



Organizational Security and Auditing



Contents

Overview	3
Create server role for managing audits (new permissions).....	4
Create server role for monitoring	17
SQL Server 2016 audit.....	26
Contained database authentication.....	39
Summary	45
Terms of use.....	46

Overview

Note: Estimated time to complete lab is 30 minutes.

Today's businesses require that everyone within an organization be able to access the information they need to best perform their roles. As businesses become more closely aligned with their suppliers and customers, this need for information expands to include growing numbers of these stakeholders as well. As a result, user access must be monitored and recorded to ensure that only the appropriate information is accessed.

But how do you guarantee that users access only the data and features required to perform their roles? Microsoft SQL Server 2016 can help you secure access with the following features:

- User-defined server roles
- New server and database permissions
- Audit resilience and security
- Contained database authentication

In this hands-on lab, you will learn how to confirm that newly installed audit features meet compliance requirements, and you'll learn how to check that user security aligns with your organization's security policies after a new application deploys.

As part of the implementation of the security policy database, permissions have been defined for the roles within the data administration team. After completing this lab, you will be able to ensure that these permissions have been implemented correctly.

Setting up your environment

1. Log on using the following account information:
Username: **Labuser**
Password: **SQLlabpass@word!**
2. During setup, you will need to record credentials and server locations. Open **Notepad.exe** to keep track of information.
3. On the lab machine, open SQL Server 2016 Management Studio.
4. You will be using the **AdventureWorks2016** database for this lab.
5. Locate the script file for this lab at **C:\SQL Server 2016 HOLs\Organizational Security and Auditing\Scripts**.

Note: A higher screen resolution will make it easier to use SQL Server 2016 Management Studio. If you have a monitor that supports a screen resolution larger than 1024 x 768, you can increase the screen resolution for the lab up to 1920 x 1080.

Create server role for managing audits (new permissions)

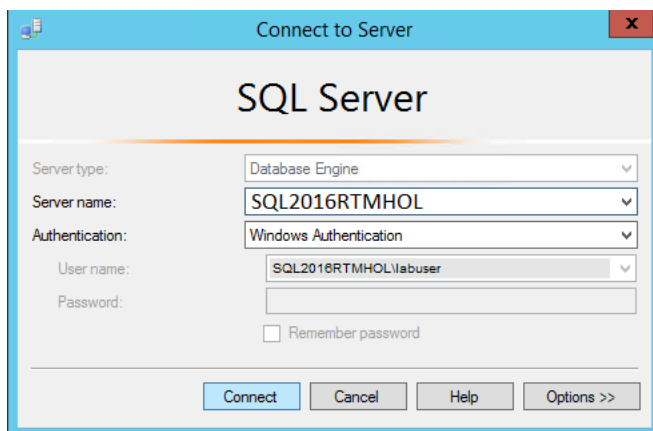
This section will help you create server roles using new permissions for managing audits. To limit the access of a person assigned the system administrator role, you will use the following new permissions available in SQL Server 2016:

- Alter a server audit object using the SQL Server Audit feature
- Connect any database using the Permissions feature
- Select all user securables and view any definition using Permissions

Alter any server audit

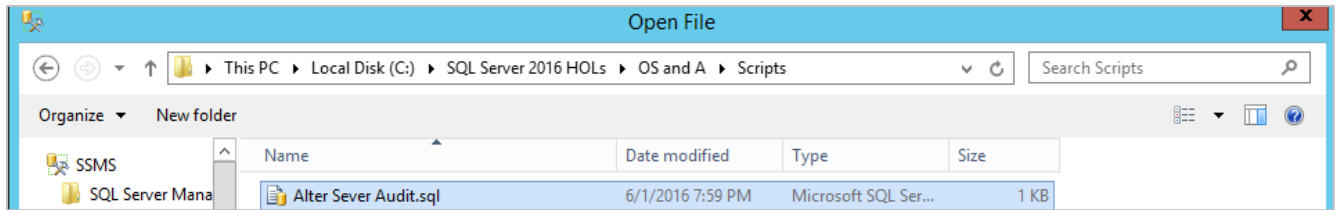
In this exercise, you will create an audit called **Monitor**. The SQL Server Audit object collects a single instance of server or database-level actions and groups of actions to monitor. The audit is at the SQL Server instance level. You can have multiple audits per SQL Server instance. When you define an audit, you specify the location for the output of the results: this is the audit destination. The audit is created in a disabled state and does not automatically audit any actions. After the audit is enabled, the audit destination receives data from the audit.

1. Open Microsoft SQL Server Management Studio with a connection to the **SQL2016RTMHOL** database engine instance. Select **Windows Authentication**, and then click **Connect**.



2. From the **File** menu, select **Open**, and then select **File**.

3. Go to **C:\SQL Server 2016 HOLs\Organizational Security and Auditing\Scripts** and select **Alter Server Audit.sql**.



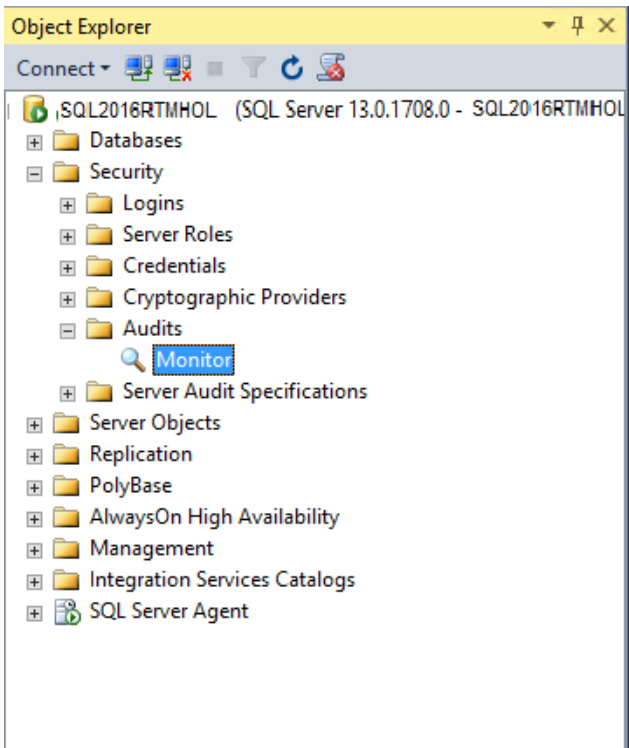
4. Click **Open**.



5. Click **Execute**.

Confirm that the server audit was created by following these steps:

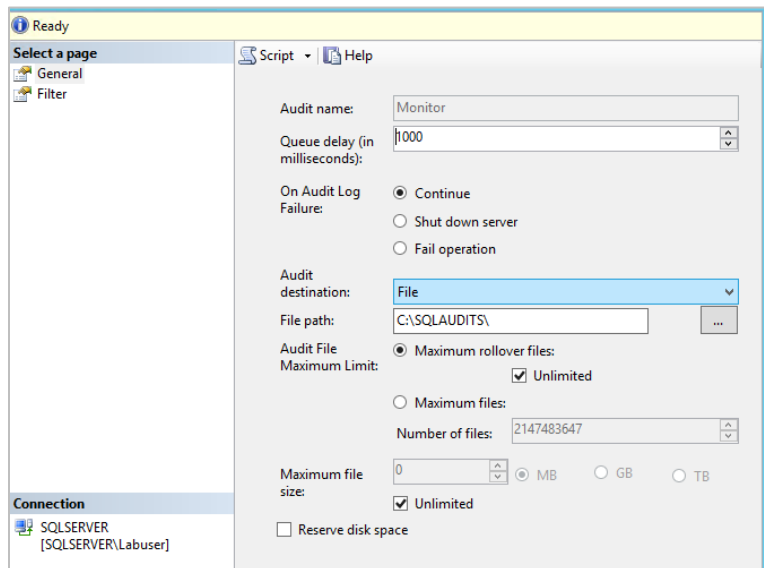
1. Expand security in **Object Explorer** and then **Audits**. You will see the **Monitor** object.



Note: You may need to click **Refresh**  to see the audit.

2. Right-click **Monitor**, and then select **Properties**.

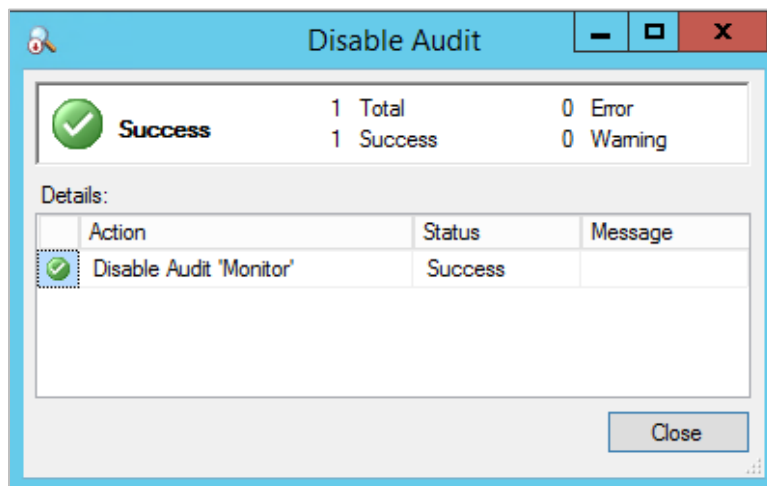
Note: The setting will continue server operation in case of an audit log failure. The audit destination (File) and file path are also defined here.



3. Click **OK**.

Now you will temporarily disable the audit. This event will be written to the audit log, but the server will remain in operation.

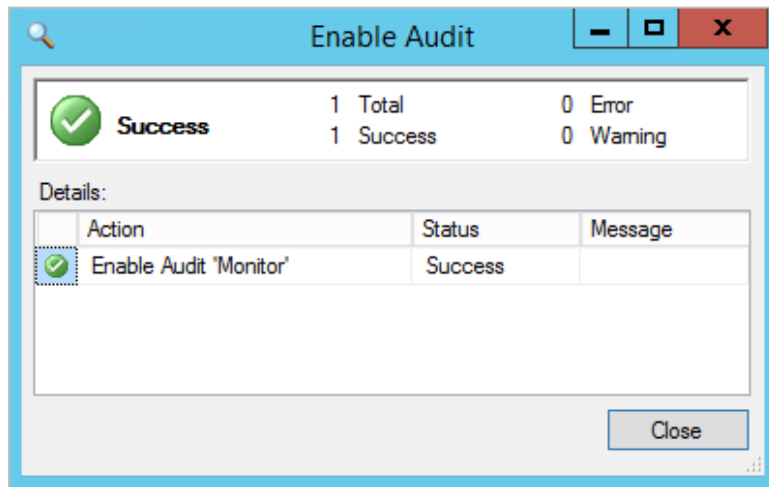
1. Right-click **Monitor**, and then select **Disable Audit**.



2. Click **Close**.

Note: Wait a few seconds before taking the next step.

- Right-click **Monitor**, and then select **Enable Audit**.



- Click **Close**.
- Right-click **Monitor**, and then select **View Audit Logs** to review the results.

Select logs

- ☒ Audit Collection
 - ☒ Monitor
 - ☐ Windows NT
- ☐ Windows NT

Log file summary: No filter applied

Date	Event Time	Server Instance Name	Action ID	Class Type	Sequence Number	Succeeded	Permission Bit Mask	Column Permission
6/17/2016 6:19:49 AM	06:19:49.4264517	SQL2016RTMHOL	AUDIT SESSION CHANGED	SERVER AUDIT	1	True	0x0000000000000000	False
6/17/2016 6:19:42 AM	06:19:42.6371775	SQL2016RTMHOL	AUDIT SESSION CHANGED	SERVER AUDIT	1	True	0x0000000000000000	False
6/17/2016 6:19:42 AM	06:19:42.6371775	SQL2016RTMHOL	AUDIT SESSION CHANGED	SERVER AUDIT	1	True	0x0000000000000000	False
6/17/2016 6:11:57 AM	06:11:57.8669618	SQL2016RTMHOL	AUDIT SESSION CHANGED	SERVER AUDIT	1	True	0x0000000000000000	False

- Note that the audit log recorded that auditing was disabled and then re-enabled.

Monitor

- ☐ Windows NT

Status

Last Refresh: 6/17/2016 6:20:08 AM

Filter: None

[View filter settings](#)

Progress

Done (4 records).

Date	Event Time	Server Instance Name	Action ID	Class Type	Sequence Num
6/17/2016 6:19:49 AM	06:19:49.4264517	SQL2016RTMHOL	AUDIT SESSION CHANGED	SERVER AUDIT	1
6/17/2016 6:19:42 AM	06:19:42.6371775	SQL2016RTMHOL	AUDIT SESSION CHANGED	SERVER AUDIT	1
6/17/2016 6:19:42 AM	06:19:42.6371775	SQL2016RTMHOL	AUDIT SESSION CHANGED	SERVER AUDIT	1
6/17/2016 6:11:57 AM	06:11:57.8669618	SQL2016RTMHOL	AUDIT SESSION CHANGED	SERVER AUDIT	1

Selected row details:

Statement

Additional Information

File Name C:\SQLAUDITSM\monitor_F37ACF2C-9F44-4255-9B22-10BEE40A237D_0_131106179894260000.sqlaudit

File Offset 5632

User Defined Event ID 0

- Click **Close**.

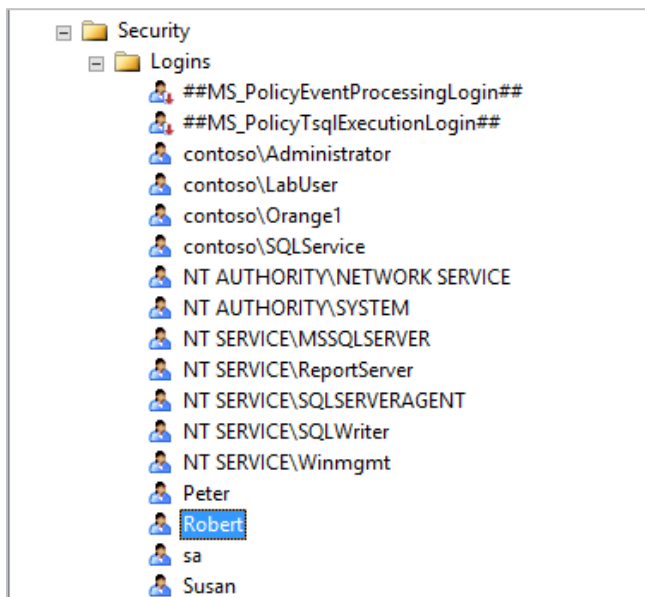
Grant Connect Any Database permission

You can grant the new server-level permission of **Connect Any Database** to a user who must connect to all currently existing databases and to any new databases created in the future. This does not, however, grant permission to any database beyond **Connect**.

Previously, any increase in the number of databases supported by the data administration team required either granting a server-level role or creating a logon for the administrator on each database. Now, with the **Connect Any Database** permission, this is no longer necessary because this permission already grants the user the ability to connect and view any database on the server.

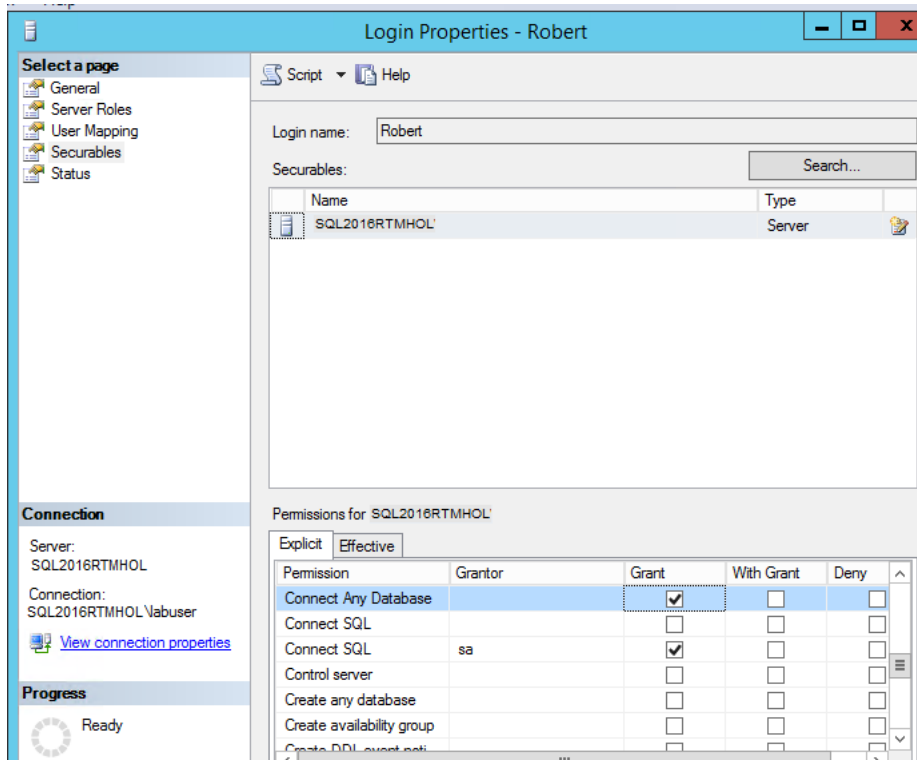
With a new data administrator ("Robert" for this scenario) soon to start, you can use **Connect Any Database** to give him access to the databases on the server. In this step, you will grant Robert the **Connect Any Database** permission.

