Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals

Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao
Hamid Pirahesh, Frank Pellow

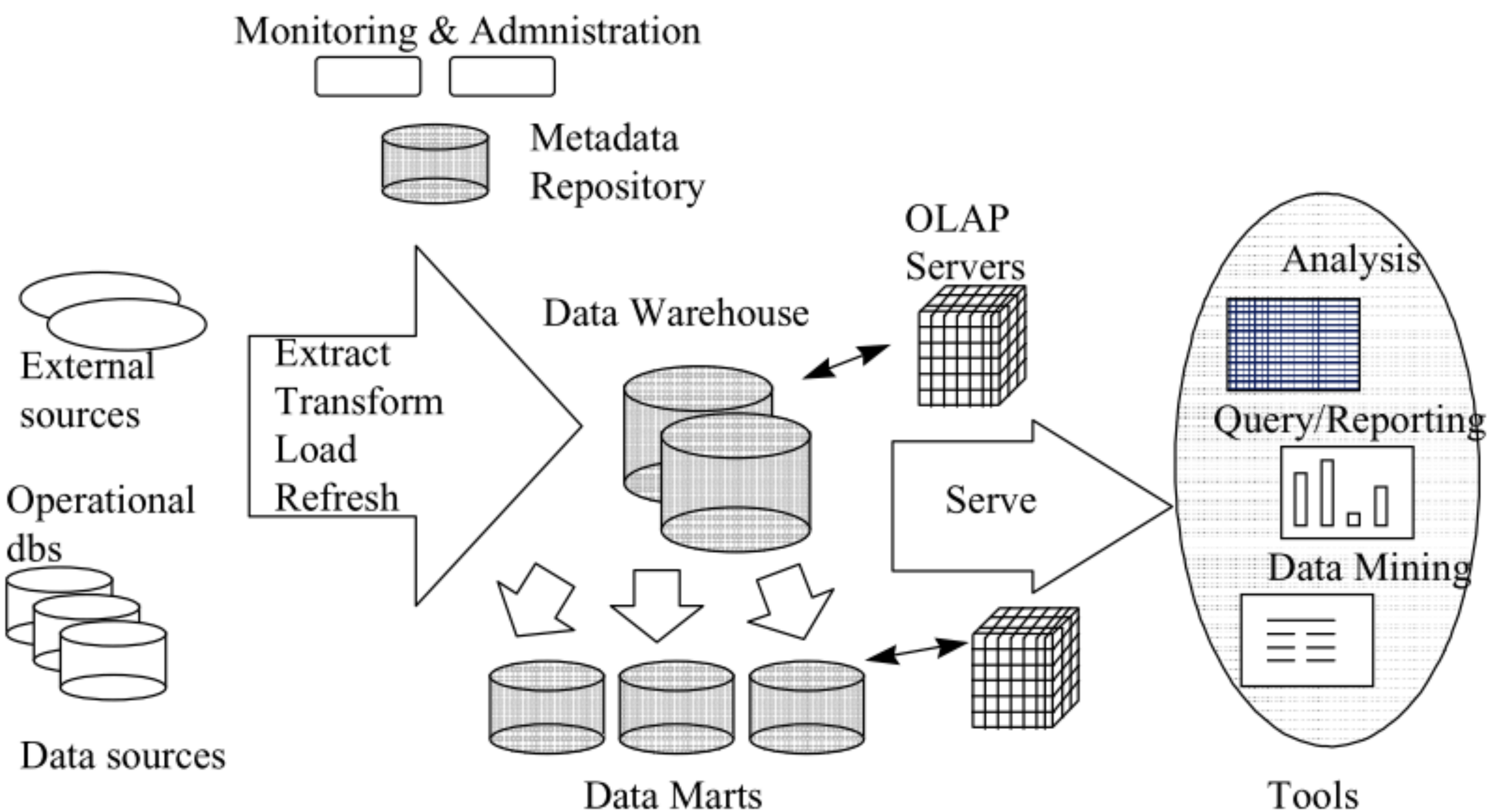
Presenter: Xiaoying Wang

How a **relational database** can support efficient **extraction** of **multidimensional** information

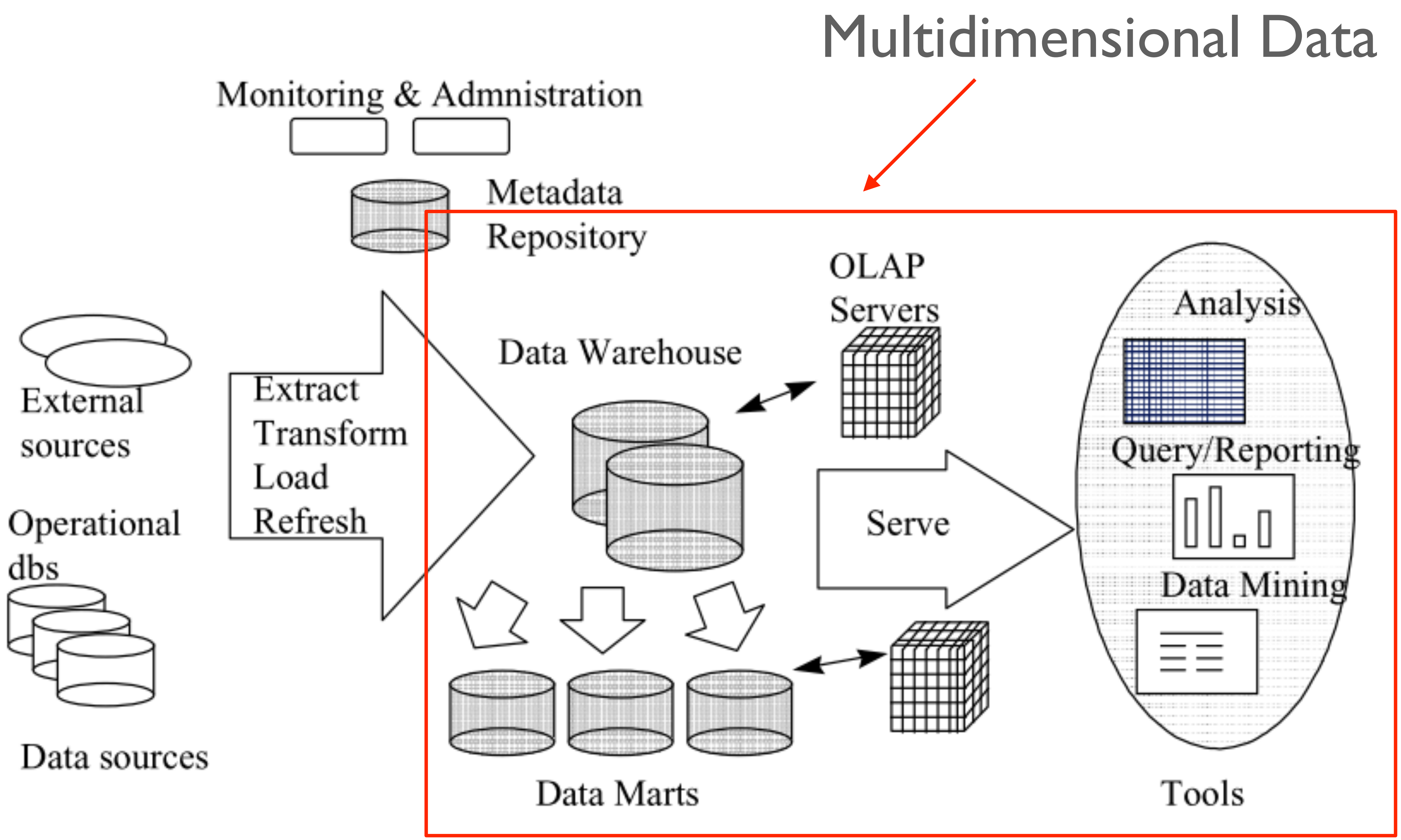
- What is a Data Cube in relational database?
- Why do we need the Cube Operator?
- How to implement the Cube Operator?

- **What is a Data Cube in relational database?**
- Why do we need the Cube Operator?
- How to implement the Cube Operator?

