THE TEACHING OF RELATIONAL ON-LINE ANALYTICAL PROCESSING (ROLAP) IN ADVANCED DATABASE COURSES

Bernadette Marie Byrne
School of Computer Science
University of Hertfordshire
Hatfield
UK
b.m.byrne@herts.ac.uk

Iftikhar Ansari
School of Computer Science
University of Hertfordshire
Hatfield
UK
Ifitkharansari2003@hotmail.com

ABSTRACT

One of the problems with teaching ROLAP and Data Warehousing in database systems courses is the difficulty of providing practical exposure of the topics to the students. The approach taken in this work is intended to help students apply the knowledge learnt through lectures into a practical environment. This will provide the basis for analysis and evaluation at the level of critical analysis that would be expected on an advanced level course and at the highest level of learning taxonomies. In our work we have a large dataset reversed engineered from some Oracle based sample data. This dataset has a fact table with 918843 records and dimension tables. We have also produced a smaller sample dataset for simplification and understanding of complex queries involving Slicing, Dicing, Pivoting, Rollup and Cube operations. In our examples ROLAP operations and their alternative SQL approaches are performed to provide a comparison of the multi-dimensional results.

Keywords

Rollup, Cube, SQL, ROLAP, Data Warehousing.

1. INTRODUCTION

Data Warehousing and Relational On-Line Analytical Processing are now often taught in university database courses and included in most textbooks on Database Systems, in references [1],[2],[3] for example. The Data Warehousing Star Schema design is usually taught and this is followed with the differences between OLTP systems and OLAP systems. The students can then have practical experience using the SQL extensions for OLAP by using queries with Rollup, Cube etc. One of the problems with providing hands-on experience for the students is providing a large enough data set in which to run queries, as obviously in a real Data Warehousing environment we are dealing very large amounts of historical data. Also, if you want the students to consider performance aspects and query optimisation aspects of the use of the ROLAP extensions you need to have large datasets.

In our work we have reverse engineered a large Oracle sample dataset and adapted it for use in the teaching of ROLAP. We have also produced our own smaller sample dataset for simplification and understanding of complex queries involving Slicing, Dicing, Pivoting, Rollup and Cube operations. In our examples in this paper we use the smaller dataset as it is easier to demonstrate results. ROLAP operations and their alternative SQL approaches are performed to provide multi-dimensional results and thus a means of comparison between an analytical query run with standard SQL and then again with OLAP extensions to SQL. Our sample dataset is based on a Star Schema design.

2. PERFORMING ROLAP OPERATIONS

We have produced a small sales data table with 36 rows for illustration purposes. The table contains four columns, Scale (size), Item, Colour and Quantity. The table overleaf is the result of query one:-

```
Select * from the sales_data table;
```

The table contains 36 rows with four attributes, or in dimensional modelling terminology, it has 3 dimension attributes; Scale, Item and Colour and a numeric measure attribute, Quantity. The measure attribute which is quantity, shows the quantities produced against each item of different sizes and colours.

SCALE	ITEM	COLOUR	QUANTITY
SMALL	PANTS	BLACK	1
SMALL	PANTS	BLUE	0
SMALL	PANTS	WHITE	2
SMALL	SHIRTS	BLACK	3
SMALL	SHIRTS	BLUE	2
SMALL	SHIRTS	WHITE	8
SMALL	DRESS	BLACK	5
SMALL	DRESS	BLUE	3
SMALL	DRESS	WHITE	1
SMALL	LEATHER-COATS	BLACK	2
SMALL	LEATHER-COATS	BLUE	10
SMALL	LEATHER-COATS	WHITE	3
MEDIUM	PANTS	BLACK	12
MEDIUM	PANTS	BLUE	1
MEDIUM	PANTS	WHITE	2
MEDIUM	SHIRTS	BLACK	6
MEDIUM	SHIRTS	BLUE	3
MEDIUM	SHIRTS	WHITE	10
MEDIUM	DRESS	BLACK	7
MEDIUM	DRESS	BLUE	4
MEDIUM	DRESS	WHITE	2
MEDIUM	LEATHER-COATS	BLACK	4
MEDIUM	LEATHER-COATS	BLUE	12
MEDIUM	LEATHER-COATS	WHITE	5
LARGE	PANTS	BLACK	7
LARGE	PANTS	BLUE	1
LARGE	PANTS	WHITE	1
LARGE	SHIRTS	BLACK	5
LARGE	SHIRTS	BLUE	2
LARGE	SHIRTS	WHITE	10
LARGE	DRESS	BLACK	8
LARGE	DRESS	BLUE	3
LARGE	DRESS	WHITE	2
LARGE	LEATHER-COATS	BLACK	2
LARGE	LEATHER-COATS	BLUE	13
LARGE	LEATHER-COATS	WHITE	2

