ГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

БАЗЫ ДАННЫХ

Методические рекомендации к лабораторным работам для студентов направлений подготовки 09.03.01 «Информатика и вычислительная техника» и 09.03.04 «Программная инженерия» дневной формы обучения

Могилев 2018

УДК 004.65
ББК 32.973.26-0.18.2
Б 17

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления» «25» мая 2018 г., протокол № 13

Составители: канд. техн. наук, доц. К. В. Захарченков;
канд. техн. наук, доц. Т. В. Мрочек

Рецензент канд. техн. наук, доц. И. В. Лесковец

Методические рекомендации содержат описание двадцати трех лабораторных работ по дисциплине «Базы данных» с использованием AllFusion Process Modeler, AllFusion ERwin Data Modeler, Microsoft Office Access, Microsoft SQL Server, Microsoft Visual Studio .Net. Предназначены для студентов направлений подготовки 09.03.01 «Информатика и вычислительная техника» и 09.03.04 «Программная инженерия» дневной формы обучения.

Учебно-методическое издание

БАЗЫ ДАННЫХ

Ответственный за выпуск	А. И. Якимов
Технический редактор	А. Т. Червинская
Компьютерная верстка	Е. С. Лустенкова

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс. Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 31 экз. Заказ №

Издатель и полиграфическое исполнение:
Государственное учреждение высшего профессионального образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 24.01.2014.
Пр. Мира, 43, 212000, Могилев.

Содержание

Введение

Целью учебной дисциплины «Базы данных» является формирование профессиональных компетенций для работы с современными технологиями проектирования баз данных, создания и модификации баз данных, а также обучение приемам поиска, сортировки, индексирования и защиты данных в базах данных.

В результате освоения учебной дисциплины студент

познает:

– основные принципы организации баз данных;

– язык SQL;

– способы работы с реляционными базами данных;

– принципы моделирования и проектирования баз данных;

– технологии создания и работы с базами данных в современных системах управления базами данных (СУБД) и языках программирования;

– технологии для работы с базами данных в сети;

научится:

– практически создавать и администрировать базы данных;

– создавать программный интерфейс с базами данных;

– устанавливать и конфигурировать серверные и клиентские приложения баз данных;

овладеет:

– методами программирования систем на основе баз данных;

– навыками проектирования баз данных.

В методических рекомендациях кратко изложены основные теоретические сведения и задания, позволяющие:

– приобрести знания и навыки для разработки и выбора эффективных методов решения задач, связанных с представлением, хранением, отображением, передачей и обработкой информации;

– сформировать навыки разработки, администрирования и эксплуатации информационных компонентов систем обработки информации на основе современных СУБД и языков программирования высокого уровня.

Методические рекомендации содержат описание 23 лабораторных работ, задания, контрольные вопросы, список литературы [1–15], которую необходимо использовать при подготовке к защите лабораторных работ.

1 Разработка технического задания на проектирование информационной системы

Цель: разработать проект технического задания (ТЗ) на проектирование автоматизированной информационной системы (ИС) для выбранной предметной области.

1.1 Теоретические положения

Проект ТЗ разрабатывается на основе [1] и имеет следующую структуру.

1 Общие сведения.

1.1 Объект автоматизации.

1.2 Документы, на основании которых создается ИС.

2 Назначение и цели создания ИС.

2.1 Назначение системы.

2.2 Цели создания системы (здесь перечисляются цели и указываются критерии оценки достижения целей системы).

3 Характеристика объектов автоматизации.

3.1 Краткие сведения.

Здесь необходимо:

– выполнить описание работы объекта автоматизации (например, отделов предприятия, для автоматизации работы которых создается ИС);

– перечислить основные функции объекта автоматизации и указать информацию, подлежащую хранению;

– перечислить категории пользователей будущей базы данных, определить права доступа разных категорий пользователей к различной информации в ИС.

3.2 Сведения об условиях эксплуатации объекта автоматизации и характеристиках окружающей среды (здесь указывается, на каких рабочих местах и для выполнения каких именно функций будет использоваться ИС).

4 Требования к системе.

4.1 Требования к системе в целом (указываются требования надежности, безопасности, защиты информации, квалификации персонала).

4.2 Требования к функциям (задачам), выполняемым ИС.

При перечислении функций необходимо указать форму получения информации по каждой функции (в виде запроса (результатом выполнения которого является виртуальная таблица) либо отчета (или иных видов бумажной документации));

4.3 Требования к видам обеспечения.

Здесь указываются требования к программному обеспечению.

Задание

Необходимо выбрать предметную область (согласовать с преподавателем тему) и разработать техническое задание на проектирование автоматизирован-

ной информационной системы для выбранной предметной области.

Содержание отчета: тема и цель работы; техническое задание.

Контрольные вопросы

1 Указать состав и содержание ТЗ на автоматизированную ИС.
2 Каковы правила оформления ТЗ?
3 Каков порядок разработки, согласования и утверждения ТЗ на ИС?

2 CASE-средство концептуального проектирования функциональных моделей информационных систем AllFusion Process Modeler

Цель: изучить основные функциональные возможности CASE-средства AllFusion Process Modeler r7; разработать функциональную модель ИС для выбранной предметной области с использованием AllFusion Process Modeler r7.

2.1 Теоретические положения

На этапе инфологического (концептуального) проектирования по итогам анализа предметной области с помощью CASE-средства Computer Associates AllFusion Process Modeler (BPwin) разрабатывается функциональная модель IDEF0 для ИС. Функциональная модель отображает структуру и функции системы, а также потоки информации и материальных объектов, связывающие эти функции, с точностью, достаточной для однозначного моделирования деятельности объекта автоматизации.

Функциональная модель IDEF0 должна содержать три типа диаграмм:

– контекстную диаграмму, описывающую основную функцию разрабатываемой ИС и показывающую взаимодействие ИС с внешней средой;

– диаграммы декомпозиции процессов, описывающие каждую подфункцию ИС и так до достижения требуемого уровня детализации анализируемой системы;

– диаграмму дерева узлов, отображающую иерархическую зависимость функций.

Разработка диаграмм подробно описана в конспекте лекций и в [11].

Задание

Разработать функциональную модель информационной системы для выбранной предметной области с использованием AllFusion Process Modeler r7.

Модель должна содержать не менее 4-х уровней декомпозиции, на последнем уровне должно быть не менее 15 блоков (минимум). На контекстной диаграмме должны быть показаны точка зрения и цель модели.

Содержание отчета: тема и цель работы; контекстная диаграмма, диаграммы декомпозиции, диаграмма дерева узлов.

Контрольные вопросы

1 Что такое точка зрения и цель модели? Сколько точек зрения может быть в модели? Что такое функциональный SA-блок и какие стрелки он содержит?

2 Назвать основные принципы и правила SADT. Перечислить основные этапы моделирования с использованием методологий SADT и IDEF0.

3 Охарактеризовать основные типы отношений между блоками в диаграммах IDEF0 (доминирование; управление; выход – вход; обратная связь по управлению; обратная связь по входу; выход – механизм).

4 Туннельные стрелки на диаграммах IDEF0.

5 Чем различаются модели системы AS-IS, TO-BE и SHOULD-BE?

3 Основы использования средства концептуального проектирования информационной модели системы AllFusion ERwin Data Modeler

Цель: изучить основные функциональные возможности AllFusion ERwin Data Modeler r7; разработать в ERwin логическую и физическую модели ИС.

3.1 Теоретические положения

На этапе логического (даталогического) проектирования на основе функциональной модели строится информационная модель. На верхних уровнях моделирования такая модель называется логической моделью и содержит наиболее общие сведения об объектах предметной области, тем самым обеспечивая независимость модели от конкретной СУБД. Целью данного этапа является идентификация сущностей, составляющих предметную область, и связей между ними. Список потенциальных сущностей формируется на основе списка названий стрелок в функциональной модели [2, 7].

Логическая модель отображается ER-диаграммой логического уровня, разрабатываемой с помощью CASE-средства Computer Associates AllFusion ERwin Data Modeler с использованием методологии построения моделей «сущность-связь» IDEF1X.

ER-диаграмма логического уровня должна отображать сущности, связи между сущностями, имена связей (verb phrase) и атрибуты. На данном этапе проектирования производится проверка корректности структуры таблиц путем применения к ним процедуры нормализации (все таблицы базы данных должны быть приведены, по крайней мере, к третьей нормальной форме).

На этапе физического проектирования создается схема базы данных (БД) для конкретной СУБД, отображаемая с помощью ER-диаграммы физического

уровня. Данная диаграмма должна отображать сущности, связи между сущностями и атрибуты, типы данных атрибутов, NULL-значения атрибутов, мощность связей (при необходимости) ограничения ссылочной целостности. Здесь же должны быть разрешены неспецифические отношения («многие-ко-многим»).

Разработка диаграмм описана в конспекте лекций и в [2, 6–8, 10, 13, 15].

Задание

Необходимо разработать ER-диаграммы логического и физического уровней.

Содержание отчета: тема и цель работы; ER-диаграмма логического уровня; ER-диаграмма физического уровня; перечень всех сущностей, входящих в модель, с описанием их назначения и указанием атрибутов каждой сущности; подробное обоснование выбранных типов связей (идентифицирующих и неидентифицирующих (обязательных и необязательных)) между сущностями.

Контрольные вопросы

1 В чем сущность методологии IDEF1X?

2 Охарактеризовать основные понятия модели «сущность-связь»: сущности, атрибуты, виды ключей (первичный, составной, естественный, суррогатный, альтернативный, инверсный, внешний), идентифицирующие и неидентифицирующие (обязательные и необязательные) связи, направленность и мощность связи, связь категоризации, связь «многие-ко-многим» и ее разрешение.

3 Какие уровни представления данных существуют в ERwin?

4 Что такое прямое и обратное проектирование в ERwin? Что такое ссылочная целостность? В чем заключается процесс нормализации базы данных?

4 Взаимодействие CASE-средства AllFusion ERwin Data Modeler с системами управления базами данных (генерация схемы базы данных)

Цель: получить навыки генерации объектов базы данных из AllFusion ERwin Data Modeler в СУБД MS Access и MS SQL Server; освоить процедуру обратного проектирования.

4.1 Теоретические положения

После разработки информационной модели ее следует связать с функциональной моделью. Такая связь гарантирует завершенность анализа, обеспечивает наличие источников данных (сущностей) для всех работ. На данном этапе производится экспорт данных из ERwin в BPwin и связывание объектов модели данных (сущностей и атрибутов) со стрелками и работами.

В таблице отчета о верификации модели должны быть показаны связи всех информационных потоков, отображаемых стрелками в BPwin, со всеми сущностями в ERwin. Нужно учитывать, что одной и той же стрелке в функциональной модели могут соответствовать несколько сущностей в информационной модели и, наоборот, одной сущности могут соответствовать несколько стрелок.

