Практическая работа 22
Основные методы защиты данных.
Управление привилегиями пользователей

Цель занятия: Получить практический опыт применения методов защиты данных и управления привилегиями пользователей.

Перечень оборудования и программного обеспечения

Персональный компьютер
Microsoft Office (Word, Visio, Access)
Microsoft SQL Server Management Studio

Краткие теоретические сведения

1 Применение методов шифрования

Шифрование – это защита информации от посторонних. Шифрование представляет собой способ скрытия данных с помощью ключа или пароля. Это делает данные бесполезными без соответствующего ключа или пароля для дешифрования.

Защита должна всегда начинаться с разграничения прав доступа, но, даже в случае обхода системы управления правами, шифрование поможет защитить информацию. Например, когда база данных попадет в чужие руки, украденная или изъятая информация будет бесполезна, если она была предварительно зашифрована.

В Microsoft SQL Server реализовано прозрачное шифрование баз данных (Transparent Data Encryption). Прозрачное шифрование кодирует базы данных целиком. Когда страница данных записывается из оперативной памяти на диск, она шифруется. Когда страница загружается обратно в оперативную память, она расшифровывается. Таким образом, база данных на диске оказывается полностью зашифрованной, а в оперативной памяти – нет. Основным преимуществом TDE является то, что шифрование и дешифрование выполняются абсолютно прозрачно для приложений. Использовать преимущества шифрования может любое приложение, использующее для хранения своих данных Microsoft SQL Server.

В MS SQL Server можно шифровать или расшифровывать данные на сервере. Делать это можно различными способами. Например, можно шифровать данные в базах данных одним из следующих методов.

Пароль. Это наименее надежный способ, так как для шифрования и расшифровки данных используется одна и та же парольная фраза. Если хранимые процедуры и функции не зашифрованы, то доступ к парольной фразе возможен через метаданные.

Симметричный ключ. Достаточно надежен, удовлетворяет большинству требований к безопасности данных и обеспечивает достаточное быстродействие. Для шифрования и расшифровки данных используется один ключ.

Асимметричный ключ. Обеспечивает надежную защиту, так как применяются различные ключи для шифрования и расшифровки данных. Однако это негативно влияет на быстродействие. Специалисты Microsoft не рекомендуют использовать его для шифрования крупных значений.

SQL Server располагает встроенными функциями для шифрования и расшифровки на уровне ячеек. Функции шифрования:

ENCRYPTBYKEY, использует симметричный ключ для шифрования данных;

ENCRYPTBYPASSPHRASE, использует парольную фразу для шифрования данных;

ENCRYPTBYASYMKEY, использует асимметричный ключ для шифрования данных.

Функции расшифровки:

DECRYPTBYKEY, использует симметричный ключ для расшифровки данных;

DECRYPTBYPASSPHRASE, использует парольную фразу для расшифровки данных;

DECRYPTBYASYMKEY, использует асимметричный ключ для расшифровки данных.

Пример. В соответствии с политикой безопасности организации, информация о заработной плате сотрудникам за предыдущие периоды должна храниться только в зашифрованном виде с использованием алгоритма шифрования AES с длиной ключа не менее 256 бит.

Для этого:

1. Создать в базе данных DB_DATE симметричный ключ PayKey для применения с алгоритмом AES_256. Ключ должен быть защищён паролем P@ssw0rd.

2. Создать в этой базе данных копию таблицы ZARPLATA. Новая копия должна называться ZARPLATA_ENCRYPTED и все данные в ней должны быть зашифрованы при помощи созданного симметричного ключа.

3. Выполнить запрос, который бы вернул все данные из зашифрованной таблицы ZARPLATA_ENCRYPTED

Зашифрованную информацию нельзя поместить в столбцы типа int, money и т.п. Поэтому создание таблицы ZARPLATA_ENCRYPTED придётся производить вручную. При этом можно использовать для всех столбцов этой таблицы один и тот же тип данных — nvarchar(100). Кроме того, несимвольные типы данных необходимо преобразовать в символьные (например, nvarchar(100)) перед передачей их шифрующей функции.

Создание симметричного ключа. Команда на создание симметричного ключа в соответствии с поставленными условиями может выглядеть так:

```
USE DB_DATE;
GO
CREATE SYMMETRIC KEY PayKey
WITH ALGORITHM = AES_256
ENCRYPTION BY PASSWORD = 'P@ssw0rd;
GO
```

Убедитесь, используя SQL ServerManagement , что ключ был успешно создан:

Databases → DB_DATE → Security → Symmetric Keys.

Создание зашифрованной копии таблицы. Команда для создания зашифрованной копии таблицы ZARPLATAможет выглядеть так:

```
USE DB_DATE;
GO
-- Создаём таблицу для вставки шифрованных данных
CREATE TABLE ZARPLATA_ENCRYPTED
(ID_ZPnvarchar(100),
ZP_Datenvarchar(100),
ZP_SUMMnvarchar(100),
PER_Datenvarchar(100));
GO
-- Открываем симметричный ключ
OPEN SYMMETRIC KEY PayKey DECRYPTION BY PASSWORD =
'P@ssw0rd';
GO
-- Вставляем данные в таблицу при помощи INSERT INTO
INSERT INTO DB_DATE.ZARPLATA_ENCRYPTED
(ID_ZP,ZP_Date, ZP_SUMM,PER_Date)
SELECT
EncryptByKey(Key_GUID('PayKey'), CONVERT(nvarchar(100), ID_ZP)),
EncryptByKey(Key_GUID('PayKey'),CONVERT(nvarchar(100), ZP_Date)),
EncryptByKey(Key_GUID('PayKey'), CONVERT(nvarchar(100), ZP_SUMM)),
EncryptByKey(Key_GUID('PayKey'), CONVERT(nvarchar(100), PER_Date))
FROM DB_DATE.ZARPLATA;
GO
```

