

Министерство образования Республики Беларусь  
Учреждение образования «Витебский государственный университет имени  
П.М. Машерова»

***Модели данных и СУБД  
Практикум***

*Витебск  
2012*

УДК 004.652(0758)  
ББК 32.973.26-018.2я73  
А28

Автор: доцент кафедры информатики и ИТ УО "ВГУ им. П.М. Машерова", кандидат педагогических наук" **Н.Д.Адаменко**

Рецензенты:

кафедра прикладной математики и механики УО "ВГУ им. П.М. Машерова", заведующий кафедрой прикладной математики и механики УО "ВГУ им. П.М. Машерова", кандидат физико-математических наук С.А. Ермоченко

**Адаменко Н.Д.**

*А28 Модели данных и СУБД: Практикум / Н.Д.Адаменко.- Витебск.; УО "ВГУ им. П.М. Машерова", 2012. - 150 с.*

Учебное пособие содержит описание технологии разработки баз данных с помощью СУБД MS SQL Server, а также методические материалы для проведения лабораторных занятий, последовательно формирующих основные умения, необходимые для эффективной работы с СУБД MS SQL Server. Материалы пособия могут найти применение при изучении курсов, связанных с освоением способов разработки многопользовательских систем баз данных.

**ISBN**

УДК 004.652(0758)  
ББК 32.973.26-018.2я73

© Адаменко Н.Д., 2012

© УО "ВГУ им. П.М. Машерова",  
2012

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	4
ЛАБОРАТОРНАЯ РАБОТА № 1 Тема: Установка "Microsoft SQL Server 2008" .....	5
ЛАБОРАТОРНАЯ РАБОТА № 2 Тема: Проектирование базы данных. ....	26
ЛАБОРАТОРНАЯ РАБОТА № 3 Тема: Создание базы данных. Типы данных, используемые в SQL-сервере. Создание пользовательских типов данных. ...	39
ЛАБОРАТОРНАЯ РАБОТА №4 Тема: Создание объектов базы данных: таблиц, индексов и первичных ключей таблицы. Использование ограничений. .....	49
ЛАБОРАТОРНАЯ РАБОТА № 5 Тема: Использование диаграмм для разработки структуры базы данных. Ввод данных в таблицу. Модификация данных таблиц. ....	60
ЛАБОРАТОРНАЯ РАБОТА № 6 Тема: Создание запросов на выборку и модификацию данных.....	64
ЛАБОРАТОРНАЯ РАБОТА № 7 Тема: Создание представлений, триггеров, хранимых процедур.....	84
ЛАБОРАТОРНАЯ РАБОТА № 8 Тема: Администрирование баз данных. Резервное копирование. ....	100
Л И Т Е Р А Т У Р А .....	109
Приложение 1 Варианты индивидуальных заданий.....	110
Приложение 2 Типы данных, используемые в SQL-сервере.....	139
Приложение 3 Типы ограничений.....	141
Приложение 4 Данные для заполнения таблиц.....	142
Приложение 5 Краткое определение основных терминов.....	145
Приложение 6 Дополнительные параметры создаваемого индекса.....	146
Приложение 7 Список функций SQL Server .....	147

## ВВЕДЕНИЕ

В связи с широким распространением корпоративных информационных систем актуальной становится задача подготовки специалистов в области распределенной обработки данных. В пособии рассмотрены теоретические основы и методы, на которых базируется практика разработки современных многопользовательских систем обработки информации на базе технологии клиент-сервер. Основное внимание уделено технологии проектирования баз данных с помощью ER-диаграмм и способам создания объектов базы – таблиц, запросов, хранимых процедур, триггеров. В процессе выполнения заданий, приведенных в пособии, у обучаемых формируется представление о способах поддержки целостности данных. Рассматриваются программные средства автоматизированного контроля соответствия базы данных постоянно изменяющемуся состоянию предметной области.

Пособие содержит систему лабораторных работ, последовательное выполнение которых обеспечивает формирование устойчивых умений проектирования, построения баз данных с распределенной обработкой. В технологии разработки многопользовательских информационных систем значительное внимание уделяется безопасности данных при их совместной обработке баз данных, поэтому в пособии рассматриваются вопросы администрирования баз данных. В частности – управление правами доступа и привилегиями, а также резервное копирование данных и восстановление после сбоев.

Каждая работа включает ряд заданий, выполнение которых способствует освоению основных возможностей СУБД MS SQL Server. Пособие может быть использовано для самостоятельного освоения приложения. В этом случае необходимо проработать теоретический материал, а затем выполнить один или несколько вариантов заданий.

# ЛАБОРАТОРНАЯ РАБОТА № 1

## Тема: Установка "Microsoft SQL Server 2008"

**Цель работы:** Изучение основных этапов установки и настройки "Microsoft SQL Server".

### ОСНОВНЫЕ СВЕДЕНИЯ

Для установки MS SQL Server необходимо наличие установленных пакетов: Microsoft .NET Framework 3.5 SP1, Windows Installer 4.5 и Windows PowerShell 1.0.

1. Запустить программу-установщик (в бесплатной версии Express Edition обычно называется SQLEXPRESS\_x86\_RUS.exe) с правами администратора на данном компьютере.

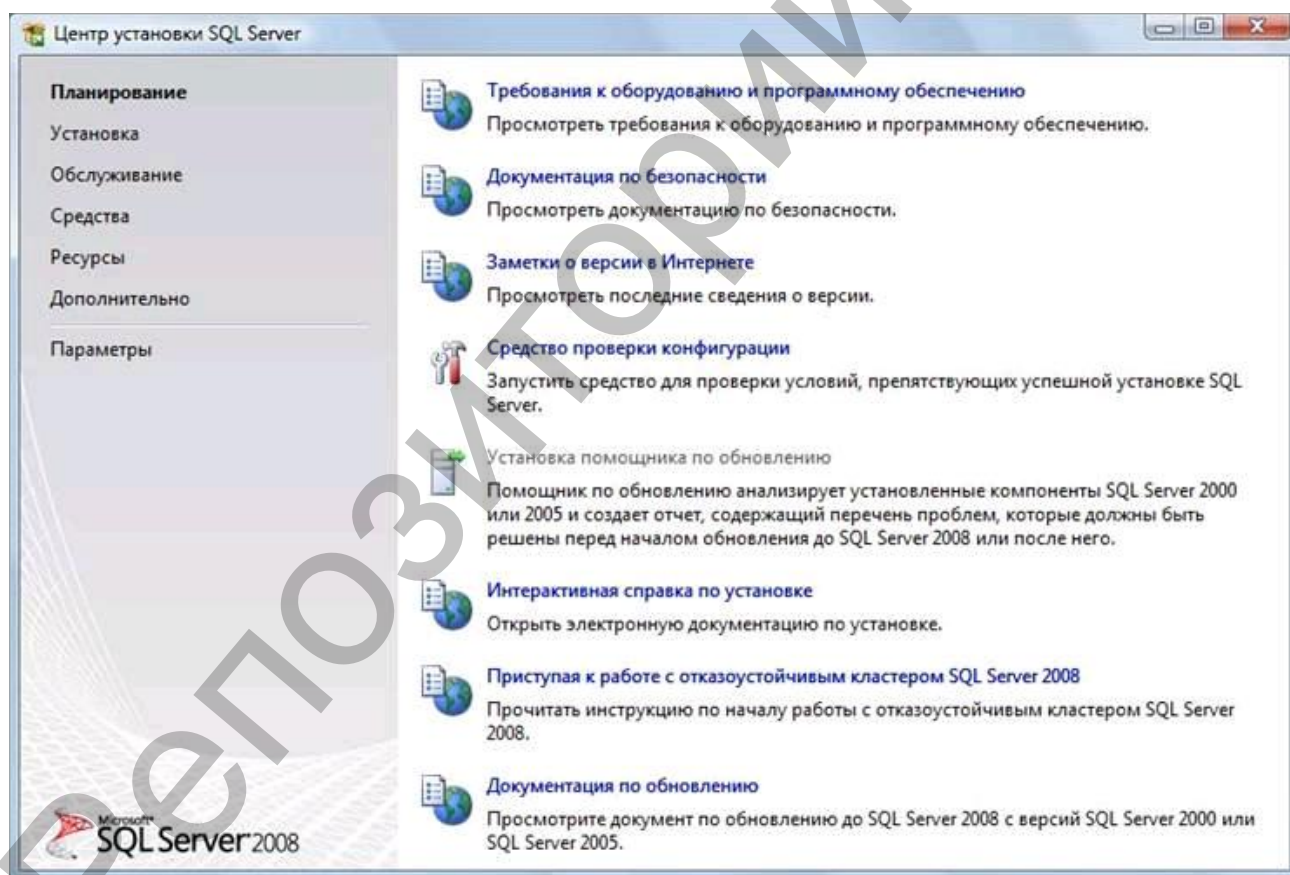


Рис. 1

2. В разделе «Планирование» нажать пункт «Средство проверки конфигурации»:

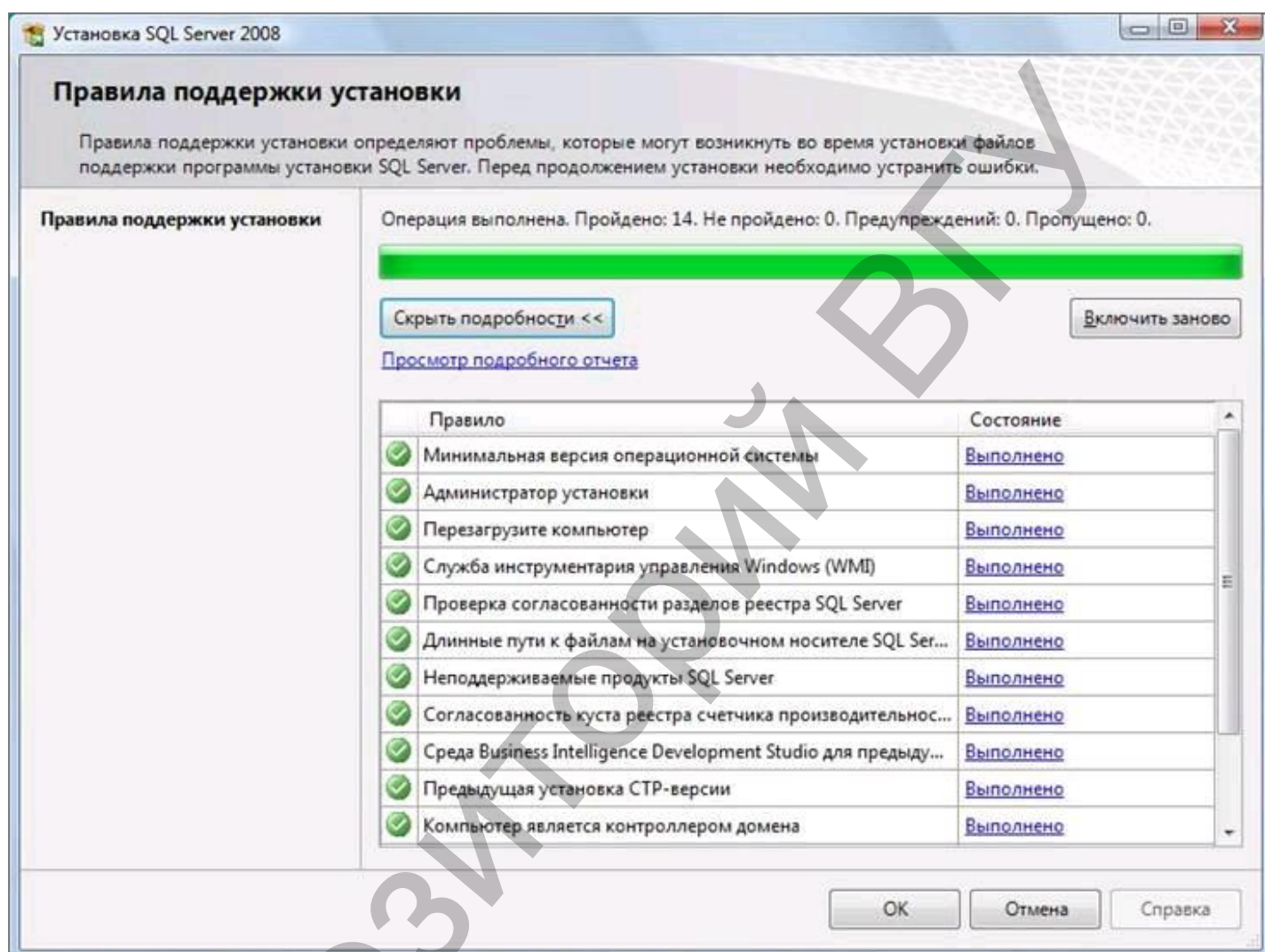


Рис. 2

3. Нажать кнопку «Показать подробности» и убедиться, что все проверки успешно пройдены. Если будут обнаружены какие-то проблемы, то необходимо их устранить и запустить повторную проверку кнопкой

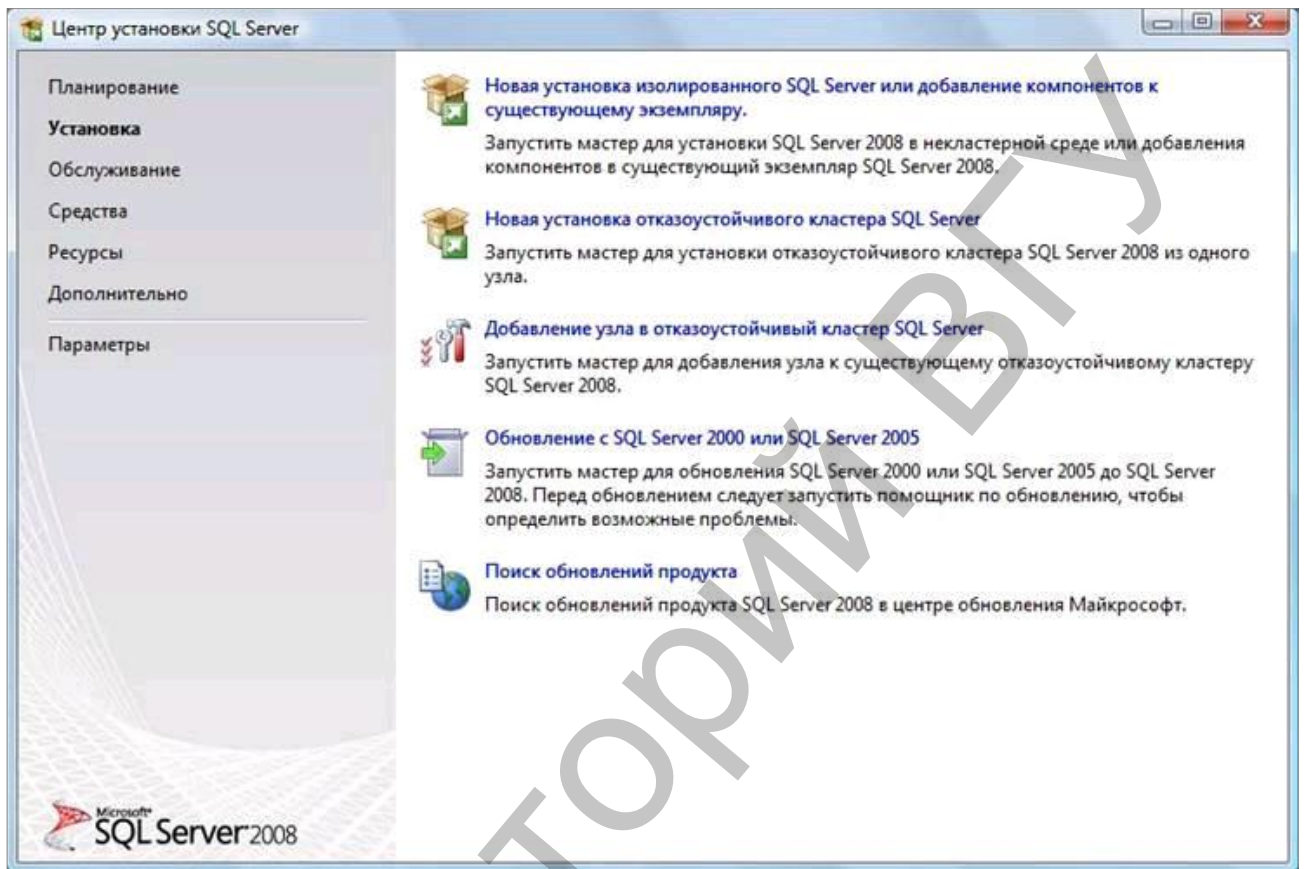


Рис. 3

«Включить заново». Затем закрыть данное окно кнопкой «ОК»:

4. Нажать на раздел «Установка» и затем пункт «Новая установка изолированного SQL Server или добавление компонентов ...»:

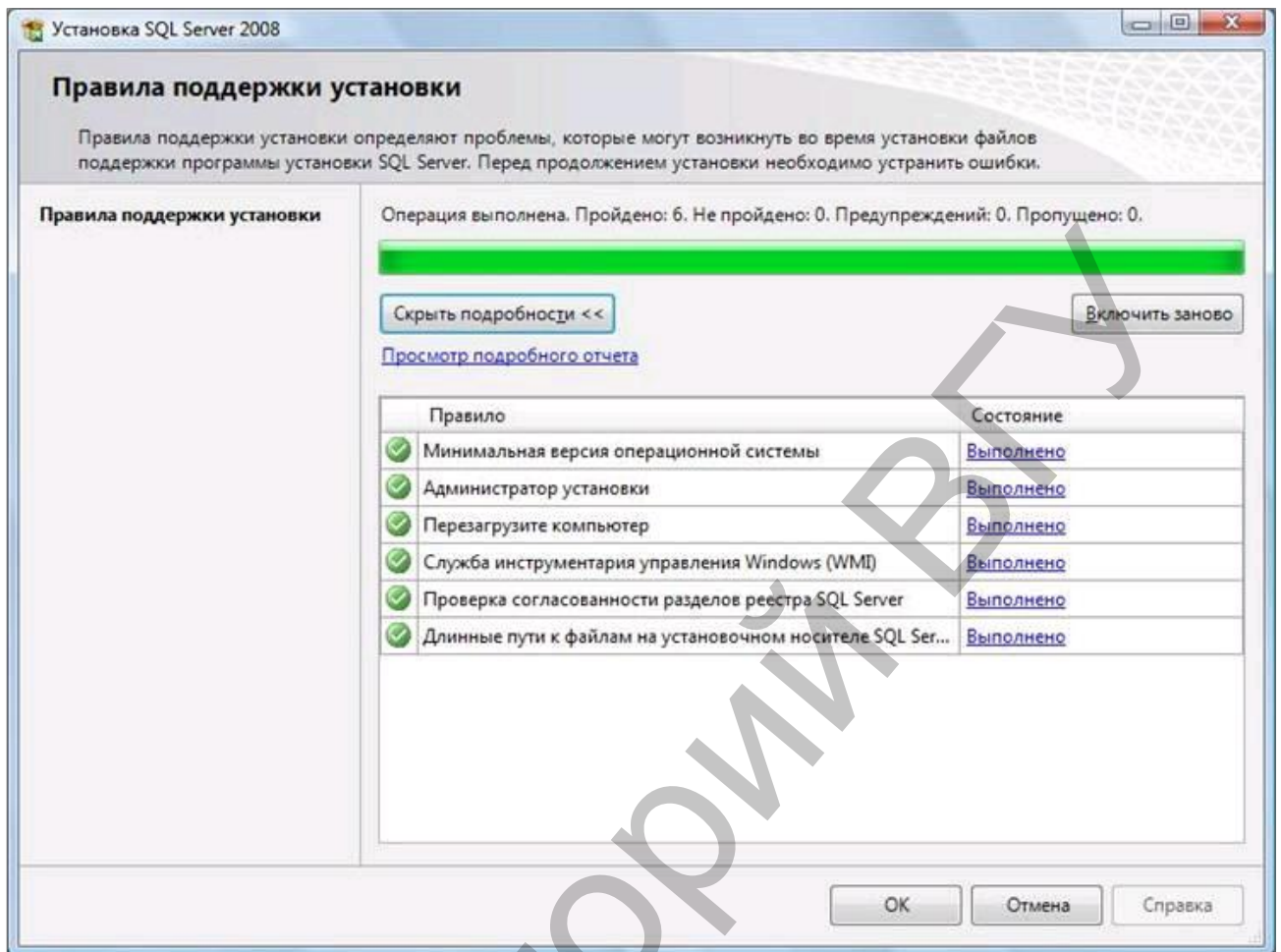


Рис. 4

5. Нажать кнопку «Показать подробности» и убедиться, что все проверки успешно пройдены. Если будут обнаружены какие-то проблемы, то необходимо их устранить и запустить повторную проверку кнопкой «Включить заново».

Затем нажать кнопку «ОК»:



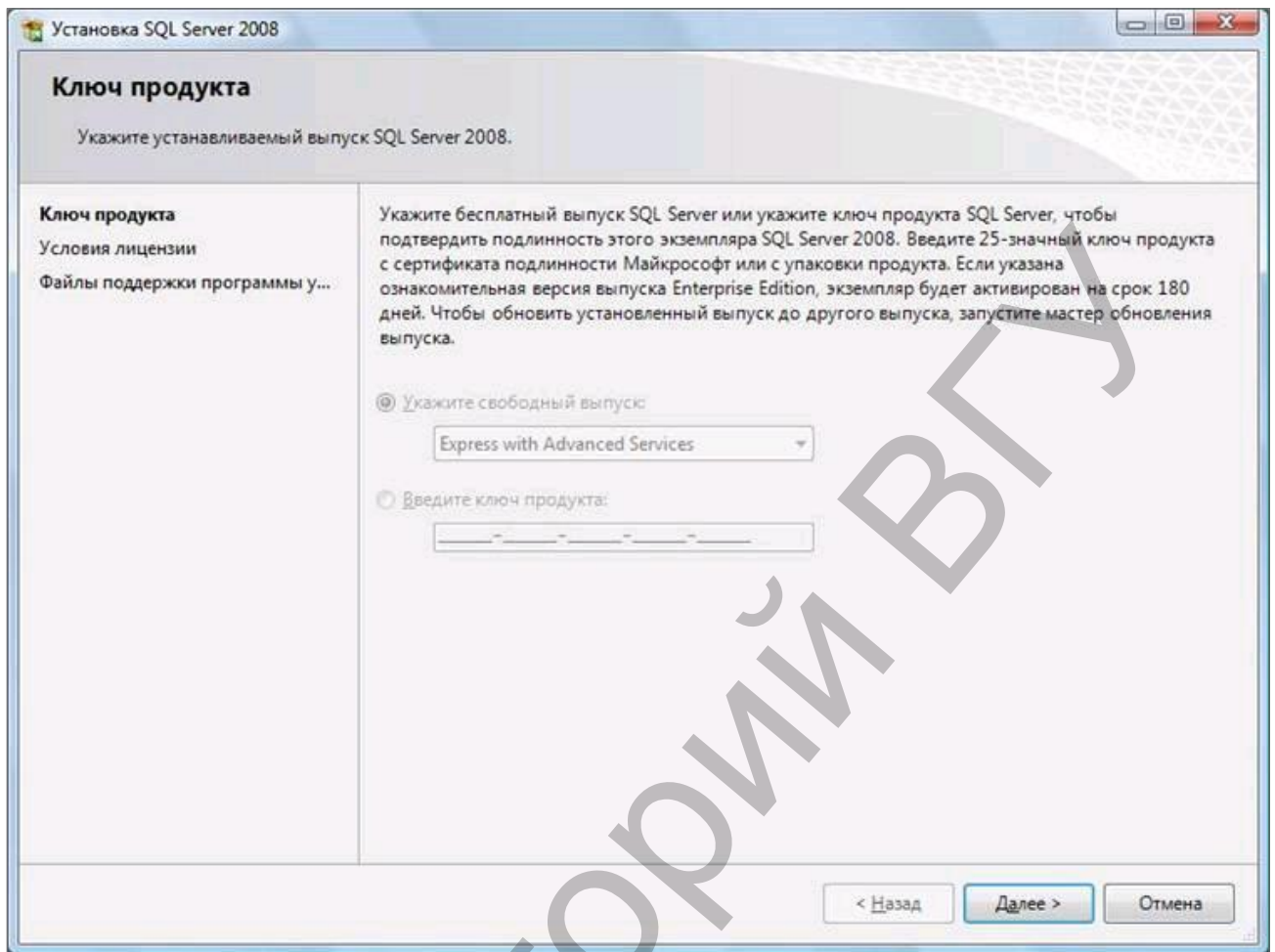


Рис. 5

6. Ввести приобретенный ключ продукта (для бесплатной версии не требуется) и нажать кнопку «Далее»:

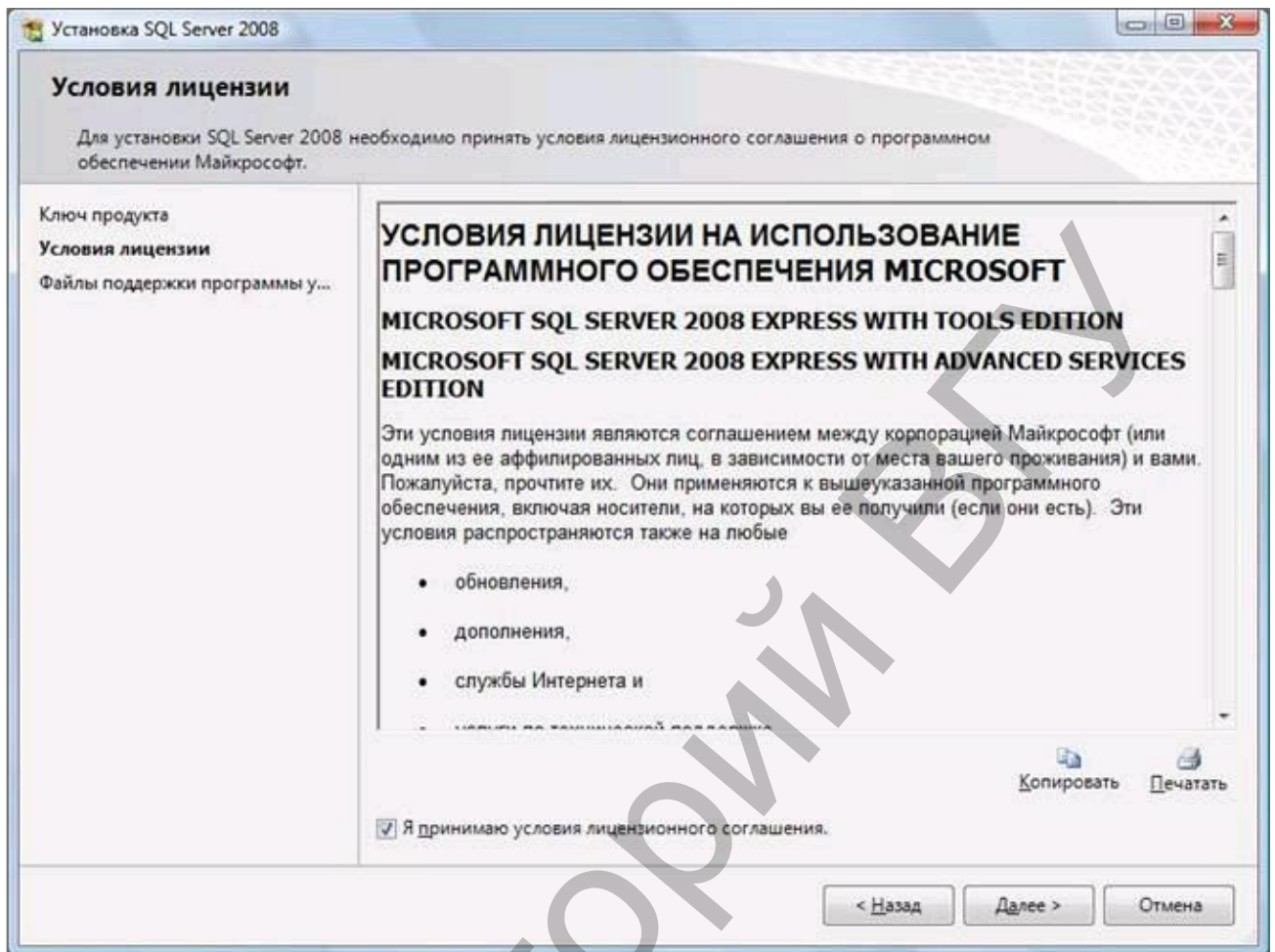


Рис. 6

7. Прочитать лицензию, установить галочку и нажать кнопку «Далее»:

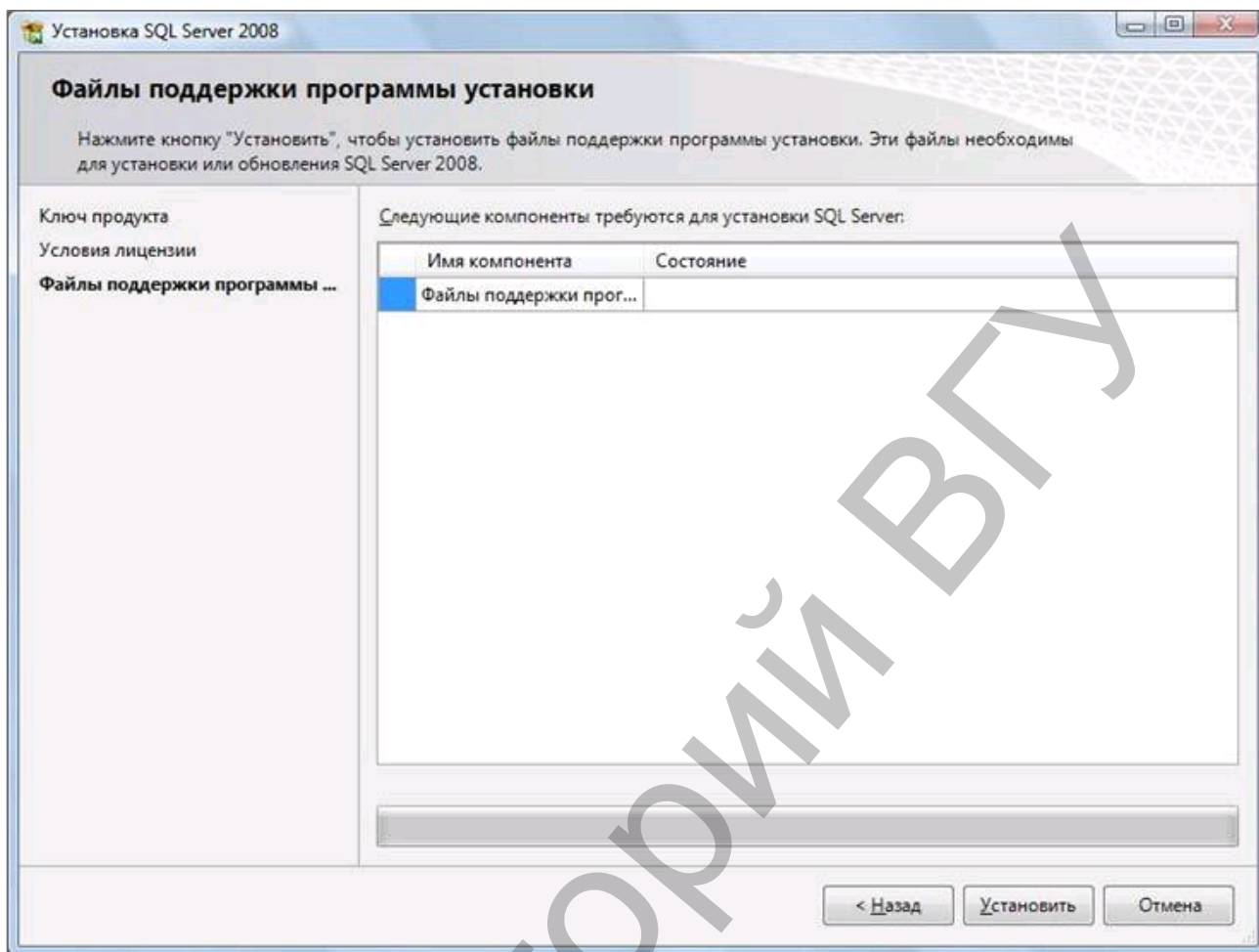


Рис. 7

8. Нажать кнопку «Установить»:

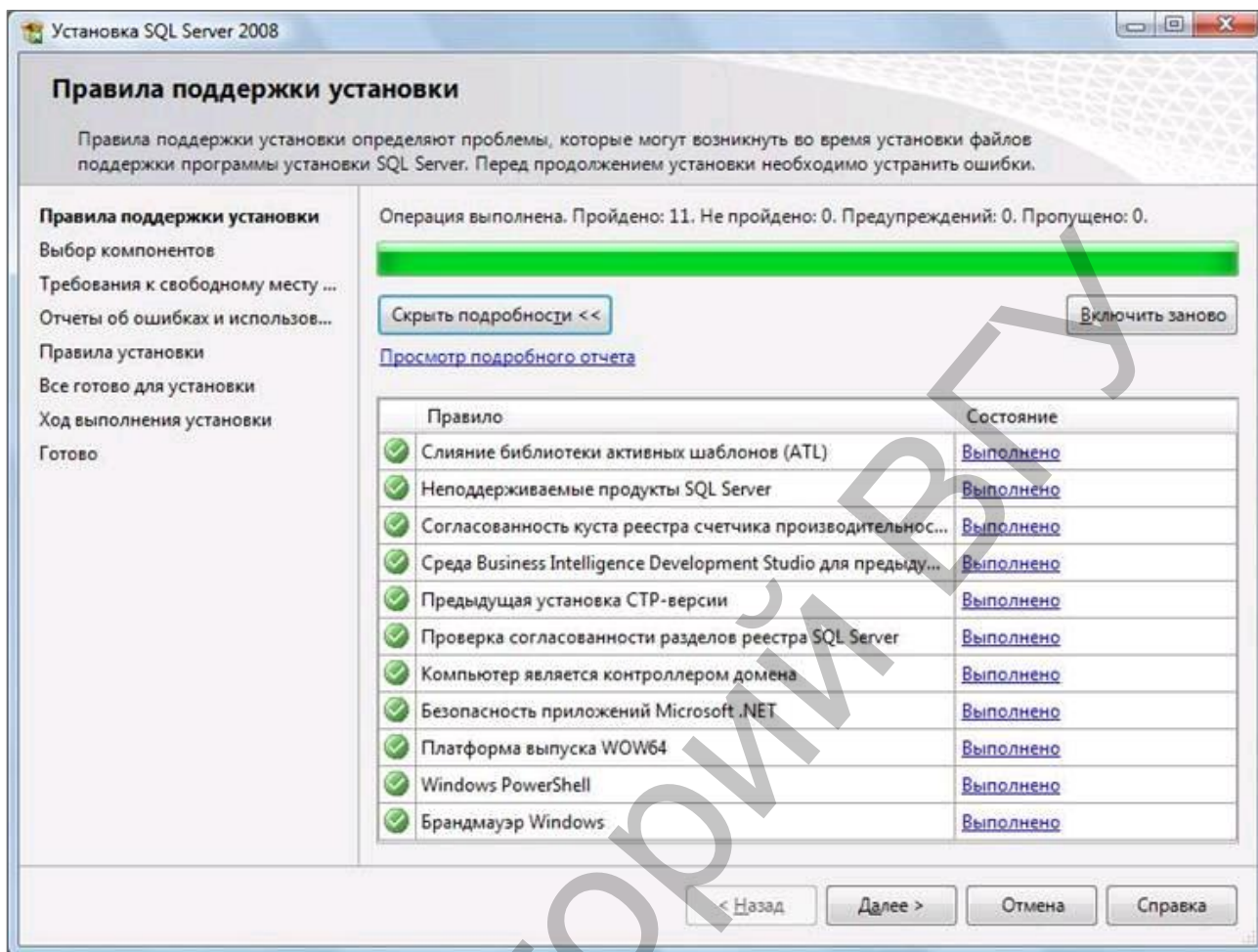


Рис. 8

9. Нажать кнопку «Показать подробности» и убедиться, что все проверки успешно пройдены. Если будут обнаружены какие-то проблемы, то необходимо их устранить и запустить повторную проверку кнопкой «Включить заново». Затем нажать кнопку «Далее»:

Если появится предупреждение в строке «Брандмауэр Windows», то его можно проигнорировать.

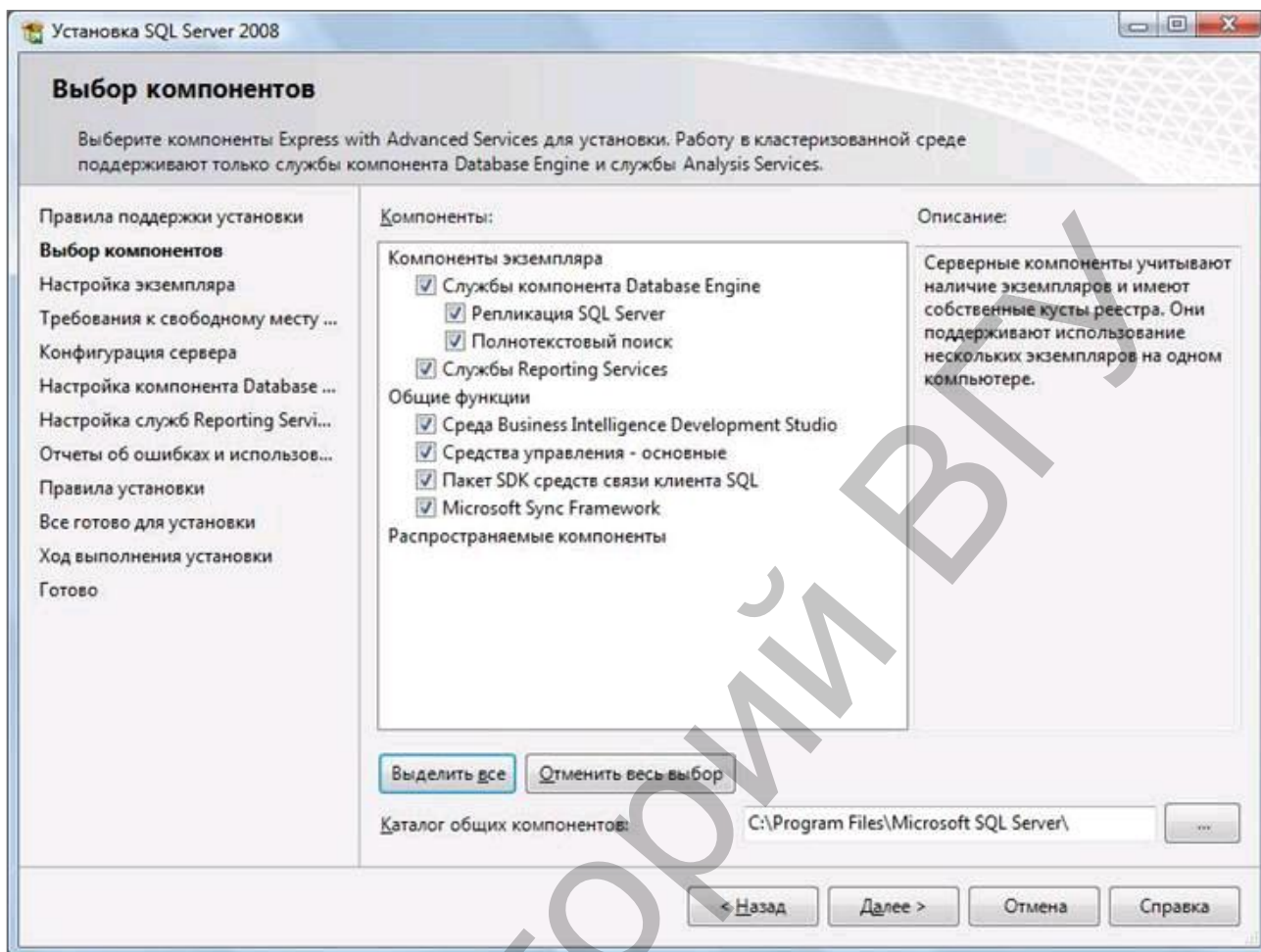


Рис. 9

10. Выбрать компоненты для установки (можно воспользоваться кнопкой «Выделить все»), и нажать кнопку «Далее»: для управления самим SQL Server на этом этапе следует установить компонент «Средства управления - основные».

