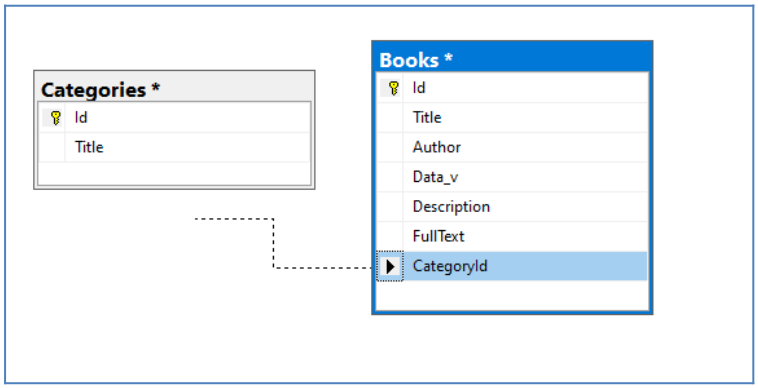
8. Устанавливаем связи между таблицами.

Определим связь ОДИН-КО-МНОГИМ. Чтобы связать таблицу книги с категориями книг в таблицу **Books** необходимо добавить специальное поле -ВНЕШНИЙ КЛЮЧ (в нашем случае это поле **CategoryId**), который по типу совпадает с тем, что является первичным ключом в таблице **Categories**.

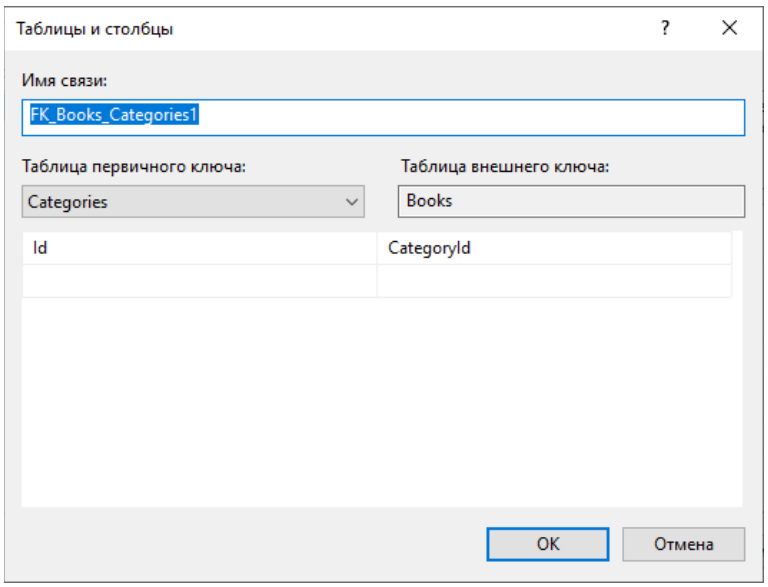
В базе данных создаем «Диаграмму данных»

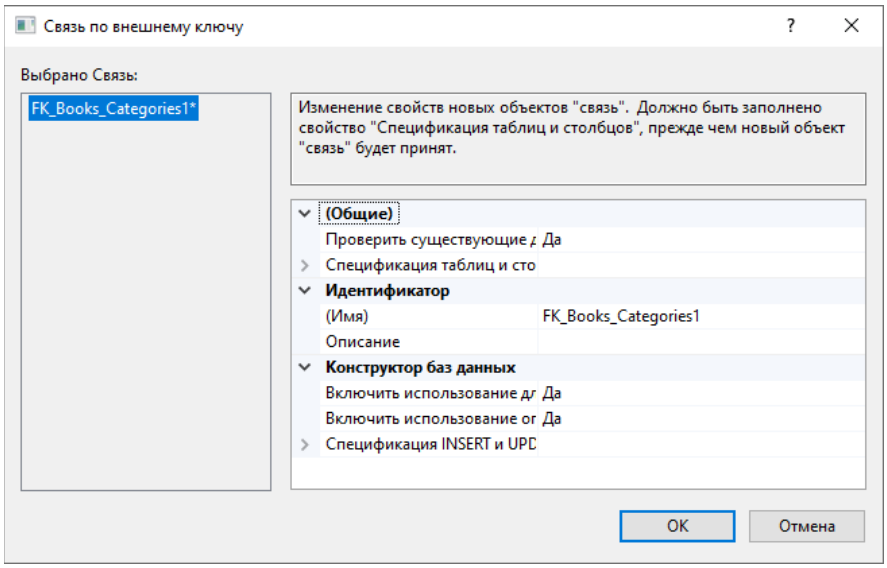


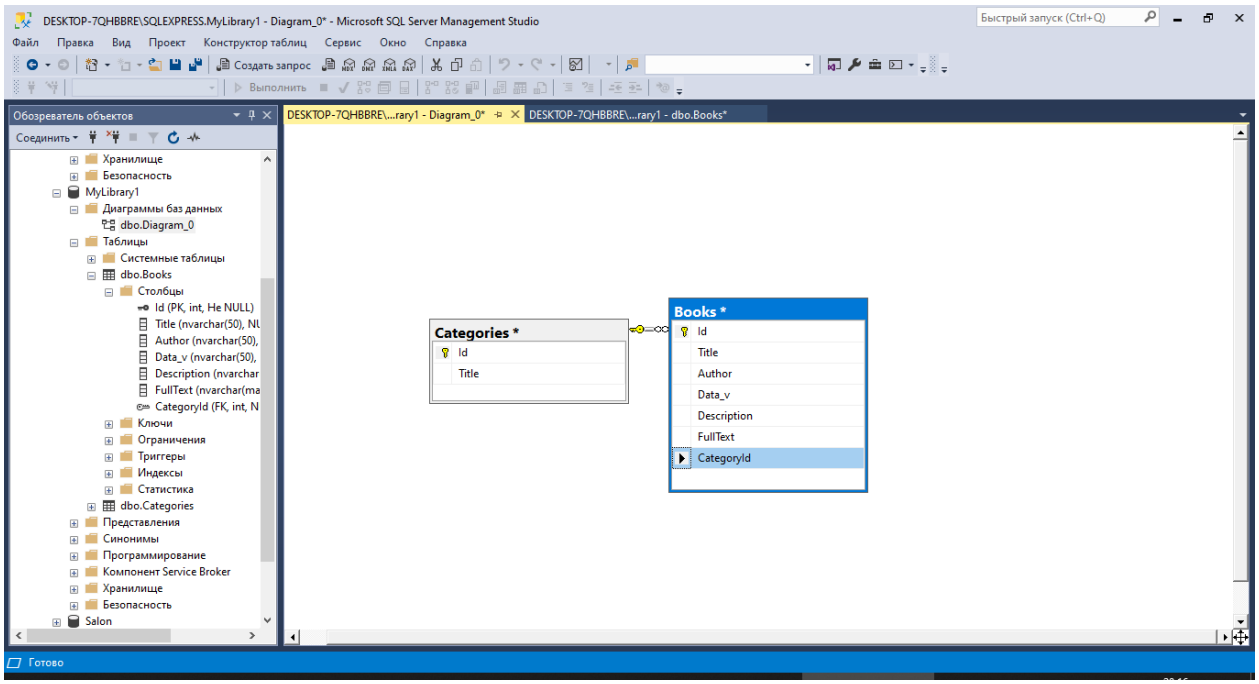
Добавляем на диаграмму таблицы. От поля-внешнего ключа **CategoryId** проводим связь к первичному ключу **Id** таблицы Catgories.



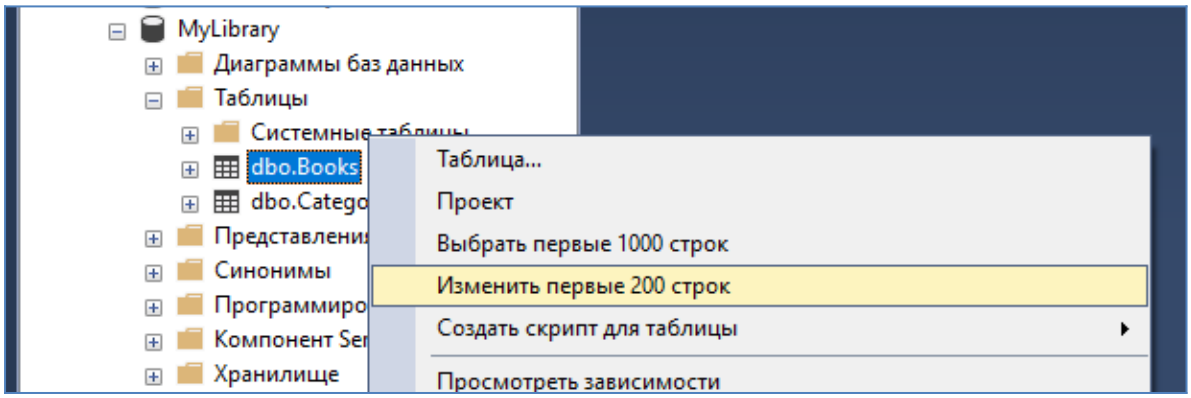
Открывается окно «Таблицы и связи», в котором внешний и первичный ключ должны быть как на скриншоте ниже. Нажимаем ОК.


