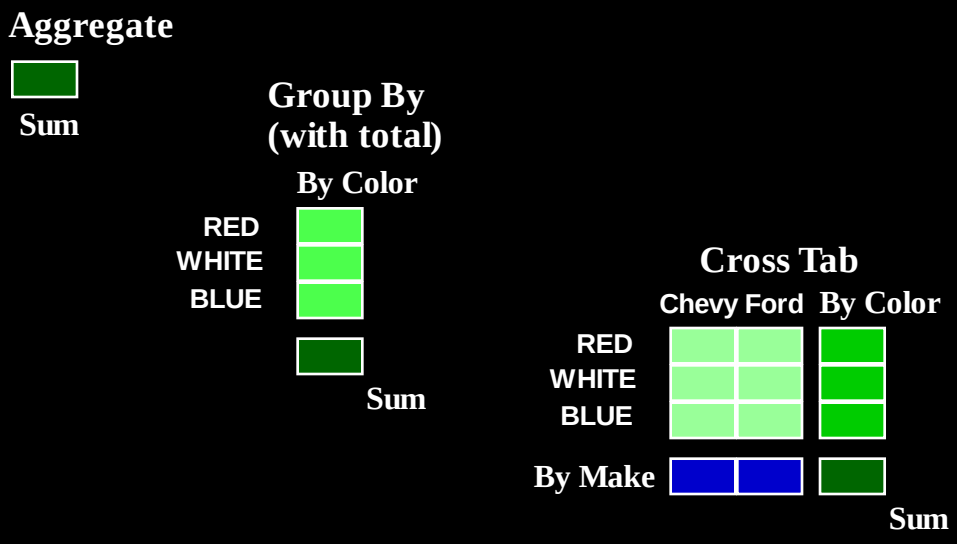
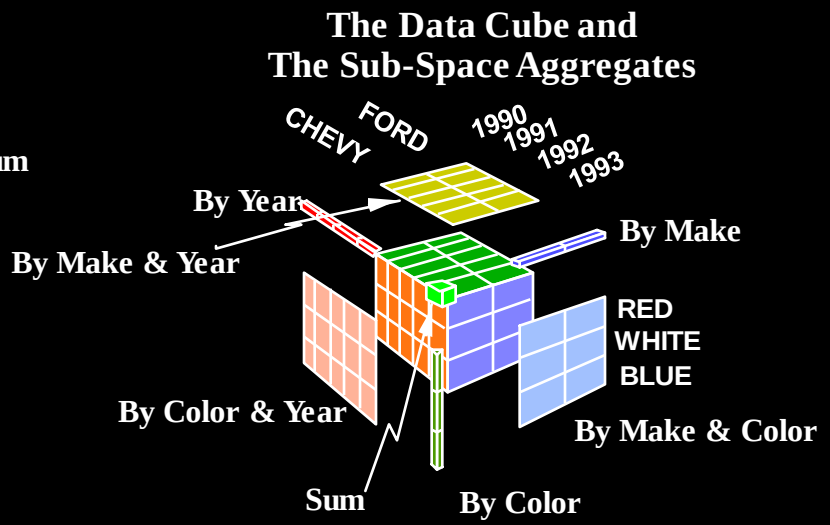
# CUBE:
# A Relational Aggregate Operator Generalizing Group By