Для генерации схемы БД из ERwin необходимо её открыть. Далее перейти к вкладке «Physical», выбрав нужный пункт из выпадающего списка на панели инструментов. В меню «Database» выбрать пункт «Choose Database». В открывшемся диалоговом окне в разделе «Target Desktop DBMS» выбрать «Access», в разделе «Access Version» из выпадающего списка выбрать 2000. Нажать ОК. Создать пустую базу данных Access (Пуск/Программы/ Microsoft Office/Microsoft Access 2000), сохранить и закрыть её. В меню «Tools» выбрать пункт «Forward Engineer/Schema Generation…». В открывшемся диалоговом окне «Access Schema Generation» нажать «Generate». В появившемся диалоговом окне «Access Connection» в поле «User Name» ввести «admin». Нажать кнопку «Browse» рядом с полем «Database» и найти ранее созданную базу данных Access. После этого нажать «Connect». Появится диалоговое окно «Generate Database Schema», в котором отражаются результаты. При возникновении ошибки генерация приостанавливается. Для продолжения необходимо выбрать «Continue». Чтобы при возникновении ошибки процесс генерации не останавливался, нужно снять метку «Stop If Failure». Нажать ОК. Открыв свою БД в Access, можно просмотреть результат генерации.

Задание

Необходимо сгенерировать схему БД из ERwin в MS Access.
Содержание отчета: тема и цель работы; схема данных БД в Access.

Контрольные вопросы

1 Что такое полное сравнение «Complete Compare»?
2 Как устранить различия между моделью в ERwin и выбранной СУБД?

5 Синхронизация функциональной и информационной моделей программной системы

Цель: выполнить синхронизацию функциональной и информационной моделей автоматизированной ИС.

5.1 Теоретические положения

После разработки информационной модели ее следует связать с функциональной моделью. Такая связь гарантирует завершенность анализа, обеспечива-

ет наличие источников данных (сущностей) для всех работ. На данном этапе производится экспорт данных из ERwin в BPwin и связывание объектов модели данных (сущностей и атрибутов) со стрелками и работами.

Первым шагом связывания модели данных и модели процессов является экспорт данных из ERwin в BPwin.

Для экспорта модели данных из ERwin в BPwin необходимо в ERwin открыть модель и выбрать пункт меню File/Bpwin/Export. В появившемся диалоге выбрать имя заранее созданного файла *.eax и нажать ОК.

Затем в BPwin нужно открыть модель процесса, выбрать в меню пункт File/Import/Erwin (EAX)..., выбрать имя файла и нажать ОК. Появится протокол импорта. Для внесения данных в модель процесса нажимают кнопку «Accept».

После внесения данных в модель процессов можно связать сущности и атрибуты со стрелками. Правой кнопкой мыши нажимают по стрелке и выбирают в контекстном меню «Arrow Data». Появится закладка «Arrow Data» диалога «Arrow Property». Для связывания атрибута со стрелкой достаточно поставить «галочку» на иконке выбора в иерархическом списке атрибутов. При этом сущность автоматически связывается со стрелкой. Каждая стрелка в модели процессов может быть связана с несколькими атрибутами различных сущностей. Кнопка «Copy In» позволяет копировать связанные данные из другой стрелки, а кнопка «Clear» – все связи стрелки с данными. Кнопка «Migrate» вызывает диалог «Changes to Arrow Data Associations», в котором отображаются данные, мигрирующие от дочерних к родительским стрелкам (для разветвляющихся и сливающихся стрелок). При миграции возможны изменения связывания данных: «Deletions» – если данные связаны с родительской стрелкой, но не связаны с дочерней, связи с родительской стрелкой удаляются; «Additions» – если данные связаны с дочерней стрелкой и не связаны с родительской, добавляется связь с родительской стрелкой. Для подтверждения изменений в диалоге «Changes to Arrow Data Associations» следует нажать кнопку «Commit». Миграция возможна только в моделях IDEF0 и DFD.

Результат связывания объектов модели процессов можно отобразить в отчете «Data Usage Report», для создания которого выбирается пункт меню Tools\Reports\Data Usage Report. Появится окно «Data Usage Report», в котором указывается, какие поля будут в отчете. Нажимается кнопка «Report». В появившемся диалоговом окне выбирается имя документа, в который будет генерироваться отчет, и нажимается ОК.

Задание

Необходимо сгенерировать таблицу отчета о верификации моделей, в которой должны быть показаны связи всех информационных потоков, отображаемых стрелками в BPwin, со всеми сущностями в ERwin. Следует учитывать, что одной и той же стрелке в функциональной модели могут соответствовать несколько сущностей в информационной модели и, наоборот, одной сущности могут соответствовать несколько стрелок.

Содержание отчета: тема и цель работы; таблица отчета о верификации.

Контрольные вопросы

1 Для чего необходимо связывание? Какие ассоциации задаются для сущностей? Что означает ассоциация IRUN?

2 Для чего служит кнопка Migrate на закладке Arrow Data?

3 Как создать отчет о связывании? Какие поля можно отразить в отчете?

6 Access. Создание и заполнение таблиц

Цель: научиться создавать и заполнять таблицы средствами MS Office Access (выпуска 2013 и выше).

6.1 Теоретические положения

При разработке структуры таблицы прежде всего необходимо определить названия полей, из которых она должна состоять, типы полей и их размеры (таблица 1). Каждому полю таблицы присваивается уникальное имя, которое не может содержать более 64 символов [6].

Таблица 1 – Характеристики типов данных в Access

Тип	Описание	Размер
Текстовый	Текст (одна строка) или числа, не требующие проведения расчетов, например, номера телефонов, коды	До 255 символов
Поле MEMO	Длинный текст или комбинация текста и чисел	До 65535 символов
Числовой	Числовые данные, используемые для проведения расчетов	1, 2, 4, 8 байт
Дата/время	Дата и/или время, относящиеся к годам с 100 по 9999	8 байт
Денежный	Специальный формат для представления числовых данных. Точность – до 15 знаков в целой и до 4 знаков в дробной части	8 байт
Счетчик	Уникальные последовательно возрастающие (на 1) или случайные числа, автоматически вводящиеся при добавлении каждой новой записи в таблицу	4 байта
Логический	Поля, которые могут содержать одно из двух возможных значений (True/False, Да/Нет)	1 бит
Поле объекта OLE	Объект, связанный или внедренный в таблицу MS Access	До 1 Гбайт
Гиперссылка	Ссылка на файл (в данном компьютере, в локальной сети, в Internet)	До 2048 символов
Вложения	Файлы произвольного типа	

Для каждого типа данных можно установить значения свойств, которые будут определять особенности обработки соответствующего поля. Среди всех свойств можно выделить общие, которые можно задать для большинства типов (таблица 2).

Таблица 2 – Основные свойства большинства типов данных

Свойство	Описание
Маска ввода	Символы редактирования, определяющие способы ввода данных
Подпись	Устанавливается информативное название поля, которое автоматически будет использоваться при создании форм и отчетов
Значение по умолчанию	Значение, которое будет использоваться по умолчанию, т. е. в том случае, если в данное поле не будет введена информация
Условие на значение	Устанавливает ограничение на вводимые данные, т. е. не позволяет вводить в поле данные, не соответствующие указанному условию
Сообщение об ошибке	Задает текст сообщения, отображаемого, если данные, введенные в поле, не соответствуют ограничению в свойстве Условие на значение
Обязательное поле	Определяет режим обязательного ввода информации в данное поле
Индексированное поле	Устанавливает режим использования индекса для данного поля, что позволяет ускорить доступ к информации в поле, а также задать режим, при котором в поле нельзя вводить повторяющиеся значения

В Access существует возможность ввода данных в определенном формате. Для этого соответствующему полю необходимо задать маску ввода, определяющую формат вводимой информации. Например, можно ограничить диапазон символов, вводимых в поле, только цифрами или только буквами.

Существует несколько способов создания пустой таблицы: мастер таблиц; ввод данных непосредственно в пустую таблицу в режиме таблицы; определение всех параметров макета таблицы в режиме конструктора.

Структура базы данных задается с помощью схемы данных. В ней определяются и запоминаются связи между таблицами. Схема данных открывается командой «Работа с базами данных» → «Схема данных». При этом появляется окно «Добавление таблицы», с помощью которого на схему добавляются таблицы базы данных. Между таблицами устанавливаются связи 1:1 и 1:М. При нажатии левой клавишей мыши по связи из контекстного меню выбирается «Изменить связь», и в появившемся диалоговом окне «Связи» нужно установить флажок «Обеспечение целостности данных».

При построении связи Access называет одну из таблиц главной, а другую – связанной. Если одно из связываемых полей является ключевым или просто имеет уникальный индекс, главной будет таблица, содержащая это поле. Или главной считается таблица, с которой было начато прокладывание связи.

Создание связи «*один-к-одному*». Оба общих поля (как правило, поля первичного ключа и внешнего ключа) должны иметь уникальный индекс. Это означает, что свойство «Индексированное поле» должно иметь для этих полей значение «Да (Совпадения не допускаются)».

Создание связи «*один-ко-многим*». Поле на стороне «один» связи (как правило, это первичный ключ) должно иметь уникальный индекс. Это означает, что свойство «Индексированное поле» данного поля должно иметь значение «Да (Совпадения не допускаются)». Полю на стороне «многие» не должен соответствовать уникальный индекс. У него может быть индекс, однако он должен допускать совпадения. Это означает, что его свойство «Индексированное поле» может иметь значение «Нет» либо «Да (Допускаются совпадения)».

Если перетащить поле, не являющееся ключевым и не имеющее уникального индекса, на другое поле, которое также не является ключевым и не имеет уникального индекса, создается неопределенное отношение. В запросах, содержащих таблицы с неопределенным отношением, MS Access по умолчанию отображает линию объединения между таблицами, но условия целостности данных не накладываются и нет гарантии уникальности записей в таблицах.

Для связей можно установить следующие свойства:

– «Обеспечение целостности данных» – запрещает помещать в поле связанной таблицы значения, которые отсутствуют в поле главной таблицы;

– «Каскадное обновление связанных полей» – при изменении значения поля в главной таблице будут автоматически меняться соответствующие значения в связанной таблице;

– «Каскадное удаление связанных записей» – если удаляется запись из главной таблицы, должны автоматически удалиться связанные с ней записи из связанной таблицы.

При включении обеспечения целостности данных должны выполняться следующие условия:

– общее поле главной таблицы должно быть первичным ключом или иметь уникальный индекс (общие поля могут иметь разные имена);

– общие поля должны иметь одинаковый тип данных. Единственное исключение – поле типа «Счетчик» можно связать с полем типа «Числовой», если свойство «Размер поля» обоих полей одинаково (например, «Длинное целое»).

Задание

Для сгенерированной из ERwin базы данных необходимо заполнить базу 200 записями (в сумме для всех таблиц).

Контрольные вопросы

1 Перечислить типы данных для полей в Access.
2 Как обеспечивается ссылочная целостность при изменении таблиц БД?
3 В каких случаях невозможно создать связь с параметрами целостности?
4 Что представляет собой режим каскадного обновления связанных полей?

7 Access. Создание запросов

Цель: приобрести навыки работы в СУБД Access по построению запросов.

