What's the difference between a Table Scan and a Clustered Index Scan?

Asked 16 years, 4 months ago Modified 8 years, 2 months ago Viewed 46k times

77

Since both a `Table Scan` and a `Clustered Index Scan` essentially scan all records in the table, why is a Clustered Index Scan supposedly better?

As an example - what's the performance difference between the following when there are many records?:

```
declare @temp table(
    SomeColumn varchar(50)
)

insert into @temp
select 'SomeVal'

select * from @temp

-----------------------------

declare @temp table(
    RowID int not null identity(1,1) primary key,
    SomeColumn varchar(50)
)

insert into @temp
select 'SomeVal'

select * from @temp
```

sql sql-server indexing

Share
Improve this question
Follow

edited Dec 19, 2013 at 18:37

Nizam
5,721 ● 9 ● 46 ● 57

asked Aug 20, 2008 at 20:38

Seibar
70.2k ● 38 ● 93 ● 100

3 Answers

Sorted by: Highest score (default)

89

In a table without a clustered index (a heap table), data pages are not linked together - so traversing pages requires a lookup into the Index Allocation Map.

A clustered table, however, has it's data pages linked in a doubly linked list - making sequential scans a bit faster. Of course, in exchange, you have the overhead of dealing with keeping the data pages in order on `INSERT`, `UPDATE`, and `DELETE`. A heap table, however, requires a second write to the IAM.

If your query has a `RANGE` operator (e.g.: `SELECT * FROM TABLE WHERE Id BETWEEN 1 AND 100`), then a clustered table (being in a guaranteed order) would be more efficient - as it could use the index pages to find the relevant data page(s). A heap would have to scan all rows, since it cannot rely on ordering.

And, of course, a clustered index lets you do a CLUSTERED INDEX SEEK, which is pretty much optimal for performance...a heap with no indexes would always result in a table scan.

So:

- For your example query where you select all rows, the only difference is the doubly linked list a clustered index maintains. This should make your clustered table just a tiny bit faster than a heap with a large number of rows.
- For a query with a `WHERE` clause that can be (at least partially) satisfied by the clustered index, you'll come out ahead because of the ordering - so you won't have to scan the entire table.
- For a query that is not satisified by the clustered index, you're pretty much even...again, the only difference being that doubly linked list for sequential scanning. In either case, you're suboptimal.
- For `INSERT`, `UPDATE`, and `DELETE` a heap may or may not win. The heap doesn't have to maintain order, but does require a second write to the IAM. I think the relative performance difference would be negligible, but also pretty data dependent.

Microsoft has a whitepaper which compares a clustered index to an equivalent non-clustered index on a heap (not exactly the same as I discussed above, but close). Their conclusion is basically to put a clustered index on all tables. I'll do my best to summarize their results (again, note that they're really comparing a non-clustered index to a clustered index here - but I think it's relatively comparable):

- `INSERT` performance: clustered index wins by about 3% due to the second write needed for a heap.
- `UPDATE` performance: clustered index wins by about 8% due to the second lookup needed for a heap.
- `DELETE` performance: clustered index wins by about 18% due to the second lookup needed and the second delete needed from the IAM for a heap.

- single `SELECT` performance: clustered index wins by about 16% due to the second lookup needed for a heap.
- range `SELECT` performance: clustered index wins by about 29% due to the random ordering for a heap.
- concurrent `INSERT`: heap table wins by 30% under load due to page splits for the clustered index.

Share

Improve this answer

Follow

edited Oct 15, 2016 at 18:49

BanksySan

28.5k ● 36 ● 125 ● 227

answered Aug 20, 2008 at 21:32

Mark Brackett

85.6k ● 17 ● 111 ● 155

2 This very question bubbled in my mind today. Thanks @Terrapin for asking this and thanks to @Marc for answering it so well! – peakit May 1, 2011 at 18:53

3 MS Exam 70461 Querying Microsoft SQL Server 2012 - Chapter 15 Lesson 1 has in-depth exploitation. – June Jan 13, 2015 at 23:28

I can't seem to gain the supposed boost indicated by this statement of yours: "For a query with a WHERE clause that can be (at least partially) satisfied by the clustered index, you'll come out ahead because of the ordering - so you won't have to scan the entire table." I have a table of 10 million rows. SELECT Id FROM Customer WHERE Id > X executes in the same amount of time no matter if I have a clustered index on Id or not. How come? I can see how it changes from table scan to clustered index scan though. – Mattias Nordqvist Apr 30, 2015 at 7:17

2 @MattiasNordqvist - If you're just looking at time taken, You're Doing It Wrong. Because of caching, concurrent access, CPU vs disk time, etc. it's difficult to perf-tune MS-SQL on time alone. Look into SET STATISTICS IO ON to check your disk reads, which is where the boost would come from. Second, it's going to depend on the number of rows returned - if it's a high enough percentage, the optimizer could conceivably choose a read + filter instead. – Mark Brackett Apr 30, 2015 at 19:42

@MattiasNordqvist "I have a table of 10 million rows. SELECT Id FROM Customer WHERE Id > X executes in the same amount of time no matter if I have a clustered index on Id or not. How come?" It will be faster only if the clustered index is on "Id" column. – IndustryUser1942 Feb 11, 2022 at 7:48

5

http://msdn.microsoft.com/en-us/library/aa216840(SQL.80).aspx

The Clustered Index Scan logical and physical operator scans the clustered index specified in the Argument column. When an optional WHERE:() predicate is present, only those rows that satisfy the predicate are returned. If the Argument column contains the ORDERED clause, the query processor has requested that the rows' output be returned in the order in which the clustered index has sorted them. If the ORDERED clause is not present, the storage engine will scan the index in the optimal way (not guaranteeing the output to be sorted).

http://msdn.microsoft.com/en-us/library/aa178416(SQL.80).aspx

The Table Scan logical and physical operator retrieves all rows from the table specified in the Argument column. If a WHERE:() predicate appears in the Argument column, only those rows that satisfy the predicate are returned.

Share Improve this answer Follow

answered Aug 20, 2008 at 20:54

Stu

15.8k ● 4 ● 45 ● 74

-4

A table scan has to examine every single row of the table. The clustered index scan only needs to scan the index. It doesn't scan every record in the table. That's the point, really, of indices.

Share Improve this answer Follow

answered Aug 20, 2008 at 20:44

DrPizza

18.3k ● 7 ● 42 ● 53

11 -1 This is wrong. The leaf level of the clustered index *is* the table. – Martin Smith Feb 14, 2014 at 19:05