Образовательное частное учреждение высшего образования
«ГУМАНИТАРНО-СОЦИАЛЬНЫЙ ИНСТИТУТ»

УТВЕРЖДЕНО
заседанием Ученого совета
протокол № 7 от 27.06.2023г.
приказ ректора об утв. ОП ВО
№ 01-03/70 П от 28.06.2023 г.
Ректор _____ Л.Ф. Логинова

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ

Б1.В.08 «УПРАВЛЕНИЕ ДАННЫМИ В БИЗНЕС-СИСТЕМАХ»

Направление подготовки
38.03.05 Бизнес-информатика

Направленность (профиль)
Информационная бизнес-аналитика

Квалификация **бакалавр**

Красково – 2023

Рабочая программа учебной дисциплины разработана на основе Федерального государственного образовательного стандарта высшего образования (далее – ФГОС ВО) по программе подготовки 38.03.05 «Бизнес-информатика».

Организация – разработчик: Образовательное частное учреждение высшего образования «Гуманитарно-социальный институт».

Разработчики:

к.т.н., с.н.с.		Шматратов В.Ю.
ученая степень, звание	*подпись*	*ФИО*
ученая степень, звание	*подпись*	*ФИО*

Рабочая программа учебной дисциплины утверждена на заседании кафедры «Управления и экономики» от 08.06.2023 г. протокол № 10

Заведующий кафедрой
Д.э.н., профессор _____ Коновалов В.М.
подпись

Часть, формируемая участниками образовательных отношений
Наименование дисциплины – Управление данными в бизнес-системах

1. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы

Цель освоения дисциплины: является в формировании знаний в области управления, хранения, обработки, анализа основных технологий реализации данных в бизнес-системах.

Изучение дисциплины способствует решению следующих **задач** профессиональной деятельности: получение студентом необходимого объёма знаний в области операционных систем и применение этих знаний для решения практических задач; освоение современных методов и технологий построения информационных систем с базами данных, использования СУБД.

Дисциплина «Управление данными в бизнес-системах» в рамках воспитательной работы направлена на формирование у обучающихся; психологической готовности к профессиональной деятельности по избранной профессии; воспитание у обучающихся уважения к труду, людям труда, трудовым достижениям и подвигам; формирование у обучающихся потребности трудиться, добросовестного, ответственного и творческого отношения к разным видам трудовой деятельности; развитие навыков высокой работоспособности и самоорганизации, гибкости, умение действовать самостоятельно, активно и ответственно, мобилизуя необходимые ресурсы, правильно оценивая смысл и последствия своих действий; коммуникативной культуры и развитие органов студенческого самоуправления; исследовательского и критического мышления у обучающихся; повышение мотивации к научно- исследовательской деятельности, интереса к науке в целом; развитие творческой культуры и эрудиции; навыков творческого применения на практике достижений научного прогресса; развитие навыков решения прикладных задач с использованием научных методов, продвижение собственных научных идей.

Планируемые результаты обучения

Процесс освоения дисциплины направлен на формирование следующих компетенций:

ПК-1 Способен анализировать данные и обосновывать решения задач бизнес-анализа.

Подготовка по дисциплине реализуется на основе профессионального стандарта: ПС 08.037 «Бизнес-аналитик».

Матрица связи дисциплины Б1.В.08 «Управление данными в бизнес-системах» и компетенций, формируемых на основе изучения дисциплины, с временными этапами освоения ее содержания

Код и наименование компетенции выпускника	Код и наименование индикатора компетенции выпускника	Код индикатора компетенции выпускника	Код и наименование дескрипторов (планируемых результатов обучения выпускников)

ПК-1 Способен анализировать данные и обосновывать решения задач бизнес-анализа	ПК-1.1. Применяет различные инструменты сбора информации о бизнес-проблемах или бизнес-возможностях	**ПК-1.1.**	**ПК-1.1.1** **Знать:** инструменты и методы сбора данных о бизнес-проблемах или бизнес-возможностях; основные технологии организации хранения и обработки больших объемов данных; назначение и компоненты систем управления базами данных; основные сведения о базовых структурах данных; способы и инструменты проектирования баз данных; особенности языка SQL; основы CASE-технологии и программного обеспечения при автоматизированной разработке баз данных; **ПК-1.1.2** **Уметь:** оценивать и анализировать различные системы управления данными. осуществлять поиск, сбор и анализ необходимой информации в глобальной сети; работать с системами управления базами данных; проектировать, разрабатывать и использовать базы данных; создавать запросы и выражения к базам данных на языке SQL и структурировать полученную информацию; применять CASE-технологии и ПО при автоматизированной разработке баз данных **ПК-1.1.3** **Владеть:** основными методами, способами и средствами сбора, хранения и переработки информации4 навыками работы с системой управления базами данных; методами и программными средствами обработки больших объемов информации; профессиональной терминологией в области баз данных.

2. Место учебной дисциплины в структуре образовательной программы

Блок:1. Дисциплины (модули) ОП часть, формируемая участниками образовательных отношений.

В структурной форме межпредметные связи изучаемой дисциплины указаны в соответствии с учебным планом образовательной программы по очной форме обучения.

Связь дисциплины «Управление данными в бизнес-системах» с предшествующими дисциплинами и сроки их изучения

Код дисциплины	*Дисциплины, предшествующие дисциплине* «Управление данными в бизнес-системах»	*Семестр*
Б1.В.ДВ.01.01	Финансовый менеджмент	5
Б1.В.ДВ.01.02	Экономический анализ	5

Связь дисциплины «Управление данными в бизнес-системах» со смежными дисциплинами, изучаемыми параллельно

Код дисциплины	*Дисциплины, изучаемые параллельно*	*Семестр*
Б1.В.07	Оценка эффективности бизнеса	6
Б2.О.03(П)	Производственная практика: технологическая (проектно-технологическая) практика	6

Связь дисциплины «Управление данными в бизнес-системах» с последующими дисциплинами и сроки их изучения

Код дисциплины	*Дисциплины, следующие за дисциплиной* «Управление данными в бизнес-системах»	*Семестр*
Б1.В.10	Анализ данных в бизнес-системах	7
Б1.В.11	Информационно-аналитические системы	8
Б1.В.12	Практикум по разработке систем бизнес-аналитики	8
Б2.О.04(Пд)	Производственная практика: преддипломная практика	8
Б2.В.01(П)	Производственная практика: практика по получению профессиональных умений и опыта профессиональной деятельности	7

3. Объем дисциплины в зачетных единицах с указанием количества академических часов, выделенных на контактную работу обучающихся с преподавателем (по видам учебных занятий) и на самостоятельную работу

Виды учебной работы	**Форма обучения**	
	Очная	**Очно-заочная**
Порядковый номер семестра	6	7
Общая трудоемкость дисциплины всего (в з.е):	4	4
Контактная работа с преподавателем всего (в акад. часах), в том числе:	44	32
Занятия лекционного типа (лекции)	14	10
Занятия семинарского типа (практические занятия, семинары в том числе в форме практической подготовки)	14	10
Лабораторные работы	14	10
Текущая аттестация	1	1
Консультации (предэкзаменационные)	-	-
Промежуточная аттестация	1	1
Самостоятельная работа всего (в акад. часах), в том	100	112

числе:		
Форма промежуточной аттестации:		
зачет/ дифференцированный зачет		
Экзамен	Диф.зачет (зачет с оценкой)	Диф.зачет (зачет с оценкой)
Общая трудоемкость дисциплины (в акад. часах)	144	144

4. Содержание дисциплины, структурированное по темам (разделам)

4.1. Тематическое планирование

Тема 1. Введение в управление данными. Понятие данных. Хранилища данных. Виды, методы и технологии сбора, хранения и анализа данных. Способы, процедуры и режимы обработки данных. Технические средства сбора данных.

Тема 2. Система управления базами данными. Разновидности баз данных. Файловые системы хранения данных. Основные понятия СУБД. Классификация СУБД. Архитектура СУБД. Методы доступа к данным. Функции и компоненты СУБД. Типовая организация современной СУБД.

Тема 3. Модели данных и моделирование. Модель «сущность-связь». Реляционная модель данных. Реляционная алгебра. Связи. Индексы. Правила Кодда. Иерархическая модель данных. Сетевая модель данных. Объектно-ориентированная модель данных.

Тема 4. Многопользовательский доступ к данным. Транзакции. Блокировки. Временные отметки. Защита данных в базах данных. Обеспечение целостности и безопасности данных. Виды сбоев. Средства физической защиты данных. Восстановление базы данных. Защита от несанкционированного доступа. Оптимизация запросов. Жизненный цикл базы данных.

Тема 5. Элементы проектирования баз данных. Требования к проекту базы данных. Этапы проектирования базы данных. Инфологическое проектирование. Логическое проектирование БД. Физическое проектирование БД. Автоматизация проектирования БД.

Тема 6. Особенности проектирования реляционных БД. Выявление нереализуемых связей. Определение первичных ключей. Определение типов данных атрибутов. Описание ограничений целостности. Аномалии модификации данных. Нормализация отношений. Перспективы развития технологии баз данных.

4.2. Содержание занятий семинарского типа

№	Содержание практических занятий	Виды практических занятий	Текущий контроль
«Проектирование и реализация информационно-поисковой системы с помощью CASE-средства DBDesigner»			
1.	Практическая работа 1. Моделирование. Создание таблиц.	устный опрос по теме практического	Индивидуальное и

		занятия; работа в группах; решение ситуационных задач; коллоквиум; выполнение практического задания	групповое собеседование. Мониторинг практических заданий.
2.	Практическая работа 2. Формирование отношений.	устный опрос по теме практического занятия; работа в группах; решение ситуационных задач; коллоквиум; выполнение практического задания	Индивидуальное и групповое собеседование. Мониторинг практических заданий.
3.	Практическая работа 3. Кодирование.	устный опрос по теме практического занятия; работа в группах; решение ситуационных задач; коллоквиум; выполнение практического задания	Индивидуальное и групповое собеседование. Мониторинг практических заданий.
4.	Практическая работа 4. Работа с базой данных. Установление соединения с БД на сервере. Синхронизация.	устный опрос по теме практического занятия; работа в группах; решение ситуационных задач; коллоквиум; выполнение практического задания	Индивидуальное и групповое собеседование. Мониторинг практических заданий.
5.	Практическая работа 5. SQL-запросы. Шаблоны. Query Mode.	устный опрос по теме практического занятия; работа в группах; решение ситуационных задач; коллоквиум;	Индивидуальное и групповое собеседование. Мониторинг практических заданий.

		выполнение практического задания	

№	Название лабораторных работ	Виды лабораторных работ	Текущий контроль
1.	Лабораторная работа 1. Знакомство с SQL Management studio. Построение ER-модели. Переход к реляционной модели.	Лабораторные задания	- изучение теоретического материала; - выполнение заданий; - -индивидуальное собеседование
2.	Лабораторная работа 2. Создание базы данных в SQL Management studio.	Лабораторные задания	- изучение теоретического материала; - выполнение заданий; - -индивидуальное собеседование
3.	Лабораторная работа 3. Заполнение таблиц базы данных.	Лабораторные задания	- изучение теоретического материала; - выполнение заданий; - -индивидуальное собеседование
4.	Лабораторная работа 4. Создание запросов и фильтров.	Лабораторные задания	- изучение теоретического материала; - выполнение заданий; - -индивидуальное собеседование
5.	Лабораторная работа 5. Хранимые процедуры.	Лабораторные задания	- изучение теоретического материала; - выполнение заданий; - -индивидуальное собеседование
6.	Лабораторная работа 6. Пользовательские функции	Лабораторные задания	- изучение теоретического материала; - выполнение заданий; - -индивидуальное собеседование
7.	Лабораторная работа 7. Триггеры.	Лабораторные	- изучение

		задания	теоретического материала; - выполнение заданий; - -индивидуальное собеседование

4.3. Самостоятельная работа студента

№	Наименование темы дисциплины	Формы подготовки
1.	Способы, процедуры и режимы обработки данных.	-изучение материала по учебникам, научным статьям - подготовка сообщений к выступлению - подготовке и написании докладов
2.	Файловые системы хранения данных.	-изучение материала по учебникам, научным статьям - подготовка сообщений к выступлению - подготовке и написании докладов
3.	Иерархическая модель данных.	-изучение материала по учебникам, научным статьям - подготовка сообщений к выступлению
4.	Сетевая модель данных.	-изучение материала по учебникам, научным статьям, - подготовка сообщений к выступлению
5.	Объектно-ориентированная модель данных.	-изучение материала по учебникам, научным статьям - подготовка сообщений к выступлению
6.	Оптимизация запросов. Жизненный цикл базы данных.	-изучение материала по учебникам, научным статьям - подготовка сообщений к выступлению - подготовке и написании докладов
7.	Жизненный цикл базы данных.	-изучение материала по учебникам, научным статьям - подготовка сообщений к выступлению - подготовке и написании докладов
8.	Инфологическое проектирование.	-изучение материала по учебникам, научным статьям, - подготовка сообщений к выступлению
9.	Перспективы развития технологии баз данных.	-изучение материала по учебникам, научным статьям - подготовка сообщений к выступлению - подготовке и написании докладов

4.4. Распределение часов по темам и видам учебных занятий

Номер раздела, темы дисциплины	***Компетенции***	***Контактная работа***		***Лекции***		***Практические занятия Семинары***		***Лабораторные занятия***		***Самост. работа студентов***	
Формы		***ОФО***	***ОЗФО***	***ОФО***	***ОЗФО***	***ОФО***	***ОЗФО***	***ОФО***	***ОЗФО***	***ОФО***	***ОЗФО***

обучения											
Тема 1.	ПК-1	2	6	2	2					16	18
Тема 2.	ПК-1	8		4		2	2	2	2	16	18
Тема 3.	ПК-1	6	6	2	2	2	2	2	2	18	18
Тема 4.	ПК-1	10	6	2	2	4	2	4	2	16	18
Тема 5.	ПК-1	10	6	2	2	4	2	4	2	18	20
Тема 6.	ПК-1	6	6	2	2	2	2	2	2	16	20
Текущая аттестация	ПК-1	1									
Промежуточная аттестация	ПК-1	1									
Всего:		***44***	***32***	***14***	***10***	***14***	***10***	***14***	***10***	***100***	***112***

4.5. Методические указания для обучающихся по освоению дисциплины

Для правильной организации самостоятельной работы необходимо учитывать порядок изучения разделов курса, находящихся в строгой логической последовательности. Поэтому хорошее усвоение одной части дисциплины является предпосылкой для успешного перехода к следующей. Для лучшего запоминания материала целесообразно использовать индивидуальные особенности и разные виды памяти: зрительную, слуховую, ассоциативную. Успешному запоминанию способствует приведение ярких свидетельств и наглядных примеров. Учебный материал должен постоянно повторяться и закрепляться.

Подготовка к практическому (семинарскому) занятию начинается с тщательного ознакомления с условиями предстоящей работы, т. е. с обращения к вопросам семинарских занятий. Определившись с проблемой, следует обратиться к рекомендуемой литературе. При подготовке к практическому (семинарскому) занятию обязательно требуется изучение дополнительной литературы по теме занятия. Без использования нескольких источников информации невозможно проведение дискуссии на занятиях, обоснование собственной позиции, построение аргументации. Если обсуждаемый аспект носит дискуссионный характер, следует изучить существующие точки зрения и выбрать тот подход, который вам кажется наиболее верным. При этом следует учитывать необходимость обязательной аргументации собственной позиции. Во время практических занятий рекомендуется активно участвовать в обсуждении рассматриваемой темы, выступать с подготовленными заранее докладами и презентациями, принимать участие в выполнении практических заданий.

С целью обеспечения успешного обучения студент должен готовиться к лекции, поскольку она является важной формой организации учебного процесса: знакомит с новым учебным материалом; разъясняет учебные элементы, трудные для понимания; систематизирует учебный материал; ориентирует в учебном процессе.

Подготовка к лекции заключается в следующем:

- внимательно прочитайте материал предыдущей лекции;
- узнайте тему предстоящей лекции (по тематическому плану);
- ознакомьтесь с учебным материалом по учебным пособиям;
- постарайтесь уяснить место изучаемой темы в своей профессиональной подготовке;
- запишите возможные вопросы, которые вы зададите преподавателю на лекции.

Во время лекции рекомендуется составлять конспект, фиксирующий основные положения лекции и ключевые определения по пройденной теме.

К диф.зачету необходимо готовится целенаправленно, регулярно, систематически и с первых дней обучения по дисциплине. Попытки освоить дисциплину в период зачётно-экзаменационной сессией, как правило, показывают не слишком хороший результат. В самом начале учебного курса студенту следует познакомиться со следующей учебно-методической документацией:

- программой дисциплины;
- перечнем знаний и умений, которыми студент должен овладеть;
- тематическими планами лекций, семинарских занятий;
- контрольными мероприятиями;
- учебными пособиями по дисциплине;
- перечнем зачетных/экзаменационных вопросов.

После этого у студента должно сформироваться четкое представление об объеме и характере знаний и умений, которыми надо будет овладеть по дисциплине. Систематическое выполнение учебной работы на лекциях, семинарских занятиях и в процессе самостоятельной работы позволит успешно освоить дисциплину и создать хорошую базу для сдачи диф.зачета.

Рекомендуемая тематика занятий максимально полно реализуется в контактной работе со студентами очной формы обучения. В случае реализации образовательной программы в заочной / очно-заочной форме трудоемкость дисциплины сохраняется, однако объем учебного материала в значительной части осваивается студентами в форме самостоятельной работы. При этом требования к ожидаемым образовательным результатам студентов по данной дисциплине не зависят от формы реализации образовательной программы.

В случае организации учебной работы с использованием дистанционных образовательных технологий занятия проводятся в электронной информационно-образовательной среде института.

5. ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ

5.1 Образовательные технологии

В освоении учебной дисциплины «Управление данными в бизнес-системах» используются следующие **традиционные образовательные технологии:**

- чтение информационных лекций с использованием доски и видеоматериалов;
- практические занятия;
-лабораторные занятия;
- контрольные опросы;
- консультации;
- самостоятельная работа студентов с учебной литературой и первоисточниками;
- зачетная аттестация- дифференцированный зачет (зачет с оценкой).

5.2.Использование информационных технологий:

- технологии, основанные на использовании ЭИОС института (методические материалы по дисциплине, размещенные на сайте ГСИ);
- Интернет-технологии;
- компьютерные обучающие и контролирующие программы;

- информационные технологии, позволяющие увеличить эффективность преподавания (за счет усиления иллюстративности):
  - ***лекция-визуализация –*** *иллюстративная форма проведения информационных и проблемных лекций;*
  - ***семинар-презентация*** *– использование студентами на семинарах специализированных программных средств.*

5.3. Активные и интерактивные методы и формы обучения

Из перечня видов: («мозговой штурм», анализ проблемных ситуаций, анализ конкретных ситуаций, инциденты, имитация коллективной профессиональной деятельности) используются следующие:
- «*мозговой штурм»;*
- *диспут* (способ ведения спора, проводимого с целью установления научной истины со ссылками на устоявшиеся письменные авторитетные источники и тщательный анализ аргументов каждой из сторон);
- *круглый стол;*
- *дискуссия (*как метод, активизирующий процесс обучения, изучения сложной темы, теоретической проблемы*) применяется на семинарах-дискуссиях, где обсуждаются спорные вопросы с выявлением мнений в студенческой группе;*
- *беседа.*

6. Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине

Контроль качества освоения дисциплины включает в себя текущий контроль успеваемости и промежуточную аттестацию обучающихся. Промежуточная аттестация обучающихся по дисциплине проводится в форме дифференцированного зачета (зачет с оценкой).

Конкретный перечень типовых контрольных заданий и иных материалов для оценки результатов освоения дисциплины, а также описание показателей и критериев оценивания компетенций приведен в фонде оценочных средств по дисциплине.