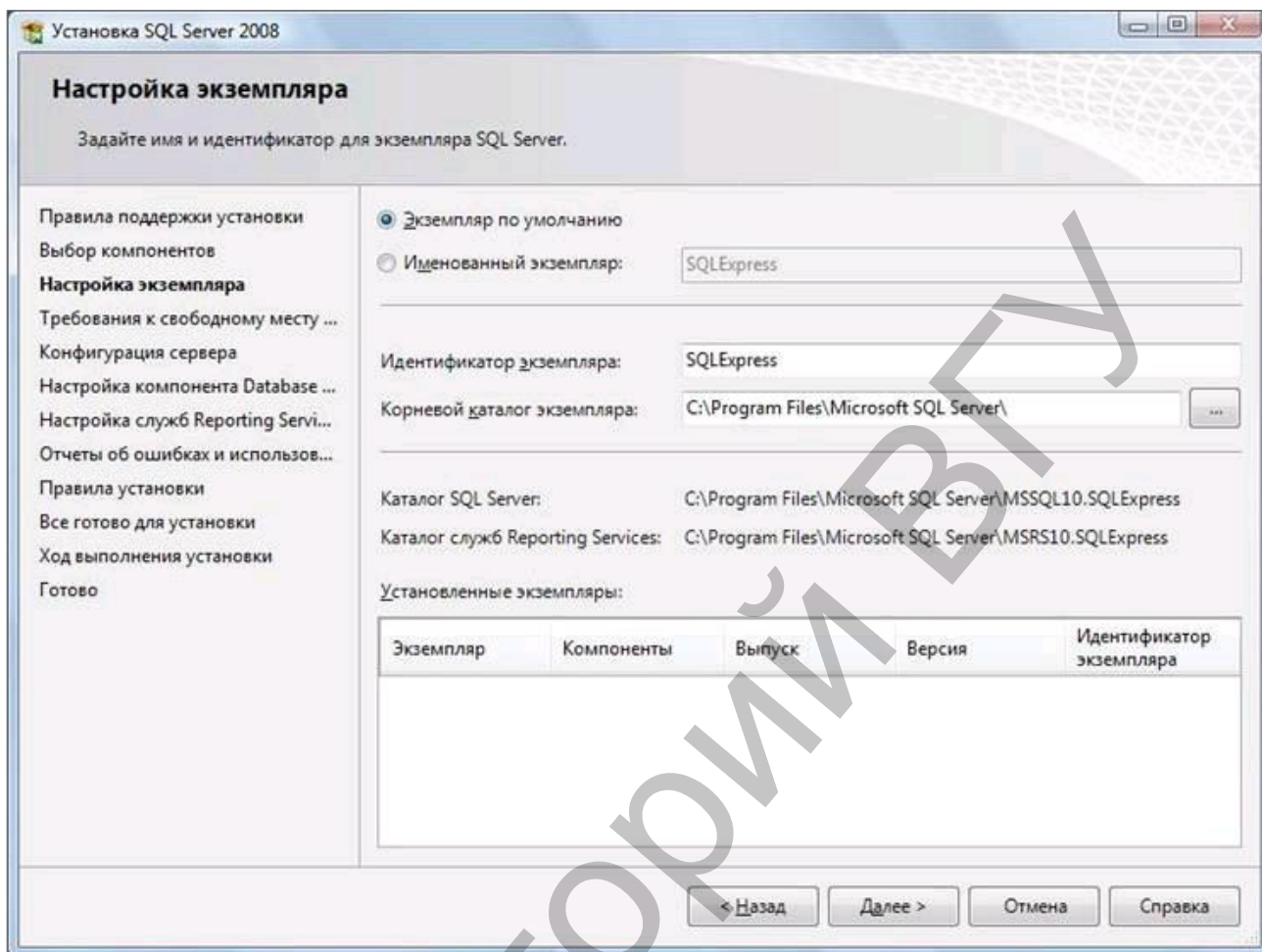


Рис. 10

11. Выбрать опцию «Экземпляр по умолчанию» и нажать кнопку «Далее»:

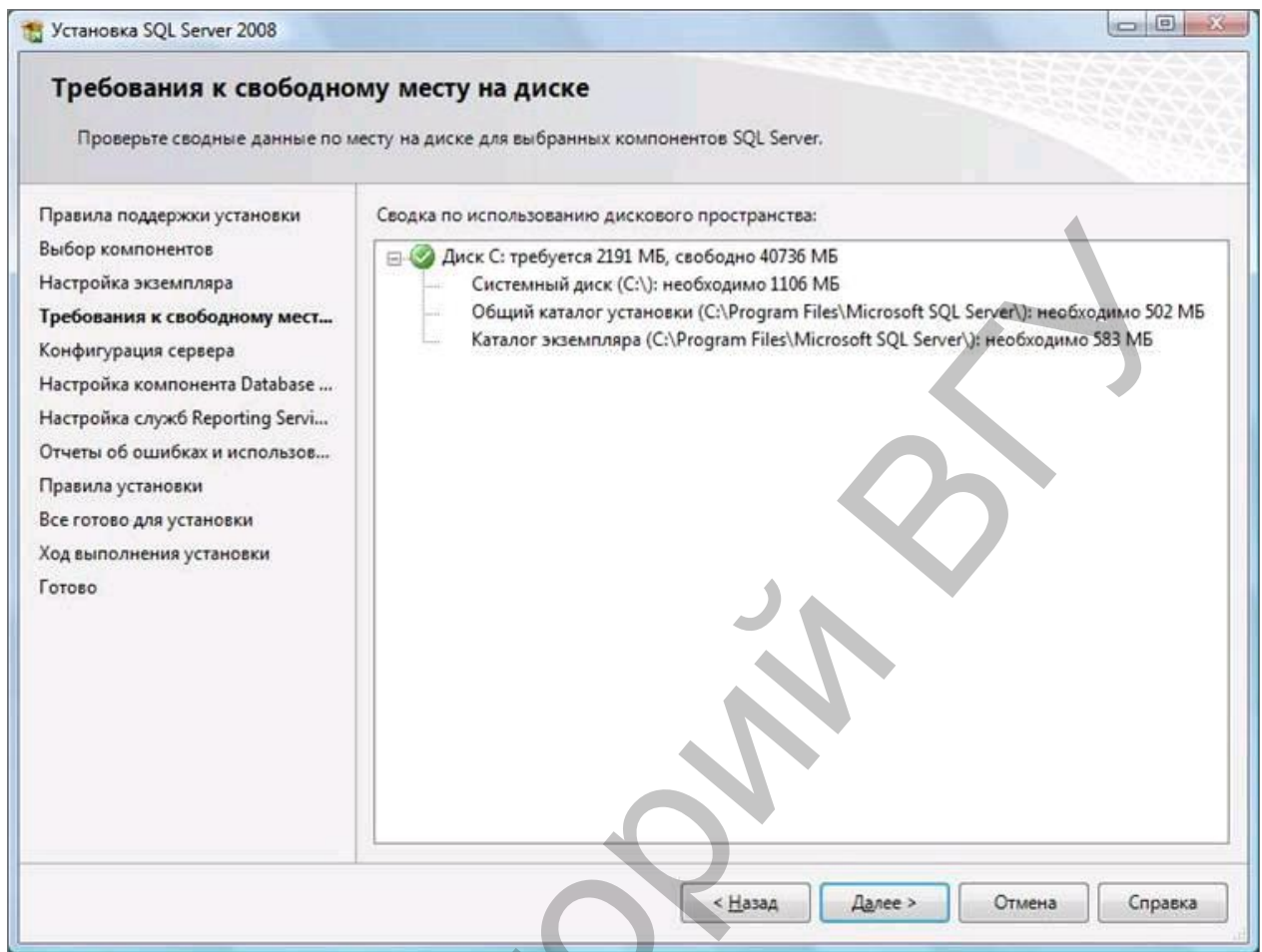


Рис. 11

12. Нажать кнопку «Далее»:

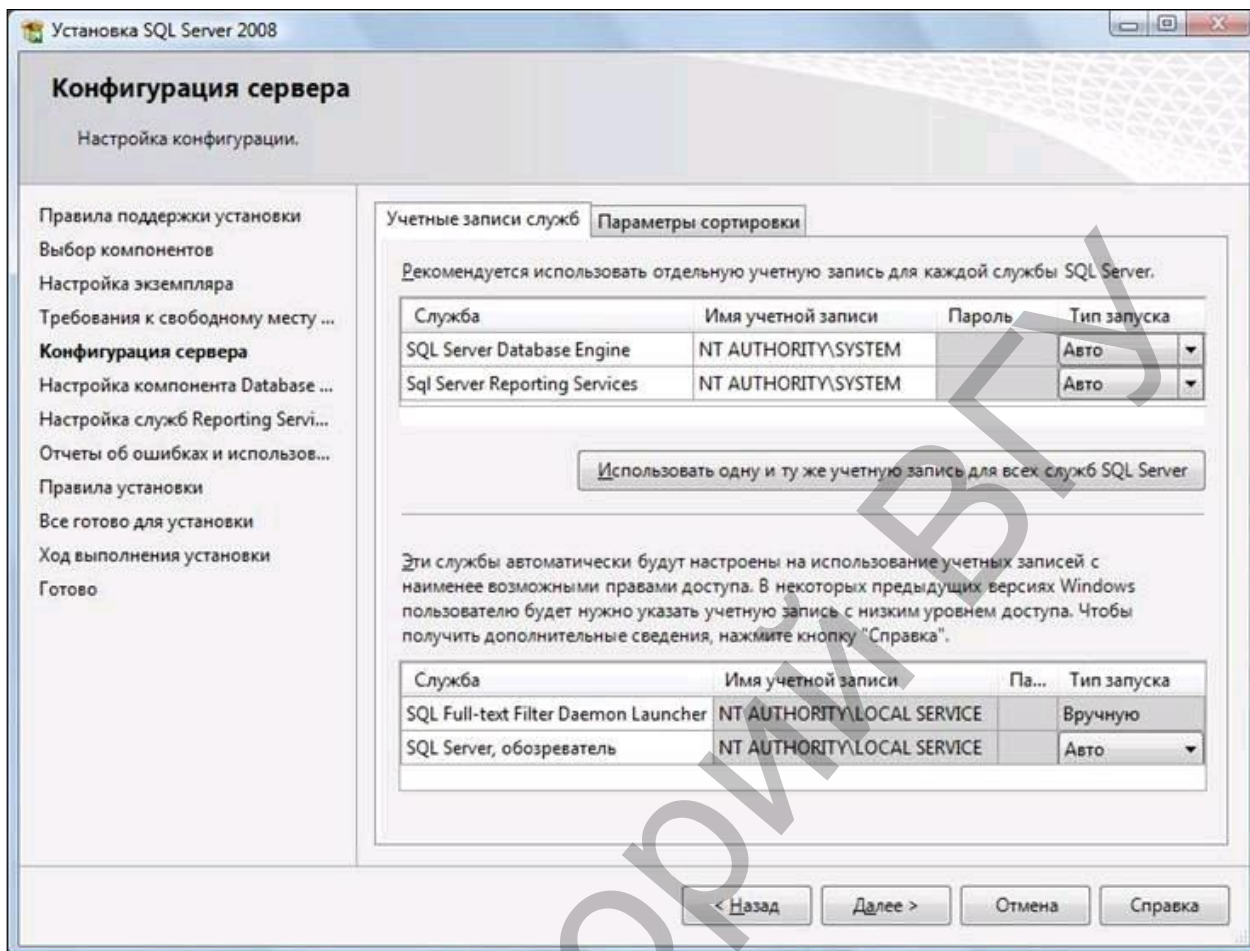


Рис. 12

13. Выбрать опции, как показано на рисунке 12, и перейти на закладку «Параметры сортировки»:



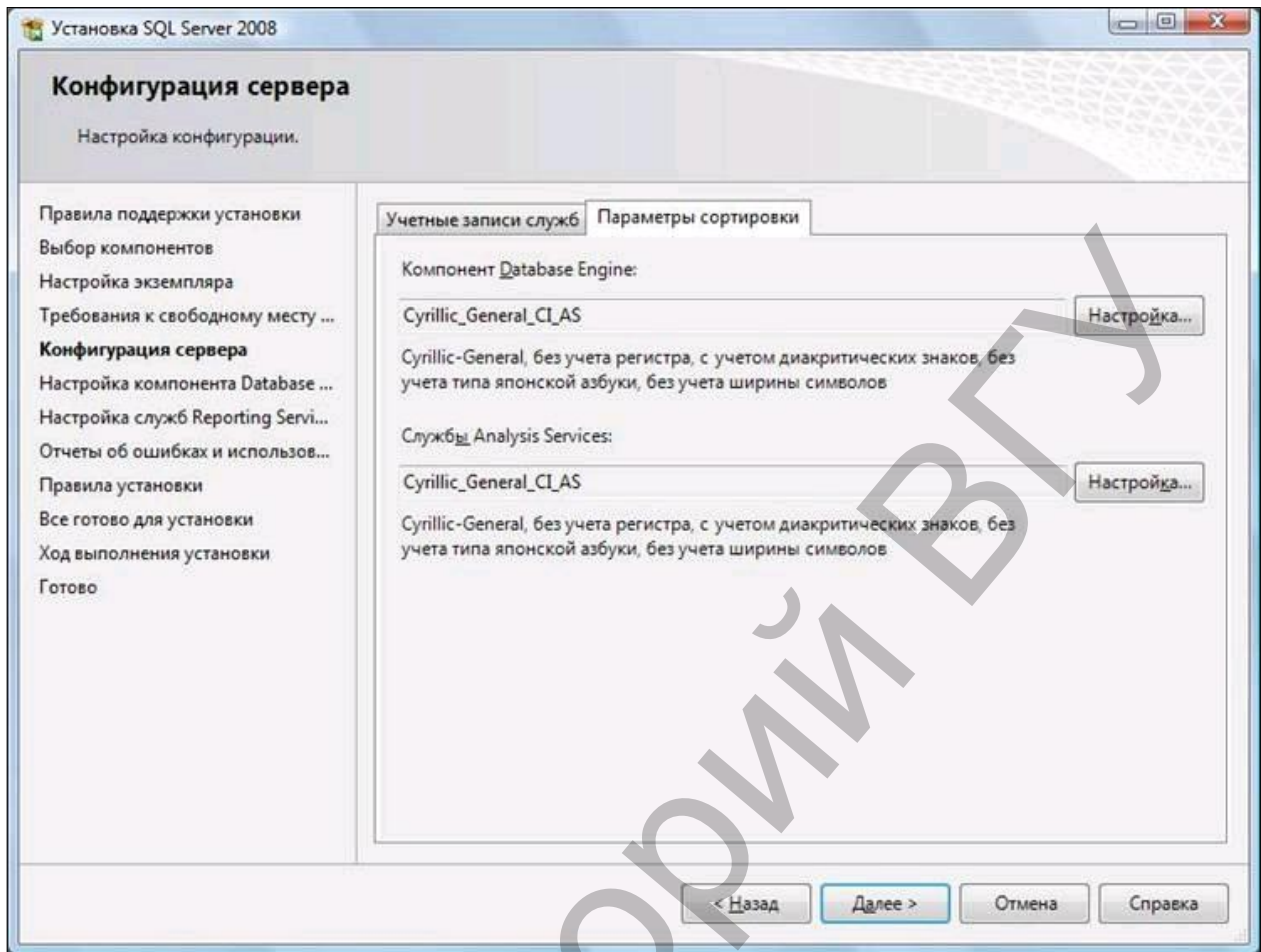


Рис. 13

14. Выбрать опции, как показано на рисунке 13, и нажать кнопку «Далее». Чтобы изменить опцию нажать расположенную рядом кнопку «Настройка» и установите параметры, как показано на рисунке 14:

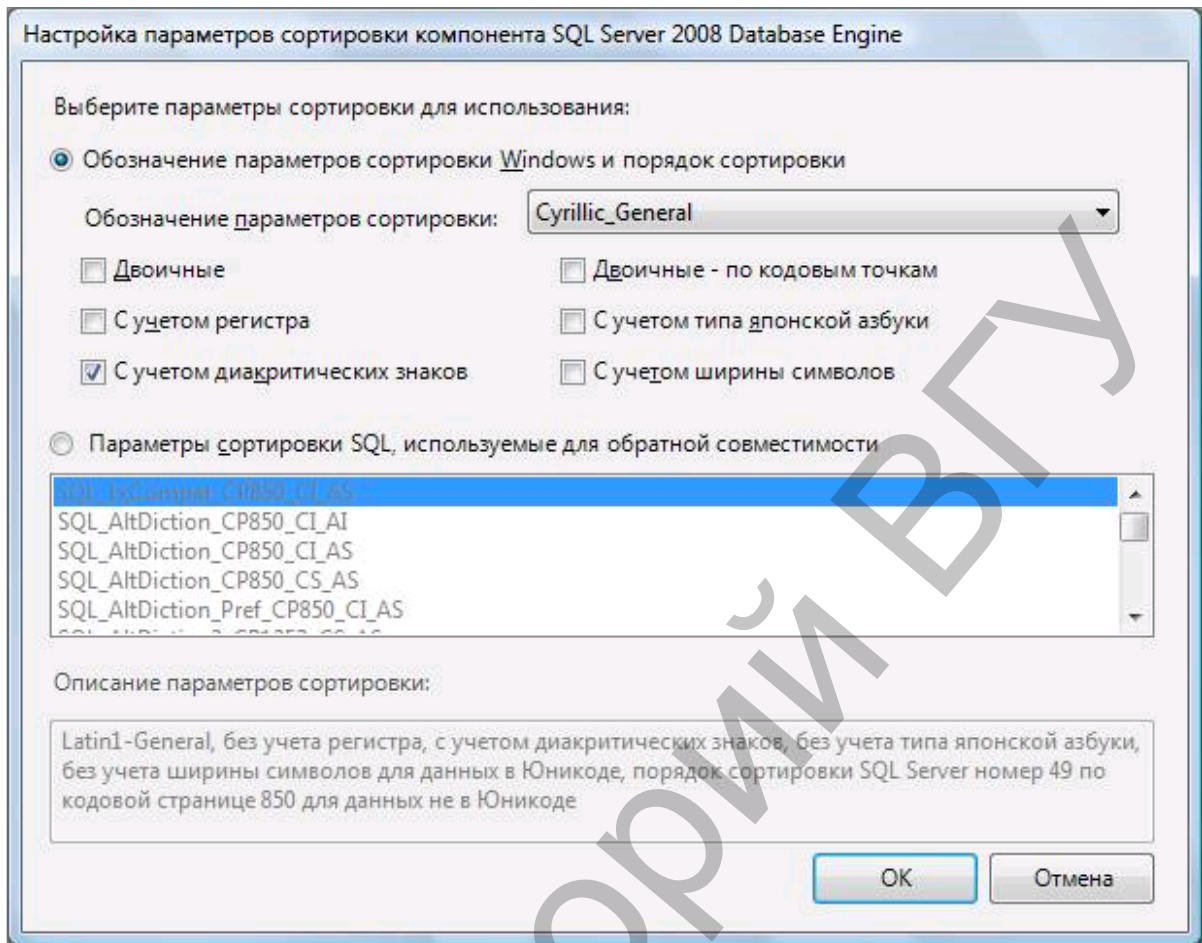


Рис. 14

Данную настройку нельзя будет изменить после установки.

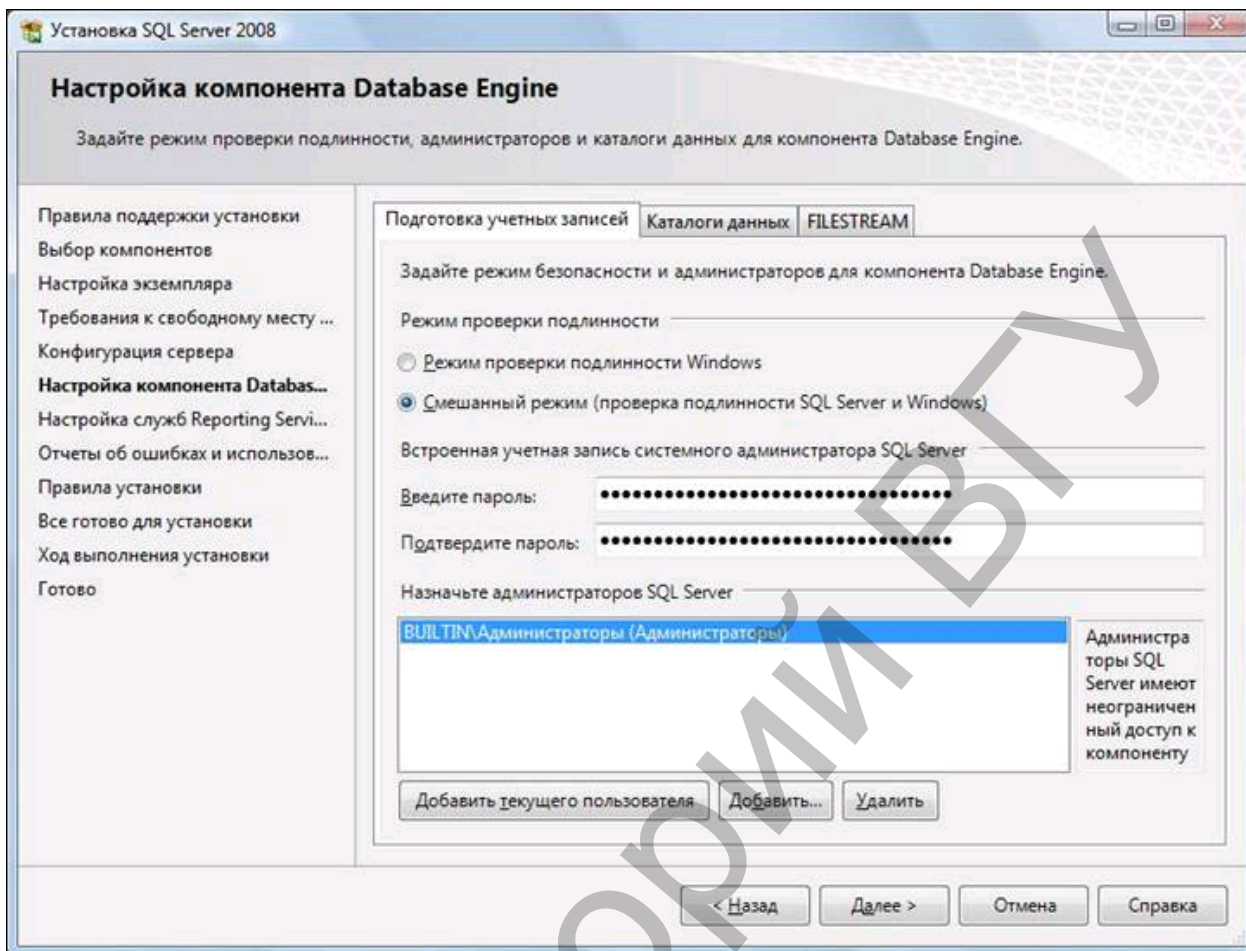


Рис. 15

15. Выбрать опцию «Смешанный режим» и задать пароль (12345) для встроенной учетной записи администратора «sa» (эта учетная запись обладает максимальными правами доступа ко всем функциям и объектам на SQL-сервере). Дополнительно можно задать учетные записи пользователей Windows или группы пользователей Windows, которые должны обладать максимальными правами доступа к SQL Server. Затем перейти на закладку «Каталоги данных»:

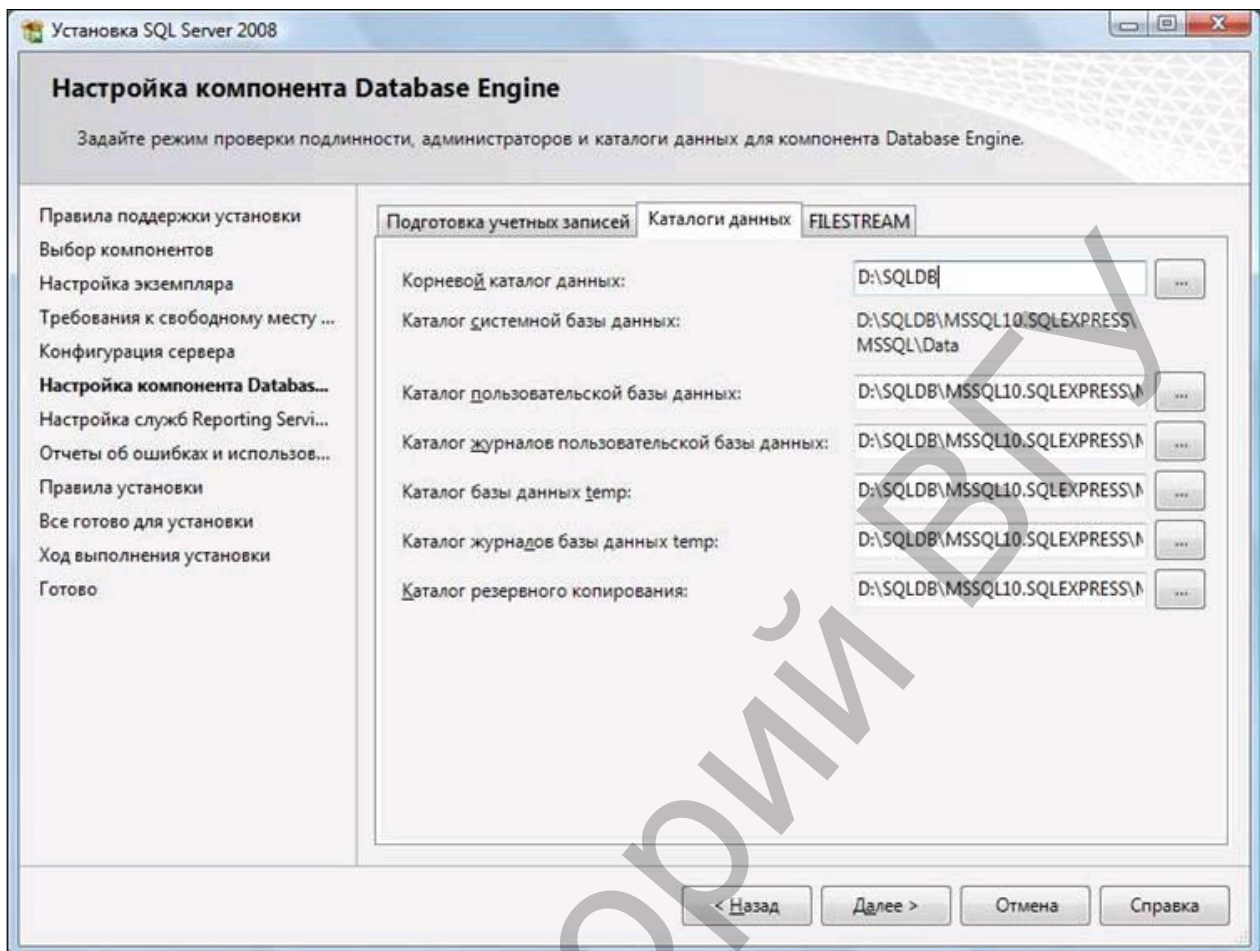


Рис. 16

16. В поле «Корневой каталог данных» ввести путь к папке, где будут размещаться файлы баз данных (**Ввести путь D\SQLDB**), и нажать кнопку «Далее»:

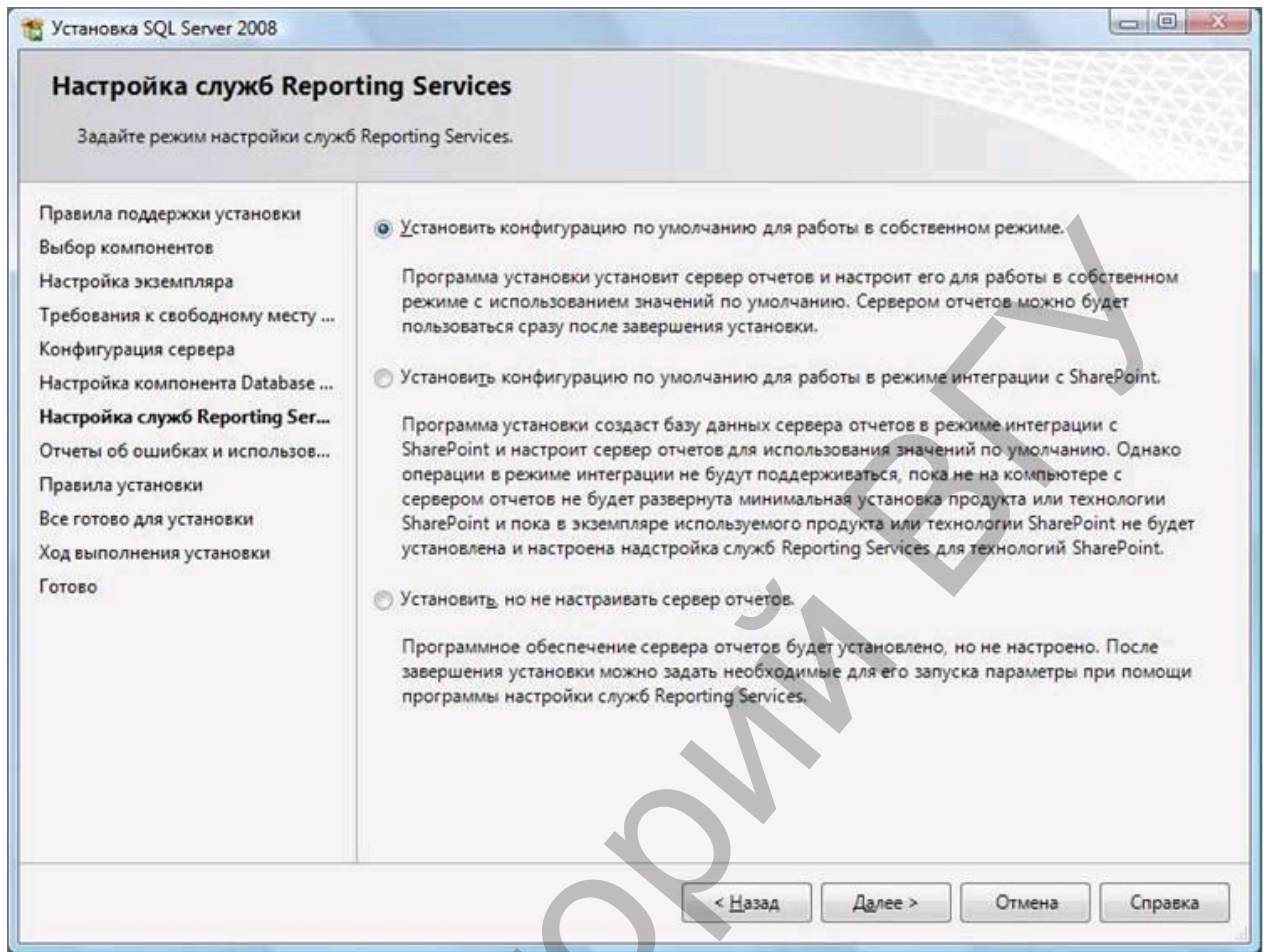


Рис. 17

17. Выбрать опции, как показано на рисунке 17, нажать кнопку «Далее».

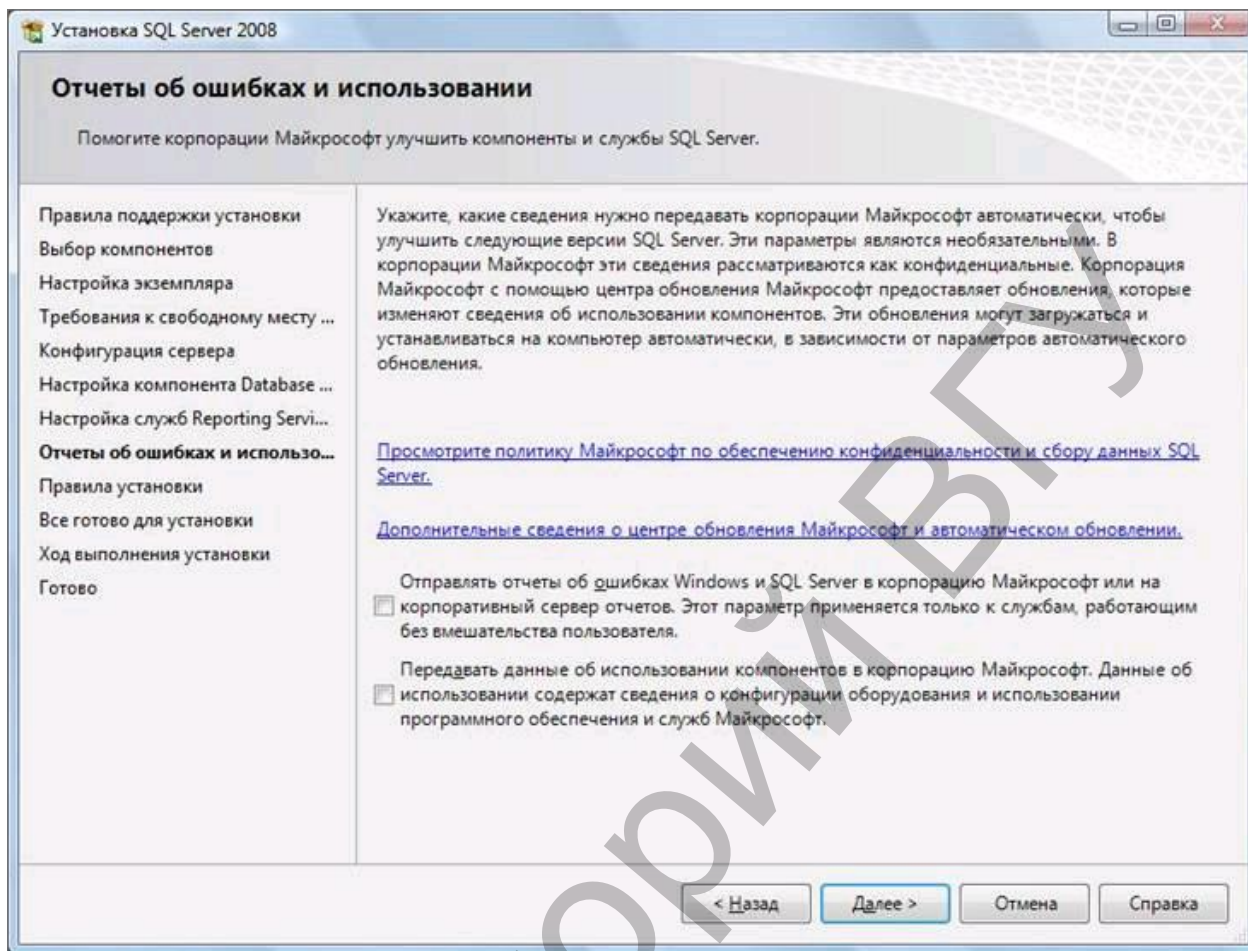


Рис. 18

18. Нажать кнопку «Далее»:

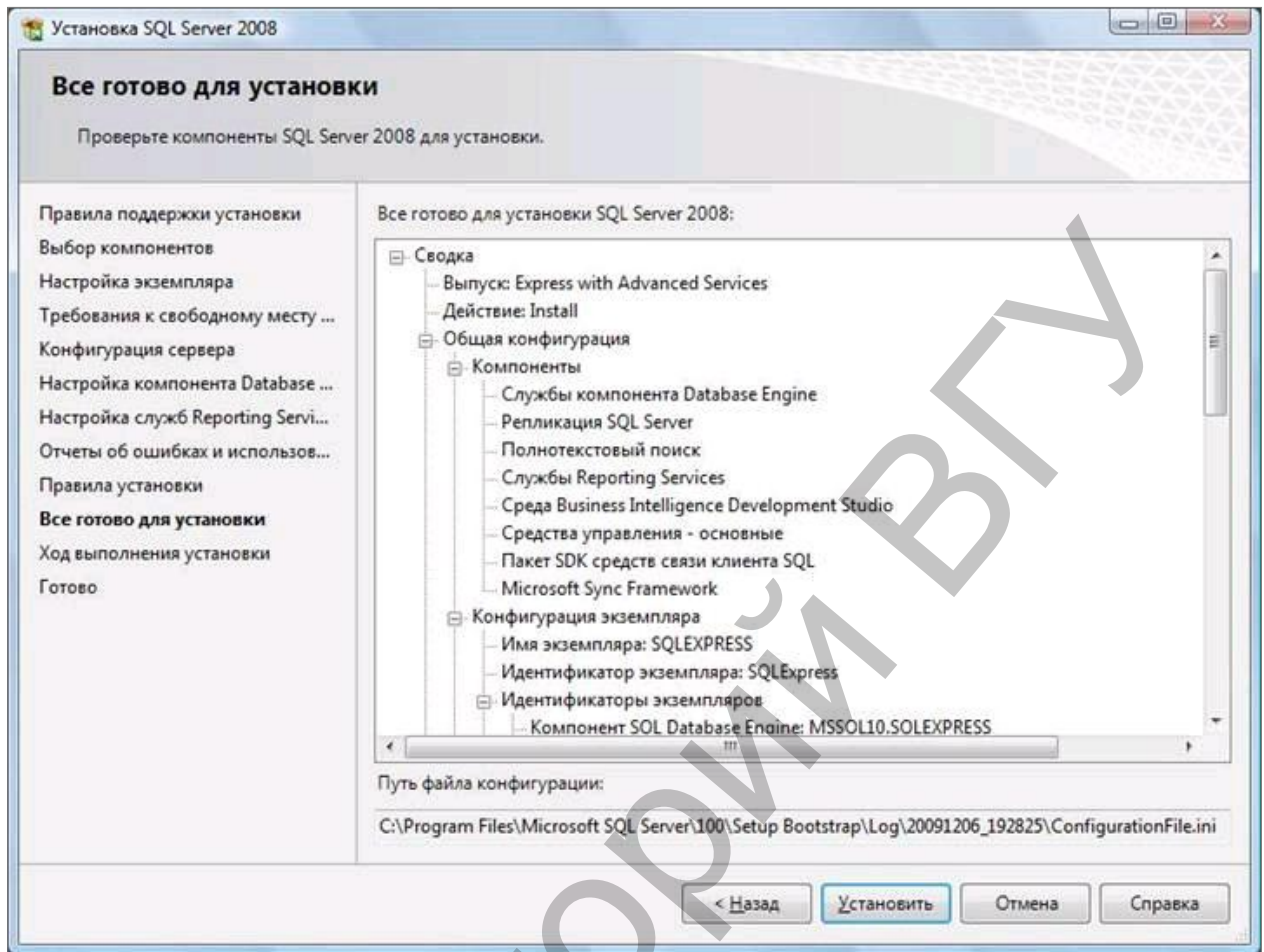


Рис. 19

19. Убедиться, что все проверки успешно пройдены. Если будут обнаружены какие-то проблемы, то необходимо их устранить и запустить повторную проверку кнопкой «Включить заново». Затем нажать кнопку «Далее»:

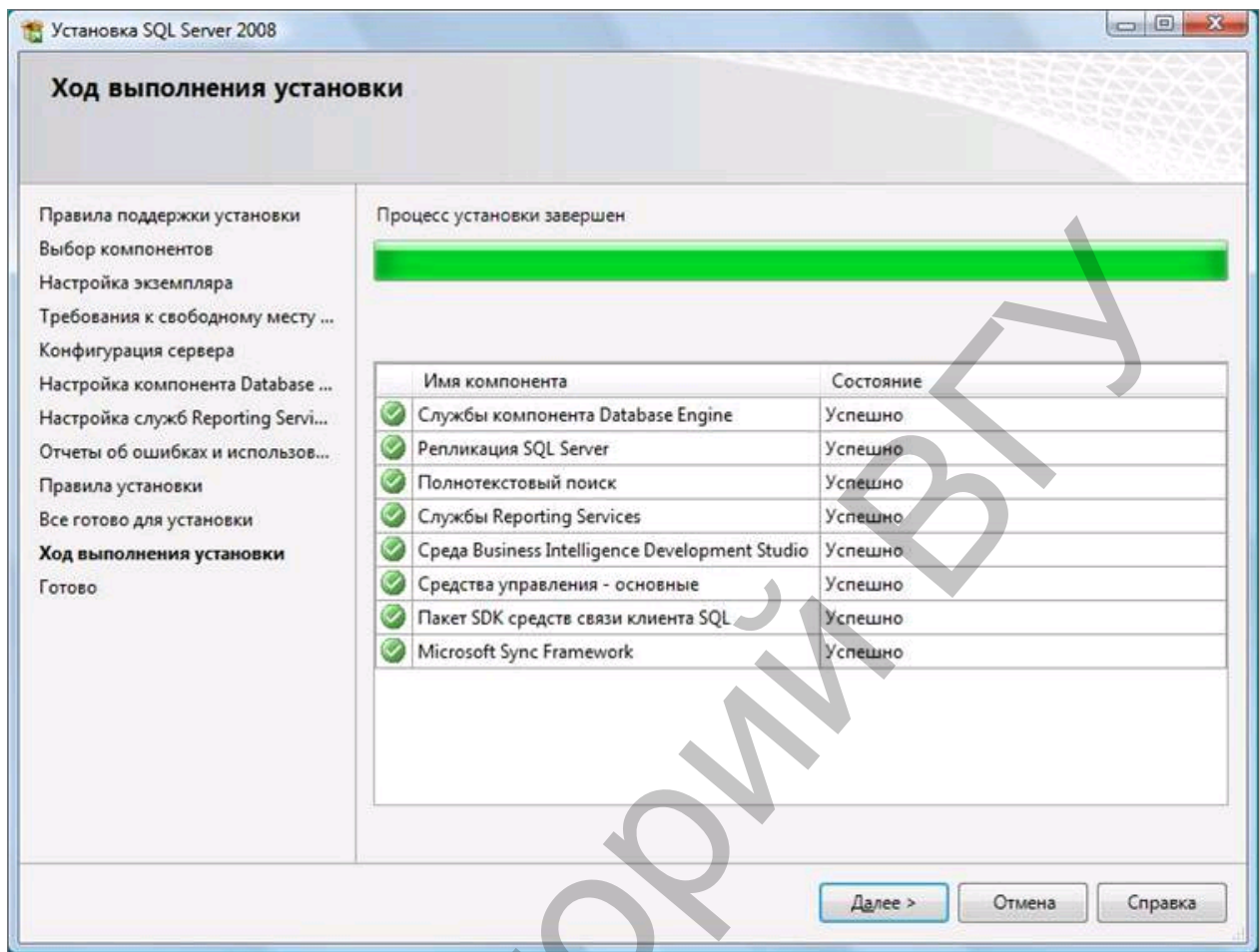


Рис. 20

20. Нажать кнопку «Установить»:



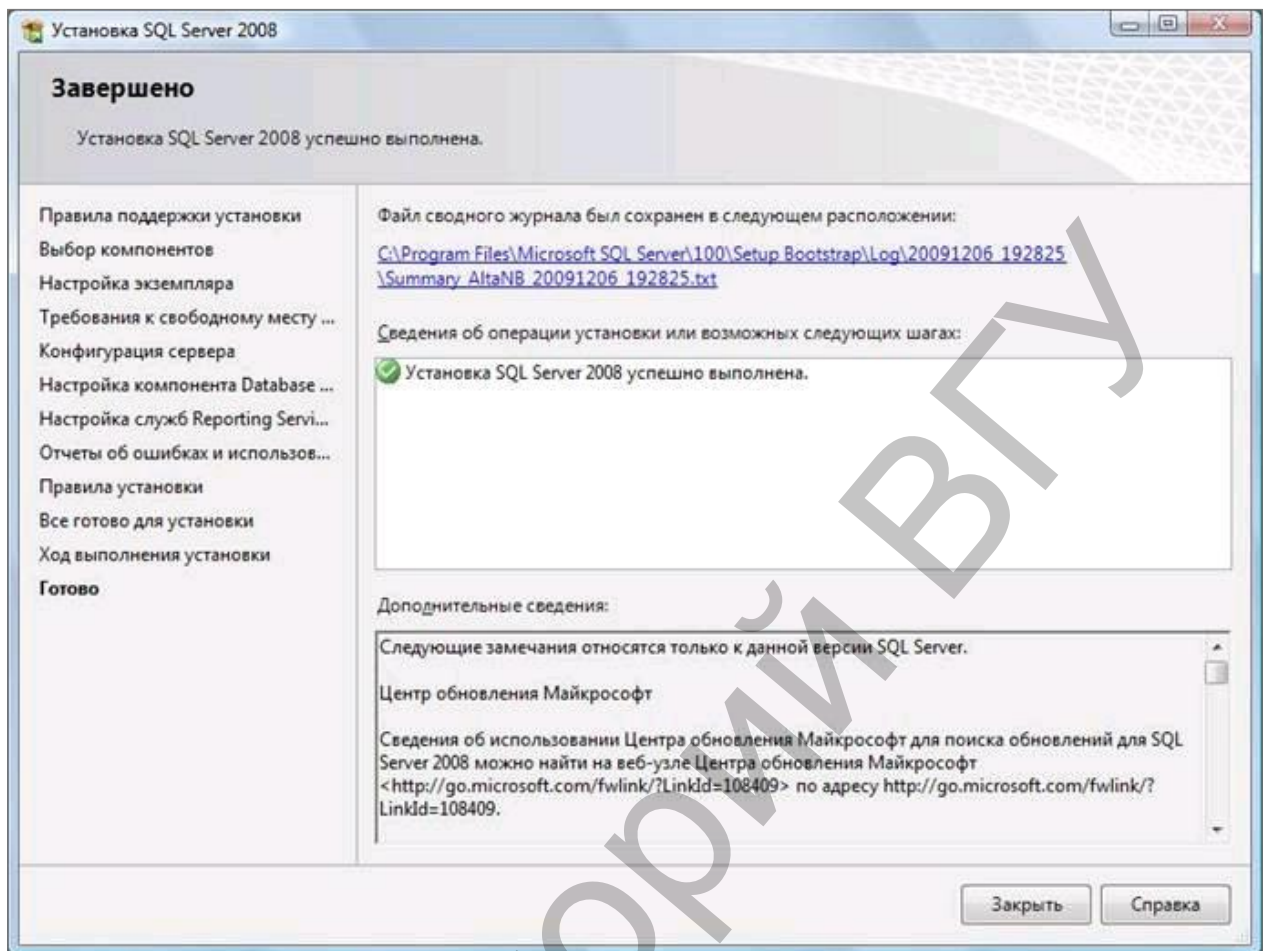


Рис. 21

21. После завершения установки нажать кнопку «закреть»:

После завершения установки и настройки рекомендуется перезагрузить компьютер.

### ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Выполните установку SQL Server в соответствии с инструкцией.
2. Выполните установку MS SQL Management Studio (Запустить программу-установщик SQLManagementStudio\_x86Rus.exe).

## ЛАБОРАТОРНАЯ РАБОТА № 2

**Тема:** Проектирование базы данных.

**Цель работы:** Изучить методы проектирования БД на основе инфологического моделирования. Ознакомиться с основными этапами разработки концептуального представления и логической структуры базы данных.

### ОСНОВНЫЕ СВЕДЕНИЯ

#### **Создание инфологической модели методом «сущность-связь». Создание ER-диаграмм**

Предварительный этап создания инфологической модели предусматривает выполнение системного анализа и словесного описания информационных объектов предметной области. На первом этапе проектирования создается концептуальная схема БД, которая затем преобразуется к реляционной схеме. В результате создается реляционная БД в 3-ей нормальной форме. Рассмотрим одну из наиболее важных и распространенных семантических моделей данных – модель “Сущность связь” (ER-модель). Основные понятия ER- модели: сущность, связь и атрибут.

**Сущность** – это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна. В диаграммах ER- модели сущность представляется в виде прямоугольника, содержащего имя сущности. При этом имя сущности (в виде существительных) – это имя типа, а не некоторого конкретного элемента этого типа. Каждый элемент сущности должен быть отличим от любого другого элемента этой же сущности.

**Связь** – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация всегда является бинарной и может существовать между 2-мя сущностями или между сущностью и ею же самой (рекурсивная связь). В любой связи выделяются 2 конца, на каждом из которых указывается имя конца связи (в виде глагола), степень конца связи (сколько элементов данной сущности связывается) и обязательность связи (то есть, любой ли элемент данной сущности должен участвовать в этой связи).

Связь представляется в виде линии, соединяющей 2 сущности, при этом в месте соприкосновения связи с сущностью используется

множественный вход в прямоугольник, если для связи могут использовать несколько элементов, и единичный – если в связи может участвовать только один элемент сущности. Обязательность связи изображается перпендикуляром, а необязательность – окружностью.

Примеры:



Конец связи **имеет**, означает, что в каждом отделении компании работает 1-5 и более сотрудников. Конец связи с именем **работает** позволяет соединить с одним сотрудником не более 1-го отделения, то есть сотрудник не может работать одновременно в нескольких отделениях. Трактовка изображенной диаграммы следующая: каждый сотрудник может работать в одном из отделений компании (не все сотрудники обязательно принадлежат отделениям, так как у компании может быть свой офис и административный аппарат), каждое отделение обязательно имеет 1-5 и более сотрудников.

**Атрибутом** сущности является любой элемент, который служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности. Имена атрибутов заносятся в прямоугольник, записываются малыми буквами под именем сущности, возможно с примерами.

Некоторый набор атрибутов назначается уникальным идентификатором (**ключом**).

Как и в реляционных схемах БД, так и в ER-схемах вводится понятие нормальных форм, причем их смысл очень близко соответствует смыслу реляционных нормальных форм.

Определения 3-х первых нормальных форм (НФ):

В 1 НФ ER-схемы устраняются повторяющиеся атрибуты или группы атрибутов. Во 2 НФ устраняются атрибуты, зависящие от части уникального идентификатора. Эта часть уникального идентификатора определяет отдельную сущность.

В 3 НФ устраняются атрибуты, зависящие от атрибутов, не входящих в уникальный идентификатор. Они являются основой отдельной сущности.

В ER-модели допускается принцип *категоризации* сущностей.

Это значит, что, как и в объектно-ориентированных языках программирования, вводится понятие подтипа сущности (категории). Все подтипы сущностей рассматриваются как взаимоисключающие. Сущность, на основе которой строятся подтипы, называется супертипом. Как в языках объектно-ориентированного программирования, вводится возможность наследования типа сущности, исходя из одного или нескольких супертипов.

Эти элементы ER-модели делают ее более мощной, но существенно усложняют ее использование. Например, сущность может быть расщеплена на 2 или более взаимоисключающих подтипа, каждый из которых включает общие атрибуты. В подтипах могут определяться собственные атрибуты и связи.