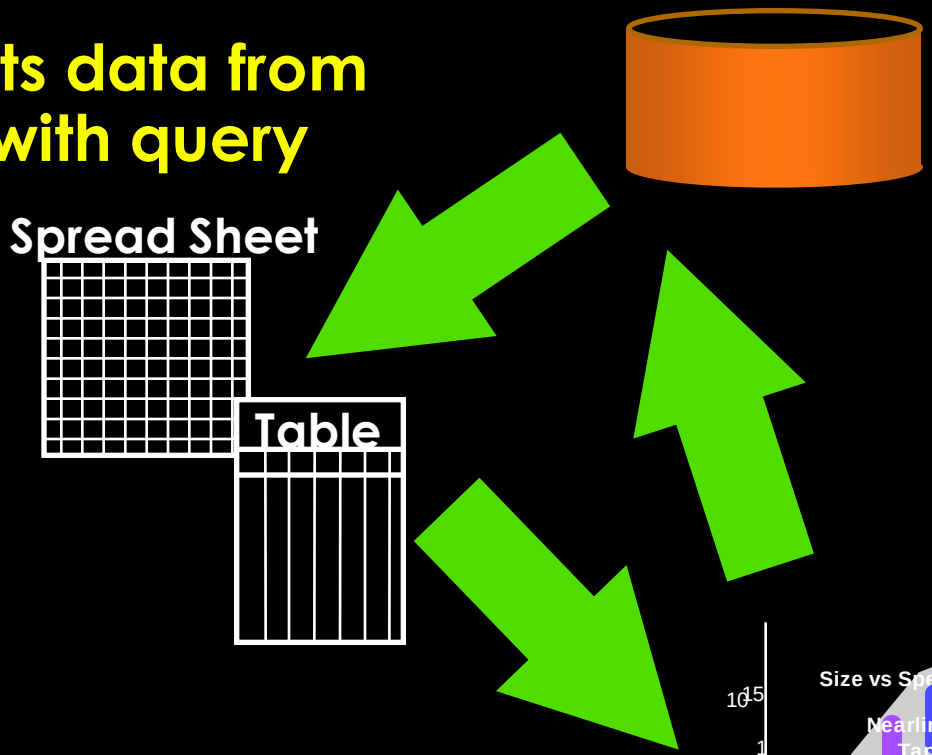
Jim Gray
Adam Bosworth Hamid Pirahesh
Andrew Layman IBM
Microsoft

Gray@ Microsoft.com

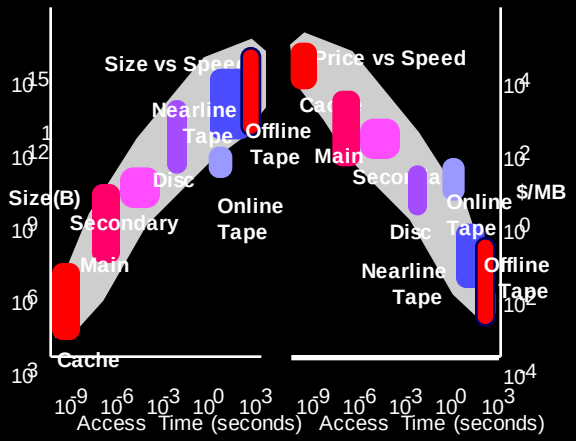



# The Data Analysis Cycle

- User extracts data from database with query

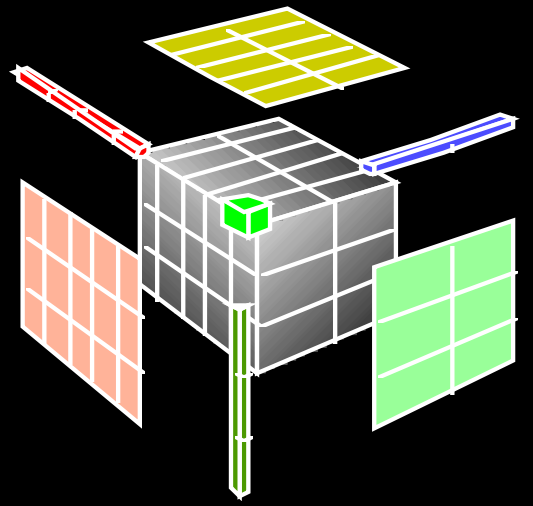



- Then visualizes, analyzes data with desktop tools

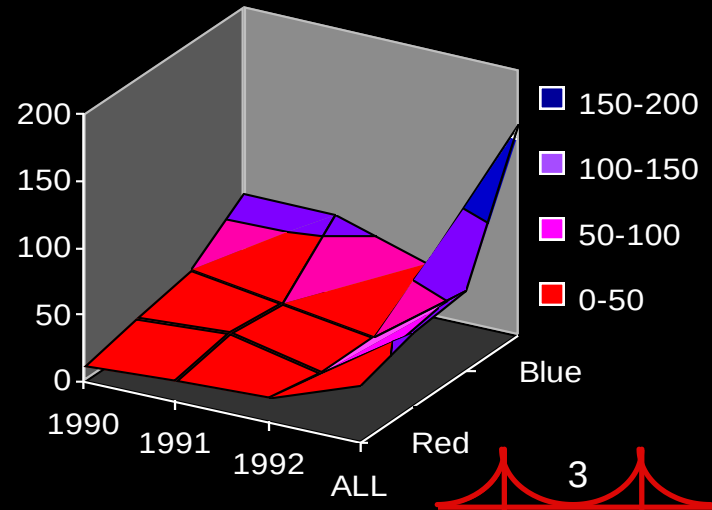


# Division of labor
# Computation vs Visualization

- Relational system builds CUBE relation
  - aggregation best done close to data
  - Much filtering of data possible
  - Cube computation may be recursive
    - (e.g., percent of total, quartile, ....)



