Steve Demonstrates Data Paging with CTE's

Transcription of Video:

Here's an example of a CTE that's been created for data paging, where the select statement inside of the CTE that shown in blue is just grabbing some table names and column names from the SIS dot columns table. And there's a lot of them in there. And then what it's doing is it's calculating the row number ordered by something, and it's ordering by the object name. So it's the alphabetical listing as the row. And normally, without a CTE, you cannot use the row number function in a where clause or having type clause to filter things out, it needs to be done. And actually, you can't even use it in a sub query. If you're going to use it for filtering that way. This is one of those things, we're wrapping it in a CTE makes it so you can filter on it later. And then we're going to select from the CTE, where the row is between a specific page number times the page size. So let's take a look at that demo. So we're going to be back in that same CTE demo database. We're just going to select from the SIS dot columns table, which is one of the sequel system tables if you're not familiar with it. And it just shows, in this example, sorted by table name, all of the tables and what all of their columns are. So what we're going to look at is how to put that into a paging CTE. So this is very similar to what we showed in the slide a moment ago. But we've declared some variables here, where we have a page number and a page size. And for this example, I want to get to page two, with page sizing of 10. Inside here, we're going to do a select of the table name and the column names and a row number. And it'll look like this. But we can't use that row number in the where clause inside of the CTE. So what we have to do is we use it in the where clause outside of the CTE we highlight the whole query. And then we get rows 11 through 20. If we change this to let's say, page four, with page size have been 15 rows we'll run the query that should give us row 46 through 60, as we're seeing there, so. But that's something that would be a little bit easier if we could wrap it in a procedure. So let's take a look at putting that into a procedure. We're just going to call it tables and column pager, where it's just going to take a page size and a page number, we're going to create the procedure with the same query that we use just above there. But instead of declaring the variables inside, we're going to pass them in as parameters. Create the procedure, and then we're going to say give me tables and columns for page one, two, and three. And we'll get three results sets. And the first result set is row one through 10. The second result set is 11, through 18. And the third one is rows 21 through 30. And when we look down here that comes from doing this, if we went in and say Okay, give me the fifth page, there's page five, which is rows 41, through 50. So that's one way of doing it. Yeah, when CTE's were first introduced, there, that was really the primary way of doing really good data paging, and a lot of people I think, picked up CTE's purely to be able to do that kind of data paging. But with SQL Server 2012. And newer, there's this new option, where we can use the OFFSET command. And we're going to do the same query here

OFFSET page number times pays page size, which would be a page number was two would be OFFSET of two minus one would be would be one times the page size of 10 would then give me 10 rows. So what I'm going to do is run these side by side, the original CTE and the page number here and compare the results. And you can see we're getting the exact same results there. But the advantage of the CTE is we can display the page number as it was shown prior to the filtering here. Let's take a look at how these actually perform. So I'm just going to turn on statistics IO and time. And I'm gonna hit Ctrl. Actually, I'm just gonna do CTRL M to turn on the actual execution plan and look at the comparison here. There we go, we run it we get the actual execution plan. And you can see comparing them 55% of the cost was associated with the CTE and 45% of the cost was associated with the rows sorry with the OFFSET and rows options Now, that's one of those things that performance as usual may vary. And there may be situations where the CTE is a better option. And there may be situations where the OFFSET and rows is the best option really depends on how it performs and the specific data behind it. So if you're looking at doing paging, you might want to compare the two of those and see which works best. So here's just an example showing the alternative to the CTE using the OFFSET and fetch like we just looked at. If you're using SQL Server 2012 or newer, which hopefully in today's day and age, most people are.