Summarizing aggregated data, Part 2

SQL Server's CUBE clause lets you summarize data on all dimensions at once, while GROUPING SETS lets you get whichever summaries you want.

Tamar E. Granor, Ph.D.

In my last article, I introduced the ROLLUP keyword that lets a single query aggregate data and then summarize those aggregations. The CUBE keyword takes that one step farther. GROUPING SETS goes even farther, letting you specify exactly which combinations to summarize; it also lets you summarize without holding on the original aggregated data.

When you use GROUP BY in a query, you get aggregated results. That is, the result contains one record totalling or counting or averaging or whatevering all the records that match in the specified fields. So you can, for example, count and total all invoices by month, or, as in Listing 1 (SalesByCountryCity.sql in this month's downloads), do that for each combination of city and month. Partial results are shown in Figure 1.

Name	City	nYear	nMonth	TotalSales	AvgSale	NumSales
Australia	Wollongong	2007	11	6691.94	836.4925	8
Australia	Wollongong	2007	12	58584.68	1105.3713	53
Australia	Wollongong	2008	1	46460.98	1548.6993	30
Australia	Wollongong	2008	2	29543.26	1477.163	20
Australia	Wollongong	2008	3	46330.36	1494.5277	31
Australia	Wollongong	2008	4	42357.88	1033.119	41
Australia	Wollongong	2008	5	39493.51	1128.386	35
Australia	Wollongong	2008	6	64000.30	1280.006	50
Australia	Wollongong	2008	7	367.84	91.96	4
Canada	Burnaby	2005	8	3578.27	3578.27	1
Canada	Burnaby	2006	3	3578.27	3578.27	1
Canada	Burnaby	2006	5	3578.27	3578.27	1
Canada	Burnaby	2006	10	1000.4375	1000.4375	1
Canada	Burnaby	2006	11	2049.0982	2049.0982	1

Figure 1. The query in **Listing 1** computes the total, average and count for each combination of country, city, year and month.

Listing 1. This query, using SQL Server's sample AdventureWorks 2008 database, computes sales totals by city and month.

```
SELECT Person.CountryRegion.Name,
       Person.Address.City,
       YEAR(OrderDate) AS nYear,
       MONTH(OrderDate) AS nMonth,
       SUM(SubTotal) AS TotalSales
       AVG(SubTotal) AS AvgSale,
       COUNT(*) AS NumSales
  FROM Sales.Customer
    JOIN Person.Person
      ON Customer.PersonID =
         Person.BusinessEntityID
    JOIN Person.BusinessEntityAddress
      ON Person.BusinessEntityID =
        BusinessEntityAddress.BusinessEntityID
    JOIN Person.Address
      ON BusinessEntityAddress.AddressID =
         Address.AddressID
    JOIN Person.StateProvince
      ON Address.StateProvinceID =
         StateProvince.StateProvinceID
    JOIN Person.CountryRegion
      ON StateProvince.CountryRegionCode =
         CountryRegion.CountryRegionCode
    JOIN Sales.SalesOrderHeader
      ON Customer.CustomerID =
         SalesOrderHeader.CustomerID
    JOIN Sales.SalesOrderDetail
      ON SalesOrderHeader.SalesOrderID =
         SalesOrderDetail.SalesOrderID
GROUP BY CountryRegion.Name, Address.City,
         YEAR(OrderDate), MONTH(OrderDate))
```

In last month's article, I showed how the ROLLUP clause lets you include in the results summaries for various subsets, such as an entire year for one city. Listing 2 shows one of the examples from that article; it provides summaries by year for each city, by city for the whole period, by country and for the whole data set. Partial results are shown in Figure 2.

