И.И. Семенова

РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНЫХ ПРИЛОЖЕНИЙ

В MICROSOFT SQL SERVER 2000

И BORLAND DELPHI 7

Учебно-методическое пособие

Омск · 2007

Федеральное агентство по образованию
Сибирская государственная автомобильно-дорожная академия (СибАДИ)

И.И. Семенова

РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНЫХ ПРИЛОЖЕНИЙ

В MICROSOFT SQL SERVER 2000

И BORLAND DELPHI 7

Учебно-методическое пособие

Омск
Издательство СибАДИ
2007

УДК 681.3.06
ББК 32.973.2
С 30

Рецензенты:
канд. техн. наук, доцент В.Г. Осипов
(Омский государственный технический университет)

Работа одобрена редакционно-издательским советом академии в качестве учебно-методического пособия для специальностей 230102 «Автоматизированные системы обработки информации и управления», 080801 «Прикладная информатика в экономике», 090105 «Комплексное обеспечение информационной безопасности автоматизированных систем».

Семенова И.И.
С 30 **Разработка клиент-серверных приложений в** Microsoft SQL Server 2000 **и** Borland Delphi 7: Учебно-методическое пособие.– Омск: Изд-во СибАДИ, 2007.– 61 с.

ISBN 978-5-93204-329-5

Основной целью создания данного учебно-методического пособия стала необходимость закрепления навыков работы в одной из современных СУБД с целью создания приложений для различных предметных областей у студентов высших учебных заведений, изучающих дисциплину “Системы управления базами данных”.

Учебно-методическое пособие по курсу «Системы управления базами данных» предназначено для студентов, обучающихся по специальностям 230102, 080801, 090105.

Табл. 10 Ил. 7 Библиогр.: 5 назв.

ISBN

ОБЩИЕ ПОЛОЖЕНИЯ

В процессе выполнения лабораторных работ по дисциплине «Системы управления базами данных» студенты должны выработать навыки физического проектирования баз данных, а также навыки разработки клиентских приложений для работы с базами данных, расположенных на сервере.

Изучение этих вопросов предусмотрено действующим государственным образовательным стандартом и в данном учебно-методическом пособии базируется на решении задач, актуальных для студентов специальности «Автоматизированные системы обработки информации и управления» (АСОИУ), «Прикладная информатика в экономике» (ПИЭ), «Комплексное обеспечение информационной безопасности автоматизированных систем» (КОИБАС).

В серии лабораторных работ используются Microsoft SQL Server 2000, Borland Delphi 7. Важной составной частью работ является освоение SQL стандарта. Для того, чтобы приступить к выполнению этих работ, студенту уже нужно иметь минимальные навыки обращения с используемыми в них программными средствами. Эти необходимые навыки даются студентам на установочных занятиях, в начале семестра, вместе с выдачей заданий и электронных материалов.

Результаты выполнения работ рекомендуется сохранять в личных папках, так как лабораторные работы взаимоувязаны.

Лабораторная работа №1

СОЗДАНИЕ БАЗ ДАННЫХ (БД) В MICROSOFT SQL SERVER

Цель работы – с помощью операторов языка Transact SQL научиться создавать базы данных и совокупность связанных таблиц, принадлежащих указанной базе данных.

Содержание работы:

1. Познакомиться с набором утилит, входящих в состав MS SQL Server 2000.
2. Познакомиться с работой утилит SQL Server Enterprise MANAGER и Query Analyzer.
3. Создать с помощью приведенных операторов пример базы данных «Книжное дело».
4. По выданным вариантам создать персональную базу данных с набором связанных таблиц.

Пояснения к выполнению работы

В качестве примера базы данных, которая будет создана программно с помощью операторов языка Transact SQL, выберем БД «Книжное дело» (рис. 1.1). Структура таблиц данной БД представлена в табл. 1.1-1.5.

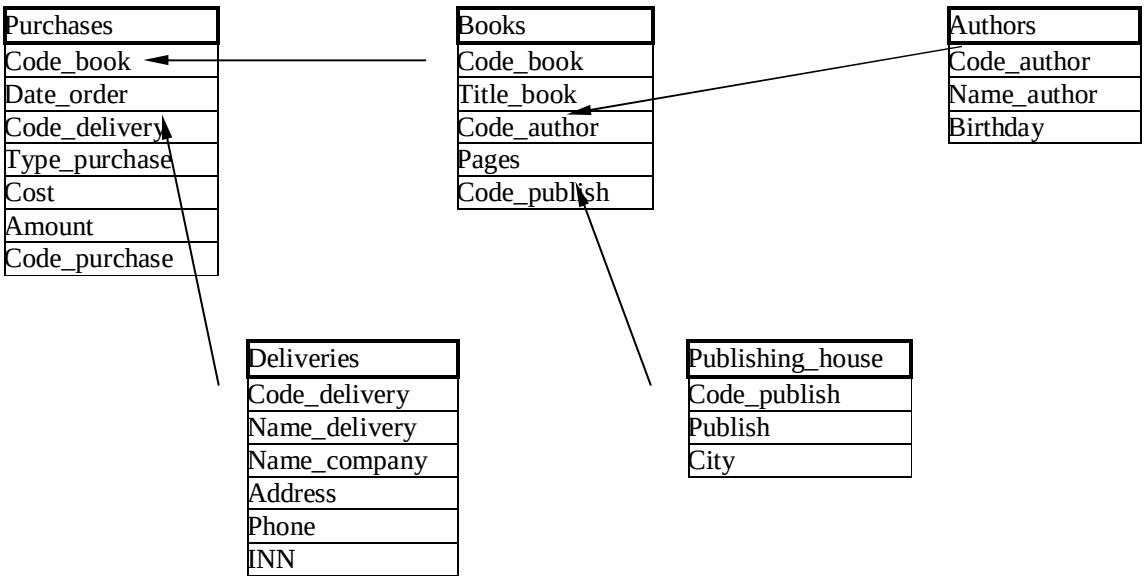

Рис. 1.1. Фрагмент базы данных «Книжное дело»

Таблица 1.1

Покупки (название таблицы Purchases)

Название поля	Тип поля	Описание поля
Code_book	Int	Код закупаемой книги
Date_order	DateTime	Дата заказа книги
Code_delivery	Int	Код поставщика
Type_purchase	Bit	Тип закупки (опт/ розница)
Cost	Money	Стоимость единицы товара
Amount	Int	Количество экземпляров
Code_purchase	Int	Код покупки

Таблица 1.2

Справочник книг (название таблицы Books)

Название поля	Тип поля	Описание поля
Code_book	Int	Код книги
Title_book	Char	Название книги
Code_author	Int	Код автора
Pages	Int	Количество страниц
Code_publish	Int	Код издательства

Таблица 1.3

Справочник авторов (название таблицы Authors)

Название поля	Тип поля	Описание поля
Code_author	Int	Код автора
Name_author	Char	Фамилия, имя, отчество автора
Birthday	DateTime	Дата рождения

Таблица 1.4

Справочник поставщиков (название таблицы Deliveries)

Название поля	Тип поля	Описание поля
Code_delivery	Int	Код поставщика
Name_delivery	Char	Фамилия, и., о. ответственного лица
Name_company	Char	Название компании-поставщика
Address	Char	Юридический адрес
Phone	Numeric	Телефон контактный
INN	Char	ИНН

Таблица 1.5

Справочник издательств (название таблицы Publishing_house)

Название поля	Тип поля	Описание поля
Code_publish	Int	Код издательства
Publish	Char	Издательство
City	Char	Город

Запустить SQL Server Enterprise MANAGER, проверить включение сервера.

Запустить Query Analyzer, подключиться к серверу с помощью пользователя sa и пароля, выданного преподавателем.

Создать новую базу данных с названием DB_Books в утилите Query Analyzer с помощью команды:

CREATE DATABASE DB_BOOKS

Для выполнения команды нажать F5.

Открыть утилиту SQL Server Enterprise MANAGER. Проверить наличие БД DB_Books, если ее не видите в разделе DataBases, то нажмите F5 для обновления.

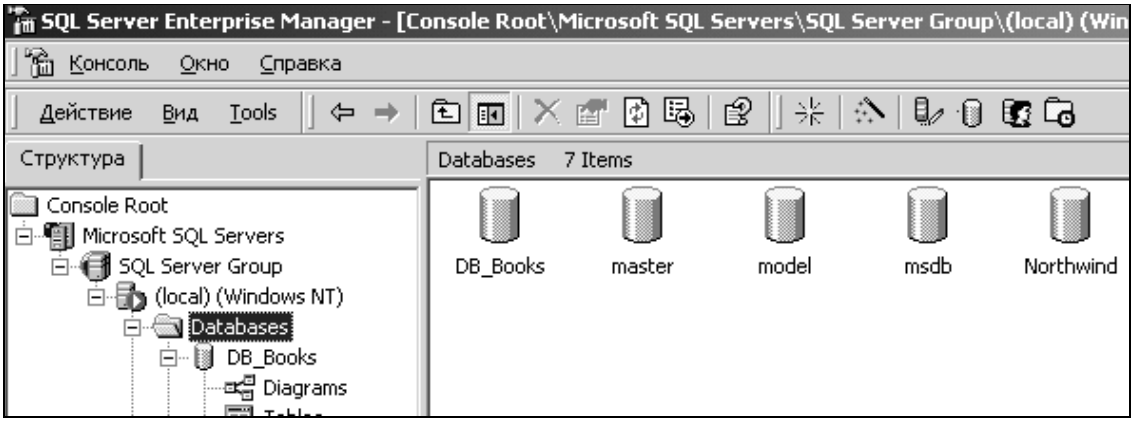

Рис. 1.2. Результат создания БД

Создать в ней перечисленные таблицы либо с помощью мастера таблиц, либо через Query Analyzer с помощью следующих команд:

use DB_BOOKS

CREATE TABLE Authors(Code_author INT PRIMARY KEY, name_author CHAR(30), Birthday DATETIME)

CREATE TABLE Publishing_house(Code_publish INT PRIMARY KEY, Publish CHAR(30), City CHAR(20))

CREATE TABLE Books(Code_book INT PRIMARY KEY, Title_book CHAR(40), Code_author INT FOREIGN KEY REFERENCES Authors(Code_author), Pages INT, Code_publish INT FOREIGN KEY REFERENCES Publishing_house(Code_publish))

CREATE TABLE Deliveries(Code_delivery INT PRIMARY KEY, Name_delivery CHAR(30), Name_company CHAR(20), Address VARCHAR(100), Phone BIGINT, INN CHAR(13))

CREATE TABLE Purchases(Code_purchase INT PRIMARY KEY, Code_book INT FOREIGN KEY REFERENCES Books(Code_book),

Date_order SMALLDATETIME, Code_delivery INT FOREIGN KEY REFERENCES Deliveries(Code_delivery), Type_purchase BIT, Cost FLOAT, Amount INT)

Открыть утилиту SQL Server Enterprise MANAGER. Проверить наличие БД DB_Books.

В разделе диаграмм создать новую диаграмму, в которую добавить из списка пять наших таблиц, проверить связи между таблицами.

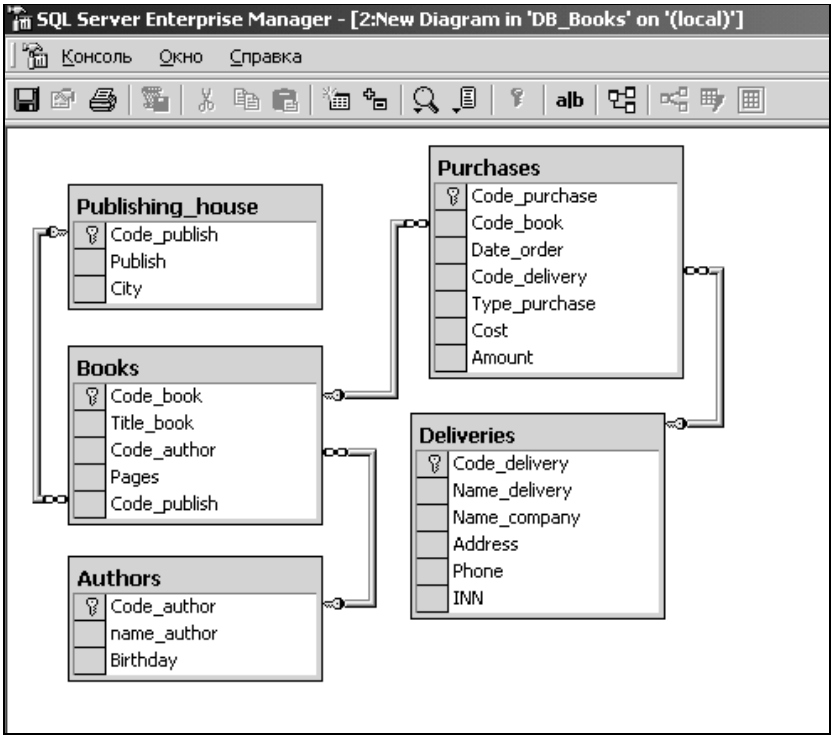

Рис. 1.3. Результат создание диаграммы

Использованные операторы:

PRIMARY KEY – признак создания ключевого поля.

FOREIGN KEY...REFERENCES… – признак создания поля связи с другой таблицей.

CREATE TABLE – команда создания таблицы в текущей БД.

USE – сделать активной конкретную БД.

CREATE DATABASE – команда создания новой БД.

Варианты заданий к лабораторной работе №1

Общие положения

В утилите Query Analyzer создать новую базу данных с помощью оператора Create Database, название БД определить, исходя из предметной

области. Закомментировать оператор (-- – однострочный комментарий, /* */ – многострочный комментарий). Программно сделать активной созданную БД с помощью оператора Use. Создать перечисленные таблицы с помощью операторов Create table, причем самостоятельно определить типы таблиц (родительская или подчиненная), типы полей и их размеры, найти поля типа Primary key и Foreign key. Сохранить файл программы с названием **ФамилияСтудента_ЛАб_1_№варианта**. В SQL Server Enterprise MANAGER в разделе диаграмм созданной БД сгенерировать новую диаграмму, проверить связи между таблицами.

