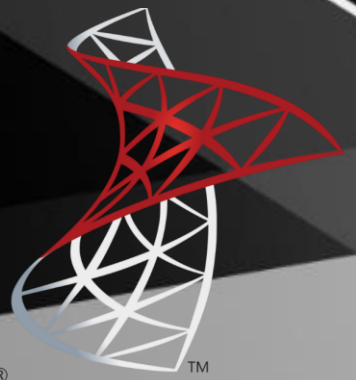
В. Б. Пикулев

БАЗЫ ДАННЫХ

3. СУБД MS SQL Server

scilink.ru, 2021



Общие сведения

Microsoft SQL Server является реляционной СУБД, поддерживающей технологию клиент-сервер вплоть до создания распределённых баз данных в масштабе локальной или глобальной компьютерной сети. SQL Server позволяет создавать базы данных различного масштаба: от уровня домашней сети до корпоративного.

- В 1988 году Microsoft и Ashton-Tate анонсировали первую версию Microsoft SQL Server — реляционную СУБД для локальных вычислительных сетей (*Sybase* DataServer для OS/2). Ashton-Tate предоставила исходные коды СУБД dBASE IV.
- В 1992 выпущен SQL Server 4.2 — 16-разрядная СУБД, результат совместной работы Microsoft и Sybase. В этой СУБД были реализованы клиентские библиотеки для MS-DOS, Windows и OS/2, впервые включены средства администрирования с графическим интерфейсом под управлением Windows.
- 1998 — выпущен Microsoft SQL Server 7.0 с радикально измененной архитектурой (без унаследованного кода). Появились встроенные службы анализа данных.
- В дальнейшем MS SQL Server в течение многих лет разрабатывался с ориентацией на платформу Windows Server. Начиная с MS SQL Server 2005 эта СУБД была интегрирована с платформой .NET.
- MS SQL Server 2019 – кроссплатформенная версия: поддержка кластеров больших данных, размещённых в различных источниках данных, включая использование данных для систем искусственного интеллекта, машинного обучения и статистического анализа.

Возможности MS SQL Server

Ядро SQL Server реализовано в виде набора сервисов, для управления которыми используются административные утилиты. В актуальную версию MS SQL Server входят:

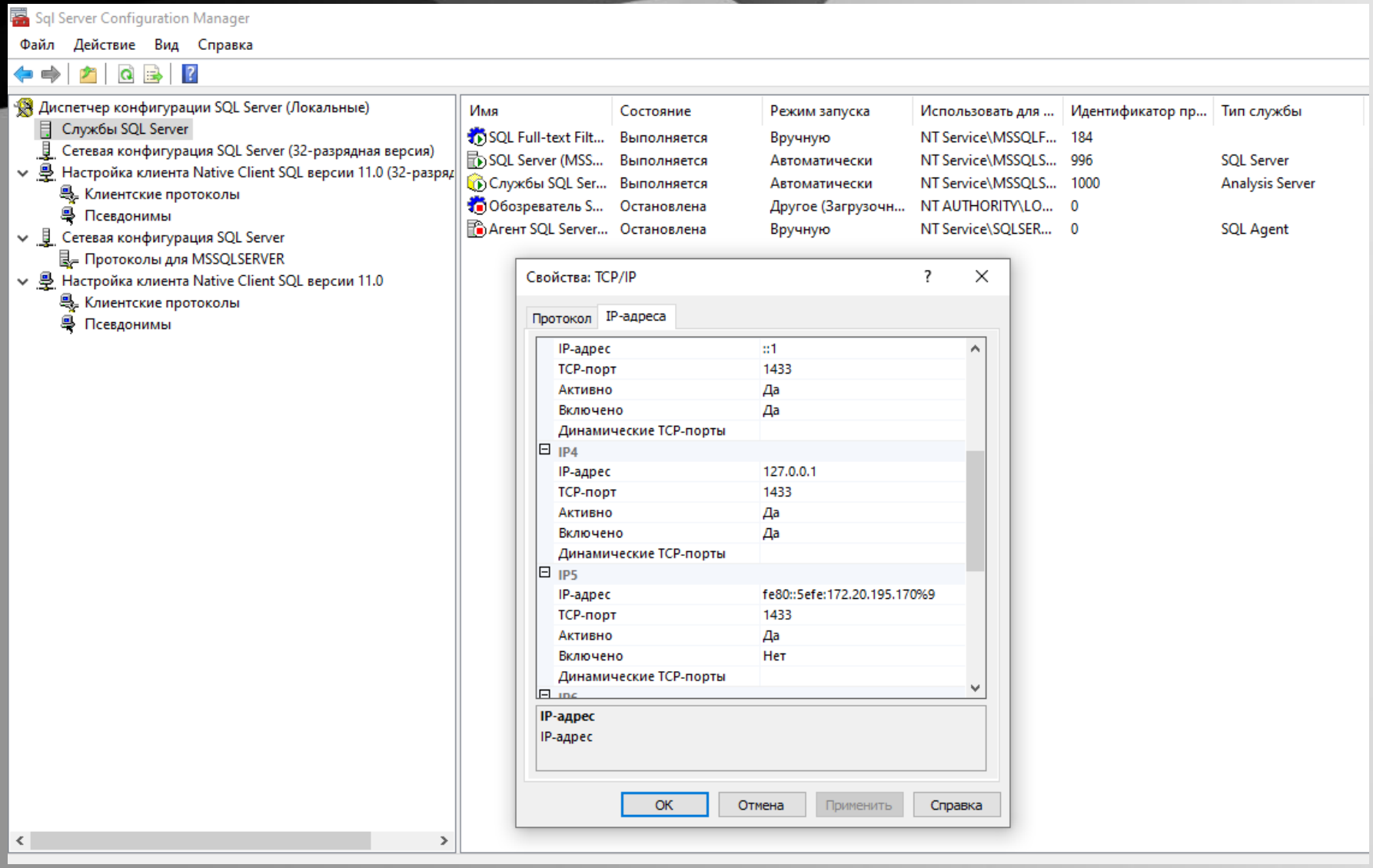
- Службы машинного обучения
- Службы SQL Server Analysis Services
- SQL Server Integration Services
- Службы Master Data Services
- Службы SQL Server Reporting Services