Listing 2. The ROLLUP clause lets you summarize results in a grouped query.

```
SELECT Person.CountryRegion.Name,
       Person.Address.City,
       YEAR(OrderDate) AS nYear,
       MONTH(OrderDate) AS nMonth,
       SUM(SubTotal) AS TotalSales,
       AVG(SubTotal) AS AvgSale,
       COUNT(SubTotal) AS NumSales
  FROM Sales.Customer
    JOIN Person.Person
      ON Customer.PersonID =
         Person.BusinessEntityID
    JOIN Person.BusinessEntityAddress
      ON Person.BusinessEntityID =
        BusinessEntityAddress.BusinessEntityID
    JOIN Person.Address
      ON BusinessEntityAddress.AddressID =
         Address.AddressID
    JOIN Person.StateProvince
      ON Address.StateProvinceID =
         StateProvince.StateProvinceID
    JOIN Person.CountryRegion
      ON StateProvince.CountryRegionCode =
         CountryRegion.CountryRegionCode
```

```
    JOIN Sales.SalesOrderHeader
      ON Customer.CustomerID =
         SalesOrderHeader.CustomerID
    JOIN Sales.SalesOrderDetail
      ON SalesOrderHeader.SalesOrderID =
         SalesOrderDetail.SalesOrderID
  GROUP BY ROLLUP(CountryRegion.Name,
           Address.City, YEAR(OrderDate),
           MONTH(OrderDate))
```

Australia	Bendigo	2008	6	41047.22	1282.7256	32
Australia	Bendigo	2008	7	307.87	61.574	5
Australia	Bendigo	2008	NULL	234093.91	1200.4815	195
Australia	Bendigo	NU...	NULL	555431.9...	1402.606	396
Australia	Brisbane	2005	7	3578.27	3578.27	1
Australia	Brisbane	2005	8	7156.54	3578.27	2
Australia	Brisbane	2005	10	14313.08	3578.27	4
Australia	Brisbane	2005	11	7156.54	3578.27	2
Australia	Brisbane	2005	12	3374.99	3374.99	1
Australia	Brisbane	2005	NULL	35579.42	3557.942	10
Australia	Brisbane	2006	3	13931.52	3482.88	4
Australia	Brisbane	2006	4	10531.53	3510.51	3

Figure 2. With ROLLUP, summaries are provided for each level you specify.

Introducing CUBE

ROLLUP is limited to summarizing only based on the hierarchy you specify. For example, the query in Listing 2 doesn't give summaries for each country for each year. While you can get that result with ROLLUP, you have to give up some other summaries to do so.

If you want to summarize based on every possibile combination of values, use CUBE rather than ROLLUP. The query in **Listing 3** is identical to the one in Listing 2, except that the GROUP BY clause specifies CUBE rather than ROLLUP. **Figure 3** shows part of the results. The items at the top of the grid include summaries you wouldn't get with ROLLUP, such as the summary for all locations in all Decembers about halfway down and the summary for Australia for all of 2005 in the last row shown.This query is included in this month's downloads as SalesByCountryCityCubeNoOrder.sql.

Listing 3. Use the CUBE clause to get summaries for all combinations of values.

```
SELECT Person.CountryRegion.Name,
       Person.Address.City,
       YEAR(OrderDate) AS nYear,
       MONTH(OrderDate) AS nMonth,
       SUM(SubTotal) AS TotalSales,
       AVG(SubTotal) AS AvgSale,
       COUNT(SubTotal) AS NumSales
  FROM Sales.Customer
    JOIN Person.Person
      ON Customer.PersonID =
         Person.BusinessEntityID
    JOIN Person.BusinessEntityAddress
      ON Person.BusinessEntityID =
        BusinessEntityAddress.BusinessEntityID
    JOIN Person.Address
      ON BusinessEntityAddress.AddressID =
         Address.AddressID
    JOIN Person.StateProvince
      ON Address.StateProvinceID =
         StateProvince.StateProvinceID
    JOIN Person.CountryRegion
      ON StateProvince.CountryRegionCode =
         CountryRegion.CountryRegionCode
    JOIN Sales.SalesOrderHeader
      ON Customer.CustomerID =
         SalesOrderHeader.CustomerID
    JOIN Sales.SalesOrderDetail
      ON SalesOrderHeader.SalesOrderID =
         SalesOrderDetail.SalesOrderID
  GROUP BY CUBE(CountryRegion.Name,
                Address.City, YEAR(OrderDate),
                MONTH(OrderDate))
```

