1

PASS SQLSATURDAY

Microsoft Azure CLOUDERA

Thank you to our St. Louis Sponsors

Covenant TECHNOLOGY PARTNERS Quest

SBS Technology Consultants Daugherty BUSINESS SOLUTIONS

redgate SentryOne

2

Agenda

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

3

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

4

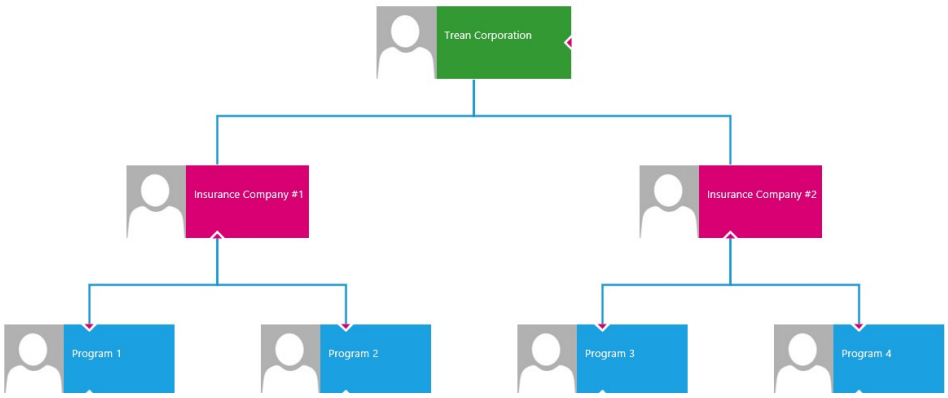
Current Role

Senior SQL Developer

Trean Corporation, Wayzata, MN

5

About Trean



Over 40 programs and growing!!

6

Limiting Data Access

Trean Staff – All Data

Insurance Companies – Data for their programs

Programs – Only their data

7

Other Requirements

- All queries **must** return filtered data
- Only 1 copy of tables allowed

8

How do we get there?

9

Row Level Security

10

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

11

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

12

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.
```

13

User Context

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

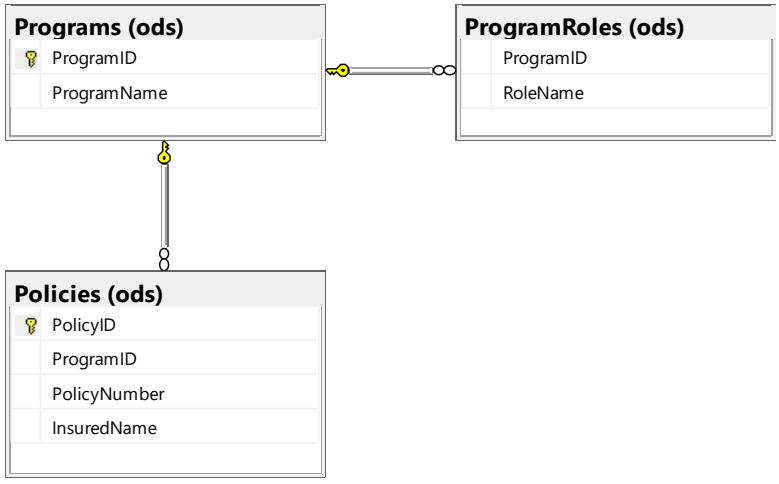
14

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

15

Sample Database



16

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

17

Predicate Function

- Inline Table-Valued Function
- Needs to return a single value

18

Predicate Function Template

