10 T-SQL Index statements needed with DBA

SQL Server DBA (Database Administrator) people - database administrators know very well that Index entries in the database are very similar to Index in the Library section. Or simply understand that Index in Database is a structure that is closely linked to tables to quickly gather information from the rows in that table.

In the following article, we will show you the T-SQL statements related to Index in SQL that are very useful for any DBA person who does the job. Specifically, we will divide into 3 main categories: the concept of Index, create query statements - Information related queries, and finally the maintenance process - Maintenance.

General concept of Index:

1. Clustered Index:

The main task is to store the data of the rows arranged in the order in the table based on the value of the key key. Only one clustered index can be created on each table, because the data of the rows can only be arranged in a certain order. One more point is that clustered index can be 'generated' while creating constraints like the Primary key on the existing data table.

Eg:

```
ALTER TABLE [MyAddress]
ADD CONSTRAINT [PK_Address_AddressID] PRIMARY KEY CLUSTERED
(
[AddressID] ASC
) ON [PRIMARY]
GO
```

Besides, clustered index can also be created on each column without the associated link. For example:

```
CREATE CLUSTERED INDEX [MyAddress_id_CIX] ON [MyAddress1]
(
[ID] ASC
) ON [PRIMARY]
GO
```

2. Non Clustered Index:

Created to improve performance, the performance of query string sequences is frequently used, but not included with clustered indexes. Within the nonclustered index block, the ordering of information in the index of the index does not match the storage order in terms of the physical aspect of the data stream on the drive.

Nonclustered Indexes can be created on existing tables, including columns not in the clustered index. For example:

```
CREATE UNIQUE NONCLUSTERED INDEX
[NIX_col5_col2_col3_col4_col6]
ON [MyAddress]
(
[AddressLine1] ASC,
[AddressLine2] ASC,
[City] ASC,
[StateProvinceID] ASC,
[PostalCode] ASC
) ON [PRIMARY]
GO
```

Or, the nonclustered index can also be done while creating links in the existing table, for example:

```
ALTER TABLE [MyAddressType]
ADD CONSTRAINT [DEFF_MyAddressType_ModifiedDate]
DEFAULT (getdate ()) FOR [ModifiedDate]
GO
```

3. XML Index:

Another concept, generated on the XML data column and the clustered index on the Primary key. 1 XML index is Frimary-like:

```
CREATE PRIMARY XML INDEX idx_xCol_MyTable on MyTable (xCol)
```

And with the XML index secondary it is as follows:

```
CREATE TABLE MyTable (Col1 INT PRIMARY KEY, XmlCol XML)
GO
- Create primary index.
CREATE PRIMARY XML INDEX PIdx_MyTable_XmlCol
ON T (XmlCol)
GO
- Create secondary indexes (PATH, VALUE, PROPERTY).
CREATE XML INDEX PIdx_MyTable_XmlCol_PATH ON MyTable (XmlCol)
USING XML INDEX PIdx_MyTable_XmlCol
FOR PATH
GO
CREATE XML INDEX PIdx_MyTable_XmlCol_VALUE ON T (XmlCol)
USING XML INDEX PIdx_MyTable_XmlCol
FOR VALUE
GO
```

4. Spatial Index:

A component in SQL Server 2008 provides users with special data columns, related to symbol data related to the spatial domain, such as geography and geometry.

1 spatial index structure can be created by the following syntax:

CREATE TABLE MySpatialTable (primary id int, geometry_col geometry);
CREATE SPATIAL INDEX SIndx_MySpatialTable_geometry_col1
ON MySpatialTable (geometry_col)
WITH (BOUNDING_BOX = (0, 0, 500, 200));

Query Index related to metadata data:

5. Search all indexes:

First, if you want to search for all indexes, use the query statement - query by table, column and index key of the existing database.

SELECT OBJECT_SCHEMA_NAME (BaseT. [Object_id], DB_ID ()) AS [Schema],
BaseT. [Name] AS [table_name], I. [name] AS [index_name], AC. [Name] AS [column_name],
I. [type_desc]
FROM sys. [Tables] AS BaseT
INNER JOIN sys. [Indexes] I ON BaseT. [Object_id] = I. [object_id]
INNER JOIN sys. [Index_columns] IC ON I. [object_id] = IC. [Object_id]
INNER JOIN sys. [All_columns] AC ON BaseT. [Object_id] = AC. [Object_id] AND IC. [Column_id] = AC. [Column_id]
WHERE BaseT. [Is_ms_shipped] = 0 AND I. [type_desc] > 'HEAP'
ORDER BY BaseT. [Name], I. [index_id], IC. [Key_ordinal]

6. Fragmentation:

In Fragmentation - search index elements in the 'Fragmentation' status of all data tables in the current database. Examples are as follows:

SELECT object_name (IPS.object_id) AS [TableName],
SI.name AS [IndexName],
IPS.Index_type_desc,
IPS.avg_fragmentation_in_percent,
IPS.avg_fragment_size_in_pages,
IPS.avg_page_space_used_in_percent,
IPS.record_count,
IPS.ghost_record_count,
IPS.fragment_count,
IPS.avg_fragment_size_in_pages
FROM sys.dm_db_index_physical_stats (db_id (DB_NAME ()), NULL, NULL, NULL, 'DETAILED') IPS
JOIN sys.tables ST WITH (nolock) ON IPS.object_id = ST.object_id
JOIN sys.indexes SI WITH (nolock) ON IPS.object_id = SI.object_id AND IPS.index_id = SI.index_id
WHERE ST.is_ms_shipped = 0
order by IPS.avg_fragment_size_in_pages desc

7. Missing Index:

With index components lost, SQL Server still has the ability to monitor and monitor the status of indexes created in order to improve the performance of the query string. The code below has the function of listing all lost index entries:

```
SELECT sys.objects.name
, (avg_total_user_cost * avg_user_impact) * (user_seeks + user_scans) AS Impact
, 'CREATE NONCLUSTERED INDEX ix_IndexName ON' + sys.objects.name COLLATE
DATABASE_DEFAULT + '(' + IsNull (mid.equality_columns, '') + CASE WHEN mid.inequality_columns IS
NULL
THEN ''
ELSE CASE WHEN mid.equality_columns IS NULL
THEN ''
ELSE ',' END + mid.inequality_columns END + ')' + CASE WHEN mid.included_columns IS NULL
THEN ''
ELSE 'INCLUDE (' + mid.included_columns + ')' END + ';' AS CreateIndexStatement
, mid.equality_columns
, mid.inequality_columns
, mid.included_columns
FROM sys.dm_db_missing_index_group_stats AS migs
INNER JOIN sys.dm_db_missing_index_groups AS mig ON migs.group_handle = mig.index_group_handle
INNER JOIN sys.dm_db_missing_index_details AS mid ON mig.index_handle = mid.index_handle AND
mid.database_id = DB_ID ()
INNER JOIN sys.objects WITH (nolock) ON mid.OBJECT_ID = sys.objects.OBJECT_ID
WHERE (migs.group_handle IN
(
SELECT TOP (500) group_handle
FROM sys.dm_db_missing_index_group_stats WITH (nolock)
ORDER BY (avg_total_user_cost * avg_user_impact) * (user_seeks + user_scans) DESC))
AND OBJECTPROPERTY (sys.objects.OBJECT_ID, 'isusertable') = 1
ORDER BY 2 DESC, 3 DESC
```

8. Index no longer uses:

The last component we mentioned in this section is that the index is no longer in use, please apply the following string of statements to list all indexes that have never been used, besides creating commands DROP:

```
SELECT o.name, indexname = i.name, i.index_id
, reads = user_seeks + user_scans + user_lookups
, writes = user_updates
, rows = (SELECT SUM (p.rows) FROM sys.partitions p WHERE p.index_id = s.index_id AND s.object_id =
p.object_id)
, CASE
WHEN s.user_updates
ELSE 1.00 * (s.user_seeks + s.user_scans + s.user_lookups) / s.user_updates
END AS reads_per_write
, 'DROP INDEX' + QUOTENAME (i.name)
```

+ 'ON' + QUOTENAME (c.name) + '.' + QUOTENAME (OBJECT_NAME (s.object_id)) as 'drop statement'
FROM sys.dm_db_index_usage_stats s
INNER JOIN sys.indexes i ON i.index_id = s.index_id AND s.object_id = i.object_id
INNER JOIN sys.objects o on s.object_id = o.object_id
INNER JOIN sys.schemas c on o.schema_id = c.schema_id
WHERE OBJECTPROPERTY (s.object_id, 'IsUserTable') = 1
AND s.database_id = DB_ID ()
AND i.type_desc = 'nonclustered'
AND i.is_primary_key = 0
AND i.is_unique_constraint = 0
AND (SELECT SUM (p.rows) FROM sys.partitions p WHERE p.index_id = s.index_id AND s.object_id = p.object_id)> 10000
ORDER BY reads

Index Maintainenance:

9. Restructuring Index:

To build, recreate index entries after performing defragmentation, or when you want to create a data table structure. The following statement is similar to **DBCC DBREINDEX** in versions of SQL Server since 2005:

USE AdventureWorks2008R2;
GO
ALTER INDEX PK_Employee_BusinessEntityID ON HumanResources.Employee
REBUILD;
GO

10. REORGANIZE:

The **REORGANIZE** syntax is applied to different leaf level indexes, specifically these **REORGANIZE** statements are always done online, and technically, this syntax is similar to **DBCC INDEXDEFRAG** in SQL Server versions after 2005.

USE AdventureWorks2008R2;
GO
ALTER INDEX PK_ProductPhoto_ProductPhotoID ON Production.ProductPhoto
REORGANIZE;
GO

Good luck!