Таблицы и столбцы
Имя связи:
FK_Books_Categories1
Таблица первичного ключа:
Categories
Таблица внешнего ключа:
Books
Id
CategoryId
OK
Отмена


Связь по внешнему ключу
Выбрано Связь:
FK_Books_Categories1*
Изменение свойств новых объектов "связь". Должно быть заполнено свойство "Спецификация таблиц и столбцов", прежде чем новый объект "связь" будет принят.
(Общие)
Проверить существующие д
Да
Спецификация таблиц и сто
Идентификатор
(Имя)
FK_Books_Categories1
Описание
Конструктор баз данных
Включить использование дл
Да
Включить использование ог
Да
Спецификация INSERT и UPD
OK
Отмена


DESKTOP-7QHBBRE\SQLEXPRESS.MyLibrary1 - Diagram_0* - Microsoft SQL Server Management Studio
Файл
Правка
Вид
Проект
Конструктор таблиц
Сервис
Окно
Справка
Обозреватель объектов
Соединить
Хранилище
Безопасность
MyLibrary1
Диаграммы баз данных
dbo.Diagram_0
Таблицы
Системные таблицы
dbo.Books
Столбцы
Id (PK, int, He NULL)
Title (nvarchar(50), NU
Author (nvarchar(50),
Data_v (nvarchar(50),
Description (nvarchar
FullText (nvarchar(ma
CategoryId (FK, int, N
Ключи
Ограничения
Триггеры
Индексы
Статистика
dbo.Categories
Представления
Синонимы
Программирование
Компонент Service Broker
Хранилище
Безопасность
Salon
DESKTOP-7QHBBRE\...rary1 - Diagram_0*
DESKTOP-7QHBBRE\...rary1 - dbo.Books*
Categories *
Id
Title
Books *
Id
Title
Author
Data_v
Description
FullText
CategoryId
Готово

9. Заполняем таблицы данными.
   Для этого нажимаем правой кнопкой на таблицу Books и в контекстном меню выбираем пункт «Изменить первые 200 строк».



Заполняем таблицы правильными данными

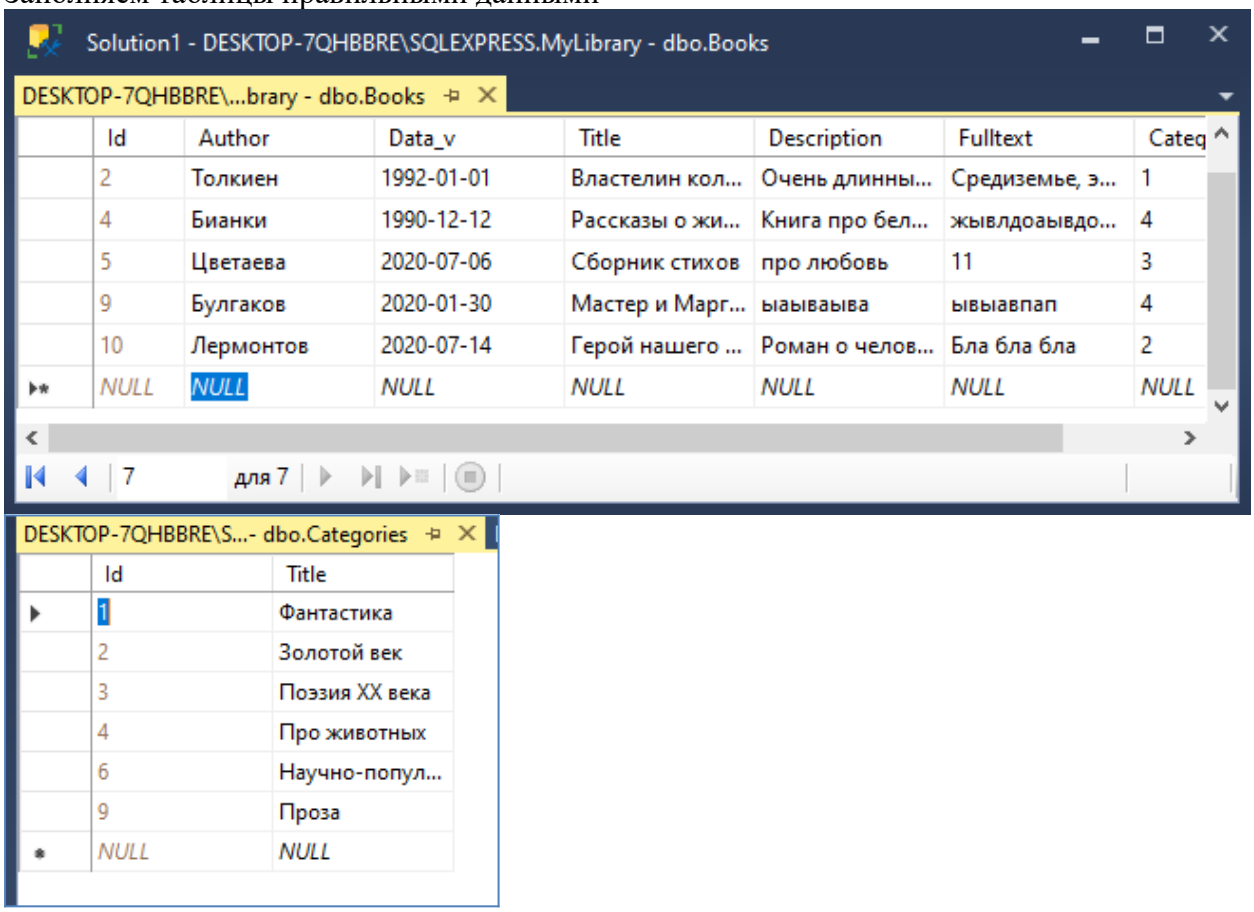
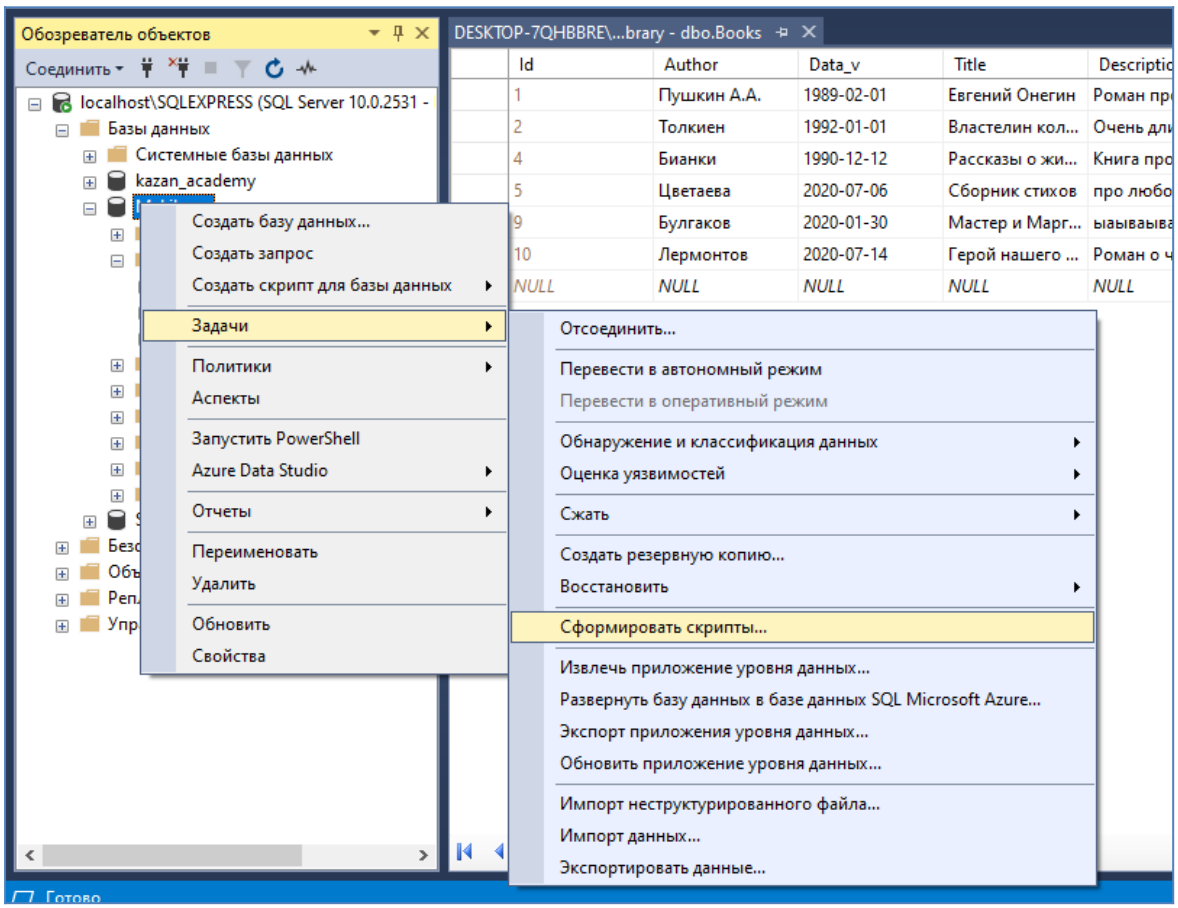


