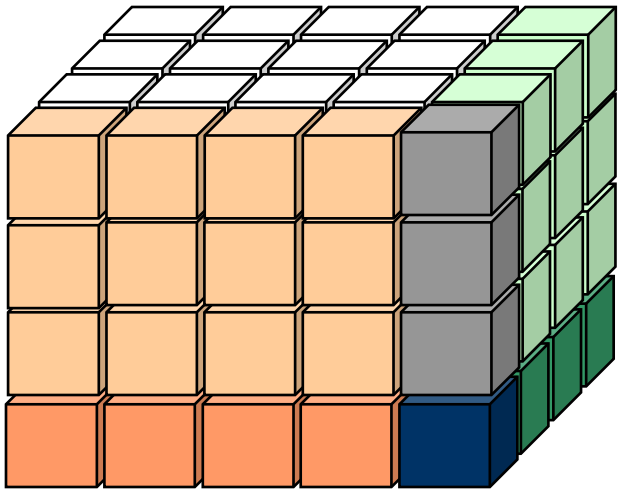
CPSC 304 – Administrative notes November 20 and 21, 2024

- Office hours for me on Monday the 25th changed to 10am-11am (sorry!)
- Project:
  - Milestone 4: Project implementation – due November 29
    - You cannot change your code after this point!
  - Milestone 5: Group demo – week of December 2
  - Milestone 6: Individual Assessment – Due November 29
- Tutorials: basically project group work time/office hours
  - No tutorials the last week of class (so the TAs have more time to grade/do demos)
- Final exam: December 16 at 12pm! Osborne A

Now where were we…

- We'd been talking about data warehousing
- We'd gotten comfortable with the idea of a data cube…



Motivating discussion

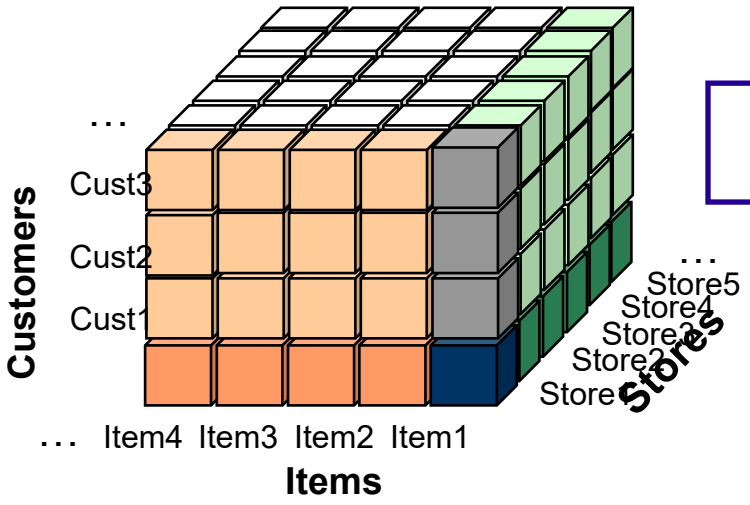
- Could we find out if younger people like bubble tea vs. older people
- I polled all of you, and pretty much everyone liked bubble tea (I'd say > 95%)
- As promised, I polled faculty, staff and grad students about bubble tea. Results:

	Like Bubble Tea	Do not Like bubble tea
(probably over 21) & < 30	9	3
30+*	27	5

* At least three people in the over 30 category wanted a new option for "what is bubble tea?"

Great! Now let's talk about how that cube is stored a bit…

Representing a Cube in a Two-Dimensional Table



storeID	itemID	custID	Sum
store1	item1	cust1	10
store1	item1	Null	70
store1	Null	cust1	145
store1	Null	Null	325
Null	item1	cust1	10
Null	item1	Null	135
Null	Null	cust1	670
Null	Null	Null	3350

. . . . . .

Add to the original cube: faces, edges, and corners … which are represented in the 2-D table using NULLs.

Finding Answers Quickly

- Large datasets and complex queries mean that we'll need improved querying capabilities
  - Materializing views
  - Finding Top N Queries
  - Using online aggregation

Queries over Views

Reminder: use a view as follows:

View

```
Create view TshirtSales AS
SELECT category, county, age, sales
FROM    AllSales F, Store S, Customer C, Item I
WHERE F.storeID = S.storeID AND F.custID = C.custID AND
              F.itemID = I.itemID ANDcategory  = 'Tshirt'
```

Query

```
SELECT category, county, age, SUM(sales)
From      TshirtSales
GROUP BY category, county, age;
```

Materializing Views

- Decision support activities require queries against complex view definitions to be computed quickly.
- Sophisticated optimization and evaluation techniques are not enough since OLAP queries are typically aggregate queries that require joining huge tables.
- Pre-computation is essential for interactive response times.
- A view whose tuples are stored in the database is said to be materialized.

Issues in View Materialization: Which views should we materialize?

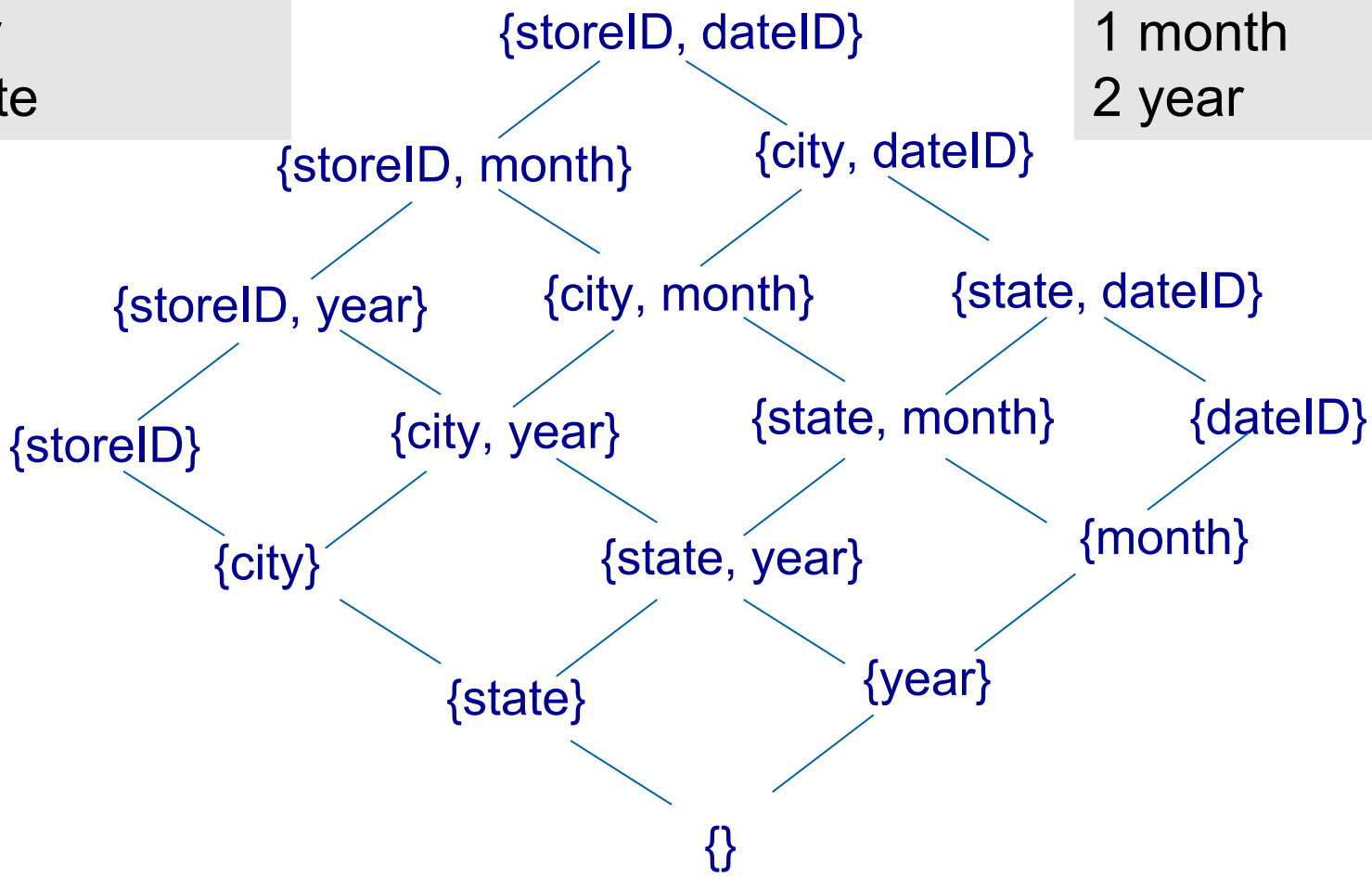
- Which views should we materialize?
- Based on size estimates for the views, suppose there is space for *k* views to be materialized. Which ones should we materialize?
- The goal is to materialize a small set of carefully chosen views to answer most of the queries quickly.
- **Fact:** Selecting *k* views to materialize such that the average time taken to evaluate all views of a lattice is minimized is a NP-hard problem.