6.1. Формы текущего контроля

- *индивидуальное собеседование;*
- *мониторинг результатов практических занятий;*
- *мониторинг результатов лабораторных занятий;*
- *тестирование*

6.2. Тестовые задания:

См. приложение – «Банк тестов»

Вопросы к дифференцированному зачету (зачету с оценкой):

1. Перечислите основные инструменты и методы сбора данных о бизнес-проблемах или бизнес-возможностях
2. Как устроена технология организации хранения и обработки больших объемов данных?
3. Дайте определение следующим терминам: база данных, хранилище данных, словарь данных, администратор БД

4. Понятие СУБД, классификация и ее основные функции
5. Назначение и компоненты СУБД
6. Понятие базы данных. Назовите области применения и известные вам примеры баз данных
7. Перечислите основные этапы проектирования БД
8. Определите основные понятия сетевой модели данных. Приведите примеры
9. Понятие иерархической модели данных. Приведите примеры
10. Понятие объектно-ориентированной модели данных. Приведите примеры
11. Дайте определение реляционной базы данных
12. Поясните основные понятия РБД: домен, атрибут, кортеж и отношение
13. СУБД MS SQL Server Management Studio и ее основные возможности
14. Перечислите и кратко охарактеризуйте основные объекты базы данных
15. Дайте определение поля, записи, столбца и строки таблицы
16. Что такое структура таблицы? В чем принципиальное отличие создания таблиц в MS SQL Server Management Studio от подобного процесса в Excel?
17. Способы создания таблиц в MS SQL Server Management Studio
18. Перечислите типы данных, поддерживаемые в MS SQL Server Management Studio
19. Назначение свойств поля. Перечислите наиболее часто используемые свойства полей таблицы
20. Дайте определение репликации. В чем отличие публикаций от статей и подписок?
21. Расскажите об обслуживании репликации. Дайте определение агентам репликации
22. Объясните особенность блочной репликации на уровне системы хранения данных
23. Перечислите отличия физической репликации от логической
24. Достоинства и недостатки логической репликации. Способы реализации логической репликации
25. Как осуществляется резервное копирование и восстановление?
26. Что понимают под форматированием таблицы? Перечислите возможные в MS SQL Server Management Studio операции форматирования данных в таблице
27. Поясните операцию поиска/замены данных в таблице
28. Что такое сортировка данных в таблице? Что является результатом сортировки таблицы по двум и более полям?
29. С какой целью применяется фильтрация данных? Какие виды фильтров существуют в MS SQL Server Management Studio?
30. С какой целью создают запросы к базе данных?
31. Какие виды обработки данных можно выполнить в процессе выполнения запроса?
32. Назовите основные виды запросов
33. Поясните основные принципы создания запроса в СУБД MS SQL Server Management Studio
34. В чем состоит суть запроса на выборку?
35. Как выглядит результирующая таблица, полученная на основе перекрестного запроса?
36. Какие возможности предоставляют запросы по обработке информации в СУБД MS SQL Server Management Studio?
37. Как формируются условия отбора записей в запросах?
38. Что такое вычисляемые поля в запросе? Как они формируются?
39. В каком режиме можно просмотреть результат выполнения запроса на обновление?
40. Как сконструировать запрос с групповыми операциями?

41. Какие статистические функции входят в набор операций группового запроса?
42. Что такое динамический набор данных?
43. Какие запросы изменяют данные в таблицах базы данных?
44. Что такое вложенные запросы?
45. Расскажите о хранимых процедурах
46. В чем отличие первичного ключа от внешнего?
47. Как выявить нереализуемую связь?
48. Расскажите про триггеры входа, триггеры DDL и триггеры DML.
49. Как сконструировать Сообщение об ошибке ввода данных в таблицу?
50. Как переименовать, переместить или удалить поля в таблицах, созданных с помощью Мастера таблиц или в режиме Таблицы?
51. Как создать новую базу данных в СУБД MS SQL Server Management Studio?
52. В чем смысл импортирования таблиц в MS SQL Server Management Studio?
53. Перечислите элементы проектирования БД.
54. Расскажите об этапах проектирования БД.
55. В чем отличие инфологического проектирования БД от логического?
56. Объясните подробности физического проектирования БД
57. Что такое аномалии модификации данных?
58. Объясните суть нормализации отношений
59. Какие CASE-средства Вам известны?
60. Достоинства и недостатки использования CASE-технологий.

7. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ

7.1. Рекомендуемая литература

Основная литература

Гаврилов, Л. П. Инновационные технологии в коммерции и бизнесе : учебник для вузов / Л. П. Гаврилов. — Москва : Издательство Юрайт, 2023. — 372 с. — (Высшее образование). — ISBN 978-5-534-15960-8. — Текст : электронный // Образовательная платформа Юрайт : [сайт]. — URL: https://urait.ru/bcode/510351

Куприянов, Ю. В. Бизнес-системы. Основы теории управления : учебное пособие для вузов / Ю. В. Куприянов. — 3-е изд., испр. и доп. — Москва : Издательство Юрайт, 2023. — 217 с. — (Высшее образование). — ISBN 978-5-534-14352-2. — Текст : электронный // Образовательная платформа Юрайт : [сайт]. — URL: https://urait.ru/bcode/515977

Дополнительная литература

Емельянова, Н. Ю. ИТ-стандарты : учебное пособие для студентов, обучающихся по направлению подготовки 38.03.05 «Бизнес-информатика» / Н. Ю. Емельянова, В. А. Емельянов. — Москва : Прометей, 2023. — 200 с. : табл., схем. — ISBN 978-5-00172-437-7. — Текст : электронный // Университетская библиотека ONLINE : [сайт]. — URL: https://biblioclub.ru/index.php?page=book&id=700940

Купцова, Е. В. Бизнес-планирование : учебник и практикум для вузов / Е. В. Купцова, А. А. Степанов. — Москва : Издательство Юрайт, 2023. — 435 с. — (Высшее образование). — ISBN 978-5-9916-8377-7. — Текст : электронный // Образовательная платформа Юрайт : [сайт]. — URL: https://urait.ru/bcode/511225

Парфенов, Ю. П. Постреляционные хранилища данных : учебное пособие для вузов /

Ю. П. Парфенов ; под научной редакцией Н. В. Папуловской. — Москва : Издательство Юрайт, 2023. — 121 с. — (Высшее образование). — ISBN 978-5-534-09837-2. — Текст : электронный // Образовательная платформа Юрайт : [сайт]. — URL: https://urait.ru/bcode/514724

Толстобров, А. П. Управление данными : учебное пособие для вузов / А. П. Толстобров. — 3-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2023. — 272 с. — (Высшее образование). — ISBN 978-5-534-14162-7. — Текст : электронный // Образовательная платформа Юрайт : [сайт]. — URL: https://urait.ru/bcode/519787

Периодическая литература (библиотека ГСИ)

1. Информатизация и связь.
2. Проблемы управления.
3. Российский журнал менеджмента.
4. Системный администратор.

ЭБС IPR BOOKS:

1. Актуальные проблемы экономики и менеджмента (доступный архив: 2019–2022). – URL: https://www.iprbookshop.ru/98831.html.
2. Вестник Российского университета дружбы народов. Серия Экономика (доступный архив: 2011–2022). – URL: https://www.iprbookshop.ru/32735.html.
3. Вестник Ростовского государственного экономического университета (РИНХ) (доступный архив: 2014–2022). – URL: https://www.iprbookshop.ru/61941.html.
4. Вестник Сибирского института бизнеса и информационных технологий (доступный архив: 2019–2022). – URL: https://www.iprbookshop.ru/102212.html.
5. Известия Саратовского университета. Новая серия. Серия Математика. Механика. Информатика (доступный архив: 2019–2022). – URL: https://www.iprbookshop.ru/99689.html.
6. Прикладная информатика (доступный архив: 2006–2022). – URL: https://www.iprbookshop.ru/11770.html.
7. Программные продукты и системы (доступный архив: 2010–2022). – URL: https://www.iprbookshop.ru/25852.html.
8. Современная конкуренция (доступный архив: 2007–2022). – URL: https://www.iprbookshop.ru/11778.html.
9. Стратегии бизнеса (доступный архив: 2020–2022). – URL: https://www.iprbookshop.ru/106278.html.

7.2. Электронные образовательные и информационные ресурсы

1. Электронно-библиотечная система «ЮРАЙТ» - https://urait.ru/
2. Университетская библиотека онлайн – www/biblioclub.ru

7.3. Профессиональные базы данных и информационные справочные системы

Информационно-справочные системы

1. «Система КонсультантПлюс» – компьютерная справочная правовая система - http://www.consultant.ru/
2. «Гарант» – справочно-правовая система по законодательству Российской Федерации - http://www.garant.ru/

3. Единое окно доступа к образовательным ресурсам. - http://window.edu.ru/
4. Национальная информационно-аналитическая система Российский индекс научного цитирования (РИНЦ). - https://www.elibrary.ru
5. Федеральный портал «Российское образование» - http://www.edu.ru/

Профессиональные базы данных

1. Научная электронная библиотека eLibrary.ru - Российский индекс научного цитирования (РИНЦ)
2. Открытый портал информационных ресурсов (научных статей, сборников работ и монографий по различным направлениям наук) https://elibrary.ru/project_risc.asp
3. База данных научных журналов на русском и английском языке ScienceDirect
4. Открытый доступ к метаданным научных статей по различным направлениям наук поиск рецензируемых журналов, статей, глав книг и контента открытого доступа http://www.sciencedirect.com/
5. Федеральный портал «Российское образование» http://www.edu.ru/
6. Бесплатная электронная библиотека онлайн «Единое окно доступа к образовательным ресурсам» http://window.edu.ru/
7. Единая коллекция цифровых образовательных ресурсов Научно-практические и методические материалы http://school-collection.edu.ru/
8. Единый реестр российских программ для электронных вычислительных машин и баз данных, в том числе свободно распространяемых, доступен по ссылке Reestr-Minsvyaz.ru
9. Сайт, посвященный SQL, программированию, базам данных, разработке информационных систем https://www.sql.ru/
10. На сайте проекта OpenNet размещается информация о Unix системах и открытых технологиях для администраторов, программистов и пользователей http://www.opennet.ru/
11. Библиотека программиста https://proglib.io
12. Сообщество IT-Специалистов https://habr.com/ru/
13. Сеть разработчиков Microsoft https://msdn.microsoft.com/ru-ru/
14. Сборник статей по информационной безопасности http://www.iso27000.ru/chitalnyi-zai

Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

1. Министерство образования и науки Российской Федерации. 100% доступ - http://минобрнауки.рф/
2. Федеральная служба по надзору в сфере образования и науки. 100% доступ - http://obrnadzor.gov.ru/
3. Федеральный портал «Российское образование». 100% доступ - http://www.edu.ru/
5. Федеральный центр информационно-образовательных ресурсов. 100% доступ - http://fcior.edu.ru/
6. Электронно-библиотечная система, содержащая полнотекстовые учебники, учебные пособия, монографии и журналы в электронном виде 5100 изданий открытого доступа. 100% доступ - http://bibliorossica.com/
7. Федеральная служба государственной статистики. 100% доступ - http://www.gks.ru

8. Программное обеспечение, используемое при осуществлении образовательного процесса по дисциплине

Операционная система Windows 10,
Microsoft office (Word, Excel, PowerPoint, Outlook, Publisher),
Microsoft Access,
Microsoft SQL Server Express Edition
SQL Server Management Studio

Microsoft Visual Studio
DB Designer Fork
1 С Предприятие (версия 8.3)
Антивирус Windows Defender (входит в состав операционной системы Microsoft Windows)

Программное обеспечение отечественного производства

INDIGO
Яндекс.Браузер

Свободно распространяемое программное обеспечение

Adobe Reader для Windows
Архиватор HaoZip

9. Материально-техническое обеспечение, необходимое для осуществления образовательного процесса по дисциплине

Для проведения учебных занятий используются учебные аудитории, оснащенные оборудованием и техническими средствами обучения: специализированной мебелью, отвечающей всем установленным нормам и требованиям; ПК с доступом к сети Интернет, переносным мультимедийным оборудованием, интерактивным комплексом, переносной аудио и видеоаппаратурой.

Для самостоятельной работы обучающихся используются помещения, оснащенные компьютерной техникой: персональные компьютеры с доступом к сети Интернет и ЭИОС института, принтеры; специализированной мебелью, отвечающей всем установленным нормам и требованиям.

Для обучения инвалидов и лиц с ограниченными возможностями здоровья институтом могут быть представлены специализированные средства обучения, в том числе технические средства коллективного и индивидуального пользования.

10. Методические рекомендации по обучению лиц с ограниченными возможностями здоровья

Профессорско-педагогический состав знакомится с психолого-физиологическими особенностями обучающихся инвалидов и лиц с ограниченными возможностями здоровья, индивидуальными программами реабилитации инвалидов (при наличии). При необходимости осуществляется дополнительная поддержка преподавания тьюторами, психологами, социальными работниками, прошедшими подготовку ассистентами.

В соответствии с методическими рекомендациями Минобрнауки РФ (утв. 8 апреля 2014 г. N АК-44/05вн) в курсе предполагается использовать социально-активные и рефлексивные методы обучения, технологии социокультурной реабилитации с целью оказания помощи в установлении полноценных межличностных отношений с другими студентами, создании комфортного психологического климата в студенческой группе. Подбор и разработка учебных материалов производятся с учетом предоставления материала в различных формах: аудиальной, визуальной, с использованием специальных технических средств и информационных систем.

Освоение дисциплины лицами с ОВЗ осуществляется с использованием средств обучения общего и специального назначения (персонального и коллективного использования). Материально-техническое обеспечение предусматривает приспособление аудиторий к нуждам лиц с ОВЗ.

Форма проведения аттестации для студентов-инвалидов устанавливается с учетом индивидуальных психофизических особенностей. Для студентов с ОВЗ предусматривается доступная форма предоставления заданий оценочных средств, а именно:

- в печатной или электронной форме (для лиц с нарушениями опорно-двигательного аппарата);
- в печатной форме или электронной форме с увеличенным шрифтом и контрастностью (для лиц с нарушениями слуха, речи, зрения);
- методом чтения ассистентом задания вслух (для лиц с нарушениями зрения).

Студентам с инвалидностью увеличивается время на подготовку ответов на контрольные вопросы. Для таких студентов предусматривается доступная форма предоставления ответов на задания, а именно:

- письменно на бумаге или набором ответов на компьютере (для лиц с нарушениями слуха, речи);
- выбором ответа из возможных вариантов с использованием услуг ассистента (для лиц с нарушениями опорно-двигательного аппарата);
- устно (для лиц с нарушениями зрения, опорно-двигательного аппарата).

При необходимости для обучающихся с инвалидностью процедура оценивания результатов обучения может проводиться в несколько этапов.

ПРИЛОЖЕНИЕ № 1:

«БАНК ТЕСТОВ»

1) Проектирование БД заключается
    a) определении структуры объектов
    b) в заполнении таблиц
    c) в архивировании БД

2) Для описания предметной области данные представляются в виде:
    a) концептуальной схемы
    b) внутренней схемы
    c) трехуровневой схемы

3) Физическая модель:
    a) производит структуризацию данных и выявляет взаимосвязи между ними
    b) определяет выбор модели данных, совместимой с выбранной СУБД
    c) определяет размещение данных, методы доступа и технику индексирования

4) Классификация БД по способу хранения данных:
    a) распределенные БД, централизованные БД
    b) централизованные БД, документальные БД
    c) фактографические БД, распределенные БД

5) Характеристики типов данных. Убери лишнее:
    a) текстовый
    b) дата/число
    c) денежный

6) Классификация БД по способу доступа к данным:
    a) с локальным доступом, с удаленным (сетевым) доступом
    b) с иерархическим доступом, с сетевым доступом
    c) с локальным доступом, с реляционным доступом

7) Количество возвращаемых записей в запросе ограничивается с помощью функции
    a) count
    b) sum
    c) avg

8) На концептуальном уровне при проектировании БД данные представляются в виде:
    a) сущностей, атрибутов, связей
    b) группирования данных, индексов, методов доступа
    c) записей, элементов данных, связей между записями

9) Что такое триггер?
    a) хранимая процедура
    b) транзакция
    c) представление

10) Какое средство необходимо использовать для миграции БД в SQL Server из других ядер СУБД?
    a) SSMA

b) DEA

c) DMA

11) Назовите четыре основных типа соединения в SQL

a) INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN

b) UNION, INNER JOIN, LEFT JOIN, RIGHT JOIN

c) UNION, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN

12) Для чего нужен оператор INSERT INTO SELECT?

a) Данный оператор копирует данные из одной таблицы и вставляет их в другую

b) Данный оператор подставляет данные в условие из выборки при вложенном запросе

c) Данного оператора не существует

13) В чем разница между командами DELETE и TRUNCATE?

a) DELETE удаляет записи из таблицы, а TRUNCATE полностью пересоздаёт таблицу

b) TRUNCATE удаляет записи из таблицы, а DELETE полностью пересоздаёт таблицу

c) DELETE удаляет записи из таблицы, а TRUNCATE полностью удаляет таблицу из БД

14) С помощью какого оператора можно переименовать таблицу?

a) UPDATE TABLE

b) ALTER TABLE

c) TRUNCATE TABLE

15) Что такое нормализация?

a) удаление избыточных данных, устранение дублей, идентификация наборов связанных данных через PRIMARY KEY и т.д.

b) конечное множество взаимосвязанных допустимых значений атрибутов, которые вместе описывают некоторую сущность

c) конечное множество атрибутов, определяющих некоторую сущность

ПРИЛОЖЕНИЕ 2

Задания для практических работ

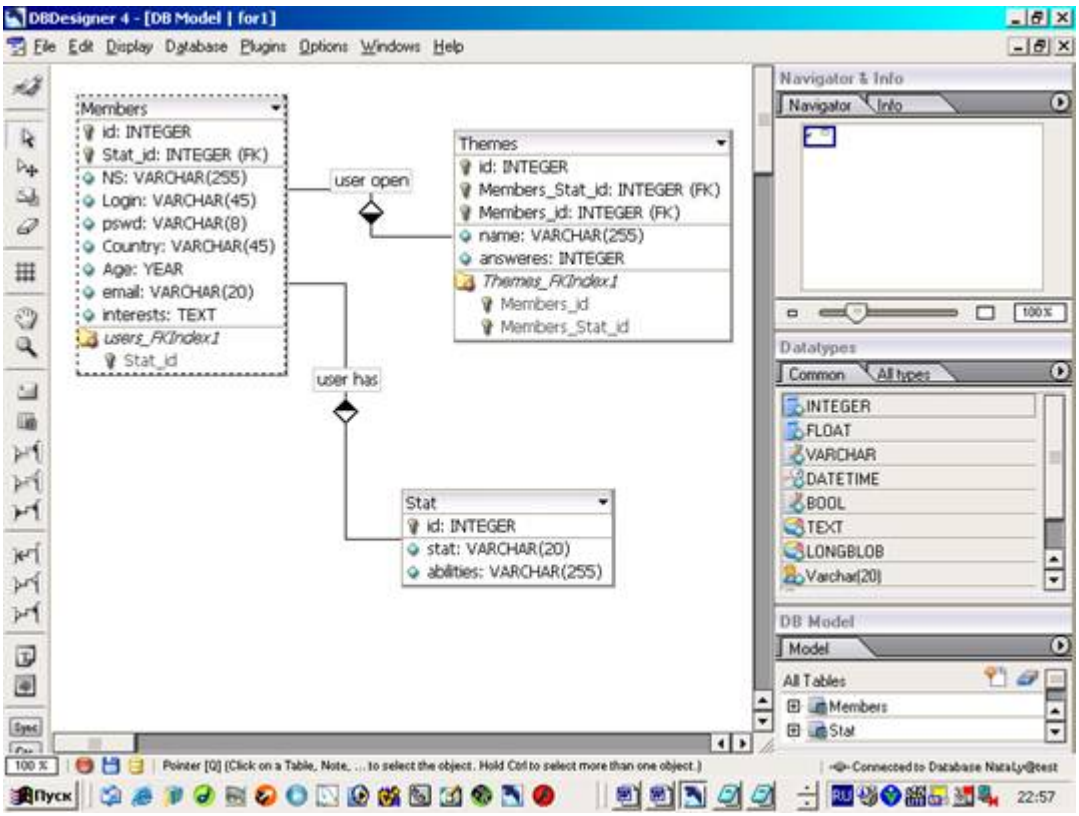
CASE-средства представляют собой программные средства, поддерживающие процессы создания и/или сопровождения информационных систем, такие как: анализ и формулировка требований, проектирование баз данных и приложений, генерация кода, тестирование, обеспечение качества, управление конфигурацией и проектом.

CASE-систему можно определить, как набор CASE-средств, имеющих определенное функциональное предназначение и выполненных в рамках единого продукта.

DBDesigner – это свободно распространяемая CASE-система, предназначенная для проектирования, моделирования, создания и поддержки информационных систем. Программа может использоваться для Windows 2000/XP, Linux KDE/Gnome и MySQL. DBDesigner позволяет:

- создавать модель проектируемой системы;
- преобразовывать модели системы в SQL-код, который можно использовать для создания базы данных с помощью DBDesigner или другого средства;
- проводить реинжиниринг – построение исходной модели программной системы путем исследования ее программных кодов. Эта функция очень удобна в случае, если необходимо разобраться уже существующей базе данных. Для проведения реинжиниринга следует выбрать в меню Database – Revers Engineering;
- создавать базу данных и автоматически вносить в нее изменения, используя соединение с сервером и синхронизацию;
- оздавать SQL-запросы для внесения изменений и проведения операций над данными.