Преобразование ER-модели в реляционную схему осуществляется в соответствии со следующими правилами:

- каждая простая сущность превращается в отношение. Имена отношений могут отличаться от имен сущностей, так как могут быть ограничены требованиями конкретной СУБД;
- каждый атрибут становится возможным столбцом с тем же именем, для каждого атрибута задается допустимый тип данных и обязательность или необязательность этого атрибута;
- компоненты уникального идентификатора сущности превращаются в первичный ключ отношения;
- в каждое отношение, соответствующее подчиненной сущности, добавляется набор атрибутов основной сущности, являющейся первичным ключом основной сущности. В отношении, соответствующем подчиненной сущности этот набор атрибутов становится внешним ключом.
- Для связи М:М используется специальный механизм преобразований, который позволяет отразить множественные связи, неспецифичные для реляционной модели. Это делается введением дополнительного связующего отношения, которое связано с каждым исходным связью 1:М, атрибутами этого связующего отношения являются первичные ключи связываемых отношений. При этом каждый из атрибутов нового отношения

является *внешним ключом*, а вместе они образуют первичный ключ новой связующей сущности.

Спроектируем инфологическую модель системы, предназначенной для компании, которая занимается продажей объектов недвижимости. Компания предлагает следующие услуги:

- сбор информации об объектах, выставляемых на продажу;
- представление данных в общую БД;
- организация просмотра объектов потенциальными покупателями;
- составление договоров на продажу недвижимости.

Компания имеет несколько отделений (агентств), расположенных в разных районах города и районных центрах. При этом компания состоит не только из отделений, так как у компании может быть свой офис и административный аппарат, следовательно, не все сотрудники обязательно принадлежат отделениям. В каждом отделении компании есть (но не весь) персонал, отвечающий за работу с выставленными на продажу объектами недвижимости.

### **Описание предметной области**

Выделим объекты предметной области:

- Каждое отделение компании может быть описано с помощью следующих характеристик: уникальный номер отделения, его адрес (почтовый индекс, город, район, улица, дом), номер телефона и номер факса.
- В штат входят работники, называемые торговыми агентами. Торговые агенты занимаются продажей недвижимости. Информация, описывающая каждого сотрудника компании, включает персональный номер, полное имя (имя и фамилию), адрес проживания, номер телефона, пол, дату рождения, занимаемую должность, а также номер и адрес отделения компании, в котором он работает. Личный номер каждого работника является уникальным в пределах всех отделений компании.
- Данные обо всей выставленной на продажу недвижимости можно получить в любом отделении компании. Информация, описывающая каждый объект недвижимости, включает номер

объекта, адрес его местонахождения (почтовый индекс, город, район, улица, дом и квартира), тип объекта, количество комнат в нём, отпускную цену, а также имя и адрес владельца этого объекта. Каждый объект недвижимости имеет единственного владельца.

- Компания управляет недвижимостью частных лиц. Частный владелец идентифицируется собственным номером, уникальным для всех отделений компании. Дополнительная информация о владельцах включает фамилию, имя, адрес и номер телефона. Каждому владельцу принадлежит, по крайней мере, один объект недвижимости.
- Потенциальный покупатель обращается в отделение компании, в котором ему могут предложить осмотреть разные объекты недвижимости. Информация, сохраняемая по каждому проведённому осмотру объекта, включает имя и адрес клиента, номер и адрес осмотренного объекта недвижимости, дату осмотра, а также комментарии по результатам осмотра. Клиент может осматривать любое количество объектов недвижимости.
- О каждом клиенте хранится следующая информация: фамилия и имя, адрес, номер телефона, предпочтительное количество комнат в покупаемой квартире, а также максимальная цена, которую клиент согласен уплатить. Каждый клиент получает личный номер, уникальный для всех отделений компании.
- При покупке некоторого объекта покупатель заключает с компанией договор на покупку выбранного им объекта (квартиры). Подробная информация о договоре на покупку включает: номер соглашения, дату заключения договора, личный номер покупателя, его имя и адрес, номер покупаемой квартиры и адрес её местонахождения, стоимость квартиры с учётом комиссионных, а также сведения о сотруднике, который составил данный договор. Клиент может купить как один, так и сразу несколько объектов недвижимости.

Необходимо предусмотреть следующие ограничения на информацию в системе:

1. В каждом отделении компании работает, по крайней мере, 5 сотрудников, а максимальное их количество не ограничено.
2. Каждый сотрудник может отвечать не более чем за 10 объектов недвижимости одновременно.

3. Компания требует сохранять данные об уволившемся сотруднике в течение года.
4. Каждый покупатель при обращении в агентство должен оставить свой номер телефона (рабочий или домашний) для быстрой с ним связи.

В данной информационной системе должны реализовываться определённые задачи, за выполнение которых несут ответственность сотрудники компании. А именно:

- Создание и корректировка записей с данными о сотрудниках каждого отделения.
- Создание отчёта со сведениями о сотрудниках каждого отделения.
- Удаление сведений об уволившемся сотруднике из базы данных и передача ответственности за все курируемые им объекты недвижимости другому сотруднику.
- Создание и корректировка записей с данными о выставленных на продажу объектах недвижимости в конкретном отделении компании.
- Создание отчёта с данными о выставленных на продажу объектах недвижимости в данном отделении компании.
- Создание и корректировка записей с описанием потенциальных покупателей и их требований.
- Поиск всех объектов недвижимости, удовлетворяющих требованиям покупателя.
- Поиск возможного покупателя для вносимого в базу данных объекта недвижимости.
- Создание и корректировка записей со сведениями об осмотре объектов недвижимости.
- Создание и корректировка записей со сведениями о заключённых договорах.
- Распечатка договора.
- При заключении договора на объект он должен автоматически удаляться из списка объектов недвижимости, выставленных на продажу.

### **Описание сущностей и типов связей**

Определим основные типы сущностей исходя из описания предметной области.

### 1. Отделение (BRANCH)

Каждое отделение имеет следующий набор атрибутов: номер отделения, адрес (почтовый индекс, город, улица, дом), номер телефона и номер факса.

### 2. Сотрудник (STAFF)

Каждый сотрудник характеризуется следующими атрибутами: номер сотрудника, фамилия, имя сотрудника, дата рождения, пол, адрес (город, улица, дом, квартира), номер домашнего телефона, дата зачисления в штат, должность и зарплата.

### 3. Объект недвижимости для продажи (PROPERTY)

Характеризуется такими атрибутами как: номер объекта недвижимости, дата регистрации, полный адрес (почтовый индекс, город, улица, дом и квартира), тип объекта (N-этажный панельный или кирпичный), этаж, количество комнат, площадь (общая, жилая, площадь кухни), балкон (балкон, лоджия, застеклён, их количество или отсутствие), наличие телефона (есть, нет) и отпускная цена.

### 4. Владелец (OWNER)

Владелец имеет атрибуты: номер владельца, фамилия, имя, адрес и номер телефона.

### 5. Покупатель (BUYER)

Каждый покупатель характеризуется следующими атрибутами: код покупателя, фамилия и имя, адрес (город, улица, дом, квартира) номер телефона (рабочий и домашний), предпочтительное количество комнат, максимальная цена.

Определим типы связей, которые существуют между основными сущностями.

Между сущностями **Отделение** и **Сотрудник** существует связь 1:M, обязательная только с одной стороны. Связь обязательна только с одной стороны, т.к. каждое из отделений компании **имеет** несколько штатных сотрудников, но не все сотрудники компании работают в отделениях. В обратном направлении, каждый из сотрудников отделений **работает** только в одном из них. Ключевой атрибут сущности **Отделение**: *Номер отделения (Branch\_no)*. Ключевой атрибут сущности **Сотрудник**: *Номер сотрудника (Staff\_no)*.

Из описания предметной области известно, что каждый объект недвижимости, выставленный на продажу, **закрепляется** за конкретным отделением компании. В каждом отделении компании



есть сотрудник, отвечающий за работу с выставленными на продажу объектами недвижимости. Для отражения этой ситуации необходимо провести связь между сущностями **Объект недвижимости для продажи** и **Отделение** (ключевым атрибутом сущности **Объект недвижимости для продажи** является *Номер объекта недвижимости (Property\_no)*). Для того чтобы узнать, какой объект недвижимости обслуживается каким сотрудником и, с другой стороны, какой сотрудник отвечает за данный объект, вводится дополнительная связь между сущностями **Объект недвижимости для продажи** и **Сотрудник**. Между сущностями **Отделение** и **Объект недвижимости для продажи** установлена связь 1:M, обязательная с 2-х сторон. Между сущностями **Сотрудник** и **Объект недвижимости для продажи** – связь 1:M, необязательная с 2-х сторон, так как из всех сотрудников компании только торговые агенты занимаются продажей недвижимости и отвечают за работу с ними. В обратном направлении, объект может быть не связан ни с одним из сотрудников. Например, когда объект впервые регистрируется в компании.

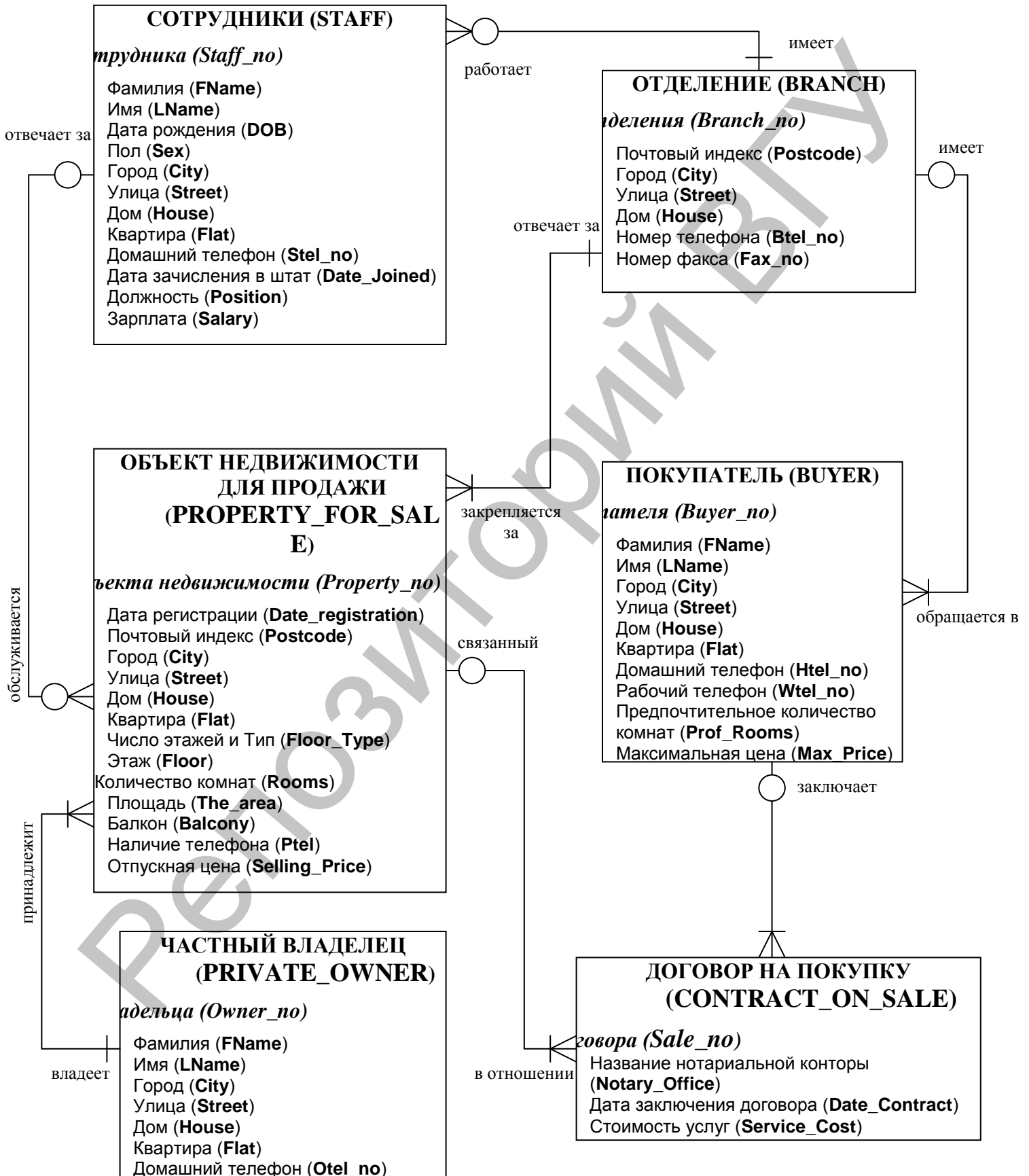
Теперь необходимо отразить связь между сущностями **Владелец** (атрибут *Номер владельца (Owner\_no)* является ключевым) и **Объект недвижимости для продажи**. Если рассмотреть эту связь с одной стороны, то можно заметить, что один владелец может владеть несколькими объектами недвижимости. С другой стороны, каждый объект принадлежит только одному владельцу. Следовательно, связь между сущностями - 1:M. Поскольку каждый владелец владеет, по крайней мере, одним объектом недвижимости, а каждый объект должен иметь одного владельца, связь является обязательной с обеих сторон.

Из описания предметной области известно, что потенциальный покупатель **обращается в** одно из отделений компании (только когда клиент становится потенциальным покупателем агентства, данные о нём, заносятся в базу данных), в котором ему могут предложить осмотреть разные объекты недвижимости. Клиент, как правило, желает осмотреть один или несколько, предлагаемых ему объектов недвижимости. Сведения о таком просмотре включают дату осмотра объекта и комментарии потенциального покупателя (согласен он или нет купить данную квартиру и др.). Образуется необязательная с двух сторон связь

М:М между сущностями **Покупатель** и **Объект недвижимости для продажи**. Отдельный клиент может осмотреть несколько выставленных на продажу объектов (1:М), а каждый объект может быть **осмотрен** несколькими клиентами (1:М). Связь необязательна со стороны клиента из-за возможного отсутствия объекта, отвечающего его требованиям. С другой стороны, сведения о некоторых объектах просто регистрируется в компании, а осмотр их клиентами не производится.

Если клиент согласен купить некоторый объект, то он заключает с компанией договор на покупку выбранного им объекта. Сотрудник компании должен оформить это соглашение. Каждый объект может быть продан единственному клиенту, и каждый клиент может купить один или более объектов в одно и то же время. Образуется необязательная с двух сторон связь 1:М между сущностями **Покупатель** и **Объект недвижимости для продажи**. Но так как, всякий раз, при покупке клиент заключает договор с компанией, мы определим две связи. Связь 1:М между сущностями **Покупатель** и **Договор на покупку**, а также связь 1:М между сущностями **Объект недвижимости для продажи** и **Договор на покупку**. Связи обязательны со стороны сущности **Договор на покупку**. Ключевым атрибутом для сущности **Покупатель** является *Код покупателя (Buyer\_no)*, а для сущности **Договор на покупку** – атрибут *Номер договора (Sale\_no)*. Кроме этого сущность **Договор на покупку (CONTRACT\_ON\_SALE)** имеет атрибуты: название нотариальной конторы, дата заключения договора, стоимость услуг.

## Инфологическая модель предметной области



## Переход к реляционной модели

В реляционной модели связи явным образом не отображаются, однако между отношениями поддерживаются иерархические связи (в каждой связи одно отношение выступает как основное, а другое как подчиненное). Это значит, что один кортеж основного отношения может быть связан с несколькими кортежами подчиненного отношения. Для поддержки этих связей оба отношения должны содержать наборы атрибутов, по которым они связаны. В основном отношении это первичный ключ отношения. В подчиненном отношении для моделирования связи должен присутствовать набор атрибутов, соответствующий первичному ключу основного отношения. Данный набор атрибутов в подчиненном отношении принято называть внешним ключом. Согласно правилу 4 перехода к реляционной модели (в каждое отношение, соответствующее подчиненной сущности, добавляется набор атрибутов основной сущности, являющейся ее первичным ключом), введем в дополнительное отношение **Договор на покупку** ключи отношений **Покупатель** и **Объект недвижимости для продажи**.

Для связи М:М между сущностями **Покупатель** и **Объект недвижимости для продажи** введем дополнительное связующее отношение, которое связано с каждым исходным связью 1:М.. Атрибутами этого связующего отношения, помимо даты осмотра и комментарии, будут первичные ключи связываемых отношений, т.е. **Property\_no** и **Buyer\_no**. Для нового отношения они являются внешними ключами, а вместе они образуют первичный ключ новой связующей сущности **Осмотр (VIEWING)**.

### ОСМОТР (VIEWING)

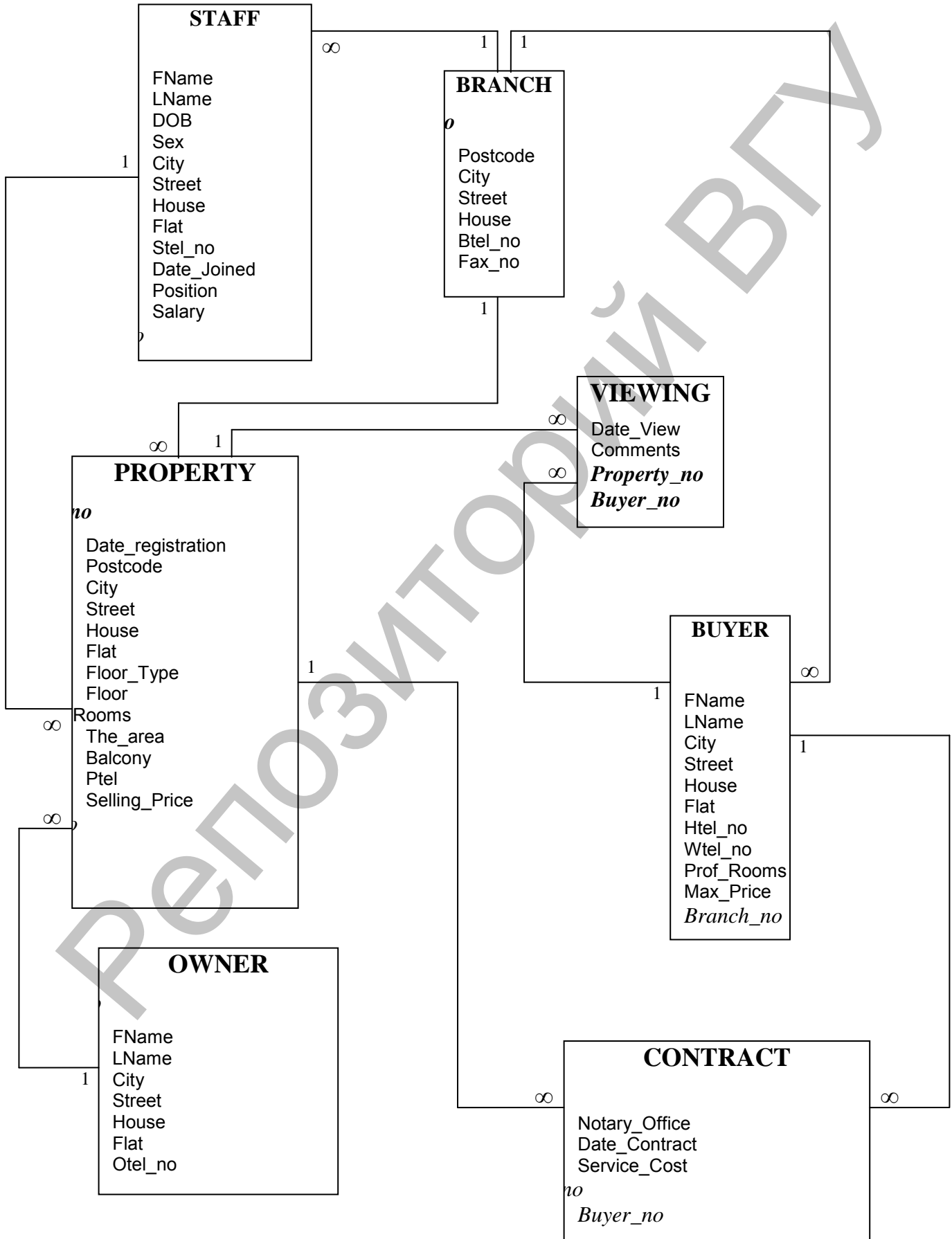
Дата осмотра (**Date\_View**)

Комментарии (**Comments**)

**Property\_no**

**Buyer\_no**

# Реляционная модель



## **КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. С чего начинается разработка базы данных?
2. Охарактеризуйте основные понятия ER- модели: сущность, связь и атрибут.
3. Каким образом на ER диаграмме отображается обязательность и необязательность связи.
4. Перечислите правила преобразования ER-модели в реляционную схему.

## **ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ**

1. Выполните проектирование базы данных (ER-модель, реляционную схему).
2. Исходные данные в соответствии с вариантом задания находятся в [Приложении 1](#).

## ЛАБОРАТОРНАЯ РАБОТА № 3

**Тема:** Создание базы данных. Типы данных, используемые в SQL-сервере. Создание пользовательских типов данных.

**Цель работы:** Изучить различные способы создания баз данных, рассмотреть основные типы данных и способы создания пользовательских типов данных.

### ОСНОВНЫЕ СВЕДЕНИЯ

#### Создание базы данных

База Данных (БД) – некоторый набор таблиц, связанных между собой определенной связью для выполнения различного рода задач.

Создание базы данных в системе SQL-сервер может осуществляться следующими способами: используя стандартные команды языка T-SQL и MS SQL Server Management Studio.

**Первый способ:** С помощью оператора CREATE DATABASE. В процессе создания базы также создаётся журнал транзакций (с помощью оператора LOG ON).

Создание базы данных – это процесс указания имени базы и определения размеров и размещения файлов базы данных (первичного и вторичных файлов базы данных, файла журнала транзакций). В primary файле базы данных (расширение .mdf) записывается информация об основных её объектах – таблицах, индексах и т. д., а в файл журнала транзакций (расширение .ldf) информация о процессе работы с транзакциями (контроль целостности данных, состояние базы данных до и после выполнения транзакции). Если в процессе использования базы данных планируется размещение её на нескольких дисках, то в этом случае создаются secondary файлы (расширение .ndf). По умолчанию базы данных имеют право создавать только те пользователи, которым назначены роли *sysadmin* и *dbcreator*.

#### Синтаксис:

```
CREATE DATABASE имя_базы_данных
[ON
[PRIMARY] (NAME=логическое_имя_файла,
           FILENAME='физическое_имя_файла'
           [, SIZE=размер]
           [, MAXSIZE=максимальный_размер]
           [, FILEGROWTH=шаг_приращения_размера])
[, ...n]
]
```

[LOG ON

(NAME=*логическое имя файла журнала*,

FILENAME=*'физическое имя файла журнала'*

[, SIZE=*размер журнала*])

[, ...*n*]

]

[FOR RESTORE]

При этом следует помнить, что пробелы используются для разделения элементов команды SQL, и поэтому они не могут быть частью имени базы данных, таблицы или какого-либо другого создаваемого объекта.

При создании базы данных можно указать следующие параметры:

- PRIMARY указывает файлы основной группы файлов, которая содержит все системные таблицы базы данных. Кроме того, здесь также содержатся объекты, не привязанные к пользовательским группам файлов. В любой базе данных должен быть лишь один основной (первичный) файл данных. Он служит отправной точкой базы данных и указывает на все прочие её файлы. Стандартное расширение имени основного файла данных – .mdf. Если ключевое слово PRIMARY опущено, основным файлом становится первый файл в операторе;
- FILENAME – задаёт физическое имя и путь к файлу. В пути (*физическое имя файла*) необходимо указывать папку локального диска сервера, на котором установлен SQL Server;
- 1. SIZE – указывает размер файла: в мегабайтах, тогда используется суффикс MB (по умолчанию), или в килобайтах – в этом случае применяется суффикс KB. Минимально возможное значение – 512 кб. Параметр *size* задает минимальный (начальный) размер файла. Файл может увеличиваться, однако его нельзя сжать так, чтобы его объем стал меньше заданного минимального размера;
- MAXSIZE – указывает максимальный размер файла. Если размер не указан, то файл будет увеличиваться до полного заполнения диска;



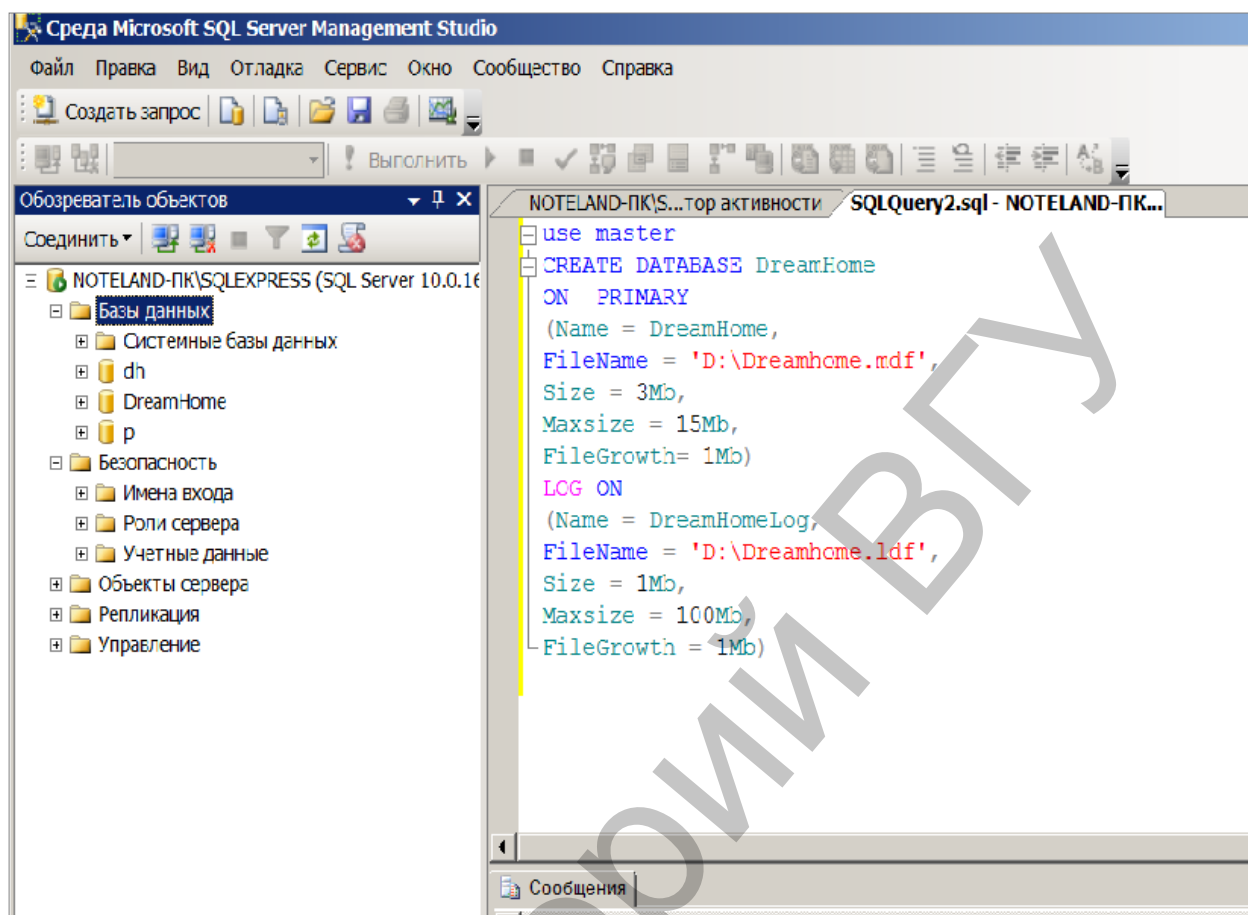


Рисунок 22

- FILEGROWTH – задает шаг приращения размера файла. Если SQL Server необходимо увеличить размер файла, он увеличит его на значение, заданное параметром FILEGROWTH, причем ноль означает запрет увеличения размера. По умолчанию (если параметр FILEGROWTH не определен) – шаг приращения равен 10%, а его минимальное значение – 64 Кб. Указанный вами размер округляется до ближайшего числа, кратного 64 Кб;
- FOR RESTORE – задаёт восстановление системы по журналу транзакций в случае её сбоя. Имеется в виду сбой системы, нарушающий все выполняемые в данный момент транзакции, но не нарушающий базу данных физически. При сбое носителей, который представляет собой физическую угрозу для данных, восстановление осуществляется с резервной копии БД.

Кроме перечисленных, при создании базы данных можно указать параметры настройки и параметры доступа к базе данных. Например, можно сделать базу данных доступной только для чтения или указать, чтобы регистрационные записи удалялись из журнала транзакций после записи изменённых страниц данных на диск. Можно также осуществить настройку доступа пользователей к базе данных. По умолчанию свойства доступа могут устанавливаться для пользователей, осуществляющих подключение,

используя роль *public*. Наиболее часто используемые параметры смотрите в Приложении 2.

**Упражнение:** При помощи оператора CREATE DATABASE создайте базу данных DreamHome. Параметры базы данных указаны в таблице:

Файл	Логическое имя	Физическое имя	Начальный размер	Шаг приращения размера	Максимальный размер
База данных	<b>DreamHome</b>	D:\DreamHome.mdf	3 Мб	1 Мб	15 Мб
Журнал	DreamHome_log	D:\ DreamHome.ldf	1 Мб	1 Мб	10 Мб

1. Для начала необходимо запустить среду разработки "MS SQL Server Management Studio". Для этого в меню "Пуск" выбрать пункт "Программы\Microsoft SQL Server 2008\SQL Server Management Studio".
2. Создайте новый запрос, нажав на панели инструментов кнопку "Создать запрос".
3. Выполните оператор CREATE DATABASE, чтобы создать базу данных (см. рисунок 22).
4. Просмотрите свойства базы данных с помощью группы процедур *sp\_helpdb*, чтобы убедиться, что база создана должным образом:

*EXEC sp\_helpdb DreamHome*

### 1. **Второй способ:**

В обозревателе объектов откройте контекстное меню папки Базы данных и в появившемся меню выберите пункт "Создать базу данных".

Появится окно настроек параметров файла данных новой БД (см. рисунок 23)

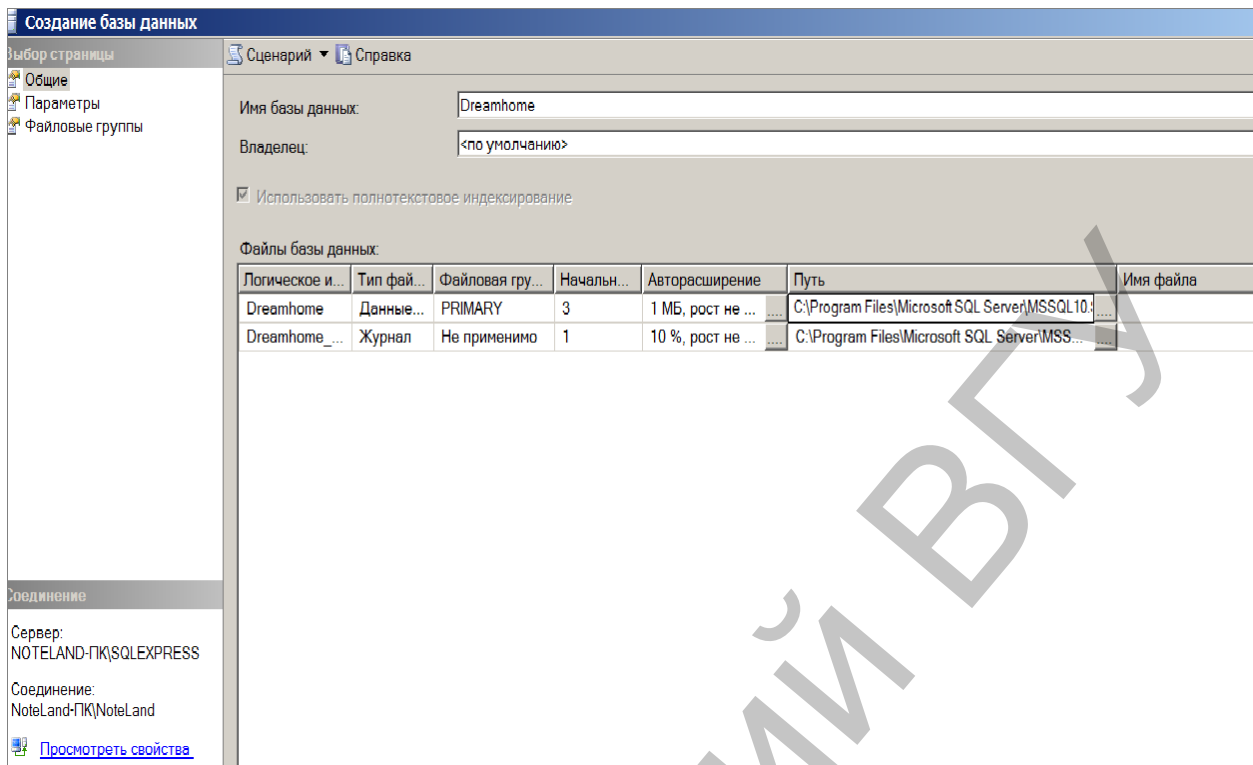


Рисунок 23

На панели **Выбор страницы** выбрать **Общие**.

В верхней части окна расположено два параметра: **Имя Базы данных** и **Владелец**). Задайте **Имя Базы данных** - **DreamHome**. Параметр **Владелец** " оставить без изменений.

Под вышеприведенными параметрами в виде таблицы располагаются настройки файла данных и журнала транзакций. Таблица имеет следующие столбцы:

- Логическое имя файла данных и журнала транзакций. По этим именам будет происходить обращение к вышеприведенным файлам в БД. Можно заметить, что файл данных имеет то же имя что и БД, а имя файла журнала транзакций составлено из имени БД и суффикса "\_log".
- Тип файла. Этот параметр показывает, является ли файл файлом данных или журналом транзакций.
- Файловая группа показывает, к какой группе файлов относится файл. **Группы файлов** настраиваются в группе настроек **Файловые группы**.
- Начальный размер файла данных и журнала транзакций в мегабайтах.
- Авторасширение размера файла. Как только файл заполняется информацией, его размер автоматически увеличивается на величину, указанную в параметре **Авторасширение**. Увеличение можно задавать как в мегабайтах так и в процентах. Здесь же можно задать

максимальный размер файлов. Для изменения этого параметра надо нажать кнопку "...".

- Путь к папке, где хранятся файлы. Для изменения этого параметра также следует нажать кнопку "...". (Укажите путь к папке, где будет храниться ваша база данных).
- имя файла. По умолчанию имена файлов аналогичны логическим именам. **Замечание:** Для добавления новых файлов данных или журналов транзакций используется кнопка **Добавить**, а для удаления кнопка **Удалить**.

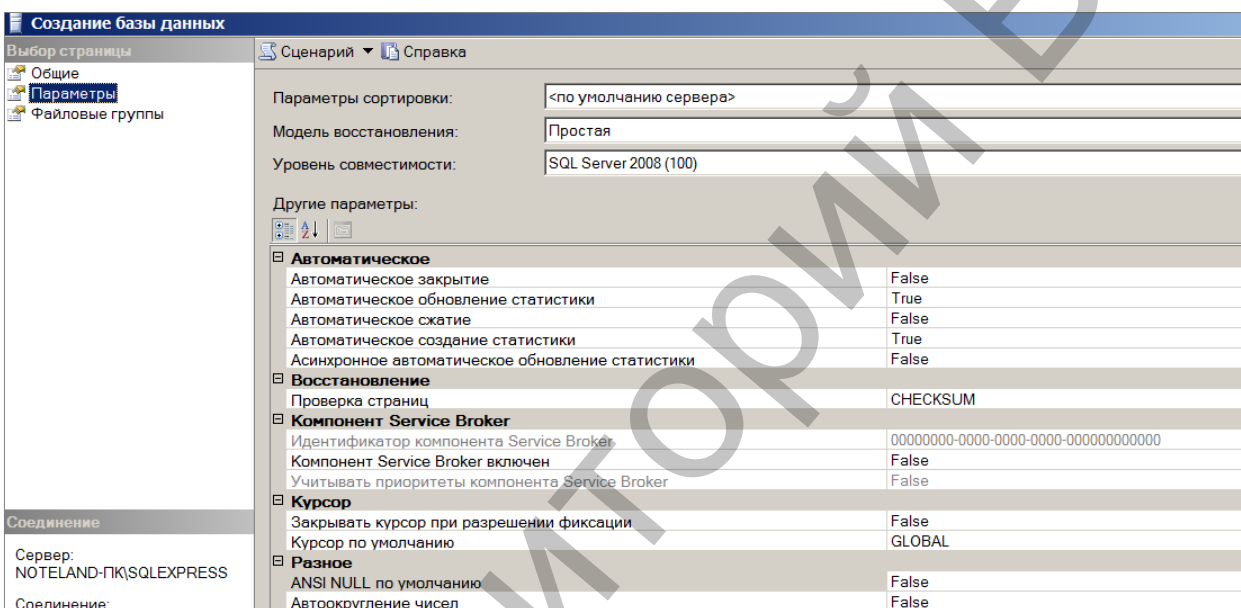


Рис.24

Перейдем к второстепенным настройкам файла данных. Для доступа к этим настройкам необходимо щелкнуть мышью по пункту **Параметры**. В правой части окна находятся следующие настройки:

- **Сортировка** - этот параметр отвечает за обработку текстовых строк, их сравнение, текстовый поиск и т.д. Рекомендуется оставить его как **по умолчанию сервера**.
- **Модель восстановления**. Данный параметр отвечает за информацию, предназначенную для восстановления БД, хранящуюся в файле транзакций. Чем полнее модель восстановления, тем больше вероятность восстановления данных при сбое системы или ошибках пользователей, но и больше размер файла журнала транзакций.
- **Уровень совместимости**, определяет совместимость файла данных с более ранними версиями сервера. Если планируется перенос данных на

другую, более раннюю версию сервера, то ее необходимо указать в этом параметре.

Рассмотрим последнюю группу настроек **Файловые группы**.

Таблица имеет следующие столбцы:

- **Имя** группы файлов.
- **Файлы** - количество файлов входящих в группу.
- **Только для чтения**. Файлы нельзя изменять.
- **По умолчанию**. Все новые файлы данных будут входить в эту группу.

Эту группу настроек оставляем без изменений.

Для принятия всех настроек и создание файла данных и журнала транзакций нашей БД в окне **Новая база данных** нажать кнопку **"Ok"**.

Произойдет возврат в окно среды разработки **"SQL Server Management Studio"**. На панели обозревателя объектов в папке **Базы данных** появится новая БД **DreamHome**.

**Упражнение:** просмотрите информацию о базе данных **DreamHome**.

Щёлкните правой кнопкой базу данных **DreamHome**.

Выберите в контекстном меню команду *Свойства* и просмотрите сведения о пространстве базы данных и журнала, параметры настройки (закладка *Параметры*) и параметры доступа к базе данных (закладка *Разрешения*).

#### **Удаление базы данных**

По умолчанию право на удаление базы данных принадлежит ее владельцу и пользователям – членам постоянной роли *sysadmin*.

Удалить базу данных можно с помощью команды **Удалить** среды MS SQL Server Management Studio или оператора:

*DROP DATABASE имя\_базы\_данных.*

**Упражнение:** Удалите базу данных **DreamHome**.

Выполните оператор **DROP DATABASE**, чтобы удалить базу данных **DreamHome**:

***DROP DATABASE DreamHome.***

Отправьте операторы серверу с помощью команды **Go**;

4. Выполните системную процедуру *sp\_helpdb*, чтобы создать список баз данных. Убедитесь, что вы удалили БД **DreamHome\_db**:

***EXEC sp\_helpdb***

#### **Типы данных, используемые в SQL-сервере**

После создания логической структуры базы данных и самой БД можно приступать к созданию в ней объектов: пользовательских типов данных и таблиц.

Одним из основных этапов в процессе создания таблиц является определение типов данных ее полей. Тип данных поля таблицы определяет тип информации, которая будет размещаться в этом поле. SQL-сервер поддерживает большое число различных типов данных: текстовые, числовые, двоичные и т.д. (см. Приложение 3).

### Пользовательский тип данных

В системе SQL-сервер имеется поддержка пользовательских типов данных. Они могут использоваться при определении или какого-либо специфического формата, или наоборот, часто употребляемого. Например, когда в нескольких таблицах базы данных имеется поле с одинаковыми свойствами (типом, размером или ограничениями на значение).

Создание пользовательских типов данных в системе SQL-сервер может осуществляться следующими способами<sup>1</sup>:

**Первый способ:** При помощи процедуры *sp\_addtype*

#### Синтаксис

*sp\_addtype* тип, системный\_тип\_данных [, 'NULL' | 'NOT NULL']

**Упражнение:** С помощью процедуры *sp\_addtype* создайте пользовательские типы данных для **Почтового индекса (Postcode)** и **Номера отделения (объекта, владельца и т.д.)** в базе данных DreamHome. Типы данных указаны в таблице:

Имя пользовательского типа	Стандартный тип данных	Описание данных
Postcode	char	Строка из 6 символов, допускающая значения NULL
Member_no	smallint	Целое число, не превышающее 30000

1. Введите и выполните следующие операторы:

```
EXEC sp_addtype postcode, 'char (6)', NULL
```

```
EXEC sp_addtype member_no, 'smallint'
```

Удаление пользовательского типа данных осуществляется при помощи процедуры *sp\_droptype*. Если на пользовательский тип ссылаются таблицы или объекты базы данных, его нельзя удалить.

**Второй способ:** С помощью обозревателя MS SQL Server Management Studio.

<sup>1</sup> Перед созданием пользовательских типов данных, выполните п.1 в разделе Порядок выполнения работы.

Для этого следует:

1. Выбрать в базе данных **DreamHome** группу **Программирование/Типы/Определяемые пользователем типы данных**. В результате выполнения этой операции на экран будет выведено диалоговое окно настройки нового типа данных.
2. В поле *Имя* этого диалогового окна следует указать его имя; выпадающий список *Тип данных* определяет стандартный тип данных, относительно которого будет создан пользовательский. Размерность нового типа данных указывается в поле *Точность*, а проверку наличия NULL-значений можно определить, воспользовавшись флажком *Разрешить значения NULL*.

Теперь при указании типа данных поля таблицы в списке стандартных типов данных будет присутствовать вновь созданный пользовательский тип, при выборе которого будут установлены описанные выше параметры.

При создании пользовательских типов данных следует придерживаться следующих правил:

3. Если длина поля варьируется, использовать типы данных переменной длины. Например, список имён лучше хранить в поле с типом *varchar*, чем *char* (имеющим фиксированную длину). Типы данных переменной длины помогают экономить дисковое пространство.
4. Целые и денежные типы данных, а также типы даты и времени поддерживают разные диапазоны значений в зависимости от объёма отведённой им памяти. Например, если для хранения уникальных идентификаторов объектов недвижимости используется тип *tinyint*, число объектов ограничивается 255.
5. Выбор числовых типов данных зависит от требуемой точности; обычно используется тип *decimal*.
6. Если объём информации превышает 8000 байт, следует использовать типы *text* или *image*. В противном случае - *binary*, *char* или *varchar*. Типы *char* или *varchar* более функциональны по сравнению с *text* и *image*.

### КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите основные способы создания базы данных в SQL Server.
2. В каких случаях создаются secondary файлы? Какой тип они имеют?
3. Какая команда служит для удаления базы данных?
4. Кому принадлежит право на удаление базы данных?
5. Какие ограничения существуют на удаление базы данных?
6. Чем отличаются типы данных Char и Varchar?
7. В каких случаях используются пользовательские типы данных?
8. Как создать пользовательский тип данных в SQL Server?

### ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Создать базу данных DreamHome (написать запрос).

В таблице перечислены её свойства:

Файл	Начальный размер, Мб	Шаг приращения размера, Мб	Максимальный размер файла, Мб
База данных	15	1	30
Журнал	2	1	7

2. Создайте с помощью команды `sp_addtype` пользовательский тип данных для **Номера телефона** (обычного, мобильного, пейджера, с междугородным кодом):

Имя пользовательского типа	Стандартный тип данных	Описание данных
Phonenumber	char	Строка из 17 символов

3. Создайте с помощью настроек MS SQL Server Management Studio пользовательский тип данных:

Имя пользовательского типа	Стандартный тип данных	Length	Precision	Scale
shortstring	varchar	20	0	0

4. Создайте базу данных в соответствии с вариантом индивидуального задания.



## ЛАБОРАТОРНАЯ РАБОТА №4

**Тема:** Создание объектов базы данных: таблиц, индексов и первичных ключей таблицы. Использование ограничений.

**Цель работы:** Освоить основные методы и приемы создания таблиц базы данных, редактирования структуры таблиц.

### ОСНОВНЫЕ СВЕДЕНИЯ

#### Таблицы баз данных

##### Создание, изменение структуры и удаление таблиц

В реляционных базах данных для хранения информации используются таблицы, представляющие собой подобие двумерных массивов. Создание таблицы базы данных в системе SQL-сервер может осуществляться следующими способами:

**Первый способ:** С помощью SQL-команды.

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

При создании таблицы необходимо указать ее имя, имена полей и их типы данных. Для каждого поля требуется указать тип данных и, при необходимости, другие параметры.

При определении таблицы можно разрешить или запретить использование в поле значений *NULL*. Если не указано, может ли поле содержать пустые значения, SQL Server определит это самостоятельно, в зависимости от стандартных параметров для текущей базы данных.

##### Синтаксис:

```
CREATE TABLE имя_таблицы  
(имя_поля тип_данных [NULL| NOT NULL]  
[, ...n])
```

Порядок расположения полей в таблице определяется тем, в какой последовательности они указаны в команде создания таблицы. Для создания полей, содержащих сгенерированные системой последовательные числовые значения, идентифицирующие каждую строку таблицы, можно использовать свойство *IDENTITY* (такие поля называются полями *IDENTITY*). Поле *IDENTITY* обычно служит первичным ключом.