Вариант 1. БД «Учет выданных подарков несовершеннолетним детям сотрудников предприятия»

Код сотрудника
Фамилия
Имя
Отчество
Должность
Подразделение
Дата приема на работу

Код сотрудника
Имя ребенка
Дата рождения
Код ребенка

Код ребенка
Стоимость подарка
Дата выдачи подарка
Код выдачи

Вариант 2. БД «Учет выполненных ремонтных работ»

Код прибора в ремонте
Название прибора
Тип прибора
Дата производства

Код прибора
Код мастера
ФИО владельца прибора
Дата приема в ремонт
Вид поломки
Стоимость ремонта
Код ремонта

Код мастера
Фамилия мастера
Имя мастера
Отчество мастера
Разряд мастера
Дата приема на работу

Вариант 3. БД «Продажа цветов»

Код цветка
Название цветка
Сорт цветка
Средняя высота
Тип листа
Цветущий
Дополнительные сведения

Код цветка
Дата продажи
Цена продажи
Код продавца
Код продажи

Код продавца
Фамилия
Имя
Отчество
Разряд
Оклад
Дата приема на работу

Вариант 4. БД «Поступление лекарственных средств»

Код лекарства
Название лекарства
Показания к применению
Единица измерения
Количество в упаковке
Название производителя

Код лекарства
Код поставщика
Дата поставки
Цена за единицу
Количество
Код поступления

Код поставщика
Сокращенное название
Полное название
Юридический адрес
Телефон
ФИО руководителя

Вариант 5. БД «Списание оборудования»

Код оборудования
Название оборудования
Тип оборудования
Дата поступления
ФИО ответственного
Место установки

Код оборудования
Причина списания
Дата списания
Код сотрудника
Код списания

Код сотрудника
Фамилия
Имя
Отчество
Должность
Подразделение
Дата приема на работу

Вариант 6. БД «Поваренная книга»

Код блюда
Тип блюда
Вес блюда
Порядок приготовления
Количество калорий
Количество углеводов

Код блюда
Код продукта
Объем продукта

Код продукта
Название продукта
Ед измерения

Вариант 7. БД «Регистрация входящей документации»

Код регистратора
Фамилия
Имя
Отчество
Должность
Дата приема на работу

Код документа
Номер документа
Дата регистрации
Краткое содержание документа
Тип документа
Код организации-отправителя
Код регистратора

Код организации-отправителя
Сокращенное название
Полное название
Юридический адрес
Телефон
ФИО руководителя

Вариант 9. БД «Увольнение сотрудника»

Код сотрудника
Фамилия
Имя
Отчество
Должность
Подразделение
Дата приема на работу

Код документа
Номер документа
Дата регистрации
Дата увольнения
Код статьи увольнения
Код сотрудника
Денежная компенсация

Код статьи увольнения
Название статьи увольнения
Причина увольнения
Номер статьи увольнения
Номер пункта/ подпункта увольнения

Вариант 9. БД «Приказ на отпуск»

Код сотрудника
Фамилия
Имя
Отчество
Должность
Подразделение
Дата приема на работу

Код документа
Номер документа
Дата регистрации
Дата начала отпуска
Дата окончания отпуска
Код сотрудника
Код отпуска

Код отпуска
Тип отпуска
Оплата отпуска
Льготы по опуску

Вариант 10. БД «Регистрация выходящей документации»

Код отправителя	Код документа	Код организации-получателя
Фамилия	Номер документа	Сокращенное название
Имя	Дата регистрации	Полное название
Отчество	Краткое содержание документа	Юридический адрес
Должность	Тип документа	Телефон
Дата приема на работу	Код организации-получателя	ФИО руководителя
	Код отправителя	

Вариант 11. БД «Назначение на должность»

Код сотрудника	Код документа	Код должности
Фамилия	Номер документа	Название должности
Имя	Дата регистрации	Льготы по должности
Отчество	Дата назначения	Требования к квалификации
Дата приема на работу	Код сотрудника	
Дата рождения	Код должности	
Пол		

Вариант 12. БД «Выдача оборудования в прокат»

Код клиента	Код выдачи	Код оборудования
Фамилия	Номер документа	Название оборудования
Имя	Дата начала проката	Тип оборудования
Отчество	Дата окончания проката	Дата поступления в прокат
Адрес	Код оборудования	
Телефон	Код клиента	
Серия и номер паспорта	Стоимость	

Вариант 13. БД «Списание оборудования из проката»

Код оборудования	Код оборудования	Код сотрудника
Название оборудования	Причина списания	Фамилия
Тип оборудования	Дата списания	Имя
Дата поступления в прокат	Код сотрудника	Отчество
	Номер документа	Должность
	Дата регистрации	Дата приема на работу
	Код списания	

Вариант 14. БД «Прием цветов в магазин»

Код цветка	Код цветка	Код поставщика
Название цветка	Дата поступления	Сокращенное название
Сорт цветка	Цена за единицу	Полное название
Средняя высота	Код поставщика	Юридический адрес
Тип листа	Код поступления	Телефон
Цветущий	Количество	ФИО руководителя
Дополнительные сведения		

Вариант 15. БД «Регистрация клиентов гостиницы»

Код номера
Тип номера
Перечень удобств
Цена за сутки

Код регистрации
Код номера
Дата заезда
Дата выезда
Стоимость
Код клиента

Код клиента
Фамилия
Имя
Отчество
Адрес
Телефон
Серия и номер паспорта

Вариант 16. БД «Возврат оборудования в службу проката»

Код клиента
Фамилия
Имя
Отчество
Адрес
Телефон
Серия и номер паспорта

Код возврата
Номер документа
Дата возврата
Состояние оборудования
Код оборудования
Код клиента
Штраф

Код оборудования
Название оборудования
Тип оборудования
Дата поступления в прокат

Вариант 17. БД «Учет материальных ценностей на предприятии»

Код ценности
Название ценности
Тип ценности
Закупочная стоимость
Срок гарантии
Дата начала гарантии

Код постановки на учет
Код ценности
Код материально ответственного
Дата постановки на учет
Место нахождения ценности

Код материально ответственного
Фамилия
Имя
Отчество
Должность
Дата приема на работу
Подразделение

Вариант 18. БД «Состав ремонтных работ»

Код ремонтной работы
Код этапа работы
Название этапа работы
Стоимость этапа

Код ремонтной работы
Код мастера
Стоимость ремонта
Количество дней ремонта
Название ремонтной работы

Код мастера
Фамилия мастера
Имя мастера
Отчество мастера
Разряд мастера
Дата приема на работу

Вариант 19. БД «Продажа лекарственных средств»

Код лекарства
Название лекарства
Показания к применению
Единица измерения
Количество в упаковке
Название производителя

Номер чека
Цена за единицу
Количество
Код лекарства
Код записи в чеках

Номер чека
Дата продажи
Сумма
ФИО кассира

Вариант 20. БД «Учет исполнения по входящей документации»

Код исполнителя
Фамилия
Имя
Отчество
Должность
Подразделение
Дата приема на работу

Код документа
Дата назначения на исполнения
Срок выполнения в днях
Тип результата
Код исполнителя
Факт исполнения

Код документа
Номер документа
Дата регистрации
Краткое содержание документа
Тип документа
Организация-отправитель
Код исполнителя

Лабораторная работа №2

ИСПОЛЬЗОВАНИЕ ОПЕРАТОРОВ МАНИПУЛИРОВАНИЯ ДАННЫМИ В MICROSOFT SQL SERVER

Цель работы – научиться использовать операторы манипулирования данными Select, Insert, Update, Delete.

Содержание работы:

1. Создать с помощью приведенных операторов пример базы данных «Книжное дело», описанный в предыдущей лабораторной работе (если БД отсутствует на сервере).
2. С помощью операторов Insert создать программу в Query Analyzer для заполнения таблиц данными (по 3-5 записей).
3. С помощью оператора Select по заданиям выполнить запросы к БД.

Пояснения к выполнению работы

Вся теория по данной работе представлена в конспекте лекций. Также при необходимости можно воспользоваться справочными материалами MS SQL Server 2000, запустив утилиту Book OnLine.

Варианты заданий к лабораторной работе №2

Общие положения

Создать новую базу данных с названием DB_Exam с помощью оператора Create Database, создать в ней перечисленные таблицы с помощью операторов Create table (если БД отсутствует на сервере). Сохранить файл программы с названием **ФамилияСтудента_ЛАб_1_B_Exam**. В утилите Query Analyzer создать отдельные программы по каждому запросу, которые сохранять на диске с названием: **ФамилияСтудента_ЛАб_2_№_задания**. В сами программы копировать текст задания в виде комментария. Для проверки работы операторов SELECT предварительно создайте программу, которая с помощью операторов INSERT заполнит

все таблицы несколькими записями, сохраните программы с названием **ФамилияСтудента_ЛАб_2_Insert.**

Список вариантов заданий

Вариант	Список номеров упражнений												
1	1	6	11	16	21	26	31	36	41	46	51	56	61
2	2	7	12	17	22	27	32	37	42	47	52	57	62
3	3	8	13	18	23	28	33	38	43	48	53	58	63
4	4	9	14	19	24	29	34	39	44	49	54	59	64
5	5	10	15	20	25	30	35	40	45	50	55	60	65
6	6	11	16	21	26	31	36	41	46	51	56	61	1
7	7	12	17	22	27	32	37	42	47	52	57	62	2
8	8	13	18	23	28	33	38	43	48	53	58	63	3
9	9	14	19	24	29	34	39	44	49	54	59	64	4
10	10	15	20	25	30	35	40	45	50	55	60	65	5
11	2	6	12	16	22	26	32	36	42	46	52	56	62
12	1	5	11	15	21	25	31	35	41	45	51	55	61
13	3	7	13	17	23	27	33	37	43	47	53	57	63
14	4	8	14	18	24	28	34	38	44	48	54	58	64
15	5	9	15	19	25	29	35	39	45	49	55	59	65
16	6	13	26	33	36	43	46	53	56	63	1	11	17
17	7	18	27	28	37	38	47	48	57	58	2	4	10
18	8	9	18	19	28	29	38	39	48	49	58	59	60
19	9	14	29	24	39	34	49	44	59	54	61	65	3
20	10	12	20	22	30	32	40	42	50	52	60	62	8

Сортировка

1. Выбрать все сведения о книгах из таблицы Books и отсортировать результат по коду книги (поле Code_book).
2. Выбрать из таблицы Books коды книг, названия и количество страниц (поля Code_book, Title_book и Pages), отсортировать результат по названиям книг (поле Title_book по возрастанию) и по полю Pages (по убыванию).
3. Выбрать из таблицы Deliveries список поставщиков (поля Name_delivery, Phone и INN), отсортировать результат по полю INN (по убыванию).

Изменение порядка следования полей

4. Выбрать все поля из таблицы Deliveries таким образом, чтобы в результате порядок столбцов был следующим: Name_delivery, INN, Phone, Address, Code_delivery.

5. Выбрать все поля из таблицы Publishing_house таким образом, чтобы в результате порядок столбцов был следующим: Publish, City, Code_publish.

Выбор некоторых полей из двух таблиц

6. Выбрать из таблицы Books названия книг и количество страниц (поля Title_book и Pages), а из таблицы Authors выбрать имя соответствующего автора книги (поле Name_ author).

7. Выбрать из таблицы Books названия книг и количество страниц (поля Title_book и Pages), а из таблицы Deliveries выбрать имя соответствующего поставщика книги (поле Name_delivery).

8. Выбрать из таблицы Books названия книг и количество страниц (поля Title_book и Pages), а из таблицы Publishing_house выбрать название соответствующего издательства и места издания (поля Publish и City).

Условие неточного совпадения

9. Выбрать из справочника поставщиков (таблица Deliveries) названия компаний, телефоны и ИНН (поля Name_company, Phone и INN), у которых название компании (поле Name_company) начинается с ‘ОАО’.

10. Выбрать из таблицы Books названия книг и количество страниц (поля Title_book и Pages), а из таблицы Authors выбрать имя соответствующего автора книг (поле Name_ author), у которых название книги начинается со слова ‘Мемуары’.

11. Выбрать из таблицы Authors фамилии, имена, отчества авторов (поле Name_ author), значения которых начинаются с ‘Иванов’.

Точное несовпадение значений одного из полей

12. Вывести список названий издательств (поле Publish) из таблицы Publishing_house, которые не находятся в городе ‘Москва’ (условие по полю City).

13. Вывести список названий книг (поле Title_book) из таблицы Books, которые выпущены любыми издательствами, кроме издательства ‘Питер-Софт’ (поле Publish из таблицы Publishing_house).

Выбор записей по диапазону значений (Between)

14. Вывести фамилии, имена, отчества авторов (поле Name_author) из таблицы Authors, у которых дата рождения (поле Birthday) находится в диапазоне 01.01.1840 – 01.06.1860.

15. Вывести список названий книг (поле Title_book из таблицы Books) и количество экземпляров (поле Amount из таблицы Purchases), которые были закуплены в период с 12.03.2003 по 15.06.2003 (условие по полю Date_order из таблицы Purchases).

16. Вывести список названий книг (поле Title_book) и количество страниц (поле Pages) из таблицы Books, у которых объем в страницах укладывается в диапазон 200 – 300 (условие по полю Pages).

17. Вывести список фамилий, имен, отчеств авторов (поле Name_author) из таблицы Authors, у которых фамилия начинается на одну из букв диапазона ‘В’ – ‘Г’ (условие по полю Name_author).

Выбор записей по диапазону значений (In)

18. Вывести список названий книг (поле Title_book из таблицы Books) и количество (поле Amount из таблицы Purchases), которые были поставлены поставщиками с кодами 3, 7, 9, 11 (условие по полю Code_delivery из таблицы Purchases).

19. Вывести список названий книг (поле Title_book) из таблицы Books, которые выпущены следующими издательствами: ‘Питер-Софт’, ‘Альфа’, ‘Наука’ (условие по полю Publish из таблицы Publishing_house).

20. Вывести список названий книг (поле Title_book) из таблицы Books, которые написаны следующими авторами: ‘Толстой Л.Н.’, ‘Достоевский Ф.М.’, ‘Пушкин А.С.’ (условие по полю Name_author из таблицы Authors).

Выбор записей с использованием Like

21. Вывести список авторов (поле Name_author) из таблицы Authors, которые начинаются на букву ‘К’.

22. Вывести названия издательств (поле Publish) из таблицы Publishing_house, которые содержат в названии сочетание ‘софт’.

23. Выбрать названия компаний (поле Name_company) из таблицы Deliveries, у которых значение оканчивается на ‘ский’.

Выбор записей по нескольким условиям

24. Выбрать коды поставщиков (поле Code_delivery), даты заказов (поле Date_order) и названия книг (поле Title_book), если количество книг (поле Amount) в заказе больше 100 или цена (поле Cost) за книгу находится в диапазоне от 200 до 500.

25. Выбрать коды авторов (поле Code_author), имена авторов (поле Name_author), названия соответствующих книг (поле Title_book), если код издательства (поле Code_Publish) находится в диапазоне от 10 до 25 и количество страниц (поле Pages) в книге больше 120.

26. Вывести список издательств (поле Publish) из таблицы Publishing_house, в которых выпущены книги, названия которых (поле Title_book) начинаются со слова ‘Труды’ и город издания (поле City) – ‘Новосибирск’.

Многотабличные запросы (выборка из двух таблиц, выборка из трех таблиц с использованием JOIN)

27. Вывести список названий компаний-поставщиков (поле Name_company) и названия книг (поле Title_book), которые они поставили в период с 01.01.2002 по 31.12.2003 (условие по полю Date_order).

28. Вывести список авторов (поле Name_author), книги которых были выпущены в издательстве ‘Мир’ (условие по полю Publish).

29. Вывести список поставщиков (поле Name_company), которые поставляют книги издательства ‘Питер’ (условие по полю Publish).

30. Вывести список авторов (поле Name_author) и названия книг (поле Title_book), которые были поставлены поставщиком ‘ОАО Книготорг’ (условие по полю Name_company).

Вычисления

31. Вывести суммарную стоимость партии одноименных книг (использовать поля Amount и Cost) и название книги (поле Title_book) в каждой поставке.

32. Вывести стоимость одной печатной страницы каждой книги (использовать поля Cost и Pages) и названия соответствующих книг (поле Title_book).

33. Вывести количество лет с момента рождения авторов (использовать поле Birthday) и имена соответствующих авторов (поле Name_author).

