✅ Tracking **SQL Server database property changes** (like changing Auto Close, Recovery Model, Compatibility Level, Read-Only status, etc.) is a key DBA auditing task. Let’s go step-by-step with a **T-SQL DDL trigger** that captures **any database property modification** event.

🧠 Goal

We want to **detect and log any DB property change** such as:

- ALTER DATABASE … MODIFY FILE
- ALTER DATABASE … SET RECOVERY / READ_ONLY / AUTO_CLOSE, etc.
- ALTER AUTHORIZATION ON DATABASE
- ALTER DATABASE COLLATE …

✅ 1. Create an Audit Table

Create a table to store who made the change, when, and what was changed.

```
USE master;
GO

        IF OBJECT_ID('dbo.DBPropertyChangeAudit', 'U') IS NOT NULL
            DROP TABLE dbo.DBPropertyChangeAudit;
        GO

        CREATE TABLE dbo.DBPropertyChangeAudit
        (
            AuditID INT IDENTITY(1,1) PRIMARY KEY,
            EventTime DATETIME NOT NULL DEFAULT GETDATE(),
            EventType NVARCHAR(100),
            DatabaseName NVARCHAR(255),
            LoginName NVARCHAR(255),
            HostName NVARCHAR(255),
            ApplicationName NVARCHAR(255),
            TSQLCommand NVARCHAR(MAX)
        );
        GO
```

✅ 2. Create a DDL Trigger on the Server

We’ll create a **server-level DDL trigger** to capture any ALTER_DATABASE events.

```
        USE master;
        GO

        CREATE TRIGGER trg_Audit_DBPropertyChanges
        ON ALL SERVER
        FOR ALTER_DATABASE
        AS
        BEGIN
            SET NOCOUNT ON;

            DECLARE @EventData XML = EVENTDATA();
```

```
    INSERT INTO dbo.DBPropertyChangeAudit
    (
        EventTime, EventType, DatabaseName, LoginName,
        HostName, ApplicationName, TSQLCommand
    )
    SELECT
        GETDATE(),
        @EventData.value('(/EVENT_INSTANCE/EventType)[1]', 'NVARCHAR(100)'),
        @EventData.value('(/EVENT_INSTANCE/DatabaseName)[1]', 'NVARCHAR(255)'),
        @EventData.value('(/EVENT_INSTANCE/LoginName)[1]', 'NVARCHAR(255)'),
        HOST_NAME(),  APP_NAME(),
        @EventData.value('(/EVENT_INSTANCE/TSQLCommand)[1]', 'NVARCHAR(MAX)');
END;
GO
```

✅ 3. Test the Trigger

Run a simple database property change and check if it’s logged.

```
-- Example property change
ALTER DATABASE AdventureWorks2019 SET AUTO_CLOSE OFF;

-- View logged change
SELECT * FROM master.dbo.DBPropertyChangeAudit ORDER BY EventTime DESC;
```

You should see:

- EventType: ALTER_DATABASE
- DatabaseName: AdventureWorks2019
- LoginName: The user who executed it
- TSQLCommand: The ALTER statement

⚙️ 4. Manage or Disable Trigger (Optional)

If you ever need to disable or drop it:

```
-- Disable temporarily
DISABLE TRIGGER trg_Audit_DBPropertyChanges ON ALL SERVER;

-- Re-enable
ENABLE TRIGGER trg_Audit_DBPropertyChanges ON ALL SERVER;

-- Drop permanently
DROP TRIGGER trg_Audit_DBPropertyChanges ON ALL SERVER;
```

💡 Notes & Recommendations

- This trigger fires for **all ALTER DATABASE** statements on the server.
- You can extend it to log **DROP_DATABASE** or **CREATE_DATABASE** by changing:
- FOR ALTER_DATABASE, DROP_DATABASE, CREATE_DATABASE
- Store audit data in a **central admin database** instead of master if you manage multiple servers.
- Optionally, send alerts (e.g., email via Database Mail) when such events occur.

✅ **DB Property Change Audit Trigger** to send **automatic email alerts** via **Database Mail** whenever someone alters database properties in SQL Server.

🚀 Goal

You'll receive an **instant email notification** when a user executes any

ALTER DATABASE command (changes to Recovery Model, Compatibility, AutoClose, etc.).

⚙ Pre-requisite: Database Mail Setup

Ensure Database Mail is configured and working.

You can check this using:

```
EXEC msdb.dbo.sysmail_help_account_sp;
```

If not configured yet, create a profile (example below):

```
EXEC msdb.dbo.sysmail_add_account_sp
   @account_name = 'SQLDBA_Alerts',
   @description = 'DBA Alert Email Account',
   @email_address = 'dba.alerts@yourcompany.com',
   @display_name = 'SQL Server Alerts',
   @mailserver_name = 'smtp.yourcompany.com';

EXEC msdb.dbo.sysmail_add_profile_sp
   @profile_name = 'SQLDBA_Profile',
   @description = 'Profile for DBA alerts';

EXEC msdb.dbo.sysmail_add_profileaccount_sp
   @profile_name = 'SQLDBA_Profile',
   @account_name = 'SQLDBA_Alerts',
   @sequence_number = 1;

EXEC msdb.dbo.sysmail_add_principalprofile_sp
   @profile_name = 'SQLDBA_Profile',
   @principal_name = 'public',
   @is_default = 1;
```

🧱 Step 1: Create the Audit Table

```
USE master;
GO
IF OBJECT_ID('dbo.DBPropertyChangeAudit', 'U') IS NOT NULL
   DROP TABLE dbo.DBPropertyChangeAudit;
GO
CREATE TABLE dbo.DBPropertyChangeAudit
(
   AuditID INT IDENTITY(1,1) PRIMARY KEY,
   EventTime DATETIME NOT NULL DEFAULT GETDATE(),
   EventType NVARCHAR(100),
   DatabaseName NVARCHAR(255),
   LoginName NVARCHAR(255),
   HostName NVARCHAR(255),
   ApplicationName NVARCHAR(255),
   TSQLCommand NVARCHAR(MAX)
);
```

