Microsoft

Security - General

Module 4

Learning Units covered in this Module

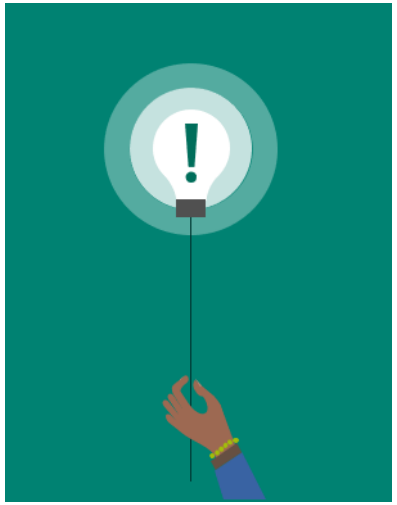
- Lesson 1: Authentication and Authorization
- Lesson 2: Row Level Security
- Lesson 3: Dynamic Data Masking
- Lesson 4: SQL Server Audit

Lesson 1: Authentication and Authorization

Objectives

After completing this learning, you will be able to:

- Understand the difference between Authentication and Authorization
- Understand the difference between Principals and Securables
- Understand how to create logins and users
- Understand how to assign permissions to objects.
- Understand the concept of SQL Server schemas

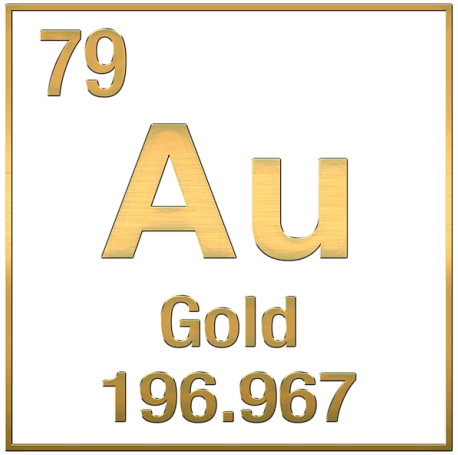


The Security Gold Standard

AUTHENTICATION – Verifies who you are

AUTHORIZATION – Assigns what you can do

AUDITING – Monitors what you did



Authentication Types

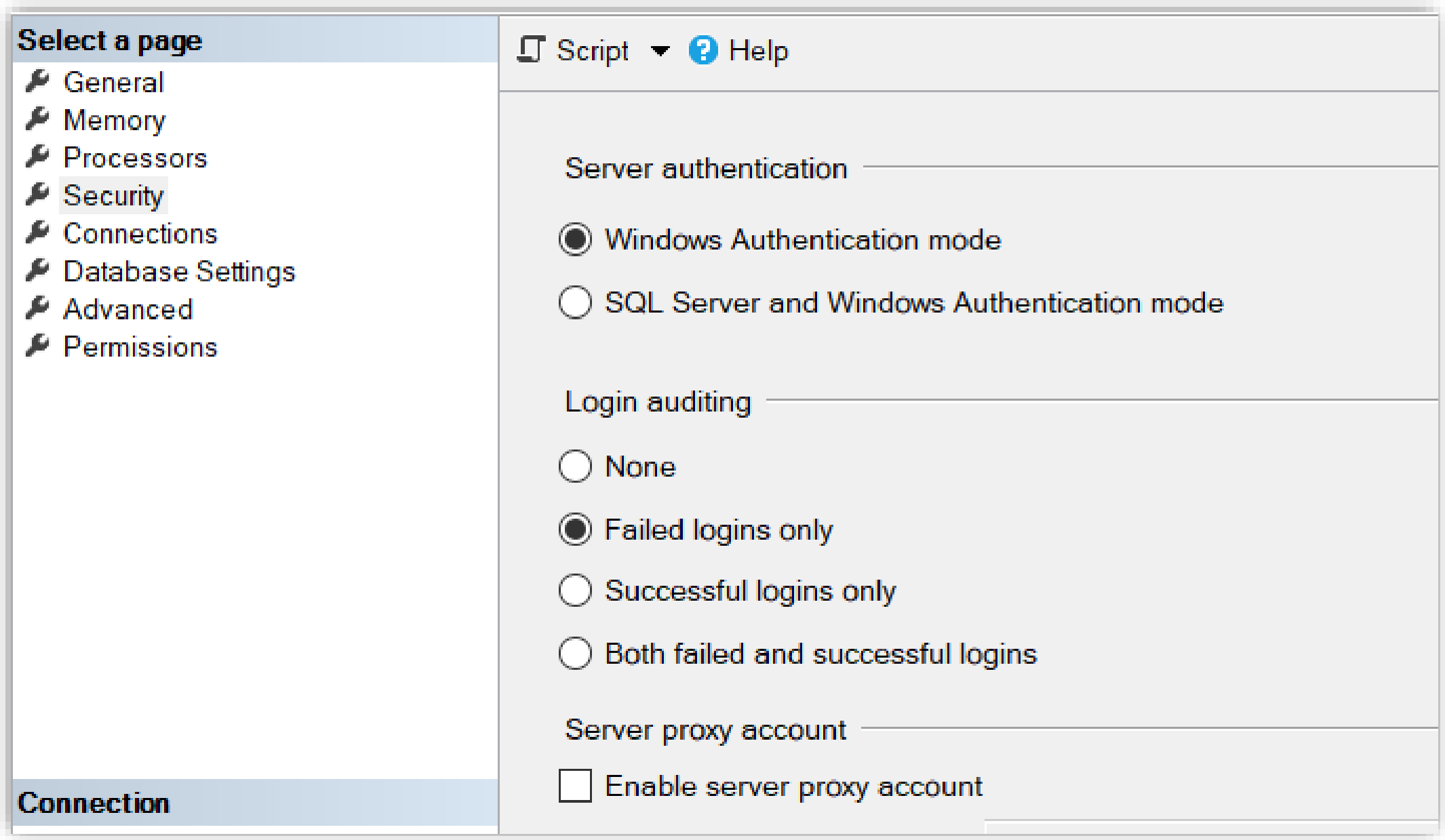
Windows Authentication

- SQL Server validates credentials using Active Directory and then verifies if it has permissions to connect.

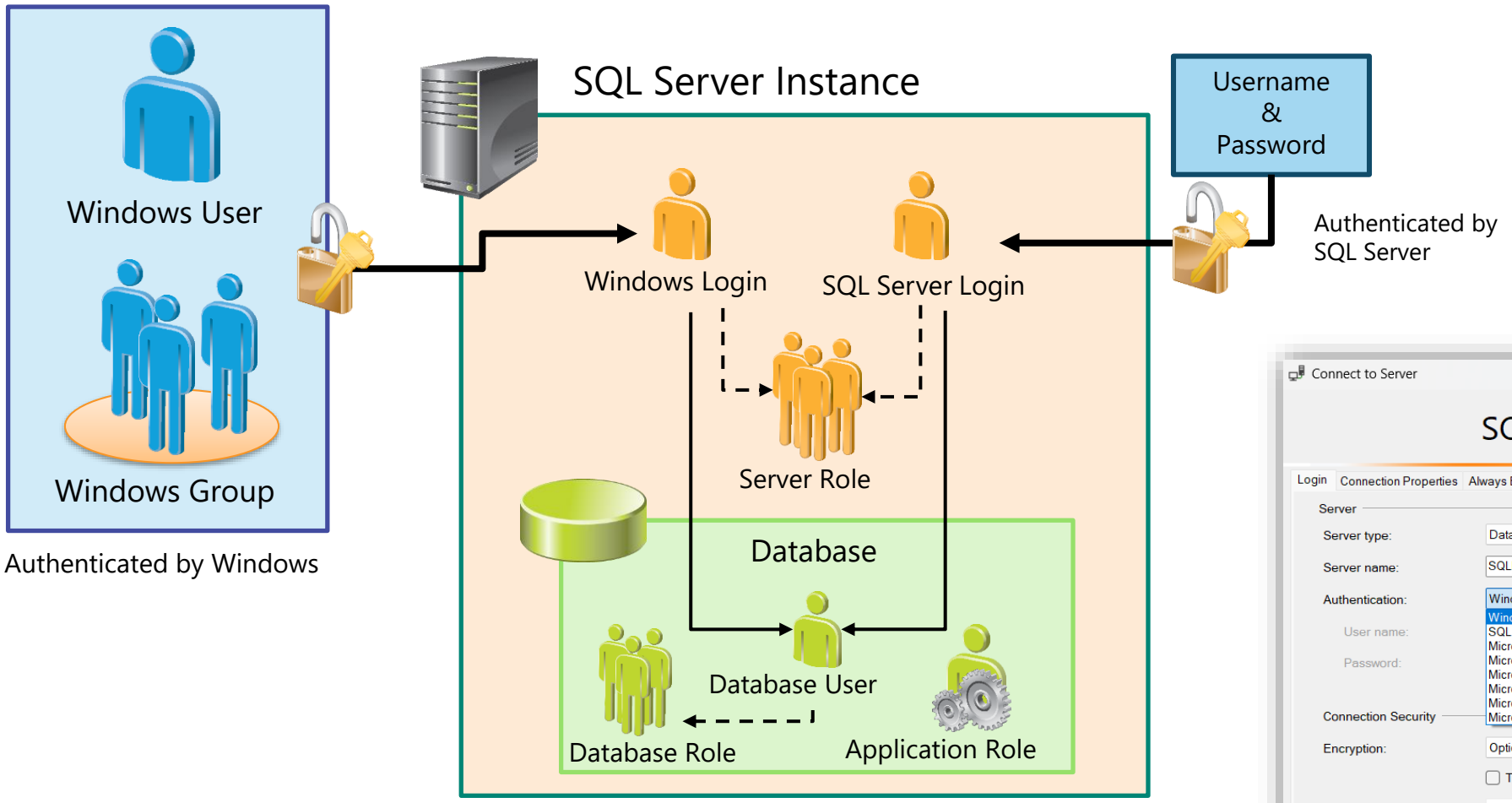
SQL Authentication

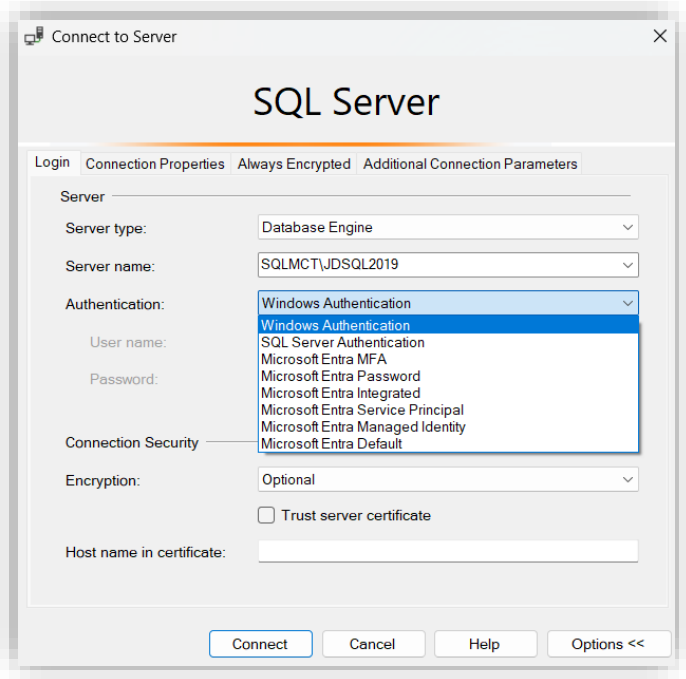
- SQL Server validates the password against a hash stored in master and then verifies if it has permissions to connect.