1. Ensure that you have SQL Server 2016 Management Studio open with a connection to the **SQL2016RTMHOL** server. (If you do not, open Management Studio from the Windows Start screen, enter **Database Engine** as the server type, **SQL2016RTMHOL** as the server name, and ensure **Windows Authentication** is selected before clicking **Connect**.)
2. In Object Explorer, expand **Security**, and then expand **Logins**.



3. Right-click **Robert**, and then select **Properties**.

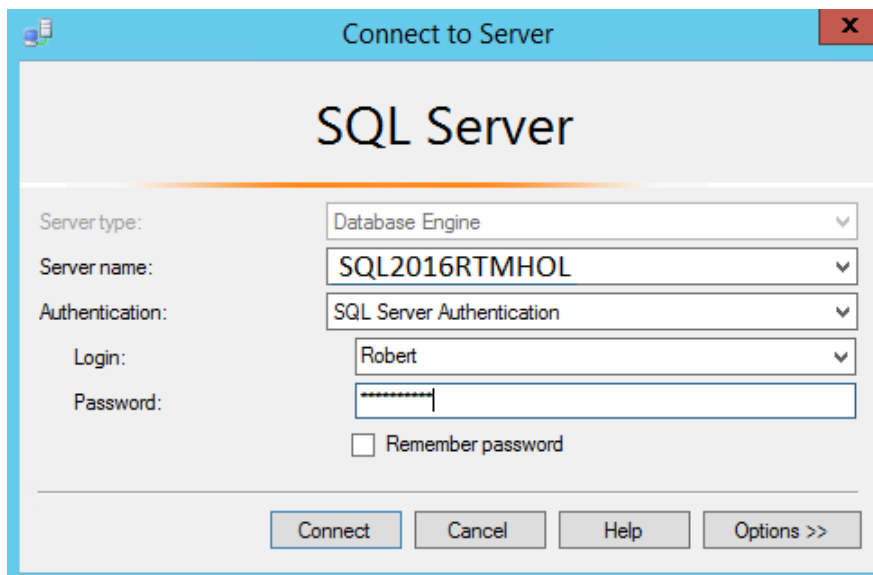
- Click **Securables**, and then select the check box to grant permission for **Connect Any Database**.



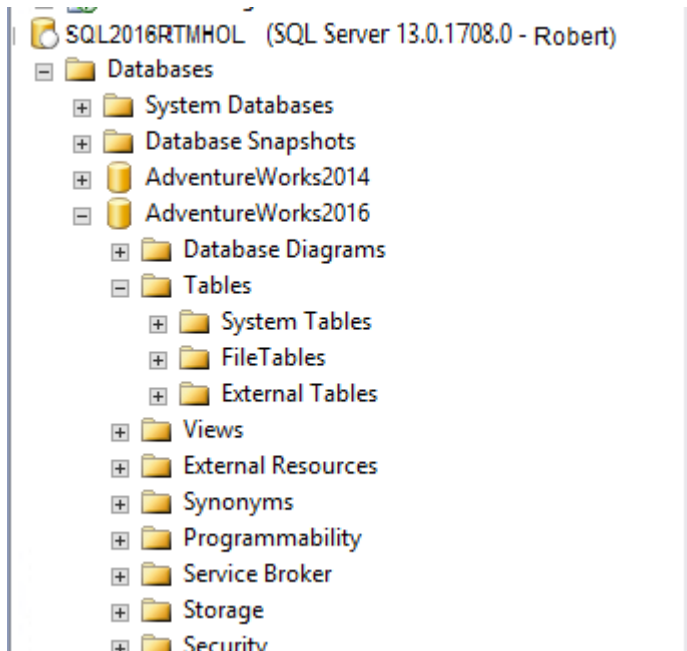
- Click **OK**.

Test that the Connect Any Database permission is working

- From SQL Server 2016 Management Studio in Object Explorer, click Connect to Server, select **SQL2016RTMHOL**, and then select **Database Engine**.
- Ensure that authentication is set to **SQL Server Authentication**.
- In the **Login** box, type **Robert**. In the **Password** box, type **pass@word1**.

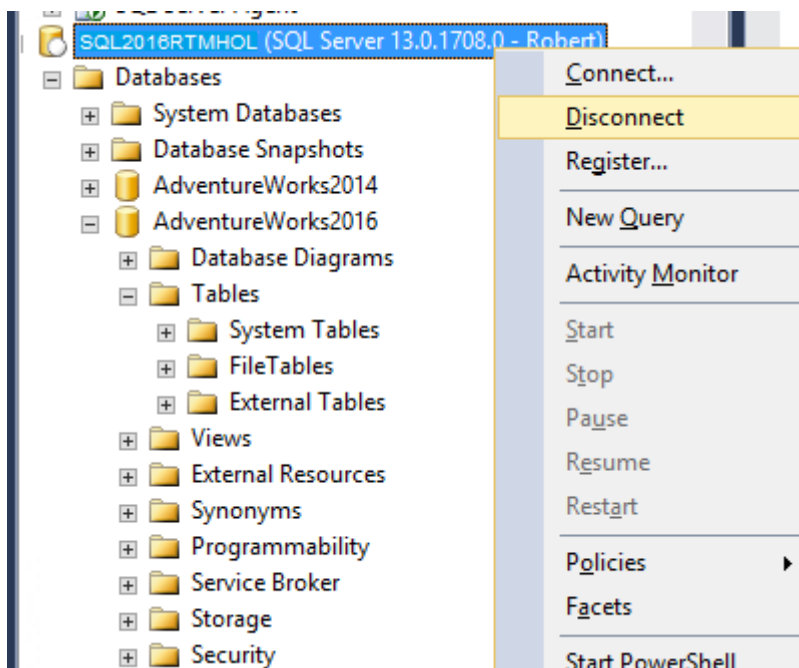


4. Click **Connect**.
5. Expand **Databases**.
6. Expand **AdventureWorks 2016**.
7. Expand **Tables**.



Although Robert is not a system administrator, he is able to see all of the databases with the **View Any Database** permission that is granted to the **Public Server Role**. Consequently, all users by default can see all databases.

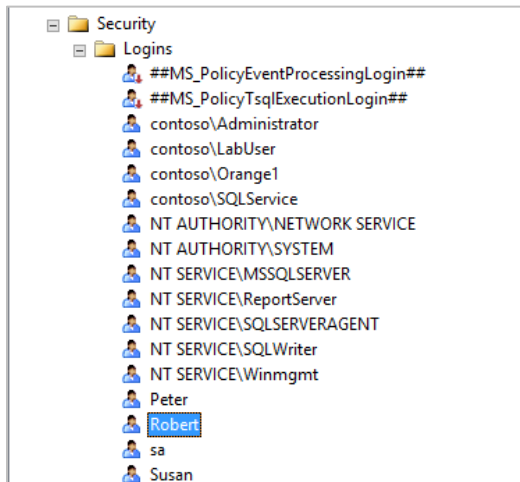
8. Right-click **SQL2016RTMHOL** for Robert in **Object Explorer**, and then click the **Disconnect** command.



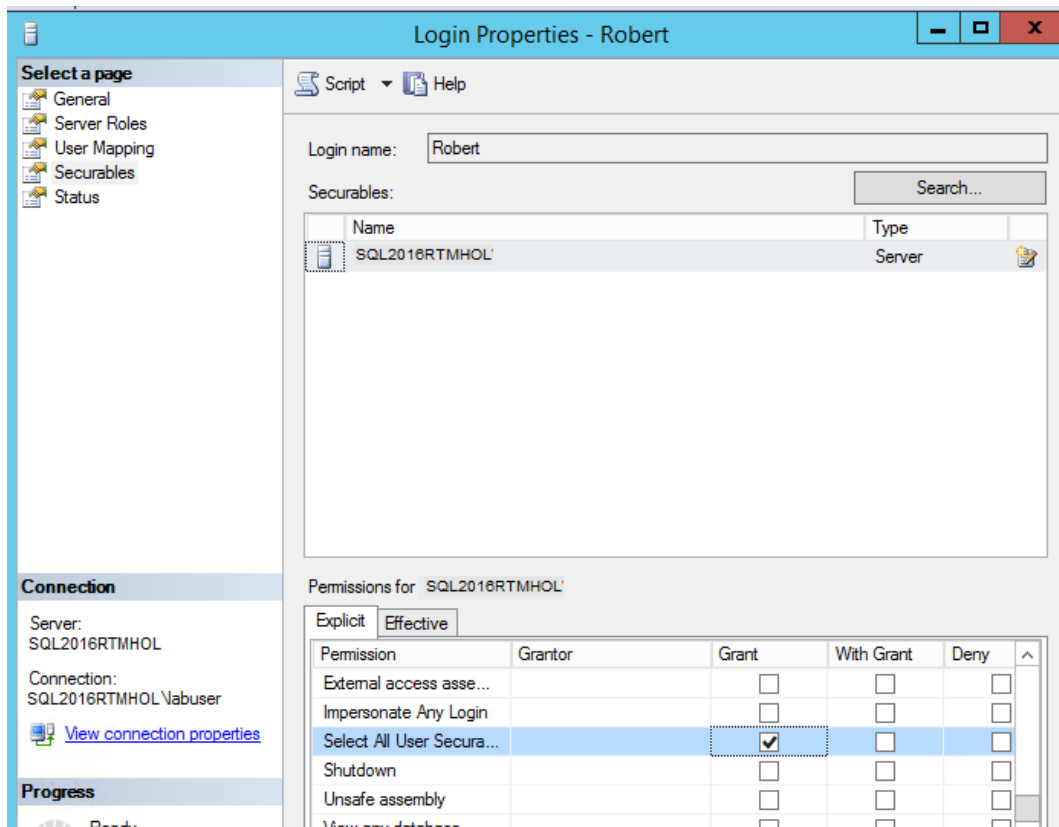
Select All User Securables

The Select All User Securables permission allows users to select from any table or view any database they can connect to. When combined with **Connect Any Database**, it allows selection of any data in any database. Now you will use the **Select All User Securables** permission to allow Robert to view data.

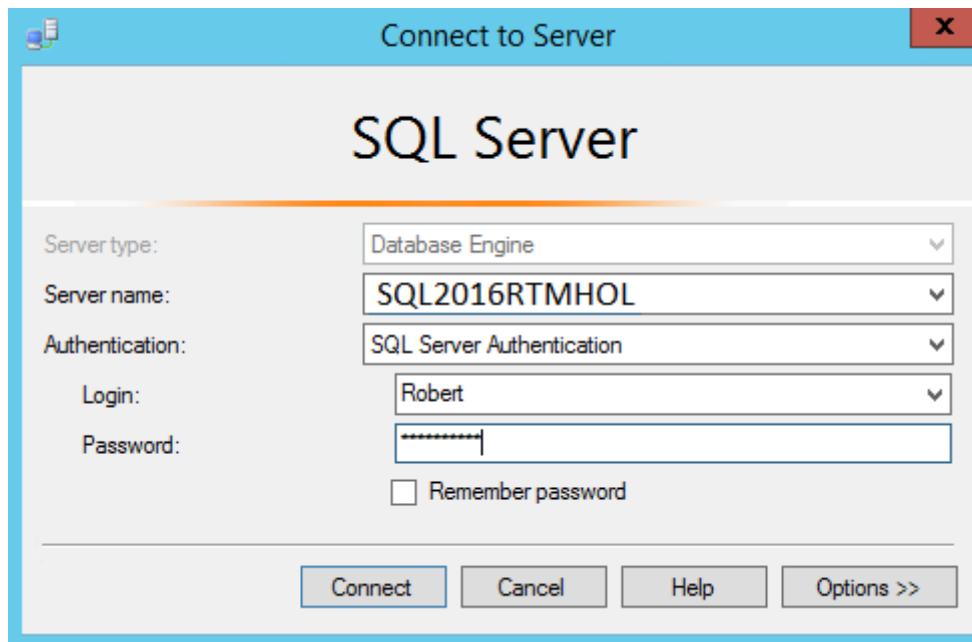
1. Ensure that you have SQL Server 2016 Management Studio open with a connection to the **SQL2016RTMHOL** server. (If you do not, open Management Studio from the Windows Start screen, enter **Database Engine** as the server type, **SQL2016RTMHOL** as the server name, and ensure **Windows Authentication** is selected before clicking **Connect**.)
2. In **Object Explorer**, expand **Security**, and then expand **Logins**.



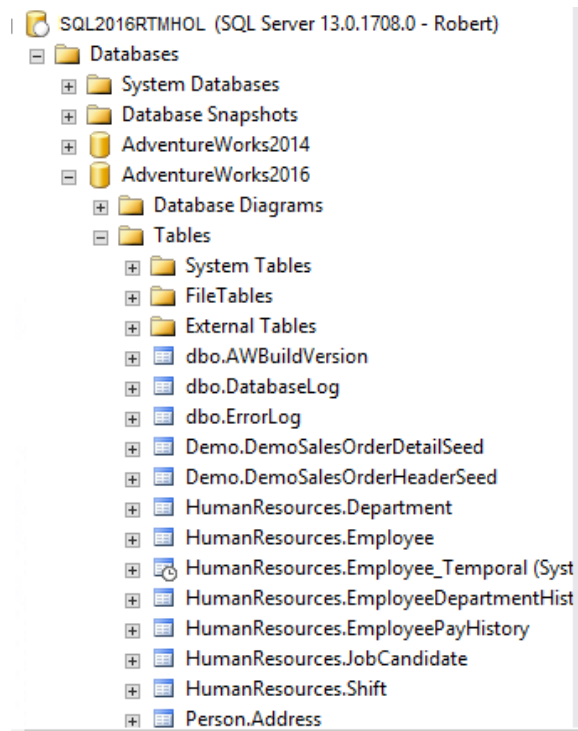
3. Right-click **Robert**, log on under **Security** → **Logins** in Object Explorer, and then click **Properties**.
4. Click **Securables**, and then select the check box to grant permission for **Select All User Securables**.



5. Click **OK**.
6. From SQL Server 2016 Management Studio in Object Explorer, click Connect to **SQL2016RTMHOL**, and then select **Database Engine**. Ensure that authentication is set to **SQL Server Authentication**. In the **Login** box, type **Robert**. In the **Password** box, type **pass@word1**.



7. Click **Connect**.
8. Expand **Databases**.
9. Expand **AdventureWorks 2016**.
10. Expand **Tables**.



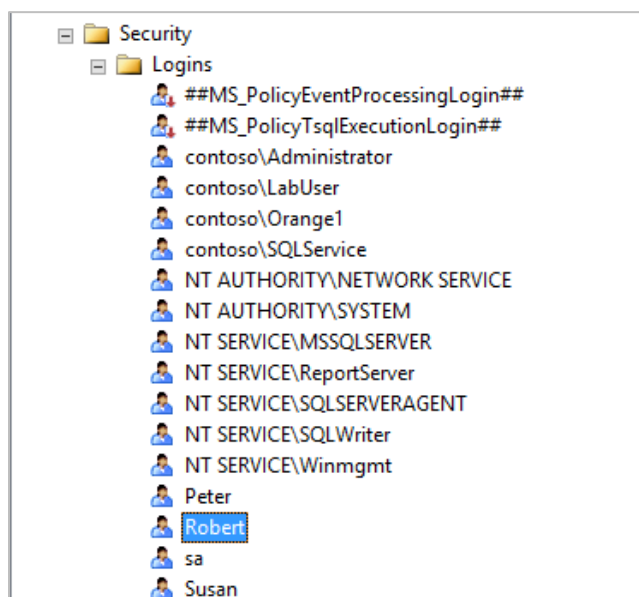
Robert now has the permissions for Connect Any Database and Select All User Securables, so he can see all data in the databases.

