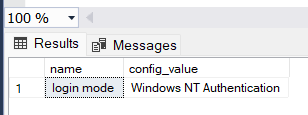
**Лабораторная работа 3. Безопасность SQL Server**

Задание (выполняется посредством ввода команд на Transact-SQL):

Измените, если это возможно, режим аутентификации SQL-сервера на смешанный режим.

--Текущий режим аутентификации SQL-сервера

EXEC master.dbo.xp\_loginconfig 'login mode'



--Изменение на смешанный режим

USE [master]

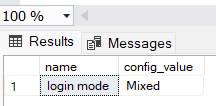
GO

EXEC xp\_instance\_regwrite N'HKEY\_LOCAL\_MACHINE',

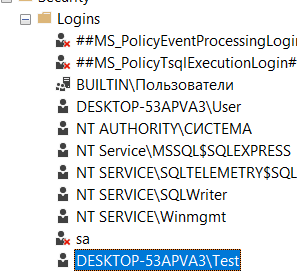
N'Software\Microsoft\MSSQLServer\MSSQLServer',

N'LoginMode', REG\_DWORD, 2

GO



Создайте локальную учётную запись пользователя Windows на рабочей станции с SQL-сервером.



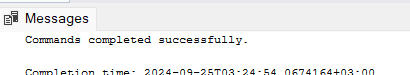
Разрешите вход на SQL-сервер под учётными данными созданного пользователя в режиме Windows-аутентификации.



Создайте логин для входа на сервер в режиме SQL-аутентификации.

CREATE LOGIN SQLNewLogin WITH PASSWORD = '1234567890';

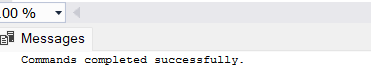
GO

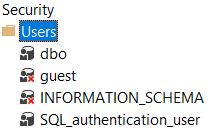


В тестовой базе данных создайте двух пользователей, соответствующих созданным ранее логинам.

USE TestDB2

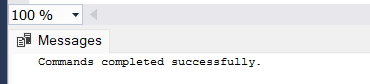
CREATE USER SQL\_authentication\_user FOR LOGIN SQLNewLogin





USE TestDB2

CREATE USER Windows\_authentication\_user FOR LOGIN [DESKTOP-53APVA3\Test]





Протестируйте подключение к тестовой базе данных с использованием созданных логинов. Выведите при помощи запроса название логина, который использовался при аутентификации на сервере, а также название соответствующего пользователя базы данных.

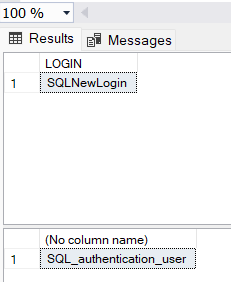
PRINT ORIGINAL\_LOGIN();

GO

SELECT SUSER\_NAME() AS 'LOGIN';

GO

SELECT CURRENT\_USER



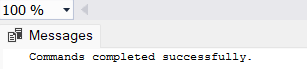
Разрешите доступ на чтение любой из таблиц первому пользователю, а второму пользователю – запуск любой созданной Вами хранимой процедуры. Протестируйте выданные разрешения.

USE TestDB2

GO

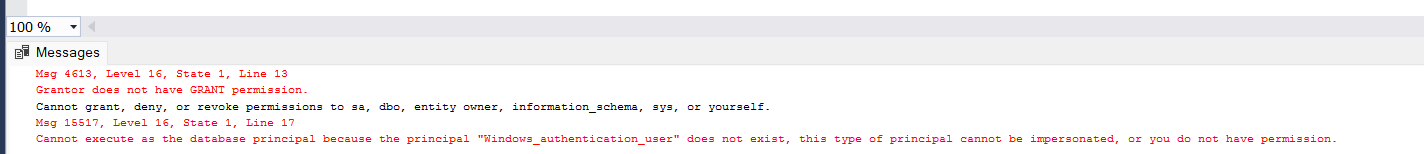
GRANT EXECUTE ON database::TestDB2 TO Windows\_authentication\_user;

GRANT SELECT ON database::TestDB2 TO SQL\_authentication\_user;



EXEC AS USER = 'Windows\_authentication\_user'