The Exponential Explosion of Views

- Assume that we have two dimensions, each with a hierarchy

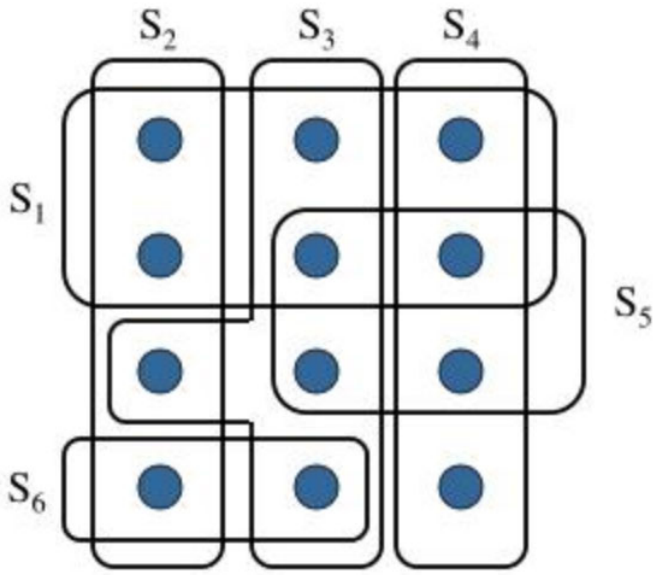
Store dimension
0 storeID
1 city
2 state

Calendar dimension
0 dateID
1 month
2 year



Maximum Coverage Problem Example

Given 12 ground facts/elements, 6 subsets, and a value for *k*, find the *k* subsets (e.g., *k* = 3) that between them cover as many ground elements as possible.



Difference between a NP-hard problem and a problem that you solve efficiently (polynomial time) is like the difference between solving a Sudoku puzzle vs. checking whether a given solution is valid.

Maximum Coverage problem has a structure similar to finding the top *k* views. We can find approximately optimal solutions quickly for both.

CPSC 304 – December 1 Administrative notes

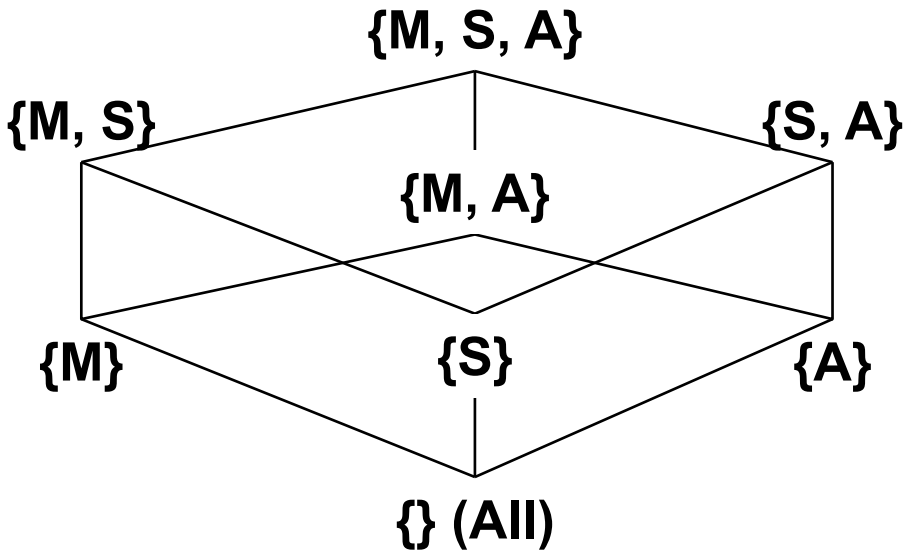
- Project Demos: November 28-December 2
- No tutorials this week
- **UNGRADED** In-class exercise released for data warehousing
- Final exam: Sunday, December 11 @3:30pm: Osborne A
  - Final exam office hours are listed on the office hour page. The page is still undergoing changes so check back regularly.

Now where were we...

- We'd been discussing data warehousing
- We'd said that it was usually visualized like a cube, and built using the cube operator
- But that cube is super big! We don't want to materialize all of it.
- How do we choose which parts to materialize?

Back to the student table: Student(snum,sname,major,standing,age)

- We can aggregate the student table based on major, standing, or age



Group by all 3 – any ordering gives you same # of tuples

```
SELECT major, standing, age, count(*)
FROM         student
GROUP BY major, standing, age
ORDER BY major, standing, age
```

MAJOR	ST	AGE	COUNT(*)
Accounting	JR	19	1
Animal Science	FR	18	1
Architecture	SR	22	1
Civil Engineering	SR	21	1
Computer Engineering	FR	18	1
Computer Engineering	SR	19	1
Computer Science	JR	18	1
Computer Science	JR	20	1
Computer Science	SO	17	1
Computer Science	SO	19	1
Economics	JR	20	1
Education	SR	21	1
Electrical Engineering	FR	17	2
English	SR	21	1
Finance	FR	18	2
History	SR	20	1
Kinesiology	SO	19	1
Law	JR	20	1
Mechanical Engineering	SO	19	1
Psychology	JR	20	1
Psychology	SO	18	1
Veterinary Medicine	SR	21	1

22 rows selected.

```
SELECT age, standing, major, count(*)
FROM student
GROUP BY age, standing, major
ORDER BY age, standing, major
```

AGE	ST	MAJOR	COUNT(*)
17	FR	Electrical Engineering	2
17	SO	Computer Science	1
18	FR	Animal Science	1
18	FR	Computer Engineering	1
18	FR	Finance	2
18	JR	Computer Science	1
18	SO	Psychology	1
19	JR	Accounting	1
19	SO	Computer Science	1
19	SO	Kinesiology	1
19	SO	Mechanical Engineering	1
19	SR	Computer Engineering	1
20	JR	Computer Science	1
20	JR	Economics	1
20	JR	Law	1
20	JR	Psychology	1
20	SR	History	1
21	SR	Civil Engineering	1
21	SR	Education	1
21	SR	English	1
21	SR	Veterinary Medicine	1
22	SR	Architecture	1

22 rows selected.