Server Authentication



Security Principals





Creating Logins

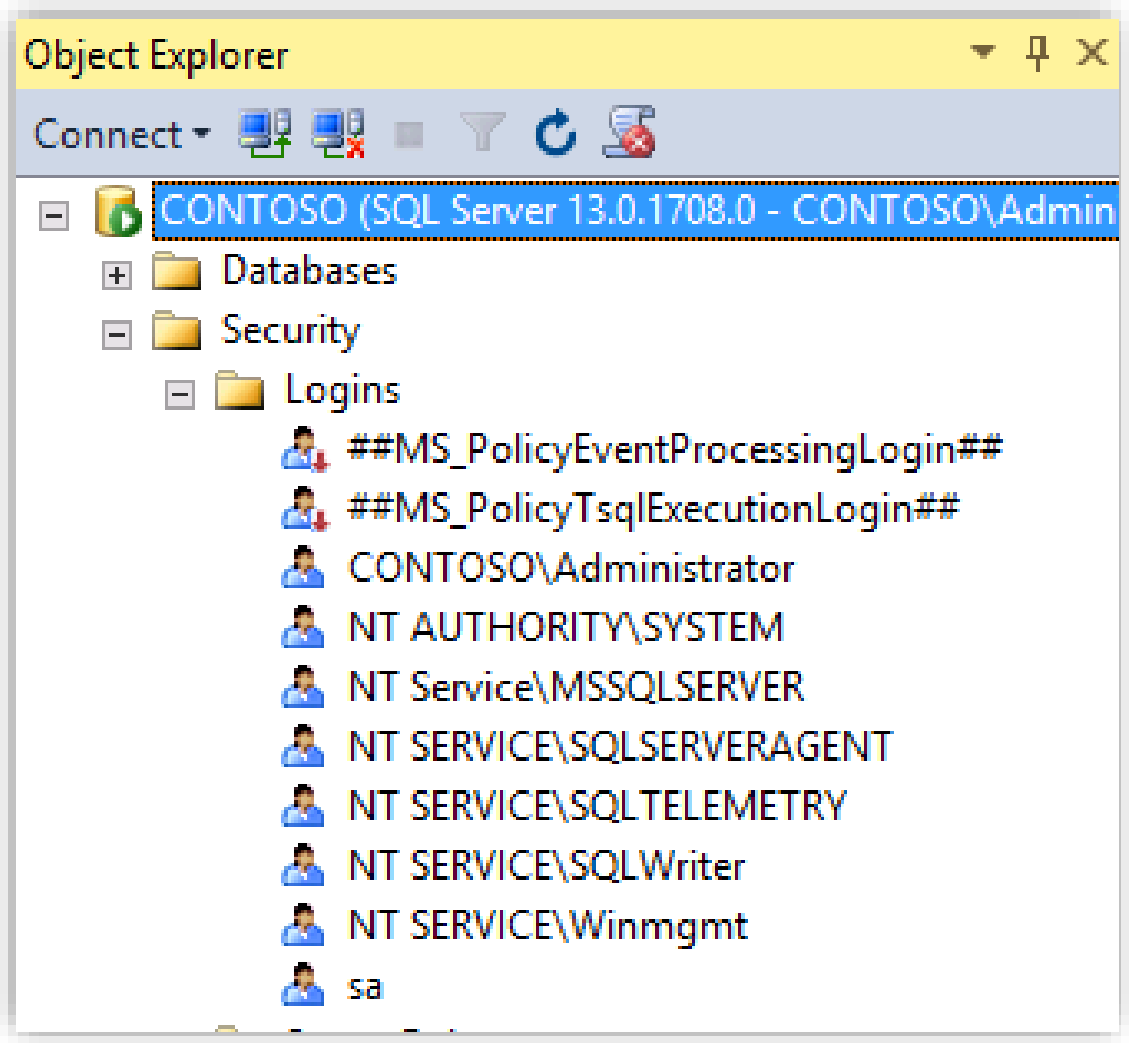
Allows connection to a SQL Server Instance

Two type of logins:

- SQL Login
- Windows Login

Can be created by:

- CREATE LOGIN statement in T-SQL
- SQL Server Management Studio



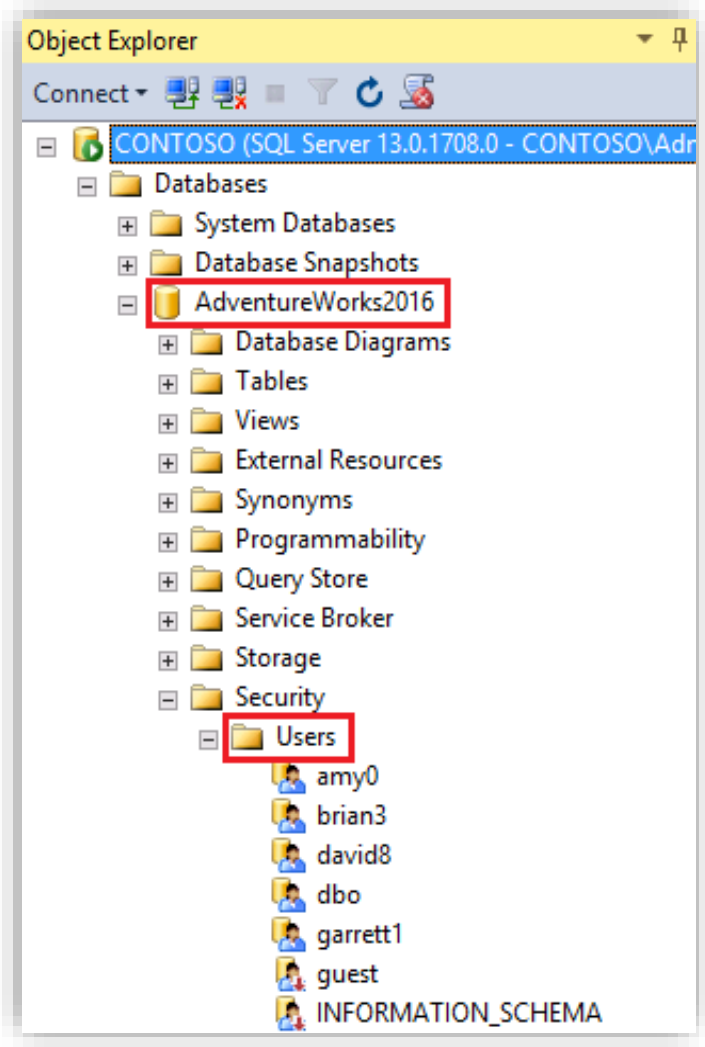
Creating Users

Allow access to a database

Specific to a single database

Type of users:

- Windows user
- SQL User with Password
- SQL User with Login
- SQL User without Password
- User mapped to a certificate
- User mapped to an asymmetric key



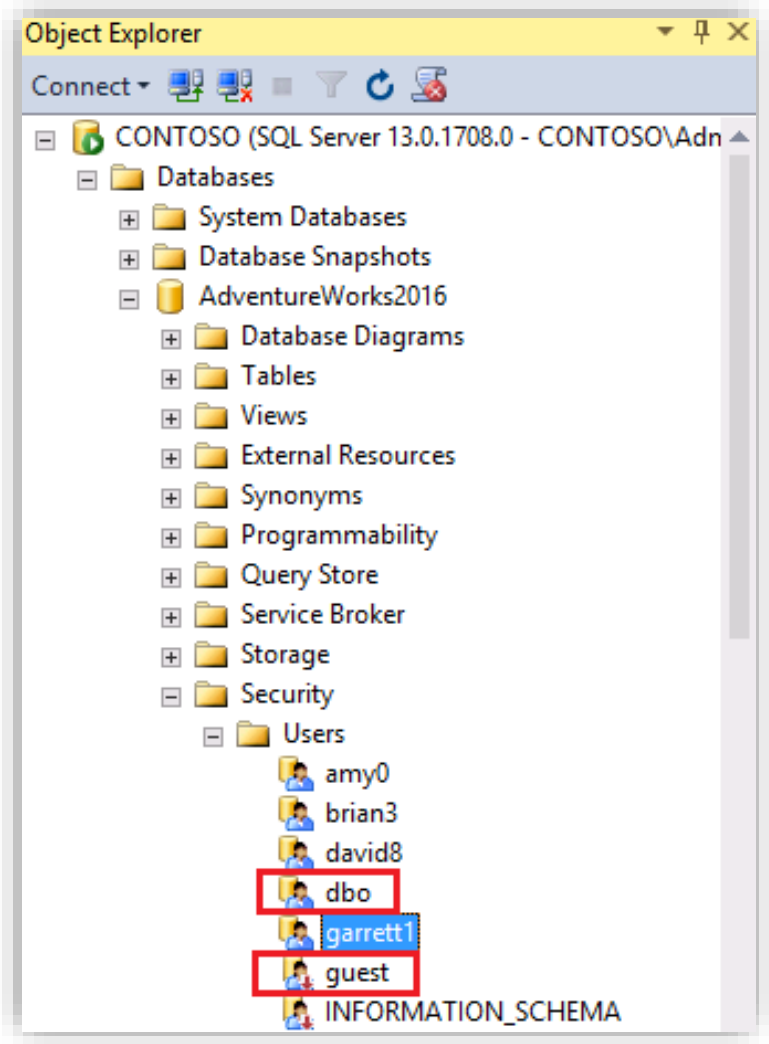
DBO and Guest User

DBO

- Performs all activities in the database
- Members of sysadmin role, SA login, and database owner are mapped to DBO.
- Cannot be deleted

Guest

- Allows logins without user accounts to access database
- Disabled by default in user databases
- Cannot be dropped but you can prevent it from accessing a database
- Must NOT be disabled in master and tempdb



Roles

Server Roles

- Fixed server roles
- User-defined server roles

Database Roles

- Fixed database roles
- User-defined database roles

Application roles

- Assign rights to applications instead of users

Fixed Server Level Roles and Permissions

Role	Description	Server-level Permission
sysadmin	Perform any activity	CONTROL SERVER (with GRANT option)
dbcreator	Create and alter databases	ALTER ANY DATABASE
diskadmin	Manage disk files	ALTER RESOURCES
serveradmin	Configure server-wide settings	ALTER ANY ENDPOINT, ALTER RESOURCES, ALTER SERVER STATE, ALTER SETTINGS, SHUTDOWN, VIEW SERVER STATE
securityadmin	Manage and audit server logins	ALTER ANY LOGIN
processadmin	Manage SQL Server processes	ALTER ANY CONNECTION ALTER SERVER STATE
bulkadmin	Run the BULK INSERT statement	ADMINISTER BULK OPERATIONS
setupadmin	Configure replication and linked servers	ALTER ANY LINKED SERVER

New Server Level Roles introduced in SQL Server 2022

Server-level role	Description
##MS_DatabaseConnector##	Connect to any database without requiring a database User-account.
##MS_LoginManager##	Create, delete and modify logins. Cannot GRANT.
##MS_DatabaseManager##	Create and delete databases.
##MS_ServerStateManager##	Same as the ##MS_ServerStateReader## role but also has the **ALTER SERVER STATE** permission.
##MS_ServerStateReader##	Read all dynamic management views (DMVs) and functions that are covered by **VIEW SERVER STATE**.
##MS_ServerPerformanceStateReader##	Read all dynamic management views (DMVs) and functions that are covered by **VIEW SERVER PERFORMANCE STATE**
##MS_ServerSecurityStateReader##	Read all dynamic management views (DMVs) and functions that are covered by **VIEW SERVER SECURITY STATE**
##MS_DefinitionReader##	Read all catalog views that are covered by **VIEW ANY DEFINITION**
##MS_PerformanceDefinitionReader##	Read all catalog views that are covered by **VIEW ANY PERFORMANCE DEFINITION.**
##MS_SecurityDefinitionReader##	Read all catalog views that are covered by **VIEW ANY SECURITY DEFINITION**.