7.1 Теоретические положения

Запросы создаются для выборки данных из одной или нескольких связанных таблиц по заданным условиям, для проведения вычислений и статистической обработки данных. С помощью запроса можно обновить, удалить, доба-

вить данные в таблицу, создать новые таблицы (таблица 3).

При выполнении запроса на выборку Access извлекает записи из таблиц и формирует результирующий набор данных – динамический (или виртуальный) набор записей, не хранящийся в базе данных, т. е. прекращающий свое существование после закрытия запроса. Сохраняется только структура запроса – перечень таблиц, список полей, порядок сортировки, ограничения на записи, тип запроса и т. д.

Запросы в СУБД Access можно создавать с помощью: режима мастера; режима конструктора, являющегося графическим инструментом языка QBE (Query-by-Example – язык запросов по образцу); языка SQL (используется диалект Jet SQL).

Таблица 3 – Виды запросов в Access

Наименование	Назначение	Способы реализации
1	2	3
Запрос на выборку	Позволяет отобразить записи из одной или нескольких таблиц по указанным полям, сгруппировать записи для вычисления сумм, средних значений и т. д.	Вкладка «Создание» → Кнопка «Конструктор запросов». Для шаблонов в поиске используются следующие символы: ? или _ заменяет любой текстовый символ; # – любую одиночную цифру (0-9); * или % – любое количество символов; [а-л] – символы в заданном интервале; [!м-ю] – символы вне заданного интервала. Например, условие Like "М*" выбирает записи со значениями поля, которые начинаются на букву М
Запрос на выборку с параметром	При выполнении выводит приглашение для ввода данных	В строку «Условие отбора» вводят имя параметра в квадратных скобках, отличающееся от имени поля
Перекрестный запрос	Представляет результаты в виде сводной таблицы с группировкой по строкам и столбцам и вычислениями по заданной функции (Sum, Min, Max, Avg, Count и т. д.). Значения группируются по двум наборам данных, один из которых расположен в левом столбце таблицы, а второй – в верхней строке	Вкладка «Создание» → Кнопка «Конструктор запросов» → Запрос на выборку → Добавить нужные таблицы или запросы и задать условия отбора → Изменить тип запроса на «Перекрестный». Для полей, значения которых группируются по строкам, в строке «Перекрестная таблица» выбрать «Заголовки строк». Для поля, значения которого представлены в запросе как заголовки столбцов, в строке «Перекрестная таблица» выбрать «Заголовки столбцов». Для поля, по которому производятся вычисления, в строке «Перекрестная таблица» выбрать «Значения», а в строке «Групповая операция» – нужную функцию. Для полей, содержащих условие, но без группировки, в строке «Групповая операция» выбирается «Условие», а строка «Перекрестная таблица» оставляется пустой

Окончание таблицы 3

1	2	3
Запрос на выборку с групповыми операциями	Позволяет выделить группы записей с одинаковыми значениями в указанных полях группировки и выполнить статистические вычисления по заданной функции (Sum, Min, Max, Avg, Count (подсчет количества непустых значений поля в группе)). Результат содержит по одной записи для каждой группы	В бланк запроса включают поля, по которым производится группировка, и поля, для которых выполняются групповые функции. При нажатии кнопки «Итоги» на вкладке «Конструктор» в бланке появится строка «Групповая операция», в которой для вычисляемых полей вместо «Группировка» указывают функцию
Запрос Повторяющиеся записи	Предназначен для поиска записей с одинаковыми значениями заданных полей	Вкладка «Создание» → Кнопка «Мастер запросов»
Запрос Записи без подчиненных	Предназначен для поиска записей основной таблицы, у которых нет связанных записей в подчиненной таблице	Вкладка «Создание» → Кнопка «Мастер запросов». Указываются анализируемая (главная) таблица, подчиненная таблица и поля связи
Запрос на обновление	Используется для обновления значений указанных полей таблицы новыми значениями	Вкладка «Создание» → Кнопка «Конструктор запросов» → Запрос на выборку → Добавить обновляемую таблицу → Изменить тип запроса на «Обновление». Для обновляемого поля в строке «Обновление» вводится выражение, вычисляющее значение
Запрос на добавление	Предназначен для вставки записей из одной или нескольких таблиц в одну целевую таблицу	Вкладка «Создание» → Кнопка «Конструктор запросов» → Запрос на добавление
Запрос на создание таблицы	Служит для создания новой таблицы на основе записей существующих таблиц, обычно использует группировку или содержит вычисляемые поля	Строится как запрос на выборку, а затем тип запроса меняется на «Создание таблицы» и в появившемся диалоговом окне задается имя новой таблицы
Запрос на удаление	Предназначен для удаления записей, удовлетворяющих заданному условию	Вкладка «Создание» → Кнопка «Конструктор запросов» → Запрос на выборку → Добавить требуемую таблицу → Изменить тип запроса на «Удаление». В первом столбце в первом поле бланка ставится «ИмяПоля.*» (т. е. «все поля»), после чего в строке «Удаление» будет выведен текст «Из». В следующих столбцах выбирают поля, для которых задают условия отбора

Задание

Для спроектированной базы данных необходимо разработать 15 запросов, представляющих изученные виды запросов (с учетом запросов в функциональной модели, разработанной в лабораторной работе № 2).

Содержание отчета: тема и цель работы; SQL-код запросов.

Контрольные вопросы

1 Что такое запрос? Какие способы создания запросов существуют?

2 Как при создании запроса установить условия отбора? Как осуществить поиск данных в диапазоне значений? Как используются выражения в запросе? Как сортируются данные в запросе? Как установить параметры в запросе? Как суммировать данные в запросе?

3 Каково назначение запросов с групповыми операциями и перекрестных запросов? Какие функции используются в данных запросах?

4 Логические операторы и их назначение. Назначение операторов BETWEEN, IN, LIKE.

5 Встроенные функции даты и времени. Встроенные функции и операторы для работы с текстом. Встроенные функции преобразования типов данных.

6 Как создаются запросы на обновление, добавление, удаление?

8 Access. Создание форм и отчетов

Цель: приобрести навыки работы в СУБД MS Access по созданию форм и отчетов.

8.1 Теоретические положения

Формы являются основным средством организации интерфейса пользователя в приложениях Access [6].

Существуют следующие способы создания формы:

– с помощью инструмента «Форма», который позволяет создать форму на основе указываемой готовой таблицы (или запроса);

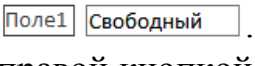
– с помощью конструктора форм, который позволяет создавать и редактировать формы любой степени сложности;

– с помощью мастера форм (создание простых настраиваемых форм);

– с помощью инструмента «Разделенная форма», что позволяет одновременно отображать данные в двух представлениях – в режиме таблицы, отображаемой в верхней части формы, и в режиме формы для ввода данных в запись, выделенную в таблице;

– с помощью инструмента «Несколько элементов», что позволяет создать форму, в которой отображается несколько записей;

– с помощью инструмента «Пустая форма».

Создание вычисляемых полей в форме. Следует открыть форму в режиме конструктора и найти панель элементов  , далее выбрать элемент «Поле» и перенести его на свободное место формы. На форме появится элемент 

. Переименовать подпись *Поле1*, например, в *Итого*. Щелкнув правой кнопкой мыши по окошку с надписью *Свободный*, выбрать «Свойства». В открывшемся диалоговом окне во вкладке «Данные» в строке «Данные» открыть с помощью кнопки  «Построитель выражений». В окне «Построитель

выражений» можно записать математические выражения, выбирая нужные таблицы и поля из этих таблиц, участвующие в вычислениях.

Создание кнопок на форме. Если на форме поместить кнопку, то в появившемся окне «Создание кнопок» в разделе «Категории» можно связать кнопку с каким-либо действием – открытием другой формы, выполнением запроса, печатью таблицы, выходом из приложения и т. д.

Кнопочная форма – это форма, содержащая набор кнопок, направляющих пользователя к другим формам (обычно при щелчке мышью кнопки формы), т. е. своего рода главное меню БД.

Для автоматического создания кнопочной формы следует выбрать на ленте «Работа с базами данных» → «Диспетчер кнопочных форм» (Database Tools → Switchboard Manager). Если диспетчер кнопочных форм отсутствует, то его надо включить. Для этого следует выбрать пункт меню «Файл», выбрать раздел «Параметры» и в нем – «Панель быстрого доступа». Затем на вкладке «Настройка панели быстрого доступа» выбрать из раскрывающегося списка строку «Вкладка «Работа с базами данных» и в списке команд выделить «Диспетчер кнопочных форм», а дальше нажать на кнопку «Добавить». При первом нажатии этой кнопки Access сообщит, что не может найти кнопочную форму и предложит ее создать. Диспетчер кнопочных форм выводит на экран список страниц. Каждая страница – отдельная часть меню кнопочной формы.

Окно «Изменение страницы кнопочной формы» позволяет создавать команды меню, удалять те, которые больше не нужны, изменять порядок их следования (этот порядок определяет порядок команд на кнопочной форме). Для создания команды меню необходимо указать текст, появляющийся на форме, и команду, которую должна выполнить Access при нажатии кнопки. На каждой странице кнопочной формы можно поместить только восемь команд меню. Если нужно больше, следует добавить дополнительные страницы в меню.

Автоматический запуск главной кнопочной формы при открытии приложения. Предполагается, что форма, которая должна отображаться первой, уже создана. Следует открыть вкладку «Файл» и выбрать «Параметры». В открывшемся окне в левой колонке в группе «Текущая база данных» в списке «Форма просмотра» выбрать форму, отображаемую при запуске БД, и нажать ОК. Чтобы отобразилась начальная форма, нужно закрыть БД и открыть ее повторно.

Макросы – это небольшие программы на языке макрокоманд СУБД Access, состоящие из последовательности определенных команд (одной или нескольких макрокоманд). Макросы являются простейшими средствами автоматизации действий над объектами Access. Если нажать правой кнопкой мыши по полю кнопки и выбрать **Обработка событий → Макросы**, **о**ткроется окно конструктора макросов, в котором надо выбрать макрокоманду из выпадающего списка (всего имеется около 50 различных макрокоманд). Все созданные макросы будут отображаться во вкладке **Макросы.**

Отчеты используются для отображения информации, содержащейся в таблицах, в отформатированном виде, который легко читается как на экране компьютера, так и на бумаге.

Отчет можно отобразить в трех режимах: в режиме конструктора (что поз-

воляет изменить внешний вид и макет отчета), в режиме просмотра образца (можно просмотреть все элементы готового отчета, но в сокращенном виде), в режиме предварительного просмотра.

Отчет можно создать в режиме конструктора и в режиме мастера.

Разработка и форматирование отчета аналогичны форме.

Задание

Для созданной базы данных необходимо для всех таблиц разработать формы, оформить главную кнопочную форму. Должны быть реализованы все отчеты, указанные в функциональной модели в лабораторной работе № 2.

Содержание отчета: тема и цель работы; скриншоты форм и отчетов.

Контрольные вопросы

1 Какие способы создания форм существуют?

2 Что такое главная кнопочная форма и как организовать автоматический запуск главной кнопочной формы при открытии приложения?

