RELATIONAL DBMS EXTENSIONS FOR DW

- SQL extensions
- Index and storage structures
- Star query physical plans
- Materialized views

MATERIALIZED VIEWS

The **views** in relational DBMS: derived relation defined in terms of base (stored) relations.

```
CREATE VIEW TotalSalesByStore AS
SELECT      Store, Product, SUM(m)  AS Tm
FROM         Sales
GROUP BY   Store, Product;
```

Materialized views: A view can be materialized by storing the result of the view in the DB.

```
CREATE MATERIALIZED VIEW TotalSalesByStore AS
SELECT      Store, Product, SUM(m)  AS Tm
FROM         Sales
GROUP BY   Store, Product;
```

Standard/Oracle, in SQL Server named '**Indexed Views**'

WHY TO MATERIALIZE VIEWS?

Sales(Product, Store, Date, m) with 1M facts but only 1K distinct Stores

```
SELECT    Store, Product, SUM(m) AS Tm
FROM      Sales
GROUP BY  Store, Product;
```

Let us materialize the result as **V**

Consider the query Q

```
SELECT    Store, SUM(m) AS Tm
FROM      Sales -- scan of 1M rows
GROUP BY  Store;
```

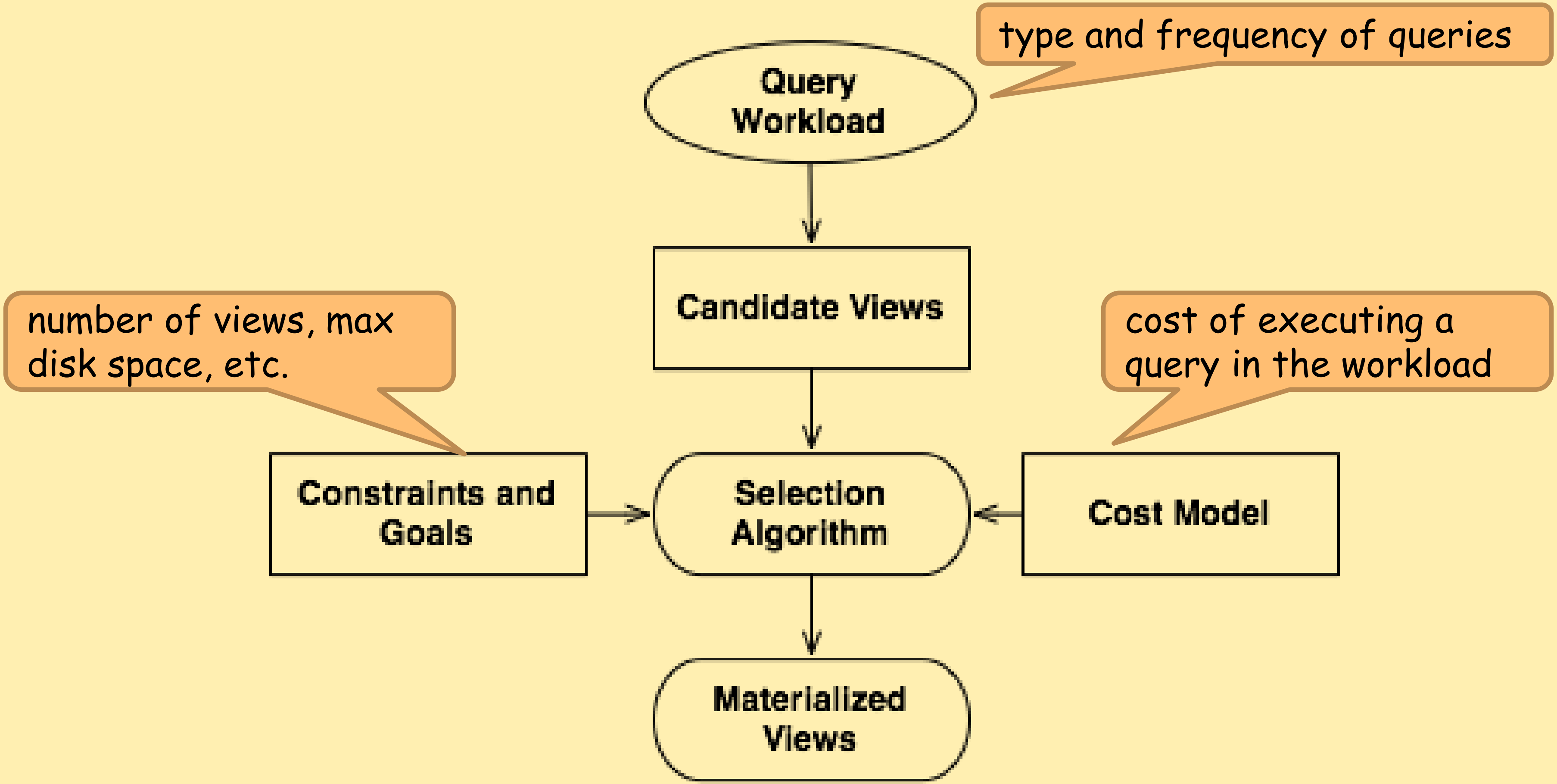
The query Q can be rewritten as the more efficient

```
SELECT    Store, SUM(Tm) AS Tm
FROM      V -- scan of 1K rows
GROUP BY  Store;
```

