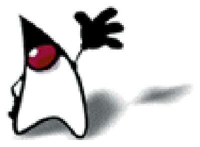
Thang Long University

Advanced querying with T-SQL

Giảng viên: Trần Quang Duy



13-Apr-09

Objectives

- Objective
    - Understanding T - SQL languages
    - Understanding Logical Query Processing
    - Advanced querying techniques using Transact-SQL

Content

Basic Concept

- SQL Server Tables
  - Constraints
  - Identity Columns
  - Computed Columns
  - User-Defined Data Types
  - Adding and Modifying Indexes
- Table Relationships
  - One-to-Many
  - One-to-One
  - Many-to-Many

Basic Concept - SQL Server Tables

- Constraints
  - Primary Key
  - Foreign Key
- Identity Columns
- Computed Columns
- User-Defined Data Types
- Adding and Modifying Indexes

Basic Concept - Table Relationships

- Ensures
  - Data integrity
  - Optimal performance
  - Ease of use in designing system objects
- Types of relationships
  - One-to-Many
  - One-to-One
  - Many-to-Many
- Database Diagrams

Query Data

```
SELECT DISTINCT <TOP_specification> <select_list>
FROM <left_table>
<join_type> JOIN <right_table>
ON <join_condition>
WHERE <where_condition>
GROUP BY <group_by_list>
WITH {CUBE | ROLLUP}
HAVING <having_condition>
ORDER BY <order_by_list>
```

Select Clause

- Syntax:
  - SELECT column-list
- Example
  - SELECT * FROM Customers
  - SELECT CustomerID, CompanyName FROM Customers
  - SELECT CustomerID, City + ', ' + Region + ' ' + PostalCode AS Address FROM Customers

From Clause

- Syntax:
  - FROM table-list [AS alias]
- Example
  - SELECT CustomerID, CompanyName FROM Customers
  - SELECT CustomerID, CompanyName FROM Customers AS Clients

Where Clause

- Syntax
  - WHERE expression1 [{AND|OR} expression2 [...]]
- Example
  - WHERE Country = 'USA' AND ContactTitle Like 'Sales%'
  - WHERE Country = 'USA' OR Country = 'Canada'
  - WHERE Country = 'USA' OR ContactTitle Like 'Sales%'
  - SELECT CustomerID, CompanyName FROM Customers WHERE Country = 'USA' OR Country = 'Canada'

ORDER BY

- Syntax
  - ORDER BY column1 [{ASC|DESC}], column2 [{ASC|DESC}] [,...]]
- Example
  - SELECT CustomerID, CompanyName FROM Customers WHERE Country = 'USA' OR Country = 'Canada'   ORDER BY CompanyName
  - SELECT CustomerID, CompanyName FROM Customers ORDER BY CustomerID DESC

DDL Language

- CREATE TABLE

```
CREATE TABLE dbo.OpenSchema
([objectid]  INT          NOT NULL,
[attribute] NVARCHAR(30) NOT NULL,
 [value]     SQL_VARIANT  NOT NULL,
 PRIMARY KEY (objectid, attribute));
```

- ALTER TABLE

```
ALTER TABLE dbo.OpenSchema
    ALTER COLUMN [attribute] NVARCHAR(50) NOT NULL
```

- DROP TABLE

```
DROP TABLE [dbo].[OpenSchema]
```

DML Language

- INSERT
  - INSERT [INTO] table_or_view [(col1, col2...)]
    VALUES (value1, value2)
  - SELECT INTO … FROM
- UPDATE
  - UPDATE tablename
    SET column1=value1, [column2=value2....]
- DELETE
  - DELETE [FROM] table-name [WHERE ….]
  - TRUNCATE TABLE

Subqueries

- Subqueries: queries that are embedded into other queries
- You can use subqueries
  - single value is expected (scalar subqueries)
  - multiple values (multivalued subqueries)
  - a table (table expressions) in From Clause