Public Role

Public is a special role that is at the server and database level.

Every SQL Server login and user belongs to the Public role

Care must be taken when granting permissions to Public server role especially when granting server-level **permissions.**

Fixed Database Level Roles and Permissions

Role	Description
db_owner	Perform any configuration and maintenance activities on the DB and can drop it
db_securityadmin	Modify role membership and manage permissions
db_accessadmin	Add or remove access to the DB for logins
db_backupoperator	Back up the DB
db_ddladmin	Run any DDL command in the DB
db_datawriter	Add, delete, or change data in all user tables
db_datareader	Read all data from all user tables
db_denydatawriter	Cannot add, delete, or change data in user tables
db_denydatareader	Cannot read any data in user tables

Listing Built-in Server and Database Permissions

```
SELECT * FROM sys.fn_builtin_permissions('SERVER')
ORDER BY permission_name;
```

Results | Messages

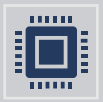
	class_desc	permission_name	type	covering_permission_name	parent_class_desc	parent_covering_permission_name
1	SERVER	ADMINISTER BULK OPERATIONS	ADBO	CONTROL SERVER		
2	SERVER	ALTER ANY AVAILABILITY GROUP	ALAG	CONTROL SERVER		
3	SERVER	ALTER ANY CONNECTION	ALCO	CONTROL SERVER		
4	SERVER	ALTER ANY CREDENTIAL	ALCD	CONTROL SERVER		
5	SERVER	ALTER ANY DATABASE	ALDB	CONTROL SERVER		
6	SERVER	ALTER ANY ENDPOINT	ALHE	CONTROL SERVER		
7	SERVER	ALTER ANY EVENT NOTIFICATION	ALES	CONTROL SERVER		

```
SELECT * FROM sys.fn_builtin_permissions('Database')
ORDER BY permission_name;
```

Results | Messages

	class_desc	permission_name	type	covering_permission_name	parent_class_desc	parent_covering_permission_name
1	DATABASE	ALTER	AL	CONTROL	SERVER	ALTER ANY DATABASE
2	DATABASE	ALTER ANY APPLICATION ROLE	ALAR	ALTER	SERVER	CONTROL SERVER
3	DATABASE	ALTER ANY ASSEMBLY	ALAS	ALTER	SERVER	CONTROL SERVER
4	DATABASE	ALTER ANY ASYMMETRIC KEY	ALAK	ALTER	SERVER	CONTROL SERVER
5	DATABASE	ALTER ANY CERTIFICATE	ALCF	ALTER	SERVER	CONTROL SERVER
6	DATABASE	ALTER ANY COLUMN ENCRYPTION KEY	ALCK	ALTER	SERVER	CONTROL SERVER
7	DATABASE	ALTER ANY COLUMN MASTER KEY	ALCM	ALTER	SERVER	CONTROL SERVER

Authorization



Process by which SQL server decides whether a given principal can access a resource

Allows granting the specific permissions required rather than granting membership in a fixed role

Provides information and metadata of a securable only to those principals who have permission to access the securable

Allows creating custom permission sets

Works on the principle of *least privilege*

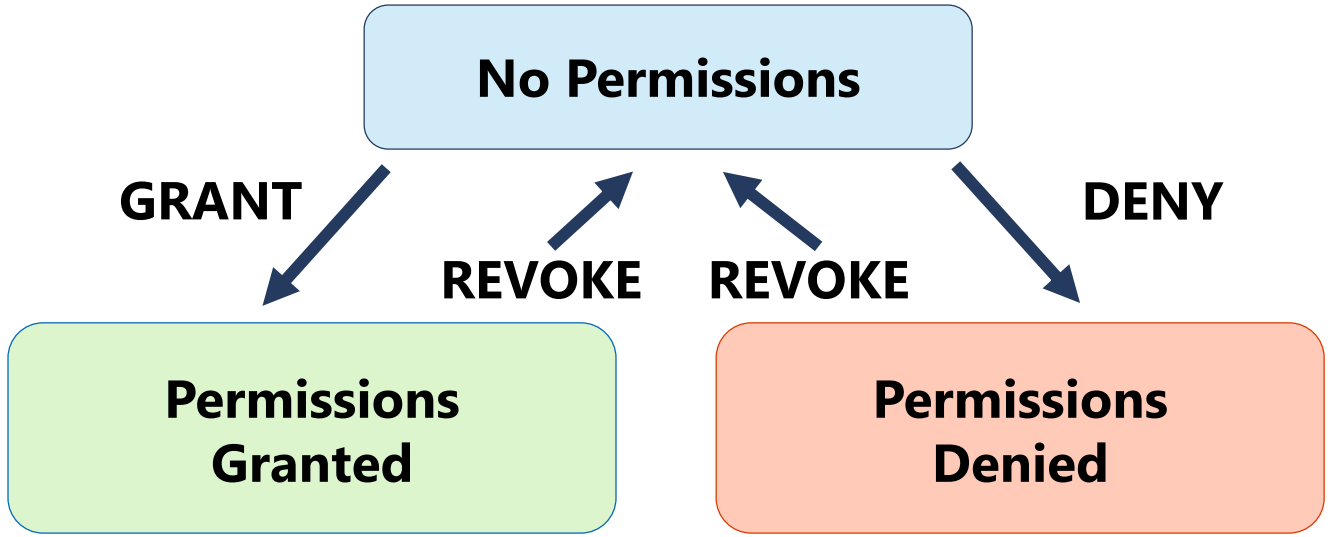
Assigning Permissions to Accounts

GRANT is used to assign a permission

DENY is used to explicitly deny a permission

- Used where permissions inherited through group or role membership
- Should only be used in exceptional circumstances

REVOKE removes either a GRANT or a DENY



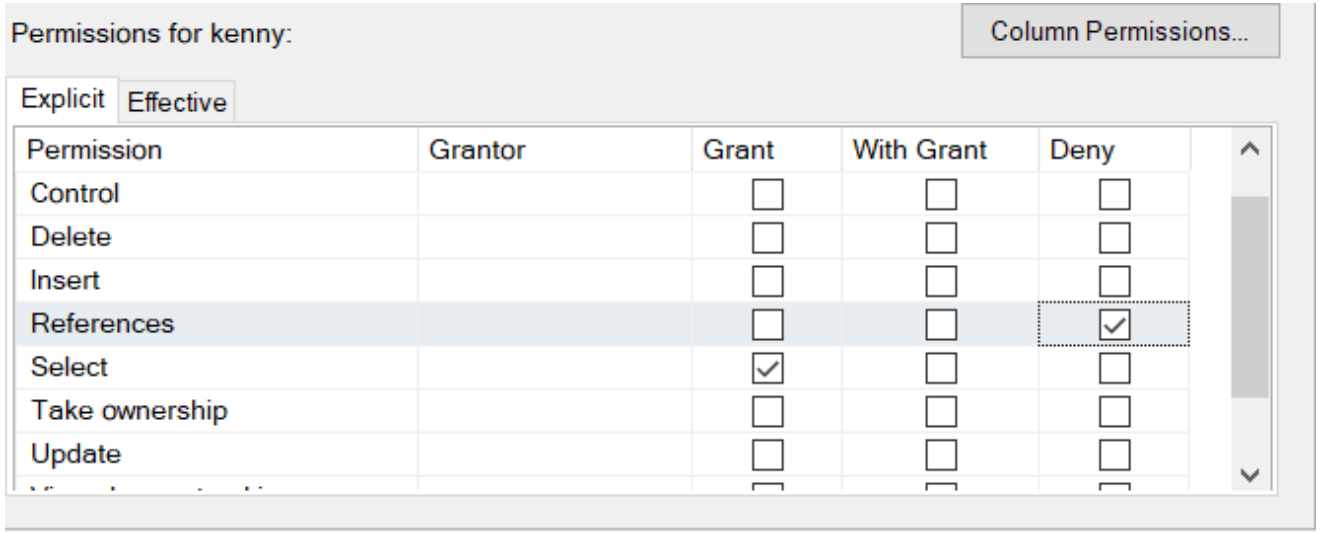
Assigning Permissions to Tables and Views

Grant with Grant allows the user to assign that permission.

Tables and Views can be assigned the same permissions.

Permissions for SELECT, UPDATE, and REFERENCES can also be set at the column level.

Select the Effective Tab to see what permissions have been granted.



Security with Schemas

FQN has the form: ***server.database.schema.object***

In a database, all objects are created within a schema (dbo is default).

Allow their owners full control over objects within the schema

Permissions can be granted at the schema level.

