Cursor

WHAT IS CURSOR?

A cursor is a database object which is used to retrieve data from result set one row at a time. The cursor can be used when the data needs to be updated row by row.

Types of Cursors in SQL:

Implicit Cursor

Explicit Cursor

Implicit Cursor: Whenever DML operations such as INSERT, UPDATE, and DELETE are processed in the database, implicit cursors are generated automatically and used by the framework. These types of cursors are used for internal processing and can’t be controlled or referred from another code area.

Explicit Cursor: This type of cursor is generated whenever data is processed by a user through an SQL block. Generally, the use of the SELECT query triggers the creation of an explicit cursor and can hold more than one row but process just one at a time.

CURSOR LIFE CYCLE:

**Declaring Cursor** A cursor is declared by defining the SQL statement.

**Opening Cursor** A cursor is opened for storing data retrieved from the result set.

**Fetching Cursor** When a cursor is opened, rows can be fetched from the cursor one by one or in a block to do data manipulation.

**Closing Cursor** The cursor should be closed explicitly after data manipulation.

**Deallocating Cursor** Cursors should be deallocated to delete cursor definition and released all the system resources associated with the cursor.

**WHY USE CURSORS?**

In relational databases, operations are made on a set of rows. For example, a SELECT statement returns a set of rows which is called a result set. Sometimes the application logic needs to work with a row at a time rather than the entire result set at once. This can be done using cursors.

CURSOR LIMITATIONSA: cursor is a memory resident set of pointers -- meaning it occupies memory from your system that may be available for other processes.

But, when you open a cursor, you are basically loading those rows into memory and locking them, creating potential blocks. Then, as you cycle through the cursor, you are making changes to other tables and still keeping all of the memory and locks of the cursor open.

**SQL Server Cursor Limitations:-**

1. When processing data, it imposes locks on a subset or the entire table.
2. The cursor updates table records one row at a time, which slows down its performance.
3. While loops are slower than cursors, they do have more overhead.
4. Another factor that influences cursor speed is the quantity of rows and columns brought into the cursor.

**Cursor Alternate:-**

Temporary tables & While Loop

1. **How to find nth highest salary in sql**

db.products.find().sort({price:-1}).skip(n-1).limit(n-1)

with Emp as

(

select \*,DENSE\_RANK() over (order by salary desc) as RankID from Employee

)

select top 1 \* from Emp where RankID=N

1. **How to find 1th highest salary in sql**

db.products.find().sort({price:-1}).limit(1)

select max(Salary) from Employee

1. **How to find 2th highest salary in sql**

db.products.find().sort({price:-1}).skip(1).limit(1)

1. **How to find 15th highest salary in sql**

db.products.find().sort({price:-1}).skip(15-1).limit(15-1)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **SQL query to get organization hierarchy**

with Emp as

(

select EmployeeID,Name,Salary,ManagerID from Employee where EmployeeID=1

union all

select Employee.EmployeeID, Employee.Name,Employee.Salary,Employee.ManagerID from Employee join Emp on Employee.EmployeeID=Emp.ManagerID

)

select EmployeeID,Name,Salary,ManagerID from Emp

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **Delete duplicate rows in sql**

WITH EmployeesCTE AS

(

SELECT \*, ROW\_NUMBER()OVER(PARTITION BY Name ORDER BY EmployeeID) AS RowNumber

FROM Employee

)

Delete from EmployeesCTE where RowNumber>1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **SQL query to find employees hired in last n months**

select \*,DATEDIFF(MONTH,HireDate,GETDATE()) as datediffnum from Employee

where DATEDIFF(MONTH,HireDate,GETDATE()) between 1 and 12

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **Transform rows into columns in sql server**

select country , city1,city2,city3 from

(

select \*,'City'+ cast(ROW\_NUMBER() over(partition by Country order by City) as nvarchar(10)) as CityID from Countries

) tmp

pivot

(

MAX(City) for CityID in(city1,city2,city3)

) as pivottable

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **SQL query to find rows that contain only numerical data**

SELECT Value FROM TestTable WHERE ISNUMERIC(Value) = 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**SQL Query to find department with highest number of employees**

select Department.Name, count(Employee.EmployeeID) from Department

inner join Employee on Department.DepartmentPID=Employee.DepartmentPID

group by Employee.DepartmentPID,Department.Name

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **Difference between inner join and left join**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **Can we join two tables without primary foreign key relation**

Yes, we can join two tables without primary foreign key relation as long as the column values involved in the join can be converted to one type.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **Difference between blocking and deadlocking**

Blocking : Occurs if a transaction tries to acquire an incompatible lock on a resource that another transaction has already locked. The blocked transaction remains blocked until the blocking transaction releases the lock.

