SQL Server 2005 Books Online (September 2007)

Using APPLY

Send Feedback

See Also

Collapse All

Updated: 14 April 2006

The APPLY operator allows you to invoke a table-valued function for each row returned by an outer table expression of a query. The table-valued function acts as the right input and the outer table expression acts as the left input. The right input is evaluated for each row from the left input and the rows produced are combined for the final output. The list of columns produced by the APPLY operator is the set of columns in the left input followed by the list of columns returned by the right input.

Note:
To use APPLY, the database compatibility level must be 90.

There are two forms of APPLY: CROSS APPLY and OUTER 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.

As an example, consider the following tables, `Employees` and `Departments`:

Copy Code

```
--Create Employees table and insert values.
CREATE TABLE Employees
(
  empid   int         NOT NULL,
  mgrid   int         NULL,
  empname varchar(25) NOT NULL,
  salary  money       NOT NULL,
  CONSTRAINT PK_Employees PRIMARY KEY(empid),
)
GO
INSERT INTO Employees VALUES(1 , NULL, 'Nancy'   , $10000.00)
INSERT INTO Employees VALUES(2 , 1   , 'Andrew'  , $5000.00)
INSERT INTO Employees VALUES(3 , 1   , 'Janet'   , $5000.00)
INSERT INTO Employees VALUES(4 , 1   , 'Margaret', $5000.00)
INSERT INTO Employees VALUES(5 , 2   , 'Steven'  , $2500.00)
INSERT INTO Employees VALUES(6 , 2   , 'Michael' , $2500.00)
INSERT INTO Employees VALUES(7 , 3   , 'Robert'  , $2500.00)
INSERT INTO Employees VALUES(8 , 3   , 'Laura'   , $2500.00)
INSERT INTO Employees VALUES(9 , 3   , 'Ann'     , $2500.00)
INSERT INTO Employees VALUES(10, 4   , 'Ina'     , $2500.00)
INSERT INTO Employees VALUES(11, 7   , 'David'   , $2000.00)
INSERT INTO Employees VALUES(12, 7   , 'Ron'     , $2000.00)
INSERT INTO Employees VALUES(13, 7   , 'Dan'     , $2000.00)
INSERT INTO Employees VALUES(14, 11  , 'James'   , $1500.00)
GO
--Create Departments table and insert values.
CREATE TABLE Departments
(
  deptid    INT NOT NULL PRIMARY KEY,
  deptname  VARCHAR(25) NOT NULL,
  deptmgrid INT NULL REFERENCES Employees
)
GO
INSERT INTO Departments VALUES(1, 'HR',           2)
INSERT INTO Departments VALUES(2, 'Marketing',    7)
INSERT INTO Departments VALUES(3, 'Finance',      8)
INSERT INTO Departments VALUES(4, 'R&D',          9)
INSERT INTO Departments VALUES(5, 'Training',     4)
INSERT INTO Departments VALUES(6, 'Gardening', NULL)
```

Most departments in the `Departments` table have a manager ID that corresponds to an employee in the `Employees` table. The following table-valued function accepts an employee ID as an argument and returns that employee and all of his or her subordinates.

Copy Code

```
CREATE FUNCTION dbo.fn_getsubtree(@empid AS INT) RETURNS @TREE TABLE
(
  empid   INT NOT NULL,
  empname VARCHAR(25) NOT NULL,
  mgrid   INT NULL,
  lvl     INT NOT NULL
)
AS
BEGIN
  WITH Employees_Subtree(empid, empname, mgrid, lvl)
  AS
  (
    -- Anchor Member (AM)
    SELECT empid, empname, mgrid, 0
    FROM Employees
    WHERE empid = @empid

    UNION all

    -- Recursive Member (RM)
    SELECT e.empid, e.empname, e.mgrid, es.lvl+1
    FROM Employees AS e
      JOIN Employees_Subtree AS es
        ON e.mgrid = es.empid
  )
  INSERT INTO @TREE
    SELECT * FROM Employees_Subtree

  RETURN
END
GO
```

To return all of the subordinates in all levels for the manager of each department, use the following query.

Copy Code

```
SELECT *
FROM Departments AS D
  CROSS APPLY fn_getsubtree(D.deptmgrid) AS ST
```

Here is the result set.

Copy Code

```
deptid      deptname   deptmgrid   empid       empname    mgrid       lvl
----------- ---------- ----------- ----------- ---------- ----------- ---
1           HR         2           2           Andrew     1           0
1           HR         2           5           Steven     2           1
```

```
1           HR          2           6           Michael     2           1
2           Marketing   7           7           Robert      3           0
2           Marketing   7           11          David       7           1
2           Marketing   7           12          Ron         7           1
2           Marketing   7           13          Dan         7           1
2           Marketing   7           14          James       11          2
3           Finance     8           8           Laura       3           0
4           R&D         9           9           Ann         3           0
5           Training    4           4           Margaret    1           0
5           Training    4           10          Ina         4           1
```

Notice that each row from the `Departments` table is duplicated as many times as there are rows returned from `fn_getsubtree` for the department's manager.

Also, the `Gardening` department does not appear in the results. Because this department has no manager, `fn_getsubtree` returned an empty set for it. By using `OUTER APPLY`, the `Gardening` department will also appear in the result set, with null values in the `deptmgrid` field, as well as in the fields returned by `fn_getsubtree`.

See Also

Other Resources

FROM (Transact-SQL)

Help and Information

Getting SQL Server 2005 Assistance