Programming SQL Server Database Triggers and Functions

WORKING SMARTER WITH TRIGGERS

Ryan Booz
AUTHOR AND SPEAKER

@ryanbooz https://www.softwareandbooz.com

Overview

Security and Execution Context

MERGE statements

Hidden or transparent work

Overusing Triggers

Asynchronous work with Service Broker

Trigger Security

Intentional Trigger Security

LOGON Trigger Anti-Patterns

Execution Context

LOGON Trigger Anti-Patterns

Do not validate against data that can be spoofed

Specifically Host Name and Application Name

Both are easily modified through applications and connection strings

```
CREATE TRIGGER ON ALL SERVER
FOR LOGON
AS
BEGIN
     IF (APP_NAME() NOT IN (‘TrustedApp1’,’TrustedApp2’))
     BEGIN
          RAISERROR(‘Application not allowed to login.', 16, 1);
          ROLLBACK;
     END
END
```

Application Name Spoofing

Connection string spoofing will bypass this LOGON Trigger constraint

Append Application Name, ApplicationName, or App Name to connection string

```
Data Source=server;Initial Catalog=master;Integrated Security=True;AppName=TrustedApp1
```

Trigger Execution Context

Triggers are executed with the permissions of the CALLER by default

A well-crafted trigger could grant access to objects through a completely different DML or DDL event

WITH EXECUTE AS

```
CREATE OR ALTER TRIGGER [Trigger Name]
ON {ALL SERVER | DATABASE } WITH EXECUTE AS [Username]
FOR { event_type | event_group }
EXECUTE AS
AS

   /*

      Insert business logic here…

   */

GO;
```

Execution Context

CALLER (Default)

The user that is currently executing the DML/DDL statement

CALLER must have permissions and access to everything in the trigger

SELF

The user that created or modified the Trigger.

Prefer ‘user_name’ instead

SELF leaves it unclear, especially when viewing generated SCHEMA

Execution Context

OWNER

The owner of the Trigger that is being executed

Only permissible with DML Triggers and most Functions

user_name/login_name

Specify the username to impersonate when executing the Trigger

Schema generation tools will display this value

Much clearer than SELF when debugging problems

Be thoughtful and explicit with permissions

Always use the principle of least privilege with user permissions

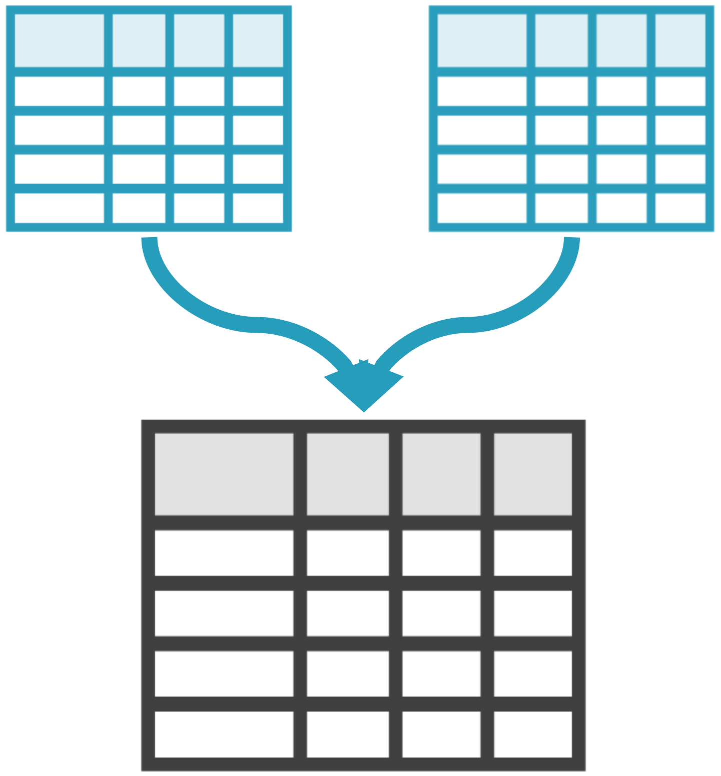
Triggers are generally transparent to most users and developers

Make a habit of checking 'sys.triggers' and 'sys.system_triggers’.