```
CREATE FUNCTION <Some Name>
(
      <Some Parameters>
)
RETURNS TABLE
[WITH SCHEMABINDING]
AS
RETURN
(
      <Single SELECT Statement>
      <doing some light, easy (QUICK) work>
      <Returning 1 value>
);
```

19

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

20

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 – Defined explicit value

Good – Filtering ProgramID AND Role Membership

21

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

22

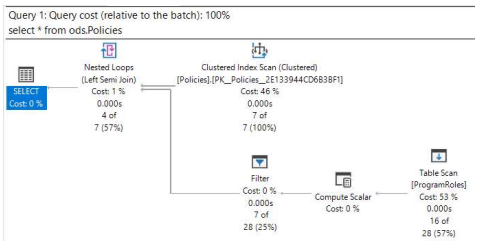
Query Filtering Behavior

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

23

Query Filtering Behavior (SQL Server 2017)

With Row Level Security Enabled



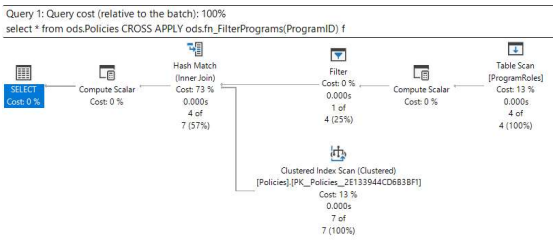

```
EXECUTE AS USER='User1'
select * from ods.Policies
```

100 %

Results | Messages | Execution plan

	PolicyID	ProgramID	PolicyNumber	InsuredName
1	1	1	PG1-0001-01	ACME Bombs
2	3	1	PG1-0002-01	ACME Anvils
3	4	1	PG1-0003-01	ACME Security
4	21	1	PG1-500-002	Bugs Bunny

Explicit use of Predicate Function w/Cross Apply




```
EXECUTE AS USER='User1'
select * from ods.Policies
CROSS APPLY ods.fn_FilterPrograms(ProgramID) f
```

100 %

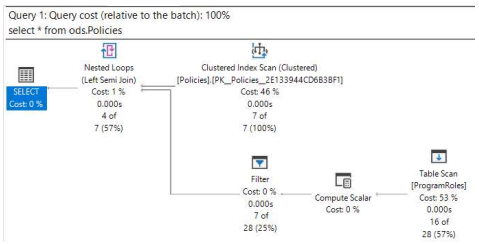
Results | Messages | Execution plan

	PolicyID	ProgramID	PolicyNumber	InsuredName	Filter
1	1	1	PG1-0001-01	ACME Bombs	1
2	3	1	PG1-0002-01	ACME Anvils	1
3	4	1	PG1-0003-01	ACME Security	1
4	21	1	PG1-500-002	Bugs Bunny	1

24

Query Filtering Behavior (SQL Server 2017)

With Row Level Security Enabled

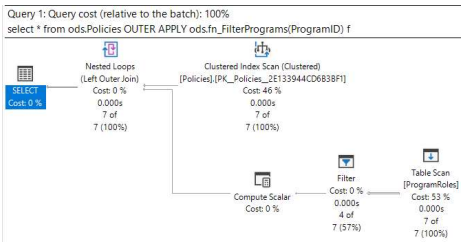



```
EXECUTE AS USER='User1'
select * from ods.Policies
```

100 % — Results | Messages | Execution plan

	PolicyID	ProgramID	PolicyNumber	InsuredName
1	1	1	PG1-0001-01	ACME Bombs
2	3	1	PG1-0002-01	ACME Anvils
3	4	1	PG1-0003-01	ACME Security
4	21	1	PG1-500-002	Bugs Bunny

Explicit use of Predicate Function w/Outer Apply




```
EXECUTE AS USER='User1'
select * from ods.Policies
OUTER APPLY ods.fn_FilterPrograms(ProgramID) f
```

100 % — Results | Messages | Execution plan

	PolicyID	ProgramID	PolicyNumber	InsuredName	Filter
1	1	1	PG1-0001-01	ACME Bombs	1
2	2	2	PG2-0010-01	Roady's Bird Seed	NULL
3	3	1	PG1-0002-01	ACME Anvils	1
4	4	1	PG1-0003-01	ACME Security	1
5	5	3	PG3-1000-01	Association of Animated Singers	NULL
6	6	4	PG4-0500-01	Sylvester del Gato	NULL
7	21	1	PG1-500-002	Bugs Bunny	1

25

Demo

26

System Considerations

- SQL Server 2016 and later
- Azure SQL Database

- Included in all editions

27

Minimum Security Requirements

Database

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

Schema

ALTER

Security Policy

ALTER

28

Wrapping up

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

29

Thank you

Blog www.endlessreporting.com
Twitter @em_dempster

31