3 Для чего служат отчеты и из каких разделов они могут состоять?

4 Какие элементы форматирования используют для оформления отчетов?

5 Как в отчете задаются уровни группировки и сортировка?

6 С какой целью и в какие разделы отчета добавляются вычисляемые поля?

9 Технология создания баз данных на основе промышленной СУБД MS SQL Server

Цель: получить навыки установки MS SQL Server; научиться создавать первичные и внешние ключи, создавать отношения между таблицами, создавать, настраивать и удалять индексы.

9.1 Теоретические положения

Microsoft SQL Server – система управления реляционными базами данных, использующая язык запросов Transact-SQL (T-SQL). Этапы установки SQL Server описаны в [15].

Для выполнения большинства действий над базой данных удобно использовать инструмент Microsoft SQL Server Management Studio (SSMS).

После генерации схемы БД из ERwin в MS SQL Server необходимо открыть получившиеся таблицы, проверить установку первичных ключей и перейти к построению схемы БД. Для этого в обозревателе объектов нужно выбрать «Диаграммы баз данных» и в контекстном меню – «Создать диаграмму базы данных», добавить все таблицы в окно схемы БД. Для определения связей следует «ухватиться» за одно поле и «перетащить» его на второе. При опреде-

лении связи появляется окно «Create Relationship», в котором можно задать название связи в поле «Relationship name».

Отсоединение и присоединение БД используется для переноса базы БД между различными экземплярами SSMS. Например, разработанная под управлением сервера S1 БД может быть отсоединена от этого сервера, скопирована на диск другого компьютера и присоединена к серверу S2 на втором компьютере. Самый простой способ присоединить БД – в обозревателе объектов SSMS выбрать «Базы данных» → «Присоединить», отсоединить БД – выбрать «Базы данных», выделить имя отсоединяемой БД и в контекстном меню выбрать «Задачи» → «Отсоединить». Отсоединение базы данных также может быть выполнено с помощью системной процедуры sp_detach_db, а присоединение – двумя способами: с помощью системной процедуры sp_attach_db и с помощью оператора *CREATE DATABASE* с опцией *FOR ATTACH* .

Задание

Необходимо сгенерировать схему БД из ERwin в MS SQL Server.
Содержание отчета: тема и цель работы; схема БД в MS SQL Server.

Контрольные вопросы

1 Перечислить типы файлов БД в SQL Server и пояснить их назначение.
2 С помощью каких операторов T-SQL создается и удаляется БД?
3 Для чего применяется отсоединение и присоединение БД?
4 Сколько схем БД можно создать? Что такое ограничения целостности БД?

10 Создание таблиц средствами SQL

Цель: получить навыки создания таблиц средствами T-SQL.

10.1 Теоретические положения

Для создания таблицы используется следующая инструкция T-SQL:

```
CREATE TABLE [ database [owner]. | owner. ] name
( { <column_definition>
| column_name AS computed_col_expr
|<table_constraint>} )
[ON {filegroup |  DEFAULT}]
```

Ключевое слово ON позволяет указать файловую группу, в которой будет располагаться таблица. Здесь есть две возможности: либо явно указать имя файловой группы (причем она уже должна существовать в базе данных), либо использовать ключевое слово DEFAULT, которое предписывает системе распо-

ложить таблицу в файловой группе по умолчанию.

Определение каждой колонки таблицы, в синтаксисе команды обозначенное как <column_definition>, имеет следующий формат:

{column_name data_type}
[DEFAULT constant
| [IDENTITY [(seed, increment) [NOT FOR REPLICATION]]]]
[ROWGUIDCOL]
[<column_constraint>]

Подобным образом необходимо описать каждую колонку в таблице. Прежде всего следует определить имя колонки (column_name), а также тип хранимых в ней данных (data_type). При описании могут быть использованы следующие ключевые слова.

DEFAULT – определяет значение по умолчанию (constant_expression), используемое, если при вводе строки явно не указано другое значение.

IDENTITY – предписывает системе осуществлять заполнение колонки автоматически. При этом также указать начальное значение (seed) и приращение (increment). В случае, когда указано NOT FOR REPLICATION, эта колонка не будет автоматически заполняться для строк, вставляемых в таблицу в процессе репликации, так что эти строки сохранят свои значения.

ROWGUIDCOL – данная колонка будет использоваться для хранения глобального идентификационного номера.

Кроме того, для колонки можно определить ограничения на значения:

[CONSTRAINT constraint_name]
{ [NULL | NOT NULL] | [{ PRIMARY KEY | UNIQUE }
[CLUSTERED | NONCLUSTERED]
[WITH FILLFACTOR = fillfactor]
[ON { filegroup | DEFAULT}
| [[FOREIGN KEY]
REFERENCES ref_table]

Наложение ограничения на значение колонки должно начинаться с ключевого слова CONSTRAINT, после которого необходимо указать имя ограничения на значение. Для каждого ограничения желательно указать, допустимо ли для колонки значение NULL, выбрав соответствующее ключевое слово (NULL или NOT NULL). После этого требуется определить тип ограничения.

PRIMARY KEY – определяет колонку как первичный ключ таблицы. В качестве альтернативы можно определить колонку как уникальную, воспользовавшись ключевым словом UNIQUE. При необходимости можно также указать, будет ли индекс, создаваемый для данного ограничения, кластерным (ключевое слово CLUSTERED) или некластерным (NONCLUSTERED). Однако кластерный индекс можно определить только для одного ограничения, поэтому требуется решить, с каким ограничением (первичный ключ или уникальная колонка)

он будет использоваться. Если создается индекс, необходимо также указать степень заполнения его страниц (ключевое слово WITH FILLFACTOR).

FOREIGN KEY – определяет колонку как внешний ключ таблицы. Одновременно, используя ключевое слово REFERENCES, необходимо указать имя таблицы, с которой будет связана создаваемая таблица. Дополнительно потребуется указать ее колонки, которые будут связаны с данной колонкой.

Следует заметить, что можно определить виртуальную колонку, которая будет производной от других колонок. При этом на хранение ее значений не требуется физической памяти, поскольку они в любой момент могут быть получены для любой строки путем вычислений. Определение виртуальной колонки имеет следующий формат: *имя_колонки AS выражение*. Выражение может включать имена колонок, функции и арифметические операции.

Вопросы создания таблиц описаны в конспекте лекций и в [3, 14].

Задание

Необходимо создать средствами T-SQL три обычных таблицы и три таблицы, обеспечивающие возможность сохранения копий строк из других таблиц при удалении данных.

Содержание отчета: тема и цель работы; прокомментированный SQL-код выполнения задания.

Контрольные вопросы

1 Перечислить этапы проектирования структуры таблиц.

2 С помощью каких команд T-SQL можно создать, изменить или удалить таблицу? Для чего нужны уникальные колонки, проверочные ограничения на значения колонок, идентификационные колонки?

3 Охарактеризовать механизмы управления значениями колонок (Primary Key, Foreign Key, Unique, Check, Defaults, NULL).

11 Изменение таблиц средствами SQL

Цель: изучить изменение структуры таблиц средствами T-SQL.

11.1 Теоретические положения

Для изменения структуры таблицы средствами T-SQL предусмотрена специальная команда:

```
ALTER TABLE table
{[ALTER COLUMN
column_name {new_data_type [(precision [, scale])]
[NULL | NOT NULL]
```

```
| {ADD | DROP}  ROWGUIDCOL}]
|ADD
{[<column_definition> ]
column_name AS computed_column_expression}[,...n]
[WITH CHECK | WITH NOCHECK] ADD
{<table_contraint>}[...n]
|DROP
[CONSTRAINT] contraint_name |  COLUMN column }[....n]
{CHECK  |  NOCHECK} CONSTRAINT {ALL | constraint_name[,...n]}
{ENABLE  | DISABLE} TRIGGER {ALL | trigger_name[....n]}}
```

Рассмотрим синтаксис данной команды. С помощью команды ALTER TABLE можно изменить определение уже существующих колонок, удалить любую из них, а также добавить в таблицу новые колонки.

Изменение определения колонки. Данная операция осуществляется с использованием ключевого слова ALTER COLUMN, после которого помещается имя изменяемой колонки (column_name). Можно изменить тип данных колонки (new_data_type), размерность (precision) и точность (scale). Можно указать, разрешено ли колонке содержать значения NULL. В этом случае обязательно нужно указать тип данных для колонки, даже если вы не хотите его изменять (просто укажите существующий тип данных). Если определяется для колонки свойство NOT NULL, следует позаботиться о том, чтобы на момент изменения колонка не содержала ни одного значения NULL.

Добавление в таблицу новой колонки. Для определения новой колонки необходимо использовать ключевое слово ADD. За ним следует описание колонки, которое имеет такой же формат, как и при создании колонки с помощью команды CREATE TABLE. Здесь же можно наложить на таблицу новые ограничения на значения колонок. Определив ключевое слово WITH CHECK, системе предписывается при добавлении новых ограничений на значения колонок FOREIGN KEY или CHECK осуществлять проверку данных в таблице на соответствие этим ограничениям. По умолчанию данная проверка проводится для всех вновь создаваемых ограничений. Когда выполнение подобной проверки не требуется, необходимо использовать ключевое слово WITH NOCHECK.

Удаление колонок из таблицы. В случае необходимости можно удалить из таблицы некоторые колонки. Для этого используется ключевое слово DROP. Можно удалить как конкретную колонку (ключевое слово COLUMN), так и определенное ограничение на значение колонки. Однако нельзя удалять следующие колонки: колонки, задействованные в индексе; колонки, полученные в результате репликации; колонки, для которых определены любые ограничения на значения; колонки, для которых определены значения по умолчанию.

Управление ограничениями на значения колонок. Иногда бывает необходимо отключить ограничения на значения колонок FOREIGN KEY или CHECK. Отключение конкретного ограничения (NOCHECK CONSTRAINT) означает, что при вводе новых строк данные не будут проверяться на соответствие этому ограничению. Когда снова потребуется сделать ограничение активным, исполь-

зуйте ключевое слово CHECK CONSTRAINT. При необходимости можно управлять всеми ограничениями сразу с помощью ключевого слова ALL.

Управление триггерами. Ключевое слово DISABLE TRIGER отключает триггер. При этом в процессе изменения данных в таблице те действия, которые определены в триггере как реакция системы на эти изменения, не производятся, хотя триггер продолжает существовать. Чтобы активизировать триггер, необходимо использовать команду с ключевым словом ENABLE TRIGGER. Если требуется управлять сразу всеми триггерами, используется ключевое слово ALL.

Более подробно вопросы изменения таблиц описаны в конспекте лекций и в [3, 14].

Задание

Необходимо изменить структуру таблиц, созданных в лабораторной работе № 10, так, чтобы в них можно было хранить информацию о времени удаления данных из таблицы.

Содержание отчета: тема и цель работы; прокомментированный SQL-код выполнения задания.

Контрольные вопросы

1 Синтаксис команды изменения структуры таблицы средствами T-SQL.
2 Как изменить определение колонки таблицы?
3 Как добавить новую колонку или удалить колонку из таблицы?
4 С помощью каких команд производится управление ограничениями на значения колонок и триггерами?