11. Now, in Object Explorer, right-click **SQL2016RTMHOL** for **Robert**, and then click the **Disconnect** command.

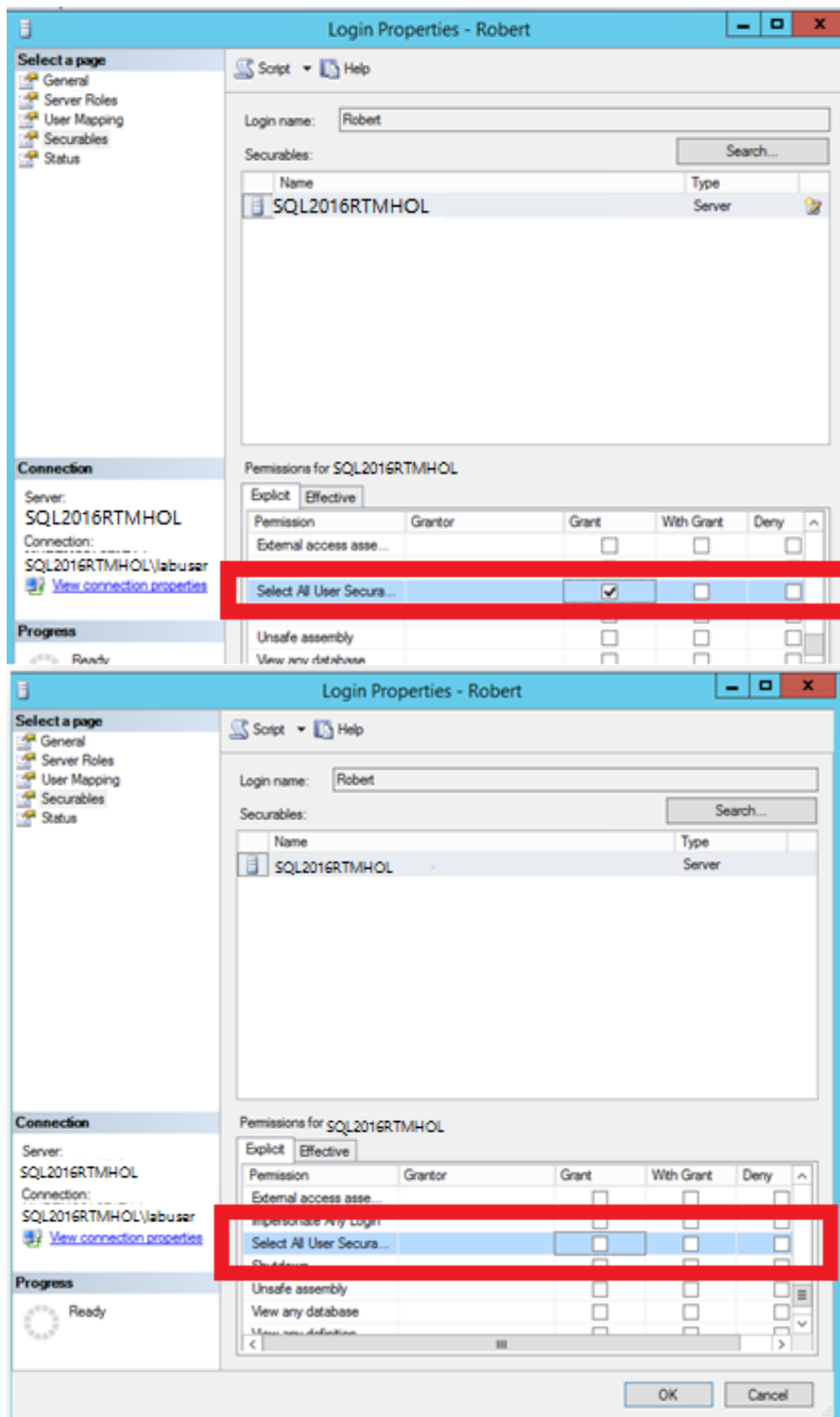
Revoking Select All User Securable permission

In the next exercise, you will be granting the View Definition permission, but first you will need to revoke the Select All User Securable permission settings for Robert. To do so, follow these steps:

1. Connect to the **SQL2016RTMHOL** server. (If you do not, open Management Studio from the Windows Start screen, enter **Database Engine** as the server type, **SQL2016RTMHOL** as the server name, and then ensure **Windows Authentication** is selected before clicking **Connect**.)
2. In Object Explorer, expand **Security**, and then expand **Logins**.



3. Right-click **Robert**, and then select **Properties**.
4. Click **Securables**, and then clear the grant permission check box for **Select All User Securables**.



This will revoke the Select All User Securable permission for Robert.

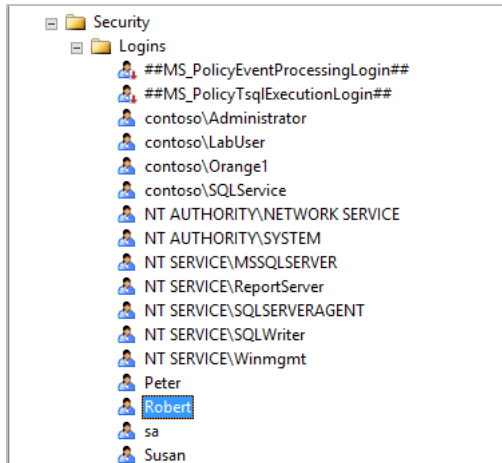
Now you can proceed with the View Definition permission.

Granting View Definition permission

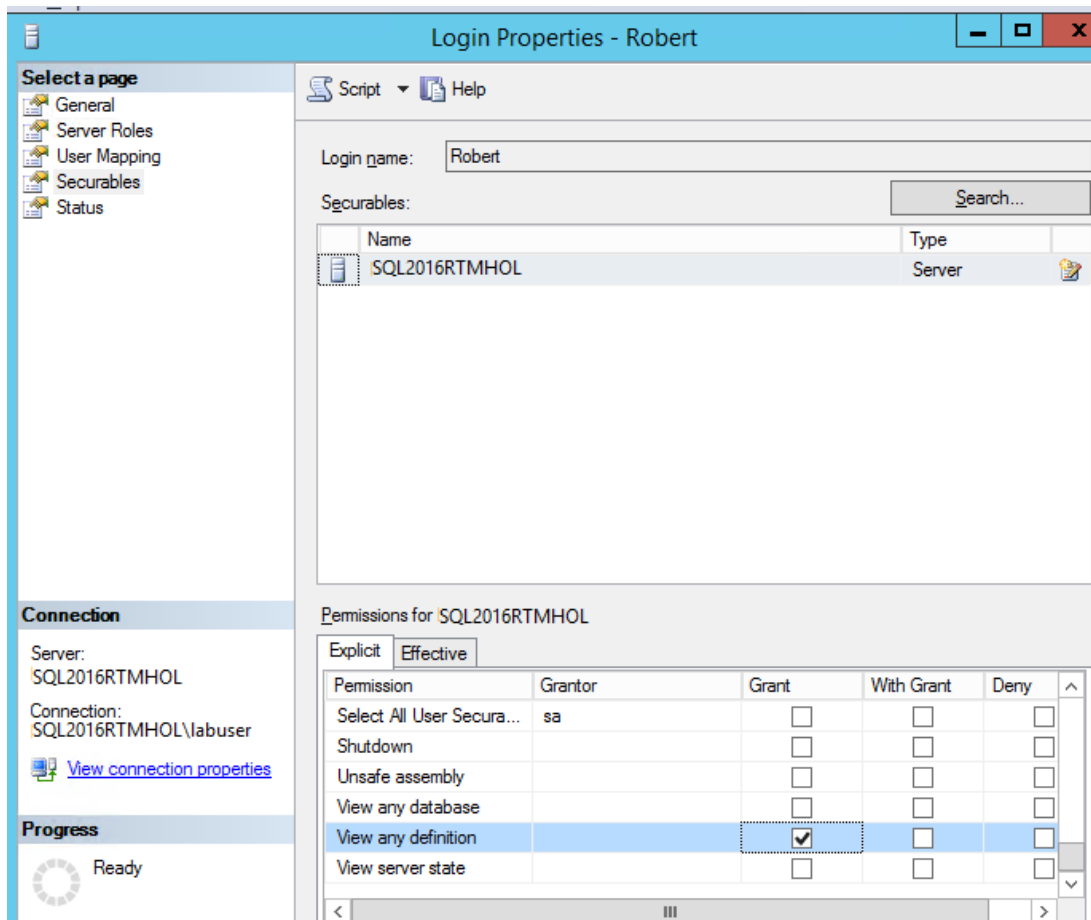
The View permission lets a user see the metadata of the securable on which the permission is granted. However, View Definition permission does not confer access to the securable itself. View Any Definition permission granted at this scope effectively negates permissions-based metadata access for the grantee. This means that the grantee can see all metadata in the instance of SQL Server unless the grantee is denied View Definition or Control permissions at the database scope, schema scope, or for an individual entity such as a table.

Note: For information about the syntax to use for this permission at this scope, see *GRANT (Transact-SQL)*.

1. Ensure that you have SQL Server 2016 Management Studio open with a connection to the **SQL2016RTMHOL** server. (If you do not, open Management Studio from the Windows Start screen, enter **Database Engine** as the server type, **SQL2016RTMHOL** as the server name, and then ensure **Windows Authentication** is selected before clicking **Connect**.)
2. In Object Explorer, expand **Security**, and then expand **Logins**.

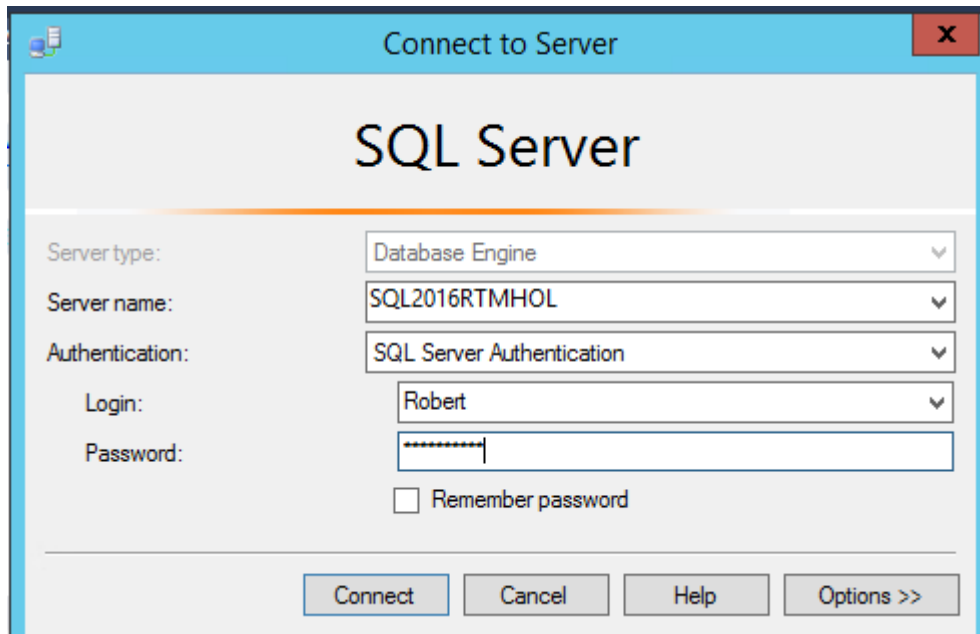


3. Right-click **Robert**, and then select **Properties**.
4. Click **Securables**, and then select the check box to grant permission for **View any definition**.

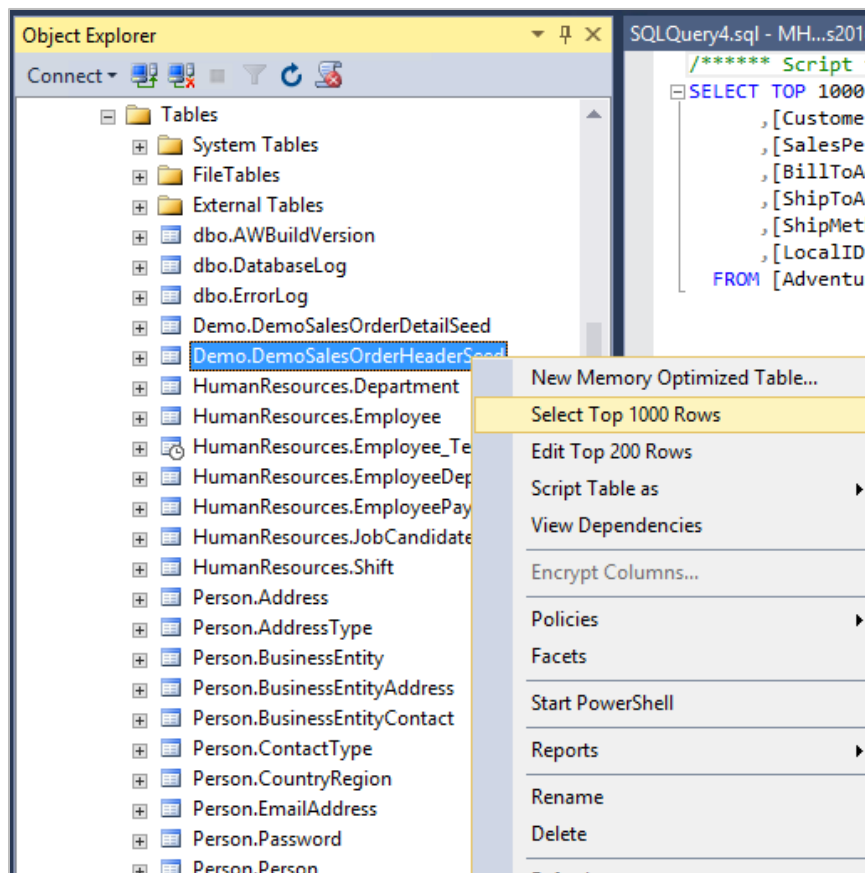


5. Click **OK**.

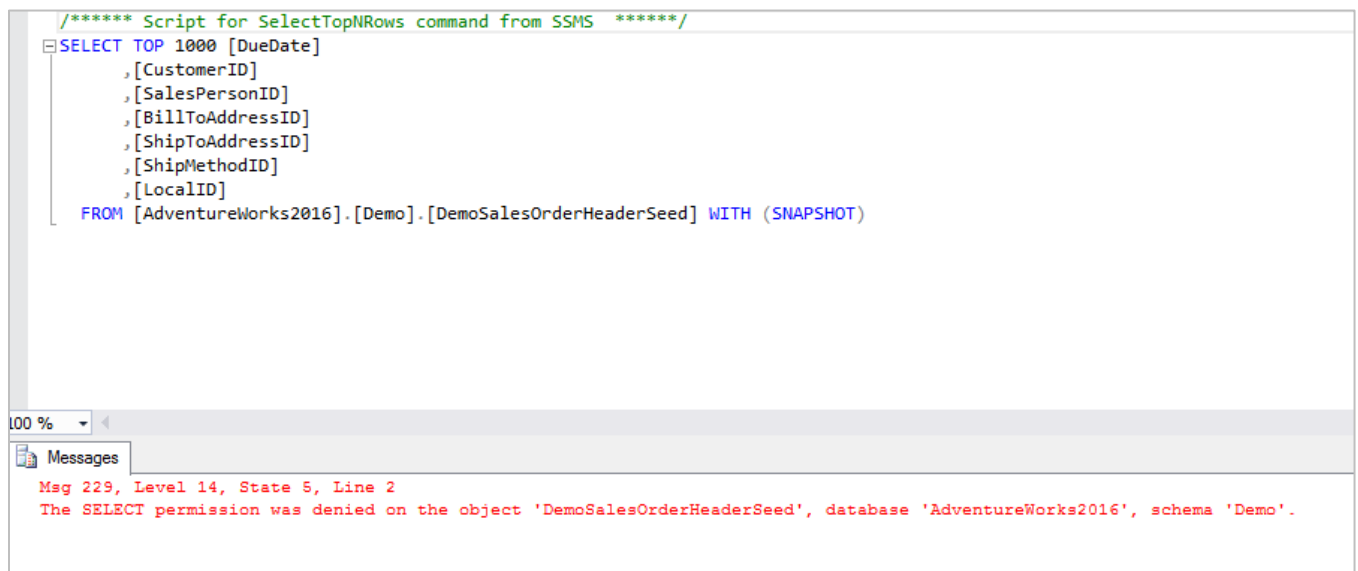
6. From SQL Server 2016 Management Studio in Object Explorer, click Connect to **SQL2016RTMHOL**, and then select **Database Engine**. Ensure that authentication is set to **SQL Server Authentication**. In the **Login** box, type **Robert**. In the **Password** box, type **pass@word1**.



7. Click **Connect**.
8. Expand **Databases**.
9. Expand **AdventureWorks2016**.
10. Expand **Tables**.
11. Right-click and select **Select Top 10000 Rows**.



You can see that Robert has assigned the permission to see all metadata in the instance of SQL Server. However, without additional permissions such as Select or Control, the user cannot read data from the table.