Data Warehouse & OLAP

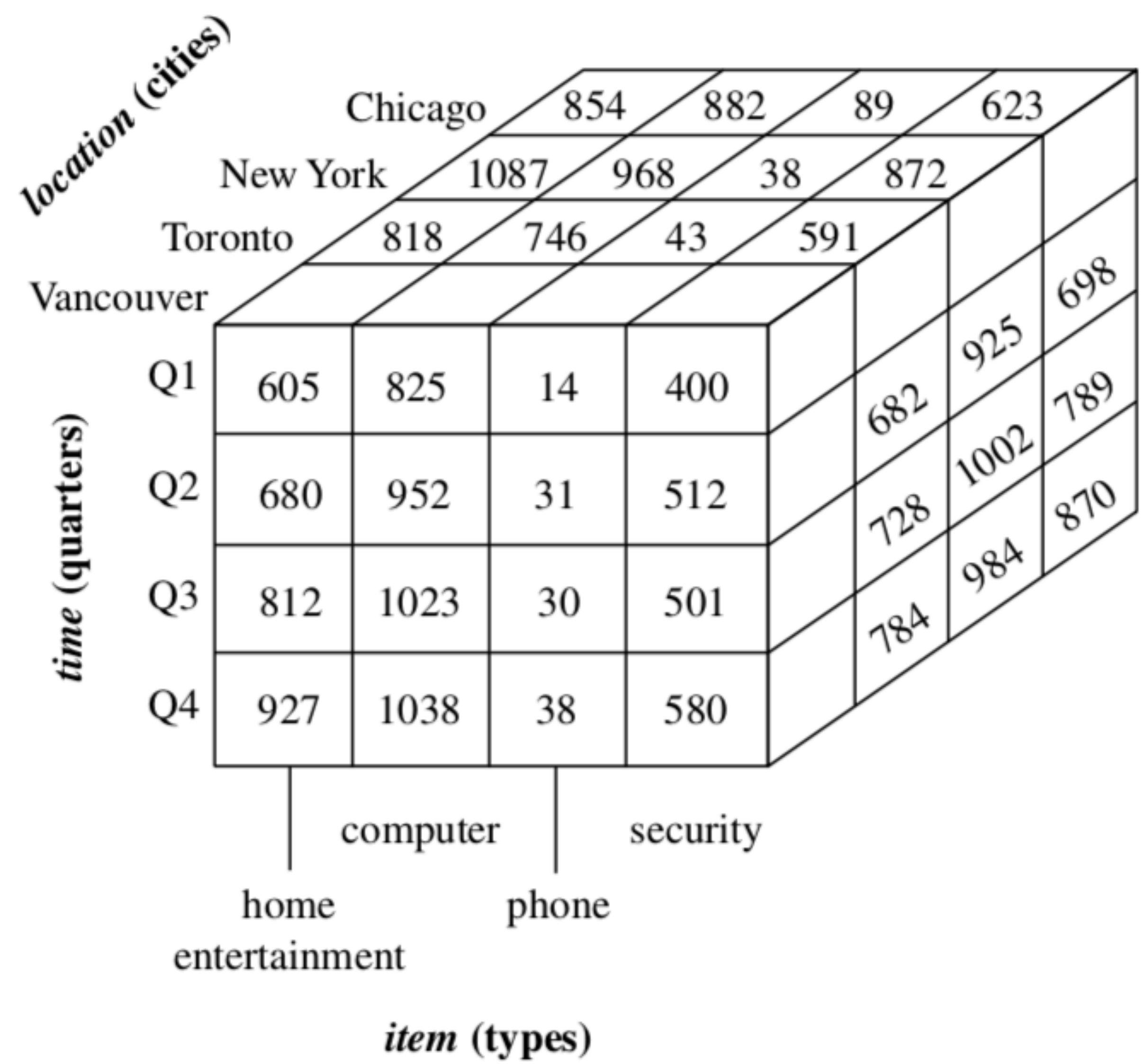


Data Warehouse & OLAP



Data Cube

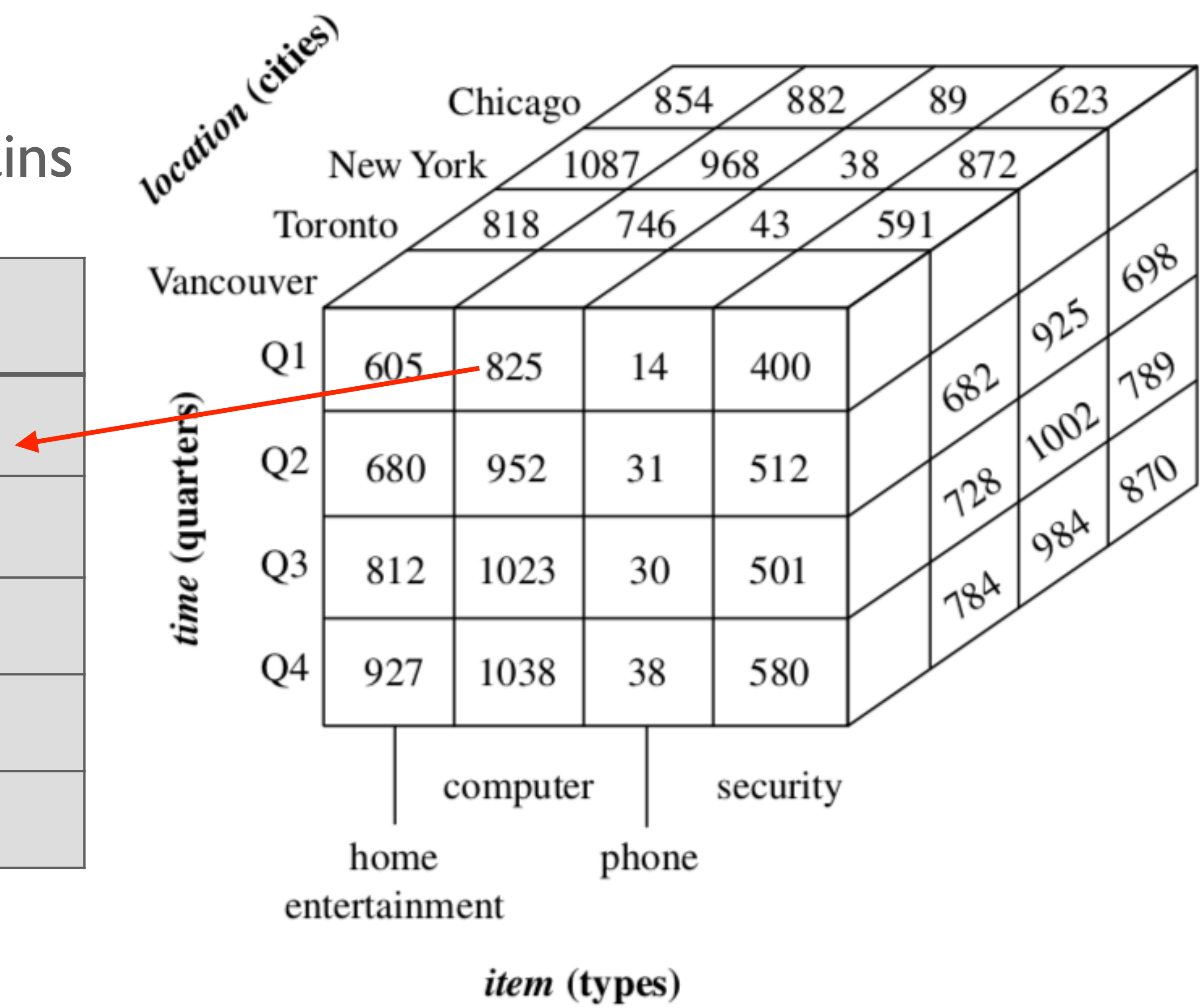
- A Multidimensional Data Model
  - Dimension
    - location
    - time
    - item
  - Measurement
    - sales



Data Cube

- In Relational Database:
  - A relation with n-attribute domains

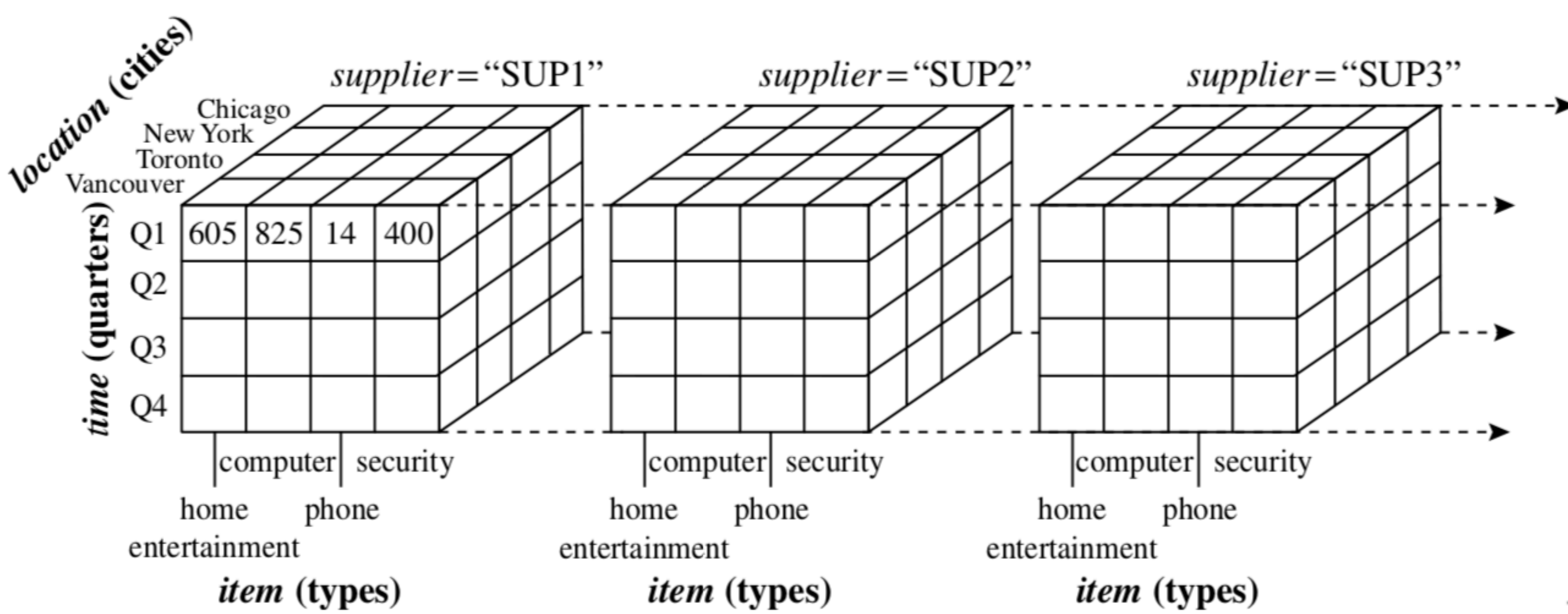
Time	Item	Location	Sales
Q1	Computer	Vancouver	825
Q1	Security	Vancouver	400
Q2	Phone	Vancouver	31
Q2	Security	New York	925
Q3	Security	Chicago	789



- What is a Data Cube in relational database?
- **Why do we need the Cube Operator?**
- How to implement the Cube Operator?