##### Синтаксис:

```
CREATE TABLE имя_таблицы  
(имя_поля, числовой тип_данных
```

IDENTITY [(начальное\_значение, шаг)] [NOT NULL]

Используя свойство IDENTITY, необходимо иметь в виду следующее:

- в таблице может быть только одно поле IDENTITY;
- значение поля IDENTITY нельзя изменить;
- значения NULL в поле IDENTITY не допускаются;
- в поле IDENTITY должны использоваться целые (*int*, *smallint* или *tinyint*), числовые или десятичные типы данных, причём два последних надо задать с нулевой разрядностью. Выбирая тип поля, надо оценить возможное число записей таблицы;
- информацию об определении поля IDENTITY задают две системные функции: IDENT\_SEED (начальное значение) и IDENT\_INCR (шаг);
- получить значение ключа последней вставленной во время текущей сессии записи можно средствами функции @@IDENTITY.

**Упражнение:** При помощи оператора CREATE TABLE создайте таблицу OWNER в базе данных DreamHome\_db, определив её поля, как указано в таблице:

Имя поля	Тип данных	Значения NULL	Свойство IDENTITY
Owner_no	member_no	Не допускаются	Seed=1 Increment=1
FName	shortstring	Не допускаются	Нет
LName	shortstring	Не допускаются	Нет
City	shortstring	Не допускаются	Нет
Street	shortstring	Не допускаются	Нет
House	nchar	Не допускаются	Нет
Flat	smallint	Допускаются	Нет
Otel_no	phononumber	Допускаются	Нет

```
CREATE TABLE OWNER  
(Owner_no member_no IDENTITY(1,1) NOT NULL,  
  FName shortstring NOT NULL,  
  LName shortstring NOT NULL,  
  City shortstring NOT NULL,  
  Street shortstring NOT NULL,  
  House nchar(6) NOT NULL,  
  Flat smallint NULL,  
  Otel_no phononumber NULL)
```

Структуру таблицы можно изменить. Для этого используется оператор ALTER TABLE. Чаще всего с его помощью добавляют поля к таблице или изменяют их тип данных. Однако следует помнить, что новое поле станет последним по порядку в списке.

**Синтаксис:**

ALTER TABLE *таблица*

ADD *имя\_поля* *тип\_данных* [NULL| NOT NULL] [, ...n]

ALTER TABLE *таблица*

ALTER COLUMN *имя\_поля* *новый\_тип\_данных* [NULL| NOT NULL]

Для того чтобы иметь возможность удалить таблицу, пользователь должен быть ее собственником. Кроме того, перед удалением, SQL потребует очистки таблицы от данных, что позволяет избежать случайной и невосполнимой потери информации.

#### **Синтаксис:**

DROP TABLE *имя\_таблицы*

После выполнения этой команды, имя таблицы больше не распознается. Перед удалением необходимо убедиться в том, что эта таблица не ссылается на другую таблицу или она не используется в каком-либо представлении.

**Второй способ:** С помощью конструктора таблиц SQL Server Management Studio.


Для создания таблицы необходимо:

1. Выбрать в списке объектов созданной базы данных группу *Таблицы*, открыть ее контекстное меню и выполнить команду *Создать таблицу*, после чего на экране отобразится окно конструктора таблиц.
2. В колонку *Имя столбца* необходимо ввести название столбца таблицы, после чего определить его тип данных, воспользовавшись колонкой **Тип данных** окна дизайнера. Здесь в выпадающем списке отображается перечень всех доступных типов данных, определенных в SQL-сервере. В зависимости от типа данных система определит доступ к свойствам этих столбцов, значения которых задаются в нижней части окна *Свойства столбцов*.

В СУБД имеется поддержка *NULL* значений. С помощью SQL-сервера можно определить их использование в таблицах. Убрав флажок в колонке *Разрешить значения Null* для некоторого поля, можно потребовать обязательный ввод значений в это поле.

При создании таблицы можно определить свойство *IDENTITY* для какого-либо ее поля. Для этого в первую очередь надо убрать флажок *Разрешить значения Null*, чтобы избежать неопределенности информации. Следующим шагом будет установка значения *да* в строке *Спецификация идентификаторов*, после чего требуется ввести *Начальное значение идентификатора* и *Шаг приращения идентификатора*.

3. После создания таблицы ее надо сохранить. Для этого следует

воспользоваться кнопкой , расположенной на панели инструментов или подтвердить сохранение при закрытии конструктора таблиц.

При необходимости можно внести изменения в структуру таблицы после ее создания. Для этого надо вызвать конструктор таблиц, воспользовавшись командой *Проект* контекстного меню таблицы.

Если необходимо в процессе работы с SQL-сервером переименовать ранее созданную таблицу, то следует выбрать команду *Переименовать* контекстного меню таблицы.

Каждая таблица в SQL-сервере обладает рядом свойств, для просмотра которых необходимо воспользоваться командой *Свойства* контекстного меню таблицы.

Для удаления таблицы из базы данных SQL-сервера необходимо сначала выбрать ее в списке, после чего выполнить команду *Удалить* контекстного меню. Воспользовавшись кнопкой *Показать зависимости*, можно просмотреть перечень таблиц, связанных с данной таблицей.

**Упражнение:** Создайте таблицу **BRANCH** в базе данных DreamHome\_db при помощи конструктора таблиц.

Выберите *Таблицы* вашей базы данных и в контекстном меню выберите команду *Создать таблицу*.

Определите поля в соответствии со следующей таблицей (каждая строка соответствует одному полю):

Имя поля	Тип данных	Значения NULL	Флажок Allow Nulls
Branch_no	member_no	Не допускаются	Не установлен
Postcode	postcode	Допускаются	Установлен
City	shortstring	Не допускаются	Не установлен
Street	shortstring	Не допускаются	Не установлен
House	nchar(10)	Не допускаются	Не установлен
Btel_no	phononumber	Не допускаются	Не установлен
Fax_no	phononumber	Допускаются	Установлен

1. Нажмите кнопку сохранения таблицы, введите имя таблицы **BRANCH** в окно сохранения таблицы. Закройте окно конструктора таблиц.

## Создание индексов и первичных ключей таблицы

### Создание ключей в системе SQL-сервер

Одним из основных понятий баз данных, используемых при контроле целостности информации, является ключ. Разделяют первичные и внешние ключи. *Первичный ключ* (PRIMARY KEY) – это уникальное поле (или несколько полей), однозначно определяющее запись таблицы базы данных. *Внешние ключи* (FOREIGN KEY) – это поля таблицы, которые соответствуют первичным ключам из других таблиц.

Создать первичный ключ можно одним из следующих способов:

**Первый способ:** При создании таблицы с помощью SQL-команд.

При создании первичного ключа надо помнить, что поле не должно содержать Null – значений.

```
CREATE TABLE имя_таблицы  
(имя_поля тип_данных [(размер)] NOT NULL PRIMARY KEY,  
имя_поля тип_данных [(размер)][NULL| NOT NULL],  
...);
```

**Второй способ:** С помощью конструктора среды SQL Server Management Studio.

Для создания ключа необходимо:

1. При создании таблицы выбрать нужное поле и установить первичный ключ с использованием кнопки  панели инструментов

Если данная операция была выполнена корректно, то слева от имени поля должен появиться соответствующий значок. Удаление первичного ключа производится аналогично его установке.

**Упражнение:** Установите ключ в таблице BRANCH базы данных DreamHome\_db с помощью дизайнера таблиц:

Откройте список объектов таблицы, выберите таблицу BRANCH, затем пункт *Проект*

1. Выделите поле *Branch\_no*.

Щёлкните по кнопке   
Закройте окно и сохраните изменения.

### Создание индексов в системе SQL-сервер

**Индексом** называется упорядоченный список полей или групп полей в таблице. Таблицы могут иметь огромное количество записей, при этом записи не находятся в каком-либо определенном порядке, поэтому на их поиск по указанному критерию может потребоваться достаточно продолжительное время. Когда создается индекс в поле, база данных запоминает соответствующий порядок всех значений этого поля в области памяти.

Индексы могут состоять из нескольких полей, при этом первое поле является как бы главным, второе упорядочивается внутри первого, третье внутри второго и т.д.

Индексы представляют собой наборы уникальных значений для некоторой таблицы с соответствующими ссылками на данные, расположенные в самой таблице. Индексы являются удобным внутренним механизмом системы SQL-сервер, с помощью которого осуществляется доступ к данным наиболее оптимальным способом.

В индексы следует включать поля, к которым часто выдаются запросы при выполнении операций поиска. К ним относятся:

- основные ключи;

- внешние ключи, а также другие поля, часто используемые для соединения таблиц;
- поля, в которых производится поиск диапазонов ключевых значений;
- поля, к которым производится упорядоченный доступ.

Создать индекс можно несколькими способами:

**Первый способ:** С помощью оператора CREATE INDEX.

**Синтаксис:**

CREATE INDEX *имя\_индекса* ON *таблица* (*поле*[, ...*n*])

Таблица, для которой создается индекс, должна уже существовать и содержать имена индексируемых полей. При этом имя индекса не может быть использовано для чего-либо другого в базе данных и SQL сам решает, когда он необходим для работы и использует его автоматически.

Для создания уникальных (не содержащих повторяющихся значений) индексов используется ключевое слово UNIQUE в операторе CREATE INDEX (CREATE UNIQUE INDEX ...).

Для удаления индекса используется оператор DROP INDEX.

**Синтаксис:**

DROP INDEX *таблица.индекс*[, ...*n*]

**Второй способ:** С помощью конструктора SQL Server Management Studio.

Для создания индекса необходимо:

1. Выбрать необходимую таблицу из базы данных, для которой будет определяться индекс, и открыть список ее компонентов.
2. Выбрать пункт *Индексы* и, затем, в контекстном меню пункт *Создать индекс*. При этом на экране отобразится диалоговое окно *Создание индекса*. Указать имя индекса, выбрать тип индекса. Нажать кнопку *Добавить* и в открывшемся окне указать столбец (столбцы таблицы), участвующие в индексе. Описание дополнительных параметров (Группа *Параметры*) приведено в Приложении 7.

**Основные различия между понятиями индекс и ключ:**

- Использование первичного ключа требует уникальности данных в таблице по определенному полю, что можно также выполнить при создании уникального индекса. Однако SQL-сервер разрешает определить только один первичный ключ, тогда как уникальных индексов можно создавать несколько.
- При использовании первичного ключа запрещается возможность ввода NULL значений, тогда как при работе с уникальными индексами этот запрет не является обязательным, однако придерживаться его желательно.

**Ограничения**

Ограничения – рекомендуемый способ обеспечения целостности данных. Ограничения гарантируют корректность вводимых значений и связей между таблицами.

Есть ограничения целостности, немедленно проверяемые, и есть откладываемые. Откладываемые ограничения целостности поддерживаются механизмом транзакций и триггеров. Среди немедленно проверяемых ограничений выделяют следующие типы:

- ограничения целостности атрибутов, или полей: значения по умолчанию, задание обязательности или необязательности значений (NULL), задание условий на значения полей;
- ограничения целостности сущностей, или таблиц: значение первичного ключа, выражения, которые применимы ко всей таблице;
- ограничения ссылочной целостности: задание обязательности связи, т.е. задание обязательности или необязательности значений внешних ключей во взаимосвязанных отношениях, определение влияния изменений значений родительских ключей на значения внешних ключей.
- ограничения целостности, определяемой пользователем: пользовательский тип данных;

Различные типы ограничений, определяемые в SQL-сервере, описаны в Приложении 4.

Ограничения определяются в операторах **CREATE TABLE** и **ALTER TABLE**.

Для анализа ошибок целесообразно именовать все ограничения, особенно если таблица содержит несколько ограничений одного типа. Для именования ограничений используется ключевое слово **CONSTRAINT**.

#### **Синтаксис:**

```
CREATE TABLE имя_таблицы  
(  
    {<определение_поля>  
    |<ограничение_таблицы>}  
    [...n]  
)
```

Где

```
<определение_поля> ::= {имя_поля тип_данных}  
[[CONSTRAINT имя_ограничения]  
 {DEFAULT константное_выражение  
 |CHECK (логическое_выражение)  
 |PRIMARY KEY [CLUSTERED [NON CLUSTERED]  
 |UNIQUE [CLUSTERED [NON CLUSTERED]  
 |[FOREIGN KEY] REFERENCES таблица_на_которую_ссылаются  
 [(поле_таблицы_на_которую_ссылаются)]
```

```
}}  
[...n]
```

<ограничение\_таблицы>::=

```
[CONSTRAINT имя_ограничения]  
|CHECK (логическое_выражение)  
|PRIMARY KEY [CLUSTERED |NON CLUSTERED] (поле [...n])  
|UNIQUE [CLUSTERED |NON CLUSTERED] (поле [...n])  
|FOREIGN KEY  
    (поле [...n])  
    REFERENCES таблица,_на_которую_ссылаются  
[(имя_родительской_таблицы,_на_которую_ссылаются [...n])]  
}
```

Используя ограничения FOREIGN KEY можно не указывать список полей родительского ключа, если родительский ключ имеет ограничение PRIMARY KEY. А также, задавая это ограничение, можно использовать только слово REFERENCES.

Например, при создании таблицы **BUYER** это может выглядеть так:

```
Branch_no member_no NOT NULL,  
FOREIGN KEY(Branch_no) REFERENCES BRANCH(Branch_no)
```

или так:

```
Branch_no member_no NOT NULL REFERENCES BRANCH
```

В соответствии со стандартом, изменение или удаление значений родительского ключа не допускается. Это означает, что нельзя изменить или удалить данные об отделении (например, при его закрытии) из таблицы **BRANCH** до тех пор, пока в таблице **BUYER** имеются клиенты, у которых в поле Branch\_no содержится номер изменяемого или удаляемого отделения. Однако довольно часто возникает необходимость в таких изменениях. В таких случаях задаются каскадирования или ограничения действий.

При необходимости, чтобы изменить или удалить текущее ссылочное значение родительского ключа существует три возможности:

1. Запретить изменения.
2. Сделав изменения в родительском ключе, произвести изменения во внешнем ключе автоматически (каскадное изменение).
3. Сделать изменение в родительском ключе и установить внешний ключ в NULL-значение автоматически.

В пределах этих возможностей выполняются все команды модификации.

Итак, изменения в родительском ключе можно разделить на ограниченные (NO ACTION), каскадируемые (CASCADES) и пустые (NULL) изменения. Например, при изменении номера отделения, данные о новом значении поля Branch\_no должны быть переданы в таблицу **BUYER**. В этом



случае следует указать оператор UPDATE с каскадируемыми изменениями. То есть, при создании таблицы **BUYER** указать:  
ON UPDATE CASCADE.

**Упражнение:** Создайте таблицу **BUYER** в базе данных DreamHome с заданием ограничений.

Запишите оператор создания таблицы BUYER, предполагая наличие следующих ограничений целостности:

Для быстрой связи с покупателем должен быть задан, по крайней мере, один из двух телефонов: рабочий или домашний.

С таблицей **BRANCH** таблица **BUYER** связана обязательной связью, потому что когда потенциальный покупатель **обращается** в одно из отделений компании, данные о нём, заносятся в базу данных. Для моделирования этой связи при создании таблицы BUYER должен быть определён внешний ключ **Branch\_no** и значение его NOT NULL (таким образом, задаётся обязательность связи).

При создании таблицы должно быть указано условие UPDATE с каскадируемым эффектом.

```
CREATE TABLE BUYER
(Buyer_no member_no NOT NULL PRIMARY KEY,
 FName shortstring NOT NULL,
 LName shortstring NOT NULL,
 City shortstring NOT NULL,
 Street shortstring NOT NULL,
 House nchar(6) NOT NULL,
 Flat smallint NULL,
 Htel_no phonenumber NULL,
 Wtel_no phonenumber NULL,
 Prof_Rooms tinyint NOT NULL,
 Branch_no member_no NOT NULL,
 Max_Price money NOT NULL,
 CONSTRAINT FK_BUYER_BRANCH FOREIGN KEY REFERENCES
 BRANCH,
 ON UPDATE CASCADE,
 CHECK (Htel_no IS NOT NULL OR Wtel_no IS NOT NULL)2
 )
```

**Упражнение:** Создание именованного первичного ключа в таблице OWNER базы данных DreamHome\_db.

Напишите и выполните оператор, добавляющий ограничение PRIMARY KEY с именем **PK\_Owner**<sup>3</sup> для поля **Owner\_no** таблицы OWNER.

---

<sup>2</sup> Булевские операторы и предикаты описаны в лабораторной работе №6

```
USE DreamHome_db
ALTER TABLE OWNER
    ADD CONSTRAINT PK_Owner PRIMARY KEY
NONCLUSTERED(Owner_no)
GO
```

Для поддержки первичного ключа и уникальных ограничений ключа автоматически создаётся уникальный индекс.

Для создания ограничений для таблицы с помощью конструктора необходимо:

1. Выбрать таблицу в списке объектов базы данных и открыть список ее компонентов;
2. В контекстном меню *Ограничения*, выбрать *Создать ограничение*, после чего на экране отобразится диалоговое окно дизайнера таблиц и откроется окно *Проверочные ограничения*;
3. В поле *Выражение* ввести выражение ограничения;
4. Нажатие кнопки *Добавить* приведет к созданию нового ограничения, после чего в поле *Выражение* надо ввести SQL-команду проверки вводимого значения.

Например: SQL команда (ROOMS >=1 AND ROOMS <=5) или (ROOMS BETWEEN 1 AND 5) позволяет контролировать ввод значения (целое число не менее 1 и не более 5) в поле ROOMS таблицы PROPERTY.

**Упражнение:** Добавьте ограничение в таблицу **BRANCH** базы данных DreamHome\_db, гарантирующее, что номер телефона соответствует принятому формату номеров.

Выберите таблицу BRANCH, выберите *Ограничения*, в контекстном меню выберите *Создать ограничение*, в поле *Выражение* введите выражение ограничения: (Btel\_no LIKE '8(021[2-6][0-9])[0-9][0-9]-[0-9][0-9]-[0-9][0-9]')

Нажмите кнопку **Закреть**.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какой оператор служит для создания таблиц в базе данных?
2. С помощью какого оператора можно добавить поля к таблице или изменить их тип данных?
3. Какой оператор позволяет удалить таблицу в базе данных?
4. Охарактеризуйте понятия внешний ключ и первичный ключ.
5. Дайте определение индекса и способы создания индексов в SQL Server. В чем различие между понятиями индекс и ключ?
6. Для чего применяются ограничения?
7. Какие типы ограничений Вам известны?

---

<sup>3</sup> Имя ограничения состоит из краткого названия типа ограничения, символа подчёркивания, имени поля или таблицы и порядкового номера ограничения данного типа, если к одному объекту задаётся несколько ограничений одного типа

## ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Для базы данных DreamHome создайте таблицы STAFF и PROPERTY с помощью запросов. При создании таблиц определите первичные ключи (поле *Staff\_no*<sup>4</sup> таблицы STAFF, поле *Property\_no* таблицы PROPERTY). При создании таблиц для таблицы STAFF задайте необязательность значения внешнего ключа *Branch\_no*, а для таблицы PROPERTY – обязательность или необязательность значений внешних ключей: *Branch\_no* (Branch\_no member\_no NOT NULL), *Staff\_no* (Staff\_no member\_no NULL), *Owner\_no* (Owner\_no member\_no NOT NULL).
2. С помощью SQL – команд задайте ограничение FOREIGN KEY для таблиц STAFF и PROPERTY. Предусмотрите каскадное изменение значения Staff\_no в таблице PROPERTY при изменении значения этого поля в таблице STAFF.
3. Таблицу VIEWING создайте с помощью SQL -команд. При создании таблицы VIEWING определите два внешних ключа (поля *Property\_no*, *Buyer\_no*) и ограничение PRIMARY KEY (первичным ключом будет набор из этих двух полей).
4. Добавьте ограничения на ввод значений в поле **Rooms** таблицы PROPERTY и в поле **Sex** (М(м) мужской, Ж(ж) женский) таблицы STAFF с помощью дизайнера ограничений и SQL -команд соответственно.
5. Задайте стандартное значение (значение по умолчанию) ‘Т’ для поля **Ptel** таблицы PROPERTY одним из способов.
6. Для полей **FName**, **Position** таблицы STAFF, создайте индекс с помощью дизайнера индексов.
7. Для поля **Date\_View** таблицы VIEWING создайте индекс с помощью SQL - запроса.
8. Просмотрите свойства созданных таблиц.

---

<sup>4</sup> В качестве Номера сотрудника можно использовать № паспорта, соответственно тип данных: nchar(9)

## ЛАБОРАТОРНАЯ РАБОТА № 5

**Тема:** Использование диаграмм для разработки структуры базы данных. Ввод данных в таблицу. Модификация данных таблиц.

**Цель работы:** Освоить использование диаграмм в SQL-сервере для разработки структуры базы данных. Научиться вводить данные в таблицу различными способами. Освоить приемы и методы вставки, удаления и модификации строк.

### ОСНОВНЫЕ СВЕДЕНИЯ

#### Использование диаграмм

Процесс разработки баз данных начинается с создания структуры данных, в которой отображаются все объекты и связи между ними. В базах данных SQL-сервера существует объект *Диаграммы баз данных*, позволяющий в графическом виде разрабатывать структуру данных. С помощью этого объекта можно создавать таблицы, определять ключи, осуществлять связи между таблицами и т.д.

Для создания диаграммы базы данных необходимо:

1. В списке объектов выбрать группу *Диаграммы баз данных*
2. В контекстном меню команду *Создать диаграмму базы данных*. Данное действие приведет к запуску мастера разработки диаграмм;
3. Выбрать необходимые таблицы в окне *Добавление таблиц*;
4. Если все действия выполнены верно, то система выведет соответствующее сообщение, после чего откроется диалоговое окно дизайнера диаграмм.


В дизайнера диаграмм существует четыре основных режима отображения таблицы:


*Стандартный* – просмотр параметров полей таблицы, причем имеется возможность изменения структуры таблицы;

*Имена столбцов* – просмотр перечня полей таблицы, причем имеется возможность установки первичных ключей;

*Ключи* – просмотр только ключевых полей;

*Только имя* – только заголовки таблицы.

Выбор данных режимов осуществляется с помощью кнопок панели инструментов или контекстного меню. Если в диаграмму необходимо добавить текстовый комментарий, следует воспользоваться кнопкой .

Для создания новой таблицы в диаграмме следует нажать кнопку , после чего на экране отобразится запрос о вводе её имени. По завершении указания имени создаваемой таблицы, последняя появится на диаграмме в режиме *создания таблицы*..

Следующим этапом разработки структуры данных является создание реляционных связей с помощью внешних ключей. Для этого необходимо:

1. Выделить поле, которое является первичным ключом основной таблицы, щелкнув мышью на кнопке, расположенной слева от поля.
2. Не отпуская кнопку мыши перетащить его к полю, которое является соответствующим внешним ключом подчинённой таблицы. На экране отобразится диалоговое окно *Таблицы и столбцы*, в котором выведено имя связи, указаны таблицы первичного и внешнего ключа и имя поля связи.
3. После подтверждения создания внешнего ключа могут быть заданы дополнительные параметры настройки связи в окне *Связь по внешнему ключу* (см. рисунок 25):

*Проверить существующие данные при создании или повторном включении* – проверяет соответствие значений таблиц условиям данной связи по завершении процесса создания;

*Спецификация INSERT and UPDATE* – указание правил добавления и удаления: Нет действия, Каскадно, Присвоить Null, присвоить значение по умолчанию.

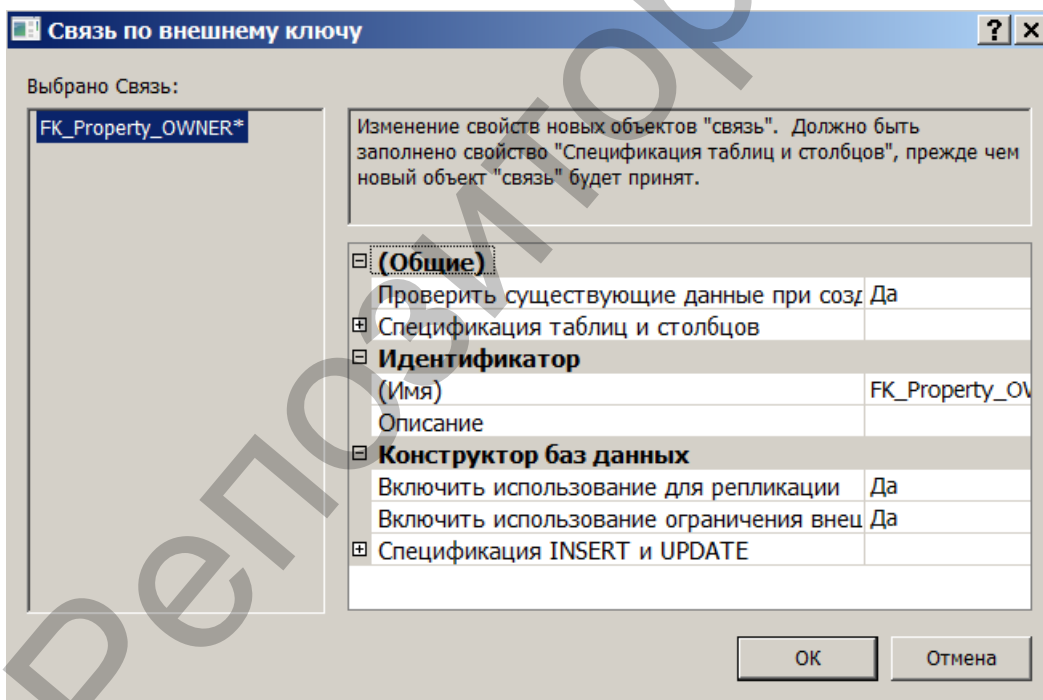


Рис. 25

При сохранении созданной диаграммы структуры данных система запросит имя диаграммы и разрешение на внесение изменений в реальные объекты базы данных. Надо определить – созданная диаграмма останется только на «листе», или все изменения необходимо внести в структуру

данных. Выбор кнопки *Yes* приведет к изменению структуры, после чего необходимо проверить корректность сделанных настроек.


**Упражнение:** *Создайте* диаграмму для базы данных DreamHome\_db. В дизайнера диаграмм создайте таблицу CONTRACT базы данных DreamHome\_db, определив её поля, как указано в таблице:

Имя поля	Тип данных	Значения NULL	Флажок Allow Nulls
<b>Sale_no</b>	member_no	Не допускаются	Не установлен
<i>Notary_Office</i>	shortstring	Не допускаются	Не установлен
<i>Date_Contract</i>	smalldatetime	Не допускаются	Не установлен
<i>Service_Cost</i>	money	Не допускаются	Не установлен
<b>Property_no</b>	member_no	Не допускаются	Не установлен
<b>Buyer_no</b>	member_no	Не допускаются	Не установлен

Выберите Диаграммы базы данных/Создать диаграмму..

Добавьте таблицы на диаграмму.

В диалоговом окне мастера создания диаграмм нажмите кнопку **Готово** для подтверждения осуществления взаимосвязи между данными в диаграмме и представленными таблицами.

Создайте таблицу CONTRACT. Для этого нажмите кнопку  на панели инструментов окна дизайнера диаграмм и в появившемся запросе введите имя CONTRACT. Определите поля созданной таблицы.

Установите первичный ключ для поля **Sale\_no**.

Установите связь между таблицами PROPERTY и CONTRACT по ключевому полю **Property\_no**, а также между таблицами BUYER и CONTRACT по ключевому полю **Buyer\_no**.

Сохраните созданную диаграмму с именем DIAGRAMMA.

### Ввод и модификация данных

**Первый способ<sup>5</sup>:** Ввод и модификацию данных таблиц можно осуществить с помощью SQL-запросов. Значения могут быть помещены или удалены из полей таблицы тремя операторами:

INSERT – вставить;

UPDATE – модифицировать;

DELETE – удалить.

**Второй способ:** Для ввода и изменения содержимого таблицы необходимо выполнить следующие действия:

---

<sup>5</sup> Этот способ будет рассматриваться в Лабораторной работе № 6 (Запросы на удаление, добавление, обновление данных)

1. выбрать требуемую таблицу в списке;
2. открыть контекстное меню и выбрать *Изменить первые 200 строк*; В рабочей области "Microsoft SQL Server Management Studio" проявится окно заполнения таблиц.

### **КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Для чего предназначены диаграммы в MS SQL Server?
2. Как создать диаграмму?
3. Можно ли в диаграмме создать новую таблицу?
4. Какими способами можно ввести данные в таблицы?
5. Как открыть окно для ввода данных в таблицы?

### **ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ**

1. Просмотрите свойства связей между таблицами в диаграмме базы данных DreamHome\_db.
2. Заполните таблицы BRANCH, OWNER, BUYER, STAFF, PROPERTY, VIEWING. Данные для таблиц находятся в [Приложении 4](#).

## ЛАБОРАТОРНАЯ РАБОТА № 6

**Тема:** Создание запросов на выборку и модификацию данных.

**Цель работы:** Изучить способы создания межтабличных запросов к базам данных, подчиненных запросов, запросов на удаление и обновление данных.

### ОСНОВНЫЕ СВЕДЕНИЯ

В SQL-сервер для манипулирования данными используется язык запросов Transact-SQL, который представляет собой версию языка SQL, переработанную компанией Microsoft.

В Transact-SQL присутствуют не только операции запросов на выборку и модификацию данных, но и операторы соответствующие DDL – языку описания данных. Кроме того, язык содержит операторы, предназначенные для управления (администрирования БД).

### Инструкция SELECT

Инструкция SELECT используется для отбора строк и столбцов таблиц базы данных.

#### Синтаксис:

```
SELECT [ALL|DISTINCT] набор_атрибутов  
FROM набор_отношений  
[WHERE условие_отбора_строк]  
[GROUP BY спецификация_группировки]  
[HAVING спецификация_выбора_групп]  
[ORDER BY спецификация_сортировки]
```

Ключевое слово ALL означает, что в результирующий набор строк включаются все строки, удовлетворяющие условиям запроса, в том числе и строки-дубликаты. Ключевое слово DISTINCT означает, что в результирующий запрос включаются только различные строки.

В разделе **SELECT** атрибуты могут указываться с помощью (\*). Например X.\* обозначает совокупность всех атрибутов отношения X, а изолированная \* – совокупность всех атрибутов всех отношений, фигурирующих в разделе FROM для создания запроса.

Таблицам могут быть присвоены имена – псевдонимы, что бывает полезно при соединении таблицы с самой собою или для доступа из вложенного подзапроса к текущей записи внешнего запроса. Псевдонимы задаются с помощью ключевого слова AS, которое может быть опущено.

Раздел **FROM** определяет таблицы или запросы, служащие источником данных. В случае если указано более одного имени таблицы, по умолчанию предполагается, что над перечисленными таблицами будет выполнена операция декартова произведения. Например, запрос

```
SELECT *  
FROM A, B
```



соответствует декартову произведению отношений A и B.

Для задания типа соединения таблиц в единый набор записей, из которого будет выбираться необходимая информация, в разделе FROM используются ключевые слова JOIN и ON. Ключевое слово JOIN и его параметры указывают соединяемые таблицы и методы соединения. Ключевое слово ON указывает общие для таблиц поля.

При внутреннем соединении таблиц (INNER JOIN или JOIN) сравниваются значения общих полей этих таблиц. В окончательный набор возвращаются только те записи, которые отвечают условиям соединения.

Операция LEFT JOIN возвращает все строки из первой таблицы, соединенные с теми строками второй, для которых выполняется условие соединения.

Если во второй таблице таких строк нет, возвращаются значения NULL в строках второй таблицы. Аналогично, операция RIGHT JOIN возвращает все строки второй таблицы, соединенные с теми строками первой, для которых выполняется условие объединения.

Операции LEFT JOIN или RIGHT JOIN могут быть вложены в операцию INNER JOIN, но операция INNER JOIN не может быть вложена в операцию LEFT JOIN или RIGHT JOIN.

Раздел **WHERE** задает условия отбора строк. Имена атрибутов, входящие в предложение WHERE могут не входить в набор атрибутов, перечисленных в предложении SELECT.

В выражении условий раздела WHERE могут быть использованы следующие предикаты:

- Предикаты сравнения {=, >, <, >=, <=, <>.}.
- Предикат BETWEEN A AND B. Предикат истинен, когда сравниваемое значение попадает в заданный диапазон, включая границы диапазона.
- Предикат вхождения во множество IN (множество) истинен тогда, когда сравниваемое значение входит во множество заданных значений. При этом множество может быть задано простым перечислением или встроением подзапросом. Одновременно существует противоположный предикат NOT IN (множество).
- Предикаты сравнения с образцом LIKE и NOT LIKE. Предикат LIKE требует задания шаблона, с которым сравнивается заданное значение.
- Предикат сравнения с неопределенным значением IS NULL. Неопределенное значение интерпретируется в реляционной модели как значение, неизвестное в данный момент времени. Это значение при появлении некоторой дополнительной информации в любой момент времени может быть заменено некоторым конкретным значением.

- Предикаты существования EXISTS и не существования NOT EXISTS.

Когда запрос включает предложение WHERE, СУБД просматривает всю таблицу по одной записи, чтобы определить является ли предикат истинным. Предикат может включать неограниченное число условий, содержащих булевы операторы. Стандартными булевыми операторами в SQL являются AND, OR и NOT.

**Упражнение:** С помощью SQL –команд создайте запросы вывода:

- перечня адресов трёхкомнатных квартир, предлагаемых для продажи в Полоцке;

```
SELECT Property_no, Street, House, Flat
FROM PROPERTY
WHERE City='Полоцк' AND Rooms=3;
```

- дат приема на работу сотрудников отделения №4;

```
SELECT Staff_no, FName, LName, Date_Joined
FROM STAFF
WHERE Branch_no=4;
```

- перечня объектов собственности, принадлежащих каждому владельцу собственности;

```
SELECT OWNER.Owner_no, OWNER.FName, OWNER.LName,
PROPERTY.Property_no, PROPERTY.City, PROPERTY.Street,
PROPERTY.House, PROPERTY.Flat
FROM OWNER, PROPERTY
WHERE OWNER.Owner_no=PROPERTY.Owner_no;
```

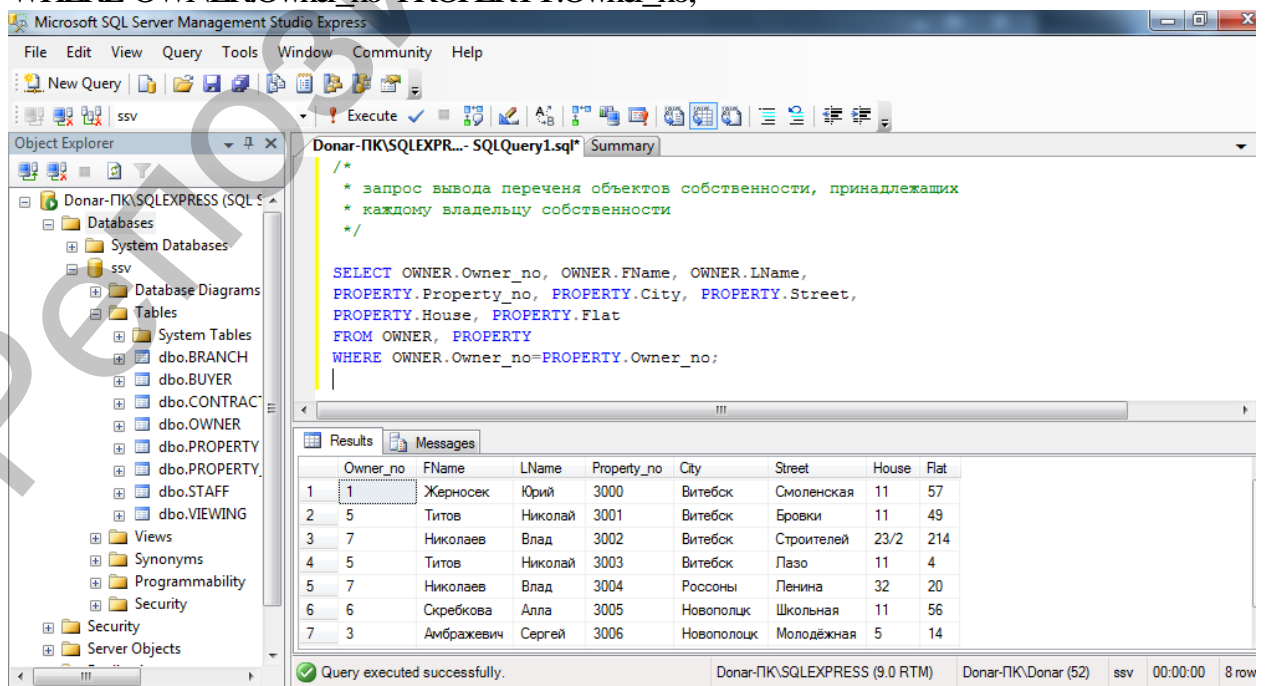


Рис. 26

- список отделений компании, которые предлагают трехкомнатные квартиры с телефонами;

```
SELECT BRANCH.Branch_no
FROM BRANCH INNER JOIN PROPERTY ON
BRANCH.Branch_no=PROPERTY.Branch_no
WHERE (PROPERTY.Rooms=3) AND (PROPERTY.Ptel='T');
```

- список шифров владельцев собственности (Owner\_no), предлагающих несколько трехкомнатных квартир для продажи;

```
SELECT DISTINCT a.Owner_no
FROM PROPERTY a, PROPERTY b
WHERE a.Owner_no=b.Owner_no AND
a.Property_no<>b.Property_no AND
a.Rooms=3 AND b.Rooms=3;
```

В запросе используются псевдонимы а и б таблицы PROPERTY, так как для выполнения запроса необходимо оценить равенство поля Owner\_no в двух экземплярах одной и той же таблицы.

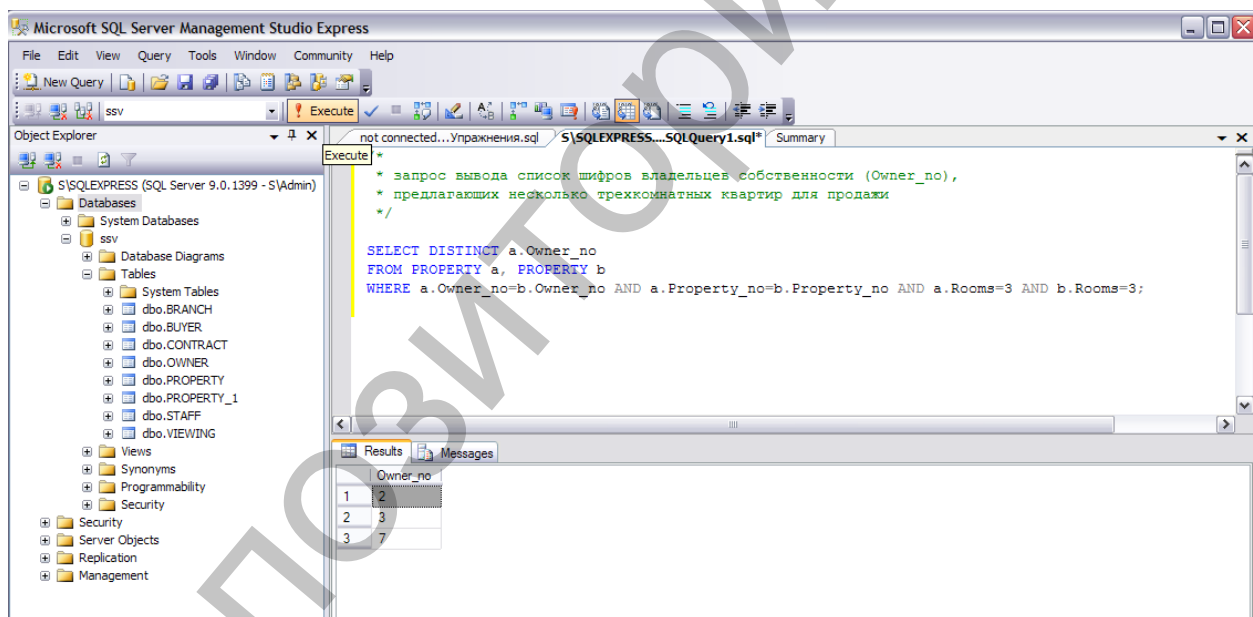


рис. 27

Раздел **GROUP BY** используется для создания итоговых запросов. Итоговые запросы имеют одно общее свойство: в предложении SELECT таких запросов используется, по крайней мере, одна агрегатная функция. (AVG, COUNT (количество непустых значений в данном столбце), SUM, MIN, MAX, FIRST (значение столбца из первой строки результирующего набора записей), LAST(значение столбца из последней строки результирующего набора записей) и др. Агрегатные функции используются подобно именам полей в операторе SELECT, но с одним исключением: они берут имя поля как аргумент. С функциями SUM и AVG могут

использоваться только числовые поля. С функциями COUNT, MAX и MIN могут использоваться как числовые, так и символьные поля.

**Синтаксис:** GROUP BY *имя\_столбца*

Имя столбца – имя любого столбца из любой из упомянутой в разделе FROM таблицы.

Если GROUP BY расположено после WHERE создаются группы из строк, выбранных после применения раздела WHERE.

При включении раздела GROUP BY в инструкцию SELECT список полей должен состоять из итоговых функций SQL и из имен столбцов, указанных в разделе GROUP BY. В раздел GROUP BY должны быть включены все атрибуты, входящие в раздел SELECT.

**Итоговые функции SQL:**

- AVG(поле) - выводит среднее значение поля;
- COUNT(\*) - выводит количество записей в таблице;
- COUNT(поле) - выводит количество всех значений поля;
- MAX(поле) - выводит максимальное значение поля;
- MIN(поле) - выводит минимальное значение поля;
- STDEV(поле) - выводит среднееквадратичное отклонение всех значений поля;
- STDEVP(поле) - выводит среднееквадратичное отклонение различных значений поля;
- SUM(поле) - суммирует все значения поля;
- TOP n [Percent] - выводит n первых записей из таблицы, либо n% записей из таблицы;
- VAR(поле) - выводит дисперсию всех значений поля;
- VARP(поле) - выводит дисперсию всех различных значений поля.

В предложение GROUP BY могут быть указаны одновременно несколько столбцов. Группы при этом определяются слева направо. Предложение GROUP BY автоматически устанавливает сортировку по возрастанию (если надо по убыванию – задать в ORDER BY).