Group by 2 option 1: Major, Standing

```
SELECT major, standing, count(*)
FROM student
GROUP BY major, standing
ORDER BY major, standing

MAJOR                     ST   COUNT(*)
------------------------- -- ----------
Accounting                JR          1
Animal Science            FR          1
Architecture              SR          1
Civil Engineering         SR          1
Computer Engineering      FR          1
Computer Engineering      SR          1
Computer Science          JR          2
Computer Science          SO          2
Economics                 JR          1
Education                 SR          1
Electrical Engineering    FR          2
English                   SR          1
Finance                   FR          2
History                   SR          1
Kinesiology               SO          1
Law                       JR          1
Mechanical Engineering    SO          1
Psychology                JR          1
Psychology                SO          1
Veterinary Medicine       SR          1

20 rows selected.
```

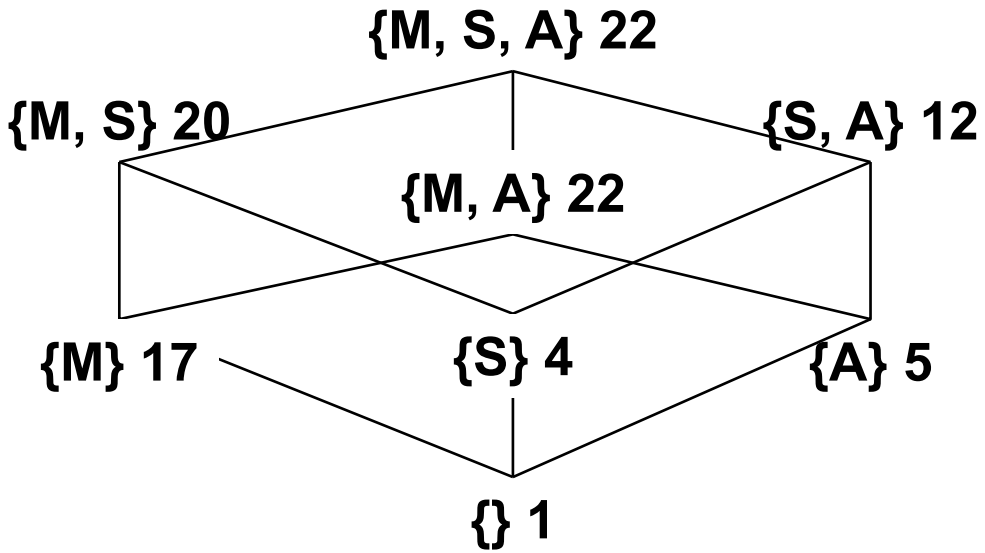
Group by 2 option 2: Standing, Age

```
SELECT standing, age, count(*)
FROM student
GROUP BY standing, age
ORDER BY standing, age

ST  AGE     COUNT(*)
-- ---------- ----------
FR   17            2
FR   18            4
JR   18            1
JR   19            1
JR   20            4
SO   17            1
SO   18            1
SO   19            3
SR   19            1
SR   20            1
SR   21            4
SR   22            1

12 rows selected.
```

Group by 2 option 3: Major, Age

```
SELECT major, age, count(*)
FROM student
GROUP BY major, age
ORDER BY major, age

MAJOR                        AGE        COUNT(*)
-------------------------- ---------- ----------
Accounting                 19                  1
Animal Science             18                  1
Architecture               22                  1
Civil Engineering          21                  1
Computer Engineering       18                  1
Computer Engineering       19                  1
Computer Science           17                  1
Computer Science           18                  1
Computer Science           19                  1
Computer Science           20                  1
Economics                  20                  1
Education                  21                  1
Electrical Engineering     17                  2
English                    21                  1
Finance                    18                  2
History                    20                  1
Kinesiology                19                  1
Law                        20                  1
Mechanical Engineering     19                  1
Psychology                 18                  1
Psychology                 20                  1
Veterinary Medicine        21                  1
22 rows selected.
```

In computing costs to query, we consider the # of tuples that you have to look at



The # is the # of tuples

Assuming all of the views were materialized, executing the query

```
SELECT standing, count(*)
FROM student
GROUP BY standing
```

would cost 4 because that's the cheapest way to access the necessary tuples

We can also use {Major, Standing} to compute {Standing} for a cost of 20

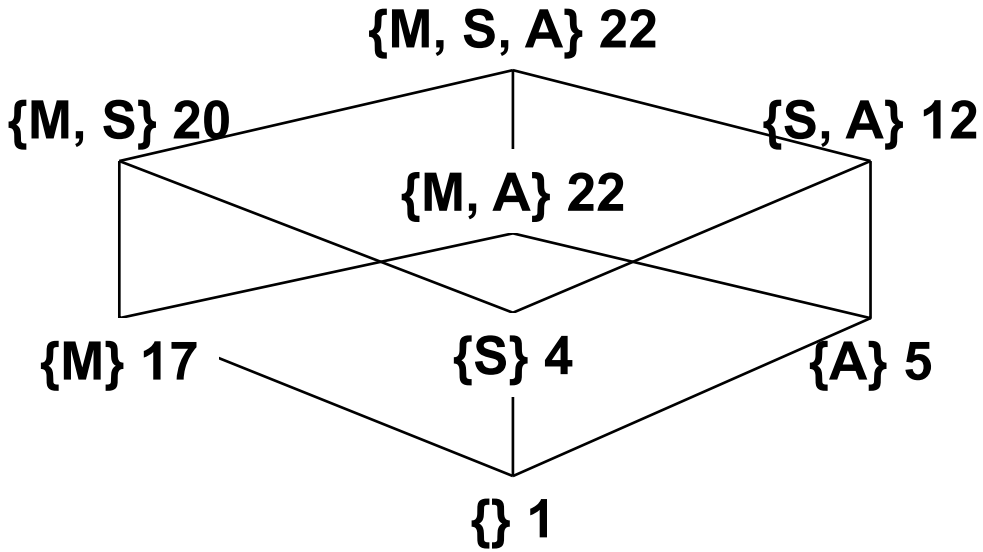
```
SELECT major, standing, count(*)
FROM student
GROUP BY major, standing
ORDER BY major, standing
```

```
SELECT standing, count(*)
FROM student
GROUP BY standing
ORDER BY standing
```

```
MAJOR                     ST   COUNT(*)
------------------------- -- ----------
Accounting                JR          1
Animal Science            FR          1
Architecture              SR          1
Civil Engineering         SR          1
Computer Engineering      FR          1
Computer Engineering      SR          1
Computer Science          JR          2
Computer Science          SO          2
Economics                 JR          1
Education                 SR          1
Electrical Engineering    FR          2
English                   SR          1
Finance                   FR          2
History                   SR          1
Kinesiology               SO          1
Law                       JR          1
Mechanical Engineering    SO          1
Psychology                JR          1
Psychology                SO          1
Veterinary Medicine       SR          1

20 rows selected.
```

```
ST   COUNT(*)
-- ----------
SR          7
SO          5
FR          6
JR          6
```

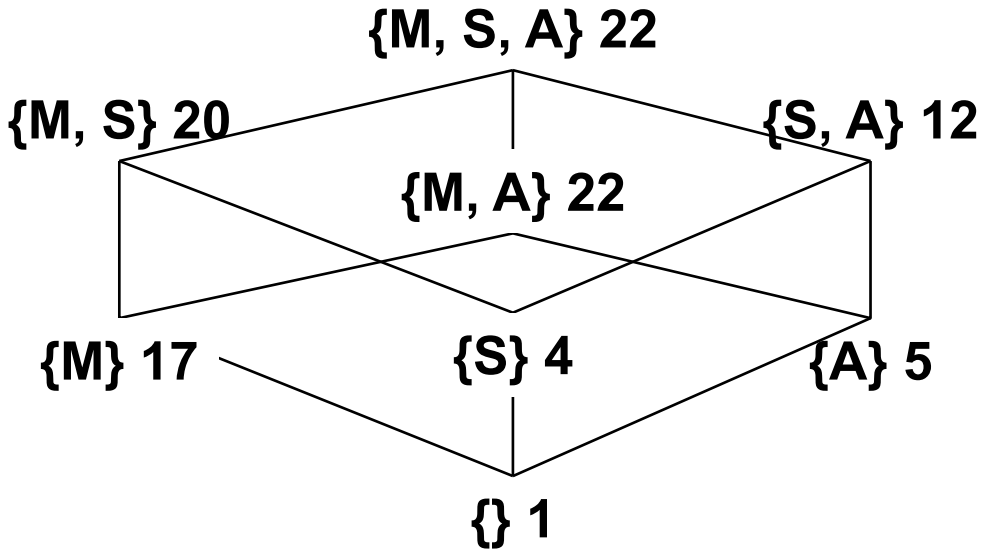
Clicker question: which views can be used for which query? Part 1



Assume that all views are materialized, which of the following views can answer a query about Standing (S)?

