عنوان مقاله: Cursor در SQL Server
نویسنده مقاله: تیم فنی نیک آموز
تاریخ انتشار: فروردین ماه ۱۴۰۱
منبع: https://nikamooz.com/Cursor-SQL-Server

مقدمه

Cursor یک شی پایگاه داده است که برای بازیابی داده ها از مجموعه نتایج یک ردیف در یک زمان استفاده می شود. در SQL Server زمانی استفاده می شود که داده ها باید سطر به سطر به روز شوند. این مقاله همه چیز را در مورد cursor در SQL Server توضیح می دهد. در این مقاله با موارد زیر آشنا خواهیم شد:

۱- مقدمه ای بر cursor در SQL Server
۲- چرخه عمر Cursor در SQL Server
۳- چرا و چه زمانی از Cursor در SQL Server استفاده کنیم.
۴- چه محدودیت هایی برای Cursor در SQL Server وجود دارد ؟
۵- چه زمانی می توانیم یک Cursor در SQL Server جایگزین کنیم.

مراحل زیر چرخه عمر Cursor در SQL Server شامل می شود

- Cursor توسط یک عبارت در SQL Server تعریف می شود.
- Cursor برای ذخیره سازی داده های بازیابی شده از مجموعه نتایج باز می شود
- هنگامی که cursor باز می شود ردیف ها را می توان از cursor یک به یک یا در یک بلوک برای دستکاری داده ها واکشی کرد
- Cursor باید به طور واضح پس از دستکاری داده ها بسته شود.
- Cursor ها باید برای تعریف حذف Cursor و آزادسازی منابع سیستم مرتبط با cursor تخصیص داده شوند.

چرا از Cursor در SQL Server استفاده می کنیم؟

در پایگاه های داده رابطه محور، عملیات روی مجموعه از ردیف ها انجام می شود به عنوان مثال زمانی که دستور select مجموعه ای از ردیف ها را برمی گرداند که مجموعه نتیجه نامیده می شود گاهی اوقات منطق برنامه این

هست که باید یک سطر در یک زمان کار کند نه کل نتیجه مجموعه در یک زمان این را می توان با استفاده از cursor انجام داد

در برنامه نویسی ما از یک حلقه for,while برای تکرار یک آیتم در یک زمان استفاده می کنیم cursor از همین رویکرد استفاده می کند و ممکن است ترجیح داده شود زیرا از منطق یکسانی پیروی می کند

نحوه نوشتن syntax دستور cursor مطابق تصویر زیر می باشد:

```
DECLARE cursor_name CURSOR [ LOCAL | GLOBAL ]
[ FORWARD_ONLY | SCROLL ]
 [ STATIC | KEYSET | DYNAMIC | FAST_FORWARD ]
[ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ]
[ TYPE_WARNING ] FOR select_statement
 [ FOR UPDATE [ OF column_name [ ,...n ] ] ] [;]
```

مثال از Cursor

cursorبرای بازیابی کد پرسنلی و نام از جدول Employee تعریف شده است. مقدار FETCH_STATUSبرابر با صفر است است تا زمانی که ردیف‌هایی وجود داشته باشد. وقتی همه ردیف‌ها واکشی شدند، FETCH_STATUS برابر با یک می‌شود. برای این منظور کوئری زیر را می نویسیم:

```
use Product_Database
SET NOCOUNT ON;

DECLARE @emp_id int ,@emp_name varchar(۲۰),
   @message varchar(max);

PRINT '-------- EMPLOYEE DETAILS --------';

DECLARE emp_cursor CURSOR FOR
SELECT emp_id,emp_name
FROM Employee
order by emp_id;

OPEN emp_cursor

FETCH NEXT FROM emp_cursor
```

```
INTO @emp_id,@emp_name

print 'Employee_ID  Employee_Name'

WHILE @@FETCH_STATUS = ۰
BEGIN
   print '    ' + CAST(@emp_id as varchar(۱۰)) +'          '+
      cast(@emp_name as varchar(۲۰))


   FETCH NEXT FROM emp_cursor
INTO @emp_id,@emp_name

END
CLOSE emp_cursor;
DEALLOCATE emp_cursor;
```

خروجی کوئری فوق مطابق تصویر زیر می باشد

چه محدودیت هایی در پیاده سازی Cursor در SQL Server وجود دارد؟

Cursorمجموعه ای از اشاره گرها در حافظه است. به این معنی که حافظه ای را از سیستم شما اشغال می کند که ممکن است برای فرآیندهای دیگر در دسترس باشد.

