Задание 1

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

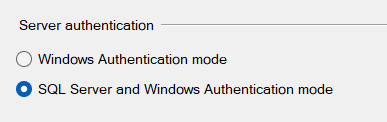
use master

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

N'Software\Microsoft\MSSQLServer\MSSQLServer',

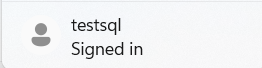
N'LoginMode', REG\_DWORD, 2

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



Задание 2

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



Задание 3

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

CREATE LOGIN [XAMEL24\testsql] FROM WINDOWS

Задание 4

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

CREATE LOGIN TmpLogin WITH PASSWORD = '123456'

Задание 5

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

USE TestDb

CREATE USER winUser FOR LOGIN [XAMEL24\testsql]

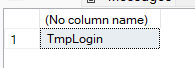
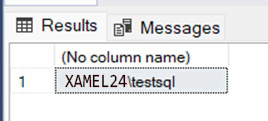
CREATE USER sqlUSER FOR LOGIN TmpLogin

Задание 6

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.

select \* from master.sys.server\_principals

SELECT SUSER\_NAME()

Задание 7

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

USE TestDb

GRANT SELECT ON DATABASE::TestDb to sqlUSER

GRANT EXECUTE ON DATABASE::TestDb to winUser

exec as user = 'winUser'

SELECT \* FROM TestTable

Задание 8

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

USE TestDb

CREATE ROLE students

ALTER ROLE students add member winUser

ALTER ROLE students add member sqlUSER

GRANT INSERT ON DATABASE::TestTable to students

EXEC as user = 'sqlUSER'

INSERT INTO TestTable VALUES(666)

SELECT TOP 1 \* FROM TestTable ORDER BY id desc

Задание 9

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

DENY INSERT on TestTable to winUser

EXEC as user = 'winUser'

INSERT INTO TestTable VALUES(666)

Задание 10

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

EXEC as user = 'sqlUSER'

SELECT TOP 10 \* FROM TestTable

EXEC as user = 'sqlUSER'

UPDATE TestTable SET num = 0 WHERE id = 1

Задание 11

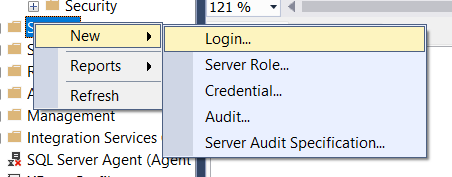
Создайте роль приложения с произвольным именем и выдайте этой роли разрешение на удаление и чтение записей в используемой таблице.

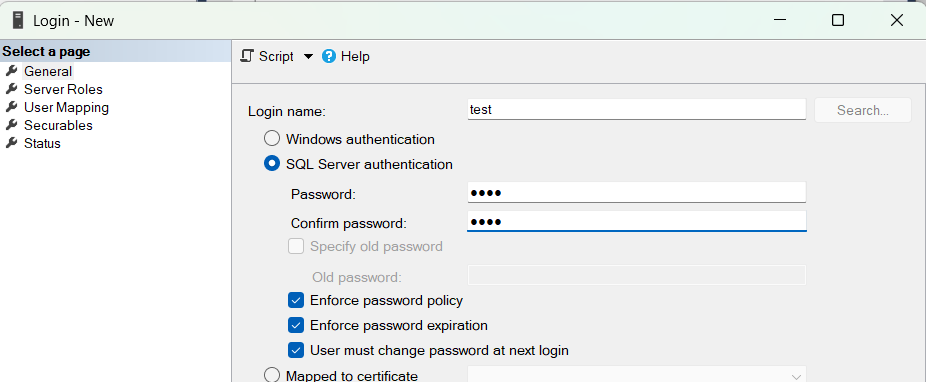
CREATE APPLICATION ROLE testRole WITH PASSWORD = '123456'

