

АСТРАХАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

А.Н. Умеров

**Администрирование
и программирование
Microsoft SQL Server**

Практикум

Астрахань
Издательство АГТУ
2011

УДК 004.42
ББК 32.97
У 52

Рецензенты:

Умеров А.Н.

У 52 Администрирование и программирование Microsoft SQL Server: практикум / А.Н. Умеров; Астрахан. гос. техн. ун-т. – Астрахань: Изд-во АГТУ, 2010. – 120 с.

Практикум предназначен для получения практических навыков по администрированию и программированию СУБД Microsoft SQL Server. Рассмотрены вопросы установки, конфигурирования и оптимизации БД, использования языка Transact SQL и применения специализированных утилит. Может использоваться для самостоятельной работы студентов.

Практикум утвержден на заседании кафедры АСОИУ
от 01.06.2010 г., протокол № 6

УДК 004.42
ББК 32.97

© Умеров А.Н., 2010

Введение

Microsoft SQL Server является одним из лидеров рынка систем управления баз данных (СУБД). Данный программный продукт отличается широкой степенью масштабируемости: от небольших и средних настольных баз данных до крупных высоконагруженных баз данных уровня предприятия.

Наличие бесплатно распространяемой версии SQL Server Express Edition позволяет минимизировать расходы на использовании как в образовательных целях в вузе, так и в производственных в масштабах небольшой организации. Также Microsoft SQL Server поставляется с удобной средой администрирования, разработки и профилирования, что немаловажно при создании крупных проектов.

В данном практикуме представлены задания, позволяющие в короткие сроки получить практические навыки и умения, необходимые для развертывания, разработки приложений баз данных с использованием реляционных команд языка SQL и расширения Transact-SQL, устранения неполадок и оптимизации баз данных под управлением СУБД Microsoft SQL Server. В ходе выполнения практикума, студентами будут получены навыки по созданию объектов базы данных, составлению сложных запросов, профилированию и оптимизации доступа к данным.

Несмотря на то, что данный практикум ориентирован на использование СУБД Microsoft SQL Server (текущая версия на момент издания практикума SQL Server 2008 R2 R2), полученные навыки будут несомненно полезны разработчикам и при разработке для других СУБД, например, Oracle.

Лабораторная работа № 1

«Установка и первый запуск SQL Server»

Цель работы

- Получить навыки установки Microsoft SQL Server 2008.

Проверка конфигурации ПК

Перед началом установки необходимо проверить соответствие системных требований конфигурации ПК, на который производится установка. Для этого необходимо проделать следующие действия:

1. Запустите командную строку, выполните команду **systeminfo**.
2. Зафиксируйте данные о названии и версии операционной системы; полном объеме физической памяти; типе системы; числу, названию и тактовой частоте процессоров.
3. Сравните полученные данные с минимальными требованиями к аппаратному и программному обеспечению SQL Server 2008 R2 с указанными в приложении данными.
4. Сделайте вывод о возможности (или невозможности) установки.

Установка

1. Запустите файл запуска установки SQL Server 2008 R2 **setup.exe**, расположенный в папке с дистрибутивом программы.
2. В раскрывшемся окне установки перейдите к разделу «Установка». Затем выберите пункт «Новая установка изолированного SQL Server или добавление компонентов к существующему экземпляру» (рис. 1.1).

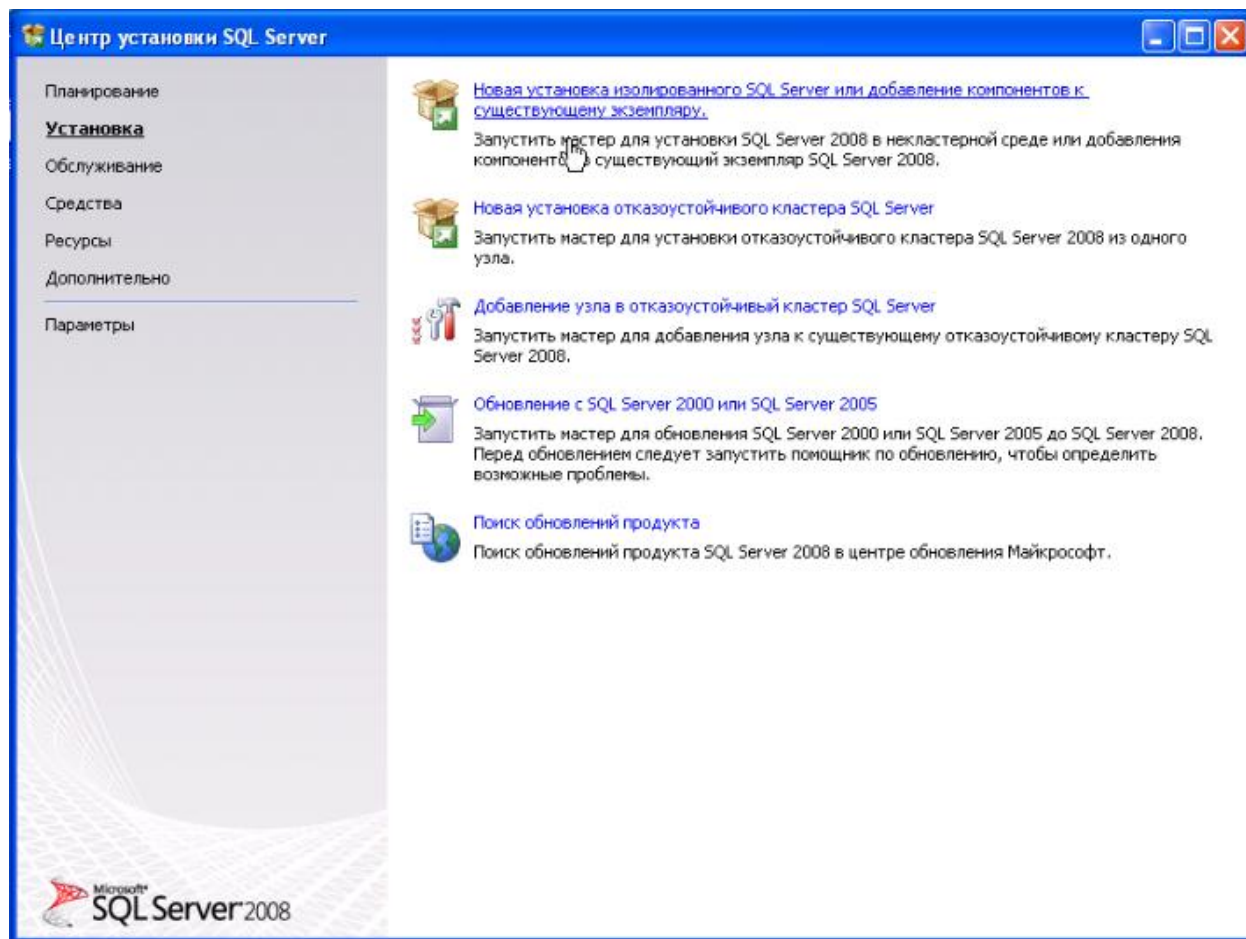


Рис. 1.1. Окно центра установки SQL Server

3. После выбора необходимого пункта, в силу вступят правила поддержки установки. Убедитесь, что проверка соответствия правилам завершена без ошибок и нажмите **ОК** (рис. 1.2).

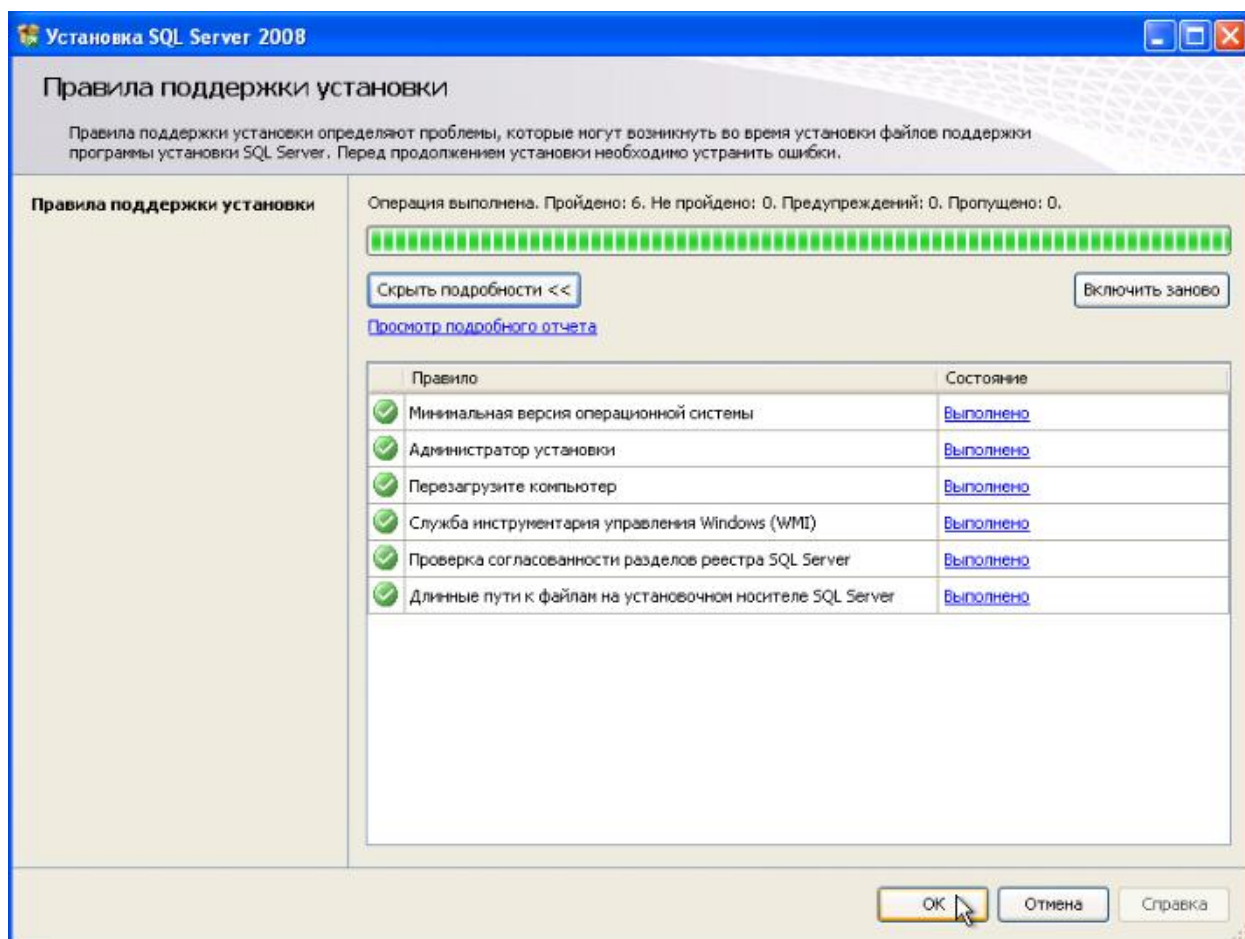


Рис. 1.2. Окно - проверка соответствия правилам установки

4. В появившемся окне выберите пункт **«Введите ключ продукта»**. Если ключ продукта не введен в поле автоматически, введите его. Затем нажмите кнопку **«Далее»**.
5. В появившемся окне нажмите кнопку **«Установить»**, чтобы установить необходимые файлы поддержки установки. Дождитесь окончания установки.
6. После установки файлов, откроется окно со 2-ой проверкой правил поддержки установки (рис. 1.3). Убедитесь, что проверка прошла без ошибок и несоответствий и нажмите **«Далее»**.

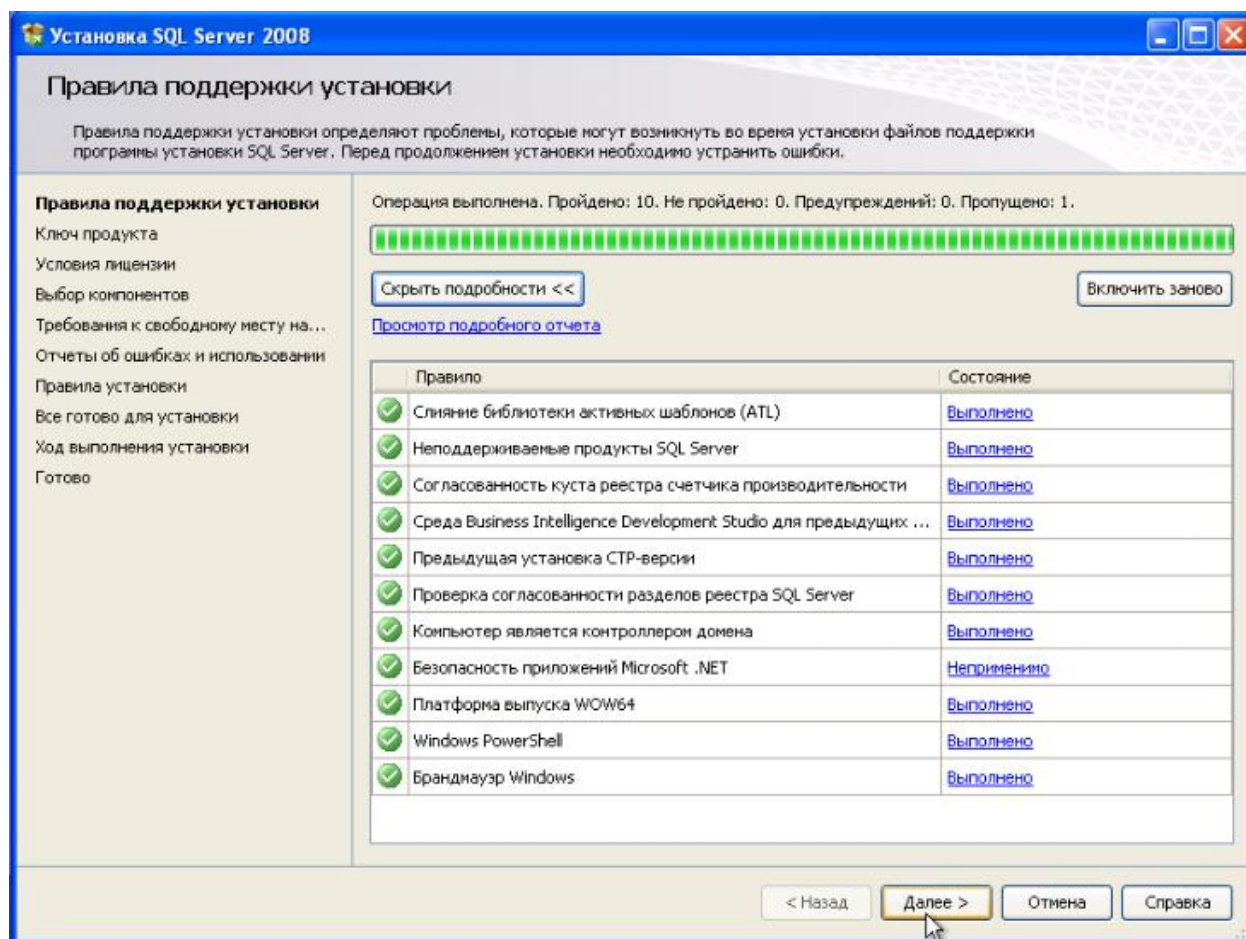


Рис. 1.3. Окно – проверка соответствий правилам установки

7. В появившемся окне выбора компонентов нажмите кнопку «**Выделить все**». Затем нажмите «**Далее**».
8. В появившемся окне настройки экземпляра укажите установки именованного экземпляра, назовите его «**newSQLSERVER**» (рис. 1.4). Нажмите «**Далее**».

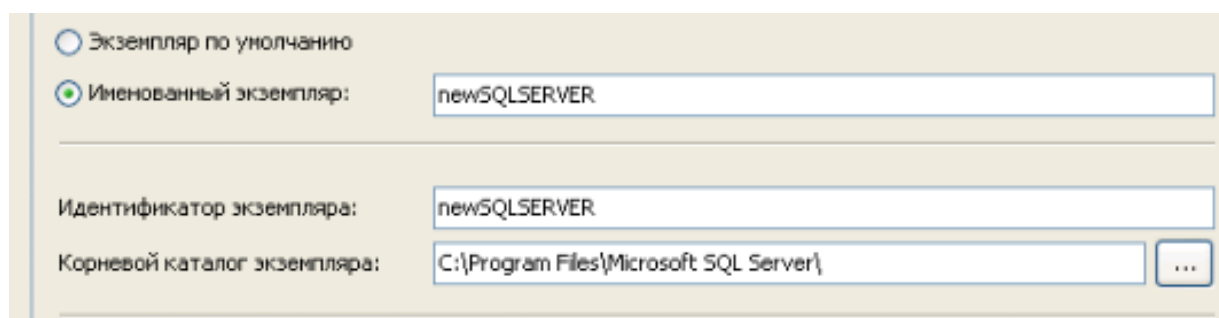


Рис. 1.4. Установка именованного экземпляра

9. В появившемся окне приведены требования к свободному месту на диске. Убедитесь, что имеется достаточное количество свободного места, и нажмите кнопку «Далее» (рис. 1.5).

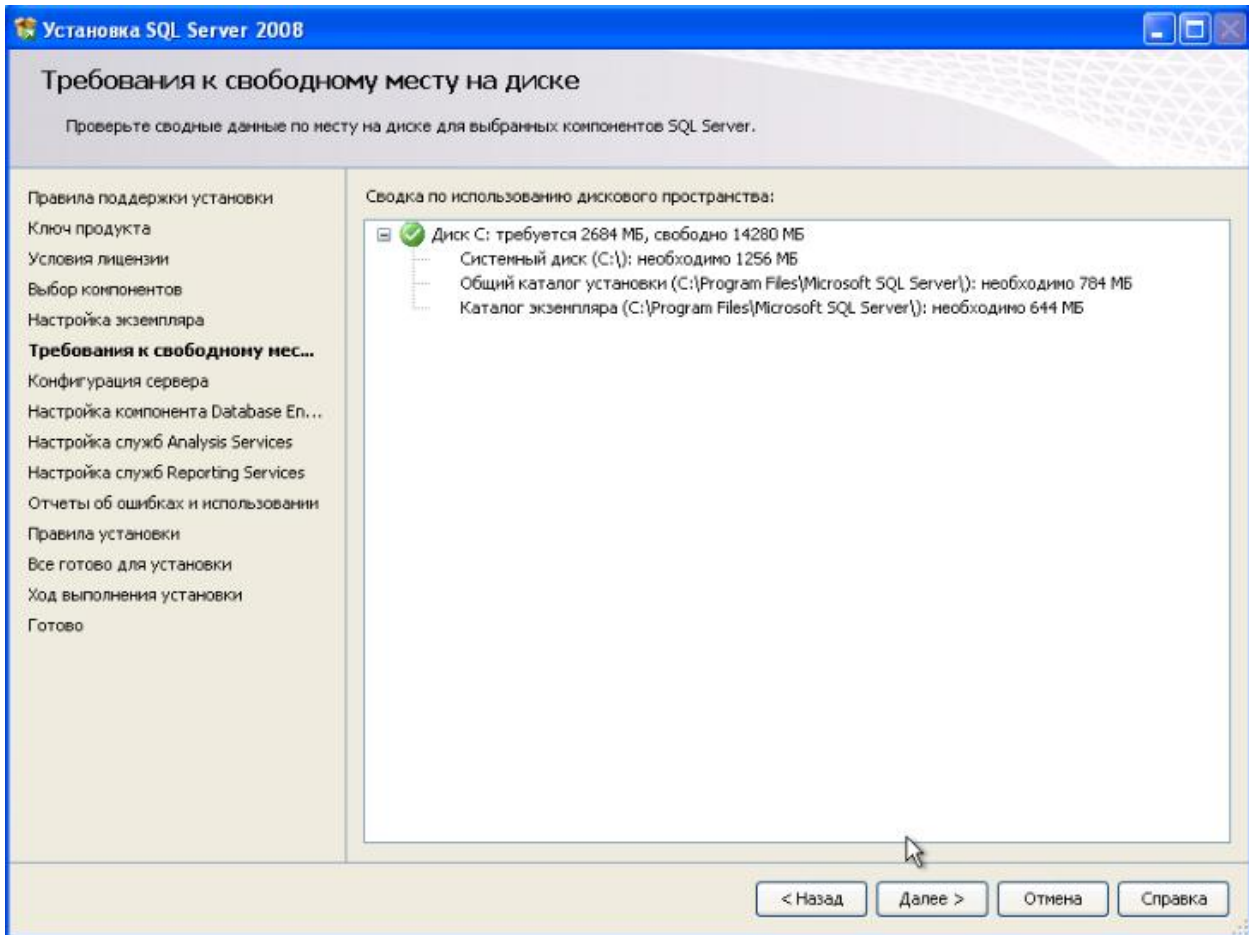


Рис. 1.5. Проверка наличия свободного места на диске

Перед выполнением следующего шага необходимо создать учетную запись операционной системы Windows «**dbuser**» со стойким паролем, например, «**X552d_(md=gEr)**». Эта учетная запись будет использована для запуска компонентов и служб СУБД SQL Server 2008. Для создания пользователя можно воспользоваться оснасткой «Локальные пользователи и группы» или выполнить команду:

```
net user dbuser "X552d_(md=gEr" /ADD
```

При создании учетной записи, она будет автоматически добавлена в группу **Пользователи**, это необходимо с целью обеспечения безопасности

(SQL Server 2008 R2 будет исполняться от имени непривилегированной учетной записи)

10. В появившемся окне конфигурации сервера необходимо указать параметры запуска служб и компонентов и задать учетные записи, их запускающие.