Too Many Dimensions!

- Human are bad at understanding high dimensional data
- Need to reduce the dimension: super aggregation



time (quarter)	item (type) home entertainment	computer	phone	security
Q1	605	825	14	400
Q2	680	952	31	512
Q3	812	1023	30	501
Q4	927	1038	38	580

Operations: Roll-Up

Table 3.a: Sales Roll Up by Model by Year by Color					
Model	**Year**	**Color**	**Sales by Model by Year by Color**	**Sales by Model by Year**	**Sales by Model**
Chevy	1994	black	50		
		white	40		
				90	
	1995	black	85		
		white	115		
				200	
					290

Roll Up →

Operations: Roll-Up

Table 3.a: Sales Roll Up by Model by Year by Color					
Model	**Year**	**Color**	**Sales by Model by Year by Color**	**Sales by Model by Year**	**Sales by Model**
Chevy	1994	black	50		
		white	40		
				90	
	1995	black	85		
		white	115		
				200	
					290

Table 5.a: Sales Summary			
Model	**Year**	**Color**	**Units**
Chevy	1994	black	50
Chevy	1994	white	40
Chevy	1994	ALL	90
Chevy	1995	black	85
Chevy	1995	white	115
Chevy	1995	ALL	200
Chevy	ALL	ALL	290

`ALL` Value: Fill in the super-aggregation items

Operation: Roll-Up

```
SELECT 'ALL', 'ALL', 'ALL', SUM(Sales)
   FROM      Sales
   WHERE     Model = 'Chevy'
UNION
SELECT Model, 'ALL', 'ALL', SUM(Sales)
   FROM      Sales
   WHERE     Model = 'Chevy'
   GROUP BY  Model
UNION
SELECT Model, Year, 'ALL', SUM(Sales)
   FROM      Sales
   WHERE     Model = 'Chevy'
   GROUP BY  Model, Year
UNION
SELECT Model, Year, Color, SUM(Sales)
   FROM      Sales
   WHERE     Model = 'Chevy'
   GROUP BY  Model, Year, Color;
```

Table 5.a: Sales Summary			
Model	**Year**	**Color**	**Units**
Chevy	1994	black	50
Chevy	1994	white	40
Chevy	1994	ALL	90
Chevy	1995	black	85
Chevy	1995	white	115
Chevy	1995	ALL	200
Chevy	ALL	ALL	290
ALL	ALL	ALL	290

N dimensions: N Unions

Operation: Roll-Up

Asymmetric!
Missing:

Table 5.b: Sales Summary rows missing form Table 5.a to convert the roll-up into a cube.			
Model	**Year**	**Color**	**Units**
Chevy	ALL	black	135
Chevy	ALL	white	155

```
UNION
SELECT Model, 'ALL', Color, SUM(Sales)
    FROM       Sales
    WHERE      Model = 'Chevy'
    GROUP BY   Model, Color;
```

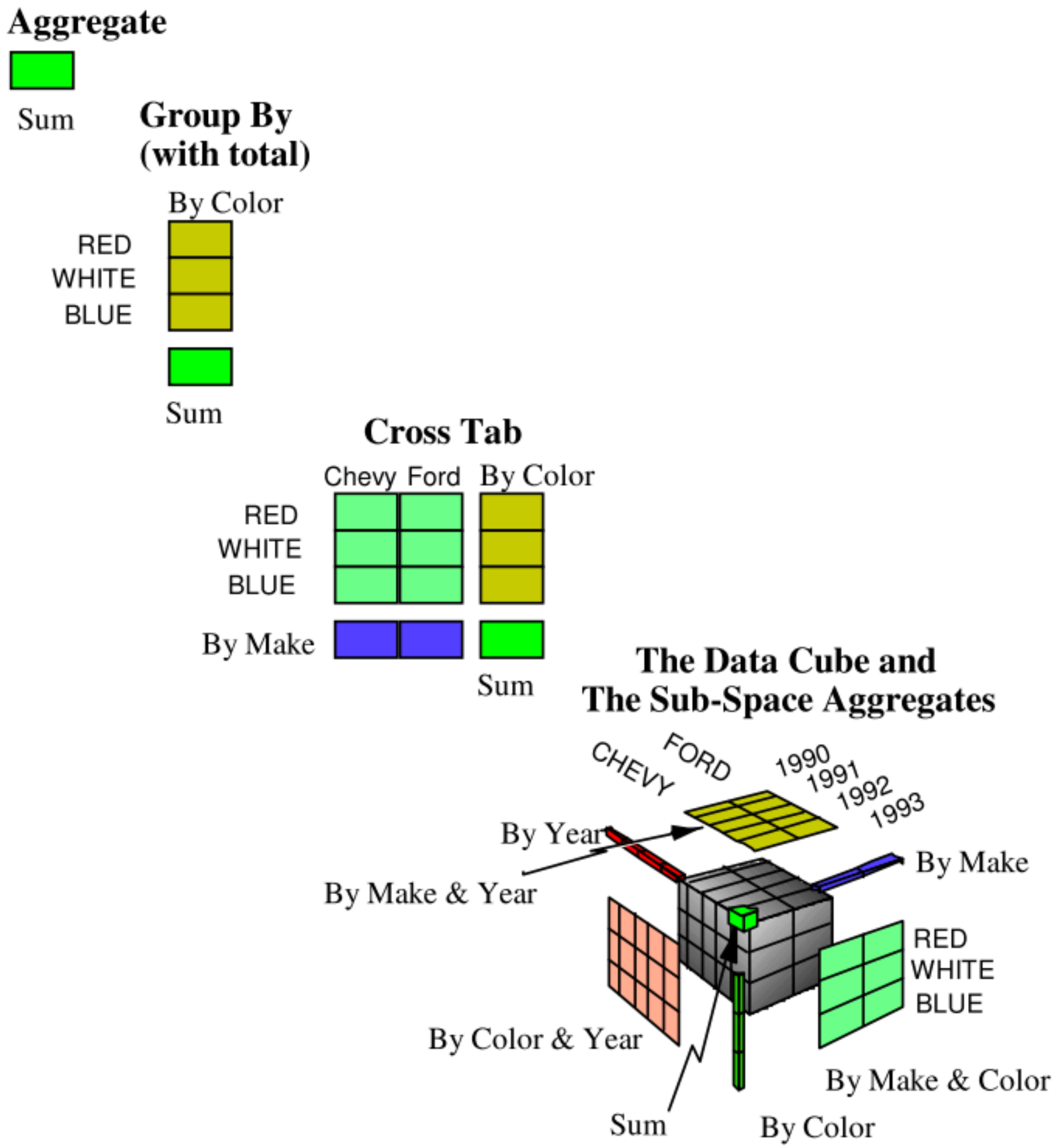
Table 5.a: Sales Summary			
Model	**Year**	**Color**	**Units**
Chevy	1994	black	50
Chevy	1994	white	40
Chevy	1994	ALL	90
Chevy	1995	black	85
Chevy	1995	white	115
Chevy	1995	ALL	200
Chevy	ALL	ALL	290
ALL	ALL	ALL	290