Track changes and ask questions if necessary

The Problem with MERGE and Triggers

MERGE



Introduced in SQL Server 2008

One statement that performs INSERT, UPDATE, and DELETE

Otherwise known as an “UPSERT” in other technologies

T-SQL MERGE

```
MERGE target_table T1
   USING source_table S1

ON (T1.id = S1.fkid)

WHEN MATCHED
    THEN UPDATE SET T1.name = S1.name

WHEN NOT MATCHED
    THEN INSERT (name,code) VALUES (S1.name, S1.code)

WHEN NOT MATCHED BY SOURCE
    THEN DELETE;
```

MERGE Misinformation

```
CREATE OR ALTER TRIGGER [Sales].[TI_OrderLines]
ON [Sales].[OrderLines] AFTER INSERT
AS
BEGIN

    IF (ROWCOUNT_BIG() = 0)
        RETURN;

    -- Do not print any result details from Trigger
    SET NOCOUNT ON;

    IF NOT EXISTS (SELECT 1 FROM INSERTED)
        RETURN;

    /* Trigger Logic */

END;
```

MERGE doesn’t always work as expected

Many SQL Server Pros advise developers to avoid MERGE

Bypassing Transactions In Triggers

Revisiting Transaction Basics

The Trigger is subject to the calling transaction

A ROLLBACK affects everything in the current transaction scope

Prevents audit logging in normal workflow

This can feel like it defeats the purpose of attempting the audit log

Tables Variables to the Rescue

Table variables provide a means of bypassing the Trigger transaction scope

They are scoped to the module they are declared in, not the overall transaction

We can use this to our advantage in a special case like this

Table Variable Misconceptions

They aren't inherently better or worse…

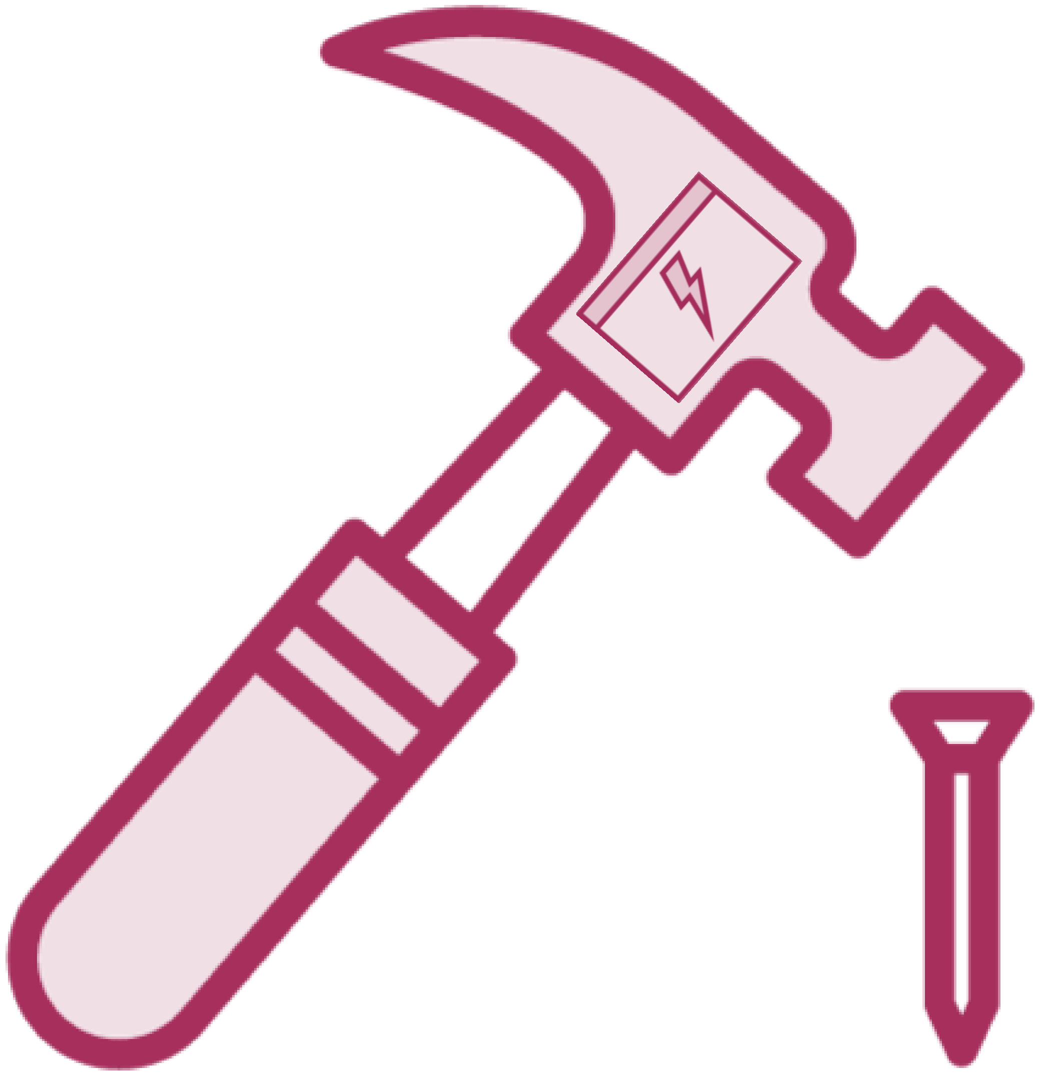
They are **not memory only** and can still spool to disk