1) Нажмите кнопку «Использовать одну и ту же учетную запись для всех служб SQL Server»;

2) В появившемся окошке ввода имени учетной записи и пароля нажмите кнопку «Обзор»;

3) Укажите учетную запись с именем «dbuser»;

4) Введите пароль учетной записи в поле ввода пароля и нажмите ОК (рис. 1.6);

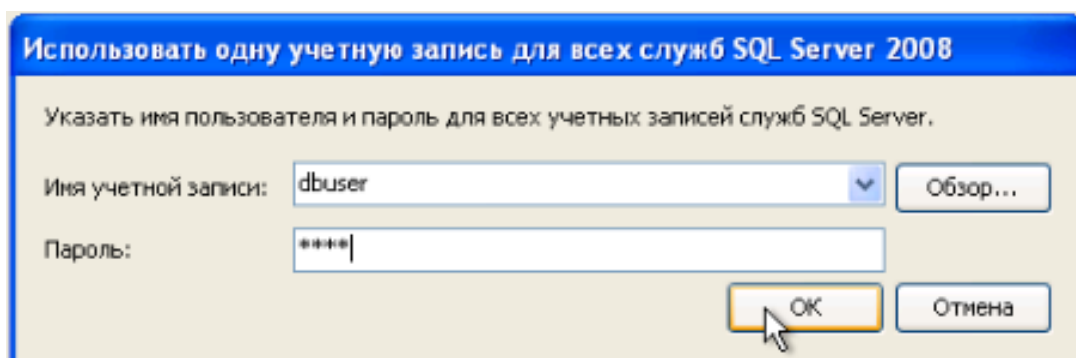


Рис. 1.6. Указание учетной записи

5) Введите данные учетной записи «**dbuser**» (имя учетной записи, пароль) в соответствующие поля напротив компонента «SQL Full-text Filter Daemon Launcher» и нажмите кнопку «Далее» (рис. 1.7)

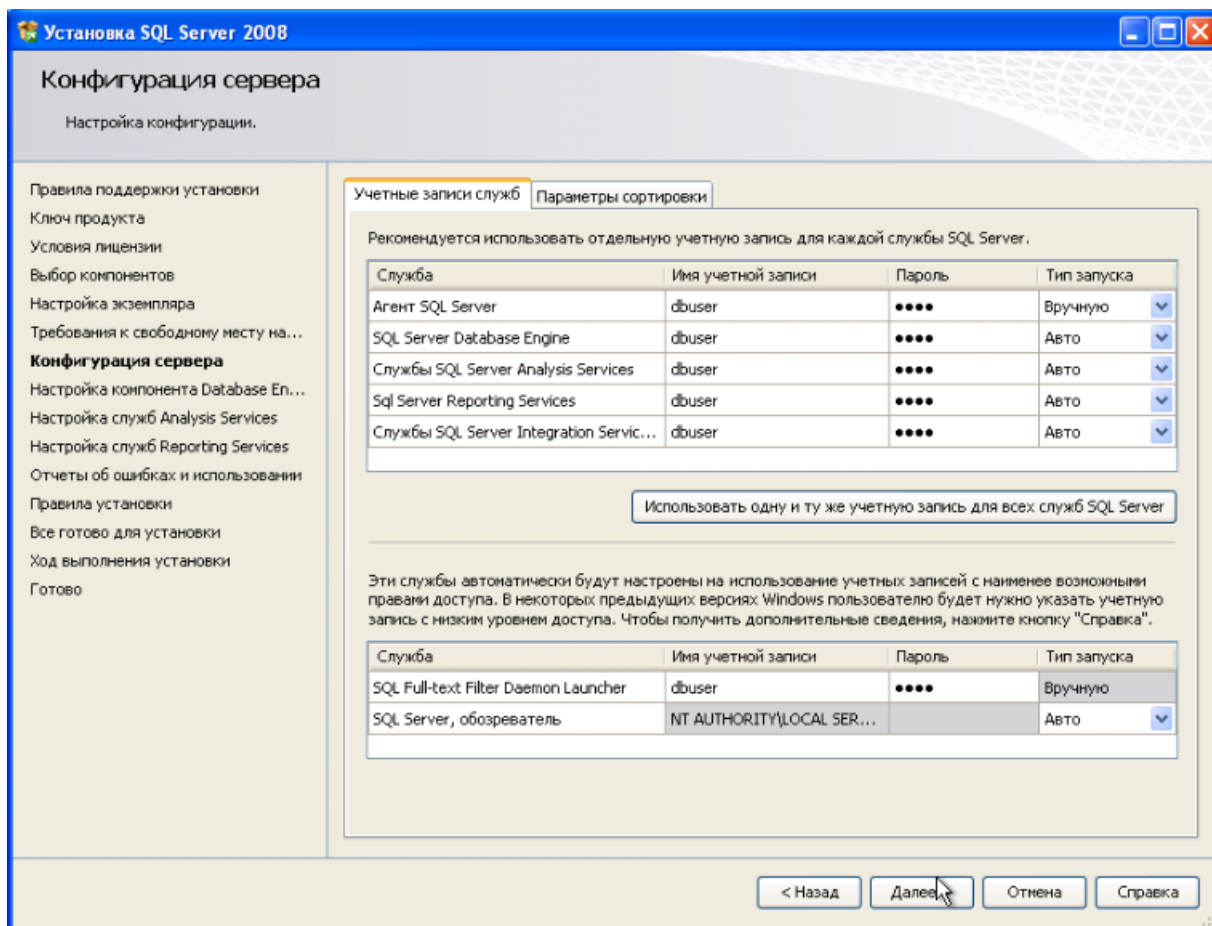


Рис. 1.7. Окно конфигурации сервера

11. В появившемся окне настройки компонента Database Engine назначьте администратором SQL Server текущего пользователя, нажав на кнопку **«Добавить текущего пользователя»**.

Перейдите на вкладку **«FILESTREAM»**. Отметьте пункт **«Разрешить FILESTREAM при доступе через Transact-SQL»**, затем отметьте пункт **«Разрешить FILESTREAM при потоковом доступе файлового ввода-вывода»** (рис. 1.8). Нажмите кнопку **«Далее»**.

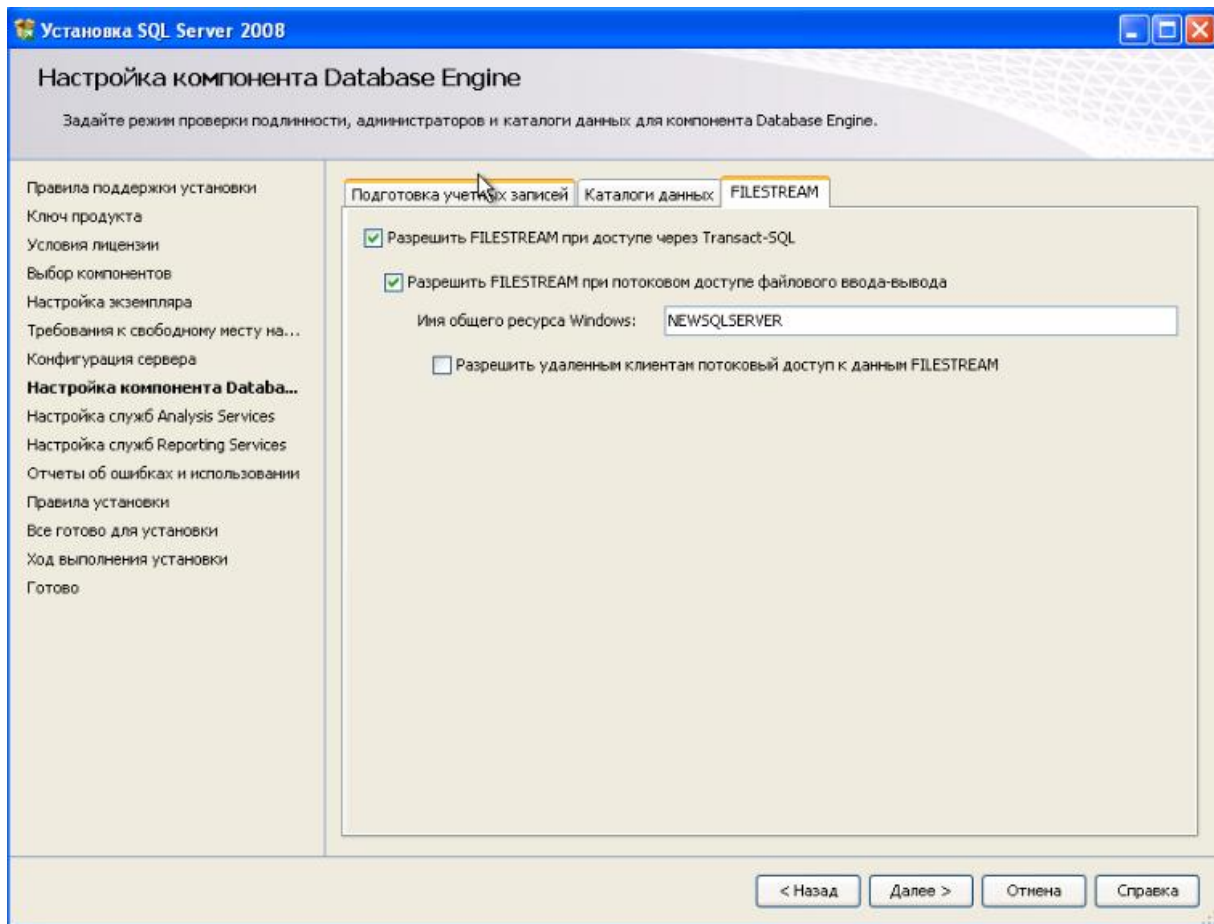


Рис. 1.8. Параметры FILESTREAM

12. В появившемся окне настройки служб Analysis Services назначьте текущего пользователя администратором службы Analysis Services, нажав на кнопку **«Добавить текущего пользователя»**.
13. В появившемся окне настройки служб Reporting Services оставьте конфигурацию по умолчанию и нажмите кнопку **«Далее»**.
14. В открывшемся окне отчетов об ошибках и использовании не вносите изменения и нажмите кнопку **«Далее»**.
15. В появившемся окне последней проверки соответствия правилам установки убедитесь, что проверка не была завершена с ошибками и соответствиями и нажмите кнопку **«Далее»** (рис. 1.9).

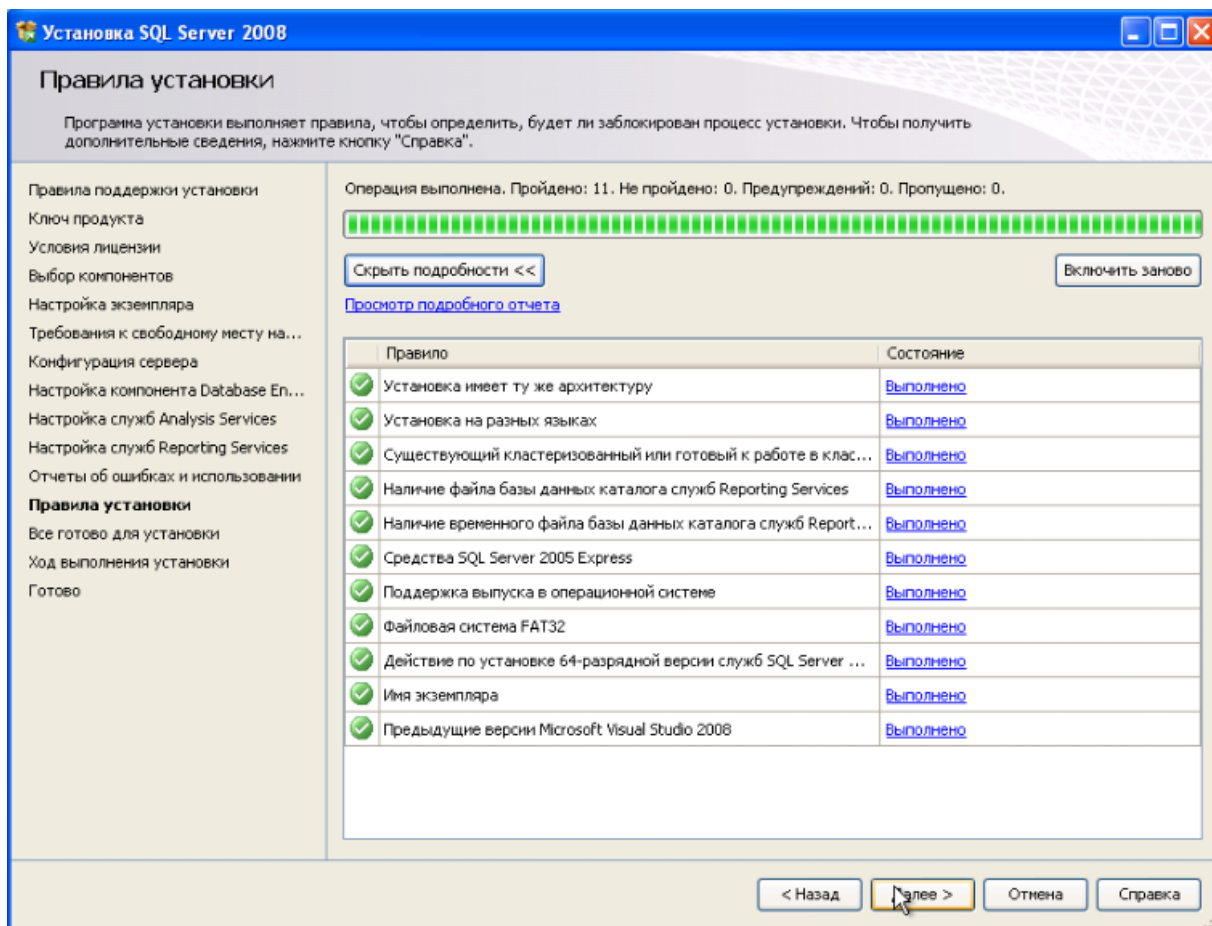


Рис. 1.9. Последняя проверка перед установкой

16. В появившемся окне проверки всех параметров установки убедитесь, что все установлено в соответствии с заданием лабораторной работы и нажмите кнопку «**Установить**» (рис. 1.10).

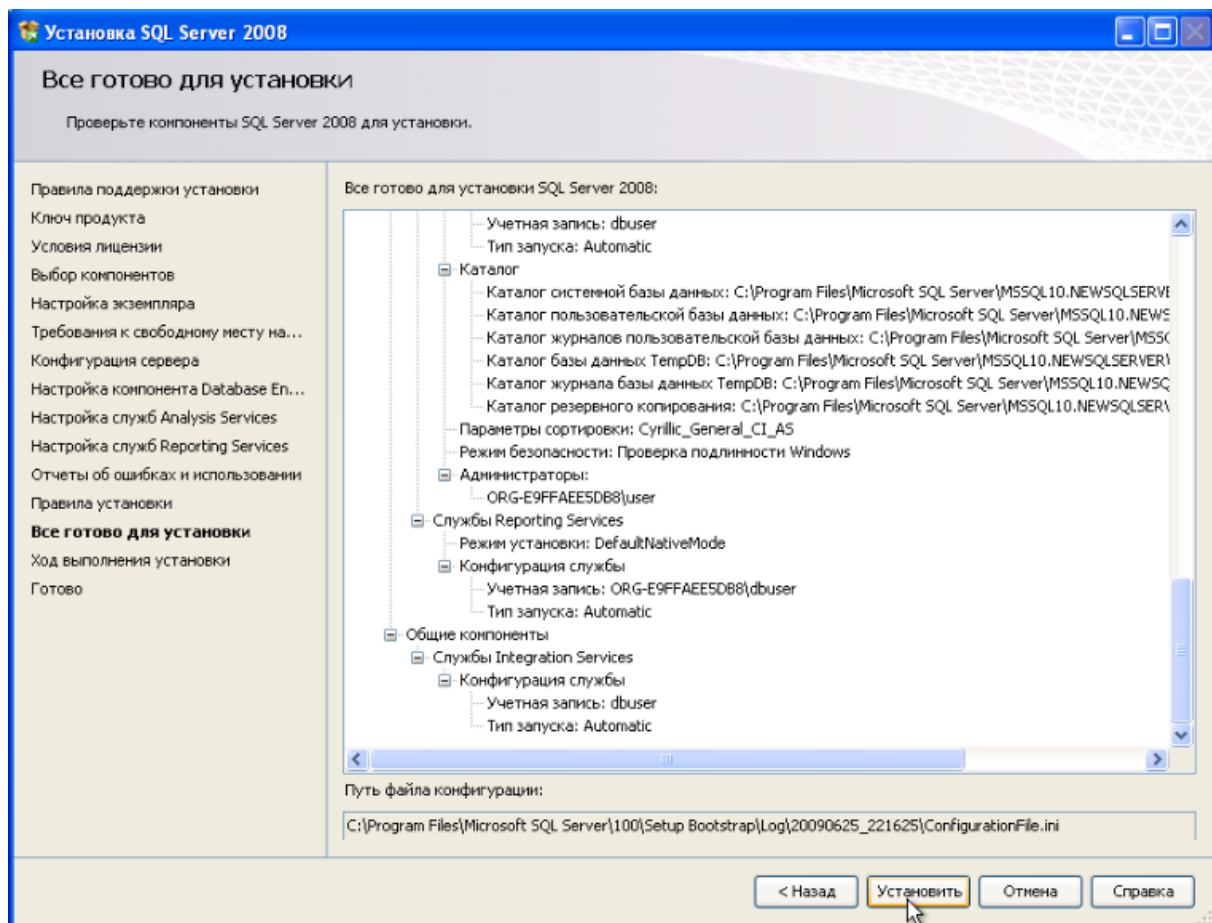


Рис. 1.10. Все готово для установки

Проследите за ходом установки SQL Server 2008. Убедитесь, что установка успешно завершится.

Первый запуск

1. Откройте Пуск -> Программы -> Microsoft SQL Server 2008-> SQL Server Management Studio
2. В окне подключения к экземпляру сервера нажмите кнопку «Соединить» (рис. 1.11)

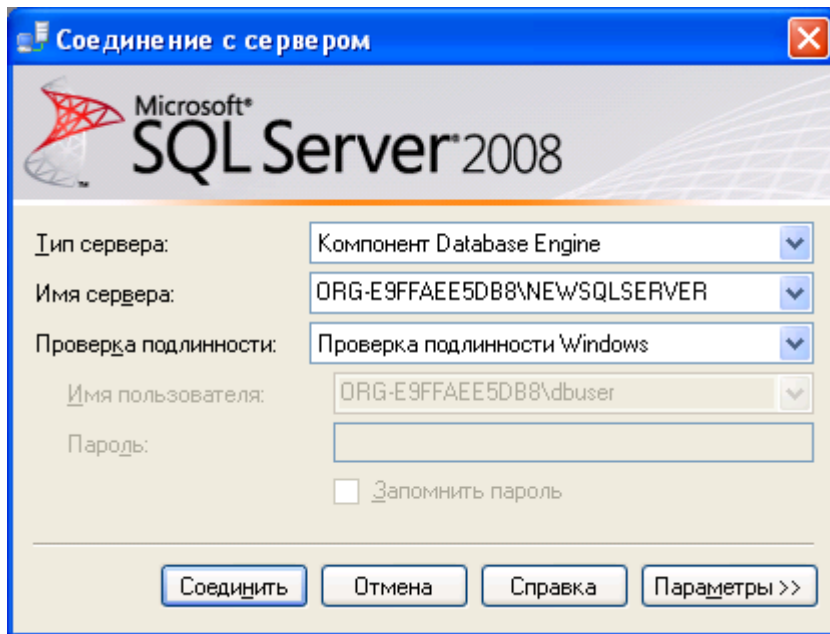


Рис. 1.11. Окно подключения к серверу

3. Создайте файл запросов в среде Management Studio, нажав на кнопку **«Создать запрос»**.

Напишите в новом файле запросов следующий скрипт:

```
CREATE database testbase
ON PRIMARY
(
    NAME = testbase,
    FILENAME = 'C:\SQL\testbase.mdf',
    SIZE = 10MB,
    MAXSIZE = UNLIMITED,
    FILEGROWTH = 1MB
)
LOG ON
(
    NAME = testbase,
    FILENAME = 'C:\SQL\testbase_log.ldf',
    SIZE = 5MB,
    MAXSIZE = UNLIMITED,
    FILEGROWTH = 1%
)
```

Запустите запрос на выполнение, нажав на кнопку «**Выполнить**» (рис. 1.12).

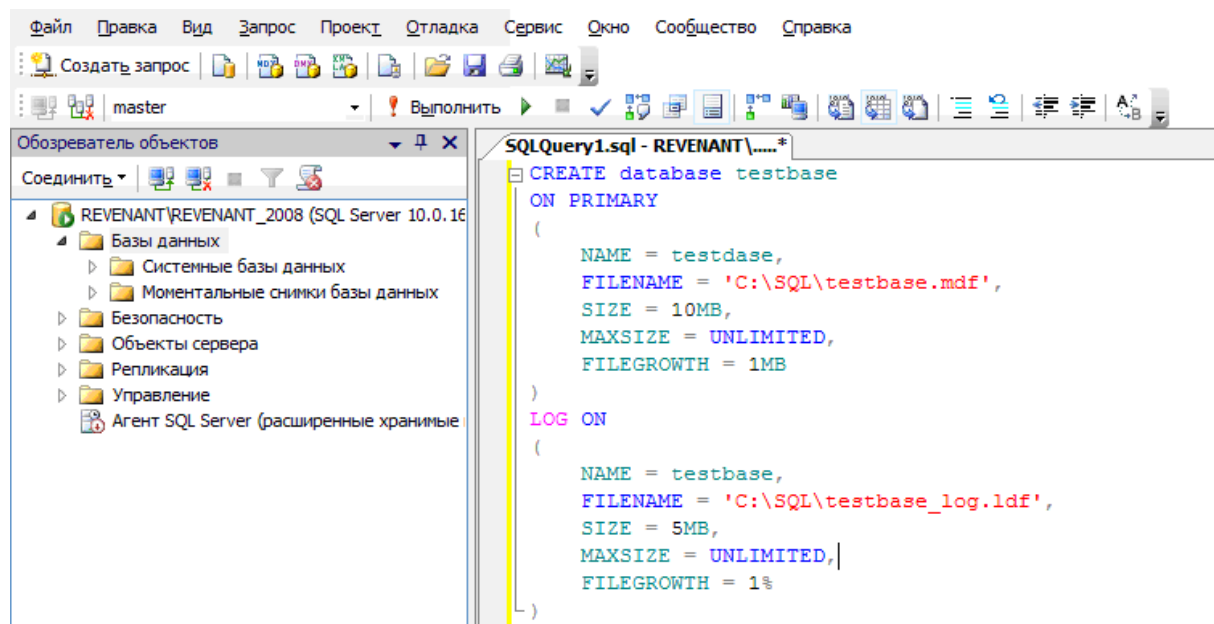


Рис. 1.12. Фрагмент окна SQL Server Management Studio

По окончании лабораторной работы необходимо предоставить отчет, содержащий конфигурацию ПК, на который была проведена установка, и системные требования установленной редакции, а также снимки экрана для каждого шага установки.

Лабораторная работа №2

«Создание баз данных»

Часть 1. Создание пользовательской базы данных при помощи SQL Server Management Studio

1. Запустите SQL Server Management Studio. Войдите в экземпляр сервера по умолчанию. В дереве объектов окна Object Explorer на ветви Database вызовите контекстное меню и выберите «New database». Имя Базы данных должно быть составлено по следующему принципу «Фамилия (в транслитерации) Инициалы (все без пробелов)», например: «IvanovAA».
2. В окне «New Database» в свойстве «General» создать пользовательский файл данных базы данных (его имя определить по аналогии с шаблоном «MyBaseIvanov»). Изменить файловую группу на собственную (имя по аналогии с шаблоном «Ivanov»). Установить первоначальный размер вновь созданного файла данных базы данных в 10 MB. Установите приращение, с которым будет увеличиваться размер файла при его заполнении в 10MB. Измените путь к вашему файлу на «D:\SQL» (предварительно создав такую папку на диске D).
3. Выберите созданную базу данных в окне «Object Explorer». В раскрывающемся списке вызовите контекстное меню папки «Tables», в нем «New Table». Создайте такое количество таблиц с такими полями и ограничениями, как указано в вашем варианте.
4. Создайте требующиеся внешние ключи в ваших таблицах. Для этого в окне «Object Explorer» в вашей базе данных в папке «Tables» раскройте ветку для соответствующей таблицы. Вызовите контекстное меню на папке «Keys» и выберите «New Foreign key...». Просмотрите результат выполнения пункта задания на диаграмме.
5. Сделайте скриншоты дизайнеров всех таблиц, скриншоты тех свойств полей таблиц, значение которых отлично от значения по умолчанию, скриншот диаграммы, скриншоты окон определения внешних ключей

для связей таблиц. Сохраните сделанные снимки с разъясняющими подписями в файле-отчете. Удалите базу данных.