Can contain objects owned by multiple database users

Can be owned by any database principal

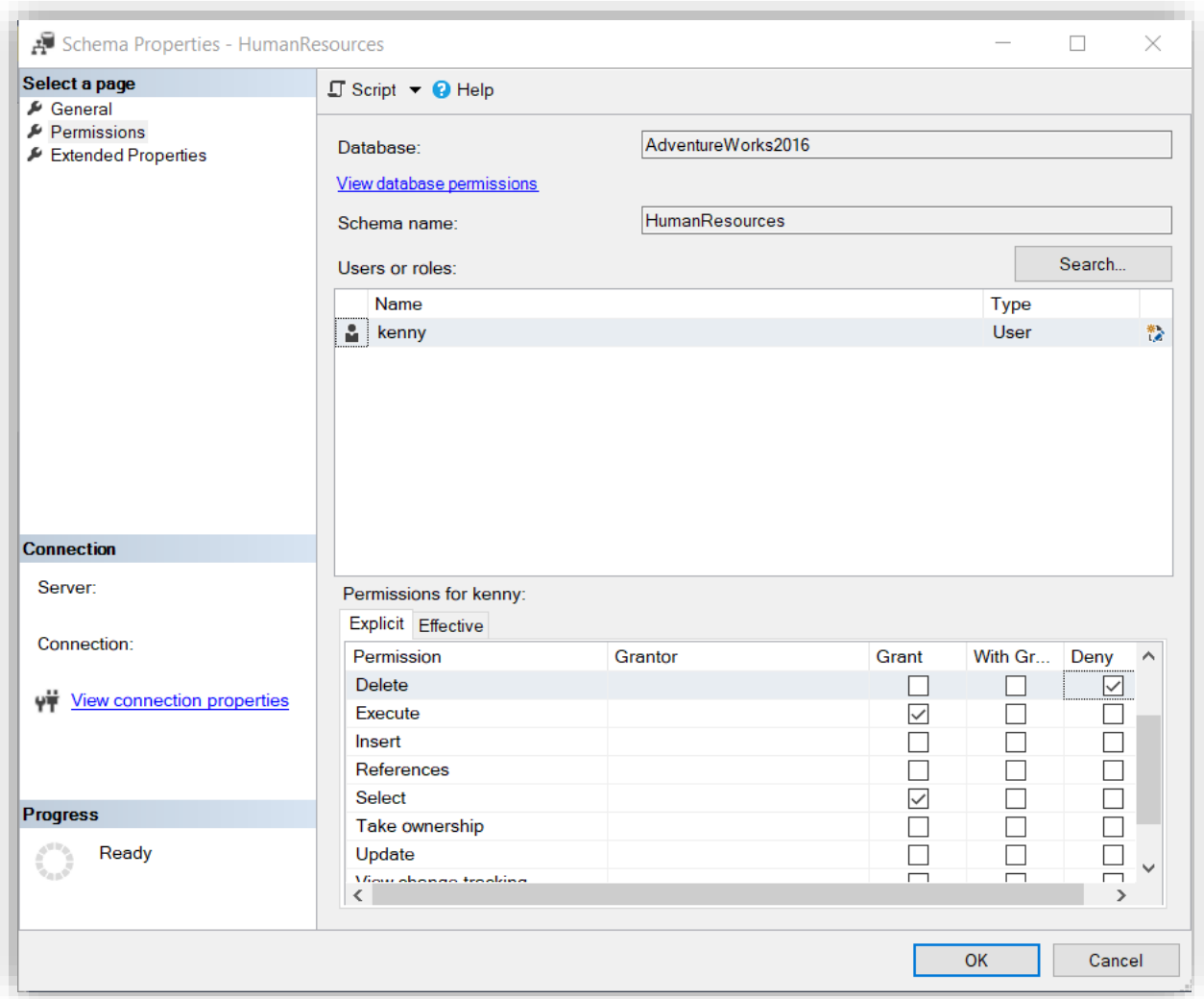
Assign Permissions to a Schema

Permissions assigned at the schema level affect all objects belonging to that schema.

Tables and Views can be assigned the same permissions.

The Execute permission will be applied to all Stored Procedures in the schema.

Select the Effective Tab to see what permissions have been granted.



Questions?

Knowledge Check

What is the difference between Authentication and Authorization?

What are the two types of Logins for SQL Server?

What is an example of a securable?

What are the two Authentication modes?

How should the (sa) account been handled?

How would an admin explicitly restrict access to a table?

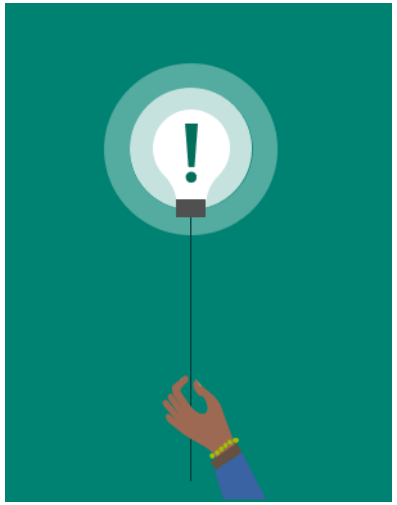
What statement would be used to remove a permission?

Lesson 2: Row Level Security

Objectives

After completing this learning, you will be able to:

- Understand row-level security and how it can be used.

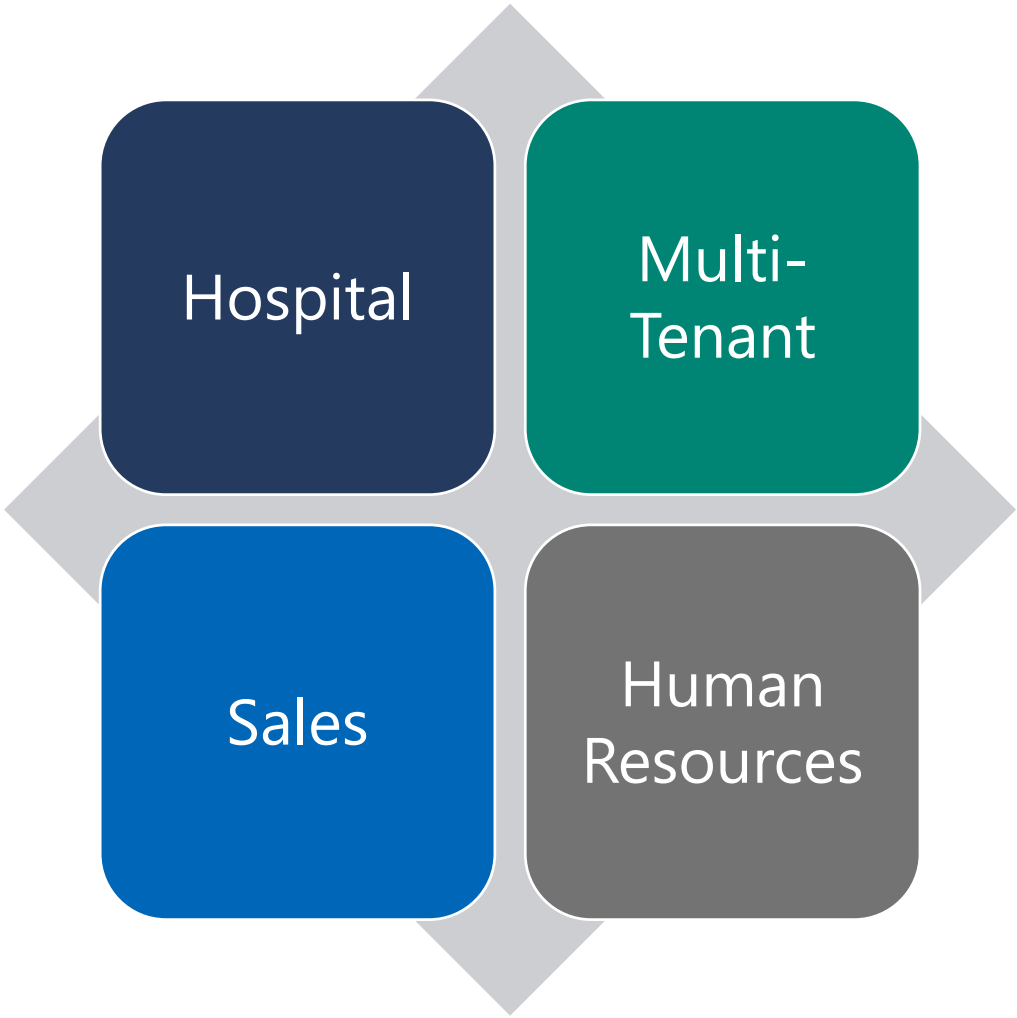


Row-Level Security Overview

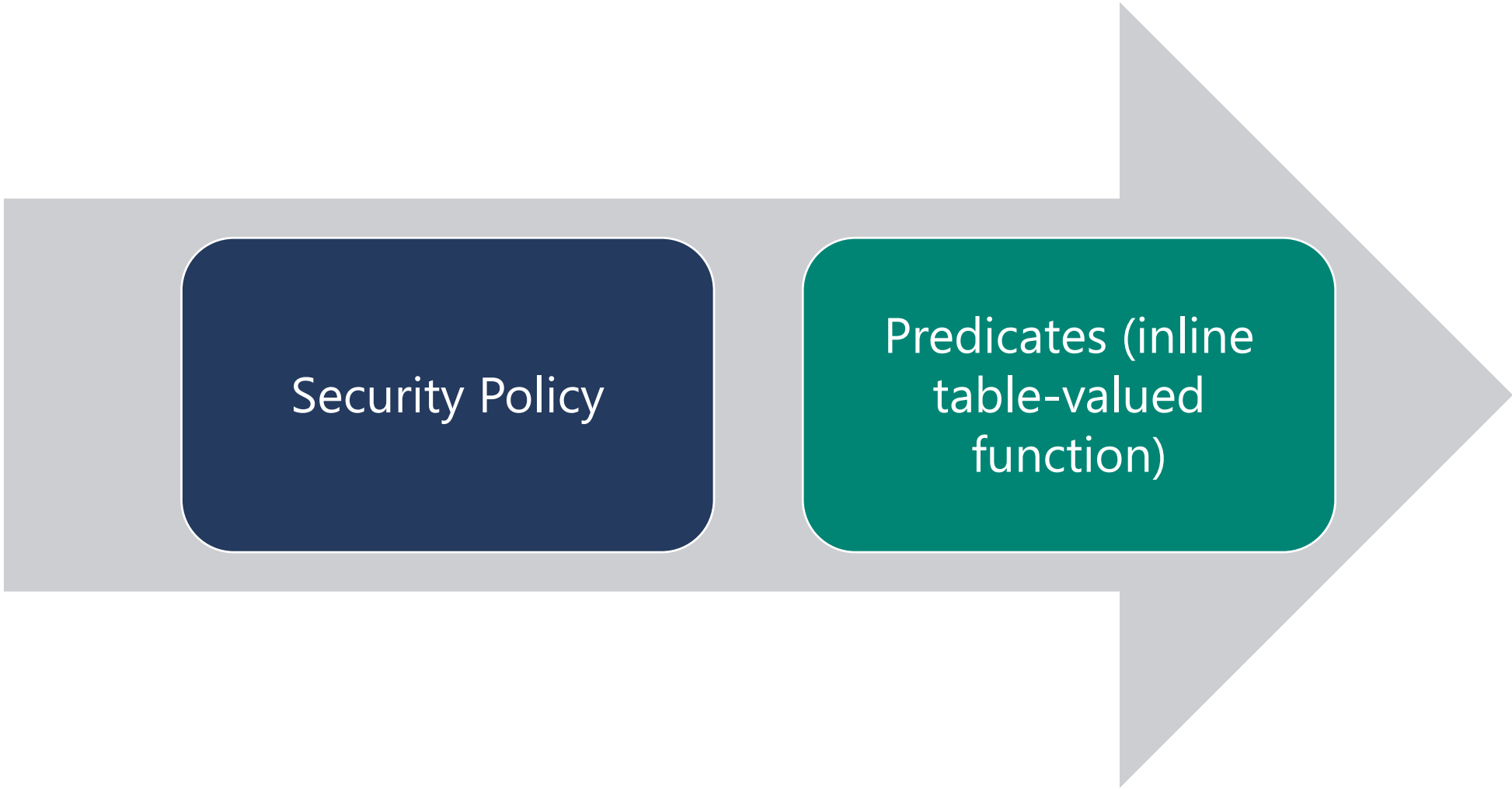
Enables fine grained access control at the row level

Security logic is controlled at the database tier instead of the application tier

Row-Level Security Use Cases



Row-Level Security Components



Row-Level Security Predicates


```
CREATE FUNCTION fn_RowLevelSecurity (@FilterColumnName sysname) RETURNS TABLE WITH
SCHEMABINDING
as
RETURN SELECT 1 as fn_SecureCustomerData
-- filter out records based on database user name
where @FilterColumnName = user_name();
```

Row-Level Security Policy

Security policies are named objects, scoped to a schema that perform filtering using an inline table-valued function.

