

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт кибербезопасности и цифровых технологий

Кафедра КБ-2 «Информационно-аналитические системы кибербезопасности»

Отчет по практической работе № 3

по дисциплине «Безопасность систем баз данных»

«Среда управления SQL Server Management Studio. ЯзыкTransact-SQL. Компоненты SQL»

Выполнил студент 3 курса по специальности (направлению подготовки):

10.05.04 «Информационно-аналитические системы безопасности»

Зубарев Василий Сергеевич

|  |  |
| --- | --- |
|  | Проверил:  старший преподаватель Войтенков Д.В.  Отметка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Москва 2025

**Цели работы:**

1. Создание объектов баз данных
2. Модифицирование объектов баз данных
3. Удаление объектов баз данных

В этой работе рассматриваются все инструкции Transact-SQL, связанные с языком описания данных DDL (Data Definition Language ). Инструкции языка DDL разбиты на три группы, которые рассматриваются последовательно. В первую группу входят инструкции для создания объектов, вторая содержит инструкции для модифицирования структуры объектов, а третья состоит из инструкций для удаления объектов.

**Пример 1. Код для создания простой базы данных**

USE master;

CREATE DATABASE sampleNew;

Код, приведенный в примере 1, создает базу данных, которая называется sampleNew. Результат выполнения представлен на рисунке 1.

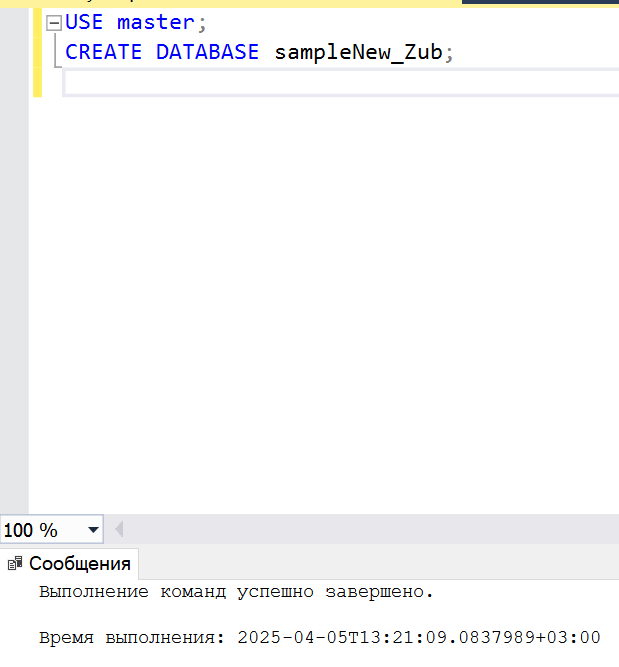


Рисунок 1 - создание БД sampleNew

**Пример 2. Создание базы данных с явным указанием файлов**

USE master;

CREATE DATABASE projects ON (NAME=projects\_dat,

FILENAME = ’C:\ projects.mdf’,

SIZE = 10,

MAXSIZE = 100,

FILEGROWTH = 5)

LOG ON

(NAME=projects\_log,

FILENAME = ’C:\ projects.ldf’,

SIZE = 40,

MAXSIZE = 100,

FILEGROWTH = 10);

Созданная в листинге 2 база данных называется project. Поскольку опция primary не указана, то первичным файлом предполагается первый файл. Этот файл имеет логическое имя projects\_dat и он сохраняется в дисковом файле projects.mdf. Исходный размер этого файла 10 Мбайт. При необходимости, система выделяет этому файлу дополнительное дисковое пространство в приращениях по 5 Мбайт. Если не указать опцию maxsize или если этой опции присвоено значение unlimited, то максимальный размер файла может увеличиваться и будет ограничиваться только размером всего дискового пространства. (Единицу размера файла можно указывать с помощью суффиксов kb, tb и mb, означающих килобайты, терабайты и мегабайты соответственно. По умолчанию используется единица размера mb, т. е. мегабайты.). Результат выполнения представлен на

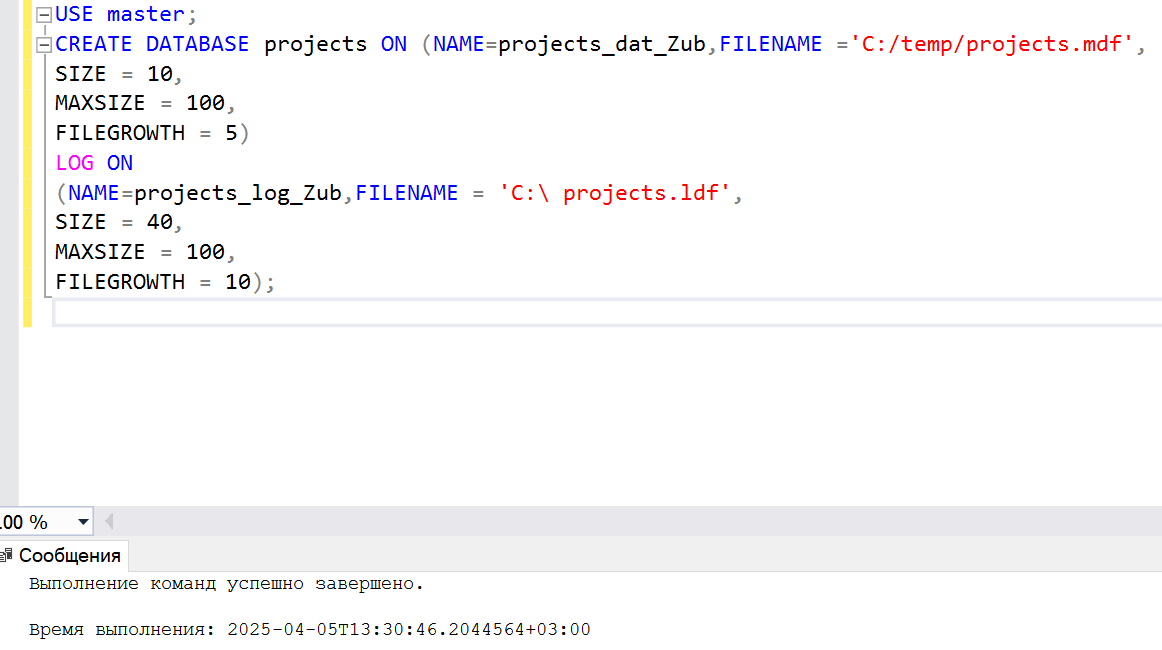


Рисунок 2 - Создание БД с передачей измененными параметрами по умолчанию

**Пример 3. Создание моментального снимка базы данных sampleNew\_Zub**

USE master;

CREATE DATABASE sample\_snapshot ON (NAME = ’sample\_Data’ ,

FILENAME = ’C:\temp\snapshot\_DB.mdf’)

AS SNAPSHOT OF sampleNew\_Zub;

Моментальный снимок существующей базы данных — это доступная только для чтения копия базы данных-источника, которая отражает состояние этой базы данных на момент копирования. Результат создания снапшота базы данных представлена на рисунке 3.



Рисунок 3 - создание снапшота БД

**Пример 4. Создание всех таблиц базы данных sampleNew**

USE sampleNew;

CREATE TABLE employee (emp\_no INTEGER NOT NULL,

empfname CHAR(20) NOT NULL,

emp\_lname CHAR(20) NOT NULL,

dept\_no CHAR(4) NULL);

CREATE TABLE department(dept\_no CHAR(4) NOT NULL,

dept\_name CHAR(25) NOT NULL,

location CHAR(30) NULL);

CREATE TABLE project (project\_no CHAR(4) NOT NULL,

project\_name CHAR(15) NOT NULL,

budget FLOAT NULL);

CREATE TABLE works\_on (emp\_no INTEGER NOT NULL,

project\_no CHAR(4) NOT NULL,

job CHAR (15) NULL,

enter\_date DATE NULL);

Выполнение скрипта представлено на

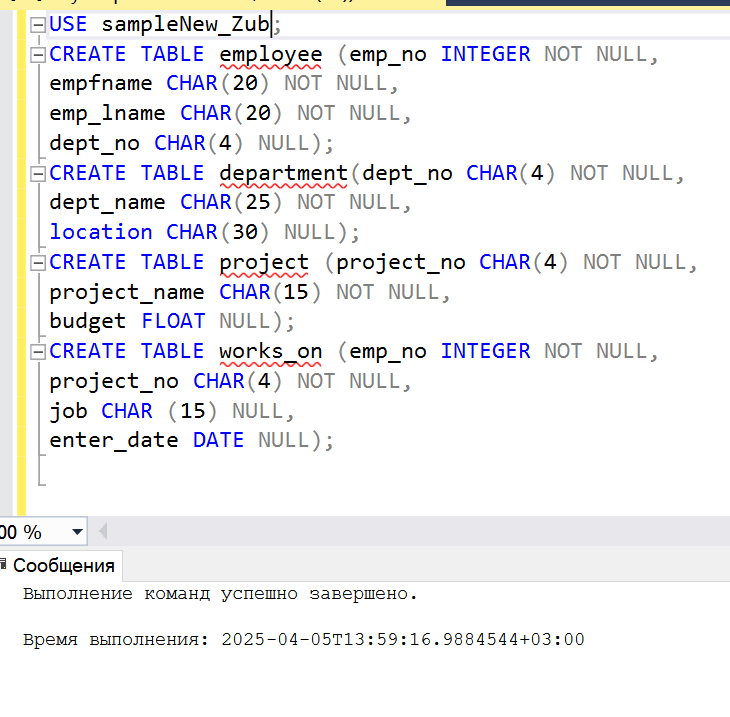


Рисунок 4 - Создание структуры БД sampleNew\_Zub

**Пример 5 Создание таблицы, содержащей столбец типа sql\_variant**

USE project;

CREATE TABLE Item\_Attributes (

item\_id INT NOT NULL,

attribute NVARCHAR(30) NOT NULL,

value SQL\_VARIANT NOT NULL,

PRIMARY KEY (item\_id, attribute) )

В примере 5 создается таблица, содержащая столбец value, который имеет тип sql\_variant. Как рассматривалось в предыдущей работе, тип данных sql\_variant можно использовать для хранения значений разных типов одновременно, таких как числовые значения, строки и даты. Обратите внимание на то, что в примере 5 столбцу присваивается тип данных SQL\_VARIANT по той причине, что значения разных атрибутов могут быть разных типов данных. Например, для атрибута размера тип данных значения attribute будет целочисленным, а для атрибута имени — строковым. Прмер выполнения представлен на рисунке 5.