SQL Server имеет в своём составе серверное и клиентское ПО. Приложения клиента выполняют всю работу по взаимодействию с пользователем, включая отображение информации и предоставление возможности работы с приложением через графический интерфейс пользователя.

Одним из преимуществ SQL Server является простота его освоения. SQL Server Enterprise Manager, входящий в состав всех редакций SQL Server, представляет собой полнофункциональное и достаточно простое средство для администрирования этой СУБД.

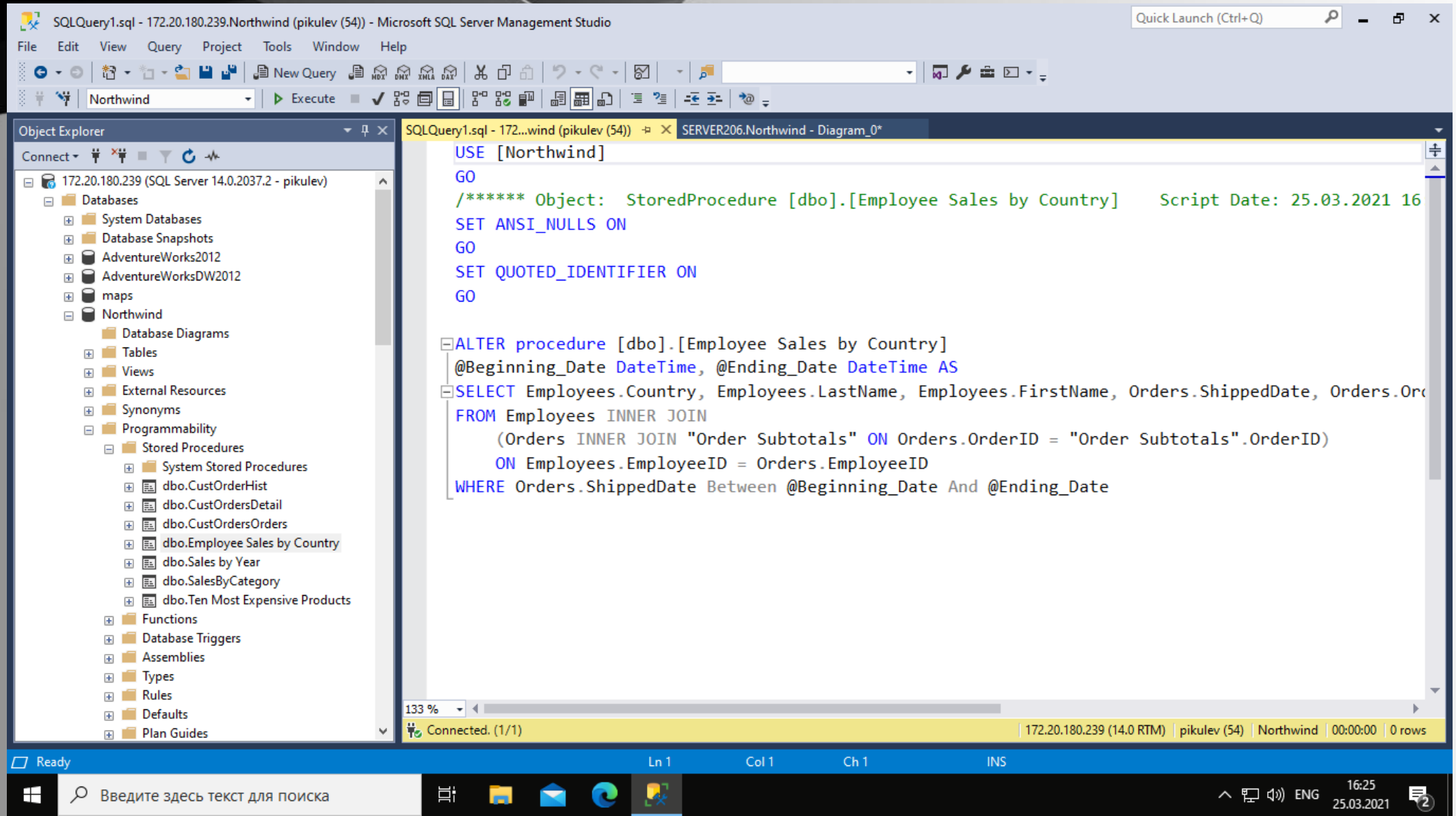
SQL Server поддерживает сетевые протоколы **TCP/IP** Sockets (IPX/SPX), **Named Pipes** NetBios (для Apple и Unix), поддерживается доступ через защищенные вызовы RPC (Remote Procedure Call).
Версия языка SQL, используемого SQL Server, называется **Transact-SQL** (T-SQL).
Физически *устройства баз данных* могут размещаться в дисковых файлах или на неразмеченных разделах жестких дисков, *устройства резервных копий* могут быть ассоциированы с дисками, съемными накопителями, устройствами записи на магнитные ленты и именованными каналами.