State setting determines if they are on or off.

```
CREATE SECURITY POLICY FilterCustomer
ADD FILTER PREDICATE dbo.fn_RowLevelSecurity(SalesPersonUserName)
ON dbo.Customer
WITH (STATE = ON);
```

Row-Level Security Permissions

Create, alter or drop policy	• Requires ALTER ANY SECURITY POLICY • Creating or dropping a security policy requires the ALTER permission on the schema
For each predicate added	• SELECT and REFERENCES permissions on the function being used as a predicate • REFERENCES on the target table • REFERENCES on every column from the target table used as an argument

Row-Level Security Best Practices

Create a separate schema for RLS objects

Monitor who has the ALTER ANY SECURITY POLICY – intended for highly privileged users

If the security policy managers have the ALTER ANY SECURITY POLICY permission, they do not need the select permission on the table

Keep predicate functions as simple as possible to prevent performance issues

Row-Level Security Limitations

- Filestream – not supported
- Polybase – not supported
- DBCC SHOW_STATISTICS report statistics on unfiltered data (potential leak)
- Memory-optimized tables – predicate must use WITH NATIVE_COMPILATION option
- Indexed views cannot be created on top of tables that have a security policy
- Change Data Capture (CDC) – can leak rows that should be filtered to db_owner

Demonstration

Row-Level Security

Create a RLS Function and Security Policy

Row-Level Security

You will setup row level security to allow different people to see their territory without seeing data for other territories.

Questions?

Knowledge Check

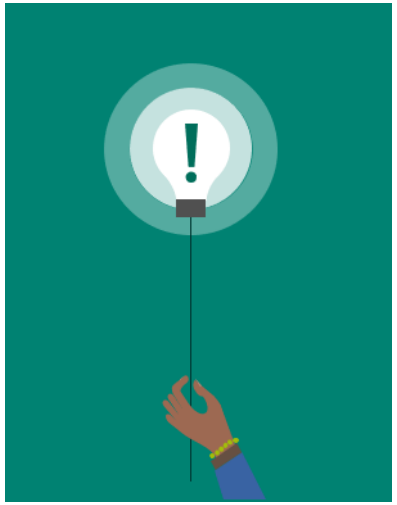
What are some scenarios where row-level security would be beneficial?

Lesson 3: Dynamic Data Masking

Objectives

After completing this learning, you will be able to:

- Understand how dynamic data masking can be utilized to enhance data security.



Dynamic Data Masking Overview

Why use Dynamic Data Masking?			
Limits sensitive data exposure to non-privileged users	Provides control over how much of the sensitive data is revealed	Minimal impact to the application layer	Data in the database remains unchanged

Types of Data Masks

Default (based on data type)

Email

Custom String

Random

Dynamic Data Masking Limitations

The following columns cannot be masked

- Using Always Encrypted
- FILESTREAM
- Computed Columns*
- Column that is a key for a full-text index

Dynamic Data Masking Permissions

ALTER ANY MASK and ALTER permission on the table

Required to add, replace or remove the mask of a column

UNMASK

Required to see unmasked data*

Demonstration

Dynamic Data Masking

Creating and Querying Masked Tables

Questions?

Knowledge Check

Which permission needs to be granted for a user to see the full data view

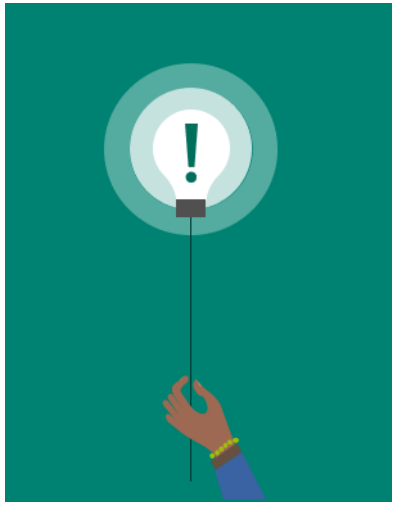
What are the four types of data masks?

Lesson 4: Introduction to SQL Server Audit

Objectives

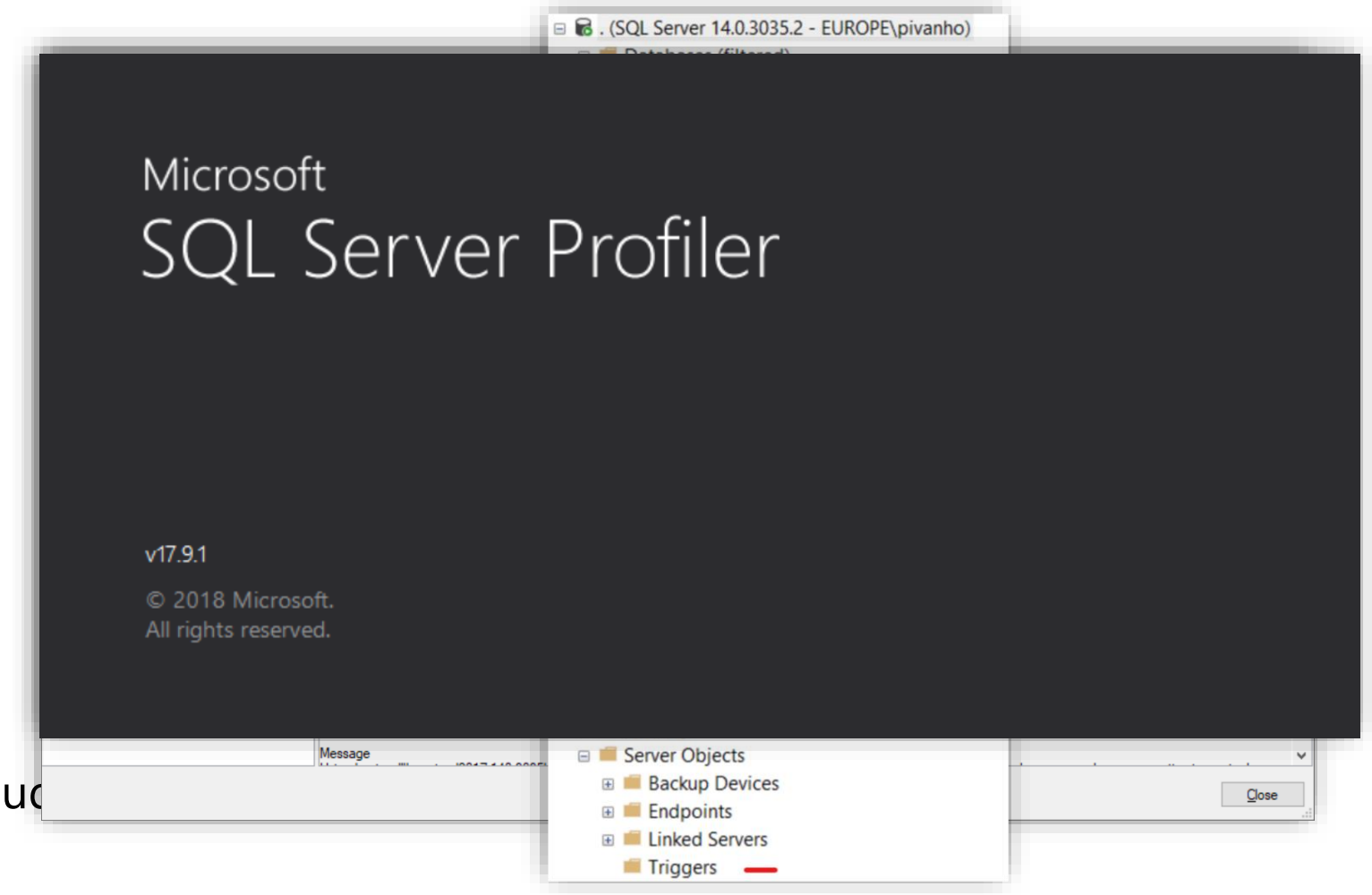
After completing this learning, you will be able to:

- Understand what SQL Server Audit is and how to configure it.



Some History...

- SQL 2005 and earlier
  - Server Level
  - Application Log
  - SQL Server Error Log
- Triggers
  - Login triggers
  - Server triggers
  - DDL triggers
- SQL Trace (Profiler)
  - Detailed activity audits
  - Individual statements, inclu



SQL Server Audit

Provides the ability to track and log events that occur in SQL Server engine

Built on Extended Events architecture

Basic Server Audit available in all SKUs (Since SQL Server 2012)

- Database Audit only available in Enterprise, Developer, and Evaluation editions

SQL Server Audit Components

- Allow pre-filtering and fine-grained auditing
- Allow multiple targets (File, Application and Security Event Logs)

Key part of security strategy

Who has accessed or attempted to access your data

Ability to detect unauthorized access attempts

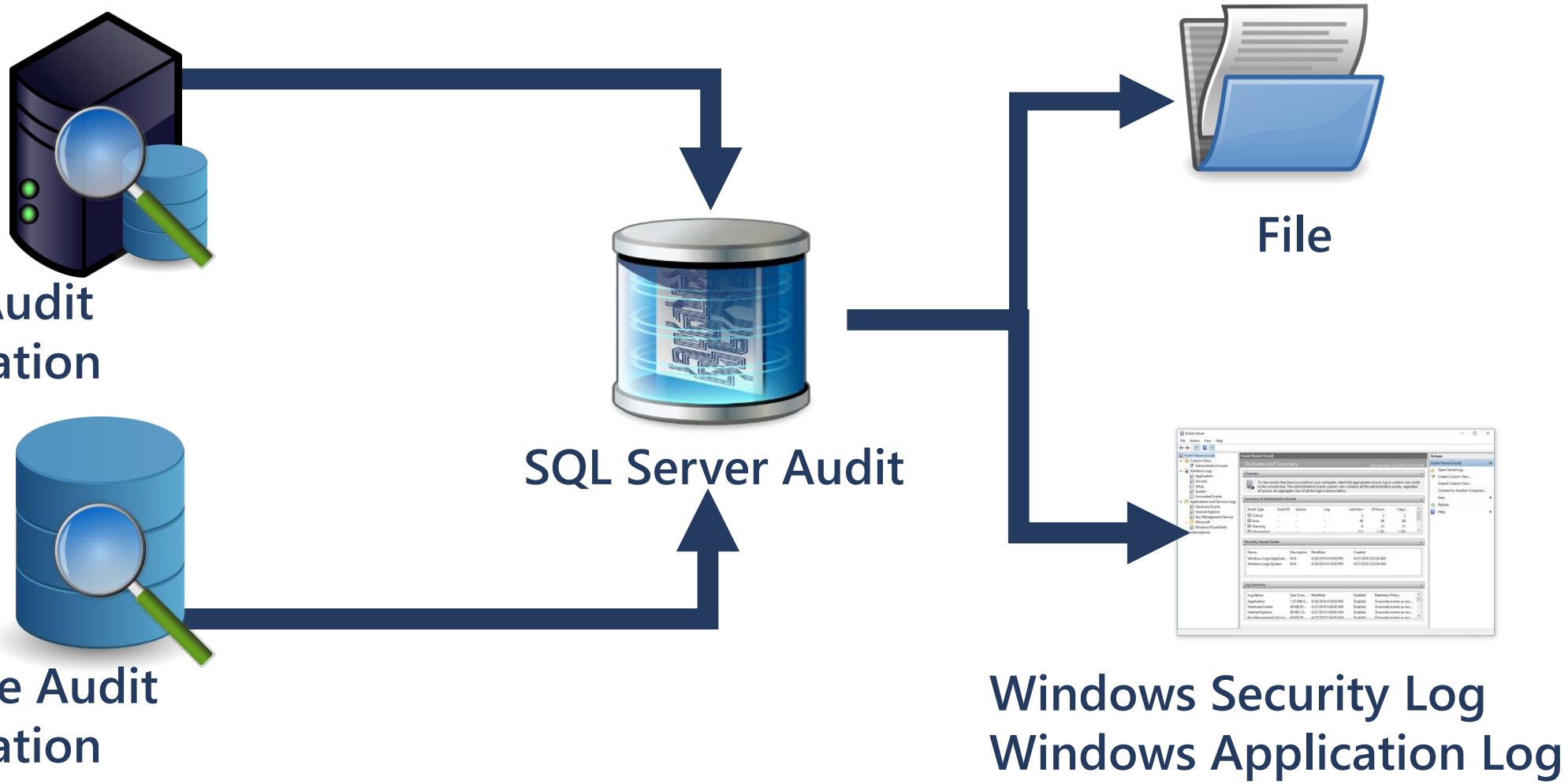
Piece together the actions of malicious insiders

