--Лаб 1, Титова

CREATE TABLE Northwind.dbo.Leto\_31\_3

(

Id INT,

Age INT,

FirstName NVARCHAR(20),

LastName NVARCHAR(20),

);

use Northwind

go

sp\_help Customers

use Northwind

sp\_help Orders

sp\_help Categories

sp\_spaceused Customers

sp\_spaceused Categories

USE NORTHWIND

GO

SELECT COL\_LENGTH ('CATEGORIES','CATEGORYID'),

COL\_LENGTH ('SYSUSERS','NAME')

GO

USE NORTHWIND

GO

SELECT DB\_ID ('NORTHWIND')

GO

EXEC sp\_helptext '[INFORMATION\_SCHEMA].[CHECK\_CONSTRAINTS]'

--ТАБЛИЦЫ ПОЛЬЗОВАНЕЛЕЙ БД

USE NORTHWIND

GO

SELECT \* FROM INFORMATION\_SCHEMA.TABLES

USE NORTHWIND

GO

SELECT \* FROM INFORMATION\_SCHEMA.CONSTRAINT\_TABLE\_USAGE

USE NORTHWIND

GO

SELECT \* FROM INFORMATION\_SCHEMA.VIEWS

--Создание таблицы TITOVA\_31

USE NORTHWIND

GO

CREATE TABLE TITOVA\_31

(MEMBER INT IDENTITY (1,1) NOT NULL,

LASTNAME CHAR(10) NOT NULL,

FHOTOGRAPH IMAGE NULL)

GO

SELECT TABLE\_NAME

FROM INFORMATION\_SCHEMA.TABLES

WHERE TABLE\_NAME = 'TITOVA\_31'

GO

DROP TABLE TITOVA\_31

SELECT TABLE\_NAME

FROM INFORMATION\_SCHEMA.TABLES

WHERE TABLE\_NAME = 'Leto\_31\_3'

use Northwind

go

sp\_helpDB 'Northwind';

--Лаб 2

--Создание таблицы

DROP TABLE Investors

USE Northwind

CREATE TABLE Investors\_31\_3

( InvestorID INT NOT NULL,

FirstName varchar(30) NOT NULL,

LastName varchar(30) NOT NULL )

--Модификация таблицы

ALTER TABLE Investors\_31\_3

ADD InvestmentCode INT NULL

--Вывод сведений о таблице

EXEC sp\_help Investors\_31\_3

--Предоставление права доступа к объекту базы данных

GRANT SELECT

ON Investors\_31\_3

TO PUBLIC

EXEC sp\_helprotect Investors\_31\_3

--Отзыв права доступа к объекту базы данных

REVOKE SELECT

ON Investors\_31\_3

TO PUBLIC

EXEC sp\_helprotect Investors\_31\_3

--Извлечение данных

INSERT Investors\_31\_3 VALUES (01, 'Amie', 'Baldwin', 103)

INSERT Investors\_31\_3 VALUES (02, 'Jo', 'Brown', 103)

INSERT Investors\_31\_3 VALUES (03, 'Scott', 'Culp', 103)

INSERT Investors\_31\_3 VALUES (04, 'Jon', 'Grande', 103)

INSERT Investors\_31\_3 VALUES (05, 'Lani', 'Ota', 103)

SELECT \* FROM Investors\_31\_3

SELECT Firstname, Lastname from Investors\_31\_3

WHERE (InvestorID = 03 or InvestorID = 05)

ORDER by Firstname

--Модификация данных

UPDATE Investors\_31\_3

SET InvestmentCode = 101

WHERE InvestorID = 04

SELECT \* FROM Investors\_31\_3

DELETE FROM Investors\_31\_3

WHERE InvestorID = 04

SELECT \* FROM Investors\_31\_3

DROP Table Investors\_31\_3

--Лаб 3

-- Выбрать базу данных.

USE Northwind

GO

--Создать таблицу.

CREATE TABLE [New\_31\_3 Table]

(ColumnA INT, ColumnB CHAR (30))

GO

SET NOCOUNT ON

GO

-- Объявить необходимую переменную

DECLARE @MyCounter INT

--Инициализировать переменную

SET @MyCounter = 0

/\* Определить с помощью переменной число циклов \*/

WHILE (@MyCounter<26)

BEGIN

--Вставить в таблицу строку.

INSERT INTO [New\_31\_3 Table] VALUES

-- С помощью переменной получить целочисленное значение для столбца

--ColumnA и сгенерировать уникальную букву для каждой строки.

-- Получить целочисленное значение символа “a” с помощью функции ASSCII

--Прибавить @MyCounter. С помощью функции CHAR преобразовать сумму

--обратно в символы @MyCounter, следующие после символа “a”.

(@MyCounter + 1, CHAR ((@MyCounter + ASCII('a') )) )

/\* Увеличить значение переменной для учета этой итерации цикла\*/

SET @MyCounter = @MyCounter + 1

END

GO

SET NOCOUNT OFF

GO

SELECT \* FROM [New\_31\_3 Table]

DROP TABLE "New\_31\_3 Table"

--Лаб 4

--Создание базы данных с помощью оператора CREATE DATABASE

USE master

GO

CREATE DATABASE BookShop\_31\_3\_DB

ON Primary

(

NAME = BookShop\_31\_3\_dat,

FILENAME ="C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\ BookShop\_31\_3.mdf",

SIZE = 4,

MAXSIZE = 10,

FILEGROWTH = 1

)

LOG ON

(

NAME = BookShop\_31\_3\_log,

FILENAME = "C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\ BookShop\_31\_3.ldf",

SIZE = 2,

MAXSIZE = 5,

FILEGROWTH = 1

)

GO

--Изменение размера файла

ALTER DATABASE BookShop\_31\_3\_DB

MODIFY FILE

(

NAME = BookShop\_31\_3\_dat,

SIZE = 10

)

--Изменение шага

ALTER DATABASE BookShop\_31\_3\_DB

MODIFY FILE

(

NAME = BookShop\_31\_3\_dat,

FILEGROWTH = 2

)

--Изменение имени файла

ALTER DATABASE BookShop\_31\_3\_DB

MODIFY FILE

(

NAME = BookShop\_31\_3\_dat,

NEWNAME = BookShop\_31\_3\_2\_dat

)

--Изменение имени БД

ALTER DATABASE BookShop\_31\_3\_DB MODIFY NAME = BookShop\_31\_3\_2\_DB;