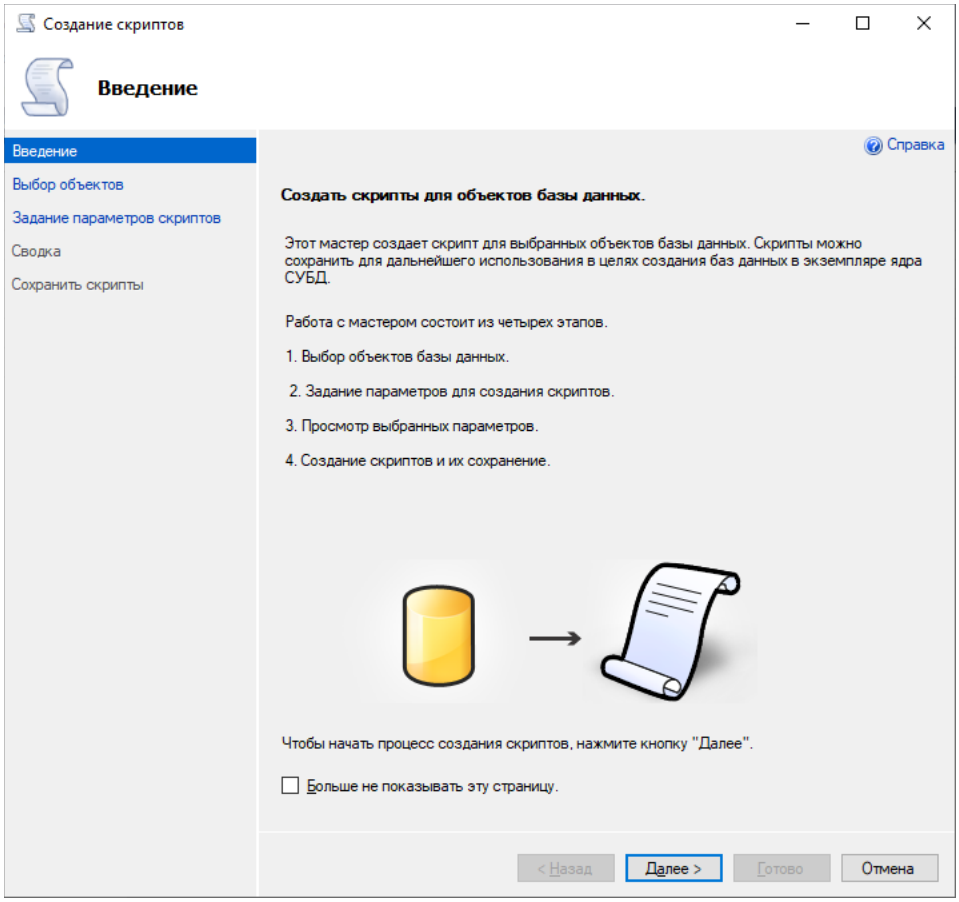
Рисунок 3. Заполненные таблицы Books и Categories

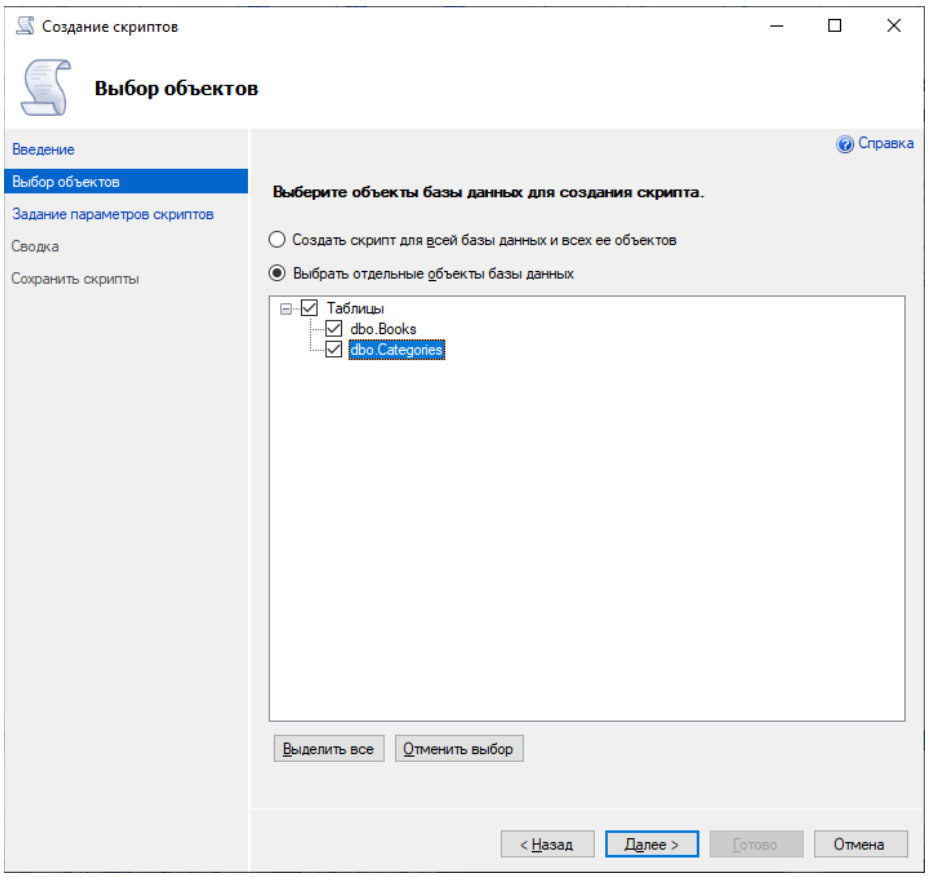
10. Сохранение БД и создание скрипта

Все созданные базы данных хранятся на сервере. Чтобы перенести базу данных на другой сервер, необходимо правильно ее сохранить. Один из методов переноса - создание скрипта базы данных:

Для этого в окне «Обозреватель объектов» нажать правой кнопкой мыши на название базы данных MyLibrary и в открывшемся контекстном меню выбрать «Задачи> Сформировать скрипты…»


Обозреватель объектов
Соединить
localhost\SQLEXPRESS (SQL Server 10.0.2531 -
Базы данных
Системные базы данных
kazan_academy
DESKTOP-7QHBBRE\...brary - dbo.Books
Id
Author
Data_v
Title
Descriptio
1
Пушкин А.А.
1989-02-01
Евгений Онегин
2
Толкиен
1992-01-01
Властелин кол...
4
Бианки
1990-12-12
Рассказы о жи...
5
Цветаева
2020-07-06
Сборник стихов
9
Булгаков
2020-01-30
Мастер и Марг...
10
Лермонтов
2020-07-14
Герой нашего ...
NULL
Создать базу данных...
Создать запрос
Создать скрипт для базы данных
Задачи
Политики
Аспекты
Запустить PowerShell
Azure Data Studio
Отчеты
Переименовать
Удалить
Обновить
Свойства
Отсоединить...
Перевести в автономный режим
Перевести в оперативный режим
Обнаружение и классификация данных
Оценка уязвимостей
Сжать
Создать резервную копию...
Восстановить
Сформировать скрипты...
Извлечь приложение уровня данных...
Развернуть базу данных в базе данных SQL Microsoft Azure...
Экспорт приложения уровня данных...
Обновить приложение уровня данных...
Импорт неструктурированного файла...
Импорт данных...
Экспортировать данные...
Готово


Создание скриптов
Введение
Введение
Выбор объектов
Задание параметров скриптов
Сводка
Сохранить скрипты
Справка
Создать скрипты для объектов базы данных.
Этот мастер создает скрипт для выбранных объектов базы данных. Скрипты можно сохранить для дальнейшего использования в целях создания баз данных в экземпляре ядра СУБД.
Работа с мастером состоит из четырех этапов.
1. Выбор объектов базы данных.
2. Задание параметров для создания скриптов.
3. Просмотр выбранных параметров.
4. Создание скриптов и их сохранение.
Чтобы начать процесс создания скриптов, нажмите кнопку "Далее".
Больше не показывать эту страницу.
< Назад
Далее >
Готово
Отмена



В данном случае выполнение скрипта приведет к восстановлению структуры таблиц и переносу записей из вашей базы данных. Так что его можно использовать не только для переноса базы на другой сервер, но и для хранения резервных копий предыдущих состояний базы данных.

Лабораторная работа №11 Создание границ и ограничений.

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.

Выполнение 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. №1.

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).