Часть 2. Создание пользовательской базы данных при помощи Transact-SQL

1. Выполните первый и второй пункт части первой при помощи языка T-SQL. Синтаксис и использование директивы CREATE DATABASE приведены ниже.

```
CREATE DATABASE database_name
```

```
[ ON
```

```
[ PRIMARY ]
```

```
[ <filespec>[ ,,...n ] ]
```

```
[ ,<filegroup>[ ,,...n ] ]
```

```
]
```

```
[ [ LOG ON {[<filespec>[ ,,...n ] ]}]
```

```
[ COLLATE collation_name ]
```

```
[ WITH <external_access_option>]
```

```
]
```

```
[ ;]
```

```
<filespec>::=
```

```
{(NAME =logical_file_name ,
```

```
FILENAME ='os_file_name '
```

```
[ ,SIZE =size [ KB |MB |GB |TB ] ]
```

```
[ ,MAXSIZE ={max_size [ KB |MB |GB |TB ] ||UNLIMITED }]
```

```
[ ,FILEGROWTH =growth_increment [ KB |MB |%] ]
```

```
)
```

```
}
```

```
<filegroup>::=
```

```
{FILEGROUP filegroup_name [ DEFAULT ] <filespec>[ ,,...n ] }
```

```
<external_access_option>::=
```

```
{DB_CHAINING {ON |OFF }
```

```
|TRUSTWORTHY {ON |OFF }
```

```
}
```

Пример использования

```
USE master
GO
CREATE DATABASE Sample
ON PRIMARY
(NAME =Sample1,
FILENAME ='c:\data \sampledat1.mdf',
SIZE =100MB,
MAXSIZE =UNLIMITED,
FILEGROWTH =10%
),
(NAME =Sample2,
FILENAME ='c:\data \sampledat2.ndf',
SIZE =100MB,
MAXSIZE =UNLIMITED,
FILEGROWTH =10%
)
LOG ON
(NAME =SampleLog1,
FILENAME ='c:\data \samplelog1.ldf',
SIZE =3MB,
MAXSIZE =UNLIMITED,
FILEGROWTH =5MB
) GO
```

2. Выполните третий и четвертый пункты части первой при помощи языка T-SQL. Синтаксис и использование директивы CREATE TABLE приведены ниже. Итоговые скрипты сохраните в файле-отчете.

```
CREATE TABLE
[ database_name .[schema_name ].[schema_name .]table_name
({<column_definition>|<computed_column_definition>}
[ <table_constraint>] [ ,...n ]
```

```

)
[ ON {partition_scheme_name (partition_column_name )
|filegroup_name

|"DEFAULT"
        }
]
[ {TEXTIMAGE_ON {filegroup_name |"DEFAULT"}}]
[ ;]
<column_definition>::=
column_name <data_type>
[ COLLATE collation_name ]
[ NULL |NOT NULL ]
[ [ CONSTRAINT constraint__name ] DEFAULT constant__expression
]
|[ IDENTITY [ ((seed ,increment ) ) [ NOT FOR REPLICATION ] ]
[ ROWGUIDCOL ] [ <column_constraint>[ .....n ] ]

<data type>::=
[ type_schema_name .]type_name
[ (precision [ ,scale ] ||max
|[ {{CONTENT |DOCUMENT }}xml_schema_collection )
]
<column_constraint>::=
[ CONSTRAINT constraint_name ]
{{PRIMARY KEY |UNIQUE }
[ CLUSTERED |NONCLUSTERED ]
[ WITH FILLFACTOR =fillfactor
|WITH (<index_option>[ ,,...n ] ))
]
[ ON {partition_scheme_name (partition_column_name )
|filegroup_name
|"DEFAULT"
        }
]
|[ FOREIGN KEY ]

```

```

REFERENCES [ schema_name . ] ref_table_name [ ( ref_column_name ) ]
[ ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
[ ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
[ NOT FOR REPLICATION ]
| CHECK [ NOT FOR REPLICATION ] ( logical_expression )
}
<computed_column_definition>::=
column_name AS computed_column_expression
[ PERSISTED [ NOT NULL ] ]
[ [ CONSTRAINT constraint__name ]
{ PRIMARY KEY | UNIQUE }
[ CLUSTERED | NONCLUSTERED ]
[ WITH FILLFACTOR = fillfactor
| WITH ( <index_option> [ , , ... n ] ) )
]
| [ FOREIGN KEY ]
REFERENCES ref_table_name [ ( ref_column_name ) ]
[ ON DELETE { NO ACTION | CASCADE } ]
[ ON UPDATE { NO ACTION } ]
[ NOT FOR REPLICATION ]
| CHECK [ NOT FOR REPLICATION ] ( ( logical_expression )
[ ON { partition_scheme_name ( partition_column_name )
| filegroup_name
| "DEFAULT"
}
]
]
<table_constraint >::=
[ CONSTRAINT constraint_name ]
{ { PRIMARY KEY | UNIQUE }
[ CLUSTERED | NONCLUSTERED ]
( column_name [ ASC | DESC ] [ , , ... n ] ) )
[ WITH FILLFACTOR = fillfactor
| WITH ( <index_option> [ , , ... n ] ) )
]
[ ON { partition_scheme_name ( partition_column_name )

```

```

| filegroup_name
| "DEFAULT"
}
]
| FOREIGN KEY (column_name [ ,...n ] ))
REFERENCES ref_table_name [ (ref_column_name [ ,...n ] )]]
[ ON DELETE {NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
[ ON UPDATE {NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
[ NOT FOR REPLICATION ]
| CHECK [ NOT FOR REPLICATION ] ((logical_expression )
}

```

```

<index_option>::=
{PAD_INDEX = {ON | OFF }
| FILLFACTOR = fillfactor
| IGNORE_DUP_KEY = {ON | OFF }
| STATISTICS_NORECOMPUTE = {ON | OFF }
| ALLOW_ROW_LOCKS = {ON | OFF }
| ALLOW_PAGE_LOCKS = {ON | OFF } }

```

Использование

```

USE OrderSystemDB
CREATE TABLE Sales.Customers
(cust_lname varchar(40) NOT NULL,
cust_fname varchar(20) NOT NULL,
phone char(12) NOT NULL,
uid uniqueidentifier NOT NULL DEFAULT newid()
)

```

3. Используя директиву «DROP DATABASE» удалите созданную базу данных. Добавьте блок удаления в отчет.

```

DROP DATABASE {database_name | database_snapshot_name } [ ,...n ]

```

Использование

```
USE master
ALTER DATABASE Customer
SET SINGLE_USER
GO
DROP DATABASE Customer
GO
```

Вариант 1

Схема базы данных состоит из четырех отношений:

Автор (id_authors int, fio_authors varchar(20))

- id_authors является первичным ключом таблицы ***Автор***
- значение поля fio_authors не может принимать значения NULL и по умолчанию равно «Пушкин»

Издательство (id_publishers int, name varchar(20))

- id_publishers является первичным ключом таблицы ***Издательство***
- значение поля name не может принимать значения NULL и не может быть равно пустой строке.

Книга (id_books int, name varchar(50), date_out datetime)

- id_books является первичным ключом таблицы ***Книга***
- значение поля date_out является вычисляемым (используется функция получения текущей даты/времени).

Сводная (id_out int, id_books int, id_authors int, id_publishers int)

- id_out является первичным ключом таблицы ***Сводная***
- id_books является внешним ключом к таблице ***Книга***. Обеспечить каскадное обновление и удаление записей.
- id_authors является внешним ключом к таблице ***Автор***. При обновлении или удалении установить значения соответствующих столбцов таблицы в NULL.
- id_publishers является внешним ключом к таблице ***Издательство***.

Схема базы данных представлена на рис. 2.1.

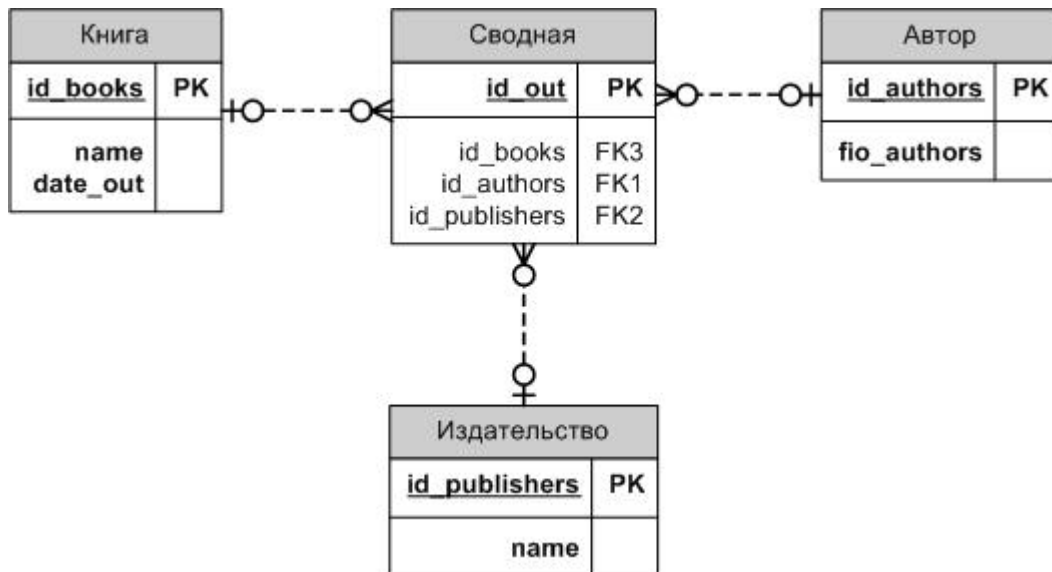


Рис. 2.1. Схема базы данных

Вариант 2

Схема базы данных состоит из трех отношений:

***Гостиница* (*id_hotel int*, *name varchar(20)*, *address varchar(50)*)**

- *id_hotel* является первичным ключом таблицы ***Гостиница***.
- значение поля *name* не может принимать значения NULL.
- значение поля *address* не может принимать значения NULL и по умолчанию равно «г. Астрахань»

***Организация* (*id_organization int*, *name varchar(50)*, *short_name varchar(5)*)**

- *id_organization* является первичным ключом таблицы ***Организация***.
- значение поля *name* не может принимать значения NULL.
- значение поля *short_name* является вычисляемым и представляет собой первые пять символов значения поля *name*.

***Участники_форума* (*id_participant int*, *fio varchar(50)*, *id_organization int*, *id_hotel int*)**

- *id_participant* является первичным ключом таблицы

Участники_форума

- значение поля fio может принимать значения NULL
- id_organization является внешним ключом к таблице *Организация*. Обеспечить каскадное обновление и удаление записей.
- id_hotel является внешним ключом к таблице *Гостиница*. При обновлении или удалении установить значения соответствующих столбцов таблицы в NULL.

Схема базы данных представлена на рис. 2.2.



Рис. 2.2. Схема базы данных

Лабораторная работа №3
«Язык модификации данных DML»

Вариант 1

Схема базы данных состоит из четырех отношений:

Автор (id_authors int, fio_authors varchar(20))

Издательство (id_publishers int, name varchar(20))

Книга (id_books int, name varchar(50), year_out varchar(4))

Сводная (id_out int, id_books int, id_authors int, id_publishers int)

Автор

id_authors	fio_authors
1	Иванов И. И.
2	Петров П. П.
3	Сидоров С. С.
4	Кузнецов К. К.

Издательство

id_publishers	name
	Питер
2	Дрофа
3	Феникс
4	Финансы и статистика

Книга

id_books	name	year_out
1	Основы программирования	2003
2	Мировая экономика	2000
3	Информатика	2007
4	Операционные системы	2007
5	Физика 11 класс	1999
6	Химия 10 класс	2005

Сводная

id_out	id_books	id_authors	id_publishers
1	1	3	1
2	2	1	4
3	6	4	2
4	5	2	2
5	3	3	1
6	3	1	1
7	4	1	1
8	4	1	3

Схема базы данных представлена на рис. 3.1.



Рис. 3.1. Схема базы данных

DML

Задание №1

Создать и добавить в таблицы *Автор*, *Издательство*, *Книга*, *Сводная* данные, указанные на первом листе задания. Добавлять данные, используя конструкцию:

insert into *name_table* (значения параметров по порядку через запятую)

Внимание!!! Первоначально заполняются таблицы-справочники.

Задание №2

Добавить в таблицу *Книга* запись (значение поля *name*='SQL Server', *year_out*=2008) используя в конструкции **insert into** *name_table* конструкцию **select**:

```
insert into name_table  
select @var as name_column
```

Задание №3

Добавить в таблицу *Автор* три записи, используя один оператор **insert into** *name_table*. (значения поля *name*: 'Андреев А.А.', 'Киреев К.К.', 'Максимов М.М.').

Задание №4

Добавить в таблицу *Книга* запись (значение поля *name*='MySQL', *year_out*= значение по умолчанию)

Задание №5

Добавить в таблицу *Сводная* запись со следующими параметрами:

```
id_books   = максимальному в таблице Книга;  
id_authors = минимальному в таблице Автор;  
id_publishers = среднему значению в таблице Издательство  
(отбросить дробную часть).
```

Задание №6

Удалить из таблицы *Сводная* все записи о книгах с минимальным годом выпуска используя конструкцию:

```
delete from name_table  
where условие
```

Задание №7

Изменить таблицу **Сводная** так, чтобы все книги написанные автором, код у которого равен 3 печатались в издательстве “Дрофа”.
Используйте конструкцию:

```
update name_table  
set name_column=@var
```

Задание №8

Добавить к году выпуска книги в таблице **Книга** единицу, но только для тех книг, вторая буква в названии которых ‘и’.

SQL

Задание №9

Найти количество книг с максимальным годом выпуска.

Вывод: *год выпуска, количество книг*

Задание №10

Найти количество авторов у каждой книги.

Вывод: *название книги, количество авторов*

Задание №11

Найти только те книги, которые написали два автора.

Вывод: *название книги*

Задание №12

Найти только те издательства, в названиях которых есть буква ‘ф’.

Вывод: *название издательства*

Задание №13

Найти книги, выпущенные до 2000 года и книги выпущенные после 2006.

Вывод: *название книги, год издания*

Задание №14

Найти количество букв “о” в фамилии и инициалах каждого автора.

Вывод: *фамилия и инициалы, количество букв “о”*

Задание №15

Найти тех авторов, которые написали не менее двух книг.

Вывод: *фамилия и инициалы, количество книг*

Задание №16

Найти сколько раз встречается каждая буква алфавита в названиях книг.

Вывод: *буква, количество повторений*

Вариант 2

Схема базы данных состоит из трех отношений:

Гостиница (*id_hotel int, name varchar(20), address varchar(50)*)

Организация (*id_organization int, name varchar(50)*)

Участники_форума (*id_participant int, fio varchar(50), id_organization int, id_hotel int*)

Гостиница

id_hotel	name	address
1	Астория	ул. Новолесная, строение 5
2	Азимут	ул. Заводская, строение 138
3	Флоренция	ул. Зеленая, дом 46
4	Сафари	бульвар Солнечный, строение 8

Организация

id_organization	name
1	Астраханский государственный технический университет
2	Астраханский государственный университет
3	Саратовская академия права
4	Волгоградская медицинская академия
5	Московский государственный университет
6	Архангельский государственный университет

Участники_форума

id_participant	fio	id_organization	id_hotel
1	Иванов Иван Иванович	1	2
2	Петров Петр Петрович	1	1
3	Сидоров Сидор Сидорович	1	1
4	Александров Александр	2	2
5	Михайлов Михаил Михайлович	3	1
6	Юрьев Юрий Юрьевич	4	3
7	Андреев Андрей Андреевич	4	4

8	Степанов Степан Степанович	5	3
9	Максимов Максим Максимович	6	2

Схема базы данных представлена на рис.3.2.

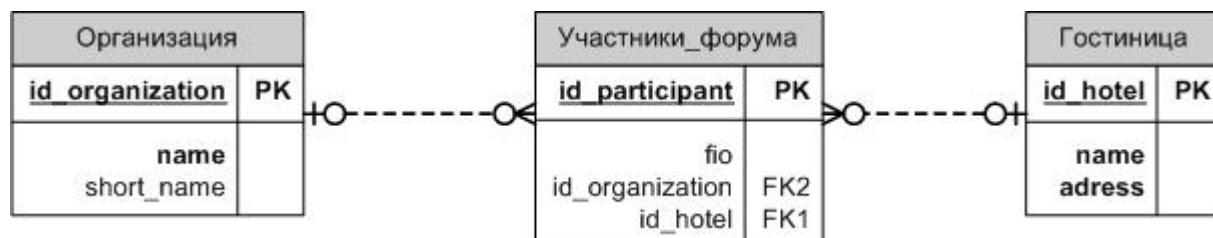


Рис.3.2. Схема базы данных

DML

Задание №1

Создать и добавить в таблицы *Гостиница*, *Организация*, *Участники_форума* данные, указанные на первом листе задания. Добавлять данные, используя конструкцию:

insert into *name_table* (значения параметров по порядку через запятую)

Внимание!!! Первоначально заполняются таблицы-справочники.

Задание №2

Добавить в таблицу *Гостиница* запись (значение поля *name*='Надежда', *address*='ул. Красная набережная, строение 90') используя в конструкции **insert into** *name_table* конструкцию **select**:

```
insert into name_table
select @var as name_column
```

Задание №3

Добавить в таблицу *Организация* три записи, используя один оператор **insert into** *name_table*. (значения поля *name*: ‘Волгоградский государственный технический университет’, ‘Волгоградский государственный университет’, ‘Московский институт статистики и информатики’).

Задание №4

Добавить в таблицу *Гостиница* запись (значение поля *name*=‘Встреча’, *address*=значение по умолчанию).

Задание №5

Добавить в таблицу *Участники_форума* запись со следующими параметрами:

```
fio = 'Викторов Виктор Викторович';  
id_organization = минимальному в таблице Организация;  
id_hotel = максимальному значению в таблице Гостиница.
```

Задание №6

Удалить из таблицы *Участники_форума* все записи об участниках форума, проживающих в гостинице ‘Азимут’ используя конструкцию:

```
delete from name_table  
where условие
```

Задание №7

Изменить таблицу *Участники_форума* так, чтобы все участники, проживающие в гостинице ‘Астория’ переселились в гостиницу ‘Сафари’. Используйте конструкцию:

```
update name_table  
set name_column=@var
```


Задание №8

Заменить полные названия организаций их аббревиатурами.

SQL

Задание №9

Найти организацию с максимальным идентификатором.

Вывод: *организация, идентификатор.*

Задание №10

Найти количество участников от каждой организации.

Вывод: *название организации, количество участников.*

Задание №11

Найти только те гостиницы, в которых проживает не менее двух участников.

Вывод: *название гостиницы.*

Задание №12

Найти только те организации, в названиях которых есть сочетание букв 'гос'.

Вывод: *название организации.*

Задание №13

Найти организации, идентификаторы которых меньше чем 2 или больше чем 4.

Вывод: *название организации, идентификатор.*

Задание №14

Найти количество букв "й" в фамилии, имени и отчестве каждого из участников форму.

Вывод: *фамилия имя и отчество, количество букв "й".*

Задание №15

Найти в какое количество гостиниц организация расселила своих участников.

Вывод: *название организации, количество гостиниц.*

Задание №16

Найти сколько раз встречается каждая буква алфавита в фамилии, имени и отчестве участников форума.

Вывод: *буква, количество повторений.*

Лабораторная работа №4

«Программируемые объекты SQL Server. Функции»

Вариант 1

Схема базы данных состоит из четырех отношений:

Автор (id_authors int, fio_authors varchar(20))

Издательство (id_publishers int, name varchar(20))

Книга (id_books int, name varchar(50), year_out varchar(4))

Сводная (id_out int, id_books int, id_authors int, id_publishers int)

Автор

id_authors	fio_authors
1	Иванов И. И.
2	Петров П. П.
3	Сидоров С. С.
4	Кузнецов К. К.

Издательство

	name
1	Питер
2	Дрофа
3	Феникс
4	Финансы и статистика

Книга

id_books	name	year_out
1	Основы программирования	2003
2	Мировая экономика	2000
3	Информатика	2007
4	Операционные системы	2007

5	Физика 11 класс	1999
6	Химия 10 класс	2005

Сводная

id_out	id_books	id_authors	id_publishers
1	1	3	1
2	2	1	4
3	6	4	2
4	5	2	2
5	3	3	1
6	3	1	1
7	4	1	1
8	4	1	3

Схема базы данных представлена на рис. 4.1.



Рис. 4.1. Схема базы данных

Скалярные функции

Задание №1

Создать функцию, принимающую в качестве параметра фамилию автора и возвращающую количество написанных им книг. Если автор не написал ни одной книги, функция должна вернуть 0. Если автора не существует в базе данных, функция должна вернуть -1.

Создание функции осуществляются при помощи следующей конструкции:

```
CREATE FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ] [ type_schema_name. ]
parameter_data_type
    [ = default ] }
    [ ,...n ]
]
)
RETURNS return_data_type
    [ WITH <function_option> [ ,...n ] ]
    [ AS ]
BEGIN
    function_body
    RETURN scalar_expression
END
[ ; ]
<function_option>::=
{
    [ ENCRYPTION ]
| [ SCHEMABINDING ]
| [ RETURNS NULL ON NULL INPUT | CALLED ON NULL INPUT ]
| [ EXECUTE_AS_Clause ]
}
[ ENCRYPTION ]
| [ SCHEMABINDING ]
```

```
| [ RETURNS NULL ON NULL INPUT | CALLED ON NULL INPUT ]  
| [ EXECUTE_AS_Clause ]  
}
```

Объявление локальных переменных осуществляется следующим образом:

```
DECLARE { @local_variable [AS] data_type }
```

Синтаксис оператора IF-ELSE следующий:

```
IF Boolean_expression  
    { sql_statement | statement_block }  
[ ELSE  
    { sql_statement | statement_block } ]
```

Проверка выражения на NULL-значения осуществляется следующим образом:

```
expression IS [ NOT ] NULL
```

Вызовите созданную скалярную функцию и проверьте корректность её работы.

Скалярная функция может быть указана в любом месте вместо скалярного выражения, в том числе в вычисляемых столбцах и определениях ограничений CHECK.

Кроме того, скалярная функция может быть выполнена инструкцией EXECUTE. Скалярные функции должны вызываться с помощью как минимум двусоставного имени.