Robust tracking capability

Primary Goals of SQL Server Audit

Security	• The audit feature must be truly secure.
Performance	• Performance impact must be minimized
Management	• The audit feature must be easy to manage.
Discoverability	• Audit-centric questions must be easy to answer

Audit Object Layout



Working with SQL Server Audit

Create an audit and define the target

Create either a server audit specification or database audit specification

Enable the audit specification

Enable the audit

Read the audit events

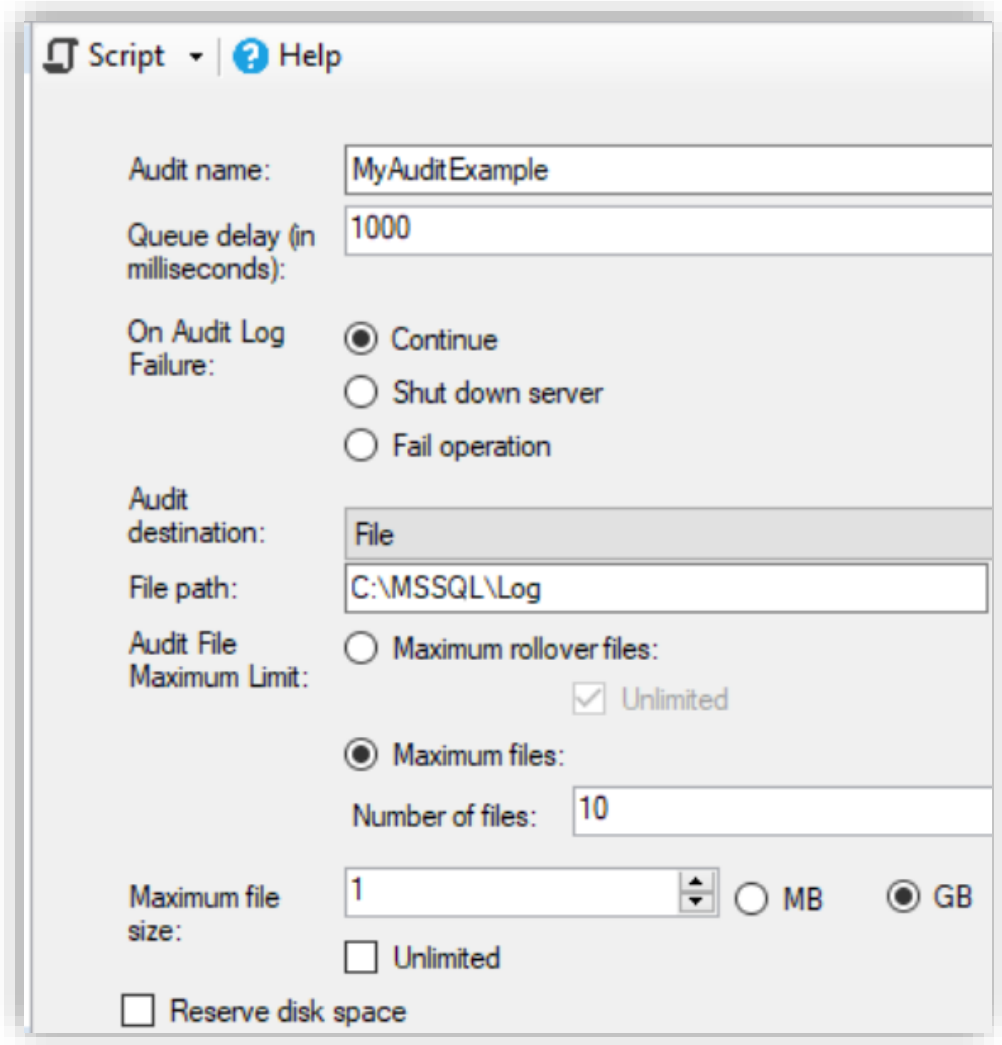
Create Audit

Queue delay (in milliseconds)

On Audit Log Failure - Continue

On Audit Log Failure - Shut down server

On Audit Log Failure - Fail operation



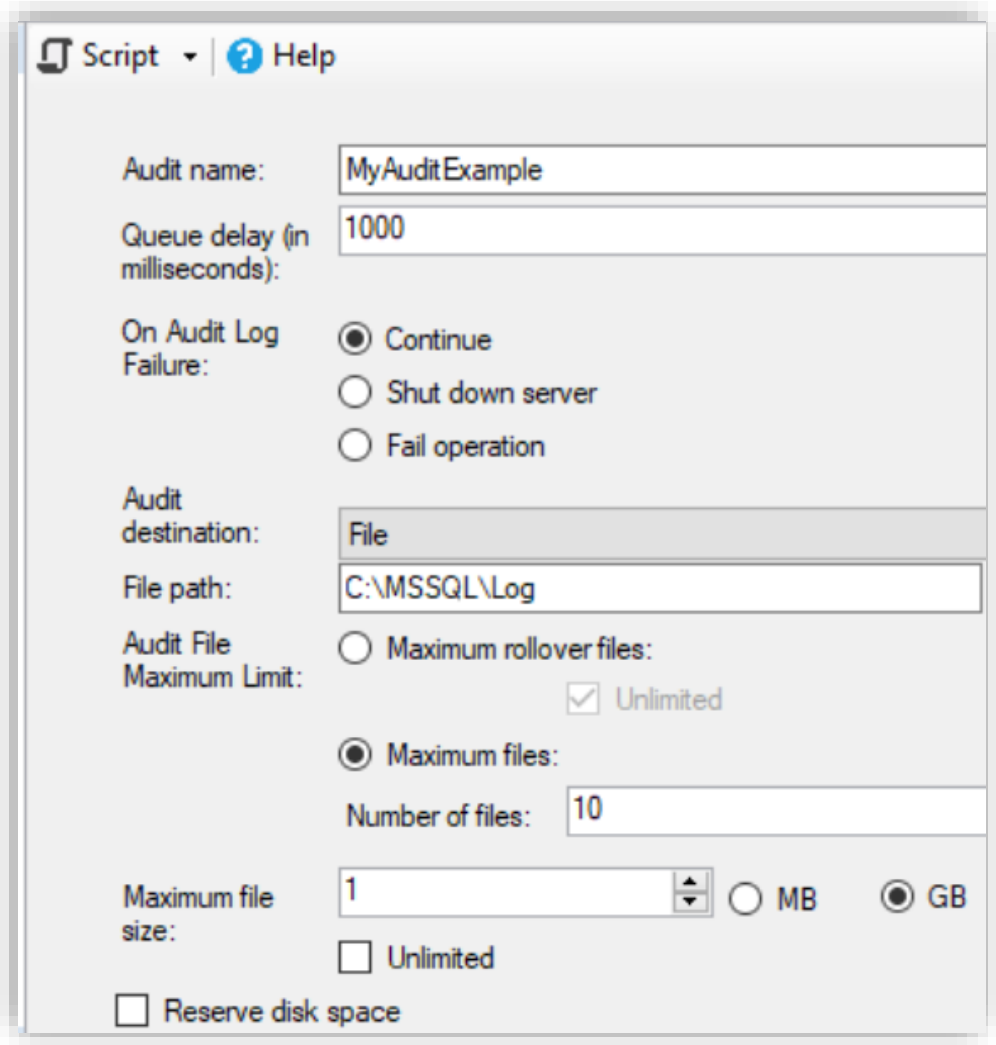
Create Audit (Continued)

Audit Destination

- Binary file
- Windows Application log
- Windows Security log

File Settings

- File Path
- Audit File Maximum Limit
- Maximum File Size
- Reserve disk space



SQL Server Audit Events to the Security Log

The Audit object

- The Audit object access setting must be configured to capture the events. The audit policy tool (auditpol.exe) exposes a variety of sub-policies settings in the audit object access category. To allow SQL Server to audit object access, configure the application generated setting.

SQL Server service Account

- The account that the SQL Server service is running under must have the generate security audits permission to write to the Windows Security log.
- secpol.msc → Generate security audits

Registry

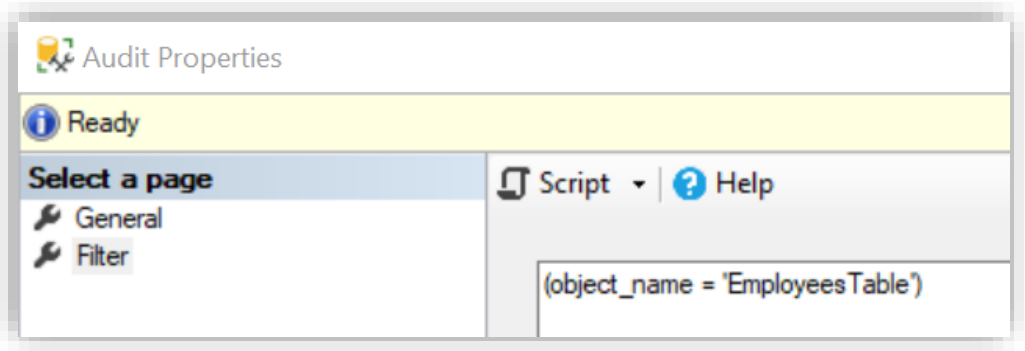
- Provide full permission for the SQL Server service account to the registry hive
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\EventLog\Security.

Create Audit Filter

Enter a predicate, or “WHERE clause”

Audit events are filtered before they are written to the audit log

You can filter on every element of the Audit Records



Server-Level Audit Action Groups

LOGIN_CHANGE_PASSWORD_GROUP

- Whenever a login password is changed

SERVER_OBJECT_CHANGE_GROUP

- CREATE, ALTER, or DROP operations on server objects

SERVER_PRINCIPAL_CHANGE_GROUP

- When server principals are created, altered, or dropped

SERVER_ROLE_MEMBER_CHANGE_GROUP

- Whenever a login is added or removed from a fixed server role.