12 Создание отношений между таблицами средствами SQL

Цель: создать отношения между таблицами средствами T-SQL.

12.1 Теоретические положения

Вопросы создания отношений между таблицами описаны в конспекте лекций и в [3, 14].

Для реализации связей 1:1 и 1:M необходимо, чтобы одна из таблиц содержала ссылку (внешний ключ) на вторую. Внешний ключ – это столбец (или группа столбцов таблицы), содержащий значения, совпадающие со значениями первичного ключа в этой же или другой таблице.

Синтаксис предложения FOREIGN KEY следующий:

```
[CONSTRAINT c_name]
[[FOREIGN KEY] ({col_name1} ,...)]
REFERENCES table_name ({col_name2},...)
```

[ON DELETE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}]
[ON UPDATE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}]

Предложение FOREIGN KEY явно определяет все столбцы, входящие во внешний ключ. В предложении REFERENCES указывается имя таблицы, содержащей столбцы, создающие соответствующий первичный ключ. Количество столбцов и их тип данных в предложении FOREIGN KEY должны совпадать с количеством соответствующих столбцов и их типом данных в предложении REFERENCES (и, конечно же, они должны совпадать с количеством столбцов и типами данных в первичном ключе таблицы, на которую они ссылаются).

Таблица, содержащая внешний ключ, называется ссылающейся (или дочерней), а таблица, содержащая соответствующий первичный ключ, – ссылочной или родительской.

Например, создадим две таблицы, связанные отношением 1:1. Столбец таблицы B нужно сделать внешним ключом или уникальным. Это гарантирует, что в таблице B может быть только один столбец, который соответствует полю PRIMARY KEY в таблице A.

```
CREATE TABLE TableB (
Id INT PRIMARY KEY IDENTITY(1,1),
Name VARCHAR(255))

CREATE TABLE TableA (
Id INT PRIMARY KEY IDENTITY(1,1),
Name VARCHAR(255),
TableBRelation  INT UNIQUE,
FOREIGN KEY (TableBRelation) REFERENCES TableB (Id))
```

Задание

Необходимо установить отношения между резервными таблицами из лабораторной работы № 10, соответствующие отношениям между основными таблицами, средствами T-SQL.

Содержание отчета: тема и цель работы; прокомментированный SQL-код выполнения задания.

Контрольные вопросы

1 Какие типы отношений поддерживаются в MS SQL Server?
2 Для чего предназначены предложения FOREIGN KEY и REFERENCES?
3 Как создаются отношения 1:1, 1:M, M:M?

13 Создание sql-скрипта заполнения базы данных

Цель: получить навыки создания sql-скриптов заполнения БД.

13.1 Теоретические положения

Сценарий (скрипт) – последовательность операторов T-SQL. Для подготовки сценария, отладки и выполнения используется SSMS.

Автоматическая генерация скриптов из обозревателя объектов. Для таблицы можно сгенерировать скрипты для создания таблицы, удаления таблицы, выборки данных из таблицы, скрипты для добавления новых данных в таблицу, а также для изменения и удаления существующих записей. В обозревателе объектов для таблицы текущей БД из контекстного меню выбирается команда «Создать сценарий для таблицы» → «Используя CREATE».

Для сценария самой БД после выделения имени БД из контекстного меню выбирается команда «Задачи» → «Сформировать сценарии» → «Внести в сценарий всю базу данных и все объекты базы данных» → «Тип вывода данных: Сохранить сценарии в заданном месте» → «Дополнительные параметры создания сценария: Типы данных для внесения в сценарий – Схема и данные».

При запуске SSMS и подключении к SQL серверу по умолчанию выполняется команда Use Master, т. е. работа идет с системной БД Master. Для выбора иной БД следует выполнить команду Use Имя_Базы_Данных. Перед запуском на исполнение сгенерированного скрипта БД аналогичным образом в первой строке скрипта нужно указывать Use Имя_Базы_Данных.

Для создания скрипта для административных операций (например, резервное копирование базы данных, создание учетной записи и т. д.) можно воспользоваться контекстным меню «Задачи» → «Создать резервную копию…» → «Сценарий», что позволит автоматически создать скрипт, в который будут подставлены введенные в полях формы на экране значения.

Задание

Необходимо, используя команды INSERT, внести в базу данных не менее 200 записей. Пример команды INSERT:

INSERT INTO Имя_таблицы(Код, Наименование) VALUES(1, ’Коробка’)

Для заполненной БД сгенерировать скрипт заполнения.
Содержание отчета: тема и цель работы; скрипт заполнения БД.

Контрольные вопросы

1 Что такое скрипт (сценарий) и для решения каких задач он используется?
2 Какие способы создания скриптов существуют?
3 Для чего предназначена системная БД Master?

14 Язык SQL. Добавление, изменение и удаление данных в таблицах средствами SQL

Цель: научиться добавлять, изменять и удалять данные в таблицах с помощью команд T-SQL.

14.1 Теоретические положения

Для вставки данных в таблицу средствами T-SQL используется команда INSERT, имеющая следующий синтаксис [3, 14]:

```
INSERT [INTO] { имя_таблицы | имя_ представления}
{[ (список_столбцов) ]
{ VALUES ( { DEFAULT | NULL
| выражение }[,... n ] ) | временная_таблица | инструкция_выполнения }}
```

Ключевое слово INSERT и необязательное ключевое слово INTO вводят инструкцию. Аргумент «имя_ таблицы» задает целевую таблицу, в которую необходимо вставить данные. Можно в качестве цели вставки задать аргумент «имя_представления». Аргумент «список_столбцов» представляет необязательный список столбцов, разделенных запятыми, который будет получать вставляемые данные. Вставляемые значения можно задать ключевыми словами DEFAULT, NULL или выражениями. В качестве альтернативы можно использовать инструкцию SELECT для создания временной таблицы, которая станет поставщиком данных для вставки.

Инструкция UPDATE реализует один из способов изменения любых данных, содержащихся в таблице. Можно написать инструкцию UPDATE таким образом, чтобы она влияла только на отдельное поле в отдельной строке либо чтобы она вычисляла изменения в столбце во всех строках таблицы. Можно также написать инструкцию, которая будет изменять все строки из множества столбцов. Синтаксис инструкции UPDATE:

```
UPDATE { имя_таблицы | имя_представления }
SET { имя_столбца = {выражение | DEFAULT | NULL}
| @ переменная = выражение | @переменная-столбец = выражение
{ [FROM {исходная_таблица} [,... n]]
[WHERE условие_поиска] }
```

В качестве исходного материала для обновляемых строк нужно задать имя таблицы или представления. Ключевое слово SET представляет вносимые изменения. Можно задать значение столбца равным выражению, значению по умолчанию или пустому значению NULL. Можно присвоить выражение локальной переменной. Можно объединить присвоение значения локальной переменной и столбцу в одном и том же выражении. В одной директиве SET

можно задать множество столбцов.

Для удаления записей при помощи запросов из существующей таблицы можно использовать инструкцию DELETE. Инструкция DELETE имеет множество опций, но базовый синтаксис ее довольно прост:

```
DELETE [FROM]
{ имя_таблицы | имя_представления }
[FROM исходная_таблица] [WHERE условия_поиска]
```

Ключевое слово DELETE идентифицирует инструкцию. Необязательное ключевое слово FROM можно использовать для удобочитаемости инструкции SQL. В качестве исходного объекта для удаляемых строк следует задать имя таблицы или представления. Предложение FROM имеет такой же синтаксис и параметры, что и предложение FROM в инструкции SELECT. Предложение WHERE имеет такой же синтаксис и параметры, как и предложение WHERE в инструкции SELECT.

Более подробно вопросы модификации данных в таблицах описаны в конспекте лекций и в [3, 4, 9, 14, 15].

Задание

Необходимо для разрабатываемой БД написать по три команды INSERT, UPDATE, DELETE для каждой таблицы.

Содержание отчета: тема и цель работы; SQL-код выполнения задания.

Контрольные вопросы

1 Объяснить синтаксис команд INSERT, SELECT INTO, UPDATE, DELETE и указать ограничения для данных команд.

2 Привести примеры использования команд INSERT, UPDATE, DELETE.

15 Язык SQL. Работа с представлениями

Цель: научиться создавать представления в MS SQL Server средствами Transact-SQL.

15.1 Теоретические положения

Представление – это именованный запрос на выборку, сохраненный в базе данных, который выглядит и работает как таблица, при обращении по имени создает виртуальную таблицу, наполняя ее актуальными данными из БД, с которой можно работать так же, как с реально существующей на диске таблицей. Физически представление реализовано в виде SQL-запроса, на основе которого производится выборка данных из одной или нескольких таблиц или представ-

лений. Представление часто применяется для ограничения доступа пользователей к конфиденциальным данным в таблице.

Для создания представлений средствами Transact-SQL используется следующая конструкция:

```
CREATE VIEW view_name [(column [,...n])]
[WITH ENCRYPTION]
AS
select_statement
[WITH CHECK OPTION]
```

Рассмотрим составляющие данной конструкции.

view_name – имя представления. При указании имени необходимо придерживаться тех же правил и ограничений, что и при создании таблицы.

column – имя колонки, которое будет использоваться в представлении (длина имени до 128 символов). Имена колонок перечисляются через запятую в соответствии с их порядком в представлении. Имена колонок можно указывать в команде SELECT, определяющей представление.

WITH ENCRYPTION – данный параметр предписывает серверу шифровать код SQL-запроса. Это гарантирует, что пользователи не смогут просмотреть код запроса и использовать его. Если при определении представления необходимо скрыть имена исходных таблиц и колонок, а также алгоритм объединения данных, то следует использовать эту опцию.

select_statement – код запроса SELECT, выполняющий выборку, объединение и фильтрацию строк из исходных таблиц и представлений. Можно использовать команду SELECT любой сложности со следующими ограничениями:

1) нельзя создавать новую таблицу на основе результатов, полученных в ходе выполнения запроса, то есть запрещается использование параметра INTO;

2) нельзя проводить выборку данных из временных таблиц, то есть нельзя использовать имена таблиц, начинающихся на # или ##;

3) в представление нельзя включать операции вычисления и группировки, т. е. запрещено указание параметров ORDER BY, COMPUTE и COMPUTE BY.

Чтобы выполнить представление, т. е. получить данные в виде виртуальной таблицы, необходимо выполнить запрос SELECT к представлению так же, как и к обычной таблице: SELECT * FROM view_name.

Для удаления представления используется команда T-SQL DROP VIEW{view […n]}. За один раз можно удалить несколько представлений.