Убедитесь, что в таблице EZARPLATA_ENCRYPTED данные зашифрованы:

```
SELECT * FROM ZARPLATA_ENCRYPTED;
GO
```

Запрос, возвращающий данные из нашей зашифрованной таблицы может выглядеть так:

```
OPEN SYMMETRIC KEY PayKey DECRYPTION BY PASSWORD =
'P@ssw0rd';
```

```
GO
SELECT
Convert(Nvarchar(100), DecryptByKey(ID_ZP)) AS ID_ZP,
Convert(Nvarchar(100), DecryptByKey(ZP_Date)) AS ZP_Date,
Convert(Nvarchar(100), DecryptByKey(ZP_SUMM)) AS ZP_SUMM,
Convert(Nvarchar(100), DecryptByKey(PER_Date)) AS PER_Date
FROMDB_DATE.ZARPLATA_ENCRYPTED;
GO
```

Откройте новую сессию подключения к Вашему SQL Server и выполните этот же запрос на выборку (SELECT ...), но без открытия симметричного ключа (OPENSYMMETRICKEY...)

2 Авторизация и аутентификация

Аутентификация — это процесс проверки права пользователя на доступ к тому или иному ресурсу. Чаще всего аутентификация осуществляется с помощью ввода имени и пароля.

SQL Server 2008 поддерживает два режима аутентификации: с помощью Windows и с помощью SQL Server. Первый режим позволяет реализовать решение, основанное на однократной регистрации пользователя и едином пароле при доступе к различным приложениям (*SingleSignOnsolution, SSO*). Подобное решение упрощает работу пользователей, избавляя их от необходимости запоминания множества паролей и тем самым снижая риск их небезопасного хранения. Кроме того, данный режим позволяет использовать средства безопасности, предоставляемые операционной системой, такие как применение групповых и доменных политик безопасности, правил формирования и смены паролей, блокировка учетных записей, применение защищенных протоколов аутентификации с помощью шифрования паролей (Kerberos или NTLM).

Аутентификация с помощью SQL Server предназначена главным образом для клиентских приложений, функционирующих на платформах, отличных от Windows. Этот способ считается менее безопасным, но в SQL Server 2008 он поддерживает шифрование всех сообщений, которыми обмениваются клиент и сервер, в том числе с помощью сертификатов, сгенерированных сервером. Шифрование также повышает надежность этого способа аутентификации. Для учетной записи SQL Server можно указать такой параметр, как необходимость сменить пароль при первом соединении с сервером. Если SQL Server 2008 работает под управлением WindowsServer 2003, можно воспользоваться такими параметрами учетной записи, как проверка срока действия пароля и локальная парольная политика Windows

Схемы, не имеющие отношения к пользователям

Принцип распределения прав доступа к объектам БД в большинстве серверных СУБД основан на наличии у каждого объекта базы данных пользователя-владельца, который может предоставлять другим пользователям права доступа к объектам базы данных. При этом набор объектов,

принадлежащих одному и тому же пользователю, называется схемой. Данный способ владения объектами создавал определенные неудобства при сопровождении приложений, использующих базы данных.

В SQL Server 2008 концепция ролей расширена: эта СУБД позволяет полностью отделить пользователя от схем и объектов базы данных. Теперь объекты базы данных принадлежат не пользователю, а схеме, не имеющей никакого отношения ни к каким учетным записям и тем более к административным привилегиям. Таким образом, схема становится механизмом группировки объектов, упрощающим предоставление пользователям прав на доступ к объектам.

Роли

Для упрощения управления правами доступа в большинстве серверных СУБД применяется механизм ролей — наборов прав доступа к объектам базы данных, присваиваемых некоторой совокупности пользователей. При использовании ролей управление распределением прав доступа к объектам между пользователями, выполняющими одинаковые функции и применяющими одни и те же приложения, существенно упрощается: создание роли и однократное назначение ей соответствующих прав осуществляется намного быстрее, нежели определение прав доступа каждого пользователя к каждому объекту. SQL Server 2008 позволяет создавать так называемые вложенные роли, то есть присваивать одной роли другую со всеми ее правами. Это упрощает управление не только правами пользователей, но и самими ролями, создавая, к примеру, сходные между собой группы ролей.

SQL Server также поддерживает так называемые роли для приложений (*applicationroles*), которые могут использоваться для ограничения доступа к объектам базы данных в тех случаях, когда пользователи обращаются к данным с помощью конкретных приложений. В отличие от обычных ролей, роли для приложений, как правило, неактивны и не могут быть присвоены пользователям. Их применение оказывается удобным в том случае, когда требования безопасности едины для всех пользователей, при этом не требуется аудит или иная регистрация деятельности конкретных пользователей в базе данных.

Рекомендации по управлению доступом

Ниже следует ряд несложных рекомендаций, касающихся применения средств управления доступом при создании и развертывании решений на основе SQL Server 2008.

Применение схем и ролей. Общие принципы

Одним из обязательных этапов проектирования баз данных должно быть детальное определение способов доступа к объектам базы данных. В частности, на этом этапе необходимо определить, каким образом использовать схемы для группировки объектов базы данных с точки зрения доступа к ним, какие роли создать, определить возможные группы пользователей и роли, которые следует им присвоить. Кроме того, на этом же этапе важно принять решение о том, как будут использоваться представления и хранимые процедуры для обеспечения

безопасного доступа к данным (например, для ограничения непосредственного доступа к таблицам).

В последнее время настоятельно рекомендуется применять роли во всех решениях (как было сказано выше, это заметно снижает затраты на процесс управления правами доступа и внесение в них изменений), а в случае наличия пользователей, имеющих частично совпадающие наборы прав доступа, рекомендуется применять вложенные роли. Для определения прав доступа и правил вложенности ролей следует разделить пользователей на группы с разными правами доступа и понять, можно ли сформулировать правила вложенности групп, а затем отразить эти сведения в проектируемых ролях.