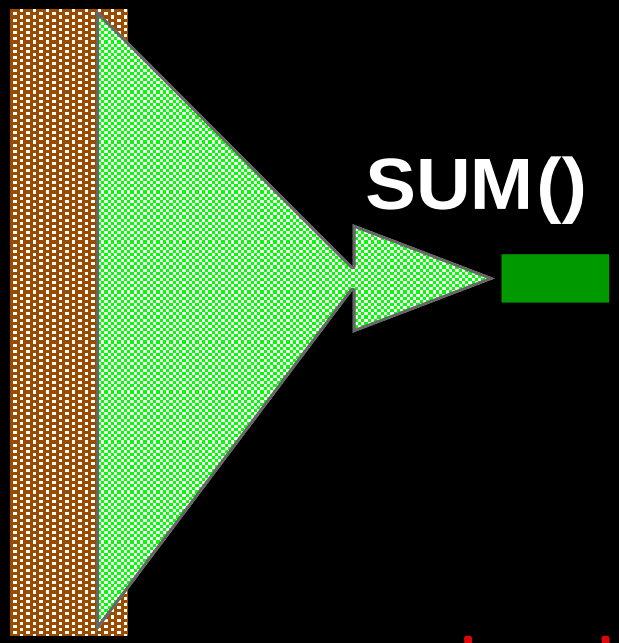
- Visualization System displays/explores the cube



# Relational Aggregate Operators

- SQL has several aggregate operators:
  - sum(), min(), max(), count(), avg()
- Other systems extend this with many others:
  - stat functions, financial functions, ...
- The basic idea is:
  - Combine all values in a column
  - into a single scalar value.
- Syntax

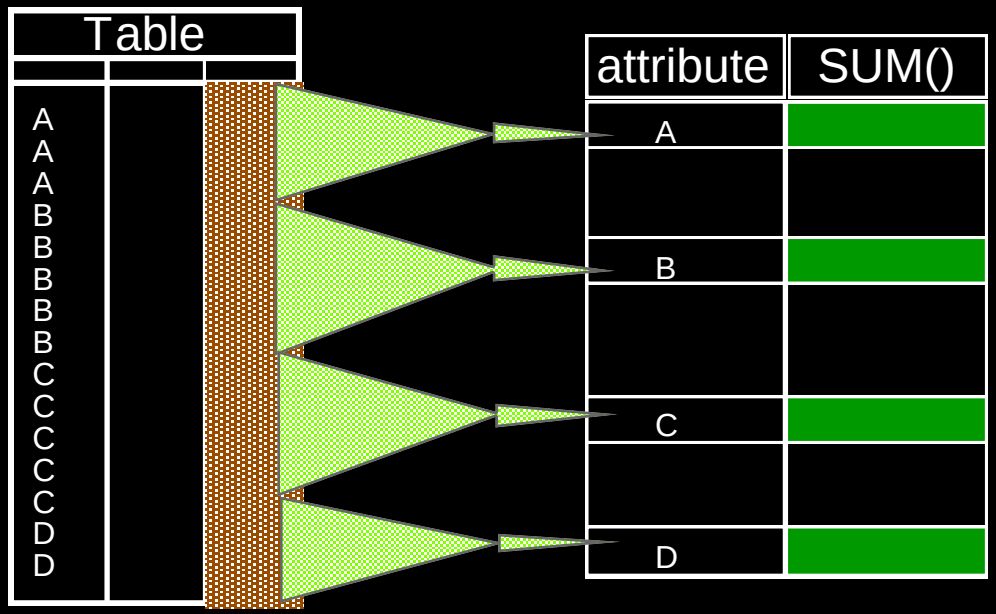
```
select  sum(units)
from    inventory;
```