MS SQL Server Configuration Manager



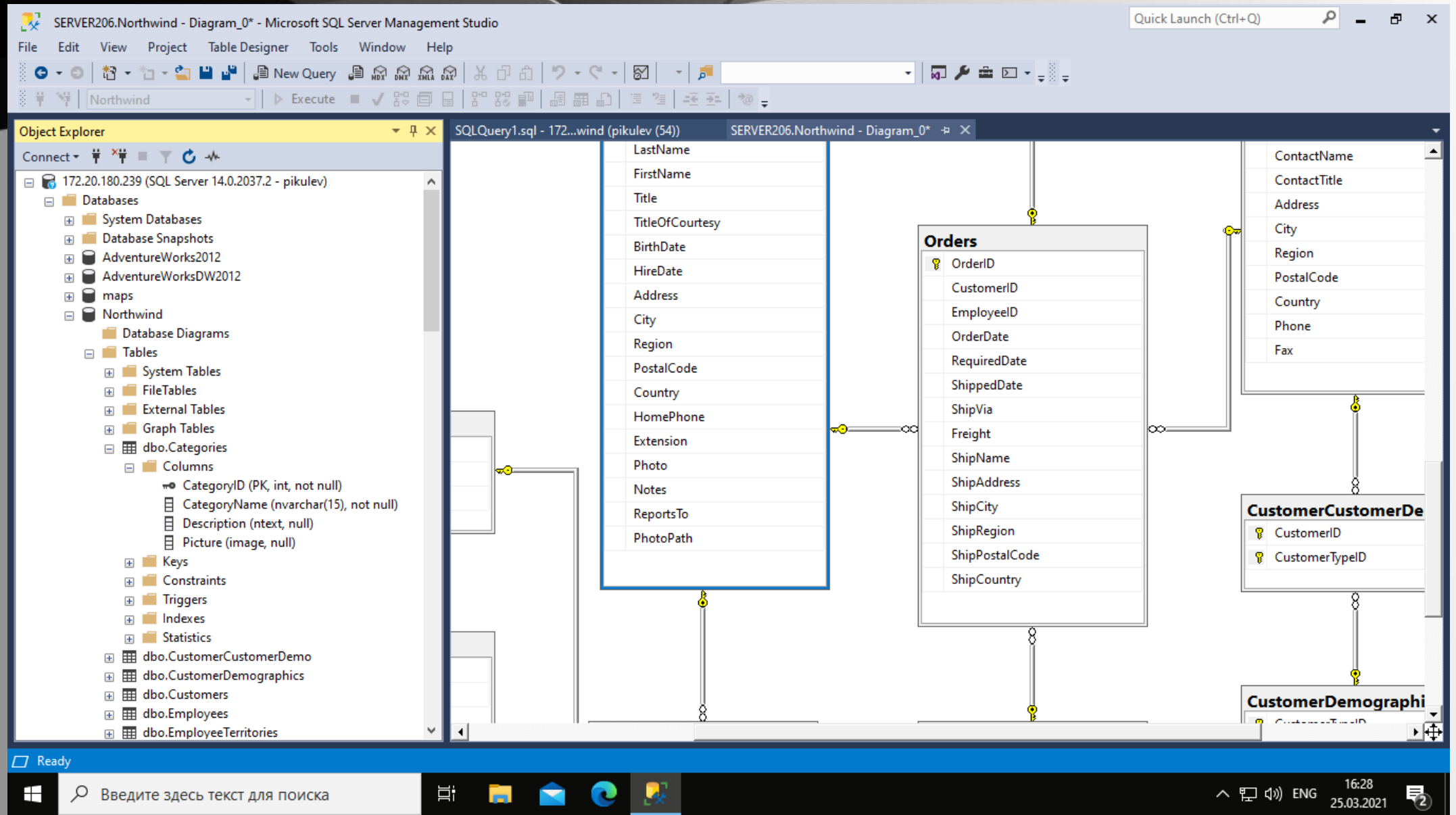
SSMS

MS SQL Server Management Studio



SSMS

MS SQL Server Management Studio



Встроенные и определённые пользователем типы данных хранятся в текущей базе данных в таблице **systypes** и могут быть просмотрены командой **SELECT * FROM systypes**