🧠 Step 2: Create the Server-Level DDL Trigger with Email Notification

```
USE master;
GO
CREATE TRIGGER trg_Audit_DBPropertyChanges
ON ALL SERVER
FOR ALTER_DATABASE
AS
BEGIN
  SET NOCOUNT ON;

  DECLARE @EventData XML = EVENTDATA();
  DECLARE @DBName NVARCHAR(255) = @EventData.value('(/EVENT_INSTANCE/DatabaseName)[1]', 'NVARCHAR(255)');
  DECLARE @LoginName NVARCHAR(255) = @EventData.value('(/EVENT_INSTANCE/LoginName)[1]', 'NVARCHAR(255)');
  DECLARE @TSQLCommand NVARCHAR(MAX) = @EventData.value('(/EVENT_INSTANCE/TSQLCommand)[1]', 'NVARCHAR(MAX)');
  DECLARE @EventType NVARCHAR(100) = @EventData.value('(/EVENT_INSTANCE/EventType)[1]', 'NVARCHAR(100)');
  DECLARE @Subject NVARCHAR(200);
  DECLARE @Body NVARCHAR(MAX);
  -- Insert into audit table
  INSERT INTO dbo.DBPropertyChangeAudit
  (
    EventTime, EventType, DatabaseName, LoginName, HostName, ApplicationName, TSQLCommand
  )
  VALUES
  (
    GETDATE(), @EventType, @DBName, @LoginName, HOST_NAME(), APP_NAME(), @TSQLCommand
  );
  -- Prepare email content
  SET @Subject = '⚠ SQL ALERT: Database Property Change Detected on ' + @DBName;
  SET @Body = 'A database property was modified in SQL Server.' + CHAR(13) + CHAR(10) +
        '----------------------------------------------------------' + CHAR(13) + CHAR(10) +
        '📅 Time: ' + CONVERT(NVARCHAR(30), GETDATE(), 120) + CHAR(13) + CHAR(10) +
        '🧩 Database: ' + @DBName + CHAR(13) + CHAR(10) +
        '👤 Login Name: ' + @LoginName + CHAR(13) + CHAR(10) +
        '💻 Host Name: ' + HOST_NAME() + CHAR(13) + CHAR(10) +
        '🧠 Application: ' + APP_NAME() + CHAR(13) + CHAR(10) +
        '📝 Event Type: ' + @EventType + CHAR(13) + CHAR(10) +
        '----------------------------------------------------------' + CHAR(13) + CHAR(10) +
        'T-SQL Executed:' + CHAR(13) + CHAR(10) + @TSQLCommand + CHAR(13) + CHAR(10) +
        '----------------------------------------------------------' + CHAR(13) + CHAR(10) +
        '📩 Sent automatically by SQL Server Audit Trigger.';

  -- Send email via Database Mail
  EXEC msdb.dbo.sp_send_dbmail
    @profile_name = 'SQLDBA_Profile',
    @recipients = 'sqlalerts@yourcompany.com',
    @subject = @Subject,
    @body = @Body;
END;
GO
```

🧪 Step 3: Test It

Run a database property change:

```
ALTER DATABASE AdventureWorks2019 SET AUTO_UPDATE_STATISTICS OFF;
```

Then check:

```
SELECT * FROM master.dbo.DBPropertyChangeAudit ORDER BY EventTime DESC;
```

And verify that you received an **email alert** with all the details.

📊 Sample Email Output

Subject:

⚠ SQL ALERT: Database Property Change Detected on AdventureWorks2019

Body:

```
A database property was modified in SQL Server.
------------------------------------------------------------
📅 Time: 2025-10-23 09:42:13
🧩 Database: AdventureWorks2019
👤 Login Name: DOMAIN\Praveen.M
💻 Host Name: DBA-SERVER01
🧠 Application: Microsoft SQL Server Management Studio
📝 Event Type: ALTER_DATABASE
------------------------------------------------------------
T-SQL Executed:
ALTER DATABASE AdventureWorks2019 SET AUTO_UPDATE_STATISTICS OFF
------------------------------------------------------------
📩 Sent automatically by SQL Server Audit Trigger.
```

🧯 Optional Enhancements

- Include CREATE_DATABASE and DROP_DATABASE:
- FOR ALTER_DATABASE, CREATE_DATABASE, DROP_DATABASE
- Filter to alert only for **production databases**:
- IF @DBName NOT IN ('tempdb', 'model', 'msdb')
- Add error handling if Database Mail fails.

✅ **Professional DBA move** — keeping all change logs centralized in a **DBA Repository Database** makes long-term tracking, reporting, and alerting much easier (especially across multiple SQL Servers).

Below is a **production-ready version** of your trigger setup, designed to log database property changes into a **DBA_Repository** database and send **email alerts** through Database Mail.

🧠 Objective

Detect and audit **any ALTER DATABASE property change** (or optionally CREATE/DROP DATABASE) across all SQL instances,

✅ Log the details into a central **DBA_Repository** database

✅ Send real-time **email alerts** to the DBA team

🏗️ Step 1: Create or Verify Your Central Repository Database

You only need to do this **once per SQL instance** (or once centrally if linked servers are used).

```
USE master;
GO

IF DB_ID('DBA_Repository') IS NULL
BEGIN
  CREATE DATABASE DBA_Repository;
  PRINT '✅ Created DBA_Repository database';
END
ELSE
  PRINT '✅ DBA_Repository database already exists';
GO
```

📦 Step 2: Create the Audit Table in DBA_Repository

```
USE DBA_Repository;
GO

IF OBJECT_ID('dbo.DBPropertyChangeAudit', 'U') IS NOT NULL
  DROP TABLE dbo.DBPropertyChangeAudit;
GO

CREATE TABLE dbo.DBPropertyChangeAudit
(
  AuditID INT IDENTITY(1,1) PRIMARY KEY,
  EventTime DATETIME NOT NULL DEFAULT GETDATE(),
  EventType NVARCHAR(100),
  DatabaseName NVARCHAR(255),
  LoginName NVARCHAR(255),
  HostName NVARCHAR(255),
  ApplicationName NVARCHAR(255),
  TSQLCommand NVARCHAR(MAX),
  ServerName NVARCHAR(255) DEFAULT @@SERVERNAME
);
GO
```

🧰 Step 3: Create the Server-Level DDL Trigger

This trigger will:

- Fire for every **ALTER DATABASE** event
- Log details into DBA_Repository.dbo.DBPropertyChangeAudit
- Send email alert via **Database Mail**

```
USE master;
GO
CREATE OR ALTER TRIGGER trg_Audit_DBPropertyChanges
ON ALL SERVER
FOR ALTER_DATABASE
AS
BEGIN
  SET NOCOUNT ON;
  DECLARE @EventData XML = EVENTDATA();
  DECLARE @DBName NVARCHAR(255) = @EventData.value('(/EVENT_INSTANCE/DatabaseName)[1]', 'NVARCHAR(255)');
  DECLARE @LoginName NVARCHAR(255) = @EventData.value('(/EVENT_INSTANCE/LoginName)[1]', 'NVARCHAR(255)');
  DECLARE @TSQLCommand NVARCHAR(MAX) = @EventData.value('(/EVENT_INSTANCE/TSQLCommand)[1]', 'NVARCHAR(MAX)');
  DECLARE @EventType NVARCHAR(100) = @EventData.value('(/EVENT_INSTANCE/EventType)[1]', 'NVARCHAR(100)');
  DECLARE @Subject NVARCHAR(200);
  DECLARE @Body NVARCHAR(MAX);
  DECLARE @ServerName NVARCHAR(255) = @@SERVERNAME;

  BEGIN TRY
    -- Insert into centralized repository
    INSERT INTO DBA_Repository.dbo.DBPropertyChangeAudit
    (
      EventTime, EventType, DatabaseName, LoginName, HostName, ApplicationName, TSQLCommand, ServerName
    )
    VALUES
    (
      GETDATE(), @EventType, @DBName, @LoginName, HOST_NAME(), APP_NAME(), @TSQLCommand, @ServerName
    );
    -- Prepare email details
    SET @Subject = N'⚠ SQL ALERT: Database Property Change Detected on [' + @ServerName + N'] - ' + @DBName;

    SET @Body =
    'A database property was modified in SQL Server.' + CHAR(13) + CHAR(10) +
    '-------------------------------------------------------------' + CHAR(13) + CHAR(10) +
    '📅 Time: ' + CONVERT(NVARCHAR(30), GETDATE(), 120) + CHAR(13) + CHAR(10) +
    '🧩 Database: ' + ISNULL(@DBName, 'N/A') + CHAR(13) + CHAR(10) +
    '👤 Login Name: ' + ISNULL(@LoginName, 'N/A') + CHAR(13) + CHAR(10) +
    '💻 Host Name: ' + ISNULL(HOST_NAME(), 'N/A') + CHAR(13) + CHAR(10) +
    '🧠 Application: ' + ISNULL(APP_NAME(), 'N/A') + CHAR(13) + CHAR(10) +
    '📝 Event Type: ' + ISNULL(@EventType, 'N/A') + CHAR(13) + CHAR(10) +
    '🏢 Server Name: ' + @ServerName + CHAR(13) + CHAR(10) +
    '-------------------------------------------------------------' + CHAR(13) + CHAR(10) +
    'T-SQL Executed:' + CHAR(13) + CHAR(10) + ISNULL(@TSQLCommand, 'N/A') + CHAR(13) + CHAR(10) +
    '-------------------------------------------------------------' + CHAR(13) + CHAR(10) +
    '📩 Sent automatically by SQL Server Audit Trigger.';
    -- Send email alert (customize recipients)
    EXEC msdb.dbo.sp_send_dbmail
      @profile_name = 'SQLDBA_Profile',  -- Your Database Mail profile
      @recipients = 'sqlalerts@yourcompany.com; dba_team@yourcompany.com',
      @subject = @Subject,
      @body = @Body;

  END TRY
  BEGIN CATCH
    -- Log trigger errors into DBA_Repository for review
    INSERT INTO DBA_Repository.dbo.DBPropertyChangeAudit
    (
      EventTime, EventType, DatabaseName, LoginName, HostName, ApplicationName, TSQLCommand, ServerName
    )
    VALUES
    (
      GETDATE(), 'TRIGGER_ERROR',
      ERROR_MESSAGE(), SYSTEM_USER, HOST_NAME(), APP_NAME(), ERROR_PROCEDURE(), @ServerName
    );
  END CATCH
END;
GO
```

🧪 Step 4: Test the Setup

Run a database property change command:

```
ALTER DATABASE AdventureWorks2019 SET RECOVERY SIMPLE;
```

Then check your central repository:

```
SELECT *
FROM DBA_Repository.dbo.DBPropertyChangeAudit
ORDER BY EventTime DESC;
```

✅ You’ll also receive an **email alert** with all details.

⚙️ Optional Enhancements

- **Expand trigger scope**:
- FOR ALTER_DATABASE, CREATE_DATABASE, DROP_DATABASE
- **Exclude non-critical databases**:
- IF @DBName IN ('tempdb', 'model', 'msdb') RETURN;
- **Centralize across multiple servers**:
  - Use **Linked Servers** to push all logs to one central DBA server.
  - Or use SQL Agent jobs to sync logs daily from each instance.

📋 Example Report Query (Centralized View)

Run this on your **DBA_Repository** to review who changed what:

```
SELECT
  ServerName, DatabaseName,  LoginName, HostName,
  ApplicationName, EventType, LEFT(TSQLCommand, 200) AS ExecutedCommand,
  EventTime
FROM dbo.DBPropertyChangeAudit
ORDER BY EventTime DESC;
```

✅ Let’s **automate daily summary reports** for all database property changes logged in your DBA_Repository — so even if an alert email was missed, the team still gets a full daily digest.

📅 Goal

Automatically send a **Daily Database Property Change Report** to the DBA team
via SQL Server Agent + Database Mail.

⚙️ Step 1: Create SQL Agent Job – "Daily_DB_Property_Change_Report"

📘 Job Overview

Property	Value
Job Name	Daily_DB_Property_Change_Report
Schedule	Every day at 8:00 AM
Database	DBA_Repository
Mail Profile	SQLDBA_Profile
Recipients	sqlalerts@yourcompany.com; dba_team@yourcompany.com