PROBLEMS

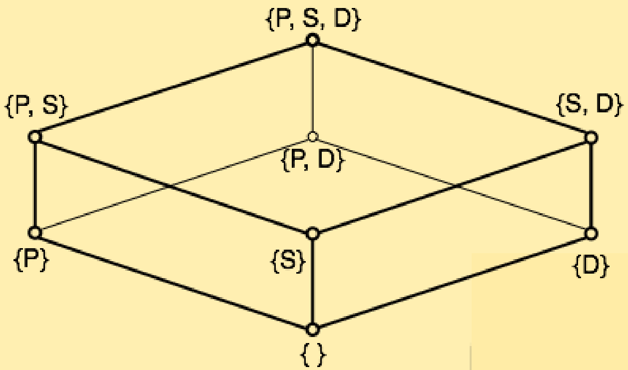
- Given a query workload Q (type and frequency of queries), how to select the views to materialized?
- How the system rewrites a query to use materialized views?
  - We'll see in future lessons
- How to update materialized views if the database is updated?
  - Incremental view maintainance: overhead to updates/inserts
  - Recomputation: applies to DW (better than incremental view maintainance):
    1. Drop materialized views
    2. ETL
    3. Re-create materialized views

APPROACH FOR SELECTION OF VIEWS TO MATERIALIZE



ASSUMPTIONS AND AN EXAMPLE OF THE DW LATTICE

The fact table **F** has **n** dimensions, without attributes, and a measure **m**



P	S	D	M
0	1	1	40
1	1	1	10
2	3	1	15
4	5	1	30
6	5	2	20

FACT TABLE **Sales**

$_X\gamma_{SUM(m)\ AS\ m}$ **(F)**

The **candidate views v** are **the possible DW lattice nodes**, different from the root **F**, defined as:

$_X\gamma_{SUM(m)\ AS\ m}$ **(F)**.

Let **g(v)** = **X** the grouping attributes of the cuboid **v**.

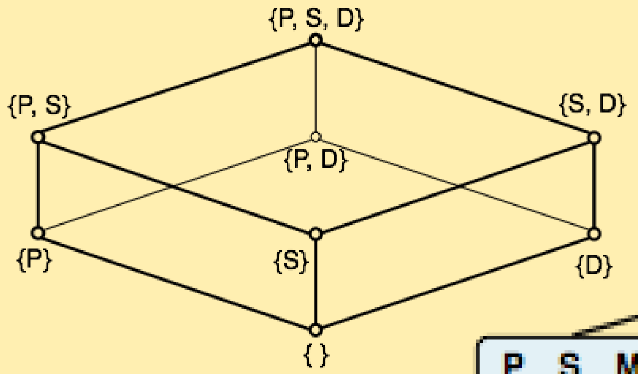
SELECT S, D, SUM(m) **AS** m
FROM Sales
GROUP BY S, D;
g(v) = {S,D}

SELECT D, SUM(m) **AS** m
FROM Sales
GROUP BY D;
g(v) = {D}

SELECT SUM(m) **AS** m
FROM Sales;
g(v) = {}

ASSUMPTIONS AND AN EXAMPLE OF THE DW LATTICE

The fact table **F** has **n** dimensions, without attributes, and a measure **m**



The **candidate views v** are **the possible DW lattice nodes**, different from the root **F**, defined as:

$_{X}\gamma_{\mathrm{SUM}(m)\ \mathrm{AS}\ m}(\mathbf{F})$.

Let **g(v) = X** the grouping attributes of the cuboid **v**.

P	S	D	M
0	1	1	40
1	1	1	10
2	3	1	15
4	5	1	30
6	5	2	20

FACT TABLE **Sales**

$_{X}\gamma_{\mathrm{SUM}(m)\ \mathrm{AS}\ m}(\mathbf{F})$

P	S	M
0	1	40
1	1	10
2	3	15
4	5	30
6	5	20

P	D	M
0	1	40
1	1	10
2	1	15
4	1	30
6	2	20

S	D	M
1	1	50
3	1	15
5	1	30
5	2	20

```
SELECT    S, D, SUM(m) AS m
FROM      Sales
GROUP BY  S, D;
```
g(v) = {S,D}

