Capgemini

SQL Server 2012 – Database Development

Lesson 4: Beginning with Transact-SQL

Lesson Objectives

- In this lesson, you will learn:
  - Transact-SQL Programming Language
  - Types of Transact-SQL Statements
  - Transact-SQL Syntax Elements
  - Restricting rows
  - Operators
  - Functions – String, Date, Mathematical, System, Others
  - Grouping and summarizing data

Transact-SQL Programming Language

- Transact SQL, also called T-SQL, is Microsoft's extension to the ANSI SQL language
- Structured Query Language(SQL), is a standardized computer language that was originally developed by IBM
- T-SQL expands on the SQL standard to include procedural programming, local variables, various support functions for string processing, date processing, mathematics, etc.
- Transact-SQL is central to using SQL Server
- All applications that communicate with SQL Server do so by sending Transact-SQL statements to the server, regardless of the user interface of the application

Querying Data – SELECT Statement

- The Transact-SQL language has one basic statement for retrieving information from a database: the SELECT statement
- With this statement, it is possible to query information from one or more tables of a database
- The result of a SELECT statement is another table, also known as a result set
- SELECT Statement Syntax

```
SELECT [DISTINCT][TOP n]   <columns >
[FROM] <table names>
[WHERE] <criteria that must be true for a row to be chosen>
[GROUP BY] <columns for grouping aggregate functions>
[HAVING] <criteria that must be met for aggregate functions>
[ORDER BY] <optional specification of how the results should be sorted>
```

SELECT statements primary properties

- Number and attributes of the columns
- The following attributes must be defined for each result set column
  - The data type of the column.
  - The size of the column, and for numeric columns, the precision and scale.
  - The source of the data values returned in the column
- Tables from which the result set data is retrieved
- Conditions that the rows in the source tables must meet
- Sequence in which the rows of the result set are ordered

SELECT statement

- Simple query that retrieves specific columns of all rows from a table

```
Use pubs
GO
SELECT  au_lname, au_fname, city, state, zip
FROM authors
GO
```

- Using WHERE clause

```
SELECT au_lname, au_fname, city, state, zip
FROM authors
WHERE au_lname='Ringer'
GO
```

SELECT statement – Order By Clause

- Specifies the sort order used on columns returned in a SELECT statement

```
USE Northwind
GO
SELECT ProductId, ProductName, UnitPrice
FROM Products
ORDER BY ProductName ASC
GO
```

Use of DISTINCT

- DISTINCT is used to eliminate duplicate rows
- Precedes the list of columns to be selected from the table(s)
- The DISTINCT considers the values of all the columns as a single unit and evaluates on a row-by-row basis to eliminate any redundant rows

- Example

```
SELECT DISTINCT Region
FROM  Northwind.dbo.Employees
```

Demo

- Using SELECT Statement

Use of Operators

- An operator is a symbol specifying an action that is performed on one or more expressions.
- Arithmetic Operators
- Logical Operators
- Assignment Operator
- String Concatenation Operator
- Comparison Operators
- Compound Assignment Operator

Arithmetic Operators

+ (Add)	Addition
- (Subtract)	Subtraction
* (Multiply)	Multiplication
/ (Divide)	Division
% (Modulo)	Returns the integer remainder of a division

Logical Operators

Operator	Meaning
ALL	TRUE if all of a set of comparisons are TRUE
AND	TRUE if both Boolean expressions are TRUE
ANY	TRUE if any one of a set of comparisons are TRUE
BETWEEN	TRUE if the operand is within a range
EXISTS	TRUE if a subquery contains any rows
IN	TRUE if the operand is equal to one of a list of expressions
LIKE	TRUE if the operand matches a pattern.
NOT	Reverses the value of any other Boolean operator
OR	TRUE if either Boolean expression is TRUE.

Like Operators

- Pattern: The pattern to search for in match_expression.

Wildcard character	Example
% Any string of zero or more characters.	WHERE title LIKE '%computer%' finds all book titles with the word 'computer' anywhere in the book title
_ (underscore)Any single character.	WHERE au_fname LIKE '_ean' finds all four-letter first names that end with ean, such as Dean or Sean.