🧩 Step 2: Job T-SQL Script

Create the SQL Agent Job with this script (run in **msdb**):

```
USE msdb;
GO
EXEC sp_add_job
    @job_name = N'Daily_DB_Property_Change_Report',
    @enabled = 1,
    @description = N'Sends daily summary of all ALTER/CREATE/DROP DATABASE property changes to DBA Team',
    @category_name = N'Database Maintenance';
GO

-- Add job step
EXEC sp_add_jobstep
    @job_name = N'Daily_DB_Property_Change_Report',
    @step_name = N'Send Daily DB Property Change Email',
    @subsystem = N'TSQL',
    @database_name = N'DBA_Repository',
    @command = N'
DECLARE @HTML NVARCHAR(MAX);
DECLARE @Yesterday DATETIME = DATEADD(DAY, -1, CAST(GETDATE() AS DATE));

SET @HTML =
    N''<h2 style="color:#2E86C1;">📊 Daily Database Property Change Report</h2>'' +
    N''<p>Date: '' + CONVERT(NVARCHAR(20), GETDATE(), 107) + N''</p>'' +
    N''<table border="1" cellpadding="6" cellspacing="0" style="border-collapse:collapse;font-family:Segoe UI;font-size:12px;">
        <tr style="background-color:#2E86C1;color:white;">
            <th>Server Name</th>
            <th>Database</th>
            <th>Event Type</th>
            <th>Login</th>
            <th>Host</th>
            <th>Application</th>
```

```
      <th>Event Time</th>
      <th>T-SQL Command</th>
    </tr>'' +
  CAST((
    SELECT
      td = ServerName, '''',
      td = DatabaseName, '''',
      td = EventType, '''',
      td = LoginName, '''',
      td = HostName, '''',
      td = ApplicationName, '''',
      td = CONVERT(VARCHAR(20), EventTime, 120), '''',
      td = LEFT(TSQLCommand, 200)
    FROM dbo.DBPropertyChangeAudit
    WHERE EventTime >= @Yesterday
    ORDER BY EventTime DESC
    FOR XML PATH(''tr''), TYPE
  ) AS NVARCHAR(MAX)) +
  N''</table><br><p style="font-size:11px;color:gray;">Report generated automatically by SQL Server Audit System.</p>'';

-- Send email
EXEC msdb.dbo.sp_send_dbmail
  @profile_name = ''SQLDBA_Profile'',
  @recipients = ''sqlalerts@yourcompany.com;dba_team@yourcompany.com'',
  @subject = ''SQL Audit: Daily DB Property Change Report'',
  @body = @HTML,
  @body_format = ''HTML'';
',
  @retry_attempts = 1,
  @retry_interval = 5;
GO

-- Add schedule (8:00 AM daily)
EXEC sp_add_schedule
  @schedule_name = N'Daily_8AM_Schedule',
  @freq_type = 4,          -- daily
  @freq_interval = 1,       -- every day
  @active_start_time = 80000; -- 8:00 AM
GO

EXEC sp_attach_schedule
  @job_name = N'Daily_DB_Property_Change_Report',
  @schedule_name = N'Daily_8AM_Schedule';
GO

EXEC sp_add_jobserver
  @job_name = N'Daily_DB_Property_Change_Report';
GO
```

🧾 Step 3: Verify Job Setup

Run this to confirm job details:

EXEC msdb.dbo.sp_help_job @job_name = 'Daily_DB_Property_Change_Report';

🧪 Step 4: Manual Test Run

You can test the report email immediately:

EXEC msdb.dbo.sp_start_job 'Daily_DB_Property_Change_Report';

Then check your email inbox for the formatted HTML table report.

Each row shows — **Server**, **DB Name**, **Event Type**, **Login**, and **T-SQL Command.**

🚀 Bonus Add-ons (Optional)

Enhancement	Description
🔄 **Centralized Reports**	Push logs from all SQL instances into one main DBA_Repository using linked servers or ETL job.
🕵️ **Severity Filter**	Modify WHERE EventType IN (...) to include/exclude CREATE/DROP DB.
📁 **Archive Old Logs**	Add a weekly step to delete or archive audit data older than 90 days.
📊 **Power BI / SSRS Dashboard**	Build a live dashboard on top of DBPropertyChangeAudit table for management visibility.

✅ **Daily Audit Job** to automatically **archive or purge** records older than 90 days from the DBPropertyChangeAudit table in your DBA_Repository. This will keep your central audit database clean and lightweight while preserving the last 3 months of change history.

🛠️ Updated Job: Daily_DB_Property_Change_Report

Now it will:

1. **Purge/Archive** data older than 90 days.
2. **Send daily email report** of last 24 hours' DB property changes.

⚙️ Option 1: Purge old records (simpler and faster)

If you just want to **delete** data older than 90 days:

```
USE msdb;
GO

EXEC sp_update_jobstep
   @job_name = N'Daily_DB_Property_Change_Report',
   @step_id = 1,
   @command = N'
DECLARE @HTML NVARCHAR(MAX);
DECLARE @Yesterday DATETIME = DATEADD(DAY, -1, CAST(GETDATE() AS DATE));
DECLARE @PurgeDate DATETIME = DATEADD(DAY, -90, GETDATE());

------------------------------
-- 🔷 Step 1: Purge Old Records
------------------------------
DELETE FROM dbo.DBPropertyChangeAudit
WHERE EventTime < @PurgeDate;

------------------------------
-- 🔷 Step 2: Generate HTML Report
------------------------------
SET @HTML =
   N''<h2 style="color:#2E86C1;">📊 Daily Database Property Change Report</h2>'' +
   N''<p>Date: '' + CONVERT(NVARCHAR(20), GETDATE(), 107) + N''</p>'' +
   N''<table border="1" cellpadding="6" cellspacing="0" style="border-collapse:collapse;font-family:Segoe UI;font-size:12px;">
      <tr style="background-color:#2E86C1;color:white;">
         <th>Server Name</th>
         <th>Database</th>
         <th>Event Type</th>
         <th>Login</th>
         <th>Host</th>
         <th>Application</th>
         <th>Event Time</th>
         <th>T-SQL Command</th>
      </tr>'' +
   CAST((
      SELECT
         td = ServerName, '''',
         td = DatabaseName, '''',
         td = EventType, '''',
        td = LoginName, '''',
        td = HostName, '''',
        td = ApplicationName, '''',
        td = CONVERT(VARCHAR(20), EventTime, 120), '''',
        td = LEFT(TSQLCommand, 200)
      FROM dbo.DBPropertyChangeAudit
      WHERE EventTime >= @Yesterday
      ORDER BY EventTime DESC
      FOR XML PATH(''tr''), TYPE
   ) AS NVARCHAR(MAX)) +
   N''</table><br><p style="font-size:11px;color:gray;">Report generated automatically by SQL Server Audit System.</p>'';

------------------------------
-- 🔷 Step 3: Send Email
------------------------------
EXEC msdb.dbo.sp_send_dbmail
   @profile_name = ''SQLDBA_Profile'',
   @recipients = ''sqlalerts@yourcompany.com;dba_team@yourcompany.com'',
   @subject = ''SQL Audit: Daily DB Property Change Report'',
   @body = @HTML,
   @body_format = ''HTML'';
';
GO
```

⚙️ **Option 2: Archive instead of purge (recommended for compliance)**

If you need to **retain history** longer, you can move old records into an archive table before deletion.