--Введение доп. файла

ALTER DATABASE BookShop\_31\_3\_2\_DB

ADD FILE

(

NAME = BookShop\_test1,

FILENAME ="C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\ BookShop\_test1.ndf",

SIZE = 5,

MAXSIZE = 10,

FILEGROWTH = 1

)

--ВВЕДЕНИЕ ДОП. ЖУРНАЛА ТРАНЗАКЦИЙ

ALTER DATABASE BookShop\_31\_3\_2\_DB

ADD LOG FILE

(

NAME = BookShop\_test2\_log,

FILENAME = "C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\ BookShop\_test2.ldf",

SIZE = 2,

MAXSIZE = 5,

FILEGROWTH = 1

)

GO

--Лаб 5

USE BookShop\_31\_3\_2\_DB

GO

--Создание таблицы

CREATE TABLE Zebra

(

first\_name VARCHAR(10),

last\_name VARCHAR(10),

groupa INT

);

--Добавление нового столбца

ALTER TABLE Zebra

ADD year DATE;

--Просмотр справки о таблице

sp\_help Zebra;

--Переименование таблицы

EXEC sp\_rename 'Zebra', 'New\_Zebra';

--Переименование столбца

EXEC sp\_rename 'New\_Zebra.groupa', 'groupa1', 'COLUMN';

--Справка

sp\_help New\_Zebra;

--Удаление столбца

ALTER TABLE New\_Zebra DROP COLUMN groupa1;

GO

--Создание пользовательских типов данных sp\_addtype

sp\_addtype Member\_no, int, 'NOT NULL';

sp\_addtype Phonenumber, 'char(13)', 'NULL';

sp\_addtype Shortstring, 'varchar(15)';

--Создание пользовательских типов данных CREATE TYPE

CREATE TYPE Item

FROM CHAR(8);

CREATE TYPE Yes\_no

FROM CHAR(1);

CREATE TYPE Letter

FROM CHAR(1) NOT NULL;

CREATE TYPE Normstring

FROM VARCHAR(13) NOT NULL;

--Проверка созданных типов данных

sp\_datatype\_info;

sp\_help 'Normstring';

--Просмотр таблицы SYSTYPES

SELECT \* FROM SYSTYPES;

--Удаление типов данных

DROP TYPE Letter

DROP TYPE Normstring

--Создание таблицы ADULT

CREATE TABLE ADULT

(

Member\_no Member\_no IDENTITY(2, 3),

street Shortstring,

city Shortstring,

"state" Shortstring,

zip Zipcode,

phone\_no Phonenumber

);

sp\_help Table\_1

6

--Структура таблиц базы данных

USE BookShop\_31\_3\_2\_DB

GO

--Создание таблиц

CREATE TABLE books

(

titleID char(8) NOT NULL,

title varchar(70) NOT NULL,

publisher varchar(50) NOT NULL DEFAULT 'N/A',

pub\_date char(4) NOT NULL DEFAULT 'N/A',

edition char(4) NOT NULL DEFAULT 'N/A',

cost money NOT NULL,

SPR money NOT NULL,

conditionID tinyint NOT NULL,

sold bit NOT NULL DEFAULT '0'

);

sp\_help books

CREATE TABLE book\_condition

(

conditionID tinyint NOT NULL,

condition\_name char(10),

"description" varchar(50)

);

sp\_help book\_condition

CREATE TABLE authors

(

authorID smallint IDENTITY(10, 2) NOT NULL,

first\_name varchar(30) NOT NULL DEFAULT 'unknown',

last\_name varchar(30) NOT NULL,

year\_born char(4) NOT NULL DEFAULT 'N/A',

year\_died char(4) NOT NULL DEFAULT 'N/A',

"description" varchar(200) NOT NULL DEFAULT 'N/A'

);

sp\_help authors

CREATE TABLE book\_authors

(

authorID smallint NOT NULL,

titleID char(8) NOT NULL

);

sp\_help book\_authors

CREATE TABLE employees

(

employeeID smallint IDENTITY NOT NULL,

first\_name varchar(30) NOT NULL,

last\_name varchar(30) NOT NULL,

address1 varchar(60) NOT NULL,

address2 varchar(60) NOT NULL DEFAULT 'N/A',

city varchar(15) NOT NULL,

"state" char(2) NOT NULL,

zip varchar(12) NOT NULL,

phone varchar(24) NOT NULL,

DOB datetime NOT NULL,

hire\_date datetime NOT NULL,

positionID tinyint NOT NULL

);

sp\_help employees

CREATE TABLE positions

(

positionID tinyint NOT NULL,

title varchar(30) NOT NULL,

job\_descrip varchar(80) NOT NULL DEFAULT 'N/A'

);

sp\_help positions

CREATE TABLE customers

(

customerID smallint IDENTITY(10, 1) NOT NULL,

first\_name varchar(30) NOT NULL DEFAULT 'unknown',

last\_name varchar(30) NOT NULL DEFAULT 'unknown',

phone varchar(24) NOT NULL DEFAULT 'unknown',

address1 varchar(60) NOT NULL DEFAULT 'unknown',

address2 varchar(60) NOT NULL DEFAULT 'unknown',

city varchar(15) NOT NULL DEFAULT 'unknown',

"state" varchar(7) NOT NULL DEFAULT 'unknown',

zip varchar(12) NOT NULL DEFAULT 'unknown'

);

sp\_help customers

CREATE TABLE orders

(

orderID smallint IDENTITY NOT NULL,

customerID smallint NOT NULL,

employeeID smallint NOT NULL,

amount money NOT NULL DEFAULT '0',

order\_date datetime NOT NULL,

delivery\_date datetime NULL,

paymentID tinyint NOT NULL,

statusID tinyint NOT NULL

);

CREATE TABLE order\_status

(

statusID tinyint NOT NULL,

status\_descrip varchar(25) NOT NULL,

);

CREATE TABLE from\_of\_payment

(

paymentID tinyint NOT NULL,

payment\_descrip varchar(12) NOT NULL,

);

CREATE TABLE book\_orders

(

orderID smallint NOT NULL,

title char(8) NOT NULL

);

EXEC sp\_rename 'book\_orders.title', 'titleID', 'COLUMN';

--Ограничение PRIMARY KEY для таблицы authors

ALTER TABLE authors

ADD CONSTRAINT authors\_pk PRIMARY KEY (authorID)