Create server role for monitoring

User-defined server roles increase flexibility and manageability and help facilitate compliance through clearer separation of duties. They can be created to suit different organizations that separate administrative duties

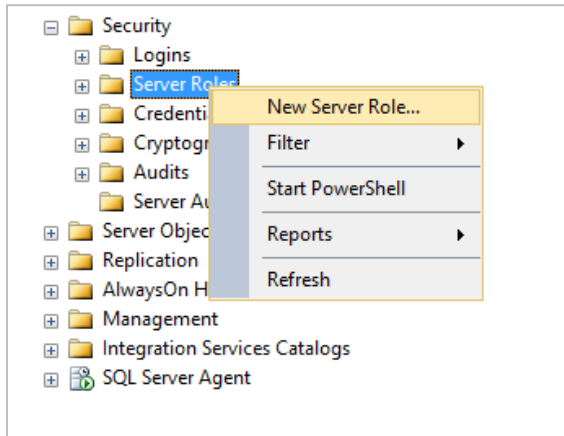
according to roles. Server roles can be nested to allow more flexibility in mapping to hierarchical structures in organizations.

User-defined server roles also help prevent organizations from using sysadmin for database administration. For example, a special database administration role—without the ability to access user data—can be created for common database administration.

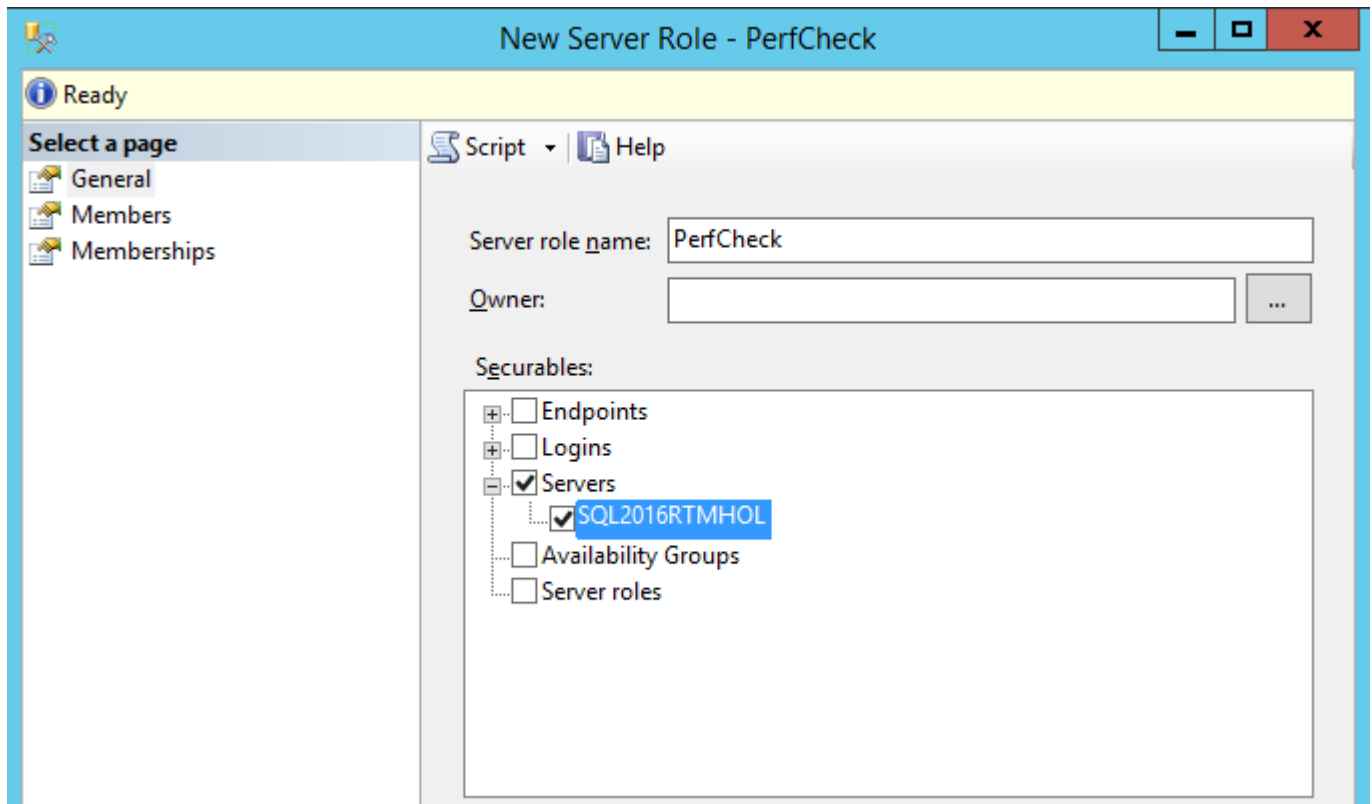
In the following steps, you will create a server role called **PerfCheck** for user Peter, who needs to monitor SQL Server performance. This role is allowed to see all of the databases but not the data in them.

Create the new role PerfCheck

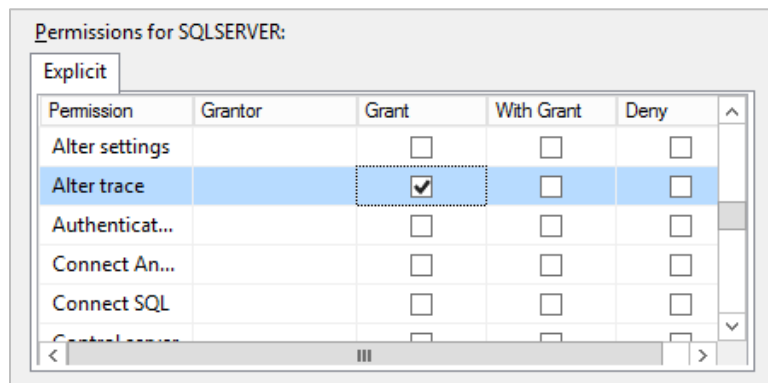
1. Ensure that you have SQL Server Management Studio open with a connection to the **SQL2016RTMHOL** server. (If you do not, open Management Studio from the Windows Start screen, enter **Database Engine** as the server type, enter **SERVER** as the server name, and select **Windows Authentication** before clicking **Connect**.)
2. Expand **Security** in Object Explorer.
3. Right-click **Server Roles**, and then select **New Server Role**.



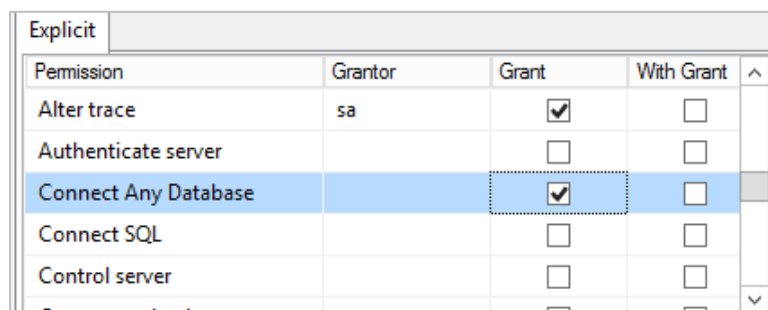
4. In **Server Role Name**, enter **PerfCheck**.
5. Select and expand the **Servers** check box, and then select **SQL2016RTMHOL**.



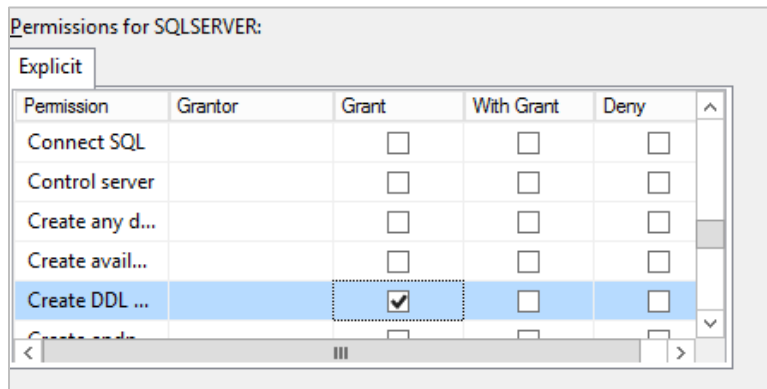
6. In the following steps, you will be granting a variety of permissions. First, under **Grant**, select the **Alter trace** permission check box.



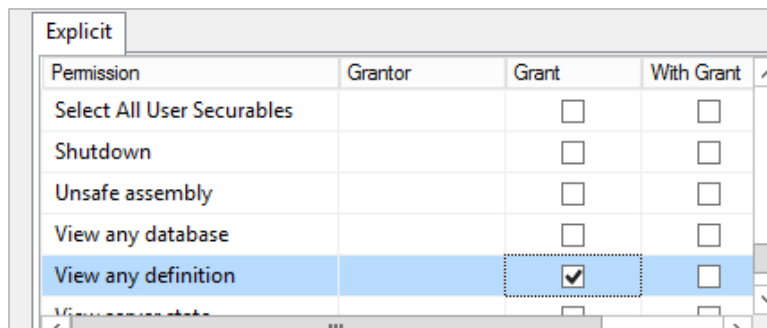
7. Next, grant permission to **Connect Any Database**.



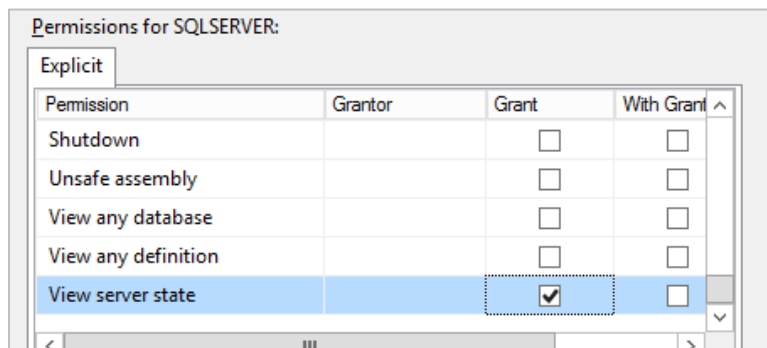
8. Select the **Grant** check box for **Create DDL** event notification.



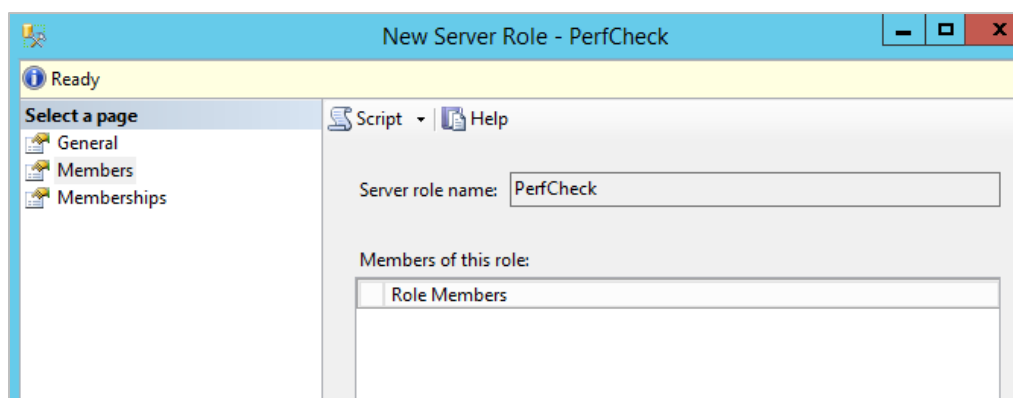
9. Grant permission for **View any definition**.



10. Select the **Grant** check box for **View server state**.

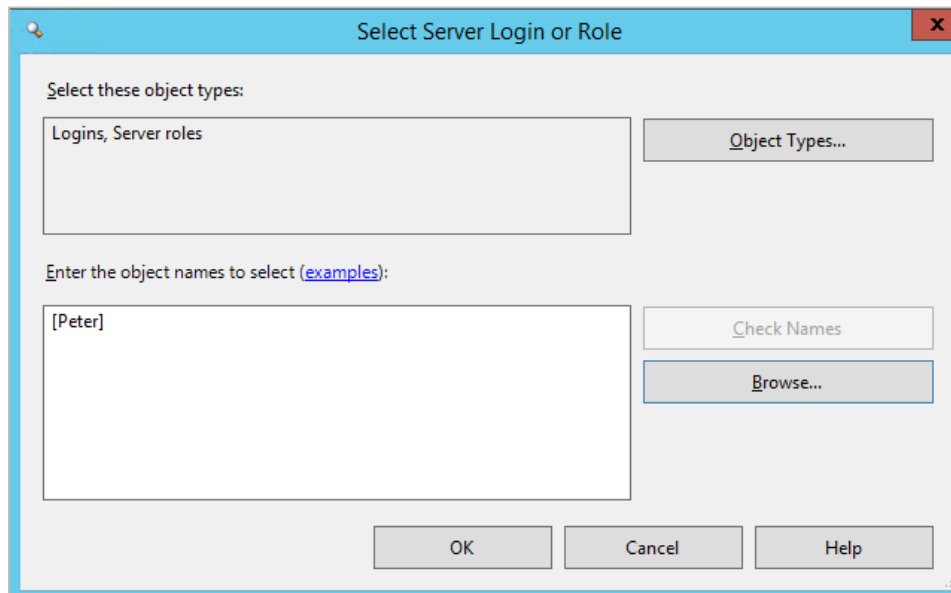


11. Click **Members**.



12. Click **Add**.

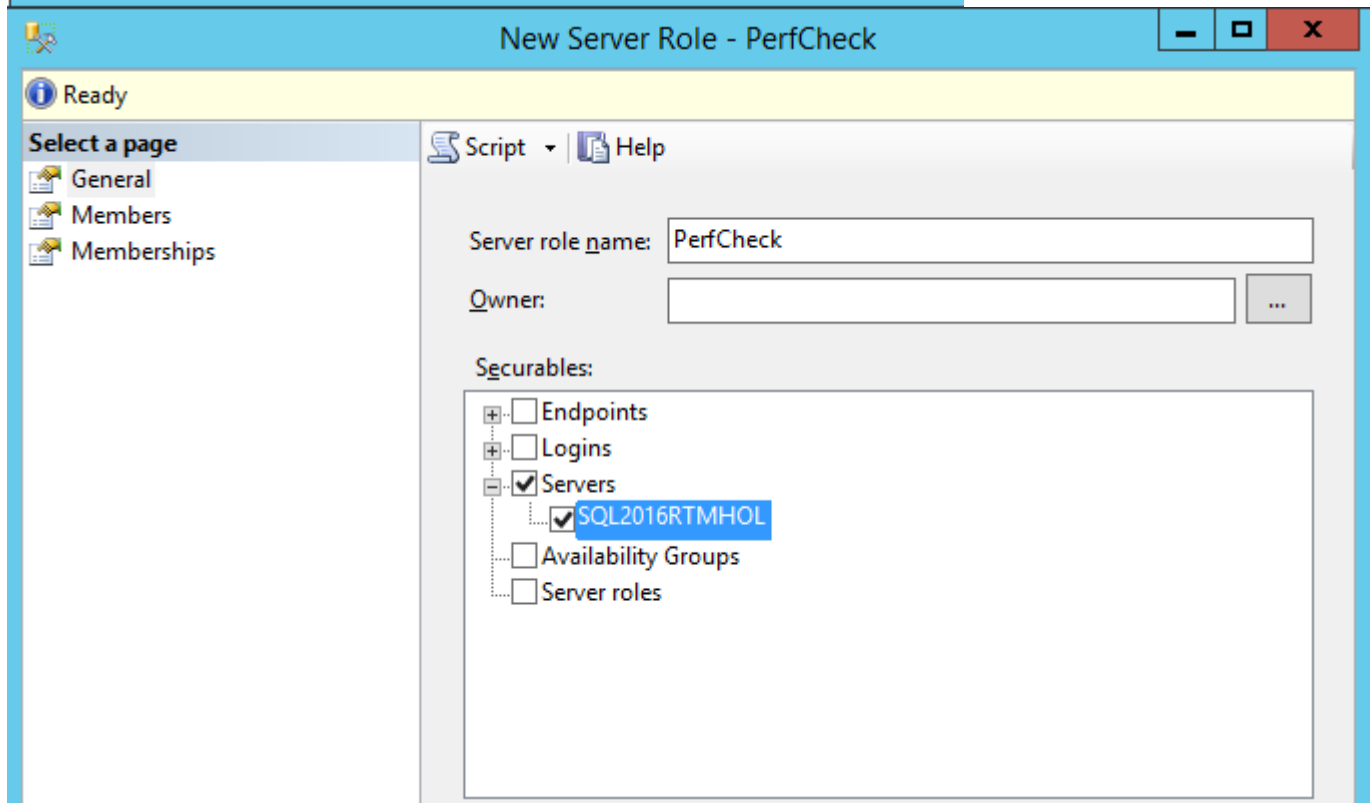
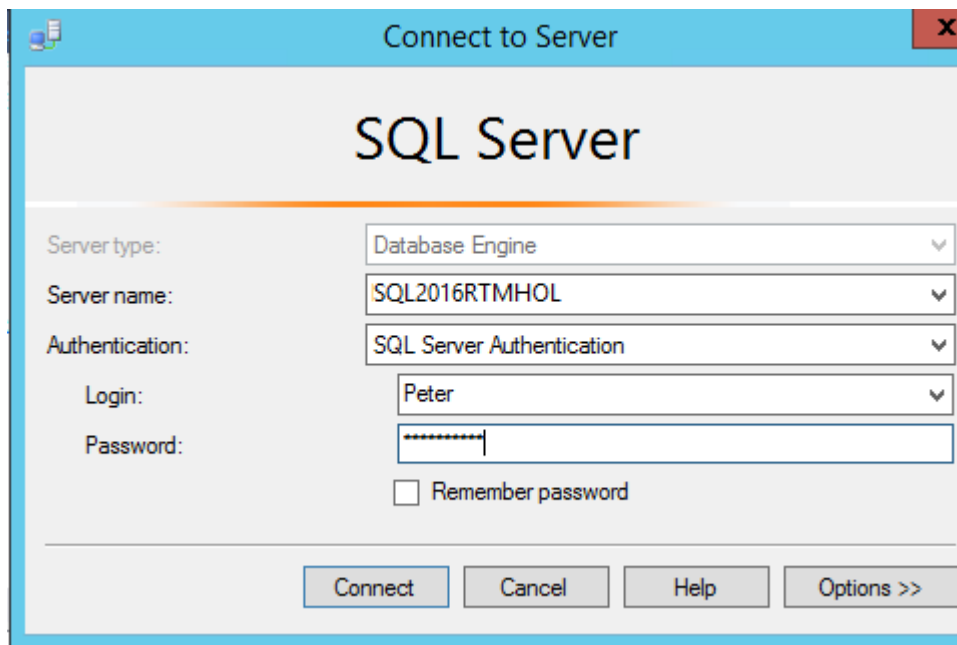
13. In the **Enter the object names to select** field, type **Peter**, and then click **OK**.



