**Создание объектов базы данных**

Transact-SQL могут быть написаны и пересланы компоненту Компонент Database Engine следующими способами:

* При помощи среды SQL Server Management Studio. Предполагается, что вы используете среду Среда Management Studio, но можно также использовать среду Среда Management Studio Express, которая может быть загружена бесплатно с веб-узла [центра загрузки Майкрософт](https://www.microsoft.com/download/details.aspx?id=14630).
* Посредством программы [sqlcmd](https://docs.microsoft.com/ru-ru/sql/tools/sqlcmd-utility?view=sql-server-ver15).
* Соединившись из создаваемого приложения.

Исходный код исполняется в компоненте Компонент Database Engine таким же образом и с теми же разрешениями, независимо от того, как был передан исходный код инструкций.

Чтобы выполнить инструкцию языка Transact-SQL в среде Среда Management Studio, откройте среду Среда Management Studio и соединитесь с экземпляром компонента Компонент SQL Server Database Engine.

## предварительные требования

Для работы с этим руководством необходима среда SQL Server Management Studio и доступ к экземпляру SQL Server.

* Установите [SQL Server Management Studio](https://docs.microsoft.com/sql/ssms/download-sql-server-management-studio-ssms).

Если у вас нет экземпляра SQL Server, создайте его. Чтобы создать экземпляр, выберите свою платформу по следующим ссылкам. При выборе проверки подлинности SQL используйте учетные данные SQL Server.

* **Windows**: [скачать выпуск SQL Server 2017 Developer Edition](https://www.microsoft.com/sql-server/sql-server-downloads).
* **macOS**: [скачать SQL Server 2017 для Docker](https://docs.microsoft.com/sql/linux/quickstart-install-connect-docker).

## Создание базы данных

Как и у многих инструкций языка Transact-SQL, у инструкции [CREATE DATABASE](https://docs.microsoft.com/ru-ru/sql/t-sql/statements/create-database-transact-sql?view=sql-server-ver15) имеется обязательный параметр: имя базы данных.CREATE DATABASE Кроме этого, у инструкции имеется ряд необязательных параметров, таких как расположение на диске, где требуется хранить файлы базы данных. При выполнении инструкции CREATE DATABASE без дополнительных параметров для многих из них SQL Server использует значения по умолчанию.

1. В окне редактора запросов введите, но не выполняйте, следующий код:

SQLКопировать

CREATE DATABASE TestData

GO

1. С помощью указателя выделите слова CREATE DATABASEи нажмите клавишу **F1**. Должен открыться раздел CREATE DATABASE электронной документации на Microsoft SQL Server. Таким же способом можно найти полный синтаксис инструкции CREATE DATABASE и других инструкций, используемых в данном учебнике.
2. В редакторе запросов нажмите клавишу **F5** , чтобы выполнить инструкцию и создать базу данных с именем TestData.

При создании базы данных сервер SQL Server создает копию базы данных **model** и присваивает ей указанное имя базы данных. Эта операция обычно занимает несколько секунд, если только с помощью дополнительного параметра не указан большой исходный размер базы данных.

**Примечание**

Когда в одном пакете представлено несколько инструкций, они разделяются с помощью ключевого слова GO. Ключевое слово GO является необязательным, если в пакете содержится только одна инструкция.

## Создание таблицы

**Область применения:** ДаSQL Server ДаБаза данных SQL Azure ДаAzure Synapse Analytics (Хранилище данных SQL) ДаParallel Data Warehouse

Чтобы создать таблицу, нужно указать имя таблицы, имена и типы данных для каждого столбца таблицы. Также рекомендуется указывать, допускаются ли значения NULL для каждого из столбцов. Для создания таблицы необходимо иметь разрешение CREATE TABLE и разрешение ALTER SCHEMA для схемы, которая будет содержать таблицу. Предопределенная роль базы данных [db\_ddladmin](https://docs.microsoft.com/ru-ru/sql/relational-databases/security/authentication-access/database-level-roles?view=sql-server-ver15) имеет эти разрешения.

Большинство таблиц имеют первичный ключ, состоящий из одной или нескольких столбцов таблицы. Первичный ключ всегда уникален. Компонент Компонент Database Engine потребует выполнения условия неповторения значения первичного ключа в таблице.

Список типов данных и ссылки на их описание см. в разделе [Типы данных (Transact-SQL)](https://docs.microsoft.com/ru-ru/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver15).

**Примечание**

Компонент Компонент Database Engine может быть установлен с учетом регистра и без учета регистра. Если компонент Компонент Database Engine установлен с учетом регистра, имена объектов должны иметь одно и тоже имя. Например, таблица с именем OrderData будет отличаться от таблицы ORDERDATA. Если компонент Компонент Database Engine установлен без учета регистра, эти два имени таблицы будут рассматриваться как одна таблица, то есть имя может быть использовано только один раз.

### Переключение соединения редактора запросов на базу данных TestData

В окне редактора запросов введите и выполните следующий код, чтобы изменить соединение на базу данных TestData .