# Relational Group By Operator

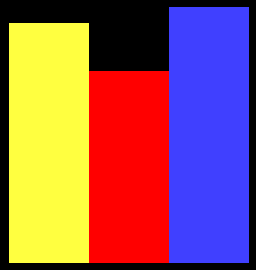
- Group By allows aggregates over table sub-groups
- Result is a new table
- Syntax:

```
select    location, sum(units)
from      inventory
group by  location
having    nation = "USA";
```

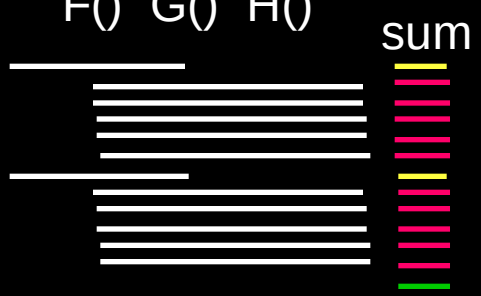


# Problems With This Design

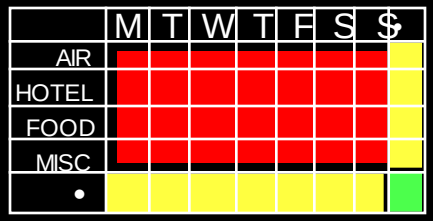
- Users Want Histograms




- Users want sub-totals and totals
  - drill-down & roll-up reports

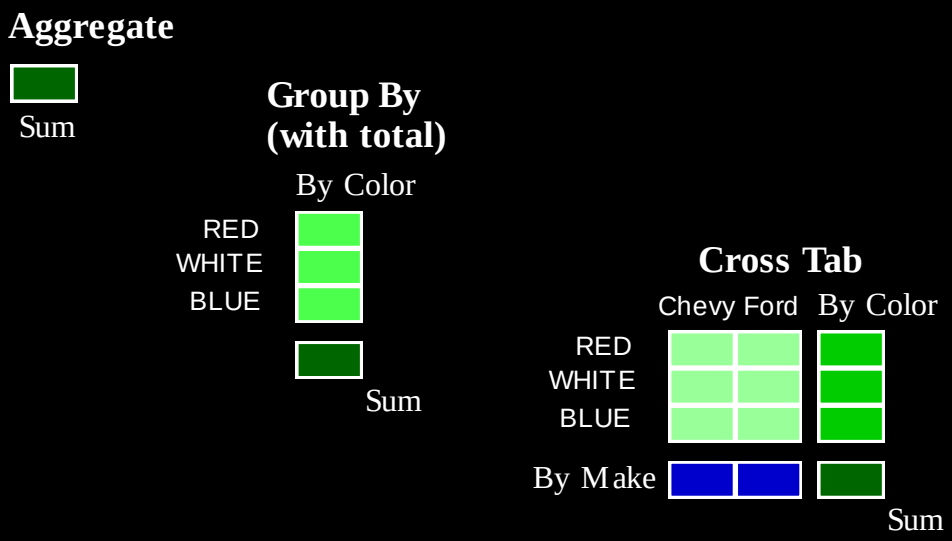


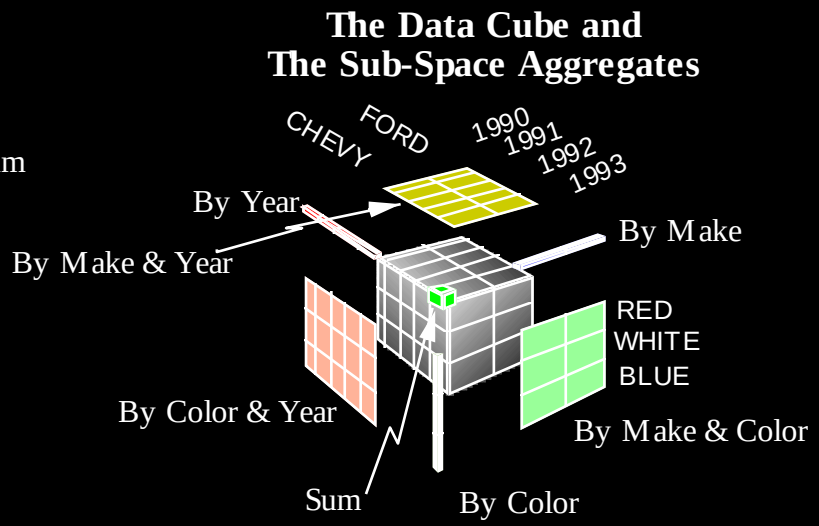

- Users want CrossTabs




- Conventional wisdom
  - These are not relational operators
  - They are in many report writers and query engines