Пользовательский интерфейс программы:



ВЫПОЛНЕНИЕ

Предметной областью разрабатываемой базы системы является Форум. На форуме зарегистрировано несколько участников, которые открывают темы и оставляют сообщения на форуме. В зависимости от частоты посещения каждый участник имеет определенный статус. Чем он выше, тем больше привилегий имеет участник. В базе данных содержится личная информация каждого из участников.

Необходимо обеспечить возможность внесения, изменения или удаления данных в базе и проведение различных поисковых операций: поиск по названию темы, по псевдониму участника.

1. Моделирование

Модель – это визуальное представление структуры данных. Модель может включать в себя следующие объекты: таблицы и отношения, которые используются обязательно, и дополнительные (например, изображения, записи) – для обеспечения лучшего «понимания» структуры модели.

Для создания модели необходимо переключится в Design Mode, выбрав меню **Display - Design Mode**.

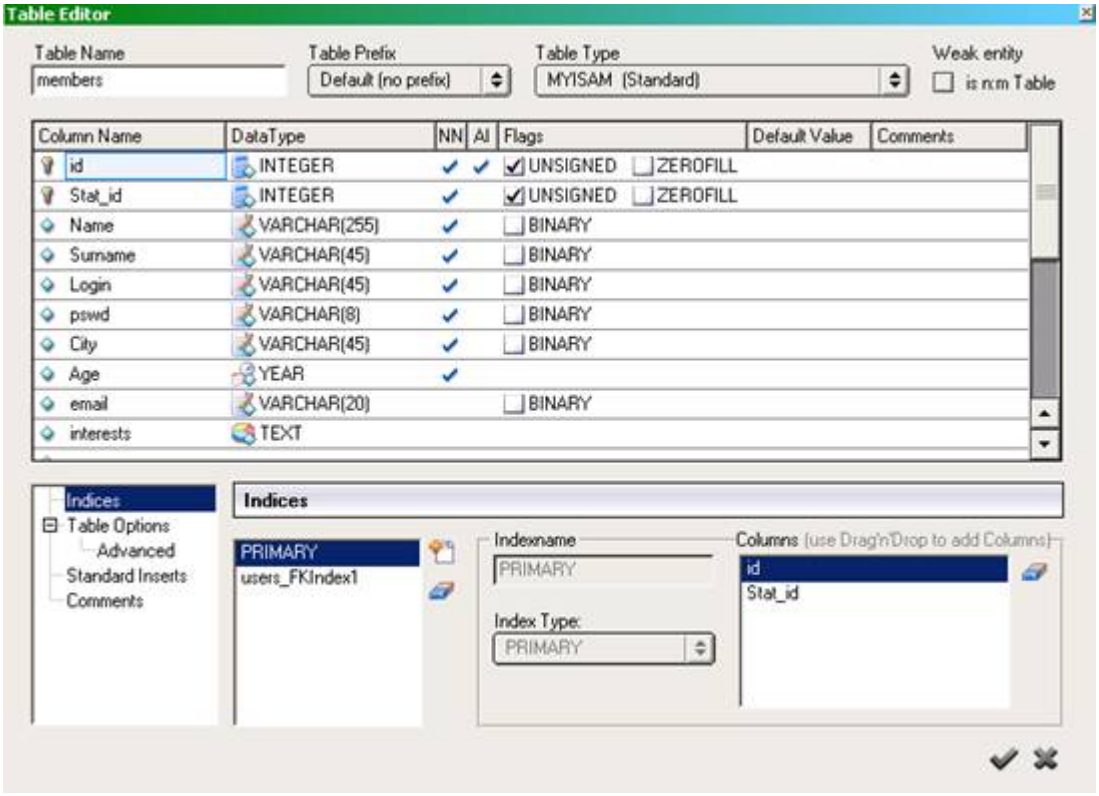
Пользовательский интерфейс делает создание модели базы данных очень легким.

DBDesigner 4 поддерживает Multible Document Interface(MDI), который позволяет открывать неограниченное число моделей одновременно. При работе, вы можете переключаться между моделями, копируя команды и объекты, чтобы обмениваться ими между моделями.

1) Создание таблиц

инструментов. Необходимо нажать на этой панели кнопку  и указать место на холсте, где будет располагаться новая сущность. Появится прямоугольное изображение пустой сущности. Чтобы задать атрибуты сущности, необходимо два раз щелкнуть на изображении сущности. В появившемся окне можно задать название сущности, а также атрибуты этой сущности.

В данной лабораторной работе область представляет собой форум с некоторым количеством участников, темами, которые они открывают их статусом. была для отображения личной информации об участниках форума была создана сущность **Members**. Структура сущности M**embers** показана на рисунке:



Для этой сущности введены следующие атрибуты:

Идентификатор участника (Members_id) (ключевой атрибут), Имя участника (Members_name), Фамилия участника (Members_surname), Логин участника в форуме (Members_login), Пароль участника при входе на форум (Members_pswd), Город, в котором проживает участник (Members_city) Возраст участника (Members_age), Электронная почта участника (Members_email), Интересы участника (Members_interests),

Флаг в поле NN означает, что содержимое данного поля не может быть нулевым (Not Null). Флаг в поле AI означает, что значение данного поля в каждой следующей строке увеличивается на 1 (Auto Increment). Иконка напротив имени атрибута означает, что этот атрибут является ключевым.

Были созданы еще две сущности: статус участника (Stat) и темы форума (Themes)

Атрибуты сущности **Stat**:

> Порядковый номер статуса (Stat_id) (ключевой атрибут), Название статуса (Stat_stat), Возможности участника, обладающего этим статусом (Stat_abilities).

Атрибуты сущности **Themes**:

> Название темы (Themes_name) (ключевой атрибут), Число ответов (Themes_ans).

2) Формирование отношений

Связь между сущностями определяет связь между будущими таблицами. Для этого необходимо

поставить флаг напротив после всех полей в разделе параметры отношений и внешних ключей (раздел **Default Relation Settings / Foreign Keys Settings**) во вкладке редактирования модели (**Editing Options**),

Между сущностями имеются следующие связи:

Участник –> открыл -> **Тему**

Участник может открыть несколько тем, каждая тема может быть открыта только одним участником. Участник может не открывать ни одной темы, тема обязательно должна быть открыта кем-то. Имеем связь 1:m, класс принадлежности H:O.

Участник -> имеет -> **Статус**

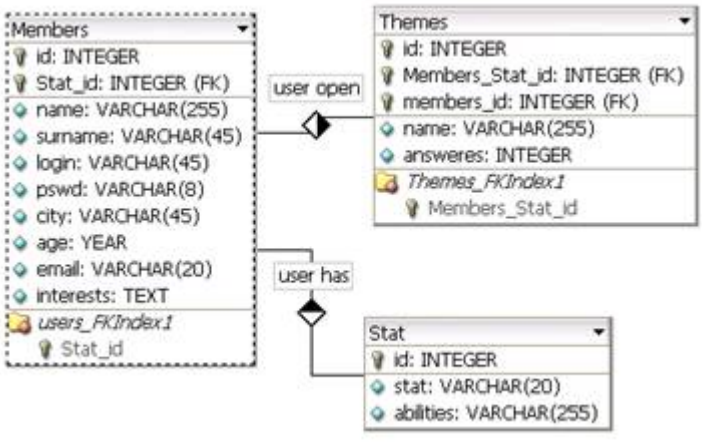
Участник в данный момент времени может иметь только один статус, один и тот же статус могут иметь несколько участников. Каждый участник должен иметь статус, но статус не обязательно должен принадлежать кому-то. Имеем связь m;1, класс принадлежности O:H.

В программе связи задаются следующим образом.

Связь 1:1 задается с помощью кнопки  . Связь 1:n задается с помощью кнопки  . Связь n:m задается с помощью кнопки  .

Задать связь между сущностями можно, нажав на соответствующую кнопку и указав связываемые таблицы. После нажатия кнопки связи, надо нажать на первую таблицу, участвующую в связи, затем на другую. Внешние ключи будут автоматически добавлены в сущности соответственно связи.

Результат связывания сущностей показан на рисунке:



Двойным щелчком по изображению связь можно редактировать свойства связи, такие как название связи и тип связи.

Особенностью программы DBDesigner является то, что в процессе создания ER-диаграмм отношения будет сформированы автоматически. Это очень удобно при проектировании, т. к.

позволяет разработчику не запоминать правила формирования отношений для различных связей между таблицами, а получать все автоматически на основе анализа предметной области.

Связи между таблицами можно корректировать, используя «Редактор связей» (**Relation Editor**), вызываемый двойным щелчком мыши. В «Редакторе связей» можно задать имя связи, изменить ее тип и задать ограничения на данные таблицы при удалении и добавлении в нее данных.

2. Кодирование

DBDesigner позволяет преобразовывать полученную модель в код на языке SQL, который может быть использован для создания базы данных с помощью других средств, например, с помощью MySQL.
Для получения кода необходимо выбрать в меню **File – Export – SQL Create Script**. Откроется диалоговое окно

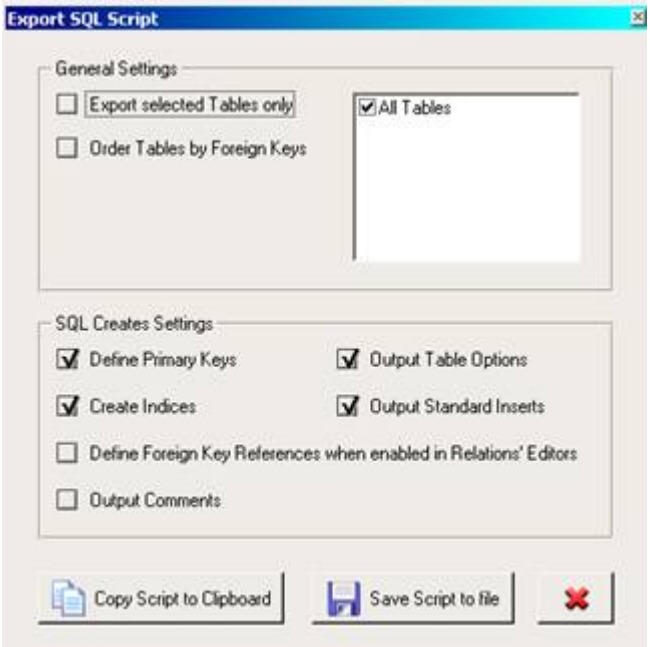


Рис. Кодирование

В основных настройках (**General Settings**) можно назначить экспортировать в SQL код только выделенные таблицы или экспортировать все таблицы модели, также можно задать упорядочить таблицы по внешним ключам.
- Export selected tables only – кодировать только выбранные таблицы
- **Order Tables by Foreign Keys** – позволяет изменить порядок кодирования
В настройках SQL кода (**SQL Creates Settings**) можно настроить параметры связанные с первичными ключами и внешними ключами, а также задать настройки относительно индексов.
- **Copy Script to Clipboard**. Позволяет скопировать SQL код в буфер обмена;-
- **Save Script to file**. Позволяет сохранить SQL код в файл. Файл сохраняется в формате *.sql. Открыть его можно и в текстовом редакторе «Блокнот»
Выбрав необходимые параметры, необходимо нажать Save Script to file. Файл с SQL кодом будет сохранен на диске.

```
CREATE TABLE Members (
id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
Stat_id INTEGER UNSIGNED NOT NULL,
name VARCHAR(255) NOT NULL,
surname VARCHAR(45) NOT NULL,
login VARCHAR(45) NOT NULL,
```

pswd VARCHAR(8) NOT NULL,
city VARCHAR(45) NOT NULL,
age YEAR NOT NULL,
email VARCHAR(20) NULL,
interests TEXT NULL,
PRIMARY KEY(id, Stat_id),
INDEX users_FKIndex1(Stat_id)
CREATE TABLE Stat (
id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
stat VARCHAR(20) NULL,
abilities VARCHAR(255) NULL,
PRIMARY KEY(id)
CREATE TABLE Themes (
id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
Members_Stat_id INTEGER UNSIGNED NOT NULL,
members_id INTEGER UNSIGNED NOT NULL,
name VARCHAR(255) NOT NULL,
answeres INTEGER UNSIGNED NULL,
PRIMARY KEY(id, Members_Stat_id, members_id),
INDEX Themes_FKIndex1(Members_Stat_id)

3. Работа с базой данных

DBDesigner позволяет также создавать базу данных на сервере и выполнять с ней различные операции. Это обеспечивается за счет подключения DBDesigner к MySQL серверу, созданию базы данных и установлению синхронизации между базой на сервере и визуальной моделью. Синхронизация – это сравнение визуальной модели и базы данных, находящейся на сервере. В случае внесения изменений в таблицу, изменения связей между таблицами или удаления таблиц в модели, DBDesigner внесет и соответствующие изменения в базу на сервере.

1) Установление соединения с базой данных на сервере

Для занесения базы данных, соответствующей полученной модели, на сервер MySQL, необходимо установить соединение с сервером.

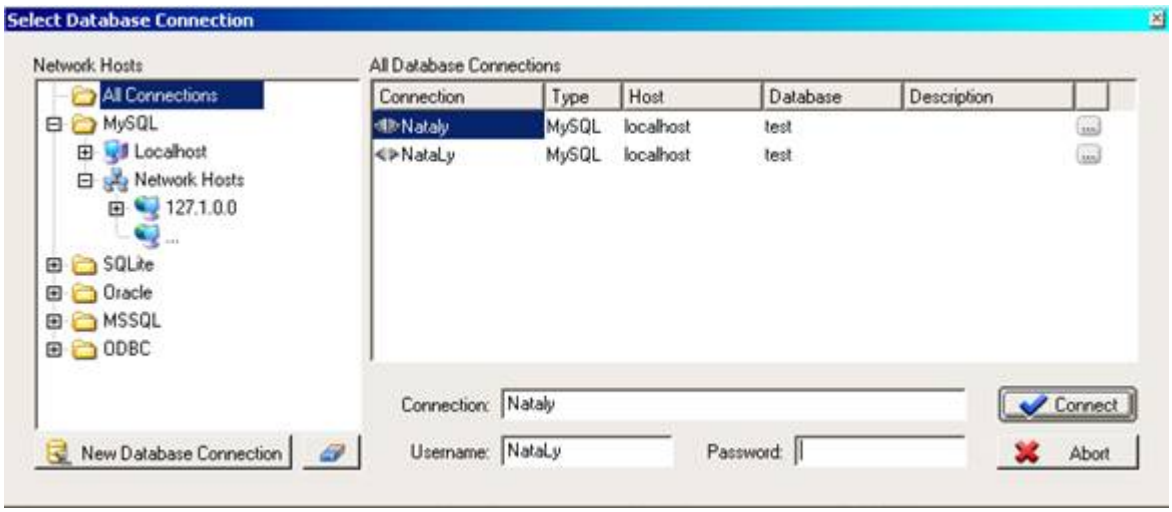


Рис. Установление соединения с сервером

- Выполните **Database –> Connect to Database**.
- В окне **Network Hosts** выберите MySQL
- В открывшемся списке баз данных, выберите либо существующую базу, либо создать новую, щелкнув два раза по значку «…» и задав имя новой базы.
- Введите название соединения (**Connection**), имя пользователя (**Username**) и пароль (**Password**), если они нужны.

- В центральном окне находится список серверов баз данных, с которыми велась работа и для которых указаны IP-адрес, тип, размещение и название. Так как в данной работе предполагается, что сервер MySQL находиться на локальном компьютере, то все необходимые параметры будут установлены автоматически. Однако при использовании сети, необходимо знать IP-адрес сервера и иметь доступ на работу.
- Нажмите на кнопку **Connect**, после чего соединение с базой будет установлено.

1) Синхронизация

Для синхронизации модели и базы на сервере необходимо:
- Выбрать в меню **Database - Database Synchronisation** и установить соединение с нужной базой.
- В диалоговом окне Database Synchronisation задать необходимые параметры:

o **Apply changes to Database** – вносить изменения модели в базу

- **Don't delete exisiting Tables** – при использовании этой опции таблицы, удаленные из модели, не будут удалены из базы

o **Execute Standard Inserts when Creating New Tables** – создавать стандартный запрос на внесение данных в таблицу
- Нажать **Execute,** после чего база данных будет занесена на сервер. Также будет выведен отчет и сообщения об ошибках в модели, если они есть.

Рис. Занесение базы данных на сервер и установление синхронизации

4. SQL-запросы

DBDesigner также позволяет создавать запросы на языке SQL. Причем код запроса можно либо непосредственно написать, либо использовать готовые шаблоны, в которые необходимо только внести какие-то изменения.

Для работы с запросами необходимо:
- Переключиться в Query Mode, выбрав в меню **Display -> Query Mode**.
- В меню инструментов слева появятся кнопки, с помощью которых можно выполнить основные запросы.
- Выбрав кнопку (например, SELECT), следует щелкнуть по заголовку таблицы, а затем, не отпуская кнопку мыши, сдвинуть указатель вниз.
- В появившемся меню выбрать нужную операцию.

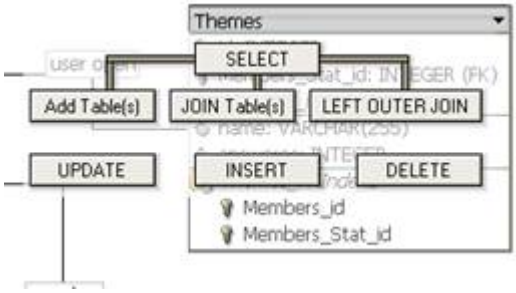


Рис. SQL-запросы

- Код на языке SQL появиться в нижней части экрана.

```
SELECT *
FROM razdel
```

- Нажав на кнопку **Execute SQL Query**, в нижней части экрана можно увидеть результат запроса.

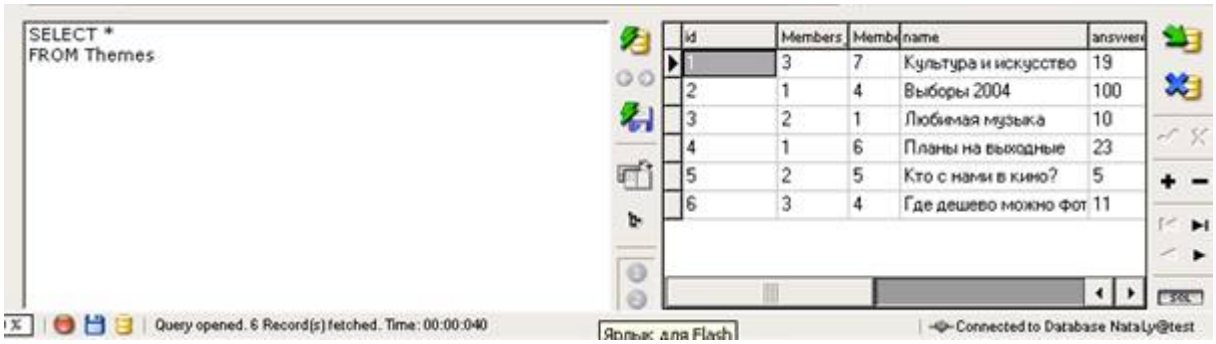


Рис. Результат выполнения запроса

DBDesigner предоставляет различные функции для работы с запросами: сохранение кода, внесение изменений в базу и отмена внесенных изменений. Благодаря этим встроенным функциям работа с запросами существенно упрощается.

Кроме того, в программе есть очень удобное средство для внесения данных в таблицу. Щелкнув правой кнопкой мыши по таблице и выбрав в меню Edit Table Data, можно заносить данные в таблицу или изменять их без использования языка SQL.

ПРИЛОЖЕНИЕ 3

Задания для лабораторных работ

Лабораторная работа 1
СОЗДАНИЕ БАЗЫ ДАННЫХ В SQL SERVER MANAGEMENT STUDIO

Цель работы: научиться создавать базы данных в среде SQL Server Management Studio.

Теоретическая часть

Родоначальником серии SQL Server и его основой является язык запросов SQL. Данный язык был предложен сотрудником компании IBM Эдгаром Коддом в начале 1970-х гг. Изначально он назывался SEQUEL (Structured English Query Language, структурированный английский язык для запросов), который впоследствии по юридическим соображениям был переименован в SQL (Structured Query Language, структурированный язык запросов).

Целью разработки было создание простого непроцедурного языка, которым мог воспользоваться любой пользователь, даже не имеющий навыков программирования. К началу 1980-х годов SQL завоевал популярность как язык реляционных систем управления базами данных (СУБД) и привлек внимание Американского национального института по стандартизации (American National Standards Institute, ANSI), который в 1986, 1989, 1992, 1999 и 2003 годах выпустил стандарты языка SQL. В 1989 году SQL был включен в стандарты международной организации по стандартизации ISO (SQL:1989), а затем были приняты и опубликованы стандарты SQL:1992, SQL:1999 и SQL:2003. В настоящее время все производители распространенных реляционных СУБД поддерживают с различной степенью соответствия стандарт SQL:2003. В основу языка SQL, используемого в SQL Server, легла разновидность языка T-SQL (Transact- SQL).