--Ограничение PRIMARY KEY для таблицы book\_authors

ALTER TABLE book\_authors

ADD CONSTRAINT bookauhtors\_pk PRIMARY KEY (authorID, titleID)

--Ограничение PRIMARY KEY для таблицы books

ALTER TABLE books

ADD CONSTRAINT books\_pk PRIMARY KEY (titleID)

--Ограничение PRIMARY KEY для таблицы book\_condition

ALTER TABLE book\_condition

ADD CONSTRAINT bookcondition\_pk PRIMARY KEY (conditionID)

--Ограничение PRIMARY KEY для таблицы book\_orders

ALTER TABLE book\_orders

ADD CONSTRAINT bookorders\_pk PRIMARY KEY (titleID)

--Ограничение PRIMARY KEY для таблицы customers

ALTER TABLE customers

ADD CONSTRAINT customers\_pk PRIMARY KEY (customerID)

--Ограничение PRIMARY KEY для таблицы orders

ALTER TABLE orders

ADD CONSTRAINT orders\_pk PRIMARY KEY (orderID)

--Ограничение PRIMARY KEY для таблицы employees

ALTER TABLE employees

ADD CONSTRAINT employees\_pk PRIMARY KEY (employeeID)

--Ограничение PRIMARY KEY для таблицы positions

ALTER TABLE positions

ADD CONSTRAINT positions\_pk PRIMARY KEY (positionID)

--Ограничение PRIMARY KEY для таблицы order\_status

ALTER TABLE order\_status

ADD CONSTRAINT orderstatus\_pk PRIMARY KEY (statusID)

--Ограничение PRIMARY KEY для таблицы order\_status

ALTER TABLE from\_of\_payment

ADD CONSTRAINT fromofpayment\_pk PRIMARY KEY (paymentID)

--Ограничение FOREIGN KEY для таблицы book\_authors, связь с authors через authorID

ALTER TABLE book\_authors

ADD CONSTRAINT authored\_fk Foreign KEY (authorID)

REFERENCES authors (authorID)

--Ограничение FOREIGN KEY для таблицы book\_authors, связь с books через titleID

ALTER TABLE book\_authors

ADD CONSTRAINT titleid\_fk FOREIGN KEY (titleID)

REFERENCES books (titleID)

--Ограничение FOREIGN KEY для таблицы books, связь с book\_condition через conditionID

ALTER TABLE books

ADD CONSTRAINT books\_fk FOREIGN KEY (conditionID)

REFERENCES book\_condition (conditionID)

--Ограничение FOREIGN KEY для таблицы book\_orders, связь с books через titleID

ALTER TABLE book\_orders

ADD CONSTRAINT bookorderstitleID\_fk FOREIGN KEY (titleID)

REFERENCES books (titleID)

--Ограничение FOREIGN KEY для таблицы book\_orders, связь с orders через orderID

ALTER TABLE book\_orders

ADD CONSTRAINT bookordersorderID\_fk FOREIGN KEY (orderID)

REFERENCES orders (orderID)

--Ограничение FOREIGN KEY для таблицы orders, связь с customers через customerID

ALTER TABLE orders

ADD CONSTRAINT orderscustomerID\_fk FOREIGN KEY (customerID)

REFERENCES customers (customerID)

--Ограничение FOREIGN KEY для таблицы orders, связь с employees через employeeID

ALTER TABLE orders

ADD CONSTRAINT ordersemployeeID\_fk FOREIGN KEY (employeeID)

REFERENCES employees (employeeID)

--Ограничение FOREIGN KEY для таблицы orders, связь с from\_of\_payment через paymentID

ALTER TABLE orders

ADD CONSTRAINT orderspaymentID\_fk FOREIGN KEY (paymentID)

REFERENCES from\_of\_payment (paymentID)

--Ограничение FOREIGN KEY для таблицы employees, связь с positions через positionID

ALTER TABLE employees

ADD CONSTRAINT employees\_fk FOREIGN KEY (positionID)

REFERENCES positions (positionID)

--убеждаемся, что ограничения добавлены к столбцам

sp\_helpconstraint book\_authors

--Добавление к таблице Customers ограничения CHECK

ALTER TABLE customers

ADD CONSTRAINT checknames\_ck CHECK

(first\_name NOT LIKE 'unknown' OR last\_name NOT LIKE 'unknown')

--Добавление к таблице authors ограничения CHECK

ALTER TABLE authors

ADD CONSTRAINT authors1\_ck

CHECK (year\_born LIKE ('[1-2][0,6-9][0-9][0-9]')

OR (year\_born = 'N/A'))

ALTER TABLE authors

ADD CONSTRAINT authors2\_ck

CHECK (year\_born NOT LIKE '[1][0][0-9][0-9]')

ALTER TABLE authors

ADD CONSTRAINT authors3\_ck

CHECK (year\_born NOT LIKE '[2][6-9][0-9][0-9]')

ALTER TABLE authors

ADD CONSTRAINT authors4\_ck

CHECK (year\_died LIKE ('[1-2][0,6-9][0-9][0-9]')

OR (year\_died = 'N/A'))

ALTER TABLE authors

ADD CONSTRAINT authors5\_ck

CHECK (year\_died NOT LIKE '[1][0][0-9][0-9]')

ALTER TABLE authors

ADD CONSTRAINT authors6\_ck

CHECK (year\_died NOT LIKE '[2][6-9][0-9][0-9]')

--Добавление к таблице books ограничения CHECK

ALTER TABLE books

ADD CONSTRAINT books1\_ck

CHECK (pub\_date LIKE ('[1-2][0,6-9][0-9][0-9]')

OR (pub\_date = 'N/A'))

ALTER TABLE books

ADD CONSTRAINT books2\_ck

CHECK (pub\_date NOT LIKE '[1][0][0-9][0-9]')

ALTER TABLE books

ADD CONSTRAINT books3\_ck

CHECK (pub\_date NOT LIKE '[2][6-9][0-9][0-9]')

--Добавление к таблице customers ограничения CHECK

ALTER TABLE customers

ADD Unique (phone);

ALTER TABLE customers

ADD CONSTRAINT customers\_uc

Unique (address1, address2);