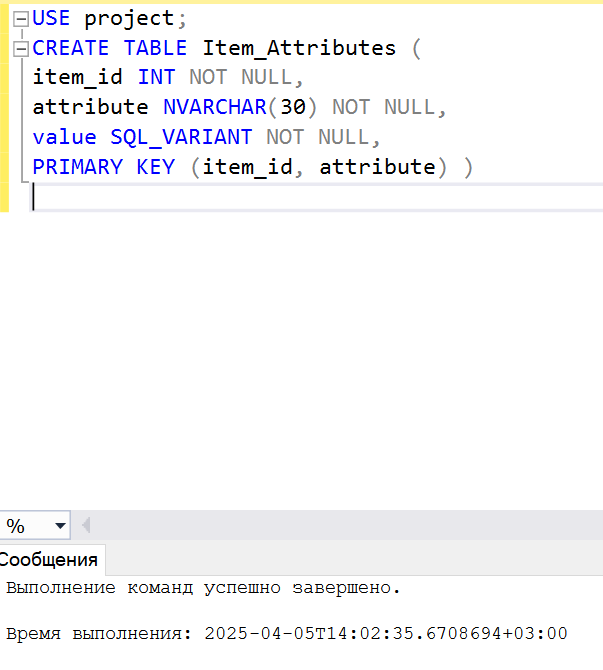


Рисунок 5 Создание таблицы, содержащей столбец типа sql\_variant

**Пример 6. Применение предложения unique**

USE sample;

CREATE TABLE projects (project\_no CHAR(4) DEFAULT ’p1’,

project\_name CHAR(15) NOT NULL,

budget FLOAT NULL

CONSTRAINT unique\_no UNIQUE (project\_no));

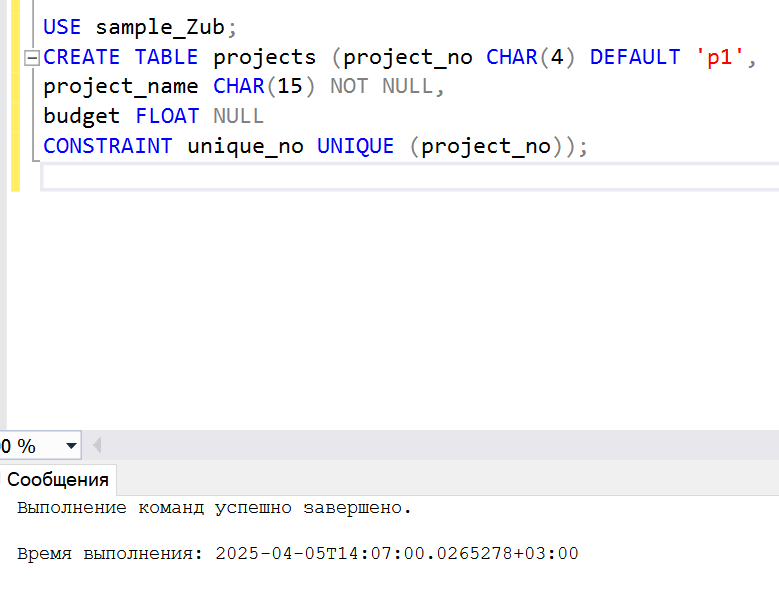


Рисунок 6 - Применение предложения unique

Каждое значение столбца project\_no таблицы projects является уникальным, включая значение NULL. (Точно так же, как и для любого другого значения с ограничением UNIQUE, если значения NULL разрешены для соответствующего столбца, этот столбец может содержать не более одной строки со значением NULL.) Попытка вставить в столбец project\_no уже имеющееся в нем значение будет неуспешной, т. к. система не примет его. Явное имя ограничения, определяемого в примере 6, — unique\_no. Результат выполнения скрипта представлен на рисунке 6.

**Пример 7. Определение первичного ключа**

USE sample;

CREATE TABLE employee (emp\_no INTEGER NOT NULL,

emp\_fname CHAR(20) NOT NULL, emp\_lname CHAR(20) NOT NULL, dept\_no CHAR(4) NULL,

CONSTRAINT prim\_empl PRIMARY KEY (emp\_no));

В результате выполнения кода в примере 7 снова создается таблица employee, в которой определен первичный ключ. Первичный ключ таблицы определяется посредством декларативного ограничения для обеспечения целостности с именем prim\_empl. Это ограничение для обеспечения целостности является ограничением уровня таблицы, поскольку оно указывается после определения всех столбцов таблицы employee. Реализация в SSMS представлена на рисунке 7.

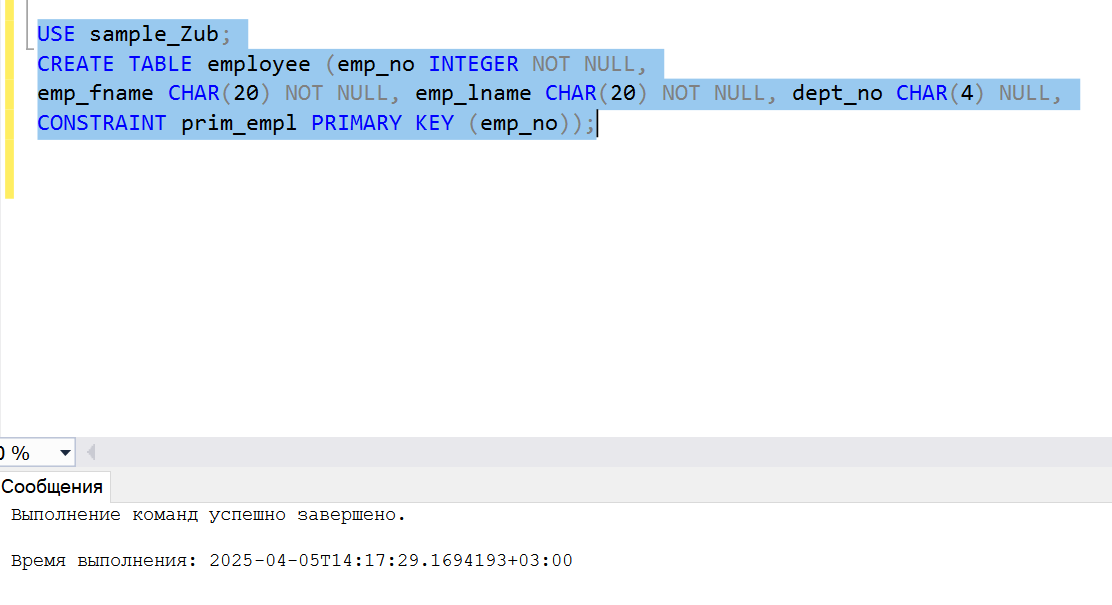


Рисунок 7 - Определение первичного ключа

**Пример 8. Определение ограничения уровня столбца**

USE sampleNew;

CREATE TABLE employee

(emp\_no INTEGER NOT NULL CONSTRAINT prim\_empl PRIMARY KEY,

emp\_fname CHAR(20) NOT NULL,

emp\_lname CHAR(20) NOT NULL,

dept\_no CHAR(4) NULL);

В примере 8 предложение primary key принадлежит к объявлению соответствующего столбца, наряду с объявлением его типа данных и свойства содержать значения null. По этой причине это ограничение называется ограничением на уровне столбца. Добавление ограничения представлено на рисунке 8.

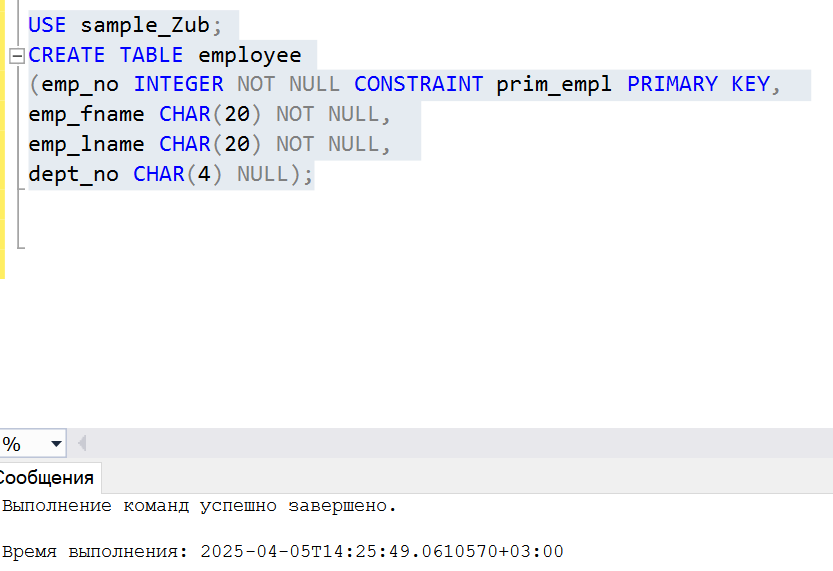


Рисунок 8 - Определение ограничения уровня столбца

**Пример 9. Применение предложения check**

USE sampleNew;

CREATE TABLE customer

(cust\_no INTEGER NOT NULL,

cust\_group CHAR(3) NULL,

CONSTRAINT ch\_cust\_gr CHECK (cust\_group IN ('c1', 'c2', 'c10')));

Создаваемая в примере 9 таблица customer включает столбец cust\_group, содержащий соответствующее проверочное ограничение. При вставке нового значения, отличающегося от значений в наборе ('cl', 'c2', 'c10'), или при попытке изменения существующего значения на значение, отличающегося от этих значений, система управления базой данных возвращает сообщение об ошибке. Пр менение параметра показано на рисунке 9.

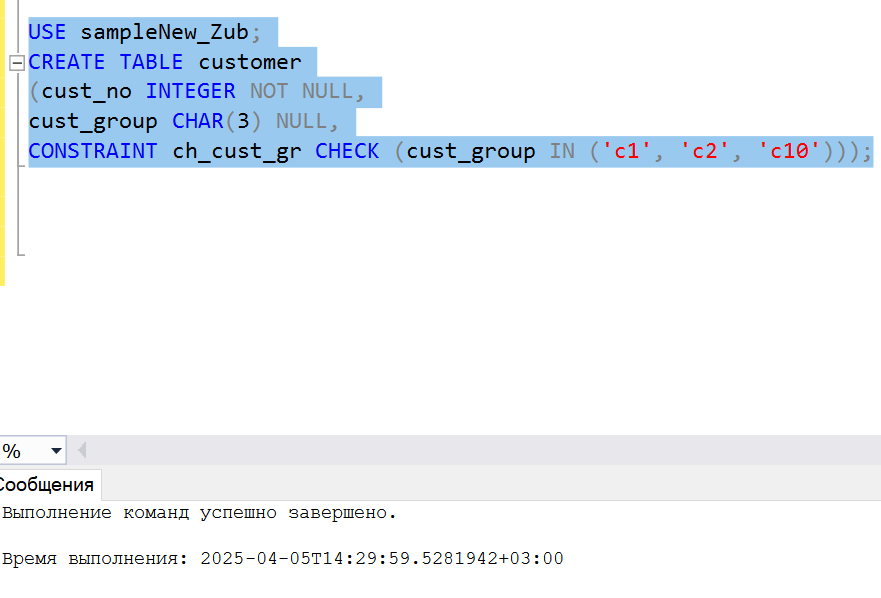


Рисунок 9 -Применение предложения check

**Пример 10. Объявление внешнего ключа**

USE sampleNew;

CREATE TABLE works\_on (emp\_no INTEGER NOT NULL,

project\_no CHAR(4) NOT NULL,

job CHAR (15) NULL,

enter\_date DATE NULL,

CONSTRAINT prim\_works PRIMARY KEY(emp\_no, project\_no),

CONSTRAINT foreign\_works FOREIGN KEY(emp\_no) REFERENCES employee (emp\_no));

Таблица works\_on в примере 10 задается с двумя декларативными ограничениями для обеспечения целостности: prim\_works и foreign\_works. Оба ограничения являются уровня таблицы, где первое указывает первичный ключ, а второе — внешний ключ таблицы works\_on. Кроме этого, ограничение foreign\_works определяет таблицу employee, как ссылочную таблицу, а ее столбец emp\_no, как соответствующий первичный ключ столбца с таким же именем в таблице works\_on. Реализация скрипта показана на рисунке 10.