Table 1 36 rows selected.

We begin with the Group By clause provided in Standard SQL.

SQL Statement 2

```
SELECT scale, item, colour, sum(quantity)
  FROM sales_data
  GROUP BY scale, item, colour
```

This query will produce information similar to Table 1. (Obviously space does not allow us to produce all of the results for the following queries.) We can also use the following Group By statements:

SQL Statement 3

```
SELECT scale, item, to_char(null),SUM(quantity)
FROM sales_data
GROUP BY (scale, item)
```

Query 3 would show the grouped data, based on the scale and item dimension attributes and answers the query of how many items there are in small, medium and large scale. It shows the quantity of items present in different sizes but does not show sub-totals, such as how many small items are there irrespective of their colour or scale. Nor does it show how many shirts are there irrespective of sizes and colours. The next query shows how many total items are there in small, medium and large sizes of any colour and of any item.

SQL Statement 4

```
SELECT scale, to_char(null), to_char(null), sum(quantity)
FROM sales_data
GROUP BY scale
```

SCALE	T	T	SUM(QUANTITY)
LARGE			56
MEDIUM			68
SMALL			40

Table-4: Grouping on sample data.

SQL Statement 5

```
SELECT SUM(quantity)FROM sales_data
GROUP BY ()
```

This query shows summary information and shows the total number of items of any colour and any size produced by the company.

3. THE PROBLEM WITH GROUP BY

The problem with the SQL standard GROUP BY clause is that neither of the above statements provides multidimensional views to decision makers that display sub-totals and totals across various dimensions, nor does it traverse different levels of hierarchies, nor does it perform cross-tabulation. In order to produce multidimensional results to provide sub-totals and totals at various different dimensions, at various different level of hierarchies, these statements can be are combined together through the use of the UNION clause of standard SQL to generate the desired results.

A second problem relates to the roll-up operation using totals and sub-totals for drill-down reports. A Report commonly aggregates data at a coarse level, and then at successively finer levels. Moving across the dimensional hierarchies from finer level to coarser level hierarchies is called rolling-up the data whereas drilling-down refers to moving from a higher level hierarchy to a lower level. A 3-dimensional roll-up, aggregating over N dimensions, requires N unions of statements showing different results from different dimensions when using standard SQL without OLAP extensions.

4. UNION OF MULTIPLE SELECTS

The results of Sql statements 2, 3, 4, 5 can be combined together to provide the same result as a Rollup. (N+1) standard sql statements are required to generate the rollup results where n represents the number of dimensions. Such that if a 3 dimensional result with sub-totals and a grand total is needed, 3+1 = 4 sql statements and UNIONs are needed to generate the rollup or drill-down result sets. In the following case, 3D rollup is carried out. Each statement's returned result is UNION or summed-up towards the single output resultant report. (i.e. all these statements are executed individually and their results are combined using UNION returning a total of 36+12+3+1 = 52 rows). As this can be seen from the result of statement 6 it returns the combined result of statements 2, 3, 4 and 5 ran separately and individually. Writing such multidimensional queries is a tedious task and increases the complexity with the increase in the number of dimensions. It does

provide a better result to decision makers in decision support systems as it does provide a multidimensional view however it still lacks information such as:

What is the total number of items of any colour and any size i.e. total number of shirts, pants etc?

How many black coloured items are there regardless of item and their sizes?