Working with NULL Values

- NULL values are treated differently from other values
- NULL is used as a placeholder for unknown or inapplicable values
- We have to use the IS NULL and IS NOT NULL operators to test for NULL Values

```
SELECT LastName, FirstName, Address
    FROM Persons
WHERE Address IS NULL
```

```
SELECT LastName, FirstName, Address
    FROM Persons
WHERE Address IS NOT NULL
```

Assignment Operator

- Can create a variable
- Sets a value returned by an expression
- Can be used to establish the relationship between a column heading and the expression that defines the values for the column

```
USE AdventureWorks;
GO
SELECT FirstColumnHeading = 'xyz',
SecondColumnHeading = ProductID
FROM Products;
GO
```

Comparison Operator

=	(Equals) Equal to
>	(Greater Than) Greater than
<	(Less Than) Less than
>=	(Greater Than or Equal To) Greater than or equal to
<=	(Less Than or Equal To) Less than or equal to
<>	(Not Equal To) Not equal to
!=	(Not Equal To) Not equal to
!<	(Not Less Than) Not less than
!>	(Not Greater Than) Not greater than

Compound Assignment Operators

+=	Plus Equals
-=	Minus Equals
*=	Multiplication Equals
/=	Division Equals
%=	Modulo Equals

Demo

- Using LIKE operator
- Working with commonly used operators

Using System Functions

- String Functions
- Date and Time Functions
- Mathematical Functions
- Aggregate Functions
- System Functions

Using System Functions – String Functions

- STR - Returns character data converted from numeric data
- REPLACE - Replaces all occurrences of a specified string value with another string value
- LEFT - Returns the left part of a character string with the specified number of characters
- RIGHT - Returns the right part of a character string with the specified number of characters

Using System Functions – String Functions

- SUBSTRING - Returns part of a character, binary, text, or image expression
- LEN - Returns the number of characters of the specified string expression, excluding trailing blanks
- REVERSE - Returns the reverse of a character expression
- LOWER - Returns a character expression after converting uppercase character data to lowercase
- UPPER - Returns a character expression with lowercase character data converted to uppercase
- + -- used for concatenating strings

Using System Functions – Date Functions

- GETDATE - Returns the current database system timestamp as a datetime value without the database time zone offset
- GETUTCDATE - Returns the current database system timestamp as a datetime value. The database time zone offset is not included
- CURRENT_TIMESTAMP - Returns the current database system timestamp as a datetime value without the database time zone offset
- SYSDATETIME - Returns a datetime2(7) value that contains the date and time of the computer on which the instance of SQL Server is running. The time zone offset is not included

Using System Functions – Date Functions

- SYSDATETIMEOFFSET - Returns a datetimeoffset(7) value that contains the date and time of the computer on which the instance of SQL Server is running. The time zone offset is included.

- SYSUTCDATETIME - Returns a datetime2(7) value that contains the date and time of the computer on which the instance of SQL Server is running. The date and time is returned as UTC time (Coordinated Universal Time).

Date Functions – To retrieve Date & Time Parts

- DATENAME - Returns a character string that represents the specified datepart of the specified date
- DATEPART - Returns an integer that represents the specified datepart of the specified date

datepart	Return value – DATEPART	Return Value - DATENAME
year, yyyy, yy	2007	2007
quarter, qq, q	4	4
month, mm, m	9	October
dayofyear, dy, y	303	303
day, dd, d	30	30
week, wk, ww	44	44
weekday, dw	3	Tuesday
hour, hh	12	12
minute, n	15	15
second, ss, s	32	32
millisecond, ms	123	123
microsecond, mcs	123456	123456

Using System Functions – Date Functions

- DATEDIFF - Returns the count (signed integer) of the specified datepart boundaries crossed between the specified startdate and enddate.

- DATEADD- Returns a specified date with the specified number interval (signed integer) added to a specified datepart of that date.

Using System Functions – Mathematical Functions

- ABS - A mathematical function that returns the absolute (positive) value of the specified numeric expression
- RAND - Returns a random float value from 0 through 1
- ROUND - Returns a numeric value, rounded to the specified length or precision
- SQRT - Returns the square root of the specified float value

Using System Functions – Aggregate Functions

- The aggregate functions are: sum, avg, count, min, max, and count(*)
- Aggregate functions are used to calculate and summarize data

```
USE pubs
GO

SELECT AVG(price * 2)
FROM titles
GO

SELECT  MAX(price) as Maxprice,MIN(price) as Minprice
FROM  titles
GO
```

System Functions – To retrieve System Information

- CURRENT_TIMESTAMP - Returns the current date and time, equivalent to GETDATE.
- CURRENT_USER - Returns the name of the current user, equivalent to USER_NAME().
- HOST_ID & HOST_NAME - Returns the workstation identification number and name.