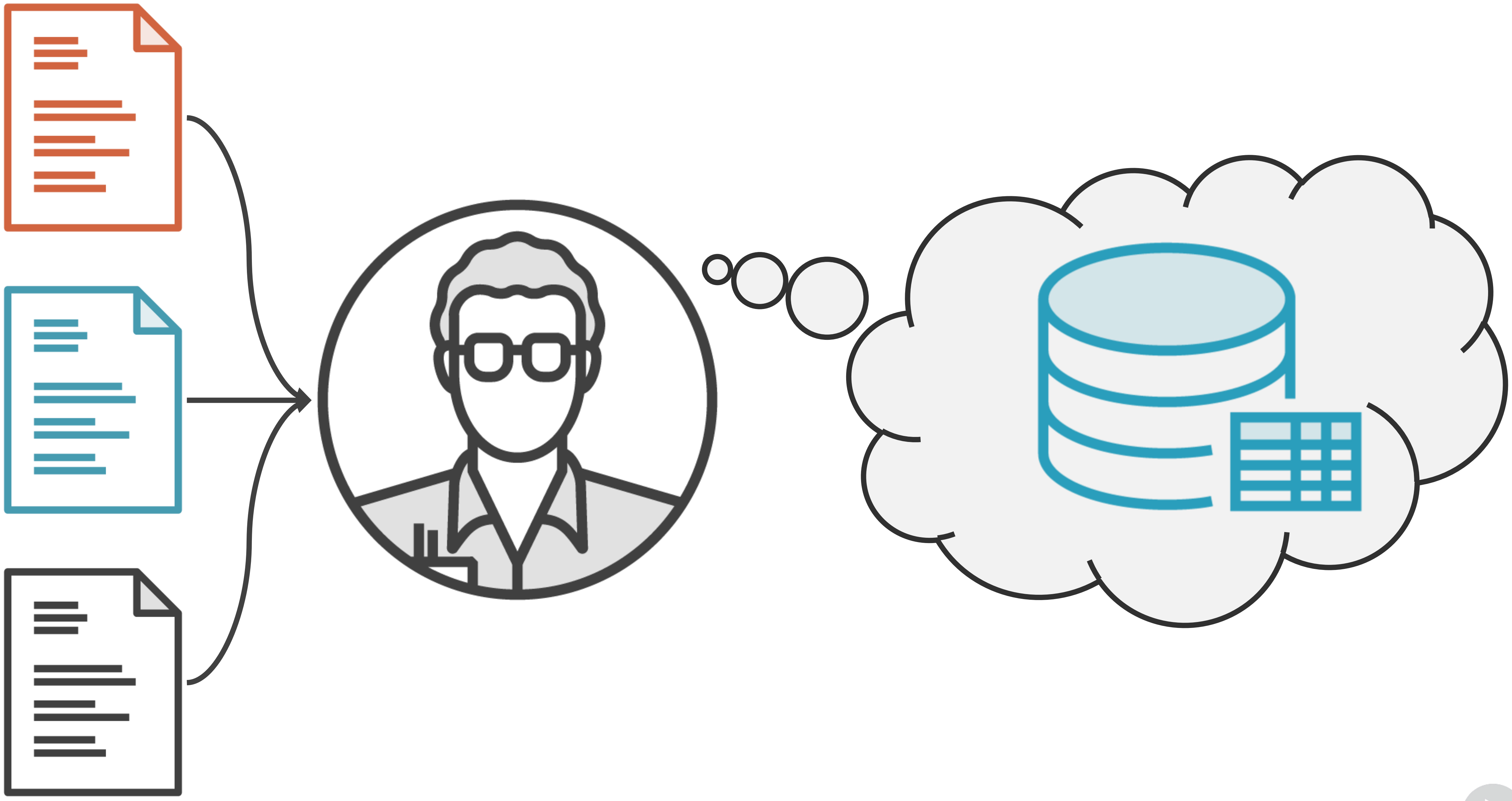
They do allow index creation on individual columns (SQL Server 2014+)