Operation: Cross Tab

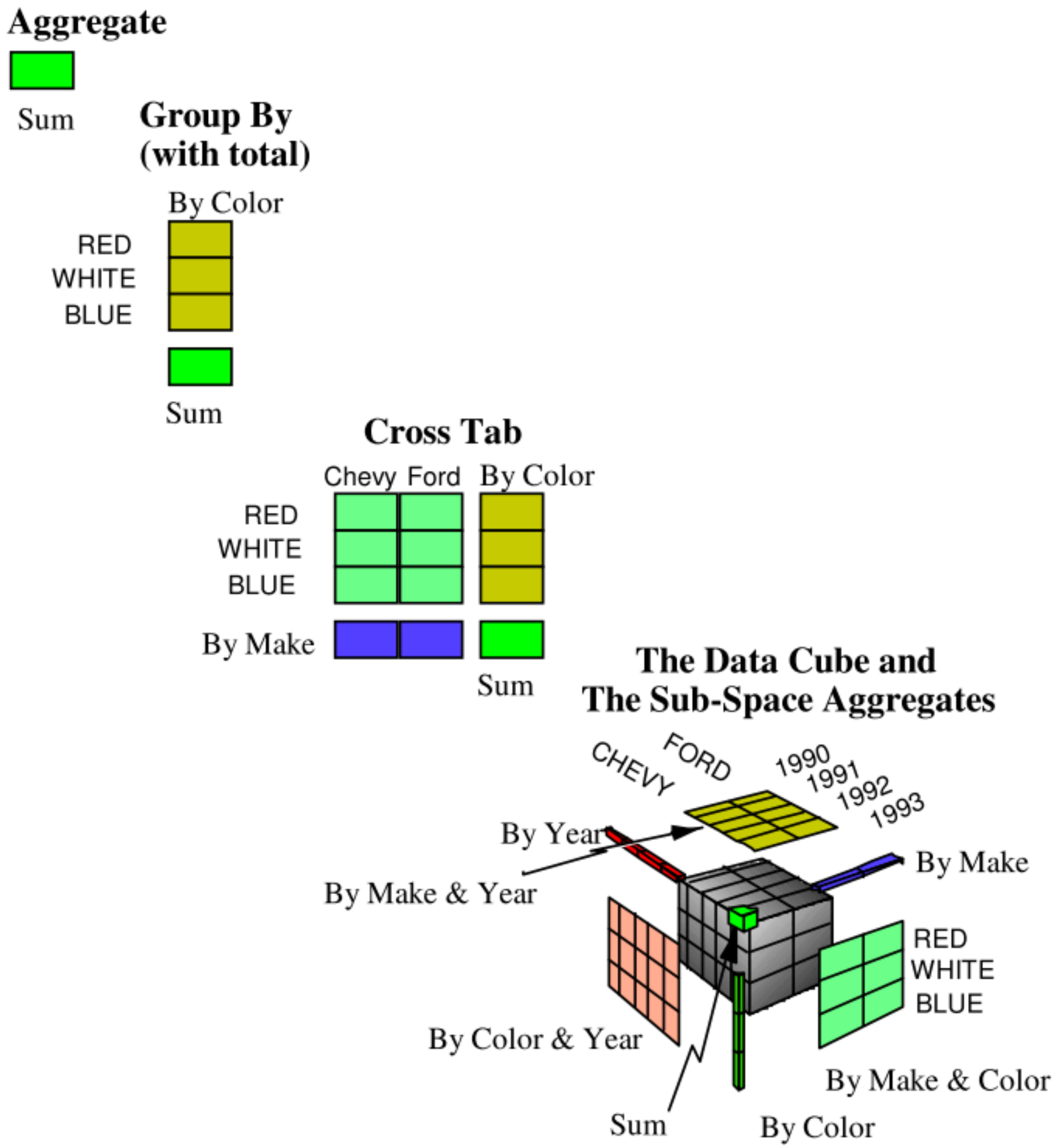
- Cross Tabulation: 2D Symmetric Aggregation Result

Table 6.a: Chevy Sales Cross Tab			
Chevy	**1994**	**1995**	**total** (ALL)
black	50	85	135
white	40	115	155
total (ALL)	90	200	290

3-D Generalization of Cross Tab



3-D Generalization of Cross Tab



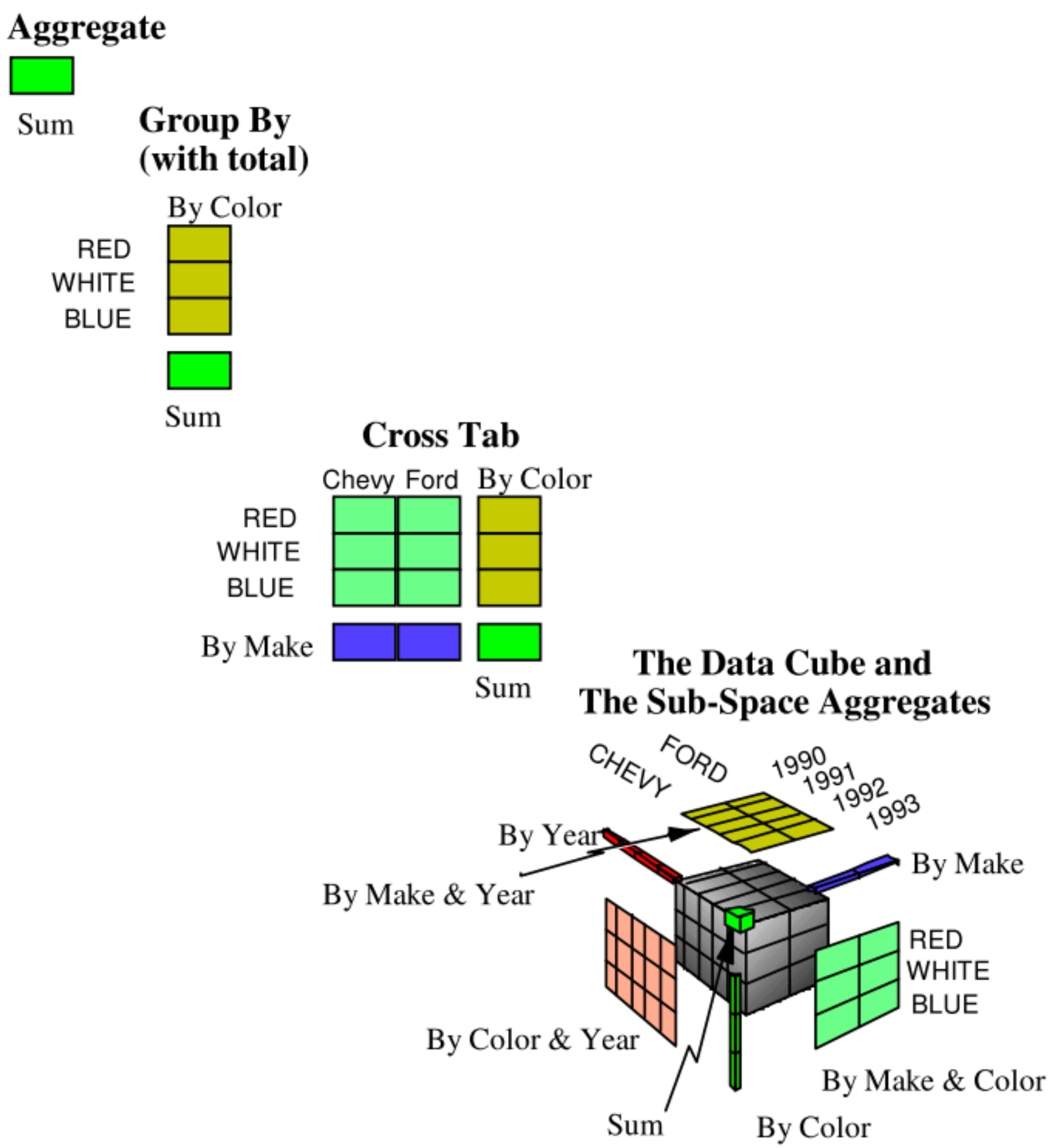

```
SELECT 'ALL', 'ALL', 'ALL', SUM(Sales)
   FROM      Sales
UNION
SELECT Model, 'ALL', 'ALL', SUM(Sales)
   FROM      Sales
GROUP BY  Model
```

-
-
-
-

Group By x 8

```
UNION
SELECT Model, Year, 'ALL', SUM(Sales)
   FROM      Sales
   GROUP BY  Model, Year
UNION
SELECT Model, Year, Color, SUM(Sales)
   FROM      Sales
GROUP BY  Model, Year, Color;
```

N-D Generalization of Cross Tab




```
SELECT 'ALL', 'ALL', 'ALL', SUM(Sales)
   FROM      Sales
UNION
SELECT Model, 'ALL', 'ALL', SUM(Sales)
   FROM      Sales
GROUP BY  Model
```