В начале 1980-х гг. фирма IBM и, в частности, в то время ее подразделениями Microsoft и Sybase была создана первая версия сетевой СУБД, которая называлась SQL Server версия 1.0, для операционной системы IBM OS/2. После этого под эту операционную систему было выпущено еще 3 версии SQL Server. В середине 1980-х гг. компании Microsoft и Sybase отделяются от фирмы IBM, и Microsoft начинает работу над своей операционной системой Windows, и вместе с компанией Sybase продолжает развитие SQL Server. В середине 1990-х гг. Microsoft создала операционную систему Windows NT и вместе с компанией Sybase выпустила первую версию SQL Server для Windows версии 4.1.

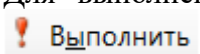
В дальнейшем компания Sybase разорвала свои отношения с Microsoft, и Microsoft сама продолжила работу над Microsoft SQL Server 6.0. Данная

версия была предназначена для работы в операционной системе Windows NT, 95 и 98. В 1999 г. выходит версия Microsoft SQL Server 7.0, которая стала одной из самых популярных серверных СУБД в мире. В 2000 г. выходит 8-я версия Microsoft SQL Server 2000. В 2005 году появляется новая версия сервера, основанная на новой технологии .NET, а в 2008 году - её улучшенная версия - Microsoft SQL Server 2008.

В СУБД Microsoft SQL Server новую базу данных (БД) можно создать, используя стандартные команды языка T-SQL.

Для создания новой БД с помощью T-SQL необходимо вначале перейти в БД «Master». Это можно сделать либо выбором ее из выпадающего списка баз данных на панели инструментов, либо набором команды USE Master на вкладке нового запроса.

Все команды языка T-SQL набираются на вкладке нового запроса (SQLQuery). Для того, чтобы создать новый запрос на панели инструментов необходимо нажать кнопку 

. Для выполнения команд языка T- SQL на панели инструментов необходимо нажать кнопку 

или на вкладке нового запроса набрать команду GO.

В Microsoft SQL Server БД состоит из двух частей:

- файл данных - файл, имеющий расширение mdf и где находятся все таблицы и запросы;
- файл журнала транзакций - файл, имеющий расширение ldf, содержит журнал, где фиксируются все действия с БД. Данный файл предназначен для восстановления БД в случае её

выхода из строя.

Для создания нового файла данных используется SQL-команда CREATE DATABASE, которая имеет следующий синтаксис:

CREATE DATABASE <Имя БД>
(Name=<Логическое имя>, FileName=<Имя файла> [Size=<Нач. размер>,] [Maxsixe=<Макс. размер>,] [FileGrowth=<Шаг>])
[LOG ON
(Name=<Логическое имя>, FileName=<Имя файла> [Size=<Нач. размер>,] [Maxsixe=<Макс. размер >,] [FileGrowth=<Шаг>])

Здесь Имя БД - имя создаваемой БД; Логическое имя - определяет логическое имя файла данных БД, по которому происходит обращение к файлу данных; Имя файла - определяет полный путь к файлу данных; Нач. размер - начальный размер файла данных в МБ; Макс. размер - максимальный размер файла данных в МБ; Шаг - шаг увеличения файла данных, либо в МБ, либо в %.

Параметры в разделе LOG ON аналогичны параметрам в разделе CREATE DATABASE. Однако они определяют параметры журнала транзакций.

Пример: Создать БД «Artworks», расположенную в файле «D:\Artworks.mdf» и имеющую начальный размер файла данных - 3 МБ, максимальный размер файла данных - 100 МБ и шаг увеличения файла данных, равный 1 МБ. Файл журнала транзакций данной БД имеет имя «ArtworksLog» и расположен в файле «D:\Artworks.ldf». Данный файл имеет начальный размер, равный 1 МБ, максимальный размер, равный 20 МБ и шаг увеличения, равный 1 МБ.

```
CREATE DATABASE Artworks ON (Name = Artworks,
FileName = 'D:\ Artworks.mdf, Size = 3MB,
Maxsixe = 100MB, FileGrowth= 1MB)
LOG ON
(Name = Artworks Log,
FileName = 'D:\ Artworks.ldf, Size = 1MB,
Maxsixe = 20MB, FileGrowth = 1MB)
```

В языке запросов T-SQL с БД возможны следующие действия:

1. Отображение сведений о БД: EXEC SP_HELPDB <Имя БД>;
2. Изменение параметров БД: EXEC SP_DBOPTION <Имя БД>, <Параметр>, <Значение>;
3. Добавление новых файлов, удаление файлов и переименование файлов, входящих в БД:
   ALTER DATABASE <Имя БД>
   ADD FILE (<Параметры>)|
   REMOVE FILE <Логическое имя файла>| MODIFY FILE (<Параметры>)
   ,где раздел ADD FILE - добавляет файл,
   REMOVE FILE - удаляет,
   MODIFY FILE - изменяет параметры файла;
4. Сжатие всей БД:
   DBCC SHRINKDATABASE <Имя БД>;
5. Сжатие конкретного файла БД:
   DBCC SHRINKFILE <Логическое имя файла>;
6. Переименование БД:
   EXEC SP_RENAMEDB <Имя БД>,<Новое имя БД>;
7. Удаление БД:
   DROP DATABASE <Имя БД>.

Вышеперечисленные команды используют следующие параметры:

- <Имя БД> - имя БД, с которой производится действие;
- <Параметр> - изменяемый параметр;
- <Значение> - новое значение изменяемого параметра;
- <Параметры> - параметры файла БД, аналогичные параметрам, используемым в команде CREATE DATABASE;
- <Логическое имя файла> - логическое имя файла, входящего в БД;

- <Новое имя БД> - новое имя БД.

Пример выполнения лабораторной работы

Для начала запустим среду разработки «SQL Server Management Studio». После запуска среды разработки появится окно подключения к серверу (рис. 1)

«Соединение с сервером» (рис. 1).

В этом окне необходимо нажать кнопку «Соединить». После нажатия кнопки «Соединить» появится окно среды разработки «SQL Server Management Studio» (рис. 2).

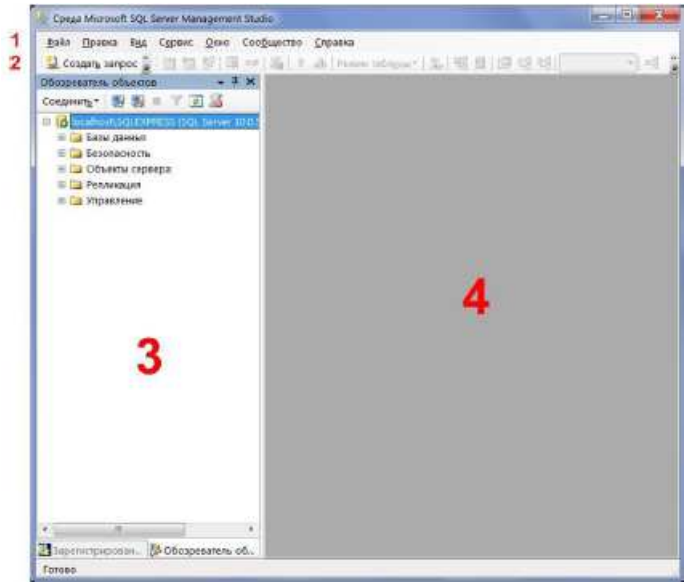
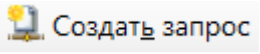


Рис. 2

Данное окно имеет следующую структуру:

1. Оконное меню - содержит полный набор команд для управления сервером и выполнения различных операций.
2. Панель инструментов - содержит кнопки для выполнения наиболее часто производимых операций. Внешний вид данной панели зависит от выполняемой операции.
3. Панель «Обозреватель объектов» - обозреватель объектов. Обозреватель объектов — это панель с древовидной структурой, отображающая все объекты сервера, а также позволяющая производить различные операции, как с самим сервером, так и с БД. Обозреватель объектов является основным инструментом для разработки БД.
4. Рабочая область. В рабочей области производятся все действия с БД, а также отображается её содержимое.

В обозревателе объектов сами объекты находятся в папках. Чтобы открыть папку, необходимо щёлкнуть по знаку «+» слева от изображения папки.

Нажатие кнопки 

приводит к открытию окна запросов (рис. 3)

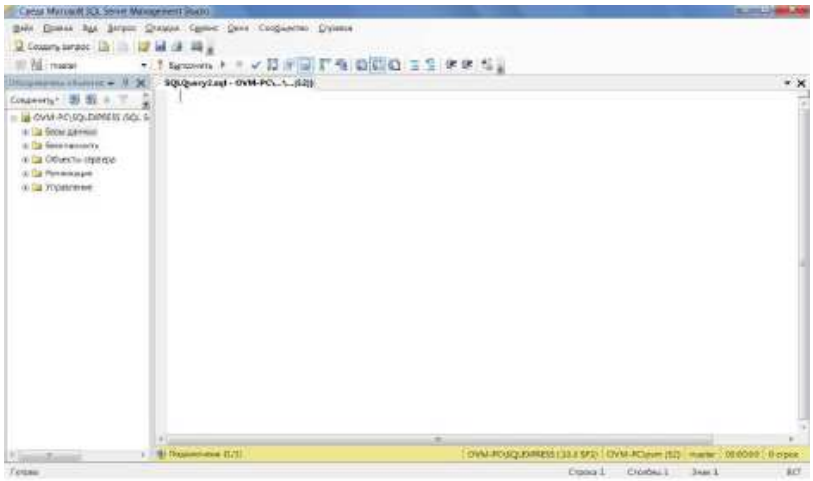
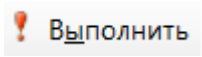


Рис. 3

Для выполнения запроса, введенного в этом окне, нужно нажать на кнопку 

.

Выполним создание базы данных «Artworks» в среде разработки SQL Server Management Studio. В новом окне запросов введем запрос на создание БД:

IF DB_ID('Artworks') IS NULL CREATE DATABASE Artworks;

Если базы данных с именем **Artworks** не существует, приведенный код создаст новую БД. Функция **DB_ID** принимает имя базы данных в качестве входного параметра и возвращает внутренний ID базы данных. Если БД с именем входного параметра не существует, функция возвращает значение NULL. Это простой способ проверить наличие заданной БД.

В инструкции **CREATE DATABASE** используются установочные параметры файла для задания его местоположения и начального размера.

После выполнения данного запроса в окне среды разработки SQL Server Management Studio на панели обозревателя объектов в папке «Базы данных» появится новая БД **Artworks** (рис. 4).

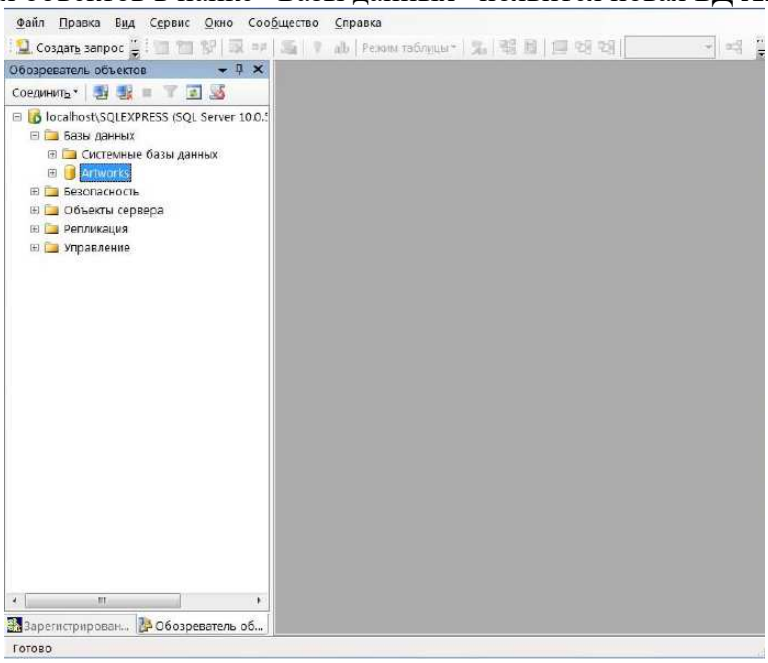


Рис. 4

Вся информация в базе данных хранится в таблицах, которые представляют собой средство

хранения данных. Таблицы состоят из строк или записей и столбцов или полей. Каждое поле имеет три характеристики:
1. Имя поля - используется для обращения к полю;
2. Значение поля - определяет информацию, хранимую в поле;
3. Тип данных поля - определяет, какой вид информации можно хранить в поле.

В SQL сервер используется следующие типы данных:

• **Двоичные типы данных,** которые содержат последовательности нулей и единиц:
1) binary(n) - двоичный тип фиксированной длины размером в n байт, где n — значение от 1 до 8000; размер при хранении составляет n байт;
2) varbinary(n) - двоичный тип с переменной длиной, n может иметь значение от 1 до 8000, **max** указывает, что максимальный размер при хранении составляет 231-1 байт;
• **Целочисленные типы данных** - типы данных для хранения целых чисел (в скобках указан диапазон значений типа данных): tinyint (0..255), smallint (-32768..+32767), int (-231..+(231-1)), bigint (-263..+(263- 1));
• **Типы данных для хранения чисел с плавающей запятой:** real занимает в памяти 4 байта; float(n), где n — это количество битов, используемых для хранения мантиссы числа в формате **float** при экспоненциальном представлении, определяет точность данных и размер для хранения; значение параметра n должно лежать в пределах от **1** до **53;** значением по умолчанию для параметра n является **53;**
• **Типы данных для хранения чисел с фиксированными точностью и масштабом:**
decimal(p, s) и numeric(p, s), где p (точность) - максимальное количество десятичных разрядов числа (как слева, так и справа от десятичной запятой). Точность должна быть значением в диапазоне от 1 до 38. По умолчанию это значение равно 18. s (масштаб)
- максимальное количество десятичных разрядов числа справа от десятичной запятой. Масштаб может принимать значение от 0 до p и может быть указан только совместно с точностью. По умолчанию масштаб принимает значение 0; поэтому 0 < s < p;
• **Символьные типы данных:** char(n) - строковые данные фиксированной длины не в Юникоде, аргумент n определяет длину строки и должен иметь значение от 1 до 8000, размер при хранении составляет n байт; varchar(n | **max**) - строковые данные переменной длины не в Юникоде, аргумент n определяет длину строки и должен иметь значение от 1 до 8000, значение **max** указывает, что максимальный размер при хранении составляет 231-1 байт (2 ГБ); text - данные переменной длины не в Юникоде в кодовой странице сервера и с максимальной длиной строки 231-1;
• **Специальные типы данных:** bit - целочисленный тип данных, который может принимать значения 1, 0 или NULL; image - тип данных для хранения рисунка размером до 2ГБ;
• **Типы данных даты и времени:** date (от 01.01.0001 до 31.12.9999); datetime (диапазон даты - от 01.01.1753 до 31.12.1999, диапазон времени - от 00:00:00 до 23:59:590,997); smalldatetime (диапазон даты -от 01.01.1900 до 6.06.2079, диапазон времени - от 00:00:00 до 23:59:59); time (от 00:00:00.0000000 до 23:59:59.9999999);
• **Денежные типы данных для хранения финансовой информации:** money (8 байт) и smallmoney (4 байта) - типы данных, представляющие денежные (валютные) значения.

Для создания таблиц в SQL Server в первую очередь необходимо сделать активной ту БД, в которой создается таблица. Для этого в новом запросе можно набрать команду: USE <Имя БД>, либо на панели инструментов необходимо выбрать в выпадающем списке рабочую БД. После выбора БД можно создавать таблицы.
Таблицы создаются командой

CREATE TABLE table_name
({ <column_definition> }) [;]
<column_definition> ::= column_name <data_type> [NULL
| NOT NULL] [DEFAULT constant_expression] | [IDENTITY [(seed,increment)]] [
<column_constraint> [...n]]
<column_constraint> ::= [CONSTRAINT constraint_name]
{ PRIMARY KEY | UNIQUE }

Здесь table_name - имя таблицы; column_name - имя столбца в таблице; data_type - тип

данных для столбца; IDENTITY указывает, что новый столбец является столбцом идентификаторов, при этом seed - значение, используемое для самой первой строки, загружаемой в таблицу, increment - значение приращения, добавляемое к значению идентификатора предыдущей загруженной строки; CONSTRAINT - необязательное ключевое слово, указывающее на начало определения ограничения, constraint_name - имя ограничения; NULL | NOT NULL определяет, допустимы ли для столбца значения NULL; PRIMARY KEY - ограничение, которое определяет столбец первичным ключом таблицы.

Если имя поля содержит пробел, то оно заключается в квадратные скобки.

Пример: Создать таблицу «Artworks», содержащую поля:
а) код произведения (**ArtworkId**);
б) название произведения (**Title**); в) жанр (**Genre**);
г) средства создания (**Tools**); д) код автора (**Authorld**);
е) дата создания (**CreatDate**);
ж) цена (**Price**;
з) отдел хранения (**DepId**).
SQL-запрос для создания этой таблицы имеет следующий вид:

```
USE Artworks;
IF OBJECT_ID('dbo.Artworks', 'U') IS NOT NULL DROP TABLE dbo.Artworks;
CREATE TABLE dbo.Artworks (
ArtworkId BIGINT IDENTITY(1,1) CONSTRAINT PK_Artworks PRIMARY KEY,
Title VARCHAR(100) NULL, Genre VARCHAR (50) NULL, Tools VARCHAR (50) NULL,
AuthorId BIGINT NULL, CreatDate DATE NULL,
Price MONEY NULL, DepId INT NOT NULL );
```

Инструкция **USE** изменяет текущую связь с БД на связь с **Artworks**.

Включение этой инструкции в сценарии создания объектов очень важно, т.к. гарантирует создание объектов в требуемой БД.

Инструкция **IF** запускает функцию **OBJECT_ID**, которая в качестве входных параметров принимает имя объекта и его тип. Тип **'U'** представляет пользовательские таблицы. Данная функция возвращает внутренний ID объекта, если объект с заданным именем и типом уже существует, и значение **NULL** в противном случае.

При создании таблицы используется схема с именем **dbo**, которая создается автоматически в каждой базе данных и используется как схема по умолчанию. Если опустить имя схемы при создании таблицы, то SQL Server свяжет с таблицей схему, используемую по умолчанию для имени пользователя, выполняющего программный код.

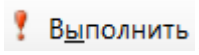
Для каждого атрибута сущности **Artworks** задается его имя, тип и допустимость значений **NULL**.

Для столбца **ArtworkId** определено ограничение в виде первичного ключа (**PK_Artworks**), при этом значения **ArtworkId** будут начинаться с 1 и увеличиваться при каждом добавлении новых строк в таблицу тоже на 1 (**IDENTITY(1,1)**).

Аналогично создаются и другие таблицы БД **Artworks** : **Authors**, **Employees**, **Departments**. SQL-запросы для создания этих таблиц приведены ниже.

```
USE Artworks;
IF OBJECT_ID('dbo.Authors', 'U') IS NOT NULL DROP TABLE dbo.Authors;
CREATE TABLE dbo.Authors (
AuthorId BIGINT IDENTITY(1,1) CONSTRAINT
PK_Authors PRIMARY KEY, Lastname
VARCHAR(25) NOT NULL,
Firstname VARCHAR (25) NOT NULL,
Middlename VARCHAR (25) NULL, DateOfBirth DATE NULL, DateOfDeath DATE NULL,
Country VARCHAR(25) NULL );
IF OBJECT_ID('dbo.Employees', 'U') IS NOT NULL DROP TABLE dbo.Employees;
CREATE TABLE dbo.Employees (
EmpId BIGINT IDENTITY(1,1) CONSTRAINT
```

```
PK_Employees PRIMARY KEY, Lastname
VARCHAR(25) NOT NULL,
Firstname VARCHAR (25) NOT NULL,
Middlename VARCHAR (25) NOT NULL, Position VARCHAR (25) NULL, Salary
MONEY NULL, BeginDate DATE NOT NULL, EndDate DATE NULL, DepId INT
NULL );
IF OBJECT_ID('dbo.Departments', 'U') IS NOT NULL DROP TABLE dbo.Departments;
CREATE TABLE dbo.Departments (
DepId INT IDENTITY(1,1) CONSTRAINT
PK_Departments PRIMARY KEY, Name
VARCHAR(25) NOT NULL );
```

Для выполнения запросов, введенных в среде SQL Server Management Studio, необходимо нажать на кнопку 

. .

Для того чтобы обеспечить ссылочную целостность в БД **Artworks** нужно добавить в созданные таблицы ограничение по внешним ключам.