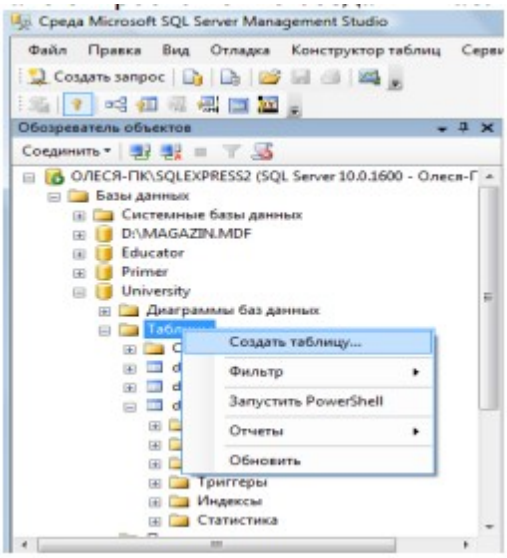


Рис. 1. Окно "Проводник" с перечнем таблиц

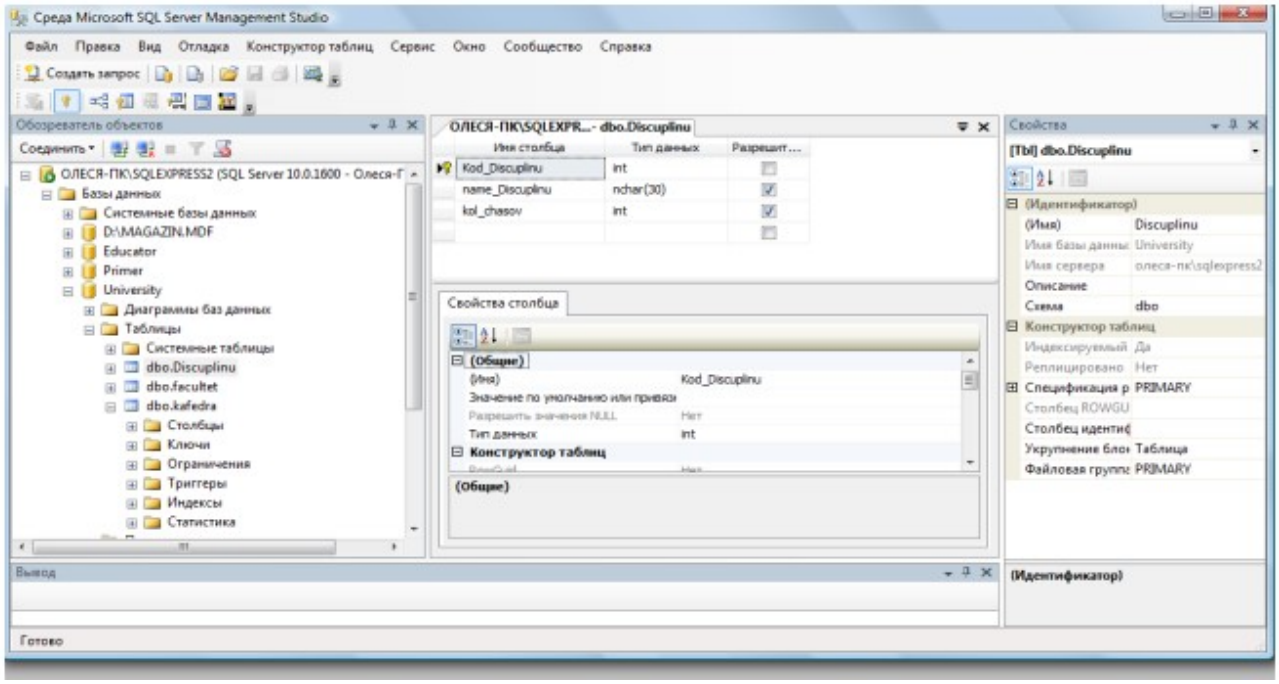


Рис. 2. Окно формирования таблицы в диалоговом режиме

Для ранее созданной базы данных по номеру варианта (созданной в лаб.работе №2) в СУБД SQL Server Management Studio:

1. Создайте все таблицы базы данных, ключи, ограничения и связи.
2. Каждая таблица должна иметь ограничение первичного ключа.
3. С помощью ограничений внешнего ключа должны быть заданы все имеющиеся связи между таблицами.
4. В зависимости от условий выданного задания в некоторых таблицах могут быть наложены дополнительные ограничения целостности на столбцы или должны быть разработаны вычисляемые поля.
5. Создайте диаграмму базы данных.
6. Заполните таблицы данными не менее 5 записей в каждой.
7. Создать текстовый отчет, в котором отобразить скриншоты результатов работы в СУБД SQL Server Management Studio (окно с базой данных с перечнем всех таблиц, проекты таблиц с перечнем столбцов, окна ограничений внешних ключей (создание), окно с перечнем ключей для

каждой таблицы, окна с данными для каждой таблицы, диаграмма базы данных).

Лабораторная работа №12 Разработка запросов.
Цель лабораторной работы Изучить используемый в реляционных СУБД оператор извлечения данных из таблиц. Получить навыки работы с оператором SELECT в программе 'SQL Server Managmant Studio'.
Для созданной базы данных, согласно номеру варианта, самостоятельно создать на языке Transact-SQL 15 многотабличных запросов:
- 1 запрос с использованием декартового произведения двух таблиц;
- 3 запроса с использованием соединения двух таблиц по равенству;
- 1 запрос с использованием соединения двух таблиц по равенству и условием отбора;
- 1 запрос с использованием соединения по трем таблицам;
- создать копии ранее созданных запросов на соединение по равенству на запросы с использованием внешнего полного соединения таблиц (JOIN).
- 1 запрос с использованием левого внешнего соединения;
- 1 запрос на использование правого внешнего соединения;
- 1 запрос с использованием симметричного соединения и удаление избыточности.
Все программные инструкции команд SQL сохранять в файлах с расширением *.sql в папке ФИО_студента/Лаб6.
Для каждого запроса сформулировать текстовое задание, которое должно быть выполнено к базе данных.
Создать текстовый отчет, в котором отобразить sql-команды разработанных запросов и скриншоты результатов работы из СУБД SQL Server Management Studio

Лабораторная работа №13 Добавление, удаление, изменение данных в таблицах
Цель: научить использовать операторы SQL для вставки, изменения и удаления данных из таблиц.

Теоретический материал: перед выполнением лабораторной работы рекомендуется изучить лекцию №4 «Выборка данных», из которой требуются знания оператора SELECT, необходимого для вставки данных в одну таблицу из другой, а также лекцию №9 «Операторы реляционной алгебры», в которой рассмотрены теоретические аспекты основных команд по управлению данными.

Требования к отчету: по результатам работы представить набор SQL-скриптов, решающих задачи из раздела «Самостоятельная работа», а также текстовый файл, необходимый для задания №2 из того же раздела.

Задание 1. В таблицу *Lect* добавьте запись с заданными значениями полей, значениями по умолчанию.

Указания к выполнению:
1. Зададим значения для трех полей: *Brdate*, *FirstName*, *LastName*.
2. Поле счетчика мы не будем задавать, т.к. оно будет сгенерировано автоматически.
3. Используйте следующий код для вставки новой записи:

```
INSERT INTO Lect (Brdate, FirstName, LastName)
VALUES (Convert(datetime,'01.01.1970',104), '...', '...')
```

Замечание. Выполнение команды **INSERT INTO Lect DEFAULT VALUES** приведет к

ошибке, т.к. не для всех полей указаны значения по умолчанию.

Задание 2. Добавьте в таблицу *Lect* данные из таблицы *Employee* базы данных *AdventureWorks2008*:

Указания к выполнению:

1. Будем заполнять 4 поля в таблице *Lect*: *FirstName*, *LastName*, *Brdate*, *HireDate*.
2. Для вставки сразу нескольких записей воспользуемся оператором SELECT:

```
INSERT INTO Lect (FirstName, LastName, Brdate, HireDate)
SELECT FirstName, LastName, BirthDate, HireDate
FROM AdventureWorks2008.HumanResources.Employee
INNER JOIN AdventureWorks2008.Person.Contact
ON Contact.ContactID = Employee.ContactID
```

Замечание. При использовании команды SELECT в таблицу вставляется несколько записей, являющихся результатом выборки данных других таблиц. Однако если хотя бы одна из вставляемых записей нарушает ограничения целостности таблицы, то вся команда INSERT отменяется.

Задание 3. Добавьте в таблицу *Students* данные из файла.

Указания к выполнению:

1. Создайте в корне диска C: текстовый файл *students.txt* со следующим содержимым:

```
1	Ivan	Petrov	1.1.1990
2	Petr	Ivanov	12.10.1992
3	Sergey	Kazakov	5.3.1991
```

Замечание. В качестве разделителя в строках используется TAB, а не пробелы.

2. Выполните следующий код:

```
BULK INSERT Students FROM 'c:\students.txt'
```

Замечание. Обратите внимание: несмотря на то, что мы указали поля *StudentID* в файле *students.txt*, они были проигнорированы при вставке. Это объясняется тем, что мы не использовали опцию KEEPIDENTITY.

Задание 4. Измените номер телефона у преподавателя по имени *King*.

Указания к выполнению:

1. Для поиска нужного имени воспользуемся оператором LIKE.
2. Установим для всех преподавателей *'King'* телефон равный 35-35-35*:*

```
UPDATE Lect
SET Phone = ‘35-35-35’
WHERE FirstName like ‘King’
```

Задание 5. Указать, что курсы по информатике должен читать преподаватель по имени *King*.

Указания к выполнению:

1. Для обновления записей в таблице *Course* нам потребуется использование подзапроса, возвращающего нужного преподавателя: SELECT LectID From Lect Where (FirstName like ‘King’).
2. При этом необходимо учитывать, что подзапрос должен возвращать единственное значение для каждой записи.
3. Код для обновления преподавателя по информатике будет выглядеть следующим образом:

```
UPDATE Course SET LectID =
(SELECT LectID From Lect Where (FirstName like ‘King’))
WHERE CourseName like ‘%[Ii]nform%’
```

Задание 6. Удалите всех студентов с фамилией *Ivanov*.

Указания к выполнению:

1. Для удаления записей по заданному условию необходимо использовать раздел

WHERE в операторе DELETE.

2. Выполните следующий код для удаления студентов:

```
DELETE Students
WHERE LastName = 'Ivanov'
```

Замечане. Удаление данных из таблицы осуществляется построчно, при этом можно выполнить удаление как одной записи, так и нескольких, удовлетворяющих некоторому условию. Однако для удаления всех записей некоторой таблицы рекомендуется команда: TRUNCATE TABLE ИмяТаблицы. При этом удаляются все записи, однако структура таблицы сохраняется, как и связанные с ней объекты. Данная команда выполняется быстрее команды DELETE, т.к. сервером регистрируется только освобождение страниц памяти, а не удаление каждой записи со страницы.

Самостоятельная работа

1. Заполните таблицы *Lect* и *Class* созданной базы данных.
2. Заполните таблицу *Students* созданной базы данных из текстового файла.
3. Заполните остальные таблицы созданной базы данных.
4. Удалите всех студентов, родившихся до 1990 года.

Лабораторная работа №14 Разработка хранимых процедур.

Изучение назначения представлений баз данных, синтаксиса и семантики команд языка Transact-SQL для их создания, изменения и удаления, системных хранимых процедур для получения информации о представлениях, а также приобретение навыков их создания с помощью графических средств утилиты EnterpriseManager и мастера CreateViewWizard.

При выполнении лабораторной работы необходимо:

- для заданной предметной области написать две хранимые процедуры и включить их в БД;
- составить отчет по лабораторной работе.

