CUBE MATERIALIZATION

E0 261

Jayant Haritsa
Computer Science and Automation
Indian Institute of Science

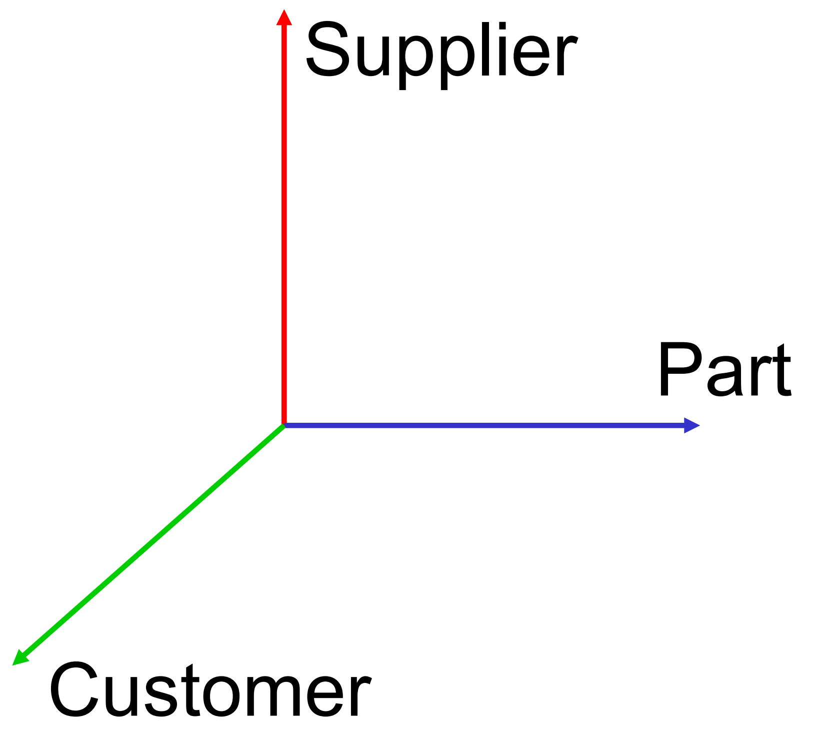
Views and Decision Support

- OLAP queries are typically aggregate queries.
  - Precomputation is essential for interactive response times.
  - The CUBE is in fact a collection of aggregate queries, and precomputation is especially important: lots of work on what is best to precompute given a limited amount of space to store precomputed results.
- Warehouses can be thought of as a collection of asynchronously replicated tables and periodically maintained views.

