# SEMINAR 7 - 7th January 2022 (group 927)

## Indexes in SQL SERVER (II)

Index recommendations:

* Every table should have a cl index
* The cl index as small as possible, selective, increasing
* Ncl indexes on foreign keys and on columns frequently used in WHERE
* Covering indexes on the most frequently used queries
* NO: index/col
* Multi-col index: put first the most selective column

A – table

Create view viewA

As

Select \* from A

Insert into A ….

Select \* from viewA

* Views do not hold the data or does not physically exist on the disk
* We can create indexes just on materialized views
* Restrictions:
* 1. Options:
  + ANSI\_NULLS ON
* Column is NULL (on) vs. column = NULL (off)
  + ANSI\_PADDING ON

ON: add spaces varchar(3) ‘ab\_’

Off: trailing whitespaces are removed

* + ANSI\_WARNINGS ON

ON: SUM, MIN, MAX on nullable fields/columns => warning

Division by 0 – error

The length of the new value exceeds the maximum length of the column (varchar(4) insert ‘abcde’ => warning)

OFF - SUM, MIN, MAX on nullable fields/columns

Division by 0 – warning

The length of the new value exceeds the maximum length of the column (varchar(4) insert ‘abcde’ => warning) => truncate

* + ARITHABORT ON
* ON: Arithmetic overflows – if any errors -> statement aborted
* OFF: returns null
  + CONCAT\_NULL\_YIELDS\_NULL ON
* ON: NULL + ‘abc’ => NULL
* OFF: NULL + ‘abc’ => ‘abc’
  + QUOTED\_IDENTIFIER ON
* ON: object names “”
* OFF: ‘ or “
* On: Create table “having” => ok
* OFF: error
  + NUMERIC\_ROUNDABORT OFF
* OFF: no warnings if there are any truncations concerning precision DECIMAL(7, 3)
* ON: warnings
* 2. View restrictions:
* Not allowed: AVG, MIN, MAX, STDEV, VAR
* COUNT -> COUNT\_BIG
* SUM -> used just on non-nullable colums

Select is\_nullable

From sys.columns

Where name= ‘ColumnName’ and object\_id=OBJECT\_ID(‘TAblename’)

SUM(ISNULL(‘ColumnName’, 0))

* Can’t reference another view
* NO: subqueries in the FROM clause, HAVING, OUTER JOIN, TOP, DISTINCT, ORDER BY, UNION, INTERSECT, EXCEPT ….
* Can’t use functions which are not deterministic, ex: GETDATE
* Names of objects must be written in a two part format: dbo.Tablename
* Use SCHEMABINDING

Create view view\_2

With **schemabinding**

As

Select col1

From dbo.table1

Drop table table1 =>> error

When creating a clustered unique index on a view, that view is getting materialized

After this, we can create as well some non-cl indexes

Select \*

From viewName

WITH (NOEXPAND) -🡪 sql will use indexes on the view, not the underlying indexes of the tables

WHEre …..

Warnings -> severity 0 ..18 -> handled in try catch if the severity is above 10.

FRAGMENTATION:

1. Internal
   1. Records are not stored in a **contiguous** area of memory (there may be some unused space between)
   2. The fullness of a page depends in time
   3. Unused page => cache is not efficiently used and more pages transfers between the disk and the main memory
2. Extent

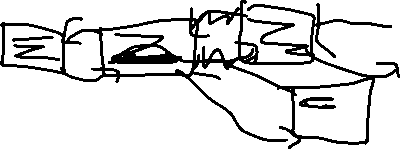
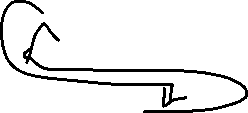
* Extent: group of 8 contiguous pages
* Table’s extent are not contiguous stored

1. Logical

* Every indexed page has a reference to the next page and the previous one based on the key index



* Page split (example: inserting a value)
* Out-of order page



Examples:

Read operation – page requests: 2

Extent switches: 0

Disk space (one page is 8KB): 16KB

Avg\_fragmentation\_in\_percent: 0

AVgpage\_space\_used\_in\_percent: 100

## Capture

Read operation – page requests: 6

Extent switches: 5

Disk space (one page is 8KB): 48KB

Avg\_fragmentation\_in\_percent > 80

AVgpage\_space\_used\_in\_percent: 33

## Capture1

Sys.dm\_db\_index\_physical\_stats

* Avg\_fragmentation\_in\_percent
  + Heaps: extent fragmentation
  + Indexes: logical fragmentation
* Avg\_page\_space\_used\_in\_percent:
  + Avg of the percentages of all the available space in all pages

Heap-table without a cls index, stored in no particular order

Reduce fragmentation:

* Heap:
  + Create a cls index on it and the drop it
  + Creating the cl index => the records will get redistributed
* Indexed table:
  + If Avg\_fragmentation\_in\_percent > 5% and <=30%:
    - Reorder the leaf pages of the index: ALTER iNDEX nameindex REORGANIZE
  + If Avg\_fragmentation\_in\_percent > 30 %:
    - Rebuild the index
    - Drop and recreating the index
* Drop and recreate the index => all the data is going to be redistributed into full data pages
* CREATE INDEX indexname

WITH (FILLFACTOR = 70) ….

## T-SQL Control of Flow Language

* BEGIN…END
* BREAK
* CONTINUE  
  RETURN
* WHILE
* THROW
* IF … ELSE
* TRY … CATCH
* WAITFOR  
  GOTO label

Label:

GOTO Label

RETURN {interger\_expression}

Exists from a stored procedure/ batch (GO separated by GO) /statement block (begin and end)

System stored procedures :

0 – success

Non-0 : failure

WAITFOR {delay ‘time\_to\_pass’ | TIME ‘time\_to execute’}

Example: WAITFOR TIME ’07:15’

WAITFOR DELAY ‘3:00’

THROW [error\_number, message, state]

Ex: THROW 51000, ‘message’, 1;

Error\_number >=50000

Message: nvarchar(2048)

State 0-255

Severity = 16

RAISERROR – specify 26 severity level

Severity 20 -25 => fatal error & the db connection is closed

Severity >=16 => errors are logged automatically

RAISERROR(….) WITH LOG

TRY CATCH

BEGIN TRY

…..

END TRY

BEGIN CATCH

….

END CATCH

* Handles execution errors with a severity >=10 and which do not close the db connection

ERROR\_NUMBER()

ERROR\_SEVERITY()

ERROR\_STATE()

ERROR\_PROCEDURE() ----

ERROR\_LINE()

ERROR\_MESSAGE()