Пример выполнения работы

1. Создадим хранимую процедуру, которая выводит число заказов покупателя по вводимому в качестве параметра процедуры коду покупателя.

```
mysql> CREATE PROCEDURE num(OUT total INT, IN user_kod INT)
    -> BEGIN
    -> SELECT COUNT(*) INTO total FROM orders WHERE o_user_ID=user_kod;
    -> END
    -> //
Query OK, 0 rows affected (0.00 sec)
```

Параметр *total* является выходным, его значение равно числу заказов покупателя, код которого записывается во входной параметр *user_kod.* Процедура считает все строки, где код клиента совпадает с параметром *user_kod.*

До вызова процедуры присваиваем параметру процедуры значение кода клиента. Затем вызываем процедуру оператором *CALL*. Для вывода результата можно воспользоваться оператором *SELECT*.

```
mysql> SET @user_kod=3//
Query OK, 0 rows affected (0.00 sec)

mysql> CALL num(@total,@user_kod)//
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @total,@user_kod//
+--------+-----------+
| @total | @user_kod |
+--------+-----------+
| 2      | 3         |
+--------+-----------+
1 row in set (0.00 sec)
```

2. Создадим хранимую процедуру, которая записывает в новую таблицу *fevral* все заказы, сделанные в феврале 2009 г. Предварительно необходимо создать новую пустую таблицу *fevral* со структурой, аналогичной структуре таблицы *orders*.

```
mysql> CREATE TABLE fevral(
    ->   f_order_ID int(6) NOT NULL,
    ->   f_o_user_ID int NOT NULL,
    ->   f_o_book_ID int NOT NULL,
    ->   f_o_time datetime NOT NULL,
    ->   f_o_number int(6) NOT NULL,
    ->   PRIMARY KEY (f_order_ID)
    -> )TYPE=InnoDB//
Query OK, 0 rows affected, 1 warning (0.08 sec)
```

Хранимая процедура *ord_fevr ( )* использует курсор *curf*, который в цикле читает данные из таблицы *orders* и добавляет их в таблицу *fevral*.

```
mysql> CREATE PROCEDURE ord_fevr()
    -> BEGIN
    -> DECLARE id int;
    -> DECLARE _end int DEFAULT 0;
    -> DECLARE userID int;
    -> DECLARE bookID int;
    -> DECLARE tim datetime;
    -> DECLARE num int;
    -> DECLARE curf CURSOR FOR SELECT * FROM orders
    -> WHERE o_time BETWEEN '2009.02.01' AND '2009.02.28';
    -> DECLARE CONTINUE HANDLER FOR NOT FOUND SET _end=1;
    -> OPEN curf;
    -> wet: LOOP
    -> FETCH curf INTO id,userID,bookID,tim,num;
    -> IF _end THEN LEAVE wet;
    -> END IF;
    -> INSERT INTO fevral VALUES(id,userID,bookID,tim,num);
    -> END LOOP wet;
    -> CLOSE curf;
    -> END
    -> //
Query OK, 0 rows affected (0.03 sec)
```

Вызов процедуры осуществляется оператором *CALL*. Для просмотра результата выполнения процедуры используем полную выборку из таблицы *fevral*.

```
mysql> CALL ord_fevr//
Query OK, 1 row affected (0.05 sec)

mysql> SELECT * FROM fevral//
+------------+-------------+-------------+---------------------+-----------
| f_order_ID | f_o_user_ID | f_o_book_ID | f_o_time            | f_o_number
+------------+-------------+-------------+---------------------+-----------
|          2 |           6 |          10 | 2009-02-10 09:40:29 |          2
|          3 |           1 |          20 | 2009-02-18 13:41:05 |          4
+------------+-------------+-------------+---------------------+-----------
2 rows in set (0.00 sec)
```

Лабораторная работа №15 Разработка технической документации.
Цель работы: научиться составлять техническую документацию программного продукта.

Ход работы

Задание 1. Для готового программного модуля, создать руководство пользователя программного продукта.

Документация должна содержать необходимые сведения по установке, обеспечению надёжной работы продукта, справочное пособие для пользователя, демонстрационные версии, примеры документов, создаваемых при помощи данного программного продукта, обучающие программы.

Время выполнения: 100 минут

Перечень объектов контроля и оценки

Контрольные вопросы:

1. Что такое ТЗ?
2. Что такое руководство пользователя?
3. Что такое руководство администратора?
4. Что такое руководство по техническому обслуживанию?
5. Что относиться к эксплуатационным документам?

Лабораторная работа №16 Разработка пользовательской документации.

Цель работы:

1. Изучите теоретические сведения. Ознакомьтесь с постановкой задачи. Подготовить отчет в тетради. Ответить на контрольные вопросы.

Теоретические сведения.

Программная документация, включает:

1. техническое задание (назначение, область применения программы, требования, предъявляемые к программе);
2. текст программы (запись программы с необходимыми комментариями);
3. описание программы (сведения о логической структуре и функционировании программы);
4. пояснительная записка (схема алгоритма, общее описание алгоритма и/или функционирования программы, обоснование принятых решений);
5. эксплуатационные документы.

К эксплуатационным документам относят:

- описание применения (сведения о назначении программы, области применения, применяемых методах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств);
- руководство системного программиста (сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения);
- руководство программиста (сведения для эксплуатации программы);
- руководство оператора (сведения для обеспечения общения оператора с вычислительной системой в процессе выполнения программы);
- описание языка (описание синтаксиса и семантики языка);
- руководство по техническому обслуживанию (сведения для применения тестовых и диагностических программ при обслуживании технических средств)

Основная часть программной документации составляется на стадии рабочего проекта. Необходимость того или иного документа определяется на этапе составления технического задания. Допускается объединять отдельные виды документов.

Эксплуатационный документ "Описание языка" включается в программную документацию, если разработанный программный продукт реализует некий язык программирования, управления заданиями, организации вычислительного процесса и т. п.

Эксплуатационный документ "Руководство по техническому обслуживанию" включается в программную документацию, если разработанный программный продукт требует использования тестовых или диагностических программ.

Описание применения

Документ "Описание применения" относится к эксплуатационным документам и состоит из следующих разделов:

- назначение программы (возможности, основные характеристики, ограничения области применения);
- условия применения (требования к техническим и программным средствам, общие характеристики входной и выходной информации, а также требования и условия организационного, технического и технологического характера);
- описание задачи (указываются определения задачи и методы её решения);
- входные и выходные данные.
- Руководство программиста

Документ "Руководство программиста" относится к эксплуатационным документам и включается в программную документацию, если разработанный программный продукт требует обслуживания программистом. Документ состоит из следующих разделов:

- назначение и условия применения программы (назначение и функции программы, сведения о технических и программных средствах, обеспечивающих выполнение данной программы);
- характеристики программы (временные характеристики, режимы работы, средства контроля правильности выполнения и т. п.);
- обращение к программе (способы передачи управления и параметров данных);
- входные и выходные данные (формат и кодирование);
- сообщения (тексты сообщений, выдаваемых программисту или оператору в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

Руководство оператора

Документ "Руководство оператора" относится к эксплуатационным документам и состоит из следующих разделов:

- назначение программы (информация, достаточная для понимания функций программы и её эксплуатации);
- условия выполнения программы (минимальный и/или максимальный набор технических и программных средств и т. п.);
- выполнение программы (последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы; описываются функции, форматы и возможные варианты команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, а также ответы программы на эти команды);
- сообщения оператору (тексты сообщений, выдаваемых оператору в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

Ход работы

Задание 1. Для готового программного модуля, создать руководство пользователя программного продукта.

Документация должна содержать необходимые сведения по установке, обеспечению надёжной работы продукта, справочное пособие для пользователя, демонстрационные версии, примеры документов, создаваемых при помощи данного программного продукта, обучающие программы.

Время выполнения: 100 минут

Перечень объектов контроля и оценки

Контрольные вопросы:
1. Что такое ТЗ?
2. Что такое руководство пользователя?
3. Что такое руководство администратора?
4. Что такое руководство по техническому обслуживанию?
5. Что относиться к эксплуатационным документам?

Критерии оценки выполнения практических заданий:
«5» – все задания выполнены правильно;
«4» – наблюдались неточности при выполнении работы;
«3» – наблюдались ошибки при выполнении работы;
«2» – работа выполнена менее 50 %.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РЕСПУБЛИКИ ДАГЕСТАН
Частное профессиональное образовательное учреждение
«РЕСПУБЛИКАНСКИЙ ПОЛИПРОФЕССИОНАЛЬНЫЙ КОЛЛЕДЖ»
(ЧПОУ «Республиканский полипрофессиональный колледж»)

Тесты

по дисциплине Программные решения для бизнеса

Тема 1.1 Структурный подход в моделировании предметной области

1
Сколько функциональных блоков рекомендуется располагать на диаграмме IDEF0 (НЕ контекстного уровня)?
a) сколько угодно
b) 4
c) 4-8
d) 1

2
Функциональный блок в IDEF0 обозначает
a) объект
b) логическое условие
c) бизнес-процесс или набор процессов
d) исполнительный механизм

3
Объект «CRM-система» для функционального блока «Обработать заявку клиента» будет
Впишите ответ «……»

4
Декомпозировать функциональный блок IDEF0 можно в нотации:
a) IDEF0, DFD, EPC, BPMN, если позволяют возможности CASE-инструмента (системы бизнес-моделирования)
b) только структурные нотации, такие как IDEF0, IDEF1x, DFD
c) в абсолютно любой нотации, включая UML
d) только IDEF0

5
Сколько функциональных блоков содержит контекстная диаграмма?
a) 6-8
b) 4
c) 1
d) сколько угодно

6
Какие обязательные стрелки должны присутствовать у каждого функционального блока?
a) Управление
b) Выход
c) Вход, Выход, Управление, Механизм
d) Вход и Выход

Тема 1.2 Объектно-ориентированное моделирование системы

1 – Вопрос с одним вариантом ответа
Основными понятиями объектно – ориентированного подхода являются?
a) Класс и объект
b) Типизация и устойчивость
c) Инкапсуляция и модульность

2 – Вопрос с одним вариантом ответа
Индивидуальность – это …
a) Поведения объекта
b) Свойство объекта
c) Действие объекта

3
Основной элемент объектной модели.
Впишите ответ «…….»

