

Performance Comparison of Common Table Expressions and Cursors

Sowndarya. S
Assistant System Engineer - Trainee
Tata Consultancy Services Limited
Chennai, TamilNadu, India
s.sowndarya1@tcsin.com

Sridhar. A
Assistant System Engineer - Trainee
Tata Consultancy Services Limited
Chennai, TamilNadu, India
a.sridhar@tcsin.com

Abstract: This paper compares the performance of common table expressions and cursors when implemented for complex queries. Cursor enables traversal of records in a database. Cursors can also be called as “Iterators”, as it performs retrieval, addition and removal of database records based on the given condition. It is mainly used for processing individual rows in a database. Common table expression is an alternative for derived tables. It increases the performance and reduces the complexity of the queries. CTE is mainly used for writing recursive queries.

Keywords— Cursors; Common Table Expressions

I. INTRODUCTION

Data plays a vital role in our day to day life. Storing and maintaining the available data is a major part. For this to happen effectively, “Databases” comes into picture. Database Management System helps in maintaining large volume of data. Data are stored in the form of tables. Sql queries are used to play with the data in the table.

To traverse over the records, cursors are used. The cursor can refer only one row at a time, but it can move to other rows of the result set. To use cursors, the following four steps should be performed.

- Declare a cursor
- Open the cursor
- Fetch the cursor
- Close the cursor

Below is the basic syntax of a cursor.

```
DECLARE Cursor Name CURSOR FOR  
Select * from Table name  
OPEN Cursor Name  
FETCH Cursor Name INTO @Name  
While (@@FETCH_STATUS = 0)  
Begin  
Print @Name  
FETCH Cursor Name INTO @Name  
End  
CLOSE Cursor Name  
DEALLOCATE Cursor Name
```

Common table expression is similar to that of the derived table as it is not stored as an object; rather, it lasts only for the duration of the query. CTE is a result set which can be referenced within a Select, Insert, Update or Delete statement. SQL Server supports two types of CTE.

- Recursive
- Non-Recursive

Query for Inner Join Operation:

```
Select * from (
  Select A.Name from Adress A
  INNER JOIN Employee E On E.EmpID = A.EmpID) T
  Where Name = 'Sowndarya'
```

Using CTE:

```
With T
AS
(
  Select A.Name from Adress A
  INNER JOIN Employee E On E.EmpID = A.EmpID
)
Select * from T
where Name = 'Sowndarya'
```

Using CURSORS:

```
DECLARE @Name varchar(50)
DECLARE @Age int
DECLARE AllIDBCursor CURSOR FOR
  Select * FROM (
    Select A.Name From Adress A
    Inner join Employee E on E.empID = A.empID ) T
  Where Name = 'Sowndarya'
OPEN AllIDBCursor;
FETCH AllIDBCursor INTO @Name;
WHILE (@@FETCH_STATUS = 0)
  Begin
    Print @Name
  FETCH AllIDBCursor INTO @Name;
  END
CLOSE AllIDBCursor;
DEALLOCATE AllIDBCursor;
```

II. SIMULATION AND RESULT

Screen Shot -1:

```

SQLQuery1.sql - V...CAIPpatnaik (931)
SET @start_time = CURRENT_TIMESTAMP;
-- query goes here
DECLARE @Name varchar(50)
DECLARE @Age int
DECLARE AllDBCursor CURSOR FOR
    SELECT * FROM (
        SELECT A.Name From Adress A
        Inner join Employee E on E.empID = A.empID ) T
WHERE Name = 'Sowndarya'
OPEN AllDBCursor;
FETCH AllDBCursor INTO @Name;
WHILE (@@FETCH_STATUS = 0) -- loop through all db-s
BEGIN
    PRINT @Name
    --PRINT @Age
    FETCH AllDBCursor INTO @Name;
END -- while
    
```

Name	
1	Sowndarya

CTEQueryExecutionTimings	
1	0

CursorQueryExecutionTimings	
1	3

Screen Shot – 2:

```

SQLQuery1.sql - V...CAIPpatnaik (931)*
SET @start_time = CURRENT_TIMESTAMP;
-- query goes here
DECLARE @Name varchar(50)
DECLARE @Age int
DECLARE AllDBCursor CURSOR FOR
    SELECT * FROM (
        SELECT A.Name From Adress A
        Inner join Employee E on E.empID = A.empID ) T
WHERE Name = 'Soundarya'
OPEN AllDBCursor;
FETCH AllDBCursor INTO @Name;
WHILE (@@FETCH_STATUS = 0) -- loop through all db-s
BEGIN
    PRINT @Name
    --PRINT @Age
    FETCH AllDBCursor INTO @Name;
END -- while
    
```

Name	
1	Sowndarya

CTEQueryExecutionTimings	
1	0

CursorQueryExecutionTimings	
1	6

III. TABULAR COLUMN

Number of Records	Query Execution Timing using CTE (In Milli Seconds)	Query Execution Timing using Cursors (In Milli Seconds)
100	0	3
1000	0	6

IV. CONCLUSION

The above screen shot compares the performance by calculating the query execution timing. There are three tables named "Employee, Address and T". An inner join operation has been performed using both the concepts and results are recorded.

Cursors will allocate resource on the server. If the cursor is not properly deallocated, the resources will not be free until the session itself gets closed. This wasting of resource on the server, not only reduces the performance but may also lead to failures. Thereby, to increase the performance, common table expressions are used.

V. REFERENCES

- [1] Paul Nielsen, Uttam Parui and Mike White, Microsoft SQL Server 2008 Bible
- [2] Ramez Elmasri and B.Navathe, 5th edition, Fundamentals of Database Systems
- [3] www.msdn.microsoft.com
- [4] Wikipedia Search [Online]
Available: <http://www.wikipedia.org/>
- [5] The IEEE Website [Online]
Available: <http://www.ieee.org/>