Задание №2

Просмотрите информацию о созданной вами функции, выбрав соответствующие поля из системных таблиц:

```
SELECT name, type, type_desc, definition  
FROM sys.sql_modules AS m
```

```
JOIN sys.objects AS o ON m.object_id = o.object_id
AND type IN ('FN', 'IF', 'TF');
```

Задание №3

Измените созданную вами функцию при помощи инструкции ALTER FUNCTION (синтаксис которой аналогичен синтаксису CREATE FUNCTION), таким образом, чтобы её определение было зашифрованным. Повторно просмотрите информацию о созданной функции.

Задание №4

Удалите созданную вами функцию, используя следующую конструкцию:

```
DROP FUNCTION { [ schema_name. ] function_name } [ ,...n ]
```

Возвращающие табличное значение функции

Задание №5

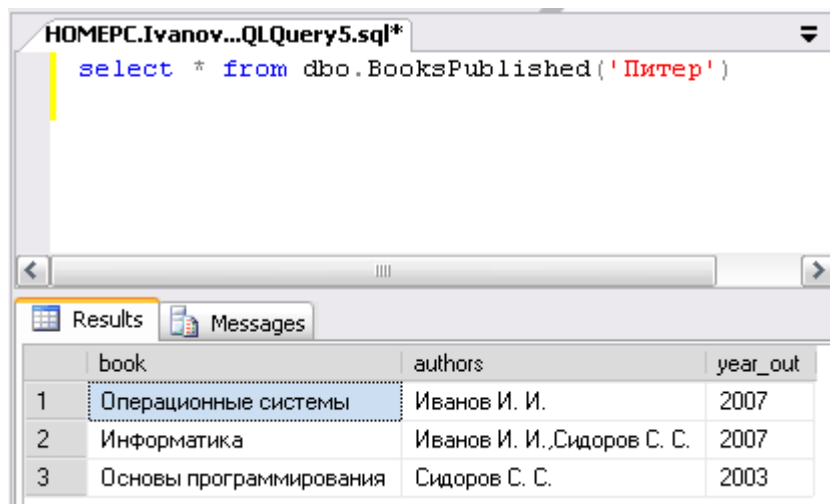
Создайте inline-функцию, принимающую параметром название издательства и возвращающую табличное значение, содержащее книги, изданные в данном издательстве (Название книги, Фамилии авторов через запятую; Год издания книги).

Синтаксис создания inline-функции, возвращающей табличное значение, следующий:

```
CREATE FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ] [ type_schema_name. ]
parameter_data_type
    [ = default ] }
  [ ,...n ]
] )
RETURNS TABLE
[ WITH <function_option> [ ,...n ] ]
[ AS ]
RETURN [ ( ) select_stmt [ ) ]
```

[;]

Протестируйте работу созданной функции, как показано на рис. 4.2:



The screenshot shows a SQL Server Enterprise Manager window titled 'HOMEPC.Ivanov...QLQuery5.sql*'. The query editor contains the SQL statement: `select * from dbo.BooksPublished('Питер')`. Below the editor, the 'Results' tab is active, displaying a table with three columns: 'book', 'authors', and 'year_out'. The table contains three rows of data.

	book	authors	year_out
1	Операционные системы	Иванов И. И.	2007
2	Информатика	Иванов И. И., Сидоров С. С.	2007
3	Основы программирования	Сидоров С. С.	2003

Рис. 4.2. Тестирование созданной функции

Задание №6

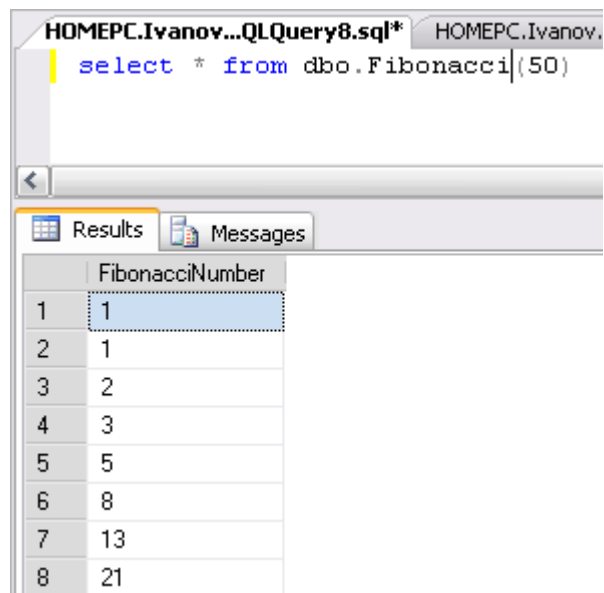
Создайте функцию, возвращающую табличное значение, принимающую параметром число возвращаемых записей (RecordCount), и возвращающую RecordCount первых чисел из последовательности Фибоначчи.

Синтаксис создания функции, возвращающей табличное значение, следующий:

```
CREATE FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ] [ type_schema_name. ]
parameter_data_type
    [ = default ] }
  [ ,...n ]
] )
RETURNS @return_variable TABLE < table_type_definition >
[ WITH <function_option> [ ,...n ] ]
[ AS ]
BEGIN
    function_body
RETURN
```


END [;]

Протестируйте работу созданной функции, как показано на рис. 4.3:



The screenshot shows a SQL query window titled 'HOMEPC.Ivanov...QLQuery8.sql*' with the query: `select * from dbo.Fibonacci(50)`. Below the query window, the 'Results' tab is active, displaying a table with two columns: an implicit index column and 'FibonacciNumber'. The table contains 8 rows of data, with the first row (index 1, value 1) highlighted.

	FibonacciNumber
1	1
2	1
3	2
4	3
5	5
6	8
7	13
8	21

Рис. 4.3. Тестирование созданной функции

Вариант 2

Схема базы данных состоит из трех отношений:

Гостиница (id_hotel int, name varchar(20), address varchar(50))

Организация (id_organization int, name varchar(50))

Участники_форума (id_participant int, fio varchar(50), id_organization int, id_hotel int)

Гостиница

id_hotel	name	address
1	Астория	ул. Новолесная, строение 5
2	Азимут	ул. Заводская, строение 138
3	Флоренция	ул. Зеленая, дом 46
4	Сафари	бульвар Солнечный, строение 8

Организация

id_organization	name
1	Астраханский государственный технический университет
2	Астраханский государственный университет
3	Саратовская академия права
4	Волгоградская медицинская академия
5	Московский государственный университет
6	Архангельский государственный университет

Участники_форума

id_participant	fio	id_organization	id_hotel
1	Иванов Иван Иванович	1	2
2	Петров Петр Петрович	1	1
3	Сидоров Сидор Сидорович	1	1
4	Александров Александр	2	2
5	Михайлов Михаил Михайлович	3	1
6	Юрьев Юрий Юрьевич	4	3
7	Андреев Андрей Андреевич	4	4

8	Степанов Степан Степанович	5	3
9	Максимов Максим Максимович	6	2

Схема базы данных представлена на рис. 4.4.

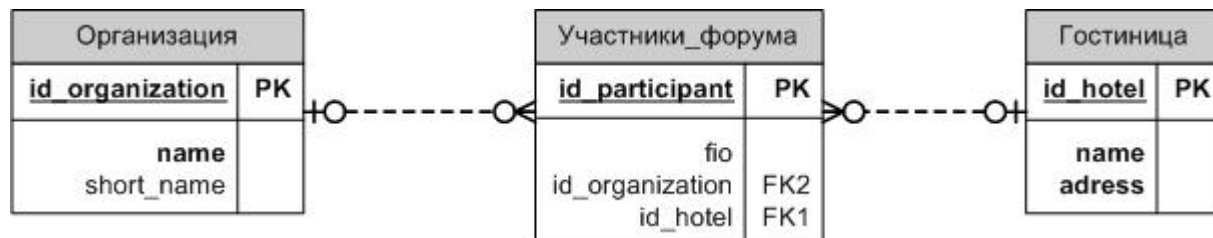


Рис. 4.4. Схема базы данных

Скалярные функции

Задание №1

Создать функцию, принимающую в качестве параметра название гостиницы и возвращающую количество заселенных в ней участников форума. Если в гостинице не остановился ни один человек, функция должна вернуть 0. Если такой гостиницы не существует в базе данных, функция должна вернуть -1.

Создание функции осуществляются при помощи следующей конструкции:

```

CREATE FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ] [ type_schema_name. ]
parameter_data_type
    [ = default ] }
    [ ,...n ]
] )
RETURNS return_data_type
[ WITH <function_option> [ ,...n ] ]
[ AS ]
BEGIN

    function_body

```

```

        RETURN scalar_expression
    END
[ ; ]
<function_option>::=
{
    [ ENCRYPTION ]
    | [ SCHEMABINDING ]
    | [ RETURNS NULL ON NULL INPUT | CALLED ON NULL INPUT ]
    | [ EXECUTE_AS_Clause ] }

```

Объявление локальных переменных осуществляется следующим образом:

```

DECLARE { @local_variable [AS] data_type }

```

Синтаксис оператора IF-ELSE следующий:

```

IF Boolean_expression
    { sql_statement | statement_block }
[ ELSE
    { sql_statement | statement_block } ]

```

Проверка выражения на NULL-значения осуществляется следующим образом:

```

expression IS [ NOT ] NULL

```

Вызовите созданную скалярную функцию и проверьте корректность её работы.

Скалярная функция может быть указана в любом месте вместо скалярного выражения, в том числе в вычисляемых столбцах и определениях ограничений CHECK.

Кроме того, скалярная функция может быть выполнена инструкцией EXECUTE. Скалярные функции должны вызываться с помощью как минимум двусоставного имени.

Задание №2

Просмотрите информацию о созданной вами функции, выбрав соответствующие поля из системных таблиц:

```
SELECT name, type, type_desc, definition
FROM sys.sql_modules AS m
JOIN sys.objects AS o ON m.object_id = o.object_id
AND type IN ('FN', 'IF', 'TF');
```

Задание №3

Измените созданную вами функцию при помощи инструкции ALTER FUNCTION (синтаксис которой аналогичен синтаксису CREATE FUNCTION), таким образом, чтобы её определение было зашифрованным.

Повторно просмотрите информацию о созданной функции.

Задание №4

Удалите созданную вами функцию, используя следующую конструкцию:

```
DROP FUNCTION { [ schema_name. ] function_name } [ ,...n ]
```

Возвращающие табличное значение функции

Задание №5

Создайте inline-функцию, принимающую параметром название организации и возвращающую табличное значение, содержащее список гостиниц, в которых остановились сотрудники данной организации (Название гостиницы; Фамилии сотрудников организации через запятую).

Синтаксис создания inline-функции, возвращающей табличное значение, следующий:

```
CREATE FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ] [ type_schema_name. ]
parameter_data_type
[ = default ] }
```

```

[ ,...n ]
]
)
RETURNS TABLE
[ WITH <function_option> [ ,...n ] ]
[ AS ]
RETURN [ ( ) select_stmt [ ) ]
[ ; ]

```

Протестируйте работу созданной функции, как показано на рис. 4.5:

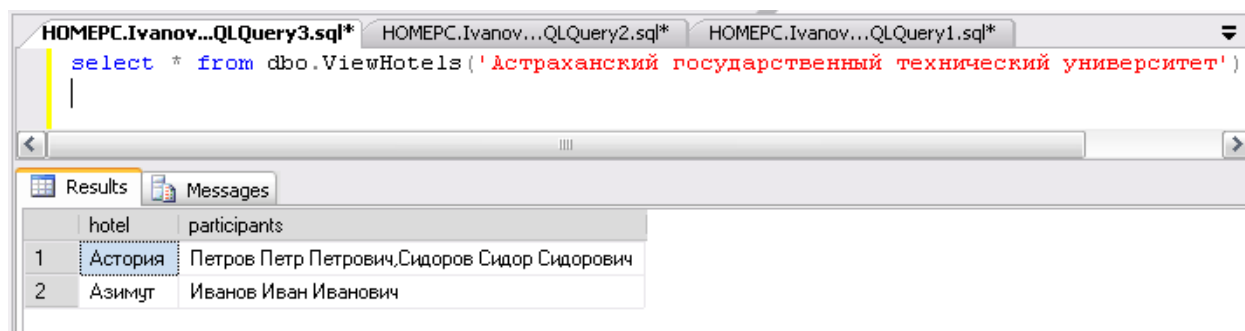


Рис.4.5. Тестирование созданной функции

Задание №6

Создайте функцию, возвращающую табличное значение, принимающую параметром число возвращаемых записей (RecordCount), и возвращающую факториалы первых RecordCount чисел.

Синтаксис создания функции, возвращающей табличное значение, следующий:

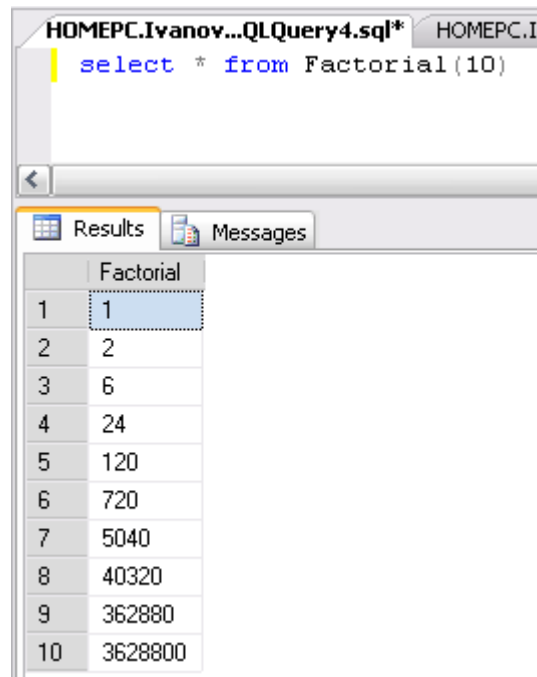
```

CREATE FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ] [ type_schema_name. ]
parameter_data_type
[ = default ] }
[ ,...n ]
]
)
RETURNS @return_variable TABLE < table_type_definition >
[ WITH <function_option> [ ,...n ] ]
[ AS ]

```

```
BEGIN  
  
        function_body  
  
    RETURN  
  
END  
  
[ ; ]
```

Протестируйте работу созданной функции, как показано на рис. 4.6:



The screenshot shows a SQL query window titled 'HOMEPC.Ivanov...QLQuery4.sql*' with the query 'select * from Factorial(10)'. Below the query, the 'Results' tab is active, displaying a table with two columns: an index and the 'Factorial' value. The table contains 10 rows of data, with the first row (index 1, value 1) highlighted.

	Factorial
1	1
2	2
3	6
4	24
5	120
6	720
7	5040
8	40320
9	362880
10	3628800

Рис.4.6. Тестирование созданной функции

Лабораторная работа №5

«Программируемые объекты SQL Server. Хранимые процедуры, триггеры»

Вариант 1

Схема базы данных состоит из четырех отношений:

- *Кассир*
- *Продукт*
- *Клиент*
- *Продажа*

Схема базы данных представлена на рис. 5.1.

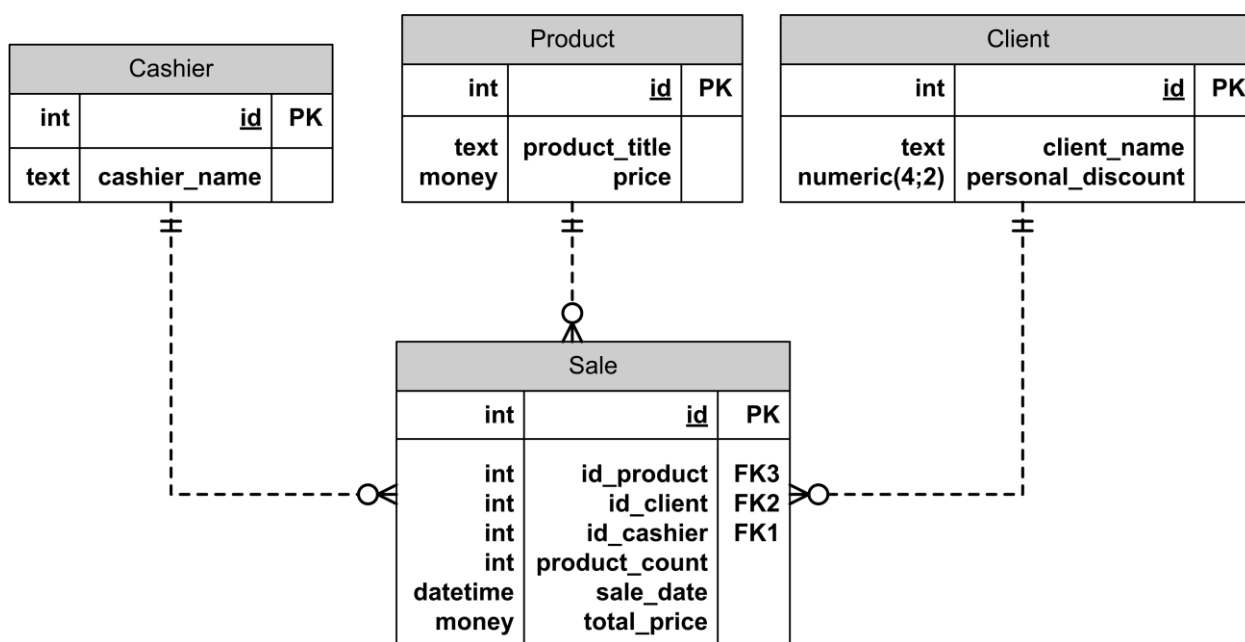


Рис. 5.1. Схема базы данных

Задание № 1

Выполните sql-скрипт для создания таблиц в базе данных и вставки в них значений.

Задание №2

Создайте хранимую процедуру, которая устанавливает базовый уровень скидок для клиентов следующим образом:

- если клиент совершил не менее 5 покупок - 3%
- если сумма покупок превышает 50 000 - 5%

Выбирается максимальная скидка.

Синтаксис создания хранимой процедуры следующий:

```
CREATE { PROC | PROCEDURE } [schema_name.] procedure_name [ ;
number ]
    [ { @parameter [ type_schema_name. ] data_type }
      [ VARYING ] [ = default ] [ OUT | OUTPUT ]
    ] [ ,...n ]
[ WITH <procedure_option> [ ,...n ] ]
[ FOR REPLICATION ]
AS { <sql_statement> [;][ ...n ] | <method_specifier> }
[;]
<procedure_option> ::=
    [ ENCRYPTION ]
    [ RECOMPILE ]
    [ EXECUTE_AS-Clause ]
<sql_statement> ::=
{ [ BEGIN ] statements [ END ] }
<method_specifier> ::=
EXTERNAL NAME assembly_name.class_name.method_name
```

Задание №3

Создайте хранимые процедуры для редактирования (добавление, удаление, изменение) значений всех справочников.

Задание №4

Создайте хранимую процедуру, которая возвращает размер скидки клиента на текущий момент:

- базовую скидку

- скидку выходного дня - 3%
 - скидку за объем покупки (при покупке на сумму более 100 000) - 1%
- Скидки суммируются.

Задание №5

Создайте триггер на изменение таблицы продаж (на вставку) для расчета общей суммы продажи на текущий момент.

Синтаксис создания DML-триггера следующий:

```
CREATE TRIGGER [ schema_name . ]trigger_name
ON { table | view }
[ WITH <dml_trigger_option> [ ,...n ] ]
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
[ WITH APPEND ]
[ NOT FOR REPLICATION ]
AS { sql_statement [ ; ] [ ,...n ] | EXTERNAL NAME <method
specifier [ ; ] > }
<dml_trigger_option> ::=
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]
<method_specifier> ::=
    assembly_name.class_name.method_name
```

Задание №6

Создайте триггер на удаление строки из таблицы продаж (на удаление), который обеспечит пересчёт базового уровня скидки клиента.

Задание №7

Создайте хранимую процедуру, которая принимает в качестве параметра id клиента и возвращает нарастающим итогом (дата, общая сумма) информацию обо всех покупках клиента.

Задание №8

Создайте хранимую процедуру, возвращающую значение бонуса кассиру (id кассира) по месяцам, в которых у него были продажи.

Размер бонуса составляет:

- 0.1% от суммы продажи, если $10\,000 > \text{сделка} > 0$
- 0.2% от суммы продажи, если $50\,000 > \text{сделка} > 10\,000$
- 0.3% от суммы продажи, если $100\,000 > \text{сделка} > 50\,000$

Задание №9

Создайте триггер, который запрещает удаление созданных вами пользовательских таблиц.

Синтаксис создания DDL-триггера следующий:

```
CREATE TRIGGER trigger_name
ON { ALL SERVER | DATABASE }
[ WITH <ddl_trigger_option> [ ,...n ] ]
{ FOR | AFTER } { event_type | event_group } [ ,...n ]
AS { sql_statement [ ; ] [ ,...n ] | EXTERNAL NAME < method
specifier > [ ; ] }
<ddl_trigger_option> ::=
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]
<method_specifier> ::=
    assembly_name.class_name.method_name
```

Вариант 2

Схема базы данных состоит из четырех отношений:

- *Компьютер*
- *Программа*
- *Операционная система*
- *Установка программы на компьютер*

Схема базы данных представлена на рис.5.2.

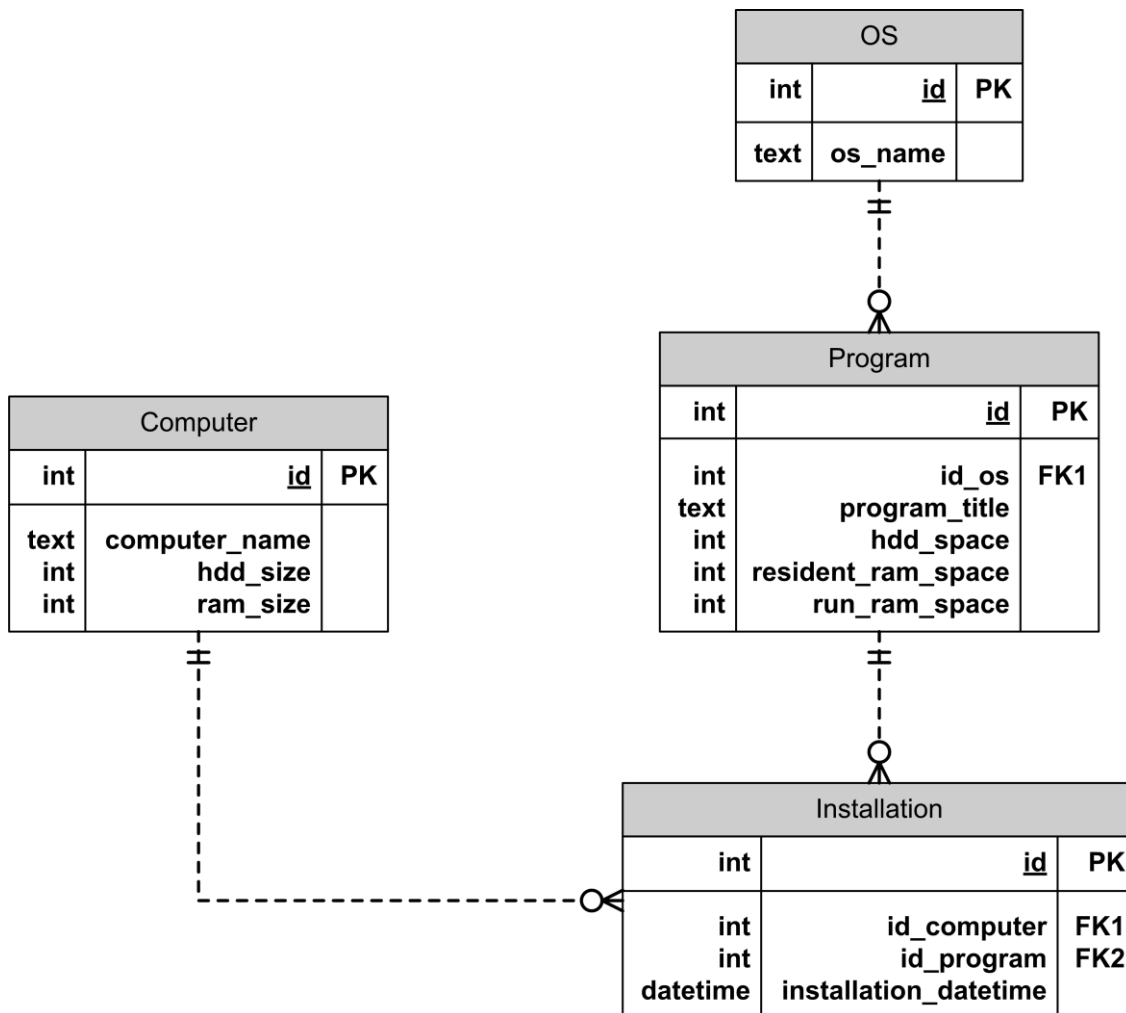


Рис. 5.2. Схема базы данных

Задание № 1

Выполните sql-скрипт для создания таблиц в базе данных и вставки в них значений.