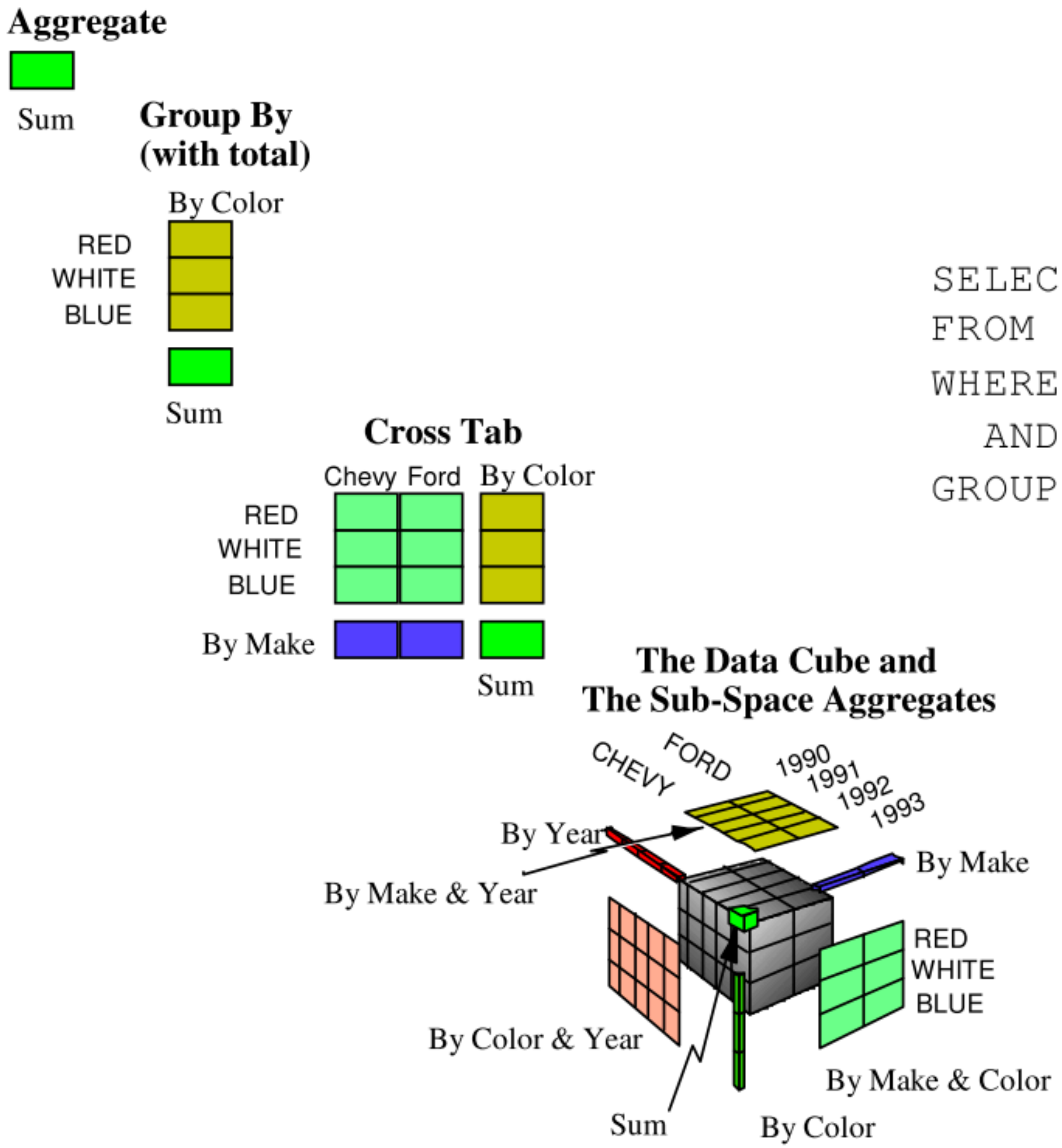
- Group By x 2^N

```
UNION
SELECT Model, Year, 'ALL', SUM(Sales)
   FROM      Sales
   GROUP BY  Model, Year
UNION
SELECT Model, Year, Color, SUM(Sales)
   FROM      Sales
GROUP BY  Model, Year, Color;
```

N-D Generalization of Cross Tab

- Problems
  - Expressing with conventional SQL is exhaustive
  - Too complex to analyze for optimization

The Cube Operator




```
SELECT Model, Year, Color, SUM(sales) AS Sales
FROM Sales
WHERE Model in {'Ford', 'Chevy'}
  AND Year BETWEEN 1990 AND 1992
GROUP BY CUBE Model, Year, Color;
```

- What is a Data Cube in relational database?
- Why do we need the Cube Operator?
- **How to implement the Cube Operator?**
  - How the Cube fit in SQL?
  - **How to compute the Cube?**

Compute The Cube

```
SELECT Model, Year, Color, SUM(sales) AS Sales
FROM Sales
WHERE Model in {'Ford', 'Chevy'}
  AND Year BETWEEN 1990 AND 1992
GROUP BY CUBE Model, Year, Color;
```

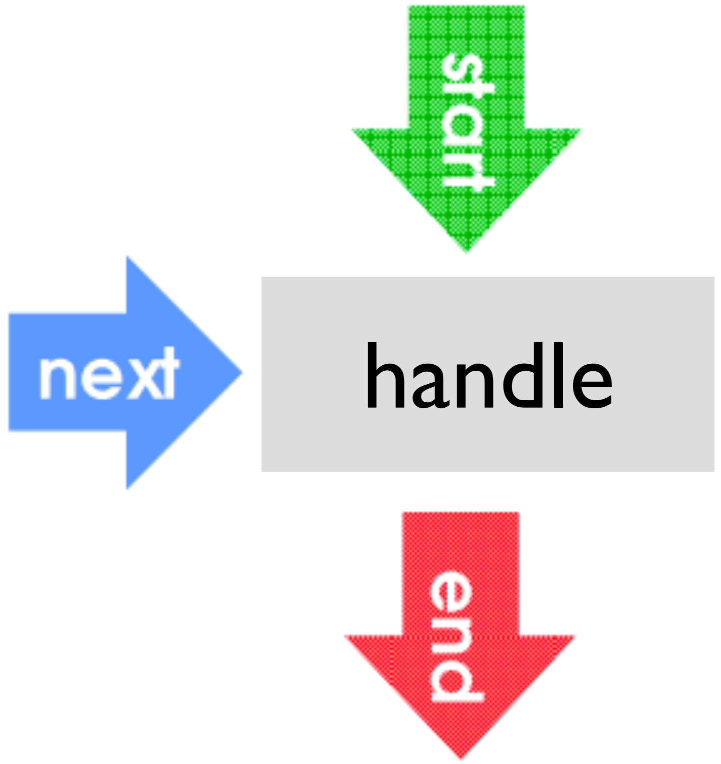
=

```
SELECT 'ALL', 'ALL', 'ALL', SUM(Sales)
   FROM      Sales
UNION
SELECT Model, 'ALL', 'ALL', SUM(Sales)
   FROM      Sales
GROUP BY  Model
```

⋮

```
UNION
SELECT Model, Year, 'ALL', SUM(Sales)
   FROM      Sales
   GROUP BY  Model, Year
UNION
SELECT Model, Year, Color, SUM(Sales)
   FROM      Sales
GROUP BY  Model, Year, Color;
```

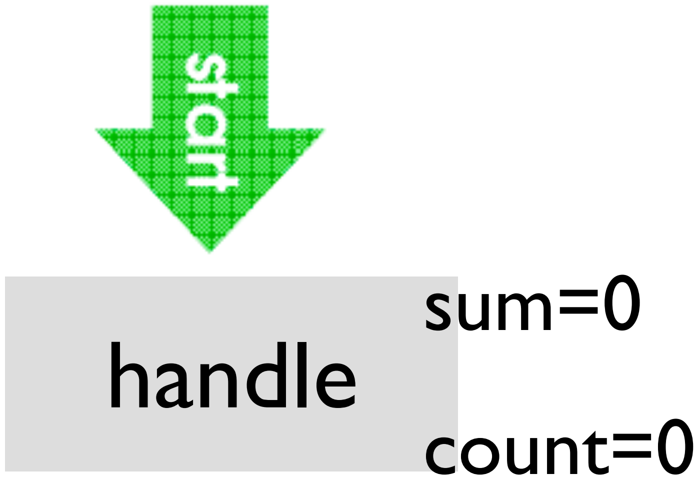
Implementation of Aggregate Functions



- **Start - Init (&handle)**: Allocates the handle and initializes the aggregate computation
- **Next - Iter (&handle, value)**: Aggregates the next value into the current aggregate
- **End - Final(&handle)**: Computes and returns the resulting aggregate by using data saved in the handle. This invocation deallocates the handle

Implementation of Aggregate Functions

Example of AVG



- **Start - Init (&handle)**: Allocates the handle and initializes the aggregate computation
- **Next - Iter (&handle, value)**: Aggregates the next value into the current aggregate
- **End - Final(&handle)**: Computes and returns the resulting aggregate by using data saved in the handle. This invocation deallocates the handle

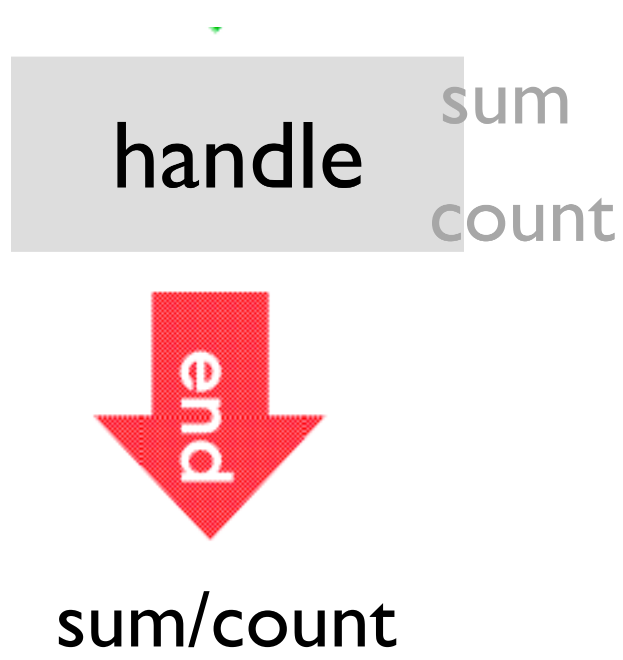
Implementation of Aggregate Functions

Example of AVG

- **Start - Init (&handle)**: Allocates the handle and initializes the aggregate computation
- **Next - Iter (&handle, value)**: Aggregates the next value into the current aggregate
- **End - Final(&handle)**: Computes and returns the resulting aggregate by using data saved in the handle. This invocation deallocates the handle