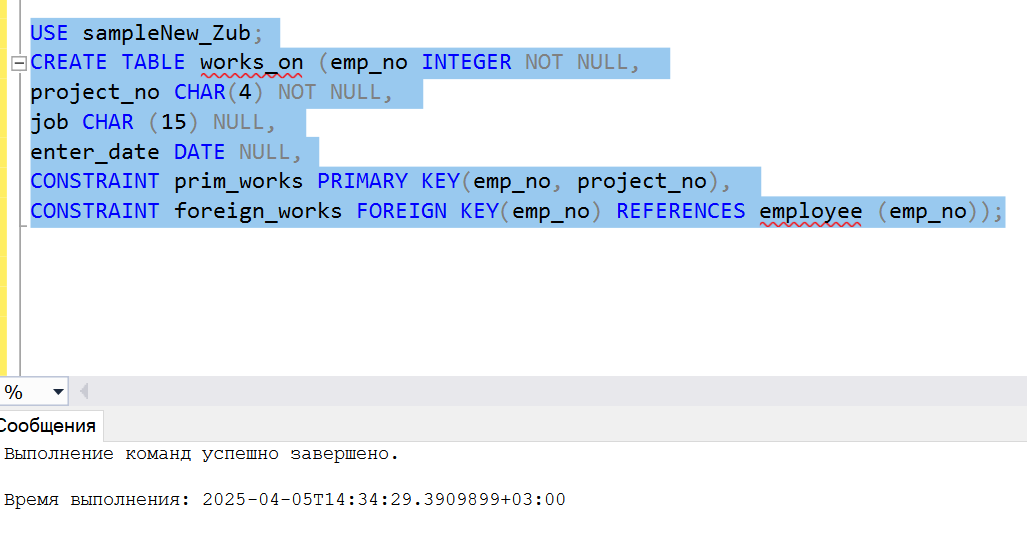


Рисунок 10 - Объявление внешнего ключа

**Пример 11. Определение всех ограничений первичного и внешнего ключей для таблиц базы данных sampleNew**

USE sampleNew;

CREATE TABLE department(dept\_no CHAR(4) NOT NULL,

dept\_name CHAR(25) NOT NULL,

location CHAR(30) NULL,

CONSTRAINT prim\_dept PRIMARY KEY (dept\_no));

CREATE TABLE employee (emp\_no INTEGER NOT NULL,

emp\_fname CHAR(20) NOT NULL,

emp\_lname CHAR(20) NOT NULL,

dept\_no CHAR(4) NULL,

CONSTRAINT prim\_emp PRIMARY KEY (emp\_no),

CONSTRAINT foreign\_emp FOREIGN KEY(dept\_no) REFERENCES department(dept\_no));

CREATE TABLE project (project\_no CHAR(4) NOT NULL,

project\_name CHAR(15) NOT NULL,

budget FLOAT NULL,

CONSTRAINT prim\_proj PRIMARY KEY (project\_no));

CREATE TABLE works\_on (emp\_no INTEGER NOT NULL,

project\_no CHAR(4) NOT NULL,

job CHAR (15) NULL,

enter\_date DATE NULL,

CONSTRAINT prim\_works PRIMARY KEY(emp\_no, project\_no),

CONSTRAINT foreign1\_works FOREIGN KEY(emp\_no) REFERENCES employee(emp no),

CONSTRAINT foreign2\_works FOREIGN KEY(project\_no) REFERENCES project(project\_no));

Результат работы скрипта представлен на рисунке 11.

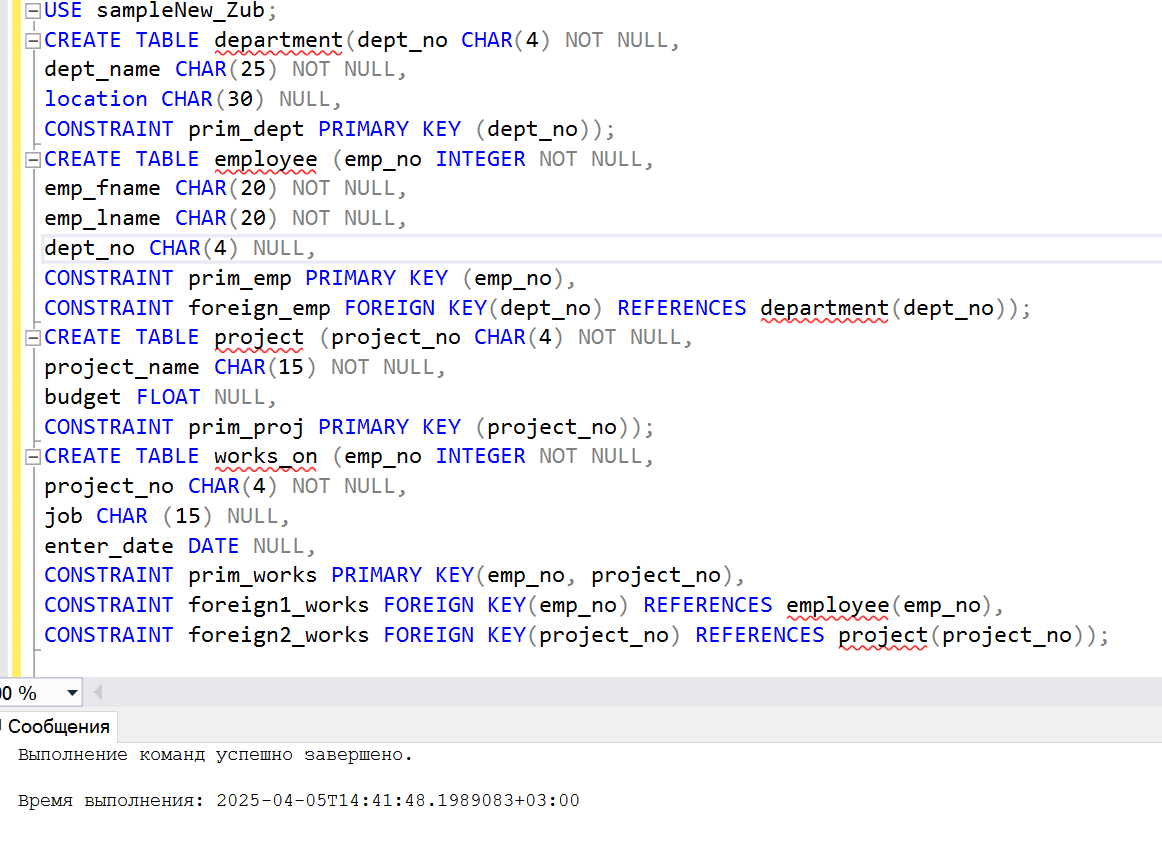


Рисунок 11 - Определение всех ограничений первичного и внешнего ключей для таблиц базы данных sampleNew\_Zub

**Пример 12. Применение опций on delete и on update**

USE sampleNew;

CREATE TABLE works\_on1

(emp\_no INTEGER NOT NULL,

project\_no CHAR(4) NOT NULL,

job CHAR (15) NULL,

enter\_date DATE NULL,

CONSTRAINT prim\_works1 PRIMARY KEY(emp\_no, project\_no),

CONSTRAINT foreign1\_works1 FOREIGN KEY(emp\_no)

REFERENCES employee(emp\_no) ON DELETE CASCADE,

CONSTRAINT foreign2\_works1 FOREIGN KEY(project\_no)

REFERENCES projact(project\_no) ON UPDATE CASCADE);

В примере 12 создается таблица works\_on1 с использованием опций ON DELETE CASCADE и ON UPDATE CASCADE. Если таблицу works\_on1 загрузить значениями из табл. 1.4, каждое удаление строки в таблице employee будет вызывать каскадное удаление всех строк в таблице works\_on1, которые имеют значения внешнего ключа, соответствующие значениям первичного ключа строк, удаляемых в таблице employee. Подобным образом каждое обновление значения столбца project\_no таблицы project будет вызывать такое же обновление всех соответствующих значений столбца project\_no таблицы works\_on1. Результат отображен на рисунке*12*.

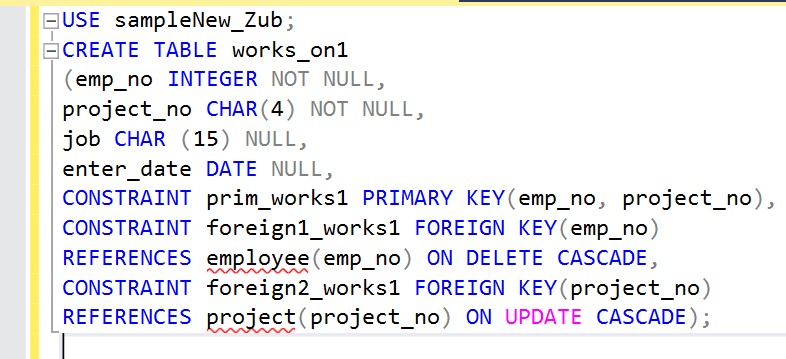


Рисунок 12 - Применение опций on delete и on update

**Пример 14. Создание псевдонимного типа данных С ПОМОЩЬЮ инструкции CREATE TYPE**

USE sampleNew;

CREATE TYPE zip

FROM SMALLINT NOT NULL;

В примере 14 создается псевдонимный тип данных zip на основе стандартного типа данных SMALLINT. Результат представлен на рисунке 13.