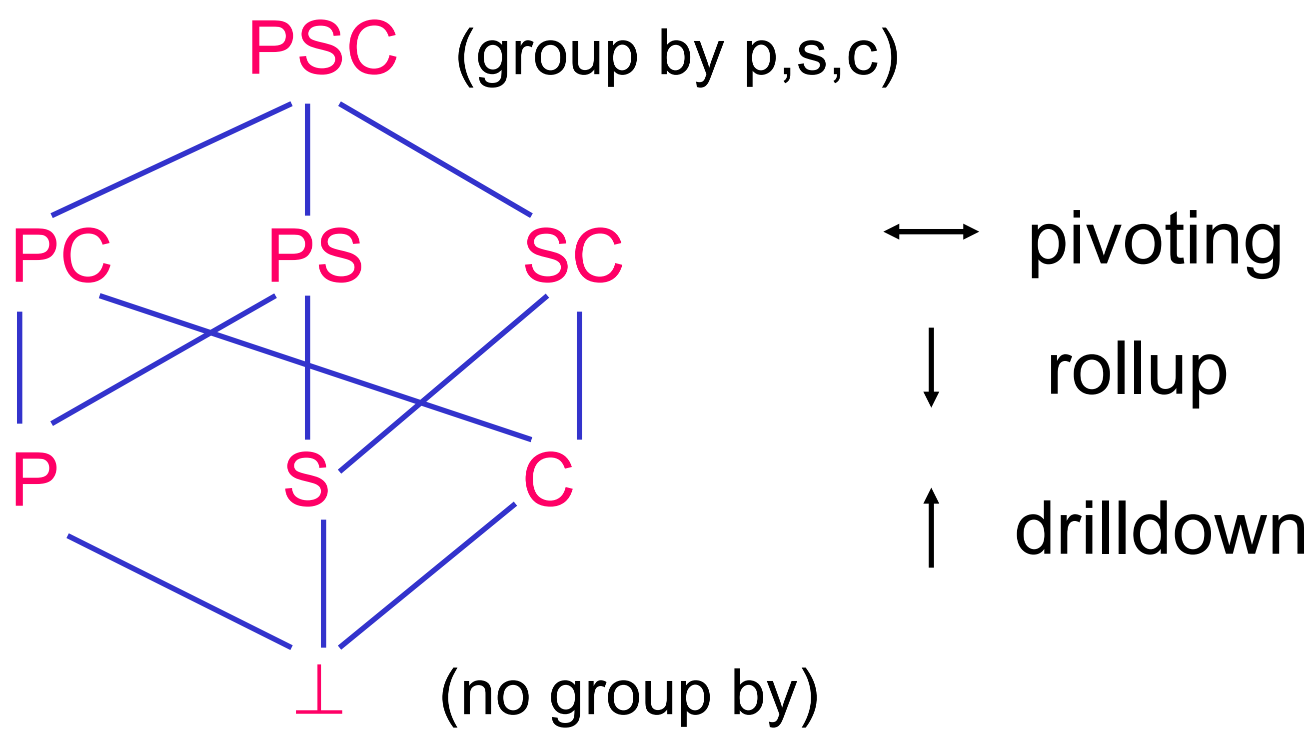
Issues in View Materialization

- What views should we materialize, and what indexes should we build on the precomputed results?
- Given a query and a set of materialized views, can we use the materialized views to answer the query?
- How frequently should we refresh materialized views to make them consistent with the underlying tables?
  (And how can we do this incrementally?)

TPC-D Example



View Lattice



- Given N dimensions, 2^N views in lattice

Materialization Options

- Materialize everything
  - minimum response time
  - space explosion
- Materialize nothing
  - maximum response time
  - zero space
- Materialize a carefully chosen subset and derive others from this subset
  - e.g. Any view can be derived from PSC
  - today's paper (received Best Paper award in Sigmod 96 !)

Problem Formulation

- Given a view lattice and a constraint on the number of views that can be materialized, which choice will result in minimizing the average cost across all views?
- Assumptions:
  - All queries equi-probable
  - Query Cost $\propto$ number of rows examined
  - No indexes

Solutions:

- Optimization problem is NP-hard
  - Reduction from Set-Cover problem
    - Given a set X of n elements, a family F of subsets of X that cover X, what is the smallest number of subsets whose union is X ?
- Therefore, heuristic-based approximate solutions are the only hope
  - greedy algorithm discussed in this paper