Вычисление итоговых значений с использованием агрегатных функций

34. Вывести общую сумму поставок книг (использовать поле Cost), выполненных ‘ЗАО Оптторг’ (условие по полю Name_company).

35. Вывести общее количество всех поставок (использовать любое поле из таблицы Purchases), выполненных в период с 01.01.2003 по 01.02.2003 (условие по полю Date_order).

36. Вывести среднюю стоимость (использовать поле Cost) и среднее количество экземпляров книг (использовать поле Amount) в одной поставке, где автором книги является ‘Акунин’ (условие по полю Name_author).

37. Вывести все сведения о поставке (все поля таблицы Purchases), а также название книги (поле Title_book) с минимальной общей стоимостью (использовать поля Cost и Amount).

38. Вывести все сведения о поставке (все поля таблицы Purchases), а также название книги (поле Title_book) с максимальной общей стоимостью (использовать поля Cost и Amount).

Изменение наименований полей

39. Вывести название книги (поле Title_book), суммарную стоимость партии одноименных книг (использовать поля Amount и Cost), поместив в результат в поле с названием Itogo, в поставках за период с 01.01.2002 по 01.06.2002 (условие по полю Date_order).

40. Вывести стоимость одной печатной страницы каждой книги (использовать поля Cost и Pages), поместив результат в поле с названием One_page, и названия соответствующих книг (поле Title_book).

41. Вывести общую сумму поставок книг (использовать поле Cost) и поместить результат в поле с названием Sum_cost, выполненных ‘ОАО Луч’ (условие по полю Name_company).

Использование переменных в условии

42. Вывести список сделок (все поля из таблицы Purchases) за последний месяц (условие с использованием поля Date_order).

43. Вывести список авторов (поле Name_author), возраст которых меньше заданного пользователем (условие с использованием поля Birthday).

44. Вывести список книг (поле Title_book), которых закуплено меньше, чем указано в запросе пользователя (условие с использованием поля Amount).

Использование переменных вместо названий таблиц

45. Вывести список названий компаний-поставщиков (поле Name_company) и названия книг (поле Title_book), которые они поставили.

46. Вывести список авторов (поле Name_author), книги которых были выпущены в издательствах ‘Мир’, ‘Питер Софт’, ‘Наука’ (условие по полю Publish).

47. Вывести список издательств (поле Name_company), книги которых были поставлены по цене 150 руб. (поле Cost).

Выбор результата в курсор

48. Вывести список названий книг (поле Title_book) и количества страниц (поле Pages) в каждой книге и поместить результат в курсор с названием Temp1.

49. Вывести список названий компаний-поставщиков (поле Name_company) и поместить результат в курсор с названием Temp2.

50. Вывести список авторов (поле Name_author) и поместить результат в курсор с названием Temp3.

Использование функций совместно с подзапросом

51. Вывести список книг (поле Title_book), у которых количество страниц (поле Pages) больше среднего количества страниц всех книг в таблице.

52. Вывести список авторов (поле Name_author), возраст которых меньше среднего возраста всех авторов в таблице (условие по полю Birthday).

53. Вывести список книг (поле Title_book), у которых количество страниц (поле Pages) равно минимальному количеству страниц книг, представленных в таблице.

Использование квантора существования в запросах

54. Вывести список издательств (поле Publish), книги которых были приобретены оптом (‘опт’ из поля Type_Purchase).

55. Вывести список авторов (поле Name_author), книг которых нет в таблице Books.

56. Вывести список книг (поле Title_book), которые были поставлены поставщиком ‘ЗАО Квантор’ (условие по полю Name_company).

Оператор обработки данных Update

57. Изменить в таблице Books содержимое поля Pages на 300, если код автора (поле Code_author) =56 и название книги (поле Title_book) =’Мемуары’.

58. Изменить в таблице Deliveries содержимое поля Address на ‘нет сведений’, если значение поля является пустым.

59. Увеличить в таблице Purchases цену (поле Cost) на 20 процентов, если заказы были оформлены в течение последнего месяца (условие по полю Date_order).

Оператор обработки данных Insert

60. Добавить в таблицу Purchases новую запись, причем так, чтобы код покупки (поле Code_purchase) был автоматически увеличен на единицу, а в тип закупки (поле Type_purchase) внести значение ‘опт’.

61. Добавить в таблицу Books новую запись, причем вместо ключевого поля поставить код (поле Code_book), автоматически увеличенный на единицу от максимального кода в таблице, вместо названия книги (поле Title_book) написать ‘Наука. Техника. Инновации’.

62. Добавить в таблицу Publish_house новую запись, причем вместо ключевого поля поставить код (поле Code_publish), автоматически увеличенный на единицу от максимального кода в таблице, вместо названия города – ‘Москва’ (поле City), вместо издательства – ‘Наука’ (поле Publish).

Оператор обработки данных Delete

63. Удалить из таблицы Purchases все записи, у которых количество книг в заказе (поле Amount) = 0.

64. Удалить из таблицы Authors все записи, у которых нет имени автора в поле Name_Author.

65. Удалить из таблицы Deliveries все записи, у которых не указан ИНН (поле INN пустое).

Лабораторная работа №3

ОСВОЕНИЕ ПРОГРАММИРОВАНИЯ С ПОМОЩЬЮ ВСТРОЕННОГО ЯЗЫКА TRANSACT SQL В MICROSOFT SQL SERVER

Цель работы – знакомство с основными приципами программирования в MS SQL Server средствами встроенного языка Transact SQL.

Содержание работы:

1. Знакомство с правилами обозначения синтаксиса команд в справочной системе MS SQL Server (утилита Books Online).
2. Изучение правил написания программ на Transact SQL.
3. Изучение правил построения идентификаторов, правил объявления переменных и их типов.
4. Изучение работы с циклами и ветвлениями.
5. Изучение работы с переменными типа Table и Cursor.
6. Проработка всех примеров, анализ результатов их выполнения в утилите Query Analyzer.
7. Выполнение индивидуальных заданий по вариантам.

Пояснения к выполнению работы

Для освоения программирования используем пример базы данных с названием DB_Books, которая была создана в лабораторной работе №1. При выполнении примеров и заданий обращайте внимание на соответствие названий БД, таблиц и других объектов проекта.

Специальные знаки и простейшие операторы в Transact SQL

Знак	Назначение	Знак	Назначение
*	Знак умножения	" "	В них заключают строковые значения, если SET QUOTED_IDENTIFIER OFF
-	Знак вычитания	‘ ’	В них заключают строковые значения
%	Остаток от деления двух чисел	<>	Не равно

Окончание таблицы

Знак	Назначение	Знак	Назначение
+	Знак сложения или конкатенации (объединение двух строк в одну)	[ ]	Аналог кавычек, в них можно заключать названия идентификаторов, если в их названиях встречаются пробелы
=	Знак равенства или сравнения	!<	Не менее чем
<=	Меньше или равно	!>	Не более чем
>=	Больше или равно	>	Больше
!=	Не равно	<	Меньше
@	Ставится перед именем переменной	.	Разделяет родительские и подчиненные объекты
@@	Указывает на системные функции	/	Знак деления
--	Однострочный комментарий или комментарий с текущей позиции и до конца строки	/* */	Многострочный комментарий

Идентификаторы – это имена объектов, на которые можно ссылаться в программе, написанной на языке Transact SQL. Первый символ может состоять из букв английского алфавита или “_”, “@”, “#”. Остальные дополнительно из цифр и «$».

Имя идентификатора не должно совпадать с зарезервированным словом.

Для ограничителей идентификаторов при установленном параметре

SET QUOTED_IDENTIFIER ON

можно использовать как квадратные скобки, так и одинарные кавычки, а строковые значения только в одинарных кавычках (режим по умолчанию).

Если использовать установленный параметр в режиме

SET QUOTED_IDENTIFIER OFF,

то в качестве ограничителей идентификаторов можно использовать только квадратные скобки, а строковые значения указываются в одинарных или двойных кавычках.

Переменные используются для сохранения промежуточных данных в хранимых процедурах и функциях. Все переменные считаются локальными. Имя переменной должно начинаться с @.

Объявление переменных

Синтаксис в обозначениях MS SQL Server:

DECLARE @имя_переменной1 тип_переменной, …, @имя_переменнойN тип_переменной

Если тип переменной предполагает указание размера, то используется следующий сисntаксис для объявления переменных:
DECLARE @имя_переменной1 тип_переменной (размер), …,
@имя_переменнойN тип_переменной(размер)
Пример:
DECLARE @a INT, @b numeric(10,2)
DECLARE @str CHAR(20)

Присвоение значений переменным и вывод значений на экран

Присвоение с помощью SET – обычное присвоение, синтаксис:
SET @имя_переменной = значение.
Пример:
DECLARE @a INT, @b numeric(10,2)
SET @a = 20
SET @b = (@a+@a)/15
SELECT @b --вывод на экран результата

Присвоение с помощью SELECT – помещение результата запроса в переменную. Если в результате выполнения запроса не будет возвращено ни одной строки, то значение переменной не меняется, т.е. остается старым.
Пример:
DECLARE @a INT
SELECT @a = COUNT(*) FROM Authors

Пример:
DECLARE @str CHAR(30)
SELECT @str = name FROM Authors
В данном примере в переменную поместится последнее значение из результата запроса.

Сочетание ключевых слов SET и SELECT

Пример:
DECLARE @a INT
SET @a = (SELECT COUNT(*) FROM Authors)

Работа с датой и временем

Оператор SET DATEFORMAT dmy | ymd | mdy задает порядок следования компонентов даты.

Пример:
SET DATEFORMAT dmy
DECLARE @d DateTime
SET @d = ’31.01.2005 13:23:15’
SET @d = @d+1
SELECT @d

Создание временной таблицы через переменную типа TABLE

Объявляется через DECLARE с указанием в скобках столбцов таблицы, их типов, размеров, значений по умолчанию, а также индексов типа PRIMARY KEY или UNIQUE.

Пример:
DECLARE @mytable TABLE(id INT, myname CHAR(20) DEFAULT ‘Введите имя’)
INSERT INTO @mytable(id) VALUES (1)
SELECT * FROM @mytable

Пример:
DECLARE @mytable TABLE(id INT, myname CHAR(20) DEFAULT ‘Введите имя’)
INSERT @mytable SELECT Code_publish, City FROM Publishing_house
SELECT * FROM @mytable

Преобразование типов переменных

Функция CAST возвращает значение, преобразованное к указанному типу:
CAST(@переменная или значение AS требуемый_тип_данных)

Пример:
DECLARE @d DateTime, @str char(20)
SET @d = ’31.01.2005 13:23:15’
SET @str = CAST(@d AS Char(20))
SELECT 2str

Функция CONVERT возвращает значение, преобразованное к указанному типу по заданному формату. Изучить дополнительно, по желанию.

Операторские скобки

BEGIN
/* в них нельзя помещать команды, изменяющие структуры объектов БД. Операторские скобки должны содержать хотя бы один оператор. Тре-

буются для конструкций поливариантных ветвлений, условных и циклических конструкций

```
    */
END
```

Условная конструкция IF

Синтаксис:

IF **условие**

Набор операторов1

ELSE

Набор операторов2

Пример:

```
DECLARE @a INT
DECLARE @str CHAR(30)
SET @a = (SELECT COUNT(*) FROM Authors)
IF @a >10
    BEGIN
        SET @str = 'Количество авторов больше 10'
        SELECT @str
    END
ELSE
    BEGIN
        SET @str = 'Количество авторов = ' + str(@a)
        SELECT @str
    END
```

Цикл WHILE

Синтаксис:

WHILE *Условие*

Набор операторов1

BREAK

Набор опреторов2

CONTINUE

Конструкции BREAK и CONTINUE являются необязательными.

Цикл можно принудительно остановить, если в его теле выполнить команду BREAK. Если же нужно начать цикл заново, не дожидаясь выполнения всех команд в теле, необходимо выполнить команду CONTINUE.

Пример:

```
DECLARE @a INT
SET @a = 1
WHILE @a <100
    BEGIN
```

```
        PRINT @a -- вывод на экран значения переменной
        IF (@a>40) AND (@a<50)
            BREAK     --выход и выполнение 1-й команды за циклом
        ELSE
            SET @a = @a+rand()*10
        CONTINUE
    END
PRINT @a
```

Объявление курсора

CURSOR – это набор строк, являющийся результатом выполнения запроса. В один момент времени доступна лишь одна строка (текущая), по курсору можно передвигаться и получать доступ к элементарным данным. При объявлении курсора создается временная копия данных, которая сохраняется в БД tempdb.
Динамический курсор – данные в курсоре могут быть изменены.
Статический курсор – данные в курсоре не меняются.

Стандартный способ объявления курсора, синтаксис в обозначениях MS SQL Server:
DECLARE *cursor_name* [INSENSITIVE] [SCROLL] CURSOR
FOR *select_statement*
[FOR { READ ONLY | UPDATE [OF *column_name* [,...*n*]] }]

Примеры объявления курсоров:

```
DECLARE MyCursor1 CURSOR FOR (select * from Authors)
/*объявили курсор с названием MyCursor1, который содержит всю информацию об авторах, двигаться по нему можно только от первой записи вниз до последней. Курсор является динамическим.*/

DECLARE MyCursor1 INSENSITIVE CURSOR FOR (select * from Authors)
/*объявили курсор с названием MyCursor1, который содержит всю информацию об авторах, двигаться по нему можно только от первой записи вниз до последней. Курсор является статическим.*/

DECLARE MyCursor1 SCROLL CURSOR FOR (select * from Authors)
/*объявили курсор с названием MyCursor1, который содержит всю информацию об авторах, двигаться по нему можно в любом направлении. Курсор является динамическим.*/
```

DECLARE MyCursor1 INSENSITIVE SCROLL CURSOR FOR (select * from Authors)
/*объявили курсор с названием MyCursor1, который содержит всю информацию об авторах, двигаться по нему можно в любом направлении. Курсор является статическим.*/

DECLARE MyCursor1 CURSOR FOR (select * from Authors) FOR READ ONLY
/*объявили курсор с названием MyCursor1, который содержит всю информацию об авторах, двигаться по нему можно только от первой записи вниз до последней. Курсор является динамическим. Данные доступны только для чтения.*/

DECLARE MyCursor1 CURSOR FOR (select * from Authors) FOR UPDATE
/*объявили курсор с названием MyCursor1, который содержит всю информацию об авторах, двигаться по нему можно только от первой записи вниз до последней. Курсор является динамическим. Данные курсора можно менять.*/

Операторы для работы с курсором

Прежде чем обратиться к данным курсора, его нужно после объявления открыть.

Синтаксис оператора OPEN в обозначениях MS SQL Server:
OPEN { { [GLOBAL] *cursor_name* } | *cursor_variable_name* }
Пример:
DECLARE MyCursor1 CURSOR FOR (select * from Authors)
OPEN MyCursor1

После прекращения работы с курсором, его нужно закрыть. Курсор остается доступным для последующего использования в рамках процедуры или триггера, в котором он создан.

Синтаксис оператора CLOSE в обозначениях MS SQL Server:
CLOSE { { [GLOBAL] *cursor_name* } | *cursor_variable_name* }
Пример:
DECLARE MyCursor1 CURSOR FOR (select * from Authors)
OPEN MyCursor1
--здесь операторы работы с курсором
CLOSE MyCursor1

Если курсором больше не будут пользоваться, то его необходимо уничтожить и освободить переменную.