Such information of sub-totals at every level of granularity at different dimensions is lacking in the ROLLUP operation of OLAP. The SQL statement 6 below will provide the same result as the ROLLUP function which is shown in SQL statement 7.

SQL Statement 6

```
SELECT scale, item, colour, SUM(quantity) FROM sales_data
GROUP BY scale, item, colour
UNION
SELECT scale, item, to_char(null),SUM(quantity) FROM sales_data
GROUP BY (scale, item)
UNION
SELECT scale, to_char(null),to_char(null),SUM(quantity)
FROM sales_data
GROUP BY (scale)
UNION
SELECT to_char(null),to_char(null),to_char(null),SUM(quantity)
FROM sales_data
GROUP BY ()
```

SCALE	ITEM	COLOUR	SUM(QUANTITY)
LARGE	DRESS	BLACK	8
LARGE	DRESS	BLUE	3
LARGE	DRESS	WHITE	2
LARGE	DRESS		13
LARGE	LEATHER-COATS	BLACK	2
LARGE	LEATHER-COATS	BLUE	13
LARGE	LEATHER-COATS	WHITE	2
LARGE	LEATHER-COATS		17
LARGE	PANTS	BLACK	7
LARGE	PANTS	BLUE	1
LARGE	PANTS	WHITE	1
LARGE	PANTS		9
LARGE	SHIRTS	BLACK	5
LARGE	SHIRTS	BLUE	2
LARGE	SHIRTS	WHITE	10
LARGE	SHIRTS		17
LARGE			56
MEDIUM	DRESS	BLACK	7
MEDIUM	DRESS	BLUE	4
MEDIUM	DRESS	WHITE	2
MEDIUM	DRESS		13
MEDIUM	LEATHER-COATS	BLACK	4
MEDIUM	LEATHER-COATS	BLUE	12
MEDIUM	LEATHER-COATS	WHITE	5

MEDIUM	LEATHER-COATS		21
MEDIUM	PANTS	BLACK	12
MEDIUM	PANTS	BLUE	1
MEDIUM	PANTS	WHITE	2
MEDIUM	PANTS		15
MEDIUM	SHIRTS	BLACK	6
MEDIUM	SHIRTS	BLUE	3
MEDIUM	SHIRTS	WHITE	10
MEDIUM	SHIRTS		19
MEDIUM			68
SMALL	DRESS	BLACK	5
SMALL	DRESS	BLUE	3
SCALE	**ITEM**	**COLOUR**	**SUM(QUANTITY)**
SMALL	DRESS	WHITE	1
SMALL	DRESS		9
SMALL	LEATHER-COATS	BLACK	2
SMALL	LEATHER-COATS	BLUE	10
SMALL	LEATHER-COATS	WHITE	3
SMALL	LEATHER-COATS		15
SMALL	PANTS	BLACK	1
SMALL	PANTS	BLUE	0
SMALL	PANTS	WHITE	2
SMALL	PANTS		3
SMALL	SHIRTS	BLACK	3
SMALL	SHIRTS	BLUE	2
SCALE	**ITEM**	**COLOUR**	**SUM(QUANTITY)**
SMALL	SHIRTS	WHITE	8
SMALL	SHIRTS		13
SMALL			40
			164

Table-6: Combination of multiple groupings for Rollup – 52 rows selected

SQL Statement 7:

```
SELECT scale, item, colour, sum(quantity)
   FROM sales_data
   GROUP BY rollup(scale, item, colour)
   ORDER BY 1
```

Statement 7 produces the same results as shown in table-6. However Rollup still does not provide the complete set of information that could be helpful for decision makers as discussed earlier. The CUBE function is used to provide more multidimensional results.

5. THE CUBE OPERATION

The following SQL statement will produce the same results as a CUBE operation.