Greedy Algorithm

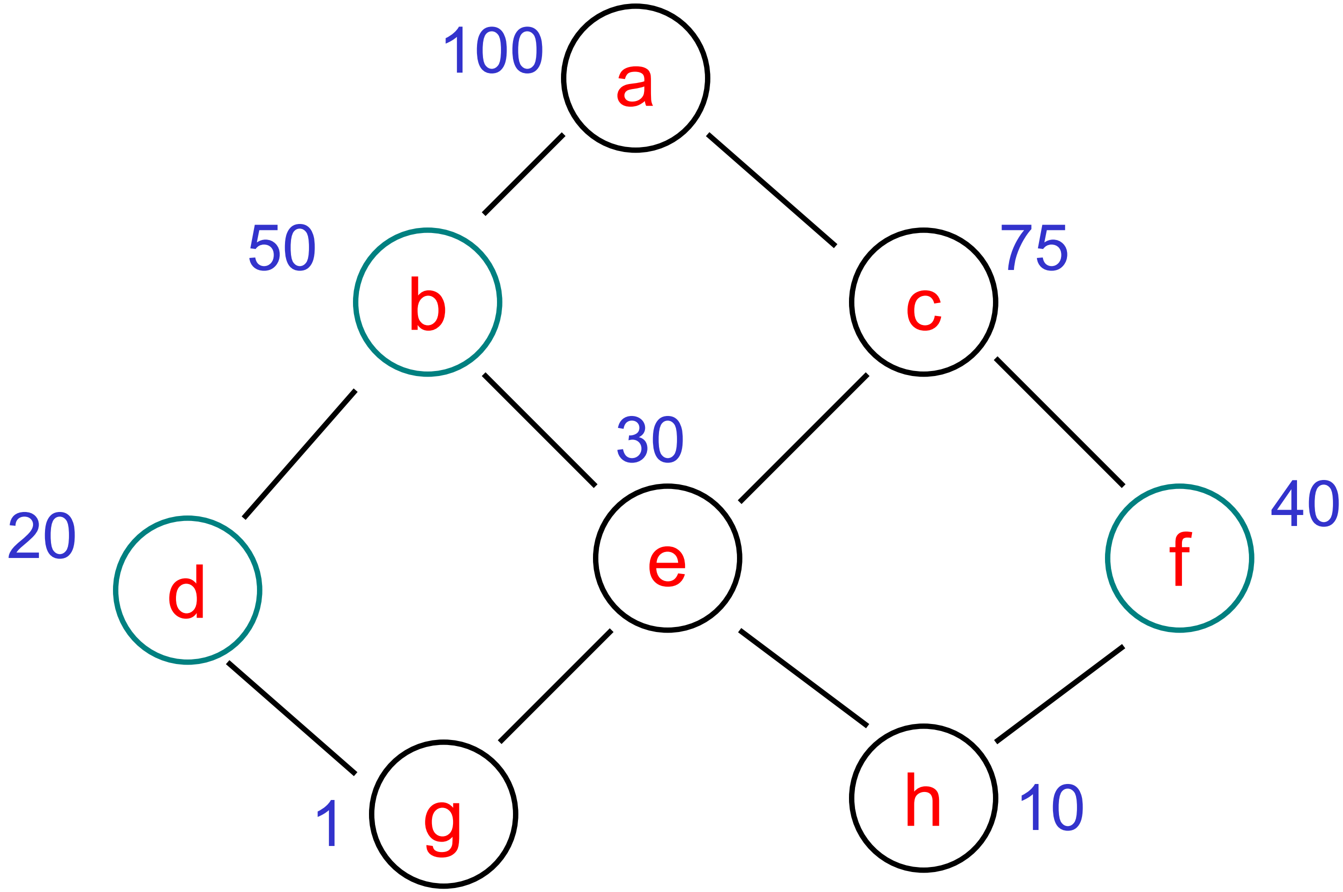
- Given view lattice V, number of (interior) views k, and result to be stored in S

- S = {Top view}
  for i = 1 to k do
  do a full traversal of V – S
  select view $v \notin S$ such that
  Benefit (v,S) is maximized
  $S = S \cup \{v\}$

Benefit Computation B(v,S)

- For each $w \preccurlyeq v$ (w derivable from v)
  - let u be the view of least cost in S such that $w \preccurlyeq u$.
    - since top view is in S, there must be at least one such view in S
  - if $C(v) < C(u)$, then $B_w = C(u) – C(v)$
    - Benefit to w of including v in set S
    - i.e. B_w = C(current_parent) – C(new_candidate_parent)
  - otherwise, $B_w = 0$
- Then, $B(v,S) = \sum_{w \preccurlyeq v} B_w$
    - overall benefit to all descendants,including itself, of v