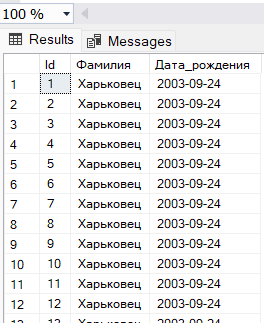
SELECT \* FROM Сотрудники



USE TestDB2

EXEC AS USER = 'SQL\_authentication\_user'

SELECT \* FROM Сотрудники



CREATE PROCEDURE SAY\_HELLO

AS

BEGIN

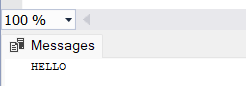
PRINT 'HELLO'

END

USE TestDB2

EXEC AS USER = 'Windows\_authentication\_user'

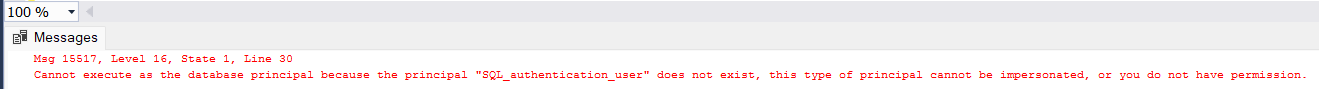
EXEC SAY\_HELLO



USE TestDB2

EXEC AS USER = 'SQL\_authentication\_user'

EXEC SAY\_HELLO



Создайте роль «students» и включите в неё двух созданных пользователей. Выдайте разрешение этой роли вставлять записи в ранее использованную таблицу.

USE TestDB2

CREATE ROLE students;

GO

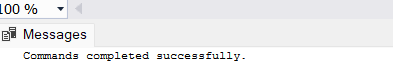
ALTER ROLE students ADD MEMBER [Windows\_authentication\_user];

GO

ALTER ROLE students ADD MEMBER [SQL\_authentication\_user];

GO

GRANT INSERT ON database::TestDB2 TO students;



USE TestDB2

EXEC AS USER = 'SQL\_authentication\_user'

INSERT INTO Сотрудники

VALUES ('Васнецова','2001-08-09');

SELECT \* FROM Сотрудники



Задайте явный запрет вставки записи в ранее использованную таблицу второму пользователю.

USE TestDB2

DENY INSERT ON Сотрудники to Windows\_authentication\_user;

GO

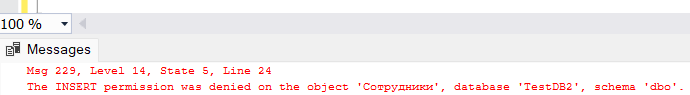


USE TestDB2

EXEC AS USER = 'Windows\_authentication\_user'

INSERT INTO Сотрудники

VALUES ('Трушин','2003-02-11');



Протестируйте все ранее выданные разрешения под учётными данными обоих созданных пользователей.



USE TestDB2

EXEC AS USER = 'SQL\_authentication\_user'

INSERT INTO Сотрудники

VALUES('Мальцев', '1999-06-10')

SELECT \* FROM Сотрудники



Создайте роль приложения с произвольным именем и выдайте этой роли разрешение на удаление и чтение записей в используемой таблице.

USE TestDB2

GO

GRANT DELETE ON Сотрудники TO Add\_role\_user;

GO

GRANT SELECT ON Сотрудники TO Add\_role\_user;