**Упражнение:** С помощью SQL- команд создайте итоговые запросы:

- Вычисления средней зарплаты сотрудников по каждому из отделений компании;

- Подсчёта количества трехкомнатных квартир, предлагаемых в Витебске и Полоцке.

```
SELECT STAFF.Branch_no, Avg(STAFF.Salary) AS Средняя_зарплата
FROM STAFF
GROUP BY Branch_no;
```

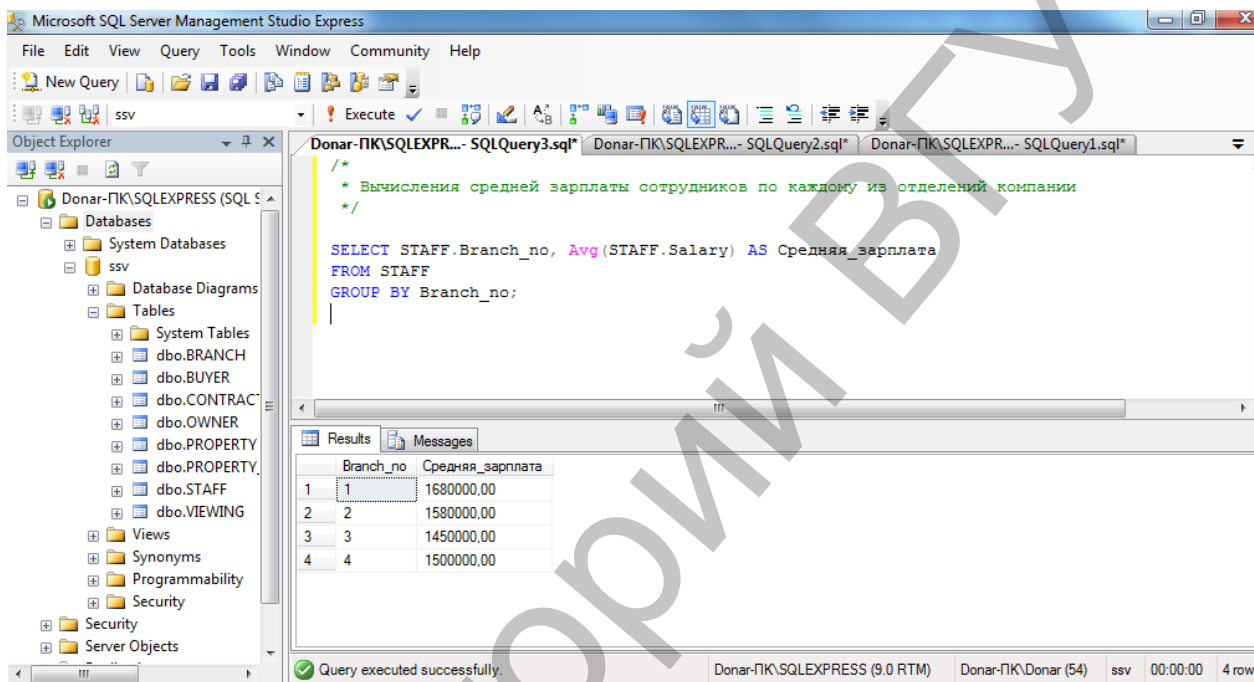


Рис. 28

```
SELECT City, COUNT(*) AS Количество_квартир
FROM PROPERTY
WHERE (Rooms=3) AND ((City='Витебск') OR (City='Полоцк'))
GROUP BY City;
```

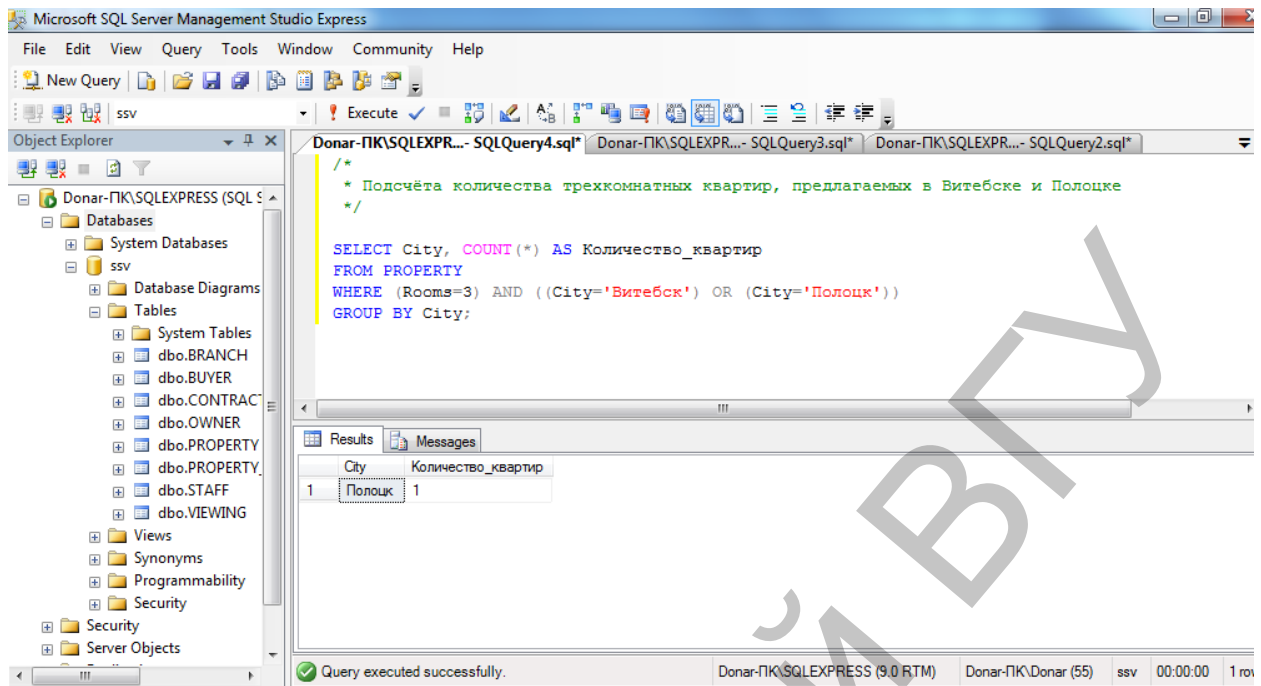


Рис. 29

Раздел **HAVING** задает условие отбора групп строк, которые включаются в таблицу, определяемую инструкцией SELECT.

Условия отбора применяется к столбцам, указанным в разделе GROUP BY, к столбцам итоговых функций или к выражениям, содержащим итоговые функции. Если некоторая группа не удовлетворяет условию отбора, она не попадает в набор записей.

**Синтаксис:** HAVING *условие\_отбора*

Разница между HAVING и WHERE заключается в том, что условие отбора, заданное в разделе WHERE применяется к отдельным записям, перед их группировкой, а условие отбора раздела HAVING применяется к группам строк.

Если раздел GROUP BY находится перед HAVING, условие отбора применяется к каждой из групп, сформированных на основе совпадения значений в заданных столбцах. В случае отсутствия раздела GROUP BY условие отбора применяется ко всей таблице определенной инструкцией SELECT. Агрегатные функции могут применяться как в выражении вывода результатов строки SELECT, так и в выражении обработки сформированных групп HAVING.

**Упражнение:** Выведите список и номера телефонов отделений, которые предлагают более одной трехкомнатной квартиры.

```

SELECT PROPERTY.Branch_no, BRANCH. Btel_no
FROM BRANCH, PROPERTY
WHERE PROPERTY.Branch_no=BRANCH.Branch_no
AND PROPERTY.Rooms=3
GROUP BY PROPERTY.Branch_no, BRANCH. Btel_no

```

HAVING COUNT(\*)>1;

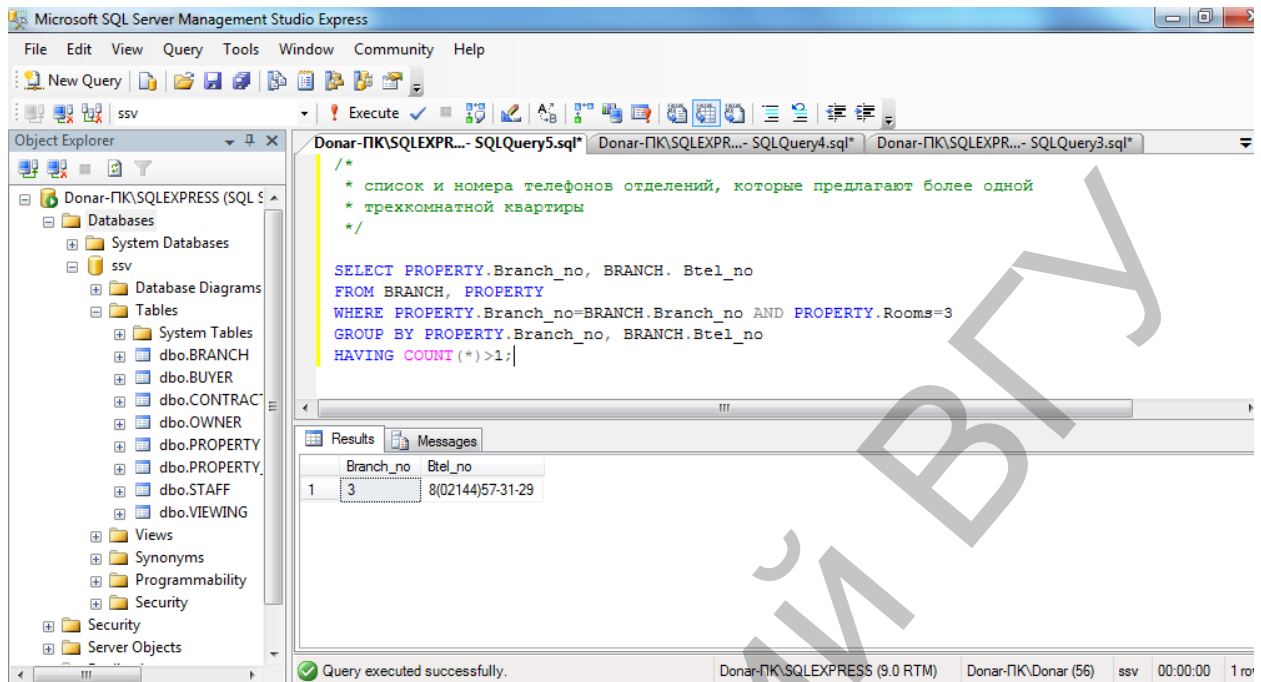


Рис. 30

### Сортировка результатов запроса

В SQL представлены специальные средства, которые позволяют совершенствовать вывод запросов:

- размещение текста в выводе запроса:

SELECT имя\_поля1, 'текст', имя\_поля2 ...

При этом все символы, в том числе и пробелы, вставляются в вывод, поэтому этот способ можно использовать для маркировки вывода вместе со вставляемыми комментариями.

- упорядочение полей вывода:

ORDER BY имя\_поля ASC|DESC;

Если указывается несколько полей, то столбцы вывода упорядочиваются один внутри другого, при этом можно определить ASC (возрастание) или DESC (убывание).

**Упражнение:** Определите количество объектов, находящихся в ведении каждого из сотрудников компании с упорядочением отделений по убыванию:

```
SELECT STAFF.Branch_no, STAFF.Staff_no, Count(*) AS
Count_Staff_no
FROM STAFF INNER JOIN PROPERTY ON STAFF.Staff_no =
PROPERTY.Staff_no
GROUP BY STAFF.Branch_no, STAFF.Staff_no
ORDER BY STAFF.Branch_no DESC, STAFF.Staff_no;
```

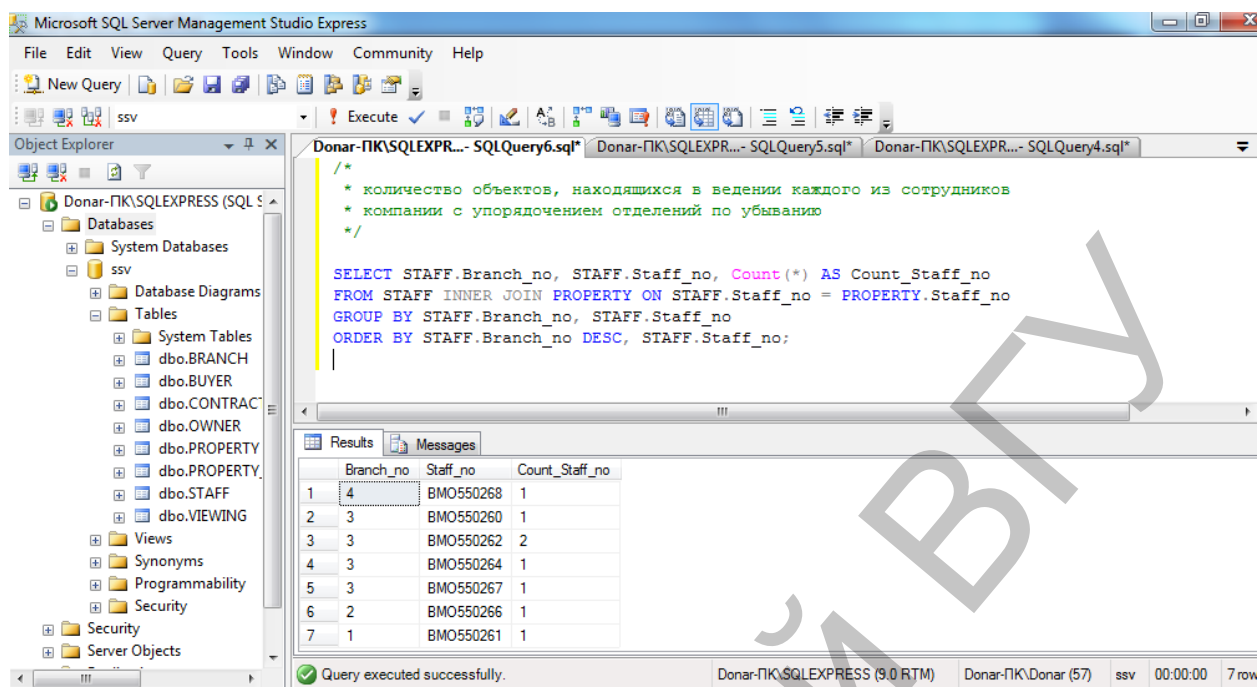


Рис. 31

### Вложение запросов

Одни запросы могут управлять другими запросами. Это можно сделать, разместив один запрос внутри другого запроса. Обычно подчиненный запрос генерирует значение, которое проверяется в предикате внешнего запроса (в предложении WHERE или HAVING), определяющего верно оно или нет. Совместно с подзапросом можно использовать предикат EXISTS, который возвращает истину, если вывод подзапроса не пуст.

Часто бывает необходимо сравнивать значения в определенных столбцах со списком значений этого же столбца из другой таблицы или запроса. В подобных случаях используется ключевое слово IN (NOT IN).

**Упражнение:** Выведите список сотрудников, за которыми не закреплен ни один из объектов недвижимости.

```

SELECT *
FROM STAFF
WHERE Staff_no NOT IN (SELECT Staff_no FROM PROPERTY);

```



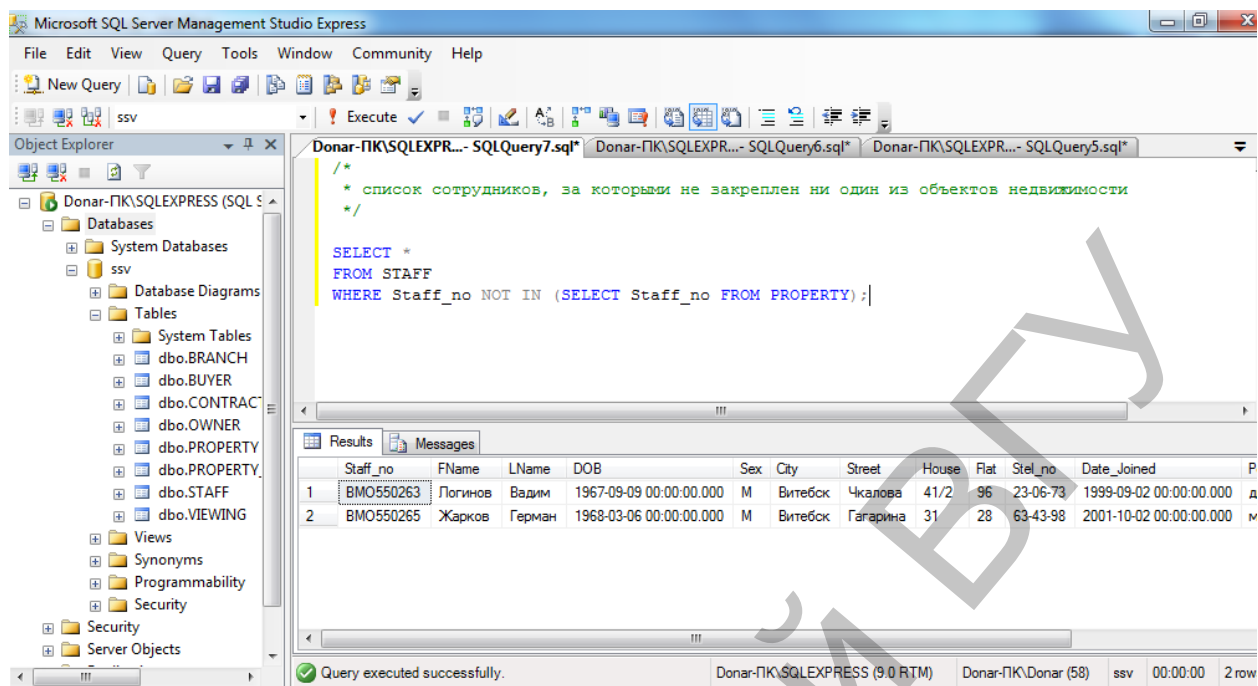


Рис. 32

В подзапросах допускается использование агрегатных функций, например, для вывода списка трехкомнатных квартир, цена которых превышает среднюю цену трехкомнатной квартиры, используется запрос:

```

SELECT City, Street, House, Flat
FROM PROPERTY
WHERE Rooms=3
AND Selling_Price > (SELECT AVG(Selling_Price) FROM Property
WHERE Rooms=3);

```

**Упражнение:** Выведите список владельцев собственности, чьи объекты были осмотрены в определенный день:

```

SELECT OWNER.Owner_no, FName, LName
FROM OWNER INNER JOIN PROPERTY
ON PROPERTY.Owner_no=OWNER.Owner_no
WHERE PROPERTY.Property_no=ANY(SELECT Property_no
FROM VIEWING
WHERE Date_View='17.01.2012');

```

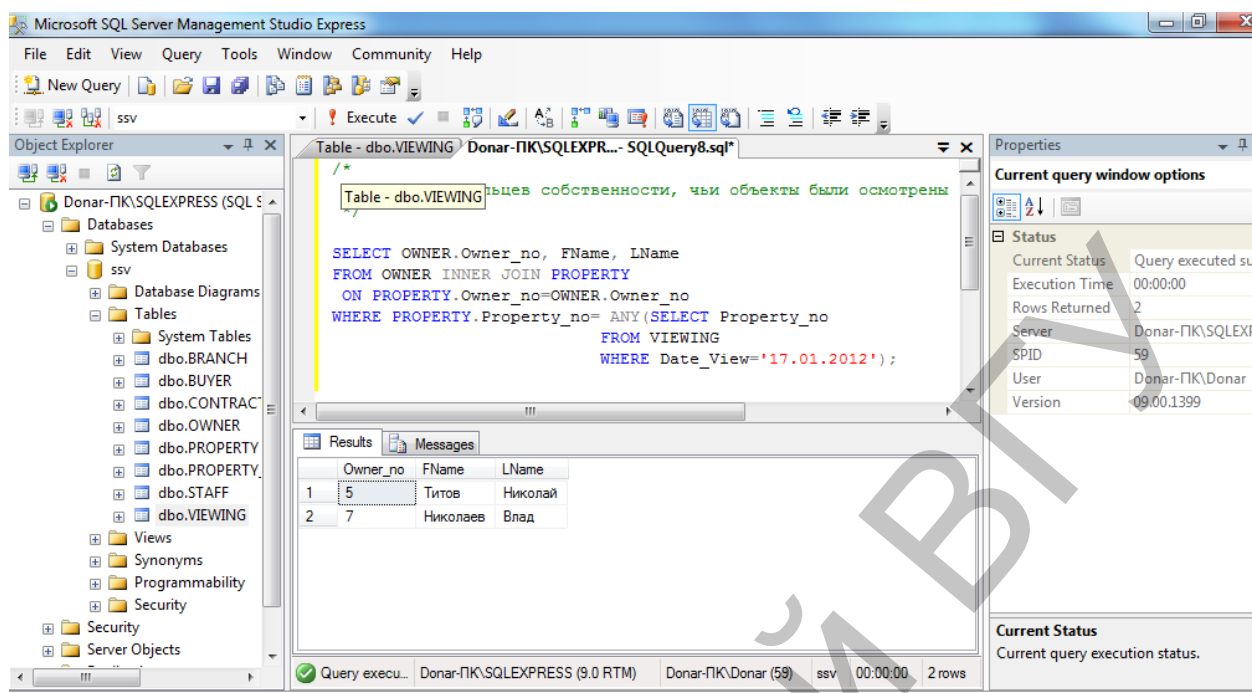


Рис. 33

В таблице VIEWING будет найдена соответствующая дата и передана в предложение WHERE. После определения даты в основном запросе из таблицы PROPERTY будут отобраны записи, удовлетворяющие заданному условию.

(В данном примере предполагается, что подзапрос должен вернуть только одно значение).

Если подчиненный запрос возвращает более одного значения, использование запроса в таком виде приведет к ошибке. В тех случаях, когда подчиненный запрос возвращает более одной строки необходимо использовать следующие ключевые слова: ANY, SOME, ALL. Подчиненный запрос в этом случае должен возвращать один столбец.

- ANY или SOME – возвращает TRUE, если заданное выражение является истинным для какой-нибудь из строк возвращаемой запросом.
- ALL - возвращает TRUE, если заданное выражение является истинным для всех строк возвращаемой запросом.

**Упражнение:** Выведите список объектов собственности, которые были осмотрены покупателями (присутствуют в таблице VIEWING):

```
SELECT *
FROM PROPERTY
WHERE Property_no =ANY (SELECT Property_no
FROM VIEWING);
```

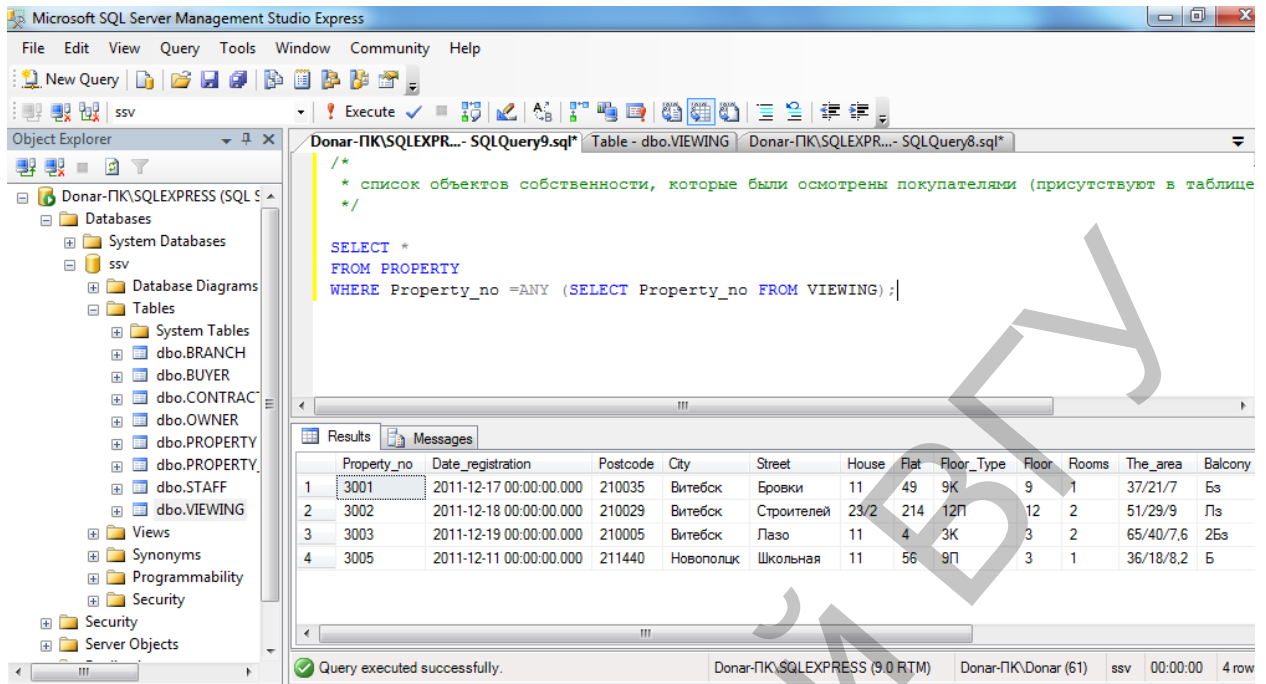


Рис. 34

Этот же результат может быть получен с помощью оператора IN

```

SELECT Property_no
FROM PROPERTY
WHERE Property_no IN (SELECT Property_no FROM VIEWING);

```

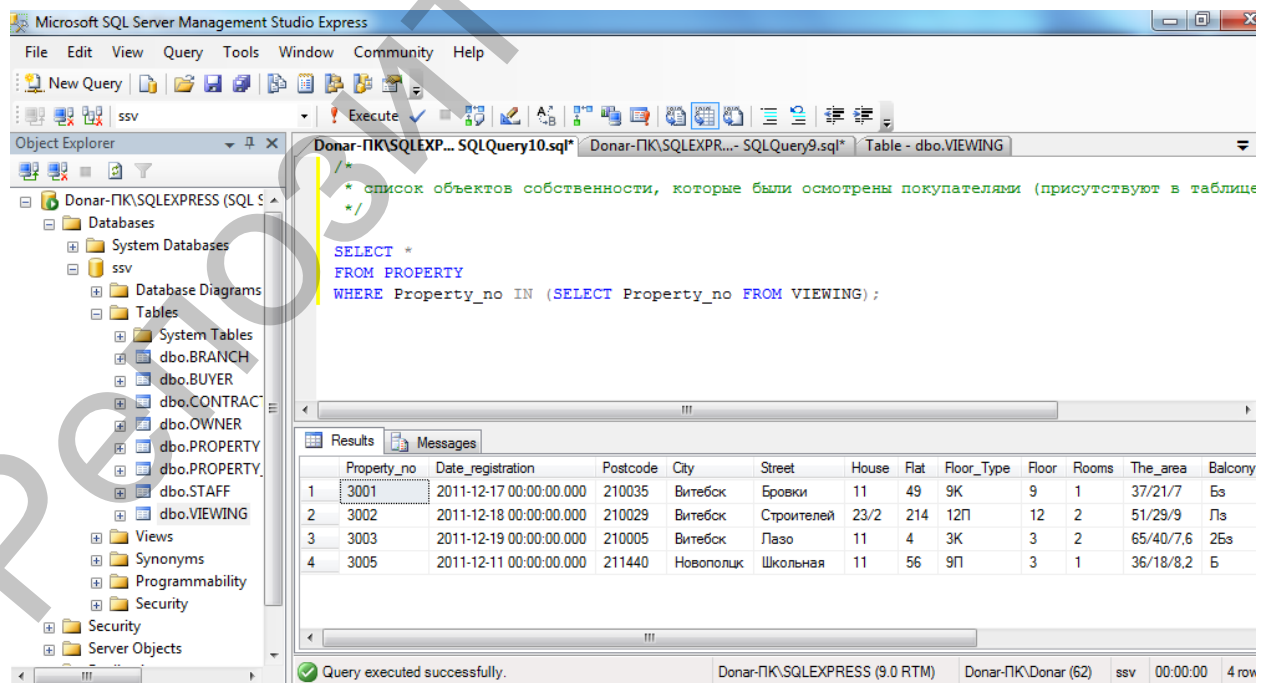


Рис. 35

Оператор ALL работает таким образом, что предикат является верным, если каждое значение, выбранное подзапросом, удовлетворяет условию в предикате внешнего запроса.

**Упражнение:** Найдите всех сотрудников, чья заработная плата выше заработной платы любого из сотрудников отделения компании под номером 3:

```
SELECT Staff_no, FName, LName, Salary
FROM STAFF
WHERE Salary > ALL (SELECT Salary FROM STAFF
WHERE Branch_no=3);
```

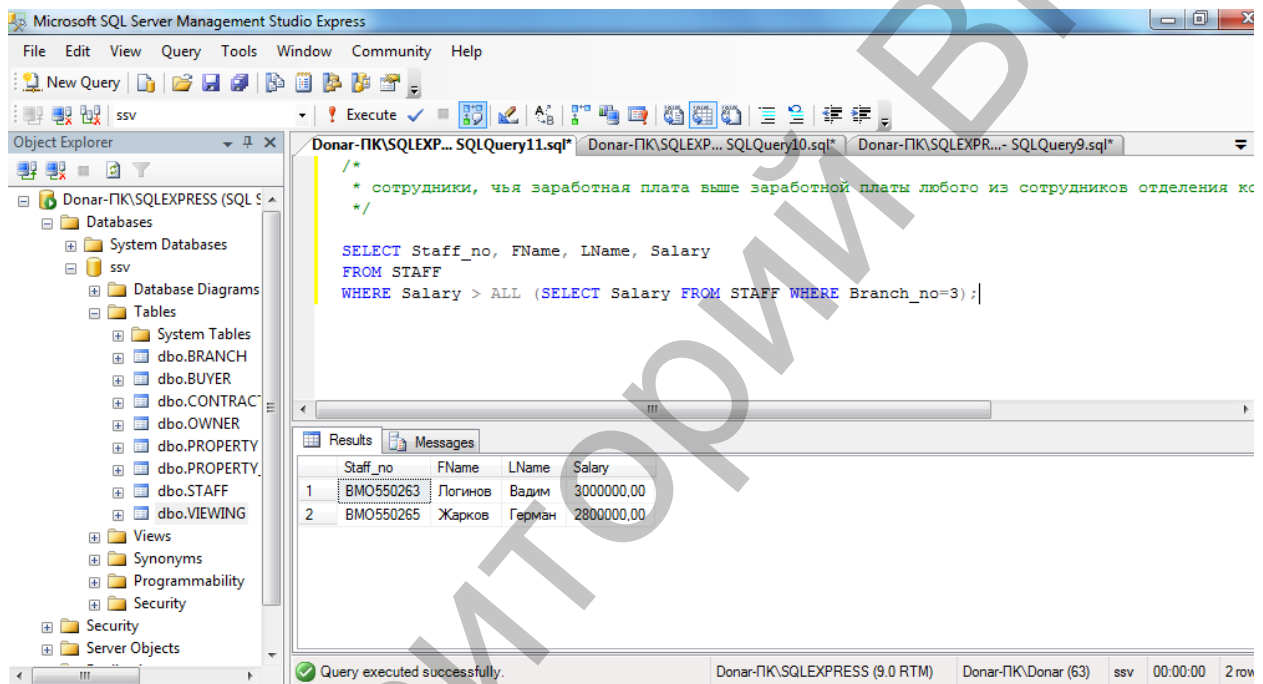


Рис. 36

### Использование оператора EXISTS

Оператор EXISTS проверяет, возвращает ли подчиненный запрос хотя бы одну строку. Для проверки противоположного значения используется предикат NOT EXISTS.

**Упражнение:** Выведите данные об объектах собственности из таблицы PROPERTY только в том случае, если хотя бы один из них был осмотрен покупателями, и было получено согласие на приобретение:

```
SELECT Property_no
FROM PROPERTY
WHERE EXISTS (SELECT Property_no FROM VIEWING
WHERE Comments='согласен');
```

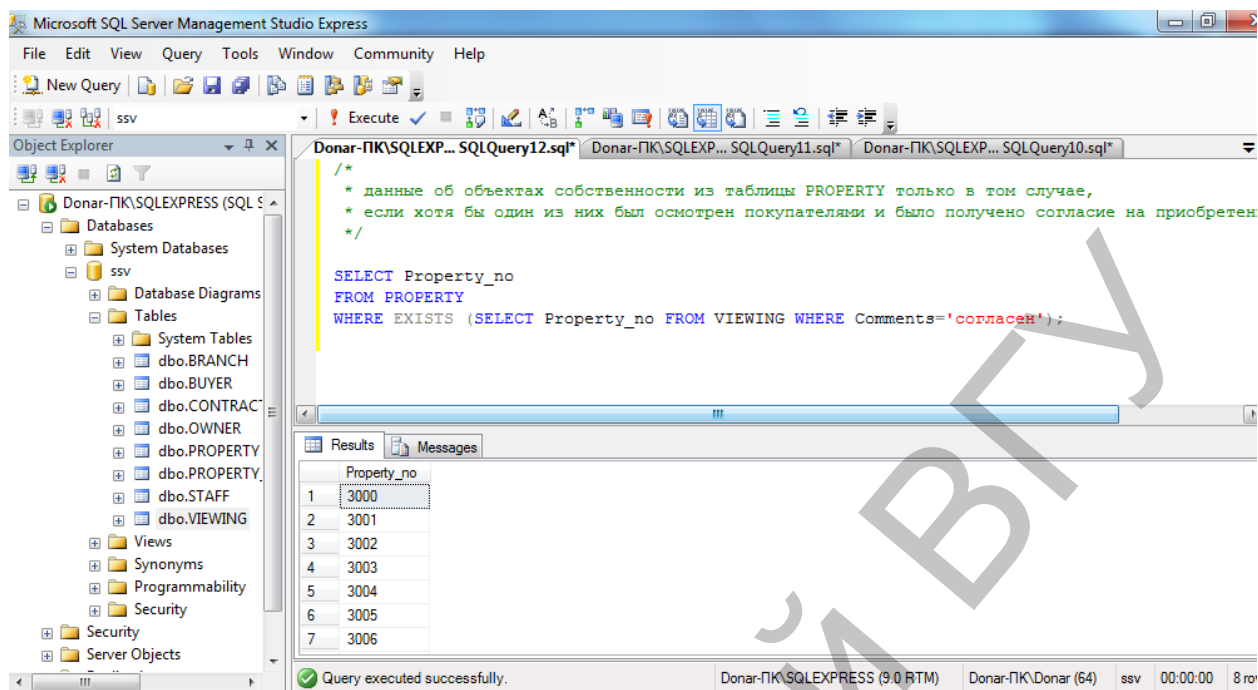


Рис. 37

### Создание таблицы из набора результатов

При помощи оператора можно поместить набор результатов запроса новую таблицу. Кроме того, этот оператор позволяет создавать и заполнять новые таблицы, а также создавать временные таблицы. Запросы к временной таблице иногда оказываются проще тех, которые пришлось бы выполнять, обращаясь к нескольким таблицам или базам данных. Оператор SELECT INTO позволяет создать локальную или глобальную временную таблицу. Для локальных таблиц используются имена, начинающиеся с символа #, а для глобальных – с символа ##.

**Упражнение:** Создайте таблицу, содержащую объекты собственности, находящиеся в городе Полоцке.

```

SELECT *
INTO ##PROPERTY_POLOCK
FROM PROPERTY
WHERE City='Полоцк';

```

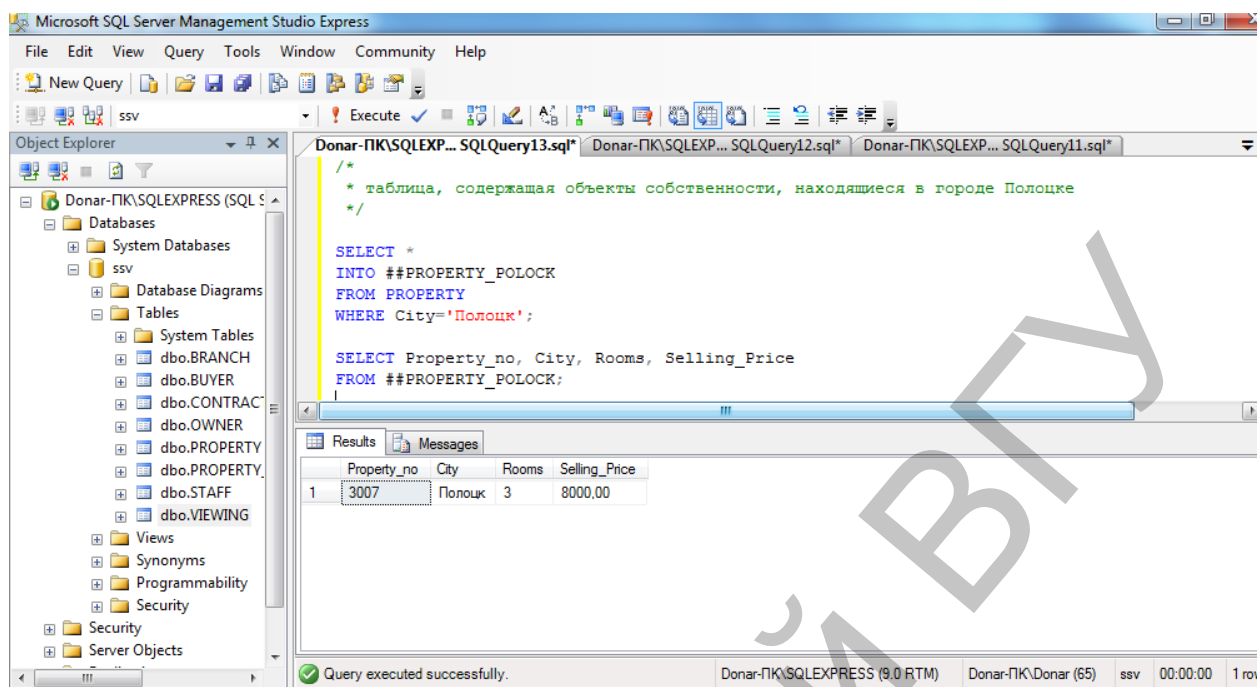


Рис. 38

### Запросы на модификацию данных

SQL позволяет не только создавать запросы, но и вносить изменения в данные. Для этого используются запросы на удаление, вставку и обновление данных.

#### Запросы на удаление

Удаление строк из таблицы можно осуществить с помощью оператора DELETE. Следует учитывать, что оператор удаляет только целые записи таблицы, а не индивидуальные значения того или иного поля.

##### Синтаксис:

```

DELETE [таблица.*]
FROM таблица
WHERE условие_отбора

```

**Упражнение:** Удалите из таблицы OWNER все записи, относящиеся к владельцу собственности, у которого значение поля Owner\_no=10:

```

DELETE
FROM OWNER
WHERE OWNER_no=10;

```

В команде удаления возможно использование вложенного запроса. Это может быть необходимо в тех случаях, когда критерий, по которому выбираются данные, базируется на другой таблице.

#### Запросы на добавление

Ввод и добавление записей в SQL осуществляется с помощью оператора INSERT. Существует несколько вариантов вставки данных.

### Вставка записей из другой таблицы

Оператор **INSERT** добавляет записи в уже существующую таблицу, вставляя в нее набор результатов оператора **SELECT**

#### Синтаксис:

```
INSERT [INTO] имя_таблицы
  SELECT список_выборки
  FROM список_таблиц
  WHERE условие_поиска
```

#### Добавление данных в указанные поля

Наиболее употребительный вариант команды **INSERT INTO** предусматривает добавление записи в существующую таблицу с указанием списка полей:

#### Синтаксис:

```
INSERT INTO имя_таблицы (поле1, поле2,...)
  VALUES (значение_поля1, значение_поля2...)
```

При этом если перечислены не все поля, то в не перечисленные поля автоматически устанавливается значение NULL.

Если задается полный список значений новой записи, форма записи становится более короткой, так как перечень заполняемых полей после имени таблицы может не задаваться. Порядок следования значений после служебного слова **VALUES** должен соответствовать структуре таблицы.

#### Синтаксис:

```
INSERT INTO имя_таблицы
  VALUES (список_значений)
```

**Упражнение:** Отберите из таблицы **BUYER** покупателей, проживающих в Витебске, и поместите их в таблицу **BUYER\_1**. Таблица **BUYER\_1** должна быть заранее создана командой **CREATE TABLE**.

```
INSERT INTO BUYER_1
  SELECT Buyer_no, FName, LName
  FROM BUYER
  WHERE City = 'Витебск';
```

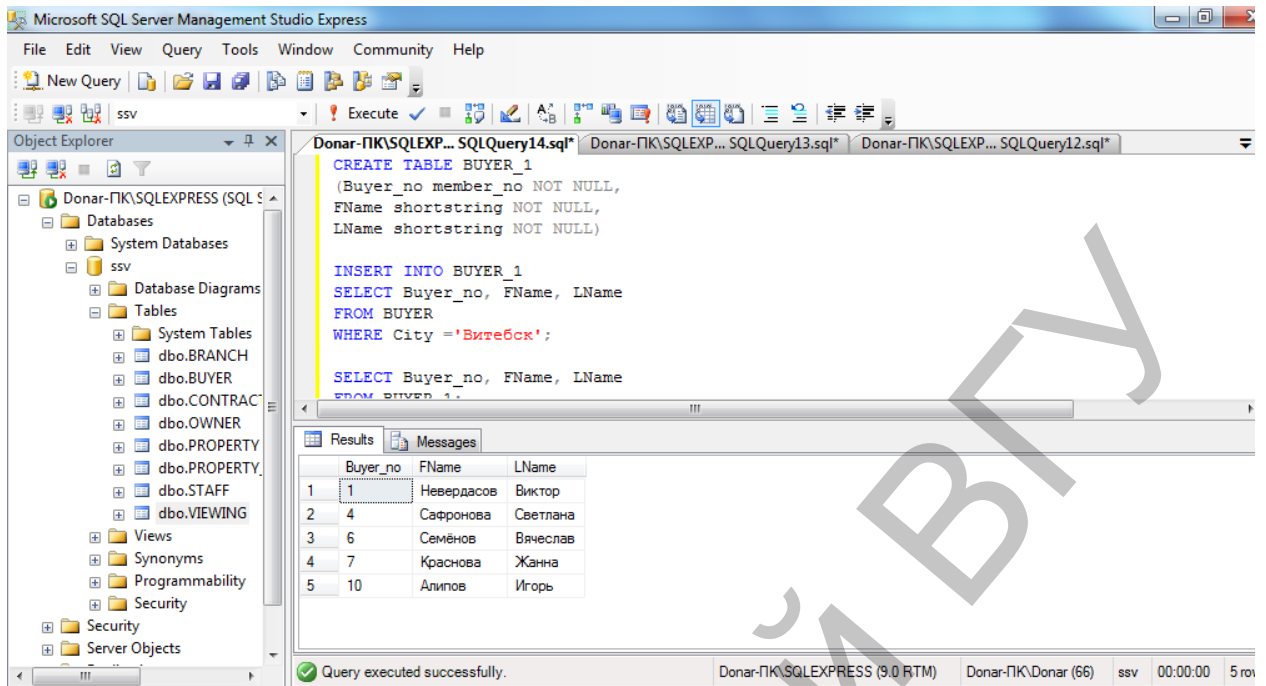


Рис. 39

В INSERT можно использовать подзапросы.

**Упражнение:** Вставьте в таблицу BUYER\_2 данные только тех покупателей, которые приобрели объекты собственности.

```

INSERT INTO BUYER_2
SELECT Buyer_no, FName, LName
FROM BUYER
WHERE Buyer_no = ANY(SELECT Buyer_no
FROM VIEWING
WHERE Comments = 'согласен');

```

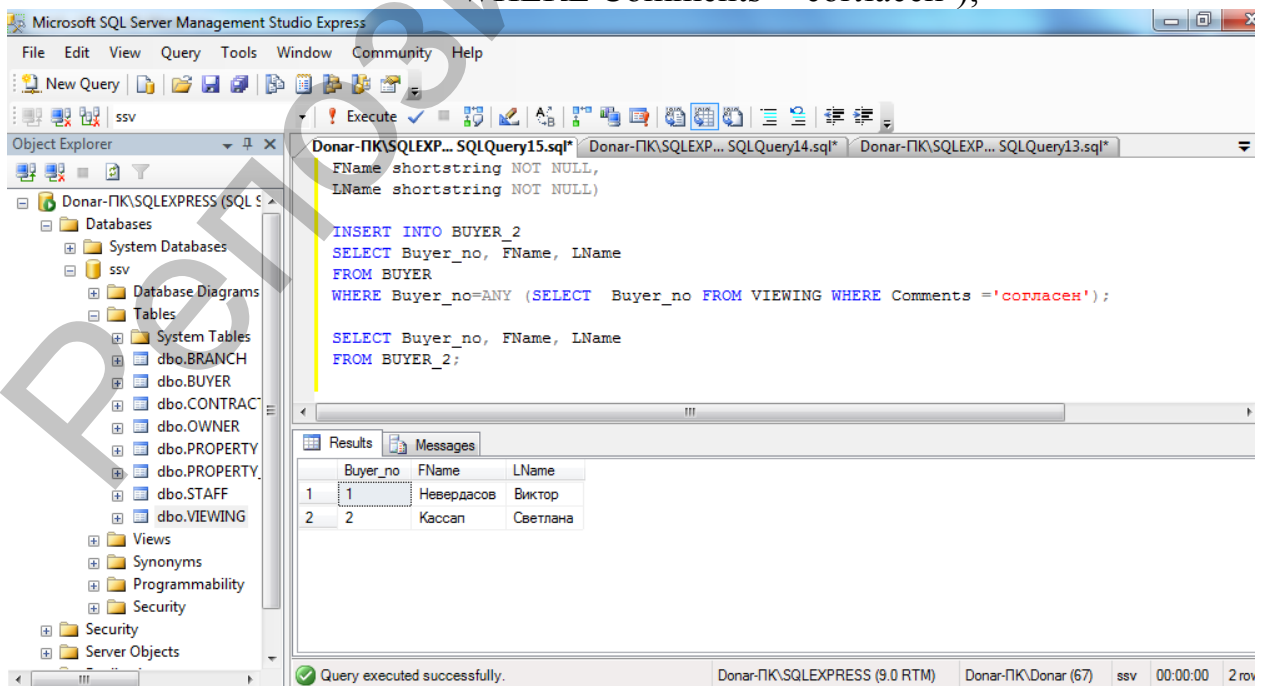


Рис. 40



**Упражнение:** Добавьте данные в таблицу VIEWING:

```
INSERT INTO VIEWING (Date_View, Comments, Property_no, Buyer_no)
VALUES('31.03.03', 'согласен', 3000, 4)
```

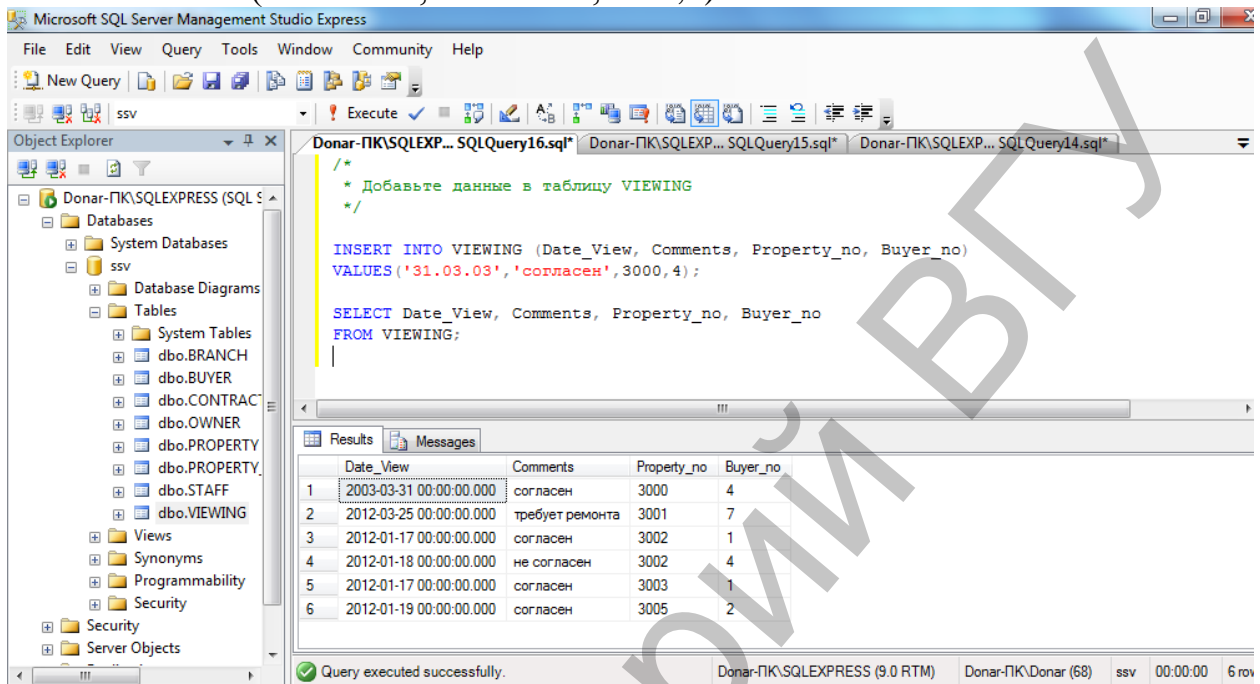


Рис. 41

### Запросы на обновление

Запрос на обновление реализуется с помощью оператора UPDATE. Он служит для изменения значений полей на основе заданного условия отбора.