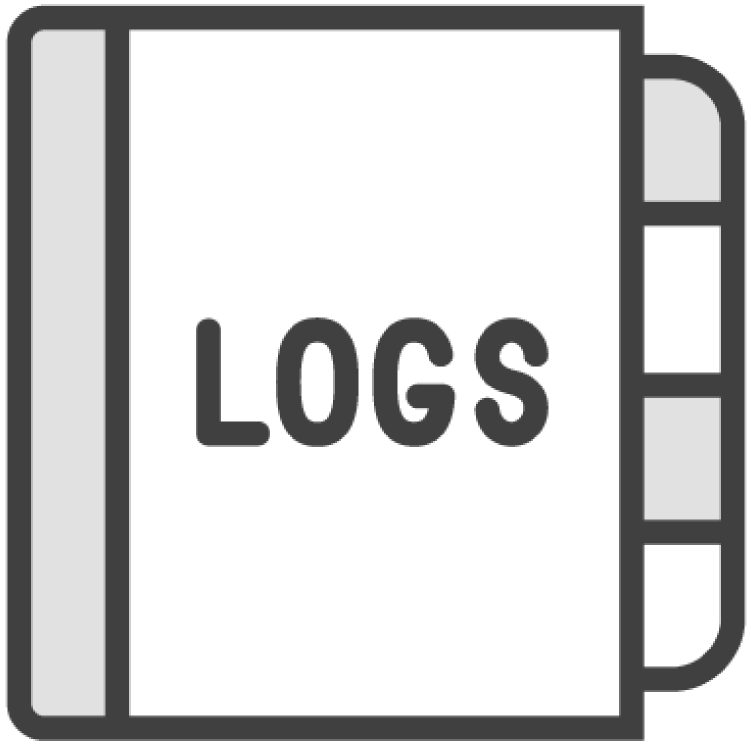
They are not subject to the calling transaction scope

Triggers In Moderation

“When all you have is a hammer, everything looks like a nail.”