Example 4.1



b,d,e,g,h: 100 (a) → 50 (b)

c,f: 100 (a) → 75 (c)
(e,g,h retain b as cheapest parent)

	Choice 1	*Cho*[illegible]	*Choice 3*
b	50 x 5 = 250		
c	25 x 5 = 125	25 x 2 =50	25 x 1 = 25
d	80 x 2 = 160	30 x 2 = 60	30 x 2 = 60
e	70 x 3 = 210	20 x 3 = 60	2 x 20 + 10 =50
f	60 x 2 = 120	60 + 10 = 70	
g	99 x 1 = 99	49 [illegible] 1 = 49	49 x [illegible] = 49
h	90 x 1 = 90	40 x [illegible] = 40	30 x [illegible]

f,h: 100 (a) → 40 (f)

f: 100 (a) → 40 (f)
h: 50 (b) → 40 (f)

e,g: 50 (b) → 30 (e)
h: 40 (f) → 30 (e)

Figure 8: Benefits of possible choices at each round

View Choices (k=3)

- Figure 8 in paper
- The greedy selection is b, f and d
- Cost reduces from 800 (100 * 8) to 420 which coincides with the optimal

TPC-D database

- Figure 11 gives a visual example of tradeoff
- After picking first five views (cp,ns,nt,c,p), almost the minimum possible total time, while total space is hardly more than the mandatory space used for just the top view.