**Синтаксис:**

```
UPDATE имя_таблицы|имя_проекции
SET имя_поля={выражение|DEFAULT|NULL}[,...n]
[[FROM таблица |соединенная_таблица}[,...n]]
WHERE условие_отбора
```

**Упражнение:** Снизить цены на квартиры, в которых не установлены телефоны на 2 %:

```
UPDATE PROPERTY
SET Selling_Price= Selling_Price*0.98
WHERE Ptel_no='-' ;
```

В команде **UPDATE** могут быть использованы подзапросы. Например, снизить цену в 2 раза на те объекты собственности, у которых поле Comments таблицы VIEWING содержит значение 'требуется ремонта':

```
UPDATE PROPERTY
SET Selling_Price= Selling_Price/2;
WHERE Property_no= (SELECT Property_no
FROM VIEWING
```

WHERE Comments ='требуется ремонт');

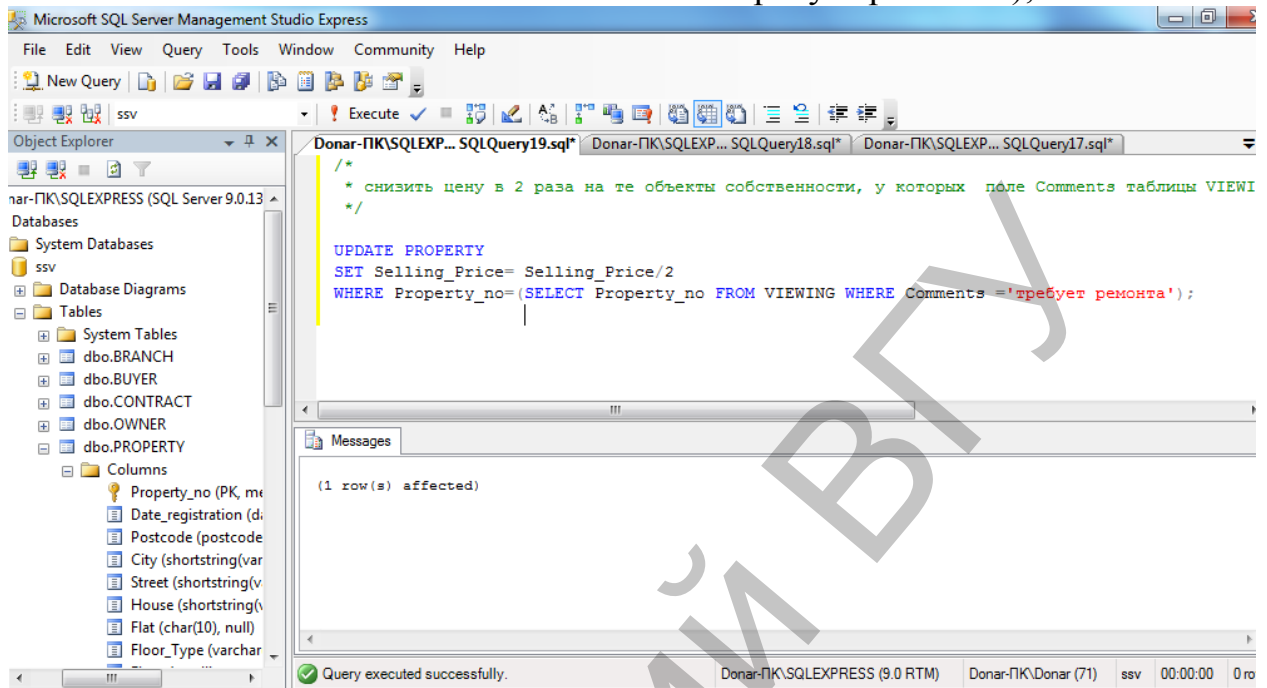


Рис. 42

### КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите основные разделы инструкции SELECT.
2. Для чего предназначены ключевые слова ALL и DISTINCT?
3. Объясните назначение ключевых слов INNER JOIN, LEFT JOIN, RIGHT JOIN.
4. К каким полям применяется оператор LIKE? Какие групповые символы, используется в этом операторе?
5. Перечислите основные агрегатные функции, которые используются в предложении SELECT?
6. Какая команда предназначена для упорядочения вывода информации?
7. В каких случаях необходимы псевдонимы таблиц?
8. В чем заключаются особенности разделов HAVING и WHERE?
9. В каком случае предикат EXISTS возвращает значение 'TRUE'?
10. Приведите примеры вложенных запросов с предикатами ANY, ALL.

### ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

С помощью SQL –команд создайте следующие запросы:

1. Вывести адреса квартир, осмотренных покупателями, у которых в поле Comment занесено значение 'требуется ремонт'.
2. Список трехкомнатных квартир в Витебске площадью не менее 60 метров, расположенных на втором – четвертом этажах, цена которых не превышает 100000\$.

3. Список однокомнатных квартир в Витебске, у которых площадь кухни не менее 10 метров.
4. Перечень квартир, проданных каждым агентом за последний месяц (квартира считается проданной, если в поле Comments таблицы VIEWING занесено значение “согласен”).
5. Среднюю заработную плату сотрудников в каждом из отделений.
6. Среднюю цену трехкомнатных квартир с балконами.
7. Вывести количество квартир, выставленных на продажу.
8. Вывести количество квартир, выставленных на продажу в каждом городе.
9. Определить, сколько однокомнатных, двухкомнатных, трехкомнатных и т.д. квартир выставлено на продажу.
10. Количество однокомнатных квартир, цены которых не превышают средней цены однокомнатной квартиры.
11. Найти самую дешевую однокомнатную квартиру.
12. Вывести количество квартир, проданных каждым агентом..
13. Вывести список агентов, у которых один и тот же объект осматривался более одного раза.
14. Вывести данные сотрудников компании, чья заработная плата выше средней заработной платы сотрудников отделения, в котором он работает.
15. Вывести все варианты объектов недвижимости из таблицы Property, удовлетворяющие требованиям каждого покупателя.
16. Повысить на 10% зарплату агентов, продавших не менее одной квартиры за последний месяц.
17. С помощью команды UPDATE уменьшить на 10% цены однокомнатных квартир, которые не были проданы в течение года с момента регистрации.
18. Таблица PROPERTY\_1 служит для хранения данных об объектах собственности уже выбранных покупателями (находятся в таблице VIEWING и содержат значение “согласен” в поле Comments). С помощью команды INSERT вставить данные об этих квартирах в таблицу PROPERTY\_1.

## ЛАБОРАТОРНАЯ РАБОТА № 7

**Тема:** Создание представлений, триггеров, хранимых процедур.

**Цель работы:** Изучить способы создания и использования представлений, триггеров, хранимых процедур.

### ОСНОВНЫЕ СВЕДЕНИЯ

#### Представление

Механизм представлений является мощным средством СУБД, позволяющим скрыть реальную структуру БД от некоторых пользователей. Реально представление является хранимым в БД запросом, отличаясь от запроса лишь тем, что при изменении данных в таблице они автоматически изменяются и в представлении, что обеспечивает актуальное состояние данных. Представление дает возможность пользователю работать только с теми данными, которые ему нужны, кроме того, механизм представлений позволяет скрыть служебные, конфиденциальные данные.

Создание представления базы данных в системе SQL-сервер может осуществляться следующими способами:

**Первый способ:** С помощью SQL запроса.

Представление создается командой CREATE VIEW, после которой указывается его имя, а далее следует запрос, формирующий тело представления:

#### **Синтаксис:**

```
CREATE VIEW имя_представления  
AS SELECT ...
```

#### **Горизонтальные представления**

Горизонтальное представление позволяет ограничить доступ пользователей определенными строками из одной или нескольких таблиц. Например, создать представление, позволяющее руководителю отделения компании под номером 3 иметь доступ только к данным сотрудников своего отделения:

```
CREATE VIEW STAFF3  
AS SELECT *  
FROM STAFF  
WHERE STAFF.Branch_no= 3;
```

Преимущество представления по сравнению с запросами к БД заключается в том, что оно будет модифицировано автоматически всякий раз, когда таблица, лежащая в его основе изменяется. Например, если в отделение номер 3 будет принят новый сотрудник, то он автоматически отобразится в представлении.

## Вертикальные представления

Вертикальные представления позволяют дать доступ к информации в таблице, исключив некоторые поля. Например, для того, чтобы скрыть данные о зарплате сотрудников надо отобразить в таблицу все поля, исключая поле Salary.

```
CREATE VIEW SALARY_OFF
AS SELECT Staff_no, FName, LName, DOB, City, Street, House, Flat, Stel_no,
Position, Branch_no
FROM STAFF;
```

В рассмотренном примере поля представлений имеют имена, полученные непосредственно из имен полей основной таблицы. Однако иногда возникает необходимость назвать столбцы новыми именами. Это, например, может потребоваться в случае, если столбцы являются вычисляемыми и поэтому не имеющими имен.

Имена, которые необходимо присвоить полям, записываются в круглых скобках после имени таблиц. Они могут не указываться, если совпадают с именами полей запрашиваемой таблицы.

В SQL существует понятие групповых представлений, то есть таких, которые содержат предложение GROUP BY. Представления могут быть основаны сразу на нескольких базовых таблицах.

**Упражнение:** С помощью SQL запроса создать представление, содержащее данные об агентах, отвечающих за продажу объектов. Представление должно включать номер отделения (Branch\_no), номер работника (Staff\_no) и сведения о количестве объектов, за которые он отвечает:

```
CREATE VIEW STAFF_PROP (Branch_no, Staff_no, Properties)
AS SELECT STAFF.Branch_no, STAFF.Staff_no, COUNT(*)
FROM STAFF INNER JOIN PROPERTY ON STAFF.Staff_no=
PROPERTY.Staff_no
GROUP BY STAFF.Branch_no, STAFF.Staff_no;
```

Одной из причин использования представлений является стремление к упрощению многотабличных запросов. После определения представления с соединением нескольких таблиц можно использовать простейшие однотабличные запросы к этому представлению. Однако при создании запросов к представлениям, созданным на основе нескольких таблиц следует учитывать следующие ограничения:

- если столбец в представлении создается с использованием обобщающей функции, то этот столбец может указываться только в предложениях SELECT и ORDER BY тех запросов, которые обеспечивают доступ к данному представлению. Этот столбец не может использоваться в предложении WHERE, а также не может быть аргументом в обобщающей функции;

- сгруппированное представление не должно соединяться с таблицами базы данных или другими представлениями.

Представление может быть обновляемым только в следующих случаях:






- в нем не используется ключевое слово DISTINCT;
- каждый элемент в списке предложения SELECT представляет собой имя столбца, а не выражение или обобщающую функцию;
- представление должно быть построено на базе одной таблицы;
- запрос, определяющий представление не должен содержать предложений GROUP BY и HAVING.


### **Второй способ:** С помощью конструктора.

Для создания представления надо:

1. Выбрать группу *Представления* в списке объектов базы данных, после чего воспользоваться командой *Создать представление*.

После выполнения этих действий загрузится конструктор представлений. Диалоговое окно дизайнера представлений состоит из следующих частей:

- Панель диаграмм – используется для добавления новых таблиц в представление, описание связей между ними, определения полей, которые будут участвовать в представлении. Для открытия/закрытия данной панели используется кнопка “Показать область схемы” ;
- Панель-список – на этой панели отображается перечень полей, выбранных в Панели диаграмм. Можно так же добавить новые поля, определить наличие различных критериев и т.д. Для открытия/закрытия данной панели используется кнопка “Показать область условий” ;
- SQL-панель – данная панель используется для ввода SQL-команд, с помощью которой создается представление. Для открытия/закрытия данной панели используется кнопка “Показать область SQL кода” ;
- Панель результатов – работу произведенных настроек удобно проверить, используя данную панель, по нажатию кнопки Run , отображаются результаты настроенного представления. Для открытия/закрытия данной панели используется кнопка “Показать область результатов” ;

2. Для создания нового представления надо добавить в него необходимые таблицы. Для этого используется кнопка “Добавление таблицы” . При выполнении этого действия на экран будет выведено диалоговое окно с перечнем имеющихся в базе данных таблиц. Используя кнопку “Добавление таблицы” можно добавить выбранные таблицы в представление.

Кроме того представления могут строиться не только на основании таблиц, но и с использованием других представлений. Для этого в диалоговом окне существует закладка *Представления*, которая позволяет добавлять существующие представления базы данных в создаваемое представление.

3. После добавления таблиц, перечень их полей будет отображен в диаграмме представления. Если были ранее установлены связи между полями данной таблицы с использованием первичных и внешних ключей, то будет добавлено соответствующее графическое отображение.
4. На панели диаграмм данного диалогового окна, слева от имени поля таблиц, имеется флажок, при установке которого данное поле будет выведено на экран в результате выполнения представления. При выборе имени этого поля, оно автоматически появляется в списке «Панель-список», и в области оператора SELECT на панели *SQL кода*. Проверить правильность создания представления можно, используя кнопку «*Выполнить код SQL*», в результате чего должны отобразиться данные из созданного представления на панели *результатов*.
5. После сохранения созданного представления его имя появится в списке объектов *Представления* базы данных. Для просмотра информации из этого представления надо выполнить команду *Выбрать первые 1000 строк*, предварительно выбрав предварительно его в списке объектов *Представления*.

Удаление представления из базы данных осуществляется командой `DROP VIEW имя_представления`. При удалении представления пользователь должен являться его владельцем.

**Упражнение:** Создайте представление STAFF\_PROP, содержащее данные о работниках, отвечающих за продажу объектов.

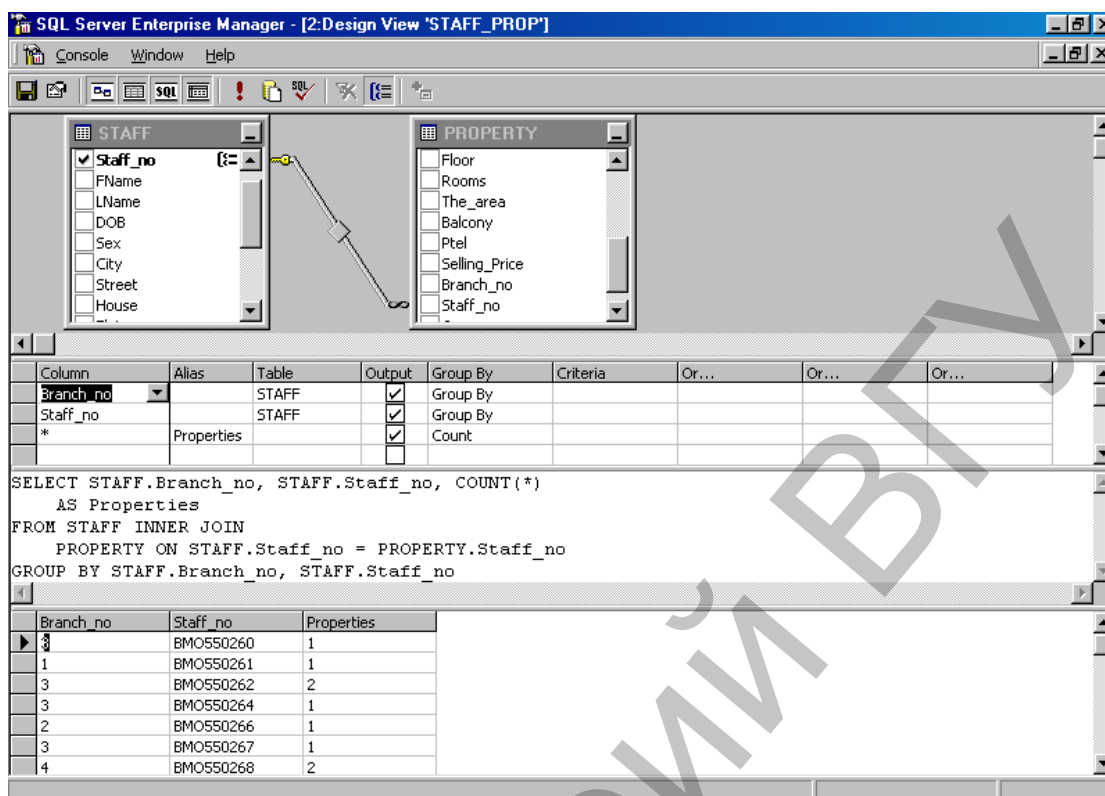


рис. 43

## Хранимые процедуры

Хранимые процедуры – это подпрограммы, выполнение которых происходит непосредственно на сервере баз данных. Все хранимые процедуры в базе данных находятся в специально отведенном списке **Хранимые процедуры** группы **Программирование**.

Для создания новой процедуры с помощью конструктора необходимо:

1. Выбрать команду **Создать хранимую процедуру**. В рабочей области окна сервера появится вкладка **SQLQuery1.sql.**, в котором будет расположена область для ввода текста процедуры.

Вместо текста [PROCEDURE NAME] необходимо ввести имя создаваемой процедуры, после чего набрать текст ее команд.

Чтобы посмотреть информацию о хранимой процедуре необходимо выполнить команду:

```
EXEC SP_HELPTEXT <Имя процедуры>
```

2. Далее необходимо проверить работоспособность созданной процедуры.

### EXEC имя процедуры

Процедура может содержать переменные-параметры, принимаемые процедурой. Каждая переменная внутри хранимой процедуры описывается следующим образом:

**@<имя переменной> <тип данных>**



**Если в процедуру передается несколько параметров, то они указываются после ее имени.**

EXEC <имя процедуры><имя переменной = значение>

Такой способ передачи значений параметра в терминах SQL Server называется передачей по ссылке. При этом значения могут передаваться в произвольном порядке.

Существует другой способ передачи значений параметров в процедуру, называемый передачей значений по позиции. В этом случае параметры указываются через запятую после имени, не нарушая порядка следования параметров в теле процедуры.

**EXEC <имя процедуры><имя переменной1> <имя переменной2>...**

Для создания процедур в MS SQL Server используется язык Transact SQL. Каждая хранимая процедура компилируется при первом выполнении. Описание процедуры совместно с планом ее работы хранится в системных таблицах БД.

Для создания хранимой процедуры используется оператор SQL **CREATE PROCEDURE**, имеющий следующий

**Синтаксис:**

```
CREATE PROCEDURE имя_процедуры (@<имя перем1> <тип данных>,
@<имя перем2> <тип данных> ...)
[ VARYING [=значение по умолчанию] [, параметр N] [OUTPUT]
[ WITH
{ RECOMPILE
| ENCRYPTION
| RECOMPILE, ENCRYPTION } ]
AS тело_процедуры
```

Создается процедура с указанным именем. Процедура может быть создана только в текущей базе данных, за исключением временных процедур, которые создаются в *tempdb*. Для создания временных процедур следует начинать ее имя с '#' или '##'. Длина имени хранимой процедуры вместе с ## не может превышать 20 символов.

Ключевое слово **VARYING** определяет заданное значение по умолчанию для определенного ранее параметра. Ключевое слово **RECOMPILE** определяет режим компиляции. Если **RECOMPILE** задано, то процедура будет перекомпилироваться всякий раз, когда она будет передаваться на выполнение. Ключевое слово **ENCRYPTION** определяет режим, при котором исходный текст хранимой процедуры не сохраняется в БД.

Кроме имени все остальные параметры являются необязательными. Процедуры могут быть процедурами или функциями. Эти понятия трактуются традиционно. В процедуре может быть использовано ключевое слово **OUTPUT**, которое определяет, что данный параметр является выходным.

Удаление процедуры осуществляется с помощью команды DROP PROCEDURE  
DROP PROCEDURE [*owner.*]*procedure\_name* [, [*owner.*]*procedure\_name*...]

В процедурах могут использоваться следующие операторы управления:

### 1. Оператор условия

```
IF <выражение>  
BEGIN  
    <операторы>  
END  
[ELSE]  
[IF <выражение>]  
BEGIN  
    <операторы>  
END
```

Если используется один оператор, то **BEGIN ... END** не используется.

### 2. Циклическое выполнение операций

```
WHILE <логическое выражение>  
    BEGIN  
        <операторы>  
    END
```

В этом операторе можно также использовать ключевое слово **BREAK**, которое позволяет прервать выполнение этого цикла.

### 3. Выбор одного из нескольких значений

```
CASE <переменная>  
WHEN <условие1> THEN <оператор1>  
WHEN <условие2> THEN <оператор2>  
WHEN <условие3> THEN <оператор3>  
...  
ELSE <оператор>  
END
```

Для объявления переменных, которые используются в процедуре надо воспользоваться директивой **DECLARE**. При этом если необходимо присвоить этой переменной какое-либо значение, используется ключевое слово **SELECT**. Оператор **PRINT** позволяет выводить текстовое сообщение на экран.

Пример:

```
DECLARE @X INT;  
SELECT @X=@X+1;  
PRINT 'Результат',@X;
```

**Упражнение:** Создайте процедуру для увеличения заработной платы сотрудников на 10 %.

1. Выберите команду **Создать хранимую процедуру** папки **Хранимые процедуры** в области обозревателя объектов. Появится окно кода хранимой процедуры:

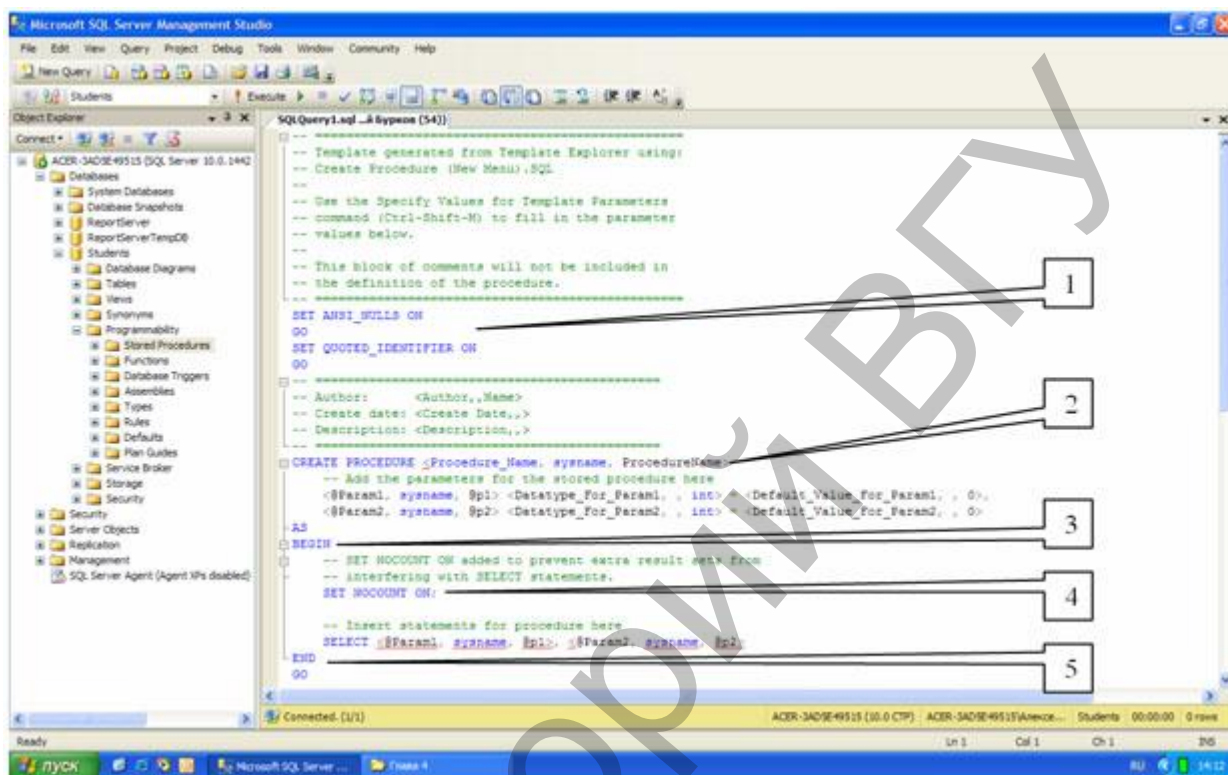


рис.44

Хранимая процедура имеет следующую структуру (см. рисунок 44)

1. Область настройки параметров синтаксиса процедуры. Позволяет настраивать некоторые синтаксические правила, используемые при наборе кода процедуры. В нашем случае это:
  - SET ANSI\_NULLS ON - включает использование значений NULL (Пусто) в кодировке ANSI,
  - SET QUOTED\_IDENTIFIER ON - включает возможность использования двойных кавычек для определения идентификаторов;
2. Область определения имени процедуры (**Procedure\_Name**) и параметров передаваемых в процедуру (**@Param1, @Param2**). Определение параметров имеет следующий синтаксис:  
@<Имя параметра> <Тип данных> = <Значение по умолчанию>  
Параметры разделяются между собой запятыми;
3. Начало тела процедуры, обозначается служебным словом "BEGIN";

4. Тело процедуры, содержит команды языка программирования запросов T-SQL;
5. Конец тела процедуры, обозначается служебным словом "END".

Введите имя процедуры и наберите ее текст, который будет иметь следующий вид:

```
CREATE PROCEDURE NEW (@procent decimal)
AS
UPDATE STAFF
SET Salary =Salary * (100+ @procent)/100
```

Для проверки работоспособности введите команду необходимо создать новый запрос и набрать команду:

```
EXEC NEW @procent = 10
```

Для просмотра результата работы процедуры выполните команду:

```
SELECT * FROM STAFF
```

**Упражнение:** Создайте процедуру для вывода окладов сотрудников заданного как параметр отделения.

```
CREATE PROCEDURE SALARY_OUT(@Branch_no member_no)
AS
SELECT Staff_no, FName, LName, Salary
FROM STAFF
WHERE Branch_no=@Branch_no
```

**Упражнение:** Создать процедуру для повышения заработной платы сотрудника только в том случае, если за ним закреплен хотя бы один объект собственности в таблице Property (номер сотрудника и процент повышения заработной платы передаются в процедуру как параметры).

```
CREATE PROC NEW_SALARY
(@Staff_no int,
@Procent decimal)
AS
IF EXISTS (SELECT property_no
FROM PROPERTY
WHERE Staff_No= @Staff_No )
UPDATE STAFF SET Salary=Salary*(100+ @Procent)/100
WHERE Staff_No= @STAFF_NO
```

Для запуска процедуры:

```
EXEC NEW_SALARY @Staff_No = BMO5502601, @PROCENT=10
```

### Пользовательские функции

Пользовательские функции находятся в папке "**Функции**" расположенной в папке "**Программирование**" обозревателя объектов. Главным отличием

пользовательских функций от хранимых процедур является то, что они возвращают какой то результат. Они вызываются только при помощи оператора SELECT, аналогично встроенным функциям. Все пользовательские функции делятся на 2 вида:

Скалярные функции - функции, которые возвращают число или текст, то есть одно или несколько значений;

Табличные функции - функции, которые выводят результат в виде таблицы.

Для создания новой пользовательской функции используется команда CREATE FUNCTION имеющая следующий синтаксис:

```
CREATE FUNCTION <Имя функции>
([@<Параметр1> <Тип1>[=<Значение1>],
 @<Параметр2> <Тип2>[=<Значение2>], . . .])
RETURNS <Тип>/TABLE
AS
BEGIN
[<function statements>]
    {RETURN < type| RETURN<Команды SQL>}
END
```

Здесь:

Имя функции - имя создаваемой пользовательской функции.

Параметр1, Параметр2, . . - параметры передаваемые в функцию.

Значение1, Значение2, . . . - значения параметров по умолчанию.

Тип1, Тип2, . . . - типы данных параметров.

После служебного слова RETURNS в скалярных функциях указывается тип данных результата, который возвращает скалярная функция, либо служебное слово TABLE в табличных функциях.

После служебного слова RETURN записывается SQL команда самой функции.

После служебного слова RETURN может быть несколько команд, которые располагаются между словами BEGIN и END.

Тип данных параметра должен совпадать с типом данных выражения, в котором он используется.

Если используются несколько SQL команд и BEGIN и END, то перед END следует записать команду RETURN <результат функции>.

**Упражнение:** Создайте скалярную пользовательскую функцию для вычисления средней цены трехкомнатной квартиры в заданном как параметр отделении:

```
CREATE FUNCTION average_price
(@Branch_no Int)
RETURNS Real
AS
RETURN (SELECT avg(selling_price)FROM PROPERTY
        WHERE Branch_no=(@Branch_no)
```

Созданная функция, вычисляющая среднюю цену трехкомнатной квартиры в первом отделении, запускается следующим образом:

```
SELECT Average_3('1').
```

**Упражнение:** Создайте табличную пользовательскую функцию для решения следующей задачи: из таблицы **STAFF** вывести поля **FName**, **LName** и столбец **Length\_of\_work**, который вычисляется как разница дат в годах, между датой приема на работу в компанию (**Date\_Joined**) и текущей датой (параметр **CurDate**).

```
CREATE FUNCTION length_of_work  
(@CurDate Date = GETDATE)  
RETURNS TABLE  
AS  
RETURN (SELECT FName, LName,  
        length_of_work = DATEDIFF (yy, Date_Joined, @CurDate) FROM  
        STAFF)
```

Данная функция вызывается следующим образом:

```
SELECT length_of_work ('31/12/2012')
```

### Триггеры

*Триггер* – это инструмент SQL-сервера, используемый для поддержания целостности данных в базе и выполнения бизнес-правил, слишком сложных для реализации ограничений. Триггеры – это специальный класс хранимых процедур, автоматически запускаемых при добавлении, изменении и удалении данных из таблицы. Каждый триггер привязывается к конкретной таблице. Когда пользователь пытается изменить данные в таблице, сервер автоматически запускает триггер, и только при его успешном завершении разрешается выполнение изменений.

В отличие от хранимых процедур, триггеры нельзя вызывать напрямую, кроме того, в них нельзя передавать параметры. Главное их преимущество в том, что они могут содержать сложную логику обработки. С помощью триггеров осуществляются каскадные изменения данных, что позволяет сократить объем кода для обновления данных в связанных таблицах и обеспечить синхронность изменений во всех таблицах. Например, при удалении данных об объекте **Property\_no** из таблицы **PROPERTY** будут удалены данные о просмотрах этого объекта из таблицы **VIEWING**.

Триггеры могут использоваться для выдачи пользовательских сообщений об ошибках при возникновении определенных условий в процессе выполнения этого триггера. Ограничения, правила и значения по умолчанию позволяют выводить лишь системные сообщения об ошибках.

Триггеры не возвращают наборы результатов. Это связано с тем, что операторы **INSERT**, **UPDATE** и **DELETE** не должны возвращать наборы результатов. Как и хранимые процедуры, триггеры содержат операторы Transact-SQL.

В зависимости от выполняемых пользователем действий, приводящих к запуску триггера, они делятся на три категории:

**триггеры изменения** (запускаются при попытке изменения данных с помощью команды **UPDATE**);

**триггеры вставки** (запускаются при попытке вставки данных с помощью команды **INSERT**);

**триггеры удаления** (запускаются при попытке удаления данных с помощью команды **DELETE**).

Триггеры могут выполняться после выполнения операции (**AFTER**), или вместо выполнения операции (**INSTEAD OF**). Эти параметры определяет то, когда выполняется код триггера – до или после модификации данных. Но в любом случае триггер выполняется прежде, чем изменения будут зафиксированы в базе данных. Важнейшей особенностью триггера **INSTEAD OF** является то, что, он предназначен для выполнения кода, предусмотренного программистом, вместо того кода, выполнение которого обусловлено запросом..

При работе с триггерами доступны две специальные таблицы: таблиц вставок (**INSERTED**) и таблица удалений (**DELETED**) со структурой идентичной структуре таблицы, с которой связан триггер. Таблицы **INSERTED** и **DELETED** заполняются строками модифицируемой таблицы. При выполнении операции **DELETE** строки, удаленные из модифицируемой таблицы помещаются в таблицу **DELETED**. При выполнении операции **INSERTED**, строки, добавленные в модифицируемую таблицу, помещаются в таблицу **INSERTED**. При выполнении операции **UPDATE** для каждой измененной строки ее исходное значение помещается в таблицу **DELETED**, а новое значение – в таблицу **INSERTED**. Данные таблиц **INSERTED** и **DELETED** можно использовать в триггере.

Для создания триггеров используется оператор **CREATE TRIGGER**. В коде оператора указывается таблица, в которой следует создать триггер, а также операторы, включаемые в триггер. Для создания триггера следует выполнить команду **Создать триггер**, выбрав папку **Триггеры** таблицы, для которой создается триггер и ввести код триггера.

**Упражнение:** Создайте триггер для поддержания целостности данных – проверки наличия связанной записи в главной таблице (**BRANCH**) при вводе данных в подчиненную таблицу (**PROPERTY**).

При вводе новых объектов собственности, каждый из объектов должен быть соотнесен с каким-либо отделением компании, то есть при вводе значения атрибута **Branch\_no** в таблицу **PROPERTY** необходимо проверить наличие этого значения в поле **Branch\_no** таблицы **BRANCH**. Создаваемый триггер не позволит добавить новую запись в таблицу **PROPERTY**, если значение в поле **Branch\_no** не совпадает ни с одним значением в поле **Branch\_no** таблицы **BRANCH**.

Для создания триггера с именем INSCHECK с помощью SQL запроса выберите таблицу **PROPERTY** в списке объектов базы данных, после чего выполните команду *Триггеры/Создать триггер*. После этого будет открыто окно триггера, в которое введите следующий код:

```
CREATE TRIGGER INSCHECK ON PROPERTY
FOR INSERT
AS
DECLARE @X Member_no
SELECT @X= Branch_no FROM          INSERTED
IF NOT EXISTS(SELECT * FROM
                BRANCH WHERE Branch_no=@X)
BEGIN
    ROLLBACK TRAN
    RAISERROR('ОШИБКА ЦЕЛОСТНОСТИ! ОТДЕЛЕНИЕ
ОТСУТСТВУЕТ В ТАБЛИЦЕ BRANCH',16,10)
END
```

Триггер активизируется при вставке (ключевое слово **INSERT**) новой записи. После определения переменной **@X** (**DECLARE @X**) ей присваивается значение поля **Branch\_no** добавляемой записи. В процессе использования триггера создается временная таблица **INSERTED**, хранящая в себе добавляемые значения. С помощью оператора **SELECT** переменной **@X** присваивается значение поля **Branch\_no** из таблицы **INSERTED**, то есть значение поля **Branch\_no** вновь добавляемой записи.

Следующий шаг работы триггера – проверка наличия в поле **Branch\_no** таблицы **BRANCH** значения переменной **@X**, то есть проверка допустимости вводимого значения. Если значение не найдено, то выполняется блок операторов, заключенных в области **BEGIN ...END**. С помощью команды **ROLLBACK TRAN**, используемой при работе с транзакциями, отменяется последняя операция. Оператор **RAISERROR** осуществляет выдачу сообщения об ошибке. Значения 16 и 10 определяют уровень критичности операции.

**Упражнение:** Создайте триггер для удаления всех подчиненных записей в таблице **VIEWING** при удалении записи из главной таблицы **PROPERTY**. Если из таблицы **PROPERTY** удаляется какой-либо объект, то предварительно должны быть удалены все записи подчиненной таблицы **VIEWING**, у которых значение поля **Property\_no** соответствует значению поля **Property\_no** удаляемой из таблицы **PROPERTY** записи. В таблице **PROPERTY** создадим триггер **DELCHECK** следующего содержания:

```
CREATE TRIGGER DELCHECK ON PROPERTY
INSTEAD OF DELETE
AS
DECLARE @X INT
```



```

SELECT @X=Property_no FROM DELETED
IF EXISTS (SELECT *
           FROM          VIEWING
           WHERE      Branch_no = @X)
DELETE FROM VIEWING WHERE Property_no=@X
DELETE FROM PROPERTY
WHERE Property_no=@X

```

В первой строке кода создается новый триггер с именем **DELCHECK** для таблицы **PROPERTY**, активизирующийся при удалении записи. Следующим шагом является определение переменной **@X**, которая будет содержать значение поля **Property\_no** удаляемой записи. Затем с помощью оператора **SELECT** данной переменной присваивается значение поля **Property\_no** удаляемой записи, в которой буферизируется удаляемая запись. С помощью оператора **EXISTS** определяется наличие данных в таблице **VIEWING**, для которых в поле **Property\_no** находится значение **@X**. Если такие записи найдены, то система выполняет их удаление. Затем выполняется удаление из главной таблицы (**PROPERTY**). Следует иметь в виду, что связанные записи также должны быть удалены и таблиц, подчиненных таблице **VIEWING**.

**Упражнение:** Создайте триггер для увеличения зарплаты сотрудника на 1% при каждой продаже:

```

CREATE TRIGGER UPDATE_SALARY
ON VIEWING
AFTER INSERT
AS
DECLARE @X Member_no
DECLARE @P Nchar(9)
DECLARE @Y Nchar(18)
SELECT @X=Property_no From INSERTED
SELECT @Y=Comments From INSERTED
SELECT @P= Staff_no From PROPERTY
WHERE Property_no = @X
IF (@Y= 'согласен')
BEGIN
UPDATE STAFF
SET STAFF.Salary = STAFF.Salary*1.1
WHERE Staff_no=@P
END

```

### КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего используются представления?
2. Происходит ли изменение данных в представлении в случае внесения изменений базовые таблицы, на основе которых они созданы?
3. В чем заключается отличие горизонтальных и вертикальных представлений?

4. Можно ли создать представление на основании нескольких таблиц?
5. Перечислите ограничения на обновление данных в представлениях.
6. Для чего предназначены триггеры?
7. В чем заключается механизм работы триггеров?
8. Какие виды триггеров существуют?
9. Какие операторы управления могут использоваться в хранимых процедурах?
10. Как запустить хранимую процедуру на выполнение?

### ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. С помощью **SQL** запроса создайте представление, содержащее данные о количестве квартир, принадлежащих каждому из владельцев собственности. Представление должно включать номер владельца, его данные и количество принадлежащих ему объектов.
2. С помощью конструктора создайте представление, содержащее данные о количестве однокомнатных, двухкомнатных и трехкомнатных квартир в таблице **PROPERTY**.
3. Создайте триггер для удаления из таблиц **PROPERTY** и **VIEWING** объекта собственности, по которому заключается контракт. Выполнить удаление данных о владельце этого объекта. Проверьте работоспособность триггера.
4. Создайте триггер для вывода сообщения о превышении количества объектов собственности, закрепленных за сотрудником, при вводе нового объекта в таблицу **PROPERTY** (количество объектов не должно быть больше трех). Проверьте работоспособность триггера.
5. Создайте триггер для снижения стоимости квартиры на 5%, если в поле **Comments** таблицы **VIEWING** вводится значение "требуется ремонт". Проверьте работоспособность триггера.
6. Создайте хранимую процедуру для индексации цен объектов собственности (увеличьте цену на заданный процент).
7. Создайте хранимую процедуру для вывода списка объектов собственности, зарегистрированных в таблице **PROPERTY** на заданную как параметр дату.
8. Создайте хранимую процедуру для подсчета количества объектов, проданных определенным сотрудником (в процедуру передается номер сотрудника).
9. Создайте процедуру для выбора объектов собственности, удовлетворяющих требованиям покупателя (в процедуру передать следующие параметры: количество комнат, этаж, общую площадь, площадь кухни)
10. Создайте процедуру для повышения на заданный как параметр процент заработной платы тех сотрудников, которые продали максимальное количество объектов собственности в своем отделении. Предусмотрите вывод списка сотрудников, заработная плата которых была повышена.

11. Создать пользовательскую функцию для вычисления стоимости самой дешевой квартиры с заданным, как параметр, количеством комнат в заданном городе.

Репозиторий ВГУ

## ЛАБОРАТОРНАЯ РАБОТА № 8

**Тема:** Администрирование баз данных. Резервное копирование.

**Цель работы:** Ознакомиться с понятием привилегий, способами их создания. Изучить основные способы резервного копирования баз данных.

### ОСНОВНЫЕ СВЕДЕНИЯ

#### Безопасность баз данных и привилегии

При хранении информации в СУБД одной из основных задач является обеспечение безопасности данных. В языке SQL используются следующие основные принципы защиты данных:

В БД действующими лицами являются пользователи. Манипуляции с данными происходят от имени конкретного пользователя;

В SQL используется система привилегий, т.е. прав пользователя на проведение тех или иных действий над определенным объектом базы данных.

Администратор БД создает пользователей и дает им привилегии, пользователи, которые создают таблицы, сами имеют права на управление этими таблицами.

Каждый пользователь в среде SQL имеет специальное идентификационное имя. Команда, посланная в БД, ассоциируется с определенным пользователем. Каждый пользователь в SQL имеет набор привилегий. Эти привилегии могут изменяться со временем – новые добавляться, старые удаляться. При этом пользователь, создавший таблицу любого вида, является ее владельцем. Это означает, что такой пользователь имеет все привилегии в этой таблице и может передавать привилегии другим пользователям для данной таблицы.

Существуют следующие привилегии:

SELECT – пользователь может выполнять запросы к таблице;

INSERT - пользователь может выполнять вставку записей;

UPDATE - пользователь может выполнять корректировку данных;

DELETE - пользователь может выполнять удаления в таблице;

REFERENCES - пользователь может определять внешние ключи;

INDEX - пользователь может создавать индекс в таблице;

SYNONYM - пользователь может создавать синонимы для объекта;

ALTER - пользователь может выполнять команду ALTER TABLE.

В SQL привилегии даются и отменяются двумя операторами – GRANT (допуск) и REVOKE (отмена).

#### Синтаксис:

GRANT *привилегия1, привилегия2...*

ON *таблица*

TO *пользователь1, пользователь2...*;

Для оператора GRANT существует два аргумента, которые имеют специальное значение – это ALL PRIVILEGES (используется вместо имени привилегии, чтобы отдать все привилегии в таблице) и PUBLIC (используется вместо имени пользователя, чтобы отдать соответствующую привилегию всем пользователям).

Для того чтобы пользователи могли передавать свои привилегии другим пользователям, используется оператор WITH GRANT OPTION.

**Синтаксис:**

```
GRANT привилегия1, привилегия2...  
    ON таблица  
    TO пользователь1, пользователь2...  
    WITH GRANT OPTION;
```

С помощью этого оператора пользователь получает особые привилегии для данной таблицы, и может предоставить эту привилегию к той же таблице другому пользователю.

Для удаления привилегии используется оператор

**Синтаксис:**

```
REVOKE привилегия  
    ON таблица  
    FROM пользователь;
```

При этом привилегии отменяются пользователем, который их предоставил.

**Управление привилегиями и правами доступа**

В системе SQL-сервер организована двухуровневая настройка ограничения доступа к данным. На первом уровне в системе необходимо создать так называемое *Имя входа* пользователя, что позволяет ему подключиться к серверу. На втором уровне для каждой базы данных необходимо создать запись пользователя, т.е. с помощью Имен входа осуществляется подключение к SQL-серверу, после чего определяются уровни доступа этого пользователя для каждой базы данных в отдельности.

Для создания учетных записей пользователей в SQL Server необходимо:

Выбрать в группе объектов SQL-сервера *Имена входа* в списке *Безопасность*, после чего выполнить команду *Создать имя входа*. На экран будет выведено диалоговое окно настройки параметров создаваемой учетной записи пользователя.