Cursor می توانند سریعتر از یک حلقه while باشند، اما سربار بیشتری دارند.

یکی دیگر از عوامل موثر بر سرعت cursorتعداد سطرها و ستون های وارد شده به cursorاست. زمان باز کردن Cursorو واکشی عبارات چقدر طول می کشد.
ستون‌های زیادی که در حافظه کشیده می‌شوند، که هرگز در عملیات cursorبعدی به آنها ارجاع نمی‌شوند، می‌توانند سرعت کار را کاهش دهند.
Cursor کندتر هستند زیرا جدول ها را ردیف به ردیف به روز می کنند.

چگونه می توانیم Cursor ها در SQL Server جایگزین کنیم؟

فرض کنید باید داده ها را از دو جدول به طور همزمان با مقایسه کلیدهای اصلی و کلیدهای خارجی بازیابی کنیم. در این نوع مشکلات، cursorهنگام پردازش در هر ستون، عملکرد بسیار ضعیفی را ارائه می دهد. از طرف دیگر استفاده از join ها در آن شرایط امکان پذیر است زیرا فقط ستون هایی را پردازش می کند که شرایط را برآورده می کنند. بنابراین در اینجا اتصالات سریعتر از مکان نما هستند.

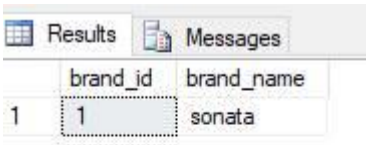
مثال زیر جایگزینی Cursorاز طریق join ها را توضیح می دهد.

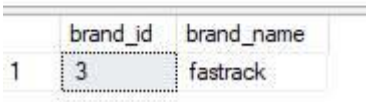
فرض کنید دو جدول ProductTable و Brand Table داریم. کلید اصلی BrandTable brand_id است که در ProductTable به عنوان کلید خارجی brand_id ذخیره می شود. حالا فرض کنید، من باید brand_name را از BrandTable با استفاده از کلید خارجی brand_id از ProductTable بازیابی کنم. در این مواقع برنامه های cursorبه صورت زیر خواهد بود:

```
use Product_Database
SET NOCOUNT ON;

DECLARE @brand_id int
DECLARE @brand_name varchar(۲۰)


PRINT '--------Brand Details --------';

DECLARE brand_cursor CURSOR FOR
SELECT distinct(brand_id)
FROM ProductTable;

OPEN brand_cursor
```

```
FETCH NEXT FROM brand_cursor
INTO @brand_id

WHILE @@FETCH_STATUS = 0
BEGIN
   select brand_id,brand_name from BrandTable where brand_id=@brand_id
--(@brand_id is of ProductTable)

   FETCH NEXT FROM brand_cursor
INTO @brand_id

END
CLOSE brand_cursor;
DEALLOCATE brand_cursor;
```

خروجی کوئری فوق مطابق تصویر زیر می باشد.

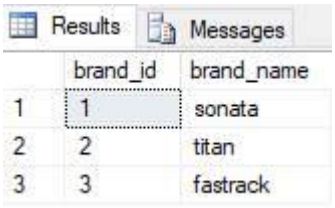




کوئری فوق را می توان با دستور inner join هم پیاده سازی کرد برای این منظور کوئری زیر را می نویسیم.

```
Select distinct b.brand_id,b.brand_name from BrandTable b inner join
ProductTable p on b.brand_id=p.brand_id
```

خروجی کوئری فوق مطابق تصویر زیر می باشد



Results | Messages

	brand_id	brand_name
1	1	sonata
2	2	titan
3	3	fastrack

همانطور که از مثال بالا می بینیم، استفاده از Join خطوط کد را کاهش می دهد و در صورت نیاز به پردازش رکوردهای بزرگ، عملکرد سریع تری می دهد.