P	M
0	40
1	10
2	15
4	30
6	20

S	M
1	50
3	15
5	50

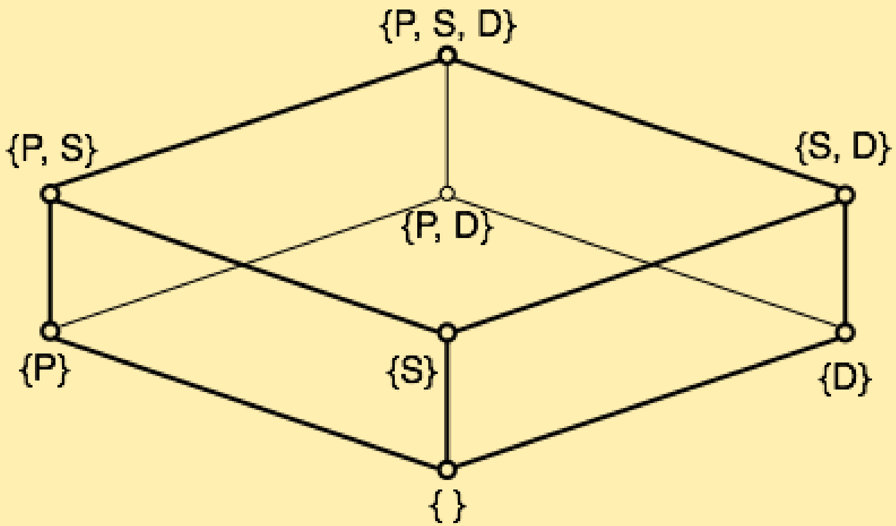
D	M
1	95
2	20

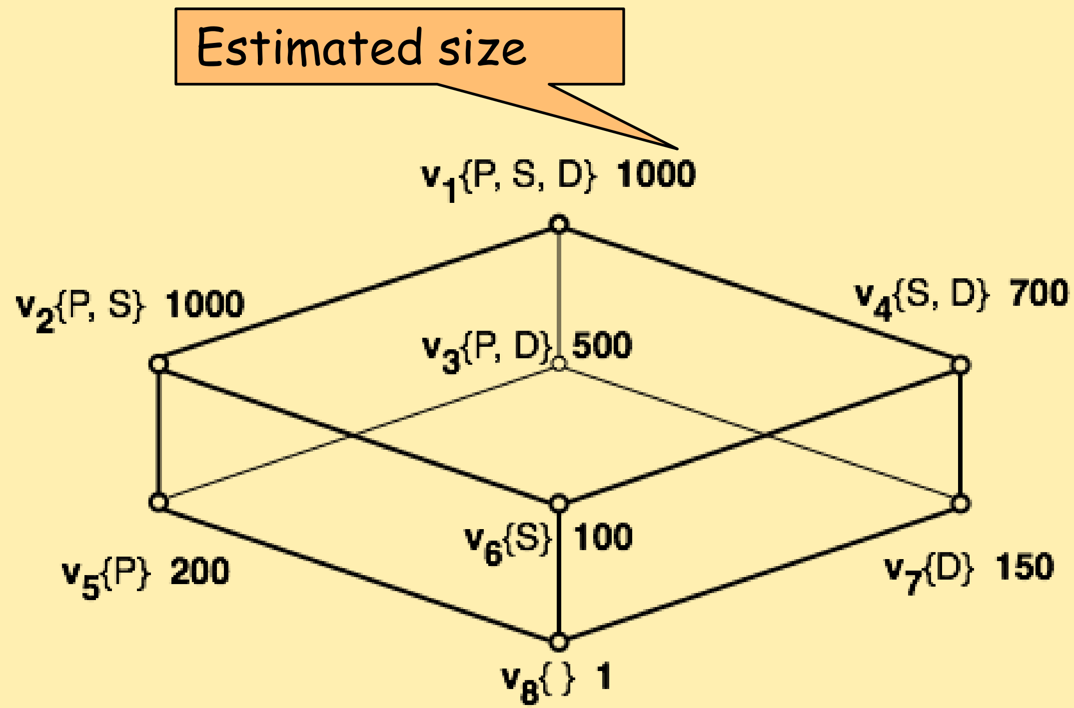
```
SELECT    D, SUM(m) AS m
FROM      Sales
GROUP BY  D;
```
g(v) = {D}

