ALTER PROCEDURE [dbo].[Stp\_SearchIndex\_UpdateEntity]

(

@IndexID bigint,

@IndexPath varchar(260),

@GroupID int,

@Type int,

@LeaseSeconds int,

@Active bit,

@DocCount int = 0,

@IsLocked bit = 0,

@LastUpdated datetime = getdate,

@SizeInMB decimal=null

)

AS

DECLARE @currentActive BIT

DECLARE @curdoccount INT

SELECT @currentActive=active,@curdoccount=DocCount

FROM searchindex (nolock)

WHERE indexid = @IndexID

IF (@currentActive<>@Active OR @curdoccount<>@DocCount)

BEGIN

SET @LastUpdated=GETDATE()

END

ELSE

BEGIN

SET @LastUpdated=NULL

END

UPDATE searchindex

SET

indexpath = @IndexPath,

groupnum = @GroupID,

[Type] = @Type,

LeaseSeconds = @LeaseSeconds,

active = @Active,

DocCount = @DocCount,

IsLocked = @IsLocked,

lastUpdate = ISNULL(@LastUpdated, lastUpdate)

SizeInMB = ISNULL(@SizeInMB,SizeInMB)

WHERE

indexid = @IndexID

CREATE PROC [dbo].[Stp\_IndexLockQueueItem\_GetIndexToProcessByApp](@AppType TINYINT)

AS

BEGIN

--Check if groups are prioritized or service/agent turned off

--==================================

DECLARE @Group TABLE (GroupID INT)

INSERT @Group(GroupID)

SELECT GroupID FROM SQLManagement.dbo.FN\_ServiceControl(5,NULL)

IF @@ROWCOUNT <> 0

BEGIN

IF EXISTS (SELECT 1 FROM @Group WHERE GroupID=0) --Service or Agent turned off

BEGIN

WAITFOR DELAY '00:00:05'

RETURN

END

END

-- select all the IndexLockQueueItems not expired

DECLARE @TblQueue TABLE(LockQueueID INT,GroupID INT, IndexID INT,OldIndexID INT, DateExpired DATETIME,IsComplete BIT,Step TINYINT)

INSERT INTO @TblQueue(LockQueueID, GroupID, IndexID, OldIndexID, DateExpired,IsComplete,Step)

SELECT A.LockQueueID,

A.GroupID,

ISNULL(B.NewSearchIndexID,B.SearchIndexID),

B.SearchIndexID,

A.DateExpired,

ISNULL(B.IsComplete,0),

A.Step

FROM IndexLockRequestQueue A (NOLOCK)

LEFT JOIN IndexLockRequestQueueItem B (NOLOCK) ON A.LockQueueID=B.LockQueueID

WHERE A.AppType=@Apptype

AND A.DateExpired>GETDATE()

--=================================

DECLARE @PriotizeType BIT

IF EXISTS (SELECT 1 FROM @Group WHERE GroupID<0)

SET @PriotizeType=0--Depriotize

ELSE IF EXISTS (SELECT 1 FROM @Group WHERE GroupID>0)

SET @PriotizeType=1--Priotize

--delete groups, which is not in retriction or which are in restriction

DELETE FROM @TblQueue

WHERE

@PriotizeType = 1 AND GroupID NOT IN (SELECT GroupID FROM @Group) OR

@PriotizeType = 0 AND GroupID IN (SELECT -1\*GroupID FROM @Group)

IF (NOT EXISTS(SELECT 1 FROM @TblQueue)) RETURN

DECLARE @Tbl TABLE(IndexId INT, GroupID INT)

UPDATE TOP (1) searchindex SET IsLocked=1,DateLeased=GETDATE(),AppType=@AppType, BatchGUID=NEWID()

OUTPUT inserted.indexid,inserted.groupnum INTO @Tbl

WHERE active=1 AND BatchGUID IS NULL AND IsLocked=0

AND EXISTS(

-- for the first attemp only update the searchindex table with index id

-- exist in our lockqueue items

SELECT 1 FROM @TblQueue A

WHERE A.IndexId=searchindex.indexid

AND A.IsComplete=0

AND A.Step>0

)

AND NOT EXISTS

(

-- exclude any group that is expired , need to be removed ! previous exist is enough

