Erin Dempster (she/her)
Sr. SQL Developer, Trean Corporation

PASS SQLSATURDAY

Navigating Row Level Security

1

Agenda

- About Me
- Row Level Security Overview
- Components
- Demo
- Wrap-up

2

About Me

- Over 15 years of SQL Server experience
  - MCDBA on SQL Server 2000 (earned in 2004)
  - MCSE: Data Management and Analytics (2017 – 2019)
- Database Developer (Applications and BI)
- Database Administrator

3

Row Level Security

5

What is Row Level Security?

"Row-Level Security enables you to use group membership or execution context to control access to rows in a database table."

https://docs.microsoft.com/en-us/sql/relational-databases/security/row-level-security?view=sql-server-ver15

6

Types of Access Control

- Filtering
  - Applies to SELECT, UPDATE and DELETE statements
  - Filters silently
- Blocking*
  - Applies to INSERT, UPDATE and DELETE statements
  - Returns an error message

* Not available in Azure Synapse Analytics (formerly Azure SQL Data Warehouse)

7

Blocking Conflict Message

```
INSERT INTO ods.Policies (ProgramID, PolicyNumber, InsuredName)
    VALUES (1, 'PG1-001-01', 'ACME Transportation')
```

```
Msg 33504, Level 16, State 1, Line 2

The attempted operation failed because the target object
'TreanDemo.ods.Policies' has a block predicate that conflicts with this operation.
If the operation is performed on a view, the block predicate might be enforced
on the underlying table. Modify the operation to target only the rows that are
allowed by the block predicate.

The statement has been terminated.
```

8

User Context

- Implement schema to support
  - Single-user accessibility
  - Role-based accessibility
- Applied to ALL queries for ALL users

9

Components

- 1 or more tables
- User-defined function (predicate function)
- Security Policy binds predicate to table
  - Filtering and Blocking at the same time
  - Only 1 function bound to a single table

10

Security Policy

Binds the table and predicate function

```
CREATE SECURITY POLICY <Schema>.<Policy Name>
ADD <BLOCK | FILTER> PREDICATE
    <Schema>.<UDF Name> (<Field Name(s)>)
ON <Schema>.<Table>
WITH
(STATE = ON, SCHEMABINDING = [ON | OFF]);
```

https://docs.microsoft.com/en-us/sql/t-sql/statements/create-security-policy-transact-sql?view=sql-server-ver15

11

Predicate Function

- Inline Table-Valued Function
- Returns a single value for success

12

Predicate Function Wisdom

"The return value is typically shown as a 1, with some column name, though the important item to remember is that as long as something is returned, the user has access to the row data. Note that the result does not need to return a 1."

- Steve Jones, SQL Server Central

https://www.sqlservercentral.com/steps/row-level-security-predicate-functions-level-2-of-the-stairway-to-row-level-security

13

Sample Predicate Function

```
CREATE FUNCTION
    Security.fn_securitypredicate(@SalesRep AS sysname)
RETURNS TABLE
WITH SCHEMABINDING
AS
RETURN

SELECT 1 AS fn_securitypredicate_result
WHERE @SalesRep = USER_NAME() OR USER_NAME() = 'Manager';
```

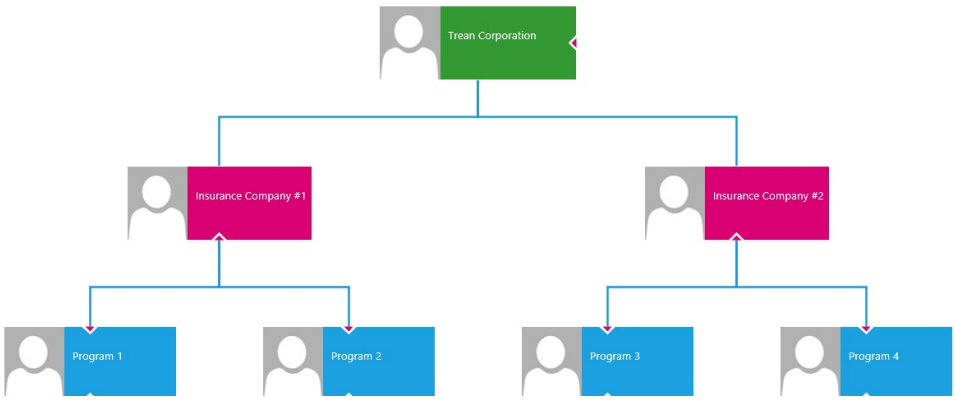
15

Scalability Issues

- What about other departments?
- What to do with multiple companies?