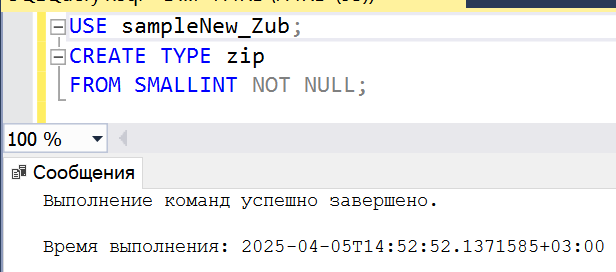


Рисунок 13 - Создание псевдонимного типа данных С ПОМОЩЬЮ инструкции CREATE TYPE

**Пример 15. Определение столбца псевдонимным типом данных**

USE sampleNew;

CREATE TABLE customer

(cust\_no INT NOT NULL,

cust\_name CHAR(20) NOT NULL,

city CHAR(20),

zip\_code ZIP,

CHECK (zip\_code BETWEEN 601 AND 99950));

В примере 15 тип данных столбца zip\_code таблицы customer определяется псевдонимным типом данных zip. Допустимые значения этого столбца требуется ограничить диапазоном целочисленных значений от 601 до 99950. Как можно видеть в примере 15, это ограничение можно наложить с помощью предложения check. Результат работы скрипта показан на рисунке 14.

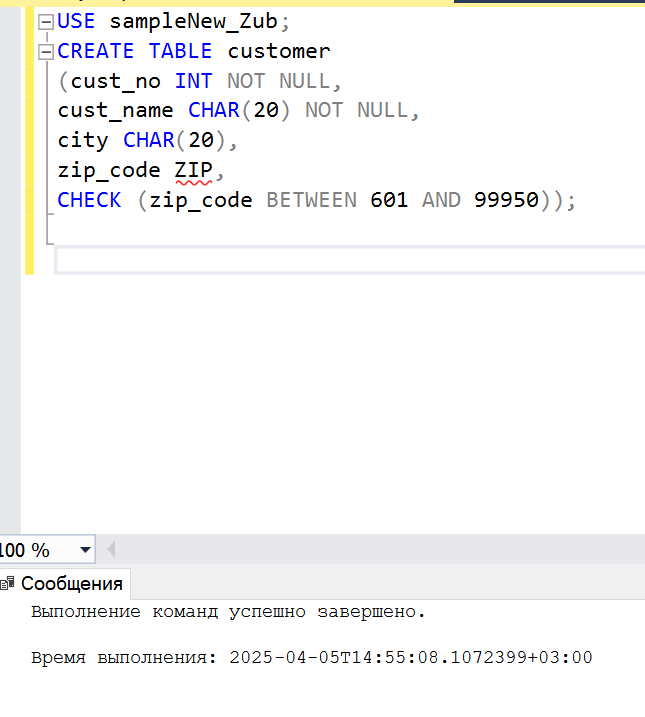


Рисунок 14 - Определение столбца псевдонимным типом данных

**Пример 16. Создание табличного типа**

USE sampleNew;

CREATE TYPE person\_table\_t AS TABLE

(name VARCHAR(30),

salary DECIMAL(8,2));

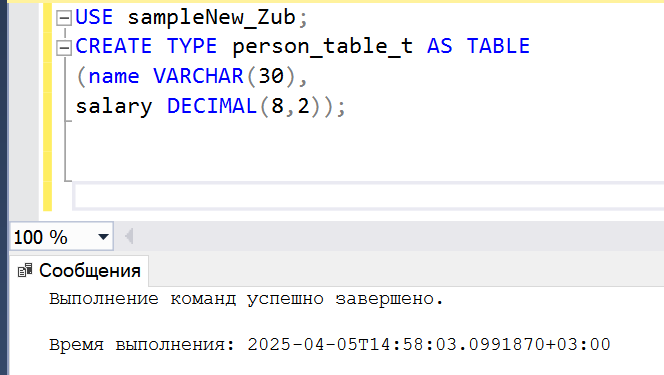


Рисунок 15 - Создание табличного типа

Создаваемый в примере 16 определяемый пользователем табличный тип данных person\_table\_t имеет два столбца: name и salary. Основное синтаксическое отличие табличных типов от псевдонимных состоит в наличии предложения as table, как это можно видеть в примере 15. Определяемые пользователем табличные типы обычно применяются с возвращающими табличные значения параметрами. Результат отображен на рисунке 15.

**Пример 17. Добавление нового файла в базу данных**

USE master;

GO

ALTER DATABASE projects ADD FILE (NAME=projects\_dat1,

FILENAME = 'C:\projects1.mdf',

SIZE = 10,

MAXSIZE = 100, FILEGROWTH = 5);

В примере 17 инструкция alter database добавляет новый файл с логическим именем projects\_dat1. Здесь же указан начальный размер файла 10 Мбайт и автоувеличение по 5 Мбайт до максимального размера 100 Мбайт. Файлы журналов добавляются так же, как и файлы баз данных. Единственным отличием является то, что вместо предложения add file используется предложение add log file. Результат отображен на рисунке 16.

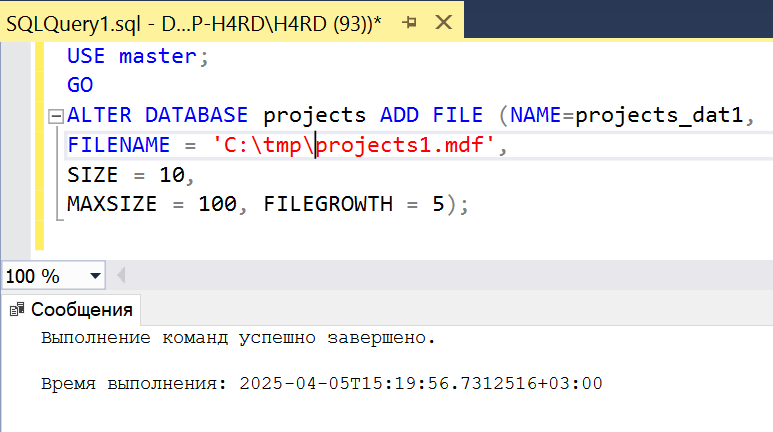


Рисунок 16 - Добавление нового файла в базу данных

Пример 21. Добавление нового столбца в таблицу

USE sampleNew;

ALTER TABLE employee

ADD telephone\_no CHAR(12) NULL;

В примере 21 инструкция alter table добавляет в таблицу employee столбец telephone\_no. Компонент Database Engine заполняет новый столбец значениями null или identity или указанными значениями по умолчанию. По этой причине новый столбец должен или поддерживать свойство содержать значения NULL, или для него должно быть указано значение по умолчанию. Пример представлен на рисунке 17.

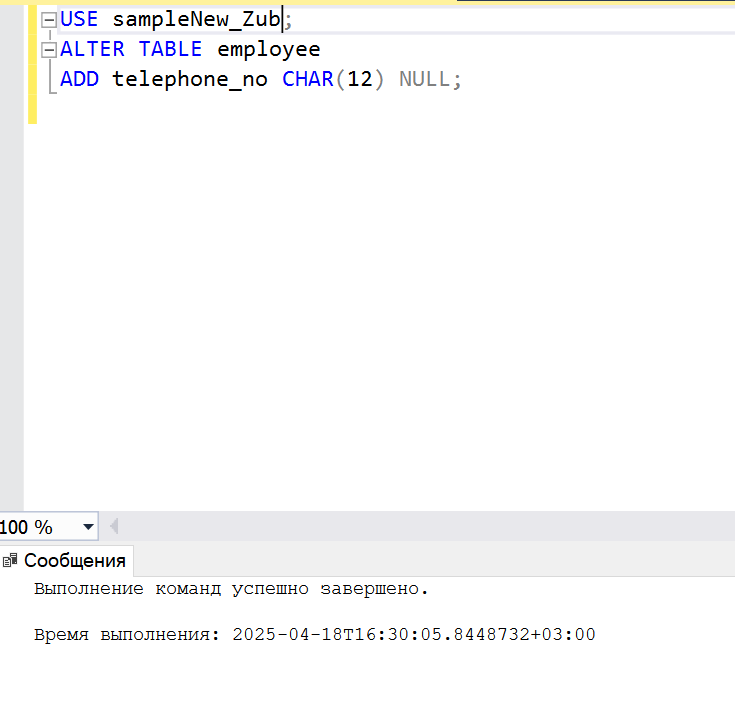


Рисунок 17- Добавление нового столбца в таблицу

Пример 22. Удаление столбца таблицы

USE sampleNew;

ALTER TABLE employee

DROP COLUMN telephone\_no;

В примере 22 инструкция alter table удаляет в таблице employee столбец telephone\_no, который был добавлен в эту таблицу предложением add в примере 21. Реализация на рисунке 18.

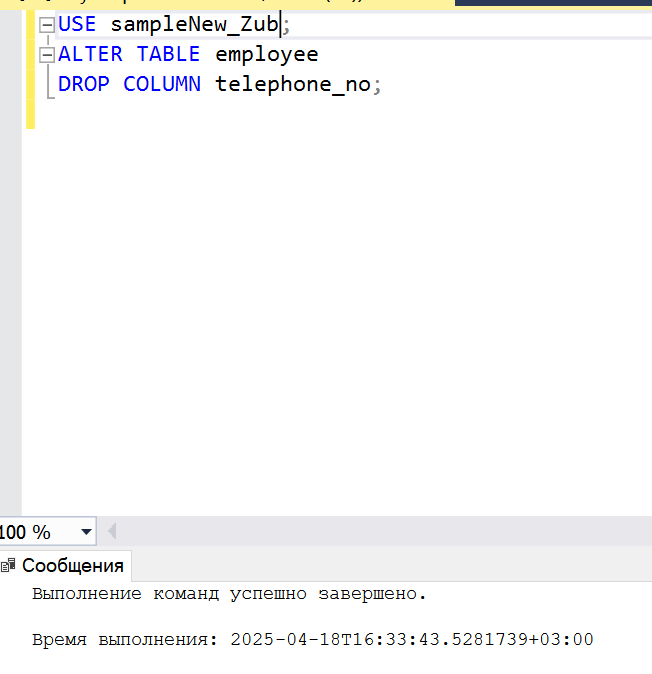


Рисунок 18 - Удаление столбца таблицы

**Пример 23. Изменение свойств столбца**

USE sampleNew;

ALTER TABLE department

ALTER COLUMN location CHAR(25) NOT NULL;

В примере 23 инструкция alter table изменяет начальные свойства (char(30), значения null разрешены) столбца location таблицы department на новые (char(25) , значения null не разрешены — not null). Реализация на рисунке 19.