🧱 **Step 1: Create Archive Table**

Run once in DBA_Repository:

```
USE DBA_Repository;
GO

IF OBJECT_ID('dbo.DBPropertyChangeAudit_Archive') IS NULL
BEGIN
   SELECT TOP 0 * INTO dbo.DBPropertyChangeAudit_Archive
   FROM dbo.DBPropertyChangeAudit;
END
GO
```

🧩 **Step 2: Modify Job Step**

Update job to archive instead of delete:

```
USE msdb;
GO

EXEC sp_update_jobstep
   @job_name = N'Daily_DB_Property_Change_Report',
   @step_id = 1,
   @command = N'
DECLARE @HTML NVARCHAR(MAX);
DECLARE @Yesterday DATETIME = DATEADD(DAY, -1, CAST(GETDATE() AS DATE));
DECLARE @ArchiveDate DATETIME = DATEADD(DAY, -90, GETDATE());
```

```
------------------------------
-- 🔷 Step 1: Archive Old Records (>90 days)
------------------------------
INSERT INTO dbo.DBPropertyChangeAudit_Archive
SELECT * FROM dbo.DBPropertyChangeAudit WHERE EventTime < @ArchiveDate;

DELETE FROM dbo.DBPropertyChangeAudit WHERE EventTime < @ArchiveDate;


------------------------------
-- 🔷 Step 2: Generate HTML Report
------------------------------
SET @HTML =
  N''<h2 style="color:#2E86C1;">📊 Daily Database Property Change Report</h2>'' +
  N''<p>Date: '' + CONVERT(NVARCHAR(20), GETDATE(), 107) + N''</p>'' +
  N''<table border="1" cellpadding="6" cellspacing="0" style="border-collapse:collapse;font-family:Segoe UI;font-size:12px;">
    <tr style="background-color:#2E86C1;color:white;">
      <th>Server Name</th>
      <th>Database</th>
      <th>Event Type</th>
      <th>Login</th>
      <th>Host</th>
      <th>Application</th>
      <th>Event Time</th>
      <th>T-SQL Command</th>
    </tr>'' +
  CAST((
    SELECT
      td = ServerName, '''',
      td = DatabaseName, '''',
      td = EventType, '''',
      td = LoginName, '''',
      td = HostName, '''',
      td = ApplicationName, '''',
      td = CONVERT(VARCHAR(20), EventTime, 120), '''',
      td = LEFT(TSQLCommand, 200)
    FROM dbo.DBPropertyChangeAudit
    WHERE EventTime >= @Yesterday
    ORDER BY EventTime DESC
    FOR XML PATH(''tr''), TYPE
  ) AS NVARCHAR(MAX)) +
  N''</table><br><p style="font-size:11px;color:gray;">Report generated automatically by SQL Server Audit System.</p>'';

------------------------------
-- 🔷 Step 3: Send Email
------------------------------
EXEC msdb.dbo.sp_send_dbmail
  @profile_name = ''SQLDBA_Profile'',
  @recipients = ''sqlalerts@yourcompany.com;dba_team@yourcompany.com'',
  @subject = ''SQL Audit: Daily DB Property Change Report'',
    @body = @HTML,
    @body_format = ''HTML'';
';
GO
```

📅 Step 3: Test Archive Job

Manually execute:

```
EXEC msdb.dbo.sp_start_job 'Daily_DB_Property_Change_Report';
```

Then confirm:

```
SELECT COUNT(*) FROM DBA_Repository.dbo.DBPropertyChangeAudit_Archive;
```

🧠 Pro Tip for Production

Feature	Recommendation
🔒 **Retention**	Keep 90 days in main table, archive yearly backups for 2 years.
📦 **Compression**	Enable row compression on the archive table.
📜 **Indexing**	Add index on (EventTime) for faster purging.
💡 **Notification**	Add a @copy_recipients parameter for your lead DBA.

✅ **Pro DBA repository** should evolve — fast, clean, and storage-optimized.
Let's now complete the setup with **indexing + compression optimization** for both live and archive audit tables.

⚙️ Goal

Optimize your DBA_Repository audit tables to improve:

- 🔍 **Query performance** (report generation + searches)
- 📦 **Storage efficiency** (via row compression)
- 🧹 **Maintenance speed** (for daily cleanup jobs)

🧱 Step 1: Add Recommended Indexes

Run in your DBA_Repository database:

```
USE DBA_Repository;
GO

-- Drop existing nonclustered indexes if any (safe cleanup)
IF EXISTS (SELECT * FROM sys.indexes WHERE name = 'IX_DBPropertyChangeAudit_EventTime')
   DROP INDEX IX_DBPropertyChangeAudit_EventTime ON dbo.DBPropertyChangeAudit;
GO

IF EXISTS (SELECT * FROM sys.indexes WHERE name = 'IX_DBPropertyChangeAudit_Archive_EventTime')
   DROP INDEX IX_DBPropertyChangeAudit_Archive_EventTime ON dbo.DBPropertyChangeAudit_Archive;
GO

-- Create indexes for fast search and purge
CREATE NONCLUSTERED INDEX IX_DBPropertyChangeAudit_EventTime
ON dbo.DBPropertyChangeAudit (EventTime)
INCLUDE (ServerName, DatabaseName, EventType, LoginName, HostName, ApplicationName);
GO

CREATE NONCLUSTERED INDEX IX_DBPropertyChangeAudit_Archive_EventTime
ON dbo.DBPropertyChangeAudit_Archive (EventTime)
INCLUDE (ServerName, DatabaseName, EventType, LoginName, HostName, ApplicationName);
GO
```

✅ **Benefits:**

- Faster filtering on EventTime for daily reports and purges.
- Reduces I/O on the main table by using a covering index.