Типы данных Microsoft SQL Server

для сравнения

Точные числа

bigint	8 байт, от -2^{63} (-9 223 372 036 854 775 808) до $2^{63}-1$ (9 223 372 036 854 775 807)
bit	Целочисленный тип данных, который может принимать значения 1, 0 или NULL
decimal (n,p) **numeric**	Числа с фиксированной точностью и масштабом. От $-10^{38}+1$ до $10^{38}-1$. Точность - максимальное количество ВСЕХ разрядов числа. Точность должна быть значением в диапазоне от 1 до 38. Точность по умолчанию составляет 18. Масштаб - максимальное количество десятичных разрядов числа справа от десятичной запятой. По умолчанию масштаб принимает значение 0.
int	4 байта, от -2^{31} ($-2\ 147\ 483\ 648$) до $2^{31}-1$ (2 147 483 647)
money	8 байт, от -922 337 203 685 477,5808 до 922 337 203 685 477,5807
smallint	2 байта, от -2^{15} (-32 768) до $2^{15}-1$ (32 767)
smallmoney	4 байта, от -214 748,3648 до 214 748,3647
tinyint	1 байт, от 0 до 255

Результат вычисления выражения приводится к типу данных, имеющих максимальный размер из всех участвующих в выражении.

```
DECLARE @VR float, @VS varchar(20)
SET @VR=3.14
SET @VS=STR(@VR,4,2)+'15'
SELECT @VS
```

Типы данных Microsoft SQL Server

Числа в формате с плавающей точкой

Float (n)	n в диапазоне от 1 до 53: 4 либо 8 байт, от 1,79E308 до -2,23E-308, 0 и от 2,23E-308 до 1,79E308
real	Соответствует float(24): 4 байта, от 3,40E38 до -1,18E-38, 0 и от 1,18E-38 до 3,40E38

Бинарные данные

binary	Двоичные данные фиксированной длины размером в n байт, где n — значение от 1 до 8000. Длина по умолчанию равна 1.
varbinary	Двоичные данные с переменной длиной. n может иметь значение от 1 до 8000.
image	Двоичные данные переменной длины, включающие от 0 до $2^{31} - 1$ (2 147 483 647) байт. Использовать не рекомендуется, альтернатива – **varbinary(max)**

требуется указывать числа в 16-ричном виде, например 0xFF.

```
DECLARE @name nvarchar(25)
SET @name = 'Д''Артаньян'
SELECT @name
```

Типы данных Microsoft SQL Server

Символьные строки

char(n)	Строковые данные фиксированной длины. Аргумент n определяет длину строки и должен иметь значение от 1 до 8000. По умолчанию длина равна 1.
nchar(n)	Строковые данные постоянной длины в Юникоде. Параметр n определяет длину строки и должен иметь значение от 1 до 4000.
varchar(n)	Строковые данные переменной длины.
nvarchar(n)	Строковые данные переменной длины в Юникоде.
text	Данные переменной длины в кодовой странице сервера и с максимальной длиной строки 2^31-1 (2 147 483 647). Использовать не рекомендуется, альтернатива – varchar(max)
ntext	Данные переменной длины в Юникоде с максимальной длиной строки 2^30 - 1 (1 073 741 823) байт. Использовать не рекомендуется, альтернатива – **nvarchar(max)**

```
SET LANGUAGE 'русский'
DECLARE @DV datetime
SET @DV='10 марта 2021 11:05'
SELECT @DV
```

Типы данных Microsoft SQL Server

Дата и время

date	От 1 января 1 года нашей эры до 31 декабря 9999 года нашей эры. Дата хранится в одной переменной типа integer размером 1 или 3 байта. Точность - один день. Календарь григорианский.
datetime	Даты с 1 января 1753 года — 31 декабря 9999 года, время от 00:00:00 до 23:59:590,997. Календарь григорианский. Округляется в третьем знаке после запятой. 8 байт.
datetime2	С 1 января 1 года нашей эры до 31 декабря 9999 года нашей эры. Время от 00:00:00 до 23:59:59.9999999. Точность по умолчанию составляет 7 цифр, точность по времени 100 нс. Календарь григорианский. 8 байт.
datetimeoffset	Определяет дату, объединенную со временем дня, с учетом часового пояса в 24-часовом формате. От 1 января 1 года нашей эры до 31 декабря 9999 года нашей эры. Время от 00:00:00 до 23:59:59.9999999. Диапазон смещения часового пояса от -14:00 до +14:00. Точность по времени 100 нс. 10 байт.
smalldatetime	1 января 1900 года — 6 июня 2079 года. Точность по времени – одна минута. 4 байта.
time	Определяет время дня в 24-часовом формате. От 00:00:00.0000000 до 23:59:59.9999999. Точность долей секунды по умолчанию равна 7 (100 нс). 5 байт.

Типы данных Microsoft SQL Server

Пространственные

geography	Представляет данные в системе координат круглой земли: различные векторные объекты с вершинами в виде координат широты и долготы в системе GPS.
geometry	Представляет данные в евклидовом пространстве (на плоскости). Имеется набор методов для создания объектов, хранящихся в полях этого типа.