Нужно также проанализировать требования к правам доступа, предъявляемые клиентскими приложениями, и учесть эти требования при проектировании ролей и создании схем. Не стоит при этом забывать и о том, что списки пользователей и ролей следует не только единожды создать, но и постоянно актуализировать. Помимо вопросов управления доступом в клиентских приложениях, рекомендуется уделить внимание и управлению доступом в различных службах SQL Server.

Управление доступом в службах отчетов

При применении служб отчетов (*ReportingServices*) необходимо отдельно позаботиться о правилах доступа к объектам базы данных для пользователей, применяющих приложения, обращающиеся к этим службам. Так, указанной категории пользователей должны быть доступны как сами описания отчетов, так и те объекты базы данных, на основании которых эти отчеты генерируются, и в этом случае наиболее уместно создание отдельной роли, обладающей указанными правами. Следует заметить, что по умолчанию службы отчетов используют службы InternetInformationServices (*IIS*) и средства безопасности Windows для аутентификации пользователей на сервере отчетов. Данный режим считается наиболее безопасным, поскольку может позволить запретить анонимный доступ к web-приложениям, выполняющимся под управлением IIS.

Отметим, что сервисы отчетов могут выполняться в режиме IntegratedSecurity. Но в этом случае они выполняют код ряда компонентов с теми же привилегиями, что и у пользователя, сгенерировавшего отчет, а это позволит пользователю, исполняющему отчет, получить более высокие привилегии, нежели ему положены. Поэтому данный режим не рекомендуется применять вместе со службами отчетов.

Управление доступом в службах уведомлений

Службы уведомлений (*NotificationServices*) выполняются от имени собственной учетной записи, которая, с одной стороны, должна обладать определенными правами для обеспечения доставки уведомлений, с другой стороны, указанный набор прав должен быть минимально необходимым (как минимум, такая учетная запись не должна входить в группу Administrators).

Отметим, что для служб уведомлений существуют специальные роли — NSEventProvider, NSGenerator, NSDistributor, которые следует присваивать

учетным записям, отвечающим запровайдеры, генераторы и механизмы рассылки, а также роль NSRunService, включающая в себя три перечисленные роли и применяющаяся в том случае, если все составные части служб уведомлений выполняются внутри одного ядра базы данных.

Управление доступом в интеграционных службах

Для управления доступом в интеграционных службах в SQL Server 2008 введены новые роли: db_dtsadmin, db_dtsltuser и db_dtsoperator. Они позволяют осуществлять контроль доступа к пакетам интеграционных служб, хранящихся в базе данных msdb.

Задания

1 Изучить теоретические сведения.

2 В соответствии с вариантом задания выполнить шифрование данных с помощью:

- симметричного ключа;
- сертификата.

3 В соответствии с вариантом задания выполнить настройку привилегий пользователей

Порядок выполнения работы

Для создания сертификата базы данных testing:

1) Для начала создадим ключ базы данных:

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = ‘1234’;
```

2) Затем создадим сертификат для шифрования:

```
CREATE CERTIFICATE dbcert WITH SUBJECT = ‘certtest’;
```

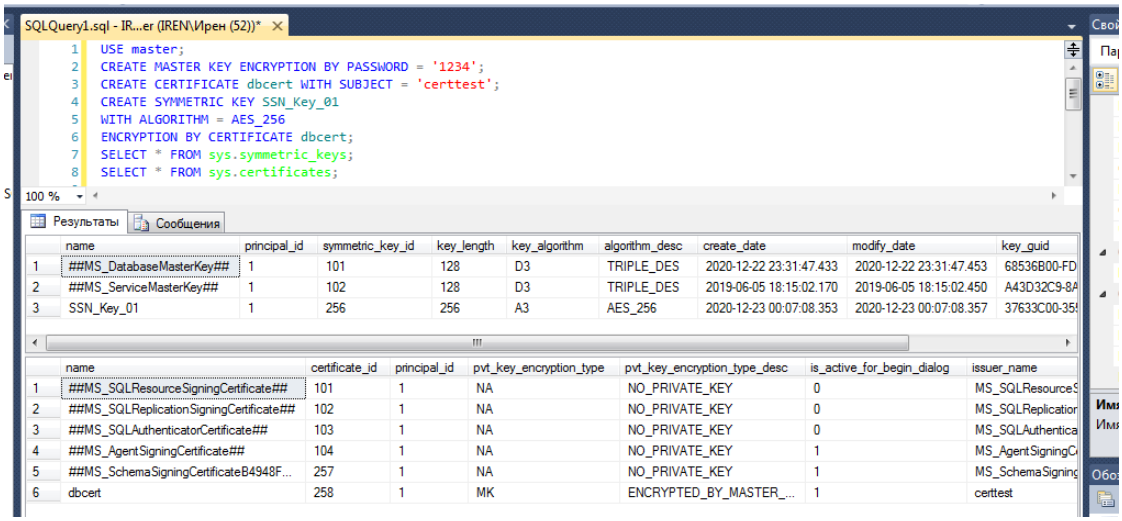
3) Далее создадим симметричный ключ шифрования с помощью сертификата:

```
CREATE SYMMETRIC KEY SSN_Key_01
WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE dbcert;
GO
```

Проверить наличие ключей и сертификатов можно, выполнив запросы:

```
SELECT * FROM sys.symmetric_keys;
```

SELECT * FROM sys.certificates



4) После того как все создали, необходимо сделать резервные копии наших ключе и сертификатов:

Создание резервной копии главного ключа базы данных

```
BACKUP MASTER KEY TO FILE = 'c:\adminbd'
ENCRYPTION BY PASSWORD = 'password@1' ;
GO
```

5) После этого можно шифровать\дешифровать данные в таблице.

Но перед тем как использовать шифрование\дешифрование, необходимо открыть ключ шифрования.

Пример вставки с шифрованием:

```
OPEN SYMMETRIC KEY SSN_Key_01
DECRYPTION BY CERTIFICATE dbcert;

INSERT INTO [dbo].[servicelogins]
        ([login]          ,[srv]                  ,[encrpsw]
         ,[bd]          ,[description]          ,[owner]          )
VALUES
         ('test_login'
        ,'test_srv'
        ,EncryptByKey(Key_GUID('SSN_Key_01'),'testpassword' )
        ,'test bd'
        ,'[description]'
        ,'[owner]'
        )
```

Чтение данных:

```
OPEN SYMMETRIC KEY SSN_Key_01
DECRYPTION BY CERTIFICATE cert1;

select  [login]
      ,[srv]
  ,[encrpsw]
  ,convert(char,DecryptByKey([encrpsw])) as p
  ,[bd] ,owner
from [dbo].[servicelogins]
go
```

Ключ открывается на время сеанса.

При чтении таблицы без ключа ошибки не будет, но данные, соответственно, будут в зашифрованном виде.

Самое, главное при создание резервной копии и восстановлении её на другом сервере, база данных будет восстановлена, данные будут, сертификаты и ключи базы данных так же будут восстановлены, но читать с помощью их данные не получится, т.к главный ключ службы другой. Для этого необходимо восстановить главный ключ из резервной копии:

```
RESTORE SERVICE MASTER KEY FROM FILE = 'c:\service_master_key'
DECRYPTION BY PASSWORD = 'Password@1' force
```

Параметр ‘Force’ указывает, что можно заменить текущий ключ. Внимание, если на сервере есть зашифрованные данные текущего мастера ключа, то они будут недоступные при восстановлении нового ключа. Данный факт стоит учитывать при резервном копировании и восстановлении.

Создание пользователей и настройка привилегий

На первый взгляд, в MS SQL нестандартная система учетных записей. В ней существуют логины и пользователи.

Логины - это понятие всего сервера SQL. У них могут быть права на какие-то административные задачи (например бэкап). Им можно дать права на какие-то действия с базой или таблицей. Для того что бы у логина были права на базу на основе него создается пользователь. Они могут быть как SQL логином, так и созданные на основе существующих пользователей Windows или AD. Например, тот пользователь, который устанавливал MS SQL так же добавился в список логинов. Рекомендуется использовать логины на основе Windows или AD, а не SQL логины.

Пользователи - это понятие в рамках одной базы. Им нельзя дать права по административной части, но можно дать права на работу с базой.

Оба типа учетных записи можно создать как в графике, так и с помощью запроса T-SQL

CREATE USER, CREATE LOGIN

Перейдем в базу данных, в которой мы работаем:

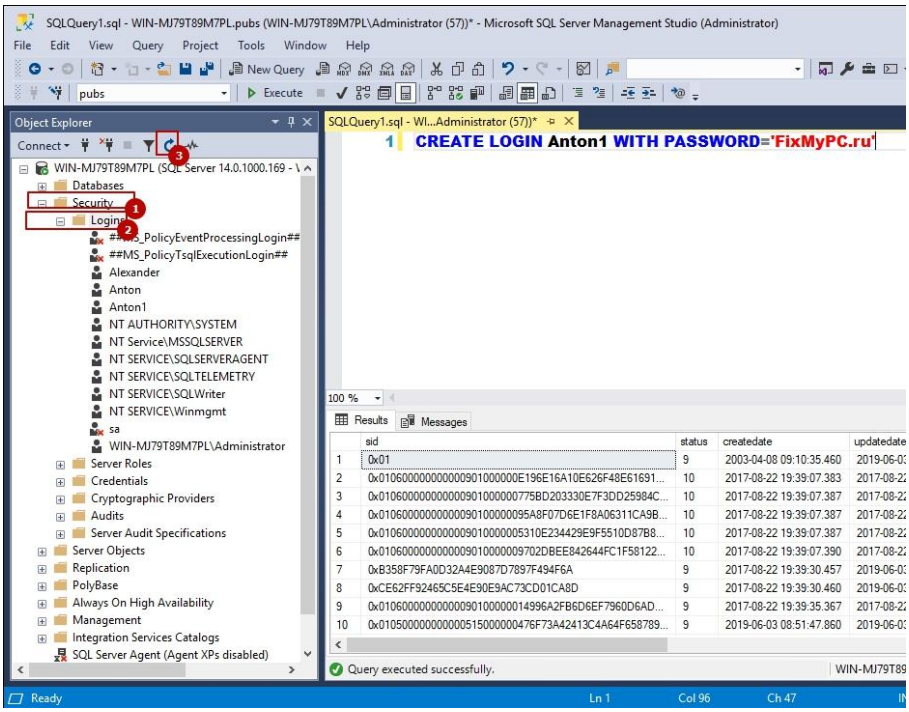
```
USE pubs
```

Для создания логина SQL выполним:

```
CREATE LOGIN Anton WITH PASSWORD='Password1910'
```

Anton - имя логина. Password1910 - пароль. У этого логина уже будут права на вход.

Посмотреть что этот логин создался можно так:



Для создания пользователя привязанного к логину нужно выполнить:

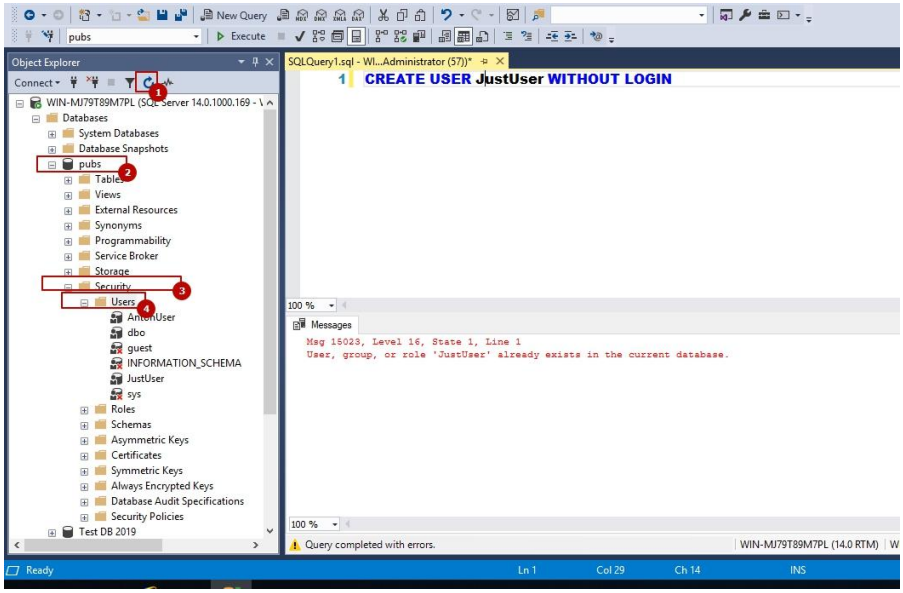
```
CREATE  USER AntonUser FOR LOGIN Anton
```

Пользователь создастся в той базе, откуда мы выполняем запрос. Чтобы просто создать пользователя нужно:

```
CREATE USER JustUser WITHOUT LOGIN
```

Где JustUser - просто пользователь

Созданных пользователей можно увидеть так:



Пользователи создаются без каких либо прав на базу.

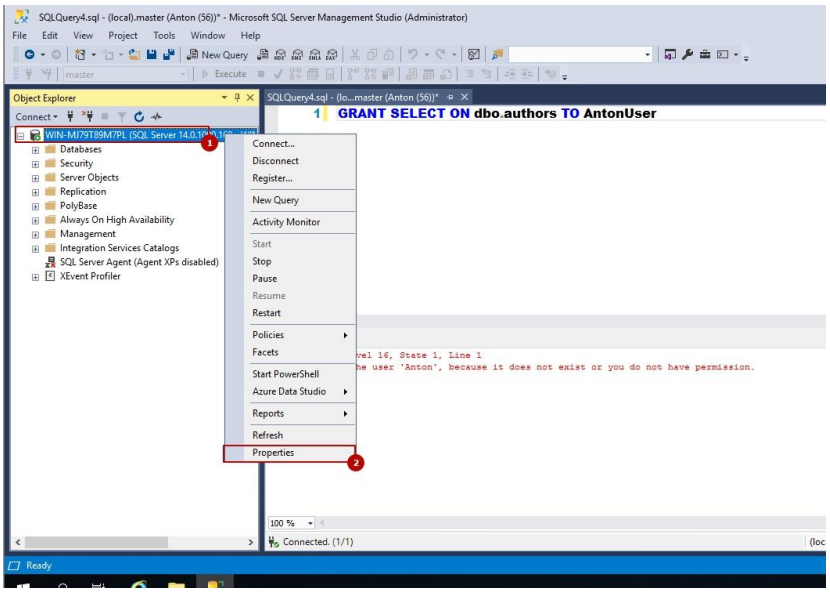
GRANT

GRANT в пер. "Разрешение" дает права на какое-то действие.

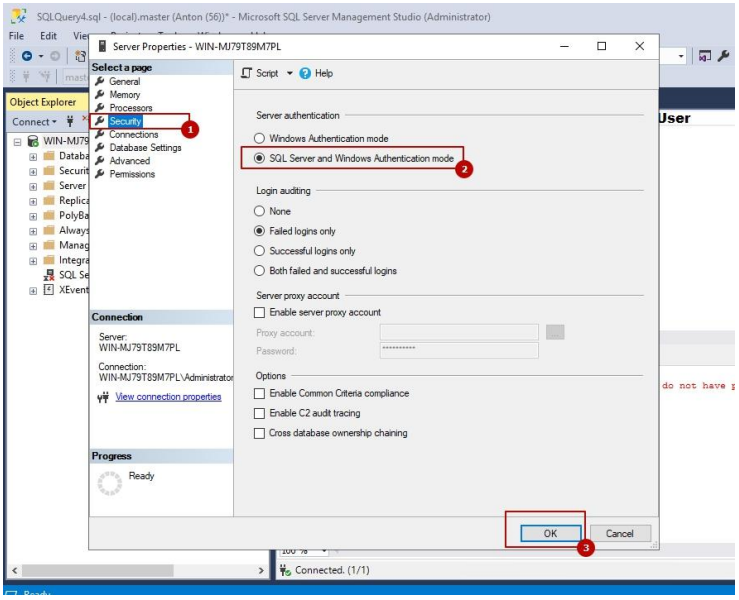
Что бы дать пользователю право получать данные из какой-то таблицы нужно выполнить:

```
GRANT SELECT ON dbo.authors TO AntonUser
```

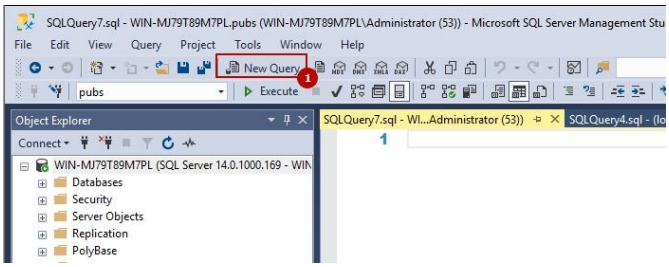
Если мы хотим зайти под созданным пользователем нам нужно выполнить ряд действий. Дело в том, что по умолчанию аутентификация по логинам SQL отключена (разрешена только Windows). Для этого зайдем в свойства сервера:



Затем в "Безопасность" и включи аутентификация SQL

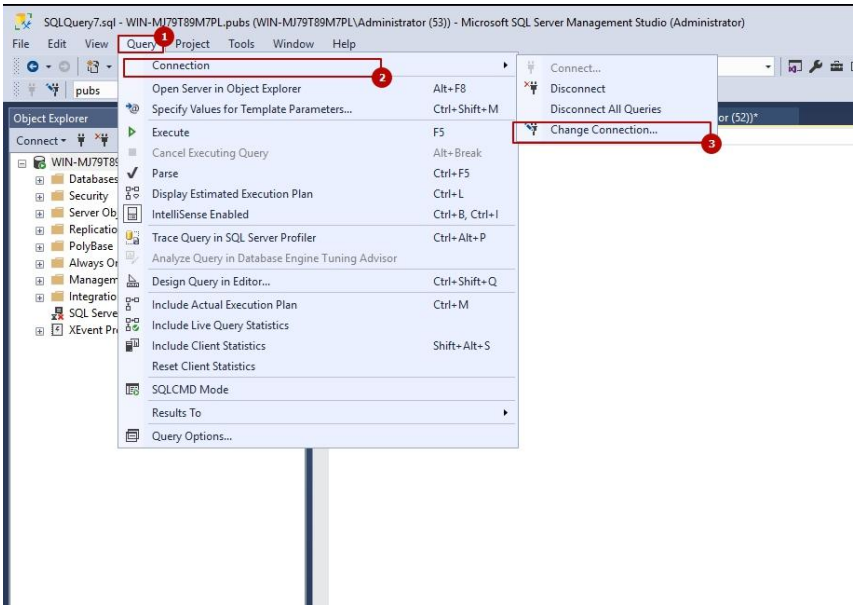


Теперь мы сможем зайти под логином SQL. Для этого сделаем следующее - откроем новое окно запросов Ctr + N или нажмем кнопку:



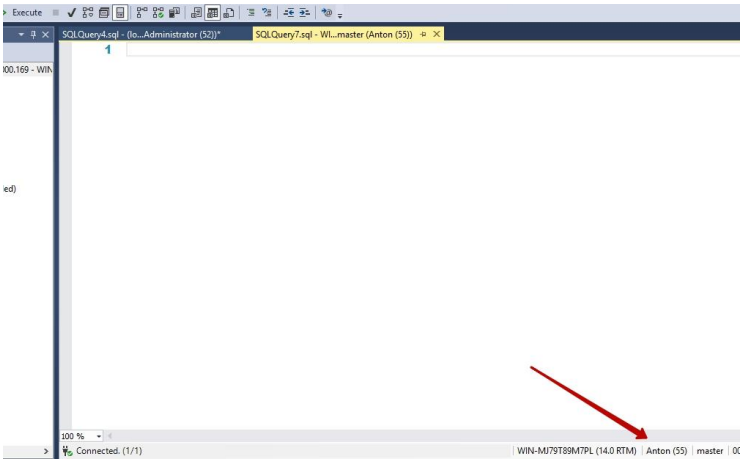
Важно сделать так, что бы у нас было 2 окошка для запросов до того как мы сменим пользователя. Благодаря этому у нас каждое окно будет работать под разными пользователями

Затем окно подключения пользователя:



Меняем тип подключения на SQL и вводим логин/пароль. Я создавал логин Anton, с паролем Password1910.

После подключения можем увидеть, что одно окно работает под логином Anton, а другое под учетной записью с помощью которой мы проводили операции выше. Так же обратите внимания, что новые пользователь подключен к другой базе (master).



Перейдем к базе, на которую мы давали права. В моем случае это база pubs

```
USE pubs
```

Права на SELECT даны для таблицы dbo.authors. Проверем работу:

```
SELECT *
FROM dbo.authors
```

Запрос дожен пройти успешно.
Для другой таблицы, на которую мы не давали прав:

```
SELECT *
FROM dbo.titles
```

Получим ошибку

```
The SELECT permission was denied on the object 'titles', database 'pubs', schema 'dbo'.
```

Что бы отозвать разрешение на SELECT нужно заменить GRANT на REVOKE (не забудьте переключиться на окно, у которого есть разрешение или переключиться на предыдущего пользователя):