Таблицы **Artworks** и **Authors** нужно связать по столбцу **AuthorId**, а таблицы **Artworks** и **Departments** - по столбцу **DepId.** Аналогично должны быть связаны между собой таблицы **Employees** и **Departments.**

```
USE Artworks;
ALTER TABLE dbo.Artworks
ADD CONSTRAINT FK_Artw_Auth FOREIGN KEY (AuthorId) REFERENCES Authors
(AuthorId);
ALTER TABLE dbo.Artworks
ADD CONSTRAINT FK_Artw_Dep FOREIGN KEY (DepId)
REFERENCES Departments (DepId);
ALTER TABLE dbo.Employees
ADD CONSTRAINT FK_Emp_Dep FOREIGN KEY (DepId)
REFERENCES Departments (DepId);
```

Теперь, когда установлены связи между всеми таблицами, можно создать диаграмму, описывающую эти взаимосвязи.

Для создания диаграммы нужно щелкнуть правой кнопкой мыши на элементе БД «Диаграммы баз данных» и в открывшемся контекстном меню выбрать пункт «Создать диаграмму базы данных» (рис. 5).

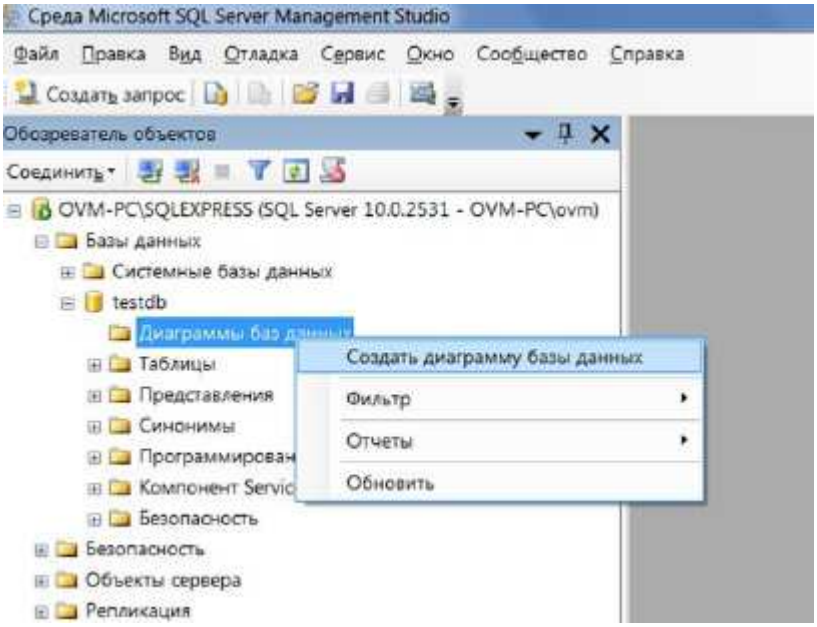


Рис. 5

В открывшимся диалоговом окне необходимо выбрать таблицы, которые будут добавлены на диаграмму (рис. 6)

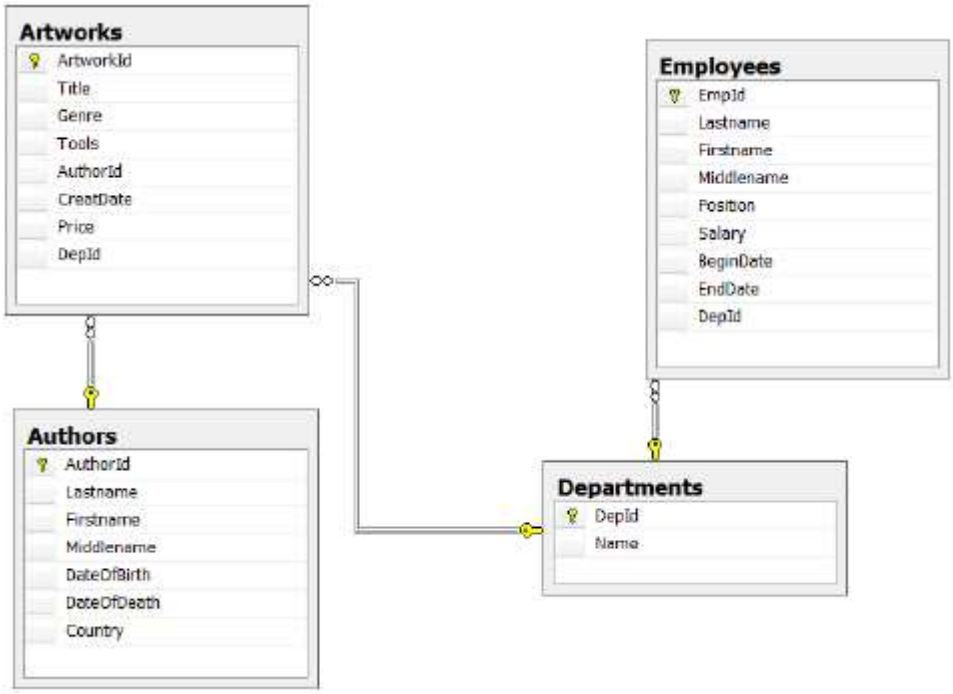


Рис. 6

В результате будет получена диаграмма следующего вида (рис. 7)

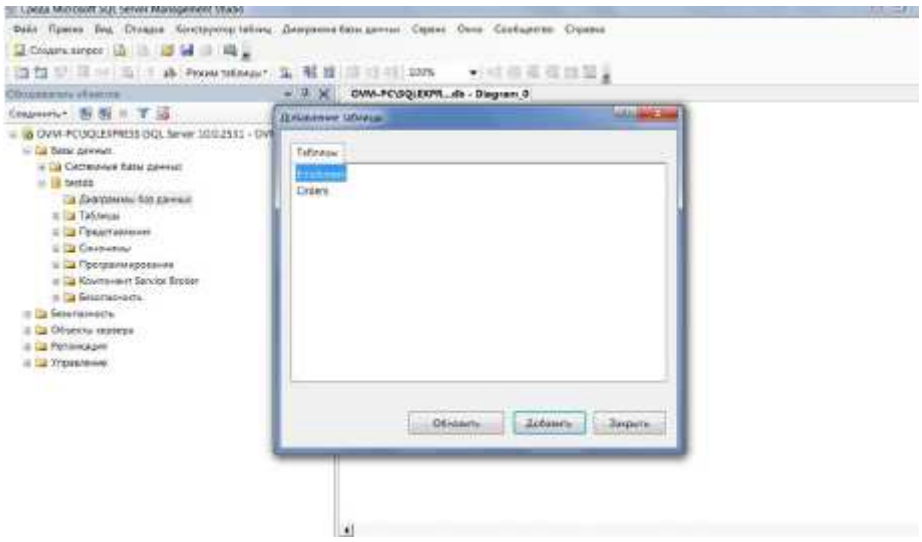


Рис.7

Оборудование и материалы:

Для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 и выше, программными продуктами: MS Office, Adobe Reader, MS SQL Management Studio.

Указания по технике безопасности:

К выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Задание:

1) Создать базу данных и ее таблицы в соответствии с вариантом. Выполнить задание с помощью стандартных команд языка T-SQL;
2) Создать диаграмму БД средствами среды.

Контрольные вопросы:

- Каково назначение СУБД?
- Дайте определение приложения, укажите, в каких случаях оно разрабатывается.
- Укажите назначение словаря данных.
- Перечислите функции администратора базы данных.

Содержание отчета:

Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал - полуторный, поля: левое - 3 см., правое - 1,5 см., верхнее и нижнее - 2 см. Абзацный отступ - 1,25. Текст должен быть выравнен по ширине.

Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчеты приводятся выводы о проделанной работе.

Лабораторная работа 2.
ЗАПОЛНЕНИЕ ТАБЛИЦ БАЗЫ ДАННЫХ

Цель работы: научиться заполнять таблицы базы данных в среде SQL Server Management Studio.

Теоретическая часть

В SQL Server 2008 заполнение таблиц производится при помощи следующей команды:

INSERT INTO имя_таблицы(список_столбцов) VALUES (список_значений);

где имя_таблицы - таблица, куда вставляются данные, список_столбцов - список полей, в которые вставляются данные, если он не указывается, то подразумевается заполнение всех полей, в списке полей поля указываются через запятую, список_значений - значение полей для вставки через запятую.

Пример: Добавление записи в таблицу «Orders»:

```
INSERT INTO dbo.Orders (orderid, empid, custid, qty) VALUES (10002, 3, 'B', 10000);
```

Из таблицы можно удалить все столбцы, либо отдельные записи. Это осуществляется командой

```
DELETE FROM <Имя таблицы> [WHERE <Условие>]
```

где <Условие> - условие, которым удовлетворяют удаляемые записи, если условие не указано, то удаляются все столбцы таблицы.

Пример: удалить записи из таблицы «Orders», у которых поле orderts < '20130930'.

```
DELETE FROM dbo.Orders WHERE orderts < '20130930';
```

Значение полей таблицы можно обновить (изменить), используя следующую команду:

```
UPDATE <Имя таблицы> SET <Имя поля1> = <Выражение1>,
[<Имя поля2> = <Выражение2>,]
…
[WHERE <Условие>]
```

,где <Имя поля1>, <Имя поля2> - имена изменяемых полей;
<Выражение1>,<Выражение 2> - значения, которые должны принять поля;
<Условие> - условие, которым должны соответствовать записи, поля которых изменяем.
В качестве выражения можно использовать математические формулы.

Если необходимо из таблицы удалить все записи, но сохранить ее структуру, нужно воспользоваться командой TRUNCATE TABLE <Имя таблицы>, при этом все данные будут удалены, но сама таблица останется.

Пример

Перейдём к заполнению таблиц БД, созданной в примере из лабораторной работы №1.

В таблицу «Departments» записываются имена отделов музея. Идентификатор каждого отдела будет заполняться автоматически, потому что при создании таблицы было использовано свойство IDENTITY.

```
INSERT INTO dbo.Departments (Name) VALUES ('Отдел искусства фотографии'),
('Отдел гравюры и рисунка'), ('Отдел искусства Античного мира'),
('Отдел искусства Древнего Востока'), ('Отдел искусства стран Европы XTX-XX вв.'),
('Отдел реставрации'), ('Отдел русского искусства XTX-XX вв.');
```

Аналогично заполняются и другие таблицы базы данных. Например, для таблицы «Employees» запрос на добавление данных может быть записан следующим образом:

```
INSERT INTO dbo.Employees
VALUES ('Иванов', 'Иван', 'Иванович', 'Реставратор', 30000, '20130109', NULL, 6),
('Петров', 'Петр', 'Петрович', 'Смотритель', 15000, '20130109', NULL, 1),
('Степанов', 'Степан', 'Степанович', 'Начальник', 50000, '20130109', NULL, 2),
('Федорова', 'Анна', 'Федоровна', 'Уборщица', 10000, '20130109', NULL, 3),
('Кузнецова', 'Наталья', 'Ивановна', 'Директор',75000, '20130109', NULL, NULL),
```

('Потапов', 'Виктор', 'Сергеевич', 'Реставратор', 30000, '20130109', NULL, 5),
('Сергеев', 'Николай', 'Петрович', 'Экскурсовод', 30000, '20130109', NULL, 7),
('Коваленко', 'Мария', 'Владимировна', 'Экскурсовод', 30000, '20130109', NULL, 1),
('Верещагин', 'Вадим', 'Петрович', 'Экскурсовод', 30000, '20130109', NULL, 2),
('Сидорова', 'Татьяна', 'Викторовна', 'Экскурсовод', 30000, '20130109', NULL, 3),
('Семенова', 'Инна', 'Николаевна', 'Экскурсовод', 30000, '20130109', NULL, 4);

Если поле является внешним ключом, то заполнять его следует значениями того поля, которое является первичным ключом для него, как и было сделано в этом случае для поля «DepId» (номер отдела из таблицы «Departments»).

Оборудование и материалы:

Для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 и выше, программными продуктами: MS office, Adobe Reader, MS SQL Management Studio.

Указания по технике безопасности:

К выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Задание:

Заполнить таблицы БД данными в соответствии с вариантом.

Контрольные вопросы:

- Охарактеризуйте основные виды программ, относящихся к СУБД.
- Назовите основные способы работы пользователя с базой данных при решении прикладных задач.
- Дайте характеристику многопользовательским СУБД.
- Перечислите классические модели представления данных.
- Укажите достоинства и недостатки иерархической модели данных.
- Как организуется физическое размещение данных в БД иерархического типа?
- Охарактеризуйте сетевую модель данных.
- Охарактеризуйте реляционную модель данных.
- Дайте определение реляционной модели и назовите составляющие ее элементы.
- Охарактеризуйте составные элементы реляционной модели данных и формы их представления.

Содержание отчета:

Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал - полуторный, поля: левое - 3 см., правое - 1,5 см., верхнее и нижнее - 2 см. Абзацный отступ - 1,25. Текст должен быть выравнен по ширине.

Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчеты приводятся выводы о проделанной работе.

Лабораторная работа 3
Создание запросов и фильтров

Цель работы: научиться создавать запросы и фильтры в среде SQL Server Management Studio.

Теоретическая часть

Запросы предназначены для связи одной или нескольких таблиц. Также они могут осуществлять отбор отдельных полей из таблицы и производить фильтрацию данных согласно условию, наложенному на одно или несколько полей. Такие запросы называют фильтрами.

Для реализации запросов используют специальный язык запросов SQL (Structured Query Language), в MS SQL Server используется процедурное расширение языка SQL компанией Microsoft, T-SQL (Transact-SQL).

В информационных системах запросы могут храниться как на стороне клиентского приложения, так и на стороне сервера. Если запрос хранится на стороне клиента, то он прописывается внутри объекта связи. В этом случае клиентское приложение не зависит от файла данных. Файл данных содержит только таблицы, поэтому, мы легко можем модифицировать клиентское приложение, не затрагивая файл данных, но в этом случае запрос передается серверу через сеть, что может вызвать проблемы с безопасностью.

Если запрос хранится или выполняется на сервере, то сам запрос выступает в качестве компонента БД, вся передача информации происходит внутри файл данных, т.е. внутри самого сервера, клиентскому приложению только передаются результаты выполнения запроса. В этом случае обеспечивается высокая защита данных, но в случае изменения запроса придется менять сам файл данных.

Все запросы делятся на статические и динамические. Структура статических запросов неизменна в ходе работы с программой, а динамические запросы могут меняться в зависимости от ситуации.

Обычно динамические запросы могут быть реализованы только при помощи запросов, выполняющихся на стороне клиента. Если необходимо реализовать динамические запросы, которые выполняются на стороне сервера, то в этом случае необходимо использовать хранимые процедуры. Подробно хранимые процедуры будут рассмотрены в лабораторной работе №5.

В основном запрос или хранимая процедура либо реализует связь между таблицами, либо осуществляет фильтрацию данных, некоторые SQL запросы также могут производить вычисления.

В случае связей между таблицами одна таблица всегда выступает первичной, а другая - вторичной, связь происходит при помощи полей связи. При связи сопоставляются записи с одинаковыми значениями полей связи. Первичная таблица всегда заполняется первой, а ее поле связи заполняется автоматически (тип данных - счётчик). Вторичная таблица всегда заполняется после заполнения первичной таблицы, значения ее поля связи подставляется из значений поля связи первичной таблицы. Поля связи должны иметь одинаковый тип данных. Существует четыре вида связи между таблицами:

1) одна к одной - одному полю в первичной таблице соответствует одно поле во вторичной таблице;
2) одна ко многим - одному полю в первичной таблице соответствует несколько полей во вторичной таблице;
3) многие к одной - нескольким полям в первичной таблице соответствует одно поле во вторичной таблице;
4) многие ко многим - одному полю в первичной таблице соответствует несколько полей во вторичной таблице и наоборот.

Запросы с первым видом связи называются простыми, а с остальными видами связи - сложными.

Чтобы создать запрос необходимо сделать активной БД, для которой создается запрос, затем в рабочей области редактора запросов создать запрос с помощью команды SELECT, имеющей следующий синтаксис:

```
SELECT [ ALL | DISTINCT ]
[ TOP (expression) [ PERCENT ] ]
{
```

```
*
| { table_name }.*
| {
[{ table_name }. ]
{ column_name }
| expression [ [ AS ] column_alias ]
}
| column_alias=expression
}
[ INTO new_table ]
[ FROM { table_name } ]
[ WHERE search_condition ]
[ GROUP BY column_expression ] [ HAVING search_condition ]
[ORDER BY { order_by_expression [ ASC | DESC ] }]
```

,где параметры ALL|DISTINCT показывают, какие записи окажутся в результирующей таблице: ALL - дублирующийся записи могут оказаться в результате, DISTINCT - только уникальные записи окажутся в результате;
TOP expression PERCENT определяет, сколько записей попадет в результат запроса (в количестве записей или в процентах);
table_name - имя таблицы;
column_name - имя столбца;
INTO new_table - результат запроса может быть помещен в таблицу с именем new_table;
search_condition определяет условие для осуществления поиска данных в таблице (после WHERE) или для группировки данных в результате запроса (после HAVING);
ORDER BY позволяет сгруппировать данные, полученные в результате запроса, по столбцам (order_by_expression) в порядке возрастания или убывания (ASC | DESC).

Замечания:
1) Если выбираются поля из разных таблиц с одинаковыми именами нужно указывать и имя таблицы table_name.column_name.
2) Полям можно присваивать псевдонимы, следующим образом: column_name AS column_alias.
3) Если необходимо выбрать все поля из таблицы, то их можно заменить значком «*».
4) Раздел INTO. Если присутствует этот раздел, то на основе результатов запроса создается новая таблица. Параметр INTO это имя новой таблицы.
5) Раздел FROM. Здесь указываются таблицы и запросы, через запятую, которые участвуют в новом запросе. В разделе FROM также можно задавать сложные связи: связь поля одной таблицы с несколькими полями другой таблицы.
6) Раздел WHERE. Данный раздел используют для создания простых запросов, в этом случае в качестве условия указываются связываемые поля, либо этот раздел используют для создания фильтров, и здесь указываются условия отбора. В условиях отбора мы можем использовать стандартные логические операторы NOT, OR, AND.
7) Раздел GROUP BY определяет поле для группировки записей в запросе.
8) Раздел ORDER BY определяет поле для сортировки записей в запросе. Если указан параметр ASC, то будет производиться сортировка по возрастанию, если DESC - по убыванию. По умолчанию используется сортировка по возрастанию.

Кроме связывания таблиц и отбора данных оператор SELECT может использоваться для вычислений. В этом случае он имеет синтаксис:

SELECT expression

,где expression - какое-то математическое выражение или функция.

В SQL Server существуют следующие встроенные агрегатные функции:

AVG	Возвращает среднее значение
CHECKSUM_A	Возвращает контрольную сумму значений
COUNT	Возвращает количество значений (результат имеет тип int)
COUNT_BIG	Возвращает количество значений (результат имеет тип bigint)
GROUPING	Определяет, является ли столбец, указанный в списке
	GROUP BY агрегированным. GROUPING возвращает 1 для
MAX	Возвращвет максимальное значение
MIN	Возвращает минимальное значение
SUM	Возвращает сумму всех значений
STDEV	Возвращает среднеквадратичное отклонение всех значений
STDEVP	Возвращает среднеквадратичное отклонение для множества
VAR	Возвращает дисперсию всех значений
VARP	Возвращает дисперсию для множества всех значений

Примеры использования агрегатных функций:

SELECT AVG(возраст) FROM Студенты - выводит средний возраст студента из таблицы «Студенты». SELECT СОЦЫТ(ФИО) FROM Студенты - выводит количество различных ФИО из таблицы «Студенты».