Implementation of Aggregate Functions

Example of AVG



- **Start - Init (&handle)**: Allocates the handle and initializes the aggregate computation
- **Next - Iter (&handle, value)**: Aggregates the next value into the current aggregate
- **End - Final(&handle)**: Computes and returns the resulting aggregate by using data saved in the handle. This invocation deallocates the handle

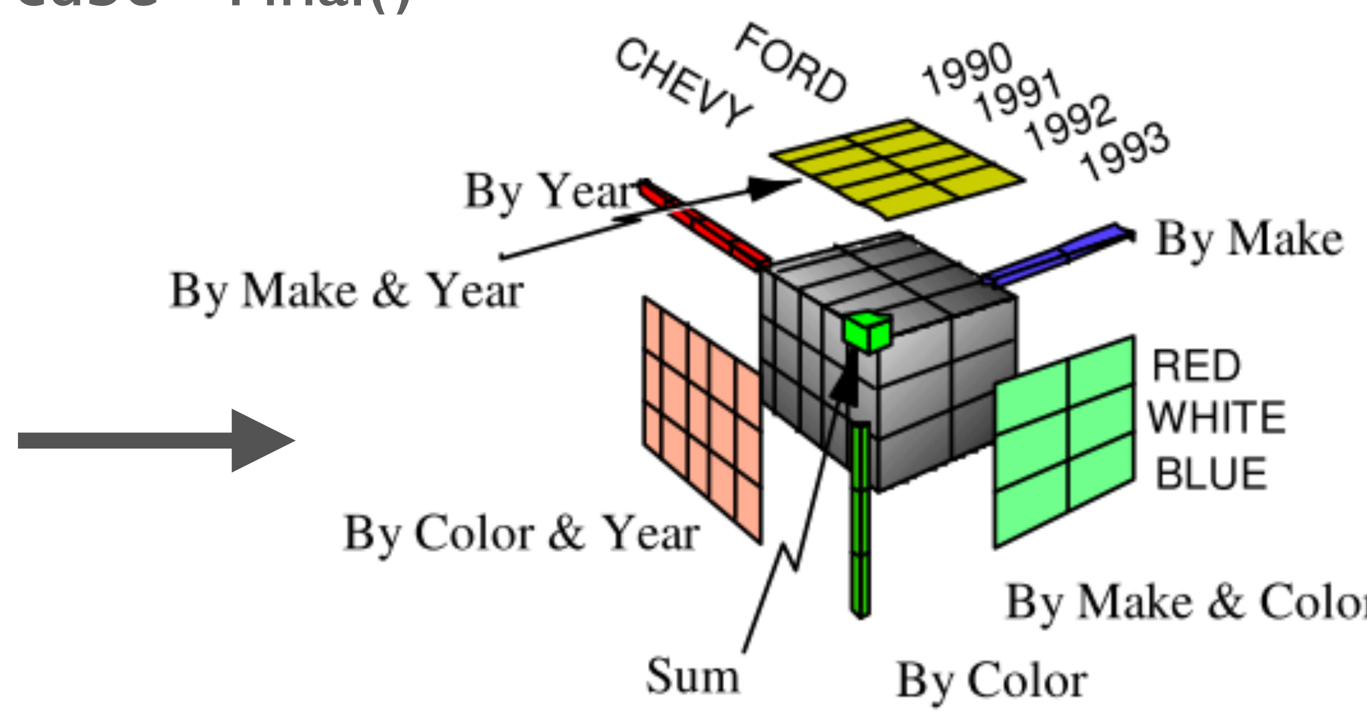
2^N Algorithm

1. Allocate a handle for each cell of the cube - Init()

2. Each tuple needs to invoke the Iter() function once for the cells that match the tuple

3. Compute result for each cell of the cube - Final()

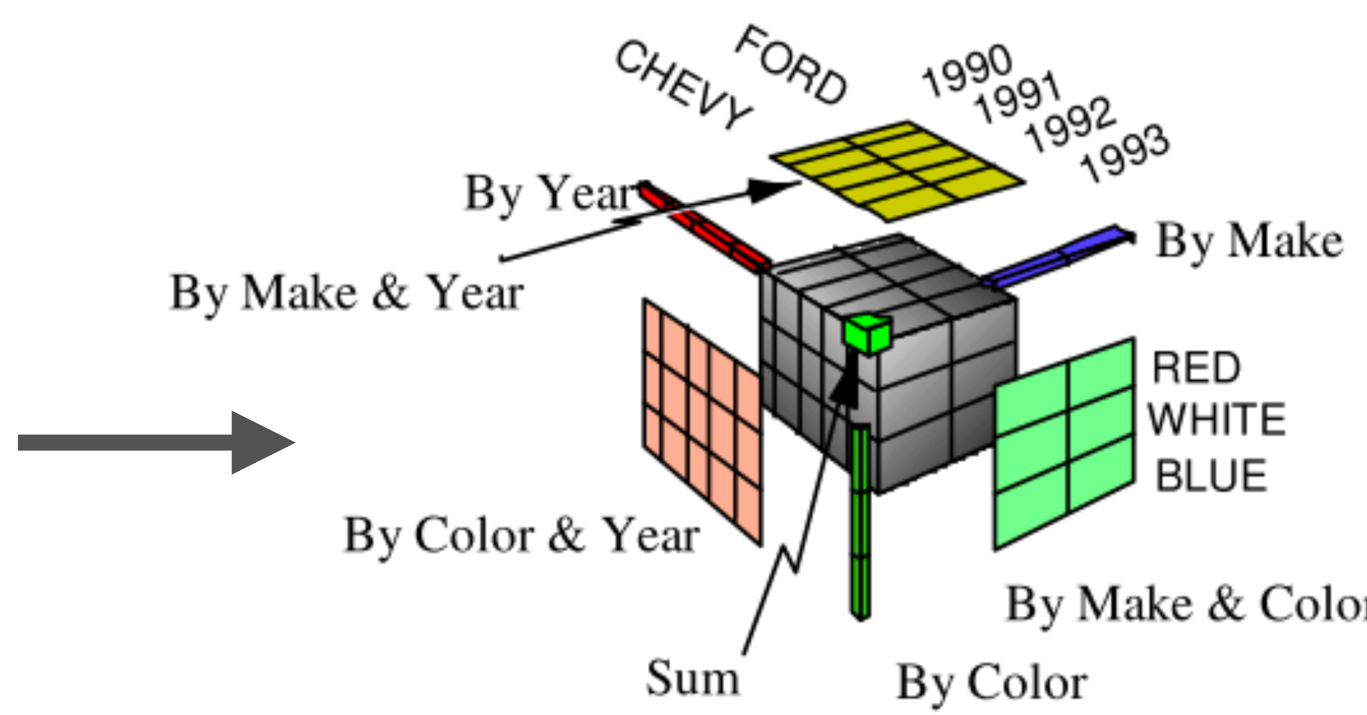
Branch	Model	Year	Color	Sales
Burnaby	Chevy	1990	red	23
Richmond	Chevy	1990	white	14
Richmond	Chevy	1990	white	31
Burnaby	Ford	1990	blue	23
Richmond	Ford	1990	red	4
Burnaby	Chevy	1991	blue	22
Richmond	Ford	1992	red	32
…	…	…	…	…



2^N Algorithm

- Invoke Init() & Final() one time for each cell
- Invoke Iter() 2^N times for each tuple