SUCCESSFUL_LOGIN_GROUP

- A principal has successfully logged in to SQL Server

Database-Level Audit Action Groups

BACKUP_RESTORE_GROUP

- Whenever a backup or restore command is issued

DATABASE_CHANGE_GROUP

- When a database is created, altered, or dropped

DATABASE_OBJECT_CHANGE_GROUP

- When a CREATE, ALTER, or DROP statement is executed on database objects

DATABASE_ROLE_MEMBER_CHANGE_GROUP

- Whenever a login is added to or removed from a database role

DBCC_GROUP

- Whenever a principal issues any DBCC command

FAILED_DATABASE_AUTHENTICATION_GROUP

- A principal tried to log on to SQL Server and failed

Database Audit Specifications Actions and Groups

```
select *
from sys.dm_audit_actions
where class_desc = 'database'
or parent_class_desc = 'database';
```

action_id	name	class_desc	covering_action_name
R	REVOKE	DATABASE	DATABASE_PERMISSION_CHANGE_GROUP
D	DENY	DATABASE	DATABASE_PERMISSION_CHANGE_GROUP
G	GRANT	DATABASE	DATABASE_PERMISSION_CHANGE_GROUP
GWG	GRANT WITH GRANT	DATABASE	DATABASE_PERMISSION_CHANGE_GROUP
RWG	REVOKE WITH GRANT	DATABASE	DATABASE_PERMISSION_CHANGE_GROUP
RWC	REVOKE WITH CASCADE	DATABASE	DATABASE_PERMISSION_CHANGE_GROUP
DWC	DENY WITH CASCADE	DATABASE	DATABASE_PERMISSION_CHANGE_GROUP
R	REVOKE	OBJECT	NULL
D	DENY	OBJECT	NULL
G	GRANT	OBJECT	NULL
GWG	GRANT WITH GRANT	OBJECT	NULL
RWG	REVOKE WITH GRANT	OBJECT	NULL
RWC	REVOKE WITH CASCADE	OBJECT	NULL
DWC	DENY WITH CASCADE	OBJECT	NULL
R	REVOKE	TYPE	NULL
D	DENY	TYPE	NULL
G	GRANT	TYPE	NULL
GWG	GRANT WITH GRANT	TYPE	NULL
RWG	REVOKE WITH GRANT	TYPE	NULL
RWC	REVOKE WITH CASCADE	TYPE	NULL
DWC	DENY WITH CASCADE	TYPE	NULL
R	REVOKE	SCHEMA	NULL
D	DENY	SCHEMA	NULL

List All Server and Database Action Groups

```
select name,class_desc from
sys.dm_audit_actions
where name in
(select containing_group_name
from sys.dm_audit_actions) order
by class_desc, name;
```

name	class_desc
DBCC_GROUP	DATABASE
FAILED_DATABASE_AUTHENTICATION_GROUP	DATABASE
SCHEMA_OBJECT_ACCESS_GROUP	DATABASE
SCHEMA_OBJECT_CHANGE_GROUP	DATABASE
SCHEMA_OBJECT_OWNERSHIP_CHANGE_GROUP	DATABASE
SCHEMA_OBJECT_PERMISSION_CHANGE_GROUP	DATABASE
SUCCESSFUL_DATABASE_AUTHENTICATION_GROUP	DATABASE
USER_CHANGE_PASSWORD_GROUP	DATABASE
USER_DEFINED_AUDIT_GROUP	DATABASE
APPLICATION_ROLE_CHANGE_PASSWORD_GROUP	SERVER
AUDIT_CHANGE_GROUP	SERVER
BACKUP_RESTORE_GROUP	SERVER
BROKER_LOGIN_GROUP	SERVER
DATABASE_CHANGE_GROUP	SERVER
DATABASE_LOGOUT_GROUP	SERVER
DATABASE_MIRRORING_LOGIN_GROUP	SERVER
DATABASE_OBJECT_ACCESS_GROUP	SERVER
DATABASE_OBJECT_CHANGE_GROUP	SERVER
DATABASE_OBJECT_OWNERSHIP_CHANGE_GROUP	SERVER
DATABASE_OBJECT_PERMISSION_CHANGE_GROUP	SERVER

Get Information About a Particular Group Name

```
select *
from sys.dm_audit_actions
where containing_group_name = 'USER_CHANGE_PASSWORD_GROUP';
```

action_id	name	class_desc	covering_action_name	parent_class_desc	covering_parent_action_name	configuration_level
PWR	RESET PASSWORD	USER	NULL	DATABASE	USER_CHANGE_PASSWORD_GROUP	NULL
PWRS	RESET OWN PASSWORD	USER	NULL	DATABASE	USER_CHANGE_PASSWORD_GROUP	NULL
PWCS	CHANGE OWN PASSWORD	USER	NULL	DATABASE	USER_CHANGE_PASSWORD_GROUP	NULL
PWC	CHANGE PASSWORD	USER	NULL	DATABASE	USER_CHANGE_PASSWORD_GROUP	NULL
USTC	COPY PASSWORD	USER	NULL	DATABASE	USER_CHANGE_PASSWORD_GROUP	NULL
UCGP	USER_CHANGE_PASSWORD_GROUP	DATABASE	NULL	SERVER	USER_CHANGE_PASSWORD_GROUP	Group
UCGP	USER_CHANGE_PASSWORD_GROUP	SERVER	NULL	NULL	NULL	Group

Database-Level Audit Actions

SELECT

UPDATE

INSERT

DELETE

EXECUTE

RECEIVE

REFERENCES

View a SQL Server Audit Log

SQL SERVER MANAGEMENT STUDIO

SYS.FN_GET_AUDIT_FILE

sys.fn_get_audit_file

- **file_pattern**
  - Specifies the directory or path and file name for the audit file set to be read.
- **initial_file_name**
  - Specifies the path and name of a specific file in the audit file set to start reading audit records from
- **audit_record_offset**
  - Specifies a known location with the file specified for the initial_file_name

```
SELECT * FROM sys.fn_get_audit_file
('\\serverName\Audit\HIPAA_AUDIT.sqlaudit',default,default);
```

Considerations

In the case of a failure during audit initiation, the server will not start.

Attaching a Database with an Audit Defined

Always On Availability Groups and SQL Server Audit

Auditing Administrators

Demonstration

Demonstrate how to Create an Audit and Audit Specification within SQL Server

Questions?

SQL Server Auditing

- Exercise 1: Create a Login with PowerShell
- Exercise 2: Create a Server audit and read its results

Knowledge Check

How do you start an audit after creating the server audit specification login?

How do you stop the audit?

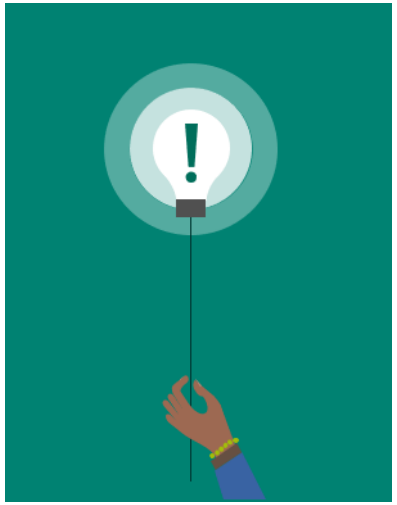
What are audit action groups in a server audit specification?

Lesson 5: Ledger for SQL Server

Objectives

After completing this learning, you will be able to:

- Understand what Ledger for SQL Server is and how to configure it.



Security enhancements - Ledger for SQL Server

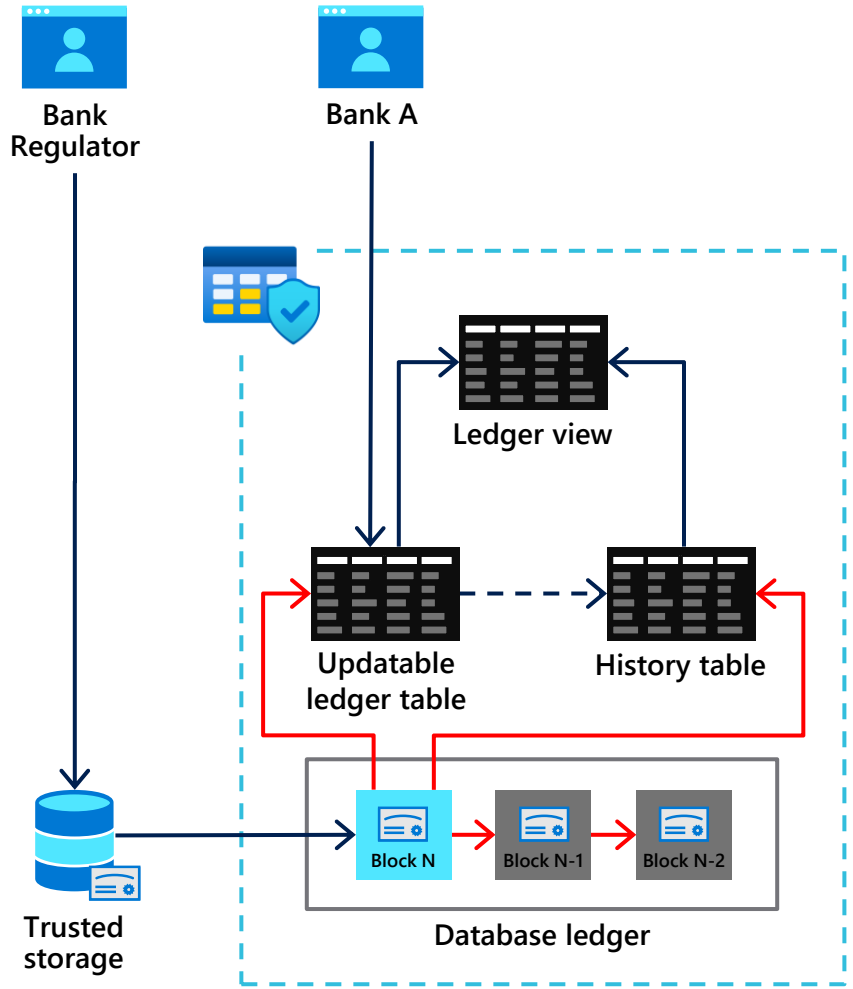
Ledger for SQL Server -The ledger feature provides tamper-evidence capabilities in your database. You can cryptographically attest to other parties, such as auditors or other business parties, that your data hasn't been tampered with.