В качестве примера приведено представление, предоставляющее информацию об экземплярах книг, которые были изданы с 2000 до текущего года.

```
CREATE VIEW InfoEkzemplar
AS
SELECT   /* Указываем какие поля будут выбраны */
Экземпляр.шифр, Книга.Автор, Книга.Название, Книга.Год_издания, Книга.Издательство, Экземпляр.Предметная_область, Экземпляр.Номер_отдела,
```

```
Экземпляр.Отметка_о_списании, Экземпляр.Отметка_о_замене
    FROM    /* Указываем таблицу и связанные с ней таблицы, из которых выбираются связанные данные */
    Книга INNER JOIN Экземпляр ON Книга.ISBN = Экземпляр.ISBN
    WHERE Книга.Год_издания BETWEEN 2000 AND YEAR (GETDATE())
    /* GETDATE()  возвращает текущую дату, YEAR – год из даты */
```

Задание

В разрабатываемой базе данных необходимо реализовать 15 представлений с использованием стандартных функций T-SQL. Перечень функций T-SQL содержится в конспекте лекций, а также в [4, 14].

Содержание отчета: тема и цель работы; SQL-код 15 представлений.

Контрольные вопросы

1 Что такое представление и в каких случаях целесообразно его использовать? Перечислить способы создания представлений.

2 Перечислить операторы SQL, с помощью которых представления создаются, удаляются и изменяются.

3 Перечислить ограничения при создании представлений.

4 Перечислить основные группы функций T-SQL и охарактеризовать наиболее часто используемые функции.

16 Язык SQL. Создание хранимых процедур

Цель: научиться создавать хранимые процедуры в СУБД MS SQL Server с использованием команд T-SQL; реализовать хранимые процедуры для вставки, удаления, изменения данных.

16.1 Теоретические положения

Хранимая процедура – это скомпилированный набор SQL-предложений, сохраненный на сервере баз данных как именованный объект и выполняющийся как единый фрагмент кода. Хранимые процедуры могут принимать и возвращать параметры. При этом клиент осуществляет только вызов хранимой процедуры по ее имени, затем сервер базы данных выполняет блок команд, составляющих тело вызванной процедуры, и возвращает клиенту результат.

Хранимые процедуры обычно используются для поддержки ссылочной целостности данных и реализации бизнес-правил. В последнем случае повышается скорость разработки приложений, поскольку если бизнес-правила изменяются, можно изменить только текст хранимой процедуры, не изменяя клиентские приложения. По сравнению с обычными SQL-запросами, посылаемыми из клиентского приложения, они требуют меньше времени для подготовки к выпол-

нению, поскольку скомпилированы и сохранены.

Хранимые процедуры имеют следующее определение:

```
CREATE PROCEDURE] procedure_name [;number]
[ {@parameter data_type} [= default] [OUTPUT] ] [,...n] ]
AS sql_statement [...n]
```

Рассмотрим составляющие данной конструкции.

procedure_name – имя создаваемой процедуры. Используя префиксы sp_, # и ##, можно определить создаваемую процедуру как системную или временную (локальную или глобальную).

number – параметр определяет идентификационный номер хранимой процедуры, однозначно определяющий ее в группе процедур.

@parameter – определяет имя параметра, который будет использоваться создаваемой хранимой процедурой для передачи входных или выходных данных. Параметры, определяемые при создании хранимой процедуры, являются локальными переменными, поэтому несколько хранимых процедур могут иметь абсолютно идентичные параметры.

data_type – определяет, к какому типу данных должны относиться значения параметра описываемой процедуры.

OUTPUT – определяет указанный параметр как выходной.

default – позволяет определить для параметра значение по умолчанию, которое хранимая процедура будет использовать в случае, если при ее вызове указанный параметр был опущен.

AS – ключевое слово, определяющее начало кода хранимой процедуры. После этого ключевого слова следуют команды Transact-SQL, которые и составляют непосредственно тело процедуры (sql statement). Здесь можно использовать любые команды, включая вызов других хранимых процедур, за исключением команд, начинающихся с ключевого слова CREATE.

Более подробно вопросы создания хранимых процедур различных видов рассмотрены в конспекте лекций и в [4, 14].

Далее приведен пример хранимой процедуры, возвращающей количество экземпляров какой-либо книги.

```
CREATE PROCEDURE KolExzemplarov
/*Объявляем необходимые переменные*/
@ISBN varchar(20)
AS
/* Следующая конструкция проверяет, существуют ли записи в таблице «Книги» с заданным ISBN*/
IF not EXISTS (SELECT * FROM Книга WHERE ISBN = @ISBN)
RETURN 0 /*Вызывает конец процедуры KolExzemplarov */
SELECT Экземпляр.ISBN
INTO TEMP1 /*Сохраняет выбранные поля во временной таблице Temp1*/
FROM Экземпляр
```

WHERE ISBN = @ISBN
SELECT COUNT(ISBN) /**Count подсчитывает количество неповторяющихся записей поля ISBN*/
FROM TEMP1

Задание

В разрабатываемой БД необходимо реализовать 20 хранимых процедур, в том числе для вставки, удаления, изменения данных с использованием стандартных функций SQL.

Содержание отчета: тема и цель работы; SQL-код 20 хранимых процедур.

Контрольные вопросы

1 Что такое хранимая процедура? Для чего используется?
2 Какие виды параметров могут использоваться в процедуре?
3 Как производится средствами T-SQL создание, модификация и удаление хранимых процедур? Как создаются хранимые процедуры на вставку, изменение и удаление данных?
4 Описать управление процессом компиляции хранимой процедуры.

17 Язык SQL. Работа с триггерами

Цель: научиться создавать триггеры в MS SQL Server Management Studio.

17.1 Теоретические положения

Триггер – это специальный тип хранимых процедур, который запускается автоматически при выполнении тех или иных действий с данными таблицы. Каждый триггер привязывается к конкретной таблице.

Существует три типа триггеров в зависимости от команд, на которые они реагируют: триггеры на вставку, на обновление и на удаление. Для одной таблицы допускается создание нескольких однотипных триггеров. Более подробно триггеры описаны в конспекте лекций и в [4, 14].

Для создания триггера используется следующая инструкция T-SQL:

CREATE TRIGGER Trigger_name
ON Table_name
{FOR {[DELETE] [,] [INSERT] [,] [UPDATE]}
[WITH APPEND]
AS
sql_statement [...n] }

Trigger_name – задает имя триггера, с помощью которого он будет распознаваться хранимыми процедурами и командами Transact SQL. Имя триггера должно быть уникальным в пределах БД.

Table_name – имя таблицы БД, к которой будет привязан триггер.

[DELETE] [,] [INSERT] [,] [UPDATE] – эта конструкция определяет, на какие автоматы будет реагировать триггер. При создании триггера должно быть указано хотя бы одно из этих ключевых слов, и допускается создание триггера, реагирующего на две или три команды.

WITH APPEND – указание этого ключевого слова требуется для обеспечения совместимости с более ранними версиями SQL Server.

sql_statement – определяет набор команд, которые будут выполняться при запуске триггера.

Далее приведен пример триггера, который будет запрещать удаление записей таблицы «Пользование библиотекой», если текущий пользователь не владелец базы данных и если поле «дата выдачи» содержит какое-либо значение.

```
CREATE TRIGGER udalenie          /*Обьявляем имя триггера*/
ON Пользование_библиотекой2  /* имя таблицы, с которой связан триггер*/
FOR  DELETE  /*Указываем операцию, на которую будет срабатывать триггер (здесь на  удаление)*/
AS
IF ( SELECT count(*)          /*проверяет*/
from Пользование_библиотекой2      /*записи  из  таблицы  «Пользование библиотекой»*/
where  Пользование_библиотекой2.дата_выдачи  is  not  null)>0  /*условие проверяет наличие записи в поле «дата выдачи». Если count возвращает значение, отличное от нуля (т.е. запись есть), то первое условие IF не выполнено*/
AND (CURRENT_USER <> 'dbo')  /*вызывается функция определения имени текущего пользователя и проверяется, владелец ли он*/
BEGIN
PRINT 'у вас нет прав на удаление этой записи'       /*выдача  сообщения  о неудаче операции*/
ROLLBACK TRANSACTION      /*откат (отмена) транзакции*/
END
```

Задание

В разрабатываемой БД необходимо реализовать 8 триггеров.

Содержание отчета: тема и цель работы; SQL-код 8 триггеров.

Контрольные вопросы

1 Что такое триггер? Какие типы триггеров различают?

2 Как создать триггер?

3 С помощью каких команд T-SQL можно изменить или удалить триггер?

18 Язык SQL. Работа с курсорами

Цель: научиться создавать курсоры в MS SQL Server Management Studio.

18.1 Теоретические положения

Курсоры в SQL Server представляют собой механизм обмена данными между сервером и клиентом. Курсор позволяет клиентским приложениям работать не с полным набором данных, а только с одной или несколькими строками.

Существует четыре основных типа курсоров, различающихся по предоставляемым возможностям, – статические, динамические, последовательные и ключевые. Тип курсора определяется на стадии его создания и не может быть изменен. Более подробно курсоры описаны в конспекте лекций и в [4, 14].

Далее рассмотрен пример создания курсора для просмотра информации о студентах и выдачи информации об их количестве.

```
DECLARE curs1 CURSOR
GLOBAL              /*Создается глобальный курсор, который
                    будет существовать до закрытия данного соединения*/
SCROLL              /*Создает прокручиваемый курсор*/
KEYSET              /*Будет создан ключевой курсор*/
TYPE_WARNING
FOR
SELECT              /*Какие поля  будут показаны в курсоре*/
Студенты.Читательский_номер,Студенты.Имя, Студенты.Фамилия, Студенты.Отчество, Студенты.год_поступления, Студенты.год_окончания, Студенты.факультет, Студенты.специальность, Студенты.номер_приказа
FROM Студенты               /*Из какой таблицы выбираются данные*/
FOR READ ONLY               /*Только для чтения*/
OPEN GLOBAL curs1           /*открываем глобальный курсор*/
DECLARE @@Counter int           /*объявляем переменную*/
SET @@Counter =@@CURSOR_ROWS    /*присваиваем ей число рядов курсора*/
Select @@Counter            /*выводим результат на экран*/
CLOSE curs1                 /*закрываем курсор*/
DEALLOCATE curs1            /*освобождаем курсор*/
```

Задание

В разрабатываемой БД необходимо реализовать 5 курсоров.

Содержание отчета: тема и цель работы; SQL-код 5 курсоров.

Контрольные вопросы

1 Что такое курсор? Какие типы курсоров различают?
2 Что такое полный и результирующий наборы строк?
3 Какие основные операции выделяют при работе с курсором? С помощью каких команд T-SQL реализуются основные операции?

19 Назначение прав доступа пользователям к объектам базы данных средствами T-SQL

Цель: изучить основы управления правами доступа к объектам базы данных в СУБД MS SQL Server.

19.1 Теоретические положения

Вопросы управления правами доступа к объектам БД подробно описаны в конспекте лекций и в [7, 14]. Здесь же рассмотрен SQL-код основных инструкций.