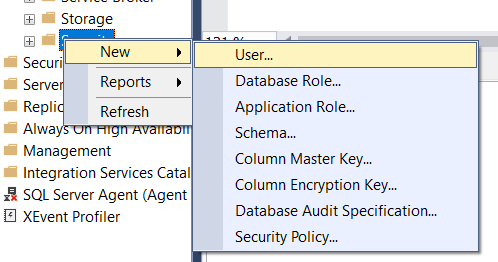
GRANT DELETE ON TestTable TO testRole

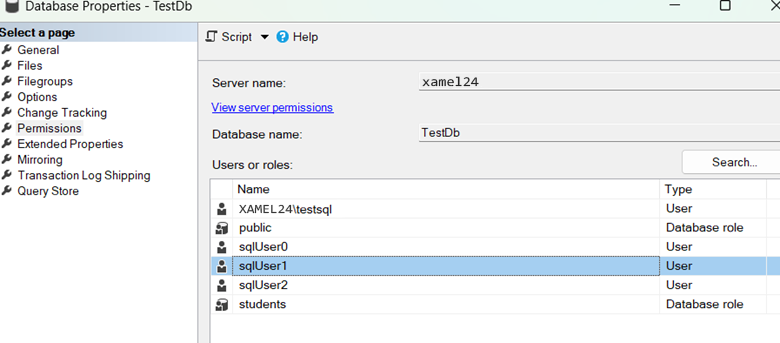
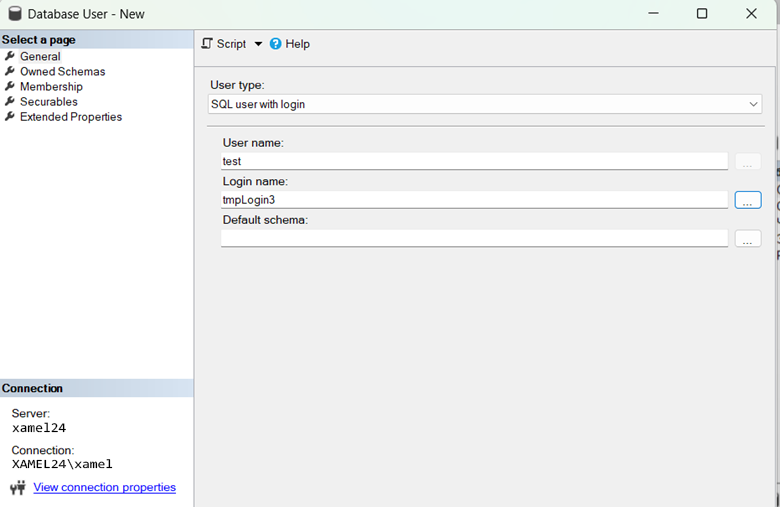
GRANT SELECT ON TestTable TO testRole

Задание 12

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

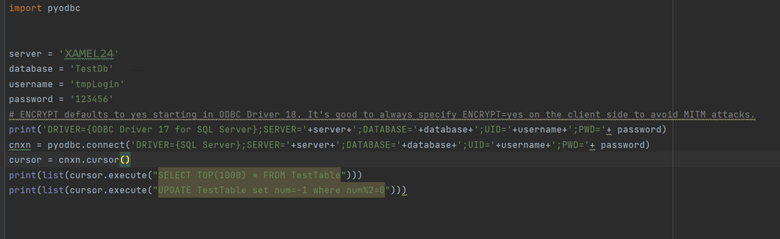


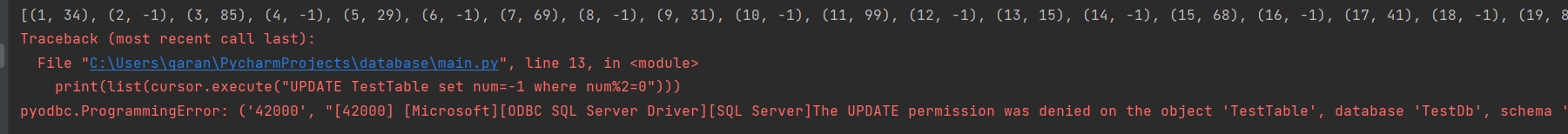




Задание 13

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.





№1. Концепция трёхуровневой модели безопасности.

Участники безопасности SQL Server 2005 - это такие сущности, как пользователи, учетные записи, группы и роли, которые могут обращаться к базе данных, серверу или к схеме. Группы и роли известны также как коллекции, поскольку могут включать в себя других участников безопасности.

В модели безопасности SQL Server 2005 участники безопасности существуют на трех уровнях: уровне Windows, уровне SQL Server и уровне базы данных.