# Thesis:
# The Data CUBE Relational Operator Generalizes Group By and Aggregates



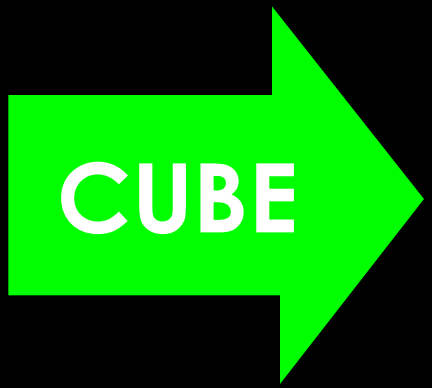



# The Idea: Think of the N-dimensional Cube Each Attribute is a Dimension

- N-dimensional Aggregate (sum(), max(),...)
  - fits relational model exactly:
    - » $a_1$, $a_2$, ...., $a_N$, f()
- Super-aggregate over *N-1* Dimensional sub-cubes
    - » ALL, $a_2$, ...., $a_N$ , f()
    - » $a_3$ , ALL, $a_3$, ...., $a_N$ , f()
    - » ...
    - » $a_1$, $a_2$, ...., ALL, f()
  - this is the *N-1* Dimensional cross-tab.
- Super-aggregate over *N-2* Dimensional sub-cubes
    - » ALL, ALL, $a_3$, ...., $a_N$ , f()
    - » ...
    - » $a_1$, $a_2$ ,...., ALL, ALL, f()

# An Example

| SALES | | | |
|---|---|---|---|
| Model | Year | Color | Sales |
| Chevy | 1990 | red | 5 |
| Chevy | 1990 | white | 87 |
| Chevy | 1990 | blue | 62 |
| Chevy | 1991 | red | 54 |
| Chevy | 1991 | white | 95 |
| Chevy | 1991 | blue | 49 |
| Chevy | 1992 | red | 31 |
| Chevy | 1992 | white | 54 |
| Chevy | 1992 | blue | 71 |
| Ford | 1990 | red | 64 |
| Ford | 1990 | white | 62 |
| Ford | 1990 | blue | 63 |
| Ford | 1991 | red | 52 |
| Ford | 1991 | white | 9 |
| Ford | 1991 | blue | 55 |
| Ford | 1992 | red | 27 |
| Ford | 1992 | white | 62 |
| Ford | 1992 | blue | 39 |



