

Choosing the Clustering Key



Kimberly L. Tripp

OWNER/PRESIDENT - SQLSKILLS.COM

@kimberlyltrippp

www.sqlskills.com/blogs/kimberly



Module Overview



Clustered index overview

Clustered index key choice

Clustered index key criteria

Clustering key suggestions

Clustering on an identity

Clustered Index Overview

Not required,
although highly
recommended

Only one per table

Physical order
applied at
creation

Logical order
maintained using
doubly-linked list

Requires ongoing
and automated
maintenance

Need to choose
wisely!



```
ALTER TABLE [Employee]  
    ADD CONSTRAINT [EmployeePK]  
    PRIMARY KEY ([EmployeeID]);
```

Primary Key Constraint Creates an Index

Defaults to unique clustered



```
ALTER TABLE [Employee]  
    ADD CONSTRAINT [EmployeePK]  
    PRIMARY KEY CLUSTERED ([EmployeeID]);
```

Primary Key Constraint Creates an Index

Explicitly state the index type desired

If a clustered index already exists, the primary key will be created as a nonclustered index instead



```
ALTER TABLE [Employee]  
    ADD CONSTRAINT [EmployeePK]  
    PRIMARY KEY NONCLUSTERED ([EmployeeGUID]);
```

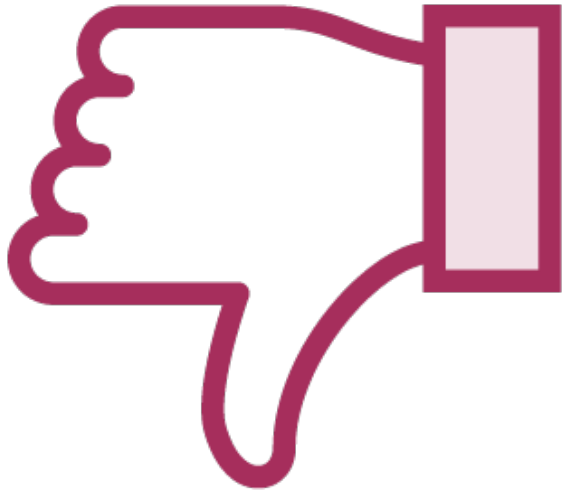
Primary Key Constraint Creates an Index

Can be created as a nonclustered index instead

Only one per table



Primary Key Does NOT Have to Be the Clustering Key



Primary key: relational integrity

Clustering key: internal mechanism for easily finding rows

If primary key is a natural key then likely want to enforce it with nonclustered index

If no column (or small set of columns) that meets these criteria then consider adding a surrogate [identity] key and then cluster it!

Clustered Index Key Choice: Historically

Reason	Explanation
Chosen to remove hotspots...	Page-level locking



Clustered Index Key Choice: Historically

Reason	Explanation
Chosen to remove hotspots...	Page-level locking
Chosen to improve “range” query performance...	Low selectivity “ranges” are obviously not bad, but are they the best?



Clustered Index Key Choice: Historically

Reason	Explanation
Chosen to remove hotspots...	Page-level locking
Chosen to improve “range” query performance...	Low selectivity “ranges” are obviously not bad, but are they the best?
Dependency on clustered index was much greater...	<p>SQL Server ONLY used ONE index per table per query (few exceptions)</p> <p>Adding nonclustered indexes or making nonclustered indexes wider degraded performance much more than the structures do in the current architecture (SQL Server 7.0 and higher)</p>



Clustered Index Key Choice: Currently

Reason

DOES NOT need to remove hot spots...

Explanation

True row-level locking



Clustered Index Key Choice: Currently

Reason

Explanation

DOES NOT need to remove hot spots...

True row-level locking

NOT the best for “range” queries...

Only gives ONE “range” query better performance, and only for queries asking for SELECT *

Range queries performance improved with better nonclustered indexes

SQL Server has improved index capabilities as indexes can be joined, scanned with lookups, aggregates, ...



Clustered Index Key Choice: Currently

Reason	Explanation
DOES NOT need to remove hot spots...	True row-level locking
NOT the best for “range” queries...	...
Dependency on clustered index has CHANGED...	Nonclustered indexes INCLUDE the clustering key for lookup
	Clustered index key criteria very important as it's included in each row of each nonclustered index!



Clustered Index Key Criteria

Unique

Yes: No extra time or space overhead, data takes care of this criteria

NO: SQL Server must “uniquify” the rows during INSERT



Clustered Index Key Criteria

Static

Yes: Reduces overhead

NO: Costly to maintain during key updates



Clustered Index Key Criteria

Narrow

Yes: Keeps nonclustered indexes narrow

NO: Unnecessarily wastes space



Clustered Index Key Criteria

Non-nullable and
fixed width

Yes: Reduces overhead

NO: Adds overhead to ALL nonclustered indexes



Clustered Index Key Criteria

Ever-increasing

Yes: Reduces fragmentation

NO: Inserts/updates might cause page splits (significant fragmentation)



Clustering Key Suggestions

Suggestion

Identity column

Explanation

Adding this column and clustering on it can be extremely beneficial, even when you don't "use" this data



Clustering Key Suggestions

Suggestion

Explanation

Identity column

...

GUID

NO: if populated by client-side call to .NET or server-side NEWID() function to generate GUID (OK as primary key but not as clustering key)

Maybe: if populated by server-side NEWSEQUENTIALID() function

But, this isn't really why you choose to use a GUID



Clustering Key Suggestions

Suggestion	Explanation
Identity column	...
GUID	...
DateCol, bigint (identity?)	<p>In that order and as a composite key (not date alone as that would need to be “uniquified” during INSERT)</p> <p>Great for very large tables</p> <p>Great for partitioned tables</p> <p>Great for ever-increasing tables where you have a lot of date-related queries</p>



Clustering on an Identity: The Good

Naturally unique

Naturally static

Naturally narrow

**Naturally non-
nullable and fixed
width**

**Naturally ever-
increasing**



Clustering on an Identity: The Bad

Can create system page contention on allocation when there are lots of tables that each have high insert volume

Can create page latch contention on insert in extremely high concurrent insert volume

What about “range” queries and optimization? Tuning focuses on other indexes for low-selectivity queries



What We Covered



Clustered index overview

Clustered index key choice

Clustered index key criteria

Clustering key suggestions

Clustering on an identity