SQLКопировать

USE TestData

GO

### Создание таблицы

В окне редактора запросов введите и выполните следующий код, чтобы создать таблицу Products. Столбцы таблицы имеют имена ProductID, ProductName, Priceи ProductDescription. Столбец ProductID является первичным ключом таблицы. int, varchar(25), moneyи varchar(max) . Только столбцы Price и ProductionDescription могут быть пустыми при вставке или изменении строки. Данная инструкция содержит необязательный элемент (dbo.), называемый схемой. Схема — это объект базы данных, к которому принадлежит таблица. Если вы являетесь администратором, схемой по умолчанию будет схема dbo . dbo означает владельца базы данных.

SQLКопировать

CREATE TABLE dbo.Products

(ProductID int PRIMARY KEY NOT NULL,

ProductName varchar(25) NOT NULL,

Price money NULL,

ProductDescription varchar(max) NULL)

GO

## Вставка данных в таблицу и их обновление

После создания таблицы **Products** можно вставлять данные в таблицу с помощью инструкции INSERT. После вставки данных содержимое строки изменяется с помощью инструкции UPDATE. Предложение WHERE предназначено для ограничения числа строк, изменяемых в процессе выполнения инструкции UPDATE до одной строки. Чтобы ввести следующие данные, потребуется четыре инструкции.

| **ТАБЛИЦА 1** | | | |
| --- | --- | --- | --- |
| **ProductID** | **ProductName** | **Price** | **ProductDescription** |
| 1 | Clamp | 12,48 | Workbench clamp |
| 50 | Screwdriver | 3,17 | Flat head |
| 75 | Tire Bar |  | Tool for changing tires. |
| 3000 | 3 mm Bracket | 0,52 |  |

Базовый синтаксис: INSERT, имя таблицы, список столбцов, VALUES, а затем список добавляемых значений. Два дефиса в начале строки означают, что строка является примечанием и текст не будет обрабатываться компилятором. В этом случае примечание описывает возможные варианты синтаксиса.

### Вставка данных в таблицу

1. Выполните следующую инструкцию, чтобы добавить строку в таблицу Products , которая была создана в предыдущей задаче.

SQLКопировать

-- Standard syntax

INSERT dbo.Products (ProductID, ProductName, Price, ProductDescription)

VALUES (1, 'Clamp', 12.48, 'Workbench clamp')

GO

**Примечание**

Если вставка выполнена, перейдите к следующему шагу.

Если вставка завершается сбоем, это может быть вызвано тем, что в таблице Product уже есть строка с таким ИД продукта. Чтобы продолжить, удалите все строки в таблице и повторите предыдущий шаг. [**TRUNCATE TABLE**](https://docs.microsoft.com/ru-ru/sql/t-sql/statements/truncate-table-transact-sql?view=sql-server-ver15) удаляет все строки в таблице.

Выполните следующую команду, чтобы удалить все строки в таблице:

SQLКопировать

TRUNCATE TABLE TestData.dbo.Products;

GO

После усечения таблицы повторите команду INSERT на этом шаге.

1. В следующей инструкции показано, как можно изменить порядок, в котором приведены параметры, изменив расположение ProductID и ProductName одновременно как в списке полей (в круглых скобках), так и в списке значений.

SQLКопировать

-- Changing the order of the columns

INSERT dbo.Products (ProductName, ProductID, Price, ProductDescription)

VALUES ('Screwdriver', 50, 3.17, 'Flat head')

GO

1. Следующая инструкция показывает, что имена столбцов перечислять не обязательно, если значения перечислены в нужном порядке. Этот синтаксис является стандартным, но не рекомендуется, поскольку другим будет трудно понять ваш код. NULL указано в столбце Price , так как цена этого товара пока неизвестна.

SQLКопировать

-- Skipping the column list, but keeping the values in order

INSERT dbo.Products

VALUES (75, 'Tire Bar', NULL, 'Tool for changing tires.')

GO

1. Имя схемы указывать не обязательно, пока доступ и изменение таблицы осуществляются с помощью схемы по умолчанию. Поскольку в столбце ProductDescription разрешены значения NULL и значение для столбца не приведено, имя и значение столбца ProductDescription в инструкции могут быть полностью опущены.

SQLКопировать

-- Dropping the optional dbo and dropping the ProductDescription column

INSERT Products (ProductID, ProductName, Price)

VALUES (3000, '3 mm Bracket', 0.52)

GO

### Обновление таблицы продуктов

Введите и выполните следующую инструкцию UPDATE , чтобы изменить значение ProductName второго продукта со значения Screwdriverна значение Flat Head Screwdriver.

SQLКопировать

UPDATE dbo.Products

SET ProductName = 'Flat Head Screwdriver'

WHERE ProductID = 50

GO

## Чтение данных из таблицы

Для чтения данных в таблице используется инструкция SELECT. Инструкция SELECT является одной из наиболее важных инструкций Transact-SQL , и для нее существует много разновидностей синтаксиса. В этом учебнике будет показана работа с пятью простыми вариантами.