A. {M,S,A}
B. {M,A}
C. {}
D. All of the above
E. None of the above

Clicker question: which views can be used for which query? Part 1

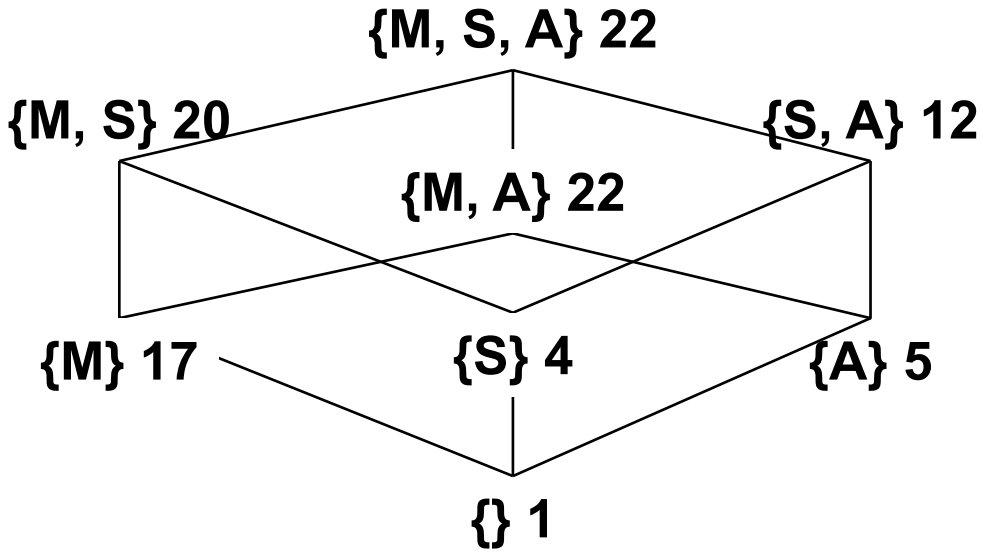


Assume that all views are materialized, which of the following views can answer a query about Standing (S)?

A. {M,S,A}
B. {M,A}
C. {}
D. All of the above
E. None of the above

The correct answer is A; any thing that aggregates S can be used.

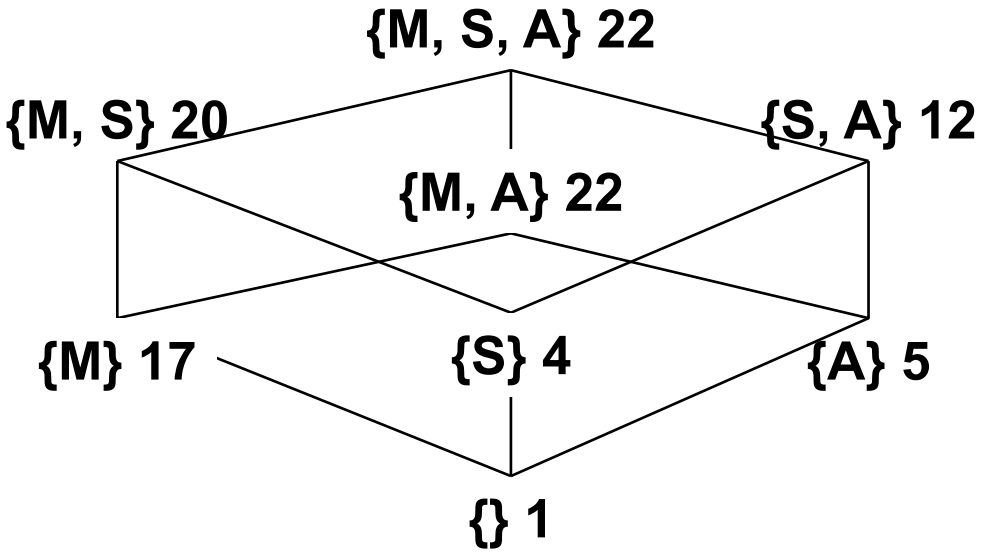
Clicker question: which views can be used for which query? Part 2



If we materialize {S,A}, queries over which lattice parts could be answered using {S,A}?

A. {S,A}
B. {{M,S,A}{S,A}
C. {{S,A},{S},{A},{}}
D. None of the above

Clicker question: which views can be used for which query? Part 2

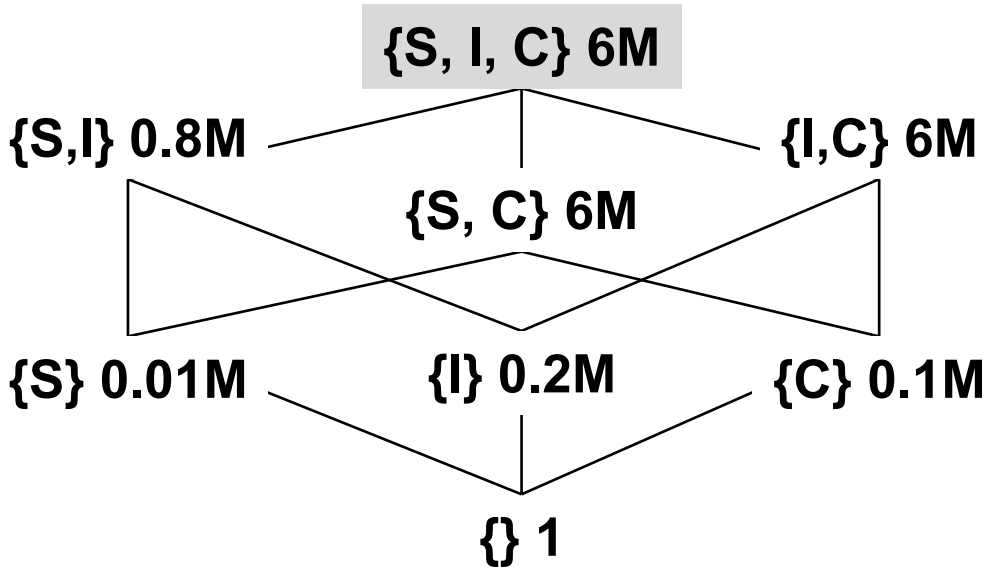


If we materialize {S,A}, queries over which lattice parts could be answered using {S,A}?

A. {S,A}
B. {{M,S,A}{S,A}
C. {{S,A},{S},{A},{}}
D. None of the above

The correct answer is C; {S,A} can be used to answer anything that uses an aggregation on {S,A}

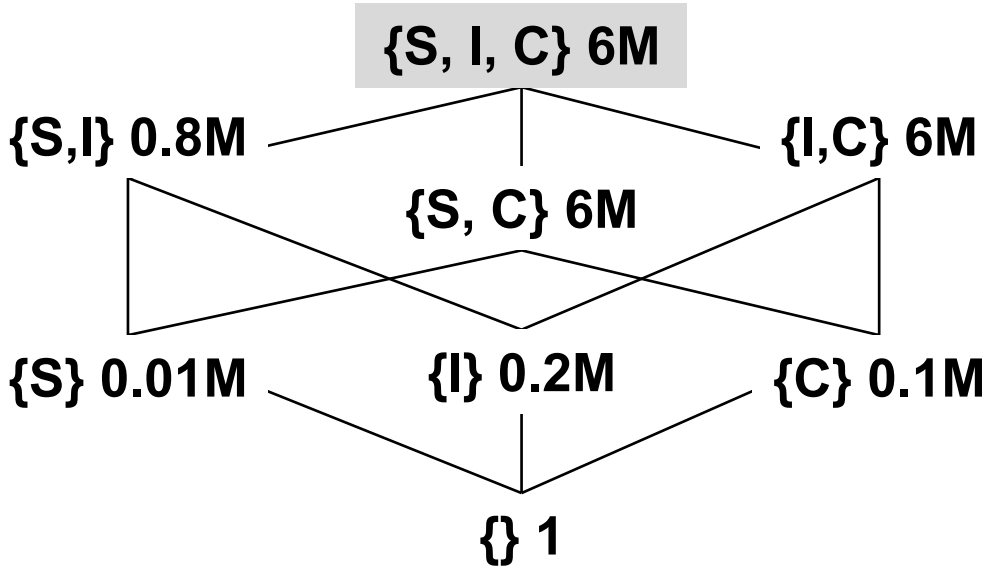
The HRU algorithm for materializing views