🧩 Step 2: Enable Row Compression (Space Saver)

This feature significantly reduces storage used by audit logs (especially on NVARCHAR + DATETIME-heavy tables).

```
-- Enable ROW compression on main audit table
ALTER TABLE dbo.DBPropertyChangeAudit
REBUILD PARTITION = ALL
WITH (DATA_COMPRESSION = ROW);
GO

-- Enable ROW compression on archive table
ALTER TABLE dbo.DBPropertyChangeAudit_Archive
REBUILD PARTITION = ALL
WITH (DATA_COMPRESSION = ROW);
```

✅ Benefits:

- Reduces disk space usage by **40–60%**
- Minimal CPU overhead for read/write
- Excellent for mostly-append workloads like audit logs

🧰 Step 3: Optional — Verify Compression Savings

You can check how much space you saved using this DMV:

```
SELECT
   t.name AS TableName,
   i.name AS IndexName,
   p.rows AS RowCount,
   au.type_desc AS StorageType,
   au.total_pages * 8 / 1024 AS SizeMB
FROM sys.partitions p
JOIN sys.allocation_units au ON p.partition_id = au.container_id
JOIN sys.tables t ON p.object_id = t.object_id
LEFT JOIN sys.indexes i ON p.object_id = i.object_id AND p.index_id = i.index_id
WHERE t.name IN ('DBPropertyChangeAudit', 'DBPropertyChangeAudit_Archive')
ORDER BY SizeMB DESC;
```

🧩 Step 4: Automate Monthly Index Maintenance (Optional)

You can add this small **index + compression maintenance step** into a monthly SQL Agent job:

```
USE DBA_Repository;
GO

ALTER INDEX ALL ON dbo.DBPropertyChangeAudit REORGANIZE;
ALTER INDEX ALL ON dbo.DBPropertyChangeAudit_Archive REORGANIZE;

ALTER TABLE dbo.DBPropertyChangeAudit
REBUILD PARTITION = ALL WITH (DATA_COMPRESSION = ROW);

ALTER TABLE dbo.DBPropertyChangeAudit_Archive
REBUILD PARTITION = ALL WITH (DATA_COMPRESSION = ROW);
GO
```

Schedule it **monthly** or **after weekend patching** to keep fragmentation low.

✅ Summary of Benefits

Optimization	**Benefit**
Nonclustered index on EventTime	Fast report queries & cleanup
ROW compression	40–60% storage reduction
Monthly index maintenance	Keeps report speed consistent
Archive table strategy	Long-term retention & compliance

Fully automated SQL Agent job that performs monthly index and compression maintenance for your audit repository.
This job ensures your audit tables remain **fast, compact, and self-healing** — a hallmark of a production-grade DBA environment.

⚙️ Job Overview

Property	Value
Job Name	Monthly_AuditRepo_Index_Compression_Maintenance
Database	DBA_Repository
Schedule	Runs on the **1st Sunday of every month at 2:00 AM**
Output	Email summary of index rebuild & compression actions
Mail Profile	SQLDBA_Profile

🧱 Step 1: Create SQL Agent Job

Run this entire script in msdb:

```
USE msdb;
GO
----------------------------------------------------------
-- Create the Job
----------------------------------------------------------
EXEC sp_add_job
   @job_name = N'Monthly_AuditRepo_Index_Compression_Maintenance',
   @enabled = 1,
   @description = N'Monthly index reorganization, rebuild, and compression maintenance for audit repository tables.',
   @category_name = N'Database Maintenance';
GO

----------------------------------------------------------
-- Add Job Step: Maintenance + Email Summary
----------------------------------------------------------
EXEC sp_add_jobstep
   @job_name = N'Monthly_AuditRepo_Index_Compression_Maintenance',
   @step_name = N'Perform Index Maintenance and Compression',
   @subsystem = N'TSQL',
   @database_name = N'DBA_Repository',
   @command = N'
DECLARE @StartTime DATETIME = GETDATE();
DECLARE @HTML NVARCHAR(MAX);
DECLARE @RowCount INT = 0;

----------------------------------------------------------
-- Step 1: Reorganize & Rebuild Indexes
----------------------------------------------------------
PRINT ''Starting index maintenance...'';
ALTER INDEX ALL ON dbo.DBPropertyChangeAudit REORGANIZE;
ALTER INDEX ALL ON dbo.DBPropertyChangeAudit_Archive REORGANIZE;
```

```
----------------------------------------------------------
-- Step 2: Rebuild with Row Compression
----------------------------------------------------------
PRINT ''Rebuilding tables with ROW compression...'';
ALTER TABLE dbo.DBPropertyChangeAudit
REBUILD PARTITION = ALL WITH (DATA_COMPRESSION = ROW);
ALTER TABLE dbo.DBPropertyChangeAudit_Archive
REBUILD PARTITION = ALL WITH (DATA_COMPRESSION = ROW);

----------------------------------------------------------
-- Step 3: Get Updated Space Report
----------------------------------------------------------
DECLARE @Report TABLE (
    TableName NVARCHAR(255),
    IndexName NVARCHAR(255),
    RowCount BIGINT,
    SizeMB DECIMAL(18,2)
);

INSERT INTO @Report
SELECT
    t.name AS TableName,
    i.name AS IndexName,
    SUM(p.rows) AS RowCount,
    SUM(au.total_pages) * 8.0 / 1024 AS SizeMB
FROM sys.partitions p
JOIN sys.allocation_units au ON p.partition_id = au.container_id
JOIN sys.tables t ON p.object_id = t.object_id
LEFT JOIN sys.indexes i ON p.object_id = i.object_id AND p.index_id = i.index_id
WHERE t.name IN (''DBPropertyChangeAudit'', ''DBPropertyChangeAudit_Archive'')
GROUP BY t.name, i.name
ORDER BY t.name;

----------------------------------------------------------
-- Step 4: Build HTML Email Summary
----------------------------------------------------------
SET @HTML =
    N''<h2 style="color:#2E86C1;">🧹 Monthly Audit Repository Maintenance Report</h2>'' +
    N''<p><b>Executed On:</b> '' + CONVERT(NVARCHAR(20), @StartTime, 120) + N''</p>'' +
    N''<p><b>Completed At:</b> '' + CONVERT(NVARCHAR(20), GETDATE(), 120) + N''</p><br>'' +
    N''<table border="1" cellpadding="6" cellspacing="0" style="border-collapse:collapse;font-family:Segoe UI;font-size:12px;">
        <tr style="background-color:#2E86C1;color:white;">
            <th>Table Name</th>
            <th>Index Name</th>
            <th>Row Count</th>
            <th>Size (MB)</th>
        </tr>'' +
    CAST((
SELECT
          td = TableName, '''',
          td = IndexName, '''',
          td = FORMAT(RowCount, ''#,##0''), '''',
          td = FORMAT(SizeMB, ''#,##0.00'')
      FROM @Report
      FOR XML PATH(''tr''), TYPE
   ) AS NVARCHAR(MAX)) +
   N''</table><br>
   <p style="font-size:11px;color:gray;">Maintenance completed successfully. Generated by SQL Server DBA Automation System.</p>'';

----------------------------------------------------------
-- Step 5: Send Email
----------------------------------------------------------
EXEC msdb.dbo.sp_send_dbmail
   @profile_name = ''SQLDBA_Profile'',
   @recipients = ''sqlalerts@yourcompany.com;dba_team@yourcompany.com'',
   @subject = ''SQL Audit Repository Maintenance Report - '' + CONVERT(NVARCHAR(20), @StartTime, 107),
   @body = @HTML,
   @body_format = ''HTML'';
',
   @retry_attempts = 1,
   @retry_interval = 5;
GO

----------------------------------------------------------
-- Add Monthly Schedule (1st Sunday, 2:00 AM)
----------------------------------------------------------
EXEC sp_add_schedule
   @schedule_name = N'Monthly_1st_Sunday_2AM',
   @freq_type = 8,           -- Weekly
   @freq_interval = 1,        -- Sunday
   @freq_recurrence_factor = 4,-- Every 4 weeks (monthly)
   @active_start_time = 20000; -- 2:00 AM
GO

EXEC sp_attach_schedule
   @job_name = N'Monthly_AuditRepo_Index_Compression_Maintenance',
   @schedule_name = N'Monthly_1st_Sunday_2AM';
GO

EXEC sp_add_jobserver
   @job_name = N'Monthly_AuditRepo_Index_Compression_Maintenance';
GO
```