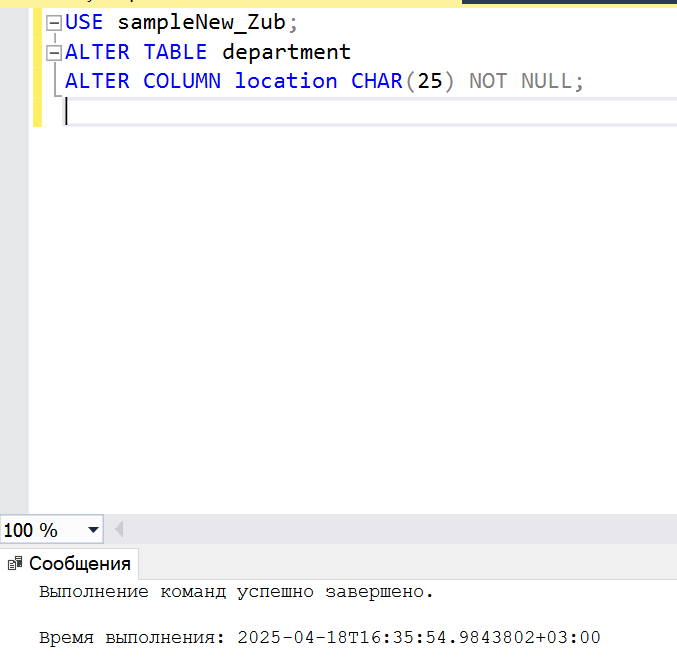


Рисунок 19 - Изменение свойств столбца

Пример 24. Добавление проверочного ограничения

USE sampleNew;

CREATE TABLE sales

(order\_no INTEGER NOT NULL,

order\_date DATE NOT NULL,

ship\_date DATE NOT NULL);

ALTER TABLE sales

ADD CONSTRAINT order\_check CHECK(order\_date <= ship\_date);

В примере 24 инструкцией create table сначала создается таблица sales, содержащая два столбца с типом данных date: order\_date и ship\_date. Далее, инструкция alter table определяет ограничение для обеспечения целостности order\_check, которое сравнивает значения обоих этих столбцов и выводит сообщение об ошибке, если дата отправки ship\_date более ранняя, чем дата заказа order\_date. Результат на рисунке 20.

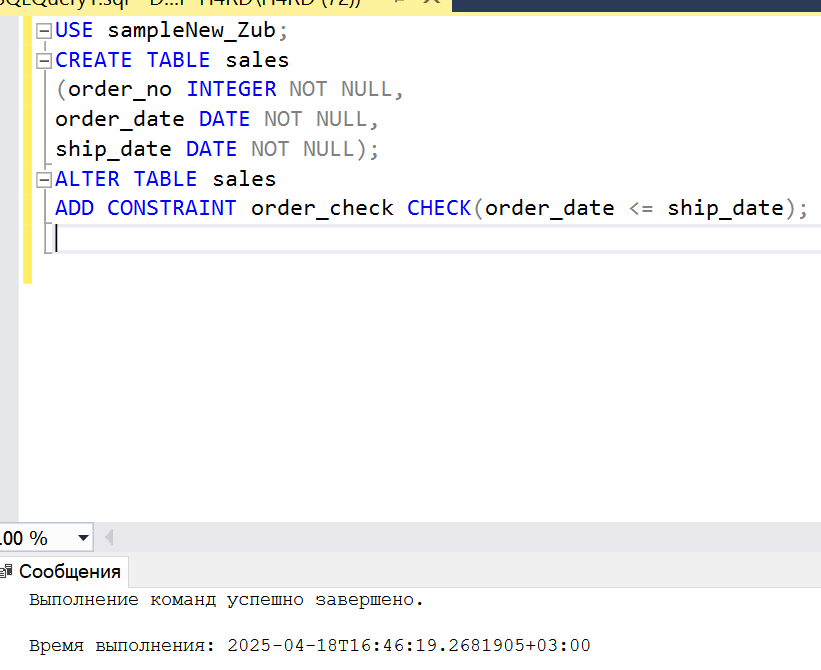


Рисунок 20 - Добавление проверочного ограничения

**Пример 25. Определение первичного ключа таблицы**

USE sampleNew;

ALTER TABLE sales

ADD CONSTRAINT primaryk\_sales PRIMARY KEY(order\_no);

В примере 25 столбец order\_no определен, как первичный ключ таблицы sales. Результат на рисунке 21.

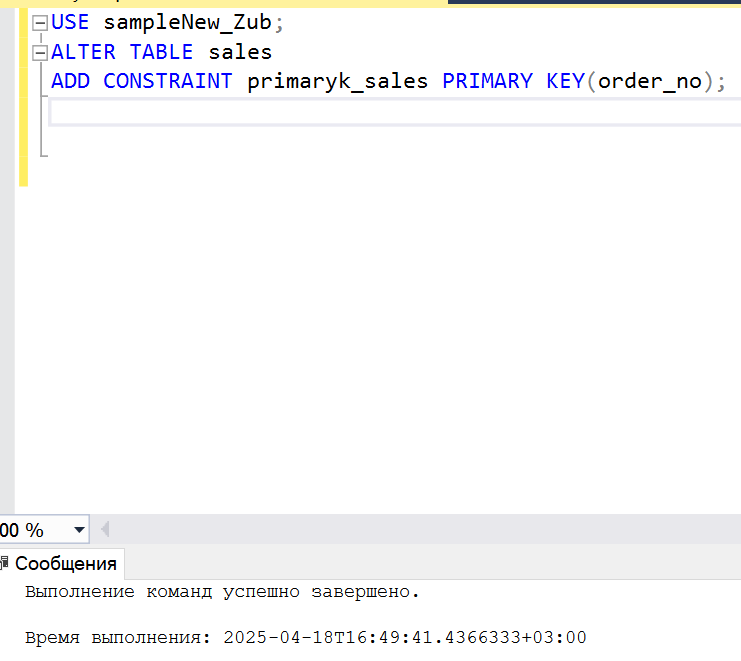


Рисунок 21 - Определение первичного ключа таблицы

Пример 26. Удаление ограничений для обеспечения целостности

USE sampleNew;

ALTER TABLE sales

DROP CONSTRAINT order check;

В примере 26 инструкция alter table удаляет проверочное ограничение order\_check, установленное в примере 24. Результат на рисунке 22.

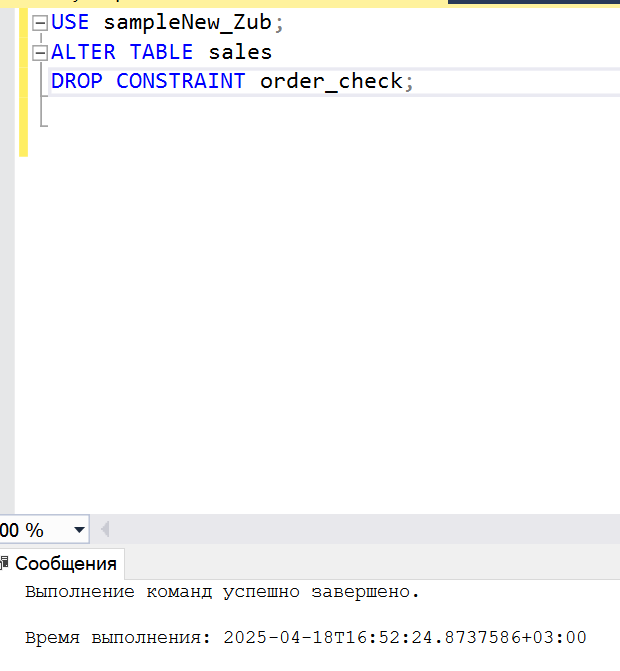


Рисунок 22 - Удаление ограничений для обеспечения целостности

**Пример 27. Отключение ограничений таблицы**

USE sampleNew;

ALTER TABLE sales NOCHECK CONSTRAINT ALL;

В примере 27 все ограничения таблицы sales отключаются посредством ключевого слова ALL. Результат на рисунке 23

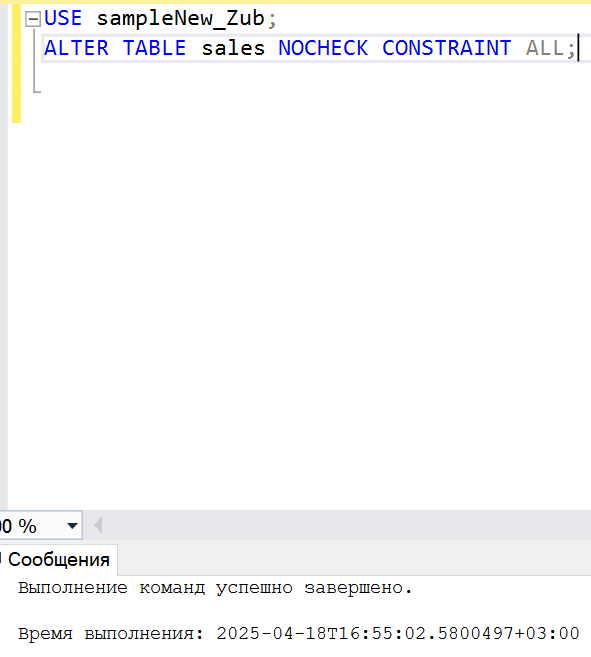


Рисунок 23 - Отключение ограничений таблицы

**Пример 28. Переименование таблицы**

USE sampleNew;

EXEC sp\_rename @objname = department, @newname = subdivision

В примере 28 таблице department присваивается новое имя subdivision. Результат на рисунке 24.

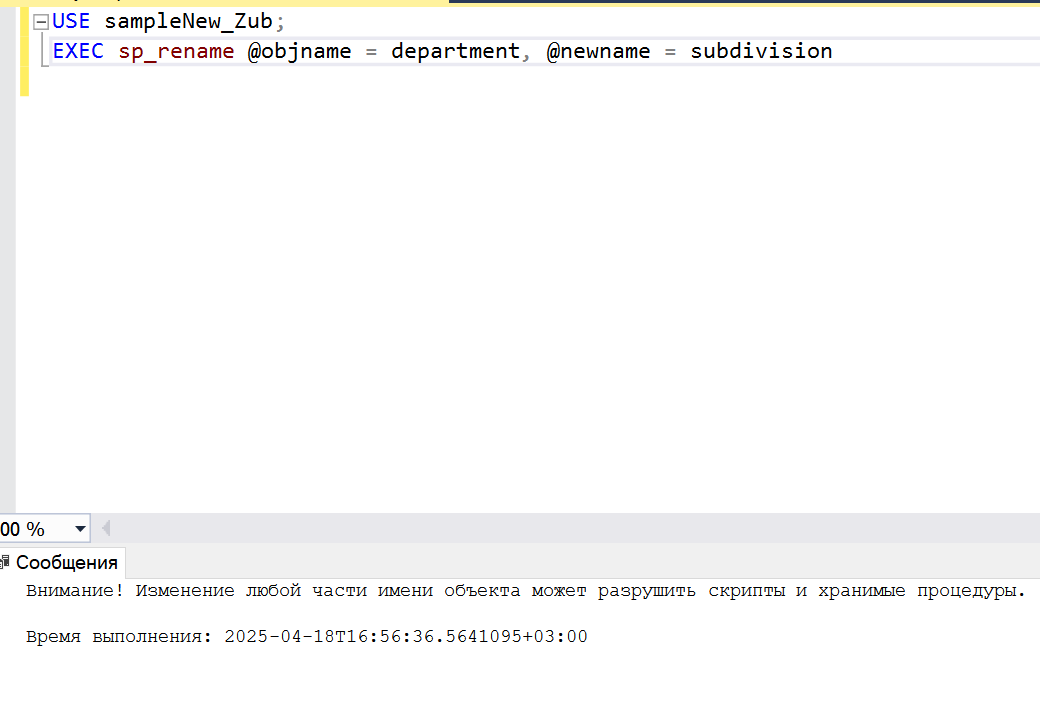


Рисунок 24 - Переименование таблицы

**Пример 29. Переименование столбца таблицы**

USE sampleNew;

EXEC sp\_rename @objname = 'sales.order\_no', Snewname = ordernumber

В примере 29 столбцу order\_no таблицы sales присваивается новое имя ordernumber. При переименовании столбца таблицы имя этого столбца требуется указывать в виде: table\_name. column\_name (т. е. имя\_таблицы.имя\_столбца). Результат на рисунке 2.