Name	City	nYear	nMonth	TotalSales	AvgSale	NumSales
United Kingdom	Woolston	2007	12	16936.33	1129.0886	15
NULL	Woolston	2007	12	16936.33	1129.0886	15
United States	Yakima	2007	12	22768.39	875.7073	26
NULL	Yakima	2007	12	22768.39	875.7073	26
United Kingdom	York	2007	12	50720.73	1334.756	38
NULL	York	2007	12	50720.73	1334.756	38
NULL	NULL	2007	12	4899275...	927.3661	5283
NULL	NULL	NU...	12	6233117...	1065.6723	5849
NULL	NULL	NU...	NULL	5927376...	980.3961	60459
Australia	NULL	2005	7	209652....	3381.4984	62
Australia	NULL	2005	8	222538....	3272.6219	68
Australia	NULL	2005	9	173993....	3346.029	52
Australia	NULL	2005	10	217993....	3353.7443	65
Australia	NULL	2005	11	210683....	3344.1835	63
Australia	NULL	2005	12	274185....	3264.1136	84
Australia	NULL	2005	NULL	1309047...	3322.4548	394

Figure 3. When you specify CUBE, every possible combination of values is summarized.

However, some of the results of this query are misleading. The first few rows in Figure 3 should give you a clue as to the problem. We're summarizing by name of a city for a month. What if we have multiple cities with the same name? In fact, this data set contains several repeated city names, among them Birmingham. Figure 4. shows that when both Birminghams have data for a given month, we get a total for that month that covers both cities, which is meaningless.

Name	City	nYear	nMonth	TotalSales	AvgSale	NumSales
United Kingdom	Birmingham	NULL	6	9782.32	543.4622	18
NULL	Birmingham	NULL	6	9782.32	543.4622	18
United Kingdom	Birmingham	NULL	7	8189.09	1169.87	7
United States	Birmingham	NULL	7	35.00	35.00	1
NULL	Birmingham	NULL	7	8224.09	1028.0112	8
United Kingdom	Birmingham	NULL	8	1714.0957	428.5239	4
NULL	Birmingham	NULL	8	1714.0957	428.5239	4
United Kingdom	Birmingham	NULL	9	7298.8825	1042.6975	7
NULL	Birmingham	NULL	9	7298.8825	1042.6975	7
United Kingdom	Birmingham	NULL	10	13124.3075	1009.5621	13
NULL	Birmingham	NULL	10	13124.3075	1009.5621	13
United Kingdom	Birmingham	NULL	11	6083.63	675.9588	9
United States	Birmingham	NULL	11	2.29	2.29	1
NULL	Birmingham	NULL	11	6085.92	608.592	10
United Kingdom	Birmingham	NULL	12	17644.17	1604.0154	11
NULL	Birmingham	NULL	12	17644.17	1604.0154	11

Figure 4. Some of the summarized results can be misleading if fields are dependent on each other. Here, we get totals for a given month for both Birminghams.

The way to avoid the problem is to group fields together if their data is linked. You do that by putting parentheses around the fields to be grouped. Listing 4 shows the same query, but with the Name

(that is, Country) and City fields grouped together. (It also has an ORDER BY clause to sort the results into a useful order.) It's included in this month's downloads as SalesByCountryCityCubeCombined.sql. Figure 5 shows partial results; note that there are no totals where Name is null, but City is not.