-- if there a case we will have 2 IndexlockQueue records for the group one expired and one not (this will prevent the group from processing)

~~SELECT 1 FROM IndexLockRequestQueue(NOLOCK) WHERE Step=0 and GroupId=searchindex.groupnum AND DateExpired>GETDATE() AND AppType=@AppType~~

)

IF @@ROWCOUNT = 0

BEGIN

-- if we couldn't locked any unlocked searchindex belonging to our Index Lock Queue items

-- try now to get any index that is already locked by retention

UPDATE TOP (1) searchindex SET IsLocked=1,DateLeased=GETDATE(),AppType=@AppType, BatchGUID=NEWID()

OUTPUT inserted.indexid,inserted.groupnum INTO @Tbl

WHERE active=1 AND BatchGUID IS NULL AND IsLocked=1 AND Apptype=@Apptype AND

EXISTS(

-- and the group id exist in our lock queue (request created for this group to be retention deleted )

SELECT 1 FROM @TblQueue A

WHERE A.GroupID=searchindex.groupnum

)

AND NOT EXISTS

(

-- exclude any group that is expired , need to be removed ! previous exist is enough

-- if there a case we will have 2 IndexlockQueue records for the group one expired and one not (this will prevent the group from processing)

~~SELECT 1 FROM IndexLockRequestQueue(NOLOCK) WHERE Step=0 and GroupId=searchindex.groupnum AND DateExpired>GETDATE() AND AppType=@AppType~~

)

END

IF (EXISTS(SELECT 1 FROM @Tbl))

BEGIN

DECLARE @LockQueueID INT

DECLARE @IndexID INT

DECLARE @OldIndexID INT

DECLARE @GroupID INT

SELECT TOP 1 @IndexID=IndexId,@GroupID=GroupID FROM @Tbl

SELECT TOP 1 @LockQueueID=A.LockQueueID,@OldIndexID=A.OldIndexID

FROM @TblQueue A

WHERE A.GroupID=@GroupID

AND A.IsComplete=0

AND (A.IndexID=@IndexID OR A.OldIndexID IS NULL)

ORDER BY A.IndexID DESC,A.DateExpired ASC

--prolong dateexpired in IndexLockRequestQueue table for two hours, because you are going to work on it. Hopefully you will finish for that time

UPDATE IndexLockRequestQueue

SET DateExpired=DATEADD(hour,2,DateExpired)

WHERE LockQueueID=@LockQueueID AND DateExpired>GETDATE() AND DateExpired<DATEADD(hour,2,GETDATE())

SELECT @LockQueueID AS LockQueueID,@OldIndexID AS OldIndexID, indexid,indexpath,groupnum,[Type],LeaseSeconds,active,DocCount,IsLocked,lastUpdate,SizeInMB,BatchGUID,DateLeased,AppType

FROM searchindex (NOLOCK)

WHERE indexid=@IndexID

END

END

For this we always pass IndexId to this stored proc ,null is for backward compatibility

ALTER PROCEDURE [dbo].[Stp\_RetentionBatchMailQueue\_GetToProcess]

(

@MaxRows int,

@AgentID varchar(50),

@ProcessStep int,

@ExpireMins int,

@LockTimeoutMsecs varchar(50) = '1000',

@LockResult int = 0 OUT,

@DebugMode bit = 0,

@IndexID INT = NULL

)

AS

DECLARE @tbl TABLE (RetentionMailQueueID bigint)

DECLARE @lockname varchar(50)

DECLARE @committed bit

SET @committed = 0

-- Lock this section so that no other agents can grab the same rows to process

IF @DebugMode = 1 PRINT 'Acquiring lock...' + CONVERT(VARCHAR, CURRENT\_TIMESTAMP, 14)

SELECT @lockname = 'RetentionBatchMailQueue' + CONVERT(VarChar, @ProcessStep) + 'AppLock'

-- For step 1, we only want to returns all queues for a particular IndexID

IF @ProcessStep = 1

BEGIN

IF @IndexID IS NULL

BEGIN

SELECT TOP 1 @IndexID=SearchIndexID

FROM RetentionBatchMailQueue (NOLOCK)