System Functions – To retrieve System Information

- CAST and CONVERT - Explicitly converts an expression of one data type to another
- CAST (expression AS data_type [(length)])
- CONVERT (data_type [(length)] , expression [, style])

Example

```
Select CONVERT(char,100)     --converts 100 to '100'
Select CAST(100 as char)
```

Demo

- Working with commonly used Functions

Organizing Query result into Groups

- Using group by clause
- Group by clause divides the output of a query into groups
- Can group by one or more column names

Example

lists no of employees in each region

```
SELECT  Region, count(EmployeeID)
FROM Northwind.dbo.Employees
GROUP by REGION
GO
```

Groups By

- SQL standards for group by are more restrictive
- SQL standard requires that:
  - Columns in a select list must be in the group by expression or they must be arguments of aggregate functions
  - A group by expression can only contain column names in the select list

```
USE pubs;
GO
select pub_id, type, avg(price), sum(ytd_sales)
from titles
group by pub_id, type
GO
```

Using Aggregation with Groups

```
USE pubs;
GO
select type, sum(advance)
from titles
group by type;
GO
```

Selecting Group

- Use the having clause to display or reject rows defined by the group by clause

  Example

```
USE pubs;
GO
select type
from titles
group by type
having count(*) > 1
GO
```

```
USE pubs;
GO
select type
from titles
where count(*)> 1
GO
```

Grouping Sets

- SQL Server 2008 introduces several extensions to the GROUP BY clause that enable you to define multiple groupings in the same query
- We can use grouping set for single result set instead of using UNION ALL with multiple queries for various grouping sets for various calculations
- SQL Server optimizes the data for access and grouping

Grouping Sets

- Example – Grouping Sets equivalent to UNION ALL

```
SELECT customer, NULL as
year, SUM(sales) FROM T
GROUP BY customer
UNION ALL SELECT NULL
as customer, year,
SUM(sales) FROM T GROUP
BY year
```

```
SELECT customer, year,
SUM(sales) FROM T
GROUP BY GROUPING
SETS ((customer),
(year))
```

Demo

- Working with Group By

Introduction

- SET operators are mainly used to combine the same type of data from two or more tables into a single result
- SET Operators supported in SQL Server are
  - UNION /UNION ALL
  - INTERSECT
  - EXCEPT

Introduction (Contd…)

- UNION /UNION ALL
  - Combine two or more result sets into a single set, without duplicates.
  - The number of columns have to match
  - UNION ALL works exactly like UNION except that duplicates are NOT removed

```
SELECT ProductID,ProductName FROM Products
WHERE categoryID=1234
UNION
SELECT ProductID,ProductName FROM Products
WHERE categoryID=5678
GO
```

Introduction (Contd…)

- INTERSECT
  - Takes the data from both result sets which are in common.
  - All the other conditions remain same

```
SELECT CustomerID FROM Customers
INTERSECT
SELECT CustomerID FROM Orders
```

- EXCEPT
  - Takes the data from first result set, which is not available in the second
  - It is like a complement operation

```
SELECT CustomerID FROM Customers
EXCEPT
SELECT CustomerID FROM Orders
```

Rules of Set Operation

- The result sets of all queries must have the same number of columns.
- In every result set the data type of each column must match the data type of its corresponding column in the first result set.
- In order to sort the result, an ORDER BY clause should be part of the last statement.
- The records from the top query must match the positional ordering of the records from the bottom query.
- The column names or aliases of the result set are given in the first select statement

Summary

- Transact-SQL is central to using SQL Server
- SELECT is the basic & commonly used data retrieval statement used in SQL Server
- SQL Server provides variety of System functions which can help us in performing our day to day activities
- For example, String Functions, Date Functions, Mathematical functions, Aggregate Functions and so on
- We can make use of Group By to divide the SQL query result into groups
- We can use grouping set for single result set instead of using UNION ALL with multiple queries for various grouping sets for various calculations

Review Question

- Question 1: _____ is central to using SQL Server
- Question 2: _____ returns the current database system timestamp as a datetime value without the database time zone offset
- Question 3: We can use _____ _____ for single result set instead of using UNION ALL with multiple queries

Review Question

- Question 1: The Set operation that will show all the rows from both the resultsets including duplicates is _____
  - Option 1: Union All
  - Option 2: Union
  - Option 3: Intersect
  - Option 4: Minus

- Question 2: The Except operator returns _____