Listing 4. Group fields with parentheses in the CUBE clause to have them treated as a single dimension.

```
SELECT Person.CountryRegion.Name,
       Person.Address.City,
       YEAR(OrderDate) AS nYear,
       MONTH(OrderDate) AS nMonth,
       SUM(SubTotal) AS TotalSales,
       AVG(SubTotal) AS AvgSale,
       COUNT(SubTotal) AS NumSales
  FROM Sales.Customer
    JOIN Person.Person
      ON Customer.PersonID =
         Person.BusinessEntityID
    JOIN Person.BusinessEntityAddress
      ON Person.BusinessEntityID =
        BusinessEntityAddress.BusinessEntityID
    JOIN Person.Address
      ON BusinessEntityAddress.AddressID =
         Address.AddressID
    JOIN Person.StateProvince
      ON Address.StateProvinceID =
         StateProvince.StateProvinceID
    JOIN Person.CountryRegion
      ON StateProvince.CountryRegionCode =
         CountryRegion.CountryRegionCode
    JOIN Sales.SalesOrderHeader
      ON Customer.CustomerID =
         SalesOrderHeader.CustomerID
    JOIN Sales.SalesOrderDetail
      ON SalesOrderHeader.SalesOrderID =
         SalesOrderDetail.SalesOrderID
  GROUP BY CUBE((CountryRegion.Name,
                 Address.City),
                YEAR(OrderDate),
                MONTH(OrderDate))
  ORDER BY Name, City, nYear, nMonth
```

Name	City	nYear	nMonth	TotalSales	AvgSale	NumSales
NULL	NULL	2008	6	5534993.75	996.7573	5553
NULL	NULL	2008	7	137184.87	62.0465	2211
Australia	Bendigo	NULL	NULL	555431.9893	1402.606	396
Australia	Bendigo	NULL	1	45717.8121	1576.4762	29
Australia	Bendigo	NULL	2	35863.8521	1434.554	25
Australia	Bendigo	NULL	3	47283.4582	1432.832	33
Australia	Bendigo	NULL	4	63837.0056	1329.9376	48
Australia	Bendigo	NULL	5	61997.1378	1265.2477	49
Australia	Bendigo	NULL	6	63823.672	1556.6749	41
Australia	Bendigo	NULL	7	60588.16	1731.0902	35
Australia	Bendigo	NULL	8	28585.1242	1242.8314	23
Australia	Bendigo	NULL	9	23499.03	1382.2958	17
Australia	Bendigo	NULL	10	52875.3221	1429.0627	37
Australia	Bendigo	NULL	11	34616.54	1081.7668	32
Australia	Bendigo	NULL	12	36744.8752	1360.9213	27
Australia	Bendigo	2005	NULL	41972.84	3497.7366	12
Australia	Bendigo	2005	7	20909.78	3484.9633	6
Australia	Bendigo	2005	9	3578.27	3578.27	1
Australia	Bendigo	2005	10	6953.26	3476.63	2

Figure 5. With country and city grouped, the results don't have totals for a city without the associated country.

If you don't want summaries for each month across the years (that is, for example, for all Aprils), you can group year and month in the CUBE clause, as well, as in Listing 5. A query that uses this CUBE clause is included in this month's downloads as SalesByCountryCityCubeCombinedBoth.sql.

Listing 5. You can have multiple groups of fields within the CUBE clause.

```
GROUP BY CUBE(
  (CountryRegion.Name, Address.City),
  (YEAR(OrderDate), MONTH(OrderDate)))
```

Fine tuning the set of summaries

ROLLUP and CUBE take care of very common scenarios, but each is restricted in which set of summaries you can get, and each includes the basic aggregated data in the result. What if you want a different set of summaries? What if you want just the summaries without the basic aggregated data?

In our example, suppose you want to see the summary for each month across all years and locations, the summary for each year across all months and locations, and the summary for each location across all months and years? You could get those results by doing a separate query for each and then combining them with UNION ALL, as in Listing 6 (SummariesUnion.SQL in this month's downloads); Figure 6. shows partial results.