4 – Вопрос с одним вариантом ответа
Сколько дополнительных элементов существует в объектной модели ?
a) 5
b) 8
c) 3

5 – Вопрос с одним вариантом ответа
Типизация – это
a) Процесс отделения друг от друга отдельных элементов
b) Свойство объекта существовать во времени
c) Ограничение, накладываемое на класс объектов

Тема 1.3 Спецификации языка C#. Технология .NET

1 – вопрос с несколькими вариантами ответов
Типы данных в языке C# принято классифицировать как:
a) фигурные
b) определенные программистом

c) простые
d) встроенные (базисные)
e) сложные (структурные)
f) правильных ответов нет

2 – Вопрос с одним вариантом ответа
Укажите правильный порядок следования приоритетов бинарных операций:
a) арифметические, логические, отношения
b) отношения, логические, арифметические
c) арифметические, отношения, логические
d) правильных ответов нет

3 – Вопрос с одним вариантом ответа
Отличительной особенностью каких языков программирования является их ориентация не на систему команд той или иной ЭВМ, а на систему операторов, характерных для записи определенного класса алгоритмов?
a) языков программирования низкого уровня
b) языков программирования высокого уровня
c) языков программирования сверхвысокого уровня
d) правильных ответов нет

4
Что используют все языки программирования высокого уровня для предоставления программисту простого и легкого доступа к различным объектам?
Впишите ответ «……..»

5 – Вопрос с одним вариантом ответа
Что понимают под языком программирования (ЯП)?
a) язык, предназначенный для решения определенного класса задач (проблем)
b) правила представления данных и записи алгоритмов их обработки, которые автоматически выполняются ЭВМ
c) язык, предназначенный для создания пакетов прикладных программ, в том числе для современных операционных систем
d) правильных ответов нет

6 – Вопрос с одним вариантом ответа
В результате выполнения фрагмента программы
```
double x = 0, y = 0, z = x/y;
```
a) ошибки не будет и значение переменной z будет равно null
b) возникнет ошибка на этапе компиляции программы
c) возникнет ошибка на этапе выполнения программы
d) ошибки не будет и значение переменной z будет равно Infinity
e) ошибки не будет и значение переменной z будет равно NaN
f) ошибки не будет и значение переменной z будет равно 0
g) правильных ответов нет

7 – Вопрос с одним вариантом ответа
Как называется именованная спецификация одного или более столбцов (для каждого столбца указывается имя, а также его тип или домен)?
a) строчный тип данных
b) объектный тип данных
c) комбинированный тип данных

d) правильных ответов нет

8 – Вопрос с одним вариантом ответа
Что понимается под наследованием типов?
a) однородная масса разрядов, имеющая какую либо структуру
b) возможность дисциплинированного создания новых типов на основе уже определенных
c) многовходовой программный модуль, точки входа которого соответствуют набору операций реализуемого типа
d) правильных ответов нет

9 – Вопрос с одним вариантом ответа
Что из перечисленного не относится к наиболее распространенным конструируемым типам данных?
a) тип записи
b) тип множества
c) тип массива
d) тип распределения
e) правильных ответов нет

10
Что представляет собой открытый массив?
a) фактический параметр подпрограммы, описывающий базовый тип элементов массива, но не определяющий его размерности и границы
b) формальный параметр подпрограммы, описывающий базовый тип элементов массива и определяющий его размерность и границы
c) формальный параметр подпрограммы, описывающий базовый тип элементов массива, но не определяющий его размерности и границы
d) правильных ответов нет

Тема 1.4 Система управления базами данных MySQL и MS SQL Server

1
Обязательные элементы запроса, которые определяют выбранные столбцы, их порядок и источник данных _____

2
Фильтрация по нескольким условиям осуществляется с помощью оператора
1) AND
2) SQL
3) WHERE
4) OR
3
Для обновления существующих данных или пустых полей строки нужно использовать запрос:
1) INSERT
2) LastName
3) VALUES
4) UPDATE

4

Основными понятиями реляционных баз данных являются:
Впишите ответ «…….»

5
Строки таблицы-отношения называются:
1) атрибутами
2) доменами
3) кортежами
4) ключами

6
SQL Server поддерживает процедурную целостность путем использования _____ — процедур, которые выполняются, когда запись вставляется, изменяется или удаляется.
Ответ: триггеров

Тема 1.5 Тестирование и отладка программных решений

1
Ошибка, возникающая при выполнении арифметических операций?
a) Ошибка интерфейса
b) Ошибка описания входных данных
c) Ошибка вычисления
d) Ошибка ввода-вывода

2
Ошибка, возникающая в процессе обмена данными между устройствами памяти?
a) Ошибка интерфейса
b) Ошибка описания входных данных
c) Ошибка вычисления
d) Ошибка ввода-вывода

3
Ошибка, допущенная в ходе описания данных?
a) Ошибка интерфейса
b) Ошибка описания входных данных
c) Ошибка вычисления
d) Ошибка описания данных

4
Функциональный подход к формированию тестовых наборов:
a) основывается на том, что структура ПО не известна
b) базируется на том, что известна структура тестируемого ПО, в том числе его алгоритмы.
c) соответствует основным критериям тестирования

5
При статическом подходе к ручному контролю ПО:
a) анализируют внешние связи
b) анализируют структуру, управляющие и информационные связи программы, ее входные и выходные данные.

c) анализируют только структуру

6
При динамическом подходе к ручному контролю ПО:
a) моделируют процесс выполнения программы на заданных исходных данных
b) анализируют структуру, управляющие и информационные связи программы, ее входные и выходные данные.
c) анализируют только структуру

7
К основным методам ручного контроля относят:
a) контроль вычислений, соответствие заданным требованиям
b) контроль обращения к данным
c) инспекции исходного текста, сквозные просмотры, проверка за столом, оценки программ

Тема 1.6 Документирование программных решений

1
Согласно классификации документов ЕСПД, документ, заверенный установленными подписями и считающийся первичным, выполненный на материальном носителе, допускающем многократное воспроизведение - это..
Впишите ответ «…….»

2
Документом, скопированным с подлинника, полностью идентичным подлиннику, соласно классификации документов по ЕСПД, называют...
1) Подлинник
2) Дубликат
3) Копия

3
Документ, скопированный с подлинника или дубликата, использующийся при сопровождении и эксплуатации программ, согласно классификации документов по ЕСПД - это...
1) Подлинник
2) Дубликат
3) Копия

4 – Вопрос с несколькими вариантами ответа
Что из нижеприведённого не является видом эксплуатационной документации?
a) Формуляр
b) Руководство оператора ЭВМ
c) Спецификация
d) Описание примененния
e) Описание языка
f) Программа и методика испытаний

5
Как расшифровывается ЕСПД?
Впишите ответ «…….»

6
Что обозначает цифра 19 в группе стандартов ГОСТ 19.XXX-XX?
1) год регистрации стандарта
2) класс стандартов
3) код группы стандартов
4) номер стандарта в группе

7
Что обозначает .X в группе стандартов ГОСТ 19.XXX-XX?
1) год регистрации стандарта
2) класс стандартов ЕСПД
3) код группы стандартов
4) номер стандарта в группе

6
Что обозначает XX после тире в группе стандартов ГОСТ 19.XXX-XX?
1) год регистрации стандарта
2) класс стандартов ЕСПД
3) код группы стандартов
4) номер стандарта в группе

9
Что обозначает XX до тире в группе стандартов ГОСТ 19.XXX-XX?
1) год регистрации стандарта
2) класс стандартов ЕСПД
3) код группы стандартов
4) номер стандарта в группе

10

Выберите работы, включаемые в стадию "Техническое задание" по ЕСПД (ГОСТ 19.102)
1) Обоснование целесообразности применения ранее разработанных программ
2) Обоснование принципиальной возможности решения поставленной задачи
3) Разработка технико-экономического обоснования разработки программы
4) Определение стадий, этапов и сроков разработки программы и документации на неё
5) Разработка общего описания алгоритма решения задачи
6) Согласование и утверждение технического проекта

11
Разработка программных документов в соответствии с требованиями ГОСТ 19.101 происходит на стадии...
1) Техническое задание
2) Эскизный проект
3) Технический проект
4) Рабочий проект
5) Внедрение

12
Окончательное определение конфигурации технических средств происходит на стадии...
1) Техническое задание
2) Эскизный проект
3) Технический проект

4) Рабочий проект
5) Внедрение

13
Определение необходимости проведения научно-исследовательских работ на последующих стадиях происходит на стадии...
1) Техническое задание
2) Эскизный проект
3) Технический проект
4) Рабочий проект
5) Внедрение

Критерии оценок студентов при прохождении тестирования:
Оценка «5» - если верно выполнено от 85% до 100% всех заданий.
Оценка «4» - если верно выполнено от 75% до 84% всех заданий.
Оценка «3» - если верно выполнено от 56% до 74 % всех заданий.
Оценка «2» - если верно выполнено менее 56% всех заданий.

3. СПЕЦИФИКАЦИИ И ВАРИАНТЫ ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

3.1. Назначение

Спецификацией устанавливаются требования к содержанию и оформлению вариантов оценочного средства – дифференцированный зачет.

Дифференцированный зачет предназначен для промежуточной аттестации и оценки знаний и умений студентов по программе учебной дисциплины «Безопасность жизнедеятельности» основной профессиональной образовательной программы 09.02.07 Информационные системы и программирование

3.2. Контингент аттестуемых: студенты 1 курса

3.3. Форма и условия аттестации:

Аттестация проводится в форме дифференцированного зачета по завершению освоения учебного материала учебной дисциплины и при положительных результатах текущего контроля.