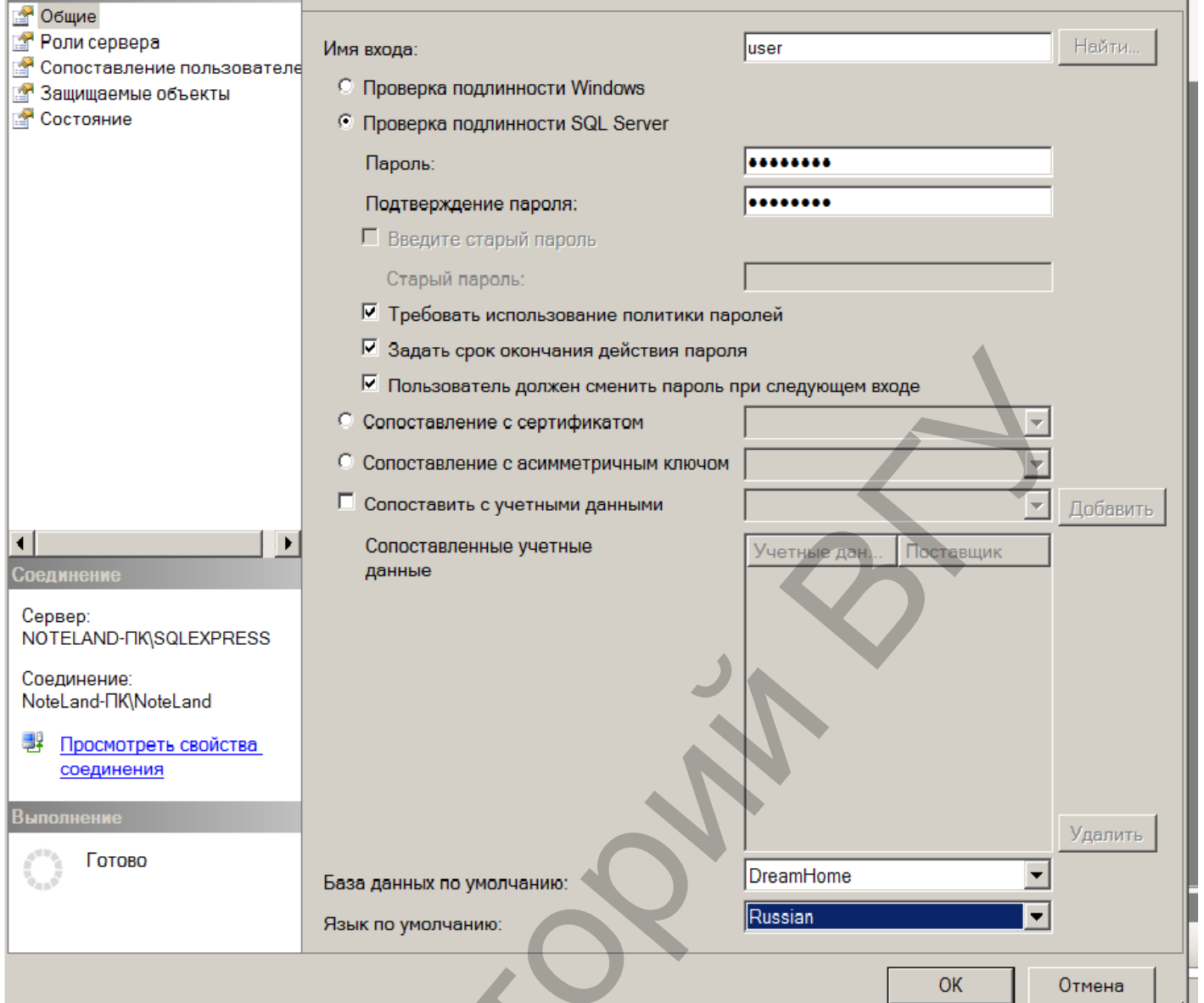


Рис.45

На закладке *Общие* этого окна в поле *Имя входа* вводится имя учетной записи, а поле *Password* – пароль пользователя (при выборе проверки подлинности SQL Server). Здесь также можно установить базу данных, подключение к которой осуществляется по умолчанию при входе пользователя в систему (список <База данных по умолчанию>), а также используемый по умолчанию для данного пользователя язык..

В закладке *Роли сервера* этого диалогового окна представлены опции, с помощью которых назначаются серверные роли для создаваемой учетной записи. Возможны следующие серверные роли:

Sysadmin – любые функции администрирования;

Securityadmin – управление доступом, возможность создания баз данных;

Serveradmin – настройка конфигураций и выполнение функций закрытия SQL-сервера;

Setupadmin – управление связями между серверами;

Processadmin – управление процессами, выполняющимися на сервере;

Diskadmin – управление файлами сервера;

Dbcreator – управление процессами создания и удаления баз данных.

С помощью закладки *Сопоставление пользователя* осуществляется настройка роли для базы данных. В верхнем списке осуществляется выбор

требуемой базы данных (базы, к которой будет разрешен доступ). А в нижнем списке отображается перечень ролей доступа к объектам базы данных. Возможны следующие роли:

public – нет специальных ролей;

db\_owner – полный доступ;

db\_accessadmin – возможность добавления и удаления пользователей базы данных;

db\_securityadmin – управление всеми процессами доступа пользователей к объектам базы данных;

db\_ddladmin – выполнение основных команд, кроме GRANT, REVOKE;

db\_backupoperator – функции запуска процедуры резервного копирования;

db\_datareader – возможность чтения всех данных из любых таблиц базы данных;

db\_datawriter – возможность изменения всех данных из любых таблиц базы данных.

Дальнейшее изменение настроек учетной записи пользователя можно осуществлять с помощью выбора необходимого пользователя в списке *Имя входа* и выбора пункта *Свойства*. Учетные записи доступа к SQL-серверу отображаются в группе объектов *Безопасность/Имена входа* сервера, а права доступа к базам данных в группе *Безопасность/Имена входа* баз данных.

Для создания пользователя в базе данных необходимо:

Выполнить команду *Безопасность/Пользователи/Создать пользователя* нужной базы данных.

На вкладке *Общие* ввести *Имя* пользователя и выбрать ранее созданное *Имя входа* в окне *Выбор имени входа*.

Для настройки прав пользователя в базе данных необходимо:

Выбрать *имя входа* в группе *Безопасность* нужной базы данных.

Выбрав в контекстном меню нужного пользователя *Свойства*, настроить параметры пользователя:

На вкладке *Общие* указать *Схемы, принадлежащие данному пользователю* и *Членство в роли базы данных*

На вкладке *Защищаемые объекты* щелкнуть *Найти*, окне *Добавление объектов* выбрать *Определенные объекты*, в окне *Выбор объектов* указать *Типы объектов* (например, *Таблицы*).

Указать, к каким таблицам определен доступ, и определить, какие манипуляции с выбранными объектами будут разрешены и какие – запрещены пользователю.

### **Резервное копирование**

При разработке систем баз данных особое внимание уделяется вопросам резервного копирования. В системе SQL-сервер имеются инструменты для создания резервных копий и последующего восстановления баз данных. При этом необходимо разработать стратегию создания резервных копий –

периодичность создания, размещение файлов, перечень баз данных, требующих копирования, время запуска процедуры и т.д.

Процесс создания и восстановления резервных копий баз данных осуществляется с помощью среды SQL Server Management Studio. Перед началом процесса резервирования баз данных можно настроить новое хранилище данных – устройство, используемое для хранения резервных копий.

Для создания хранилища данных надо:

Выбрать в списке объектов SQL-сервера *Устройства резервного копирования*, группу *Объекты сервера*.

Выполнить команду *Создать устройство резервного копирования*, после чего на экране отобразится диалоговое окно настройки параметров нового хранилища данных.

В поле *Имя устройства* необходимо указать имя создаваемого устройства. После сохранения параметров устройства его имя отобразится в перечне объектов.

В поле *Файл* необходимо указать путь к файлу, в котором будет храниться резервная копия.



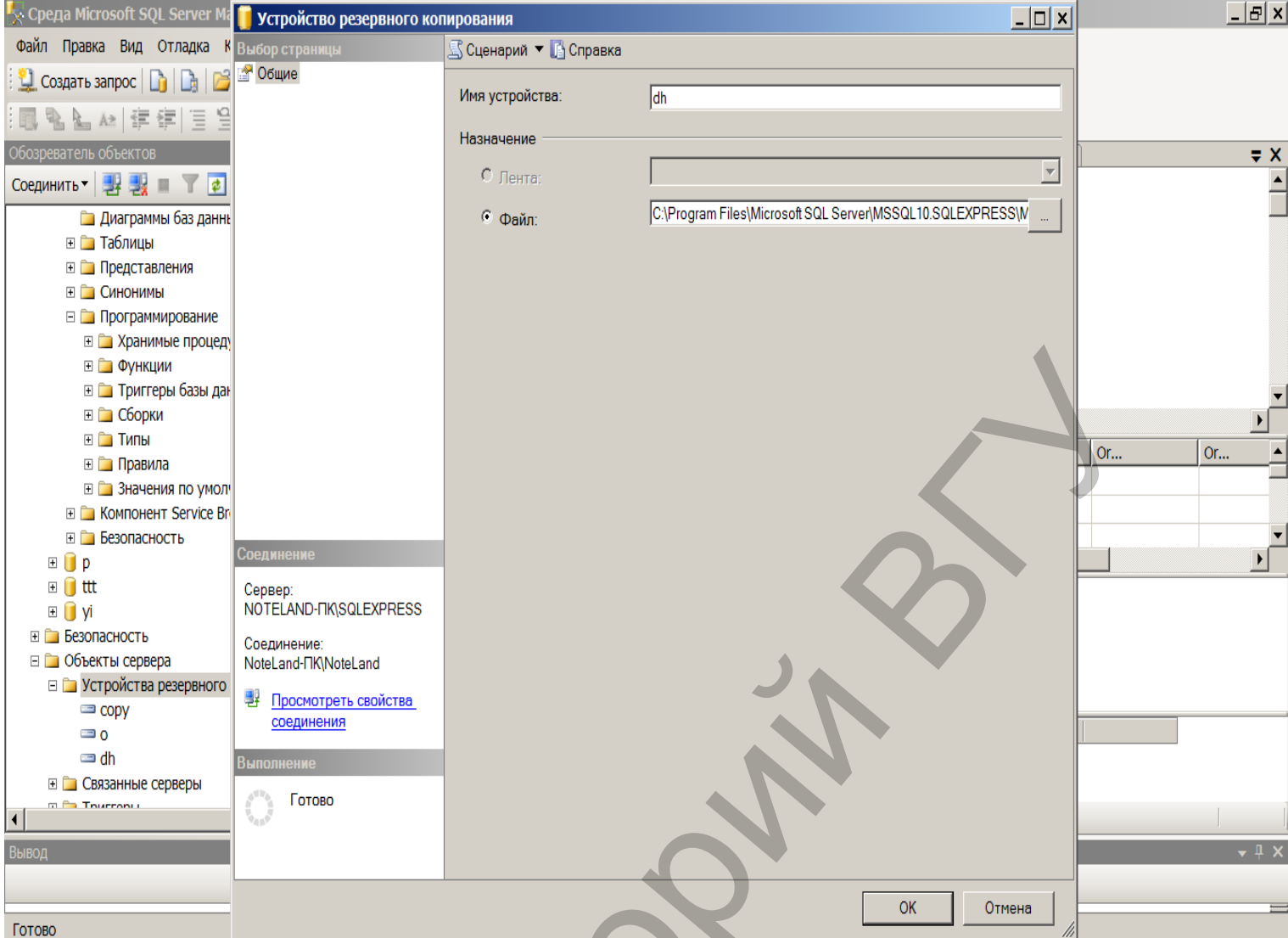


Рис 46

Выбрать в списке объектов SQL-сервера базу данных, резервное копирование которой необходимо осуществить, после чего выполнить команду *Задачи/Создать резервную копию*. При выполнении этого действия на экране отобразится диалоговое окно настройки процесса резервного копирования.

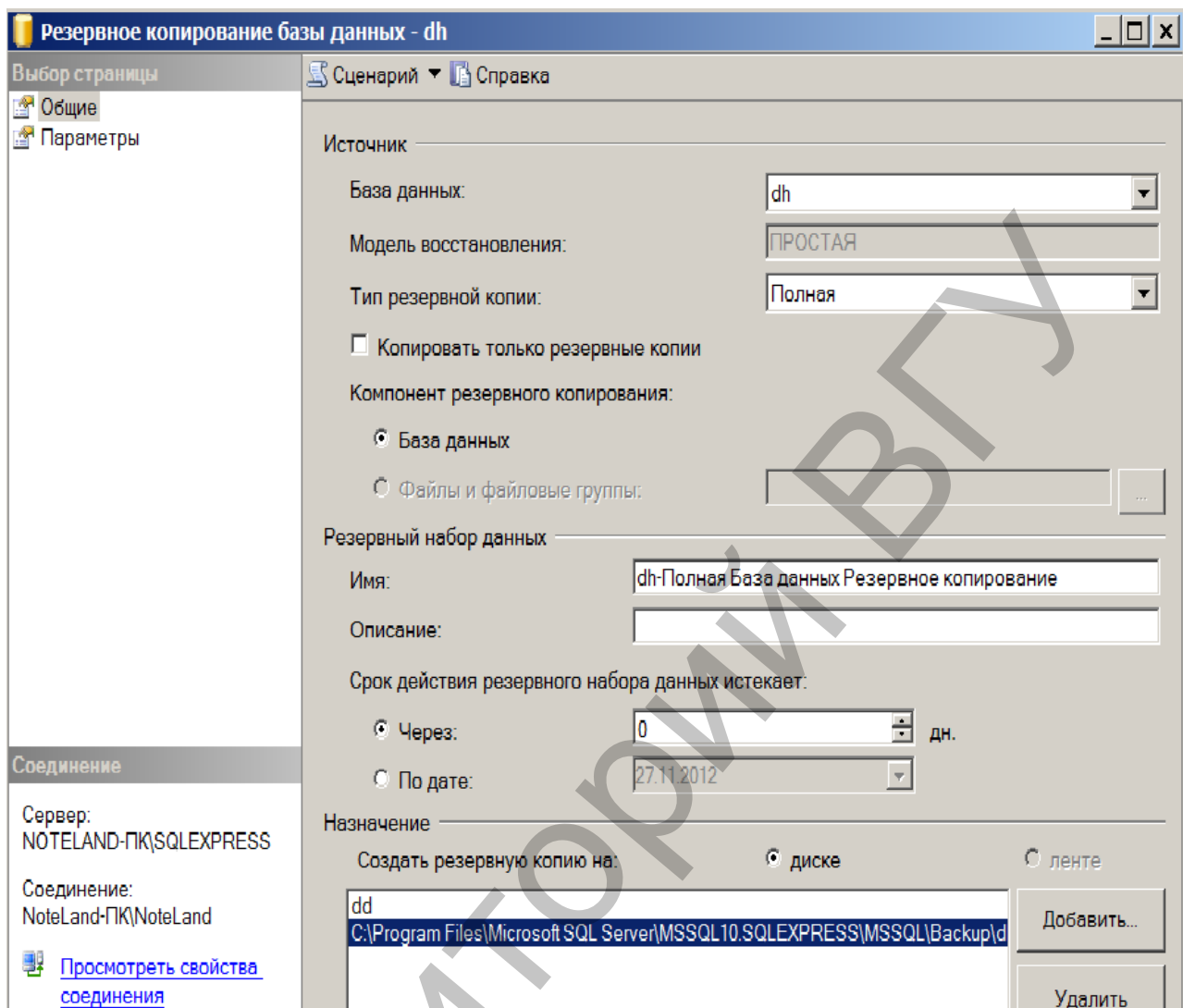


Рис.47

В поле *База данных* отображается имя базы данных, резервное копирование которой надо выполнить. В поле *Описание* можно определить краткое описание создаваемого процесса.

Указать, когда истекает срок действия резервного набора данных.

В поле *Создать резервную копию на* выбрать нужное хранилище либо щелкнуть на кнопке *Добавить*, и в окне *Выбор места расположения резервной копии* выбрать либо *Имя файла* (и задать путь к файлу, в котором будет храниться копия), либо *Устройство резервного копирования*.

Способы восстановления данных из резервной копии:

Выбрать в списке объектов SQL-сервера базу данных, которую надо восстановить или установить курсор на группе .

Выполнить команду *Задачи/Восстановить/База данных*, что приведет к открытию диалогового окна восстановления базы данных.

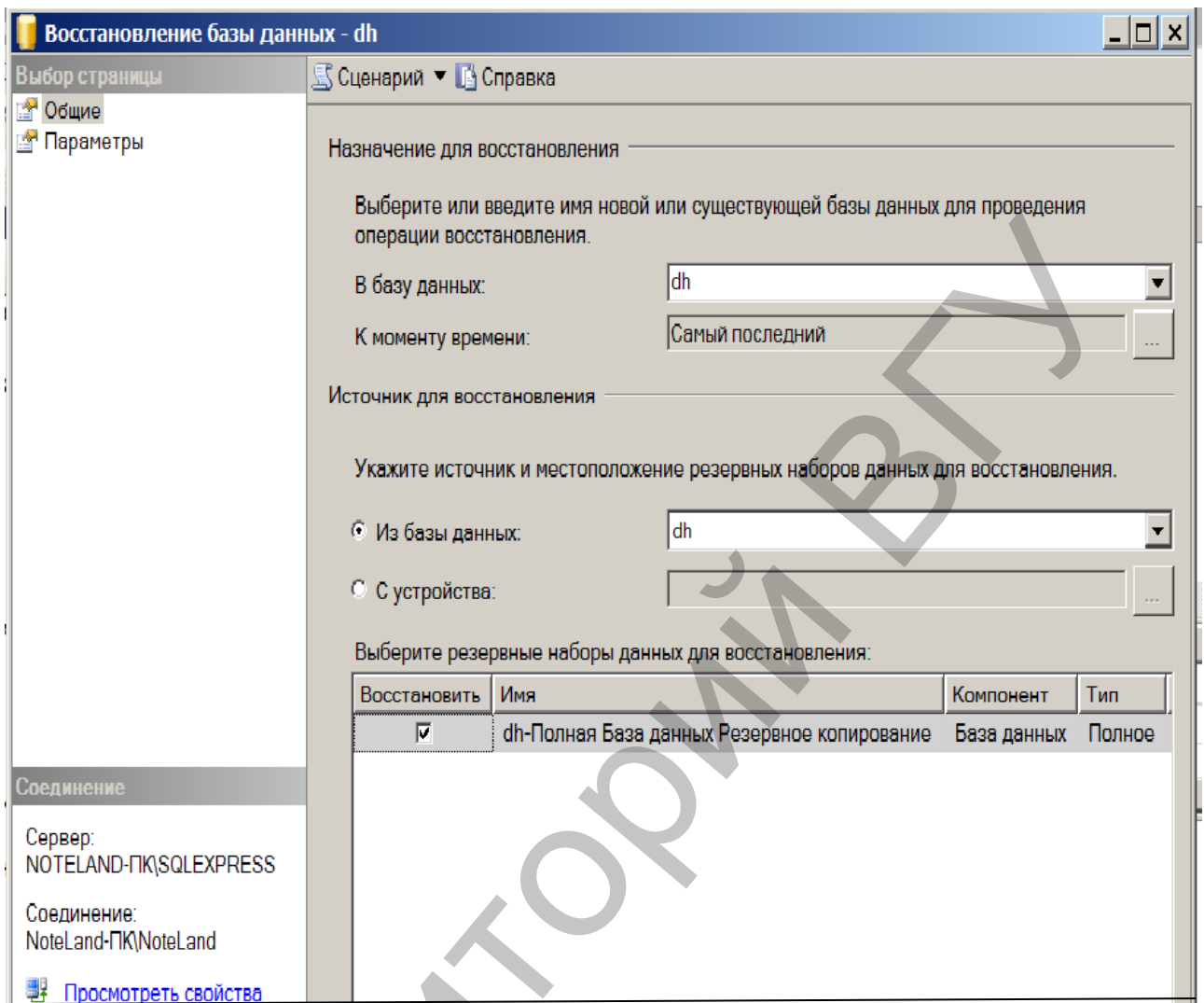


рис. 48

В выпадающем списке надо определить базу данных, требующую восстановления.

Далее следует выбрать местоположение резервной копии и нажать *OK*, что приведет к запуску процесса восстановления базы данных,

### КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие реализуются в основные принципы защиты данных в языке SQL?
2. Какие привилегии могут быть созданы в SQL?
3. С помощью каких команд в SQL даются и отменяются привилегии?
4. Что представляет собой двухуровневая настройка ограничения доступа к данным?
5. Как изменить настройки созданной учетной записи?
6. Для чего нужны системные таблицы?
7. Что такое системный каталог?
8. Что необходимо сделать перед началом процесса резервного копирования баз данных?

### ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Создайте имя входа и определите для него роль **Public**.
2. Обеспечьте для этого имени входа доступ к своей БД с привилегиями добавления данных в таблицу **STAFF**
3. Создайте новое хранилище данных для дальнейшего резервного копирования.
4. Настройте параметры вновь созданного хранилища данных.
5. Проверьте записанную информацию в хранилище данных.
6. Удалите некоторые записи из любой таблицы, а затем восстановите базу данных из созданной резервной копии.

Репозиторий ВГУ

## ЛИТЕРАТУРА

1. Базы данных: модели, разработка, реализация / Т.С. Карпова. СПб., 2001.
2. Программирование баз данных MS SQL Server 2005/ Базовый курс. / Роберт Виейра. -М. Диалектика. 2007.-831 с.
3. Конолли, Томас, Берг, Каролин, Страчан, Анна. Базы даны: проектирование, реализация и сопровождение. Теория и практика, 2-е изд, Пер с англ.: Уч. пос. -М.- 2000
4. Астахова И. Ф. SQL в примерах и задачах : Учеб. пособие для студ. высш. учеб. заведений, обуч. . - Мн. : Новое знание, 2002. - 176с.
5. Хомоненко А. Д. Базы данных : учеб. для высш. учеб. заведений / под ред. А.Д. Хомоненко. - 4-е изд., доп. и перераб. - СПб. : КОРОНА принт, 2004. - 736 с.
6. Марков А. С. Базы данных. Введение в теорию и методологию : учеб. для студ., обуч. по спец. "Прикладная математика и информатика", а также при подготовке бакалавров и магистров по этому напр. - Москва : Финансы и статистика, 2004. - 512 с.
7. Кузин А. В. Базы данных : учеб. пособие для студ. высш. учеб. заведений, обучающихся по напр. подготовки дипломированных спец. 654600 "Информатика и вычислительная техника". - Москва : Академия, 2005. - 315 с.
8. Полякова Л. Н. Основы SQL : курс лекций : учеб. пособие для студ. вузов, обуч. по спец. 351400 "Прикладная информатика". - Москва : Интернет-Университет Информационных Технологий, 2004. - 368 с.

## Приложение 1

### Варианты индивидуальных заданий на проектирование и разработку БД

#### Разработка включает следующие этапы:

- проектирование БД;
- анализ предметной области в соответствии с вариантом индивидуального задания;
- выявление сущностей предметной области (не менее 5) и их атрибутов (минимально необходимый перечень атрибутов приведен и может быть расширен)
- определение ключевых атрибутов сущностей;
- определение связей между сущностями;
- разработку инфологической модели;
- преобразование инфологической модели в реляционную модель;
- определение правил, действующих в предметной области, определение возможных пользователей и решаемых ими задач.
- Создание БД в среде MS SQ Server, создание таблиц с определением ограничений целостности данных, создание диаграммы.
- Ввод данных таблицы. Объем введенных данных должен обеспечивать получение результатов всех запросов, проверку работы триггеров и хранимых процедур.
- Создание представлений для каждого пользователя, триггеров (не менее 3), запросов (не менее 10), хранимых процедур, реализующих задачи пользователей.
- Создание приложения для каждого пользователя. Приложение должно поддерживать решение задач пользователей по работе с информационной системой, обеспечивать ввод информации в таблицы созданной базы данных с помощью соответствующих форм, поиск необходимой информации, поддерживать целостность базы данных, используя соответствующие средства, выполнять запросы.

#### Описание предметной области и примерный состав атрибутов каждого варианта:

##### Вариант 1 БД "Отель"

##### Описание предметной области:

В БД хранятся сведения об отелях, принадлежащих одной компании. Отели находятся в разных городах. Цены на номера одного типа во всех отелях одинаковы. Номер может быть забронирован или свободен. При заезде в отель постояльцы проходят регистрацию. Информация о регистрации постояльцев отеля (выехавших из отеля) хранится в течение года и 1 января удаляется в архив.

БД должна содержать следующий минимальный набор сведения:

- Адрес отеля.
- Название отеля.
- Номер комнаты.
- Тип комнаты.
- Количество мес.
- Цена за сутки проживания.
- Имя и постояльца.
- Фамилия постояльца.
- Отчество постояльца.
- Адрес постоянного проживания.
- Дата заезд.
- Дата отъезд.

### Задания

1. Создайте таблицы, используя необходимые средства поддержки целостности данных для реализации следующих требований:

В поле *Тип комнаты* должно помещаться одно из следующих значений “однокомнатный”, “двухкомнатный” или “семейный”.

значение в поле *Цена* должно находиться в диапазоне от 100 т.р. до 400 т.р.

Значение в поле *Номер комнаты* должно находиться в пределах от 10 до 100.

Значения, помещаемые в поля “Дата прибытия” и “Дата убытия” должны быть по умолчанию равны текущей дате.

2. Создайте запросы:

- Составить список всех 2-комнатных номеров отелей, с ценой менее 200 т.р., упорядочив данные в порядке уменьшения стоимости.
- Выбрать все записи регистрации постояльцев, которые выехали из отелей в течение двух последних недель.
- Найти среднюю стоимость номера в каждом из отелей компании.
- Чему равен общий суточный доход от всех номеров каждого типа?
- Составить список свободных номеров одного из отелей.
- Найти общие потери от наличия в этом отеле свободных номеров за текущий день.
- Определить количество номеров в каждом из отелей.
- Создать таблицу со структурой аналогичной структуре таблицы регистрации для хранения архивных записей. Скопируйте в нее все записи, созданные до 1 января 2011 года. Удалите из основной таблицы регистрации все записи, занесенные в архив.

3. Создайте представления:

- Для турагентов (поиск свободных номеров в отелях).
- Для владельца компании (информация о доходах).

4. Создайте хранимые процедуры:

- для увеличения цены всех номеров на 5 %, если в гостинице нет свободных номеров.
- для получения информации о свободных одноместных номерах гостиницы. Если таких номеров нет, то выдать соответствующее сообщение.
- бронирования двухместного номера в гостинице.

5. Создайте необходимые триггеры.

## **Вариант 2**

### **БД “Сессия”**

Описание предметной области:

БД содержит сведения о сдаче сессии студентами. Номер зачетной книжки однозначно идентифицирует студента. Количество групп на одном курсе не может быть более 5.

БД должна содержать следующий минимальный набор сведений:

- Номер зачетной книжки.
- Фамилия студента.
- Имя студента.
- Отчество студента.
- Курс.
- Группа.
- Код дисциплины.
- Название дисциплины.
- Оценка.
- Фамилия преподавателя.
- Имя преподавателя.
- Отчество преподавателя.
- Кафедра.
- Дата сдачи экзамена.
- Аудитория.

### **Задания**

1. Создайте таблицы, используя необходимые средства поддержки целостности данных для реализации следующих требований:

Значение в поле Курс должно находиться в диапазоне от 1 до 5.

Значение в поле Оценка должно находиться в пределах от 2 до 10.

2. Создайте запросы:

- Составить список дисциплин, которые должны быть сданы каждой группой с указанием дат сдачи и фамилий преподавателей.
- Вывести список студентов, получивших более двух двоек.
- Вывести список студентов, получивших двойки с указанием фамилии преподавателя, которым они должны пересдать экзамен.
- Вычислить средний балл каждого студента.



- Создать рейтинговый список групп по результатам сдачи сессии, упорядочить его по убыванию.
  - Создайте списки студентов, упорядоченные по группам и фамилиям студентов, содержащие данные о средних баллах и размерах стипендии. Формулу для вычисления стипендии (стипендия зависит от среднего балла) задайте самостоятельно.
  - Вывести список студентов, получивших несколько двоек.
  - Вывести список студентов, сдавших все положенные экзамены.
  - Составить список на отчисление (отчисляются студенты, имеющие две и более задолженности).
  - Составьте запрос для назначения повышенной стипендии студентам, сдавшим все экзамены на отлично.
  - Рассчитать количество оценок “3”, “4”, “5”, полученных студентами.
3. Создайте представление для учебного отдела, содержащее данные о результатах сдачи сессии.
4. Создайте хранимые процедуры:
- Для повышения стипендии отличникам на 10%.
  - Для перевода студентов на следующий курс.
  - Для отчисления студента, получившего более одной двойки в сессию.
  - Для изменения оценки при успешной пересдаче экзамена.
5. Создайте триггер для занесения данных о студенте в таблицу “К отчислению” при получении им третьей оценки “2”.

### **Вариант 3**

#### **БД “Библиотека”**

Описание предметной области:

Каждая книга может храниться в нескольких экземплярах.

Для каждого экземпляра известно место его хранения (комната, стеллаж, полка).

Читателю не может быть выдано более 3-х книг одновременно.

Книги выдаются читателям на срок не более 10 дней.

БД должна содержать следующий минимальный набор сведений:

- Автор (фамилия и имя (инициалы) или псевдоним автора издания).
- Название (заглавие) издания.
- Номер тома (части, книги, выпуска).
- Составитель (фамилия и имена (инициалы) каждого из составителей издания).
- Язык, с которого выполнен перевод издания.
- Вид издания (сборник, справочник, монография ...).
- Область знания.
- Переводчик (фамилия и инициалы переводчика).
- Место издания (город).
- Издательство (название издательства).

- Год выпуска издания.
- Библиотечный шифр (например, ББК 32.973).
- Количество книг.
- Номер (инвентарный номер) экземпляра.
- Номер комнаты (помещения для хранения экземпляров).
- Номер стеллажа в комнате.
- Номер полки на стеллаже.
- Цена конкретного экземпляра.
- Дата изъятия экземпляра с установленного места.
- Номер читательского билета (формуляра).
- Фамилия читателя.
- Имя читателя.
- Отчество читателя.
- Адрес читателя.
- Телефон читателя.

### **Задания**

1. Создайте таблицы, используя необходимые средства поддержки целостности данных для реализации следующих требований:

- В библиотеке хранятся книги, выпущенные не позднее 1970 года.
- В библиотеке имеется 10 комнат для хранения книг, в каждой комнате 30 стеллажей, каждый стеллаж состоит из 50 полок.
- Дата изъятия экземпляра по умолчанию равна текущей дате.
- Возраст читателей должен быть не меньше 16 лет.

2. Создайте запросы:

- Вывести список читателей, не вернувших книги в назначенный срок.
- Вывести список читателей, имеющих на руках книги, переведенные с английского языка, изданные позднее 2000 года.
- Вывести список читателей, не вернувших в срок книги и имеющих на руках более трех книг.
- Вывести список книг, которые находятся в библиотеке в единственном экземпляре.
- Вывести книгу, для которой наибольшее количество экземпляров находится "на полках" (не выданы читателям).
- Подсчитать количество читателей, которые не обращались в библиотеку в течение года.
- Исключить из библиотеки читателей, которые не обращались в библиотеку в течение года.
- Вывести список книг по программированию, экземпляры которых отсутствуют в библиотеке.

- Вывести список книг по программированию на C#, экземпляры которых отсутствуют в библиотеке, и которые должны быть возвращены не позднее, чем через 3 дня.

3. Создать представления для администрации библиотеки, содержащие: сведения о должниках.

сведения о наиболее популярных книгах (все экземпляры находятся на руках у читателей).

4. Создать хранимые процедуры:

- Для проверки наличия экземпляров заданной книги в библиотеке (процедура должна возвращать количество экземпляров книги).
- Для ввода в базу данных новой книги.
- Для ввода нового читателя (необходимо проверить наличие читателя в картотеке, чтобы не назначить ему номер вторично).

5. Создать необходимые триггеры.

#### **Вариант 4.**

#### **БД "Учет выполнения заданий"**

Описание предметной области:

Сотрудники организации выполняют проекты. Проекты состоят из нескольких заданий. Каждый сотрудник может участвовать в одном или нескольких проектах, или временно не участвовать ни в каких проектах. Над каждым проектом может работать несколько сотрудников нескольких организаций и отделов, или временно проект может быть приостановлен, тогда над ним не работает ни один сотрудник. Над каждым заданием в проекте работает ровно один сотрудник. Каждый сотрудник числится в одном отделе.

БД должна содержать следующий минимальный набор сведений:

- Номер сотрудника.
- Фамилия сотрудника.
- Имя сотрудника.
- Отчество сотрудника.
- Оклад сотрудника.
- Название организации.
- Номер организации.
- Адрес организации.
- Номер телефона отдела.
- Номер отдела.
- Название отдела.
- Код проекта.
- Название проекта.
- Номер задания.
- Дата начала выполнения задания.
- Срок выполнения задания.

- Отметка о выполнении задания.
- Дата контроля выполнения задания.
- Причина невыполнения задания.

Репозиторий ВГУ

## **Задания**

1. Создайте таблицы, используя необходимые средства поддержки целостности данных для реализации следующих требований:

Оклад сотрудника должен находиться в пределах от 200\$ до 500\$.

Срок выполнения задания не должен превышать 30 дней.

Дата начала выполнения задания и дата контроля выполнения задания по умолчанию равны текущей дате.

Поле причина невыполнения задания может содержать 2 значения, имеющих следующий смысл: “уважительная”, ”неуважительная”.

2. Создайте запросы:

- Составить список всех заданий каждого проекта с указанием организаций, отделов и исполнителей, занятых в его выполнении.
- Составить список проектов, работа над которыми была начата больше месяца назад.
- Вычислить средний оклад сотрудника каждого отдела.
- Подсчитать количество проектов, выполняемых каждым отделом.
- Составить список сотрудников, проектов, заданий, в выполнении которых они участвуют и дат предполагаемого выполнения ими заданий.
- Составить список сотрудников, не выполнивших задание в срок по неуважительной причине.
- Составить список сотрудников, не выполнивших задания в срок с указанием проектов и заданий, которые они должны были выполнить и количества дней просрочки выполнения заданий.
- Составить список сотрудников, участвующих в выполнении более чем одного проекта.
- Составить список проектов, в выполнении которого участвует более трех человек.
- Составить список проектов, срок выполнения которых истекает сегодня, и которые включают больше трех невыполненных заданий.
- Составить список отделов, сотрудники которых не выполнили задания в срок.

3. Создать представление для руководителей проектов, содержащее сведения об исполнителях, отделах, сроках выполнения заданий, включенных в проект.

4. Создать хранимые процедуры:

- Для повышения зарплаты сотрудников, выполнивших задания с трехдневным опережением графика.
- Для печати предупреждения сотруднику, не сдавшему задание в срок по неуважительной причине.
- Для поиска номера телефона сотрудника (телефон установлен в каждом отделе).

5. Создать триггер для запрета удаления данных о сотруднике в случае, если он не завершил выполнение всех своих заданий.

### **Вариант 5**

### **БД "Издательство компьютерной литературы"**

Описание предметной области:

Издательство занимается выпуском литературы по различным областям информатики. Покупатели книг приобретают книги на базе издательства. Когда на базе заканчиваются книги, издается дополнительный тираж.

БД должна содержать следующий минимальный набор сведений:

- Фамилия автора.
- Имя автора.
- Отчество автора.
- Код автора.
- E-mail автора.
- Код ISBN.
- Название книги.
- Код категории книги.
- Категория книги.
- Количество страниц.
- Год начала издания.
- Розничная цена книги.
- Тираж.
- Количество экземпляров на базе издательства.
- Код заказчика.
- Фамилия заказчика.
- Имя заказчика.
- Отчество заказчика.
- Адрес заказчика.
- Телефон заказчика.
- Код заказа.
- Количество экземпляров книги в заказе.

### **Задания**

1. Создайте таблицы, используя необходимые средства поддержки целостности данных для реализации следующих требований:

Количество страниц книги находится в интервале от 50 до 2000.

Год начала издания по умолчанию равен текущему году.

Розничная цена книги находится в диапазоне от 500 до 40000 тысяч рублей.

Тираж не превышает 10000 штук.

2. Создать запросы:

- Список книг, изданных в текущем году и относящихся к категории "Базы данных".

- Список покупателей, заказавших книг на сумму более 100 тыс. рублей.
  - Список книг, которые не заказывались в течение последних двух кварталов.
  - Список авторов, не написавших ни одной книги, относящейся к категории “Базы данных”.
  - Список книг, в названиях которых содержится слово “проектирование” и которые присутствуют на базе в количестве, превышающем 50 экземпляров.
  - Список всех книг, которые дороже любой книги по категории “Базы данных”.
  - Покупателя, сделавшего заказ на максимальную сумму.
  - Список книг, не попавших ни в один из заказов.
3. Создать представление, содержащее сведения о количестве заказанных экземпляров каждой книги, изданной в текущем году.
4. Создать хранимые процедуры:
- Для снижения цен на книги, которые находятся на базе в количестве, превышающем 1000 штук.
  - Для ввода новой книги.
  - Для оформления заказа.
  - Для поиска книг заданного автора.
5. Создать триггеры для увеличения на 1 % стоимость книги, если число проданных экземпляров превышает 5 штук.

#### **Вариант 6**

##### **БД “Пассажир”**

Информационная система служит для продажи железнодорожных билетов. Билеты могут продаваться на текущие сутки или предварительно (не более чем за 45 суток). Цена билета при предварительной продаже снижается на 5 %.

БД должна содержать следующий минимальный набор сведений:

- Номер поезда.
- Название поезда.
- Тип поезда.
- Пункт назначения.
- Расстояние до конечного пункта.
- Пункт назначения для проданного билета.
- Тип вагона.
- Количество мест в вагоне.
- Цена билета.
- Дата отправления.
- Время отправления.
- Номер вагона.

- Номер билета.
- Место.
- Фамилия пассажира.

### **Задания**

1. Создайте таблицы, используя необходимые средства поддержки целостности данных для реализации следующих требований:

Дата отправления по умолчанию равна текущей дате.

Билет может быть продан предварительно не позднее, чем за 45 суток.

Количество вагонов в поезде не может быть меньше 3 и больше 30.

Задайте ограничения на типы вагонов и типы поездов.

2. Создать запросы:

- Свободные места на все поезда, отправляющиеся с вокзала в течение текущих суток.
- Список пассажиров, отправившихся из Витебска в Москву всеми рейсами за прошедшие сутки.
- Количество билетов, проданных до промежуточных пунктов за прошедший день.
- Свободные места в купейные вагоны всех рейсов до Москвы на текущие сутки.
- Выручка от продажи билетов на все поезда за прошедшие сутки.
- Общее количество билетов, проданных по всем направлениям в вагоны типа "СВ".
- Количество непроданных билетов на все поезда, формирующиеся в Витебске, за прошедшие сутки.
- Номера и названия поездов, формирующихся в Витебске все вагоны которых были заполнены менее чем наполовину за прошедшие сутки.
- Список скорых поездов, на которые были проданы билеты за текущие сутки, имеющих вагоны СВ.

3. Создать представление для пассажиров о наличии свободных мест.

4. Создать хранимые процедуры:

- Для повышения цен в пригородные поезда на 20%.
- Для продажи билета.

5. Создать необходимые триггеры.

### **Вариант 7**

#### **БД "Курсы"**

Описание предметной области:

Подразделение занимается организацией внебюджетного образования.

Имеется несколько типов краткосрочных курсов, предназначенных для определенных специальностей, связанных с программным обеспечением ИТ.

Каждый тип курсов имеет определенную длительность и свой перечень изучаемых дисциплин. На каждую специальность может быть набрано несколько групп. По каждой дисциплине могут проводиться лекционные и



лабораторные занятия. Подразделение обеспечивает следующие ресурсы: учебные классы, лекционные аудитории и преподавателей. Необходимо составить расписание занятий.

БД должна содержать следующий минимальный набор сведений:

- Фамилия слушателя.
- Имя слушателя.
- Специальность.
- Номер группы.
- Количество человек в группе.
- Название дисциплины.
- Количество часов.
- День недели.
- Номер пары.
- Номер аудитории.
- Вид занятий (лекционные или практические).
- Фамилия преподавателя.

### **Задания**

1. Создайте таблицы, используя необходимые средства поддержки целостности данных для реализации следующих требований:

- Количество пар в день не может быть больше 3.
- Количество часов, отводимых на изучение дисциплины, находится в диапазоне от 10 до 100.

2. Создать запросы:

- Вывести все номера группы и специальности, где количество слушателей меньше 10.
- Вывести перечень изучаемых дисциплин по тем специальностям, где количество слушателей меньше 10.
- Вывести список преподавателей, которые не проводят занятия на третьей паре ни в один из дней недели.
- вывести список свободных лекционных аудиторий на понедельник.
- Вычислить общее количество учебных часов по каждой специальности.

3. Создать представление для потенциальных слушателей, содержащее перечень специальностей, изучаемых на них дисциплин и количество часов.

4. Создать хранимые процедуры:

- Получить расписание занятий для группы на определенный день недели.
- Осуществить процедуру записи на курс слушателя, если в группе по данному курсу уже набрано больше 15 человек, то выдать сообщение об отказе в приеме.
- Выполнить процедуру увольнения преподавателя, при этом в расписании заменить его другим.

- Получить перечень свободных лекционных аудиторий (с указанием времени) на любой день недели. Если свободных аудиторий не имеется, то выдать соответствующее сообщение.

5. Создать триггер для вставки данных о новой специальности.

### **Вариант 8**

#### **БД "Приказы"**

Описание предметной области:

База данных содержит сведения о сотрудниках организации. В соответствии с приказами сотрудники могут быть приняты на работу, переведены на другую должность, отправлены в отпуск, уволены, и т.д. Нумерация приказов ежегодно начинается заново с 1 января.

БД должна содержать следующий минимальный набор сведений:

- Номер личного дела.
- Фамилия.
- Имя.
- Отчество.
- Дата рождения.
- Адрес проживания.
- Структурное подразделение
- Должность.
- Специальность.
- Квалификация.
- Образование.
- Номер кабинета.
- Номер телефона.
- Дата начала трудовой деятельности.
- Размер оклада.
- Вид приказа.
- Номер приказ.
- Дата приказа.
- Дата приема.
- Дата назначения.
- Вид перевода.
- Вид отпуска.
- Срок отпуска.
- Дата аттестации.
- Форма повышения квалификации (с отрывом или без отрыва от работы).
- Дата начала обучения.
- Дата окончания обучения.

1. Создайте таблицы, используя необходимые средства поддержки целостности данных. Ограничения задайте самостоятельно.
2. Создать запросы:
  - Вывести список поощрений одного из сотрудников.
  - Сколько сотрудников не имеют высшего образования.
  - Вывести список уволенных сотрудников.
  - Сколько сотрудников повысили квалификацию в январе?
  - Найти среднюю продолжительность отпуска сотрудников.
  - Вывести список сотрудников, не повышавших квалификацию.
  - Вывести количество сотрудников в каждом структурном подразделении, имеющих стаж работы более 5 лет.
  - Вывести список сотрудников уволенных из каждого структурного подразделения.
3. Создать представление для руководителя организации, содержащее сведения о руководителях структурных подразделений и их окладах.
4. Создать хранимые процедуры:
  - Для ввода данных приказа.
  - Для изменения номера телефона в одном из кабинетов.
  - Найти данные сотрудника по заданным параметрам.
  - Повысить оклады сотрудников на заданный процент.
5. Создать триггер: при увольнении сотрудника (при заполнении приказа об увольнении) удалить сведения о сотруднике из базы.

#### **Вариант 9.**

##### **БД Аэропорт**

Описание предметной области:

Необходимо обеспечить продажу билетов на нужный рейс, при отсутствии билетов (необходимого количества билетов) предложить билет на ближайший рейс.

БД должна содержать следующий минимальный набор сведений:

- Бортовой номер самолета.
- Тип самолета.
- Количество мест.
- Грузоподъемность.
- Скорость.
- Дата выпуска.
- Налётано часов.
- Дата последнего ремонта.
- Назначение самолета.
- Расход топлива.
- Код экипажа.
- Паспортные данные членов экипажа.

- Номер рейса,
- Дата вылета,
- Время вылета.
- Аэропорт вылета,
- Аэропорт назначения.
- Расстояние.
- ФИО пассажира.
- Паспортные данные.
- Номер места.
- Тип места.
- Цена билета.

### **Задания**

1. Создайте таблицы, используя необходимые средства поддержки целостности данных для реализации следующих требований:

а) Самолеты, выпущенные ранее 10 лет назад должны проходить профилактический ремонт ежегодно.

б) Билеты продаются не более чем за 30 дней до вылета и не менее чем за 1 час до вылета.

2. Создать запросы:

- Определить расчетное время полета по всем маршрутам.
- Определить расход топлива по всем маршрутам.
- Вывести экипаж, совершивший максимальное количество полетов за прошедшую неделю.
- Вывести данные о том, сколько свободных мест оставалось в самолетах, совершавших полет по одному из рейсов за вчерашний день.
- Рассчитать убытки компании за счет непроданных билетов за вчерашний день.
- Вывести список самолетов, которые не ремонтировались в течение более чем 3 лет.
- Определить каким количеством самолетов каждого типа владеет компания.
- Определить средний “возраст” самолетов компании.

3. Создать представление для пассажиров авиакомпании.

4. Создать хранимые процедуры:

Для поиска и продажи билетов на нужный рейс.

5. Создать необходимые триггеры.

## Вариант 10

### БД ” Оптовая база ”

Описание предметной области:

Оптовая база закупает товары у компаний-поставщиков и поставляет их компаниям – покупателю. Доход оптовой базы составляет 5 % от стоимости товара проданного компании – покупателю. Один и тот же товар может доставляться несколькими поставщиками и один и тот же поставщик может доставлять несколько видов товаров. Цены товара у разных поставщиков могут отличаться.

БД должна содержать следующий минимальный набор сведений:

- Код сотрудника.
- Паспортные данные сотрудника.
- Код товара.
- Название товара.
- Единица измерения товара.
- Количество товара.
- Минимальный запас товара.
- Стоимость единицы товара.
- Примечание – описание товара.
- Код поставщика.
- Название компании поставщика.
- Адрес поставщик.
- Дата поставки.
- Количество товара в партии.
- Номер счета.
- Код организации – покупателя.
- Название компании покупателя.
- Адрес покупателя.
- Дата вывоза.
- Количество товара в партии.
- Продажная цена товара.

### Задания

1. Создайте таблицы, используя необходимые средства поддержки целостности данных для реализации следующих требований:

В поле Тип комнаты должно помещаться одно из следующих значений “однокомнатный”, “двухкомнатный” или “семейный”.

значение в поле Цена должно находиться в диапазоне от 100 т.р. до 400 т.р.