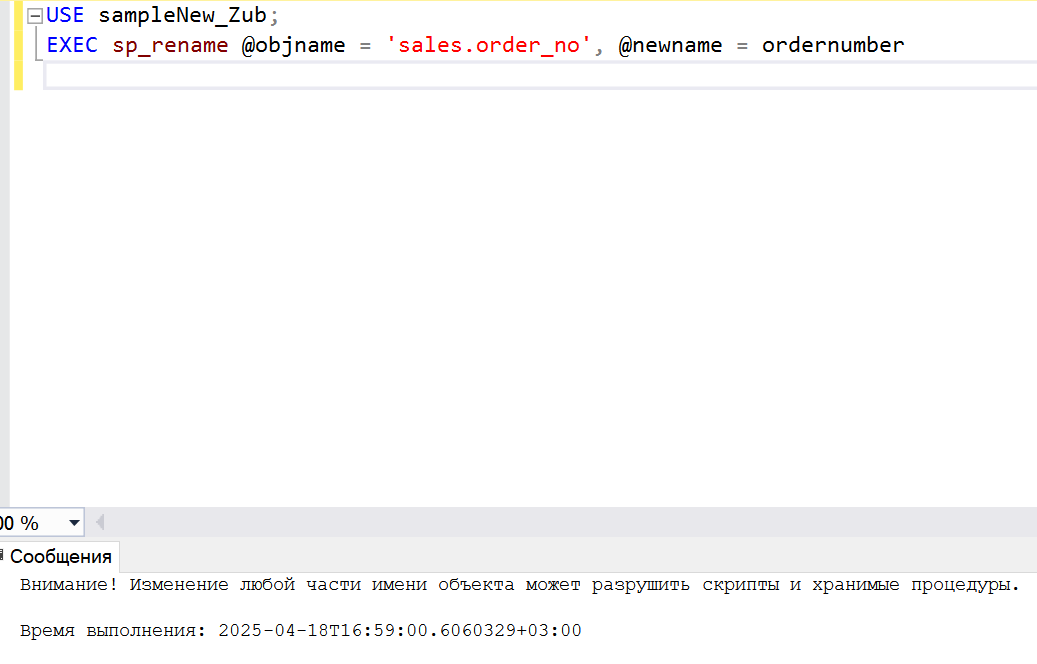


Рисунок 25 - Переименование столбца таблицы

Упражнения

**Упражнение 1**

Используя инструкцию CREATE DATABASE, создайте новую базу данных test\_db, задав явные спецификации для файлов базы данных и журнала транзакций. Файл базы данных с логическим именем test\_db\_dat сохраняется в физическом файле C:\tmp\test\_db.mdf, его начальный размер — 5 Мбайт, автоувеличение по 8%, максимальный размер не ограничен. Файл журнала транзакций с логическим именем test\_db\_log сохраняется в физическом файле C:\tmp\test\_db\_log.ldf, его начальный размер — 2 Мбайта, автоувеличение по 500 Кбайт, максимальный размер 10 Мбайт.

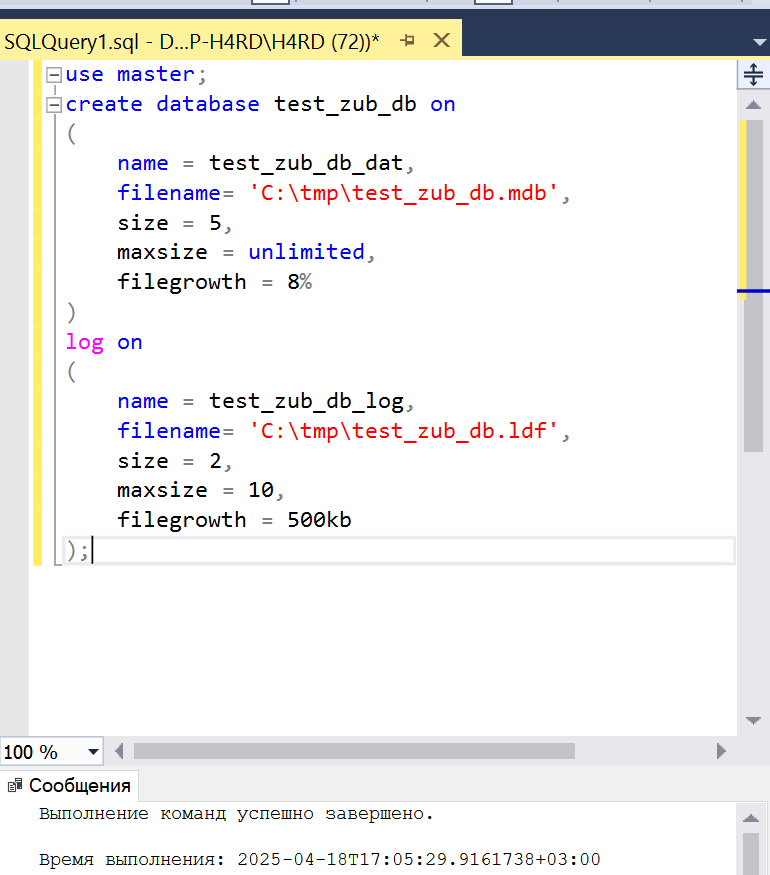


Рисунок 26 - код упражнения 1

**Упражнение 2**

Используя инструкцию alter database, добавьте новый файл журнала в базу данных test\_db. Файл сохраняется в физическом файле C:\tmp\emp\_log.ldf, его начальный размер — 2 Мбайта, автоувеличение по 2 Мбайта, максимальный размер не ограничен.

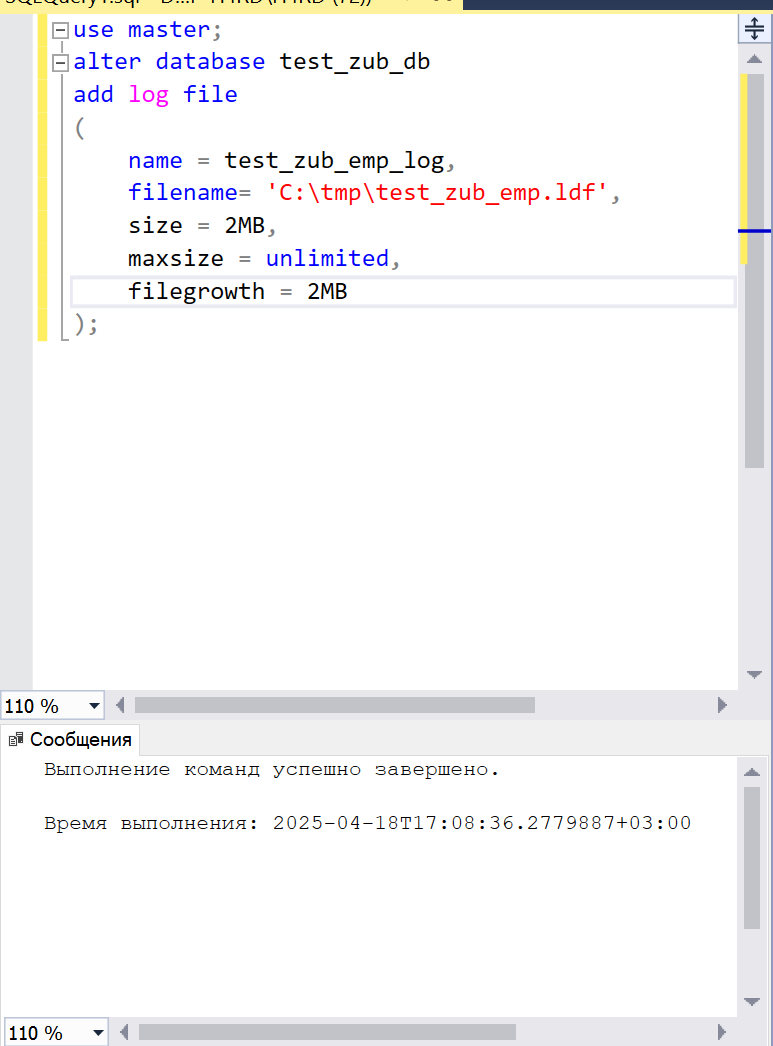


Рисунок 27 - код упражнения 2

**Упражнение 3**

Используя инструкцию alter database, измените начальный размер файла базы данных test\_db на 10 Мбайт.

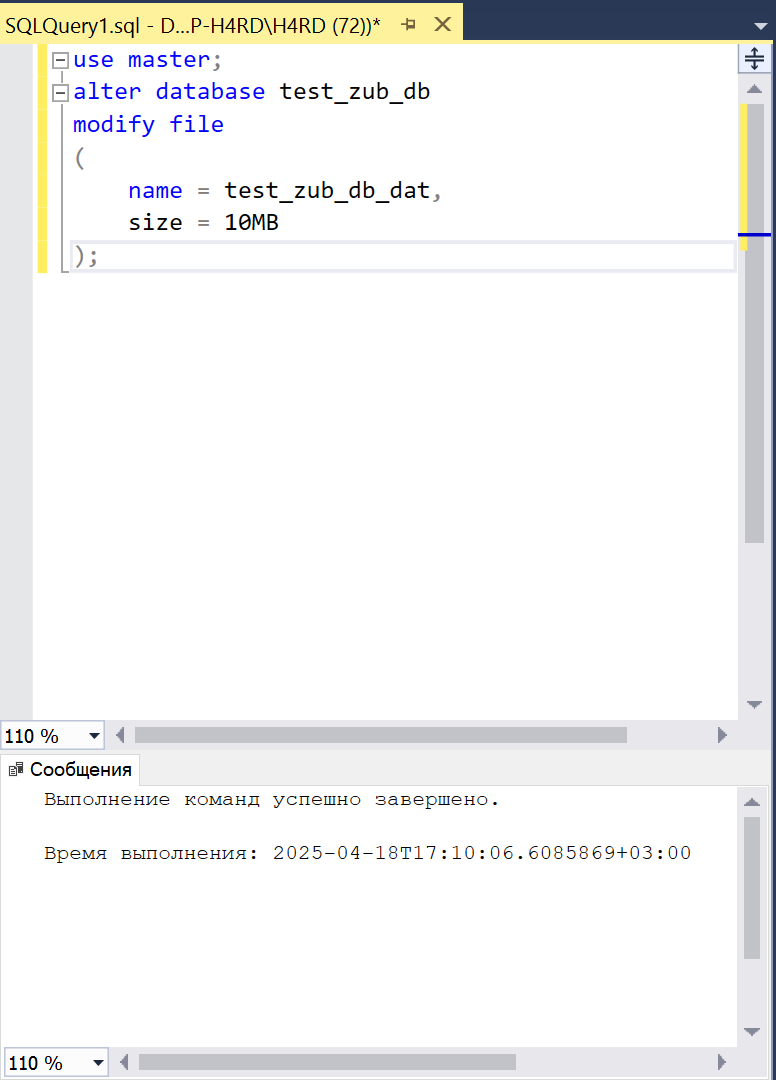


Рисунок 28 - код упражнения 4

**Упражнение 4**

В примере 4 для некоторых столбцов четырех созданных таблиц запрещены значения NULL. Для каких из этих столбцов это определение является обязательным, а для каких нет?