Deadlock : Occurs when two or more transactions have a resource locked, and each transaction requests a lock on the resource that another transaction has already locked. Neither of the transactions here can move forward, as each one is waiting for the other to release the lock. So in this case, SQL Server intervenes and ends the deadlock by cancelling one of the transactions, so the other transaction can move forward.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **Sql query to select all names that start with a given letter without like operator**

SELECT \* FROM Employee WHERE Name LIKE 'M%'

SELECT \* FROM Students WHERE CHARINDEX('M',Name) = 1

SELECT \* FROM Students WHERE LEFT(Name, 1) = 'M'

SELECT \* FROM Students WHERE SUBSTRING(Name, 1, 1) = 'M'

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **Sql date interview questions**

Write a SQL query to get

**1. All people who are born on a given date (For example, 9th October 2017)**

SELECT Name, DateOfBirth, CAST(DateOfBirth AS DATE) AS [DatePart]

FROM Employees

WHERE CAST(DateOfBirth AS DATE) = '2017-10-09'

**2. All people who are born between 2 given dates (For example, all people** **born between Nov 1, 2017 and Dec 31, 2017)**

SELECT Name, DateOfBirth, CAST(DateOfBirth AS DATE) AS [DatePart]

FROM Employees

WHERE CAST(DateOfBirth AS DATE) BETWEEN '2017-11-01' AND '2017-12-31'

**3. All people who are born on the same day and month excluding the year (For example, 9th October)**

SELECT Name, DateOfBirth, CAST(DateOfBirth AS DATE) AS [DatePart]

FROM Employees

WHERE DAY(DateOfBirth) = 9 AND Month(DateOfBirth) = 10

**4. All people who are born yesterday, today, tomorrow, last seven days, and next 7 days**

**--yesterday**

SELECT Name, DateOfBirth, CAST(DateOfBirth AS DATE) AS [DatePart]

FROM Employees

WHERE CAST(DateOfBirth AS DATE) = DATEADD(DAY, -1, CAST(GETDATE() AS DATE))

**--tomorrow**

SELECT Name, DateOfBirth, CAST(DateOfBirth AS DATE) AS [DatePart]

FROM Employees

WHERE CAST(DateOfBirth AS DATE) = DATEADD(DAY, 1, CAST(GETDATE() AS DATE))

**--yesterday and today\since yesterday**

SELECT Name, DateOfBirth, CAST(DateOfBirth AS DATE) AS [DatePart]

FROM Employees

WHERE CAST(DateOfBirth AS DATE)

BETWEEN DATEADD(DAY, -1, CAST(GETDATE() AS DATE))

AND CAST(GETDATE() AS DATE)

**--last 7 days (excluding today)**

SELECT Name, DateOfBirth, CAST(DateOfBirth AS DATE) AS [DatePart]

FROM Employees

WHERE CAST(DateOfBirth AS DATE)

BETWEEN DATEADD(DAY, -7, CAST(GETDATE() AS DATE))

AND DATEADD(DAY, -1, CAST(GETDATE() AS DATE))

**--today**

SELECT Name, DateOfBirth, CAST(DateOfBirth AS DATE) AS [DatePart]

FROM Employees

WHERE CAST(DateOfBirth AS DATE) = CAST(GETDATE() AS DATE)

**5. All people whose birth year is the same (For example, all people born in 2017, 2018 etc.)**

SELECT Name, DateOfBirth, CAST(DateOfBirth AS DATE) AS [DatePart]

FROM Employees

WHERE YEAR(DateOfBirth) = 2017

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Sql query to delete from multiple tables\ delete cascade**

Alter table Employees

add constraint FK\_Dept\_Employees\_Cascade\_Delete

foreign key (DeptId) references Departments(Id) on delete cascade

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Sql function to get number from string**

Create function UDF\_ExtractNumbers

(

-- Input is alphanumeric string

@input varchar(255)

)

-- Returns numbers as a string

Returns varchar(255)

As

Begin

-- Returns the index of a character that is not a number

-- If the specified pattern is not found, ZERO is returned

Declare @alphabetIndex int = Patindex('%[^0-9]%', @input)

Begin

While @alphabetIndex > 0

Begin

-- In the input string (@input) at the position (@alphabetIndex)

-- where we have a non-numeric chracter, replace that 1

-- character with an empty string ('')

Set @input = Stuff(@input, @alphabetIndex, 1, '' )

-- Find the next non-numeric character and repeat the above step

-- until all non-numeric characters are replaced with an empty string

Set @alphabetIndex = Patindex('%[^0-9]%', @input )

End

End

Return @input

End

Select dbo.UDF\_ExtractNumbers(IDName) as ID as Numbers from TestTable

Create function UDF\_ExtractAlphabets

(

-- Input is alphanumeric string

@input varchar(255)

)

-- Returns numbers as a string

Returns varchar(255)

As

Begin

-- Returns the index of a character that is not an alphabet

-- If an alphabet is not found, ZERO is returned

Declare @numberIndex int = Patindex('%[^a-zA-Z]%', @input)

Begin

While @numberIndex > 0

Begin

-- In the input string (@input) at the position (@numberIndex)

-- where we have an alphabetic chracter, replace that 1 alphabetic

-- character with an empty string ('')

Set @input = Stuff(@input, @numberIndex, 1, '' )

-- Find the next alphabetic character and repeat the above step

-- until all alphabetic characters are replaced with an empty string

Set @numberIndex = Patindex('%[^a-zA-Z]%', @input )

End

End

Return @input

End

Select dbo.UDF\_ExtractAlphabets(IDName) as Name from TestTable

Select dbo.UDF\_ExtractNumbers(IDName) as ID, dbo.UDF\_ExtractAlphabets(IDName) as Name from TestTable

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sql server select where in list

Declare @FirstNamesList nvarchar(100) = 'Mark,John,Sara'

SELECT \* FROM Employees where FirstName IN (SELECT \* FROM STRING\_SPLIT(@FirstNamesList, ','))

Declare @FirstNamesList nvarchar(100) = 'Mark,John,Sara'

SELECT Employees.\* FROM Employees

JOIN STRING\_SPLIT(@FirstNamesList, ',') Result

ON Result.VALUE = Employees.FirstName

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sql select most repeated value

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*