WHERE ProcessStep=1 AND StepInProgress=0 AND SearchIndexID IS NOT NULL

ORDER BY NEWID()

END

IF @IndexID IS NULL

RETURN

SET @lockname = 'RetentionBatchMailQueue' + CONVERT(VarChar, @ProcessStep) + '\_' + CONVERT(VARCHAR(20),@IndexID) + 'AppLock'

END

BEGIN TRANSACTION;

EXEC @LockResult = sp\_getapplock @Resource=@LockName, @LockMode='Exclusive', @LockOwner='Transaction', @LockTimeout=@LockTimeoutMsecs

PRINT @LockResult

-- If lock was acquired...

IF @LockResult IN (0, 1)

BEGIN

-- Get the rows to process

IF @DebugMode = 1 PRINT 'Lock acquired, selecting rows...' + CONVERT(VARCHAR, CURRENT\_TIMESTAMP, 14)

-- @MaxRows never used do we remove it , do we need to use it ?

IF @ProcessStep <> 1

SET ROWCOUNT @MaxRows

-- if step inprogress stuck , what will reset it ?

-- if is failed =1 what will reset it ? nothing will move to step 2 after 24 hours

-- if process count = 5 what will reset it? Nothing will move to step2 after 24 hours

INSERT INTO @tbl (RetentionMailQueueID)

SELECT a.RetentionMailQueueID FROM RetentionBatchMailQueue a with (NOLOCK)

WHERE

a.ProcessStep = @ProcessStep AND

a.StepInProgress = 0 AND

a.IsFailed = 0 AND

a.ProcessCount < 5 AND

(@IndexID IS NULL OR a.SearchIndexID = @IndexID)

SET ROWCOUNT 0

-- Set the processing flags

IF @DebugMode = 1 PRINT 'Rows selected, updating process flags...' + CONVERT(VARCHAR, CURRENT\_TIMESTAMP, 14)

-- Redundant code?

~~IF @ProcessStep = 1~~

~~BEGIN~~

~~UPDATE RetentionBatchMailQueue~~

~~SET AgentID = @AgentID,~~

~~StepInProgress = 1,~~

~~ProcessCount = ProcessCount + 1,~~

~~DateStarted = getdate(),~~

~~ProcessDate = getdate(),~~

~~ProcessExpDate = DATEADD(MINUTE, @ExpireMins, getdate())~~

~~WHERE~~

~~RetentionMailQueueID IN (SELECT RetentionMailQueueID FROM @tbl)~~

~~END~~

~~ELSE~~

~~BEGIN~~

UPDATE RetentionBatchMailQueue

SET AgentID = @AgentID,

StepInProgress = 1,

ProcessCount = ProcessCount + 1,

DateStarted = getdate(),

ProcessDate = getdate(),

ProcessExpDate = DATEADD(MINUTE, @ExpireMins, getdate())

WHERE

RetentionMailQueueID IN (SELECT RetentionMailQueueID FROM @tbl)

~~END~~

-- Remove the lock

EXEC sp\_releaseapplock @lockname, 'Transaction'

COMMIT TRANSACTION;

SET @committed = 1

END

IF @committed = 0

ROLLBACK TRANSACTION;

-- Return the rows

IF @DebugMode = 1 PRINT 'Returning results...' + CONVERT(VARCHAR, CURRENT\_TIMESTAMP, 14)

SELECT \* FROM RetentionBatchMailQueue(NOLOCK) WHERE RetentionMailQueueID IN (SELECT RetentionMailQueueID FROM @tbl)

Stp\_RetentionBatchMailQueue\_ResetExpired

For this stored procedure we only delete the Items that stuck for more than 22 hours

In processtep=1 and stepinprogress =0 and failed=0 and processcount=0

The remaining logic of :

1. Reset items that is stuck in stepinprogress=1 and processed less than 5 times and not failed for more than 60 minutes (#ToReset)
2. Move items processed more than 5 times and stuck to Fail (#ToReset)
3. Move the items that is in Fail state for more than 24 hours to Step 2 so customer doesn’t complain and think we deleted them , **why we are waiting 24 hours? , to give OPS time to investigate these items , if not why we don’t move them directly to Step =2 cause we will never reset or delete the items in fail state**