# The CROME based computing of redundant Data cubes

Dr.C.Shoba bindu
Dept of CSE,
JNTUA , Anantapur.
Anantapur.
shobabindhu@gmail.com

K.Dhanasree
Dept of CSE,
DRK institute of science and technology.
Bowrampet, Hyd.
dsree_99@yahoo.co.in

***Abstract*—Very often computation of multiple group -bys are done by OLAP applications. Many of the OLAP applications use cube operator. The cube operator requires computing group-bys on all possible combinations of list of attributes. If multiple copies of the same data are stored at different data bases then the OLAP applications evaluate same data cubes may number of times. This increases the cost of evaluation and performance time. Here we present a CROME based method of computing the data cubes. Data cubes with same CROME are identified as duplicates and can be evaluated once.**

***Keyword: olap,cube,lattice,crome.***

# 1. Introduction

OLAP applications are at a great advent in solving modern business problems. Since we are required to retrieve large number of records from different data bases we are at an urge to summarize them on multidimensions. This multidimensional nature of data has led to OLAP applications. Recently introduced data cube operator is supporting such aggregates in OLAP data bases. The precomputation of all or part of data cube can greatly reduce the response time and enhance the performance of online analytical processing. The OLAP cubes are used to summarize data. Previous cube computation and representation approaches were classified into two main categories: (i) full and (ii) partial cube computation and representation. The first category approaches compute all the cells of all of the *cuboids* for a given data cube, while the second category approaches compute a subset of a given set of dimensions, or a smaller range of possible values for some of the dimensions.

The precomputation of the different summary views (group-bys) of a data cube is critical to improving the response time of data cube queries for On-Line Analytical Processing (OLAP). Many solutions have been proposed for generating the entire data cube. In contrast, this paper studies the problem of computing the *partial* data cube and full data cubes from many redundant data bases. That is, given a relation *R* of size *n* and dimension *d* as well as an arbitrary *subset S* of the set of all 2*d* possible view identifies, we wish to compute the subset of views that are non redundant identified in S. Despite the practical significance of partial cube generation for OLAP systems, little algorithmic work has been presented in the literature.

The group by operator in SQL is typically used to compute aggregates on a set of attributes. There are many approaches to calculate the cube aggregates. The sort based and the hash based approaches makes use of techniques like sorting the possible combinations of group bys. They calculated the lower group bys from the higher group bys. The hash based evaluation has made use of partitions. When the hash table is too large enough to fit into the available memory the table is partitioned and evaluated. Both the methods suffered from redundant evaluations when applied on multiple data bases.. In our work we present a method which calculates the CROME for each data cuboid. Later the CROMS are compared ,if the CROMS are equal the cuboids are evaluated only once. Our approach has reduced the cost of evaluating ones for each cuboid redundancy.

# 2. Data cube

For supporting decision queries effectively, a new operator, CUBE BY, was proposed [3]. It is a multidimensional extension of the relational operator GROUP BY. The CUBE BY operator computes GROUP BY corresponding to all possible combinations of grouping attributes in the CUBE BY clause. A cuboid of a cube is one group by. As the cube by attributes increases cube by becomes more expansive. Given a base relation *S* with *A* attributes, the number of tuples in a *r*-attribute cuboid(GROUP BY), $0 <= r <= A$, is the number of tuples in *S* that have distinct combination of attribute values on the *r*-attributes.. The huge size of a data cube makes data cube computation time-consuming. Recently several methods have been introduced to reduce the size of a data cube and hence its computation time and storage overhead, including condensed cube , Dwarf , Quotient cube and QC-trees . The basic idea of all these methods is trying to remove redundancies existing among cube tuples.

In SQL the collection of aggregate queries can be expressed using the cube operator as follows

```
Select A, B, C sum(s)
Cube by A, B, C;
```

This query will result in the computation of 8 Group-bys:

ABC, AB, BC,AC, A, B, C and ALL. ALL is the aggregate of all attributes, as shown in FIG 1 and executes them separately.