Задание №2

Создайте хранимую процедуру, которая осуществляет проверку и исправление ошибок следующим образом:

- первая установленная по времени на компьютер программа считается операционной системой
- следующие установленные программы должны быть предназначены для этой операционной системы

- для установки программы на компьютере должно быть достаточное количество места на жестком диске и в оперативной памяти.

Хранимая процедура должна удалить с каждого компьютера программы, не соответствующие требованиям операционной системы и минимальным системным требованиям.

Синтаксис создания хранимой процедуры следующий:

```
CREATE { PROC | PROCEDURE } [schema_name.] procedure_name [ ;
number ]
    [ { @parameter [ type_schema_name. ] data_type }
        [ VARYING ] [ = default ] [ OUT | OUTPUT ]
    ] [ ,...n ]
[ WITH <procedure_option> [ ,...n ] ]
[ FOR REPLICATION ]
AS { <sql_statement> [;][ ...n ] | <method_specifier> }
[;]
<procedure_option> ::=
    [ ENCRYPTION ]
    [ RECOMPILE ]
    [ EXECUTE_AS_Clause ]
<sql_statement> ::=
{ [ BEGIN ] statements [ END ] }
<method_specifier> ::=
EXTERNAL NAME assembly_name.class_name.method_name
```

Задание №3

Создайте хранимые процедуры для редактирования (добавление, удаление, изменение) значений всех справочников.

Задание №4

Создайте хранимую процедуру, которая возвращает размер доступной оперативной памяти, и места на жестком диске на текущий момент времени для заданного компьютера.

Задание №5

Создайте триггер на изменение таблицы установки программы (на вставку) для проверки допустимости установки программы на данный момент времени, с учетом требований операционной системы и минимальных системных требований.

Синтаксис создания DML-триггера следующий:

```
CREATE TRIGGER [ schema_name . ]trigger_name
ON { table | view }
[ WITH <dml_trigger_option> [ ,...n ] ]
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
[ WITH APPEND ]
[ NOT FOR REPLICATION ]
AS { sql_statement [ ; ] [ ,...n ] | EXTERNAL NAME <method
specifier [ ; ] > }
<dml_trigger_option> ::=
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]
<method_specifier> ::=
    assembly_name.class_name.method_name
```

Задание №6

Создайте триггер на удаление строки из таблицы установки программ (на удаление), который обеспечит в случае удаления операционной системы с компьютера удаление всех остальных программ с данного компьютера.

Задание №7

Создайте хранимую процедуру, которая принимает в качестве параметра id компьютера и возвращает убывающим итогом (дата и время, размер свободной оперативной памяти и свободное места на жестком диске) информацию обо всех программах, установленных на компьютере.

Задание №8

Создайте хранимую процедуру, позволяющую по id компьютера определить, какие программы на него можно установить с учетом установленной операционной системы и минимальными системными требованиями.

Задание №9

Создайте триггер, который запрещает удаление созданных вами пользовательских таблиц.

Синтаксис создания DDL-триггера следующий:

```
CREATE TRIGGER trigger_name
ON { ALL SERVER | DATABASE }
[ WITH <ddl_trigger_option> [ ,...n ] ]
{ FOR | AFTER } { event_type | event_group } [ ,...n ]
AS { sql_statement [ ; ] [ ,...n ] | EXTERNAL NAME < method
specifier > [ ; ] }
<ddl_trigger_option> ::=
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]
<method_specifier> ::=
    assembly_name.class_name.method_name
```

Лабораторная работа №6

«Создание индексов. Индексированные представления»

Вариант 1

Схема базы данных состоит из четырех отношений:

- *Кассир*
- *Продукт*
- *Клиент*
- *Продажа*

Схема базы данных представлена на рис.6.1.

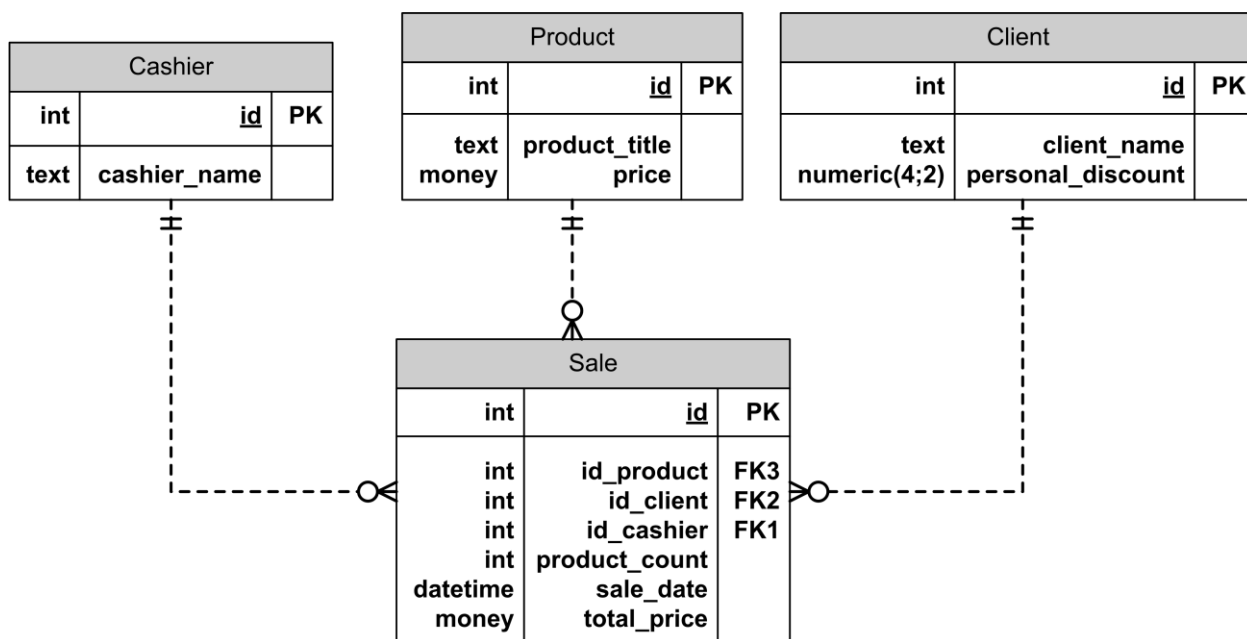


Рис. 6.1. Схема базы данных

Задание № 1

Заполните вашу базу данных тестовыми значениями одним из способов:

- выполнение sql-скрипта;

- восстановление из резервной копии;
- подключение файла базы данных.

Обоснуйте причину вашего выбора.

Задание №2

Создайте следующие SQL-запросы:

Запрос 1

Определите, сколько продаж осуществил каждый кассир.

Вывод: имя кассира, число продаж.

Запрос 2

Определите число продаж, осуществленных после 1 января 2005 г.

Вывод: число продаж.

Запрос 3

Определите первую десятку самых продаваемых (по количеству – product_count) товаров.

Вывод: наименование товара, общий проданный объем.

При помощи SQL Server Profiler оцените среднюю (по результатам 7 запусков) производительность ваших запросов.

Оцениваются параметры CPU, Reads, Duration.

Результаты измерений занесите в таблицу 6.1:

Таблица 6.1

Запрос	CPU	Reads	Duration

Просмотрите планы выполнения ваших запросов, выпишите операции, которые осуществляются при их выполнении.

Задание №3

Создайте следующие некластеризованные индексы для ускорения выполнения SQL-запросов для таблицы Sale:

1. Индекс: [id_product] ASC INCLUDE ([product_count])
2. Индекс: [sale_date] ASC
3. Индекс: [id_cashier] ASC

Синтаксис создания индексов следующий:

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ] INDEX index_name
    ON <object> ( column [ ASC | DESC ] [ ,...n ] )
    [ INCLUDE ( column_name [ ,...n ] ) ]
    [ WITH ( <relational_index_option> [ ,...n ] ) ]
    [ ON { partition_scheme_name ( column_name )
        | filegroup_name
        | default
        }
    ]
[ ; ]
```

<object> ::=

```
{
    [ database_name. [ schema_name ] . | schema_name. ]
    table_or_view_name
}
```

<relational_index_option> ::=

```
{
    PAD_INDEX = { ON | OFF }
    | FILLFACTOR = fillfactor
    | SORT_IN_TEMPDB = { ON | OFF }
    | IGNORE_DUP_KEY = { ON | OFF }
    | STATISTICS_NORECOMPUTE = { ON | OFF }
    | DROP_EXISTING = { ON | OFF }
    | ONLINE = { ON | OFF }
```

```
| ALLOW_ROW_LOCKS = { ON | OFF }  
| ALLOW_PAGE_LOCKS = { ON | OFF }  
| MAXDOP = max_degree_of_parallelism  
}
```

Создайте отсутствующие статистики для полей задействованных в индексах, при помощи следующей системной процедуры:

```
EXEC sp_createstats 'indexonly';
```

Обновите статистики при помощи следующей системной процедуры:

```
EXEC sp_updatestats;
```

Повторно оцените производительность и занесите результаты измерений в таблицу.

Просмотрите планы выполнения ваших запросов, выпишите операции, которые осуществляются при их выполнении.

Задание №4

Создайте представления с привязкой к схеме на основе следующих запросов:

Представление 1

```
SELECT id_product, SUM(product_count), count_big(*)  
FROM Sale GROUP BY id_product
```

Создайте для данного представления кластеризованный индекс по первому столбцу.

Создайте для данного представления некластеризованный индекс по второму столбцу.

Представление 2

```
SELECT id_cashier, count_big(*)  
FROM Sale
```

```
GROUP BY id_cashier
```

Создайте для данного представления кластеризованный индекс по первому столбцу.

Важно! Если вы используете редакцию SQL Server отличную от Enterprise/Developer, вам необходимо переписать ваши запросы, чтобы они использовали созданное вами индексированное представление.

Синтаксис создания представлений следующий:

```
CREATE VIEW [ schema_name . ] view_name [ (column [ ,...n ] ) ]  
[ WITH <view_attribute> [ ,...n ] ]  
AS select_statement  
[ WITH CHECK OPTION ] [ ; ]  
<view_attribute> ::=  
{  
    [ ENCRYPTION ]  
    [ SCHEMABINDING ]  
    [ VIEW_METADATA ]      }
```

Повторно оцените производительность и занесите результаты измерений в таблицу. Сведите полученные результаты в одну таблицу и постройте графики, с целью наглядного сравнения полученных результатов.

Просмотрите планы выполнения ваших запросов, выпишите операции, которые осуществляются при их выполнении. Объясните, за счет чего увеличивается производительность запросов на каждом из этапов оптимизации.

Вариант 2

Схема базы данных состоит из четырех отношений:

- *Компьютер*
- *Программа*
- *Операционная система*
- *Установка программы на компьютер*

Схема базы данных представлена на рис.6.2.

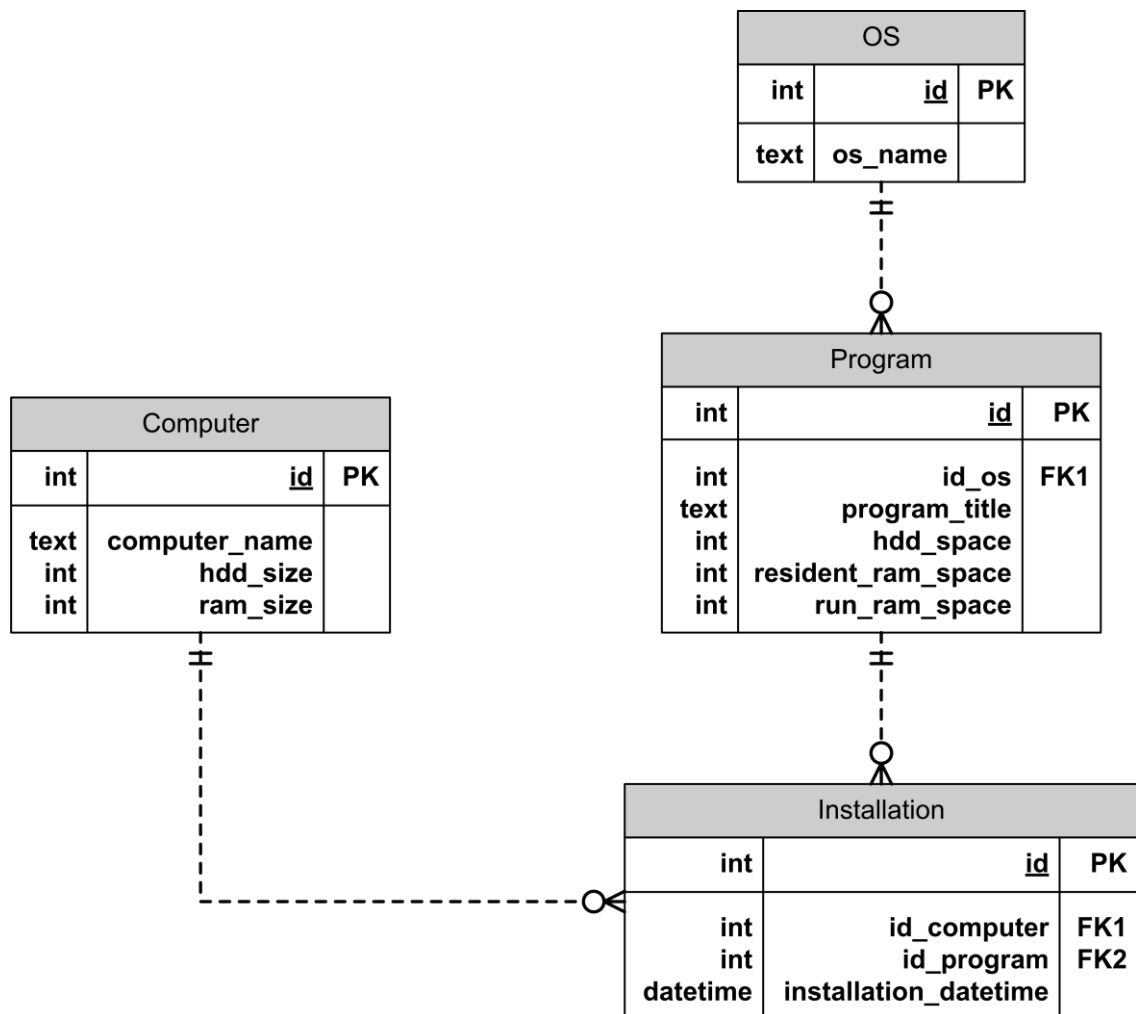


Рис. 6.2. Схема базы данных

Задание № 1

Заполните вашу базу данных тестовыми значениями одним из способов:

- выполнение sql-скрипта;

- восстановление из резервной копии;
- подключение файла базы данных.

Обоснуйте причину вашего выбора.

Задание №2

Создайте следующие SQL-запросы:

Запрос 1

Определите, сколько программ установлено на каждый компьютер

Вывод: имя компьютера, число установленных программ.

Запрос 2

Определите число установок программ, осуществленных после 1 января 2006 года.

Вывод: число установок программ.

Запрос 3

Определите первую десятку самых загруженных компьютеров (по параметру resident_ram_space).

Вывод: имя компьютера, занято оперативной памяти под резидентные программы.

При помощи SQL Server Profiler оцените среднюю (по результатам 7-запусков) производительность ваших запросов. Оцениваются параметры CPU, Reads, Duration.

Результаты измерений занесите в таблицу 6.2:

Таблица 6.2

Запрос	CPU	Reads	Duration

Просмотрите планы выполнения ваших запросов, выпишите операции, которые осуществляются при их выполнении.

Задание №3

Создайте следующие некластеризованные индексы для ускорения выполнения SQL-запросов для таблицы Installation:

1. **Индекс:** [id_program] ASC, [id_computer] ASC
2. **Индекс:** [id_computer] ASC
3. **Индекс:** [installation_datetime] ASC

Для таблицы Computer

Индекс: [id] ASC INCLUDE ([computer_name])

Для таблицы Program

Индекс: [id] ASC INCLUDE ([resident_ram_space])

Синтаксис создания индексов следующий:

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ] INDEX index_name
    ON <object> ( column [ ASC | DESC ] [ ,...n ] )
    [ INCLUDE ( column_name [ ,...n ] ) ]
    [ WITH ( <relational_index_option> [ ,...n ] ) ]
    [ ON { partition_scheme_name ( column_name )
        | filegroup_name
        | default
        }
    ] [ ; ]
<object> ::=
{      [ database_name. [ schema_name ] . | schema_name. ]
      table_or_view_name  }
<relational_index_option> ::=
{
    PAD_INDEX    = { ON | OFF }
  | FILLFACTOR = fillfactor
  | SORT_IN_TEMPDB = { ON | OFF }
  | IGNORE_DUP_KEY = { ON | OFF }
  | STATISTICS_NORECOMPUTE = { ON | OFF }
```

```

| DROP_EXISTING = { ON | OFF }
| ONLINE = { ON | OFF }
| ALLOW_ROW_LOCKS = { ON | OFF }
| ALLOW_PAGE_LOCKS = { ON | OFF }
| MAXDOP = max_degree_of_parallelism
}

```

Создайте отсутствующие статистики для полей задействованных в индексах, при помощи следующей системной процедуры:

```
EXEC sp_createstats 'indexonly';
```

Обновите статистики при помощи следующей системной процедуры:

```
EXEC sp_updatestats;
```

Повторно оцените производительность и занесите результаты измерений в таблицу.

Просмотрите планы выполнения ваших запросов, выпишите операции, которые осуществляются при их выполнении.

Задание №4

Создайте представления с привязкой к схеме на основе следующих запросов:

Представление 1

```

SELECT [Installation].[id_computer],
SUM([Program].[resident_ram_space]), count_big(*)
FROM [Program], [Installation]
WHERE [Program].[id] = [Installation].[id_program]
GROUP BY [Installation].[id_computer]

```

Создайте для данного представления кластеризованный индекс по первому столбцу.

Создайте для данного представления некластеризованный индекс по второму столбцу.

Представление 2

```
SELECT  [Installation].[id_computer],  count_big(*)
FROM    [Installation]
GROUP BY  [Installation].[id_computer]
```

Создайте для данного представления кластеризованный индекс по первому столбцу.

Важно! Если вы используете редакцию *SQL Server* отличную от *Enterprise/Developer*, вам необходимо переписать ваши запросы, чтобы они использовали созданное вами индексированное представление.

Синтаксис создания представлений следующий:

```
CREATE VIEW [ schema_name . ] view_name [ (column [ ,...n ] ) ]
[ WITH <view_attribute> [ ,...n ] ]
AS select_statement
[ WITH CHECK OPTION ] [ ; ]
```

```
<view_attribute> ::=
{
    [ ENCRYPTION ]
    [ SCHEMABINDING ]
    [ VIEW_METADATA ] }
```

Повторно оцените производительность и занесите результаты измерений в таблицу.

Сведите полученные результаты в одну таблицу и постройте графики, с целью наглядного сравнения полученных результатов.

Просмотрите планы выполнения ваших запросов, выпишите операции, которые осуществляются при их выполнении.

Объясните, за счет чего увеличивается производительность запросов на каждом из этапов оптимизации.

Лабораторная работа № 7

«Система безопасности SQL Server»

Создайте базу данных соответствующую следующей диаграмме (в скобках указано имя схемы), как показано на рис.7.1:

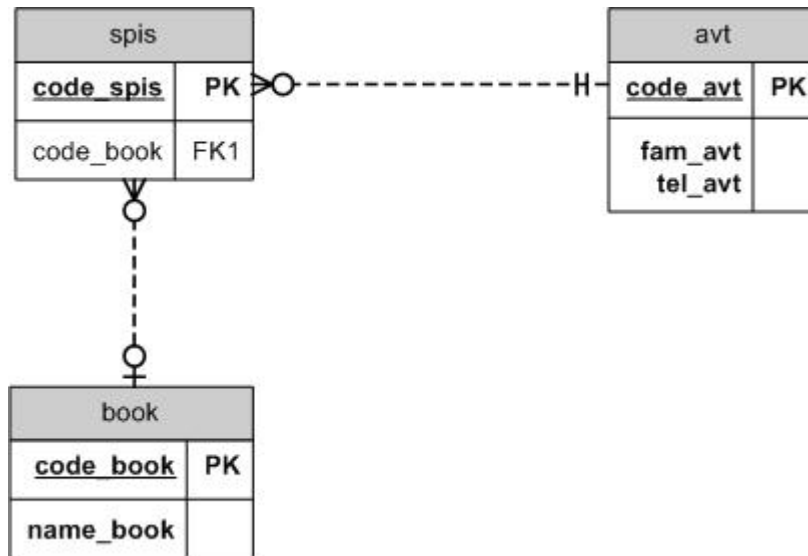


Рис.7.1. Диаграмма базы данных

Задание №1

Чтобы определить роли сервера и группы Windows в которых состоит в данное время подключенный пользователь выполните следующий запрос.

```
USE master
go
SELECT * FROM sys.login_token
```

Задание №2

Просмотрите информацию о текущей учетной записи используя процедуру sp_helplogins:

```
EXEC sp_helplogins [ [ @@LoginNamePattern =] 'login ']
```

Задание №3

Используя конструкцию CREATE LOGIN создайте:

- учетную запись SQL Server (login='login', password='password\$123456789');
- учетную запись SQL Server сопоставленную учетным данным, укажите при этом базу данных по умолчанию Lab (login = 'login2', password = 'password\$123456789');
- учетную запись SQL Server на основе пользовательской группы Windows.

```
CREATE LOGIN login {WITH <option_list1>|FROM <sources>}
```

```
<sources>::=
```

```
WINDOWS [ WITH <windows_options>[ ,...n ] ]
```

```
|CERTIFICATE certificate_name
```

```
|ASYMMETRIC KEY asym_key_name
```

```
<option_list1>::=
```

```
PASSWORD = 'password' [ HASHED ] [ MUST__CHANGE ]
```

```
[ ,<option_list2>[ ,...n ] ]
```

```
<option_list2>::=
```

```
SID =sid
```

```
|DEFAULT_DATABASE =database_name
```

```
|DEFAULT_LANGUAGE =language_id
```

```
|CHECK_EXPIRATION ={ON |OFF }
```

```
|CHECK_POLICY ={ON |OFF }
```

```
[ CREDENTIAL =credential_name ]
```

```
<windows_options>::=
```

```
DEFAULT_DATABASE =database_name
```

```
|DEFAULT_LANGUAGE =language_id
```

Установите смешанный тип аутентификации для экземпляра сервера.