📧 Sample Email Report

Subject:

SQL Audit Repository Maintenance Report – Oct 2025

Body (HTML):

Table Name	Index Name	Row Count	Size (MB)
DBPropertyChangeAudit	IX_DBPropertyChangeAudit_EventTime	3,425	18.45
DBPropertyChangeAudit_Archive	IX_DBPropertyChangeAudit_Archive_EventTime	124,520	420.78

Maintenance completed successfully.

Generated by SQL Server DBA Automation System.

📜 Step 2: Test Job

Manually trigger it anytime:

```
EXEC msdb.dbo.sp_start_job 'Monthly_AuditRepo_Index_Compression_Maintenance';
```

Then confirm email delivery and log entries in the SQL Agent History tab.

⚙️ Pro Tips for Production

Recommendation	Description
🧠 **Compression Level:** Use ROW (default) for minimal CPU overhead; use PAGE only if archive size is huge and mostly read-only.	
🕒 **Schedule:** Run early Sunday morning — low load time.	
🔐 **Least Privilege:** Run under a dedicated SQLAgentOperator proxy account.	
📊 **Retention:** Optionally purge archive older than 2 years quarterly.	

✅ Create a **Quarterly Archive Purge Job** for DBPropertyChangeAudit_Archive to complete your **fully automated audit lifecycle**.
This ensures your repository never grows indefinitely while retaining compliance data for 2 years.

⚙ Job Overview – “Quarterly_AuditRepo_Archive_Purge”

Property	Value
Job Name	Quarterly_AuditRepo_Archive_Purge
Database	DBA_Repository
Schedule	Quarterly, 1st Sunday of Jan, Apr, Jul, Oct at 3:00 AM
Task	Delete/archive data older than 2 years and send email confirmation
Mail Profile	SQLDBA_Profile

🧱 Step 1: SQL Agent Job Script

```
USE msdb;
GO

-- Create Job
EXEC sp_add_job
   @job_name = N'Quarterly_AuditRepo_Archive_Purge',
   @enabled = 1,
   @description = N'Quarterly purge of audit archive older than 2 years with email confirmation',
   @category_name = N'Database Maintenance';
GO

-- Add Job Step
EXEC sp_add_jobstep
   @job_name = N'Quarterly_AuditRepo_Archive_Purge',
   @step_name = N'Purge Old Audit Archive Records',
   @subsystem = N'TSQL',
   @database_name = N'DBA_Repository',
   @command = N'
DECLARE @StartTime DATETIME = GETDATE();
DECLARE @PurgeDate DATETIME = DATEADD(YEAR, -2, GETDATE());
DECLARE @RowsDeleted INT;

-- Delete old records
DELETE FROM dbo.DBPropertyChangeAudit_Archive
WHERE EventTime < @PurgeDate;

SET @RowsDeleted = @@ROWCOUNT;

-- Send email confirmation
EXEC msdb.dbo.sp_send_dbmail
   @profile_name = ''SQLDBA_Profile'',
   @recipients = ''sqlalerts@yourcompany.com;dba_team@yourcompany.com'',
   @subject = ''Quarterly Audit Archive Purge Completed - '' + CONVERT(NVARCHAR(20), @StartTime, 107),
   @body = ''Quarterly audit archive purge completed successfully. '' +
         ''Date of Purge Threshold: '' + CONVERT(NVARCHAR(20), @PurgeDate, 120) + ''\n'' +
         ''Number of rows deleted: '' + CAST(@RowsDeleted AS NVARCHAR(20)) + ''\n'' +
         ''Executed on Server: '' + @@SERVERNAME + ''\n'' +
         ''Job executed at: '' + CONVERT(NVARCHAR(20), @StartTime, 120);
',
   @retry_attempts = 1,
   @retry_interval = 5;
GO

-- Add quarterly schedule (1st Sunday of Jan, Apr, Jul, Oct at 3:00 AM)
EXEC sp_add_schedule
   @schedule_name = N'Quarterly_1st_Sunday_3AM',
   @freq_type = 8,              -- Weekly
   @freq_interval = 1,          -- Sunday
   @freq_recurrence_factor = 13, -- Approx every 13 weeks (quarterly)
   @active_start_time = 30000;   -- 03:00 AM
GO

EXEC sp_attach_schedule
   @job_name = N'Quarterly_AuditRepo_Archive_Purge',
   @schedule_name = N'Quarterly_1st_Sunday_3AM';
GO

EXEC sp_add_jobserver
   @job_name = N'Quarterly_AuditRepo_Archive_Purge';
GO
```

🧪 Step 2: Test Purge Job

Manually run:

```
EXEC msdb.dbo.sp_start_job 'Quarterly_AuditRepo_Archive_Purge';
```

Check your email for the **confirmation report**, including:

- Purge threshold date
- Number of rows deleted
- Server name
- Job execution timestamp

⚙️ Step 3: Optional Enhancements

Enhancement	Description
📁 **Archive Before Delete**	Move old records to a separate “long-term” archive table for compliance instead of deleting.
🔄 **Logging**	Log each quarterly purge into a DBA_Repository.dbo.PurgeAuditLog table.
🔔 **Alert on Zero Deletes**	Send a warning if @@ROWCOUNT = 0 to ensure job ran properly.
📊 **Monthly Statistics**	Add a query to summarize table growth, helping capacity planning.