Синтаксис оператора DEALLOCATE в обозначениях MS SQL Server:

DEALLOCATE { { [GLOBAL] *cursor_name* } | *@cursor_variable_name* }

Пример:

```
DECLARE MyCursor1 CURSOR FOR (select * from Authors)
OPEN MyCursor1
--здесь операторы работы с курсором
CLOSE MyCursor1
DEALLOCATE MyCursor1
```

FETCH – оператор движения по записям курсора и извлечения данных текущей записи в указанные переменные.

Синтаксис оператора FETCH в обозначениях MS SQL Server:

```
FETCH
    [ [ NEXT | PRIOR | FIRST | LAST
        | ABSOLUTE { n | @nvar }
        | RELATIVE { n | @nvar }
      ]
      FROM
    ]
{ { [ GLOBAL ] cursor_name } | @cursor_variable_name }
[ INTO @variable_name [ ,...n ] ]
```

Пример:

```
DECLARE MyCursor1 SCROLL CURSOR FOR (select * from Authors)
DECLARE @i bigint, @s char(20), @d smalldatetime
OPEN MyCursor1
FETCH FIRST FROM MyCursor1 INTO @i, @s, @d
PRINT @i
PRINT @s
PRINT @d
CLOSE MyCursor1
DEALLOCATE MyCursor1
```

@@FETCH_STATUS – данная функция определяет признак конца или начала текущего курсора. Функция принимаетодно из следующих значений: 0 – находимся в пределах курсора, не в конце; 1 – попытка выйти за пределы первой записи вверх (в никуда); 2 – попытка выйти за пределы последней записи вниз (в никуда).

Пример:

```
DECLARE MyCursor1 SCROLL CURSOR FOR (select * from Authors)
DECLARE @i bigint, @s char(20), @d smalldatetime
OPEN MyCursor1
```

```
FETCH FIRST FROM MyCursor1 INTO @i, @s, @d
WHILE @@FETCH_STATUS = 0
    BEGIN
        FETCH NEXT FROM MyCursor1 INTO @i, @s, @d
        PRINT @i
        PRINT @s
        PRINT @d
    END
CLOSE MyCursor1
DEALLOCATE MyCursor1
```

Встроенные функции

Встроенные функции, имеющиеся в распоряжении пользователей при работе с SQL, можно условно разделить на следующие группы:

- математические функции;
- строковые функции;
- функции для работы с датой и временем;
- функции конфигурирования;
- функции системы безопасности;
- функции управления метаданными;
- статистические функции.

Использование функций для работы со строковыми переменными

Краткий обзор строковых функций

Название функции	**Действие, выполняемое функцией**
ASCII	Возвращает код ASCII левого символа строки
CHAR	По коду ASCII возвращает символ
CHARINDEX	Определяет порядковый номер символа, с которого начинается вхождение подстроки в строку
DIFFERENCE	Возвращает показатель совпадения строк
LEFT	Возвращает указанное число символов с начала строки
LEN	Возвращает длину строки
LOWER	Переводит все символы строки в нижний регистр
LTRIM	Удаляет пробелы в начале строки
NCHAR	Возвращает по коду символ Unicode
PATINDEX	Выполняет поиск подстроки в строке по указанному шаблону
REPLACE	Заменяет вхождения подстроки на указанное значение
QUOTENAME	Конвертирует строку в формат Unicode
REPLICATE	Выполняет тиражирование строки определенное число раз
REVERSE	Возвращает строку, символы которой записаны в обратном порядке

Окончание таблицы

Название функции	**Действие, выполняемое функцией**
RIGHT	Возвращает указанное число символов с конца строки
RTRIM	Удаляет пробелы в конце строки
SOUNDEX	Возвращает код звучания строки
SPACE	Возвращает указанное число пробелов
STR	Выполняет конвертирование значения числового типа в символьный формат
STUFF	Удаляет указанное число символов, заменяя новой подстрокой
SUBSTRING	Возвращает для строки подстроку указанной длины с заданного символа
UNICODE	Возвращает Unicode-код левого символа строки
UPPER	Переводит все символы строки в верхний регистр

Использование функций для работы с числами

Краткий обзор математических функций

Название функции	**Действие, выполняемое функцией**
ABS	Вычисляет абсолютное значение числа
ACOS	Вычисляет арккосинус
ASIN	Вычисляет арксинус
ATAN	Вычисляет арктангенс
ATN2	Вычисляет арктангенс с учетом квадратов
CEILING	Выполняет округление вверх
COS	Вычисляет косинус угла
COT	Возвращает котангенс угла
DEGREES	Преобразует значение угла из радиан в градусы
EXP	Возвращает экспоненту
FLOOR	Выполняет округление вниз
LOG	Вычисляет натуральный логарифм
LOG10	Вычисляет десятичный логарифм
PI	Возвращает значение "пи"
POWER	Возводит число в степень
RADIANS	Преобразует значение угла из градуса в радианы
RAND	Возвращает случайное число
ROUND	Выполняет округление с заданной точностью
SIGN	Определяет знак числа
SIN	Вычисляет синус угла
SQUARE	Выполняет возведение числа в квадрат
SQRT	Извлекает квадратный корень
TAN	Возвращает тангенс угла

Использование функций для работы с типом дата/время

Краткий обзор основных функций для работы с датой и временем

Название функции	Действие, выполняемое функцией
DATEADD	Добавляет к дате указанное значение дней, месяцев, часов и т.д.
DATEDIFF	Возвращает разницу между указанными частями двух дат
DATENAME	Выделяет из даты указанную часть и возвращает ее в символьном формате
DATEPART	Выделяет из даты указанную часть и возвращает ее в числовом формате
DAY	Возвращает число из указанной даты
GETDATE	Возвращает текущее системное время
ISDATE	Проверяет правильность выражения на соответствие одному из возможных форматов ввода даты
MONTH	Возвращает значение месяца из указанной даты
YEAR	Возвращает значение года из указанной даты
MINUTE	Возвращает значение минут из указанной даты/времени
HOUR	Возвращает значение часов из указанной даты/времени
SECOND	Возвращает значение секунд из указанной даты/времени

Варианты заданий к лабораторной работе №3

Общие сведения

Для получения более подробной информации о работе тех или иных операторов или функций можно запустить утилиту Books Online из состава MS SQL Server и в разделе «Указатель» набрать искомый ключевой элемент.

Для выполнения заданий ориентироваться на вариант и список номеров заданий во второй лабораторной работе.

Специальные знаки и простейшие операторы в Transact SQL

1. Проверить работу описанной установки SET QUOTED_IDENTIFIER.
2. Проверить работу описанной установки SET DATEFIRST.

Объявление переменных

3. Объявить переменную Perem1 типа денежный, а переменную Perem2 типа число с целой частью равной 8 и дробной частью равной 2.
4. Объявить переменную Perem1 типа строка длиной 100, а переменную Perem2 типа длинное целое.
5. Объявить переменную Perem1 типа динамическая строка с максимальной длиной 1000, а переменную Perem2 типа целое число.

6. Объявить переменную Perem1 типа строка длиной 30, а переменную Perem2 типа число с целой частью равной 10 и дробной частью равной 3.

7. Объявить переменную Perem1 типа дата/ время, а переменную Perem2 типа число в диапазоне от 0 до 255.

Присвоение значений переменным и вывод значений на экран

8. Подсчитать среднюю цену закупленных книг (с помощью запроса SELECT) и умножить ее на значение 123,34, которое необходимо сохранить в отдельной переменной, вывести значение переменной на экран.

9. Подсчитать суммарную цену всех закупок книг, результат поместить в переменную, вывести значение переменной на экран.

10. Подсчитать количество книг в справочнике книг, результат поместить в переменную, вывести значение переменной на экран.

11. Определить минимальную дату рождения автора в справочнике авторов, результат поместить в переменную, вывести значение переменной на экран.

Сочетание ключевых слов SET и SELECT

12. Подсчитать количество поставщиков книг, результат поместить в переменную.

13. Подсчитать сумму закупок книг, результат поместить в переменную.

14. Подсчитать среднюю цену в таблице покупок книг, результат поместить в переменную.

15. Подсчитать максимальную стоимость книг в закупке, результат поместить в переменную.

Работа с датой и временем

16. Определить переменную Date1 типа дата/время. Присвоить ей значение даты 31.12.2006 в формате dd.mm.yyyy.

17. Определить переменную Date1 типа дата/время. Присвоить ей значение даты 31.12.2006 в формате mm.dd.yyyy.

18. Определить переменную Date1 типа дата/время. Присвоить ей значение даты 31.12.2006 в формате yyyy.mm.dd.

Создание временной таблицы через переменную типа TABLE

19. Создать локальную таблицу с названием TEMP и полями типа дата/время, длинное целое, строка. Добавить в нее две записи с данными и вывести результат на экран.

20. Создать локальную таблицу с названием TEMP и полями типа длинное целое, строка и значением по умолчанию «введите что-нибудь»,

денежный. Добавить в нее две записи с данными и вывести результат на экран.

21. Создать локальную таблицу с названием TEMP и полями типа целое, динамическая строка, бит со значением по умолчанию «1». Добавить в нее две записи с данными и вывести результат на экран.

22. Создать локальную таблицу с названием TEMP и полями типа дата/время, длинное целое, строка. Добавить в нее две записи с данными и вывести результат на экран.

23. Создать локальную таблицу с названием TEMP и полями типа дата/время, длинное целое с автонаращиванием, динамическая строка. Добавить в нее две записи с данными и вывести результат на экран.

Преобразование типов переменных

24. Объявить переменные типа FLOAT, CHAR, TINYINT. Присвоить значения, соответствующие типам. Выполнить преобразование переменных типа FLOAT, CHAR, TINYINT в INT, DATETIME, BIT соответственно и вывести результат на экран.

25. Объявить переменные типа INT, DATETIME, BIT. Присвоить значения, соответствующие типам. Выполнить преобразование переменных типа INT, DATETIME, BIT в FLOAT, CHAR, TINYINT соответственно и вывести результат на экран.

26. Объявить переменные типа NUMERIC, VARCHAR, DATETIME. Присвоить значения, соответствующие типам. Выполнить преобразование переменных типа NUMERIC, VARCHAR, DATETIME в FLOAT, CHAR, BIGINT соответственно и вывести результат на экран.

27. Объявить переменные типа BIT, NVARCHAR, DATETIME. Присвоить значения, соответствующие типам. Выполнить преобразование переменных типа BIT, NVARCHAR, DATETIME в FLOAT, INT, BIGINT соответственно и вывести результат на экран.

Условная конструкция IF

28. Подсчитать количество поставщиков в таблице Deliveries. Если их в таблице от 2 до 5, то ничего не сообщать, в противном случае вывести сообщение вида "В таблице ... поставщиков" (вместо многоточия поставить точное количество поставщиков).

29. Подсчитать сумму закупок книг в таблице покупок. Если полученная сумма в диапозоне от 1000 до 5000, то ничего не сообщать, в противном случае вывести сообщение вида "Сумма закупок = …" (вместо многоточия поставить точную сумму).

30. Подсчитать среднюю стоимость закупки книг в таблице покупок. Если полученная стоимость в диапозоне от 1000 до 5000, то ничего не со-

общать, в противном случае вывести сообщение вида "Средняя стоимость закупки = …" (вместо многоточия поставить точную среднюю стоимость).

31. Определить минимальную стоимость закупки книг в таблице покупок. Если полученная стоимость в диапозоне от 200 до 300, то ничего не сообщать, в противном случае вывести сообщение вида "Минимальная стоимость закупки = …" (вместо многоточия поставить точную стоимость).

Цикл WHILE

32. Определить количество записей в таблице Authors. Пока записей меньше 15, делать в цикле добавление записи в таблицу с автоматическим наращиванием значения ключевого поля, а вместо имени автора ставить значение 'Автор не известен'.

33. Определить количество записей в таблице издательств. Пока записей меньше 20, делать в цикле добавление записи в таблицу с автоматическим наращиванием значения ключевого поля, а вместо названия издательства ставить значение 'не известно'.

34. Определить количество записей в таблице поставщиков. Пока записей меньше 17, делать в цикле добавление записи в таблицу с автоматическим наращиванием значения ключевого поля, а вместо названия поставщика ставить значение 'не известен'.

Объявление курсора

35. Создать статический курсор по данным таблицы Books с полями Code_book, Title_book.

36. Создать динамический курсор по данным таблицы поставщиков (таблица Deliveries) с полями Name_delivery, Name_company.

37. Создать статический курсор по данным таблицы Books и Authors с полями Code_book, Title_book, Name_author.

38. Создать статический курсор по данным таблицы Books и Publishing_house с полями Code_book, Title_book, Publish.

Операторы для работы с курсором

39. Создать динамический курсор для чтения по данным таблицы Deliveries с полями Code_delivery, Name_delivery. Вывести данные 3-й записи.

40. Сделать текущей БД db_books. Поместить в курсор данные таблицы Purchases. Перебрать все записи таблицы Purchases. Просуммировать значения произведений полей Cost и Amount и результат сохранить в переменной Sum_table, которую после суммирования вывести на экран. Закрыть и удалить из памяти курсор.

41. Объявить статический курсор по данным таблиц Authors и Books. Вывести данные 5-й записи.

Использование функций для работы со сторовыми переменными

Базовый текст дан в отдельном файле по вариантам. Для выполнения этого блока заданий в Query Analyzer объявите переменную типа varchar и присвойте ей в качестве значения строку с базовым текстом, который будет анализироваться и/или исправляться в заданиях.

42. Удалить в тексте лишние пробелы. Лишними считаются те, которые идут непосредственно за пробелом. Подсчитать количество исправлений.

43. Подсчитать количество встреч каждой из следующих букв: "а", "в", "и", "п" в базовом тексте.

44. Подсчитать доли процентов встречи следующих букв: "е", "о", если суммарный процент встречаемости всех этих букв равен 100% или процент встречаемости е% + о% равен 100%.

45. По правилам оформления машинописных текстов перед знаками .,!?:; пробелы не ставятся, но обязательно ставятся после этих знаков. Удалите лишние пробелы. Подсчитать количество исправлений.

46. По правилам оформления машинописных текстов перед знаками .,!?:; пробелы не ставятся, но обязательно ставятся после этих знаков. Расставьте недостающие пробелы. Подсчитать количество исправлений.

47. Найти из исходного текста второе предложение и вернуть его в переменную Perem, а также вывести на экран весь исходный текст и найденное предложение.

48. Удалить из базового текста 2, 4, 6, 8 слова.

49. Удалить из базового текста 3, 5, 7, 10 слова.

50. Вставить в базовый текст вместо букв «а» - «АА».

51. Вставить в базовый текст вместо букв «е» и «о» - «ББ».

52. Поменять местами первое и последнее слова в базовом тексте.

