**Efficient Data Cube Computation**

**Data Warehouse Implementation**

The big data which is to be analyzed and handled to draw insights from it will be stored in data warehouses.  
These warehouses are run by OLAP servers which require processing of a query with seconds.  
So, a data warehouse should need highly efficient cube computation techniques, access methods, and query processing techniques.  
The core of multidimensional data analysis is the efficient computation of aggregations across many sets of dimensions.  
In SQL aggregations are referred to as group-by’s.  
  
Each group-by can be represented as a cuboid.  
  
Set of group-by’s forms a lattice of a cuboid defining a data cube.

**Efficient Data Cube Computation**

The compute cube Operator and the Curse of Dimensionality  
  
The compute cube operator computes aggregates over all subsets of the dimensions specified in the operation.

It requires excessive storage space, especially for a large number of dimensions.  
  
A data cube is a lattice of cuboids.

Suppose that we create a data cube for ProElectronics(Company) sales that contains the following: city, item, year, and sales\_in\_dollars.

Compute the sum of sales, grouping by city, and item.  
Compute the sum of sales, grouping by city.  
Compute the sum of sales, grouping by item.

What is the total number of cuboids, or group-by’s, that can be computed for this data cube?  
 **Three attributes:**

city, item, year (dimensions), sales\_in\_dollars (measure).

The total number of cuboids or group-by’s computed for this cube is 2^3=8.

**Group-by’s:** {(city,item,year), (city, item), (city, year), (item, year), (city), (item), (year),()}.  
() : group-by is empty i.e. the dimensions are not grouped.

The base cuboid contains all three dimensions.

Apex cuboid is empty.

On-line analytical processing may need to access different cuboids for different queries.

So we have to compute all or at least some of the cuboids in the data cube in advance.

Precomputation leads to fast response time and avoids some redundant computation.

A major challenge related to precomputation would be storage space if all the cuboids in the data cube are computed, especially when the cube has many dimensions.

The storage requirements are even more excessive when many of the dimensions have associated concept hierarchies, each with multiple levels.

This problem is referred to as the **Curse of Dimensionality**.

**Cube Operation**

Diagram

Description automatically generated

Cube definition and computation in DMQL

* define cube sales\_cube[ city, item, year] (sales\_in\_dollars)
* compute cube sales\_cube

Transform it into a SQL-like language (with a new operator cube by, introduced by Gray et al.’96)

* SELECT item, city, year, SUM (amount) FROM SALES CUBE BY item, city, year

**Data cube can be viewed as a lattice of cuboids**

* The bottom-most cuboid is the base cuboid.
* The top-most cuboid (apex) contains only one cell.
* How many cuboids in an n-dimensional cube with L levels? (T=SUM(Li+1))
* For example, the time dimension as specified above has 4 conceptual levels, or 5 if we include the virtual level all.
* If the cube has 10 dimensions and each dimension has 5 levels (including all), the total number of cuboids that can be generated is 510 =(5^10)=  9.8x106.

**Data Cube Materialization**

There are three choices for data cube materialization given a base cuboid.

* No Materialization
* Full Materialization
* Partial Materialization

**How to select which materialization to use**

* Identify the subsets of cuboids or subcubes to materialize.
* Exploit the materialized cuboids or subcubes during query processing.
* Efficiently update the materialized cuboids or subcubes during load and refresh.

**Selection of which cuboids to materialize**

* Based on the size, queries in the workload, accessing cost, their frequencies, etc.

CUBE MATERIALIZATION

* Full Cube
* Iceberg Cube
* Closed Cube
* Shell Cube

**FULL CUBE**

Full Cube:-all the cells of all of the cuboids for a given data cube. Thus, precomputation of the full cube can require huge and often excessive amounts of memory.

**Full materialization** refers to the computation of all of the cuboids in a data cube lattice.

**ICEBERG CUBE**

**Iceberg Cube:-**partially materialized cubes are known as iceberg cubes. The minimum threshold is called the minimum support threshold, or minimum support(min sup) •

**Partial materialization** refers to the selective computation of a subset of the cuboid cells in the lattice. Iceberg cubes and shell fragments are examples of partial materialization.

• An iceberg cube is a data cube that stores only those cube cells whose aggregate value (e.g., count) is above some minimum support threshold.

compute cube sales iceberg as

select month, city, customer group, count(\*)

from salesInfo

cube by month, city, customer group

having count(\*) >= min sup

**SHELL CUBE**

 Another strategy for partial materialization is to precompute only the cuboids involving a small number of dimensions, These cuboids form a cube shell for the corresponding data cube

 For shell fragments of a data cube, only some cuboids involving a small number of dimensions are computed. Queries on additional combinations of the dimensions

can be computed on the fly.

Text, letter

Description automatically generated

A picture containing diagram

Description automatically generated