

# SQL Cursors...Any use cases you would defend?

Asked 16 years, 2 months ago    Modified 14 years, 4 months ago

Viewed 3k times



I'll go first.

2



I'm 100% in the set-operations camp. But what happens when the set logic on the entire desired input domain leads to a such a large retrieval that the query slows down significantly, comes to a crawl, or basically takes infinite time?



That's one case where I'll use a itty-bitty cursor (or a while loop) of perhaps most dozens of rows (as opposed to the millions I'm targeting). Thus, I'm still working in (partitioned sub) sets, but my retrieval runs faster.

Of course, an even faster solution would be to call the partitioned input domains in parallel from outside, but that introduces an interaction with an external system, and when "good enough" speed can be achieved by looping in serial, just may not be worth it (especially during development).

sql

sql-server

t-sql

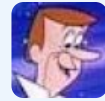
set

Share

Improve this question

Follow

asked Oct 5, 2008 at 19:46



George Jetson

2,235 ● 1 ● 19 ● 16

10 Answers

Sorted by:

Highest score (default)



3

I've got plenty of cases where rows from a configuration table have to be read and code generated and executed, or in many meta-programming scenarios.



There are also cases when cursors simply outperform because the optimizer is not smart enough. In those cases, either there is meta-information in your head which is simply not revealed to the optimizer through the indexes or statistics on the tables, or the code is just so complicated that the joins (and usually, re-joins) simply can't be optimized in the way you can visualize them in a cursor-based fashion. In SQL Server 2005, I believe CTEs tend to make this look a lot simpler in the code, but whether the optimizer also sees them as simpler is hard to know - it comes down to comparing the execution plan to how you think it could be done most efficiently and making the call.

General rule - don't use a cursor unless necessary. But when necessary, don't give yourself a hard time about it.

Share Improve this answer

Follow

answered Oct 6, 2008 at 0:34



Cade Roux

89.6k ● 40 ● 184 ● 266



3



There are lots of different [cursor behaviors](#).

- STATIC vs KEYSET vs DYNAMIC
- SCROLL vs FORWARD ONLY vs FAST FORWARD
- INSENSITIVE or not
- OPTIMISTIC or READ ONLY or not
- LOCAL vs GLOBAL (at least this is easy)

You should **never use a cursor** unless you can **explain all of these options** and which ones are **on by default**.

And so, I never do.

Instead, when I feel the urge to loop over something in T-SQL... I load it into a variable table, which is something like a LOCAL STATIC SCROLL cursor... except that it can be indexed and joined (edit: and the downside of preventing the use of parallelism).

Share Improve this answer

edited Oct 6, 2008 at 18:57

Follow

answered Oct 6, 2008 at 4:36



Amy B

110k ● 21 ● 139 ● 190



Sure, there are a number of places where cursors might be better than set-based operations.

2



One is if you're updating a lot of data in a table (for example a SQL Agent job to pre-compute data on a schedule) then you might use cursors to do it in multiple small sets rather than one large one to reduce the amount of concurrent locking and thus reduce the chance of lock contention and/or deadlocks with other processes accessing the data.

Another is if you want to take application-level locks using the `sp_getapplock` stored procedure, which is useful when you want to ensure rows that are being polled for by multiple processes are retrieved exactly once ([example here](#)).

In general though, I'd agree that it's best to start using set based operations if possible, and only move to cursors if required either for functionality or performance reasons (with evidence to back the latter up).

Share Improve this answer

answered Oct 6, 2008 at 0:04

Follow



**Greg Beech**

136k ● 45 ● 209 ● 250

---

There are much more straight forward ways of using a table as a queue than using cursors and `sp_getapplock`. See for example [rusanu.com/2010/03/26/using-tables-as-queues](http://rusanu.com/2010/03/26/using-tables-as-queues) – [Martin Smith](#) Dec 2, 2010 at 10:15

---



Very occasionally you will get an operation that needs a cursor but in T-SQL it is fairly rare. Identity(int) columns or sequences order things in ways within set operations.

2



Aggregations where calculations might change at certain points (such as accumulating claims from ground up to a limit or excess point) are inherently procedural, so those are a candidate for a cursor.



Other candidates would be inherently procedural such as looping through a configuration table and generating and executing a series of queries.

Share Improve this answer

answered Oct 5, 2008 at 23:44

Follow



[ConcernedOfTunbridgeWells](#)

66.5k ● 15 ● 148 ● 198



Along with what [David B](#) said, I, too, prefer the loop/table approach.

2



With that out of the way, one use case for cursors and the loop/table approach involves extremely large updates.

Let's say you have to update 1 billion rows. In many instances, this may not need to be transactional. For example, it might be a data warehouse aggregation where you have the potential to reconstruct from source files if things go south.