Перезапустите сервер.

1. Войдите в экземпляр сервера, используя первую учетную запись.

Выполните следующие два запроса:

```
select * from [master].dbo.spt_monitor
```

```
select * from [Lab].dbo.book
```

Сделайте выводы.

2. Войдите в экземпляр сервера, используя вторую учетную запись.

Проделайте аналогичные манипуляции.

Создайте учетные записи-двойники (в конце имени записи удвоить последнюю букву или цифру) при помощи SQL Server Management Studio. Сохранить в файл-отчет изображение окон при создании учетных записей.

Задание №4

Используя конструкцию ALTER LOGIN измените пароль первой учетной записи.

```
ALTER LOGIN login
{<status_option>
|WITH <set_option>[ ,,...n ]
}
<status_option>::=
ENABLE |DISABLE
<set_option>::=
PASSWORD ='password '
[ OLD_PASSWORD ='old_password '
|<secadmin_pwd_option>[ <<secadmin_pwd_option>]
]
|DEFAULT_DATABASE =database_name
|DEFAULT_LANGUAGE =language_id
|NAME =login
|CHECK_POLICY ={ON |OFF }
|CHECK_EXPIRATION ={ON |OFF }
|CREDENTIAL =credential_name
|NO CREDENTIAL
<secadmin_pwd_opt>::=
MUST_CHANGE |UNLOCK
```

Задание №5

Пользуясь процедурами `sp_grantlogin` и `sp_denylogin` запретите и разрешите вновь доступ к серверу для учетной записи с именем 'login3'.

Задание №6

Удалите учетную запись основанную на пользовательской группе Windows, используя процедуру `sp_revokelogin`.

```
sp_revokelogin [ @@loginame =] 'login '
```

Удалите учетную запись с именем login2 используя инструкцию DROP LOGIN.

```
DROP LOGIN login
```

Удалите учетные записи-двойники используя SQL Server Management Studio.

Задание №7

Используя процедуру `sp_addsrvrolemember`, добавьте учетную запись с именем 'login' в роль сервера 'sysadmin'.

```
sp_addsrvrolemember [ @@loginame =] 'login ',[ @@rolename =]  
'role_name '
```

И вновь войдите в экземпляр сервера под учетной записью login и выполните следующий запрос:

```
select * from [Lab].dbo.book
```

Отметьте изменения, произошедшие с момента выполнения задания №3. Удалите запись с именем 'login' из роли сервера 'sysadmin' и добавьте ее в роль сервера 'dbcreator'.

```
sp_dropsrvrolemember [ @@loginame =] 'login ',[ @@rolename =]  
'role_name '
```

Прodelайте описанную выше процедуру с конструкцией SELECT еще раз. Зафиксируйте изменения.

Удалите базу данных от имени администратора и пересоздайте от имени login.

Создайте учетные записи с параметрами ('lab_admin', 'password\$123456789'), ('ba_admin', 'password\$123456789'). Добавьте учетные записи в роль сервера 'sysadmin', используя SQL Server Management Studio. Сохраните изображение окна свойств учетной записи в файл-отчет.

Задание №8

Используя функцию sp_helprole, просмотрите информацию о ролях уровня базы данных таких как 'Db_backupoperator', 'Db_datereader', 'Db_datawriter'.

```
sp_helprole [ [ @@rolename =] 'role_name ']
```

Задание №9

Войдите учетной записью с именем 'login'. Назначьте данной учетной записи права db_datareader.

Лабораторная работа №8

«Система безопасности SQL Server (уровень баз данных)»

Отчет по каждому заданию лабораторной работы состоит из двух частей:

- набора SQL-команд;
- скриншотов окон SQL Management Studio.

Выполните приведенный ниже SQL-скрипт для создания таблиц в базе данных и вставки в них значений:

```
USE master
go
CREATE DATABASE Lab          ON PRIMARY
(NAME=Lab, FILENAME='c:\Program Files\Microsoft SQL
Server\MSSQL\Data\Lab_Data.MDF', SIZE=3MB, MAXSIZE=UNLIMITED,
FILEGROWTH=10%) LOG ON (NAME=Lab_Data,
FILENAME= 'c:\Program Files\Microsoft SQL
Server\MSSQL\Data\Lab_Data.LDF',
    SIZE=3MB, MAXSIZE=UNLIMITED, FILEGROWTH=10%)
go
USE Lab
go
create schema F_S
go
create schema S_S
go
CREATE TABLE [Lab].[F_S].[book] (
    [code_book] [INTEGER] PRIMARY KEY CLUSTERED identity,
    [name_book] [VARCHAR] (40) NOT NULL,
) go
INSERT INTO [Lab].[F_S].[book]
VALUES ('Microsoft Office')
```

```

INSERT INTO [Lab].F_S.[book]
VALUES ('Программирование в Delphi 7')
INSERT INTO [Lab].F_S.[book]
VALUES ('SQL Server 2005') go
CREATE TABLE [Lab].[S_S].[avt] (
    [code_avt][INTEGER] PRIMARY KEY CLUSTERED identity,
    [fam_avt][VARCHAR] (20) NOT NULL,
    [tel_avt][VARCHAR] (20) NOT NULL,
) go
INSERT INTO [Lab].S_S.[avt]
VALUES ('Иванов', '111-11-11')
INSERT INTO [Lab].S_S.[avt]
VALUES ('Петров', '222-22-22')
INSERT INTO [Lab].S_S.[avt]
VALUES ('Сидоров', '444-44-44')
INSERT INTO [Lab].S_S.[avt]
VALUES ('Кузнецов', '555-55-55')
go
CREATE TABLE [Lab].[dbo].[spis] (
    [code_spis][INTEGER] PRIMARY KEY CLUSTERED identity,
    [code_book][INTEGER] foreign key REFERENCES
[Lab].[F_S].[book],
    [code_avt][INTEGER] foreign key REFERENCES
[Lab].[S_S].[avt],
)
go
INSERT INTO [Lab].dbo.[spis]
VALUES (1,1)
INSERT INTO [Lab].dbo.[spis]
VALUES (1,2)
INSERT INTO [Lab].dbo.[spis]
VALUES (2,3)
INSERT INTO [Lab].dbo.[spis]
VALUES (2,1)
INSERT INTO [Lab].dbo.[spis]
VALUES (3,4)

```


Добавьте три учетных записи, ассоциированные с учетными данными (базой Lab):

```
use Lab
go
create login reader with password='pa$$word',
DEFAULT_DATABASE=Lab
go
create login writer with password='pa$$word',
DEFAULT_DATABASE=Lab
go
create login bu_operator with password='pa$$word',
DEFAULT_DATABASE=Lab
```

Задание №1

Создайте по одному пользователю базы данных Lab для каждой учетной записи, используя директиву CREATE USER (u_reader, u_writer, u_bu_operator соответственно).

```
CREATE USER user_name
[ {FOR |FROM }
  {LOGIN login
    |CERTIFICATE certificate_name
    |ASYMMETRIC KEY asym_key_name
  }
]
[ WITH DEFAULT_SCHEMA =schema_name ]
```

Задание №2

С помощью системной процедуры sp_addrolemember определите роли пользователей уровня базы данных:

- для u_reader – роль “db_datareader”
- для u_writer – роль “db_datawriter”
- для u_bu_operator – роль “db_backupoperator”

```
sp_addrolemember [ @@rolename =] 'role_name ', [ @membername =]  
'database_principal '
```

Задание №3

Войдите в экземпляр SQL Server используя учетную запись reader и выполните следующие SQL-команды:

```
SELECT * FROM [Lab].[F_S].[book]  
go  
SELECT * FROM [Lab].[S_S].[avt]  
go  
SELECT * FROM [Lab].[dbo].[spis]
```

Затем попробуйте ставить в таблицы новые записи и зафиксируйте результат:

```
INSERT INTO [Lab].[F_S].[book] VALUES ('Новая книга')  
go  
INSERT INTO [Lab].[S_S].[avt] VALUES ('Новый автор', '000-00-  
00')  
go  
INSERT INTO [Lab].[dbo].[spis] VALUES (1,1)
```

Попробуйте создать резервную копию базы данных, используя следующий SQL-скрипт:

```
BACKUP DATABASE [Lab] TO DISK = 'c:\video\lab_backup.bak'
```

Проделайте аналогичные операции для учетных записей writer и bu_operator. Сделайте выводы.

Задание №4

Добавьте две новых учетных записи (book и avt). Создайте для них по одному пользователю базы данных Lab (u_book и u_avt) соответственно. С помощью команды CREATE ROLE создайте две собственные роли базы данных Lab (booker и authors).

```
CREATE ROLE role_name [ AUTHORIZATION owner_name ]
```

Используя конструкцию предоставления разрешений GRANT, добавьте разрешение выполнять инструкцию SELECT участнику роли booker на таблице book, участнику роли authors на таблице avt, а на таблице spis разрешить выполнять инструкцию SELECT участникам обеих ролей.

```
GRANT
{ALL [ PRIVILEGES ] }
|permission_name [ (column_name [ ,...n ] )] [ ,...n ]
[ ON [class ::]securable ]
TO principal [ ,...n ] [ WITH GRANT OPTION ]
[AS principal ]
```

Определите роли пользователей: для пользователя u_book роль booker, для пользователя u_avt роль authors.

Войдите в экземпляр сервера учетными записями book и avt. Проверьте исполнение набора запросов для каждой учетной записи:

```
SELECT * FROM [Lab].[F_S].[book]
go
SELECT * FROM [Lab].[S_S].[avt]
go
SELECT * FROM [Lab].[dbo].[spis]
```

Задание №5

От имени администратора, создайте представление my_view, включающее в себя поля всех трех таблиц (рис. 8.1).

	name_book	fam_avt	tel_avt
1	Microsoft Office	Иванов	111-11-11
2	Microsoft Office	Петров	222-22-22
3	SQL Server 2005	Кузнецов	555-55-55
4	Программирование в Delphi 7	Иванов	111-11-11
5	Программирование в Delphi 7	Сидоров	444-44-44

Рис. 8.1. Результат выполнения запроса из представления.

Войдите в экземпляр сервера учетными записями book и avt. Проверьте исполнение запроса для каждой учетной записи:

```
select * from my_view
```

От имени администратора добавьте участникам роли booker разрешение выполнять инструкцию SELECT над представлением my_view. Выполните проверку на исполнение предыдущего запроса для каждой учетной записи (book, avt). Зафиксируйте результат.

Задание №6

От имени администратора, создайте хранимую процедуру с именем my_proc (передаваемый параметр – фамилия автора, возвращаемый набор данных – все книги, написанные этим автором).

Добавьте участникам роли authors разрешение исполнять директиву EXECUTE для хранимой процедуры my_proc.

Войдите в экземпляр сервера учетными записями book и avt. Проверьте исполнение нижеследующей SQL-команды для каждой учетной записи:

```
EXECUTE my_proc 'Иванов'
```

Сделайте соответствующие выводы. Подготовьте отчет.

Лабораторная работа №9

«Резервное копирование и восстановление данных»

Схема базы данных состоит из четырех отношений:

- *Компьютер*
- *Программа*
- *Операционная система*
- *Установка программы на компьютер*

Схема базы данных представлена на рис. 9.1.

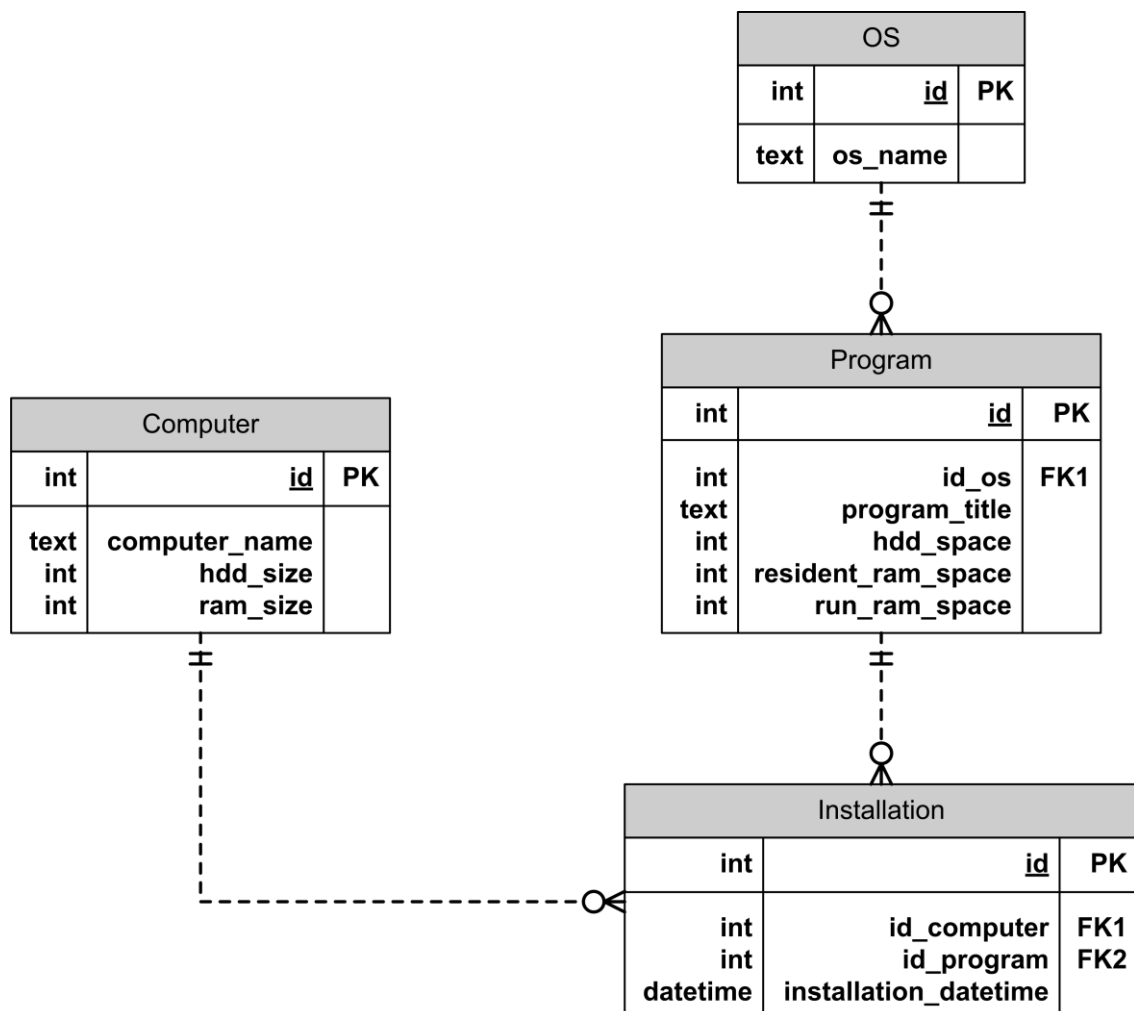


Рис. 9.1. Схема базы данных

Задание № 1

Восстановите базу данных «edu» из прилагаемой к лабораторной работе резервной копии в каталоге «mssql_lab8»:

```
RESTORE DATABASE [eduIvanovII]
FROM DISK = 'путь к файлу\edu.bak'
WITH
    MOVE 'edu' TO 'D:\SQL\edu_IvanovII.mdf',
    MOVE 'edu_log' TO 'D:\SQL\edu_IvanovII.ldf',
    STATS = 5,
    RECOVERY
```

Задание №2

Добавьте в таблицу OS следующие операционные системы: «FreeBSD» и «MacOS»

Задание №3

Выполните **полное** резервное копирование вашей базы данных при помощи команды BACKUP DATABASE (расширение целевого файла .bak):

```
BACKUP DATABASE { database_name | @database_name_var }
    TO <backup_device> [ ,...n ]
    [ <MIRROR TO clause> ] [ next-mirror-to ]
    [ WITH { DIFFERENTIAL | <general_WITH_options> [ ,...n ] } ]
[;]
<backup_device>::=
{
    { logical_device_name | @logical_device_name_var }
    | { DISK | TAPE } =
        { 'physical_device_name' | @physical_device_name_var }
}
<general_WITH_options> [ ,...n ]::=
--Backup Set Options
    COPY_ONLY
    | DESCRIPTION = { 'text' | @text_variable }
    | NAME = { backup_set_name | @backup_set_name_var }
```

```
| PASSWORD = { password | @password_variable }  
| [ EXPIREDATE = { date | @date_var }  
    | RETAINDAYS = { days | @days_var } ]  
| NO_LOG
```

Предельно внимательно отнеситесь к именованию резервных копий, давайте им логичные и понятные имена. Не перезаписывайте и не удаляйте ни одну из резервных копий, которую создадите в ходе выполнения данной лабораторной работы.

Задание №4

Добавьте в таблицу «Computer» следующие компьютеры: «9_202_1» и «9_202_2» с параметрами, аналогичными компьютеру с именем «9_204_1»

Отметьте для себя (запишите, запомните) текущее время при помощи следующей команды:

```
select getdate()
```

Создайте представление, отображающее только компьютеры с размером оперативной памяти не менее 512 МБ.

Задание №5

Выполните разностное резервное копирование вашей базы данных при помощи команды `BACKUP DATABASE WITH DIFFERENTIAL` (расширение целевого файла .bak).

Задание №6

Добавьте в таблицу «Computer» следующий компьютер: «9_202_3» параметрами, аналогичными компьютеру с именем «9_204_1»

Задание №7

Выполните разностное резервное копирование вашей базы данных при помощи команды BACKUP DATABASE WITH DIFFERENTIAL (расширение целевого файла .bak).

Задание №8

Выполните резервное копирование журнала транзакций вашей базы данных при помощи команды BACKUP LOG (расширение целевого файла .trn)

```
BACKUP LOG { database_name | @database_name_var }  
    TO <backup_device> [ ,...n ]  
    [ <MIRROR TO clause> ] [ next-mirror-to ]  
    [ WITH { <general_WITH_options> | <log-specific_optionspec> }  
    [ ,...n ] ]  
[;]
```

Задание №9

Выполните полное резервное копирование вашей базы данных при помощи команды BACKUP DATABASE (расширение целевого файла .bak)

Задание №10

Добавьте в таблицу «Installation» строки, указанные в таблице 9.1:

Таблица 9.1

Строки для добавления в таблицу «Installation»

Id	Id_computer	Id_program	Installation_datetime
16	7	1	01.04.2009 12:10:00
17	6	1	02.04.2004 12:30:00

Отметьте для себя (запишите, запомните) текущее время при помощи следующей команды:

```
select getdate()
```


Удалите все строки из таблицы OS:

```
delete from OS
```

Просмотрите таблицу Installation:

```
select * from Installation
```

Вот к чему приводит каскадное удаление.

Задание №11

Выполните резервное копирование журнала транзакций вашей базы данных при помощи команды BACKUP LOG (расширение целевого файла .trn)

Задание №12

Подготовьте и выполните скрипты для восстановления базы данных (за минимальное число операций восстановления) к следующим состояниям:

- После добавления записей в таблицу Installation, но до удаления записей из таблиц;
- После добавления компьютеров «9_202_1» и «9_202_2» и создания представления;
- После добавления компьютера «9_202_3»;
- Непосредственно до создания представления;
- После добавления записей в таблицу Installation, но до удаления записей из таблиц (используя только первую полную резервную копию и резервные копии журнала транзакций).

Синтаксис команды восстановления базы данных следующий:

```
RESTORE DATABASE { database_name | @database_name_var }  
[ FROM <backup_device> [ ,...n ] ]  
[ WITH  
    [ { CHECKSUM | NO_CHECKSUM } ]  
    [ [ , ] { STOP_ON_ERROR | CONTINUE_AFTER_ERROR } ]
```

```

[ [ , ] FILE = { backup_set_file_number |
@backup_set_file_number } ]
[ [ , ] KEEP_REPLICATION ]
[ [ , ] MEDIANAME = { media_name | @media_name_variable } ]
[ [ , ] MEDIAPASSWORD = { mediapassword |
                        @mediapassword_variable } ]
[ [ , ] MOVE 'logical_file_name_in_backup' TO
'operating_system_file_name' ]
[ ,...n ]
[ [ , ] PASSWORD = { password | @password_variable } ]
[ [ , ] BLOCKSIZE = { blocksize | @blocksize_variable } ]
[ [ , ] BUFFERCOUNT = { buffercount | @buffercount_variable
} ]
[ [ , ] MAXTRANSFERSIZE = { maxtransfersize |
@maxtransfersize_variable } ]
[ [ , ] ENABLE_BROKER ]
[ [ , ] ERROR_BROKER_CONVERSATIONS ]
[ [ , ] NEW_BROKER ]
[ [ , ] { RECOVERY | NORECOVERY | STANDBY =
        {standby_file_name | @standby_file_name_var }
} ]
[ [ , ] REPLACE ]
[ [ , ] RESTART ]
[ [ , ] RESTRICTED_USER ]
[ [ , ] { REWIND | NOREWIND } ]
[ [ , ] { UNLOAD | NOUNLOAD } ]
[ [ , ] STATS [ = percentage ] ]
[ [ , ] { STOPAT = { 'date_time' | @date_time_var }
| STOPATMARK = { 'lsn:lsn_number' }
[ AFTER 'datetime' ]
| STOPBEFOREMARK = { 'lsn:lsn_number' }
[ AFTER 'datetime' ]
} ]
]
[;]
<backup_device> ::=

```

```
{
    { logical_backup_device_name |
        @logical_backup_device_name_var }
    | { DISK | TAPE } = { 'physical_backup_device_name' |
        @physical_backup_device_name_var }
}
```

Синтаксис восстановления по журналу транзакций следующий:

--To Restore a Transaction Log:

```
RESTORE LOG { database_name | @database_name_var }
    [ <file_or_filegroup_or_pages> [ ,...n ] ]
[ FROM <backup_device> [ ,...n ] ]
[ WITH
    [ { CHECKSUM | NO_CHECKSUM } ]
    [ [ , ] { STOP_ON_ERROR | CONTINUE_AFTER_ERROR } ]
    [ [ , ] FILE = { backup_set_file_number |
@backup_set_file_number } ]
    [ [ , ] KEEP_REPLICATION ]
    [ [ , ] MEDIANAME = { media_name | @media_name_variable } ]
    [ [ , ] MEDIAPASSWORD = { mediapassword |
@mediapassword_variable } ]
    [ [ , ] MOVE 'logical_file_name_in_backup' TO
'operating_system_file_name' ]
        [ ,...n ]
    [ [ , ] PASSWORD = { password | @password_variable } ]
    [ [ , ] { RECOVERY | NORECOVERY | STANDBY =
        {standby_file_name | @standby_file_name_var } }
]
[ [ , ] REPLACE ]
[ [ , ] RESTART ]
[ [ , ] RESTRICTED_USER ]
[ [ , ] { REWIND | NOREWIND } ]
[ [ , ] { UNLOAD | NOUNLOAD } ]
[ [ , ] STATS [=percentage] ]
[ [ , ] { STOPAT = { 'date_time' | @date_time_var }
| STOPATMARK = { 'mark_name' | 'lsn:lsn_number' } }
```

```

[ AFTER 'datetime' ]
| STOPBEFOREMARK = { 'mark_name' | 'lsn:lsn_number' }
[ AFTER 'datetime' ]
}]
[;]

```

Задание №13

Выполните **полное** резервное копирование системных базы данных «master», «msdb» и «model» при помощи команды BACKUP DATABASE.