Ledger for SQL Server

Tamper-evidence track record of data over time

Challenge: I want the power of blockchain in a centralized system like SQL Server

- Use a cryptographically hashed ledger detect tampering by malicious actors
- Built into SQL Server with T-SQL
- Establish digital trust in a centralized system using blockchain technology.
- Attest to other parties that data integrity has not been compromised
- Automatic digest storage



Ledger Tables – Updatable and Append-Only

Updatable Ledger Tables are standard SQL tables which allow updates and deletes

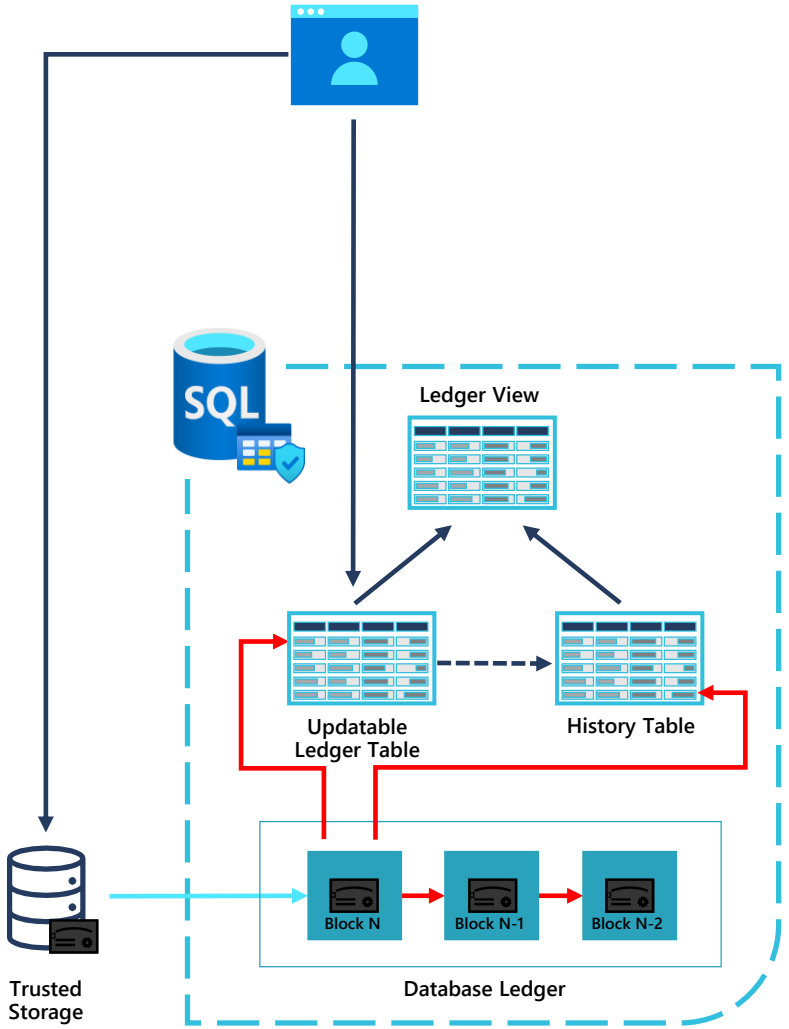
The history of rows that have been updated or deleted are preserved in the history table and easy-to-query Ledger View

Integrity of the updatable and history tables are maintained through cryptographic links from the Database Ledger

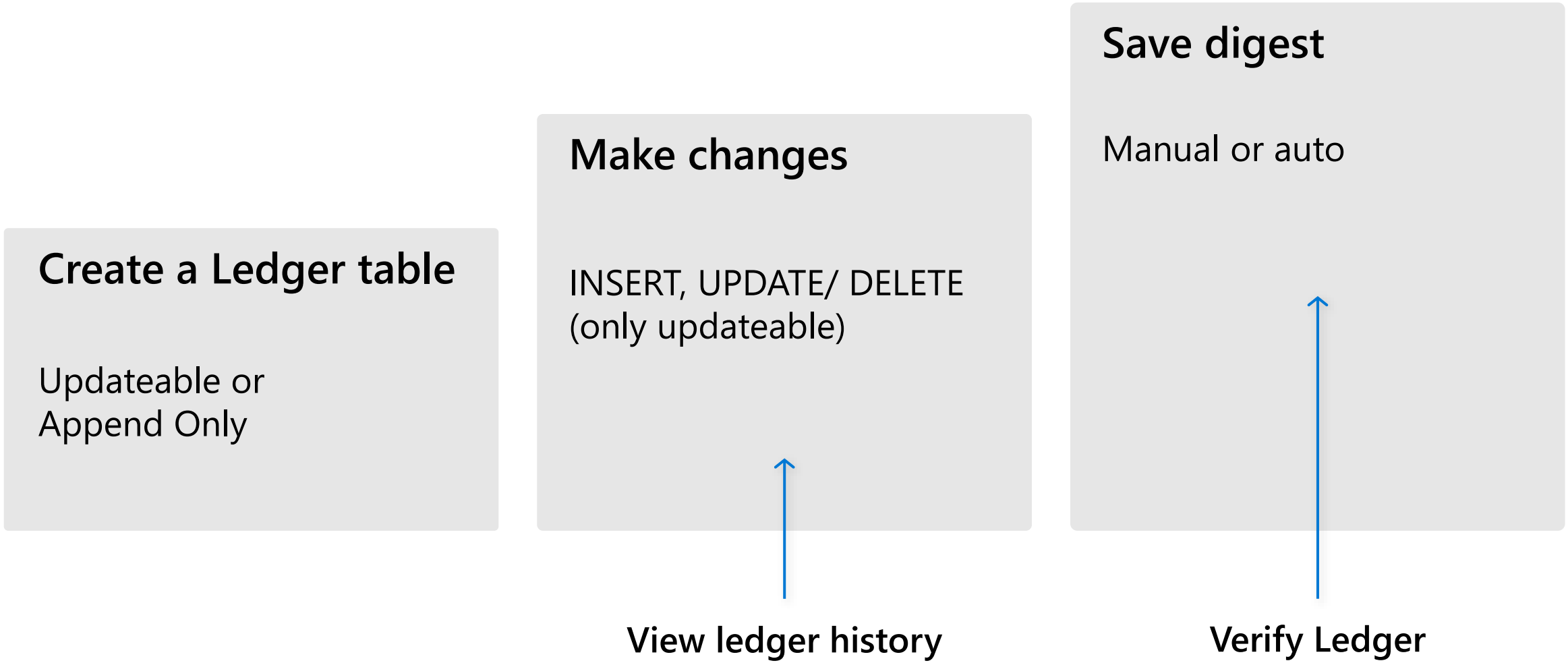
System periodically uploads digital receipts to a customer-configured trusted storage service

Customer can use digital receipts to verify the integrity of data in Ledger tables

Append-Only Ledger Tables block UPDATE/DELETE at the API and remove the need for a history table



How to use Ledger for SQL Server



Creating an Account Balance Updatable Ledger Table

```
CREATE SCHEMA [Account];
GO
CREATE TABLE [Account].[Balance]
([CustomerID] INT NOT NULL PRIMARY KEY CLUSTERED,
    [LastName] VARCHAR (50) NOT NULL,
    [FirstName] VARCHAR (50) NOT NULL,
    [Balance] DECIMAL (10,2) NOT NULL)
WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Account].[BalanceHistory]),
 LEDGER = ON);
```

	ledger_table_name	history_table_name	ledger_view_name
1	Account.Balance	Account.MSSQL_LedgerHistoryFor_1525580473	Account.Balance_Ledger

Viewing the Account Balance Updatable Ledger Table

```
SELECT ts.[name] + '.' + t.[name] AS [ledger_table_name]
, hs.[name] + '.' + h.[name] AS [history_table_name]
, vs.[name] + '.' + v.[name] AS [ledger_view_name]
FROM sys.tables AS t
JOIN sys.tables AS h ON (h.[object_id] = t.[history_table_id])
JOIN sys.views v ON (v.[object_id] = t.[ledger_view_id])
JOIN sys.schemas ts ON (ts.[schema_id] = t.[schema_id])
JOIN sys.schemas hs ON (hs.[schema_id] = h.[schema_id])
JOIN sys.schemas vs ON (vs.[schema_id] = v.[schema_id])
WHERE t.[name] = 'Balance';
```

	ledger_table_name	history_table_name	ledger_view_name
1	Account.Balance	Account.MSSQL_LedgerHistoryFor_1525580473	Account.Balance_Ledger

Add 4 Accounts In 2 Separate Transactions

Tx1: Add Nick with an opening balance of $50

Tx2: Add John, Joe and Mary

1. Each transaction has it's own unique transaction ID
2. Tx2 modified 3 rows, each tracked with a ledger sequence number

Updatable ledger table

Results Messages

	CustomerID	LastName	FirstName	Balance	ledger_start_transaction_id (1)	ledger_end_transaction_id	ledger_start_sequence_number (2)	ledger_end_sequence_number
1	1	Jones	Nick	50.00	999	*NULL*	0	*NULL*
2	2	Smith	John	500.00	1002	*NULL*	0	*NULL*
3	3	Smith	Joe	30.00	1002	*NULL*	1	*NULL*
4	4	Michaels	Mary	200.00	1002	*NULL*	2	*NULL*

Update Nick's Balance From $50 To $100

Applies to: Azure SQL Database, Managed Instance preview

Updatable ledger table – Nick's balance is now $100

	CustomerID	LastName	FirstName	Balance	ledger_start_transaction_id	ledger_end_transaction_id	ledger_start_sequence_number	ledger_end_sequence_number
1	1	Jones	Nick	100.00	1055	*NULL*	0	*NULL*
2	2	Smith	John	500.00	1002	*NULL*	0	*NULL*
3	3	Smith	Joe	30.00	1002	*NULL*	1	*NULL*
4	4	Michaels	Mary	200.00	1002	*NULL*	2	*NULL*

History Table – Shows the historical value of row containing Nick's opening balance

	CustomerID	LastName	FirstName	Balance	ledger_start_transaction_id	ledger_end_transaction_id	ledger_start_sequence_number	ledger_end_sequence_number
1	1	Jones	Nick	50.00	999	1055	0	1

Ledger View – Shows Nick's update as a delete followed but a subsequent insert

	CustomerID	LastName	FirstName	Balance	ledger_transaction_id	ledger_sequence_number	ledger_operation_type_id	ledger_operation_type_desc
1	1	Jones	Nick	50.00	999	0	1	INSERT
2	2	Smith	John	500.00	1002	0	1	INSERT
3	3	Smith	Joe	30.00	1002	1	1	INSERT
4	4	Michaels	Mary	200.00	1002	2	1	INSERT
5	1	Jones	Nick	50.00	1055	1	2	DELETE
6	1	Jones	Nick	100.00	1055	0	1	INSERT

Ledger Views

Applies to: Azure SQL Database, Managed Instance preview

Sys.database_ledger_transactions - Records the table hashes for each transaction in the database as well as the user who issued the transaction

	transaction_id	block_id	transaction_ordinal	commit_time	principal_name	table_hashes
1	999	0	0	2021-03-23 20:18:08.2700000	janders	0xB982EE5A88DFE8EF08BE7564D62273BD17306231C8E22E052644805…
2	1002	0	1	2021-03-23 20:18:12.9300000	janders	0xB982EE5AB931133CF9B8E6FCD06C9AF25C0F0C6A9A91A12C89A84AB…
3	1055	0	2	2021-03-23 20:40:08.9500000	janders	0xB982EE5A38F20FA9D8ABFC3C3523284FE65466DAA9E91166447648B…
4	1091	0	3	2021-03-23 21:36:22.2533333	janders	0x9D13BF5E345245E7456EC748BC895E0E1323379BD04EBC35638D91E…

Sys.database_ledger_blocks – Records the hash of each block created in the database, along with the # of transactions in the block

	block_id	transactions_root_hash	block_size	previous_block_hash
1	0	0x8F3C4C8ADF99EAEE24A783CB1AC282A12E9C9ECA619DDE19B2C98D8ECCA5E4A5	4	*NULL*

Ledger FAQ

How is this different from a temporal table?

- Built-in transaction auditing
- Append-only
- Database Ledger and digest for tamper evidence

How is this different than SQL Server Audit?

- Transaction history and audit built into database
- Digest verification

Can I disable ledger from a table?

- You cannot ALTER a table to “turn off” ledger
- You cannot turn off system versioning

Can I drop a ledger table?

- Yes, but a history of the dropped table and ledger is kept

Ledger FAQ

How often do I need to save the digest?

- As frequent as you need to ensure the ledger is tamper proof

Does Ledger require more space?

- Updateable requires similar extra space as temporal
- **Plus** database ledger requires some minimal extra space for hashes and blockchain
- Digests small and separate but you may need a long history
- You can’t archive or truncate ledger tables and database ledger

Any performance impact?

- Append only should see minimal impact
- Updateable would have similar impact as temporal

Demonstration

Create and Update a Ledger Table