M
115

```
SELECT  SUM(m) AS m
FROM    Sales;
```
g(v) = {}

FROM THE LATTICE OF CUBOIDS TO THE LATTICE OF VIEWS

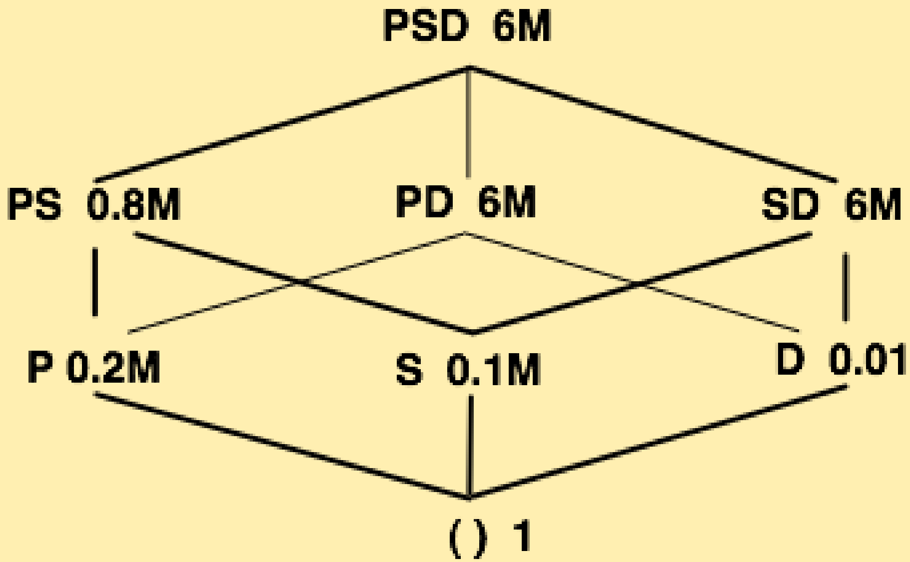




How is the size of a view estimated?

Analytic, sampling, Pareto approaches (see lecture notes)

WHY VIEWS ARE MATERIALIZED?



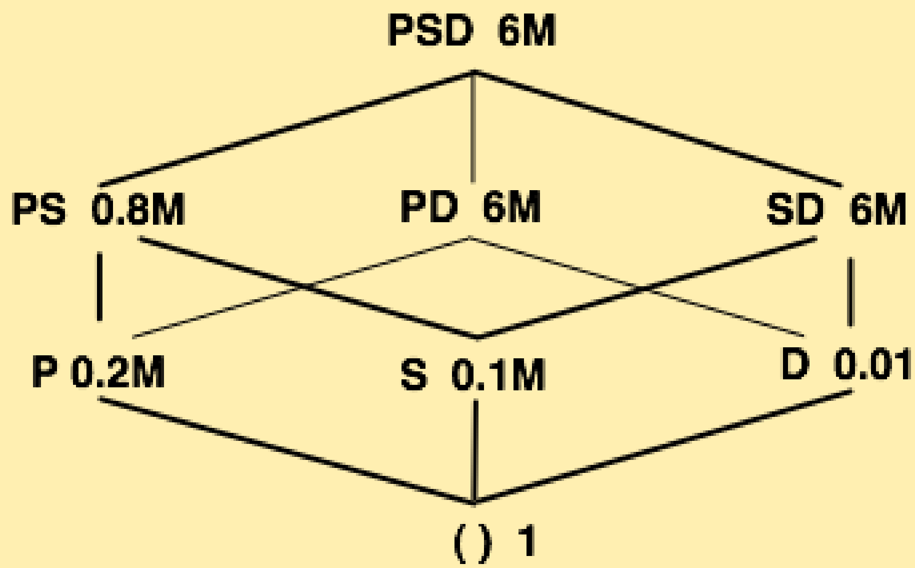
Business question: Total sales **by Product**.

Case 1: data **(PSD)** = 6M

Case 2: if **(PS)** is materialized = 0.8M

Case 3: if **(P)** is materialized = 0.2M

WHY NOT TO MATERIALIZE ALL VIEWS



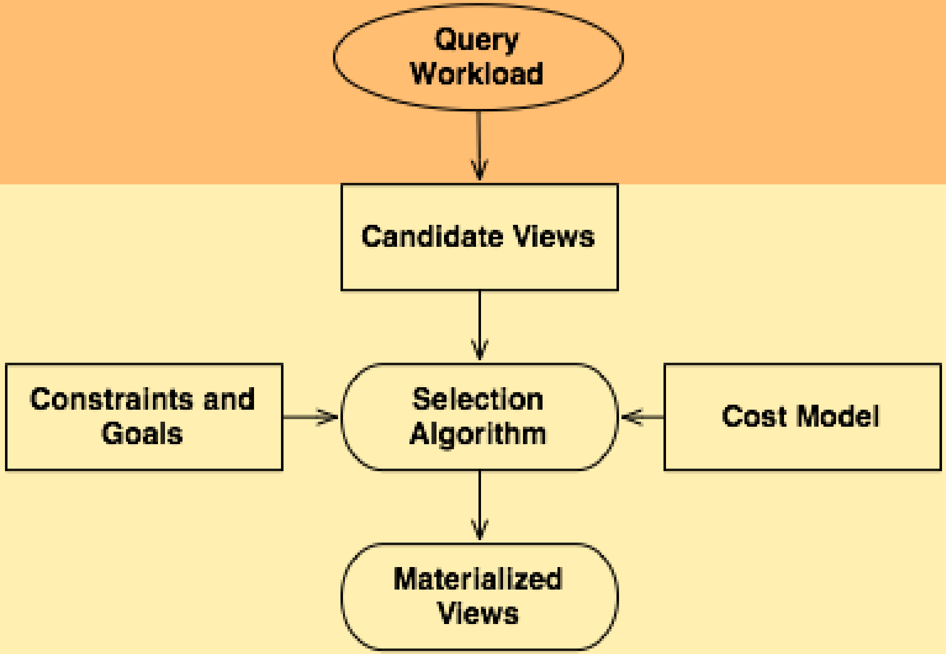
Full materialization: ~19M record

Partial materialization:

- include: **PSD**, the DW
- useless: **PD**, **SD**

total: ~ 7M

ASSUMPTIONS



- The **query workload** (used to evaluate quality of a set of materialized views) is the set of queries in the DW lattice of views.
- The **candidate views v** are **the possible DW lattice of views** different from the root **F (which is already materialized)**, defined as: $_{X}\gamma_{SUM(m)\ AS\ m}$ **(F)**.
- The execution **cost** of a query **q** using the view **v** is **|v|**, the number of records of **v**, which is assumed to be known (estimated)
- **Notice:** q can be rewritten using v (written: **q ≤ v**) iff **g(q) ⊆ g(v) ie g(q) is a descendant of g(v) in the lattice**

THE SELECTION OF MATERIALIZED VIEWS

Let Q be the query workload (Q = { queries in the lattice of views }).

Let M be a set of materialized views.

Let C(q, M) the execution cost of **q** $\in$ Q using **the best view (wrt q)** from M.

The goal is to select the set of views M which minimizes the overall execution cost of the query workload Q, i.e., the quantity:

$$\tau(M) = \sum_{q \in Q} C(q, M)$$

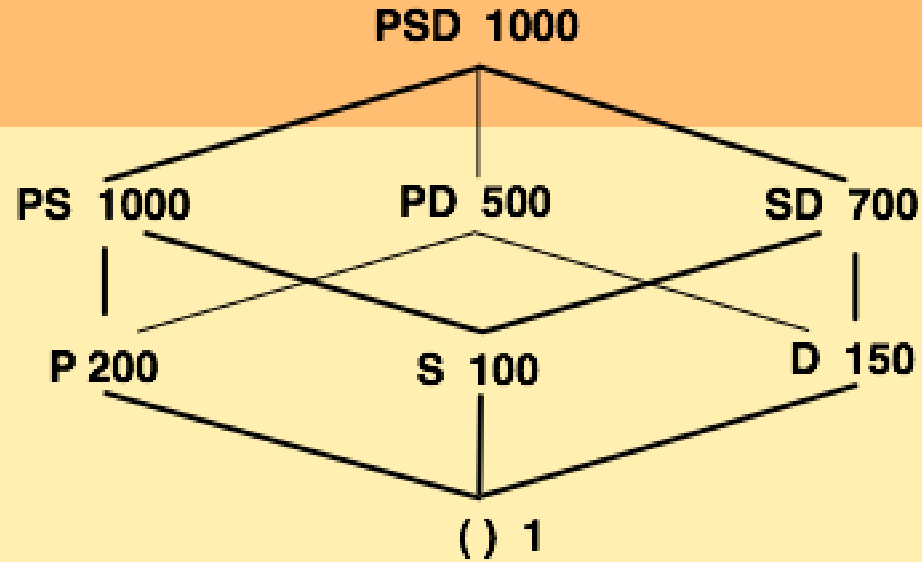
The optimization problem has been proved to be NP-complete. An approximate **greedy algorithm** has been proposed:

Initially M = { F } only the fact table is materialized.

Each iteration calculates the **benefit** of the remaining candidate views and selects for materialization the one with the maximum benefit.

BENEFIT OF A VIEW

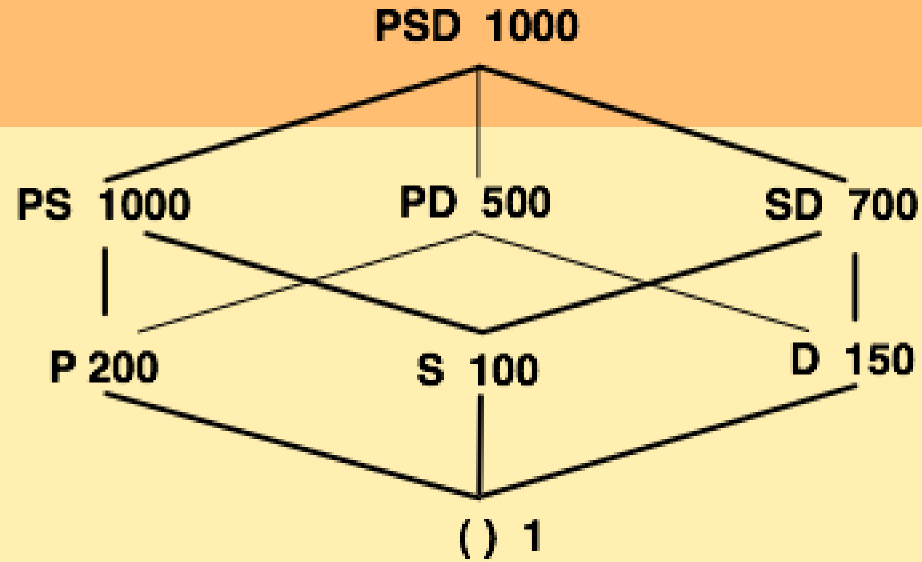
$\tau(M) = \Sigma_{q \in Q} C(q, M)$



Informally, the **benefit** of a view not yet materialized is the produced reduction of the execution cost of query workload.