16

About Trean



Over 40 programs and growing!!

17

Limiting Data Access

- Trean Staff – All Data
- Insurance Companies – Data for their programs
- Programs – Only their data

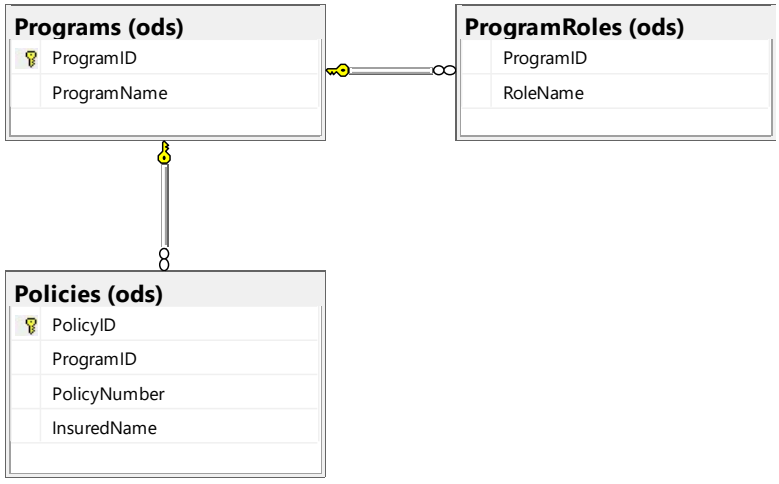
18

Scalability Options

- Find a higher level of granularity
- Assign Active Directory groups and DB Roles
- Use IS_MEMBER() function

19

Demo Tables



21

Predicate Function #1

```
CREATE FUNCTION [ods].[fn_FilterPrograms]
(
	@ProgramID INT
)
RETURNS TABLE
WITH SCHEMABINDING
AS
RETURN
(
	SELECT IS_MEMBER(RoleName) ReturnValue
	FROM ods.ProgramRoles
	WHERE ProgramID = @ProgramID
);
```

Bad – IS_MEMBER() returns 1 of 3 possible values: TRUE, FALSE and NULL

Good – Filtering on ProgramID

22

Predicate Function #2

```
CREATE FUNCTION [ods].[fn_FilterPrograms]
(
	@ProgramID INT
)
RETURNS TABLE
WITH SCHEMABINDING
AS
RETURN
(
	SELECT 1 ReturnValue
	FROM ods.ProgramRoles
	WHERE ProgramID = @ProgramID
		AND IS_MEMBER(RoleName) = 1
);
```