Задание №14

Выполните проверку **всех** созданных вами резервных копий при помощи команды RESTORE VERIFYONLY:

```

RESTORE VERIFYONLY
FROM <backup_device> [ ,...n ]
[ WITH
    [ { CHECKSUM | NO_CHECKSUM } ]
    [ [ , ] { CONTINUE_AFTER_ERROR | STOP_ON_ERROR } ]
    [ [ , ] FILE =backup_set_file_number ]
    [ [ , ] LOADHISTORY ]
    [ [ , ] MEDIANAME = { media_name | @media_name_variable } ]
    [ [ , ] MEDIAPASSWORD = { mediapassword |
        @mediapassword_variable } ]
    [ [ , ] MOVE 'logical_file_name' TO
'operating_system_file_name' ]
    [ ,...n ]
    [ [ , ] PASSWORD = { password | @password_variable } ]
    [ [ , ] { REWIND | NOREWIND } ]
    [ [ , ] STATS [ = percentage ] ]
[ [ , ] { UNLOAD | NOUNLOAD } ]
]
[;]

<backup_device> ::=
{

```

```

{ logical_backup_device_name |
    @logical_backup_device_name_var }
| { DISK | TAPE } = { 'physical_backup_device_name' |
    @physical_backup_device_name_var }
}

```

Задание №15

Убедитесь, что **SQL Server Agent** запущен (рис. 9.2):

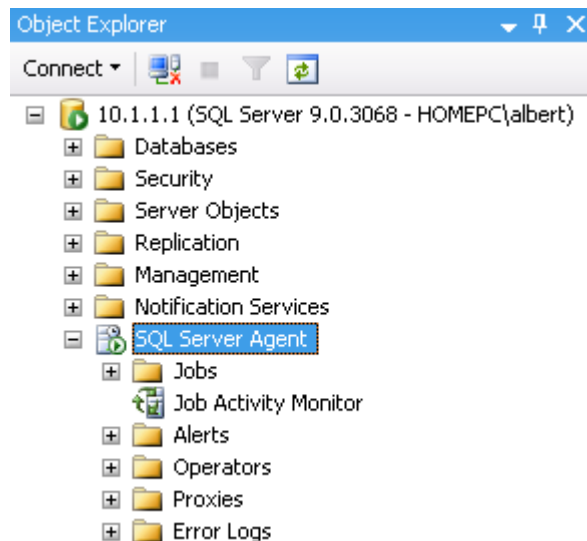


Рис. 9.2. Проверка **SQL Server Agent**

SQL Server Agent необходим для планового выполнения различных операций, например, резервного копирования.

При помощи мастера создания планов обслуживания создайте следующие задания для SQL Server Agent:

- Полное резервное копирование вашей базы данных – каждый день в 00:00
- Разностное резервное копирование вашей базы данных – каждые 4 часа
- Резервное копирование журнала транзакций – каждые 15 минут
- Полное резервное системных баз данных «master», «msdb» и «model» – каждый день в 23:30

Вызовите мастера создания планов обслуживания (рис. 9.3):

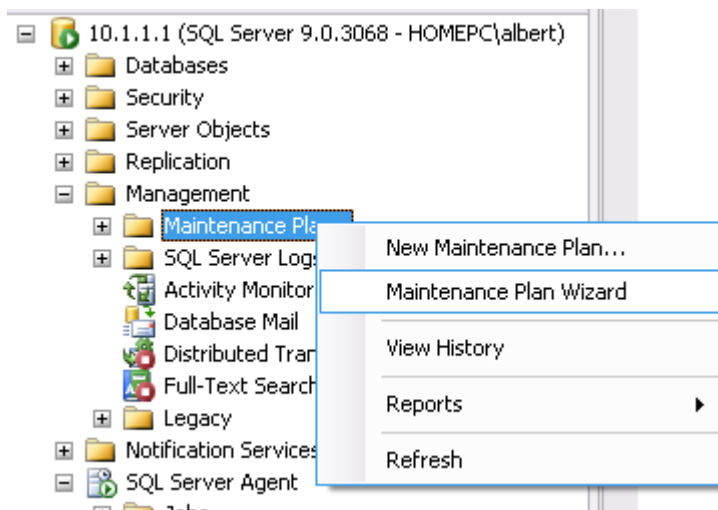


Рис. 9.3. Вызов мастера создания планов обслуживания

Задание №16

По окончании выполнения работы (после окончания занятия или отчета по лабораторной работы) удалите созданные вами планы обслуживания и задания SQL Server Agent.

Лабораторная работа №10

«Работа с транзакциями»

Необходимое программное обеспечение для выполнения лабораторной работы:

- Microsoft Visual Studio 2008
- Microsoft SQL Server 2008

Задание к лабораторной работе

1. Восстановите базу данных «edu» из прилагаемой к лабораторной работе резервной копии в каталоге «mssql_lab10»:

```
RESTORE DATABASE [eduIvanovII]
FROM DISK = 'путь к файлу\edu.bak'
WITH
    MOVE 'edu' TO 'D:\SQL\edu_IvanovII.mdf',
    MOVE 'edu_log' TO 'D:\SQL\edu_IvanovII.ldf',
    STATS = 5,
    RECOVERY
```

2. Изучите возможности SQL Server по явному и неявному запуску транзакций. Для этого выполните приведенный ниже код и проанализируйте исполняемые в нем команды.

```
USE edu
GO
SET NOCOUNT ON;
GO
SET IMPLICIT_TRANSACTIONS OFF;
GO
PRINT N'Tran count at start = '
    + CAST(@@TRANCOUNT AS NVARCHAR(10));
GO
IF OBJECT_ID(N'dbo.t1',N'U') IS NOT NULL
    DROP TABLE dbo.t1;
```

```

GO
CREATE table dbo.t1 (a int);
GO
INSERT INTO dbo.t1 VALUES (1);
GO
PRINT N'Use explicit transaction.';
BEGIN TRANSACTION;
GO
INSERT INTO dbo.t1 VALUES (2);
GO
PRINT N'Tran count in explicit transaction = '
      + CAST(@@TRANCOUNT AS NVARCHAR(10));
COMMIT TRANSACTION;
GO
PRINT N'Tran count after explicit transaction = '
      + CAST(@@TRANCOUNT AS NVARCHAR(10));
GO

PRINT N'Setting IMPLICIT_TRANSACTIONS ON.';
GO
SET IMPLICIT_TRANSACTIONS ON;
GO
PRINT N'Use implicit transactions.';
GO
-- No BEGIN TRAN needed here.
INSERT INTO dbo.t1 VALUES (4);
GO
PRINT N'Tran count in implicit transaction = '
      + CAST(@@TRANCOUNT AS NVARCHAR(10));
COMMIT TRANSACTION;
PRINT N'Tran count after implicit transaction = '
      + CAST(@@TRANCOUNT AS NVARCHAR(10));
GO

PRINT N'Nest an explicit transaction with IMPLICIT_TRANSACTIONS
ON.';

```



```

GO
PRINT N'Tran count before nested explicit transaction = '
      + CAST(@@TRANCOUNT AS NVARCHAR(10));

BEGIN TRANSACTION;
PRINT N'Tran count after nested BEGIN TRAN in implicit
transaction = '
      + CAST(@@TRANCOUNT AS NVARCHAR(10));
INSERT INTO dbo.t1 VALUES (5);
COMMIT TRANSACTION;
PRINT N'Tran count after nested explicit transaction = '
      + CAST(@@TRANCOUNT AS NVARCHAR(10));
GO
-- Commit outstanding tran.
COMMIT TRANSACTION;
GO

```

Ответьте на следующие вопросы:

- Как установить для соединения режим неявных транзакций?
- Как установить для соединения режим с автоматической фиксацией транзакций?
- Когда соединение находится в режиме неявных транзакций, и соединение в этот момент не участвует в транзакции, выполнение каких инструкций начнет транзакцию?
- Каковы особенности режима неявных транзакций, какие дополнительные действия должен осуществить пользователь перед закрытием соединения?

3. Определите, какой уровень изоляции транзакций установлен в данный момент, для чего используйте инструкцию DBCC USEROPTIONS:

```

DBCC USEROPTIONS;
GO

```

4. При помощи следующей инструкции определите текущую установку параметра LOCK_TIMEOUT:

```
SELECT @@lock_timeout;
```

5. Выполните следующие команды при разных уровнях изоляции транзакций (READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SNAPSHOT, SERIALIZABLE) и при различных значениях параметра LOCK_TIMEOUT (-1, 200) (табл. 10.1).

Объясните возникающие побочные эффекты параллелизма («грязные» операции чтения, неповторяющееся чтение, фантомные операции чтения).

Таблица 10.1

Сеансы выполнения команд

Сеанс 1	Сеанс 2
begin transaction	begin transaction
select * from Program	
-----	-----
update Program	select * from Program
set hdd_space = 14100,	
run_ram_space = 512	
where id = 11	
-----	-----
commit transaction	--чтение незафиксированных данных
begin transaction	select * from Program
	update Program
	set hdd_space = 13500,
	run_ram_space = 256
	where id = 11
-----	-----

<pre> update Program set hdd_space = 14500, run_ram_space = 192 where id = 11 ----- --неповторяемое чтение select * from Program --фантом(исчезла строка) select * from Program ----- commit transaction </pre>	<pre> commit transaction begin transaction ----- delete from Program where id = 11 rollback transaction ----- </pre>
---	---

Уровень изоляции транзакций устанавливается при помощи команды:

SET TRANSACTION ISOLATION LEVEL:

SET TRANSACTION ISOLATION LEVEL

```

{ READ UNCOMMITTED
  | READ COMMITTED
  | REPEATABLE READ
  | SNAPSHOT
  | SERIALIZABLE
}
```

[;]

6. Придумайте примеры (напишите код запросов), демонстрирующие побочные эффекты параллелизма («грязные» операции чтения, неповторяющееся чтение, фантомные операции чтения) на данной базе данных.

7. Изучите обработку ошибок при помощи конструкции TRY CATCH на следующем примере:

BEGIN TRY

SELECT 1/0;

END TRY

BEGIN CATCH

```

SELECT
    ERROR_NUMBER() AS ErrorNumber,
    ERROR_SEVERITY() AS ErrorSeverity,
    ERROR_STATE() AS ErrorState,
    ERROR_PROCEDURE() AS ErrorProcedure,
    ERROR_LINE() AS ErrorLine,
    ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
GO

```

Ответьте на следующие вопросы:

- Какие ошибки перехватывает конструкция TRY...CATCH?
- Какие ограничения накладываются на инструкции в блоке TRY CATCH?
- Какие уровни серьезности ошибок формируются SQL Server Database Engine?

8. Выполните следующий код для создания таблицы my_sales и хранимой процедуры usp_MyErrorLog:

```

USE edu;
GO

-- Verify that the table does not exist.
IF OBJECT_ID (N'my_sales',N'U') IS NOT NULL
    DROP TABLE my_sales;
GO

-- Create and populate the table for deadlock simulation.
CREATE TABLE my_sales
(
    Itemid          INT PRIMARY KEY,
    Sales           INT not null
);
GO

```

```
INSERT my_sales (itemid, sales) VALUES (1, 1);
```

```
INSERT my_sales (itemid, sales) VALUES (2, 1);
```

```
GO
```

```
-- Verify that the stored procedure for error printing  
-- does not exist.
```

```
IF OBJECT_ID (N'usp_MyErrorLog',N'P') IS NOT NULL
```

```
    DROP PROCEDURE usp_MyErrorLog;
```

```
GO
```

```
-- Create a stored procedure for printing error information.
```

```
CREATE PROCEDURE usp_MyErrorLog
```

```
AS
```

```
    PRINT
```

```
        'Error ' + CONVERT(VARCHAR(50), ERROR_NUMBER()) +  
        ', Severity ' + CONVERT(VARCHAR(5), ERROR_SEVERITY()) +  
        ', State ' + CONVERT(VARCHAR(5), ERROR_STATE()) +  
        ', Line ' + CONVERT(VARCHAR(5), ERROR_LINE());
```

```
    PRINT
```

```
        ERROR_MESSAGE();
```

```
GO
```

```
-- Verify that the stored procedure for error printing  
-- does not exist.
```

```
IF OBJECT_ID (N'usp_MyErrorLog',N'P') IS NOT NULL
```

```
    DROP PROCEDURE usp_MyErrorLog;
```

```
GO
```

```
-- Create a stored procedure for printing error information.
```

```
CREATE PROCEDURE usp_MyErrorLog
```

```
AS
```

```
    PRINT
```

```
        'Error ' + CONVERT(VARCHAR(50), ERROR_NUMBER()) +  
        ', Severity ' + CONVERT(VARCHAR(5), ERROR_SEVERITY()) +  
        ', State ' + CONVERT(VARCHAR(5), ERROR_STATE()) +  
        ', Line ' + CONVERT(VARCHAR(5), ERROR_LINE());
```

```
    PRINT
```

```
ERROR_MESSAGE ( ) ;
```

```
GO
```

9. Выполните одновременно следующие сценарии кода для сеанса 1 и сеанса 2 с использованием двух отдельных соединений среды SQL Server Management Studio (см. Таблица 10.2).

В ходе обоих сеансов предпринимаются попытки обновить одни и те же строки таблицы. Один из сеансов успешно проведет операцию обновления с первой попытки, а второй будет выбран жертвой взаимоблокировки. Ошибка жертвы взаимоблокировки приведет к тому, что управление внезапно будет передано блоку CATCH и транзакция перейдет в нефиксируемое состояние.

Внутри блока CATCH жертва взаимоблокировки может выполнить откат транзакции и вновь пытаться обновить таблицу до тех пор, пока процесс обновления не завершится успешно или пока не будет исчерпан лимит попыток. К прекращению попыток обновления таблицы приведет то из перечисленных событий, которое наступит первым.

Прокомментируйте команды, исполняемые в каждом сеансе и результаты их совместного выполнения.

Сценарии кода

Сеанс 1	Сеанс 2
<pre> USE edu; GO -- Declare and set variable -- to track number of retries -- to try before exiting. DECLARE @retry INT; SET @retry = 5; -- Keep trying to update -- table if this task is -- selected as the deadlock -- victim. WHILE (@retry > 0) BEGIN BEGIN TRY BEGIN TRANSACTION; UPDATE my_sales SET sales = sales + 1 WHERE itemid = 1; WAITFOR DELAY '00:00:13'; UPDATE my_sales SET sales = sales + 1 WHERE itemid = 2; SET @retry = 0; </pre>	<pre> USE edu; GO -- Declare and set variable -- to track number of retries -- to try before exiting. DECLARE @retry INT; SET @retry = 5; --Keep trying to update -- table if this task is -- selected as the deadlock -- victim. WHILE (@retry > 0) BEGIN BEGIN TRY BEGIN TRANSACTION; UPDATE my_sales SET sales = sales + 1 WHERE itemid = 2; WAITFOR DELAY '00:00:07'; UPDATE my_sales SET sales = sales + 1 WHERE itemid = 1; </pre>

<pre> COMMIT TRANSACTION; END TRY BEGIN CATCH -- Check error number. -- If deadlock victim error, -- then reduce retry count -- for next update retry. -- If some other error -- occurred, then exit -- retry WHILE loop. IF (ERROR_NUMBER() = 1205) SET @retry = @retry - 1; ELSE SET @retry = -1; -- Print error information. EXECUTE usp_MyErrorLog; IF XACT_STATE() <> 0 ROLLBACK TRANSACTION; END CATCH; END; -- End WHILE loop. GO </pre>	<pre> SET @retry = 0; COMMIT TRANSACTION; END TRY BEGIN CATCH -- Check error number. -- If deadlock victim error, -- then reduce retry count -- for next update retry. -- If some other error -- occurred, then exit -- retry WHILE loop. IF (ERROR_NUMBER() = 1205) SET @retry = @retry - 1; ELSE SET @retry = -1; -- Print error information. EXECUTE usp_MyErrorLog; IF XACT_STATE() <> 0 ROLLBACK TRANSACTION; END CATCH; </pre>
--	--

	END; -- End WHILE loop. GO
--	-------------------------------

10. Запустите SQL Server Profiler. Создайте трассировку, в которой включите обработку событий «Deadlock graph».

Повторно запустите сценарии кода из предшествующего задания. Отобразите и прокомментируйте граф взаимоблокировок.

Лабораторная работа №11

«Обслуживание базы данных»

Необходимое программное обеспечение для выполнения лабораторной работы:

- Microsoft Visual Studio 2008
- Microsoft SQL Server 2008

Задание к лабораторной работе

1. Восстановите базу данных «edu» из прилагаемой к лабораторной работе резервной копии в каталоге «mssql_lab11»:

```
RESTORE DATABASE [eduIvanovII]
FROM DISK = 'путь к файлу\edu.bak'
WITH
    MOVE 'edu' TO 'D:\SQL\edu_IvanovII.mdf',
    MOVE 'edu_log' TO 'D:\SQL\edu_IvanovII.ldf',
    STATS = 5,
    RECOVERY
```

2. Просмотрите степень фрагментированности индексов таблицы «Computer» при помощи следующих команд:

```
SELECT * FROM sys.dm_db_index_physical_stats
    (DB_ID(N'edu'), OBJECT_ID(N'dbo.Computer'), NULL, NULL ,
    'DETAILED');
dbcc showcontig
dbcc showcontig with tableresults
```

Проанализируйте значения следующих показателей:

- Средняя плотность страницы (полная) (Avg. Page Density full));
- Плотность просмотра (Scan Density);
- Логическое разбиение просмотра (Logical Scan Fragmentation).

Ответьте на следующие вопросы:

- Как определить, что индекс подвержен внешней фрагментации?
- Как определить, что индекс подвержен внутренней фрагментацией?

3. Выполните запросы, содержащиеся в файле «do_fragments.sql».

Оцените степень фрагментированности индексов для таблицы «Computer».

Выполните реорганизацию индексов для таблицы «Computer» при помощи команды ALTER INDEX REORGANIZE. Зафиксируйте полученные результаты.

4. Оцените степень фрагментированности индексов для таблицы «Computer».

Выполните перестроение индексов для таблицы «Computer» при помощи команды ALTER INDEX REBUILD.

Зафиксируйте полученные результаты.

5. Запустите SQL Server Profiler, выполните повторно задания 3-4 и сравните трудоёмкости выполнения операций реорганизации и перестроения индексов.

6. Создайте отсутствующие статистики для полей задействованных в индексах, при помощи следующей системной процедуры:

```
EXEC sp_createstats 'indexonly';
```

Обновите существующие статистики при помощи следующей системной процедуры EXEC sp_updatestats;

7. Оцените размер базы данных «edu» и таблицы «Computer» при помощи команд:

```
EXEC sp_spaceused
```

```
EXEC sp_spaceused 'dbo.Computer'
```

8. Выполните сжатие базы данных при помощи команды

```
DBCC SHRINKDATABASE (edu, NOTRUNCATE )
```

Повторно выполните задание 7.

9. Выполните усечение базы данных при помощи команды

```
DBCC SHRINKDATABASE (edu, TRUNCATEONLY )
```

Повторно выполните задание 7.

Назовите отличия в результатах выполнения команд в заданиях 8 и 9.

10. Создайте резервную копию базы данных!

11. Выполните проверку файловой системы для следующих томов:

- Системный том (где установлена операционная система);
- Том с бинарными файлами экземпляра SQL Server;
- Том с файлами данных и журналов базы данных.

Для проверки используется утилита Windows **chkdsk.exe**.

Синтаксис данной утилиты следующий:

```
CHKDSK [том:[[путь]имя_файла]] [/F] [/V] [/R] [/X] [/I] [/C]  
[/L[:размер]]
```

Том – определяет точку подключения, имя тома или букву проверяемого диска с двоеточием;

имя_файла – файлы, проверяемые на наличие фрагментации (только FAT/FAT32);

/F – исправление ошибок на диске;

/V – для FAT/FAT32: вывод полного пути и имени для каждого файл на этом диске; для NTFS: также вывод сообщений об очистке.

/R – поиск поврежденных секторов и восстановление их содержимого (требует /F);

/L:размер – только для NTFS: изменение размера файла журнала до указанной величины (в КБ). Если размер не указан, выводится текущее значение размера;

/X – при необходимости предварительное отключение тома. Все открытые дескрипторы для этого тома будут недействительны (требуется /F);

/I – только для NTFS: менее строгая проверка индексных элементов;

/C – только для NTFS: пропуск проверки циклов внутри структуры папок.

Ключи /I или /C укорачивают время выполнения CHKDSK за счет пропуска некоторых проверок тома.

12. Выполните проверку базы данных при помощи команды **DBCC CHECKDB**.

Установите однопользовательский режим базы данных при помощи следующей команды:

```
USE master;  
GO  
ALTER DATABASE [edu]  
SET SINGLE_USER  
WITH ROLLBACK IMMEDIATE;  
GO
```

Выполните проверку и исправление ошибок базы данных при помощи команды:

```
DBCC CHECKDB (edu, REPAIR_REBUILD) .
```

Выполните проверку и исправление ошибок базы данных при помощи команды:

```
DBCC CHECKDB (edu, REPAIR_ALLOW_DATA_LOSS) .
```

Установите многопользовательский режим базы данных при помощи следующей команды:

```
USE master;  
GO  
ALTER DATABASE [edu]  
SET MULTI_USER  
WITH ROLLBACK IMMEDIATE;  
GO
```

Ответьте на следующий вопрос:

1. Каким образом оценить пространство в базе tempdb, необходимое для выполнения операций проверки и исправления ошибок?