--Контрольная В1

--Создание базы данных Uspevaemost\_31\_3\_DB

USE master

GO

CREATE DATABASE Uspevaemost\_31\_3\_DB

ON Primary

(

NAME = Uspevaemost\_31\_3\_dat,

FILENAME ="C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\ Uspevaemost\_31\_3.mdf",

SIZE = 5,

MAXSIZE = 10,

FILEGROWTH = 1

)

LOG ON

(

NAME = Uspevaemost\_31\_3\_log,

FILENAME = "C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\ Uspevaemost\_31\_3.ldf",

SIZE = 2,

MAXSIZE = 5,

FILEGROWTH = 1

)

GO

USE Uspevaemost\_31\_3\_DB

GO

--Сортировка для ввода русских букв

ALTER DATABASE Uspevaemost\_31\_3\_DB COLLATE Cyrillic\_General\_BIN

--Создание таблицы zach

CREATE TABLE zach

(

"Номер" smallint IDENTITY(1, 3) NOT NULL,

"Фамилия" varchar(30) NOT NULL,

"Имя" varchar(30) NOT NULL,

"Паспорт" bigint NOT NULL UNIQUE

);

--Ограничение PRIMARY KEY для таблицы zach

ALTER TABLE zach

ADD CONSTRAINT zach\_pk PRIMARY KEY ("Номер")

--Ограничение для фамилии

ALTER TABLE zach

ADD CONSTRAINT zach1\_ck

CHECK ("Фамилия" LIKE '[А-Я]%' OR "Фамилия" LIKE '[A-Z]%' COLLATE Cyrillic\_General\_BIN)

ALTER TABLE zach

DROP CONSTRAINT zach1\_ck;

GO

--Ограничение для имени

ALTER TABLE zach

ADD CONSTRAINT zach2\_ck

CHECK ("Имя" LIKE '[А-Я]%' OR "Имя" LIKE '[A-Z]%' COLLATE Cyrillic\_General\_BIN)

ALTER TABLE zach

DROP CONSTRAINT zach2\_ck;

GO

--Ограничение для паспорта

ALTER TABLE zach

ADD CONSTRAINT zach3\_ck

CHECK ("Паспорт" LIKE '[4-6][0-9][0-9][0-9][4,7][0-9][0-9][0-9][0-9][0-9]')

ALTER TABLE zach

DROP CONSTRAINT zach3\_ck;

GO

ALTER TABLE zach

ADD CONSTRAINT unik

UNIQUE ("Паспорт")

--Создание таблицы otmetki

CREATE TABLE otmetki

(

"Номер" smallint NOT NULL UNIQUE,

"ПМ01" int NOT NULL,

"ПМ02" int NOT NULL,

"ПМ03" int NOT NULL

);

--Ограничение PRIMARY KEY для таблицы otmetki

ALTER TABLE otmetki

ADD CONSTRAINT otmetki\_pk PRIMARY KEY ("Номер")

--Ограничение FOREIGN KEY для таблицы otmetki, связь с zach через "Номер"

ALTER TABLE otmetki

ADD CONSTRAINT otmetki\_fk FOREIGN KEY ("Номер")

REFERENCES zach ("Номер")

INSERT INTO zach ("Фамилия", "Имя", "Паспорт")

VALUES ('Titova', 'Elizaveta', 4012456789);

SELECT \* FROM zach;

INSERT INTO zach ("Фамилия", "Имя", "Паспорт")

VALUES ('Титова', 'Елизавета', 4012456788);

SELECT \* FROM zach;

--Лаб 7

USE pubs

GO

--Извлечение всех данных из таблицы Titles

SELECT \* FROM titles;

--Получение данных из определенных столбцов таблицы Titles

SELECT title\_id,title,price,ytd\_sales FROM titles;

--Задание условия, которому должен соответствовать результирующий набор

SELECT title\_id, title, price, ytd\_sales

FROM titles

WHERE price > 10;

--Задание порядка, в котором выводится результирующий набор

SELECT \* FROM titles

ORDER BY price DESC;

SELECT \* FROM titles

ORDER BY title;

--Группировка данных в результирующем наборе

SELECT "type", Avg(price) AS AvgPrice

FROM titles

GROUP BY "type"

ORDER BY AvgPrice DESC;

--Создание таблицы для размещения результирующего набора

SELECT [type], Avg(price) AS AvgPrice

INTO TypeAvgPrice\_31\_3

FROM titles

GROUP BY [type]

ORDER BY AvgPrice DESC;

GO

SELECT \* FROM TypeAvgPrice\_31\_3

--Создание временной таблицы для размещения результирующего набора

SELECT DISTINCT TOP 7 \*

INTO #NewTable\_31\_3

FROM titles

ORDER BY price ASC;

SELECT \* FROM #NewTable\_31\_3

GO

--Создание расчетных полей

SELECT COUNT([title\_id]) AS "кол-во"

FROM titles

WHERE ([type] LIKE 'psychology' OR [type] LIKE 'business')

AND ([price] BETWEEN 10 AND 50);

--Задание на поиск данных

--Найти в таблице Titles книги, которые содержат пустые значения в поле “price” и в поле “advance”.

SELECT \*

FROM titles

WHERE [price] IS NULL OR [advance] IS NULL

--Найти в таблице Titles книги с названиями Net Etiquette, Sushi, Anyone?, The Gourmet Microwave. Информацию выдать по полям title\_id, title, type, price, advance, royalty

SELECT title\_id, title, [type], price, advance, royalty

FROM titles

WHERE title = 'Net Etiquette' OR title = 'Sushi, Anyone?' OR title = 'The Gourmet Microwave';

--Лаб 7 Дополнение

USE Northwind

GO

--Получение данных посредством внутреннего соединения

SELECT o.CustomerID, o.OrderID, o.ShipCountry, s.CompanyName

FROM Orders AS o JOIN Shippers AS s

ON o.ShipVia = s.ShipperID

WHERE o.ShipCountry = 'USA';