1. Уровень Windows
2. Уровень SQL Server
3. Уровень схемы: пользователи, роли бд, роли приложений

№2. Режимы аутентификации сервера

* Windows Authentification mode (Режим аутентификации Windows) — этот режим, который выбирается по умолчанию при установке SQL Server 2005, разрешает использовать для подключения к серверу только логины Windows;
* SQL Server and Windows Authentification mode (Режим аутентификации SQL Server и Windows) — в этом режиме можно использовать оба типа логинов — и логины SQL Server, и логины Windows. Другое название этого режима — Mixed mode (Смешанный режим).

№3. Преимущества и недостатки Windows-аутентификации и SQL-аутентификации

Windows:

Когда вы получаете доступ к SQL Server с того же компьютера, на котором он установлен, вам не должно быть предложено ввести имя пользователя и пароль. А вы-нет,если используете аутентификацию Windows. При проверке подлинности Windows служба SQL Server уже знает, что кто-то вошел в операционную систему с правильными учетными данными, и использует эти учетные данные, чтобы позволить пользователю войти в свои базы данных. Конечно, это работает до тех пор, пока клиент находится на том же компьютере, что и SQL Server, или пока подключающийся клиент соответствует учетным данным Windows сервера. Аутентификация Windows часто используется как более удобный способ входа в экземпляр SQL Server без ввода имени пользователя и пароля, однако когда подключается больше пользователей или устанавливаются удаленные соединения с SQL Server, следует использовать аутентификацию SQL.

SQL Server:

SQL аутентификация-это типичная аутентификация, используемая для различных систем баз данных, состоящая из имени пользователя и пароля. Очевидно, что экземпляр SQL Server может иметь несколько таких учетных записей пользователей (используя аутентификацию SQL) с разными именами пользователей и паролями. На общих серверах, где разные пользователи должны иметь доступ к разным базам данных, следует использовать аутентификацию SQL. Кроме того, когда клиент (удаленный компьютер) подключается к экземпляру SQL Server на другом компьютере, чем тот, на котором работает клиент, требуется аутентификация SQL Server. Даже если вы не определяете никаких учетных записей пользователей SQL Server, во время установки добавляется корневая учетная запись - sa - с предоставленным вами паролем. Как и любая учетная запись SQL Server, она может быть использована для локального или удаленного входа в систему, однако если приложение выполняет вход и должно иметь доступ только к одной базе данных, настоятельно рекомендуется не использовать учетную запись sa, а создать новую с ограниченным доступом. В целом, аутентификация SQL является основным методом аутентификации, который будет использоваться, в то время как тот, который мы рассмотрим ниже - аутентификация Windows - является более удобным.

№4. Концепции трёхуровневой модели безопасности SQL-Server.

В модели безопасности SQL Server 2005 участники безопасности существуют на трех уровнях: уровне Windows, уровне SQL Server и уровне базы данных.

Уровень Windows

Участники безопасности на уровне Windows - это доменные имена входа Windows, локальные имена входа Windows и группы Windows. Для простоты управления и лучшей безопасности желательно везде, где только возможно, использовать в SQL Server 2005 участников безопасности Windows.

Уровень SQL Server

На уровне SQL Server участниками безопасности являются имена входа SQL Server и серверные роли.

Имена входа SQL Server обычно используются для внешних пользователей компании, например для тех, кто подключается к базе данных через веб-сайт. Кроме того, у каждого экземпляра SQL Server есть встроенное имя входа sa и могут также быть имена входа NETWORK SERVICE и SYSTEM (в зависимости от конфигурации экземпляра).

Уровень базы данных

На уровне базы данных участниками безопасности являются пользователи базы данных, роли приложения и роли базы данных. Пользователи базы данных - это сущности, связанные с именами входа Windows или SQL Server, которым назначен определенный набор разрешений и привилегий к отдельным объектам (например, к таблицам) базы данных.

№5. Логины на уровне сервера. Создание, поддержка, настройка.

1. В случае проверки подлинности Windows: нужно указать имя сервера и имя учетной записи домена windows.
2. В случае проверки подлинности SQL Server: указывается имя входа и пароль.
3. Создание пользователя
4. Присоединение роли
5. Выдача разрешений

№6. Серверные роли: основные концепции, предназначение.