Прочие

cursor	Тип данных для переменных или выходных параметров хранимых процедур, которые содержат ссылку на курсор.
table	Данные в виде таблицы
timestamp **rowversion**	Автоматически сформированные уникальные двоичные числа. 8 байт.
uniqueidentifier	16-байтовый идентификатор GUID.
hierarchyid	Используется для представления положения в иерархии. Значение типа данных hierarchyid представляет позицию в древовидной иерархии.
sql_variant	Тип данных, хранящий значения различных типов данных SQL Server.
xml	Тип данных, в котором хранятся XML-данные. Размер хранимого представления экземпляров типа данных xml не может превышать 2 ГБ.

Transact-SQL: примеры

SQLQuery2.sql - 172...work (pikulev (57))*

```
DECLARE @I1 int
SET @I1 = 1
WHILE @I1 < 8
  BEGIN
    PRINT 'Квадрат числа '+Str(@I1) + ' есть ' + Str(Square(@I1))
    SET @I1 = @I1 + 1
  END
```

133 %

Messages

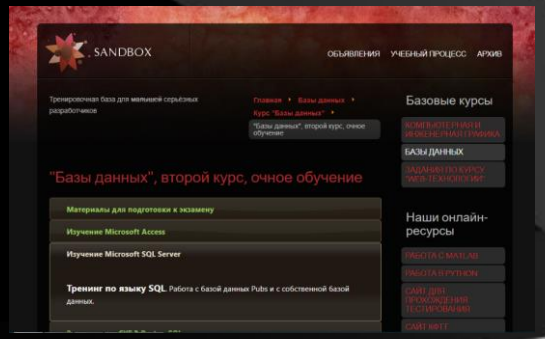
```
Квадрат числа          1 есть          1
Квадрат числа          2 есть          4
Квадрат числа          3 есть          9
Квадрат числа          4 есть         16
Квадрат числа          5 есть         25
Квадрат числа          6 есть         36
Квадрат числа          7 есть         49

Completion time: 2021-03-25T16:36:10.9427355+03:00
```

```
SELECT name, year,
CASE grade
  WHEN 5 THEN 'отлично'
  WHEN 4 THEN 'хорошо'
  WHEN 3 THEN 'удовл.'
  ELSE state
END
FROM session
WHERE group_id LIKE '21217'
```

```
DECLARE @VarTable TABLE (
  Col1 int NOT NULL IDENTITY (1,1) PRIMARY KEY,
  Col2 nvarchar(15) )
INSERT INTO @VarTable (Col2) VALUES ('Одна строка')
SELECT * FROM @VarTable
```

```
CREATE TABLE MyMoney (
  ID bigint IDENTITY (1,1) PRIMARY KEY,
  Value money NULL
)
INSERT MyMoney VALUES ($127.35)
SELECT * FROM MyMoney
```



Transact-SQL: примеры

```
use pubs

-- Вывести название книги, у которой был самый высокий объём продаж
select top(1) title, ytd_sales from titles
order by ytd_sales desc

-- Какие книги имеют более одного автора?
select t.title, count(*) as [число авторов]
from titleauthor ta inner join titles t on t.title_id = ta.title_id
group by t.title
having count(*) > 1

-- ??
select max(a.au_lname), max(a.au_fname)
from titles t inner join titleauthor ta on t.title_id = ta.title_id
inner join authors a on ta.au_id = a.au_id
group by a.au_id
having min(t.pub_id) <> max(t.pub_id)
```

Конструктивные элементы MS SQL Server

Индексы

Кластеризованный – индекс, который хранит данные таблицы в отсортированном, по значению ключа индекса, виде.

Некластеризованный – индекс, который содержит значение ключа и указатель на строку данных, содержащую значение этого ключа.

Полнотекстовый – индекс, который обеспечивает эффективную поддержку сложных операций поиска слов в символьных строковых данных.

Одним из важнейших путей достижения высокой производительности SQL Server является использование индексов. **Индекс** - это конструктивный элемент базы данных, который представляет собой структуру данных, состоящую из ключей, построенных на основе одного или нескольких столбцов таблицы или представления, и указателей, которые сопоставляются с местом хранения заданных данных. Индекс ускоряет процесс запроса, предоставляя быстрый доступ к строкам данных в таблице.

```
CREATE INDEX i1 ON table1 (col_1, col_2)
```

```
ALTER INDEX IX_NonClustered ON TestTable REBUILD
```

Однако по умолчанию работу по созданию индексов берёт на себя СУБД, так что для простых баз данных, не требующих от системы высокой производительности, индексные конструкции создавать не требуется.

Конструктивные элементы MS SQL Server

Временные таблицы

Временные таблицы существуют только на протяжении сессии базы данных. После создания все временные таблицы сохраняются в базе данных **tempdb**, к которой имеют доступ по умолчанию пользователи MS SQL Server.

Названия **локальных временных таблиц** следует начинать с символа **#**. Такие таблицы существуют до тех пор, пока действует соединение с SQL Server, в котором эти таблицы были созданы, и автоматически уничтожаются при закрытии соединения. Локальные таблицы видимы только для соединения, создавшего их.