Confirm the use of the new role

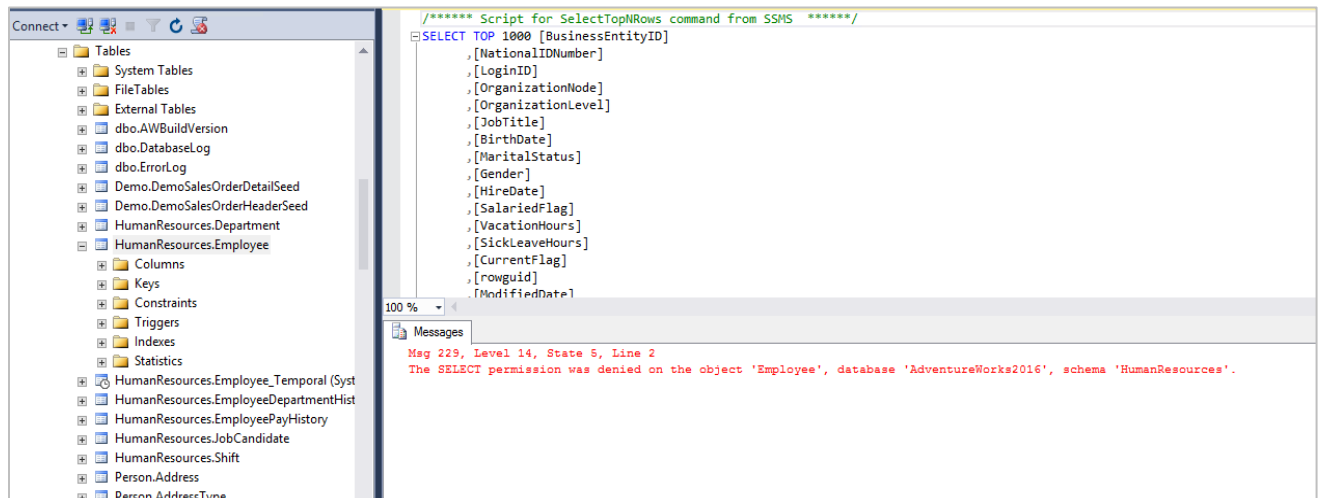
Now that you have created the new role for Peter, you are going to confirm that it is working.

1. At the top of Object Explorer, click **Connect**, and then select **Database Engine**.
2. Ensure that Authentication is set to **SQL Server Authentication** and that Server name is **SQL2016RTMHOL**.
3. In the **Login** box, type **Peter**. In the **Password** box, type **pass@word1**.



4. Click **Connect**.
5. Expand **Databases**, and then expand the database **AdventureWorks 2016**.
6. Expand any table, and then right-click and select **–SELECT TOP 1000 Rows**.

Notice the error message in the query windows.

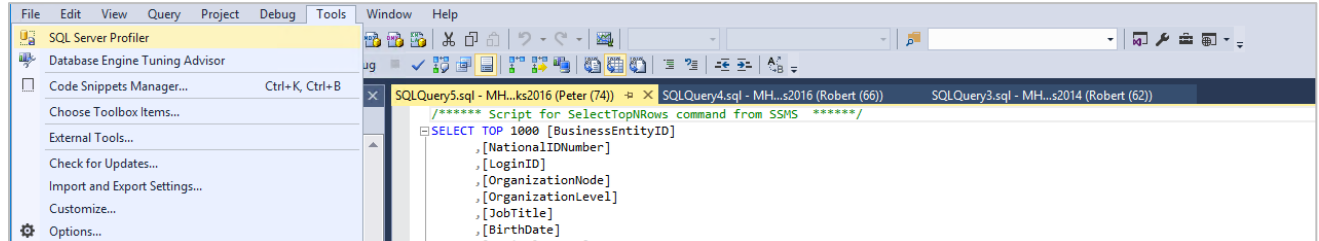


Note: The role **PerfCheck** is allowed to see all the databases and view any definition, but is not allowed access to the data in them. However, the role does allow appropriate operations to monitor SQL Server performance without the user requiring sysadmin rights.

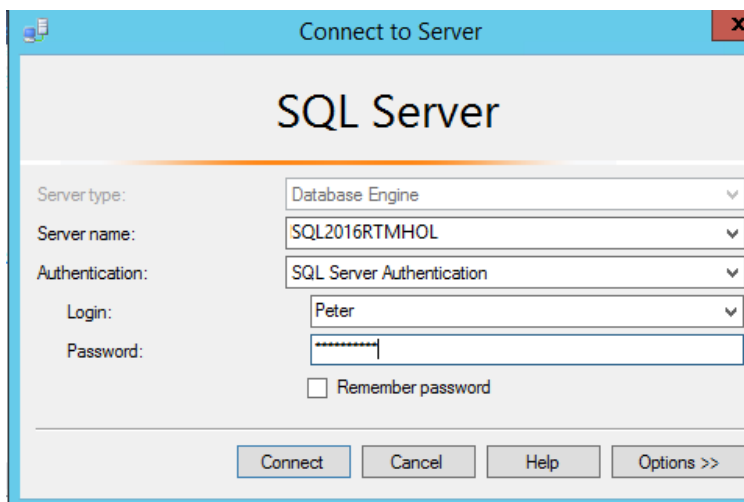
7. Click **OK**.

Test for ability to perform a SQL Trace

1. To confirm that user Peter can perform a SQL Trace, click **Tools**, and then select **SQL Server Profiler** (as indicated below).



2. Ensure that authentication is set to **SQL Server Authentication**.
3. In the **Login** box, type **Peter**. In the **Password** box, type **pass@word1**.



Click **Connect**.

- Click **Run** to start a trace.

Trace Properties

General | Events Selection

Trace name:

Trace provider name:

Trace provider type: version:

Use the template:

☐ Save to file:

Set maximum file size (MB):

☒ Enable file rollover

☐ Server processes trace data

☐ Save to table:

☐ Set maximum rows (in thousands):

☐ Enable trace stop time:


Run Cancel Help

Notice how traces are captured from all server databases, even though the role **PerfCheck** does not have sysadmin rights and access to user data.

EventClass	TextData	ApplicationName	NTUserName	LoginName	CPU	Reads	Writes	Duration	ClientProcessID	SPID
ExistingConnection	-- network protocol: LPC set quote...	Microsoft SQ...		Peter					17320	65
ExistingConnection	-- network protocol: LPC set quote...	Microsoft SQ...		Robert					17320	66
ExistingConnection	-- network protocol: LPC set quote...	SQLAgent - E...	SQLSERV...	NT SER...					7492	71
ExistingConnection	-- network protocol: LPC set quote...	Microsoft SQ...		Peter					17320	74
ExistingConnection	-- network protocol: LPC set quote...	SQLAgent - G...	SQLSERV...	NT SER...					7492	76
ExistingConnection	-- network protocol: LPC set quote...	SQLAgent - J...	SQLSERV...	NT SER...					7492	79
Audit Logout		DWDiagnostics	MHRTMSQ...	WORKGR...	0	0	0	60064	11336	58
RPC:Completed	exec sp_reset_connection	DWDiagnostics	MHRTMSQ...	WORKGR...	0	0	0	0	11336	58
Audit Login	-- network protocol: LPC set quote...	DWDiagnostics	MHRTMSQ...	WORKGR...					11336	58
SQL:BatchStarting	SELECT @@SPID	DWDiagnostics	MHRTMSQ...	WORKGR...					11336	58
SQL:BatchCompleted		DWDiagnostics	MHRTMSQ...	WORKGR...	0	0	0	0	11336	58
SQL:BatchStarting	SELECT [session_name], [definition]...	DWDiagnostics	MHRTMSQ...	WORKGR...					11336	58
SQL:BatchCompleted	SELECT [session_name], [definition]...	DWDiagnostics	MHRTMSQ...	WORKGR...	0	0	0	0	11336	58

Trace is running.

Ln 25, Col 1 Rows: 25

- Stop the trace by clicking .
- Close the **SQL Server Profiler**.
- Close all windows.

SQL Server 2016 audit

In this exercise, you will use the following auditing features in SQL Server 2016:

- **Audit resilience.** Ensures that audit logs are not lost in failover during temporary file and network issues.
- **User-defined audit.** Creates events that allow applications to write custom information to the audit log.
- **Audit filtering.** Improves filtering to simplify audit reporting.

Audit resilience

The resilience features implemented in SQL Server 2016 auditing mean that the audit logging is now tolerant to loss of connectivity to the target directory and will recover automatically once the network connection is re-established.

You will implement a policy on a new server to ensure that processing should continue in the event of an audit log failure. The audit process is also able to record that auditing has been paused or stopped.

In the following exercise, you can check that the audit log correctly records when auditing is disabled and then re-enabled.

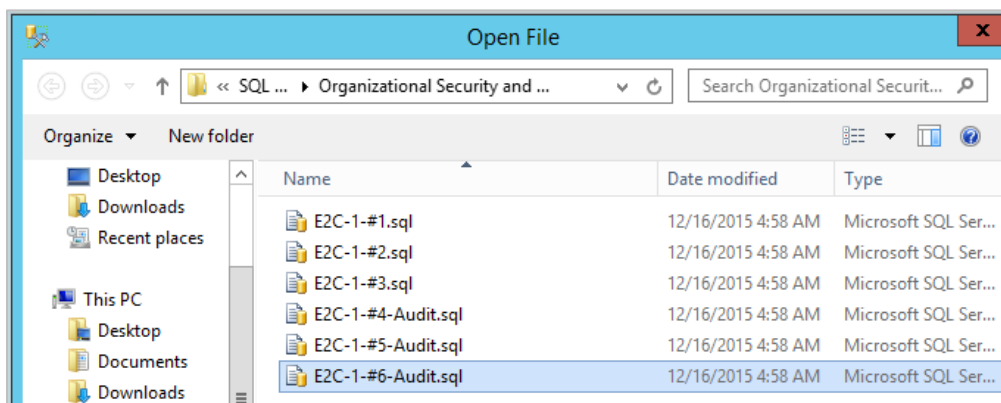
User-defined audit

The new user-defined audit events allow applications to write custom information to the audit log.

In this scenario, as part of an auditing policy, you need to log employees whose salary increases by more than 20 percent. To help with this task, you will create a user-defined audit event that will be triggered whenever an employee's salary is increased by more than 20 percent.

Begin by creating an audit called **LargePayIncrease**:

1. From SQL Server 2016 Management Studio, open the **File** menu. Select **Open**, and then select **File**.
2. Browse to **C:\SQL Server 2016 HOLs\Organizational Security and Auditing\Scripts**, and then select **E2C-1-#6-Audit**.



- Click **Open**, and then select **Execute** to run the script.

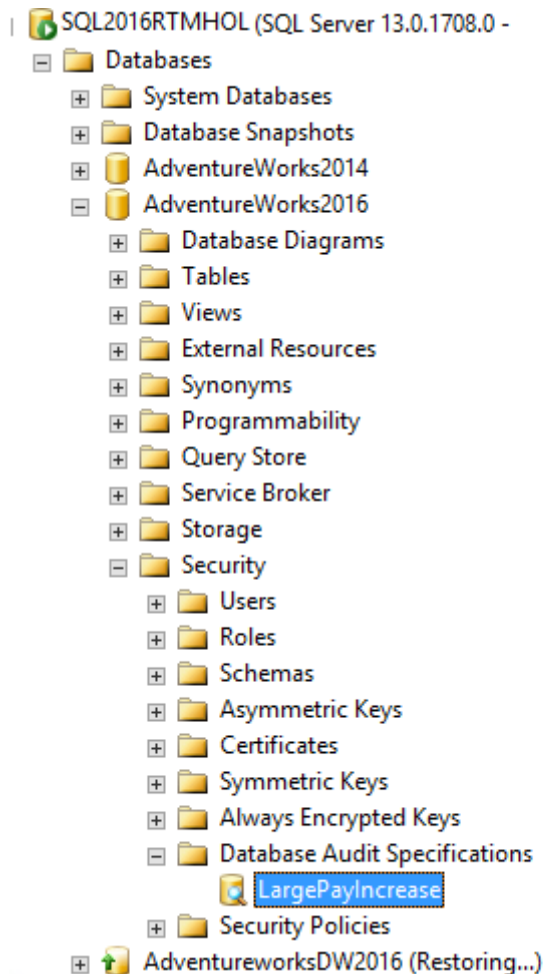
```
USE [master]
GO
CREATE SERVER AUDIT [TestingUserDefinedEvents]
TO FILE
( FILEPATH = N'C:\SQLAUDITS'
,MAXSIZE = 5 MB
,MAX_ROLLOVER_FILES = 5
,RESERVE_DISK_SPACE = OFF
)
WITH
( QUEUE_DELAY = 1000
,ON_FAILURE = CONTINUE
)
GO
ALTER SERVER AUDIT [TestingUserDefinedEvents] WITH (STATE = ON);
GO
USE [AdventureWorks2016]
```

100 %

Messages

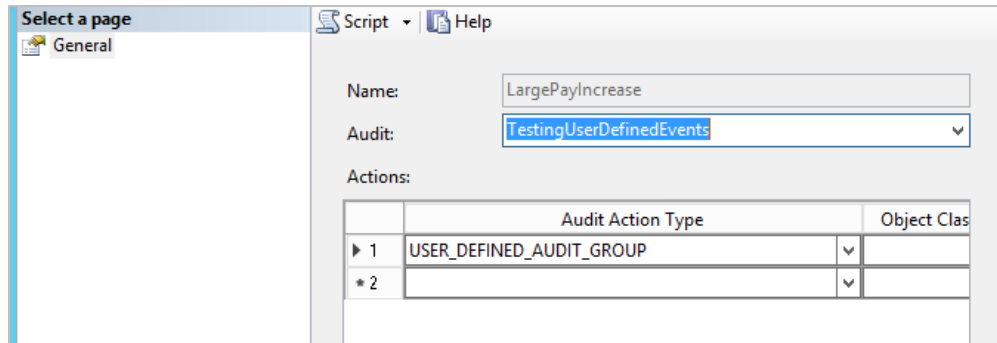
Command(s) completed successfully.

- Expand databases and, from Object Explorer, select **AdventureWorks2016** → **Security** → **Database Audit Specifications**.
- You will see that the audit report called **LargePayIncrease** was created.



- Right-click **LargePayIncrease**, and then select **Properties**.

Notice that the Audit Action Type is set to USER_DEFINED_AUDIT_GROUP. (This group tracks events raised by using the **sp_audit_write** stored procedure.)