✅ With this, your **SQL Server DBA Audit System** is fully automated:

1. **Real-time trigger** → logs ALTER/CREATE/DROP DATABASE property changes
2. **Daily report job** → emails daily change summary
3. **Monthly maintenance job** → rebuilds indexes, applies row compression, sends maintenance report
4. **Quarterly purge job** → cleans archive older than 2 years and confirms by email

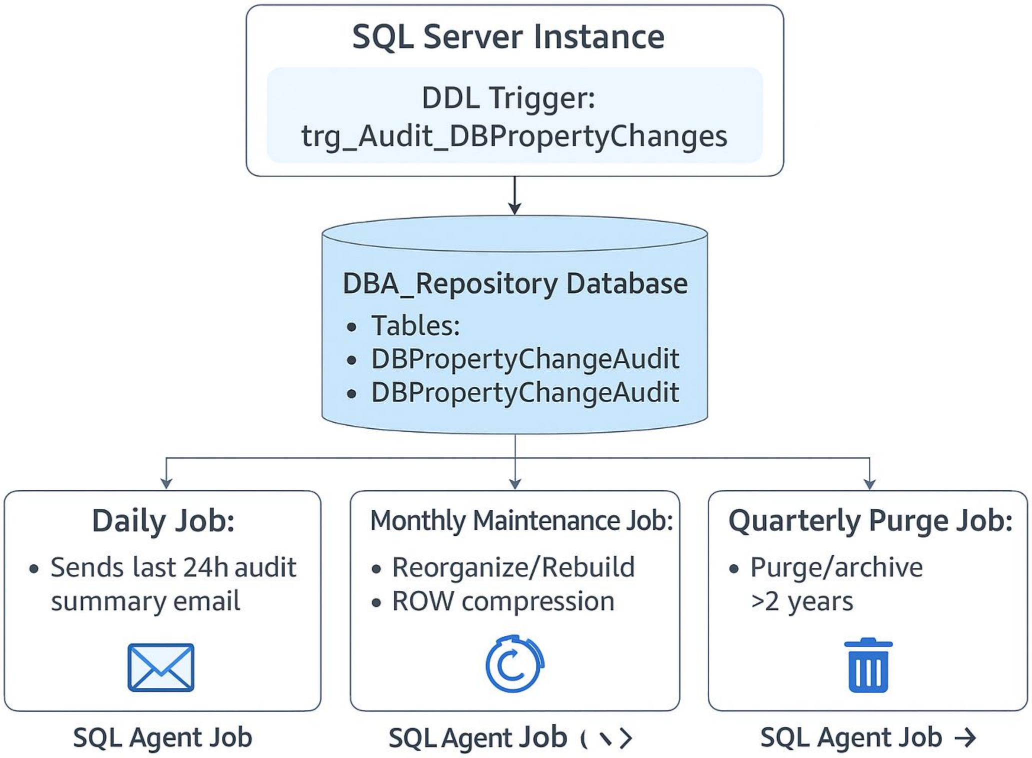
✅ **Visual architecture diagram** for your **SQL Server DBA Audit & Maintenance Lifecycle**. This will make it crystal clear for management, your team, or documentation.

🖼️ SQL Server DBA Audit & Maintenance Lifecycle Diagram

```
┌─────────────────────────────┐
|     SQL Server Instance     |
| ┌─────────────────────────┐ |
| | DDL Trigger:            | |
| | trg_Audit_DBPropertyChanges|
| | - Fires on ALTER/CREATE/ | |
| |   DROP DATABASE         | |
| | - Logs changes to       | |
| |   DBA_Repository        | |
| | - Sends immediate email  | |
| └─────────────────────────┘ |
└─────────────┬───────────────┘
              |
              ▼
┌─────────────────────────────┐
|   DBA_Repository Database   |
| ┌─────────────────────────┐ |
| | Tables:                 | |
| | - DBPropertyChangeAudit | |
| | - DBPropertyChangeAudit_Archive |
| └─────────────────────────┘ |
└─────────────┬───────────────┘
              |
              ▼
┌─────────────────────────────┐
| SQL Agent Jobs              |
| ┌─────────────────────────┐ |
| | Daily Job:              | |
| | - Sends last 24h audit  | |
| |   summary email         | |
| └─────────────────────────┘ |
| ┌─────────────────────────┐ |
| | Monthly Maintenance Job: | |
| | - Reorganize/Rebuild    | |
| | - ROW compression       | |
| | - Sends maintenance email||
| └─────────────────────────┘ |
| ┌─────────────────────────┐ |
| | Quarterly Purge Job:    | |
| | - Purge/archive >2 years | |
| | - Sends confirmation email | |
| └─────────────────────────┘ |
└─────────────────────────────┘
```

🔷 Workflow Summary

1. **Real-time Logging**
   - Trigger logs every database property change to DBA_Repository.
   - Immediate email alert to DBA team.
2. **Daily Monitoring**
   - SQL Agent job sends **daily HTML summary** of last 24 hours.
3. **Monthly Maintenance**
   - SQL Agent job rebuilds indexes and applies row compression on both live and archive tables.
   - Sends **monthly maintenance report** email.
4. **Quarterly Archive Purge**
   - SQL Agent job deletes or archives records older than 2 years.
   - Sends **quarterly confirmation email**.
5. **Central Repository Advantages**
   - All changes from multiple SQL instances can be centralized.
   - Easy reporting, audit, and compliance tracking.
   - Storage optimized via compression and purging.


SQL Server Instance
DDL Trigger:
trg_Audit_DBPropertyChanges
DBA_Repository Database
• Tables:
• DBPropertyChangeAudit
• DBPropertyChangeAudit
Daily Job:
• Sends last 24h audit summary email
SQL Agent Job
Monthly Maintenance Job:
• Reorganize/Rebuild
• ROW compression
SQL Agent Job (↘ >
Quarterly Purge Job:
• Purge/archive >2 years
SQL Agent Job →