GO

USE TestDB2

GO

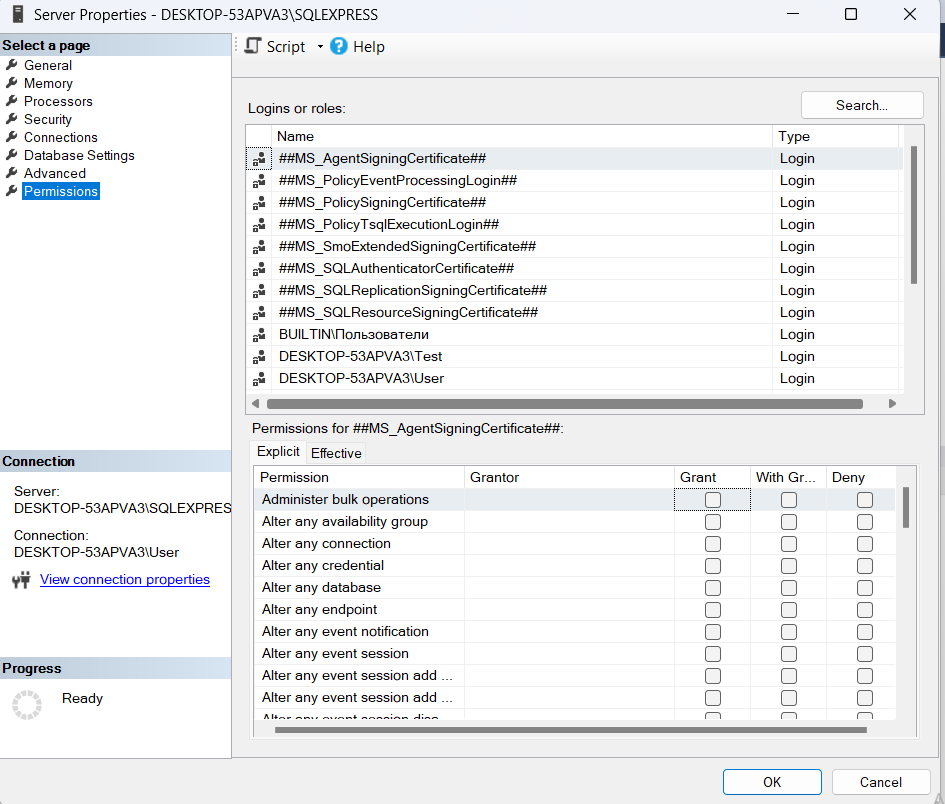
EXECUTE sys.sp\_setapprole 'Add\_role\_user', 'role\_user';