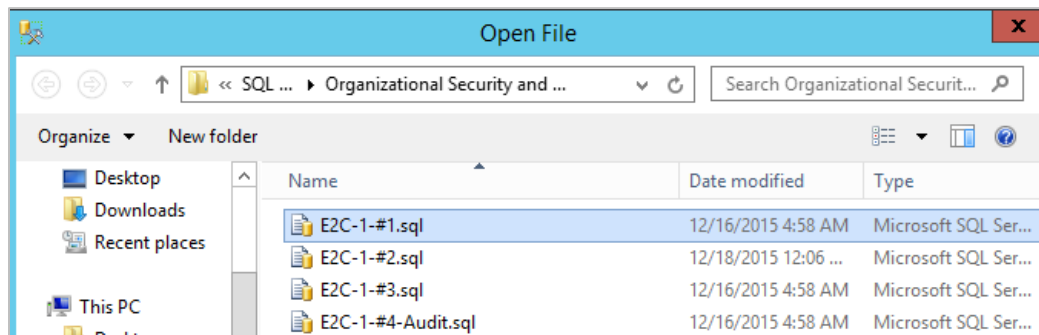


7. Click **OK**.

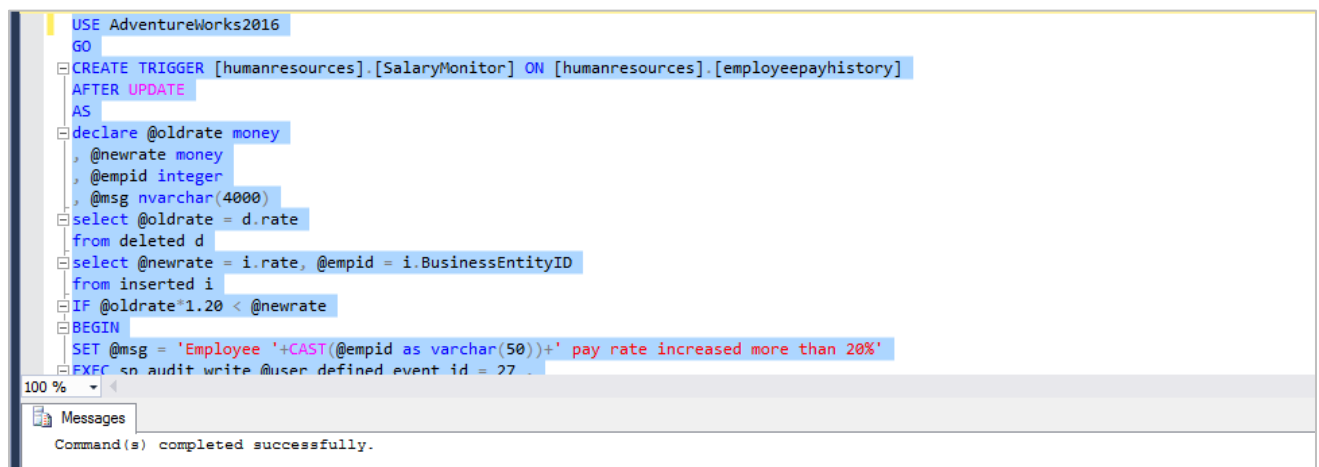
Use a trigger to write an audit record using **sp_audit_write**

Now you will create a trigger for when an employee's salary is increased by 20 percent. This trigger uses the **sp_audit_write** stored procedure, which means an event will be logged when it is used and recorded in the **LargePayIncrease** audit.

1. From SQL Server 2016 Management Studio, select **File**. Next, select **Open**, and then click **File**.
2. Browse to **C:\SQL Server 2016 HOLs\Organizational Security and Auditing\Scripts** and select **E2C-1-#1**.



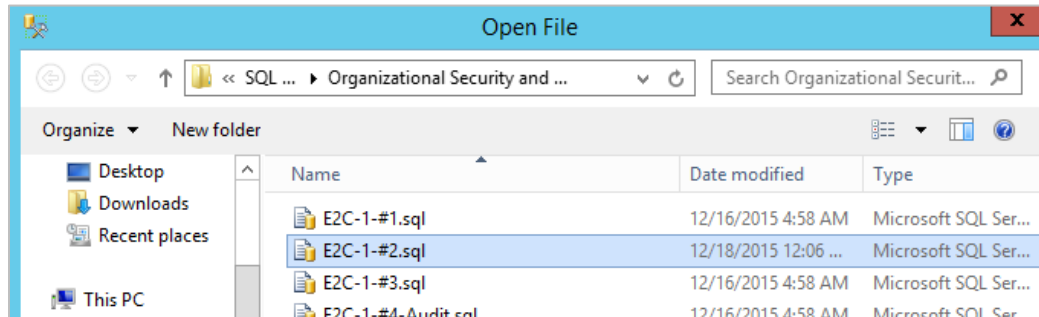
3. Click **Open**.



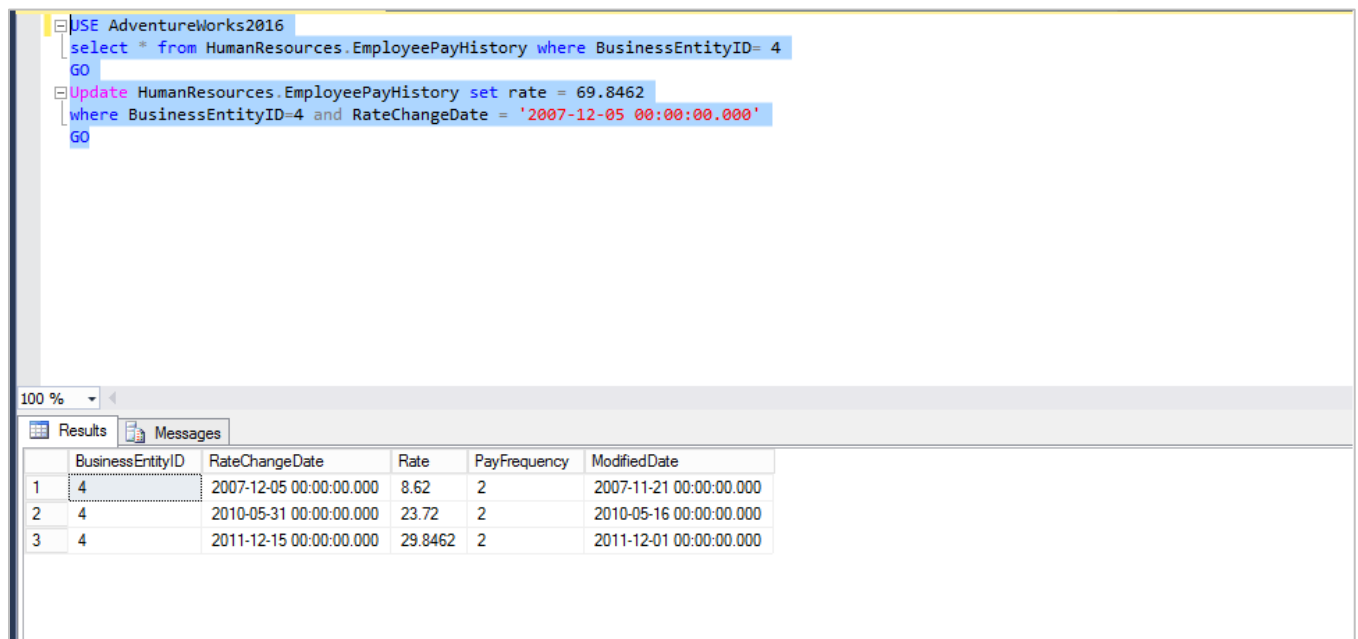
4. Click **Execute**.

Now that you have created a trigger, you can test that the user-defined audit event is working by running the following script, which virtually doubles the pay rate of Employee 4 (Business Entity ID=4).

1. Begin by selecting another script file. Click **Open**, and then click **File**.
2. Browse to **C:\SQL Server 2016 HOLs\Organizational Security and Auditing\Scripts** and select **E2C-1-#2**.



3. Click **Open**.
4. Click **Execute**.

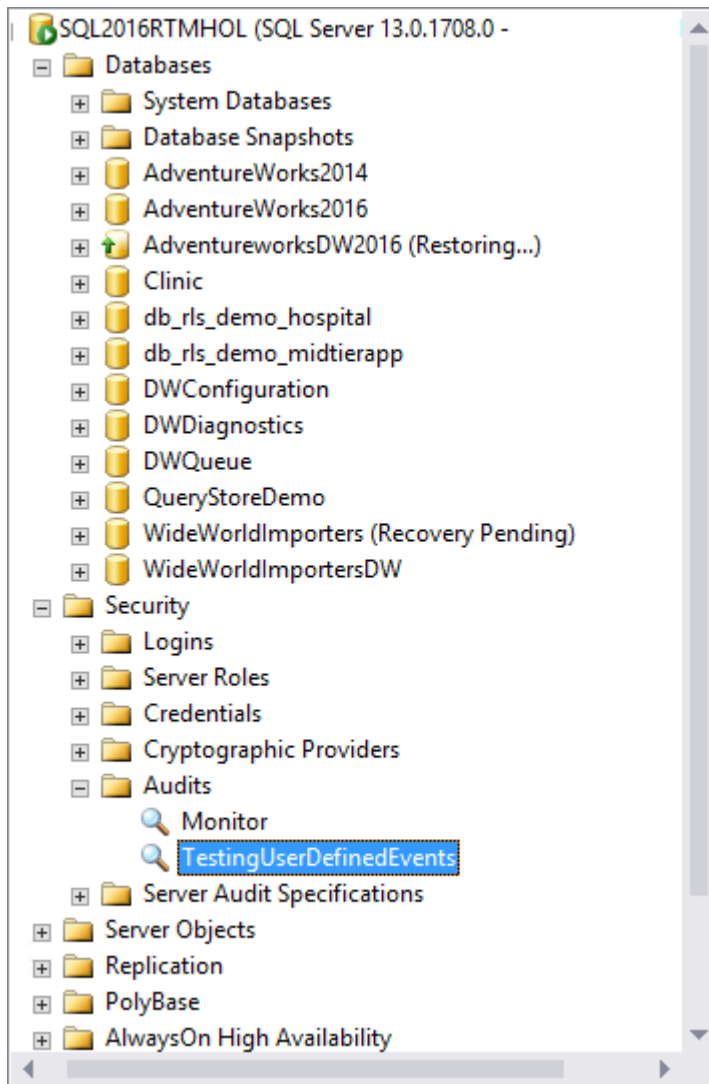


```
USE AdventureWorks2016
select * from HumanResources.EmployeePayHistory where BusinessEntityID= 4
GO
Update HumanResources.EmployeePayHistory set rate = 69.8462
where BusinessEntityID=4 and RateChangeDate = '2007-12-05 00:00:00.000'
GO
```

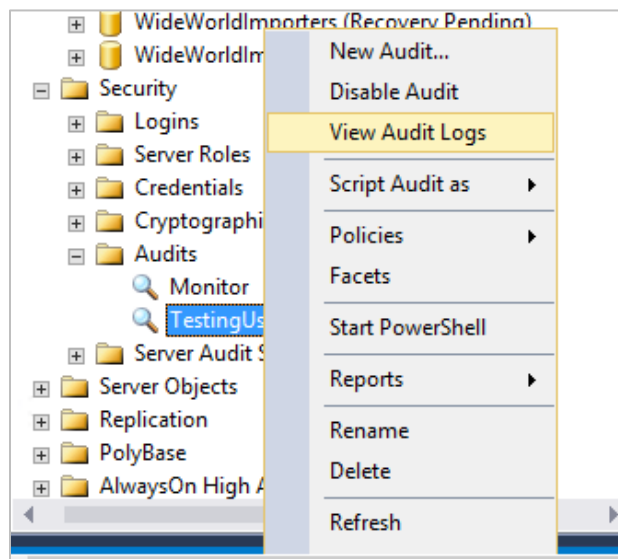
	BusinessEntityID	RateChangeDate	Rate	PayFrequency	ModifiedDate
1	4	2007-12-05 00:00:00.000	8.62	2	2007-11-21 00:00:00.000
2	4	2010-05-31 00:00:00.000	23.72	2	2010-05-16 00:00:00.000
3	4	2011-12-15 00:00:00.000	29.8462	2	2011-12-01 00:00:00.000

Check that pay increase was recorded

1. From SQL Server 2016 Management Studio in Object Explorer, expand **Security** and then expand **Audits**. (If this node is already open, refresh it by right-clicking the node and then selecting **Refresh**.)



2. Right-click **TestingUserDefinedEvents**, and then select **View Audit Logs**.



3. You can see that the user-defined event was recorded in the logs for Employee 4.

- ☒ Audit Collection
 - ☐ Monitor
 - ☒ TestingUserDefinedEvents
- ☐ Windows NT

Status

Last Refresh:

6/17/2016 7:55:59 AM

Filter: None

[View filter settings](#)

Progress



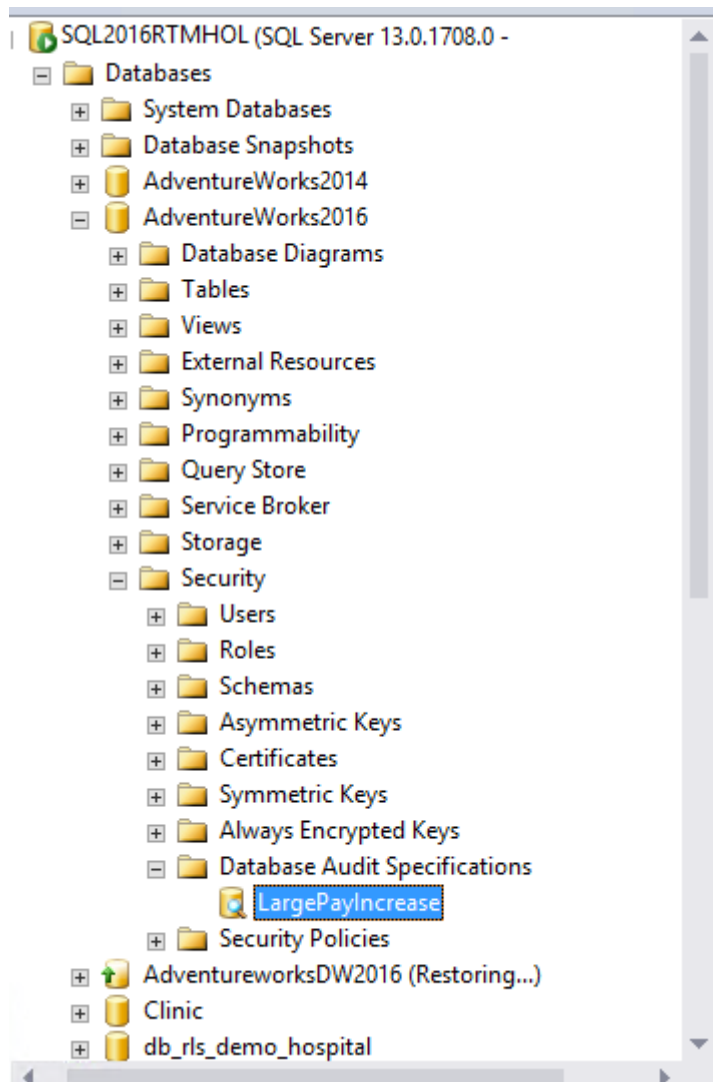
Done (2 records).

Log file summary: No filter applied

Date	Event Time	Server Instance Name	Action ID
✓ 6/17/2016 7:52:56 AM	07:52:56.0848730	SQL2016RTMHOL	USER DEFINED
✓ 6/17/2016 7:46:56 AM	07:46:56.7011932	SQL2016RTMHOL	AUDIT SESSION

Selected row details:

'SalaryMonitor'/></tsql_stack>
 File Name C:\SQLAUDITS\TestingUserDefinedEvents_B1CBE008-02B8-4F7F-B1BF-E6DC4D44C435_0_131106232166850000.sqlaudit
 File Offset 6656
 User Defined Event ID 27
 User Defined Information Employee 4 pay rate increased more than 20%



4. Click **Close**.

Audit filtering

Audit filtering allows the filtering of unwanted audit events before they are written to the audit log. As part of an auditing policy, you can log which users are accessing selected tables that contain sensitive data. To do this, you will create an audit filter to record the table, user, date, and time when the table was accessed. You will first create an audit called **Payrole_Security_Audit**.

1. From Microsoft SQL Server 2016 Management Studio, select **File**. Next, select **Open**, and then click **File**.
2. Browse to **C:\SQL Server 2016 HOLs\Organizational Security and Auditing\Scripts** and select **E2C-1-#4-Audit**.
3. Click **Open**.
4. Click **Execute**.