GO

--Получение данных посредством левого внешнего соединения

SELECT o.CustomerID, o.OrderID, c.ContactName, c.City

FROM Orders AS o LEFT OUTER JOIN Customers AS c

ON o.CustomerID = c.CustomerID AND o.ShipCity = c.City

GO

--Получение данных с помощью правого соединения

SELECT o.CustomerID, o.OrderID, c.ContactName, c.City

FROM Orders AS o RIGHT OUTER JOIN Customers AS c

ON o.CustomerID = c.CustomerID AND o.ShipCity = c.City

GO

--Получение данных с помощью полного внешнего соединения

SELECT o.CustomerID, o.OrderID, c.ContactName, c.City

FROM Orders AS o FULL OUTER JOIN Customers AS c

ON o.CustomerID = c.CustomerID AND o.ShipCity = c.City

GO

--Использование в подзапросе ключевого слова IN

SELECT o.CustomerID, o.OrderID, c.ContactName, c.City

FROM Orders AS o FULL OUTER JOIN Customers AS c

ON o.CustomerID = c.CustomerID AND o.ShipCity = c.City

WHERE o.ShipCity IN ('Seattle');

GO

--Применение подзапроса с операторами сравнения и ключевым словом ALL

SELECT \* from Orders

SELECT OrderID, UnitPrice

FROM [Order Details]

WHERE UnitPrice > ALL

(SELECT UnitPrice

FROM [Order Details] AS od JOIN Orders AS o

ON od.OrderID = o.OrderID

AND o.EmployeeID = '5'

)

ORDER BY OrderID;

--Применение подзапроса с ключевым словом EXISTS

SELECT OrderID, CustomerID

FROM Orders AS o

WHERE EXISTS

(SELECT \* FROM Customers AS c

WHERE c.CustomerID = o.CustomerID

AND City = 'London'

)

ORDER BY OrderID

GO

--Лаб 8

USE BookShop\_31\_3\_2\_DB

GO

--Создайте таблицу Test1

CREATE TABLE Test1

(

RowID int IDENTITY(1, 1) NOT NULL,

Title varchar(80) NOT NULL,

[Type] char(12) NOT NULL DEFAULT 'unknown',

City varchar(50) NULL,

Cost money NULL

);

GO

--Добавление к таблице Test1 данных с помощью оператора INSERT…VALUES

INSERT INTO Test1 (Title, Type, Cost)

VALUES ('Test Title','business', 27.00);

SELECT \* FROM Test1;

INSERT INTO Test1

VALUES ('SQL', DEFAULT, 'Moscow', 50.00);

SELECT \* FROM Test1;

--Добавление данных в таблицу Test1 с помощью оператора Insert…Select

/\*С помощью этого оператора выбрать данные из полей Title,Type, Price таблицы Titles в базе данных Pubs.

И вставить эти значения в поля Title, Type, Cost таблицы Test1\*/

INSERT INTO Test1 (Title, Type, Cost)

SELECT Title, Type, Price

FROM Pubs.dbo.Titles;

SELECT \* FROM Test1;

go

--Запишите названия нескольких книг, значение поля Type которых равно business, а также их цену

INSERT INTO Test1 (Title, Type, Cost)

VALUES

('Book 1', 'business', 10.00),

('Book 2', 'business', 15.00),

('Book 3', 'business', 20.00);

SELECT \* FROM Test1;

go

--Используя оператор Update, увеличьте в два раза значение поля Cost по сравнению с исходным значением для книг типа business

UPDATE Test1

SET Cost = Cost \* 2

WHERE Type = 'business';

SELECT \* FROM Test1

WHERE Type = 'business';

go

--Удаление данных из таблицы Test1 с помощью оператора Delete

DELETE FROM Test1

WHERE Title = 'Test Title';

SELECT \* FROM Test1;

go

/\*TRUNCATE удаляет всю запись из таблицы сразу,

тогда как команда DROP удаляет таблицу или базы данных,

а также структуру. TRUNCATE нельзя использовать с условием\*/

drop table Test1;

go

--Лаб 9

--Создание представления BookAuthorView в БД BookShopDB

USE BookShop\_31\_3\_2\_DB

GO

/\*Создайте представление, которое содержит запрос,

соединяющий таблицу Authors с BookAuthors (поле внешнего ключа – AuthorID),

а таблицу BookAuthors – с таблицей Books (поле внешнего ключа – TitleID).

В результирующий набор этого запроса войдут имена (FirstName) и фамилии(LastName) авторов,

а также написанные ими книги(Title)\*/

CREATE VIEW BookAuthorView AS

SELECT authors.first\_name AS FirstName,

authors.last\_name AS LastName,

books.title AS Title

FROM authors INNER JOIN book\_authors ON authors.authorID = book\_authors.authorID

INNER JOIN books ON book\_authors.titleID = books.titleID

--Модификация представления BookAuthorView из БД BookShopDB

/\*С помощью оператора модификации представления добавьте к оператору SELECT столбец TitleID.

Теперь в результирующем наборе появится дополнительный столбец.\*/

ALTER VIEW BookAuthorView AS

SELECT authors.first\_name AS FirstName,

authors.last\_name AS LastName,

books.title AS Title,

book\_authors.titleID AS TitleID

FROM authors INNER JOIN book\_authors ON authors.authorID = book\_authors.authorID

INNER JOIN books ON book\_authors.titleID = books.titleID

--Удаление представления BookAuthorView из БД BookShopDB

DROP VIEW BookAuthorView

GO

--Для получения результатов работы представления, введите 3 записи в таблицу AUTHORS

INSERT INTO authors

VALUES ('Alex','Li', 1950, 2001, 'adsfadf'),

('Andro', 'Pak', 1947, 2005, 'akijwiok'),

('Bob', 'Finc', 1938, 1997, 'oijio');

SELECT \* FROM authors;

--Создать представление, которое позволит просматривать все значения записи из таблицы AUTHORS

CREATE VIEW Author AS

SELECT \* FROM authors

--Изменить созданное представление так, чтобы пользователь не смог видеть значения из полей Description и FirstName

ALTER VIEW Author AS

SELECT authorID,last\_name, year\_born, year\_died

FROM authors

--Добавить две записи в таблицу AUTHORS, используя созданное представление

INSERT INTO Author

VALUES ('Lin', 1950, 2001),

('Pako', 1947, 2005);

SELECT \* FROM Author;

/\*Используя созданное представление, измените значение в полях YearBorn и YearDied на 1959 и 2020

соответственно для записи с идентификатором 11(указанные года и идентификатор могут быть любые)\*/

UPDATE Author

SET year\_born = 1959, year\_died = 2020

WHERE authorID = 12;

SELECT \* FROM Author

GO

/\*Используя созданное представление, удалите запись с идентификатором 12

(номер идентификатора может быть любым)\*/

DELETE FROM Author

WHERE authorID = 14;