Time-Space tradeoff

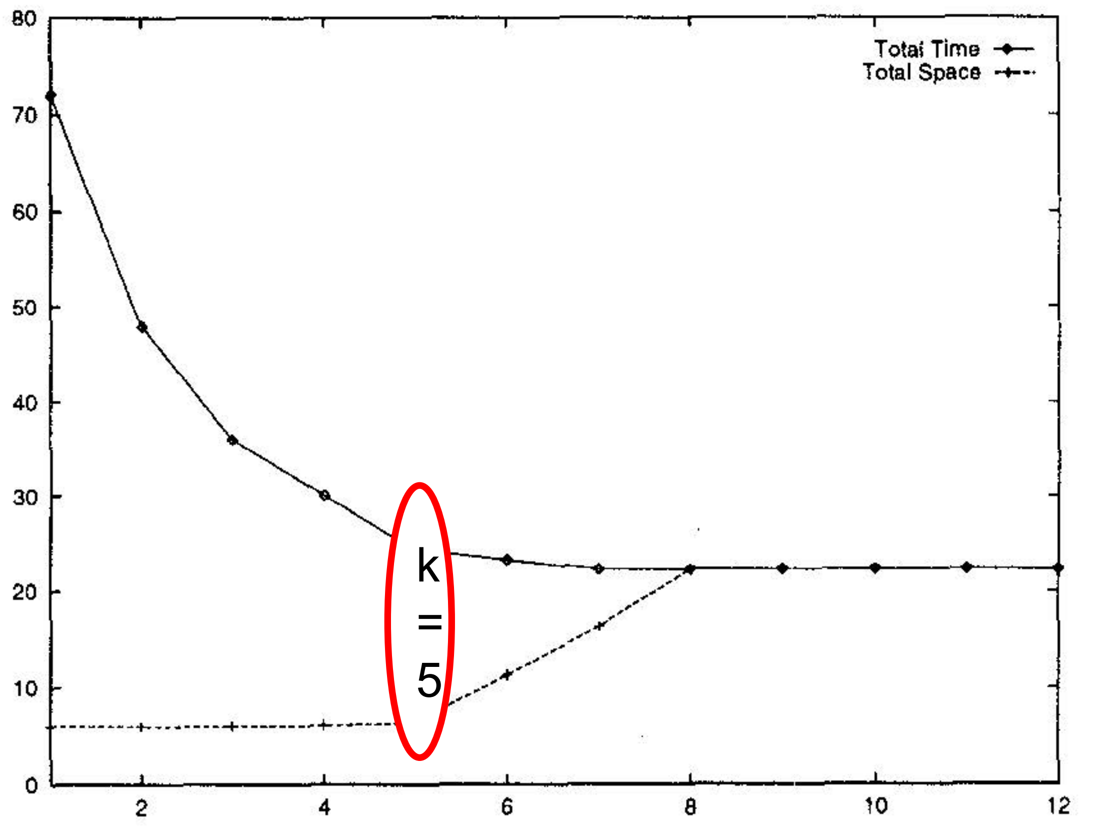


Figure 11: Time and Space versus number of views selected by the greedy algorithm

Performance Profile

- $B_{greedy} / B_{opt} \geq 1 - ((k - 1)/k)^k$
  - $k = 2$, ratio is 0.75
  - $k \rightarrow \infty$, ratio is $1 - 1/e = 0.63$
- Tight bound ! (Figure 9)
- No better algorithm possible!
  - problem closed ? no, randomized algorithms possible
- Special cases
  - Close to optimal if first view delivers most of the benefit
  - Equal to optimal if the benefit of each successive view is the same

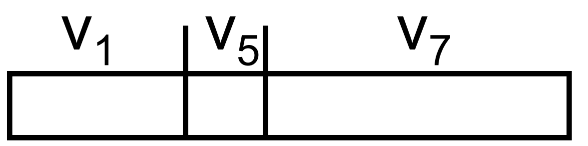
Proof

- Let $v_1, v_2, \ldots, v_k$ be the views chosen in sequence by the greedy algorithm.
- Let a_i be the benefit achieved by choosing v_i (w.r.t. $v_1, \ldots, v_{i-1}$)
- Similarly, let $w_1, w_2, \ldots, w_k$ be the views chosen by optimal, and b_i be the benefit achieved by choosing w_i (w.r.t. $w_1, \ldots, w_{i-1}$)

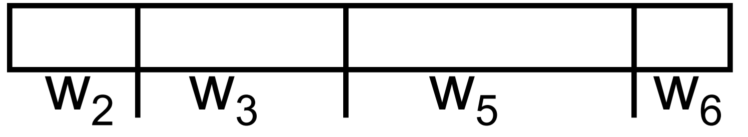
- Need to put an upper bound on the b's in terms of the a's

Proof

- Partition the improvement to an arbitrary view u effected by the v's and by the w's.
  - e.g. for view g, cost improved from 100 to 20. 50 came from b and 30 from d.
- Greedy



- Optimal



- Assign contribution of w_i's to v_j's : e.g., contribution of w_2 is wholly assigned to v_1; w_3 is divided among v_1, v_5, v_7; w_6 is not assigned;

Proof (contd)

- Define x_{ij} to be the sum over all views u in the lattice of the amount of the benefit b_i (from w_i) that is assigned to v_j .
- Then,
  - $\sum_i x_{ij} \leq a_j$ (total attribution cannot exceed complete value)

  Also
  - $\forall_i \quad b_i \leq a_1$ (o.w. w_i would have been chosen instead of v_1 by greedy algorithm)
  - $\forall_i \quad b_i - x_{i1} \leq a_2$ (benefit of w_i minus that already assigned to v_1)
  - …
  - $\forall_i \quad b_i - x_{i1} - x_{i2} - \ldots - x_{i,j-1} \leq a_j$

Proof (contd)

- Summing each equation over i, and with the constraints that $\sum_i b_i = B$, $\sum_i a_i = A$, $\sum_i x_{ij} \leq a_j$, we get
  - $B \leq ka_1$
  - $B \leq ka_2 + a_1$
  - $B \leq ka_3 + a_1 + a_2$
  - ...
  - $B \leq ka_k + a_1 + a_2 + \ldots + a_{k-1}$
- The bounds give maximum value of B when all right sides are equal. That is $ka_{i+1} - (k-1)\, a_i = 0$

