CREATE TABLE Sales (EmpId INT, Yr INT, Sales MONEY)  
INSERT Sales VALUES(1, 2005, 12000)  
INSERT Sales VALUES(1, 2006, 18000)  
INSERT Sales VALUES(1, 2007, 25000)  
INSERT Sales VALUES(2, 2005, 15000)  
INSERT Sales VALUES(2, 2006, 6000)  
INSERT Sales VALUES(3, 2006, 20000)  
INSERT Sales VALUES(3, 2007, 24000)

SELECT EmpId, Yr, SUM(Sales) AS Sales  
FROM Sales  
GROUP BY EmpId, Yr WITH ROLLUP

SELECT EmpId, Yr, SUM(Sales) AS Sales  
FROM Sales  
GROUP BY EmpId, Yr WITH CUBE

We can rewrite these two queries using the new syntax as:

SELECT EmpId, Yr, SUM(Sales) AS Sales  
FROM Sales  
GROUP BY ROLLUP(EmpId, Yr)

SELECT EmpId, Yr, SUM(Sales) AS Sales  
FROM Sales  
GROUP BY CUBE(EmpId, Yr)

The new GROUPING SETS syntax is considerably more powerful. It allows us to specify precisely which aggregations we want to compute. As the following table illustrates, our simple two dimensional schema has a total of only four possible aggregations:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **Yr** | | | |
| **2005** | **2006** | **2007** | **ALL** |
| **EmpId** | **1** | GROUP BY (EmpId, Yr) | | | GROUP BY (EmpId) |
| **2** |
| **3** |
| **ALL** | GROUP BY (Yr) | | | GROUP BY () |

ROLLUP and CUBE are just shorthand for two common usages of GROUPING SETS. We can express the above ROLLUP query as:

SELECT EmpId, Yr, SUM(Sales) AS Sales  
FROM Sales  
GROUP BY GROUPING SETS((EmpId, Yr), (EmpId), ())

EmpId Yr Sales

----------- ----------- ---------------------

1 2005 12000.00

1 2006 18000.00

1 2007 25000.00

1 NULL 55000.00

2 2005 15000.00

2 2006 6000.00

2 NULL 21000.00

3 2006 20000.00

3 2007 24000.00

3 NULL 44000.00

NULL NULL 120000.00

This query explicitly asks SQL Server to aggregate sales by employee and year, to aggregate by employee only, and to compute the total for all employees for all years. The () syntax with no GROUP BY columns denotes the total. Similarly, we can express the above CUBE query by asking SQL Server to compute all possible aggregate combinations:

SELECT EmpId, Yr, SUM(Sales) AS Sales  
FROM Sales  
GROUP BY GROUPING SETS((EmpId, Yr), (EmpId), (Yr), ())

EmpId Yr Sales

----------- ----------- ---------------------

1 2005 12000.00

2 2005 15000.00

NULL 2005 27000.00

1 2006 18000.00

2 2006 6000.00

3 2006 20000.00

NULL 2006 44000.00

1 2007 25000.00

3 2007 24000.00

NULL 2007 49000.00

NULL NULL 120000.00

1 NULL 55000.00

2 NULL 21000.00

3 NULL 44000.00

We can also use GROUPING SETS to compute other results. For example, we can perform a partial rollup aggregating sales by employee and year and by employee only but without computing the total for all employees for all years:

SELECT EmpId, Yr, SUM(Sales) AS Sales  
FROM Sales  
GROUP BY GROUPING SETS((EmpId, Yr), (EmpId))

EmpId Yr Sales

----------- ----------- ---------------------

1 2005 12000.00

1 2006 18000.00

1 2007 25000.00

1 NULL 55000.00

2 2005 15000.00

2 2006 6000.00

2 NULL 21000.00

3 2006 20000.00

3 2007 24000.00

3 NULL 44000.00

We can skip certain rollup levels. For example, we can compute the total sales by employee and year and the total sales for all employees and all years without computing any of the intermediate results:

SELECT EmpId, Yr, SUM(Sales) AS Sales  
FROM Sales  
GROUP BY GROUPING SETS((EmpId, Yr), ())

EmpId Yr Sales

----------- ----------- ---------------------

1 2005 12000.00

1 2006 18000.00

1 2007 25000.00

2 2005 15000.00

2 2006 6000.00

3 2006 20000.00

3 2007 24000.00

NULL NULL 120000.00

We can even compute multiple unrelated aggregations along disparate dimensions. For example, we can compute the total sales by employee and the total sales by year:

SELECT EmpId, Yr, SUM(Sales) AS Sales  
FROM Sales  
GROUP BY GROUPING SETS((EmpId), (Yr))

EmpId Yr Sales

----------- ----------- ---------------------

NULL 2005 27000.00

NULL 2006 44000.00

NULL 2007 49000.00

1 NULL 55000.00

2 NULL 21000.00

3 NULL 44000.00

Note that we could also write GROUPING SETS (EmpId, Yr) without the extra set of parenthesis, but the extra parenthesis make the intent of the query more explicit and clearly differentiate the previous query from the following query which just performs a normal aggregation by employee and year:

SELECT EmpId, Yr, SUM(Sales) AS Sales  
FROM Sales  
GROUP BY GROUPING SETS((EmpId, Yr))

EmpId Yr Sales

----------- ----------- ---------------------

1 2005 12000.00

2 2005 15000.00

1 2006 18000.00

2 2006 6000.00

3 2006 20000.00

1 2007 25000.00

3 2007 24000.00

Here are some additional points worth noting about the GROUPING SETS syntax:

As with any other aggregation query, if a column appears in the SELECT list and is not part of an aggregate function, it must appear somewhere in the GROUP BY clause. Thus, the following is not valid:

SELECT EmpId, Yr, SUM(Sales) AS Sales  
FROM Sales  
GROUP BY GROUPING SETS((EmpId), ())

Msg 8120, Level 16, State 1, Line 1  
Column 'Sales.Yr' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

The order of the columns within each GROUPING SET and the order of the GROUPING SETS does not matter. So both of the following queries compute the same CUBE although the order that the rows are output differs:

SELECT EmpId, Yr, SUM(Sales) AS Sales  
FROM Sales  
GROUP BY GROUPING SETS**((EmpId, Yr), (EmpId), (Yr), ())**

SELECT EmpId, Yr, SUM(Sales) AS Sales  
FROM Sales  
GROUP BY GROUPING SETS**((), (Yr), (EmpId), (Yr, EmpId))**

If the order that the rows are output matters, use an explicit ORDER BY clause to enforce that order.

We can nest CUBE and ROLLUP within a GROUPING SETS clause as shorthand for expressing more complex GROUPING SETS. This shorthand is most useful when we have more than three dimensions in our schema. For example, suppose we add a month column to our sales table:

CREATE TABLE Sales (EmpId INT, Month INT, Yr INT, Sales MONEY)

Now, suppose we want to compute sales for each employee by month and year, by year, and total. We could write out all of the GROUPING SETS explicitly:

SELECT EmpId, Month, Yr, SUM(Sales) AS Sales  
FROM Sales  
GROUP BY GROUPING SETS((EmpId, **Yr, Month**), (EmpId, **Yr**), (EmpId))

Or we can use ROLLUP to simplify the query:

SELECT EmpId, Month, Yr, SUM(Sales) AS Sales  
FROM Sales  
GROUP BY GROUPING SETS(**(**EmpId, **ROLLUP(Yr, Month))**)

Note that once again the correct use of parenthesis is critical. If we omit one set of parenthesis from the above query, the meaning changes significantly and we end up separately aggregating by employee and then computing the year and month ROLLUP for all employees.