- The question is: which views can we materialize to answer queries the cheapest?
- Initially, only the top-most view is materialized

HRU [Harinarayan, Rajaraman, and Ullman, 1996]—SIGMOD Best Paper award—is a greedy algorithm that does not guarantee an optimal solution, though it usually produces a good solution. This solution is a good trade-off in terms of the space used and the average time to answer an OLAP query.

Benefit of Materializing a View



Intuitively, for each view under consideration, determine (1) if it can be used to answer a query and (2) if so, how much does it save?

Formally:

Define the benefit (savings) of view v relative to S as **B(v,S)**.

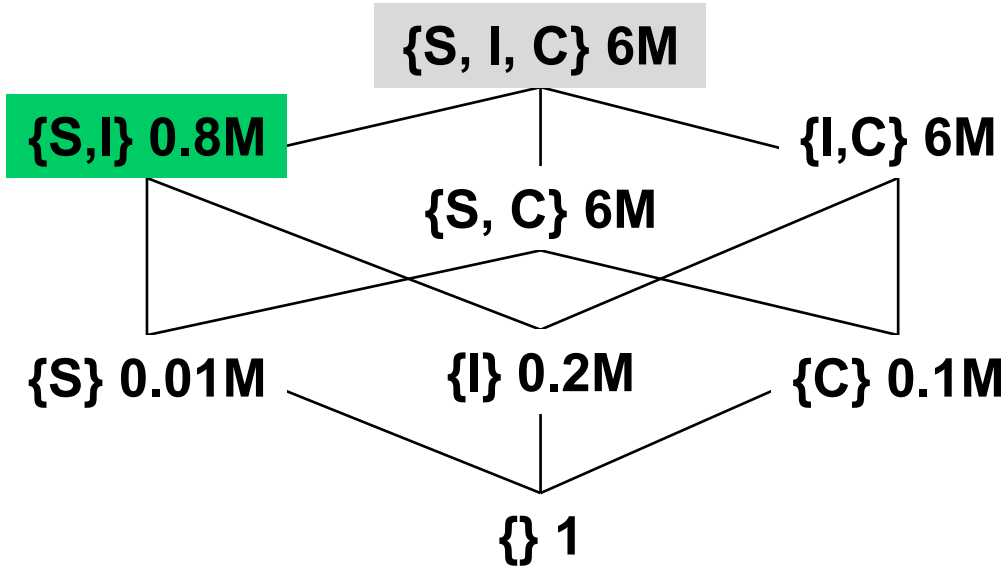
B(v, S) = 0
For each w ≦ v
    u = view of least cost in S such that w ≦ u
    if C(v) < C(u) then B_w = C(u) – C(v)
    else B_w = 0
    B(v,S) = B(v,S) + B_w
end

S = set of views selected for materialization

b ≦ a means b is a descendant of a (including itself) – b can be answered using only a (e.g., {S} ≦ {S,I}

C(v) = cost of view v, which we're approximating by its size

Benefit of Materializing a View



- The number associated with each node represents the number of rows in that view (in millions)
- Initial state has only the top most view materialized

Define the benefit (savings) of view v relative to S as **B(v,S)**.

B(v, S) = 0
For each w ≦ v
    u = view of least cost in S such that w ≦ u
    if C(v) < C(u) then $B_w = C(u) – C(v)$
    else $B_w = 0$
    $B(v,S) = B(v,S) + B_w$
end

Example

S = {{S, I, C}}, **v = {S, I}**

$B_{\{S, I\}}$ **= 5.2 M**

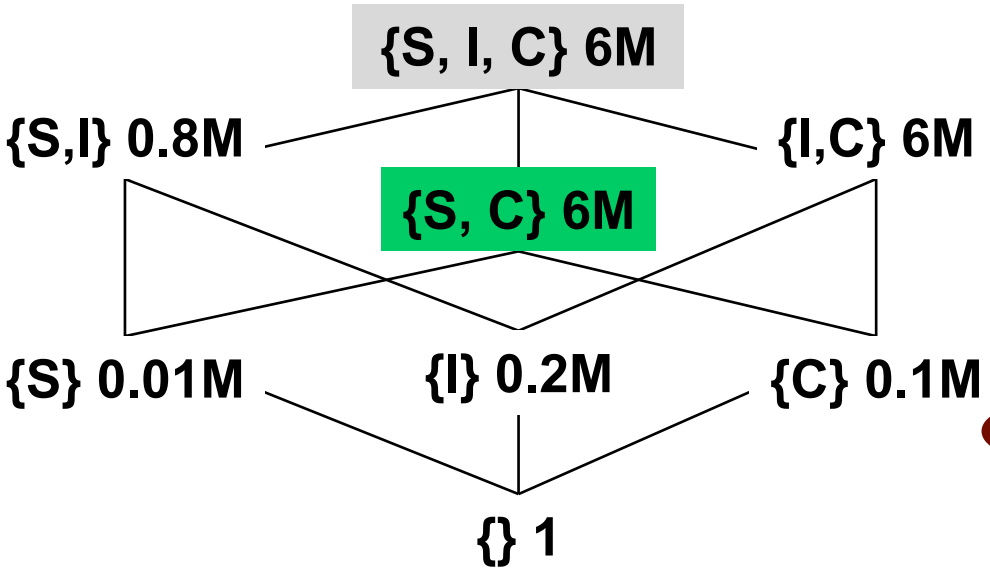
$B_{\{S\}}$ **= 5.2 M**

$B_{\{I\}}$ **= 5.2 M**

$B_{\{\}}$ **= 5.2 M**

B(v,S) = 5.2M *4

Benefit of Materializing a View: Clicker Question



What is the saving of {S,C} relative to {{S,I,C}}?

A. 6M
B. 6M x 4
C. 6M x 2
D. 0
E. None of the above

Define the benefit (savings) of view v relative to S as **B(v,S)**.

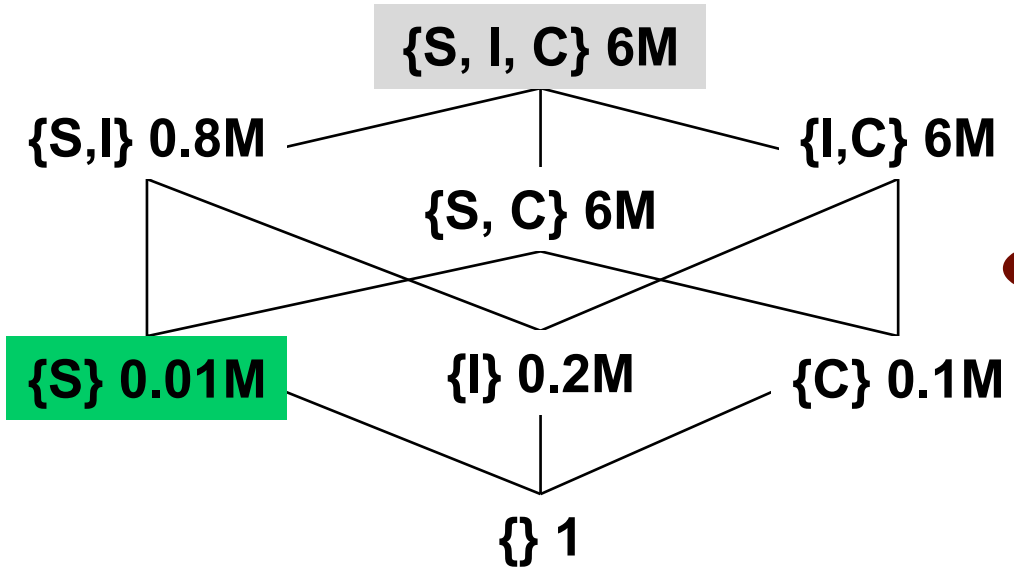
B(v, S) = 0
For each w ≦ v
    u = view of least cost in S such that w ≦ u
    if C(v) < C(u) then $B_w = C(u) – C(v)$
    else $B_w = 0$
    $B(v,S) = B(v,S) + B_w$
end