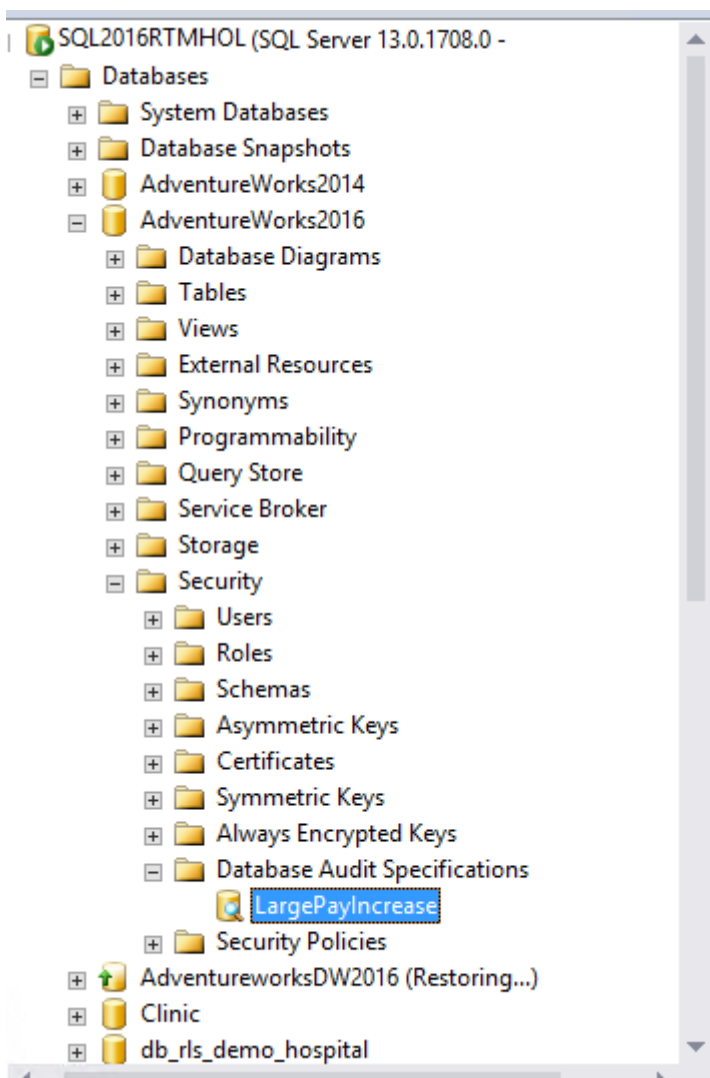
```
USE master ;
GO
-- Create the server audit
CREATE SERVER AUDIT Payrole_Security_Audit
TO FILE (FILEPATH = 'C:\SQLAUDITS' ) ;
GO
-- Enable the server audit
ALTER SERVER AUDIT Payrole_Security_Audit WITH (STATE = ON);
GO
-- Move to the target database
USE AdventureWorks2016 ;
GO
-- Create the database audit specification
CREATE DATABASE AUDIT SPECIFICATION Audit_Pay_Tables
FOR SERVER AUDIT Payrole_Security_Audit
ADD (SELECT , INSERT ON HumanResources.EmployeePayHistory BY dbo )
WITH (STATE = ON) ;
```

100 %

Messages

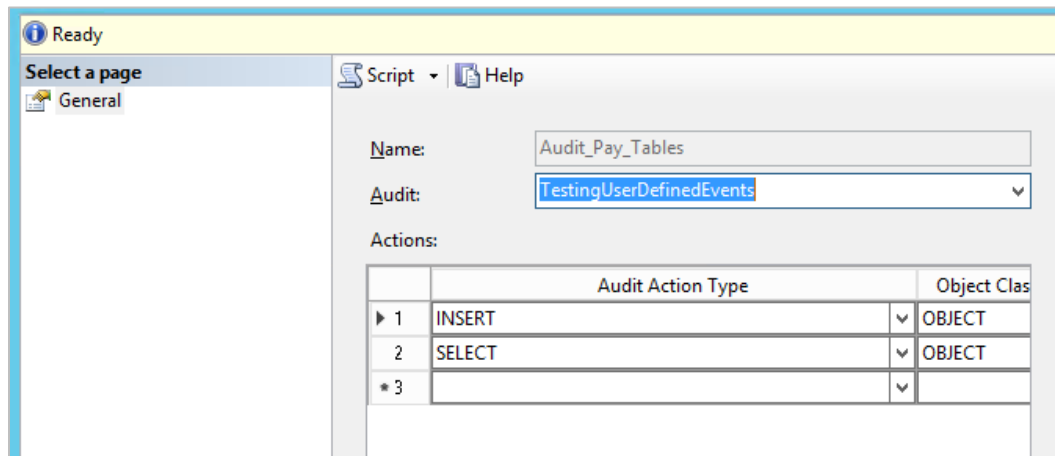
Command(s) completed successfully.

5. To confirm that the audit was created: expand **Databases**, and then expand **AdventureWorks 2016** → **Security** → **Database Audit Specifications** in Management Studio Object Explorer. (If this node is already open, refresh it by right-clicking the node and then selecting **Refresh**.) Locate the audit report called **Audit_Pay_Tables**.



6. Right-click **Audit_Pay_Tables**, and then select **Properties**.

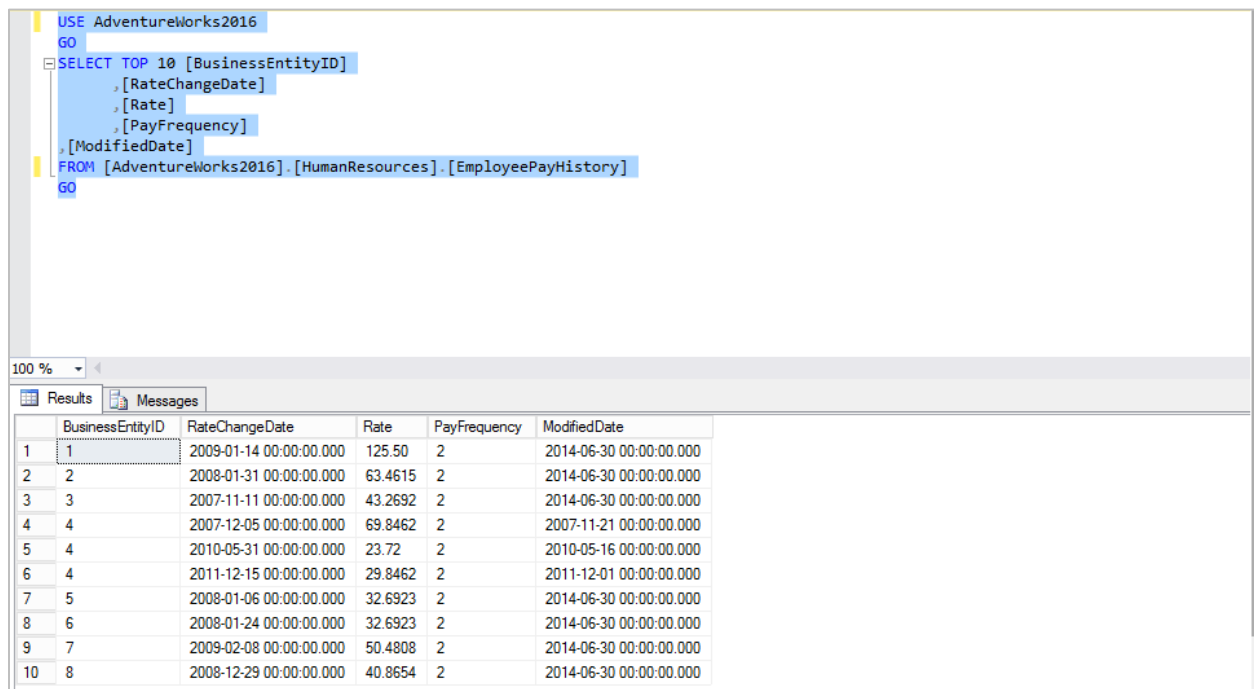
Notice that, if anyone uses SELECT in the **EmployeePayHistory** table, it is recorded in the audit logs.



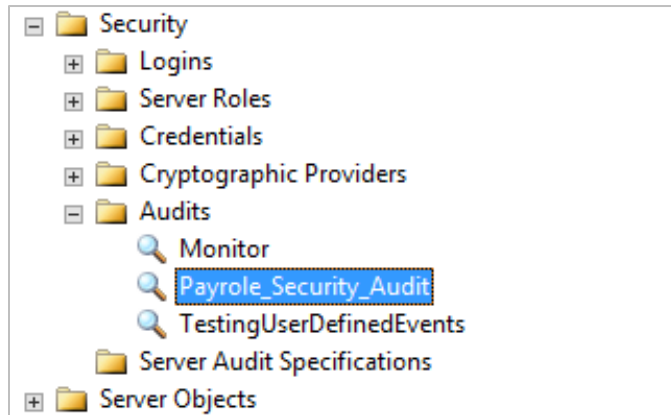
7. Click **OK**.

Now you will test that the audit is working by running the following SELECT query:

1. From SQL Server 2016 Management Studio, select **File**. Next, select **Open** and then click **File**.
2. Browse to **C:\SQL Server 2016 HOLs\Organizational Security and Auditing\Scripts** and select **E2C-1-#3**. This script selects information from the **EmployeePayHistory** table, which should create an entry in the audit log.
3. Click **Open**.
4. Click **Execute**.



- Check that the query was recorded in the audit log: in Object Explorer, expand **Security**, and then expand **Audits**. (If this node is already expanded, right-click the node, and then select **Refresh**.)



- Right-click **Payroll_Security_Audit**, and then select **View Audit Logs**.

You can see that, when the SELECT query was run on the **EmployeePayHistory** table, the user, date, and time were recorded.

Date	Event Time	Server Instance Name	Action ID
6/17/2016 8:03:29 AM	08:03:29.9307581	SQL2016RTMHOL	SELECT
6/17/2016 8:00:02 AM	08:00:02.5995541	SQL2016RTMHOL	AUDIT SESSION C

Selected row details:

Date: 6/17/2016 8:03:29 AM
Log: Audit Collection (Payroll_Security_Audit)

Event Time: 08:03:29.9307581
Server Instance Name: SQL2016RTMHOL
Action ID: SELECT
Class Type: TABLE
Sequence Number: 1
Succeeded: True
Permission Bit Mask: 0x0000000000000001
Column Permission: True
Session ID: 68
Server Principal ID: 259
Database Principal ID: 1
Target Server Principal ID: 0
Target Database Principal ID: 0
Object ID: 1813581499

Status

Last Refresh: 6/17/2016 8:04:19 AM
Filter: None
[View filter settings](#)

Progress

Done (2 records).

Session Server Principal Name: SQL2016RTMHOL\labuser
Server Principal Name: SQL2016RTMHOL\labuser
Server Principal SID: 0x13000000210001314314039700322017213010210320244100
Database Principal Name: dbo
Target Server Principal Name:
Target Server Principal SID: NULL
Target Database Principal Name:
Database Name: AdventureWorks2016
Schema Name: HumanResources
Object Name: EmployeePayHistory
Statement: SELECT TOP 10 [BusinessEntityID]
.[RateChangeDate]

Deadlocking

A deadlock occurs when there is a cyclic dependency between two or more threads for some set of resources. For example:

- Transaction A acquires a share lock on row 1.
- Transaction B acquires a share lock on row 2.
- Transaction A now requests an exclusive lock on row 2 but is blocked until transaction B finishes and releases the share lock it has on row 2.
- Transaction B now requests an exclusive lock on row 1 but is blocked until transaction A finishes and releases the share lock it has on row 1.

Both transactions in a deadlock will wait forever unless the deadlock is broken by an external process. The Microsoft SQL Server Database Engine deadlock monitor periodically checks for tasks that are in a deadlock. If the monitor detects a cyclic dependency, it chooses one of the tasks as a victim and terminates its transaction with an error. This allows the other task to complete its transaction. The application with the transaction that terminated with an error can retry the transaction, which usually completes after the other deadlocked transaction has finished.

Learn more about [minimizing and troubleshooting deadlocks](#).

New transaction — commit/rollback audit group events

This event is raised for BEGIN TRANSACTION, ROLLBACK TRANSACTION, and COMMIT TRANSACTION operations, both for explicit calls to those statements and implicit transaction operations. This event is also raised for UNDO operations for individual statements caused by the rollback of a transaction.

Learn more about transaction rollback group audit events:

- [Rollback Transaction](#)
- [Commit Transaction](#)

Contained database authentication

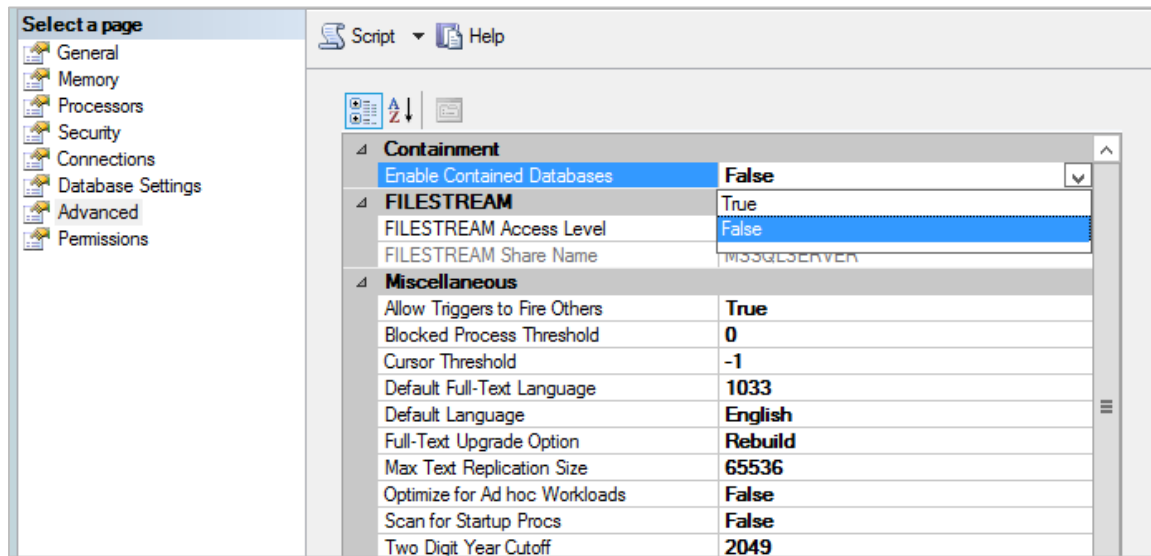
A contained database includes all database settings and metadata required to define the database, without configuration dependencies on the instance of the database engine where the database is installed. Users can connect to the database without authenticating a login at the database engine level. Isolating the database from the database engine makes it possible to move the database easily to another instance of SQL Server.

In this exercise, you will create a new contained database to which user John has read-only access.

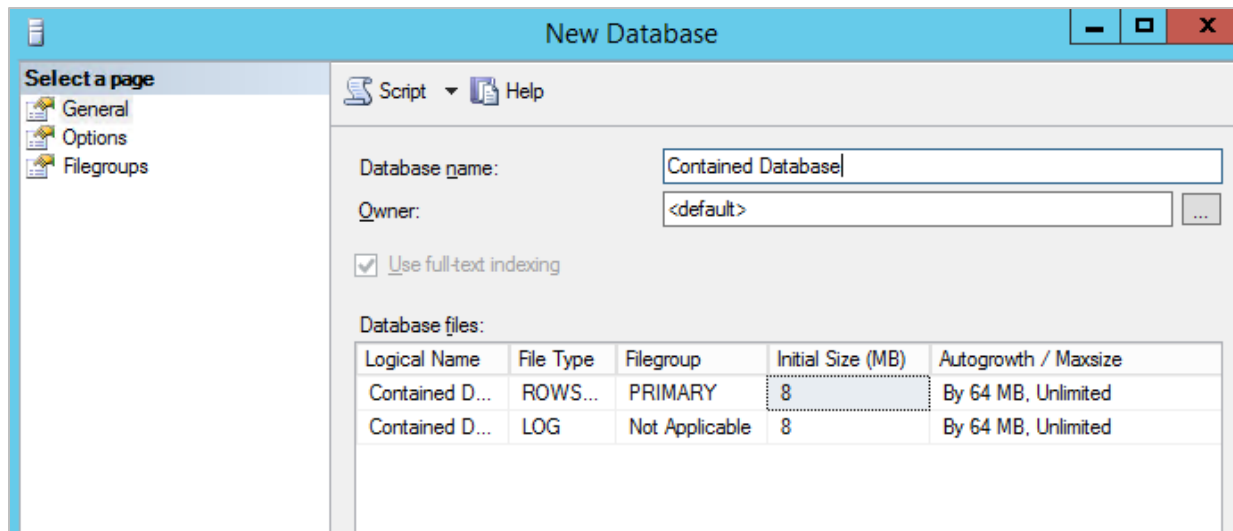
Enable server for contained databases


1. Ensure that you have SQL Server Management Studio open with a connection to the server. (If you do not, open Management Studio from the Windows Start screen, enter **Database Engine** as the server type, **SQL2016RTMHOL** as the server name, and then select **Windows Authentication** before clicking **Connect**.)
2. In Object Explorer, right-click **SQL2016RTMHOL**, and then select **Properties**.