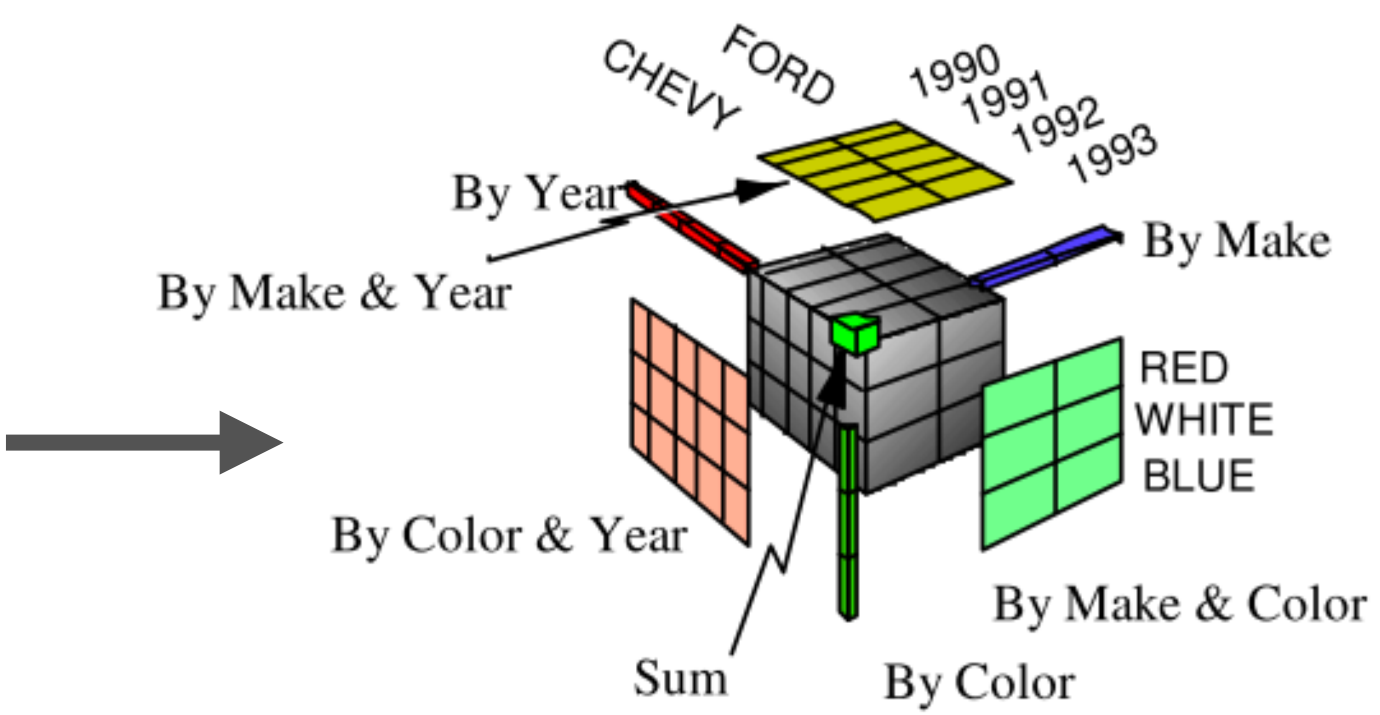
Branch	Model	Year	Color	Sales
Burnaby	Chevy	1990	red	23
Richmond	Chevy	1990	white	14
Richmond	Chevy	1990	white	31
Burnaby	Ford	1990	blue	23
Richmond	Ford	1990	red	4
Burnaby	Chevy	1991	blue	22
Richmond	Ford	1992	red	32
…	…	…	…	…



2^N Algorithm

- Invoke Init() & Final() one time for each cell
- Invoke Iter() 2^N times **for each tuple** ← Can be optimized

Branch	Model	Year	Color	Sales
Burnaby	Chevy	1990	red	23
Richmond	Chevy	1990	white	14
Richmond	Chevy	1990	white	31
Burnaby	Ford	1990	blue	23
Richmond	Ford	1990	red	4
Burnaby	Chevy	1991	blue	22
Richmond	Ford	1992	red	32
…	…	…	…	…

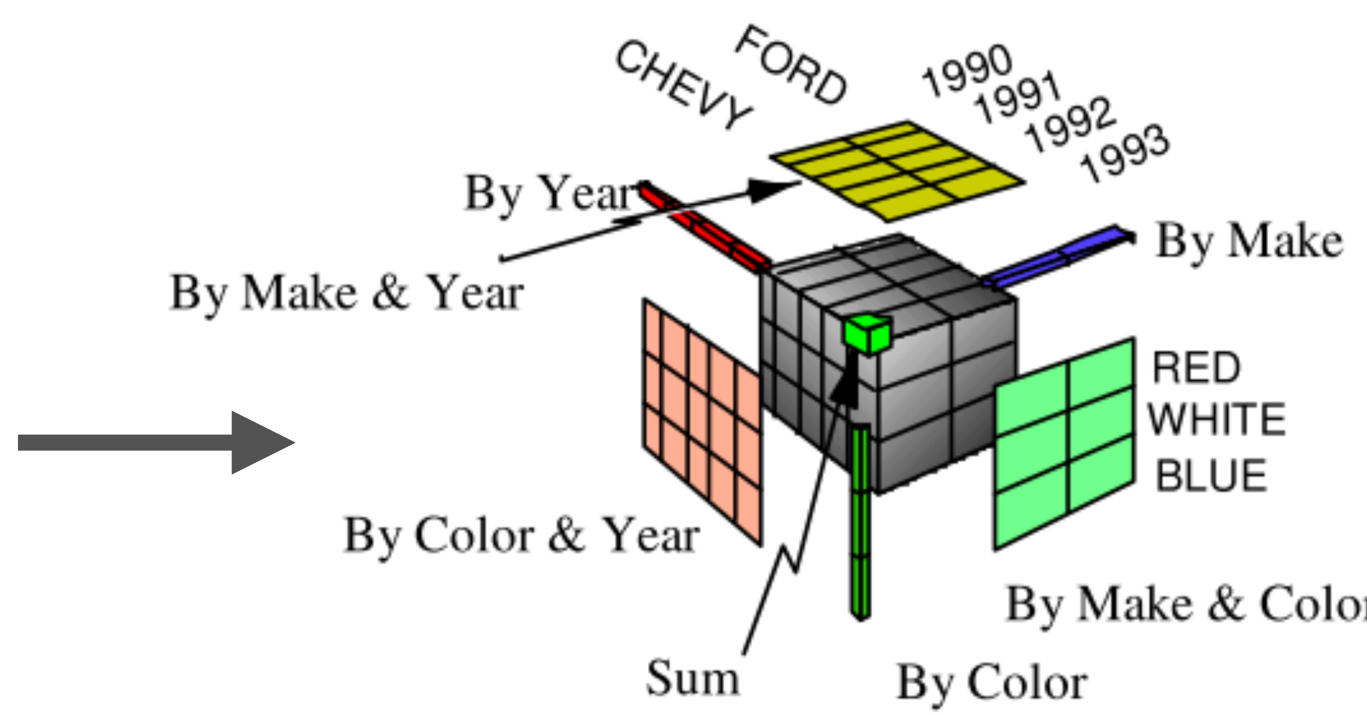


Computing the Cube

- Speed Up the Process
  - Make use of the middle result:

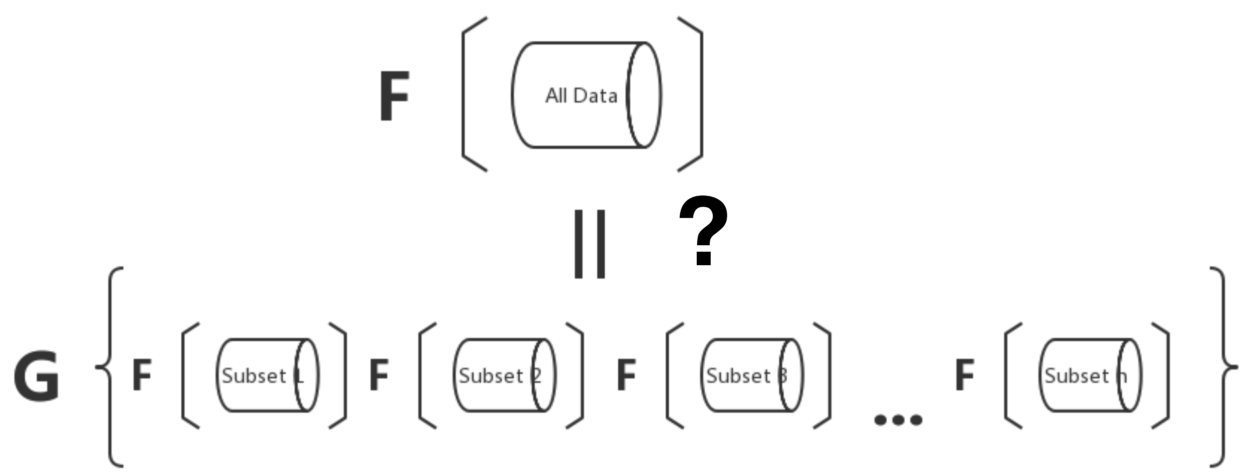
    N-D Aggregate -> (N-1)-D Aggregate

Branch	Model	Year	Color	Sales
Burnaby	Chevy	1990	red	23
Richmond	Chevy	1990	white	14
Richmond	Chevy	1990	white	31
Burnaby	Ford	1990	blue	23
Richmond	Ford	1990	red	4
Burnaby	Chevy	1991	blue	22
Richmond	Ford	1992	red	32
…	…	…	…	…