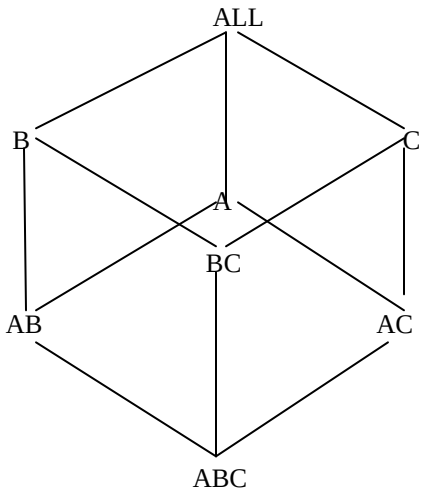



Fig 1 : Base cube

A lattice framework to represent the hierarchy of the group-bys was introduced in [2]. This is an elegant model for representing the dependencies in the calculations and also to model costs of the aggregate calculations. A scheduling algorithm can be applied to this framework substituting the appropriate costs of computation and communication. A lattice for the group-by calculations for a four dimensional cube is shown in Figure 2. Each node represents an aggregate and an arrow represents a possible aggregate calculation which is also used to represent the cost of the calculation. Calculation of the order in which the GROUP-BYs are created depends on the cost of deriving a lower order (one with a lower number of attributes) group-by from a higher order (also called the *parent*) group-by

When the OLAP application with the cube operator in implemented on multiple data bases ,these group bys are evaluated redundantly. This increases the cost of evaluation as well as the execution time.

## 3. CROME based method

A data cube with 4 attributes A, B, C, D have 16 possible group bys as shown in Fig 2

ABCD, ABC, ABD, BCD, ACD, AB, AC,AD, BC,BD,CD,A,B, C, D, and ALL.

Definition 1 ( ordered Maximal group ). The combinations consisting of maximum attributes at each level in an order of alphabetical combinations

We built crome codes for each combination as follows

| Level | group | ordered maximal groups |
|---|---|---|
| Level 0 | ABCD | ABCD |
| Level 1 | ABC, BCD ABD, CDA | ABC,ABD, BCD,CDA |
| Level 2 | AB,AD,AC,BD,BC,CD | AB,AC,AD,BC,BD,CD |
| Level 3 | A, C,B, D | A,B,C,D |

Definition 2 (crome codes) crome codes are binary equivalents of ordered numeric values of ordered maximal groups.

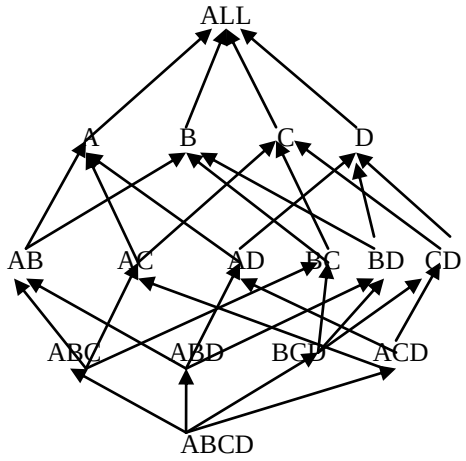



Fig 2 : **Lattice for cube operator**

| Ordered maximal Group | number equivalent | binary equivalent |
|---|---|---|
| ABCD | 0 | 0000 |
| ABC | 1 | 0001 |
| ABD | 2 | 0010 |
| BCD | 3 | 0011 |
| CDA | 4 | 0100 |
| AB | 5 | 0101 |
| AC | 6 | 0110 |
| AD | 7 | 0111 |
| BC | 8 | 1000 |
| ALL | 16 | 1111 |

For an SQL cube query

```
Select A, B, C, sum(s)
 from   sales
cube by A,B,C;
```

we take the base node as the ordered maximal group in the cube by condition. In the above SQL query base node is ABC with n=3 attributes.

Definition 3 (crome id) : we define crome id of a node as no two adjacent nodes with common n-1 base node attributes should be given the same id.

Definition 4(crome cube): The cube in which the possible group bys are given crome ids.

Definition 5 (minimal crome cube): Minimal crome cube is the crome cube which uses minimal crome ids.