Proof (contd)

- Therefore, $a_i = (k/k\text{-}1)\ a_{i+1}$
- For these values of a's,

$$A = \sum_{i=0 \text{ to } k-1} (k/k\text{-}1)^i a_k$$

and from first (or any) inequality

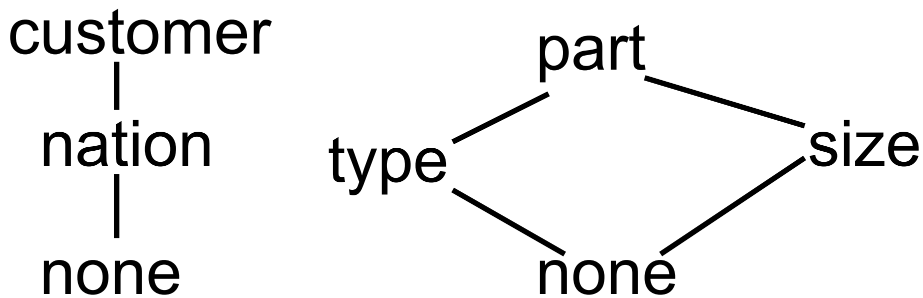
$$B \leq k\ (k/k\text{-}1)^{k-1} a_k$$

- Therefore, $A/B \geq 1 - ((k\text{-}1)/k)^k$

$$\geq 1 - 1/e \text{ as } k \rightarrow \infty$$

Dimension hierarchies

- Each dimension has a hierarchy



- Equivalent to “multiplying” lattices
  – Example Figure 4: beautiful picture !

Hierarchy

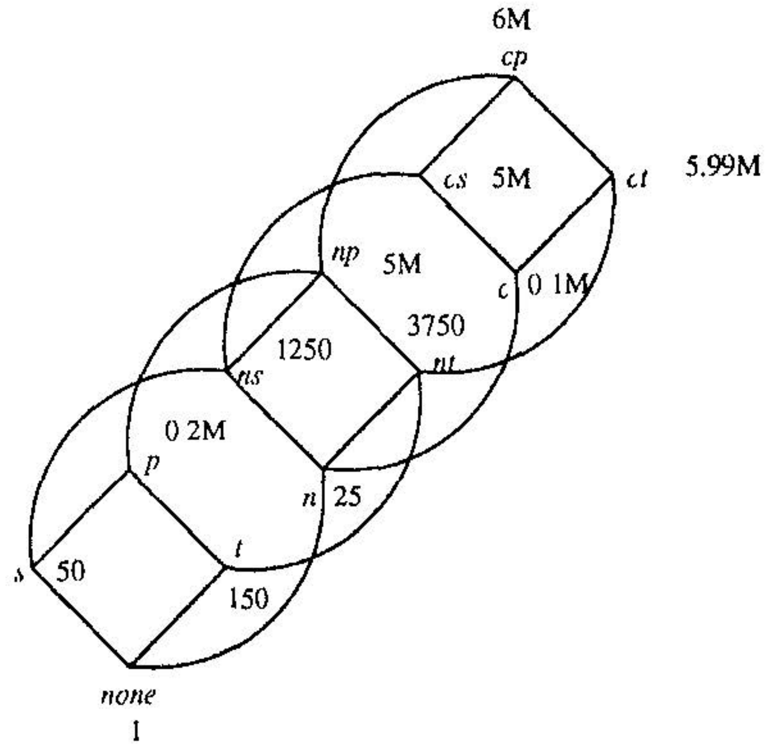


Figure 4: Combining two hierarchical dimensions

Alternative Problem Formulation

- Total Space is fixed, not number of views
- Means that Benefit per unit space needs to be computed.
- Performance guarantees still remain the same (ignoring boundary condition effects)

END CUBE MATERIALIZATION

E0 261