SQL Statement 8:

```
SELECT scale, item, colour, SUM(quantity) FROM sales_data
GROUP BY (scale, item, colour)
UNION
SELECT scale, item, to_char(null), SUM(quantity)
FROM sales_data
GROUP BY (scale, item)
UNION
SELECT scale, to_char(null), colour, SUM(quantity)
FROM sales_data
GROUP BY (scale, colour)
UNION
SELECT to_char(null), item, colour, SUM(quantity)
FROM sales_data
GROUP BY (item, colour)
UNION
SELECT scale, to_char(null), to_char(null), SUM(quantity)
FROM sales_data
GROUP BY (scale)
UNION
SELECT to_char(null), item, to_char(null), SUM(quantity)
FROM sales_data
GROUP BY (item)
UNION
SELECT to_char(null), to_char(null), colour, SUM(quantity)
FROM sales_data
GROUP BY (colour)
UNION
SELECT to_char(null), to_char(null), to_char(null), SUM(quantity)
FROM sales_data
GROUP BY ()
```

Statement 8 will display information that the rollup operation in SQL statements 6 and 7 did in addition to answering questions such as How many coloured items (White, Black, Blue) are there regardless of Item type and their Scales? Plus the sub-total of quantities of each item.

Statement 8 can be replaced with the use of a single CUBE operator as shown below in statement 9 to produce the same results as queries 2-8.

```
SELECT scale, item, colour, sum(quantity)
   FROM sales_data
   GROUP BY cube(scale, item, colour)
   ORDER BY 1
```

Space does not permit us to produce the full set of results in which there are now 80 rows. Below are the final 14 rows of this query. The null values which occur in the table are very obvious in this case.

	LEATHER-COATS	WHITE	10
	LEATHER-COATS		53
	PANTS	BLACK	20
	PANTS	BLUE	2
	PANTS	WHITE	5
	PANTS		27
	SHIRTS	BLACK	14
	SHIRTS	BLUE	7
	SHIRTS	WHITE	28
	SHIRTS		49
		BLACK	62
		BLUE	54
		WHITE	48
			164

Table-7 (in part): Combining multiple groupings for a CUBE operation

We can also provide examples of ROLAP Sliding (Horizontal and Vertical) and vertical slicing of a CUBE result. Slicing refers to using a filter (WHERE or HAVING Clause) to make a slice of a CUBE on a selected dimension and Dicing is to view the slice in multidimensional manner.

6. PERFORMANCE

Below are examples of the types of activity which can be carried out and examined as regards query execution and performance. The first diagram illustrates the query strategy for a union of multiple selects. The second diagram shows the result of a cube operation.

Figure 1 **–** Union of multiple statements for CUBE

Figure 2 – Execution Steps for single CUBE operation

7. CONCLUSION

So far we have used the smaller dataset with students and queries in standard SQL and SQL with OLAP extensions. The majority of the students were surprised at all the null values in the results and commented on this or asked for clarification. They could not understand why the null values were there and also had difficulty interpreting the results. In other words they were not expecting a relational database to produce something which looked like a spreadsheet report but without the helpful headings. For such tabular results, C.J. Date [4] says that "this result might perhaps be thought of as a table (an SQL Style table, at any rate), but it is hardly a relation".

At a higher level of critical analysis by using results of the practical exercises we can encourage the students to reflect on C J Date's criticisms of, firstly, the Star Schema Design and secondly the ROLAP operators.[4] In this way we are also raising the learning to the higher levels of learning taxonomies for the more advanced groups. We have also found that this can also lead on to some interesting Undergraduate and Post-Graduate topics for project work.

In future sessions we intend to use the larger dataset to explore areas of physical database design, such as query processing, performance of ROLAP operators, parallelising and fragmentation. For example, evaluating the performance of ROLAP by identifying how to reduce full tables scans on the same table/relations and viewing the results multi-dimensionally from different angles and different perspectives.

8. REFERENCES

[1] R Ramakrishnan, J Gehrke, *Database Management Systems* Third Edition, International Edition Chapter 25. McGraw Hill 2003

[2] R Ramakrishnan, J Gehrke, *Fundamentals of Database Systems* Fourth Edition, Chapter 28 Pearson, 2003.

[3] "T Connolly, C Begg, Database Systems – *A Practical Approach to Design, Implementation, and Management* Fifth Edition, 2010.

[4] C. J. Date *An Introduction to Database Systems* Addison Wesley, Eighth Edition, Chapter 22 International Edition 2004