Listing 6. You can retrieve just the summaries using UNION ALL.

```
SELECT Person.CountryRegion.Name,
       Person.Address.City,
       null AS nYear,
       null AS nMonth,
       SUM(SubTotal) AS TotalSales,
       AVG(SubTotal) AS AvgSale,
       COUNT(SubTotal) AS NumSales
  FROM Sales.Customer
    JOIN Person.Person
      ON Customer.PersonID =
         Person.BusinessEntityID
    JOIN Person.BusinessEntityAddress
      ON Person.BusinessEntityID =
        BusinessEntityAddress.BusinessEntityID
    JOIN Person.Address
      ON BusinessEntityAddress.AddressID =
         Address.AddressID
    JOIN Person.StateProvince
      ON Address.StateProvinceID =
         StateProvince.StateProvinceID
    JOIN Person.CountryRegion
      ON StateProvince.CountryRegionCode =
         CountryRegion.CountryRegionCode
    JOIN Sales.SalesOrderHeader
      ON Customer.CustomerID =
         SalesOrderHeader.CustomerID
    JOIN Sales.SalesOrderDetail
      ON SalesOrderHeader.SalesOrderID =
         SalesOrderDetail.SalesOrderID
    GROUP BY Person.CountryRegion.Name, City
UNION ALL
SELECT NULL AS Name,
       NULL City,
       NULL AS nYear,
       MONTH(OrderDate) AS nMonth,
       SUM(SubTotal) AS TotalSales,
       AVG(SubTotal) AS AvgSale,
       COUNT(SubTotal) AS NumSales
  FROM Sales.SalesOrderHeader
```

```
    JOIN Sales.SalesOrderDetail
      ON SalesOrderHeader.SalesOrderID =
         SalesOrderDetail.SalesOrderID
  GROUP BY MONTH(OrderDate)
UNION ALL
SELECT NULL AS Name,
       NULL AS City,
       YEAR(OrderDate) AS nYear,
       NULL AS nMonth,
       SUM(SubTotal) AS TotalSales,
       AVG(SubTotal) AS AvgSale,
       COUNT(SubTotal) AS NumSales
  FROM Sales.SalesOrderHeader
    JOIN Sales.SalesOrderDetail
      ON SalesOrderHeader.SalesOrderID =
         SalesOrderDetail.SalesOrderID
    GROUP BY YEAR(OrderDate)
    ORDER BY Name, City, nYear, nMonth
```

Name	City	nYear	nMonth	TotalSales	AvgSale	NumSales
NULL	NULL	NULL	7	228486423.7421	27584.9841	8283
NULL	NULL	NULL	8	346541500.3052	29440.2769	11771
NULL	NULL	NULL	9	314385828.5747	29177.339	10775
NULL	NULL	NULL	10	166185042.8543	20133.8796	8254
NULL	NULL	NULL	11	271459474.9601	24909.1094	10898
NULL	NULL	NULL	12	235620076.8434	20664.8023	11402
NULL	NULL	2005	NULL	141944504.3041	27556.6888	5151
NULL	NULL	2006	NULL	779150362.1894	40259.9267	19353
NULL	NULL	2007	NULL	1169638118.0044	22827.9976	51237
NULL	NULL	2008	NULL	505737472.1795	11096.5743	45576
Australia	Bendigo	NULL	NULL	555431.9893	1402.606	396
Australia	Brisbane	NULL	NULL	546014.4579	1403.6361	389
Australia	Caloundra	NULL	NULL	527130.8302	1301.5576	405
Australia	Cloverdale	NULL	NULL	384307.4948	1311.6296	293
Australia	Coffs Harbour	NULL	NULL	394188.9185	1018.576	387

Figure 6. Sometimes, you want only the summaries, not the original aggregations.

That's a lot of code. SQL Server offers an alternative way to do this, using a feature called GROUPING SETS. They let you fine tune which summaries you get. With GROUPING SETS, you explicitly tell the query which combinations to summarize. The GROUPING SETS equivalent of the UNIONed query in Listing 6 is shown in Listing 7 (included in this month's downloads as SummariesGroupingSets.SQL).