Calculation:

S = {{S, I, C}}, **v = {S, C}**

$B_{\{S,C\}}$ **= 0**
$B_{\{S\}}$ **= 0**
$B_{\{C\}}$ **= 0**
$B_{\{\}}$ **= 0**
B(v,S) = 0 x 4

Benefit of Materializing a View: Clicker Question



What is the saving of {S} relative to {{S,I,C}}?

A. 5.99M

B. 5.99M x 2

C. .79M

D. .79M x 2

E. None of the above

Define the benefit (savings) of view v relative to S as **B(v,S)**.

B(v, S) = 0
For each w ≦ v
    u = view of least cost in S such that w ≦ u
    if C(v) < C(u) then $B_w = C(u) – C(v)$
    else $B_w = 0$
    $B(v,S) = B(v,S) + B_w$
end

Calculation:

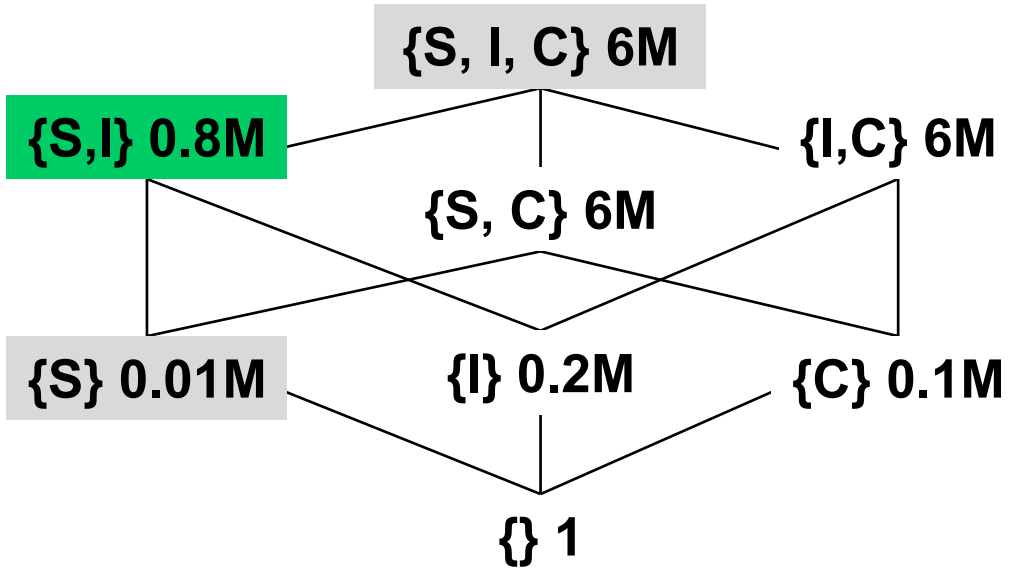
S = {{S, I, C}}, **v = {S}**

$B_{\{S\}}$ **= 5.99 M**

$B_{\{\}}$ **= 5.99 M**

B(v,S) = 5.99M *2

Benefit of Materializing a View: Clicker Question



What is the saving of {S,I} relative to {{S,I,C},{S}}?

A. 5.2M
B. 5.2M x 2
C. 5.2M x 4 M
D. .79M x 2
E. None of the above

Define the benefit (savings) of view v relative to S as **B(v,S)**.

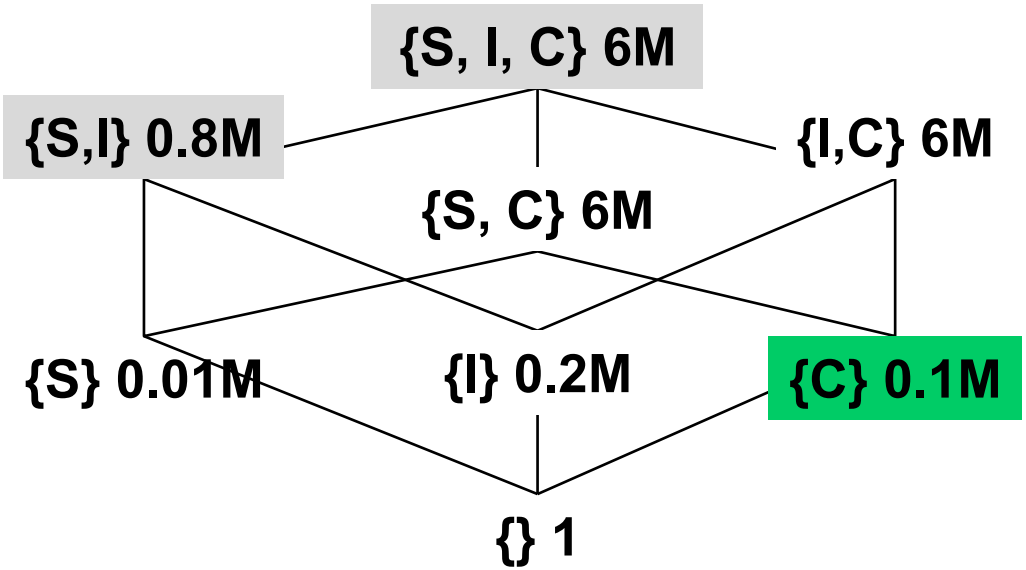
B(v, S) = 0
For each w ≦ v
    u = view of least cost in S such that w ≦ u
    if C(v) < C(u) then $B_w = C(u) – C(v)$
    else $B_w = 0$
    $B(v,S) = B(v,S) + B_w$
end

Calculation:

S = {{S, I, C}, {S}}, **v = {S,I}**

$B_{\{S,I\}}$ **= 5.2 M**
$B_{\{S\}}$ **= 0**
$B_{\{I\}}$ **= 5.2 M**
$B_{\{\}}$ **= 0**
B(v,S) = 5.2M *2

Benefit of Materializing a View: Clicker Question



What is the saving of {C} relative to {{S,I,C},{S,I}}?