Использование функций для работы с числами

53. Вывести значение формулы (1), переменные которой нужно описать и присвоить произвольные значения.

$$v = v_0 \cdot e^{\sqrt{\frac{R \cdot T}{45}}}. \qquad (1)$$

54. Подсчитать значение формулы (2), переменные которой нужно описать и присвоить произвольные значения.

$$y = 2^x \cdot \exp(\ln(x^2)). \qquad (2)$$

55. Подсчитать значение формулы (3), переменные которой нужно описать и присвоить произвольные значения.

$$y = \frac{sin(a)}{x^2 - b^3} \cdot a. \quad (3)$$

56. Подсчитать значение формулы (4), переменные которой нужно описать и присвоить произвольные значения.

$$y = \sum_{n=1}^{10} I_n \cdot a \quad . \quad (4)$$

57. Подсчитать значение формулы (5), переменные которой нужно описать и присвоить произвольные значения.

$$y = \frac{tg(a)}{a + b - c} \cdot \sqrt{a \cdot b \cdot c}. \quad (5)$$

58. Подсчитать значение формулы (6), переменные которой нужно описать и присвоить произвольные значения.

$$y = \sqrt{\sin(a) \cdot \exp(b \cdot c)} \quad . \quad (6)$$

59. Подсчитать значение формулы (7), переменные которой нужно описать и присвоить произвольные значения.

$$y = x^4 \cdot ln(a) - b \cdot c. \quad (7)$$

60. Подсчитать значение формулы (8), переменные которой нужно описать и присвоить произвольные значения.

$$y = \frac{\sqrt{x - a}}{b^3} \quad . \quad (8)$$

61. Подсчитать значение формулы (9), переменные которой нужно описать и присвоить произвольные значения.

$$y = \frac{a \cdot cos(x)}{b^2 - a^2} \cdot sin(x). \quad (9)$$

Использование функций для работы с типом дата/время

62. Вывести на экран название текущего месяца и текущее время. Записать в таблицу Purchases в поле Date_order одинаковую дату поступления, которая равна 12.03.2000.

63. Разобрать на отдельные составляющие текущую дату и время и вывести значения на экран в следующем порядке (вместо многоточий):

64. "Сегодня: День = …, Месяц = …, Год = …, Часов = …, Минут = …, Секунд= …"

65. В исходный текст, сохраненный в переменной Perem, после слова " время " вставить текущее время. Результат сохранить в той же переменной Perem и вывести на экран.

Лабораторная работа №4

СОЗДАНИЕ ХРАНИМЫХ ПРОЦЕДУР В MICROSOFT SQL SERVER

Цель работы – научиться создавать и использовать хранимые процедуры на сервере БД.

Содержание работы:

1. Проработка всех примеров, анализ результатов их выполнения в утилите Query Analyzer. Проверка наличия созданных процедур в текущей БД.
2. Выполнение всех примеров и заданий по ходу лабораторной работы.
3. Выполнение индивидуальных заданий по вариантам.

Пояснения к выполнению работы

Для освоения программирования хранимых процедур используем пример базы данных с названием DB_Books, которая была создана в лабораторной работе №1. При выполнении примеров и заданий обращайте внимание на соответствие названий БД, таблиц и других объектов проекта.

Хранимые процедуры представляют собой набор команд, состоящий из одного или нескольких операторов SQL или функций и сохраняемый в базе данных в откомпилированном виде.

Типы хранимых процедур [1]

Системные *хранимые процедуры* предназначены для выполнения различных административных действий. Практически все действия по администрированию сервера выполняются с их помощью. Можно сказать, что системные *хранимые процедуры* являются интерфейсом, обеспечивающим работу с системными таблицами. Системные *хранимые процедуры* имеют префикс sp_, хранятся в системной базе данных и могут быть вызваны в контексте любой другой базы данных.

Пользовательские *хранимые процедуры* реализуют те или иные действия. *Хранимые процедуры* – полноценный объект базы данных. Вследствие этого каждая *хранимая процедура* располагается в конкретной базе данных, где и выполняется.

Временные *хранимые процедуры* существуют лишь некоторое время, после чего автоматически уничтожаются сервером. Они делятся на локальные и глобальные. Локальные временные *хранимые процедуры* могут

быть вызваны только из того соединения, в котором созданы. При *создании* такой процедуры ей необходимо дать имя, начинающееся с одного символа #. Как и все временные объекты, *хранимые процедуры* этого типа автоматически удаляются при отключении пользователя, перезапуске или остановке сервера. Глобальные временные *хранимые процедуры* доступны для любых соединений сервера, на котором имеется такая же процедура. Для ее определения достаточно дать ей имя, начинающееся с символов ##. Удаляются эти процедуры при перезапуске или остановке сервера, а также при закрытии соединения, в контексте которого они были созданы.

Создание, изменение хранимых процедур [1]

Создание хранимой процедуры предполагает решение следующих задач: планирование прав доступа. При создании хранимой процедуры следует учитывать, что она будет иметь те же права доступа к объектам базы данных, что и создавший ее пользователь; определение параметров хранимой процедуры, хранимые процедуры могут обладать входными и выходными параметрами; разработка кода хранимой процедуры. Код процедуры может содержать последовательность любых команд SQL, включая вызов других хранимых процедур.

Синтаксис оператора создания новой или изменения имеющейся хранимой процедуры в обозначениях MS SQL Server:

```
{CREATE | ALTER } PROC[EDURE] имя_процедуры [;номер]
[{@имя_параметра тип_данных } [VARYING ] [=default][OUTPUT] ][,...n]
[WITH { RECOMPILE | ENCRYPTION | RECOMPILE,
  ENCRYPTION }]
[FOR REPLICATION]
AS
  sql_оператор [...n]
```

Рассмотрим параметры данной команды.

Используя префиксы sp_, #, ##, создаваемую процедуру можно определить в качестве системной или временной. Как видно из синтаксиса команды, не допускается указывать имя владельца, которому будет принадлежать создаваемая процедура, а также имя базы данных, где она должна быть размещена. Таким образом, чтобы разместить создаваемую хранимую процедуру в конкретной базе данных, необходимо выполнить команду CREATE PROCEDURE в контексте этой базы данных. При обращении из тела хранимой процедуры к объектам той же базы данных можно использовать укороченные имена, т. е. без указания имени базы данных. Когда же требуется обратиться к объектам, расположенным в других базах данных, указание имени базы данных обязательно.

Для передачи входных и выходных данных в создаваемой хранимой процедуре имена параметров должны начинаться с символа @. В одной

хранимой процедуре можно задать множество параметров, разделенных запятыми. В теле процедуры не должны применяться локальные переменные, чьи имена совпадают с именами параметров этой процедуры.

Для определения типа данных параметров хранимой процедуры подходят любые типы данных SQL, включая определенные пользователем. Однако тип данных CURSOR может быть использован только как выходной параметр хранимой процедуры, т.е. с указанием ключевого слова OUTPUT.

Наличие ключевого слова OUTPUT означает, что соответствующий параметр предназначен для возвращения данных из хранимой процедуры. Однако это вовсе не означает, что параметр не подходит для передачи значений в хранимую процедуру. Указание ключевого слова OUTPUT предписывает серверу при выходе из хранимой процедуры присвоить текущее значение параметра локальной переменной, которая была указана при вызове процедуры в качестве значения параметра. Отметим, что при указании ключевого слова OUTPUT значение соответствующего параметра при вызове процедуры может быть задано только с помощью локальной переменной. Не разрешается использование любых выражений или констант, допустимое для обычных параметров.

Ключевое слово VARYING применяется совместно с параметром OUTPUT, имеющим тип CURSOR. Оно определяет, что выходным параметром будет результирующее множество.

Ключевое слово DEFAULT представляет собой значение, которое будет принимать соответствующий параметр по умолчанию. Таким образом, при вызове процедуры можно не указывать явно значение соответствующего параметра.

Так как сервер кэширует план исполнения запроса и компилированный код, при последующем вызове процедуры будут использоваться уже готовые значения. Однако в некоторых случаях все же требуется выполнять перекомпиляцию кода процедуры. Указание ключевого слова RECOMPILE предписывает системе создавать план выполнения хранимой процедуры при каждом ее вызове.

Параметр FOR REPLICATION востребован при репликации данных и включении создаваемой хранимой процедуры в качестве статьи в публикацию.

Ключевое слово ENCRYPTION предписывает серверу выполнить шифрование кода хранимой процедуры, что может обеспечить защиту от использования авторских алгоритмов, реализующих работу хранимой процедуры.

Ключевое слово AS размещается в начале собственно тела хранимой процедуры. В теле процедуры могут применяться практически все команды SQL, объявляться транзакции, устанавливаться блокировки и вызы-

ваться другие хранимые процедуры. Выход из хранимой процедуры можно осуществить посредством команды RETURN.

Удаление хранимой процедуры

DROP PROCEDURE {имя_процедуры} [,...n]

Выполнение хранимой процедуры [1]

Для выполнения хранимой процедуры используется команда:

[[EXEC [UTE] имя_процедуры [;номер]
[[@имя_параметра=]{значение | @имя_переменной}
[OUTPUT]|[DEFAULT]][,...n]

Если вызов хранимой процедуры не является единственной командой в пакете, то присутствие команды EXECUTE обязательно. Более того, эта команда требуется для вызова процедуры из тела другой процедуры или триггера.

Использование ключевого слова OUTPUT при вызове процедуры разрешается только для параметров, которые были объявлены при создании процедуры с ключевым словом OUTPUT.

Когда же при вызове процедуры для параметра указывается ключевое слово DEFAULT, то будет использовано значение по умолчанию. Естественно, указанное слово DEFAULT разрешается только для тех параметров, для которых определено значение по умолчанию.

Из синтаксиса команды EXECUTE видно, что имена параметров могут быть опущены при вызове процедуры. Однако в этом случае пользователь должен указывать значения для параметров в том же порядке, в каком они перечислялись при создании процедуры. Присвоить параметру значение по умолчанию, просто пропустив его при перечислении, нельзя. Если же требуется опустить параметры, для которых определено значение по умолчанию, достаточно явного указания имен параметров при вызове хранимой процедуры. Более того, таким способом можно перечислять параметры и их значения в произвольном порядке.

Отметим, что при вызове процедуры указываются либо имена параметров со значениями, либо только значения без имени параметра. Их комбинирование не допускается.

Использование RETURN **в хранимой процедуре**

Позволяет выйти из процедуры в любой точке по указанному условию, а также позволяет передать результат выполнения процедуры числом, по которому можно судить о качестве и правильности выполнения процедуры.

Пример создания процедуры без параметров:

```
CREATE PROCEDURE Count_Books AS
    Select count(Code_book) from Books
    Go
```

Задание 1. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Enterprise Manager. Запустите ее в утилите Query Analyzer с помощью команды

```
EXEC Count_Books
```

Проверьте результат.

Пример создания процедуры с входным параметром:

```
CREATE PROCEDURE Count_Books_Pages @Count_pages as Int
AS
    Select count(Code_book) from Books WHERE Pages>=@Count_pages
    Go
```

Задание 2. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Enterprise Manager. Запустите ее в утилите Query Analyzer с помощью команды

```
EXEC Count_Books_Pages 100
```

Проверьте результат.

Пример создания процедуры с входными параметрами:

```
CREATE PROCEDURE Count_Books_Title @Count_pages as Int, @Title AS
Char(10)
AS
    Select count(Code_book) from Books WHERE Pages>=@Count_pages
AND Title_book LIKE @Title
    Go
```

Задание 3. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Enterprise Manager. Запустите ее в утилите Query Analyzer с помощью команды

```
EXEC Count_Books_Title 100, 'П%'
```

Проверьте результат.

Пример создания процедуры с входными параметрами и выходным параметром:

```
CREATE PROCEDURE Count_Books_Itogo @Count_pages  Int, @Title
Char(10) , @Itogo Int OUTPUT
AS
```

```
    Select @Itogo = count(Code_book) from Books WHERE
Pages>=@Count_pages AND Title_book LIKE @Title
Go
```

Задание 4. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Enterprise Manager. Запустите ее в утилите Query Analyzer с помощью набора команд:

```
    Declare @q As int
    EXEC Count_Books_Itogo 100, 'П%', @q output
    select @q
```

Проверьте результат.

Пример создания процедуры с входными параметрами и RETURN:

```
CREATE PROCEDURE checkname @param int
AS
IF (SELECT Name_author FROM authors WHERE Code_author = @param) =
'Пушкин А.С.'
  RETURN 1
ELSE
  RETURN 2
```

Задание 5. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Enterprise Manager. Запустите ее в утилите Query Analyzer с помощью команд:

```
DECLARE @return_status int
EXEC @return_status = checkname 1
SELECT 'Return Status' = @return_status
```

Пример создания процедуры без параметров для увеличения значения ключевого поля в таблице Purchases **в 2 раза**:

```
CREATE PROC update_proc
AS
UPDATE Purchases SET Code_purchase = Code_purchase*2
```

Процедура не возвращает никаких данных.

Задание 6. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Enterprise Manager. Запустите ее в утилите Query Analyzer с помощью команды

```
    EXEC update_proc
```

Пример процедуры с входным параметром для получения всей информации о конкретном авторе:

```
CREATE PROC select_author  @k CHAR(30)
AS
SELECT * FROM Authors WHERE name_author=@k
```

Задание 7. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Enterprise Manager. Запустите ее в утилите Query Analyzer с помощью команд:

```
EXEC select_author 'Пушкин А.С.' или
select_author @k='Пушкин А.С.' или
EXEC select_author @k='Пушкин А.С.'
```

Пример создания процедуры с входным параметром и значением по умолчанию для увеличения значения ключевого поля в таблице Purchases **в заданное количество раза (по умолчанию в** 2 **раза)**:

```
CREATE PROC update_proc @p INT = 2
AS
UPDATE Purchases SET Code_purchase = Code_purchase *@p
```

Процедура не возвращает никаких данных.

Задание 8. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Enterprise Manager. Запустите ее в утилите Query Analyzer с помощью команд:

```
EXEC update_proc 4 или
EXEC update_proc @p = 4 или
EXEC update_proc      --будет использовано значение по умолчанию.
```

Пример создания процедуры с входным и выходным параметрами. Создать процедуру для определения количества заказов, совершенных за указанный период:

```
CREATE PROC count_purchases
 @d1 SMALLDATETIME,  @d2 SMALLDATETIME,
 @c INT OUTPUT
AS
SELECT @c=count(Code_purchase) from Purchases WHERE Date_order
BETWEEN   @d1 AND @d2
SET @c = ISNULL(@c,0)
```

Задание 9. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Enterprise Manager. Запустите ее в утилите Query Analyzer с помощью команд:

```
DECLARE  @c2 INT
EXEC count_purchases ’01-jun-2006’, ’01-jul-2006’, @c2 OUTPUT
SELECT @c2
```

Варианты заданий к лабораторной работе №4

Общие положения

В утилите Query Analyzer создать новую программу. Программно сделать активной созданную БД DB_Books с помощью оператора Use. Создать хранимые процедуры с помощью операторов Create procedure, причем самостоятельно определить имена процедур. Каждая процедура будет выполнять по одному SQL запросу, которые были выполнены во второй лабораторной работе. Причем код SQL запросов нужно изменить таким образом, чтобы в них можно было передавать значения полей, по которым осуществляется поиск.