```
REVOKE SELECT ON dbo.authors TO AntonUser
```

Разрешений, которые мы можем выдать пользователю достаточно много. Это основные:

ALTER	разрешение на изменение
DELETE	разрешение на удаление
EXECUTE	разрешение на исполнение функций и процедур
INSERT	разрешение на вставку в таблицу
SELECT	разрешение на получение данных из таблицы
UPDATE	разрешение на изменение данных в таблице

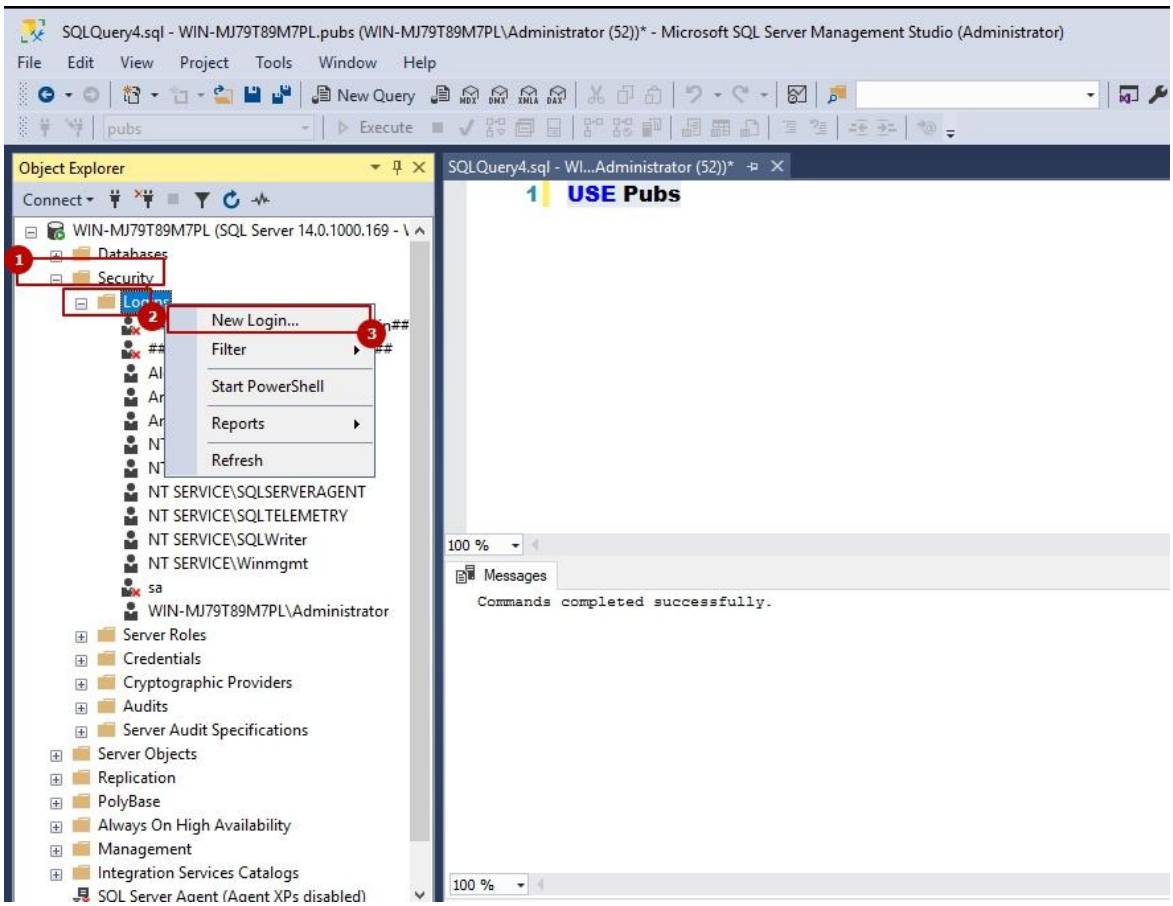
Мы можем так же явно запретить пользователю выполнять определенные запросы. Для этого есть DENY:

```
DENY SELECT ON dbo.authors TO AntonUser
```

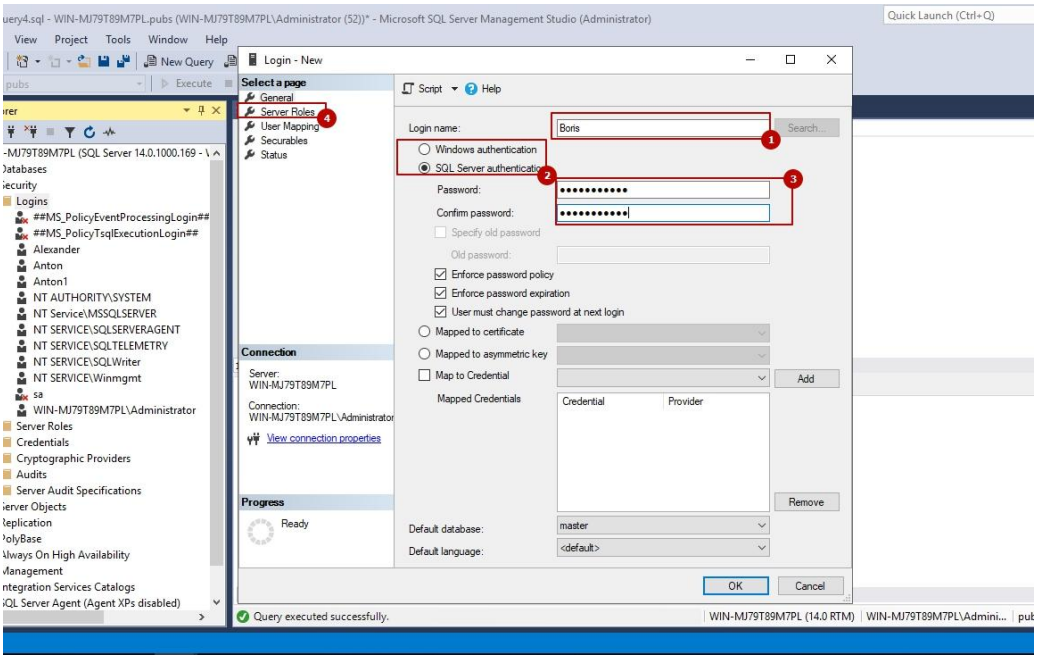
Чтобы дать разрешение на создание таблиц:

```
USE Pubs;
GRANT CREATE TABLE TO Anton;
```

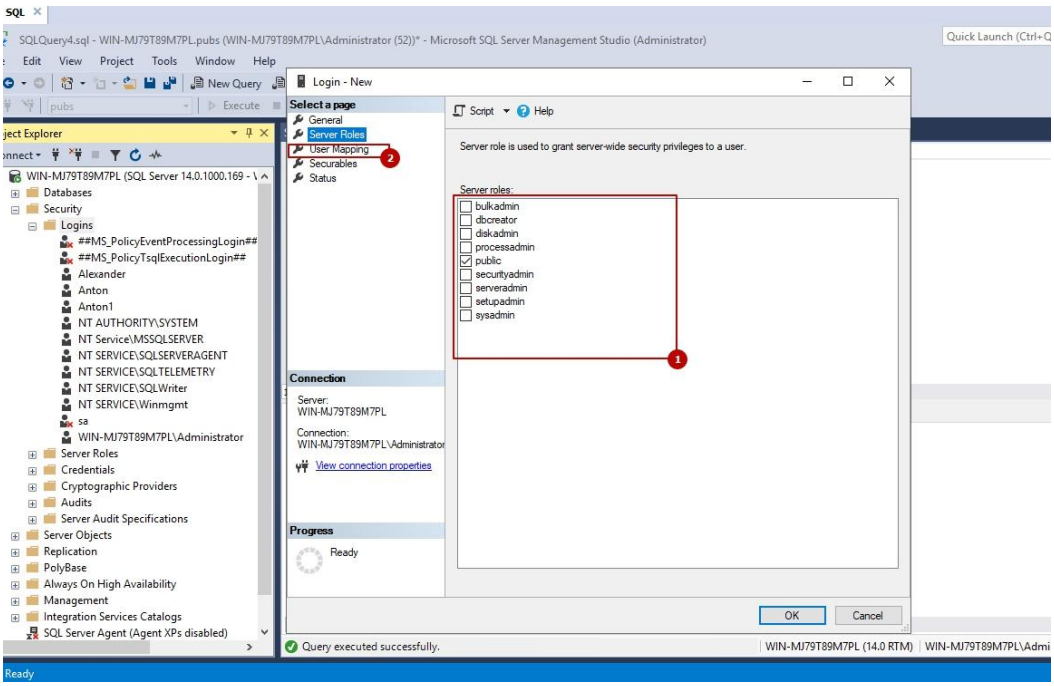
Теперь создадим логин через графический интерфейс. Нажмем следующие кнопки:



В новом окне мы должны заполнить логин (1) и пароль (3). Под цифрой 2 мы можем увидеть выбор способа аутентификации. В случае Windows мы должны будем выбрать локального пользователя или пользователя AD. Перейдем на закладку Server Roles (4).



Перед нами находятся роли уровня сервера. Все эти роли это просто набор прав. В Microsoft SQL мы так же можем создать и свои группы. Public - роль по умолчанию для всех, кому разрешено вход. Из всех созданных ролей только public можно изменять.

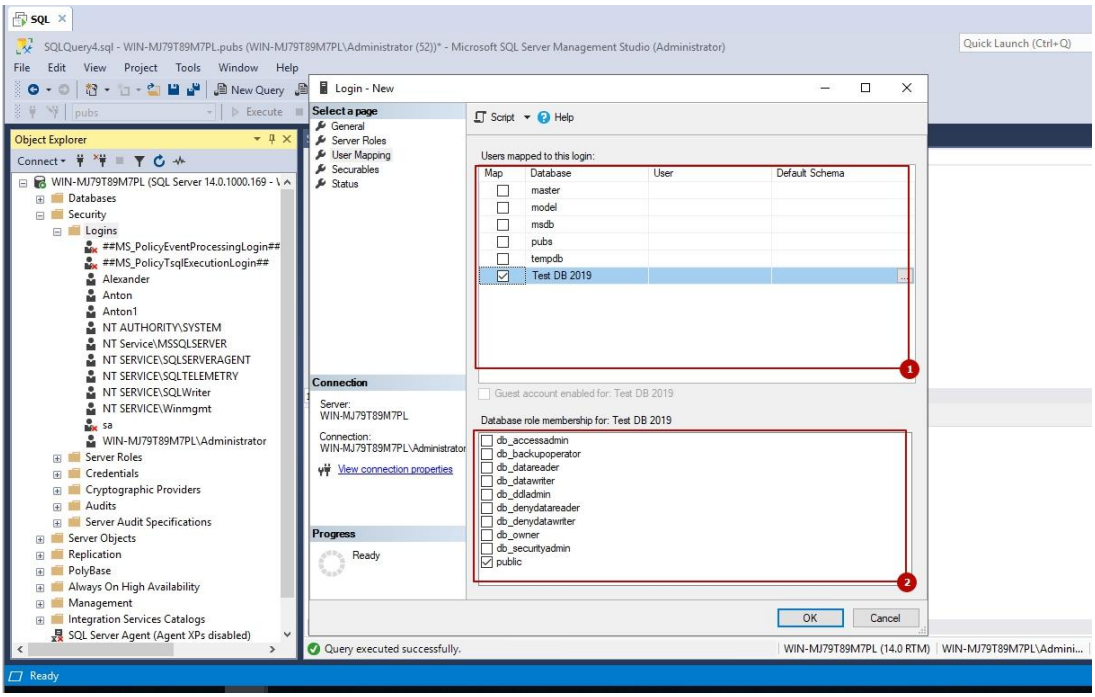


Вот описание самых популярных ролей:

sysadmin	Можно выполнять любые действия в MS SQL
serveradmin	Настройка сервера, включение и выключение.
dbcreator	Создание, изменение и удаление любых баз данных
processadmin	Может завершать любые процессы

Перейдем на следующую вкладку (2)

На этой закладке можно сразу создать пользователя и привязать его к базе данных. На этой закладке отображаются 4 базы, которые вы скорее всего раньше не видели. Это системные базы данных, которые хранят в себе транзакции, структуру подключенных баз, скрипты агента и т.д. В области 2 можно дать права на выделенную базу данных.



Коротко о нескольких ролях баз данных:

db_owner	Может выполнять все действия с базой
db_backupoperator	Бэкап базы данных
db_datareader	Может читать все данные

Роль public, в отличии от всех остальных, можно изменять. Т.е. можно дать роли public разрешение на SELECT и тогда все пользователи с логином смогут это делать. По умолчанию эта роль может делать запросы типа:

SELECT 1+1

Таким образом, существуют различные способы защиты данных в информационных системах, позволяющих обеспечить их бесперебойную работу.

Содержание отчета

1 Название работы
2 Цель работы
3 Перечень технических средств обучения
4 Порядок выполнения работы
5 Вывод

Варианты заданий

Варианты заданий представлены в практической работе № 13.

Используемая литература

– Г.Н.Федорова Основы проектирования баз данных. М.: Академия, 2020
– Г.Н.Федорова Разработка, администрирование и защита баз данных. М.: Академия, 2018
– http://dbasimple.blogspot.com/2013/07/ms-sql-server.html
– http://sd-company.su/article/sql/grant_sql_server
– https://www.osp.ru/winitpro/2013/05/13035359#list1