Listing 7. GROUPING SETS let you ask for the specific set of summaries you want.

```
SELECT Person.CountryRegion.Name,
       Person.Address.City,
       YEAR(OrderDate) AS nYear,
       MONTH(OrderDate) AS nMonth,
       SUM(SubTotal) AS TotalSales,
       AVG(SubTotal) AS AvgSale,
       COUNT(SubTotal) AS NumSales
  FROM Sales.Customer
    JOIN Person.Person
      ON Customer.PersonID =
         Person.BusinessEntityID
    JOIN Person.BusinessEntityAddress
      ON Person.BusinessEntityID =
        BusinessEntityAddress.BusinessEntityID
    JOIN Person.Address
      ON BusinessEntityAddress.AddressID =
         Address.AddressID
    JOIN Person.StateProvince
      ON Address.StateProvinceID =
         StateProvince.StateProvinceID
    JOIN Person.CountryRegion
      ON StateProvince.CountryRegionCode =
         CountryRegion.CountryRegionCode
    JOIN Sales.SalesOrderHeader
      ON Customer.CustomerID =
         SalesOrderHeader.CustomerID
    JOIN Sales.SalesOrderDetail
      ON SalesOrderHeader.SalesOrderID =
         SalesOrderDetail.SalesOrderID
  GROUP BY GROUPING SETS (
    (CountryRegion.Name, Address.City),
    (YEAR(OrderDate)), (MONTH(OrderDate)))
  ORDER BY Name, City,
           YEAR(OrderDate), MONTH(OrderDate)
```

The GROUP BY clause indicates three grouping sets here, each enclosed in parentheses: (CountryRegion.Name, Address.City) which says to show totals for each city and country combination, across all years and months; (YEAR(OrderDate)), which asks for totals for each year, across all locations and months; and (MONTH(OrderDate)), which requests totals for each month, across all locations and years. The parentheses are required in the first case, to show that city and country are to be treated as a set. While they're not required for the other two items, they do make clear that each is to be handled separately.

ROLLUP and CUBE are actually special cases of GROUPING SETS. You can use GROUPING SETS to get the same results, though it actually makes the code longer. Listing 8 shows the GROUP BY clause for the GROUPING SETS equivalent of the ROLLUP query in Listing 2. (The complete version of this query is included in this month's downloads as GroupingSetsRollupEquiv.sql.)

Listing 8. You can use GROUPING SETS instead of ROLLUP, but it calls for more code in the GROUP BY clause.

```
GROUP BY GROUPING SETS (
  (CountryRegion.Name, Address.City,
   YEAR(OrderDate), MONTH(OrderDate)),
  (CountryRegion.Name, Address.City,
   YEAR(OrderDate)),
  (CountryRegion.Name, Address.City),
  (CountryRegion.Name),
  ())
```

There are five grouping sets shown. The first set, which includes all four non-aggregated fields is the equivalent of simply doing a GROUP BY with that list. It does the aggregation, but no summaries.

Each grouping set after that contains one fewer field than the preceding one, until the last contains no field, indicating that the summary should be computed over the entire data set. Looking at this GROUP BY clause actually helps to clarify what ROLLUP does. It aggregates on all the fields listed, then one by one, removes fields from the right and aggregates again.

For the equivalent of CUBE, the GROUPING SETS list is even more unwieldy, but again it sheds light on what's going on when you use CUBE. Listing 9 shows the GROUP BY clause for a query (GroupingSetsCubeCombinedEquiv.sql in this month's downloads) that produces the same results as Listing 4.