Subqueries

- Single Value

```
SELECT OrderID
FROM dbo.Orders
WHERE EmployeeID =
 (SELECT EmployeeID FROM dbo.Employees
  WHERE LastName LIKE N'Davolio');
```

- Multiple values

```
SELECT OrderID, CustomerID, EmployeeID, OrderDate
FROM dbo.Orders WHERE OrderDate IN
   (SELECT MAX(OrderDate) FROM dbo.Orders GROUP BY
  CONVERT(CHAR(6), OrderDate, 112));
```

Subqueries

- Table expressions

```
SELECT OrderYear, COUNT(DISTINCT CustomerID) AS NumCusts
FROM (SELECT YEAR(OrderDate) AS OrderYear, CustomerID
            FROM dbo.Orders) AS D
GROUP BY OrderYear;
```

- Several rules
  - All columns must have names.
  - The column names must be unique.
  - ORDER BY is not allowed (unless TOP is also specified).

Subqueries - table expressions

- Table expressions: Nested

  -
    ```
    SELECT OrderYear, NumCusts
    FROM (SELECT OrderYear, COUNT(DISTINCT CustomerID)AS
                          NumCusts
            FROM  (SELECT YEAR(OrderDate) AS OrderYear, CustomerID
                      FROM  dbo.Orders) AS D1
            GROUP BY OrderYear) AS D2
    WHERE NumCusts > 70;
    ```

Subqueries - Common table expressions

- Common Table Expressions (CTE): new type
- Syntax

```
WITH cte_name
 AS ( cte_query )
outer_query_referring to_cte_name;
```

- Example

```
WITH C
AS ( SELECT YEAR(OrderDate) AS OrderYear, CustomerID
     FROM dbo.Orders )
SELECT OrderYear, COUNT(DISTINCT CustomerID) AS NumCusts
FROM C
GROUP BY OrderYear;
```

T-SQL Functions

- Numeric Functions
- String Functions
- Date/Time Functions
- Ranking Functions

Numeric Functions

Function	Description
Abs()	Returns a number’s absolute value
Cos()	Returns the trigonometric cosine of a specified angle
Exp()	Returns the exponential value of a specific number
IsNumeric ()	Returns information on whether a value is numeric
Pi()	Returns the value of pi
Rand()	Returns a random number
Round()	Returns a number rounded to a specified length or precision
Sin()	Returns the trigonometric sine of a specified angle
Sqrt()	Returns the square root of a specified number
Tan()	Returns the trigonometric tangent of a specified angle

String Functions

Functions	**Description**
CharIndex()	Returns the position of a specified character within a string
Left()	Returns characters from the left of a string
Len()	Returns the length of a string
Lower()	Converts string to lowercase
LTrim()	Trimswhite space from the left of a string
Replace()	Replaces characters within a string with other specified characters
Right()	Returns characters from the right of a string
RTrim()	Trims white space from the right of a string
Soundex()	Returns a string’s SOUNDEX value
Str()	Converts a numeric value to a string
SubString()	Returns characters from within astring
Upper()	Converts string to uppercase

Date/Time Functions

Functions	**Description**
DateAdd()	Adds to a date (days,weeks,and so on)
DateDiff()	Calculates the difference between two dates
DateName()	Returns a string representation of date parts
DatePart()	Returns parts of a date (day of week,month,year, and so on)
Day()	Returns the day portion of a date
GetDate()	Returns the current date and time
Month()	Returns the month portion of a date
Year()	Returns the year portion of a date

Ranking Functions

Functions	**Description**
ROW_NUMBER()	Returns the sequential number of a row within a partition of a result set
RANK ( )	Returns the rank of each row within the partition of a result set
DENSE_RANK()	Returns the rank of rows within the partition of a result set, without any gaps in the ranking
NTILE()	Distributes the rows in an ordered partition into a specified number of groups

Joins

- Joins: Match rows between tables
- ANSI SQL:1989

  FROM T1, T2 WHERE where_filter
- ANSI SQL:1992
  - FROM T1 <join_type> JOIN T2 ON <on_filter> WHERE where_filter

Join Types

- CROSS JOIN: Cartesian product between two tables