У сотрудника обязан быть номер в компании и ФИО, а вот номер отдела может быть пустым (когда, например сотрудник самозанятый).

**Упражнение 5**

Почему в примере 4 тип данных для столбцов dept\_no и project\_no определен как char, а не как один из целочисленных типов?

Так как и номер отдела ка ки номер проекта может быть определен как идентификатор с содержанием букв.

**Упражнение 6**

Создайте таблицы customers и orders, содержащие перечисленные в следующей таблице столбцы. Не объявляйте соответствующие первичный и внешние ключи.

|  |  |
| --- | --- |
| customers | orders |
| customerid char(5) not null | orderid integer not null |
| companyname varchar(40) not null | customerid char(5) not null |
| contactname char(30) null | orderdate date null |
| address varchar(60) null | shippeddate date null |
| city char(15) null | freight money null |
| phone char(24) null | shipname varchar(40) null |
| fax char(24) null | shipaddress varchar(60) null |
|  | quantity integer null |

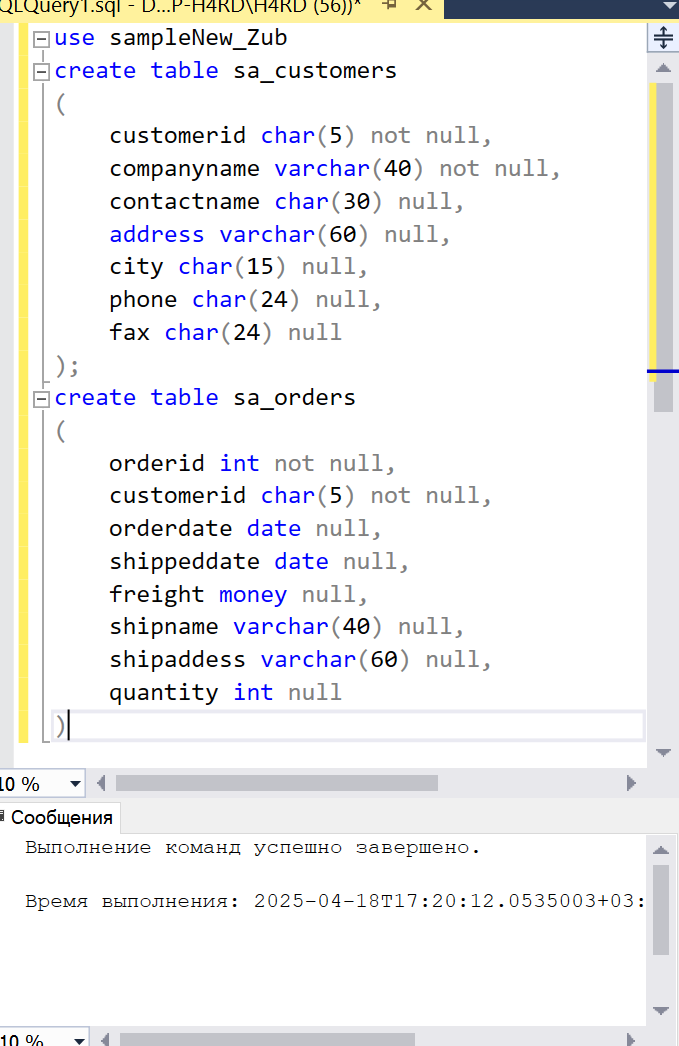


Рисунок 29 - код упражнения 6

**Упражнение 7**

Используя инструкцию alter table, добавьте в таблицу orders новый столбец shipregion. Столбец должен иметь целочисленный тип данных и разрешать значения NULL.

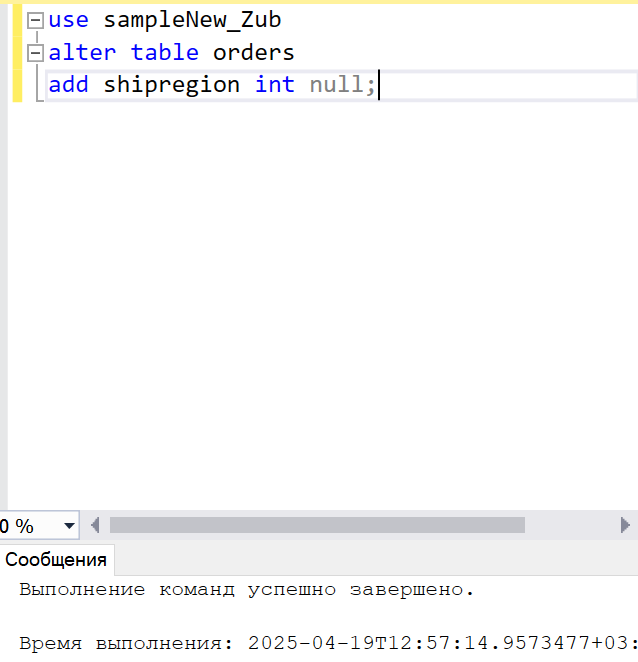


Рисунок 30 - код упражнения 7

**Упражнение 8**

Используя инструкцию ALTER TABLE, измените тип данных столбца shipregion с целочисленного на буквенно-цифровой длиной 8 символов. Столбец может содержать значения NULL.

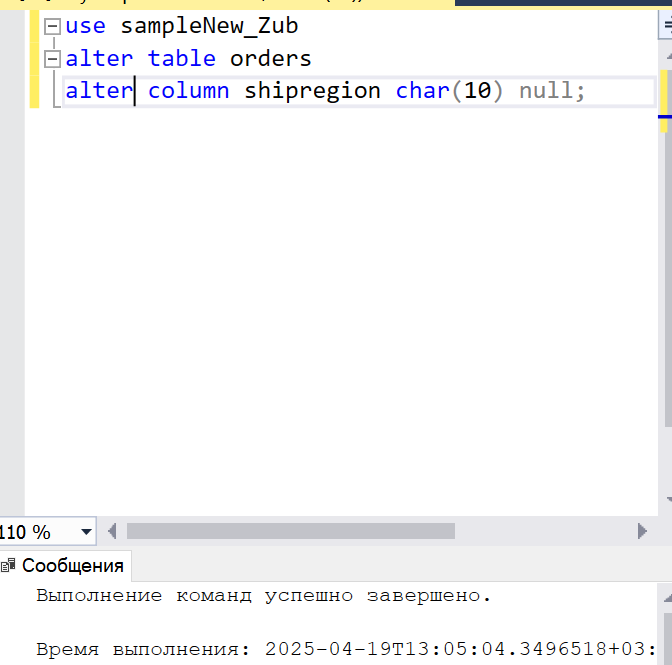


Рисунок 31 - код упражнения 8

**Упражнение 9**

Удалите созданный ранее столбец shipregion.

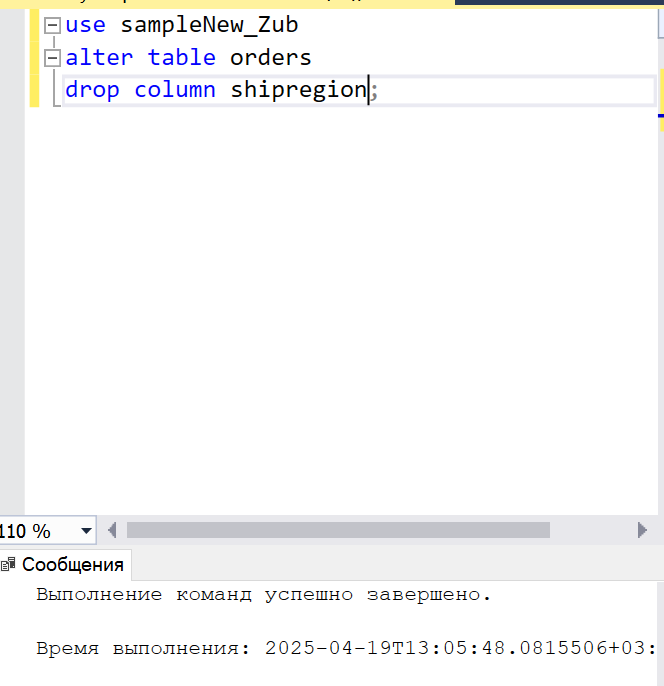


Рисунок 32 - код упражнения 9

**Упражнение 10**

Дайте точное описание происходящему при удалении таблицы с помощью инструкции DROP TABLE.

1. Проверка существования таблицы
2. Проверка зависимостей и ограничений
3. Физическая очистка с диска
4. Удаление метаданных
5. Фиксация изменений

**Упражнение 11**

Создайте заново таблицы customers и orders, усовершенствовав их определение всеми ограничениями первичных и внешних ключей.

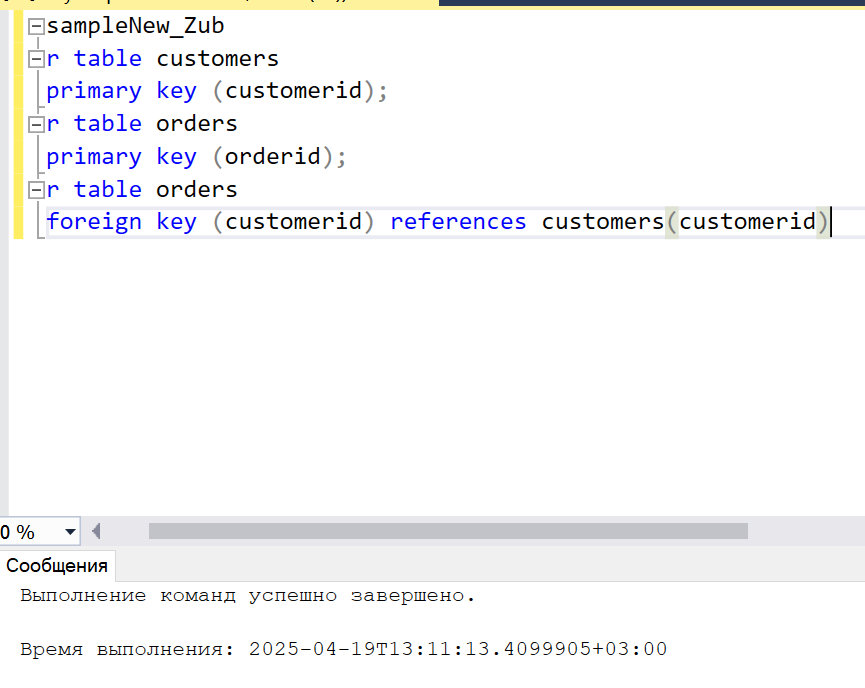


Рисунок 33 - код упражнения 11

**Упражнение 12**

Используя среду SQL Server Management Studio, попробуйте вставить следующую новую строку в таблицу orders:

(10, 'ord0l', getdate(), getdate(), 100.0, 'Windstar', 'Ocean', 1).