Listing 9. Replacing CUBE with GROUPING SETS lets you see all the cases that CUBE handles.

```
GROUP BY GROUPING SETS(
  (CountryRegion.Name, Address.City,
   YEAR(OrderDate), MONTH(OrderDate)),
  (CountryRegion.Name, Address.City,
   YEAR(OrderDate)),
  (CountryRegion.Name, Address.City,
   MONTH(OrderDate)),
  (CountryRegion.Name, Address.City),
  (YEAR(OrderDate), MONTH(OrderDate)),
  (YEAR(OrderDate)),
  (MONTH(OrderDate)),
  ())
```

Note that unlike the CUBE query, you don't have to (in fact, can't) enclose the country/city pair in parentheses here. You just omit any grouping sets that include one without the other.

Of course, there's no reason to write out the long version when you can use ROLLUP or CUBE. But when you need something else, having GROUPING SETS available is a big help.

As Listing 7 demonstrates, grouping sets also let you get summaries without including the basic aggregated data. Just omit the grouping set that lists all the fields on which to aggregate. Be aware, though, that as with any other GROUP BY clause, every field in the field list that doesn't include an aggregate function must appear somewhere in the list of grouping sets.

Listing 10 shows the GROUP BY clause for a query that's equivalent to Listing 4, but without the first grouping set, so that only the summaries are included. Figure 7 shows partial results; if you compare to Figure 5, you can see that the rows where nothing is null have been eliminated. This query is included as GroupingSetsWithoutAggregates.sql in this month's downloads.

Listing 10. By omitting the grouping set that includes all non-aggregated fields, you can get just the summaries you want without the base aggregated data.

```
GROUP BY GROUPING SETS(
  (CountryRegion.Name, Address.City,
   YEAR(OrderDate)),
  (CountryRegion.Name, Address.City,
   MONTH(OrderDate)),
  (CountryRegion.Name, Address.City),
  (YEAR(OrderDate), MONTH(OrderDate)),
  (YEAR(OrderDate)),
  (MONTH(OrderDate)),
  ())
```

Name	City	nYear	nMonth	TotalSales	AvgSale	NumSales
NULL	NULL	2008	6	5534993.75	996.7573	5553
NULL	NULL	2008	7	137184.87	62.0465	2211
Australia	Bendigo	NULL	NULL	555431.9893	1402.606	396
Australia	Bendigo	NULL	1	45717.8121	1576.4762	29
Australia	Bendigo	NULL	2	35863.8521	1434.554	25
Australia	Bendigo	NULL	3	47283.4582	1432.832	33
Australia	Bendigo	NULL	4	63837.0056	1329.9376	48
Australia	Bendigo	NULL	5	61997.1378	1265.2477	49
Australia	Bendigo	NULL	6	63823.672	1556.6749	41
Australia	Bendigo	NULL	7	60588.16	1731.0902	35
Australia	Bendigo	NULL	8	28585.1242	1242.8314	23
Australia	Bendigo	NULL	9	23499.03	1382.2958	17
Australia	Bendigo	NULL	10	52875.3221	1429.0627	37
Australia	Bendigo	NULL	11	34616.54	1081.7668	32
Australia	Bendigo	NULL	12	36744.8752	1360.9213	27
Australia	Bendigo	2005	NULL	41972.84	3497.7366	12
Australia	Bendigo	2006	NULL	68205.8315	2435.9225	28
Australia	Bendigo	2007	NULL	211159.4078	1311.5491	161

Figure 7. When you exclude the grouping set that contains all aggregated fields, the result contains only the summaries.

Make it pretty

As with the ROLLUP clause, for both CUBE and GROUPING SETS, you can make the results easier to understand by using ISNULL() to replace the nulls with meaningful descriptions. (Reminder: ISNULL() is SQL Server's equivalent to VFP's NVL().)

Listing 11 shows the query from Listing 4 with the descriptions added. Figure 8 shows partial results. The query is included in this month's downloads as SalesByCountryCityCubeCombinedWDesc.sql.