### Чтение данных в таблице

1. Чтобы прочитать данные из таблицы Products , введите и выполните следующие инструкции.

SQLКопировать

-- The basic syntax for reading data from a single table

SELECT ProductID, ProductName, Price, ProductDescription

FROM dbo.Products

GO

1. Чтобы выбрать все столбцы в таблице, можно использовать звездочку (\*). Звездочка используется для нерегламентированных запросов. В постоянном коде укажите список всех столбцов, чтобы инструкция возвращала нужные столбцы, даже если какой-то столбец будет добавлен в таблицу позднее.

SQLКопировать

-- Returns all columns in the table

-- Does not use the optional schema, dbo

SELECT \* FROM Products

GO

1. Если нет необходимости возвращать определенные столбцы, их можно опустить. Столбцы возвращаются в том порядке, в котором они перечислены.

SQLКопировать

-- Returns only two of the columns from the table

SELECT ProductName, Price

FROM dbo.Products

GO

1. Чтобы ограничить количество строк, возвращаемых пользователю, используйте предложение WHERE .

SQLКопировать

-- Returns only two of the records in the table

SELECT ProductID, ProductName, Price, ProductDescription

FROM dbo.Products

WHERE ProductID < 60

GO

1. Можно работать со значениями столбцов, по мере того как столбцы возвращаются. В следующем примере выполняется математическая операция над столбцом Price . Столбцы, изменяемые подобным образом, не имеют имени, если только имя не указывается с использованием ключевого слова AS .

SQLКопировать

-- Returns ProductName and the Price including a 7% tax

-- Provides the name CustomerPays for the calculated column

SELECT ProductName, Price \* 1.07 AS CustomerPays

FROM dbo.Products

GO

### Полезные функции в инструкции SELECT

Сведения о работе с функциями, которые используются в инструкциях SELECT, см. в следующих разделах:

| **ТАБЛИЦА 2** | |
| --- | --- |
|  |  |
| [Строковые функции (Transact-SQL)](https://docs.microsoft.com/ru-ru/sql/t-sql/functions/string-functions-transact-sql?view=sql-server-ver15) | [Типы данных и функции даты и времени (Transact-SQL)](https://docs.microsoft.com/ru-ru/sql/t-sql/functions/date-and-time-data-types-and-functions-transact-sql?view=sql-server-ver15) |
| [Математические функции (Transact-SQL)](https://docs.microsoft.com/ru-ru/sql/t-sql/functions/mathematical-functions-transact-sql?view=sql-server-ver15) | [Функции для работы с типами данных text и image (Transact-SQL)](https://msdn.microsoft.com/library/b9c70488-1bf5-4068-a003-e548ccbc5199) |

## Создание представлений и хранимых процедур

Представление является хранимой инструкцией SELECT, а хранимая процедура представляет собой одну или более инструкций Transact-SQL , выполняемых в виде пакета.

Представления запрашиваются так же, как таблицы, и не принимают параметры. Хранимые процедуры сложнее, чем представления. Хранимые процедуры содержат как входные, так и выходные параметры и могут содержать инструкции, которые управляют потоком кода, например IF и WHILE. Использование хранимых процедур для всех повторяющихся действий в базе данных является хорошим стилем программирования.

В этом примере используется инструкция CREATE VIEW, чтобы создать представление, которое выбирает только два столбца в таблице **Products** . Затем с помощью инструкции CREATE PROCEDURE создается хранимая процедура, которая принимает цену в качестве параметра и возвращает только те продукты, цена которых меньше значения, указанного в качестве параметра.

### Создание представления

Выполните следующую инструкцию, создающую представление, которое выполняет инструкцию select и возвращает названия и цены продуктов пользователю.

SQLКопировать

CREATE VIEW vw\_Names

AS

SELECT ProductName, Price FROM Products;

GO

### Тестирование представления

С представлениями обращаются так же, как с таблицами. Используйте инструкцию SELECT , чтобы получить доступ к представлению.

SQLКопировать

SELECT \* FROM vw\_Names;

GO

### Создание хранимой процедуры

В следующем примере создается хранимая процедура pr\_Namesс входным параметром @VarPrice типа money. Эта хранимая процедура печатает инструкцию Products less than , соединенную операцией сцепления с входным параметром, тип которого преобразуется из money в varchar(10) . Затем процедура выполняет инструкцию SELECT на представлении, передавая входной параметр в предложение WHERE . Возвращаются все продукты, цена которых меньше значения входного параметра.

SQLКопировать

CREATE PROCEDURE pr\_Names @VarPrice money

AS

BEGIN

-- The print statement returns text to the user

PRINT 'Products less than ' + CAST(@VarPrice AS varchar(10));

-- A second statement starts here

SELECT ProductName, Price FROM vw\_Names

WHERE Price < @varPrice;

END

GO