In this case, it may be best to do the update in "chunks", perhaps 1 million or 10 million rows at a time. This helps keep resource usage to a minimum, and allows concurrent use of the machine to be maximized while you update that billion rows. A looped/chunked approach

might be best here. Billion row updates on less-than-stellar hardware tend to cause problems.

Share Improve this answer

edited May 23, 2017 at 9:57

Follow



Community Bot

1 • 1

answered Oct 6, 2008 at 16:47



Pittsburgh DBA

6,772 • 2 • 43 • 70



2



In a pure SQL environment, I'd rather avoid cursors as you suggest. But once you cross over into procedural language (like PL/SQL), there are a number of uses. For example, if you want to retrieve certain rows and want "to do" something more complex than update it with them.



Share Improve this answer

edited Aug 12, 2010 at 8:33



Follow



Rob

45.7k • 24 • 125 • 155

answered Oct 5, 2008 at 20:19



Thorsten

13.1k • 17 • 64 • 79



1



Cursors are also handy when you want to run a system proc multiple times with different input values. I have no intention of trying to rewrite system procs to be set-based, so I will use a cursor then. Plus you are usually going through a very limited number of objects. You can do the same thing with an existing proc that inserts only



one record at a time, but from a performance view, this is usually a bad thing if you have a lot of records to run through. Those I will rewrite to be set-based.

Running totals as discussed by others can be faster.

If you are emailing from the database (not the best idea but sometimes it is what you are stuck with), then a cursor can ensure that customer a doesn't see customer b's email address when you send both the same email.

Share Improve this answer

answered Mar 4, 2009 at 15:24

Follow



**HLGEM**

96.4k ● 15 ● 119 ● 189



0



Having to use a cursor is generally a sign that you are doing in the database what ought to be done in the application. As others have said, cursors are generally needed when a stored procedure is calculating running totals, or when you're generating code and/or meta-programming.



But why are you doing that kind of work in a stored procedure in the first place? Is that really the best use of your database server? Is T-SQL really the right language to use when generating code?

Sure, sometimes the answer is "yes," or, more likely, "no, but it's simpler this way." In my view, keeping things simple trumps premature optimization any day of the week. So I use cursors. But when I think I need to use a

cursor, the universe is asking me a question that I should really have a good answer to.

Share Improve this answer

answered Oct 6, 2008 at 19:53

Follow



Robert Rossney

96.6k ● 24 ● 148 ● 218



0



If a table is un-indexed for some reason, a cursor will be faster than other methods of iterating over a table. I found this information in [this blog post](#) on cursors in SQL Server last year.

While the author is in favor of "use only as a last resort" approach (as is everyone here), she does find a case or two where cursors perform as well as other available alternatives (including the running totals pointed out by Robert Rossney). Among other interesting points she makes, she indicates that cursors operate more efficiently inside stored procedures than as ad-hoc queries. The author also does an excellent job of pointing out when the performance problems we all associate with cursors begin to occur.

The blog post contains actual code, so readers can try the queries themselves and see the results.

Share Improve this answer

answered Oct 6, 2008 at 20:24

Follow



Scott Lawrence

7,243 ● 13 ● 47 ● 64



---

If a table is unindexed and a cursor is faster, it will probably be still faster to index the table and use set-based logic.

– [HLGEM](#) Mar 4, 2009 at 15:21

---

No doubt. When I wrote my answer last year, I couldn't think of a reason why you'd have an un-indexed table.

– [Scott Lawrence](#) Mar 4, 2009 at 16:59

---



0



Well one operation where cursors are better than sets is when calculating a running total and similar stuff.

Share Improve this answer

edited Aug 12, 2010 at 8:34

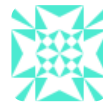
Follow



[Rob](#)

45.7k ● 24 ● 125 ● 155

answered Oct 5, 2008 at 20:02



[Mladen](#)

1,720 ● 12 ● 4

---

An analytic function calculates running totals in the database as fast as calculating in a cursor. – [David Aldridge](#) Oct 6, 2008 at 0:09

---


1 ... or faster, as with analytic functions you can essentially get the totals on a single scan of the table as opposed to getting the data then looping through it all again. – [Nick Pierpoint](#) Oct 6, 2008 at 0:37

---

1 Analytic functions let you calculate totals (and lots of other things) *before* the group by stage of a query. They're fantastic - you get the totals before you start looping through a cursor. – [Nick Pierpoint](#) Oct 7, 2008 at 9:45

---

@Nick, @David Analytic functions in SQL Server (2005 and 2008) do not support any way of referencing the next or

previous row so running totals is indeed a classic case where cursors can out perform the set based solution. Hopefully the next version will implement this. – [Martin Smith](#) Dec 2, 2010 at 10:19 

---

---