3. Select **Advanced**, and then change **Enable Contained Databases** from False to **True**.



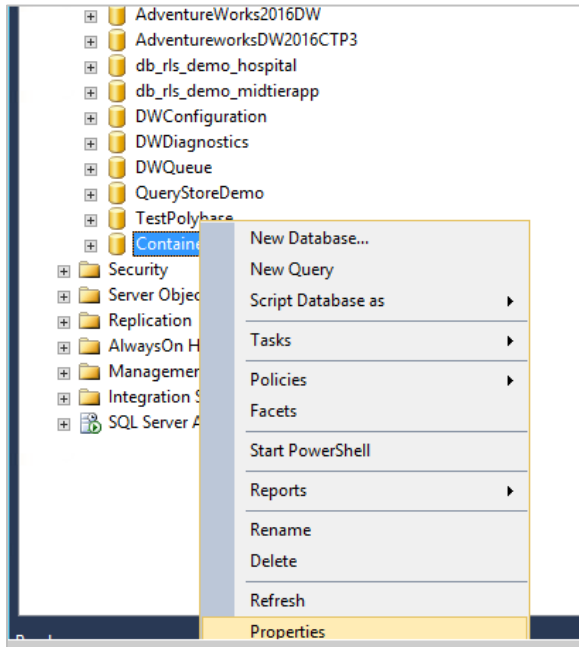
4. Click **OK**.
5. In Object Explorer, right-click **Databases**, and then select **New Database**.
6. For Database name, enter **ContainedDatabase**.



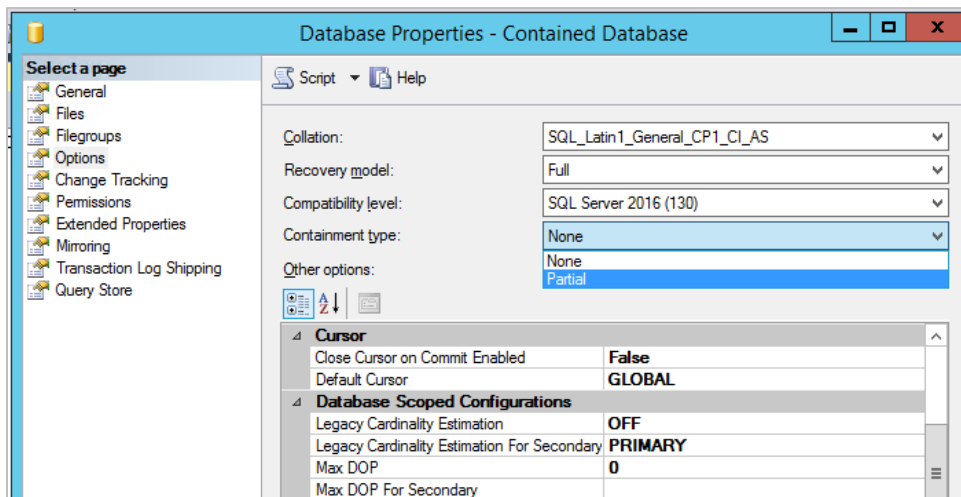
7. Click **OK**.
8. In Object Explorer, click **Refresh** .

Set the database as being contained

1. Expand **Databases**.
2. Right-click **ContainedDatabase** in Object Explorer, and then select **Properties**.



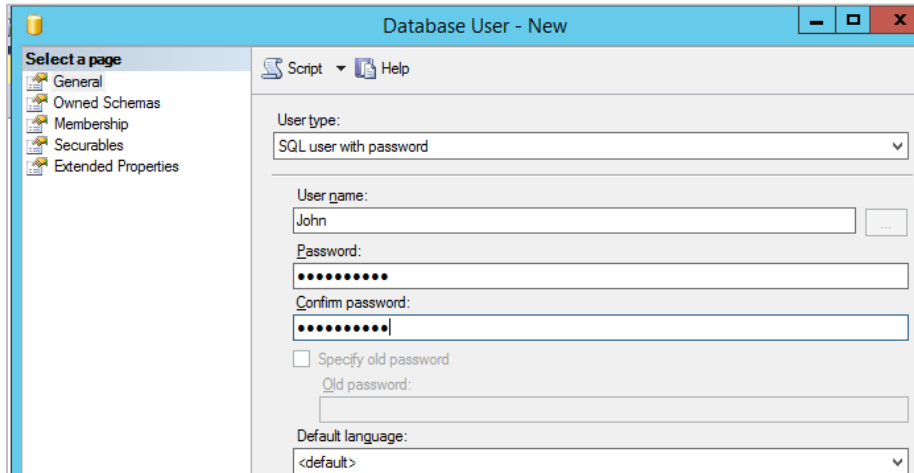
3. Click the **Options** page, and then change **Containment type** from None to **Partial**.



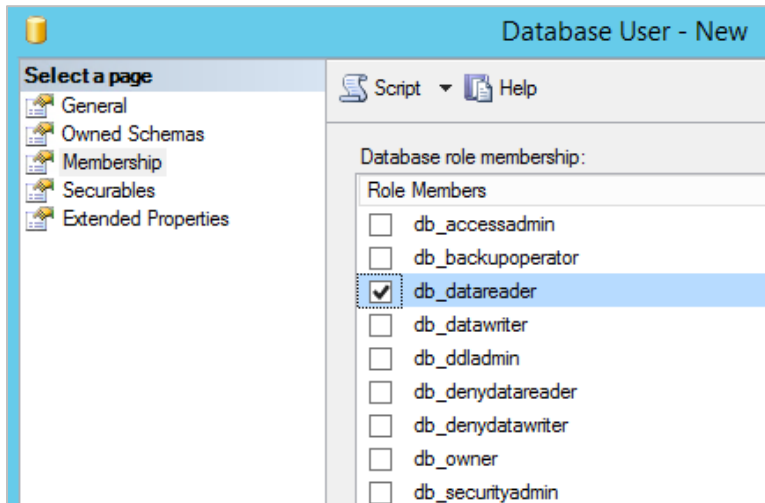
4. Click **OK**.

Add the user John to the database

1. Expand the **ContainedDatabase** node in **Object Explorer** → **Security** → **Users**.
2. Right-click **Users**, and then select **New User**.
3. Ensure the user type is **SQL user with password**, and then type **John** in the **User name** field. In the **Password** box, type **pass@word1**, and then type the same password in the **Confirm password** field.



4. Click **Membership**, and then select **db_datareader**.



5. Click **OK**.

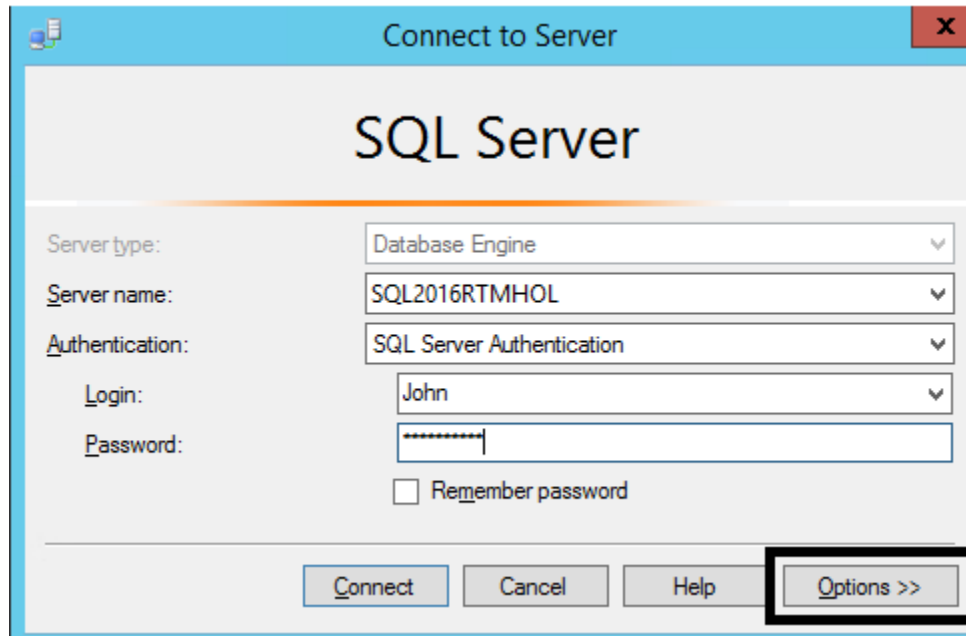
You have now created a contained database that allows John read-only access.

Note: John does not have a SQL logon account, as he only has a logon account to the database called **ContainedDatabase**.

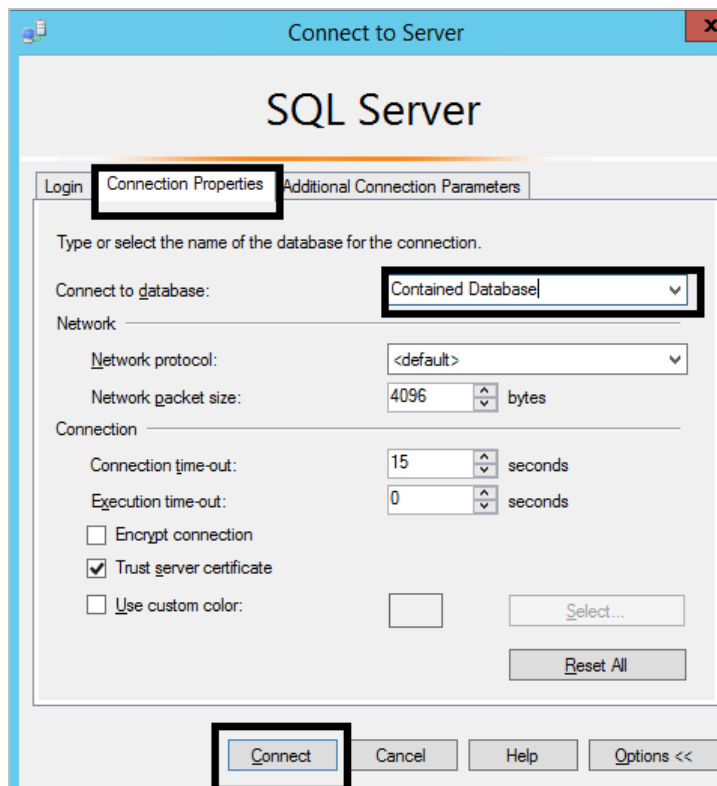
Test user access to the contained database

1. To confirm that John can now access the contained database, click **Connect** at the top of Object Explorer, and then click **Database Engine**.
2. Ensure that authentication is set to **SQL Server Authentication** and the server name is **SQLSERVER**.
3. In the **Login** box, type **John**. In the **Password** box, type **pass@word1**.
4. Click **Options**.

Note: Do **not** click the connect button.

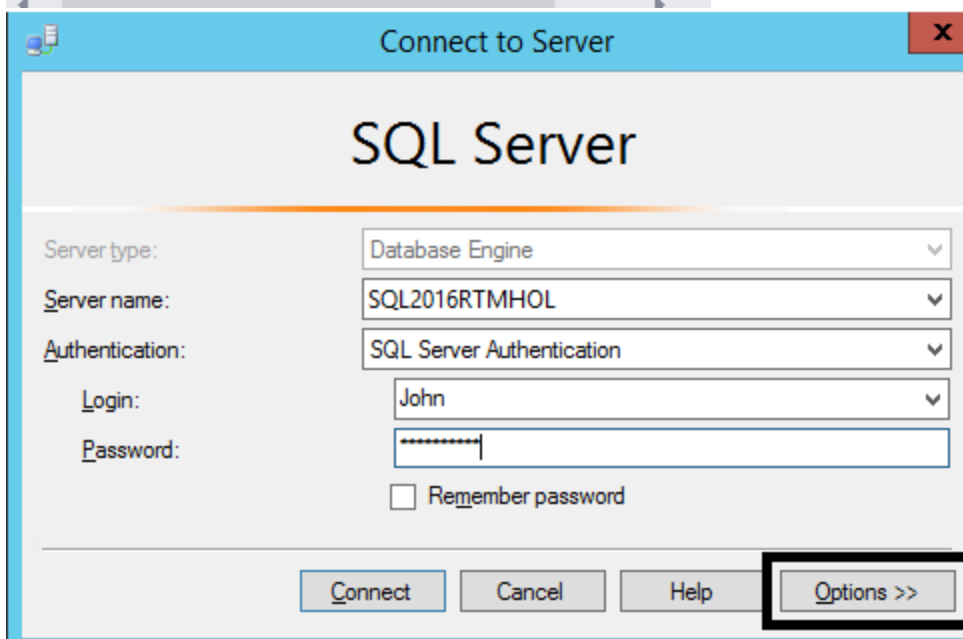
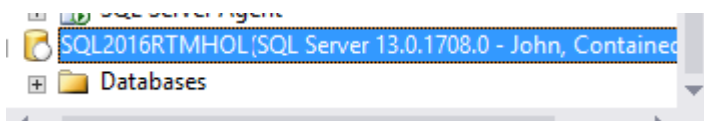


5. Select the tab **Connection Properties** and, in **Connect to database**, manually enter **ContainedDatabase**.

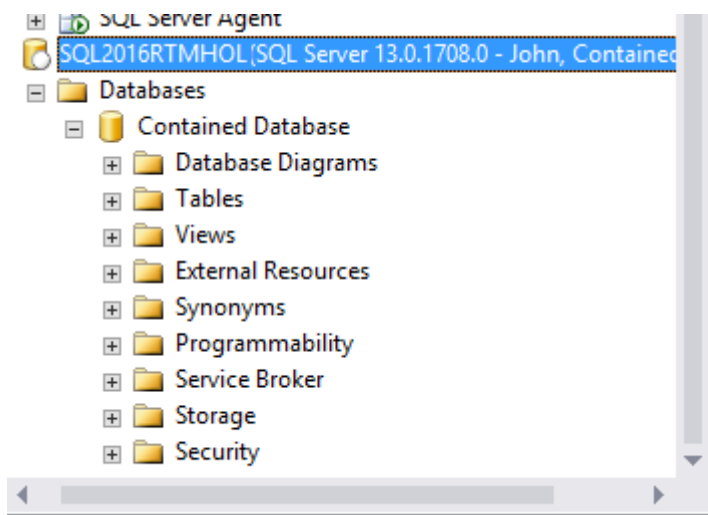


6. Click **Connect**.

You are now connected to the **ContainedDatabase**.



7. Expand **Databases**.



Summary

Every SQL Server securable has associated permissions that can be granted to a principal. Permissions in the Database Engine are managed at the server level assigned to logins and server roles, and, at the database level, assigned to database users and database roles.

Auditing an instance of the SQL Server Database Engine or an individual database involves tracking and logging events that occur on the Database Engine. SQL Server audit lets you create server audits that can contain server audit specifications for server-level events as well as database audit specifications for database-level events. Audited events can be written to the event logs or to audit files.

Terms of use

© 2016 Microsoft Corporation. All rights reserved.

By using this Hands-on Lab, you agree to the following terms:

The technology/functionality described in this Hands-on Lab is provided by Microsoft Corporation in a “sandbox” testing environment for purposes of obtaining your feedback and to provide you with a learning experience. You may only use the Hands-on Lab to evaluate such technology features and functionality and provide feedback to Microsoft. You may not use it for any other purpose. You may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, or sell this Hands-on Lab or any portion thereof.

COPYING OR REPRODUCTION OF THE HANDS-ON LAB (OR ANY PORTION OF IT) TO ANY OTHER SERVER OR LOCATION FOR FURTHER REPRODUCTION OR REDISTRIBUTION IS EXPRESSLY PROHIBITED.

THIS HANDS-ON LAB PROVIDES CERTAIN SOFTWARE TECHNOLOGY/PRODUCT FEATURES AND FUNCTIONALITY, INCLUDING POTENTIAL NEW FEATURES AND CONCEPTS, IN A SIMULATED ENVIRONMENT WITHOUT COMPLEX SET-UP OR INSTALLATION FOR THE PURPOSE DESCRIBED ABOVE. THE TECHNOLOGY/CONCEPTS REPRESENTED IN THIS HANDS-ON LAB MAY NOT REPRESENT FULL FEATURE FUNCTIONALITY AND MAY NOT WORK THE WAY A FINAL VERSION MAY WORK. WE ALSO MAY NOT RELEASE A FINAL VERSION OF SUCH FEATURES OR CONCEPTS. YOUR EXPERIENCE WITH USING SUCH FEATURES AND FUNCTIONALITY IN A PHYSICAL ENVIRONMENT MAY ALSO BE DIFFERENT.

FEEDBACK. If you give feedback about the technology features, functionality, and/or concepts described in this Hands-on Lab to Microsoft, you give to Microsoft, without charge, the right to use, share, and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies, and services to use or interface with any specific parts of a Microsoft software or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement.

MICROSOFT CORPORATION HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE HANDS-ON LAB, INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. MICROSOFT DOES NOT MAKE ANY ASSURANCES OR REPRESENTATIONS WITH REGARD TO THE ACCURACY OF THE RESULTS, OUTPUT THAT DERIVES FROM USE OF THE VIRTUAL LAB, OR SUITABILITY OF THE INFORMATION CONTAINED IN THE VIRTUAL LAB FOR ANY PURPOSE.