For the above SQL query the minimal crome cube is

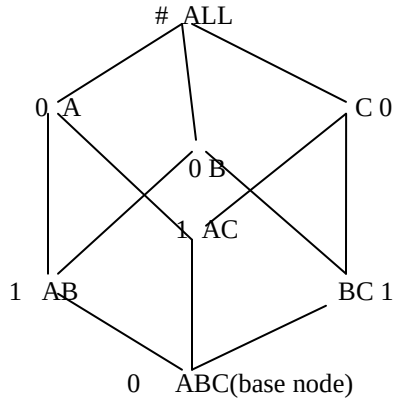



Fig 3: minimal crome cube

Definition 5 (D-sequence of minimal crome cube): D-sequence is sequence of crome ids at each levels of ordered maximal groups starting from the bas e node.

For the minimal crome cube of Fig 3 the D-sequence is
D- sequence (0, 1, 1, 1, 0, 0, 0, #).

## Crome values:

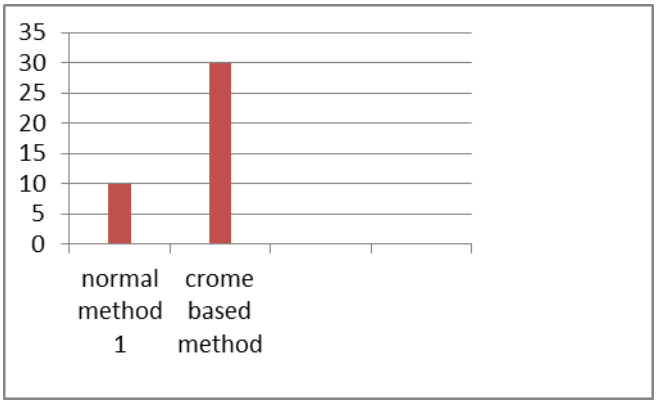
The crome values are taken as D-sequence together with the ordered maximal groups binary equivalents. for the cube shown in FIG 1 the crome values are

Crome value of cube (ABC,AB ,AC, BC,A,B,C,ALL)
= ( 0001, 0101, 0110, 1000,1100,1101,1110)
+ (0,111,0,0,0,#)

When the OLAP application executes the SQL query on the same sales table of the multiple data bases the aggregate cubes will be evaluated redundant times. Instead we first caluate the crome values for SQL cube by at multiple data bases. we compare the crome values as a global applicationthe resultant crome values are equal then we evaluate only one cube. The method reduces the cost of evaluation of same cube multiple times.the method has given a better performance , as evaluation of aggregates is timely when compared to the calucation of crome values.

# 3. Implementation and results

To test how well our method performs, we implemented using Informatica. We constructed two separate data bases with common sales table. The values for each attribute is independently chosen and the tuple size taken is 24 bytes. We have noticed that when compared to drawf or other methods crome based method reduced the cost of calculation of redundant cubes. Normal methods could reduce a 10% cost when compared to crome based which has reduced the cost of evaluating the redundant cubes upto 30% .




## Conclusion:

The heart of all OLAP or multidimensional data analysis applications is the ability to simultaneously aggregate across many sets of dimensions. Computing multidimensional aggregates is a performance bottleneck for these applications. The cube operator requires computing group bys on all possible combinations of attributes. We showed a crome based method of computing redundant data cubes in multiple data

bases. The approach has eliminated the redundant evaluation cost and showed a better performance.

## References:


[1] Goil S. and Choudhary A., "Parallel Data Cube Construction for High Performance On-Line Analytical Processing", Proc. 4th Intl. Conf. on High Performance Computing, Bangalore, India, 1997.

[2] Harinarayan V., Rajaraman A. and Ullman J.D."Implementing Data Cubes Efficiently", Proc.SIGMOD'96.

[3] Sarawagi S., Agrawal R., and Gupta A., "On Computing the Data Cube", Research Report 10026, IBM Almaden Research Center, San Jose, California, 1996.

[4 H. Gupta et al. Index selection for OLAP. In *Proceedings of the 13th ICDE*, 1997.

[5] Y. Zhou et al. An array-based algorithm for simultaneous multidimensional aggregates. In *Proceedings of the 1997 ACM SIGMOD Conference*.

[6] A. Shukla, P. Deshpande, and J. Naughton. Materialized view selection for multidimensional datasets. In *Proceedings of the 24th International VLDB Conference*, 1998.

[7] K. Ross and K. Zaman. Optimizing selections over data cubes. Technical Report CUCS-011-98, Department of Computer Science, Columbia University, 1997.