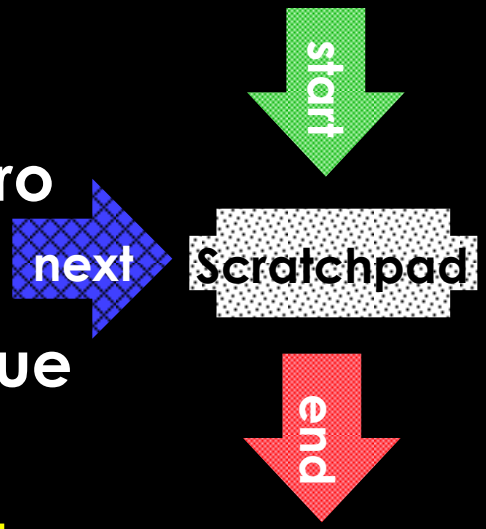

| DATA CUBE | | | |
|---|---|---|---|
| Model | Year | Color | Sales |
| ALL | ALL | ALL | 942 |
| chevy | ALL | ALL | 510 |
| ford | ALL | ALL | 432 |
| ALL | 1990 | ALL | 343 |
| ALL | 1991 | ALL | 314 |
| ALL | 1992 | ALL | 285 |
| ALL | ALL | red | 165 |
| ALL | ALL | white | 273 |
| ALL | ALL | blue | 339 |
| chevy | 1990 | ALL | 154 |
| chevy | 1991 | ALL | 199 |
| chevy | 1992 | ALL | 157 |
| ford | 1990 | ALL | 189 |
| ford | 1991 | ALL | 116 |
| ford | 1992 | ALL | 128 |
| chevy | ALL | red | 91 |
| chevy | ALL | white | 236 |
| chevy | ALL | blue | 183 |
| ford | ALL | red | 144 |
| ford | ALL | white | 133 |
| ford | ALL | blue | 156 |
| ALL | 1990 | red | 69 |
| ALL | 1990 | white | 149 |
| ALL | 1990 | blue | 125 |
| ALL | 1991 | red | 107 |
| ALL | 1991 | white | 104 |
| ALL | 1991 | blue | 104 |
| ALL | 1992 | red | 59 |
| ALL | 1992 | white | 116 |
| ALL | 1992 | blue | 110 |

# Interesting Aggregate Functions

- From RedBrick systems
  - Rank (in sorted order)
  - N-Tile (histograms)
  - Running average (cumulative functions)
  - Windowed running average
  - Percent of total
- Users want to define their own aggregate functions
  - statistics
  - domain specific

# User Defined Aggregates

- Idea:
  - User function is called at start of each group
  - Each function instance has scratchpad
  - Function is called at end of group
- Example: SUM
  - START: allocates a cell and sets it to zero
  - NEXT: adds next value to cell
  - END: deallocates cell and returns value
  - Simple example: MAX()
- This idea is in Illustra, IBM's DB2/CS, and others
- Needs extension for rollup and cube



# User Defined Aggregate Function Generalized For Cubes

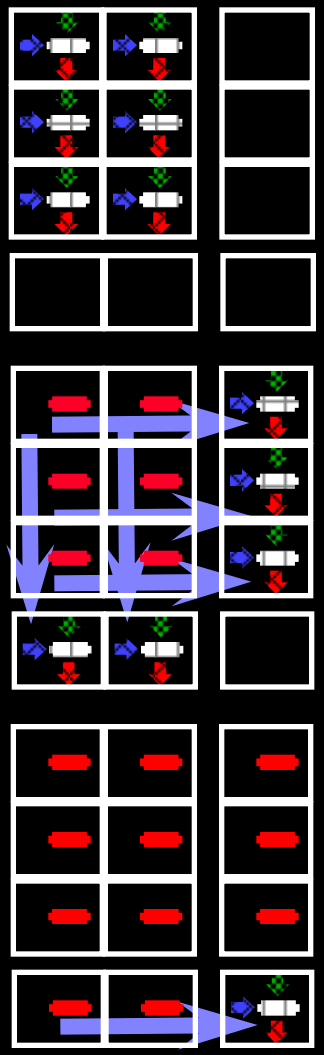
- Aggregates have graduated difficulty
  - Distributive: can compute cube from next lower dimension values (count, min, max,...)
  - Algebraic: can compute cube from next lower lower scratchpads (average, ...)
  - Holistic: Need base data (Median, Mode, Rank..)
- Distributive and Algebraic have simple and efficient algorithm: build higher dimensions from core
- Holistic computation seems to require multiple passes.
  - real systems use sampling to estimate them
    - (e.g., sample to find median, quartile boundaries)

# How To Compute the Cube?

- **If each attribute has $N_i$ values**
  **CUBE has $\Pi (N_i+1)$ values**
- **Compute N-D cube with hash if fits in RAM**
- **Compute N-D cube with sort if overflows RAM**
- **Same comments apply to subcubes:**
  - compute N-D-1 subcube from N-D cube.
  - Aggregate on "biggest" domain first when >1 deep
  - Aggregate functions need hidden variables:
    - e.g. average needs sum and count.
- **Use standard techniques from query processing**
  - arrays, hashing, hybrid hashing
  - fall back on sorting.

# Example:

- Compute 2D core of 2 x 3 cube
- Then compute 1D edges
- Then compute 0D point
- Works for algebraic and distributive functions
  Saves "lots" of calls



# Summary

- CUBE operator generalizes relational aggregates
- Needs ALL value to denote sub-cubes
  - ALL values represent aggregation sets
- Needs generalization of user-defined aggregates
- Decorations and abstractions are interesting
- Computation has interesting optimizations
- Relationship to “rest of SQL” not fully worked out.