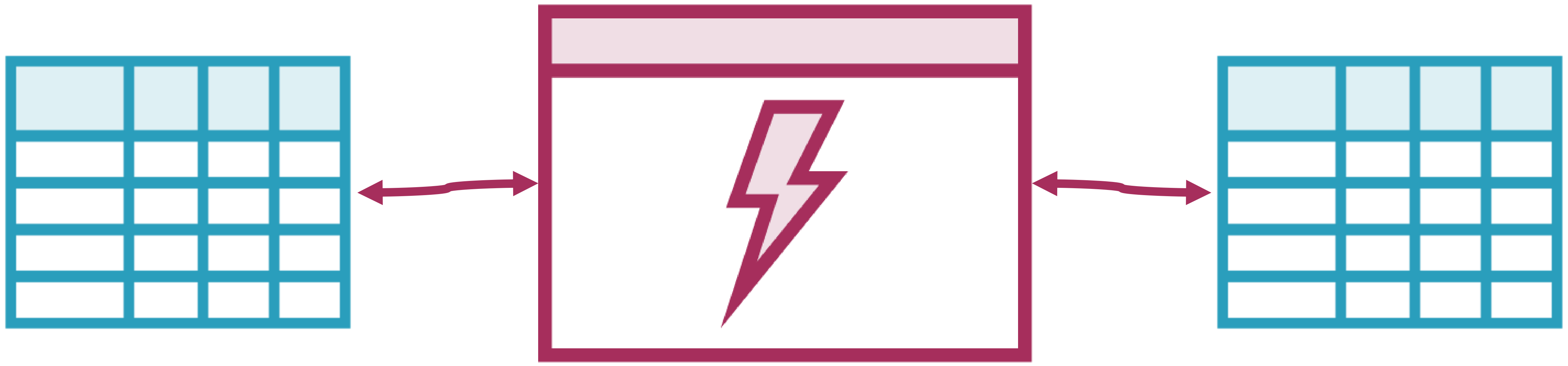


How often do you review the logs?

Do you have an archive strategy?

Will you really remember to use them when something goes wrong?

Foreign Key Triggers?



Unnecessary ROLLBACK’s can cause the transaction log to grow

Triggers are generally *transparent* to developers

The work that Triggers perform is essentially *hidden* from developers

Improving Performance with Service Broker

SQL Server Service Broker (SSB)

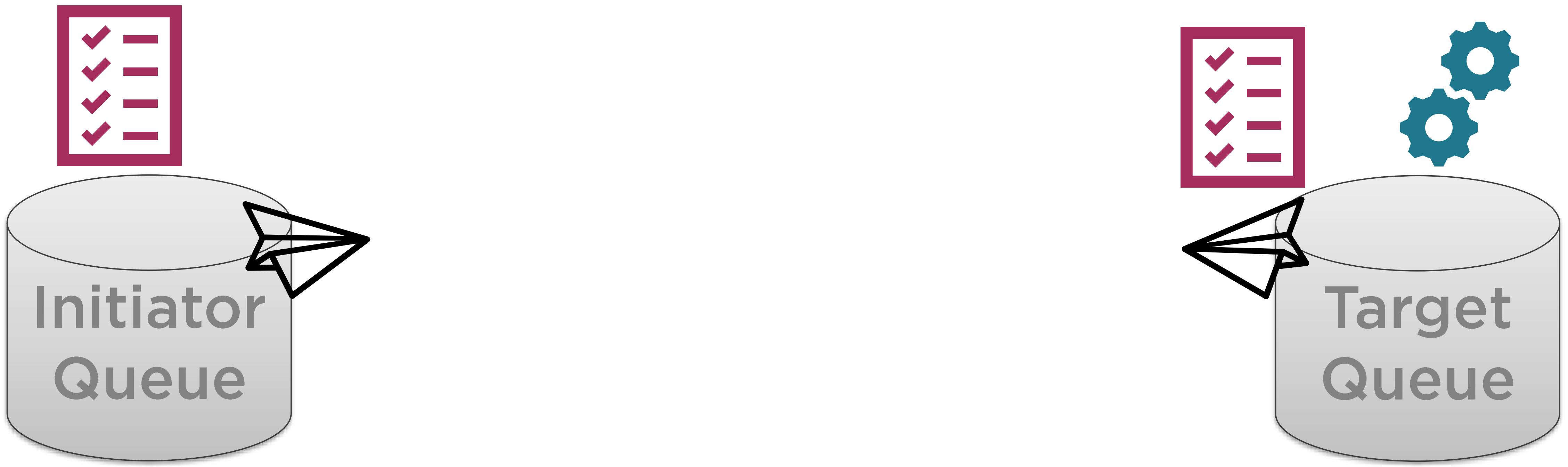
In-Database messaging system (queue)