Названия **глобальных временных таблиц** начинаются с символов **##**. Существуют эти таблицы так же, как и локальные, однако во время своего существования являются видимыми и из любого другого соединения с сервером. Имя глобальной таблицы должно быть уникальным для сервера.

```
CREATE TABLE #myTempTable
(UserId INT IDENTITY,
UserName NVARCHAR(20),
Score INT)
```

Конструктивные элементы MS SQL Server

Представления

```
CREATE VIEW My_view AS
SELECT au_lname, au_fname, [address]
FROM authors WHERE state='CA'

ALTER VIEW ...

DROP VIEW ...
```

Представление для пользователей базы данных выглядит как таблица, однако на самом деле его содержимое формируется запросом. Физически данные, виртуально принадлежащие представлению, находятся в таблицах, к которым обращается этот запрос. Представление может быть использовано:

- для защиты конфиденциальной информации
- для упрощения доступа к информации
- для сокращения времени доступа.

Для указанных целей представление может быть проиндексировано.

Недостатки:

- в запросе, определяющим представление, нельзя использовать разделы ORDER BY и INTO
- имеется ряд ограничений на изменение, добавление и удаление данных в представлении, созданном для нескольких таблиц.

Конструктивные элементы MS SQL Server

```
DECLARE @S nvarchar(max)
SET @S='<table>'
DECLARE tabloid CURSOR FOR
  SELECT * from MyTable ORDER BY id
OPEN tabloid
DECLARE @S1 nvarchar(300)
DECLARE @S2 nvarchar(300)
FETCH NEXT FROM tabloid INTO @S1, @S2

WHILE @@FETCH_STATUS=0
BEGIN
  SET @S = @S+'<tr><td>' + @S1 +
'</td><td>' + @S2 + '</td></tr>'
  FETCH NEXT FROM tabloid INTO @S1, @S2
END

CLOSE tabloid
DEALLOCATE tabloid
SET @S = @S + '</table>'
```

Курсоры

Вполне может случиться, что ответом на простой запрос клиента будет выборка из сотен тысяч строк, что для большинства клиентов неудобоваримо. В таком случае решением проблемы взаимодействия с клиентами является использование курсоров как универсального механизма для опосредования обмена данными между сервером и клиентом. **Курсоры** работают с результирующим набором данных (результатом выполнения запроса), давая пользователям дополнительные возможности по обработке данных:

- курсоры позволяют работать со строками таблицы посредством указания их порядкового номера в наборе данных
- курсоры позволяют реализовать сложные операции изменения данных, например когда для изменения значения столбца требуется многократно обращаться к значениям других столбцов.

Конструктивные элементы MS SQL Server

Хранимые процедуры

```
-- описываем хранимую процедуру
CREATE PROCEDURE MyProc
@lastname char(64),
@firstname char(64)
AS BEGIN
  SELECT * FROM authors a
  WHERE a.au_lname = @lastname
  AND a.au_fname = @firstname
END

-- создаём хранимую процедуру
GO

-- вызываем созданную процедуру
MyProg 'Иван','Бездомный'
```

Хранимая процедура – это именованный набор команд T-SQL, хранящийся непосредственно на сервере и представляющий собой самостоятельный конструктивный элемент базы данных. Хранимая процедура может быть вызвана клиентской программой, другой хранимой процедурой или триггером. Когда хранимая процедура выполняется первый раз, сервер создаёт план исполнения процедуры, выполняет её оптимизацию и компиляцию. В дальнейшем при повторном вызове процедуры используется уже сгенерированный план, что позволяет оптимизировать её время исполнения.

В состав MS SQL Server входит большое количество встроенных системных хранимых процедур. Все они имеют префикс ***sp_*** и охватывают практически все аспекты управления и конфигурирования сервера, позволяя изменять значения в системных таблицах пользовательских и системных баз данных.

```
sp_help
```

Конструктивные элементы MS SQL Server

Триггеры

```
CREATE TRIGGER Products_INSERT
ON Products
AFTER INSERT
AS
INSERT INTO History (ProductId, Operation)
SELECT Id, 'Добавлен товар: ' + ProductName +
', фирма: ' + Manufacturer
FROM INSERTED

INSERT INTO Products
(ProductName, Manufacturer, ProductCount,
Price)
VALUES('iPhone X', 'Apple', 2, 79900)

SELECT * FROM History
```

Триггером в SQL Server называется специальная хранимая процедура, привязанная к конкретной таблице (представлению) и запускаемая сервером автоматически при обращении к этой таблице. Когда пользователь, например, успешно изменил данные в таблице, сервер автоматически запускает триггер, причём если произойдёт откат триггера, то это повлечёт и отмену пользовательских изменений данных.

Триггеры могут использоваться:

- для нестандартного контроля целостности данных
- для вычисления значений в полях таблицы по значениям других полей
- для ограничения действий различных групп пользователей.

Конструктивные элементы MS SQL Server

Пользовательские функции

```
CREATE FUNCTION MyFunc ( @State char(2) )
RETURNS TABLE AS
  RETURN SELECT au_id, au_lname, au_fname
  FROM authors
  WHERE state = @state

GO

SELECT * FROM MyFunc('CA')
ORDER BY au_lname, au_fname
```

Пользовательские функции представляют возможность их вызова непосредственно из выражений (как это принято для встроенных функций) и способны возвращать результат (как скалярный, так и табличное значение). В теле функции разрешается объявление локальных переменных, использование циклов, ветвлений и любых других программных конструкций, разрешается вызов других функций. Не разрешается использование в теле функции команды PRINT, а также команды SELECT для непосредственного возвращения данных.