Let M be a set of materialized views. The benefit B(v, M) of a view **v** $\notin$ M is defined as: B(v, M) = $\tau(M) - \tau(M \cup \{v\})$

BENEFIT OF A VIEW



$\tau(M) = \Sigma_{q \in Q} C(q, M)$

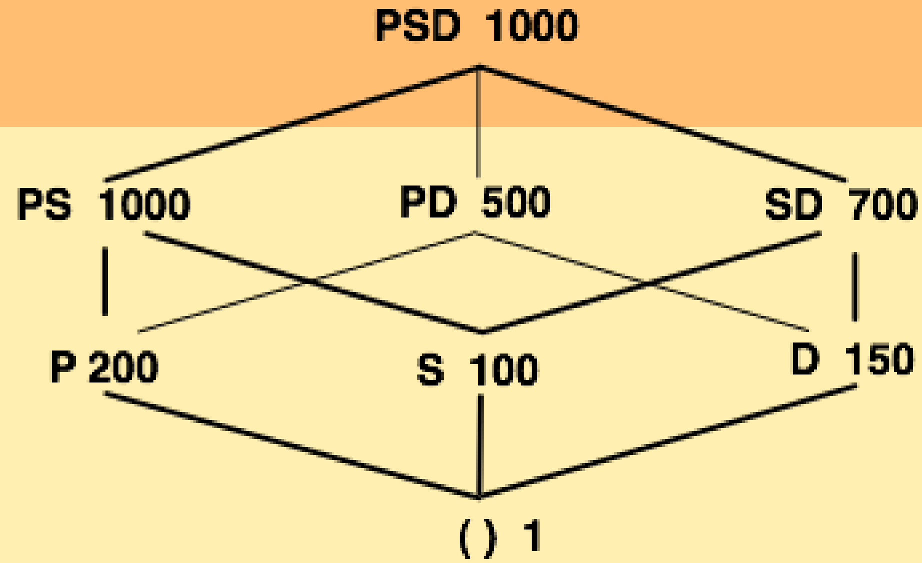
$B(v, M) = \tau(M) - \tau(M \cup \{v\})$

$= \Sigma_{q \in Q} (C(q, M) - C(q, M \cup \{v\}))$

Consider **q** such that **q** $\leq$ **v** does **not** hold:

- $C(q, M \cup \{v\}) = C(q, M)$, hence benefit for q is zero.

BENEFIT OF A VIEW



$\tau(M) = \Sigma_{q \in Q} C(q, M)$

$B(v, M) = \tau(M) - \tau(M \cup \{v\})$

$= \Sigma_{q \leq v} (C(q, M) - C(q, M \cup \{v\}))$

Consider each $\mathbf{q} \leq \mathbf{v}$:

a) Let $\mathbf{u}_q$ be the view with **least cost** in M such that $\mathbf{q} \leq u_q$, i.e., $|\mathbf{u}_q| = C(\mathbf{q}, M)$

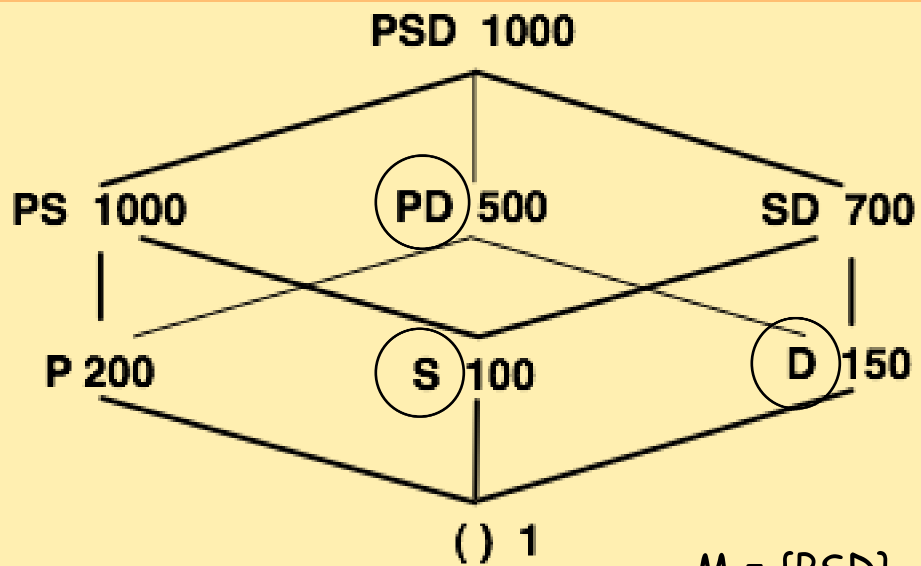
b) $C(\mathbf{q}, M \cup \{\mathbf{v}\}) = \min\{ |\mathbf{v}|, |\mathbf{u}_q| \}$ because either **v** is better than $\mathbf{u}_q$ or not.

