**Index Rebuilding Tool new Flow**

1. create a request lock available L3 indexes , insert the locked

one by other apps to waiting queue **IndexLocRequestQueueItem**

1. Make Sure SQL job that reset L3 indexes locked exclude the ones that is locked by your AppType 5 (Index Rebuilding)
2. keep waiting till all the indexes in your queue is locked for

your AppType 5 (Index Rebuilding)

Stp\_IndexLockRequestQueueItem\_AllDone

Stp\_IndexLockQueueItem\_GetIndexesToProcess

1. now get the max end mail id belong to the

list of all the L3 indexes own and locked by

apptype 5 (Index Rebuilding).

1. get the **mails expired** from Retention DB for that group
2. rebuild the ranges for this group in Rebuilding DB excluding

**mails** **expired** by Retention.

1. Make sure Remerge service (which is merge) configured with AppType remerge , and modify these stored procs:

stp\_SearchIndex\_GetLevel3IndexBestCandidateAndLock

stp\_SearchIndex\_GetNewIndexPath

make sure to switch the IndexType to 3 if AppType is Remerge (2) and any update or insert should use IndexType.

1. wait till all the record merged step 4 in SearchIndexRebuildingDB.MailsToIndex for that group
2. delete MailsToIndex Records for this group in SearchIndexingDB.MailsToIndex less than LastMail .
3. Insert all the records from SearchIndexRebuildingDB.MailsToIndex to SearchIndexingDB.MailsToIndex for target group
4. Set complete all the indexes that you AppType 5 waiting on. Because these indexes will be deleted set anything waiting on these indexes to complete so it not processed by other apps.
5. deactivate L3 Indexes that islocked by AppType 5 (indexRebuilding) for that group (SearchManager/DB)

Update any Index of IndexType 3 to IndexType 2 so they become searchable in one transaction or no !?

TODO : Ask Sahiti how the activation and de activation for big indexes work ? do we do it on search Restart , especially for big groups.

1. delete all L3 indexes files/folders that islocked by AppType

5 **indexRebuilding** for that group

1. delete all the records islocked by AppType 5 indexrebuilding

for that group

**stored Procedure changes**

**Search DB**

USE [Search]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[stp\_SearchIndex\_GetNewIndexPath] Script Date: 6/9/2014 6:34:13 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[stp\_SearchIndex\_GetNewIndexPath]

@Group\_id INT,

@IndexType TINYINT,

@AppType TINYINT,

@LeaseSec INT,

@BatchGuid UNIQUEIDENTIFIER,

@IndexPathEmpty varchar(260)

AS

BEGIN

DECLARE @LockResult INT

DECLARE @MaxLevelNum INT

DECLARE @IndexId INT

DECLARE @IndexPath VARCHAR(260)

IF @AppType = 2 -- ReMerge service

BEGIN

SET @IndexType = 3 -- the index type that belong to ReMerge service

END

SELECT TOP 1 @IndexPath=indexpath,@IndexId=indexid FROM searchindex (NOLOCK)

WHERE groupnum = @Group\_id and active=0 and IsLocked=0 and DocCount=0

--if no indexpath found select any

IF (ISNULL(@IndexPath,'')='')

BEGIN

SET @IndexId=NULL

SELECT TOP 1 @MaxLevelNum=CONVERT(INT,REVERSE(SUBSTRING(REVERSE(indexpath),0,CHARINDEX('\', REVERSE(indexpath))))),

@IndexPath=@IndexPathEmpty

FROM searchindex (NOLOCK)

WHERE groupnum = @group\_id

ORDER BY CONVERT(INT,REVERSE(SUBSTRING(REVERSE(indexpath),0,CHARINDEX('\', REVERSE(indexpath))))) DESC

END

IF (ISNULL(@IndexPath,'')='')

BEGIN

SET @IndexPath=@IndexPathEmpty

SET @MaxLevelNum=0

END

IF (ISNULL(@IndexId,-1)=-1)

BEGIN

SET @MaxLevelNum=@MaxLevelNum+1

--because we have indexpath unique index, i don't use locking for some it will fail if both will receive same indexpath

IF @MaxLevelNum<10

SET @IndexPath=@IndexPath+'0'+CONVERT(VARCHAR(200),@MaxLevelNum)

ELSE

SET @IndexPath=@IndexPath+CONVERT(VARCHAR(200),@MaxLevelNum)

INSERT INTO searchindex(indexpath,groupnum,Type,LeaseSeconds,active,DocCount,IsLocked,lastUpdate,SizeInMB,BatchGUID,DateLeased,AppType)

VALUES (@IndexPath,@group\_id,@IndexType,@LeaseSec,0,0,1,GETDATE(),0,@BatchGuid,GETDATE(),@AppType)

SET @IndexId = SCOPE\_IDENTITY()

IF (@IndexId<=0)

SET @IndexPath=''