SP из реального проекта

```
-- ============================================
-- Формируем список образовательных организаций для страницы "образовательные организации"
-- ============================================
CREATE PROCEDURE [dbo].[ShowEduTable]
          @Region int = 0,
          @EduLevel int = 0
AS
BEGIN
--определяем местоположение
declare @T1 table (locusId int)
declare @T2 table (ID int, Title nvarchar(255), NumOfProgramms int, MinCosts int, MinTreshold int)

if @Region = 1
          insert into @T1 select Id from [SmartCareer].[dbo].[Locations] where (code_okato = 45000000000) or (ParentId = 45000000000)
if @Region in (0, 2)
          insert into @T1 select Id from [SmartCareer].[dbo].[Locations] where (code_okato = 86000000000) or (ParentId = 86000000000)
--формируем запрос
INSERT INTO @T2
          SELECT DISTINCT EO.Id, ltrim(rtrim(EO.Title)), 0, 0, 0
          FROM [SmartCareer].[dbo].[EduOrganizations] EO
          WHERE EO.LocationId IN (SELECT locusId FROM @T1) and ((EO.EduLevels_id = @EduLevel) or (@EduLevel = 0))
UPDATE @T2 SET NumOfProgramms = X.N from
          (select count(distinct SpecId) as N, E.EduOrgId as Id
          from SmartCareer.data.EduOrgData E inner join @T2 T on E.EduOrgId = T.ID
          group by E.EduOrgId) X inner join @T2 T on X.Id = T.ID
UPDATE @T2 SET MinCosts = X.minValue
          FROM (SELECT EduOrgId, min(aValue) as minValue FROM [SmartCareer].[data].[EduOrgData] WHERE ParamId=3 GROUP BY EduOrgId) X
          INNER JOIN @T2 T ON X.EduOrgId = T.ID
UPDATE @T2 SET MinTreshold = X.minValue FROM (SELECT EduOrgId, min(TRY_CONVERT(int, sValue)) as minValue
          FROM [SmartCareer].[data].[EduOrgData] WHERE ParamId=4 GROUP BY EduOrgId) X
          INNER JOIN @T2 T ON X.EduOrgId = T.ID
UPDATE @T2 SET MinCosts = 0 WHERE MinCosts IS NULL
UPDATE @T2 SET MinTreshold = 0 WHERE MinTreshold IS NULL

SELECT * FROM @T2
ORDER BY Title
END
```

Ограничения

Ограничение – метод обеспечения целостности данных. Ограничения могут быть определены как на уровне столбца, так и на уровне таблицы в целом. Ограничения обеспечивают правильность данных, введенных в поля таблицы, и корректность связей между таблицами. Имена ограничений должны быть уникальными для базы данных. С помощью ключевого слова **CONSTRAINT** можно задать имя для ограничения.

Типы ограничений:

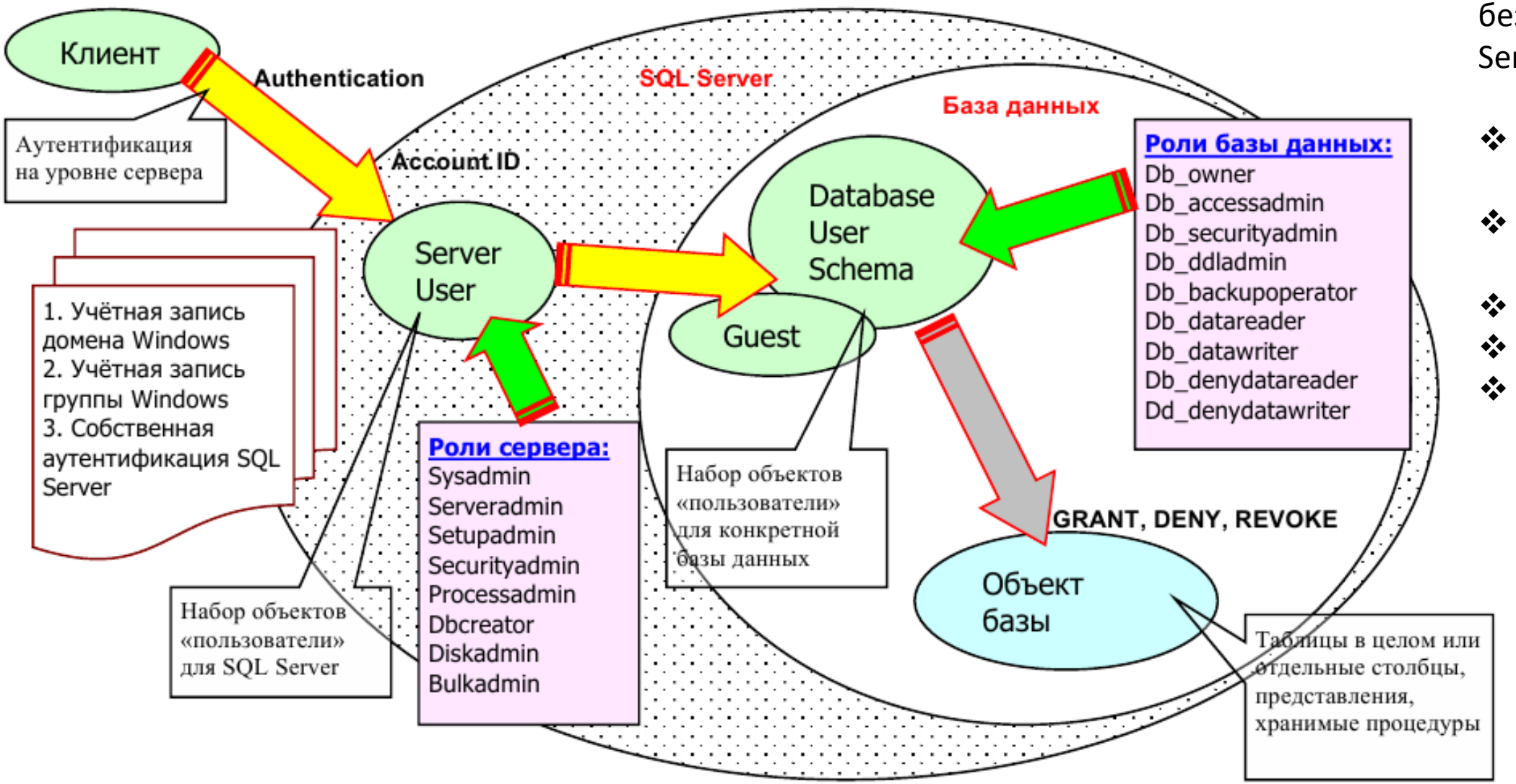
- NULL | NOT NULL
- PRIMARY KEY
- UNIQUE
- FOREIGN KEY … REFERENCES
- ON DELETE
- ON UPDATE
- CHECK
- DEFAULT

```
-- 1
CREATE TABLE publishers (
pub_id int NOT NULL PRIMARY KEY,
pub_name varchar(40) DEFAULT ('неизвестно')
CHECK (pub_id LIKE '99[0-9][0-9]')
)

-- 2
ALTER TABLE MyTable
ADD CONSTRAINT MyKey
PRIMARY KEY (id)

--3
CREATE TABLE TestTable (
 Num int DEFAULT 1,
 CONSTRAINT check_ID CHECK (Num is NOT NULL)
)
```

Администрирование MS SQL Server



Основой системы безопасности SQL Server являются:

- учётные записи (accounts)
- пользователи (users)
- роли (roles)
- группы (groups)
- схемы данных (schemas).

Администрирование MS SQL Server

Когда пользователь подключается к SQL Server, действия, которые он может выполнять, определяются правами, выданными ему как **пользователю** и **члену роли**. Права выдаются администратором, владельцем базы данных или владельцем конкретного объекта базы данных. Сервер позволяет передавать права владения от одного пользователя другому. Права в базе данных можно разделить на три категории:

- права на доступ к объектам баз данных
- права на выполнение команд TSQL
- неявные права.

В SQL Server существует пользователь, наделенный всеми административными полномочиями - это System Administrator или **sa**. Пользователь, создающий новую базу данных, автоматически становится её владельцем (**dbo** – Data Base Owner). В момент создания базы определяется и пользователь **guest**. Если учётная запись пользователя явно не отображается в пользователя конкретной базы данных, пользователю предоставляется неявный доступ с использованием гостевого имени guest. Обычно guest запрещают.

Пользователь, создавший объект в базе данных, автоматически становится его владельцем, и никто, включая dbo и sa, не могут использовать этот объект, пока владелец не назначит им права на него. Но чтобы пользователь мог создать объект, владелец базы данных должен сначала ему предоставить соответствующие права.

Transact-SQL в действии

```
-- скрипт для создания студенческих БД
DECLARE @user NVARCHAR(50)
DECLARE @pass NVARCHAR(50)
DECLARE @command NVARCHAR(300)
USE tempdb
-----
-- Редактировать только здесь!
SET @user = 'std-01'
SET @pass = 'dssp7251'
-----
```

```
SET @command = 'CREATE DATABASE ['+ @user +']'
EXEC sp_executesql @command
SET @command = 'CREATE LOGIN [' + @user + ']
WITH PASSWORD = '''+ @pass +''', CHECK_POLICY = OFF,
DEFAULT_DATABASE = ['+ @user +']'
EXEC sp_executesql @command
SET @command = 'USE ['+ @user +']
CREATE USER ['+ @user +'] FOR LOGIN ['+ @user +']'
EXEC(@command)
SET @command = 'USE ['+ @user +']
ALTER ROLE db_owner ADD MEMBER ['+ @user +']'
EXEC(@command)
USE [pubs]
SET @command = 'CREATE USER ['+ @user +']
FOR LOGIN ['+ @user +']'
EXEC sp_executesql @command
SET @command = 'ALTER ROLE db_datareader
ADD MEMBER ['+ @user +']'
EXEC sp_executesql @command
```

Спасибо за внимание!