```
SELECT E1.FirstName, E1.LastName AS emp1, E2.FirstName,
                E2.LastName AS emp2
 FROM dbo.Employees AS E1 CROSS JOIN dbo.Employees AS E2;
```

- INNER JOIN: matching rows between two tables

```
SELECT C.CustomerID, CompanyName, OrderID
FROM dbo.Customers AS C JOIN dbo.Orders AS O
            ON C.CustomerID = O.CustomerID
WHERE Country = 'USA';
```

- OUTER JOIN; matching rows from both tables based on some criterion
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN

SET Operations

- UNION
  - UNION DISTINCT
  - UNION ALL
- EXCEPT
  - EXCEPT DISTINCT
  - EXCEPT ALL: not in SQL
- INTERSECT
  - INTERSECT DISTINCT
  - INTERSECT ALL: Not in SQL

Aggregating Data

- GROUP BY
  - GROUP BY group-by-expression1 [,group-by-expression2 [,...]]

    HAVING expression1 [{AND|OR} expression2[...]]
- Aggregate Functions: COUNT, SUM, AVG, MIN, and MAX….

  -
    ```
    SELECT Customers.Country, Customers.City,
                Sum(Orders.Freight) AS SumOfFreight
    FROM Customers INNER JOIN Orders ON
             Customers.CustomerID = Orders.CustomerID
    GROUP BY Customers.Country, Customers.City
    ```

OVER Clause

- OVER: Determines the partitioning and ordering of the rowset before the associated window function is applied
  - OVER ([PARTITION BY *value_expression* , ... [*n*]]
- Example
  - SELECT OrderID,
    Freigh AS cSum, cast (1.* Freight/tsum * 100 AS DECIMAL(5, 2)) AS Perc
    FROM orders,
    (SELECT Sum(Freight) AS tsum FROM orders) AS C
  - SELECT OrderID, Freight AS cSum,
    cast (1.* Freight/Sum(Freight) over() * 100 AS DECIMAL(5, 2)) AS Perc
    FROM orders
  - SELECT CustomerID,OrderID, Freight AS cSum,
    cast (1.* Freight/Sum(Freight) over(partition by CustomerID) * 100 AS DECIMAL(5, 2)) AS Perc
    FROM orders

Pivoting Data

- Pivoting: to rotate rows to columns
  - **<pivot_clause> ::=** (*aggregate_function* (*value_column*) FOR *pivot_column* IN (<column_list>))
- Example

objectid	**attribute**	**value**
1	attr1	ABC
1	attr2	10
1	attr3	2004-01-01
2	attr2	12
2	attr3	2006-01-01
2	attr4	Y
2	attr5	13.700
3	attr1	XYZ
3	attr2	20
3	attr3	2005-01-01

Pivoting Data

- Pivoting: to rotate rows to columns

objectid	**attr1**	**attr2**	**attr3**	**attr4**	**attr5**
1	ABC	10	2004-01-01	NULL	NULL
2	NULL	12	2006-01-01	Y	13.700
3	XYZ	20	2005-01-01	NULL	NULL

Pivoting Data

- Using Case When
  - SELECT objectid,
    MAX(CASE WHEN attribute = 'attr1' THEN value END) AS attr1,
    MAX(CASE WHEN attribute = 'attr2' THEN value END) AS attr2,
    MAX(CASE WHEN attribute = 'attr3' THEN value END) AS attr3,
    MAX(CASE WHEN attribute = 'attr4' THEN value END) AS attr4,
    MAX(CASE WHEN attribute = 'attr5' THEN value END) AS attr5
    FROM dbo.OpenSchema
    GROUP BY objectid;
- Using Pivot
  - SELECT objectid, attr1, attr2, attr3, attr4, attr5
    FROM dbo.OpenSchema
    PIVOT(MAX(value) FOR attribute IN([attr1],[attr2],[attr3],[attr4],[attr5]))
    AS P;