Listing 11. You can replace the nulls that indicate summary records with descriptions.

```
SELECT ISNULL(Person.CountryRegion.Name,
              'All countries') AS Name,
       ISNULL(Person.Address.City,
              'All cities') AS City,
       ISNULL(STR(YEAR(OrderDate)),
              'All years') AS cYear,
       ISNULL(STR(MONTH(OrderDate)),
              'All months') AS cMonth,
       SUM(SubTotal) AS TotalSales,
       AVG(SubTotal) AS AvgSale,
       COUNT(SubTotal) AS NumSales
  FROM Sales.Customer
    JOIN Person.Person
      ON Customer.PersonID =
         Person.BusinessEntityID
    JOIN Person.BusinessEntityAddress
      ON Person.BusinessEntityID =
        BusinessEntityAddress.BusinessEntityID
    JOIN Person.Address
      ON BusinessEntityAddress.AddressID =
         Address.AddressID
    JOIN Person.StateProvince
      ON Address.StateProvinceID =
         StateProvince.StateProvinceID
    JOIN Person.CountryRegion
      ON StateProvince.CountryRegionCode =
```

```
        CountryRegion.CountryRegionCode
    JOIN Sales.SalesOrderHeader
      ON Customer.CustomerID =
         SalesOrderHeader.CustomerID
    JOIN Sales.SalesOrderDetail
      ON SalesOrderHeader.SalesOrderID =
         SalesOrderDetail.SalesOrderID
GROUP BY CUBE((CountryRegion.Name,
               Address.City),
              YEAR(OrderDate),
              MONTH(OrderDate))
ORDER BY Name, City, cYear, cMonth
```

Name	City	cYear	cMonth	TotalSales	AvgSale	NumSales
All countries	All cities	All years	9	3649551.8024	862.5742	4231
All countries	All cities	All years	10	4014666.7773	884.6775	4538
All countries	All cities	All years	11	4190634.2945	923.0472	4540
All countries	All cities	All years	12	6233117.4491	1065.6723	5849
All countries	All cities	All years	All months	59273769.3...	980.3961	60459
Australia	Bendigo	2005	7	20909.78	3484.9633	6
Australia	Bendigo	2005	9	3578.27	3578.27	1
Australia	Bendigo	2005	10	6953.26	3476.63	2
Australia	Bendigo	2005	11	3578.27	3578.27	1
Australia	Bendigo	2005	12	6953.26	3476.63	2
Australia	Bendigo	2005	All months	41972.84	3497.7366	12
Australia	Bendigo	2006	1	10734.81	3578.27	3
Australia	Bendigo	2006	2	10734.81	3578.27	3
Australia	Bendigo	2006	3	3578.27	3578.27	1
Australia	Bendigo	2006	4	3578.27	3578.27	1

Figure 8. You can use ISNULL() to substitute descriptions for nulls, and make the results easier to comprehend.

What about VFP?

My last article showed how you do the equivalent of ROLLUP in VFP. The second approach shown there, using a separate query for each summary you want, and then combining the results with UNION, works for CUBE and GROUPING SETS, as well. Of course, the resuling code is fairly opaque. That's why having these shortcuts in SQL Server is so nice.

Author Profile

Tamar E. Granor, Ph.D. is the owner of Tomorrow's Solutions, LLC. She has developed and enhanced numerous Visual FoxPro applications for businesses and other organizations. Tamar is author or co-author of a dozen books including the award winning Hacker's Guide to Visual FoxPro, Microsoft Office Automation with Visual FoxPro *and* Taming Visual FoxPro's SQL. *Her latest collaboration is* VFPX: Open Source Treasure for the VFP Developer, *available at www.foxrockx.com. Her other books are available from Hentzenwerke Publishing (www.hentzenwerke.com). Tamar was a Microsoft Support Most Valuable Professional from the program's inception in 1993 until 2011. She is one of the organizers of the annual Southwest Fox conference. In 2007, Tamar received the Visual FoxPro Community Lifetime Achievement Award. You can reach her at tamar@thegranors.com or through www.tomorrowssolutionsllc.com.*