Почему система отказывается вставлять эту строку в таблицу?

Так как нарушается ограничение целостности данных. Данные для вставки ссылаются на несуществующие данные в таблице customer.

**Упражнение 13**

Используя инструкцию alter table, определите значение по умолчанию столбца orderdate таблицы orders в виде текущей даты и времени системы.

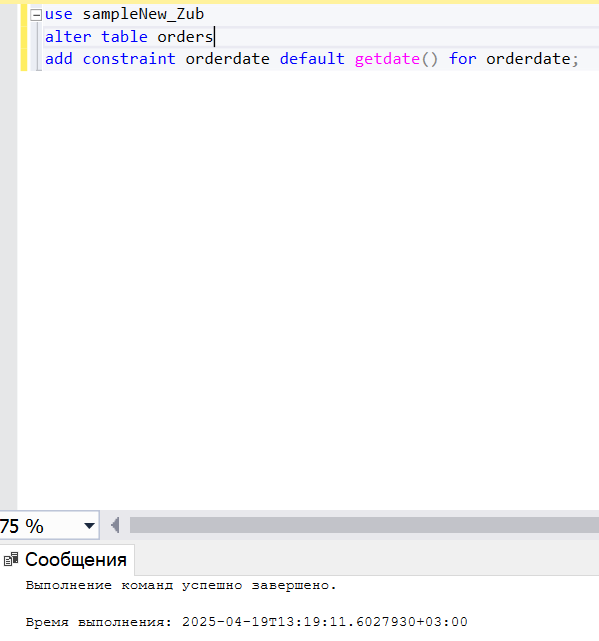


Рисунок 34 - код упражнения 13

**Упражнение 14**

Используя инструкцию alter table, создайте ограничение для обеспечения целостности, ограничивающее допустимые значения столбца quantity таблицы orders диапазоном значений от 1 до 30.

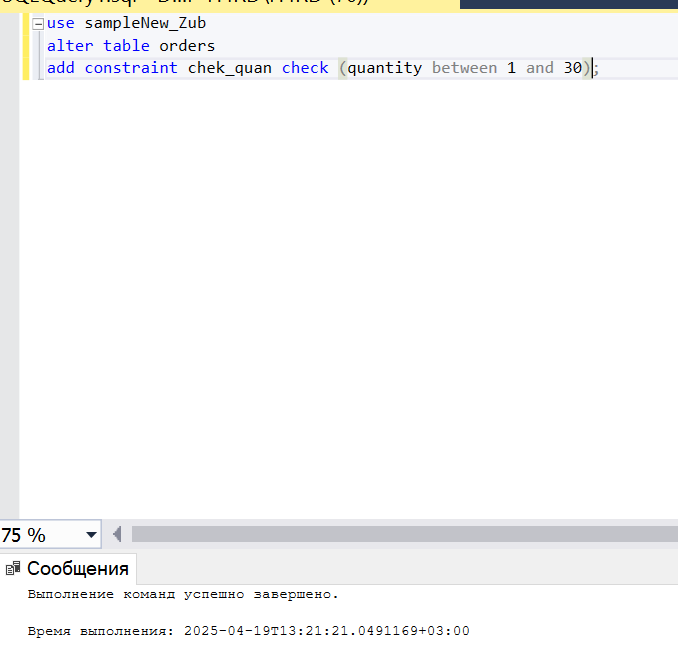


Рисунок 35 - код упражнения 14

**Упражнение 15**

Отобразите все ограничения для обеспечения целостности таблицы orders.

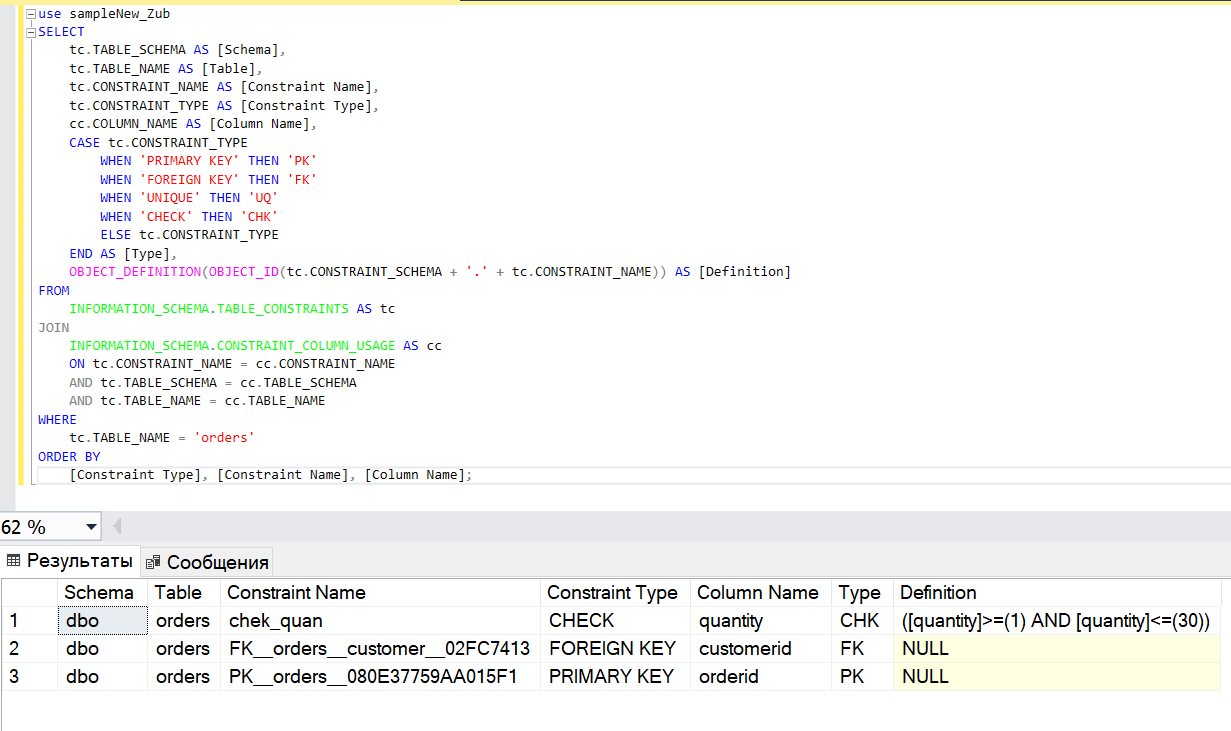


Рисунок 36 - код упражнение 15

**Упражнение 16**

Попытайтесь удалить первичный ключ таблицы customers. Почему это не удается?

На первичный ключ таблицы customers ссылается вторичный ключ таблицы orders. Для обеспечения целостности данных СУБД блокирует выполнение такого запроса.

**Упражнение 17**

Удалите ограничение для обеспечения целостности prim\_empl, определенное в примере 7.

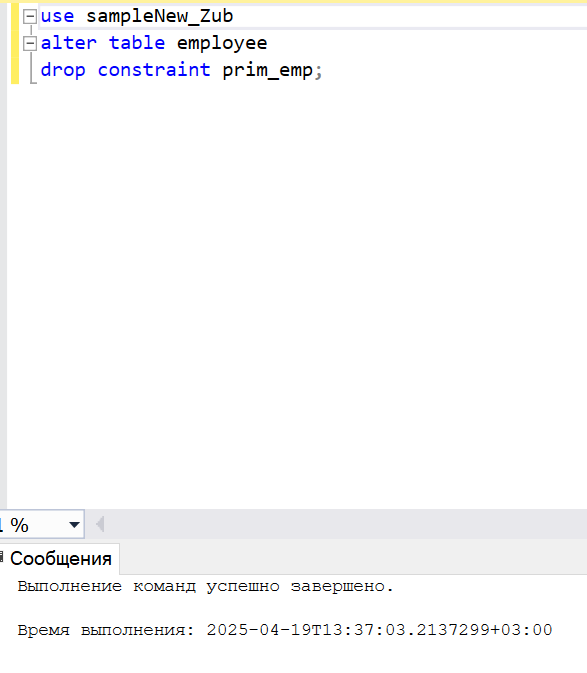


Рисунок 37 - код упражнения 17

**Упражнение 18**

В таблице customers измените имя столбца city на town.

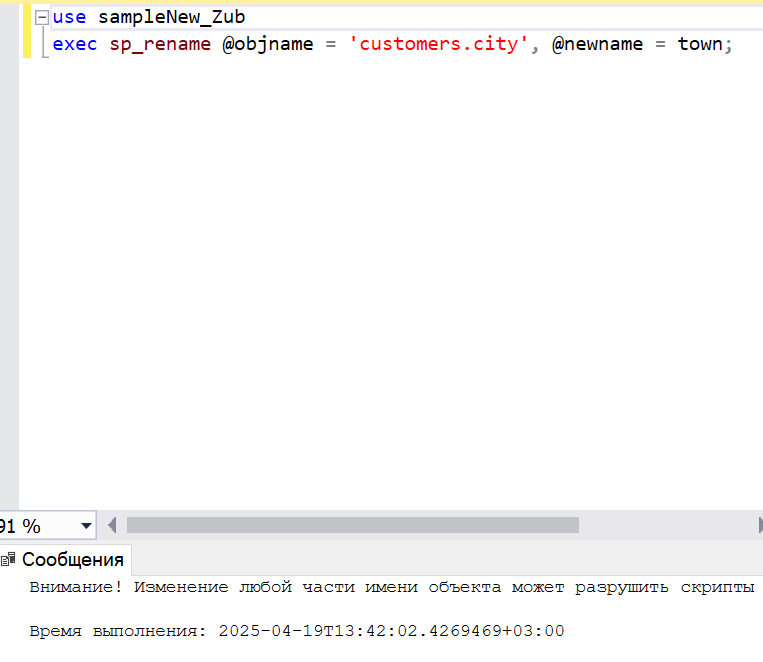


Рисунок 38 - код упражнения 18