

Automated Tuning and Partitioned Tables

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:28 AM PST

Contents

- [Issue summary](#)
- [More Information](#)
 - [Regarding Automatic Index Create](#)
 - [Regarding Automatic Index Drop](#)
- [Public Doc Reference](#)

This TSG summarizes the dependencies between Automatic Tuning and partition switches when using table partitioning

Issue summary

When you are using the partition switch feature to load or remove data into/from partitioned tables, the schema of the partitioned table and the stage table must be identical. If you are considering using Automatic Index Create or Automatic Index Drop for your Azure SQL Database, you might be worried about its consequences as it might take table schemas out of sync.

A partition switch between tables is an operation similar to the following:

```
-- create a partitioned table, assuming that partition scheme myRangePS1 is already created in the database
CREATE TABLE PartitionTable (col1 INT, col2 CHAR(10))
ON myRangePS1 (col1);

-- create non-partitioned table with the same structure as the partitioned table
CREATE TABLE NonPartitionTable (col1 INT, col2 CHAR(10));

-- Switch the data of PARTITION 2 of table PartitionTable into NonPartitionTable
ALTER TABLE PartitionTable SWITCH PARTITION 2 TO NonPartitionTable;
```

More Information

Regarding Automatic Index Create

The auto-create index feature will add indexes for tables based on the database workload. It might however fail on tables that are partitioned or are used as partition stage table.

If the Automatic Tuning system detects that the index is causing a partition switch to fail, it will drop the index automatically.

You can also track auto-created indexes by reviewing the `sys.indexes` system table for the `auto_created` column that shows which indexes were created by automatic tuning:

```

SELECT
    ObjectName = OBJECT_NAME(i.object_id),
    ObjectSchema = OBJECT_SCHEMA_NAME(i.object_id),
    IndexName = i.name,
    i.index_id,
    i.type_desc,
    i.auto_created
FROM sys.indexes i
WHERE
    i.type IN (0 /*HEAP*/, 1/*CLUSTERED*/, 2/*NONCLUSTERED*/, 5/*CLUSTERED COLUMNSTORE*/, 6/*NONCLUSTERED COLU
    AND OBJECT_SCHEMA_NAME(i.object_id) != 'sys'
ORDER BY ObjectName, i.index_id

```

Sample output:




	ObjectName	ObjectSchema	IndexName	index_id	type_desc	auto_created
1	Address	Person	PK_Address_AddressID	1	CLUSTERED	0
2	Address	Person	AK_Address_rowguid	2	NONCLUSTERED	0
3	Address	Person	IX_Address_AddressLine1_AddressLine2_City_StatePr...	3	NONCLUSTERED	0
4	Address	Person	IX_Address_StateProvinceID	4	NONCLUSTERED	0
5	Address Type	Person	PK_AddressType_AddressTypeID	1	CLUSTERED	0
6	Address Type	Person	AK_AddressType_Name	2	NONCLUSTERED	0
7	Address Type	Person	AK_AddressType_rowguid	3	NONCLUSTERED	0
8	AWBuildVersion	dbo	PK_AWBuildVersion_SystemInformationID	1	CLUSTERED	0
9	BillOfMaterials	Production	AK_BillOfMaterials_ProductAssemblyID_ComponentID...	1	CLUSTERED	0

Regarding Automatic Index Drop

Automatic Index Drop is not compatible with application that uses partition switch, because it is unable to identify which indexes cannot be dropped due to its usage by partition switch. This feature might be automatically disabled when queries with index hints are present in the workload, or when the workload performs partition switching. The portal GUI will also warn you about this scenario.

In case you decided to enable Automatic Index Drop on a database that uses partition switch and the system detects that, it will be automatically set to OFF to protect your database from errors.

Public Doc Reference

- [Partition Switch](#) 
- [Automatic tuning in Azure SQL Database and Azure SQL Managed Instance](#) 
- [Automatic tuning options](#) 

How good have you found this content?