Например, исходное задание и запрос в лабораторной работе №2:

```
/*Выбрать из справочника поставщиков (таблица Deliveries) названия компаний, телефоны и ИНН (поля Name_company, Phone и INN), у которых название компании (поле Name_company)  'ОАО МИР'.
    SELECT Name_company, Phone, INN FROM Deliveries
    WHERE Name_company = 'ОАО МИР'
*/
--В данной работе будет создана процедура:
CREATE PROC select_name_company  @comp CHAR(30)
AS
    SELECT Name_company, Phone, INN FROM Deliveries
    WHERE Name_company = @comp

--Для запуска процедуры используется команда:
    EXEC select_name_company  'ОАО МИР'
```

Сохранить файл программы с названием **ФамилияСтудента_ЛАб_4**. В SQL Server Enterprise MANAGER в разделе хранимых процедур БД DB_Books проверить наличие процедур.

Список заданий

В утилите Query Analyzer создать новую программу. Программно сделать активной индивидуальную БД, созданную в лабораторной работе №1, с помощью оператора Use. Создать хранимые процедуры с помощью операторов Create procedure, причем самостоятельно определить имена про-

цедур. Каждая процедура будет выполнять по одному SQL запросу, которые представлены в виде отдельных заданий по вариантам.

Сохранить файл программы с названием **Фамилия Студента_Лаб_4_№варианта**. В SQL Server Enterprise Manager в разделе хранимых процедур индивидуальной БД проверить наличие процедур.

***Вариант** 1*

1. Вывести список сотрудников, у которых есть хотя бы один ребенок.
2. Вывести список детей, которым выдали подарки в указанный период.
3. Вывести список родителей, у которых есть несовершеннолетние дети.
4. Вывести информацию о подарках со стоимостью больше указанного числа, отсортированных по дате.

***Вариант** 2*

1. Вывести список приборов с указанным типом.
2. Вывести количество отремонтированных приборов и общую стоимость ремонтов у указанного мастера.
3. Вывести список владельцев приборов и количество их обращений, отсортированный по количеству обращений по убыванию.
4. Вывести информацию о мастерах с разрядом больше указанного числа или с датой приема на работу меньше указанной даты.

***Вариант** 3*

1. Вывести список цветков с указанным типом листа.
2. Вывести список кодов продаж, по которым продано цветов на сумму больше указанного числа.
3. Вывести дату продажи, сумму, продавца и цветок по указанному коду продажи.
4. Вывести список цветов и сорт для цветов с высотой больше указанного числа или цветущий.

***Вариант** 4*

1. Вывести список лекарств с указанным показанием к применению.
2. Вывести список дат поставок, по которым продано больше указанного числа одноименного лекарства.
3. Вывести дату поставки, сумму, ФИО руководителя от поставщика и название лекарства по коду поступления больше указанного числа.
4. Вывести список лекарств и единицы измерения для лекарств с количеством в упаковке больше указанного числа или кодом лекарства меньше определенного значения.

***Вариант** 5*

1. Вывести список сотрудников с указанной должностью.
2. Вывести список списанного оборудования по указанной причине.

3. Вывести дату поступления, название оборудования, ФИО ответственного и дату списания для оборудования, списанного в указанный период.
4. Вывести список оборудования с указанным типом или с датой поступления больше определенного значения.

***Вариант** 6*

1. Вывести список блюд с весом больше указанного числа.
2. Вывести список продуктов, в названии которых встречается указанный фрагмент слова.
3. Вывести объем продукта, название блюда, название продукта с кодом блюда от указанного начального значения по определенному конечному значению.
4. Вывести порядок приготовления блюда и название блюда с количеством углеводов больше определенного значения или количеством калорий больше указанного значения.

***Вариант** 7*

1. Вывести список сотрудников с указанной должностью.
2. Вывести список документов, в содержании которых встречается указанный фрагмент слова.
3. Вывести дату регистрации, тип документа, ФИО регистратора и название организации для документов, зарегистрированных в указанный период.
4. Вывести список зарегистрированных документов с определенным типом документа или с датой регистрации больше указанного значения.

***Вариант** 8*

1. Вывести список сотрудников с указанной должностью.
2. Вывести список документов, в содержании которых встречается указанный фрагмент слова.
3. Вывести дату регистрации, тип документа, ФИО отправителя и название организации для документов, зарегистрированных в указанный период.
4. Вывести список зарегистрированных документов с указанным типом документа или с кодом документа меньше определенного значения.

***Вариант** 9*

1. Вывести список сотрудников с указанной причиной увольнения.
2. Вывести список документов с датой регистрации в указанный период.
3. Вывести дату регистрации, причину увольнения, ФИО сотрудника для документов, зарегистрированных в указанный период.
4. Вывести список зарегистрированных документов с кодом документа в указанном диапазоне.

***Вариант** 10*

1. Вывести список сотрудников, бравших отпуск указанного типа.
2. Вывести список документов с датой регистрации в указанный период.
3. Вывести дату регистрации, тип отпуска, ФИО сотрудника для документов, зарегистрированных в указанный период.
4. Вывести список зарегистрированных документов с кодом документа в указанном диапазоне.

***Вариант** 11*

1. Вывести список сотрудников, назначенных на указанную должность.
2. Вывести список документов с датой регистрации в указанный период.
3. Вывести дату регистрации, должность, ФИО сотрудника для документов, зарегистрированных в указанный период.
4. Вывести список зарегистрированных документов с кодом документа в указанном диапазоне.

***Вариант** 12*

1. Вывести список оборудования с указанным типом.
2. Вывести список оборудования, которое брал в прокат определенный клиент.
3. Вывести список лиц, бравших оборудование в прокат и количество их обращений, отсортированный по количеству обращений по убыванию.
4. Вывести информацию о клиентах, отсортированных по адресам.

***Вариант** 13*

1. Вывести список оборудования с указанным типом.
2. Вывести список оборудования, которое списал определенный сотрудник.
3. Вывести количество списанного оборудования, сгруппированного по типам оборудования.
4. Вывести информацию о сотрудниках с датой приема на работу больше определенной даты.

***Вариант** 14*

1. Вывести список цветков с указанным типом листа.
2. Вывести список кодов поступлений, по которым продано цветов на суммы больше определенного значения.
3. Вывести дату поступления, сумму, названия поставщика и цветов по определенному коду поставщика.
4. Вывести список цветов и сорт для цветов с высотой больше определенного числа или цветущий.

***Вариант** 15*

1. Вывести список клиентов, заехавших в номера в указанный период.

2. Вывести общую сумму оплат за номера для каждого клиента.
3. Вывести дату заезда, тип номера, ФИО клиентов, зарегистрированных в указанный период.
4. Вывести список зарегистрированных клиентов в номерах определенного типа.

***Вариант** 16*

1. Вывести список оборудования с указанным типом.
2. Вывести список оборудования, которое брал в прокат определенный клиент.
3. Вывести список лиц, бравших оборудование в прокат и количество их обращений, отсортированных по количеству обращений по убыванию.
4. Вывести информацию о клиентах, отсортированных по адресам.

***Вариант** 17*

1. Вывести список ценностей с закупочной стоимостью больше определенного значения или сроком гарантии больше указанного числа.
2. Вывести список мест нахождения материальных ценностей, в названии которых встречается указанное слово.
3. Вывести сумму стоимости ценностей с кодом в указанном диапазоне.
4. Вывести список материально ответственных лиц с датой приема на работу в указанном диапазоне.

***Вариант** 18*

1. Вывести список ремонтных работ, выполненных определенным мастером.
2. Вывести список этапов работ, входящих в работы, в названии которых встречается указанное слово.
3. Вывести сумму стоимости этапов ремонтных работ для работ с кодом в указанном диапазоне.
4. Вывести список мастеров с датой приема на работу в указанном диапазоне.

***Вариант** 19*

1. Вывести список лекарств с определенным показанием.
2. Вывести список номеров чеков, по которым продано больше определенного числа лекарств.
3. Вывести дату продажи, сумму, ФИО кассира и лекарство по чеку с указанным номером.
4. Вывести список лекарств и единицы измерения для лекарств с количеством в упаковке больше указанного числа или кодом лекарства меньше определенного значения.

***Вариант** 20*

1. Вывести список сотрудников с указанной должностью.

2. Вывести список документов, в содержании которых встречается указанный фрагмент слова.
3. Вывести дату регистрации, тип документа, ФИО исполнителя и факт исполнения для документов, зарегистрированных в указанный период.
4. Вывести список зарегистрированных документов с указанным типом документа или с кодом документа в определенном диапазоне.

Лабораторная работа №5

СОЗДАНИЕ КЛИЕНТСКОЙ ЧАСТИ ПРИЛОЖЕНИЯ ДЛЯ ПРОСМОТРА, РЕДАКТИРОВАНИЯ ДАННЫХ БД. ВЫЗОВ ХРАНИМЫХ ПРОЦЕДУР ИЗ КЛИЕНТСКОЙ ЧАСТИ

Цель работы – научиться создавать клиентское приложение для работы с базой данных с применением встроенных инструментов в среде Delphi 7.
Содержание работы:

1. Выполнение всех заданий по ходу лабораторной работы.
2. Выполнение индивидуальных заданий.

Пояснения к выполнению работы

Для создания клиентского приложения в Delphi 7 используем пример базы данных с названием DB_Books, которая была создана в лабораторной работе №1, выполним подключение БД через ODBC драйвер, выберем тип «системный источник данных» (System Name) и дадим название для DSN - DB_BooksDSN. При выполнении примеров и заданий обращайте внимание на соответствие названий БД, таблиц и других объектов проекта.

В данной работе при создании клиентского приложения будем использовать встроенные инструменты для работы с данными (в Delphi 7 - вкладка Data Controls).
В Delphi 7:

1. В новом проекте создадим модуль данных (меню File - New - DataModule). Переименуем форму в DM (свойство Name - DM).

2. В основной форме (например, Form1, переименовать ее в MainForm) в коде добавить описание модуля DM, для этого после ключевого слова USES в конце списка поставить запятую и дописать название программного модуля, например UNIT2, который соответствует созданной форме DM.

3. На форму DM добавить компонент ADOConnection (название, например ADOConnection1), 5 компонентов типа ADOTable (переименовать компоненты в ADOPurchases, ADOBooks, ADOAuthors, ADODeliveries,

ADOPublish), 5 компонентов типа DataSource (переименовать компоненты в DataPurchases, DataBooks, DataAuthors, DataDeliveries, DataPublish).

4. У компонента ADOConnection1 настроить свойства:
   Connected String = нажать кнопку Build\ выбрать Поставщик данных - Microsoft OLE DB Provider for ODBC Drivers
   Подключение: выбрать DSN - DB_BooksDSN.
   Получится в результате - Provider=MSDASQL.1;Persist Security Info=False;User ID=sa;Data Source=DB_BooksDSN
   LoginPrompt = False
   Connected = True
5. У ADOAuthors изменить следующие свойства:
   Connection на ADOConnection1;
   TableName на Authors;
   Active на True.
6. У DataAuthors изменить следующие свойства (это будет ссылка на таблицу):
   DataSet на ADOAuthors.
7. У ADOPurchases изменить следующие свойства:
   Connection на ADOConnection1;
   TableName на Purchases;
   Active на True.
8. У DataPurchases изменить следующие свойства (это будет ссылка на таблицу):
   DataSet на ADOPurchases.
9. У ADOBooks изменить следующие свойства:
   Connection на ADOConnection1;
   TableName на Books;
   Active на True.
10. У DataBooks изменить следующие свойства (это будет ссылка на таблицу):
   DataSet на ADOBooks.
11. У ADODeliveries изменить следующие свойства:
   Connection на ADOConnection1;
   TableName на Deliveries;
   Active на True.
12. У DataDeliveries изменить следующие свойства (это будет ссылка на таблицу):
   DataSet на ADODeliveries.
13. У ADOPublish изменить следующие свойства:
   Connection на ADOConnection1;
   TableName на Publishing_house;
   Active на True.

14. У DataPublish изменить следующие свойства (это будет ссылка на таблицу):

DataSet на ADOPublish.

15. На основной форме (MainForm) добавить компонент Меню с вкладки Standart. В редакторе меню сделать первый пункт «Работа с таблицами» и в подменю пункты: «Авторы», «Книги», «Издательства», «Поставщики», «Поставки».

16. Создать пять форм, каждую из которых назвать (изменить свойство Name): FormAuthors, FormPurchases, FormBooks, FormDeliveries, FormPublish. В главной форме, в коде добавить описание этих форм, для этого после ключевого слова USES в конце списка через запятую дописать названия программных модулей, которые соответствуют описанным формам. В каждой из созданных форм в коде добавить описание модуля DM, для этого после ключевого слова USES в конце списка поставить запятую и дописать название программного модуля, например UNIT2, который соответствует созданной форме DM.

17. На основной форме в подпунктах меню в соответствующих методах Click вызвать соответствующие формы с помощью кода:

для FormAuthors:

```
DM.ADOAuthors.Open;
FormAuthors:=TFormAuthors.Create(Application);
FormAuthors.Show;
```

для FormPurchases:

```
DM.ADOPurchases.Open;
FormPurchases:=TFormPurchases.Create(Application);
FormPurchases.Show;
```

для FormBooks:

```
DM.ADOBooks.Open;
FormBooks:=TFormBooks.Create(Application);
FormBooks.Show;
```

для FormDeliveries:

```
DM.ADODeliveries.Open;
FormDeliveries:=TFormDeliveries.Create(Application);
FormDeliveries.Show;
```

для FormPublish:

```
DM.ADOPublish.Open;
FormPublish:=TFormPublish.Create(Application);
FormPublish.Show.
```

18. На формы FormAuthors, FormPurchases, FormBooks, FormDeliveries, FormPublish добавить с вкладки Data Controls по паре компонент типа DBGrid и DBNavigator. Настроить у DBGrid и DBNavigator свойство DataSource для связи с соответствующим источником данных.

19. Проверить работу приложения.

20. На форму FormBooks с вкладки Data Controls добавить 3 компонента типа DBEdit, 2 компонента типа DBLookupComboBox.

У 1-го компонента DBEdit изменить свойства:

Name на DBE_Code_Book;
DataSource на DM.DataBooks;
DataField на Code_book.

У 2-го компонента DBEdit изменить свойства:

Name на DBE_Title_book;
DataSource на DM.DataBooks;
DataField на Title_book.

У 3-го компонента DBEdit изменить свойства:

Name на DBE_Pages;
DataSource на DM.DataBooks;
DataField на Pages.

У 1-го компонента DBLookupComboBox изменить свойства:

Name на DBL_Code_author;
DataSource на DM.DataBooks;
DataField на Code_author;
ListSource на DM.DataAuthors;
ListField на name_author;
KeyField на Code_author.

У 2-го компонента DBLookupComboBox изменить свойства:

Name на DBL_Code_ Publish;
DataSource на DM.DataBooks;
DataField на Code_publish;
ListSource на DM.DataPublish;
ListField на Publish;
KeyField на Code_publish.

У компонента DBGrid настроить свойство ReadOnly в режим True.

21. Аналогично для остальных форм добавить элементы типа DBEdit (для полей данных которых будут просто набираться или редактироваться пользователем) и DBLookupComboBox (для полей связи, к которым подходит связь типа «много», они позволят выбирать данные из соответствующего справочника, и пользователю не придется помнить значения кодов).

22. Проверить работу приложения.

23. На форму FormBooks с вкладки Standart добавить 5 компонентов типа Button.