Дифференцированный зачет проходит в форме тестирования.

3.4. Время выполнения:

подготовка 20 минут
собеседование 10 минут;
всего 30 минут.

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

Библиографическое описание издания (автор, заглавие, вид, место и год издания, кол. стр.)	**Основная/ дополнительная литература**	**Книгообеспеченность**	
		Кол-во. экз. в библ.	**Электронные ресурсы**
Сети и телекоммуникации : учебник и практикум для среднего профессионального образования / К. Е. Самуйлов [и др.] ; под редакцией К. Е. Самуйлова, И. А. Шалимова,	Основная	-	https://urait.ru/bcode/517817

Д. С. Кулябова. — Москва : Издательство Юрайт, 2023. — 363 с.			
Дибров, М. В. Сети и телекоммуникации. Маршрутизация в IP-сетях : учебник и практикум для среднего профессионального образования / М. В. Дибров. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2023. — 423 с.	Основная	-	https://urait.ru/bcode/531278
Замятина, О. М. Инфокоммуникационные системы и сети. Основы моделирования : учебное пособие для среднего профессионального образования / О. М. Замятина. — Москва : Издательство Юрайт, 2023. — 159 с.	Основная	-	https://urait.ru/bcode/518012

3.6. Перечень материалов, оборудования и информационных источников.

Кабинет № 12 безопасности жизнедеятельности (для проведения занятий лекционного типа и занятий семинарского типа, курсового проектирования (выполнения курсовых работ) групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации) оборудован мультимедийным комплексом. Специализированная мебель: Учебная мебель на 40 посадочных мест (столов 20 шт., стульев 40 шт.), рабочее место преподавателя (стол 1 шт., стул 1 шт.), кафедра 1 шт. доска меловая 3х секционная 1шт.

Стрелковый тир (электронный) Интерактивный тир «Профессионал» (бессрочная лицензия), комплект: КСУ PRO TARGET CONSTRUCTOR ARMY+GTO. Камера-детектор PSS BASIC CAM. Лазерный пистолет PSS ПМ, Лазерный автомат PSS Калашников. Сейф металлический.

Компьютер Intel i5 4460/1Тб/8Гб/монитор Samsung 23" - 1 шт., Мультимедийный проектор Тип 1 Optoma x 400 - 1 шт.

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

3.7.Варианты оценочных средств

Перечень современных профессиональных баз данных (СПБД)

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

Перечень информационных справочных систем (ИСС)

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

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РЕСПУБЛИКИ ДАГЕСТАН
Частное профессиональное образовательное учреждение
«РЕСПУБЛИКАНСКИЙ ПОЛИПРОФЕССИОНАЛЬНЫЙ КОЛЛЕДЖ»
(ЧПОУ «Республиканский полипрофессиональный колледж»)

Вопросы для дифференцированного зачета

по дисциплине Программные решения для бизнеса

1. Сущность структурного подхода.
2. Методология функционального моделирования SADT.
3. Диаграммы потоков данных DFD.
4. Диаграмма «сущность связь».
5. Сущность объектно-ориентированного подхода.
6. UML- язык универсального моделирования.
7. Программная платформы MVC (Model-View-Control).
8. Фреймворки. Использование шаблонов проектирования.
9. Структура языка. Объекты языка.
10. Структурированные типы данных языка.
11. Среда быстрой разработки.
12. Разработка многооконного интерфейса.
13. Этапы разработка клиентского приложения программного решения.
14. Технология разработки клиентского приложения программного решения.
15. Компоненты доступа к БД.
16. Интерфейс системы управления БД.
17. Этапы разработки серверной части программного решения.
18. Реализация доступа к БД.
19. План тестирования.
20. Виды тестирования.
21. Модульное тестирование.
22. Объемное испытание.
23. Интеграционное тестирование.
24. Приемочные испытания.
25. Тест-кейсы, результаты тест-кейсов.
26. Отчет о процессе тестирования.
27. Структура технической документации.
28. Техническое задание.
29. Технический проект.
30. Руководство программиста.
31. Руководство пользователя

4. ОСОБЕННОСТИ ОСВОЕНИЯ ДИСЦИПЛИНЫ ДЛЯ ИНВАЛИДОВ И ЛИЦ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ

Адаптированные оценочные материалы содержатся в адаптированной ОПОП. Обучение обучающихся с ограниченными возможностями здоровья при необходимости

осуществляется на основе адаптированной рабочей программы с использованием специальных методов обучения и дидактических материалов, составленных с учетом особенностей психофизического развития, индивидуальных возможностей и состояния здоровья таких обучающихся (обучающегося).

Самостоятельная работа обучающихся с ограниченными возможностями здоровья и инвалидов позволяет своевременно выявить затруднения и отставание и внести коррективы в учебную деятельность. Конкретные формы и виды самостоятельной работы обучающихся лиц с ограниченными возможностями здоровья и инвалидов устанавливаются преподавателем. Выбор форм и видов самостоятельной работы, обучающихся с ограниченными возможностями здоровья и инвалидов осуществляется с учетом их способностей, особенностей восприятия и готовности к освоению учебного материала. Формы самостоятельной работы устанавливаются с учетом индивидуальных психофизических особенностей (устно, письменно на бумаге или на компьютере, в форме тестирования, электронных тренажеров и т.п.).

Основные формы представления оценочных средств – в печатной форме или в форме электронного документа. Для обучающихся с нарушениями зрения предусматривается возможность проведения текущего и промежуточного контроля в устной форме. Для обучающихся с нарушениями слуха предусматривается возможность проведения текущего и промежуточного контроля в письменной форме.

Таблица 4.1. – Категории обучающихся с ОВЗ, способы восприятия ими информации и методы их обучения.

<table>
<tr><th colspan="2">Категории обучающихся по нозологиям</th><th>Методы обучения</th></tr>
<tr><td rowspan="2">с нарушениями зрения</td><td>Слепые.
Способ восприятия информации: осязательно-слуховой</td><td rowspan="2">Аудиально-кинестетические, предусматривающие поступление учебной информации посредством слуха и осязания.
Могут использоваться при условии, что визуальная информация будет адаптирована для лиц с нарушениями зрения:
визуально-кинестетические, предполагающие передачу и восприятие учебной информации при помощи зрения и осязания;
аудио-визуальные, основанные на представление учебной информации, при которых задействовано зрительное и слуховое восприятие;
аудио-визуально-кинестетические, базирующиеся на представлении информации, которая поступает по зрительному, слуховому и осязательному каналам восприятие.</td></tr>
<tr><td>Слабовидящие.
Способ восприятия информации: зрительно-осязательно-слуховой</td></tr>
<tr><td rowspan="2">С нарушениями слуха</td><td>Глухие.
Способ восприятия информации: зрительно-осязательный</td><td rowspan="2">визуально-кинестетические, предполагающие передачу и восприятие учебной информации при помощи зрения и осязания.
Могут использоваться при условии, что аудиальная информация будет адаптирована для лиц с нарушениями слуха:
аудио-визуальные, основанные на представление учебной информации, при которых задействовано зрительное и слуховое восприятие;
аудиально-кинестетические, предусматривающие поступление учебной информации посредством слуха и осязания;
аудио-визуально-кинестетические, базирующиеся на представлении информации, которая поступает по зрительному, слуховому и осязательному каналам восприятие.</td></tr>
<tr><td>Слабослышащие.
Способ восприятия информации: зрительно-осязательно-слуховой</td></tr>
<tr><td>С нарушениями опорно-двигательного</td><td>Способ восприятия информации:</td><td>– визуально-кинестетические;
– аудио-визуальные;
– аудиально-кинестетические;</td></tr>
</table>

аппарата	зрительно-осязательно-слуховой	– *аудио-визуально-кинестетические.*

Таблица 4.2. – Способы адаптации образовательных ресурсов.

Условные обозначения:

«+» —образовательный ресурс, не требующий адаптации;

«АФ» — адаптированный формат к особенностям приема-передачи информации обучающихся инвалидов и лиц с ОВЗ формат образовательного ресурса, в том числе с использованием специальных технических средств;

«АЭ»— альтернативный эквивалент используемого ресурса

Категории обучающихся по нозологиям		**Образовательные ресурсы**				
		Электронные				**Печатные**
		мультимедиа	**графические**	**аудио**	**текстовые, электронные аналоги печатных изданий**	
С нарушениями зрения	Слепые	АФ	АЭ (например, создание материальной модели графического объекта (3Dмодели)	+	АЭ (например, аудио описание)	АЭ (например, печатный материал, выполненный рельефно-точечным шрифтом Л. Брайля)
	Слабовидящие	АФ	АФ	+	АФ	АФ
С нарушениями слуха	Глухие	АФ	+	АЭ (например, текстовое описание, гиперссылки)	+	+
	Слабослышащие	АФ	+	АФ	+	+
С нарушениями опорно-двигательного аппарата		+	+	+	+	+

Таблица 4.3. - Формы контроля и оценки результатов обучения инвалидов и лиц с ОВЗ

Категории обучающихся по нозологиям	**Форма контроля и оценки результатов обучения**
С нарушениями зрения	– *устная проверка:* дискуссии, тренинги, круглые столы, собеседования, устные коллоквиумы и др.; – *с использованием компьютера и специального ПО*: работа с электронными образовательными ресурсами, тестирование, рефераты, курсовые проекты, дистанционные формы, если позволяет острота зрения - графические работы и др.
С нарушениями слуха	– *письменная проверка:* контрольные, графические работы, тестирование, домашние задания, эссе, письменные коллоквиумы, отчеты и др.; – *с использованием компьютера и специального ПО:* работа с электронными образовательными ресурсами, тестирование, рефераты, курсовые проекты, графические работы, дистанционные формы и др.