- **If** $|\mathbf{v}| < |\mathbf{u}_q|$, **then** $C(\mathbf{q}, M) - C(\mathbf{q}, M \cup \{\mathbf{v}\}) = |\mathbf{u}_q| - |\mathbf{v}|$, **otherwise** it is 0.
- In general, $C(\mathbf{q}, M) - C(\mathbf{q}, M \cup \{\mathbf{v}\}) = \max\{0, |\mathbf{u}_q| - |\mathbf{v}|\}$

In summary: $B(v, M) = \Sigma_{q \leq v} \max\{0, |\mathbf{u}_q| - |\mathbf{v}|\}$

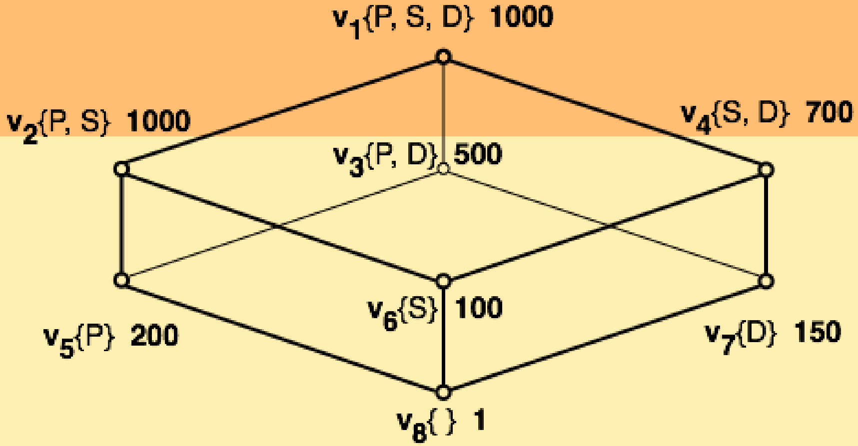
EXAMPLE

$$B(v, M) = \sum_{q \leq v} \max\{0, |u_q| - |v|\}$$



	M = {PSD}	M = {PSD, PD}	M = {PSD, PD, S}
	First Choice	**Second Choice**	**Third Choice**
PS	0	0	0
PD	**500 × 4 = 2000**		
SD	300 × 4 = 1200	300 × 2 = 600	300
P	800 × 2 = 1600	300 × 2 = 600	300
S	900 × 2 = 1800	**900 + 400 = 1300**	
D	850 × 2 = 1700	350 × 2 = 700	**350**
( )	999	499	99

Solution when selecting k=3 materialized views **M = {PSD, PD, S, D}**



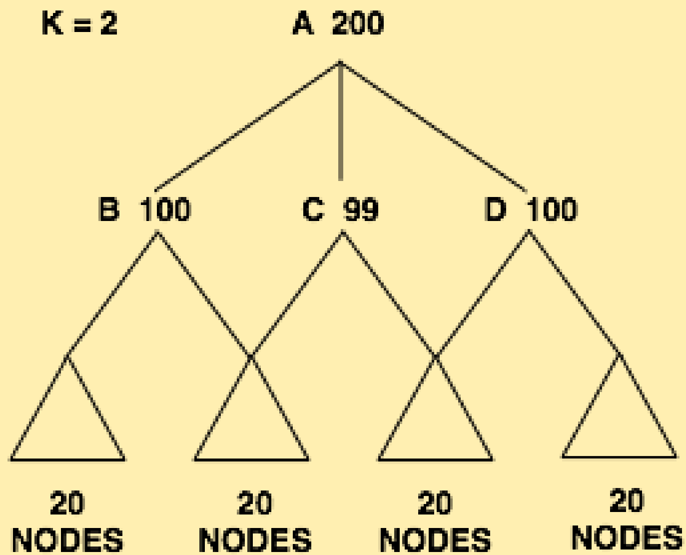
Constraint:

There are only **k** candidate views to materialize, different from the top view

Algorithm HRU(k)

% Let v_1 be the lattice root
$M = \{v_1\}$;
$N = V - M$;
for $i = 1$ **to** k
{ $v =$ the view in N, such that $B(v, M)$ the maximum;
 $M = M \cup \{v\}$;
 $N = N - \{v\}$ };
return M ;

HRU DOES NOT FIND THE BEST SOLUTION



First Choice	Second Choice
B = 100+ 40*100 = 4100	**B** = 100+20*100 = 2100
C = 101 + 40*101 = 4141	
D = 100 + 40*100 = 4100	**D** = 100+20*100 = 2100

Greedy

M = {**A**, C, B}

B_{greedy} = 6241

Optimal Choice

Pick B and D

M = {A, **B**, **D**}

B_{opt} = (100+100*40)*2 = 8200

B_{greedy}/B_{opt} = 0.76

PERFORMANCE

- In general, the algorithm does not find the optimal solution, but the authors have shown that it provides good results and the following interesting properties hold:

For each lattice, let B_{greedy} be the **benefit** of k views selected by the algorithm **greedy** and B_{opt} be the **benefit** of the optimum choice of k views, then B_{greedy} can never be less than 0,63 * B_{opt}.