Пример

Рассмотрим пример создания статических запросов для база данных информационной системы по обслуживанию хранилищ, музеев, выставок и аукционов (см. примеры из предыдущих лабораторных работ). На следующем рисунке показана диаграмма этой БД (рис. 1):

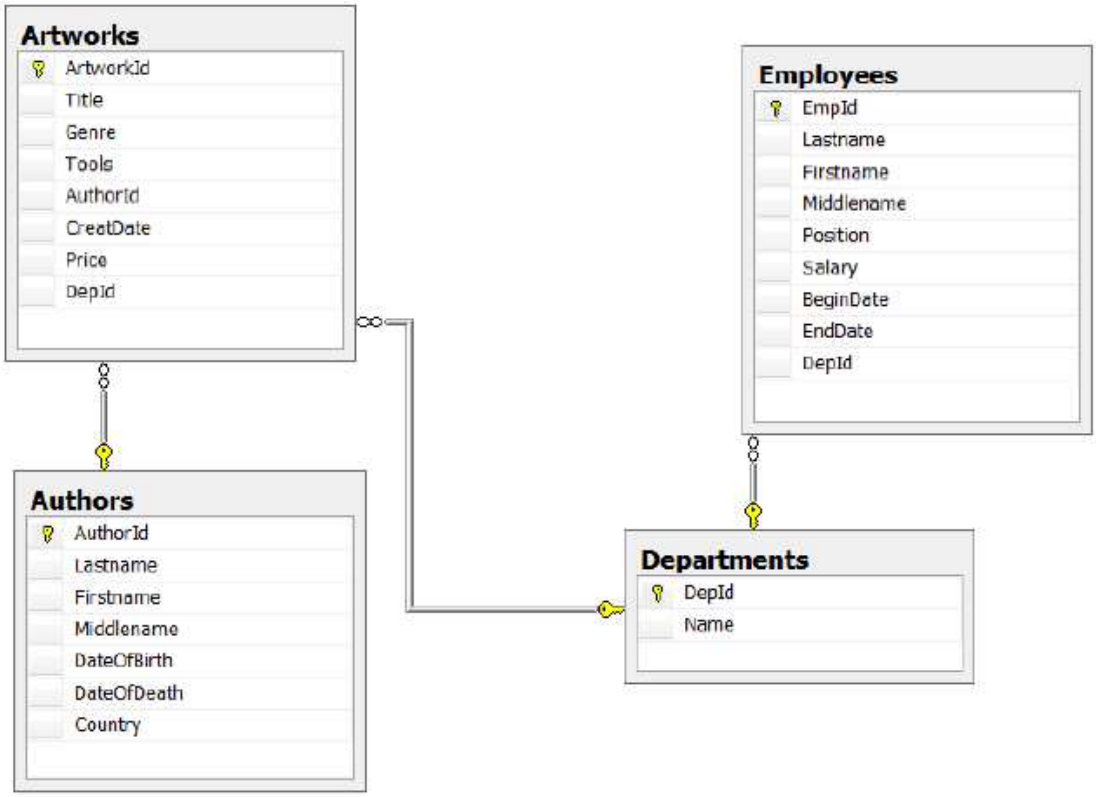


Рис. 1

Создадим запрос «Отдел кадров» (связывает таблицы «Employees» и «Departments» по полю «DepId») и фильтр для отображения сотрудников отдельных отделов (На основе запроса «Отдел кадров»).

Следующий программный код осуществляет внутреннее соединение таблиц «Employees» и «Departments» по внешнему ключу «DepId»:

```
SELECT E.Lastname, E.Firstname, E.Middlename, E.Position, D.Name, E.BeginDate
FROM dbo.Employees AS E
JOIN dbo.Departments AS D ON E.DepId = D.DepId;
```

Результатом выполнения этого запроса будет динамическая таблица, содержащая все строки из таблицы «Employees», в которых значение атрибута «DepId» равно значению этого же атрибута из таблицы «Departments» (рис. 2):

Результаты | Сообщения

	Lastname	Firstname	Middlename	Position	Name	BeginDate
1	Иванов	Иван	Иванович	Реставратор	Отдел реставрации	2021-01-09
2	Петров	Петр	Петрович	Смотритель	Отдел искусства фотографии	2021-01-09
3	Степанов	Степан	Степанович	Начальник	Отдел гравюры и рисунка	2021-01-09
4	Федорова	Анна	Федоровна	Уборщица	Отдел искусства Античного мира	2021-01-09
5	Потапов	Виктор	Сергеевич	Реставратор	Отдел искусства стран Европы XIX-XX вв.	2021-01-09
6	Сергеев	Николай	Петрович	Экскурсовод	Отдел русского искусства XIX-XX вв.	2021-01-09
7	Коваленко	Мария	Владимировна	Экскурсовод	Отдел искусства фотографии	2021-01-09
8	Верещагин	Вадим	Петрович	Экскурсовод	Отдел гравюры и рисунка	2021-01-09
9	Сидорова	Татьяна	Викторовна	Экскурсовод	Отдел искусства Античного мира	2021-01-09
10	Семенова	Инна	Николаевна	Экскурсовод	Отдел искусства Древнего Востока	2021-01-09

Рис. 2

Для создания фильтра, выводящего данные сотрудников отдельных отделов, нам нужно модифицировать предыдущий запрос следующим образом:

```
SELECT E.Lastname, E.Firstname, E.Middlename, E.Position, D.Name, E.BeginDate
FROM dbo.Employees AS E
JOIN dbo.Departments AS D ON E.DepId = D.DepId
AND D.Name = 'Отдел искусства фотографии';
```

Поле «Name» таблицы «Departments» содержит названия отдела, сравнивая его с конкретным именем отдела, мы можем организовать вывод информации о сотрудниках этого отдела (рис. 3):

Результаты | Сообщения

	Lastname	Firstname	Middlename	Position	Name	BeginDate
1	Петров	Петр	Петрович	Смотритель	Отдел искусства фотографии	2021-01-09
2	Коваленко	Мария	Владимировна	Экскурсовод	Отдел искусства фотографии	2021-01-09

Рис. 3

Оборудование и материалы:

Для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 и выше, программными продуктами: MS office, Adobe Reader, MS SQL Management Studio.

Указания по технике безопасности:

К выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Задание

Создать фильтры и запросы в соответствии с заданием для своего варианта.

Контрольные вопросы:

- Что представляет собой первичный ключ отношения, для чего он задается?
- Назовите условия, при соблюдении которых таблицу можно считать отношением.
- Что такое индекс, для чего используется индексирование?
- Изобразите схемы одноуровневой и двухуровневой индексаций и дайте им характеристику.
- Что такое вторичный индекс, в чем его отличие от первичного индекса?
- Приведите схему возможной организации связи вторичного индекса с элементами базы данных.

Лабораторная работа 4.
Хранимые процедуры

Цель работы: научиться создавать хранимые процедуры в среде SQL Server Management Studio.

Теоретическая часть

Хранимая процедура - SQL-запрос, который имеет параметры, то есть он выполняется как обычная процедура. В зависимости от значения параметров хранимой процедуры мы получаем тот или иной результат запроса. В SQL сервере хранимые процедуры реализуют динамические запросы, выполняемые на стороне сервера.

Рассмотрим создание хранимых процедур при помощи команд языка SQL.

Чтобы отобразить хранимые процедуры рабочей БД панели «Обозреватель объектов» нужно выбрать пункт «Программирование», а в нем - «Хранимые процедуры». Чтобы создать новую процедуру

при помощи команд языка SQL нужно щелкнуть левой кнопкой мыши по кнопке 

на панели инструментов. В рабочей области окна сервера появится вкладка SQLQueryl.sql, где нужно набрать код новой процедуры, который имеет следующий синтаксис:

```
CREATE { PROC | PROCEDURE } procedure_name [ ; number ] [ { @parameter data_type }
[ = default ] [ OUT | OUTPUT ] [READONLY]
] [ ,.n ]
[ WITH [ RECOMPILE ] [ ,...n ] ] [ FOR REPLICATION ]
AS { <sql_statement> [;][ ...n ] } [;]
<sql_statement> ::=
{ [ BEGIN ] statements [ END ] }
```

,где:

- procedure_name - имя новой хранимой процедуры. Имена процедур должны соответствовать правилам, предъявляемым к идентификаторам, и должны быть уникальными.
- number - необязательное целое число, используемое для группирования процедур с одним именем. Все сгруппированные процедуры можно удалить, выполнив одну инструкцию DROP PROCEDURE.
- @parameter - параметр процедуры. В инструкции CREATE PROCEDURE можно объявить один или более параметров. При выполнении процедуры значение каждого из объявленных параметров должно быть указано пользователем, если для параметра не определено значение по умолчанию или значение не задано равным другому параметру. Хранимая процедура может иметь не более 2 100 параметров. Определяет имя параметра, используя знак @ как первый символ. Имя параметра должно соответствовать правилам для идентификаторов. Параметры являются локальными в пределах процедуры; в разных процедурах могут быть использованы одинаковые имена параметров.
- data_type - тип данных параметра. Все типы данных, которые могут использоваться в качестве параметра хранимой процедуры Transact-SQL. Можно использовать определяемый пользователем табличный тип, чтобы объявить возвращающий табличное значение параметр в качестве параметра хранимой процедуры Transact-SQL.
- default - значение параметра по умолчанию. Если значение default определено, процедуру можно выполнить без указания значения соответствующего параметра. Значение по умолчанию должно быть константой или может равняться NULL.
- OUTPUT показывает, что параметр процедуры является выходным. Значение этого параметра можно получить при помощи инструкции EXECUTE. Используйте параметры OUTPUT для возврата значений коду, вызвавшему процедуру.
- READONLY указывает, что параметр не может быть обновлен или изменен в тексте процедуры. Если тип параметра является определяемым пользователем табличным типом, должно быть указано ключевое слово READONLY.
- RECOMPILE показывает, что компонент Database Engine не кэширует план выполнения процедуры и что процедура компилируется во время выполнения.

– EXECUTE AS определяет контекст безопасности, в котором должна быть выполнена хранимая процедура.
– <sql_statement> - одна или несколько инструкций языка SQL, которые будут включены в состав процедуры.

Если параметры сравниваются с какими-то полями или выражениями, то они должны иметь точно такой же тип данных, как эти поля или выражения.

После создания процедура помещается в раздел «Хранимые процедуры» текущей БД на панели «Обозреватель объектов».

Чтобы посмотреть информацию о хранимой процедуре необходимо выполнить команду

EXEC SP_HELPTEXT <Имя процедуры>

Хранимые процедуры могут быть запущены следующей командой

EXEC <Имя процедуры> [<Параметр1>, <Параметр2>, ...]

,где
<Имя процедуры> - имя выполняемой процедуры;
<Параметр1>, <Параметр2> - значения параметров.

Пример: создание хранимой процедуры, которая выводит имена студентов со средним баллом, большим заданной величины:

```
CREATE PROCEDURE AvgPoints
@X Real AS SELECT *
FROM students WHERE (Оценка1 + Оценка2 + Оценка3)/3>@X
```

Команда вызова этой процедуры выглядит следующим образом:

```
EXEC AvgPoints 4
```

Команда выводит всех студентов, у которых средний балл больше 4.

Пример: для работы с хранимыми процедурами в обозревателе объектов необходимо выделить папку «Программирование/Хранимые процедуры» базы данных (рис. 1).

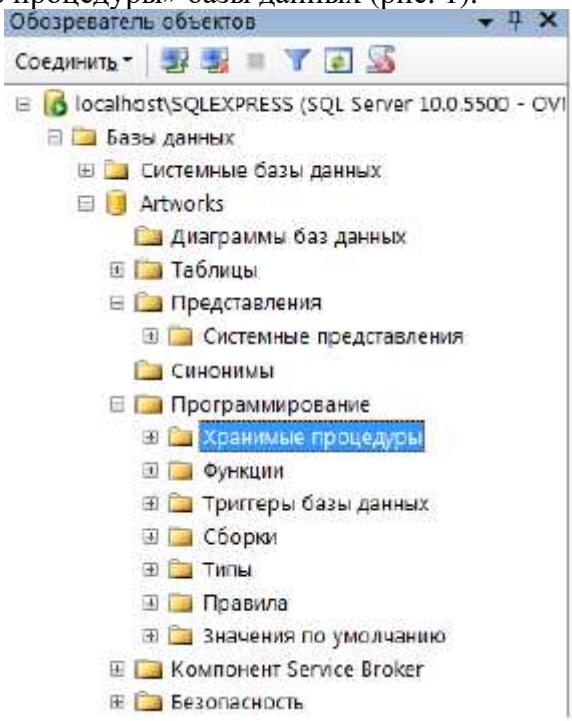


Рис. 1

Создадим процедуру, вычисляющую среднее трёх чисел. Для создания новой хранимой процедуры нужно щелкнуть правой кнопкой мыши по папке «Хранимые процедуры» и в появившемся меню выбрать пункт «Создать хранимую процедуру». Появится окно кода новой хранимой процедуры (рис. 2)

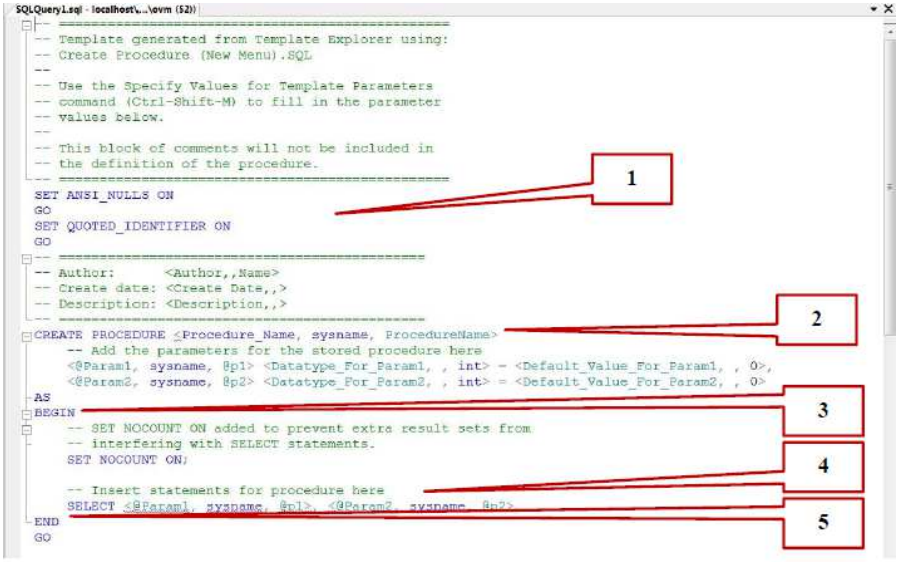


Рис. 2

Хранимая процедура имеет следующую структуру:

1. Область настройки параметров синтаксиса процедуры. Позволяет настраивать некоторые синтаксические правила, используемые при наборе кода процедуры. В нашем случае это:
SET ANSI_NULLS ON включает использование значений NULL в кодировке ANSI,
SET QUOTED_IDENTIFIER ON включает возможность использования двойных кавычек для определения идентификаторов;
2. Область определения имени процедуры и параметров, передаваемых в процедуру. Определение параметров имеет следующий синтаксис:
@<Имя параметра> <Тип данных> = <Значение по умолчанию>
Параметры разделяются между собой запятыми;
3. Начало тела процедуры, обозначается служебным словом BEGIN;
4. Тело процедуры, содержит команды языка SQL;
5. Конец тела процедуры, обозначается служебным словом END.

В коде зелёным цветом выделяются комментарии. Они не обрабатываются сервером и выполняют функцию пояснений к коду. Строки комментариев начинаются с подстроки «--». Далее в коде, мы не будем отображать комментарии, они будут свёрнуты. Слева от раздела с комментариями будет стоять знак «+», щёлкнув по которому можно развернуть комментарий.

Наберём код процедуры, вычисляющей среднее трёх чисел, как это показано на следующем рисунке (рис. 3).

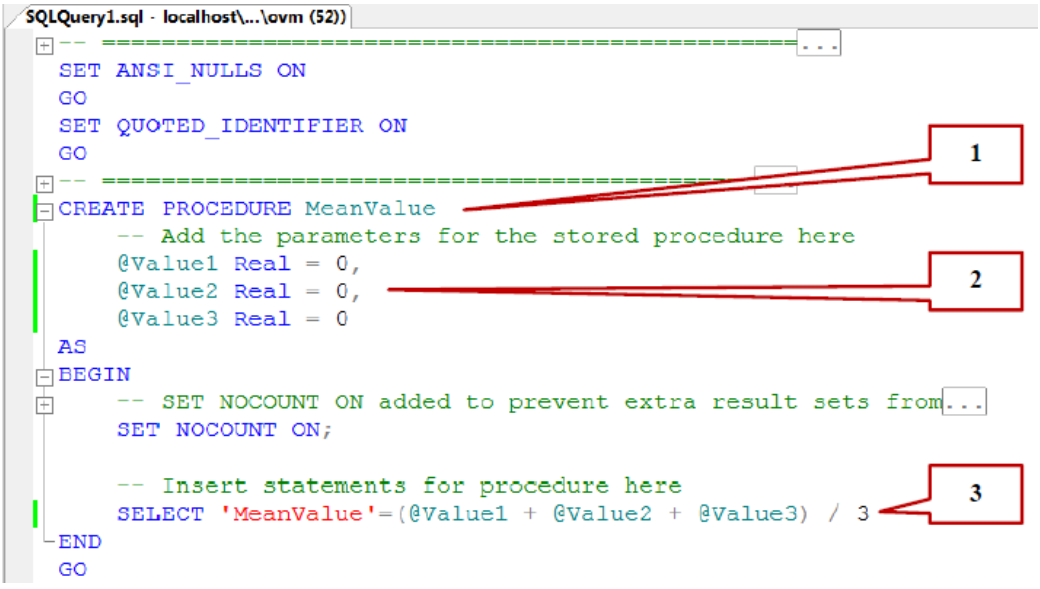


Рис. 3

Рассмотрим код данной процедуры более подробно:

1. CREATE PROCEDURE MeanValue определяет имя создаваемой процедуры как «MeanValue»;
2. @Value1 Real = 0, @Value2 Real = 0, @Value3 Real = 0 - определение трех параметра процедуры Valuel, Value2 и Value3. Тип параметров - Real, значения по умолчанию равны 0;
3. SELECT ‘Mean Value’= (@Value1+@Value2+@Value3)/3 - вычисление среднего и вывод результата с подписью «Среднее значение».

Для создания процедуры выполним ее код, нажав кнопку Выполнить на панели инструментов.

В нижней части окна с кодом появится сообщение «Выполнение команд успешно завершено». Закройте окно с кодом, щёлкнув мышью по кнопке закрытия, расположенной в верхнем правом углу окна с кодом процедуры.

Проверим работоспособность созданной хранимой процедуры. Для запуска хранимой процедуры необходимо создать новый пустой запрос, нажав на кнопку Создать запрос на панели инструментов. В появившемся окне с пустым запросом наберите команду:

EXEC MeanValue 1, 7, 9

и нажмите кнопку Выполнить на панели инструментов.

В нижней части окна с кодом появиться результат выполнения новой хранимой процедуры: Среднее значение равно 5,66667 (рис.4).

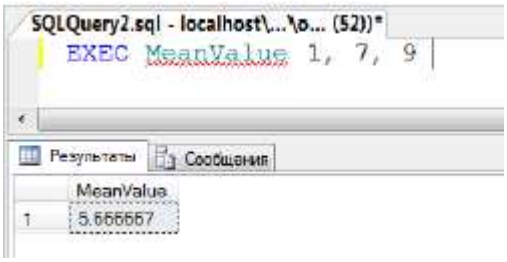


Рис. 4

Теперь создадим хранимую процедуру для отбора художников из таблицы «Authors» по их фамилиям. Для этого создадим новую хранимую процедуру по описанию, приведенному выше, и наберем следующий код новой процедуры (рис. 5).

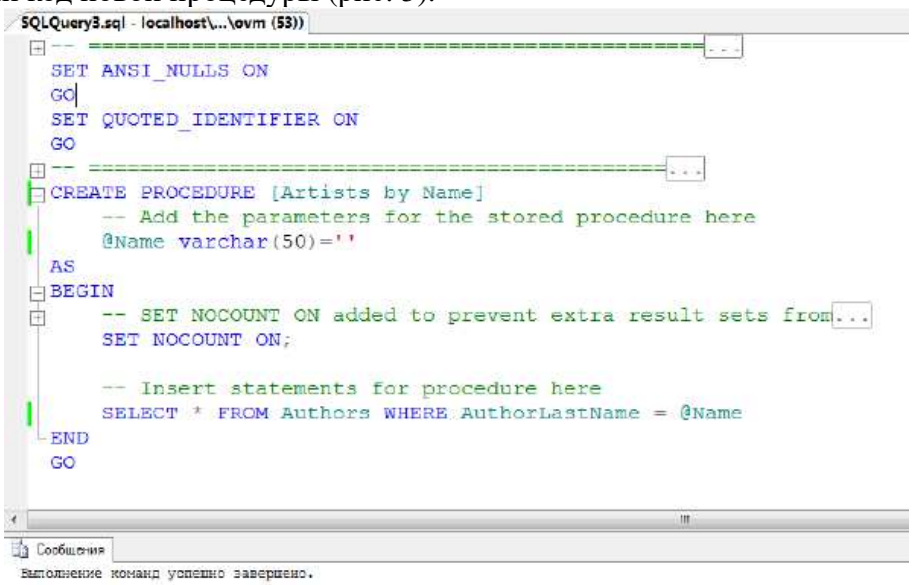


Рис. 5

Проверим работоспособность созданной хранимой процедуры. Создадим новый пустой запрос. В появившемся окне с пустым запросом наберем команду EXEC [Artists by Name] ‘Рерих’ и выполним запрос (рис. 6).