A. 5.9M x 2
B. 0.7M x 2
C. 5.9M x 1
D. 0.7 M x 1
E. None of the above

Define the benefit (savings) of view v relative to S as **B(v,S)**.

```
B(v, S) = 0
For each w ≦ v
    u = view of least cost in S such that w ≦ u
    if C(v) < C(u) then  B_w = C(u) – C(v)
    else  B_w = 0
    B(v,S) = B(v,S) + B_w
end
```

Calculation:

S = {{S, I, C}, {S,I}}, **v = {C}**

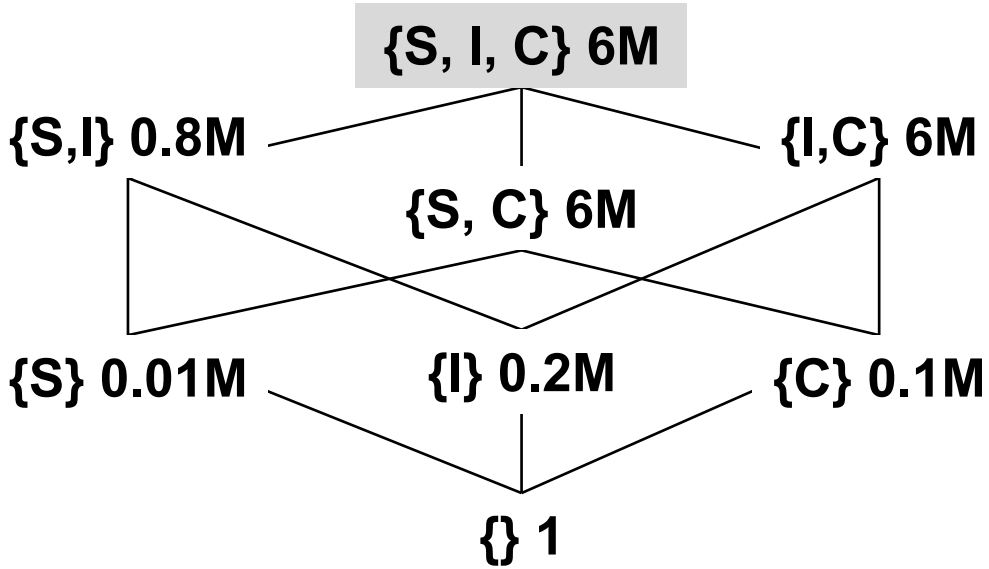
$B_{\{C\}}$ **= 6M - .1M**

$B_{\{\}}$ **= .8M - .1M**

B(v,S) = 5.9M + .7M = 6.6M

$B_{\{\}}$ previously could be computed using {S,I}, hence subtracting from .8M

Finding the Best *k* Views to Materialize



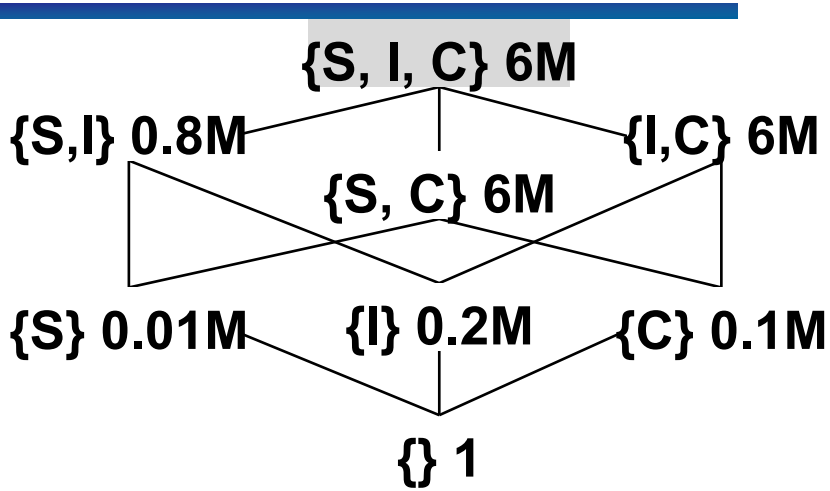
- The number associated with each node represents the number of rows in that view (often in millions)
- Initial state has only the top most view materialized

A greedy algorithm for finding the best *k* views to materialize

```
S = {top view}
for i=1 to k do begin
    select v ⊄ S such that B(v,S) is maximized
    S = S union {v}
end
```

HRU Algorithm Example. Pick the best 2 views beyond {S,I,C} to materialize: Round 1

{S,I} offers biggest benefit: materialize it.



View	1st choice
{S, I}	(6-0.8)M *4 = 20.8M
{S, C}	(6-6) *4 = 0
{I, C}	(6-6) *4 = 0
{S}	(6-0.01) M*2 = 11.98M
{I}	(6-0.2) M*2 = 11.6M
{C}	(6-0.1) M*2 = 11.8M
{}	6M – 1

Explanation

Can impact {S,I}, {S}, {I}, {}. Benefit is over {S,I,C}

Can impact {S,C}, {S},{C},{}. Benefit from {S,I,C} is zero

Can impact {I,C}, {I},{C},{}. Benefit from {S,I,C} is zero

Can impact {S}, {}. Benefit is over {S,I,C}

Can impact {I}, {}. Benefit is over {S,I,C}

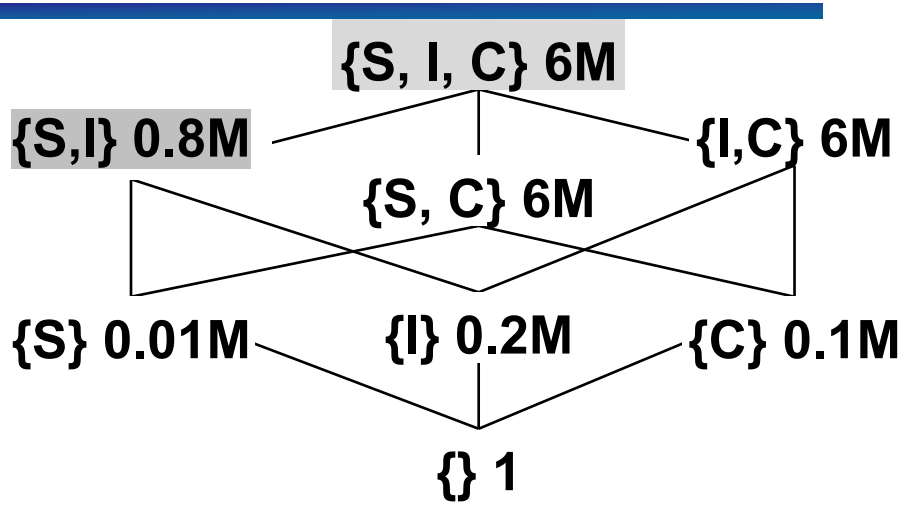
Can impact {C}, {}. Benefit is over {S,I,C}

Can impact {}. Benefit is over {S,I,C}

HRU Algorithm Example. Pick the best 2 views beyond {S,I,C} to materialize: Round 2

{S,I} is already materialized.

{C} offers biggest benefit: materialize it



View	2nd choice
{S, I}	
{S, C}	(6-6) *2 = 0
{I, C}	(6-6) *2 = 0
{S}	(0.8-0.01)M*2 = 1.58M
{I}	(0.8-0.2)M*2 = 1.2M
{C}	**(6-0.1)M + (0.8–0.1)M = 6.6M**
{}	0.8M – 1

Explanation

Already materialized. Not an option
Can impact {S,C}, {S},{C},{}. Benefit of 0 from {S,I,C} for {S,C},{C}. {S,I} is cheaper for {S},{}