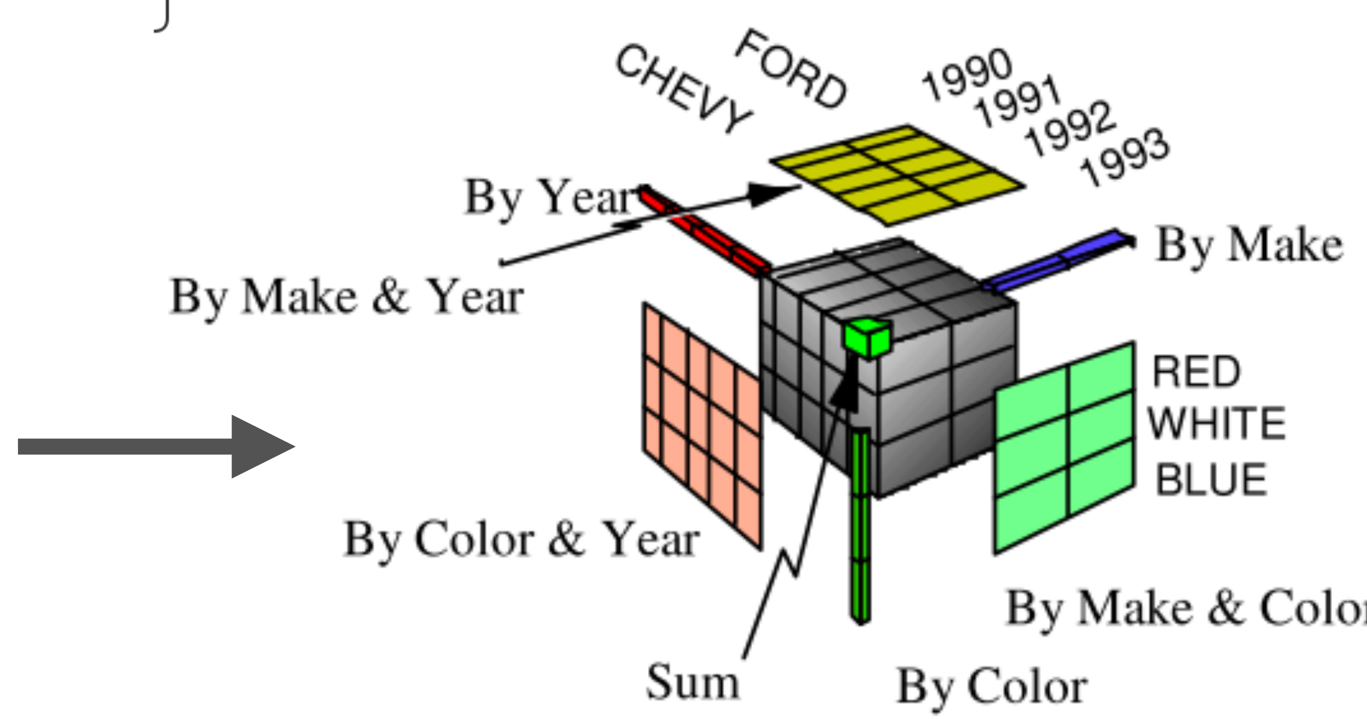


Computing the Cube

Can F be computed in distributive manner?



Branch	Model	Year	Color	Sales
Burnaby	Chevy	1990	red	23
Richmond	Chevy	1990	white	14
Richmond	Chevy	1990	white	31
Burnaby	Ford	1990	blue	23
Richmond	Ford	1990	red	4
Burnaby	Chevy	1991	blue	22
Richmond	Ford	1992	red	32
…	…	…	…	…



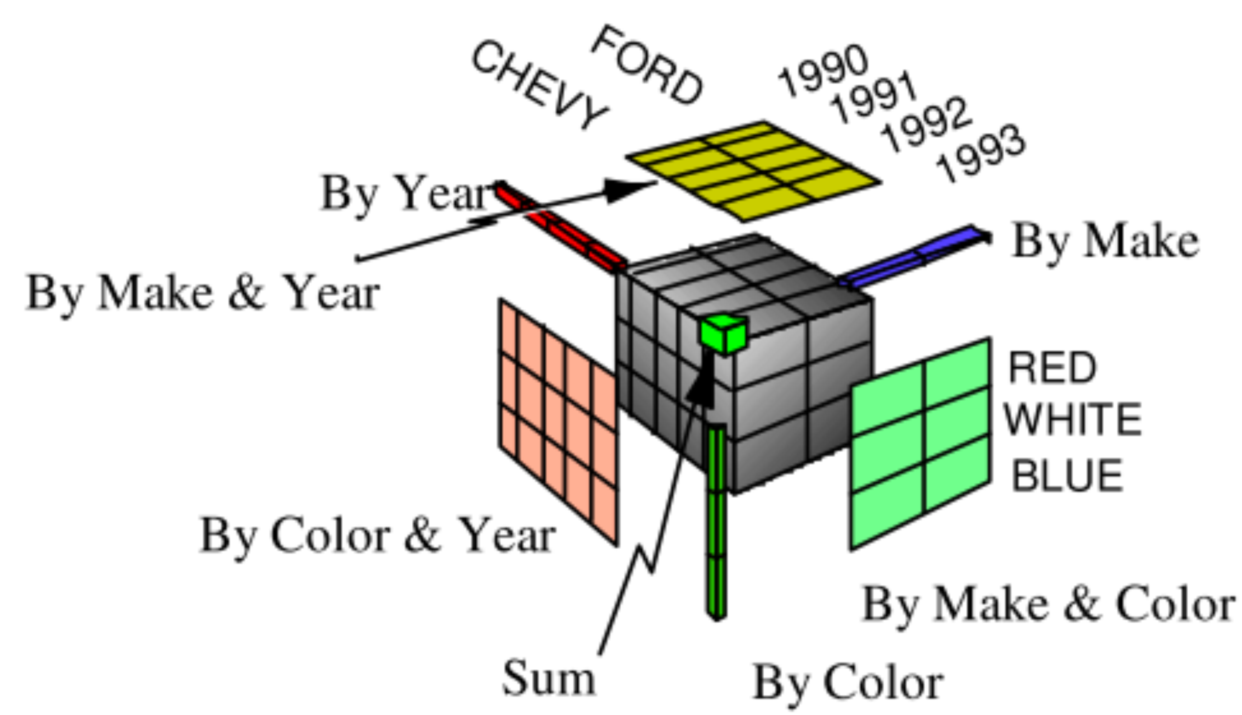
Aggregate Functions Classification

- Distributive: SUM(), MIN(), MAX(), COUNT()
  - Can be computed in a distributive manner
- Algebraic: AVG(), MaxN(), MinN()
  - Can be computed in a distributive manner with m arguments - need to keep both the handle & the result for each cell
- Holistic: Median()
  - No constant m exists - need to scan all the tuples

Computing the Cube

- Speed Up the Process for Distributive & Algebraic Functions
  - Make use of the middle result

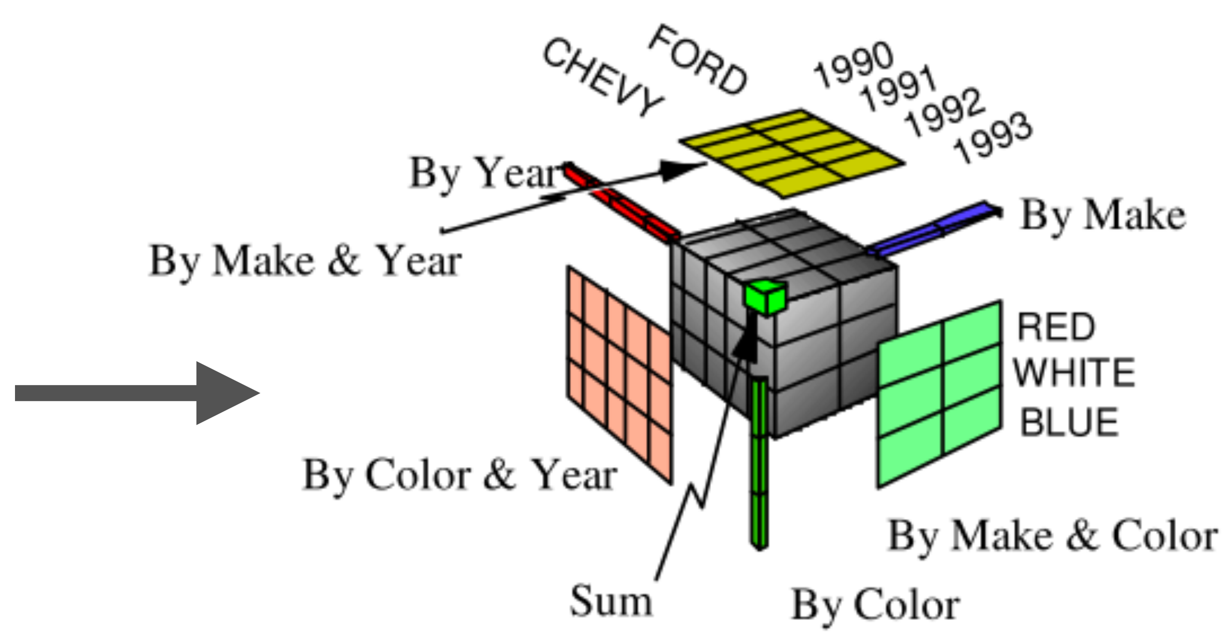
  Aggregate on the smallest list



Maintaining The Cube

- Trigger Conditions: UPDATE, INSERT, DELETE
- Can be different for the same function: MAX()
  - INSERT: Distributive
  - DELETE/UPDATE: Holistic

Branch	Model	Year	Color	Sales
Burnaby	Chevy	1990	red	23
Richmon	Chevy	1990	white	14
Richmon	Chevy	1990	white	31
Burnaby	Ford	1990	blue	23
Richmon	Ford	1990	red	4
Burnaby	Chevy	1991	blue	22
Richmon	Ford	1992	red	32
…	…	…	…	…



TakeAways

- The cube operator computes aggregations over all possible subsets of the specified dimensions
- The result of the cube operator can be modeled as a data cube
- We can speed up the computation of the cube for many common aggregate functions by using the middle result

Q & A