SQL Extensions for OLAP

OLAP requires almost invariably data aggregations, and SQL does support such aggregations through its Group-by instruction. However, during OLAP, each data analysis session usually requires repeated aggregations of similar type over the same data. Formulating many similar but distinct queries is not only tedious for the analyst but also quite expensive in execution time, as all these similar queries pass over the same data over and over again.

It thus seems worthwhile to try to find a way of requesting several levels of aggregation in a single query, thereby offering the implementation the opportunity to compute all of those aggregations more efficiently (i.e., in a single pass). Such considerations are the motivation behind the currently available options of the Group-by clause, namely,

- *Grouping Sets*
- *Rollup*
- *Cube*

These options are supported in several commercial products and they are also included in the current versions of SQL. We illustrate their use below, assuming the table SP(S#, P#, QTY), where attributes stand for “Supplier Number”, “Product Number”, and “Quantity”, respectively.

The Grouping sets option allows the user to specify exactly which particular groupings are to be performed:

Select S#, P#, Sum (QTY) As TOTQTY
From SP
Group By Grouping Sets ((S#), (P#))

This instruction is equivalent to two Group-by instructions, one in which the grouping is by S# and one in which the grouping is by P#. Changing the order in which the groupings are written does not affect the result. The remaining two options, **Rollup** and **Cube**, are actually shorthand for certain Grouping Sets combinations.

Consider first the following example of Rollup:

Select S#, P#, Sum (QTY) As TOTQTY
From SP
Group By Rollup (S#, P#)

This instruction is equivalent to (or shorthand for) the following instruction:

Select S#, P#, Sum (QTY) As TOTQTY
From SP
Group By Grouping Sets ((S#, P#), (S#), ( ))

Note that, in the case of Rollup, changing the order in which the attributes are written affects the result.

Finally, consider the following example of Cube:

Select S#, P#, Sum (QTY) As TOTQTY
From SP
Group By Cube (S#, P#)

This instruction is equivalent to the following one:

Select S#, P#, Sum (QTY) As TOTQTY
From SP
Group By Grouping Sets ((S#, P#), (S#), (P#), ( ))

In other words, the Cube option forms all possible groupings of the attributes listed in the Group-by clause. Therefore, in the case of Cube, changing the order in which the attributes are written does not affect the result.

We note that, although the result of each of the above Group-by options usually consists of two or more distinct answer-tables, SQL bundles them (unfortunately) into a single table, using nulls.

We also note that OLAP products often display query results not as SQL tables but as *cross tabulations* of SQL tables. The cross tabulation of a SQL table is a multi-dimensional table indexed by the values of the key attributes in the SQL table and in which the entries are the values of the dependent attributes. Cross tabulation - together with various visualization techniques - is especially useful for producing reports out of query results, and several report generating tools are available today in the market.

Finally, we note that the decision making process in an enterprise usually requires a number of reports produced periodically from the answers to a specific, fixed set of queries (such as monthly average sales per store, or per region, etc.). Such queries are usually called "continuous queries" or "temporal queries". It is important to note here that a temporal query does not change over time; what changes over time is the answer to the query and, as a result, the report produced from the answer changes as well.