У 1-го компонента Button изменить свойства и метод:

Name на B_Publish;
Caption на Фильтр по текущему издательству;
В методе Click кнопки написать код:
dm.ADOBooks.Filter:= 'Code_Publish = '+ form-books.DBGrid1.Columns.Items[4].Field.Text;
dm.ADOBooks.Filtered:= true.

У 2-го компонента Button изменить свойства и метод:
Name на B_Title_Book;
Caption на Фильтр по текущему названию книги.
В методе Click кнопки написать код:
dm.ADOBooks.Filter:= 'Title_book = '+ QuotedStr(form-books.DBGrid1.Columns.Items[1].Field.Text);
dm.ADOBooks.Filtered:= true.

У 3-го компонента Button изменить свойства и метод:
Name на B_Author;
Caption на Фильтр по текущему автору.
В методе Click кнопки написать код:
dm.ADOBooks.Filter:= 'Code_Author = '+ form-books.DBGrid1.Columns.Items[0].Field.Text;
dm.ADOBooks.Filtered:= true.

У 4-го компонента Button изменить свойства и метод:
Name на B_Pages;
Caption на Фильтр по количеству книг.
В методе Click кнопки написать код:
dm.ADOBooks.Filter:= 'Pages = '+ form-books.DBGrid1.Columns.Items[3].Field.Text;
dm.ADOBooks.Filtered:= true.

У 5-го компонента Button изменить свойства и метод:
Name на B_Cancel;
Caption на Снять фильтр.
В методе Click кнопки написать код:
dm.ADOBooks.Filter:= '';
dm.ADOBooks.Filtered:= false.

24. Аналогично для остальных форм добавить элементы типа Button, которые будут запускать фильтры по соответствующим значениям полей текущей записи в объекте Grid.

25. Проверить работу приложения.

26. Создать форму, назвать (изменить свойство Name) FormProcedure. В главной форме в коде добавить описание этой формы, для этого после ключевого слова USES в конце списка через запятую дописать название программного модуля, которое соответствует созданной форме. В созданной форме в коде добавить описание модуля DM, для этого после ключе-

вого слова USES в конце списка поставить запятую и дописать название программного модуля, например UNIT2, который соответствует созданной форме DM.

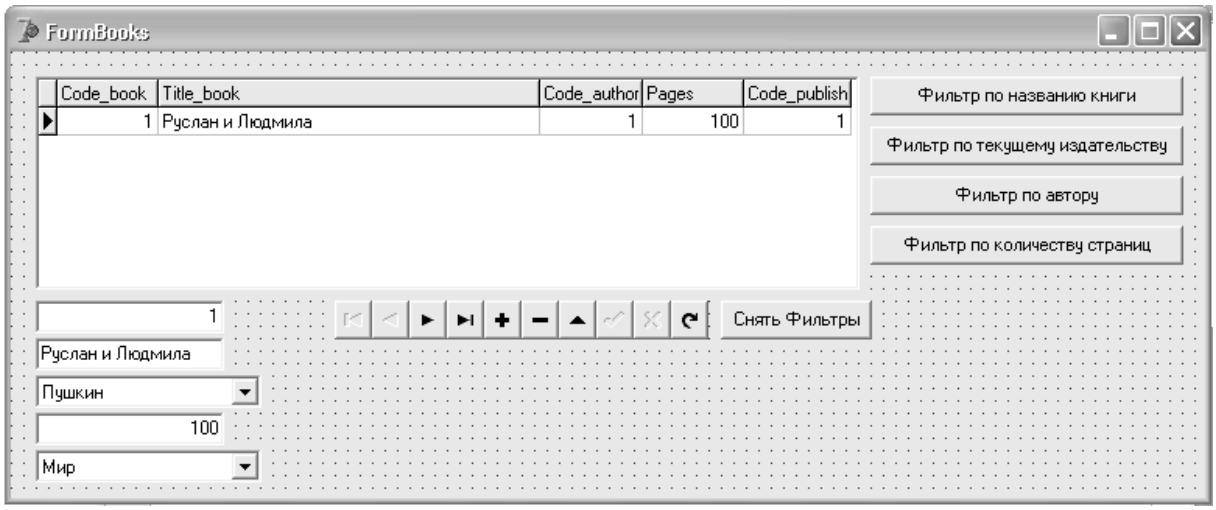

Рис. 5.1. Пример расположения компонентов на форме FormBooks

27. Добавить на главной форме в меню пункт с названием **Работа с процедурами**. В методе Click пункта меню написать код для запуска формы FormProcedure (см. пример кода в пункте 17 текущей лабораторной работы).

28. Подключить хранимую процедуру Count_purchases, выполненную в задании 9 лабораторной работы №4. На форму DM добавить компонент ADOStoredProc, переименовать компонент в ADOSP_Count. У ADOSP_Count изменить следующие свойства:

Connection на ADOConnection1;

ProcedureName на Count_purchases.

29. У компонента ADOSP_Count выбрать свойтсво Parameters и в свойствах каждого входного параметра исправить свойство Value – Type на Date, а для выходного параметра свойство Value – Type на Integer.

30. На форму FormProcedure добавить 3 компонента типа Edit (имена соответственно Edit1, Edit2, Edit3) и 1 компонент типа Button. Рядом с каждым компонентом Edit поставить Label и исправить их свойства Caption соответственно на «Количество покупок за указанный период», «Введите дату начала периода», «Введите дату конца периода».

31. На кнопке поменять название на «Выполнить запрос». В методе Click кнопки написать следующий код:

```
DM.ADOSP_Count.Prepared;
try
 DM.ADOSP_Count.Parameters.ParamByName('@d1').Value:=StrToDate(Edit2.Text);
```

```
  DM.ADOSP_Count.Parameters.ParamByName('@d2').Value:= StrToDate(Edit3.Text);
  DM.ADOSP_Author.ExecProc;
  Edit1.Text:= inttostr(DM.ADOSP_Count.Parameters.ParamByName('@c').Value);
  ShowMessage(Edit1.Text);
finally
  DM.ADOSP_Count.Prepared:=not(DM.ADOSP_Count.Prepared);
end;// try
```

32. Проверить работу приложения.

Задания к лабораторной работе №5

В Delphi 7 создать новый проект, далее для индивидуальной БД, созданной в лабораторной работе №1, создать интерфейс, включающий все функции и процедуры, которые описаны по ходу текущей лабораторной работы.

Лабораторная работа №6

СОЗДАНИЕ АДМИНИСТРАТИВНОЙ СТРАНИЦЫ

Цель работы – научиться организовывать со стороны клиентского приложения удаленное управление правами доступа к данным БД.

Содержание работы:

1. Выполнение всех заданий по ходу лабораторной работы.
2. Выполнение индивидуальных заданий.

Пояснения к выполнению работы

Для создания в приложении административной страницы используем пример базы данных с названием DB_Books, которая была создана в лабораторной работе №1, к которой сделано подключение через ODBC драйвер типа системного источника данных и названием DB_BooksDSN, а также используем клиентское приложение, которое было создано по ходу пояснений в лабораторной работе №5. При выполнении примеров и заданий обращайте внимание на соответствие названий БД, таблиц и других объектов проекта.

В Enterprise MANAGER

Создадим процедуру добавления логина в БД DB_Books в разделе Stored Procedures базы данных DB_Books, используя утилиту Enterprise Manager:

```
CREATE PROCEDURE addlogin1  @login_ char(15), @password1 char(15)
```

```
 AS
exec sp_addlogin @login_,@password1, 'DB_Books'
exec sp_adduser @login_,@login_
GO
```

В Delphi 7

1. В проекте, который был создан в лабораторной работе №5, на форму DM добавим компонент ADOStoredProc (название по умолчанию будет ADOStoredProc1).

   У компонента ADOStoredProc1 изменить свойства:

   Connection на ADOConnection1;

   ProcedureName на addlogin1.

2. У компонента ADOStoredProc1 выбрать Parameters и в свойствах каждого параметра исправить свойство Value – Type на String.

3. Добавить на основной форме (Form1) в меню пункт с названием **Администрирование**. Создать форму FormAdmin, которая будет запускаться из пункта меню **Администрирование** основной формы. В главной форме, в коде добавить описание формы FormAdmin, для этого после ключевого слова USES в конце списка через запятую дописать название программного модуля, которое соответствует созданной форме. В созданной форме в коде добавить описание модуля DM, для этого после ключевого слова USES в конце списка поставить запятую и дописать название программного модуля, например UNIT2, который соответствует созданной форме DM.

4. На форме FormAdmin расположить два компонента Edit (например, Edit1 и Edit2), один компонент Button. Рядом с компонентами типа Edit поставить элементы Label, в которых соответственно изменить свойства Caption на «Введите имя нового пользователя» и «Введите пароль».

5. В методе Click созданной кнопки написать:

```
try
 dm.ADOStoredProc1.Parameters.ParamByName('@login_').Value:=
Edit1.Text;
 dm.ADOStoredProc1.Parameters.ParamByName('@password1').Value:=
Edit2.Text;
 DM.ADOStoredProc1.ExecProc;
except
  ShowMessage('Невозможно добавление пользователя');

end; // try
```

6. Запустить приложение и проверить работу.

В Enterprise MANAGER

Создадим процедуру добавления разрешений в БД DB_Books в разделе Stored Procedures базы данных DB_Books:

```
CREATE PROCEDURE grantlogin  @text1 char(250)
 AS
declare @SQLString nvarchar(250)
SET @SQLString =  CAST( @text1 AS NVARCHAR(250) )
EXECUTE sp_executesql @SQLString
GO
```

В Delphi 7

7. На форму DM добавить еще одну ADOStoredProc (название по умолчанию будет ADOStoredProc2).

У компонента ADOStoredProc2 изменить свойства:

Connection на ADOConnection1;

ProcedureName на grantlogin.

8. У компонента ADOStoredProc2 выбрать Parameters и в свойствах каждого параметра исправить свойство Value – Type на String.

9. На форме FormAdmin расположить два компонента Combobox (например, Combobox1 и Combobox2). Рядом с компонентами типа Combobox поставить элементы Label, в которых соответственно изменить свойства Caption на «Выберите операцию» и «Выберите пользователя».

10. В список ComboBox1 занести перечень значений:

- INSERT,
- UPDATE,
- DELETE.

11. В список ComboBox2 занести перечень значений:

- Authors,
- Books,
- Purchases,
- Deliveries,
- Publishing_house.

12. На форме FormAdmin расположить компонент Edit (например, Edit3), один компонент Button. Рядом с компонентом типа Edit поставить элемент Label, в котором изменить свойства Caption на «Введите имя пользователя, которому назначается привилегия».

13. В методе Click созданной кнопки написать:

```
try
 dm.ADOStoredProc2.Parameters.ParamByName('@text1').Value:=
'GRANT '+ ComboBox1.Text + ' ON ' + ComboBox2.Text + ' TO ' +
Edit3.Text;
 DM.ADOStoredProc2.ExecProc;
```

```
except
  ShowMessage('Невозможно добавление разрешения');

end; // try
```

14. Запустить приложение и проверить работу.

Задания к лабораторной работе №6

В вашей индивидуальной базе данных, которая была выдана по вариантам (из лабораторной работы №1), создать 4 хранимые процедуры, которые будут выполнять операции по добавлению пользователя, удалению пользователя, добавлению разрешения на одну из таблиц, удалению разрешения на одну из таблиц. В клиентском приложении, которое было создано в лабораторной работе №5 по вашему варианту, добавить на основную форму в меню пункт **Администрирование**, который будет запускать форму **Администрирование**. На форме организовать запуск четырех созданных хранимых процедур с передачей данных в процедуры из клиентского приложения. Цель задания – создание удаленного управления правами доступа к вашей БД.

Лабораторная работа №7

СОЗДАНИЕ ОТЧЕТНЫХ ФОРМ В КЛИЕНТСКОМ ПРИЛОЖЕНИИ

Цель работы – научиться создавать формы отчетных документов по данным БД.

Содержание работы:

1. Выполнение всех заданий по ходу лабораторной работы.
2. Выполнение индивидуальных заданий.

Пояснения к выполнению работы

Для выполнения трех первых заданий используем пример базы данных с названием DB_Books, которая была создана в лабораторной работе №1. При выполнении примеров и заданий обращайте внимание на соответствие названий БД, таблиц и других объектов проекта.

Отчеты во многом похожи на формы и тоже позволяют получить результаты работы запросов в наглядной форме, но только не на экране, а в виде распечатки на принтере. Таким образом, в результате работы отчета создается *бумажный документ*.

В Delphi 7 есть несколько способов создания отчетов, познакомимся с

компонентами QReport. По умолчанию вкладка с QReport недоступна, для ее подключения необходимо выбрать меню Component – далее Install Packages – далее Add – найти файл C:\Program Files\Borland\Delphi7\Bin\ dclqrt70.bpl. В результате самой последней вкладкой на панели инструментов будет QReport.

Для создания отчета необходимо создать новую форму, на которую с панели инструментов поместить компоненту QuickReport. Она на форме имитирует представление данных на листе формата А4.

Структура отчета

Отчеты состоят из разделов или секций (Bands), а разделы могут содержать элементы управления. Для добавления раздела в отчет необходимо в свойствах компонента QuickReport1 найти BANDS и перевести в режим True необходимые разделы.

1. Структура отчета состоит из следующих разделов: *заголовка отчета (HasTitle), верхнего колонтитула (HasPageHeader), заголовка области данных (HasColumnHeader), области данных (HasDetail или HasSubDetail), нижнего колонтитула* (*HasPage*) и *примечания отчета (HasSummary).*
2. Раздел ***заголовка*** служит для печати общего заголовка отчета.
3. Раздел ***верхнего колонтитула*** можно использовать для печати подзаголовков, если отчет имеет сложную структуру и занимает много страниц. Здесь можно также помещать и номера страниц, если это не сделано в нижнем колонтитуле.
4. В ***области данных*** размещают элементы управления, связанные с содержимым полей таблиц базы. В эти элементы управления выдаются данные из таблиц для печати на принтере. Эти разделы будут на печати воспроизводиться столько раз, сколько записей присутствует в привязанном запросе или таблице.
5. Раздел ***нижнего колонтитула*** используют для тех же целей, что и раздел верхнего колонтитула. Можно использовать для подстановки полей для подписей должностных лиц, если есть необходимость подписывать отчет на каждой странице.
6. Раздел ***примечания*** используют для размещения дополнительной информации или итоговой информации по всем данным отчета. Печатается сразу после окончания области данных.
7. Для предварительного просмотра отчета в том виде, как он будет расположен на бумаге, необходимо вызвать метод Preview компонента QuickReport (на главной форме установить кнопку, в методе Click которой напишите, например Form1.QuickReport1.Preview). Пример отчета в режиме «Конструктор» представлен на рис. 7.1, а в режиме предваритеного просмотра – на рис. 7.2.