Good – Explicitly defined value

Good – Filtering ProgramID AND Role Membership

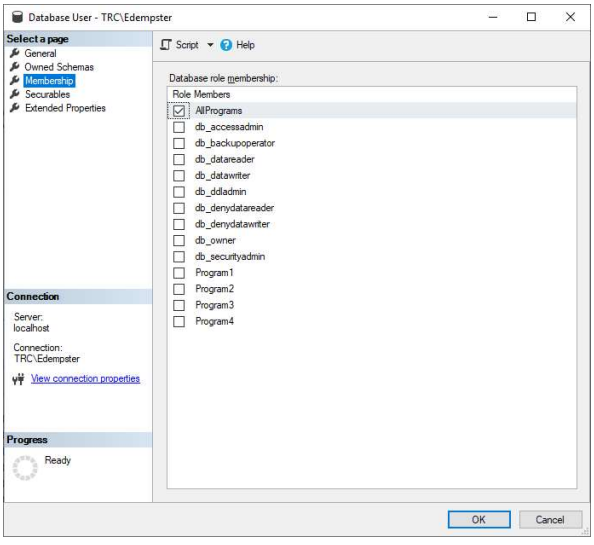
23

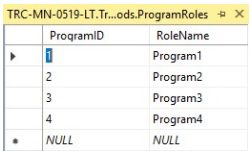
Testing Predicate Function

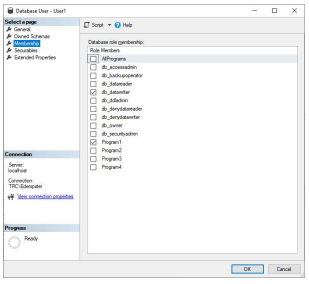
```
SELECT PolicyNumber, InsuredName
FROM ODS.Policies p
OUTER APPLY ODS.fn_FilterPrograms(p.ProgramID) f

SELECT PolicyNumber, InsuredName
FROM ODS.Policies p
CROSS APPLY ODS.fn_FilterPrograms(p.ProgramID) f
```

24

Demo Information

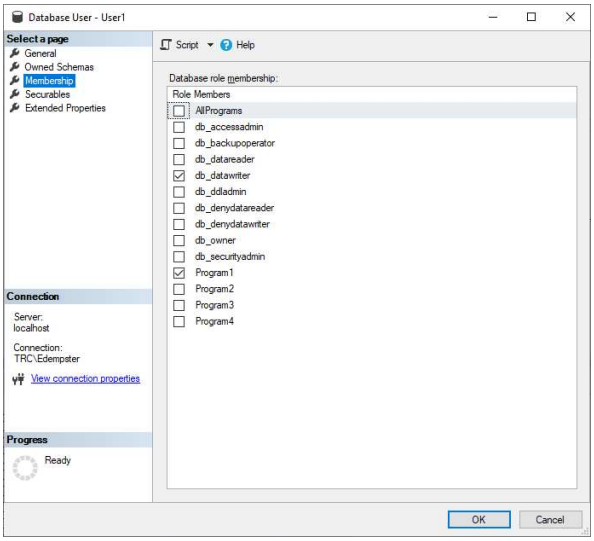


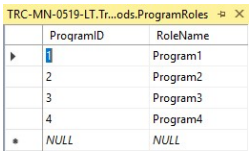


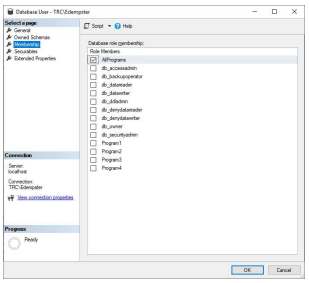


27

Demo Information

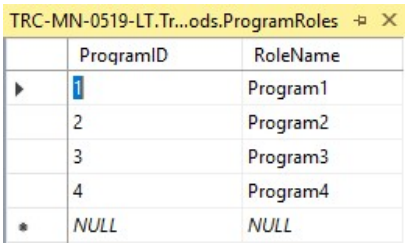


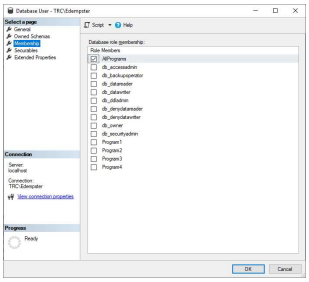


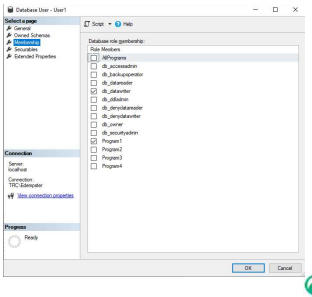


28

Demo Information







29

30

System Considerations

- SQL Server 2016 and later
- Azure SQL Database & Azure Synapse Analytics*

- Included in all editions

31

Minimum Security Requirements

Database

- ALTER ANY SECURITY POLICY
- EXECUTE (predicate functions)
- REFERENCES (predicate functions)
- SELECT (tables involved in RLS)

Schema

- ALTER

Security Policy

- ALTER

32

Wrapping up

- Keep predicate logic quick and simple
- Single table supports 1 security policy
- Use Roles for enterprise DBs
- Predicate applied, regardless of user

33


Thank you
Blog www.endlessreporting.com
Twitter @em_dempster

35