Для предоставления прав доступа используется команда GRANT:

```
GRANT
{ ALL / permissions [..n]}
{ [ (column [..n] )] ON { table / view }
| ON { table / view} [ (column[..n]) ]
| ON {stored_procedure}
TO security_account
[WITH GRANT OPTION]  [AS {group / role}]
```

Для запрещения пользователям доступа к объектам БД используется команда DENY:

```
DENY
{ ALL / permissions [..n]}
{ [ (column [..n] )] ON { table / view }
| ON { table/view } [(column[..n])]
| ON {stored_procedure}
TO security_account [..n]  [CASCADE]
```

Для неявного отклонения доступа к объектам БД используется REVOKE:

```
REVOKE
{ ALL / permissions [..n] }
{ [ (column [..n])] ON { table / view }
| ON { table / view } [(column[..n])]
| ON { stored_procedure }
TO security_account [..n]  [ CASCADE ]  [ AS { role / group} ]
```

Значения параметров команд:

ALL – означает, что пользователю будут предоставлены все возможные разрешения;

Permissions – список доступных операций, которые предоставляются пользователю. Можно предоставлять одновременно несколько разрешений;

Statement – предоставление разрешений на выполнение команд T-SQL;

Security_account – указывается имя объекта системы безопасности, которую необходимо включить в роль. Это могут быть как учетные записи SQL-сервер, так и группы пользователей Windows;

With Grant Option – позволяет пользователю, которому предоставляются права, назначать права на доступ к объекту для других пользователей;

As role / group – позволяет указать участие пользователя в роли, которому предоставляется возможность предоставлять права другим пользователям;

CASCADE – позволяет отзывать права не только у данного пользователя, но и у всех пользователей, которым он предоставил данные права.

Задание

Для разрабатываемой БД необходимо реализовать систему безопасности: создать 5 пользователей и назначить им права доступа к объектам БД.

Содержание отчета: тема и цель работы; SQL-код выполнения задания.

Контрольные вопросы

1 На какие категории можно разделить права в SQL Server?

2 Перечислить стандартные роли сервера и базы данных.

3 Какие команды T-SQL используются для предоставления прав доступа, запрещения и неявного отклонения доступа?

20 Взаимодействие СУБД MS SQL Server с системой программирования MS Visual Studio.NET

Цель: изучить основы создания приложения Windows Forms в среде Visual Studio .Net для ввода, изменения и удаления данных в базе данных.

20.1 Теоретические положения

Вопросы, изучаемые в данной теме, подробно описаны в конспекте лекций и в [5, 12]. После создания приложения Windows Forms нужно установить соединение с БД. Для этого выбирается пункт меню *Tools/ConnectionDatabase*. Появляется окно «*Add Connection*». В пункте «*Server name*» задается имя сервера. В пункте «*Select or enter database name*» – имя БД. Для проверки правильности подключения к БД можно нажать клавишу «*TestConnection*».

Для создания набора данных в меню «Данные» нужно выбрать команду «Показать источники данных», в окне «Источники данных» выбрать «Добавить новый источник данных». На странице «Выбор типа источника данных» нужно выбрать пункт «База данных» и «Набор данных». На странице «Выбор подключения базы данных» выбрать «Новое подключение» для настройки нового подключения к данным. В диалоговом окне «Выбор источника данных» выбрать Microsoft SQL Server Database File, ввести путь к файлу БД. Нажать кнопку «Далее» на странице «Сохранить строку подключения в файле конфигурации приложения». Затем развернуть узел «Таблицы» на странице «Выбор объектов базы данных», выбрать таблицы, добавляемые в проект.

Далее нужно выбрать таблицы в окне «Источники данных» и перетащить их на форму. Для подчиненных таблиц в качестве источника данных выбирается не сама вторая таблица, а связь (Binding Source) между таблицами.

Источник данных нужно открыть в конструкторе набора данных для добавления или изменения объектов, составляющих набор данных.

Чтобы добавить логику обновления в приложение, следует:

– дважды щелкнуть кнопку «Сохранить»  на BindingNavigator, чтобы открыть редактор кода для обработчика событий bindingNavigatorSaveItem_Click;

– заменить код в обработчике событий на вызов методов Update связанных адаптеров таблиц TableAdapter [5].

Задание

Необходимо создать приложение Windows Forms в среде Visual Studio .Net для ввода, изменения и удаления данных в таблицах базы данных.

Содержание отчета: тема и цель работы; код на C# и скриншоты приложения.

Контрольные вопросы

1 Как установить соединение с БД?

2 Как выполняется удаление и обновление записей в Windows-формах?

21 Работа с базами данных с использованием технологии ADO.NET

Цель: изучить основы технологии ADO.NET и компоненты доступа к данным в MS Visual Studio .Net.

21.1 Теоретические положения

ADO.NET – это набор средств Microsoft .NET Framework, позволяющих приложению легко управлять и взаимодействовать со своим файловым или серверным хранилищем данных.

Для работы с различными СУБД подключаются (using) соответствующие пространства имен: для MS Access – System.Data.OleDb; для MS SQL – System.Data.SqlClient.

В объектной модели ADO.NET можно выделить несколько уровней.

Уровень данных. Это, по сути дела, базовый уровень, на котором располагаются сами данные (например, таблицы БД MS SQL Server). На данном уровне обеспечивается физическое хранение информации на магнитных носителях и манипуляция с данными на уровне исходных таблиц (выборка, сортировка, добавление, удаление, обновление и т. п.).

Уровень бизнес-логики. Это набор объектов, определяющих, с какой БД предстоит установить связь и какие действия необходимо будет выполнить с содержащейся в ней информацией. Для установления связи с БД используется объект DataConnection. Для хранения команд, выполняющих какие-либо действия над данными, используется объект DataAdapter. И, наконец, если выполнялся процесс выборки информации из БД, для хранения результатов выборки используется объект DataSet. Объект DataSet представляет собой набор данных, «вырезанных» из таблиц основного хранилища, который может быть передан любой программе-клиенту, способной отобразить эту информацию пользователю, либо выполнить какие-либо манипуляции с полученными данными.

Каждый объект DataAdapter обеспечивает обмен данными между одной таблицей источника данных (базы данных) и одним объектом DataTable в наборе данных DataSet. Если DataSet сдержит несколько таблиц (объектов DataTable), то необходимо иметь и несколько адаптеров данных.

Используя объект DataAdapter, можно читать, добавлять, модифицировать и удалять записи в источнике данных. Чтобы определить, как каждая из этих операций должна произойти, DataAdapter поддерживает следующие свойства: SelectCommand – описание команды, которая обеспечивает выборку нужной информации из базы данных; InsertCommand – описание команды, которая обеспечивает добавление записей в базу данных; UpdateCommand – описание команды, которая обеспечивает обновление записей в базе данных; DeleteCommand – описание команды, которая обеспечивает удаление записей из базы данных. Каждая из команд реализована в виде SQL-запроса или хранимой процедуры. Эти свойства являются самостоятельными объектами и относятся к элементам класса OleDbCommand или SqlCommand. Данные объекты поддерживают свойство CommandText, содержащее описание SQL-запроса или хранимой процедуры.

Уровень приложения. Это набор объектов, позволяющих хранить и отображать данные на компьютере конечного пользователя. Для хранения информации используется объект DataSet, а для отображения данных имеется набор элементов управления (DataGrid, TextBox, ComboBox, Label и т. д.).

Обмен данными между приложениями и уровнем бизнес-логики происходит с использованием формата XML, а средой передачи данных служат либо локальная сеть (Интранет), либо глобальная сеть (Интернет).

В ADO.NET для манипуляции с данными могут использоваться команды, реализованные в виде SQL-запросов или хранимых процедур (DataCommand).

Например, если нужно получить некий набор информации БД, формируется команда SELECT или вызывается хранимая процедура по ее имени.

Когда требуется получить набор строк из БД, необходимо выполнить следующую последовательность действий:

– открыть соединение (connection) с БД;

– вызвать на исполнение метод или команду, указав ей в качестве параметра текст SQL-запроса или имя хранимой процедуры;

– закрыть соединение с базой данных.

Связь с базой данных остается активной только на достаточно короткий срок – на период выполнения запроса или хранимой процедуры.

Когда команда вызывается на исполнение, она возвращает либо данные, либо код ошибки. Если в команде содержался SQL-запрос на выборку SELECT, то команда может вернуть набор данных. Можно выбрать из БД только определенные строки и колонки, используя объект DataReader, который работает достаточно быстро, т. к. использует курсоры read-only, forward-only.

Более подробно применение технологии ADO.NET описано в конспекте лекций и в [5, 12].

Задание

Необходимо, используя стандартные компоненты Visual Studio.Net, подключиться к базе данных.

Содержание отчета: тема и цель работы; код C# подключения.

Контрольные вопросы

1 Как осуществляется взаимодействие с БД через объект DataSet?

2 Для чего предназначены объекты SqlConnection, OleDbConnection, DataAdapter DataReader?

22 Технология ADO.NET Entity Framework

Цель: изучить основы работы с технологией ADO.NET Entity Framework.

22.1 Теоретические положения

Entity Framework (EF) – это компонент API-интерфейса ADO.NET, предоставляющий возможность взаимодействия с реляционными БД через объектную модель, которая отображается на бизнес-объекты приложения. В основе EF лежит сущностная модель данных Entity Data Model (EDM). В модели EDM определяются типы сущностей, отношения и контейнеры, а разработчик взаимодействует со всем этим посредством кода. Платформа EF использует модель EDM через XML, в котором определяется концептуальная модель приложения. К сущностям можно выполнять запросы с использованием LINQ. Исполняю-

щая среда EF транслирует запросы LINQ в соответствующие запросы SQL [12].

Вначале выбирается один из трех способов создания EDM:

– Database First – разработка начинается с существующей БД и используется мастер для генерации на ее основе концептуальной модели;

– Model First – разработка начинается с нуля. С помощью EDM Designer проектируется EDM, а затем на ее основе генерируется схема БД;

– Code First – вначале разрабатываются классы, которые описывают конкретную концептуальную модель (визуальная модель здесь отсутствует).

Рассмотрим принцип Model First, т. е. разработка начнется с создания модели данных, из которой впоследствии будет сгенерирована схема данных (таблицы и связи в БД). В консольном приложении добавляется в проект модель данных: Add → New Item... → ADO.NET Entity Data Model, назовем ее MyEFModel.edmx. Выберем вариант создания пустой модели (Empty Model). После этого откроется пустое поле EDM Designer. Если дизайнер не открылся, то следует выбрать модель MyEFModel.edmx в Solution Explorer.

Сущность создается нажатием правой клавиши мыши на диаграмме модели данных и выбором контекстного меню Add → Entity. Задаются свойства, например, Entity Name = User, Entity Set = Users. По умолчанию EDM Designer создает первичный ключ Id целочисленного типа. Добавляем в эту сущность свойства (Properties): Login (Type = String, Max Length = 255), Registered (Type = DateTime). Свойства добавляются в контекстном меню самой сущности (Add → Scalar Property).

Связь создается выбором контекстного меню Add →Association, задаются свойства, например, Association Name = UserGroup; начало связи: Entity = User, Multiplicity = Many; конец связи: Entity = Group, Multiplicity = One. Это соотношение 1:M, т. е. каждый User может состоять только в одной группе, но в каждой группе может быть много пользователей.