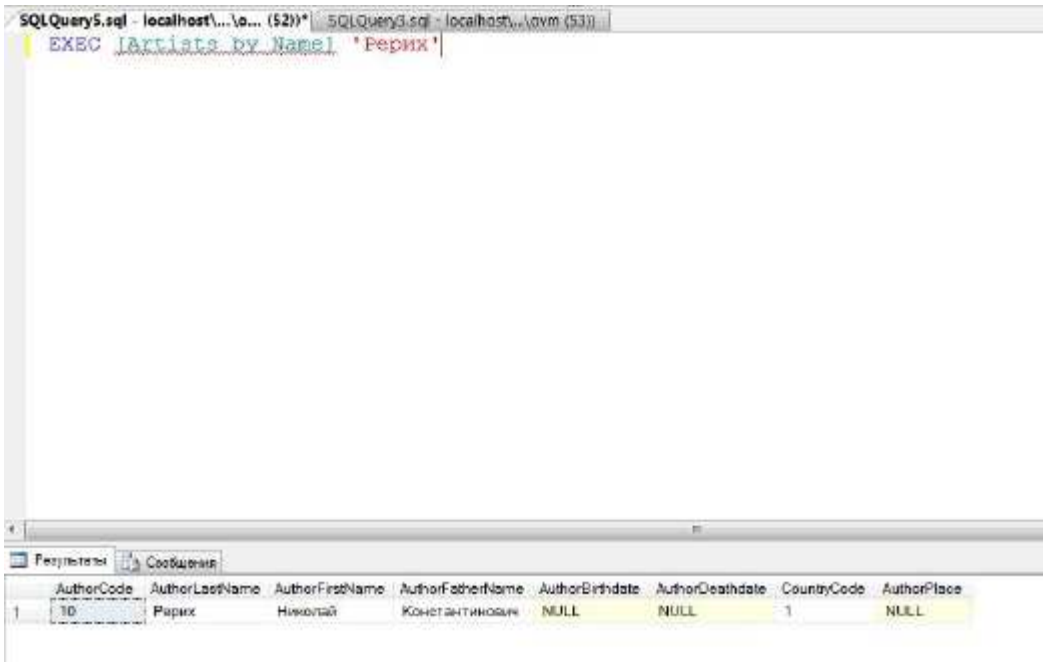


Рис. 6

В заключение решим более сложную задачу: отображение имен художников, родившихся в XV веке. Создадим новую хранимую процедуру и наберем следующий код новой процедуры (рис. 7).

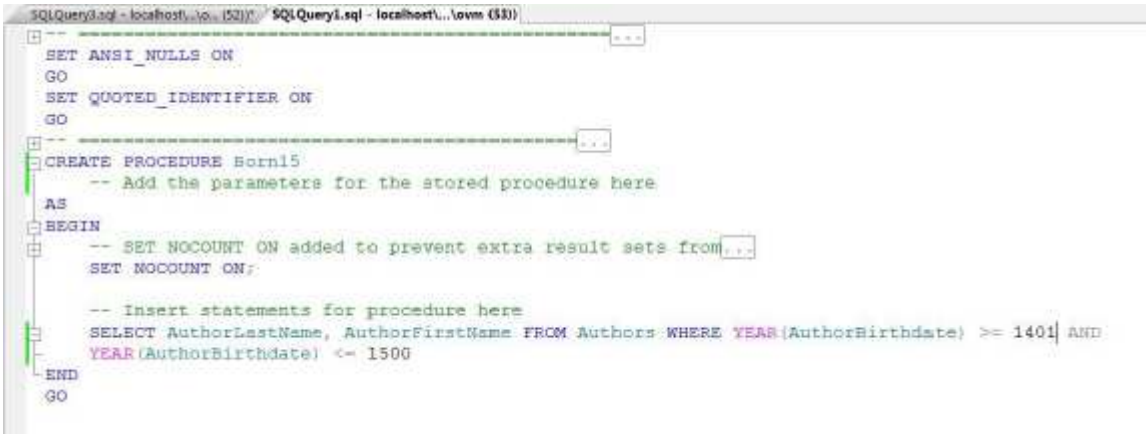


Рис. 7

Функция YEAR имеет следующий синтаксис YEAR(date^ возвращает целое число, представляющее год указанной даты date.

Полученный с помощью функции YEAR год рождения проверяется на вход в диапазон годов, определяющих XV век.

Результат выполнения этой хранимой процедуры приведен на следующем рисунке (рис. 8).

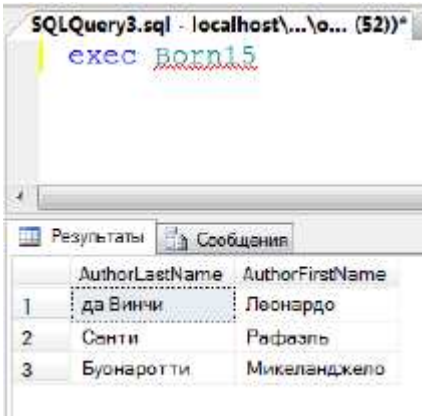


Рис. 8

Оборудование и материалы:

Для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 и выше, программными продуктами: MS office, Adobe Reader, MS SQL Management Studio.

Указания по технике безопасности:

К выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Задание:

Создать хранимые процедуры в соответствии с заданием для своего варианта.

Контрольные вопросы:

- Опишите действие механизма контроля целостности при манипулировании данными в таблицах.
- Назовите подходы к проектированию структур данных.
- В чем состоит избыточное и неизбыточное дублирование данных?
- Назовите и охарактеризуйте основные виды аномалий.
- Как формируется исходное отношение при проектировании БД?
- Приведите примеры явной и неявной избыточности.
- Назовите основные виды зависимостей между атрибутами отношений.
- Приведите примеры функциональной и частичной функциональной зависимостей.
- Приведите примеры отношений с зависимыми атрибутами.
- Охарактеризуйте нормальные формы.
- Дайте определение первой нормальной формы.
- Дайте определение второй нормальной формы.
- Дайте определение третьей нормальной формы.
- Дайте определение усиленной третьей нормальной формы.
- Поясните на примере используемых в разделе таблиц требования 4НФ.
- Поясните на примере используемых в разделе таблиц требования 5НФ.

Содержание отчета:

Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал полуторный, поля: левое 3 см., правое - 1,5 см., верхнее и нижнее - 2 см. Абзацный отступ - 1,25. Текст должен быть выравнен по ширине.

Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчеты приводятся выводы о проделанной работе.

Лабораторная работа 5.
Пользовательские функции

Цель работы: научиться создавать пользовательские функции в среде SQL Server Management Studio.

Теоретическая часть:

В среде SQL Server Management Studio все пользовательские функции находятся в папке «Функции», расположенной в папке «Программирование» в обозревателе объектов (рис. 1).

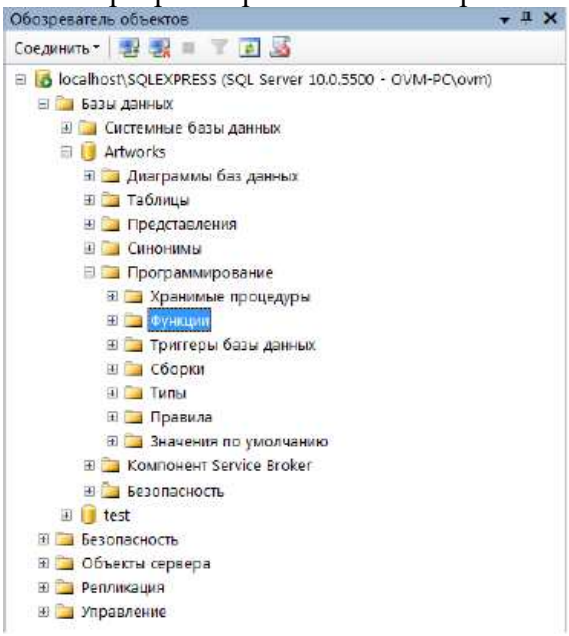


Рис. 1

Начнём с создания скалярных пользовательских функций. Для создания новой скалярной пользовательской функции в обозревателе объектов щёлкните правой кнопкой мыши по папке «Функции» и в появившемся меню выберите пункт «Создать» → «Скалярная функция». Появится окно новой скалярной пользовательской функции (рис. 2).

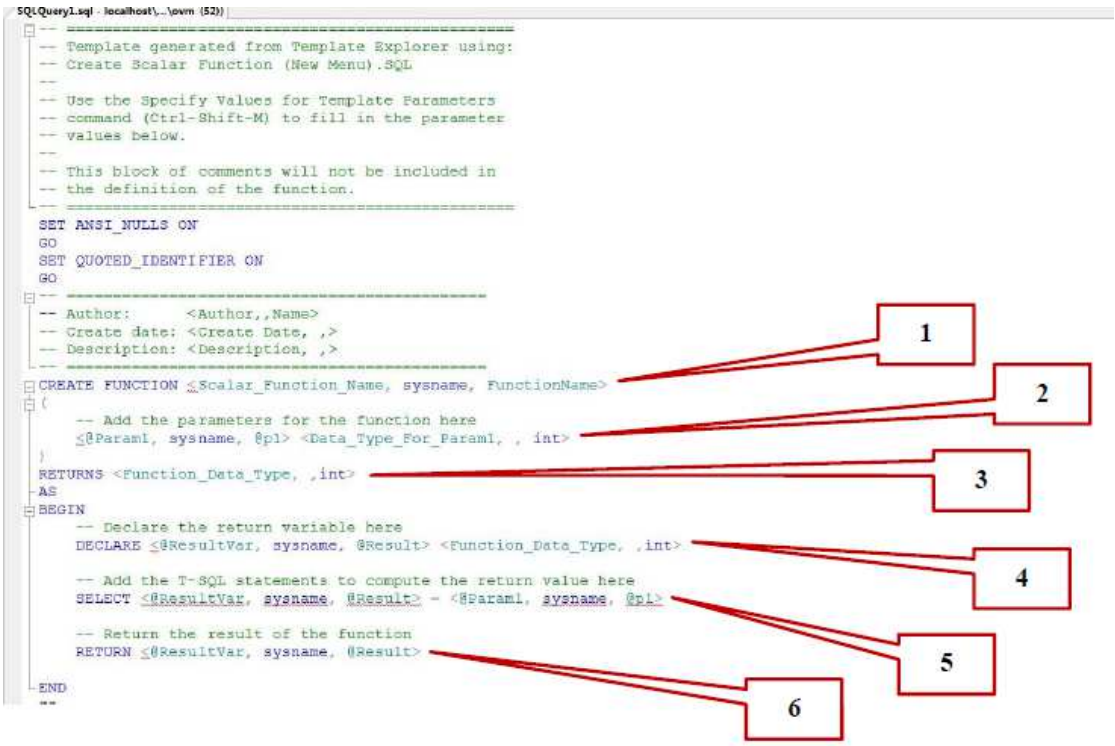


Рис. 2

Синтаксис скалярной пользовательской функции похож на синтаксис хранимой процедуры. Но есть и существенные отличия.

1. Область определения имени функции (Scalar_Function_Name);
2. Параметры, передаваемые в функцию (@param1). Определение параметров аналогично определению параметров в хранимой процедуре;
3. Тип данных значения, возвращаемого функцией;
4. Область объявления переменных, используемых внутри функции. Объявление переменных имеет следующий синтаксис:

DECLARE @<Имя переменной> <Тип данных>

5. Тело самой пользовательской функции, содержит команды языка T-SQL;
6. Команда RETURN, возвращающая результат выполнения функции, имеет следующий синтаксис:

RETURN @<Имя переменной с результатом>

Переменная должна быть того же типа данных, который был указан в пункте 3.

Создадим скалярную пользовательскую функцию, вычисляющую среднее трёх величин. В окне новой пользовательской функции наберите код представленный на рис. 3.

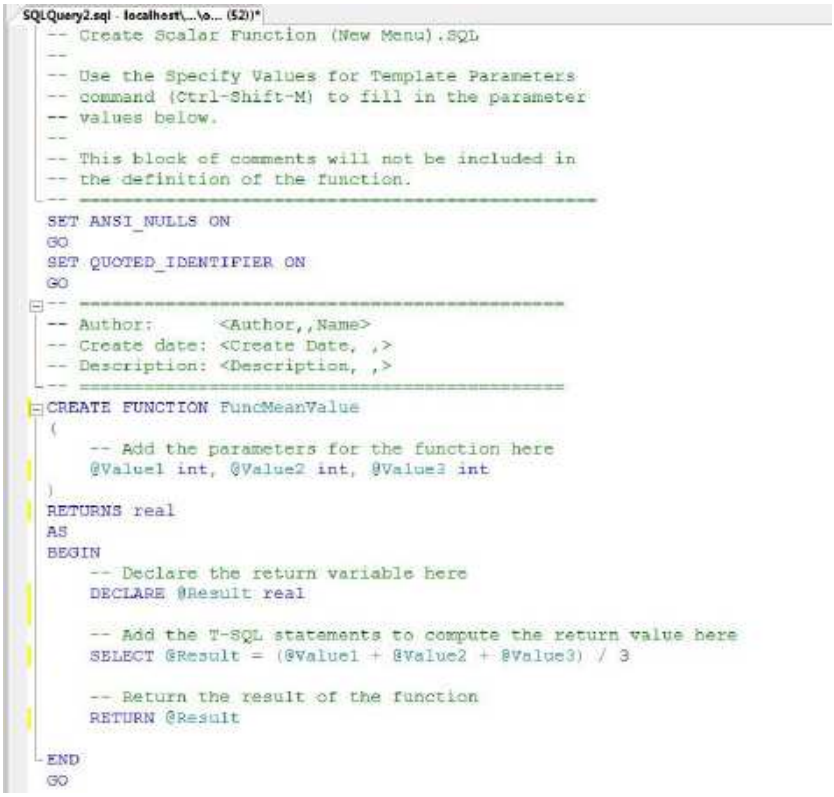
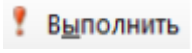


Рис. 3

Рассмотрим более подробно код данной скалярной пользовательской функции:

1. CREATE FUNCTION FuncMeanValue определяет имя создаваемой функции как FuncMeanValue;
2. @Value1, @Value2, @Value3 определяют три параметра процедуры Value1, Value2 и Value3. Данным параметрам можно присвоить целые числа (тип данных int);
3. RETURNS real показывает, что функция возвращает вещественные числа (тип данных real);
4. DECLARE @Result real - объявляется переменная @Result для хранения результата работы функции вещественного типа (тип данных real);
5. SELECT @Result = (@Value1 + @Value2 + @Value3) / 3 вычисляет среднее и помещает результат в переменную @Result;
6. RETURN @Result возвращает значение переменной @Result.

Для создания функции, выполним вышеописанный код, нажав кнопку 

на панели инструментов. В нижней части окна с кодом появится сообщение «Выполнение команд успешно завершено.».

Для проверки работы созданной скалярной пользовательской функции необходимо создать новый пустой запрос, нажав на кнопку Создать запрос на панели инструментов. В появившемся окне с пустым запросом наберите команду SELECT dbo.FuncMeanValue (3, 5, 4) и нажмите кнопку Выполнить на панели инстументов (рис. 4).

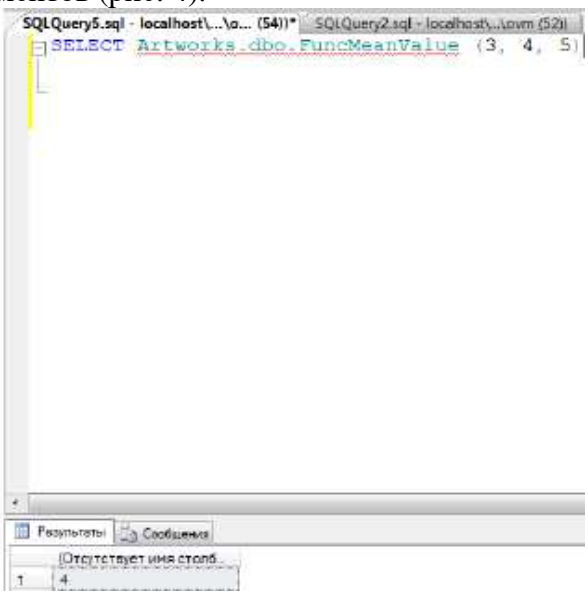


Рис. 4

В нижней части окна с кодом появится результат выполнения новой скалярной пользовательской функции.

Теперь перейдём к созданию табличных пользовательских функций. Для создания табличной пользовательской функции в обозревателе объектов щёлкните правой кнопкой мыши по папке «Функции» и в появившемся меню выберите пункт «Создать» → «Встроенная функция, возвращающая табличное значение». Появится окно новой табличной пользовательской функции (рис. 5).

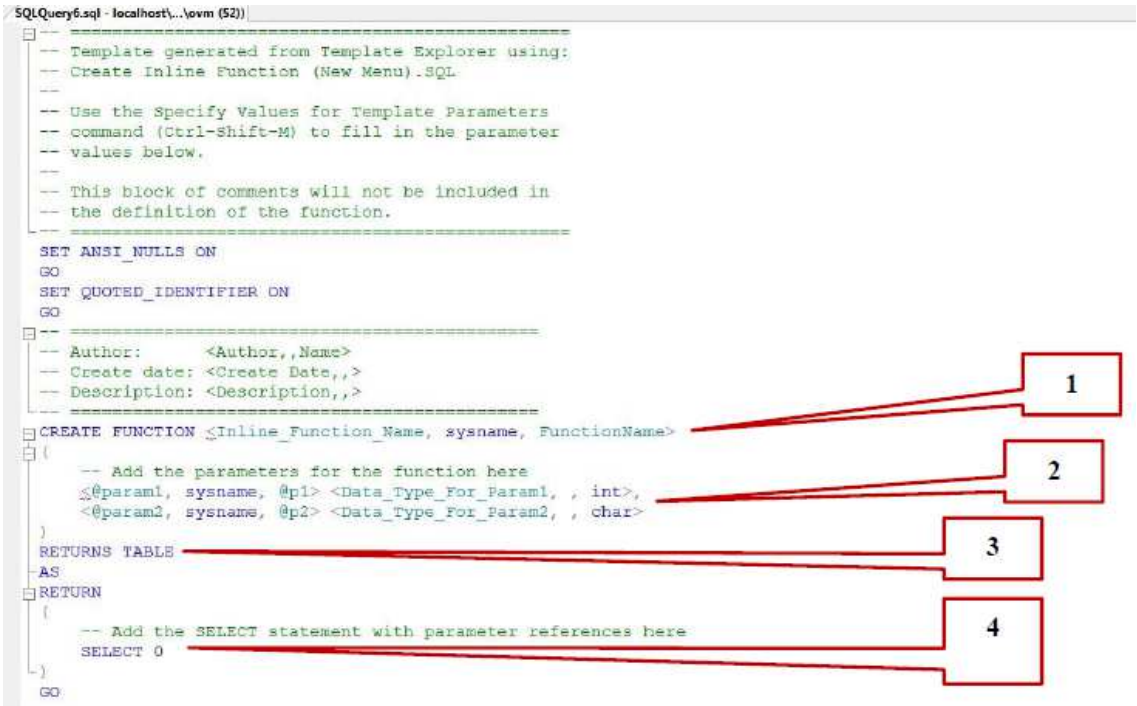


Рис. 5

Табличная пользовательская функция состоит из следующих разделов:

1. Область определения имени функции (Inline_Function_Name);
2. Параметры, передаваемые в функцию (@param1, @param2);
3. RETURNS TABLE показывает, что функция возвращает таблицу;
4. Тело самой пользовательской функции состоит из команды SELECT языка T- SQL.

Рассмотрим создание табличной пользовательской функции, выбирающей информацию о художниках по стране их проживания. В окне новой пользовательской функции (рис. 5) наберем следующий код (рис. 6):

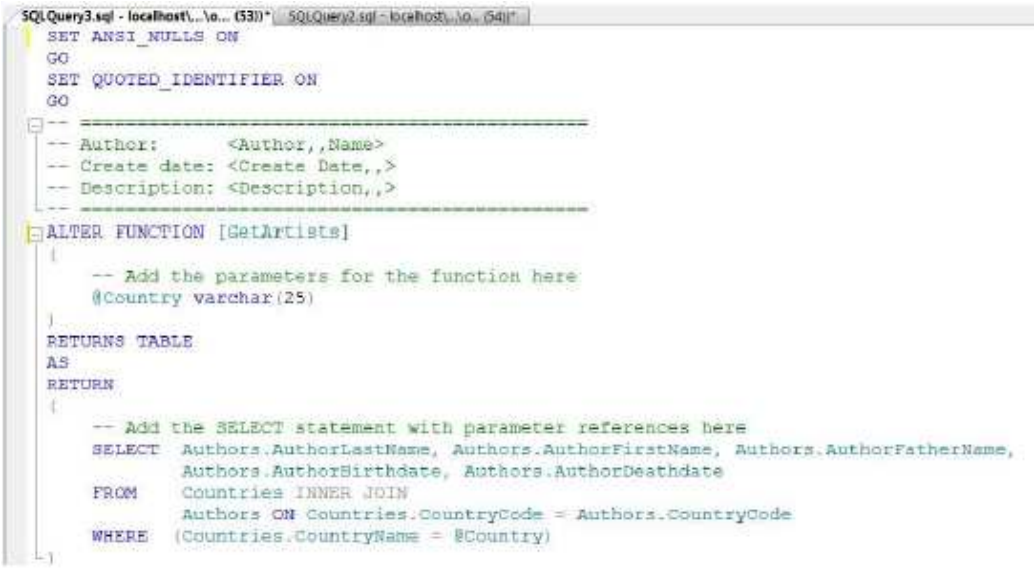


Рис. 6

Из кода функции видно, что она принимает один параметр типа varchar (название страны) и реализуется запросом:

SELECT Authors.AuthorLastName, Authors.AuthorFirstName,
Authors.AuthorFatherName, Authors.AuthorBirthdate, Authors.AuthorDeathdate
FROM Countries
INNER JOIN Authors
ON Countries.CountryCode = Authors.CountryCode
WHERE (Countries.CountryName = @Country)

Из таблицы «Authors» выбираются фамилии, имена, отчества, даты рождения и смерти художников, при этом таблица «Authors» соединяется (INNER JOIN) с таблицей «Countries» по столбцу «CountryCode». Условием соединения является равенство содержимого поля «CountryCode» передаваемому в функцию параметру.