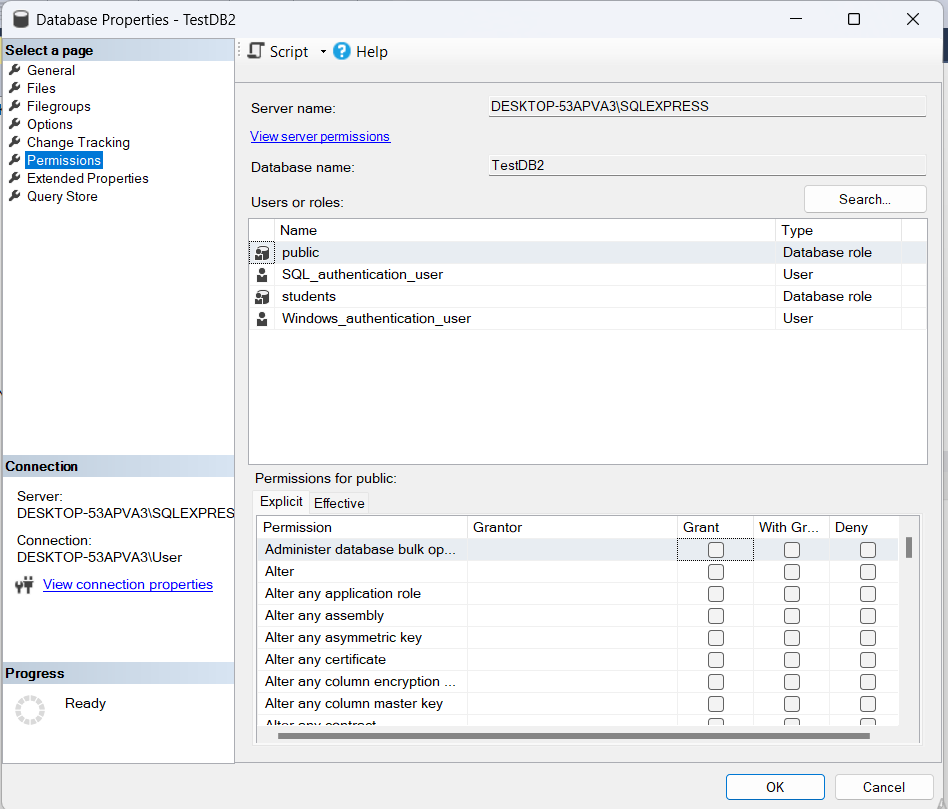
CREATE APPLICATION ROLE Add\_role\_user

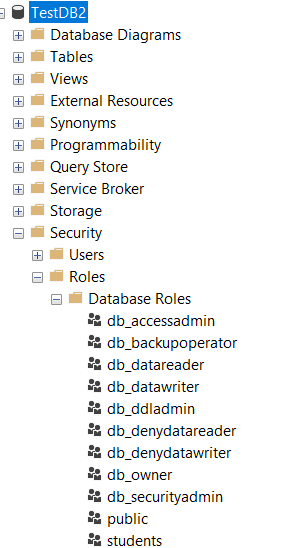
WITH PASSWORD = 'role\_user';

GO

Продемонстрируйте технологию выполнения вышеперечисленных шагов при помощи графического интерфейса Management Studio.







Продемонстрируйте на простейшем примере использование созданной роли приложения. Для этого создайте простой проект с использованием технологии ADO.NET, JDBC, либо любой другой аналогичной. Можно использовать любой язык программирования. Продемонстрируйте факт того, что приложение действует в контексте созданной Вами роли приложения, а также то, что приложению разрешено выполнять операции, разрешённые для созданной Вами роли приложения и не разрешено выполнять другие операции.

Вопросы к отчёту:

1. Что представляет собой концепция трёхуровневой модели безопасности?

2. Какие режимы аутентификации сервера существуют?

1. Концепция трёхуровневой модели безопасности заключается в разделении безопасности на уровни сервера, базы данных и приложения для обеспечения комплексной защиты информации и доступа к данным.

2. Существуют два основных режима аутентификации сервера: Windows-аутентификация, основанная на учетных записях Windows, и SQL-аутентификация, где учетные записи хранятся непосредственно в базе данных.

3. Какие преимущества и недостатки у Windows-аутентификации и SQL-аутентификации?

3. Преимущества Windows-аутентификации включают интеграцию с существующей системой учетных записей Windows и повышенную безопасность, так как необходимо вводить пароль только один раз. Однако недостатком является ограничение в возможности работы с удаленными пользователями. SQL-аутентификация обеспечивает гибкость при работе с удаленными пользователями, но может быть менее безопасной из-за хранения учетных данных в базе.

4. Какая концепция трёхуровневой модели безопасности применяется в SQL Server?

4. В SQL Server применяется концепция разделения безопасности на уровни сервера, базы данных и приложения, где каждый уровень имеет свои механизмы защиты и контроля доступа.

5. Как осуществляется настройка логинов на уровне сервера?

5. Логины на уровне сервера настраиваются путем создания учетных записей, назначения прав доступа и настройки аутентификационных параметров для обеспечения безопасного доступа к серверу.

6. Что представляют собой серверные роли и каково их назначение?

6. Серверные роли представляют собой наборы прав и привилегий, которые назначаются группе пользователей для упрощения управления доступом к ресурсам сервера и выполнения определенных задач.

7. Как осуществляется создание, настройка и поддержка пользователей на уровне базы данных?

7. Пользователи на уровне базы данных создаются путем добавления учетных записей к базе данных, назначения прав доступа к объектам и обеспечения безопасности данных. Поддержка пользователей включает в себя регулярное администрирование прав доступа и обновление учетных данных.

8. Что представляют собой роли базы данных и каково их предназначение?

8. Роли базы данных представляют собой наборы прав и привилегий, которые назначаются группе пользователей для упрощения управления доступом к объектам базы данных и выполнения определенных задач.

9. Как осуществляется настройка прав доступа на уровне объектов базы данных?

9. Настройка прав доступа на уровне объектов базы данных выполняется с помощью операторов Grant, Deny и Revoke для предоставления, запрещения или удаления прав на выполнение операций с объектами. Взаимодействие разрешений позволяет определить конечные права доступа для пользователя или группы пользователей.

10. Чем отличаются роли приложений от пользовательских ролей и в каких случаях их следует использовать?

10. Роли приложений предназначены для управления доступом к определенным функциональным возможностям приложения, в то время как пользовательские роли определяют права и привилегии на уровне базы данных. Роли приложений следует использовать для ограничения доступа к определенным функциями приложения, в то время как пользовательские роли для управления доступом к данным в базе данных.