VLDM Lecture 2 - Exercises

Calin Rares Turliuc

Tuesday 20th October, 2015

1 Cube

Consider a schema similar to that on slides 47-48: `car_sales(model, year, color, sales)`. To experiment with the data, go to `http://sqlfiddle.com/`, select MS SQL Server 2014 from the drop down menu, then click 'Text to DDL'. Use the table name 'car_sales'. In the second text box, paste the data copied from this link. Click 'Append to DDL', then 'Build Schema'. We are ready to query the data.

We want to know how many cars of models Chevy or Ford and some color were sold in each year between 1990 and 1992, as well as sub-totals aggregated on all possible one or two columns. Finally, we are also interested in the grand total of sales. We use group by `cube`:

```
SELECT model, year, color, sum(sales) AS sales
FROM car_sales
WHERE model in ('ford','chevy') AND year BETWEEN 1990 and 1992
GROUP BY CUBE(model, year, color)
ORDER BY model, year, color;
```

The result contains answers to questions such as:

Q1 How many Chevy's were sold in 1990?

```
model   year    color   sales
chevy   1990    (null)  154
```

Q2 How many red Ford and Chevy's were sold between 1990 and 1992?

```
model   year    color   sales
(null)  (null)  red     233
```

etc.

To experiment yourself, enter the query in the right editor window and click 'Run SQL'.

2 Rollup, Grouping Sets

Consider a schema `pc_orders(date, orders)`, with data avalable at this link.

We want to know how many orders were processed overall and each year, and each year and month, and each year, month and day. We use group by `rollup`:

```
SELECT DATEPART(yy,date) AS year, DATEPART(mm,date) AS month,
DATEPART(dd,date) AS day, SUM(orders) AS orders
FROM pc_orders
GROUP BY ROLLUP (DATEPART(yy,date), DATEPART(mm,date), DATEPART(dd,date))
ORDER BY DATEPART(yy,date), DATEPART(mm,date), DATEPART(dd,date);
```

We can then answer questions such as: how many orders were placed in March, 2012.

```
year    month   day     orders
2012    3       (null)  36
```

Note. We cannot answers questions such as: how many orders were placed in March across all years and days, because `rollup` uses a hierarchy defined by the order of the columns in the grouping. The difference between `cube` and `rollup` is that:

- `CUBE(a,b,c)` is equivalent to

  `GROUPING SETS( (a,b,c), (a,b),(a,c),(b,c),(a),(b),(c), ()).`

- `ROLLUP(a,b,c)` is equivalent to

  `GROUPING SETS ((a,b,c),(a,b),(a), ()).`

Suppose we were interested in the number of orders on each day of each month across all years, and in the number of orders on each day of each year across all months, but not in anything else. We then explicitly specify the grouping sets:

```
SELECT DATEPART(yy,date) as year, DATEPART(mm,date) as month,
DATEPART(dd,date) as day, SUM(orders) as orders
FROM pc_orders
GROUP BY GROUPING SETS(
    ( DATEPART(mm,date), DATEPART(dd, date)),
    ( DATEPART(yy,date), DATEPART(dd, date)) )
ORDER BY DATEPART(yy,date), DATEPART(mm,date), DATEPART(dd,date);
```

We are now able to answer questions such as: how many orders were placed on Christmas Eve?

```
year    month   day     orders
(null)  12      24      111
```

. . . or how many orders were placed on the first day of the month in 2011?

```
year    month   day     orders
2011    (null)  1       43
```

3 Assignment (not graded)

3.1 Result size

Consider a relation with schema `r(c1, c2, ..., cn`. We wish to query a `GROUP BY CUBE` on columns `c1, c2, ..., cm` with no `WHERE` clause, but we are concerned the result will be too large to inspect visually. Therefore we wish to know what is the maximum number of rows in the result.

a) First, we need to know how many distinct values there are in each column `c1, ..., cm`. Write a query that returns 1 row with m columns containing the distinct values for each column.

b) Assume the values from a) are $k_1, \ldots, k_m$. What is the maximum number of rows of the result?

c) Answer the same question if the grouping was done using `GROUP BY ROLLUP`.

d) Create a table assuming $m = 2$ where all combinations of distinct values appear in the m columns. Verify the results of b) and c) using queries.

3.2 More queries

Consider a relation `orders(orderID, productID, count)` with data avalable at ths link and a relation `products(productID, price, category, instore)` with data available at ths link. The `instore` column in `products` specifies whether a product is available in store or not.

a) Write a query whose result answers (at least) the following two questions:

Q1 What is the income generated by product A?

Q2 What is the income generated by order 34?

b) Run the following query:

```
SELECT O.orderID, P.category, P.instore, SUM(P.price),
FROM orders O
    INNER JOIN products P
        ON O.productID = P.productID
GROUP BY CUBE(
    O.orderID,
    (P.category, P.instore))
ORDER BY O.orderID, P.category, P.instore;
```

Explain the result.

3.3 More examples

a) Think of at least one more example of relations and queries for each of `CUBE` and `ROLLUP`, where you would use an aggregation function different from `SUM`.

b) Give an example of a relation and query where you would use a `GROUP BY GROUPING SETS` where one of the grouping sets is defined by `ROLLUP`, e.g.

```
GROUP BY GROUPING SETS(
    c1,
    c2,
    ROLLUP(c3, c4))
```

Motivate your answer.

4 Answers

4.1 Result size

a)

```
SELECT COUNT(DISTINCT c1) , ..., COUNT(DISTINCT cm)
FROM r;
```

b)

$$\prod_{i=1}^{m}(k_i+1)$$

c)

$$1+\sum_{i=1}^{m}\left(\prod_{j=1}^{i}k_j\right)$$

4.2 More queries

a)

```
SELECT O.orderID, P.productID, SUM(P.price)
FROM orders O
    INNER JOIN products P
        ON O.productID = P.productID
GROUP BY CUBE(O.orderID, P.productID)
ORDER BY O.orderID, P.productID;
```

`ORDER BY` is optional.

b)

The grouped columns `category` and `instore` are treated as a single column by `CUBE`.

4.3 More examples

b) See example J at this link for inspiration.