Значение в поле Номер комнаты должно находиться в пределах от 10 до 100.

Значения, помещаемые в поля “Дата прибытия” и “дата убытия” должны быть по умолчанию равны текущей дате.

2. Создайте запросы:

- Вывести список поставщиков, которые поставляют все товары.

- Определить поставщика, который поставляет один из товаров по самой низкой цене.
  - Вывести названия товаров, цены на которые никогда не повышались.
  - Чему равен общий суточный доход оптового склада за прошедший день?
  - Вычислить стоимость каждого вида товара, находящегося на базе.
  - В какой день было вывезено минимальное количество товара?
  - Сколько различных видов товара имеется на базе?
  - Создать таблицу со структурой аналогичной структуре таблицы регистрации для хранения архивных записей. Скопируйте в нее все записи, созданные до 1 января 2011 года. Удалите из основной таблицы регистрации все записи, занесенные в архив.
3. Создайте представления:
- Для компаний - покупателей (поиск нужного товара).
4. Создайте хранимые процедуры:
- для снижения цены на заданный процент для товаров, у которых срок пребывания на складе превысил заданный норматив.
5. Создайте триггер удаления, запуск которого происходит при отпуске товара покупателю. Предусмотреть вывод сообщения при недостаточном количестве товара на базе.

### **Вариант 11**

БД "Автовокзал"

Описание предметной области:

По одному и тому же маршруту отправляется несколько рейсов ежедневно. Номер рейса определяется маршрутом и временем отправления. Билеты могут продаваться предварительно, но не ранее чем за 10 суток. Места в билете не указываются. На каждый рейс может продаваться не более 10 билетов без места, цена на которые снижается на 10%.

БД должна содержать следующий минимальный набор сведений:

- Номер рейса.
- Номер водителя.
- Номер автобуса.
- Паспортные данные водителя.
- Пункт отправления.
- Пункт назначения.
- Дата отправления.
- Время отправления.
- Время в пути.
- Номер автобуса.
- Тип автобуса.
- Количество мест в автобусе.
- Номер билета.

- Цена билета.

### **Задания**

1. Создайте таблицы, используя необходимые средства поддержки целостности данных. (Ограничения задать самостоятельно)
2. Создать запросы:
  - Вывести количество автобусов каждого типа, отправляющихся с автовокзала.
  - Вывести фамилии водителей и номера автобусов, отправившиеся в рейсы до 12 часов текущего дня.
  - Рассчитать выручку от продажи билетов за прошедший день.
  - Вывести список водителей, которые не выполнили ни одного рейса за прошедший день.
  - Вывести сумму убытков из-за непроданных мест в автобусе за прошедшую неделю.
  - Сколько рейсов выполнил каждый водитель.
3. Создать представление для пассажиров (количество свободных мест на все рейсы).
4. Создать хранимые процедуры:
  - Продажи билета.
  - Возврата билета.
  - Добавления нового рейса.
5. Создать триггер для занесения стоимости каждого проданного билета во временную таблицу «выручка за текущий день».

### **Вариант 12**

#### **БД Автомастерская**

Описание предметной области:

Автомастерская осуществляет ремонт автомашин, используя для этих целей штат мастеров и свои мастерские. Стоимость ремонта включает цену деталей и стоимость работы. Заработная плата мастеров составляет 50 % стоимости работы.

БД должна содержать следующий минимальный набор сведений:

- Табельный номер мастера.
- ФИО мастера.
- Разряд мастера.
- Адрес.
- Дата заказа.
- Гос.номер автомобиля.
- Марка.
- Мощность автомобиля.
- Год выпуска.
- Цвет автомобиля.
- Дата принятия в ремонт.

- Плановая дата окончания ремонта.
- Фактическая дата окончания ремонта.
- Вид ремонта.
- Стоимость ремонта.
- Название детали.
- Цена детали.
- Марка автомобиля.
- ФИО владельца.
- Номер телефона владельца.

### **Задания**

1. Создайте таблицы, используя необходимые средства поддержки целостности данных. (Ограничения задать самостоятельно)
2. Создать запросы:
  - Выбрать фамилию того механика, который чаще всех работает с автомобилями марки "Тойота".
  - Определить тех владельцев автомобилей, которых всегда обслуживает один и тот же механик. Вывести фамилии механика и его постоянного клиента.
  - Вывести фамилии механиков, которые не выполняли работы в срок и количество дней просрочки выполнения заказа.
  - Вывести данные владельца самого старого автомобиля.
  - Сколько автомобилей отремонтировал каждый механик.
  - Сколько заработал каждый водитель за прошедший месяц?
  - За каждый день просрочки выполнения заказа механику назначается штраф в размере 5%. Рассчитать штраф каждого механика за прошедший месяц.
3. Создать представление для заказчиков (фамилию механика и модель автомобиля, которую он ремонтирует чаще всего).
4. Создать хранимые процедуры:
  - Повышения цены деталей для автомобиля "Ford" на 10 %.
  - Создайте процедуру для повышения разряда тех мастеров, которые отремонтировали максимальное количество автомобилей.
5. Создать триггер для занесения стоимости каждого выполненного заказа во временную таблицу «выручка водителя за текущий день».

### **Вариант 13.**

#### **БД Прокат автомобилей**

Описание предметной области:

Компания предоставляет прокат автомобилей. В пункт проката обращаются клиенты, данные которых регистрируют в базе. Цена проката зависит от марки автомобиля, технических характеристик и года выпуска. За каждый час просрочки возврата автомобиля начисляется штраф. При каждом обращении фиксируется дата выдачи автомобиля и дата возврата, номер.



Если клиент не вернул автомобиль в срок и не оформил продление, ему назначается штраф и автомобиль больше не выдается. Постоянным клиентам предоставляются скидки.

БД должна содержать следующий минимальный набор сведений:

- ФИО.
- Паспортные данные.
- Код должности.
- Наименование должности.
- Оклад.
- Обязанности.
- Код марки.
- Наименование,
- Технические характеристики,
- Описание.
- Код автомобиля.
- Регистрационный номер.
- Номер кузова.
- Номер двигателя.
- Год выпуска.
- Пробег,
- Цена автомобиля.
- Цена проката.
- Дата последнего ТО.
- Код сотрудника-механика.
- Специальные отметки.
- Отметка о возврате.
- Код клиента.
- ФИО.
- Адрес.
- Телефон.
- Паспортные данные.
- Дата и время выдачи автомобиля.
- На сколько часов.
- Дата и время возврата автомобиля.

### **Задания**

1. Создайте таблицы, используя необходимые средства поддержки целостности данных. (Ограничения задать самостоятельно)

2. Создать запросы:

- Какой автомобиль находился в прокате максимальное количество часов?
- Какой автомобиль ни разу не был в прокате?

- Определить убытки от простоя автомобилей за вчерашний день.
- Вывести данные автомобиля, имеющего максимальный пробег.
- Вывести данные клиента обратившегося в прокат больше двух раз.
- Вывести данные клиентов, не вернувших автомобиль вовремя.
- Определить каким количеством автомобилей каждого типа владеет компания.

Определить средний “возраст” автомобилей компании.

3. Создать представление для клиентов компании.

4. Создать хранимые процедуры:

- Выполнить списание автомобилей, выпущенных ранее заданного года.
- Выдачи автомобиля и расчета стоимости с учетом скидки постоянным клиентам.

5. Создать триггер, который помещает в поле *специальные отметки* признак, который характеризует постоянного клиента (если клиент обращается в прокат третий раз).

#### **Вариант 14**

##### **БД ”Ресторан”**

Описание предметной области:

Сотрудники ресторана – повара и официанты. За каждым официантом закреплены определенные столы. Каждый повар готовит определенный набор блюд. Запас продуктов на складе не должен быть ниже заданного значения. Цена заказа складывается из стоимости ингредиентов и наценки, которая составляет 40 % стоимости ингредиентов.

БД должна содержать следующий минимальный набор сведений:

- ФИО сотрудника.
- Паспортные данные сотрудника.
- Категория сотрудника.
- Должность сотрудника.
- Оклад сотрудника.
- Наименование ингредиента
- Код ингредиента.
- Дата закупки.
- Объем закупки.
- Количество продукта на складе.
- Необходимый запас продукта.
- Срок годности.
- Цена ингредиента.
- Поставщик.
- Наименование блюда.
- Код блюда.
- Объем ингредиента.

- Номер стола.
- Дата заказа.
- Код заказа.
- Количество.
- Название блюда.
- Ингредиенты, входящие в блюдо

### **Задания**

1. Создайте таблицы, используя необходимые средства поддержки целостности данных. (Ограничения задать самостоятельно)
2. Создать запросы:
  - Вывести данные официанта, принявшего максимальное число заказов.
  - Вывести данные официанта, принявшего заказы на максимальную сумму.
  - Рассчитать премию каждого официанта за последние 10 дней (5% от стоимости каждого заказа).
  - Подсчитать, сколько ингредиентов содержит каждое блюдо.
  - Вывести название блюда, содержащее максимальное число ингредиентов.
  - Какой повар может приготовить максимальное число блюд?
3. Создать представление для посетителей ресторана, содержащее сведения обо всех блюдах и их ценах.
4. Создать хранимые процедуры:
  - Вывести сведения о заказах заданного официанта на заданную дату.
  - Формирования заказа. При этом необходимо выполнить расчет стоимости заказа по заданному названию блюда и количеству единиц в заказе, изменить значение количества продукта на складе, и если количество продукта на складе достигло критического значения – вывести сообщение и занести название продукта и дату во временную таблицу.
  - Повышения оклада заданного сотрудника на 30 % при повышении его категории.
5. Создать триггер для удаления данных из связанных таблиц при удалении блюда из БД.

### **Вариант 15**

#### **БД "Справочная аптек"**

Описание предметной области:

Цена лекарств, срок годности которых истекает через месяц, снижается на 50 %. Если запас лекарств снижается до минимума, его данные помещают во временную таблицу **Заказ**.

БД должна содержать следующий минимальный набор сведений:

- Код лекарства.
- Название лекарства,

- Показания к использованию.
- Противопоказания.
- Производитель.
- Наличие лекарства.
- Минимальный запас.
- Тип.
- Дозировка.
- Цена,
- Количество.
- Дата продажи.
- Сколько продано.
- Дата выпуска.
- Срок годности.
- Номер аптеки.
- Специализация аптеки.
- Район.
- Телефон.
- Тип: таблетки, микстура, мазь и т.д.

### **Задания**

#### 2. Создать запросы:

- Определить, в каких аптеках дешевле всего анальгин.
- По ассортименту предлагаемых лекарств определить, какой болезнью чаще всего страдают покупатели аптеки №1.
- Вывести список лекарств, у которых срок годности истекает через три дня.
- Определить, какие убытки понесет аптека, если в течение месяца не реализует все лекарства, у которых истекает срок годности.
- Выбрать список лекарств, которые подходят для больного, страдающего болезнями 'язва желудка' и 'корь' одновременно.

#### 3. Создать представление, содержащее сведения обо всех лекарствах и их ценах.

#### 4. Создать хранимые процедуры:

- Для вывода данных о наличии и ценах заданного лекарства в аптеках в порядке возрастания цен.
- Для ввода данных о новом лекарстве в БД.
- Вывести сведения об объеме продажи заданного как параметр лекарства за прошедший месяц.
- Для вычисления суммарного дохода аптеки (параметр) за последний месяц.

#### 5. Создать триггер для фиксации продажи лекарства (уменьшение количества, проверка запаса, помещение данных в таблицу **Заказ**)

## Вариант 16

### БД Таксопарк

Описание предметной области:

Система должна фиксировать все вызовы такси. Каждому водителю ежедневно начисляется заработная плата в зависимости от количества вызовов и их тарифа (50% от заработанной им суммы).

- ФИО сотрудника.
- Адрес сотрудника.
- № телефона сотрудника.
- Паспортные данные сотрудника.
- Должность сотрудника.
- Категория сотрудника.
- Наименование марки автомобиля.
- Технические характеристики.
- Стоимость.
- Код тарифа.
- Наименование тарифа.
- Стоимость тарифа.
- Код автомобиля.
- Код марки.
- Регистрационный номер.
- Номер кузова.
- Номер двигателя.
- Год выпуска.
- Пробег.
- Дата последнего ТО.
- Дата вызова.
- Время посадки пассажира.
- Время высадки пассажира.
- Номер телефона пассажира.
- Откуда.
- Куда.

### Задания

1. Создайте таблицы, используя необходимые средства поддержки целостности данных. (Ограничения задать самостоятельно)
2. Создать запросы:
  - Вывести данные о водителе, который чаще всего доставляет пассажиров на улицу Чкалова.
  - Вывести данные об автомобилях, которые имеют пробег более 250 тыс. км. и которые не проходили ТО в текущем году.
  - Сколько раз каждый пассажир воспользовался услугами таксопарка?

- Вывести данные пассажиров, которые воспользовались услугами таксопарка более трех раз.
  - Вывести данные о водителе, который ездит на самом дорогом автомобиле.
3. Создать представление, содержащее сведения о незанятых на данный момент водителях
4. Создать хранимые процедуры:
- Вывести данные о зарплате заданного водителя за прошедшие сутки.
  - Для ввода данных о пассажирах, которые заказывали такси в заданном, как параметр, временном интервале.
  - Вывести сведения о том, куда был доставлен пассажир по заданному, как параметр, номеру телефона.
  - Для вычисления суммарного дохода таксопарка за текущий месяц.
5. Создать триггер для фиксации в БД заработанной водителем суммы и начисления ему заработной платы (после высадки пассажира)

### **Вариант 17**

#### **БД "Распределение аудиторного фонда"**

Описание предметной области:

БД содержит сведения об аудиториях и расписании проводимых в них занятий. Время начала и окончания занятия по дням недели фиксировано.

База данных используется для получения справок о наличии свободных аудиторий в указанное время, о месте и времени проведения определенных занятий.

БД должна содержать следующий минимальный набор сведений:

- Номер аудитории.
- Количество мест,
- Тип аудитории.
- Код дисциплины.
- Название дисциплины.
- Вид занятия.
- ФИО преподавателя.
- Номер студенческой группы.
- Максимально возможное количество студентов, посещающих занятие.
- Дата.
- День недели.
- Время начала занятия.

#### **Задания**

1. Создать таблицы, используя необходимые средства поддержки целостности данных. (Ограничения задать самостоятельно)
2. Создать запросы:
  - Вывести список преподавателей, не имеющих занятий в понедельник
  - Найти недельную нагрузку студентов каждой группы

- Вывести список свободных лекционных аудиторий в данное время.
  - Вывести количество аудиторий каждого типа.
  - Вывести еженедельное количество часов занятий для каждой группы.
3. Создать представление, содержащее данные о расписании на каждый день
4. Создать хранимые процедуры:
- Вывести список свободных аудиторий для проведения практических занятий заданной группы в заданное время.
  - Вывести расписание занятий для заданного преподавателя.
  - Вывести список аудиторий, в которых может разместиться заданная, как параметр группа.
5. Создать необходимые триггеры.

### **Вариант 18**

БД "Спортивный клуб".

#### **Описание предметной области:**

БД должна осуществлять: ведение списков спортсменов и тренеров. Тренеры разделены по категориям. При достижении спортсменами определенного рейтинга категория тренера повышается; учёт проводимых соревнований (с ведением их архива); учёт травм, полученных спортсменами.

Предусмотреть: возможность перехода спортсмена от одного тренера к другому; составление рейтингов спортсменов; составление рейтингов тренеров; выдачу информации по соревнованиям; выдачу информации по конкретному спортсмену; подбор возможных кандидатур на участие в соревнованиях (соответствующего уровня мастерства, возраста и без травм)

БД должна содержать следующий минимальный набор сведений:

- ФИО тренера.
- № телефона тренера.
- Паспортные данные тренера.
- Категория тренера.
- Оклад тренера.
- Вид соревнования.
- Категория соревнования.
- Место проведения соревнования.
- Дата проведения соревнования.
- Фамилия спортсмена.
- Имя спортсмена.
- Результат спортсмена.
- Отчество спортсмена.
- Место, которое занял спортсмен.
- Количество баллов спортсмены за место.
- Количество баллов тренера за место.
- Дата рождения спортсмена.

- Категория спортсмена.
- Рейтинг спортсмена.
- Вид травмы.

### **Задания**

1. Создать таблицы, используя необходимые средства поддержки целостности данных. (Ограничения задать самостоятельно).
2. Создать запросы:
  - с каким количеством спортсменов работает каждый тренер,
  - найти тренеров, чьи спортсмены не имеют травм.
  - тренер, получающий минимальную зарплату.
  - количество соревнований каждой категории.
  - тренер, работающий с самыми молодыми спортсменами,
  - сколько спортсменов участвует в соревнованиях каждой категории.
3. Создать представление, содержащее сведения обо всех тренерах, соревнованиях, в которых участвовали их спортсмены и местах которые они заняли.
4. Создать хранимые процедуры:
  - Для вывода данных о результатах заданного спортсмена за прошедший год.
  - Для ввода данных о соревнованиях, проводимых в первом квартале текущего года.
5. Создать триггер для повышения рейтинга спортсмена, рейтинга и оклада тренера после участия в соревновании

### **Вариант 19**

#### **БД "Телефонная станция"**

Описание предметной области:

Информационная система служит для хранения информации об абонентах телефонной станции и для учета оплаты всех видов услуг абонентами. В системе должны храниться сведения о продолжительности разговоров каждого абонента, о стоимости внутренних и междугородных переговоров, о задолженности абонента. Цена минуты в ночное время снижается на 20%.

БД должна содержать следующий минимальный набор сведений:

- ФИО абонента
- Номер телефона.
- Адрес абонента.
- Город.
- Продолжительность.
- Дата звонка.
- Время звонка.
- Код зоны.
- Цена минуты.
- Сумма оплаты.



- Дата оплаты.

### **Задания**

1. Создать таблицы, используя необходимые средства поддержки целостности данных. (Ограничения задать самостоятельно).
2. Создать запросы:
  - Вывести суммарное время переговоров каждого абонента.
  - Найти среднюю продолжительность разговора абонента АТС.
  - Вывести количество междугородных переговоров каждого абонента.
  - Вывести список абонентов, не внесших оплату за прошедший месяц.
  - Сколько звонков было сделано в каждый из следующих городов: в Москву, Лондон, Париж .
  - Вывести список абонентов, звонивших только в ночное время.
3. Создать представление, содержащее сведения обо всех абонентах и их переговорах за прошедший месяц
4. Создать хранимые процедуры:
  - вывести список всех звонков заданного абонента.
  - вывести задолженность по оплате для заданного абонента.
  - рассчитать сумму, которую должен внести каждый абонент
5. Создать необходимые триггеры.

### **Вариант 20**

#### **БД "ГАИ"**

Гаи производит регистрацию автомобилей и следит за безопасностью движения. БД служит для ведения статистики нарушений правил движения. БД должна содержать следующий минимальный набор сведений:

номер водительского удостоверения

- ФИО водителя.
- Адрес.
- Номер телефона.
- Номер автомобиля.
- Марка автомобиля.
- Модель автомобиля.
- Год выпуска.
- Дата регистрации в ГАИ.
- Код нарушения.
- Вид нарушения.
- Сумма штрафа.
- Срок лишения прав управления автомобилем.
- Дата нарушения.
- Время нарушения.
- Район нарушения.
- Личный номер инспектора.

- ФИО инспектора.

### **Задания**

1. Создать таблицы, используя необходимые средства поддержки целостности данных. (Ограничения задать самостоятельно).
2. Создать запросы:
  - Вывести данные водителей многократно (более одного раза) нарушивших правила движения.
  - В каком районе чаще нарушают правила движения.
  - Вывести данные водителей, которые были лишены прав управления автомобилем в текущем месяце.
  - Вывести данные водителей, которые нарушили правила движения в ночное время.
  - Вывести данные инспектора, оштрафовавшего максимальное число водителей.
3. Создать представление, содержащее следующие данные: вид нарушения, время нарушения, номер водительского удостоверения, сумма штрафа.
4. Создать хранимые процедуры:
  - Вывести все сведения о владельце автомобиля по заданному, как параметр номеру автомобиля.
  - Вывести данные инспектора, оштрафовавшего одного и того же водителя более одного раза.
  - Вывести количество нарушений, повлекших лишение прав в заданном, как параметр районе.
5. Создать необходимые триггеры.

Приложение 2  
**Типы данных, используемые в SQL-сервере**

**Текстовые типы данных:**

Название	Описание
<b>Char</b>	Размерность до 8000 символов.
<i>Nchar</i>	Размерность до 4000 символов.
<i>Varchar</i>	Используется для хранения текстовой информации переменной длины. Размерность до 8000 символов.
<i>Nvarchar</i>	Используется для хранения текстовой информации переменной длины. Размерность до 4000 символов.

**Числовые типы данных:**

Название	Описание
<i>Int</i>	Положительные и отрицательные целые числа. Диапазон: от $-2^{31}$ до $+2^{31}$
<i>Smallint</i>	Диапазон: от -32768 до 32767
<i>Tinyint</i>	Положительные целые числа. Диапазон: от 0 до 255
<i>Decimal</i>	Точный числовой, 5-17 байт
<i>Real</i>	Положительные и отрицательные числа с плавающей точкой с точностью до 7 цифр. Диапазон: от $-3,4E-38$ до $+3,4E+38$
<i>Float</i>	Положительные и отрицательные числа с плавающей точкой с точностью до 15 цифр. Диапазон: от $-1,7E-308$ до $+1,7E+308$
<i>Money</i>	Для хранения денежных значений. Диапазон от -922 337 203 685 477,5808 до + 922 337 203 685 477,5807
<i>Smallmoney</i>	от -214 748,3648 до 214 748,3647

**Типы данных даты и времени:**

Название	Описание
<i>Datetime</i>	Хранение комбинации даты и времени. Диапазон: от 01.01.1753 до 31.12.9999
<i>Smalldatetime</i>	Хранение комбинации даты и времени. Диапазон: от 01.01.1900 до 06.06.2079

**Типы данных для хранения больших объемов информации:**

Название	Описание
<i>Text</i>	Большие объемы текстовой информации. Размерность: от 1 до 214 783 647 байт.
<i>Ntext</i>	Большие объемы текстовой информации в Unicode. Размерность: от 1 до 1 073 741 823 байт.
<i>Image</i>	Длинные цепочки двоичных данных, что позволяет, например, записывать в таблицу рисунки, фотографии.

**Типы данных специального назначения:**

Название	Описание
<i>Bit</i>	Информация, принимающая только одно значение. Диапазон: от 0 до 1.
<i>Binary</i>	Хранение битовых цепочек. Размерность: до 80000 байт.
<i>Varbinary</i>	Хранение битовых цепочек варьируемой длины. Размерность: до 80000 байт.
<i>Timestamp</i>	Автоматически размещает значение счетчика каждый раз при вставке

	новой записи.
--	---------------

Репозиторий ВГУ

Приложение 3  
**Типы ограничений**

Тип целостности	Тип ограничения	Краткое название	Описание
Целостность полей	DEFAULT	DF	Определяет значение поля по умолчанию (на случай, если значение не было явно указано в операторе INSERT)
	CHECK	СК	Указывает правило достоверности для значений поля
	FOREIGN KEY	FK	Значения поля (полей) внешнего ключа должны соответствовать значениям поля (полей) первичного ключа таблицы, на которую ссылаются
Целостность сущностей	PRIMARY KEY	PK	Уникально идентифицирует каждую запись. Это гарантирует отсутствие в таблице повторяющихся значений и позволяет создать индекс для повышения производительности. Значения NULL не допускаются
	UNIQUE	U	Предотвращает дублирование непервичных ключей, что позволяет создать индекс для повышения производительности. Допускаются значения NULL
Ссылочная целостность	FOREIGN KEY	FK	Определяет поле или набор полей, значения которых соответствуют первичному ключу связанной таблицы
Целостность, определяемая пользователем	CHECK	СК	Определяет правило допустимости значений данных для поля

Приложение 4  
Данные для заполнения таблиц

**Таблица BRANCH**

<i>Branch_no</i>	<b>Postcode</b>	<b>City</b>	<b>Street</b>	<b>House</b>	<b>Btel_no</b>	<b>Fax-no</b>
1	210009	Витебск	Замковая	4/офис404	8(02122)37-73-56	37-73-58
2	210033	Витебск	Суворова	9/11	8(02122)36-01-80	33-25-23
3	211440	Новополоцк	Молодёжная	18	8(02144)57-31-29	57-25-30
4	211460	Россоны	Ленина	3	8(02159)25-55-20	

**Таблица OWNER**

<i>Owner_no</i>	<b>FName</b>	<b>LName</b>	<b>City</b>	<b>Street</b>	<b>House</b>	<b>Flat</b>	<b>Otel_no</b>
1	Жерносек	Юрий	Витебск	Терешковой	28/1	7	62-07-94
2	Панкратова	Инна	Новополоцк	Парковая	26	12	57-12-48
3	Амбражевич	Сергей	Новополоцк	Двинская	5	18	52-14-89
4	Поскрёбышева	Елена	Витебск	П.Бровки	26/1	40	23-00-72(а6978)
5	Титов	Николай	Витебск	Интернационалистов	35	187	8(029)633-76-68
6	Скребкова	Алла	Новополоцк	Молодёжная	1	22	8(029)688-84-46
7	Николаев	Влад	Витебск	Фрунзе	33	214	8(029)673-07-30
8	Цалко	Сергей	Лепель	Ленина	14/2	4	25-17-90
9	Цыркунова	Наталья	Россоны	Цветочная	90	15	26-32-48
10	Яковлев	Андрей	Витебск	Лазо	65		21-72-25

**Таблица BUYER**

<i>Buyer_no</i>	<b>FName</b>	<b>LName</b>	<b>City</b>	<b>Street</b>	<b>House</b>	<b>Flat</b>	<b>Htel_no</b>	<b>Wtel_no</b>	<b>Prof_Rooms</b>	<b>Max_Price</b>	<b>Branch_no</b>
1	Невердасов	Виктор	Витебск	Московский пр-т	16/4	117	62-08-19	36-40-80	2	110000	2
2	Кассап	Светлана	Новополоцк	Гайдара	17а	4	57-12-48		1	78000	3
3	Орлов	Александр	Минск	Либнехта	93	15	8(017)286-13-21		3	14500	1
4	Сафронова	Светлана	Витебск	пр-т Победы	3/1	324	8(029)661-07-30	22-67-94	2	60000	4
5	Окорков	Вадим	Минск	Лермонтова	35	187		8(017)224-84-24	5	300000	1
6	Семёнов	Вячеслав	Витебск	Замковая	4	13	23-00-72(аб964)		2	10500	1
7	Краснова	Жанна	Витебск	Клиническая	104			23-50-70	1	90000	2
8	Будда	Елена	Лепель	Ленина	3	5	25-17-90		4	200000	3
9	Боровая	Наталья	Орша	Смоленская	12/4	26	26-32-48		2	140000	2
10	Алипов	Игорь	Витебск	Московский пр-т	22	4	21-72-25		3	150000	2

**Таблица STAFF**

<i>Staff_no</i>	<b>FName</b>	<b>LName</b>	<b>DOB</b>	<b>Sex</b>	<b>City</b>	<b>Street</b>	<b>House</b>	<b>Flat</b>	<b>Stel_no</b>	<b>Date_Joined</b>	<b>Position</b>	<b>Salary</b>	<b>Branch_no</b>
ВМО550260	Батуркин	Александр	17.10.68	М	Новополоцк	Парковая	5	13	23-79-77	1.01.2001	менеджер	2500000	3
ВМО550261	Чубаро	Наталья	25.05.72	Ж	Витебск	Чкалова	21/1	12	8(029)662-47-32	10.02.2002	торговый агент	1800000	1
ВМО550262	Коваленко	Светлана	1.02.70	Ж	Витебск	Чкалова	24	49	62-51-23	15.01.2001	менеджер	2500000	3
ВМО550263	Логинов	Вадим	9.09.67	м	Витебск	Чкалова	41/2	96	23-06-73	2.09.1999	директор	3000000	1
ВМО550264	Суворов	Виталий	11.07.65	М	Новополоцк	Школьная	47/1	157	22-29-03	1.02.1999	торговый агент	1800000	3
ВМО550265	Жарков	Герман	6.03.68	М	Витебск	Гагарина	31	28	63-43-98	2.10.2001	менеджер	2800000	2
ВМО550266	Ганущенко	Галина	5.11.69	ж	Витебск	Лазо	90/1	54	62-43-64	3.11.2004	торговый агент	1800000	2
ВМО550267	Сотникова	Ольга	7.12.67	Ж	Полоцк	Вокзальная	7			2.02.2002	маклер	1000000	3
ВМО550268	Янчиленко	Сергей	28.02.70	М	Россоны	Ленина	28/2	25		17.05.2005	торговый агент	1500000	4

Таблица PROPERTY

<i>Property_no</i>	<i>Data_registration</i>	<i>Postcode</i>	<i>City</i>	<i>Street</i>	<i>House</i>	<i>Flat</i>	<i>Floor_Type</i>	<i>Floor</i>	<i>Rooms</i>	<i>The_area</i>	<i>Balcony</i>	<i>Ptel</i>	<i>Selling_Price</i>	<i>Branch_no</i>	<i>Staff_no</i>	<i>Owner_no</i>
3000	16.12.11	210033	Витебск	Смоленская	11	57	5П	3	1	31/18/10	Б	Т	60000	1	BMO550262	1
3001	17.12.11	210035	Витебск	Бровки	11	49	9К	9	1	37/21/7	Бз	-	70000	2	BMO550262	5
3002	18.12.11	210029	Витебск	Строителей	23/2	214	12П	2	2	81/29/9	Лз	Т	92000	2	BMO550266	7
3003	19.12.11	210005	Витебск	Лазо	11	4	3К	3	2	65/40/7,6	2Бз	-	15000	1	BMO550261	5
3004	15.12.11	211460	Россоны	Ленина	32	20	5П	5	3	68,4/44,3/9,81	2Бз	Т	100000	4	BMO550268	7
3005	11.12.11	211440	Новопол	Школьная	11	56	9П	3	1	36/18/8,2	Б	Т	75000	3	BMO550260	6
3006	14.12.11	211440	Новопол	Молодёжная	5	14	9П	2	2	46/27/6,8	Лз	Т	60000	3	BMO550264	3
3007	20.12.11	211180	Полоцк	Вокзальная	8	15	5К	5	3	65/38/7	Б	Т	80000	3	BMO550267	2
3008	16.01.11	211460	Россоны	Советская	17	1	5К	1	3	65/38/7	-	Т	47500	4	BMO550268	7

Таблица VIEWING

<i>Date_View</i>	<i>Comments</i>	<i>Property_no</i>	<i>Buyer_no</i>
17.01.12	согласен	3002	1
17.01.12	согласен	3003	1
18.01.12	не согласен	3002	4
19.01.12	согласен	3005	2
25.03.12	требует ремонта	3001	7



## Приложение 5

### Краткое определение основных терминов

**База Данных (БД)** – некоторый набор таблиц, связанных между собой определенной связью для выполнения различного рода задач.

**Блокировка** – временно накладываемое ограничение на выполнение некоторых операций обработки данных.

**Внешние ключи** – это поля таблицы, которые соответствуют первичным ключам из других таблиц.

**Данные** – это информация, зафиксированная в определенной (структурированной) форме, пригодной для последующей обработки, хранения и передачи.

**Запрос** – специальным образом описанное требование, определяющее состав производимых над базой данных операций по выборке или модификации хранимых данных.

**Импорт данных** – это процесс, в результате которого данные извлекаются из внешних источников и после соответствующей обработки вставляются в таблицы базы данных SQL Server.

**Индекс** – упорядоченный список полей или групп полей в таблице, используемый для ускорения операции поиска записей в таблице, а также выполнения других операций, использующих поиск.

**Нормализация** - удаление избыточных данных из каждой таблицы в базе данных, с целью устранения дублирования и обеспечения непротиворечивости хранимых данных.

**Первичный ключ** – это уникальное поле (или несколько полей), однозначно определяющее запись таблицы базы данных.

**Представление** - некоторое подобие таблицы, содержание которого выбирается из других таблиц с помощью выполнения запроса, причем, при изменении значений в таблицах, данные автоматически меняются и в представлении.

**Привилегии** - права пользователя на проведение тех или иных действий над определенным объектом базы данных.

**Резервное копирование** - создание резервной копии базы данных.

**Репликация** – это копирование данных, расположенных на одном сервере, на один или несколько других серверов.

**Система Управления Базами Данных (СУБД)** представляет собой комплекс инструментальных средств (программных и языковых) реализующих централизованное управление БД и обеспечивающих доступ к данным (изменения, добавления, удаления, резервного копирования и т.д.).

**Стандартное значение (умолчание)** – это значение, которое будет присвоено колонке таблицы при вставке строки, если в команде явно не указано значение для этой колонки

**Транзакция** – последовательность операций над базой данных, отслеживаемая системой управления базами данных от начала до завершения как единое целое.

**Триггеры** – это специальный класс хранимых процедур, автоматически запускаемых при добавлении, изменении и удалении данных из таблицы.

**Хранимая процедура** – программа (процедура) обработки данных, хранящаяся и выполняемая на компьютере-сервере.

**Экспорт данных** – это процесс, в результате которого данные из таблицы SQL Server трансформируются в указанный формат.

**Языковые средства** – языки, с помощью которых описывается структура данных (DDL) и языки манипулирование данными (DML).

Приложение 6

### **Дополнительные параметры создаваемого индекса**

**Unique values** – ввод в определённое поле только уникальных значений. Позволяет осуществлять автоматическую проверку уникальности при каждом добавлении новой записи. Рекомендуется установить обязательный ввод значений в поле, для которого планируется создание уникального индекса;

**Clustered index** – в системе SQL-сервер имеется возможность физического индексирования данных. Другими словами, использование индексов приводит к созданию отдельной структуры, которая связывается с физическим расположением данных в таблице. Использование этой опции позволяет произвести так называемое кластерное индексирование, в результате чего будут отсортированы данные в самой таблице согласно порядку этого индекса, и вся добавляемая информация будет приводить к изменению физического порядка данных. При этом нужно учитывать, что в таблице может быть определён только один кластерный индекс;

**Ignore duplicate values** – игнорирование ввода повторяющихся значений в проиндексированных полях. Использование данного параметра совместно с *Unique values* позволяет игнорировать уникальность значений для этого поля. Обычно использование этого параметра имеет значение при ориентации на распределение данных в разрабатываемых структурах;

**Do not recompute statistics (not recommended)** – определяет функцию автоматического обновления статистики для таблицы. Устанавливать не рекомендуется.

**File group** – осуществляет выбор файловой группы, в которой будет находиться создаваемый индекс. Использование индекса из другой файловой группы повышает производительность некластерных индексов в связи с параллельностью выполнения процессов ввода/вывода и работы с самим индексом. При выборе данного параметра активизируется список файловых групп, позволяющий определить необходимую группу для размещения индекса;

**Fill factor** – осуществляется настройка разбиения индекса на страницы. Коэффициент *fillfactor* определяет в процентном соотношении размер создаваемых индексных страниц. Если планируется частое изменение, удаление и добавление информации в таблице базы данных, то коэффициент следует установить как можно меньше, например, 20. А установка коэффициенту значения 100 рекомендуется при использовании больших таблиц, обращение к которым обычно происходит только для чтения;

**Pad index** – определяет заполнение внутреннего пространства индекса и используется совместно с параметром *fillfactor*;

**Drop existing** – определяет повторное создание кластерного индекса, что позволяет предотвратить нежелательное обновление кластерных индексов.

## Приложение 7 Список функций SQL Server

### Компоненты даты

<i>Компонент</i>	<i>Сокращение</i>	<i>Допустимые значения</i>
<i>Год (Year)</i>	<i>Yy</i>	<i>1753-9999</i>
<i>Месяц (Month)</i>	<i>Mm</i>	<i>1-12</i>
<i>День года (Dayofyear)</i>	<i>Dy</i>	<i>1-366</i>
<i>День месяца (Day)</i>	<i>Dd</i>	<i>1-31</i>
<i>День недели (Weekday)</i>	<i>Dw</i>	<i>1-7 (1 = воскресенье)</i>

### Некоторые глобальные переменные

<i>Глобальная переменная</i>	<i>Описание</i>
<i>@@CONNECTIONS</i>	<i>Количество успешных подключений плюс количество неудачных попыток</i>
<i>@@CURSOR_ROWS</i>	<i>Количество записей в последнем открытом курсоре</i>
<i>@@ERROR</i>	<i>Номер ошибки для последней команды TSQL</i>
<i>@@IDENTITY</i>	<i>Последнее вставленное значение счётчика</i>
<i>@@LANGUAGE</i>	<i>Название текущего языка</i>
<i>@@LANGID</i>	<i>Идентификатор текущего языка</i>
<i>@@MAX_CONNECTIONS</i>	<i>Максимальное количество одновременных подключений, разрешенное SQL Server</i>

<i>Глобальная переменная</i>	<i>Описание</i>
<i>@@ROWCOUNT</i>	<i>Количество записей, участвовавших в выполнении последней команды</i>
<i>@@TRANCOUNT</i>	<i>Количество активных транзакций</i>
<i>@@VERSION</i>	<i>Дата, версия SQL Server и тип процессора</i>

#### Список функций

<i>Функция</i>	<i>Описание</i>
<i>ABS(числовое_выражение)</i>	<i>Модуль числа</i>
<i>ACOS(вещественное_выражение)</i>	<i>Арккосинус (в радианах)</i>
<i>ASCII(символьное_выражение)</i>	<i>ASCII-код первого символа в строке</i>
<i>ASIN(вещественное_выражение)</i>	<i>Арксинус (в радианах)</i>
<i>ATAN(вещественное_выражение)</i>	<i>Арктангенс (в радианах)</i>
<i>CASE</i>	<i>Вычисление результата по списку выражений</i>
<i>CAST</i>	<i>Преобразование типа данных</i>
<i>CEILING</i>	<i>Наименьшее целое, большее либо равное заданной величине</i>
<i>CHAR(целое_выражение)</i>	<i>Преобразование целого числа в ASCII-символ</i>
<i>CHARINDEX(символьное_выражение, выражение, начало)</i>	<i>Начальная позиция подстроки в выражении (0, если подстрока не найдена)</i>
<i>COALESCE(выражение1, выражение2)</i>	<i>Первое выражение, отличное от NULL</i>
<i>CONVERT(тип_данных, выражение [, стиль])</i>	<i>Преобразование типа данных</i>
<i>COS(вещественное_выражение)</i>	<i>Косинус угла</i>
<i>COT(вещественное_выражение)</i>	<i>Котангенс угла</i>
<i>CURRENT_USER</i>	<i>Эквивалент USER_NAME( )</i>

<i>Функция</i>	<i>Описание</i>
<i>DATALENGTH</i> (символьное_выражение)	<i>Целое число символов в выражении, не считая завершающих пробелов</i>
<i>ROUND</i> (числовое_выражение, целое_выражение[,функция])	<i>Округление числового выражения до позиции, задаваемой целым выражением. Если третий аргумент отличен от 0, ROUND не округляет, а отбрасывает цифры</i>
<i>DATENAME</i> (компонент, выражение_дата)	<i>Компонент даты, выраженный в виде строки. По возможности преобразуется в имя (например, June)</i>
<i>DATEPART</i> (компонент, выражение_дата)	<i>Заданный компонент даты в виде целого числа</i>
<i>DATEDIFF</i> (компонент, выражение_дата1, выражение_дата2)	<i>Разность дат, выраженная в заданных компонентах</i>
<i>DAY</i> (дата)	<i>Целое число, определяющее день месяца</i>
<i>EXP</i> (вещественное_выражение)	<i>Экспонента заданного числа</i>
<i>FLOOR</i> (числовое_выражение)	<i>Наибольшее целое, меньшее либо равное заданной величине</i>
<i>GETDATE</i> ( )	<i>Текущие системная дата и время</i>
<i>GETANSINULL</i> (имя БД)	<i>Выводит допустимо или недопустимо использовать в БД значение NULL</i>
<i>IDENT_SEED</i> (таблица)	<i>Выводит начальное значение счетчиков в таблице</i>
<i>IDENT_INCR</i> ({имя_таблицы / имя_представления})	<i>Величина приращения столбца счётчика</i>
<i>IS_MEMBER</i> ({группа / роль})	<i>1, если текущий пользователь является членом группы NT или роли SQL Server, 0 — если не является и NULL, если группа/роль не определена</i>

<i>Функция</i>	<i>Описание</i>
<i>IS_SRVROLEMEMBER</i> (роль{,имя} )	<i>1, если имя пользователя входит в роль SQL Server</i>
<i>ISDATE</i> (выражение)	<i>1, если выражение определяет допустимую дату</i>
<i>ISNUMERIC</i> (выражение)	<i>1, если выражение является числовым</i>
<i>LEFT</i> (символьное_выражение, целое_выражение)	<i>Заданное число символов от начала строки</i>
<i>LEN</i> (символьное_выражение)	<i>Длина строки в символах без учета завершающих пробелов</i>
<i>LOG10</i> (вещественное_выражение)	<i>Десятичный логарифм</i>
<i>LTRIM</i> (символьное_выражение)	<i>Удаление начальных пробелов</i>
<i>LOWER</i> (символьное_выражение)	<i>Преобразование строки в нижний регистр</i>
<i>MONTH</i> (дата)	<i>Целый номер месяца</i>
<i>PI</i> ( )	<i>Константа 3,1415926...</i>
<i>POWER</i> (основание, степень)	<i>Возведение в степень</i>
<i>QUOTENAME</i> (символьное_выраж	<i>Преобразование строки в идентификатор.</i>
<i>RADIANS</i> ( )	<i>Преобразование градусов в радианы</i>
<i>RAND</i> ([целое_выражение])	<i>Получение случайного вещественного числа в интервале от 0 до 1 (необязательное целое • выражение используется для раскрутки генератора случайных чисел)</i>
<i>REPLACE</i> (символьное_выражение 1, символьное_выражение2, <i>REVERSE</i> (символьное_выражение )	<i>Замена всех экземпляров строки 2 в строке 1 на строку 3</i> <i>Обратная перестановка строки</i>
<i>RIGHT</i> (символьное_выражение, целое_выражение)	<i>Заданное число символов в конце строки</i>
<i>ROUND</i> (числовое_выражение, целое_выражение [, функция])	<i>Округление числового выражения до позиции, задаваемой целым выражением. Если третий аргумент отличен от 0, ROUND не округляет, а отбрасывает цифры</i>

<i>Функция</i>	<i>Описание</i>
<i>RTRIM</i> (символьное_выражение)	<i>Удаление завершающих пробелов</i>
<i>SIGN</i> (целое_выражение)	<i>+1 для положительных чисел, -1 для отрицательных и 0 для нуля</i>
<i>SIN</i> (вещественное_выражение)	<i>Синус угла</i>
<i>SPACE</i> (целое_выражение)	<i>Построение строки из заданного</i>
<i>SQRT</i> (вещественное_выражение)	<i>Извлечение квадратного корня</i>
<i>SQUARE</i> (вещественное_выражение)	<i>Возведение в квадрат</i>
<i>STR</i> (вещественное_выражение)	<i>Преобразование числа в строку</i>
<i>STUFF</i> (символьное_выражение1, <i>n</i> , символы строки 1, начиная с заданной	<i>Получение подстроки</i>
<i>SUBSTRING</i> (выражение, начало, <i>длина</i> )	<i>Получение подстроки</i>
<i>SUSER_ID</i> ([имя_пользователя])	<i>Идентификатор пользователя SQL Server</i>
<i>SYSTEM_USER</i>	<i>Имя текущего пользователя SQL Server</i>
<i>TAN</i> (вещественное_выражение)	<i>Тангенс угла</i>
<i>UPPER</i> (символьное_выражение)	<i>Преобразование строки в верхний регистр</i>
<i>USER_ID</i> (имя_в_базе)	<i>Идентификатор пользователя в базе данных</i>
<i>USER_NAME</i> ([идентификатор_пользователя] или <i>SESSION_USER</i> )	<i>Имя пользователя в базе данных</i>
<i>YEAR</i> (выражение_дата)	<i>Год в виде четырёх цифр</i>
<i>SUM</i> ([ <i>all</i>   <i>distinct</i> ] выражение)	<i>Вычисляет общее количество (разных) значений в числовом столбце</i>
<i>AVG</i> ([ <i>all</i>   <i>distinct</i> ] выражение)	<i>Вычисляет среднее арифметическое</i>
<i>COUNT</i> ([ <i>all</i>   <i>distinct</i> ] выражение)	<i>Вычисляет количество (разных) значений в числовом столбце, отличных от NULL</i>
<i>COUNT</i> (*)	<i>Вычисляет количество выбранных записей. Единственная агрегатная функция, учитывающая значения NULL</i>
<i>MAX</i> (выражение)	<i>Определяет максимум среди выбранных значений</i>
<i>MIN</i> (выражение)	<i>Определяет минимум среди выбранных значений</i>

---

<i>Функция</i>	<i>Описание</i>
<i>STDEV(выражение)</i>	<i>Вычисляет среднее квадратичное отклонение</i>
<i>STDEVP(выражение)</i>	<i>Вычисляет среднее квадратичное отклонение для выборки</i>
<i>VAR(выражение)</i>	<i>Вычисляет статистическую дисперсию</i>
<i>VARP(выражение)</i>	<i>Вычисляет статистическую дисперсию</i>

Репозиторий ВГУ



Министерство образования Республики Беларусь  
Учреждение образования «Витебский государственный университет имени  
П.М. Машерова»

*Модели данных и СУБД*  
*Практикум*

*Витебск*  
*2012*