07 | Using Table Expressions

Graeme Malcolm | Senior Content Developer, Microsoft
Geoff Allix | Principal Technologist, Content Master

Module Overview

- Views
- Temporary Tables
- Table Variables
- Table-Valued Functions
- Derived Tables
- Common Table Expressions

Querying Views

- Views are named queries with definitions stored in a database
  - Views can provide abstraction, encapsulation and simplification
  - From an administrative perspective, views can provide a security layer to a database
- Views may be referenced in a SELECT statement just like a table

```
CREATE VIEW Sales.vSalesOrders
AS
SELECT  oh.OrderID, oh.Orderdate, oh.CustomerID,
        od.LineItemNo, od.ProductID, od.Quantity
FROM Sales.OrderHeaders AS oh
JOIN Sales.OrderDetails AS od
ON od.OrderID = oh.OrderID;
```

```
SELECT OrderID, CustomerID, ProductID
FROM Sales.vSalesOrder
ORDER BY OrderID;
```

DEMO

Querying Views

Temporary Tables

```
CREATE TABLE #tmpProducts
(ProductID INTEGER,
 ProductName varchar(50));
GO
…
SELECT * FROM #tmpProducts;
```

- Temporary tables are used to hold temporary result sets within a user's session
  - Created in tempdb and deleted automatically
  - Created with a # prefix
  - Global temporary tables are created with ## prefix

Table Variables

```
DECLARE @varProducts table
(ProductID INTEGER,
 ProductName varchar(50));
…
SELECT * FROM @varProducts
```

- Introduced because temporary tables can cause recompilations
- Used similarly to temporary tables but scoped to the batch
- Use only on very small datasets

DEMO

Temporary Tables and Table Variables

Table-Valued Functions

```
CREATE FUNCTION Sales.fn_GetOrderItems (@OrderID AS Integer)
RETURNS TABLE
AS
RETURN
(SELECT ProductID, UnitPrice, Quantity
FROM Sales.OrderDetails
WHERE OrderID = @OrderID);
…
SELECT * FROM Sales.fn_GetOrderItems (1025) AS LineItems;
```

- TVFs are named objects with definitions stored in a database
- TVFs return a virtual table to the calling query
- Unlike views, TVFs support input parameters
  - TVFs may be thought of as parameterized views

DEMO

Using Table-Valued Functions

Derived Tables
Introduction

```
SELECT orderyear, COUNT(DISTINCT custid) AS cust_count
FROM
        (SELECT YEAR(orderdate) AS orderyear, custid
        FROM Sales.Orders) AS derived_year
GROUP BY orderyear;
```

- Derived tables are named query expressions created within an outer SELECT statement
- Not stored in database – represents a virtual relational table
- Scope of a derived table is the query in which it is defined

Derived Tables
Guidelines

- Derived tables **must**:
  - Have an alias
  - Have unique names for all columns
  - Not use an ORDER BY clause (without TOP or OFFSET/FETCH)
  - Not be referred to multiple times in the same query
- Derived tables **may**:
  - Use internal or external aliases for columns
  - Refer to parameters and/or variables
  - Be nested within other derived tables

Derived Tables
Specifying Column Aliases

- Column aliases may be defined inline:

```
SELECT orderyear, COUNT(DISTINCT custid) AS cust_count
FROM (  SELECT YEAR(orderdate) AS orderyear, custid
        FROM Sales.Orders) AS derived_year
GROUP BY orderyear;
```

- Or externally:

```
SELECT orderyear, COUNT(DISTINCT custid) AS cust_count
FROM (  SELECT YEAR(orderdate), custid
        FROM Sales.Orders) AS
        derived_year(orderyear, custid)
GROUP BY orderyear;
```

DEMO

Using Derived Tables

Common Table Expressions (CTEs)

```
WITH CTE_year (OrderYear, CustID)
AS
(
        SELECT YEAR(orderdate), custid
        FROM Sales.Orders
)
SELECT OrderYear, COUNT(DISTINCT CustID) AS Cust_Count
FROM CTE_year
GROUP BY orderyear;
```

- CTEs are named table expressions defined in a query
- CTEs are similar to derived tables in scope and naming requirements
- Unlike derived tables, CTEs support multiple references and recursion

Common Table Expressions
Recursion

```
WITH OrgReport (ManagerID, EmployeeID, EmployeeName, Level)
AS
(
        SELECT e.ManagerID, e.EmployeeID, EmployeeName, 0
        FROM HR.Employee AS e
        WHERE ManagerID IS NULL
        UNION ALL
        SELECT e.ManagerID, e.EmployeeID, e.EmployeeName, Level + 1
        FROM HR.Employee AS e
        INNER JOIN OrgReport AS o ON e.ManagerID = o.EmployeeID
)
SELECT * FROM OrgReport
OPTION (MAXRECURSION 3);
```

- Specify a query for the anchor (root) level
- Use UNION ALL to add a recursive query for other levels
- Query the CTE, with optional MAXRECURSION option

DEMO

Using Common Table Expressions

Using Table Expressions

- Views
- Temporary Tables
- Table Variables
- Table-Valued Functions
- Derived Tables
- Common Table Expressions

- Lab: Using Table Expressions

Microsoft