Результат выполнения этой функции приведен на рис.7.

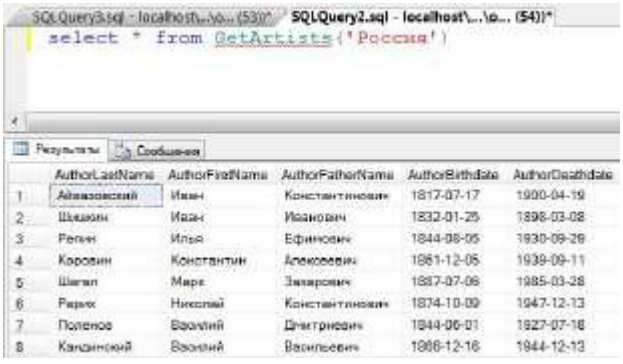


Рис. 7

Работа с табличной функцией осуществляется так же, как и с обыкновенной таблицей.

Оборудование и материалы:

Для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 и выше, программными продуктами: MS office, Adobe Reader, MS SQL Management Studio.

Указания по технике безопасности:

К выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Задание:

Создать функции в соответствии с заданием для своего варианта.

Контрольные вопросы:

- Сформулируйте основное правило создания таблиц сущностей.
- Назовите рекомендации по организации связи сущностей.
- Дайте определение физической и логической целостности БД.
- Приведите примеры ограничений значений и структурных ограничений.
- Поясните понятия внешнего и первичного ключей таблиц.
- Перечислите основные понятия метода сущность-связь.
- Охарактеризуйте понятие ключа сущности.
- Что представляют собой диаграммы ER-экземпляров и диаграммы ER-типа.
- Что определяет степень связи между сущностями?
- Каким может быть класс принадлежности?

Содержание отчета:

Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал полуторный, поля: левое - 3 см., правое - 1,5 см., верхнее и нижнее - 2 см. Абзацный отступ - 1,25. Текст должен быть выравнен по ширине.

Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчеты приводятся выводы о проделанной работе.

Лабораторная работа 6.
Триггеры

Цель работы: научиться создавать триггеры в среде SQL Server Management Studio.

Теоретическая часть:

При работе БД должна обеспечиваться целостность данных. При удалении записей из первичных таблиц автоматически должны удаляться связанные с ними записи из вторичных таблиц. В случае несоблюдения этого принципа со временем в БД накопится большое количество записей во вторичных таблицах, связанных с несуществующими записями в первичных таблицах, что приведёт к сбоям в работе БД и её засорению. Для обеспечения целостности данных в SQL Server используют триггеры.

Триггер — это сочетание хранимой в базе данных процедуры и события, которое заставляет ее выполняться. Такими событиями могут быть: ввод новой строки таблицы, изменение значений одного или нескольких ее столбцов и (или) удаление строки таблицы. При любом из этих событий автоматически запускаются один или несколько заранее созданных триггеров, которые производят проверку запрограммированных в них условий, и если они не выполняются, отменяют ввод, изменение или удаление, посылая об этом заранее подготовленное сообщение пользователю.

Триггеры похожи на процедуры и функции тем, что также являются именованными блоками и имеют раздел объявлений, выполняемый раздел и раздел обработки исключительных ситуаций. Подобно процедурам и функциям, триггеры хранятся как автономные объекты в базе данных.

Триггеры позволяют:

– реализовывать сложные ограничения целостности данных, которые невозможно реализовать через ограничения, устанавливаемые при создании таблицы;
– контролировать информацию, хранимую в таблице, посредством регистрации вносимых изменений и пользователей, производящих эти изменения;
– автоматически оповещать другие программы о том, что необходимо делать в случае изменения информации, содержащейся в таблице;
– публиковать информацию о различных событиях.

Триггеры также делятся на три основных типа:

– Триггеры DML активизируются предложениями ввода, обновления и удаления информации (INSERT, UPDATE, DELETE) до или после выполнения предложения, на уровне строки или таблицы.
– Триггеры замещения (instead of) можно создавать только для представлений (либо объектных, либо реляционных). В отличие от триггеров DML, которые выполняются в дополнение к предложениям DML, триггеры замещения выполняются вместо предложений DML, вызывающих их срабатывание. Триггеры замещения должны быть строковыми триггерами.
– Системные триггеры активизируется не на предложение DML, выполняемое над таблицей, а на системное событие, например, на запуск или останов базы данных. Системные триггеры срабатывают и на предложения DDL, такие как создание таблицы.

Создадим триггеры для таблицы «Artworks». Триггеры создаются отдельно для каждой таблицы и располагаются в обозревателе объектов в папке «Триггеры» (рис. 1).

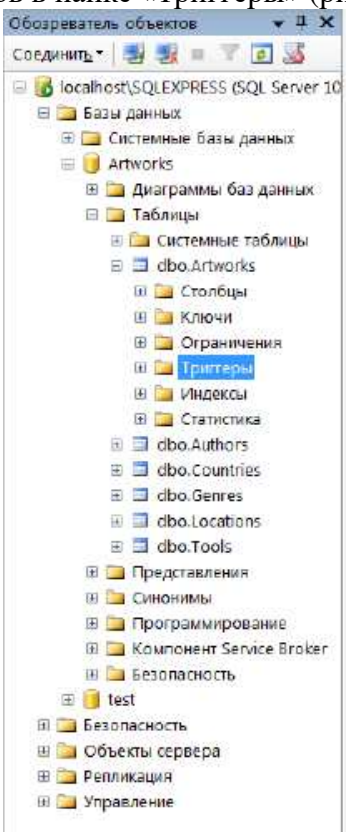


Рис. 1

Для начала создадим триггер, выводящий сообщение «Запись добавлена» при добавлении записи в таблицу «Artworks». Создадим новый триггер, щёлкнув правой кнопкой мыши по папке «Триггеры» в таблице «Artworks» и выбрав в появившемся меню пункт «Создать триггер». Появится окно с новым триггером (рис. 2).

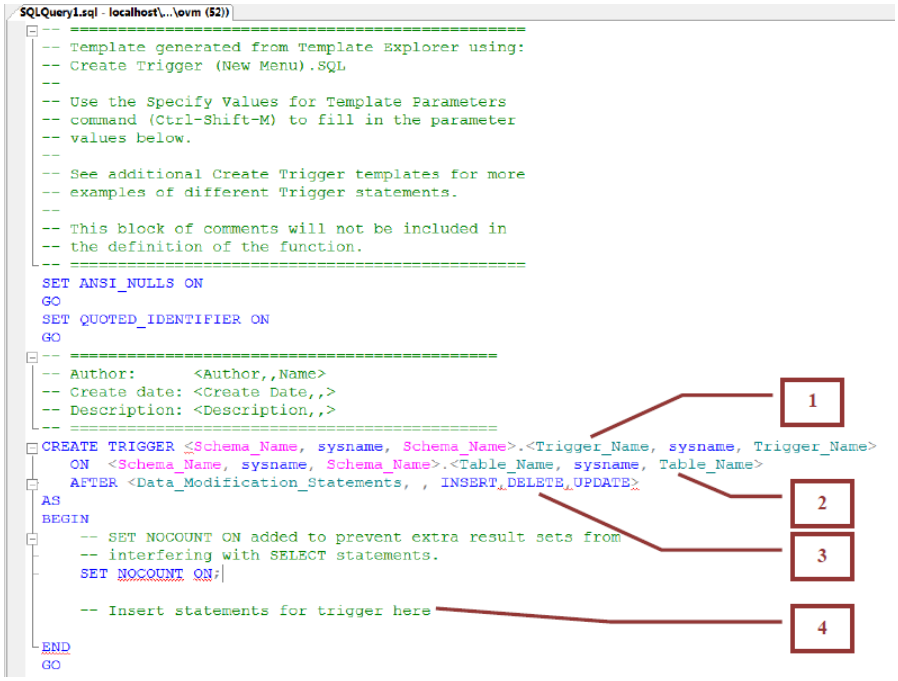


Рис 2.

Структура триггера:

1) область определения имени триггера (Trigger_Name);
2) область, показывающая для какой таблицы создаётся триггер (Table_Name);
3) область, показывающая, когда выполнять триггер (INSERT - при создании записи в таблице, DELETE - при удалении и UPDATE - при изменении);

4) тело триггера, содержащее команды языка T-SQL.

В окне нового триггера наберем следующий код (рис. 3).

```
SQLQuery2.sql - localhost\...\o... (53))*  SQLQuery1.sql - localhost\...\ovm (52))
-- ============================================
-- Template generated from Template Explorer using:
-- Create Trigger (New Menu).SQL
--
-- Use the Specify Values for Template Parameters
-- command (Ctrl-Shift-M) to fill in the parameter
-- values below.
--
-- See additional Create Trigger templates for more
-- examples of different Trigger statements.
--
-- This block of comments will not be included in
-- the definition of the function.
-- ============================================
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =============================================
CREATE TRIGGER Insert_trigger
   ON  dbo.Artworks
   AFTER INSERT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    PRINT('Запись добавлена!');

END
GO
```

Рис. 3

Созданный триггер «Insert_trigger» выполняется после добавления записи (AFTER INSERT) в таблицу «Artworks» (ON dbo.Artworks). После добавления записи триггер выведет на экран сообщение «Запись добавлена!» (PRINT ‘Запись добавлена!’). Выполним набранный код, нажав кнопку «Выполнить» на панели инструментов. В нижней части окна с кодом появится сообщение «Выполнение команд успешно завершено».

Проверим, как работает новый триггер. Создадим новый пустой запрос и в нём выполним добавление данных в таблицу «Artworks» (рис.4):

```
SQLQuery2.sql - localhost\...\o... (53))*  SQLQuery1.sql - localhost\...\ovm (52))
insert into Artworks (ArtworkTitle,GenreCode,ToolsCode,AuthorCode,CreationDate,CountryCode,PlacementCode,Price)
values('Утро в сосновом лесу', 13, 3, 2, '1889-01-01', 1, 4, 1000000)

Сообщения
Запись добавлена!

(строк обработано: 1)
```

Рис. 4.

Выполним набранную команду, нажав кнопку «Выполнить» на панели инструментов. В таблицу будет добавлена новая запись, и триггер выведет сообщение «Запись добавлена!» (рис. 4).

Аналогичным образом создаются триггеры, выводящие сообщения при изменении данных в таблице (рис. 5) и при их удалении (рис. 6).

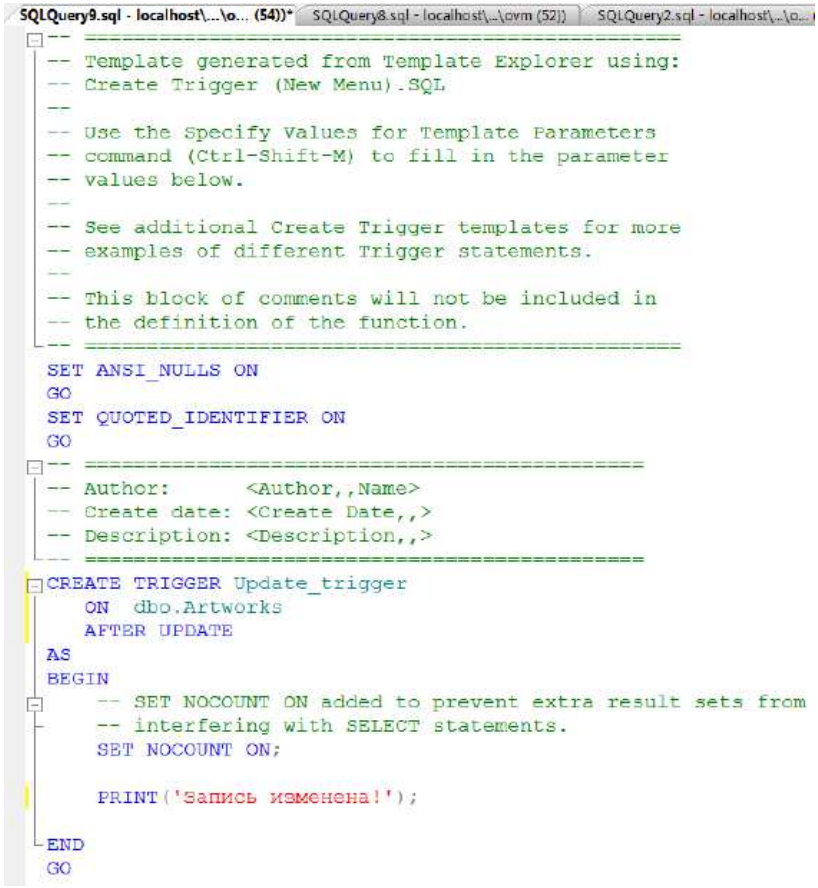


Рис. 5.

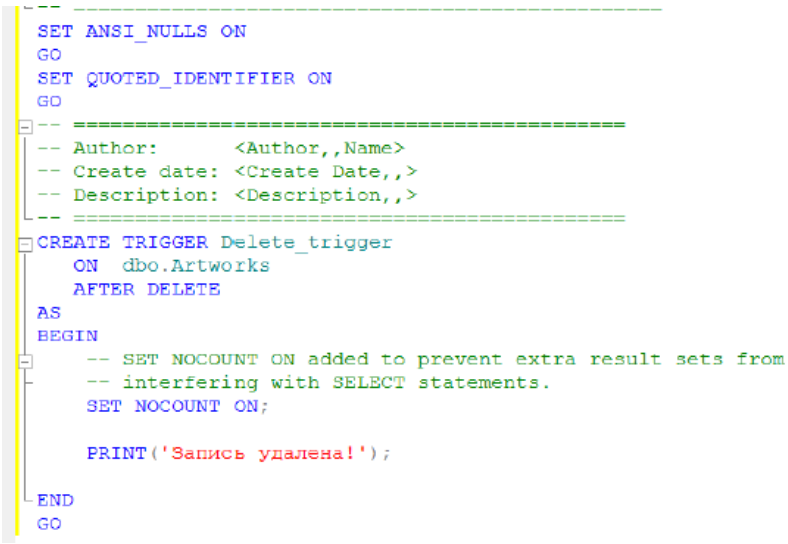


Рис. 6

Рассмотрим пример применения триггеров для обеспечения целостности данных. Создадим триггер «Delete_author» (рис. 7), который при удалении записи из таблицы «Authors» сначала удаляет все связанные с ней записи из таблицы «Artworks», а затем удаляет саму запись из таблицы «Authors».

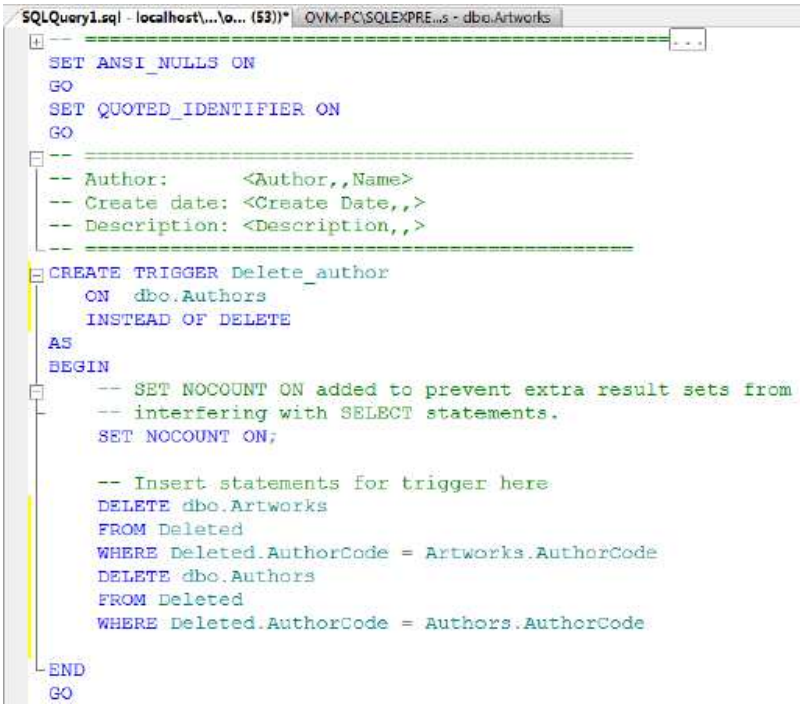


Рис. 7

При срабатывании триггера вместо удаления записи создаётся временная константа Deleted, содержащая имя таблицы, из которой должно было быть произведено удаление.

После срабатывания триггера из таблицы «Artworks» удаляется запись, у которой значение поля «AuthorCode» равно значению такого же поля у удаляемой записи из таблицы «Authors». Затем удаляется запись из таблицы «Authors», которую удаляли до срабатывания триггера.

Выполним набранный код, нажав кнопку «Выполнить» на панели инструментов. В нижней части окна с кодом появится сообщение «Выполнение команд успешно завершено.».

Проверим, как работает триггер «Author_delete» (рис. 8). При срабатывании триггера сначала из таблицы «Artworks» удалятся все связанные с удаляемой записью записи, а затем удаляется сама удаляемая запись из таблицы «Artworks», при этом сохраняется целостность данных.

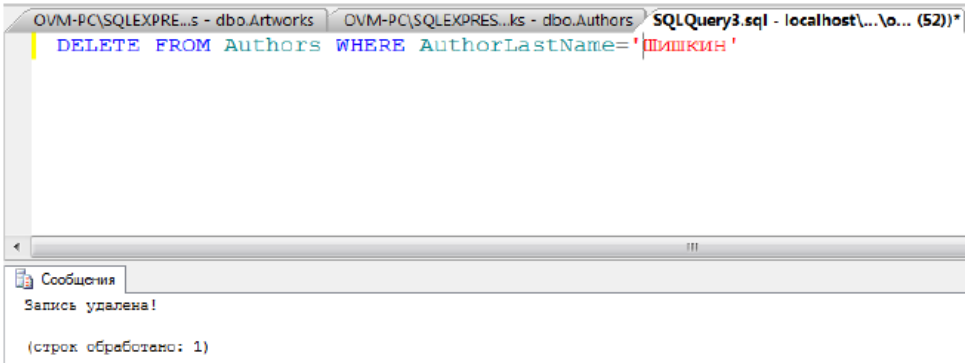


Рис. 8

Оборудование и материалы:

Для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 и выше, программными продуктами: MS office, Adobe Reader, MS SQL Management Studio.

Указания по технике безопасности:

К выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Задание:

Создать триггеры в соответствии с заданием для своего варианта.

Контрольные вопросы:

- Приведите пример диаграммы ER-экземпляров со степенью связи между сущностями 1:1 и обязательным классом принадлежности двух сущностей.
- Как на диаграммах ER-типа обозначаются степень связи, обязательное и необязательное участие в связи экземпляров сущности?
- Приведите пример диаграммы ER-экземпляров для связи типа 1:M варианта H-O.
- Назовите этапы проектирования базы данных.
- Как осуществляется формирование отношений для связи 1:1?
- Сформулируйте правило формирования отношений, если степень связи 1:1 и класс принадлежности обеих сущностей является необязательным.
- Сформулируйте правило формирования отношения для случая степени связи между сущностями 1:M (M:1) и обязательного класса принадлежности M-связной сущности.
- Укажите правила формирования отношений для связи M:M.

Содержание отчета:

Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал - полуторный, поля: левое - 3 см., правое - 1,5 см., верхнее и нижнее - 2 см. Абзацный отступ - 1,25. Текст должен быть выравнен по ширине.

Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчеты приводятся выводы о проделанной работе.