Для написания программного кода, работающего с данными, нужно для начала создать саму базу данных. После нажатия правой клавшей мыши на диаграмме и выбора в контекстном меню Generate Database from Model… можно выбрать существующее подключение к БД или создать новое. Создадим новое подключение (кнопка New Connection…). Для этого нужно выбрать сервер БД, способ и параметры аутентификации и имя базы данных (если нужно создать новую, то следует просто ввести имя новой БД в поле Select or enter a database name), нажать OK. В мастере генерации БД установить галочку «Save entity connection settings in App.config as» и нажать Next. После этого во вкладке DDL мастер отобразит DDL-скрипт схемы данных.

После окончания работы мастера будет получено следующее:

1) скрипт схемы данных в файле MyEFModel.edmx.sql;

2) строка подключения MyEFModelContainer в файле App.config (следует открыть его и найти раздел connectionStrings);

3) пустая база данных.

Теперь нужно выполнить DDL-скрипт, чтобы создать все необходимые объекты в БД. Можно выполнить файл скрипта в Query Analizer, а можно сделать это прямо в Visual Studio, открыв скрипт и нажав кнопку Execute SQL в

панели инструментов или нажав комбинацию Ctrl+Shift+E. Studio запросит параметры подключения к БД, а затем выполнит скрипт. Открыв теперь базу в Enterprise Manager, можно увидеть, что там созданы таблицы, отношения, первичные и внешние ключи, индексы для всех первичных и внешних ключей.

Чтобы обратиться к данным в БД, нужно создать экземпляр контейнера модели (класс контейнера называется MyEFModelContainer):

```
MyEFModelContainer cnt =
        new MyEFModelContainer("name = MyEFModelContainer");
```

В качестве аргумента конструктора задают имя строки подключения к БД. Далее можно записать следующий код:

```
Right canWrite = new Right()              // создание объектов прав
{    Description = "Редактировать статьи" };
    Group groupWriters = new Group()     // создание объектов групп
{    Name = "Авторы"};
    groupReaders.Rights.Add(canRead);    // присвоение прав группе
    // создание объекта пользователей и занесение его в нужную группу
    User userIvanov = new User()
    {    Login = "ivanov", Registered = DateTime.Now, Group=groupWriters};
    // добавление объектов в соответствующие коллекции контейнера
    Using (MyEFModelContainer cnt =
    new MyEFModelContainer("name = MyEFModelContainer"))
    {
    cnt.Rights.AddObject(canWrite);
    cnt.Groups.AddObject(groupWriters);
    cnt.Users.AddObject(userIvanov);
    cnt.SaveChanges();       //  сохранение всех изменений в БД
    }
```

Задание

Необходимо, используя принцип Model First, создать 3 таблицы базы данных и отношения между ними (1:M, M:M), две группы пользователей.

Содержание отчета: тема и цель работы; код C# выполнения задания.

Контрольные вопросы

1 Что такое EDM и какие существуют способы создания EDM?
2 Каковы основные этапы реализации принципа Model First?
3 Как обновить EDMX-файл при изменении базы данных?

23 Технология LINQ to SQL

Цель: изучить основы использования технологии LINQ to SQL.

23.1 Теоретические положения

Технология LINQ (Language Integrated Query – «запрос, интегрированный в язык») расширяет возможности языка программирования путем включения в него дополнительных средств высокого уровня (запросов LINQ), предназначенных для преобразования различных наборов данных. Входящие в нее средства можно применять для обработки наборов данных самых разных видов: массивов и других локальных коллекций данных, удаленных баз данных, XML-документов, а также любых других источников данных, для которых реализован программный интерфейс LINQ (LINQ API) [5].

Технология LINQ to SQL обеспечивает преобразование запросов LINQ в запросы языка SQL (LINQ to SQL ориентирована на SQL-серверы баз данных (в частности, Microsoft SQL Server)).

Все операции запроса LINQ состоят из трех действий: получение источника данных; создание запроса; выполнение запроса.

Вначале нужно создать *решение LINQ to SQL*, которое содержит ссылки, необходимые для построения и выполнения проекта LINQ to SQL. В среде Visual Studio следует создать консольное приложение. В начало Program.cs нужно добавить следующие директивы: using System.Data.Linq; using System.Data.Linq.Mapping.

Для возможности обработки БД с применением интерфейса LINQ to SQL необходимо предварительно построить *объектную модель* этой базы, определив *классы сущностей* (entity classes), связанные с ее компонентами (сопоставляемые с таблицами БД). Сопоставление осуществляется простым добавлением атрибута TableAttribute. Свойство Name задает имя таблицы в БД.

Для создания класса сущностей вводится следующий код в Program.cs перед объявлением класса Program:

```
[Table(Name = "Customers")]
public class Customer
{
    // назначение классу сущностей Customer свойства CustomerID,
      // представляющего столбец первичного ключа в БД
private string _CustomerID;
[Column(IsPrimaryKey=true, Storage="_CustomerID")]
public string CustomerID
{
            get   { return this._CustomerID; }
             set    { this._CustomerID = value; }
}
// назначение классу сущностей свойства City (столбец в таблице БД)
```

```
private string _City;
[Column(Storage="_City")]
public string City
{
        get  { return this._City;  }
        set   { this._City=value; }
}
}
```

Для установки подключения между основанными на коде структурами данных и самой БД используется объект DataContext. Основным каналом, через который извлекаются объекты из БД и отправляются изменения, является класс DataContext. Также объявляется объект Table<Customer>, который действует как логическая типизированная таблица для запросов к таблице «Customers» в БД. Ниже показан пример указания подключения к БД:

```
// Use a connection string.
DataContext db = new DataContext
   (@"c:\linqtest5\northwnd.mdf");
// Get a typed table to run queries.
Table<Customer> Customers = db.GetTable<Customer>();
```

Далее приведен пример создания запроса для поиска клиентов, находящихся в Москве, из таблицы Customers БД. Код запроса, создаваемый на этом шаге в методе Main после объявления Table<Customer>, только описывает запрос, но не выполняет его.

```
// Attach the log to show generated SQL.
db.Log = Console.Out;
// Query for customers in Moscow.
IQueryable<Customer> custQuery =
from cust in Customers
where cust.City == " Moscow"
select cust;
```

Представлен пример кода для выполнения запроса в Main. После начала итерации foreach выполняется команда SQL для БД и объекты материализуются.

```
foreach (Customer cust in custQuery)
{ Console.WriteLine("ID={0}, City={1}", cust.CustomerID, cust.City); }
// Prevent console window from closing
Console.ReadLine();
```

Следующий код создает класс сущностей Order, включающий код, который указывает, что свойство Order.Customer связано как внешний ключ со свойством Customer.CustomerID.

```
[Table(Name = "Orders")]
public class Order
{
private int _OrderID = 0;
private string _CustomerID;
private EntityRef<Customer> _Customer;
public Order() { this._Customer = new EntityRef<Customer>(); }

[Column(Storage = "_OrderID", DbType = "Int NOT NULL IDENTITY",
IsPrimaryKey = true, IsDbGenerated = true)]
public int OrderID
{
get { return this._OrderID; }
// No need to specify a setter because IsDBGenerated is true.
}

[Column(Storage = "_CustomerID", DbType = "NChar(5)")]
public string CustomerID
{
get { return this._CustomerID; }
set { this._CustomerID = value; }
}

[Association(Storage = "_Customer", ThisKey = "CustomerID")]
public Customer Customer
{
get { return this._Customer.Entity; }
set { this._Customer.Entity = value; }
}
}
```


Задание

Необходимо написать 5 запросов к БД на основе технологии LINQ to SQL.
Содержание отчета: тема и цель работы; код C# выполнения задания.

Контрольные вопросы

1 Охарактеризовать назначение технологии LINQ и LINQ to SQL.
2 Для чего предназначен интерфейс IQueryable<T>?
3 Что такое отложенное выполнение запроса LINQ?

Список литературы

1 ГОСТ 34.602–89 «Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы» [Электронный ресурс]. – Москва: Стандартинформ, 2009 (© АО «Кодекс», 2018). – Режим доступа: http://docs.cntd.ru/document/gost-34-602-89. – Дата доступа: 29.03.2018.

2 **Агальцов, В. П.** Базы данных: учебник для вузов: в 2 кн. Кн. 1. Локальные базы данных / В. П. Агальцов. – 2-е изд., перераб. и доп. – Москва: Форум: Инфра-М, 2012. – 352 с.

3 **Агальцов, В. П.** Базы данных. В 2-х т. Т. 2. Распределенные и удаленные базы данных : учебник / В. П. Агальцов. – Москва: Форум: Инфра-М, 2015. – 272 с. : ил.

4 **Бен-Ган, И.** Microsoft SQL Server 2012. Создание запросов: учебный курс Microsoft: пер. с англ. Н. Сержантовой / И. Бен-Ган, Д. Сарка, Р. Талмейдж. – Москва: Русская редакция, 2015. – 720 с.: ил.

5 **Ватсон, Б.** C# 4.0 на примерах / Б. Ватсон. – Санкт-Петербург: БХВ-Петербург, 2011. – 608 с.: ил.

6 **Гурвиц, Г. А.** Microsoft Access 2010. Разработка приложений на реальном примере / Г. А. Гурвиц. – Санкт-Петербург: БХВ-Петербург, 2010. – 496 с.

7 **Кузин, А. В.** Базы данных: учебное пособие для студентов высших учебных заведений / А. В. Кузин, С. В. Левонисова. – 6-е изд., стер. – Москва: Издательский центр «Академия», 2016. – 320 с.

8 **Кузнецов, С. Д.** Базы данных: учебник / С. Д. Кузнецов. – Москва: Академия, 2012. – 496 с.

9 **Куликов, С. С.** Работа с MySQL, MS SQL Server и Oracle в примерах: практическое пособие / С. С. Куликов. – Минск: БОФФ, 2016. – 556 с.

10 **Олейник, П. П.** Корпоративные информационные системы: учебник / П. П. Олейник. – Санкт-Петербург: Питер, 2012. – 176 с.

11 РД IDEF0-2000. Методология функционального моделирования IDEF0. Руководящий документ. – Москва: Изд-во стандартов, 2000. – 75 с.

12 **Рихтер, Д.** CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#: пер. с англ. Е. Матвеев / Д. Рихтер. – 4-е изд. – Санкт-Петербург: Питер, 2016. – 896 с.: ил.

13 **Советов, Б. Я.** Базы данных: теория и практика: учебник для бакалавров. – 2-е изд. – Москва: Юрайт, 2012. – 463 с.

14 **Тернстрем, Т.** Microsoft SQL Server 2008. Разработка баз данных. Учебный курс Microsoft: пер. с англ. / Т. Тернстрем, Э. Вебер, М. Хотек совместно с компанией GrandMasters. – Москва: Русская редакция, 2010. – 496 с.

15 **Шустова, Л. И.** Базы данных: учебник / Л. И. Шустова, О. В. Тараканов. – Москва: Инфра-М, 2017. – 304 с. + Доп. материалы [Электронный ресурс. – Режим доступа: http://www.znanium.com]. – (Высшее образование: Бакалавриат). www.dx.doi.org/10.12737/11549. – Дата доступа: 29.04.2018.