SELECT \* FROM Author;

go

--Поиск всех представлений с определенной таблицей

SELECT v.name AS ViewName

FROM sys.views v

JOIN sys.sql\_expression\_dependencies sed ON v.object\_id = sed.referencing\_id

WHERE sed.referenced\_entity\_name = 'authors';

--Напишите оператор по удалению вашего представления

DROP VIEW Author;

Go

--ЛАБ 10

USE BookShop\_31\_3\_2\_DB

GO

/\*Создайте процедуру, которая:

- позволит вводить записи в таблицу Authors;\*/

CREATE PROC sp\_my\_proc1

@f varchar(30),

@l varchar(30),

@yb char(4),

@yd char(4),

@d varchar(200)

AS

begin

INSERT INTO authors

VALUES (@f, @l, @yb, @yd, @d)

end

EXEC sp\_my\_proc1 'Ustinova', 'Masha', 1977, 2019, 'kjfghklejfgg'

--обновлять фамилию автору;

CREATE PROC sp\_my\_proc2

@aID smallint,

@ln varchar(30)

AS

BEGIN

UPDATE authors

SET last\_name = @ln

WHERE authorID = @aID

END

EXEC sp\_my\_proc2 @aID = 18, @ln = 'Filipov'

--удалять записи из таблицы

CREATE PROC sp\_my\_proc3

@aID smallint

AS

BEGIN

DELETE FROM authors

WHERE authorID = @aID

END

EXEC sp\_my\_proc3 @aID = 12

--Создание простых триггеров для таблицы Authors из базы данных

/\*Оператор CREATE TRIGGER создает триггер insertindicator и привязывает его к таблице Authors в базе данных BookShopDB.

Этот триггер срабатывает при добавлении данных в таблицу Authors и выводит на панели Grids вкладки Results сообщение\*/

CREATE TRIGGER dbo.insertindicator

ON dbo.Authors

AFTER INSERT

AS

PRINT 'The insert trigger fired'

CREATE TRIGGER dbo.updateindicator

On dbo.authors

AFTER UPDATE

AS

PRINT 'The update trigger fired'

GO

CREATE TRIGGER dbo.deleteindicator

ON dbo.authors

AFTER DELETE

AS

IF @@ROWCOUNT <> 0

PRINT 'The delete trigger fired'

--Проверка триггеров таблицы Authors

INSERT INTO Authors (first\_name, last\_name, year\_born, year\_died)

VALUES ('Max', 'Doe', 1962, 'N/A')

UPDATE authors

SET Authors.first\_name = 'Tucker'

WHERE authors.first\_name = 'Max'

DELETE authors WHERE first\_name='Tucker'

--Переименование, модификация и просмотр триггера

Sp\_rename @objname= insertindicator, @newname=insupdcontrol

--Системная хранимая процедура выводит список триггеров, применяемых к таблице authors

Sp\_helptrigger @tabname = Authors

/\*В результате модификации переименованный триггер преобразуется в триггер INSTEAD OF,

который предотвращает все операции добавления и обновления в таблице Authors.

С помощью триггеров этого типа можно временно запретить внесение изменений в таблицу.

Для включения триггеров используют оператор ALTER TABLE\*/

ALTER TRIGGER dbo.insupdcontrol

ON dbo.Authors

INSTEAD OF INSERT, UPDATE

AS

print 'Insert and update are not allowed at this time'

/\*Триггер INSTEAD OF срабатывает и выводит сообщение, в котором говорится, что в данный момент обновление запрещено.

Активирован параметр NOCOUNT, поэтому на вкладке Messages панели Results не выводится сообщение с числом строк,

на которое повлияло исполнение запроса\*/

SET NOCOUNT ON

INSERT INTO authors (first\_name, last\_name, year\_born, year\_died)

VALUES ('Max', 'Doe', 1962, 'N/A')

SET NOCOUNT OFF

SELECT \* FROM authors

--Хранимая процедура выводит содержимое триггера insupdcontrol

Sp\_helptext @objname=insupdcontrol

--Отключение и удаление триггера

--Этот оператор отключает триггер insupdcontrol в таблице Authors

ALTER TABLE authors DISABLE TRIGGER insupdcontrol

--Запись успешно добавляется в таблицу Authors, добавленная запись выводится на вкладке Grids

INSERT INTO authors (first\_name, last\_name, year\_born, year\_died)

VALUES ('Max', 'Doe', 1962, 'N/A')

SELECT \* FROM authors WHERE first\_name='Max'

/\*Оператор DROP TRIGGER удаляет все три триггера, привязанных к таблице authors.

Оператор DELETE удаляет запись, ранее добавленную к таблице authors\*/

DROP TRIGGER insupdcontrol, updateindicator, deleteindicator

DELETE authors WHERE first\_name='Max'

--Лаб 11

--Создание устройства резервного копирования при помощи команд языка Transact SQL

USE master

GO

EXEC sp\_addumpdevice 'disk','G\_31\_3',

'C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\Backup\G\_31\_3.bak'

go

EXEC sp\_addumpdevice 'disk','G\_31\_3\_log',

'C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\Backup\G\_31\_3\_log.bak'

go

--Создайте базу данных NewPrimer\_31\_3

CREATE DATABASE APrimer\_31\_3

GO

--Создайте в базе данных таблицу Products

USE APrimer\_31\_3

GO

CREATE TABLE Products

(

Member int,

[Name] char(10),

Price int

);

--Добавьте в таблицу пять записей. Например: (1, ’книга’, 236)

INSERT Products

VALUES (1, 'BOOK', 236)

INSERT Products

VALUES (2, 'ALBUM', 250)

INSERT Products

VALUES (3, 'JORNAL', 336)

INSERT Products

VALUES (4, 'PEN', 567)

INSERT Products

VALUES (5, 'MAGAZINE', 124)

INSERT Products

VALUES (6, 'YUGE', 367)

INSERT Products

VALUES (7, 'Note', 246)

SELECT \* FROM Products

--Полное резервное копирование (полное)

BACKUP DATABASE APrimer\_31\_3 TO G\_31\_3 with init

--резервное копирование (журнал транзакций)

BACKUP LOG APrimer\_31\_3 TO G\_31\_3\_log

drop database APrimer\_31\_3

--ЗАКЧЮЧ ФРАГМЕНТ ЖУРНАЛА ТРАНЗАКЦИЙ

BACKUP LOG APrimer\_31\_3 TO G\_31\_3\_log WITH NORECOVERY

--Восстановление полное

RESTORE DATABASE APrimer\_31\_3 FROM G\_31\_3 WITH NORECOVERY, REPLACE

--Восстановление журнала транзакций

RESTORE log APrimer\_31\_3 from G\_31\_3\_log WITH NORECOVERY, file = 1

RESTORE log APrimer\_31\_3 from G\_31\_3\_log WITH NORECOVERY, file = 2

RESTORE log APrimer\_31\_3 from G\_31\_3\_log WITH RECOVERY, file = 3

go

DELETE Products

where [Name] = 'Book';

--Лаб 13

USE BookShop\_31\_3\_2\_DB

GO

/\*Создайте три учётные записи с именами anna\_31\_3, karl\_31\_3 и max\_31\_3.