Allows messages to be acted upon in a new thread

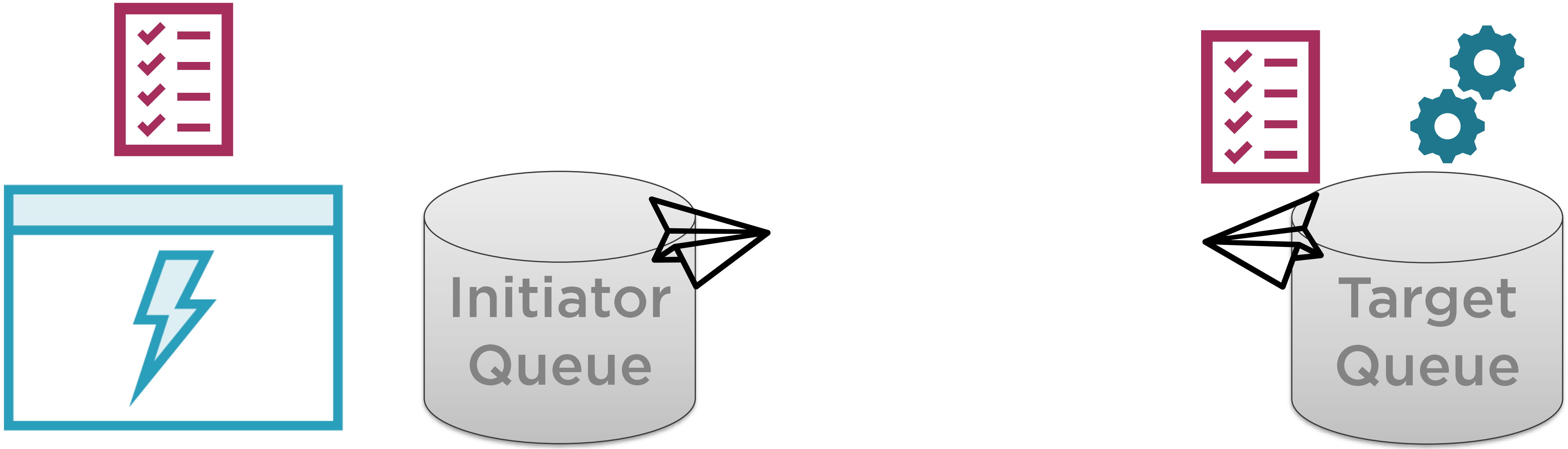
Triggers can use SSB to defer work until later

Especially helpful when Triggers begin a chain reaction of additional work

Basic Service Broker Conversation



Basic Service Broker Conversation



Service Broker Asynchronous Trigger Caution

Any solution involving Asynchronous Triggers must account for many rows

Large XML documents generated by updates to entire tables might be slower than the original Trigger implementation

Async Triggers can be a great tool if used appropriately

Summary

Security and Execution Context

Unexpected Trigger behavior with MERGE

Working outside of the Transaction with Table Variables

Not overusing Triggers

Using Service Broker for Async Triggers