Роль - это определенный набор прав, который можно назначить определенному пользователю или группе пользователей. В SQL Server есть созданные по умолчанию роли уровня сервера и уровня базы данных, которые имеют предопределенный набор разрешений, назначенных для них. Также можно создавать свои роли, для которых администратор может самостоятельно определить список разрешений и запретов. Созданные по умолчанию роли уровня сервера и роли уровня базы данных имеют предопределенный набор разрешений, каждый на своем уровне и они не могут быть изменены.

№7. Пользователи на уровне базы данных. Создание, настройка, поддержка.

В Microsoft SQL Server администратор базы данных добавляет учетные записи к экземпляру SQL Server, эти учетные записи сопоставляются с пользователями в отдельных базах данных в экземпляре SQL Server. Пользователи базы данных, которые создают таблицы и классы объектов, должны иметь соответствующие права доступа для создания объектов в базе данных и схему, в которой будут созданы эти объекты.

* Создать или добавить учетную запись к экземпляру SQL Server.
* Создать пользователя, соответствующего указанной учетной записи.
* Создать схему сопоставления для пользователя в указанной базе данных.
* Предоставить пользователю права доступа, необходимые для создания таблиц, классов объектов или представлений в указанной базе данных.

№8. Роли базы данных: основные концепции, предназначение, встроенные роли.

См. пункт 6.

В таблице ниже описаны роли уровня базы данных существующие во всех базах данных, а также кратно описаны права, которые дают эти роли . Все перечисленные ниже роли уровня БД встроенные, и их права не подлежат изменению.

db\_owner Эта роль дает право выполнять все действия по настройке и обслуживанию базы данных, а также удалять БД.

db\_securityadmin Роли db\_securityadmin дает возможность управлять ролями и разрешениями пользователей, в том числе своими .

db\_accessadmin Эта роль позволяет добавлять или удалять права удаленного доступа к БД.

db\_backupoperator Роль db\_backupoperator позволяет создавать резервные копии БД.

db\_ddladmin Эта роль позволяет выполнять любые команды Data Definition Language (DDL) в БД.

db\_datawriter Эта роль позволяет добавлять, удалять или изменять данные во всех таблицах.

db\_datareader Роль db\_datareader дает право считывать все данные из всех таблиц.

db\_denydatawriter Те, кому назначена эта роль, не могут добавлять, изменять или удалять данные в таблицах БД.

db\_denydatareader Обладатели этой роли, не могут считывать данные из таблиц базы данных.

№9. Настройка прав доступа на уровне объектов базы данных. Операторы Grant, Deny, Revoke. Взаимодействие разрешений.

GRANT. Предоставляет разрешения на защищаемый объект участнику. Общая концепция заключается в GRANT <some permission> ON <some object> TO <some user, login, or group>.

DENY. Запрещает разрешение для участника. Предотвращает наследование разрешения участником через его членство в группе или роли. DENY имеет приоритет над всеми разрешениями, но не применяется к владельцам объектов или членам с предопределенной ролью сервера sysadmin. Примечание по безопасности. Членам предопределенной роли сервера sysadmin и владельцам объектов не может быть отказано в разрешениях.

REVOKE. Удаляет разрешение, выданное или запрещенное ранее.

№10. Роли приложений. Предназначение и отличия от пользовательских ролей. Порядок их использования.

Альтернативный метод предоставления разрешений в базе данных SQL Server 2005 — воспользоваться ролью приложения (application role). Отличием этого метода является то, что разрешения предоставляются не пользователю, а приложению, из которого пользователь подключается к базе данных.

Применение этого способа выглядит так:

* пользователь от имени любого логина (которому должен соответствовать какой-нибудь объект пользователя в базе данных с правом CONNECT) подключается к серверу и делает нужную базу данных текущей;
* затем пользователь выполняет хранимую процедуру sp\_setapprole, чтобы активизировать указанную им роль приложения. При этом в базе данных он получает права этой роли приложения (и теряет свои текущие права);
* после того, как необходимость в правах роли приложения закончилась, пользователь может опять переключиться на свою исходную учетную запись (и получить соответствующие ей права). Эта новая возможность SQL Server 2005 обеспечивается при помощи хранимой процедуры sp\_unsetapprole.