Соответствующими паролями являются: A1B2c3d4e5, d4E3f2g1H0 и F102gH285\*/

CREATE LOGIN anna\_31\_3

WITH PASSWORD = 'A1B2c3d4e5'

CREATE LOGIN karl\_31\_3

WITH PASSWORD = 'd4E3f2g1H0'

CREATE LOGIN max\_31\_3

WITH PASSWORD = 'F102gH285'

/\*Создайте три новых имени пользователей базы данных для учетных записей из п.1.

Новыми именами будут s\_anna\_31\_3, s\_karl\_31\_3 и s\_max\_31\_3\*/

CREATE USER s\_anna\_31\_3

FOR LOGIN anna\_31\_3

CREATE USER s\_karl\_31\_3

FOR LOGIN karl\_31\_3

CREATE USER s\_max\_31\_3

FOR LOGIN max\_31\_3

/\*Создайте новую определённую пользователем роль базы данных с именем Managers

и добавьте три члена в эту роль (см. п.2). После этого отобразите информацию об этой роли и о трёх её членах\*/

CREATE Role Managers\_31\_3

GO

sp\_addrolemember 'Managers\_31\_3', 's\_anna\_31\_3'

sp\_addrolemember 'Managers\_31\_3', 's\_karl\_31\_3'

sp\_addrolemember 'Managers\_31\_3', 's\_max\_31\_3'

/\*Используя оператор Grant, предоставьте пользователю s\_karl возможность создавать таблицы,

а пользователю s\_anna создавать хранимые процедуры в вашей базе данных\*/

GRANT CREATE TABLE TO s\_karl\_31\_3

GRANT CREATE PROC TO s\_anna\_31\_3

sp\_helprotect Null, s\_karl\_31\_3

sp\_helprotect Null, s\_anna\_31\_3

/\*Используя оператор Grant, предоставьте пользователю s\_max

возможность изменять значения столбцов lname и fname nf, таблицы sample\*/

GRANT SELECT, UPDATE

ON books (title, publisher)

TO s\_max\_31\_3

sp\_helprotect Null, s\_max\_31\_3

/\*Используя оператор Grant, предоставьте пользователю s\_karl и s\_anna возможность

читать значения столбцов emp\_lname и emp\_fname таблицы employee.

Для этого сначала создайте соответствующее представление\*/

CREATE VIEW Employee AS

SELECT employees.first\_name AS FirstName,

employees.last\_name AS LastName

FROM employees

GRANT SELECT

ON Employee

TO s\_karl\_31\_3, s\_anna\_31\_3

GO

/\*Используя оператор Transact SQL, предоставьте определённую пользователем роль Managers

для добавления новых строк в таблицу Project\*/

GRANT INSERT

ON authors

TO Managers\_31\_3

/\*Отмените право SELECT у пользователя s\_karl\*/

DENY SELECT

TO s\_karl\_31\_3

GO

/\*Используя Transact SQL, не давайте возможности пользователю s\_anna добавлять новые строки

в таблицу project ни явно, ни опосредованно (используя роли)\*/

DENY INSERT

TO s\_anna\_31\_3

GO

REVOKE INSERT

TO s\_anna\_31\_3

GO

/\*Отобразите существующую информацию о пользователе s\_anna в отношении своей базы данных,

используя системную процедуру sp\_helpuser\*/

sp\_helpuser s\_anna\_31\_3

--проверка прав пользователя

SELECT \* FROM Employee

SELECT \* FROM employees

SELECT \* FROM authors

--Лаб 15

use BookShop\_31\_3\_DB

go

if exists (SELECT name from dbo.sysobjects

where name = 'table01' AND type = 'U')

DROP TABLE table01

CREATE TABLE table01(uniqueID int IDENTITY,

longcol02 char(300) DEFAULT

'This is default value for this column',

coi03 char(1))

go

DECLARE @counter int

SET @counter=1

WHILE @counter<=5000

BEGIN

INSERT table01 (coi03) VALUES('a')

INSERT table01 (coi03) VALUES('b')

INSERT table01 (coi03) VALUES('c')

INSERT table01 (coi03) VALUES('d')

INSERT table01 (coi03) VALUES('e')

SET @counter=@counter+1

END

--

SELECT coi03, longcol02 FROM table01 WHERE coi03='a'

SELECT uniqueID, longcol02 FROM table01 WHERE uniqueID=10000

SELECT \*FROM table01 WHERE uniqueID BETWEEN 5000 and 10000

go

--

SELECT coi03, longcol02 FROM table01 WHERE coi03='a'

go

SELECT uniqueID, longcol02 FROM table01 WHERE uniqueID=10000

go

SELECT \*FROM table01 WHERE uniqueID BETWEEN 5000 and 10000

go

--Создание трассировки аудита безопасности

use NewPrimer\_31\_3

go

CREATE LOGIN janna

WITH PASSWORD='ABCabc123',

Default\_database=NewPrimer\_31\_3

CREATE USER s\_janna for login janna

grant execute to s\_janna

sp\_helpprotect Null, s\_janna

--

use APrimer\_31\_3

go

if exists (SELECT name from dbo.sysobjects

where name = 'table01' AND type = 'U')

DROP TABLE table01

CREATE TABLE table01(uniqueID int IDENTITY,

longcol02 char(300) DEFAULT

'This is default value for this column',

coi03 char(1))

go

DECLARE @counter int

SET @counter=1

WHILE @counter<=5000

BEGIN

INSERT table01 (coi03) VALUES('a')

INSERT table01 (coi03) VALUES('b')