Unpivoting

- Unpivoting: rotating columns to rows.
  - **<unpivot_clause> ::=** (*value_column* FOR *pivot_column* IN (<column_list>))
- Example

```
SELECT custid, orderyear, qty
FROM dbo.PvtCustOrders
UNPIVOT(qty FOR orderyear IN ([2002],[2003],[2004])) AS U
```

CUBE and ROLLUP

- CUBE, ROLLUP
  - GROUP BY [ALL] *group_by_expression* [**,**...*n*]
    [WITH { CUBE | ROLLUP }]]
- CUBE: every possible combination of group and subgroup in the result set
  - SELECT empid, custid, YEAR(orderdate) AS orderyear, SUM(qty) AS totalqty
     FROM dbo.Orders
    GROUP BY empid, custid, YEAR(orderdate)
    WITH CUBE;
- ROLLUP: Groups are summarized in a hierarchical order

```
SELECT YEAR(orderdate) AS orderyear, MONTH(orderdate) AS
   ordermonth, DAY(orderdate) AS orderday, SUM(qty) AS totalqty
FROM dbo.Orders
GROUP BY YEAR(orderdate), MONTH(orderdate), DAY(orderdate)
WITH ROLLUP;
```

TOP

- TOP: used with an ORDER BY clause to limit the result to rows
  - TOP (*expression*) [PERCENT] [WITH TIES]
  - PERCENT : Percent of rows from the result set.
  - WITH TIES: Specifies that additional rows be returned
- Example

  -
    ```
    SELECT TOP(3) OrderID, CustomerID, OrderDate
    FROM dbo.Orders
    ORDER BY OrderDate DESC, OrderID DESC;
    ```

  -
    ```
    SELECT TOP(1) PERCENT OrderID, CustomerID, OrderDate
    FROM dbo.Orders
    ORDER BY OrderDate DESC, OrderID DESC;
    ```

  -
    ```
    SELECT TOP(3) WITH TIES OrderID, CustomerID, OrderDate
    FROM dbo.Orders
    ORDER BY CustomerID;
    ```

APPLY

- APPLY table operator
  - applies the right-hand table expression
- Types of APPLY
  - CROSS APPLY: returns only rows from the outer table that produce a result set from the table-valued function
  - OUTER APPLY : returns both rows that produce a result set, and rows that do not, with NULL values in the columns produced by the table-valued function

APPLY - Example

```
CREATE FUNCTION dbo.fn_top_products
(@supid AS INT, @catid INT, @n AS INT)
RETURNS TABLE
AS
RETURN
SELECT TOP(@n) WITH TIES ProductID,
ProductName, UnitPrice FROM dbo.Products WHERE
SupplierID = @supid AND CategoryID = @catid
ORDER BY UnitPrice DESC;
```

APPLY - Example

-
  ```
  SELECT S.SupplierID, CompanyName, ProductID,
  ProductName, UnitPrice
   FROM dbo.Suppliers AS S
  CROSS APPLY dbo.fn_top_products(S.SupplierID, 1, 2) AS P;
  ```

-
  ```
  SELECT S.SupplierID, CompanyName, ProductID,
  ProductName, UnitPrice
   FROM dbo.Suppliers AS S
   OUTER APPLY dbo.fn_top_products(S.SupplierID, 1, 2) AS P
  ```

Logical Query Processing

(8) SELECT (9) DISTINCT (11) <TOP_specification>
 <select_list>
(1) FROM <left_table>
(3) <join_type> JOIN <right_table>
(2) ON <join_condition>
(4) WHERE <where_condition>
(5) GROUP BY <group_by_list>
(6) WITH {CUBE | ROLLUP}
(7) HAVING <having_condition>
(10) ORDER BY <order_by_list>

Reference

- Books online
- Inside Microsoft® SQL Server™ 2005 T-SQL Querying, Microsoft Press, 2006