END

ELSE

BEGIN

--lock found levelnum record

UPDATE searchindex SET IsLocked=1,DateLeased=GETDATE(),BatchGUID=@BatchGuid,AppType=@AppType,LeaseSeconds=@LeaseSec,[Type]=@IndexType WHERE indexid=@IndexId and IsLocked=0

IF @@ROWCOUNT = 0

SET @IndexPath=''

END

IF ISNULL(@IndexPath,'')<>''

BEGIN

SELECT \* FROM searchindex(NOLOCK) WHERE indexid=@IndexId

END

END

USE [Search]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[stp\_SearchIndex\_GetLevel3IndexBestCandidateAndLock] Script Date: 6/9/2014 6:32:59 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[stp\_SearchIndex\_GetLevel3IndexBestCandidateAndLock]

@TotalDocCount INT,

@MaxIndexDocCount INT,

@GroupId INT,

@Type TINYINT,

@AppType TINYINT,

@MaxShards INT=4,

@BatchGuid UNIQUEIDENTIFIER

AS

BEGIN

DECLARE @Tbl TABLE(IndexId INT)

IF @AppType = 2 -- Remerge Service

BEGIN

SET @Type= 3 -- index type distinguish indexes created by remerge service.

END

DECLARE @Tbl\_IndexesToExclude TABLE(IndexId INT)

INSERT INTO @Tbl\_IndexesToExclude

SELECT DISTINCT ISNULL(NewSearchIndexID,SearchIndexID)

FROM IndexLockRequestQueueItem (NOLOCK) A INNER JOIN

IndexLockRequestQueue (NOLOCK) B ON A.LockQueueID=B.LockQueueID AND A.IsComplete=0

WHERE A.IsComplete=0 AND B.Step < 2 AND

GETDATE()<B.DateExpired

UPDATE searchindex SET IsLocked=1,BatchGuid=@BatchGuid,DateLeased=GETDATE(),AppType=@AppType

OUTPUT inserted.indexid INTO @Tbl

WHERE indexid IN

(

SELECT TOP (@MaxShards) indexid FROM searchindex (NOLOCK)

WHERE groupnum = @GroupId and TYPE=@Type and active=1 and IsLocked=0 and DocCount<=@MaxIndexDocCount-@TotalDocCount

ORDER BY DocCount DESC

)

AND NOT EXISTS(SELECT IndexId FROM @Tbl\_IndexesToExclude B WHERE B.IndexId=searchindex.IndexID)

AND IsLocked=0

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

FROM searchindex (NOLOCK)

WHERE indexid IN

(

SELECT indexid FROM @Tbl

)

END

Create PROC [dbo].[Stp\_IndexLockRequestQueueItem\_SetCompleteIndexForWaitingApps](@Indexes Tp\_SearchIndexEntityType READONLY)

AS

BEGIN

UPDATE IndexLockRequestQueueItem SET IsComplete=1,DateFinished=GETDATE() WHERE IsComplete=0

EXISTS(SELECT 1 FROM @Indexes A WHERE A.IndexID=ISNULL(NewSearchIndexID,SearchIndexID))

END

Create PROC [dbo].[Stp\_SearchIndex\_DeActivateRebuildedIndexes](@AppType TINYINT, @GroupID int)

AS

BEGIN

UPDATE SearchIndex SET Active=0

WHERE Active=1 AND IsLocked=1 and AppType=@AppType AND groupnum=@GroupID

END

Create PROC [dbo].[Stp\_SearchIndex\_ActivateReMergedIndexes](GroupID INT )

AS

BEGIN

UPDATE SearchIndex SET [Type]=2

WHERE Active=1 AND [Type]=3 AND groupnum=@GroupID

END

Create PROC [dbo].[Stp\_SearchIndex\_DeleteRebuildedIndexes](@AppType TINYINT ,@GroupID INT )

AS

BEGIN

delete from SearchIndex

WHERE IsLocked=1 AND [Type]=2 AND AppType=@AppType AND groupnum=@GroupID

END

**SearchIndexing DB**

Create PROC [dbo].[Stp\_MailsToIndex\_GetLastMailIDForIndexes]

(

@Indexes Tp\_SearchIndexEntityType READONLY

)

AS

BEGIN

SELECT MAX(EndMailID)

from MailsToIndex (nolock)

where IndexID in

(select IndexID from @Indexes)

END

CREATE PROC [dbo].[Stp\_MailsToIndex\_DeleteRemergedRanges]

(

@LatMailIdRemerged BIGINT,

@GroupID int

)

AS

BEGIN

      DECLARE @i INT=1

      WHILE @i>0

      BEGIN

            BEGIN TRANSACTION

                  DELETE TOP(1000)

                  FROM MailsToIndex

                  WHERE GroupID= @GroupID

                  AND EndMailID <=@LatMailIdRemerged

                  SET @i = @@ROWCOUNT;