Лабораторная работа №12

«Создание полнотекстовых каталогов»

Необходимое программное обеспечение для выполнения лабораторной работы:

- Microsoft Visual Studio 2008
- Microsoft SQL Server 2008

Задание к лабораторной работе

1. Создайте базу с именем вида «ftIvanovII».

В данной базе данных создайте таблицу Books:

```
CREATE TABLE [dbo].[Books] (  
    [id] [bigint] NOT NULL,  
    [authors] [varchar](max) NOT NULL,  
    [book_title] [varchar](max) NOT NULL,  
    [book_file] [varbinary](max) NULL,  
    [book_file_type] [varchar](50) NULL,  
    CONSTRAINT [PK_Books] PRIMARY KEY CLUSTERED  
(  
        [id] ASC  
) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,  
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS  
    = ON) ON [PRIMARY]  
) ON [PRIMARY]
```

2. Заполните таблицу Books следующими данными на основе информации из каталога mssql_lab12 (см. рис.12.1).

```
select *
from Books
```

	id	authors	book_title	book_file	book_file_type
1	1	Макс Фрай	Чужак	0xCAEDE8E3E020F1EEF5F0E0E...	.txt
2	2	Макс Фрай	Волонтеры вечности	0xCAEDE8E3E020F1EEF5F0E0E...	.txt
3	3	Макс Фрай	Простые волшебные вещи	0xCAEDE8E3E020F1EEF5F0E0E...	.txt
4	4	Холм Ван Зайчик	Дело жадного варвара	0xD0CF11E0A1B11AE100000000...	.doc

Рис.12.1. Таблица Books

3. Для заполнения столбца «book_file» типа varbinary необходимо использовать инструкцию следующего вида:

```
update Books
set book_file =
    (
        select *
        from OPENROWSET(BULK D:\mssql_lab12\Sample.doc',
SINGLE_BLOB) AS x
    )
where id = 1
```

4. Просмотрите информацию о зарегистрированных фильтрах текста:

```
exec sp_help_fulltext_system_components 'filter'
```

5. Создайте папку для хранения файлов полнотекстового каталога, например «d:\sql\test»

6. Создайте файловую группу «FTFG1» для хранения данных полнотекстового каталога:

```
use master
go
alter database ftIvanovII add filegroup FTFG1
go

alter database ftIvanovII
```



```
add file (name = 'tIvanovIIFT_data', filename =  
'D:\sql\test\tIvanovIIFT_data.ndf', size = 2048KB, filegrowth  
= 1024KB)  
to filegroup [FTFG1]
```

7. Включите возможности полнотекстового поиска в БД «ftIvanovII» и создайте полнотекстовый каталог в файловой группе FTFG1:

```
use master  
go  
alter database ftIvanovII add filegroup FTFG1  
go  
  
alter database ftIvanovII  
add file (name = 'tIvanovIIFT_data', filename =  
'D:\sql\test\tIvanovIIFT_data.ndf', size = 2048KB, filegrowth  
= 1024KB)  
to filegroup [FTFG1]  
use ftIvanovII  
go  
exec sp_fulltext_database 'enable'  
go  
create fulltext catalog BookCatalog on filegroup FTFG1 in path  
'D:\sql\test' as default
```

8. Создайте полнотекстовый индекс:

```
create fulltext index on Books (book_file type column  
book_file_type)  
key index PK_Books on BookCatalog with change_tracking auto
```

Выполните полное заполнение полнотекстового индекса:

```
alter fulltext index on Books start full population
```

Объясните предупреждение о том, что последняя команда будут проигнорирована.

9. Выполните поисковые запросы и проанализируйте результаты их выполнения.

Запрос, использующий FREETEXT:

```
select authors, book_title
from Books
where freetext(book_file, N' Шурф Лонли-Локли')
```

Запрос, использующий CONTAINS:

```
select authors, book_title
from Books
where freetext(book_file, N' Шурф Лонли-Локли')
select authors, book_title
from Books
where contains(book_file, N'меч')
```

Запрос к FREETEXTTABLE:

```
select *
from freetexttable (Books, book_file, N'Джуфффин')
```

Запрос, использующий соединение с FREETEXTTABLE:

```
select b.authors, b.book_title, k.[key], k.[rank]
from Books as b
inner join freetexttable (Books, book_file, N'Джуфффин') as k
on b.id = k.[key]
```

Запрос, использующий соединение с CONTAINSTABLE:

```
select b.authors, b.book_title, k.[key], k.[rank]
from Books as b
inner join containstable (Books, book_file, N'Джуфффин') as k
on b.id = k.[key]
```

Ответьте на следующие вопросы:

- Каково назначение полнотекстового каталога?

- Где хранится полнотекстовый каталог?
- Каковы требования к созданию полнотекстового индекса?
- Почему необходимо периодически выполнять заполнение индекса?
- В чем разница между функциями CONTAINS и FREETEXT?
- Для чего предназначены файлы тезаурусов?

10.Создайте представление **BookView** с привязкой к схеме, содержащее следующие столбцы таблицы **Book**: **id**, **authors**, **book_title**.

Создайте для данного представления первичный кластеризованный индекс по столбцу **id**.

Создайте для данного представления полнотекстовый индекс для столбцов **book_title** и **author**.

11.Выполните выборку строк из представления **BookView** на основе того, что в названии или авторах книги

- присутствует слово «макс»
- присутствует слово «вещи»
- присутствует слово, начинающееся с «вечн»

Выполните выборку строк из таблицы **Books** на основе того, что в книге

- присутствует слово «мяу»
- присутствует слово «меч» рядом с фразой «король Менин»
- присутствует слово «меч» рядом со словом «Багатур»

Ответьте на следующий вопрос:

- Каково назначение параметра **WEIGHT** предиката **CONTAINS**?

Приложение

«Требования к оборудованию и программному обеспечению для установки SQL Server 2008 R2 R2»

Следующие требования относятся и к 32-разрядным, и к 64-разрядным версии SQL Server 2008 R2 R2.

- Выпуск SQL Server 2008 R2 R2 Datacenter доступен для ознакомления в течение 180 дней. Дополнительные сведения см. на веб-узле SQL Server: инструкции по приобретению.
- Корпорация Майкрософт рекомендует устанавливать SQL Server 2008 R2 R2 на компьютерах с файловой системой NTFS. Обновление до SQL Server 2008 R2 R2 в случае использования файловой системы FAT32 не будет блокироваться.
- Программа установки SQL Server заблокирует возможность установки на сжатые диски и диски, доступные только для чтения.
- Сведения об использовании средств SQL Server 2008 R2 R2 для подготовки к обновлению до версии SQL Server 2008 R2 R2 см. в разделе Использование помощника по обновлению для подготовки к обновлениям.
- При установке SQL Server пакет SDK для .NET Framework 3.5 не устанавливается. Однако в нем содержатся средства, которые могут пригодиться при разработке приложений для SQL Server при использовании среды .NET Framework. Пакет .NET Framework SDK можно загрузить с веб-сайта .NET Framework.
- Требование перезагрузки компьютера во время установки SQL Server: после установки платформы .NET Framework необходимо перезагрузить операционную систему. Если для установки установщика Windows также необходима перезагрузка, программа установки отложит перезагрузку до завершения установки компонентов платформы .NET Framework и установщика Windows.

Следующие требования относятся ко всем видам установки SQL Server 2008 R2 R2.

Компонент	Требование
Платформа	<p>Программа установки SQL Server устанавливает следующие необходимые компоненты:</p> <ul style="list-style-type: none"> • .NET Framework3.5 с пакетом обновления 1 (SP1) • Собственный клиент SQL Server • Файлы поддержки программы установки SQL Server
Программное обеспечение	<p>Программе установки SQL Server требуется установщик Microsoft Windows версии 4.5 или более поздней. После установки необходимых компонентов программа установки SQL Server проверяет компьютер, на котором устанавливается SQL Server 2008 R2 R2, на соответствие всем остальным требованиям.</p>
Сетевое программное обеспечение	<p>Требования к сетевому программному обеспечению для 64-разрядных версий SQL Server 2008 R2 R2 те же, что и для 32-разрядных версий.</p> <p>Поддерживаемые операционные системы имеют встроенное сетевое программное обеспечение. Изолированные именованные экземпляры и экземпляры по умолчанию поддерживают следующие сетевые протоколы:</p> <ul style="list-style-type: none"> • общая память; • именованные каналы; • протоколы TCP/IP; • протокол VIA. <p>Примечание. Общая память и протокол VIA не поддерживаются в отказоустойчивых кластерах.</p> <p><i>Примечание</i></p> <p>Протокол VIA является устаревшим. В будущей версии Microsoft SQL Server эта возможность будет удалена. Избегайте использования этой возможности в новых разработках и запланируйте изменение существующих приложений, в которых она применяется.</p>

Компонент	Требование
Виртуализация	<p>SQL Server 2008 R2 R2 поддерживается в средах виртуальных машин, работающих в роли Hyper-V в выпусках операционных Windows Server 2008 с пакетом обновления 2 (SP2) Standard, Enterprise и Data Center. В виртуальной машине должна работать операционная система, поддерживаемая выбранным выпуском SQL Server 2008 R2 R2.</p> <p>Помимо ресурсов, необходимых для родительской секции, каждой виртуальной машине (дочерней секции) необходимо предоставить достаточный объем ресурсов процессора, памяти и диска для экземпляра SQL Server 2008 R2 R2.</p> <p>В роли Hyper-V в Windows Server 2008 с пакетом обновления 2 (SP2) можно выделить до четырех виртуальных процессоров для виртуальных машин, в которых работают 32-разрядные или 64-разрядные версии Windows Server 2008 с пакетом обновления 2 (SP2). Для виртуальных компьютеров, в которых работают 32-разрядные версии Windows Server 2003, можно выделить не более 2 виртуальных процессоров. Для виртуальных компьютеров, на которых работают другие операционные системы, можно выделить только один виртуальный процессор.</p> <p>Примечания.</p> <p>Рекомендуется завершить работу SQL Server 2008 R2 R2 перед завершением работы виртуальной машины.</p> <p>Гостевой режим для отказоустойчивого кластера поддерживается в SQL Server 2008 R2 R2.</p>
Программное обеспечение для доступа в Интернет	<p>Обозреватель Microsoft Internet Explorer 6 с пакетом обновления 1 (SP1) или выше необходим для установки SQL Server 2008 R2 R2. Обозреватель Internet Explorer 6 с пакетом обновления 1 (SP1) или более поздней версии необходим для консоли управления</p>

Компонент	Требование
	(MMC), среды Среда SQL Server Management Studio и Business Intelligence Development Studio, компонента «Конструктор отчетов» служб Службы Reporting Services, а также для HTML-справки.
Жесткий диск	Требования к месту на диске определяются набором устанавливаемых компонентов SQL Server 2008 R2 R2.
Диск	Для установки с CD или DVD-диска необходим соответствующий дисковод.
Отобразить	Для графических средств SQL Server 2008 R2 R2 необходимо разрешение Super VGA или более высокое — как минимум 800x600 пикселей.
Другие устройства	Указывающее устройство: Требуется мышь Майкрософт или совместимое указывающее устройство.

Необходима платформа .NET Framework следующих версий:

- SQL Server 2008 R2 R2 в 64-разрядной версии Windows Server 2003 IA64 — платформа .NET Framework 2.0 с пакетом обновления 2 (SP2);
- SQL Server Express — платформа .NET Framework 2.0 с пакетом обновления 2 (SP2);
- для всех прочих выпусков SQL Server 2008 R2 R2 — платформа .NET Framework 3.5 с пакетом обновления 1 (SP1).

В следующей таблице перечислены системные требования, относящиеся к 32-разрядной версии SQL Server 2008 R2 Developer.

Компонент	Требование
Процессор	<p><i>Тип процессора:</i> Процессор, совместимый с Pentium III или выше</p> <p><i>Быстродействие процессора:</i> Не менее: 1,0 ГГц Рекомендуется: 2 ГГц и выше</p>

Компонент	Требование
Операционная система	Windows XP Home Edition с пакетом обновления 3 (SP3)
	Windows XP Professional с пакетом обновления 3 (SP3)
	Windows XP Tablet с пакетом обновления 3 (SP3)
	Windows XP Professional x64 с пакетом обновления 2 (SP2)
	Windows XP Media Center 2002 с пакетом обновления 3 (SP3)
	Windows XP Media Center 2004 с пакетом обновления 3 (SP3)
	Windows XP Media Center 2005 с пакетом обновления 3 (SP3)
	Windows XP Professional Reduced Media с пакетом обновления 3 (SP3)
	Windows XP Home Edition Reduced Media с пакетом обновления 3 (SP3)
	Windows Server 2003 Datacenter с пакетом обновления 2 (SP2)
	Windows Server 2003 Enterprise с пакетом обновления 2 (SP2)
	Windows Server 2003 Standard с пакетом обновления 2 (SP2)
	64-разрядная версия Windows Server 2003 x64 Datacenter с пакетом обновления 2 (SP2)
	64-разрядная версия Windows Server 2003 x64 Enterprise с пакетом обновления 2 (SP2)
	64-разрядная версия Windows Server 2003 x64 Standard с пакетом обновления 2 (SP2)
	Windows Server 2003 R2 Datacenter с пакетом обновления 2 (SP2)
	Windows Server 2003 R2 Enterprise с пакетом обновления 2 (SP2)
	Windows Server 2003 R2 Standard с пакетом обновления

Компонент	Требование
	<p>2 (SP2)</p> <p>64-разрядная версия Windows Server 2003 R2 x64 Datacenter с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Server 2003 R2 x64 Enterprise с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Server 2003 R2 x64 Standard с пакетом обновления 2 (SP2)</p> <p>Windows Vista Ultimate с пакетом обновления 2 (SP2)</p> <p>Windows Vista Home Premium с пакетом обновления 2 (SP2)</p> <p>Windows Vista Home Basic с пакетом обновления 2 (SP2)</p> <p>Windows Vista Enterprise с пакетом обновления 2 (SP2)</p> <p>Windows Vista Business с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Vista Ultimate x64 с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Vista x64 Home Premium x64 с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Vista x64 Home Basic x64 с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Vista x64 Enterprise с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Vista Business x64 с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 Datacenter с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 Datacenter без Hyper-V с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 Enterprise с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 Enterprise без Hyper-V с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 Standard с пакетом обновления 2</p>

Компонент	Требование
	<p>(SP2)</p> <p>Windows Server 2008 Standard Edition без Hyper-V с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 Web с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 Datacenter с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 Datacenter без Hyper-V с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 Enterprise с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 Enterprise без Hyper-V с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Server 2008 x64 Standard с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Server 2008 x64 Standard без Hyper-V с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Server 2008 x64 Web с пакетом обновления 2 (SP2)</p> <p>Windows 7 Максимальная</p> <p>Windows 7 Домашняя расширенная</p> <p>Windows 7 Домашняя базовая</p> <p>Windows 7 Корпоративная</p> <p>Windows 7 Professional</p> <p>Windows 7 x64 Максимальная</p> <p>Windows 7 x64 Домашняя расширенная</p> <p>Windows 7 x64 Домашняя базовая</p> <p>Windows 7 x64 Корпоративная</p> <p>Windows 7 x64 Профессиональная</p> <p>64-разрядная версия Windows Server 2008 R2 Datacenter</p> <p>64-разрядная версия Windows Server 2008 R2 x64 Enterprise</p> <p>64-разрядная версия Windows Server 2008 R2 x64</p>

Компонент	Требование
	Standard 64-разрядная версия Windows Server 2008 R2 x64 Web Windows Server 2008 R2 x64 для серверных решений Windows Essential
Память	ОЗУ: Не менее: 1 ГБ Рекомендуется: 4 ГБ и более не более: Максимум, поддерживаемый операционной системой

В следующей таблице перечислены системные требования для SQL Server Express with Tools и 32-разрядных версий SQL Server Express и SQL Server Express with Advanced Services.

Компонент	Требование
Процессор	<i>Тип процессора:</i> Процессор, совместимый с Pentium III или выше <i>Быстродействие процессора:</i> Не менее: 1,0 ГГц Рекомендуется: 2 ГГц и выше
Операционная система	Windows XP Home с пакетом обновления 3 (SP3) Windows XP Professional с пакетом обновления 3 (SP3) Windows XP Tablet с пакетом обновления 3 (SP3) Windows XP Media Center 2002 с пакетом обновления 3 (SP3) Windows XP Media Center 2004 с пакетом обновления 3 (SP3) Windows XP Media Center 2005 с пакетом обновления 3 (SP3) Windows XP Professional Reduced Media с пакетом обновления 3 (SP3) Windows XP Home Edition Reduced Media с пакетом обновления 3 (SP3)

Компонент	Требование
	<p>Windows Server 2003 Datacenter с пакетом обновления 2 (SP2)</p> <p>Windows Server 2003 2003 Enterprise с пакетом обновления 2 (SP2)</p> <p>Windows Server 2003 Standard с пакетом обновления 2 (SP2)</p> <p>Windows Server 2003 Web Edition с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Server 2003 x64 Datacenter с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Server 2003 x64 Enterprise с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Server 2003 x64 Standard с пакетом обновления 2 (SP2)</p> <p>Windows Server 2003 Small Business Server R2 Premium с пакетом обновления 2 (SP2)</p> <p>Windows Server 2003 R2 Datacenter с пакетом обновления 2 (SP2)</p> <p>Windows Server 2003 R2 Enterprise с пакетом обновления 2 (SP2)</p> <p>Windows Server 2003 R2 Standard с пакетом обновления 2 (SP2)</p> <p>Windows Server 2003 R2 Web Edition с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Server 2003 R2 x64 Datacenter с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Server 2003 R2 x64 Enterprise с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Server 2003 R2 x64 Standard с пакетом обновления 2 (SP2)</p> <p>Windows Vista Ultimate с пакетом обновления 2 (SP2)</p> <p>Windows Vista Home Premium с пакетом обновления 2 (SP2)</p>

Компонент	Требование
	<p>Windows Vista Home Basic с пакетом обновления 2 (SP2)</p> <p>Windows Vista Enterprise с пакетом обновления 2 (SP2)</p> <p>Windows Vista Business с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Vista Ultimate x64 с пакетом обновления 2 (SP2)3</p> <p>64-разрядная версия Windows Vista x64 Home Premium x64 с пакетом обновления 2 (SP2)3</p> <p>64-разрядная версия Windows Vista x64 Home Basic x64 с пакетом обновления 2 (SP2)3</p> <p>64-разрядная версия Windows Vista Enterprise x64 с пакетом обновления 2 (SP2)3</p> <p>64-разрядная версия Windows Vista Business x64 с пакетом обновления 2 (SP2)3</p> <p>Windows Server 2008 Datacenter с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 Datacenter без Hyper-V с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 Enterprise с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 Enterprise без Hyper-V с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 Standard Server с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 Standard Server без Hyper-V с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 Web Edition с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 64-разрядная версия x64 Datacenter с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Server 2008 x64 Datacenter без Hyper-V с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Server 2008 x64 Enterprise с пакетом обновления 2 (SP2)</p>

Компонент	Требование
	<p>64-разрядная версия Windows Server 2008 x64 Enterprise без Hyper-V с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Server 2008 x64 Standard с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Server 2008 x64 Standard без Hyper-V с пакетом обновления 2 (SP2)</p> <p>64-разрядная версия Windows Server 2008 x64 Web Edition с пакетом обновления 2 (SP2)</p> <p>Windows Server 2008 x64 Foundation Server с пакетом обновления 2 (SP2)</p> <p>Windows XP Embedded с пакетом обновления 3 (SP3) и пакетом возможностей 20072</p> <p>Windows Embedded for Point of Service с пакетом обновления 3 (SP3)2</p> <p>Windows 7 Максимальная</p> <p>Windows 7 Домашняя расширенная</p> <p>Windows 7 Домашняя базовая</p> <p>Windows 7 Корпоративная</p> <p>Windows 7 Professional</p> <p>Windows 7 x64 Максимальная</p> <p>Windows 7 x64 Домашняя расширенная</p> <p>Windows 7 x64 Домашняя базовая</p> <p>Windows 7 x64 Корпоративная</p> <p>Windows 7 x64 Professional</p> <p>64-разрядная версия Windows Server 2008 R2 Datacenter</p> <p>64-разрядная версия Windows Server 2008 R2 x64 Enterprise</p> <p>64-разрядная версия Windows Server 2008 R2 x64 Standard</p> <p>64-разрядная версия Windows Server 2008 R2 x64 Web</p> <p>Windows Server 2008 R2 x64 для серверных решений</p> <p>Windows Essential</p> <p>64-разрядная версия Windows Server 2008 R2 x64 Foundation Server</p>

Компонент	Требование
Память	<p>ОЗУ:</p> <p>Не менее: 256 МБ для SQL Server Express</p> <p>Не менее: 512 МБ для SQL Server Express with Tools и SQL Server Express with Advanced Services</p> <p>Рекомендуется: 1,024 ГБ</p> <p>Не более: 1 ГБ для компонента Database Engine, который устанавливается с выпуском SQL Server Express, SQL Server Express with Tools и SQL Server Express with Advanced Services; 4 ГБ для служб Службы Reporting Services, которые устанавливаются с выпуском SQL Server Express with Advanced Services</p>

Более подробно с программными и аппаратными требованиями можно ознакомиться по следующему адресу:

<http://msdn.microsoft.com/ru-ru/library/ms143506.aspx>

Литература

1. Microsoft SQL Server 2005. Реализация и обслуживание. Учебный курс Microsoft / пер. с англ. – М.: «Русская редакция», СПб. : «Питер», 2007. – 786 с. : ил.
2. Томас Орин, Маклин Йен. Оптимизация и администрирование баз данных Microsoft SQL Server 2005. Учебный курс Microsoft / пер. с англ. – М.: «Русская редакция», 2007. – 624 с. : ил.
3. Microsoft SQL Server. Полезные алгоритмы от SQL.RU / под общ. ред. А.Гладченко. - СПб.: Питер, 2007. - 269с.: ил.

Содержание

Введение	3
Лабораторная работа № 1 «Установка и первый запуск SQL Server»	4
Лабораторная работа №2 «Создание баз данных»	16
Лабораторная работа №3 «Язык модификации данных DML»	25
Лабораторная работа №4 «Программируемые объекты SQL Server. Функции»	35
Лабораторная работа №5 «Программируемые объекты SQL Server. Хранимые процедуры, триггеры»	48
Лабораторная работа №6 «Создание индексов. Индексированные представления»	56
Лабораторная работа № 7 «Система безопасности SQL Server»	66
Лабораторная работа №8 «Система безопасности SQL Server (уровень баз данных)»	71
Лабораторная работа №9 «Резервное копирование и восстановление данных»	77
Лабораторная работа №10 «Работа с транзакциями»	87
Лабораторная работа №11 «Обслуживание базы данных»	98
Лабораторная работа №12 «Создание полнотекстовых каталогов»	103
Приложение «Требования к оборудованию и программному обеспечению для установки SQL Server 2008 R2 R2»	108
Литература	120