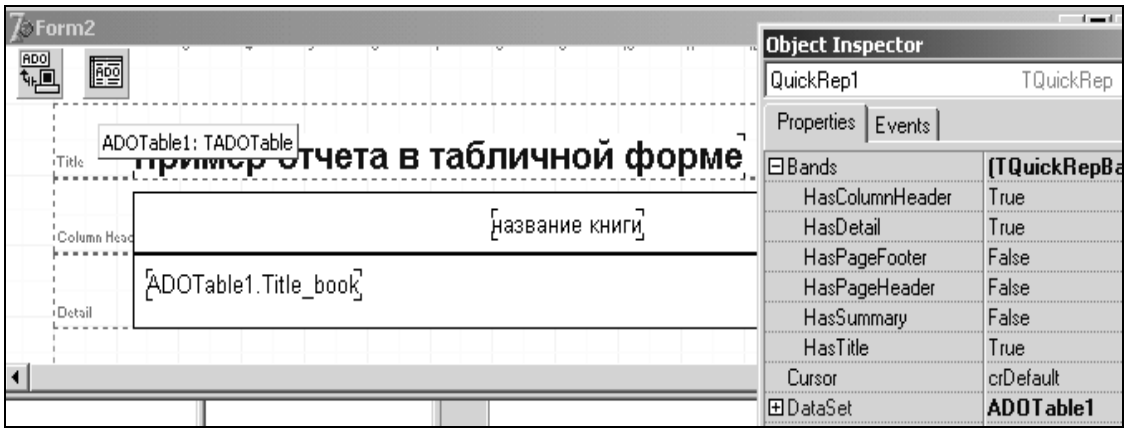

Рис 7.1. Пример отчета в режиме «Конструктор»

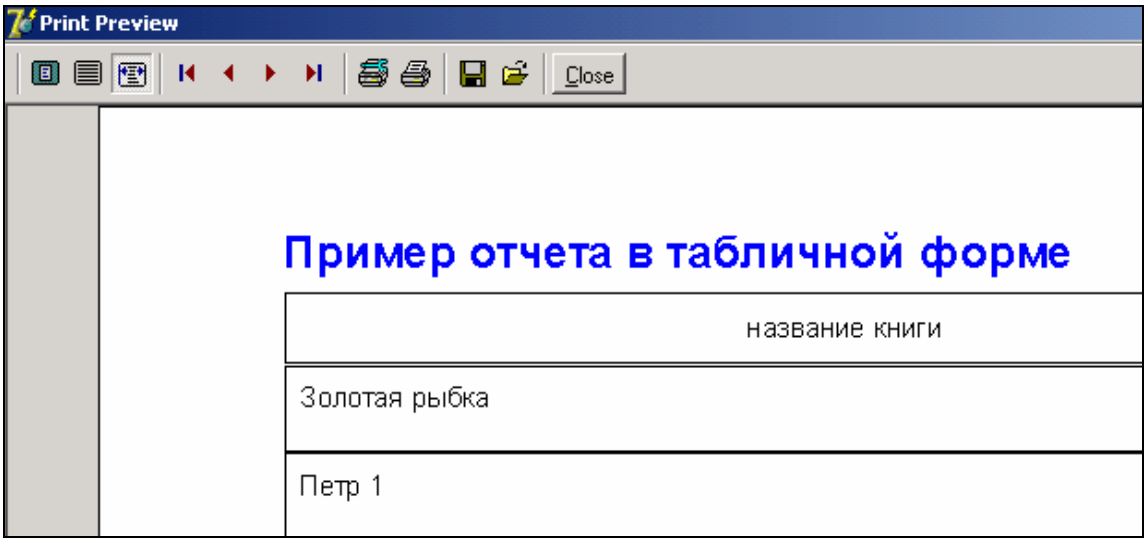

Рис. 7.2. Пример отчета в предварительном просмотре

Задание 1. Создание отчета в табличной форме по запросу, который выбирает из таблицы Books все поля, кроме кодов, из таблицы Publish_house – название издательства и место издательства, из таблицы Authors – имя автора.

1. В проекте на главной форме в меню добавить пункт меню **Отчеты**, а также подпункты:

   Отчет в табличной форме;
   Отчет в свободной форме;
   Отчет с группировкой по двум таблицам.

2. В проекте на форму DM добавить компонент ADOQUERY1, у которого изменить SQL на соответствующий запрос (см. текст задания), а свойство Connection на ADOConnection1. Активизировать запрос.

3. Создать новую форму (например, Form2). Связать эту форму с DM (в коде после ключевого слова USES добавить название соответстующего UNIT от модуля данных).

4. На форме разместить компонент QuickReport1. У свойства DataSet установить ссылку на ADOQUERY1. В свойствах включить такие разделы (BANDS), как *HasTitle*, *HasColumnHeader*, *HasDetail* и *HasSummary*.

5. В разделе Title QuickReport1 разместить метку (компонент QRLabel). В свойствах изменить его внешний вид и подпись «Пример табличного отчета по запросу».

6. В разделе ColumnHeader установить компоненты QRShape (для имитации обрамления шапки таблицы) и QRLabel (написать в нем **Название книги Автор Издательство**).

7. В разделе Detail установить 3 компонента QREXPR. Расположить компоненты симметрично под надписями. В каждом компоненте по очереди в свойстве Expression выбрать соответствующие поля запроса ADOQUERY1: title_book, name_author, publish.

8. В главной форме приложения в подпункте **Отчет в табличной форме** в методе Click написать команду: Form2.QuickReport1.Preview. В главной форме, в коде добавить описание формы Form2, для этого после ключевого слова USES в конце списка через запятую дописать название программного модуля, которое соответствует форме.

9. Запустить приложение, проверить работу.

Задание 2. Создание отчета в свободной форме по запросу из первого задания. Создадим карточку книги для библиотечной картотеки.

Особенность отчета в свободной форме в том, что он создает шаблон на каждую отдельную запись таблицы, другими словами, он создается по документам, у которых нет шапки и примечаний. Примером таких документов может служить приходный или расходный кассовый ордер, этикетка для товара или ценник в магазине, пригласительное письмо и т.д.

1. Создать новую форму (например, Form3). Связать эту форму с DM (в коде после ключевого слова USES добавить название соответстующего UNIT от модуля данных).

2. На форме разместить компонент QuickReport1. У свойства DataSet установить ссылку на ADOQUERY1. В свойствах включить такой раздел (свойство BANDS), как *HasDetail*.

3. В разделе Detail установить компоненты QREXPR **и** QRShape. В каждом компоненте QREXPR по очереди в свойстве Expression выбрать соответствующие поля запроса ADOQUERY1: title_book, name_author, publish и т.д.

4. В главной форме приложения в подпункте **Отчет в свободной форме** в методе Click написать команду: Form3.QuickReport1.Preview. В главной форме в коде добавить описание формы Form3, для этого после ключевого слова USES в конце списка через запятую дописать название программного модуля, которое соответствует форме.

5. Запустить приложение, проверить работу.

Задание 3. Создание отчета по двум таблицам. Создадим отчет с группировкой, в котором сначала будут выводиться данные автора книги из таблицы Authors, а затем список книг, которые написал этот автор.

1. Добавить на DM компоненты ADOTABLE1, ADOTABLE2 и DataSource1.

1.1. У ADOTABLE1 изменить следующие свойства (это будет главная таблица в отчете):

свойство Connection на ADOConnection1;

свойство TableName на Authors.

1.2. У DataSource1 изменить следующее свойство (это будет ссылка на главную таблицу):

свойство DataSet на ADOTABLE1.

1.3. У ADOTABLE2 изменить следующие свойства (это будет главная таблица в отчете):

свойство Connection на ADOConnection1;

свойство TableName на Books;

свойство MasterSource на DataSource1;

свойство Masterfields на Code_Author (ссылка на ключевое поле в главной таблице Authors);

свойство IndexFieldName на Code_Author (ссылка на поле связи в зависимой таблице Books).

1.4. Активизировать таблицы.

2. Создать новую форму (например, Form4). Связать эту форму с DM (в коде после ключевого слова USES добавить название соответстующего UNIT от модуля данных).

3. На форме разместить компонент QuickReport1. У свойства DataSet установить ссылку на ADOTABLE1. В свойствах включить такие разделы (BANDS), как *HasTitle, HasColumnHeader, HasDetail*.

4. После раздела *HasDetail* поместить с панели инструментов Qreport элемент HasSubDetail.

5. В разделе Title QuickReport1 разместить метку (компонент QRLabel). В свойствах изменить ее внешний вид и подпись «Отчет по авторам и написанным книгам».

6. В разделе ColumnHeader установить компоненты QRShape (для имитации обрамления шапки таблицы) и QRLabel (написать в нем **ФИО автора**).

7. В разделе Detail установить 1 компонент QREXPR. В свойстве Expression выбрать соответствующие поле ADOTABLE1: name_author. Ниже расположить компоненту QRLabel, в свойствах изменить ее внешний вид и подпись «Названия книг данного автора».

8. В разделе SubDetail изменить свойство DataSet на ADOTABLE2, а свойство LinkBand на DetailBand1. Установить 1 компонент QREXPR. В свойстве Expression выбрать соответствующие поле ADOTABLE2: title_book. Кроме того, отключить свойство авторазмера и растянуть поле до конца раздела SubDetail. Пример представлен на рис. 7.3.

9. В главной форме приложения в подпункте **Отчет с группировкой** в методе Click написать команду: Form4.QuickReport1.Preview. В главной форме в коде добавить описание формы Form4, для этого после ключевого слова USES в конце списка через запятую дописать название соответствующего форме программного модуля.

10. Запустить приложение, проверить работу.

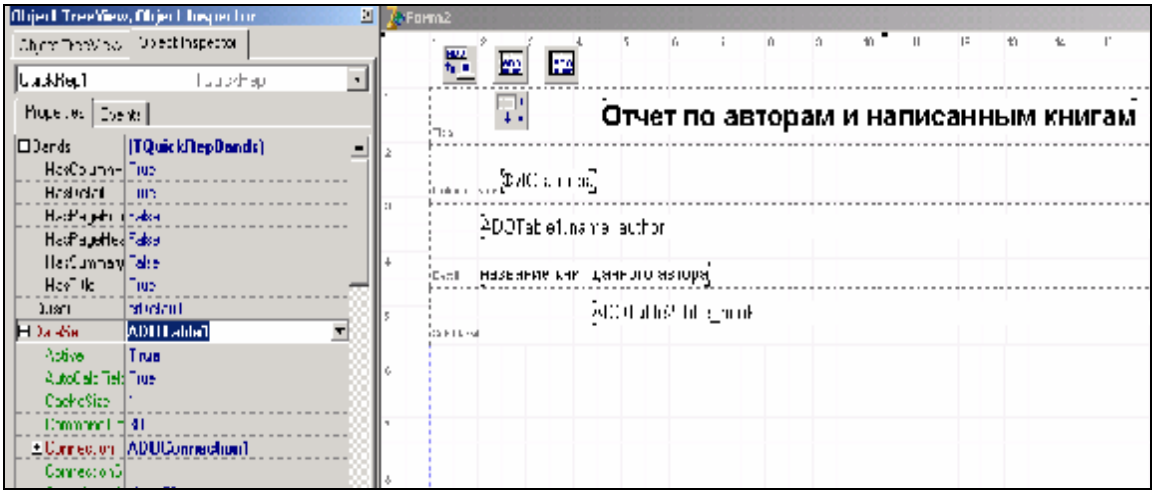

Рис. 7.3. Пример отчета

Варианты заданий к лабораторной работе №7

По индивидуальной базе данных, которая выдана по вариантам (из лабораторной работы №1) сделать в клиентском приложении четыре отчета, которые будут запускаться через меню главной формы:

- отчет в табличной форме по одному из справочников, причем в разделе «Примечание» вывести итоговое количество записей в отчете;

- отчет в свободной форме. Выберите одну из таблиц, по которой можно сделать или бейдж, или ценник, или пригласительный билет. При создании отчета используйте рисунок в качестве подложки;

- отчет по запросу. Соедините данные всех трех таблиц, кодовые поля в запрос не помещайте. Создайте отчет в табличной форме с итоговым полем в разделе «Примечание» (это может быть сумма или количество и т.п., в зависимости от содержания запросов). Каждая строка в отчете должна иметь номер по порядку. Например (см. фрагмент ниже):

1) Крупа 10 кг
2) Мука 20 кг
и т.д.;

- отчет с группировкой по нескольким таблицам. Выберите одну пару связанных таблиц, определите главную и зависимую таблицы и сделайте отчет в табличной форме, в котором данные из главной таблицы расшифровываются (дополняются) данными из зависимой таблицы.

Цель задания – создание единого приложения для ввода/ вывода данных и удаленного управления доступом к БД.

Библиографический список

1. Интернет-институт информационных технологий.– www.intuit.ru.– Курс «Основы SQL».
2. *Мамаев Е.В.* Microsoft SQL Server 2000.– СПб.: БХВ-Петербург, 2005.– 1280 с.
3. *Остринская Л.И., Семенова И.И., Дороболюк Т.Б.* Теория и практика работы с современными базами и банками данных: Учебное пособие. – Омск: Изд-во СибАДИ, 2005.– 250 с.
4. *Семенова И.И.* Сборник упражнений по стандарту SQL. – Омск: Изд-во СибАДИ, 2005.– 43 с.
5. *Шкрыль А.А.* Разработка клиент-серверных приложений в Delphi.– СПб.: БХВ-Петербург, 2006.– 480 с.

Содержание

ОБЩИЕ ПОЛОЖЕНИЯ 3

Лабораторная работа №1 4
СОЗДАНИЕ БАЗ ДАННЫХ (БД) В MICROSOFT SQL SERVER 4

Лабораторная работа №2 12
ИСПОЛЬЗОВАНИЕ ОПЕРАТОРОВ МАНИПУЛИРОВАНИЯ ДАННЫМИ В MICROSOFT SQL SERVER 12

Лабораторная работа №3 19
ОСВОЕНИЕ ПРОГРАММИРОВАНИЯ С ПОМОЩЬЮ ВСТРОЕННОГО ЯЗЫКА TRANSACT SQL **В** MICROSOFT SQL SERVER 19

Лабораторная работа №4 35
СОЗДАНИЕ ХРАНИМЫХ ПРОЦЕДУР В MICROSOFT SQL SERVER 35

Лабораторная работа №5 47
СОЗДАНИЕ КЛИЕНТСКОЙ ЧАСТИ ПРИЛОЖЕНИЯ ДЛЯ ПРОСМОТРА, РЕДАКТИРОВАНИЯ ДАННЫХ БД. ВЫЗОВ ХРАНИМЫХ ПРОЦЕДУР ИЗ КЛИЕНТСКОЙ ЧАСТИ 47

Лабораторная работа №6 53
СОЗДАНИЕ АДМИНИСТРАТИВНОЙ СТРАНИЦЫ 53

Лабораторная работа №7 56
СОЗДАНИЕ ОТЧЕТНЫХ ФОРМ В КЛИЕНТСКОМ ПРИЛОЖЕНИИ 56

Библиографический список 62

Учебное издание

Ирина Ивановна Семенова

РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНЫХ ПРИЛОЖЕНИЙ В MICROSOFT SQL SERVER 2000 И BORLAND DELPHI 7

Учебно-методическое пособие

***

Редактор Т.И. Калинина

***

Подписано к печати___.___.2007
Формат 60x90 1/16. Бумага писчая
Оперативный способ печати
Гарнитура Таймс
Усл.п. л.___, уч.-изд. л. ____
Тираж 100 экз. Заказ № _____
Цена договорная

Издательство СибАДИ
644099, г. Омск, ул. П. Некрасова, 10

Отпечатано в ПЦ издательства СибАДИ
644099, г. Омск, ул. П. Некрасова, 10

64