OTHER ALGORITHMS: PGA

HRU has a time complexity $O(km^2)$, where **k** is the **number of views selected** and **m** the **number of lattice views**. This is **polinomial** with the number **m** of views, but **exponential** with the number of **dimensions n** $O(km^2) = O(k2^{2n})$

The **exponential complexity** of HRU depends on two choices:

At each iteration, it considers all remaining views on the entire lattice that have not yet materialized.

At each iteration, it considers for each **v** all its descendants.

An algorithm with **polynomial time complexity on the number of dimensions** is the **Polynomial Greedy Algorithm, PGA** (see lecture notes).

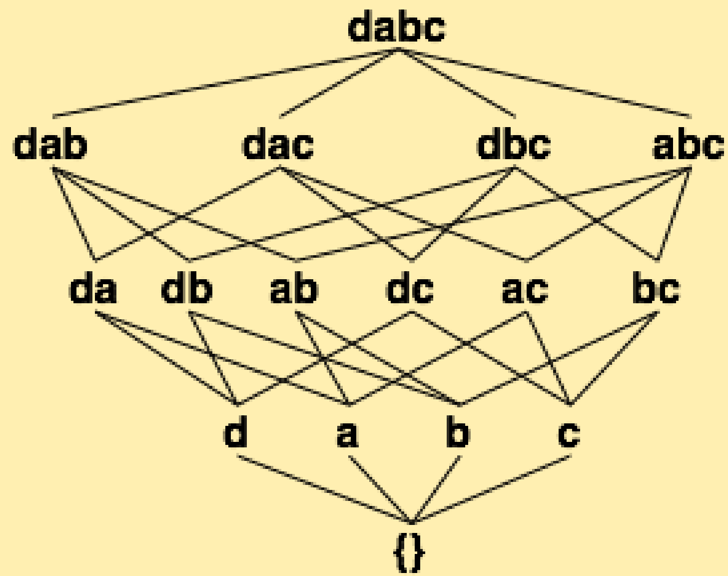
OTHER ALGORITHMS

- Queries of the workload are not equally likely.

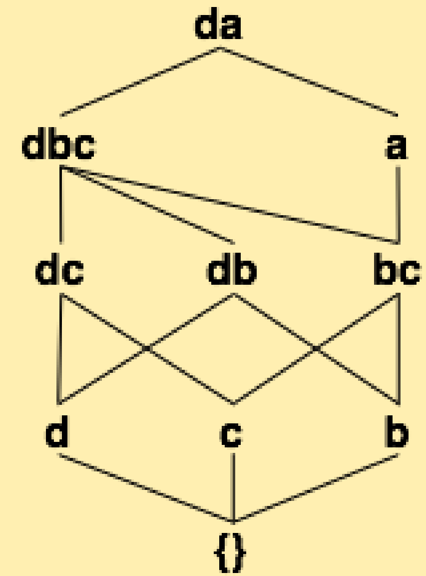
  Algorithm for a particular workload

- Instead of having a limit on the number of views **k** that can be materialized, there is an upper bound on the total storage space **S** that the set of materialized views **M** can occupy.

  Algorithm **PBS (Pick By Size)**



Hypothetic



Simplified

Hypothetic: Consider the join of F with all the dimensions.

It can be simplified:
- The root is F
- If **a -> b** a view with **a** has the same groups of one on **ab**.

WHAT ABOUT MORE COMPLEX QUERIES?

```
SELECT      <Grouping attributes>, SUM(m) AS m
FROM        <Fact Table>
WHERE       <Condition on some attributes>
GROUP BY    <Grouping attributes>;
```

q defines a slice of a cuboid, i.e., q = ${}_X\gamma_{SUM(m)\ AS\ m}(\sigma_C(F))$.

Eg., q = ${}_P\gamma_{SUM(m)\ AS\ m}(\sigma_{S=1}(F))$

q « **v** for a candidate view **v** = ${}_Z\gamma_{SUM(m)\ AS\ m}(\mathbf{F})$ when $X \cup var(C) \subseteq Z$.

How? Eg., v = ${}_{P,S}\gamma_{SUM(m)\ AS\ m}(F) \rightarrow q = {}_P\gamma_{SUM(m)\ AS\ m}(\sigma_{S=1}(v))$

MATERIALIZED VIEW SELECTION TECHNIQUES