С нарушениями опорно-двигательного аппарата	– *письменная проверка, с использованием специальных технических средств* (альтернативных средства ввода, управления компьютером и др.): контрольные, графические работы, тестирование, домашние задания, эссе, письменные коллоквиумы, отчеты и др.; – *устная проверка, с использованием специальных технических средств* (средств коммуникаций): дискуссии, тренинги, круглые столы, собеседования, устные коллоквиумы и др.; – *с использованием компьютера и специального ПО* (альтернативных средств ввода и управления компьютером и др.): работа с электронными образовательными ресурсами, тестирование, рефераты, курсовые проекты, графические работы, дистанционные формы - предпочтительнее обучающимся, ограниченным в передвижении и др.

4.1. Задания для текущего контроля для инвалидов и лиц с ограниченными возможностями

Текущий контроль и промежуточная аттестация обучающихся инвалидов и лиц с ОВЗ осуществляется с использованием оценочных средств, адаптированных к ограничениям их здоровья и восприятия информации, в том числе с использованием специальных технических средств.

Текущий контроль успеваемости для обучающихся инвалидов и лиц с ОВЗ направлен на своевременное выявление затруднений и отставания в обучении и внесения коррективов в учебную деятельность. Возможно осуществление входного контроля для определения его способностей, особенностей восприятия и готовности к освоению учебного материала.

4.2. Задания для промежуточной аттестации для инвалидов и лиц с ограниченными возможностями

Форма промежуточной аттестации устанавливается с учетом индивидуальных психофизических особенностей (устно, письменно на бумаге, письменно на компьютере, в форме тестирования и т.п.). При необходимости обучающимся предоставляется дополнительное время для подготовки ответа.

Промежуточная аттестация, при необходимости, может проводиться в несколько этапов. Для этого рекомендуется использовать рубежный контроль, который является контрольной точкой по завершению изучения раздела или темы дисциплины, междисциплинарного курса, практик и ее разделов с целью оценивания уровня освоения программного материала. Формы и срок проведения рубежного контроля определяются преподавателем (мастером производственного обучения) с учетом индивидуальных психофизических особенностей обучающихся.

Приложение 1

Кодификатор (примерный перечень) оценочных средств для оценки знаний, умений и уровня сформированности компетенций

№ п/п Код оценочного средства	Тип оценочного средства	Краткая характеристика оценочного средства	Представление оценочного средства в фонде
1.	Деловая и/или ролевая игра	Совместная деятельность группы обучающихся и преподавателя под управлением преподавателя с целью решения учебных и профессионально-ориентированных задач путем игрового моделирования реальной проблемной ситуации. Позволяет оценивать умение анализировать и решать типичные профессиональные задачи	Тема (проблема), концепция, роли и ожидаемый результат
2.	Кейс-задача	Учебный материал подается студентам в виде проблем (кейсов), в которых обучающимся предлагается осмыслить реальную профессиональную ситуацию для решения данной проблемы. Знания приобретаются в результате активной и творческой работы: самостоятельного осуществления целеполагания, сбора необходимой информации, ее анализа с разных точек зрения, выдвижения гипотезы, выводов, заключения, самоконтроля процесса получения знаний и его результатов.	Задания для решения кейс - задачи
3.	Коллоквиум	Средство контроля усвоения учебного материала темы, раздела или разделов дисциплины, организованное как учебное занятие в виде собеседования преподавателя с обучающимися.	Вопросы по темам / разделам дисциплины или профессионального модуля
4.	Контрольная работа	Средство проверки умений применять полученные знания для решения задач определенного типа по теме или разделу	Комплект контрольных заданий по вариантам
5.	Круглый стол, дискуссия, диспут, дебаты	Оценочные средства, позволяющие включить обучающихся в процесс обсуждения спорного вопроса, проблемы и оценить их умение аргументировать собственную точку зрения	Перечень дискуссионных тем для проведения круглого стола, дискуссии, диспута, дебатов
6.	Портфолио	Целевая подборка работ студента, раскрывающая его индивидуальные образовательные достижения в одной или нескольких учебных дисциплин, в профессиональном модуле.	Структура портфолио

№ п/п Код оценочного средства	Тип оценочного средства	Краткая характеристика оценочного средства	Представление оценочного средства в фонде
7.	Проект	Конечный продукт, получаемый в результате планирования и выполнения комплекса учебных и исследовательских заданий. Позволяет оценить умения обучающихся самостоятельно конструировать свои знания в процессе решения практических задач и проблем, ориентироваться в информационном пространстве и уровень сформированности аналитических, исследовательских навыков, навыков практического и творческого мышления. Может выполняться в индивидуальном порядке или группой обучающихся.	Тема групповых и/или индивидуальных проектов
8.	Рабочая тетрадь	Дидактический комплекс, предназначенный для самостоятельной работы обучающегося и позволяющий оценивать уровень усвоения им учебного материала	Образец рабочей тетради
9.	Разноуровневые учебные задачи и задания	Различают задачи и задания: а) репродуктивного уровня, позволяющие оценивать и диагностировать знание фактического материала (базовые понятия, алгоритмы, факты) и умение правильно использовать специальные термины и понятия, узнавание объектов изучения в рамках определѐнного раздела дисциплины; б) реконструктивного уровня, позволяющие оценивать и диагностировать умения синтезировать, анализировать, обобщать фактический и теоретический материал с формулированием конкретных выводов, установлением причинно-следственных связей; в) творческого уровня, позволяющие оценивать и диагностировать умения, интегрировать знания различных областей, аргументировать собственную точку зрения	Комплект разноуровневых задач и заданий
10.	Расчетно-графическая работа	Средство проверки умений применять полученные знания по заранее определенной методике для решения задач или заданий по модулю или дисциплине в целом.	Комплект заданий для выполнения расчетно-графической работы

№ п/п Код оценочного средства	Тип оценочного средства	Краткая характеристика оценочного средства	Представление оценочного средства в фонде
11.	Реферат	Продукт самостоятельной работы студента, представляющий собой краткое изложение в письменном виде полученных результатов теоретического анализа определенной темы, где автор раскрывает суть исследуемой проблемы, приводит различные точки зрения, а также собственные взгляды на нее.	Темы рефератов
12.	Доклад, сообщение	Продукт самостоятельной работы студента, представляющий собой публичное выступление по представлению полученных результатов решения определенной темы.	Темы докладов, сообщений
13.	Собеседование	Средство контроля, организованное как специальная беседа преподавателя с обучающимся на темы, связанные с изучаемой дисциплиной, и рассчитанное на выяснение объема знаний обучающегося по определенному разделу, теме, проблеме и т. п.	Вопросы по темам / разделам дисциплины
14.	Творческое задание	Частично регламентированное задание, имеющее нестандартное решение и позволяющее диагностировать умения, интегрировать знания различных областей, аргументировать собственную точку зрения. Может выполняться в индивидуальном порядке или группой обучающихся	Темы групповых и/или индивидуальных творческих заданий
15.	Тест	Средство контроля, направленное на проверку уровня освоения контролируемого теоретического и практического материала по дидактическим единицам дисциплины или профессионального модуля. Система стандартизированных заданий, позволяющая автоматизировать процедуру измерения уровня знаний и умений обучающихся	Фонд тестовых заданий
16.	Эссе	Средство, позволяющее оценить умение обучающегося письменно излагать суть поставленной проблемы, самостоятельно проводить анализ этой проблемы.	Тематика эссе
17.	Практические работы (практическое задание)	Это задания, с помощью которых у учащихся формируются и развиваются правильные практические действия.	Виды: наблюдение, измерение, опыт,

№ п/п Код оценочного средства	Тип оценочного средства	Краткая характеристика оценочного средства	Представление оценочного средства в фонде
			конструирование и др. задания для практических работ
18.	Лабораторные работы	Это проведение учащимися по заданию преподавателя опытов с использованием приборов, применением инструментов и других технических приспособлений.	Задания для лабораторных работ
19.	Тренажер	Техническое средство, которое может быть использовано для контроля приобретенных студентом профессиональных навыков и умений по управлению конкретным материальным объектом	Комплект заданий для работы на тренажере
20.	Отчеты по практикам	Средство контроля, позволяющая обучающемуся продемонстрировать обобщенные знания, умения и практический опыт, приобретенные за время прохождения учебной и производственной практик. Отчеты по практикам позволяют контролировать в целом усвоение ОК и ПК обозначенных в ППССЗ.	Виды работ и задания на учебную и производственную практику
21.	Контент-анализ документации	Анализ и оценка в соответствии с критериями документов (журналов теоретического и производственного обучения, характеристик, творческих работ, дневников и отчетов по практике, ВКР и др.), свидетельствующих об уровне компетентности обучающегося.	Перечень документов подлежащих анализу, критерии оценки
22.	Наблюдение	Инструмент сбора информации для установления фактов	Цель, объекты наблюдения, образец листа для фиксирования результатов наблюдения
23.	Задание на ВКР (дипломный проект, дипломная работа)	Перечень основных вопросов, которые должны быть раскрыты в работе, а также указания на основные информационные источники.	ВКР по специальности СПО

№ п/п Код оценочного средства	Тип оценочного средства	Краткая характеристика оценочного средства	Представление оценочного средства в фонде
24.	Зачет	Средство проверки теоретических знаний по темам, разделам, всему курсу УД.	Перечень вопросов, заданий
25.	Дифференцированный зачет	Средство проверки теоретических знаний по темам, разделам, всему курсу УД.	Перечень вопросов, заданий
26.	Экзамен	В перечень вопросов включены все темы УД.	Экзаменационные билеты