INSERT table01 (coi03) VALUES('c')

INSERT table01 (coi03) VALUES('d')

INSERT table01 (coi03) VALUES('e')

SET @counter=@counter+1

END

SELECT coi03, longcol02 FROM table01 WHERE coi03='a'

SELECT uniqueID, longcol02 FROM table01 WHERE uniqueID=10000

SELECT \*FROM table01 WHERE uniqueID BETWEEN 5000 and 10000

Go

--Лаб 16

USE NewPrimer\_31\_3

GO

--шифрование данных с применением алгоритма хеширования

--Выполните следующий код и сравните результаты для разных алгоритмов хеширования:

DECLARE @Hash varchar(100)

SELECT @Hash = 'location'

SELECT HashBytes ('MD5', @Hash)

SELECT @Hash = 'location'

SELECT @Hash = HashBytes ('SHA', @Hash)

--Выполните следующий код и обратите внимание, что алгоритм хеширования чувствителен к регистру:

DECLARE @Hash varchar(100)

SELECT @Hash = 'вап'

SELECT HashBytes ('SHA1', @Hash)

SELECT @Hash = 'вап'

SELECT HashBytes ('SHA1', @Hash)

--Шифрование данных с помощью парольной фразы

DECLARE @EncryptedText VarBinary(80)

SELECT @EncryptedText = EncryptByPassphrase('1234567','Liza')

SELECT @EncryptedText,

CAST (DecryptByPassphrase('1234567',@EncryptedText)

AS VARCHAR(MAX))

--Шифрование данных с помощью симметричного ключа

--Выполните следующий код в базе данных NewPrimer\_31\_3, чтобы создать симметричный ключ:

CREATE SYMMETRIC KEY TestSymmetricKey WITH ALGORITHM = RC4

ENCRYPTION BY PASSWORD = 'A1234567a'

SELECT\*FROM sys.symmetric\_keys

--Выполните следующий код, чтобы открыть симметричный ключ:

OPEN SYMMETRIC KEY TestSymmetricKey

DECRYPTION BY PASSWORD = 'A1234567a'

--Выполните следующий код, чтобы просмотреть данные, зашифрованные с помощью симметричного ключа:

DECLARE @EncryptedText VARBINARY(80)

SELECT @EncryptedText = EncryptByKey(Key\_GUID('TestSymmetricKey'), 'Liza')

SELECT @EncryptedText,

CAST (DecryptByKey(@EncryptedText) AS VARCHAR(30))

--Закройте симметричный ключ, выполнив код:

CLOSE SYMMETRIC KEY TestSymmetricKey

GO

--Шифрование данных с помощью сертификата

/\*Вы создадите сертификат и с его помощью зашифруете данные,

чтобы пользователи, не имеющие разрешения на доступ к данным,

не смогли их просмотреть.\*/

--Выполнив следующий код, создайте тестовую таблицу, двух пользователей и разрешения.

CREATE TABLE dbo.CertificateEncryption

(ID INT IDENTITY(1,1),

SalesRep VARCHAR(30) NOT NULL,

SalesLead VARBINARY(500) NOT NULL)

GO

CREATE USER SalesRep1\_31\_3 WITHOUT LOGIN

GO

CREATE USER SalesRep2\_31\_3 WITHOUT LOGIN

GO

GRANT SELECT, INSERT ON dbo.CertificateEncryption TO SalesRep1\_31\_3

GRANT SELECT, INSERT ON dbo.CertificateEncryption TO SalesRep2\_31\_3

GO

--Создайте главный ключ для БД:

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'P@ssw0rd';

SELECT\*FROM sys.symmetric\_keys;

GO

--Создайте сертификат для каждого пользователя:

CREATE CERTIFICATE SalesRep1Cert AUTHORIZATION SalesRep1\_31\_3

WITH SUBJECT = 'SalesRep 1 certificate'

GO

CREATE CERTIFICATE SalesRep2Cert AUTHORIZATION SalesRep2\_31\_3

WITH SUBJECT = 'SalesRep 2 certificate'

GO

SELECT\*FROM sys.certificates

GO

--Вставьте данные для каждого пользователя:

EXECUTE AS USER='SalesRep1\_31\_3'

GO

INSERT INTO dbo.CertificateEncryption

(SalesRep, SalesLead)

VALUES('SalesRep1\_31\_3',EncryptByCert(Cert\_ID('SalesRep1Cert'), 'Fabrikam'))

REVERT

GO

EXECUTE AS USER='SalesRep2\_31\_3'

GO

INSERT INTO dbo.CertificateEncryption

(SalesRep, SalesLead)

VALUES('SalesRep2\_31\_3',EncryptByCert(Cert\_ID('SalesRep2Cert'), 'Contoso'))

REVERT

GO

--Просмотрите содержимое таблицы за каждого пользователя:

SELECT ID, SalesRep, SalesLead

FROM dbo.CertificateEncryption

GO

EXECUTE AS USER='SalesRep1\_31\_3'

GO

SELECT ID, SalesRep, SalesLead,

CAST(DecryptByCert(Cert\_Id('SalesRep1Cert'), SalesLead)

AS VARCHAR(MAX))

FROM dbo.CertificateEncryption

REVERT

GO

EXECUTE AS USER='SalesRep2\_31\_3'

GO

SELECT ID, SalesRep, SalesLead,

CAST(DecryptByCert(Cert\_Id('SalesRep2Cert'), SalesLead)

AS VARCHAR(MAX))

FROM dbo.CertificateEncryption

REVERT

GO

--Реализация прозрачного шифрования

--Создайте главный ключ и сертификат в главной базе данных:

USE MASTER

GO

--CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'A1234567a'

--GO

CREATE CERTIFICATE ServerCert\_31\_3 WITH SUBJECT = 'My Server Cert for TDE'

GO

--Создайте резервную копию сертификата и закрытого ключа, чтобы гарантировать возможность восстановления:

BACKUP CERTIFICATE ServerCert\_31\_3 TO FILE = 'C:\Program Files\Microsoft

SQL Server\MSSQL10.MSSQLSERVER\MSSQL\Backup\servercert.cer'

WITH PRIVATE KEY (FILE = 'C:\Program Files\Microsoft

SQL Server\MSSQL10.MSSQLSERVER\MSSQL\Backup\servercert.key',

ENCRYPTION BY PASSWORD = 'A1234567a')