            COMMIT TRANSACTION

      END

END

Create PROC [dbo].[Stp\_MailsToIndex\_InsertRemergedRanges]

(

      @GroupID INT

)

AS

BEGIN

DECLARE @UID BIGINT

DECLARE @rootnum BIGINT,@startmailid BIGINT,@endmailid BIGINT

DECLARE @isatt BIT,@stepinprogress BIT,@failed BIT

DECLARE @agentid INT,@step INT,@processcount INT,@expectedcount INT,@successcount INT,@indexid INT,@excludecount INT, @retrycount INT,@skipcount INT,@failcount INT,@grpid INT

DECLARE @path VARCHAR(256)

DECLARE @dateadded DATETIME,@datestarted DATETIME,@datefinished DATETIME

DECLARE @delaymin TINYINT,@actiontype TINYINT

DECLARE @btchguid UNIQUEIDENTIFIER

DECLARE db\_Cursor CURSOR FOR

SELECT

       [RootNum] ,[StartMailID] ,[EndMailID] ,[IsAtt] ,[Failed],[Step],[DelayMin],[GroupID],[ActionType],

       [StepInProgress],[DateAdded],[ProcessCount],

       [AgentID] ,[Path],[DateStarted] ,[DateFinished] ,[ExpectedCount] ,[SuccessCount] ,[IndexID] ,[ExcludeCount] ,[RetryCount] ,[SkipCount] ,[FailCount] ,[BatchGUID]

FROM SearchIndexingRebuilding..MailsToIndex (NOLOCK)

WHERE GroupID = @GroupID

OPEN db\_Cursor

FETCH NEXT FROM db\_Cursor INTO @rootnum,@startmailid,@endmailid,@isatt,@failed,@step,@delaymin,@grpid,@actiontype,

                                             @stepinprogress,@dateadded,@processcount,

                                             @agentid,@path,@datestarted,@datefinished,@expectedcount,@successcount,@indexid,@excludecount, @retrycount,@skipcount,@failcount,@btchguid

WHILE @@FETCH\_STATUS = 0

BEGIN

            UPDATE MailsToIndexIdentity

            set [UID]=[UID]+1,

                 @UID=[UID]+1

            INSERT INTO MailsToIndex(

                                     [Uid],RootNum, StartMailID, EndMailID, IsAtt, Failed, Step, DelayMin, GroupID, ActionType,

                                 StepInProgress,DateAdded,ProcessCount,

                                 Agentid,[Path],DateStarted,DateFinished,ExpectedCount,SuccessCount,IndexID,ExcludeCount,RetryCount,SkipCount,FailCount,BatchGUID

                                )

                                    SELECT

                               @UID,@rootnum,@startmailid,@endmailid,@isatt,@failed,@step,@delaymin,@grpid,@actiontype,

                                             @stepinprogress,@dateadded,@processcount,

                                             @agentid,@path,@datestarted,@datefinished,@expectedcount,@successcount,@indexid,@excludecount, @retrycount,@skipcount,@failcount,@btchguid

            FETCH NEXT FROM db\_cursor INTO @rootnum,@startmailid,@endmailid,@isatt,@failed,@step,@delaymin,@grpid,@actiontype,

                                                         @stepinprogress,@dateadded,@processcount,

                                                     @agentid,@path,@datestarted,@datefinished,@expectedcount,@successcount,@indexid,@excludecount, @retrycount,@skipcount,@failcount,@btchguid

END

CLOSE db\_cursor

DEALLOCATE db\_cursor

END

**SearchIndexing DB**

**GetExpired Mails for a group , will be implemented.**

select \* from RetentionBatchMail (nolock) A

inner join RetentionBatch (nolock) B on A.BatchID=B.BatchID

where A.IndexDeleted=1 and B.GroupID=@groupid

but it can be huge amount of data, it can billions records

select MailID from RetentionPolicy..RetentionBatchMail (nolock)

where BatchID in (select BatchID from RetentionPolicy..RetentionBatch (nolock)

where GroupID = <>)

this will give you all mailids which have been collected

including the one whch have not been deleted and are sitting in the queue

select MailID from RetentionPolicy..RetentionBatchMail (nolock)

where BatchID in (select BatchID from RetentionPolicy..RetentionBatch (nolock)

where GroupID = 346 and ExecutionStatus > 1)

this will exclude mails which are collected but not ready to be deleted

Create PROC [dbo].[Stp\_MailsToIndex\_GetExpiredMailIDs]

(

@GroupID INT,

@LastExpiredMailID BIGINT,

@MaxRows INT

)

AS

BEGIN

select TOP (@MaxRows) MailID

from RetentionBatchMail (nolock)

where

MailID > @LastExpiredMailID

AND BatchID in

(select BatchID from RetentionBatch (nolock)

where GroupID = @GroupID)

END