Same reasoning as {S,C}

Can impact {S}, {}. Benefit is over {S,I}

Can impact {I}, {}. Benefit is over {S,I}

Can impact {C}, {}. Benefit over {S,I,C} for {C}, {S,I} for {}

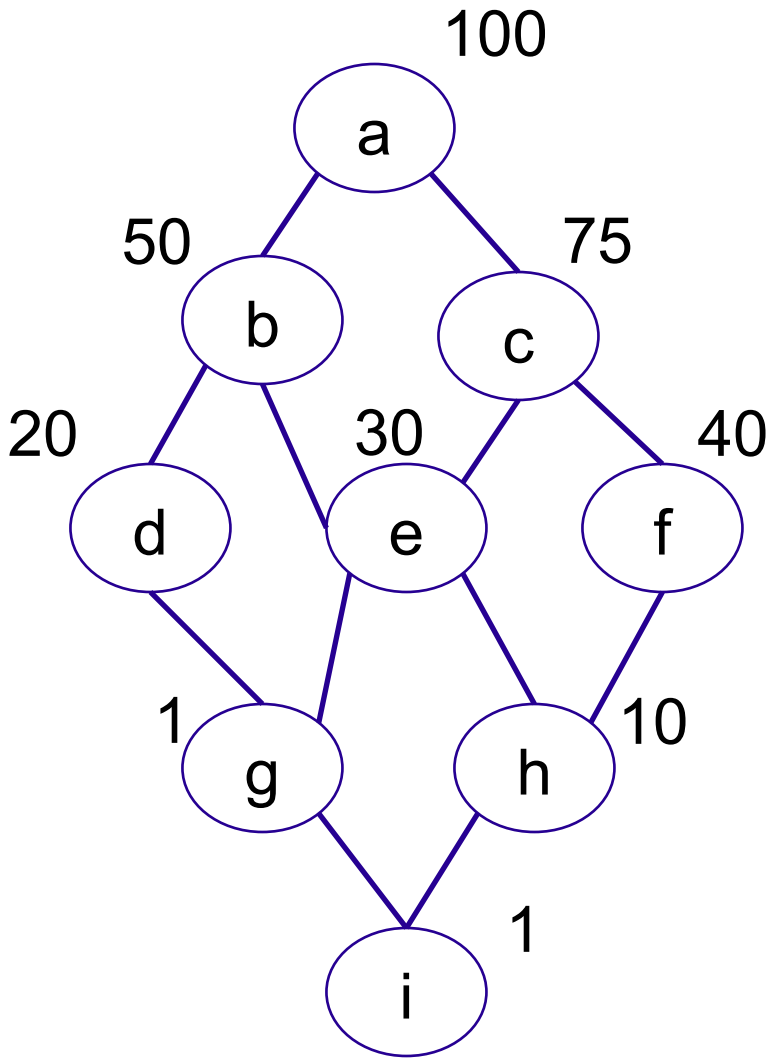
Can impact {}. Benefit is over {S,I}

Administrative Notes
April 1, 2021

- **Project milestone 4 due tonight!!**
  - See the syllabus for late penalty policies
  - Reminder: you cannot change your code after this point
- Upcoming due dates:
  - April 5-9: Project milestone 5
  - April 9: Tutorial 8 (SQL Server)
    - Start early; there is a lot of configuration required
  - April 13: Project milestone 6

In-class Exercise (not for credit)

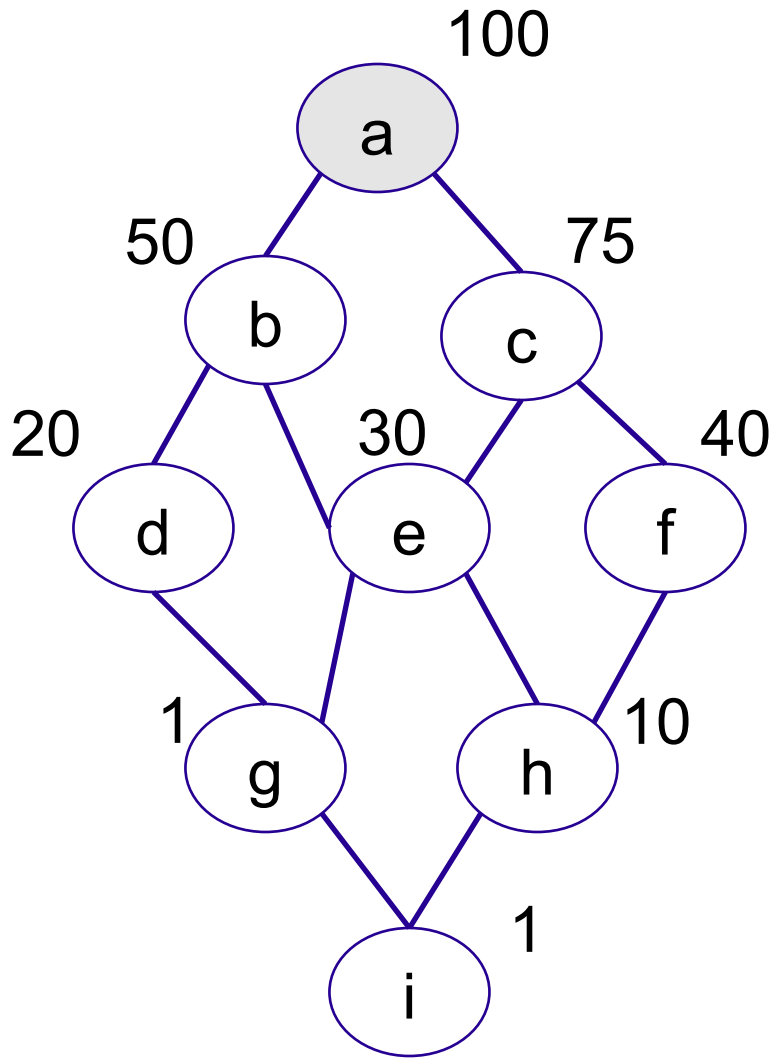
Assuming 'a' is already materialized, what are the best 3 other views that we should materialize? Begin by picking the “best” view to materialize.



In-class Exercise

Assuming 'a' is already materialized, what are the best 3 other views that we should materialize?

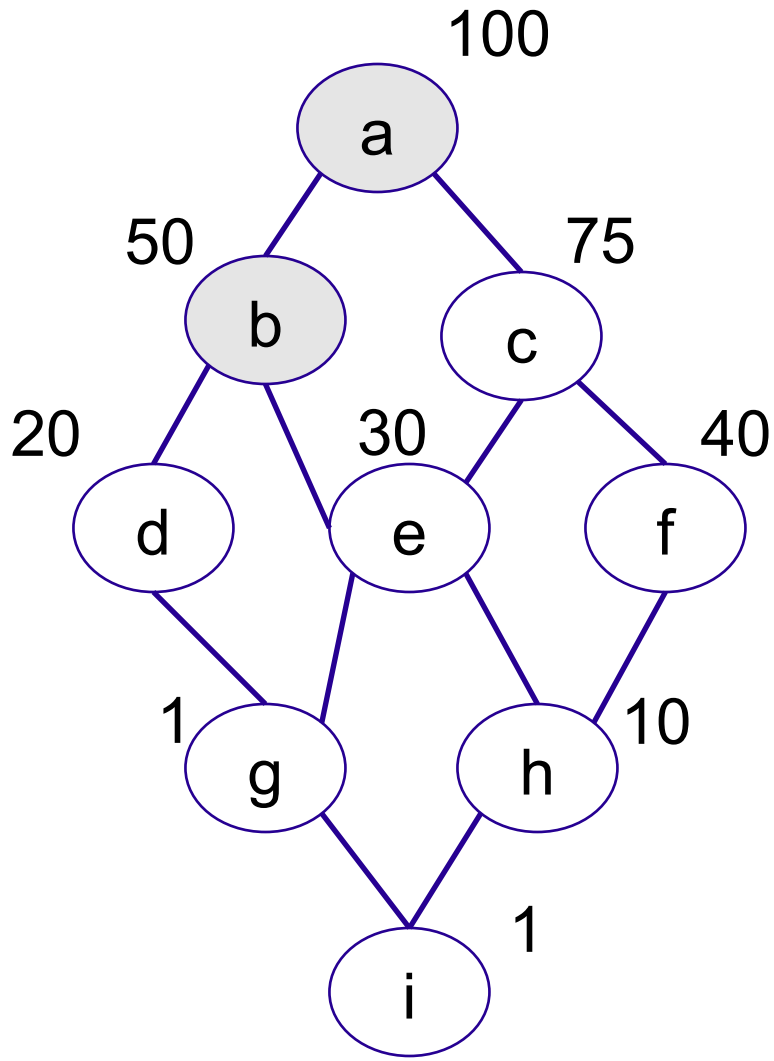
View	1st choice
b	50*6=300
c	25*6=150
d	80*3=240
e	70*4=280
f	60*3=180
g	99*2=198
h	90*2=180
i	99



In-class Exercise

Assuming 'a' is already materialized, what are the best 3 other views that we should materialize?

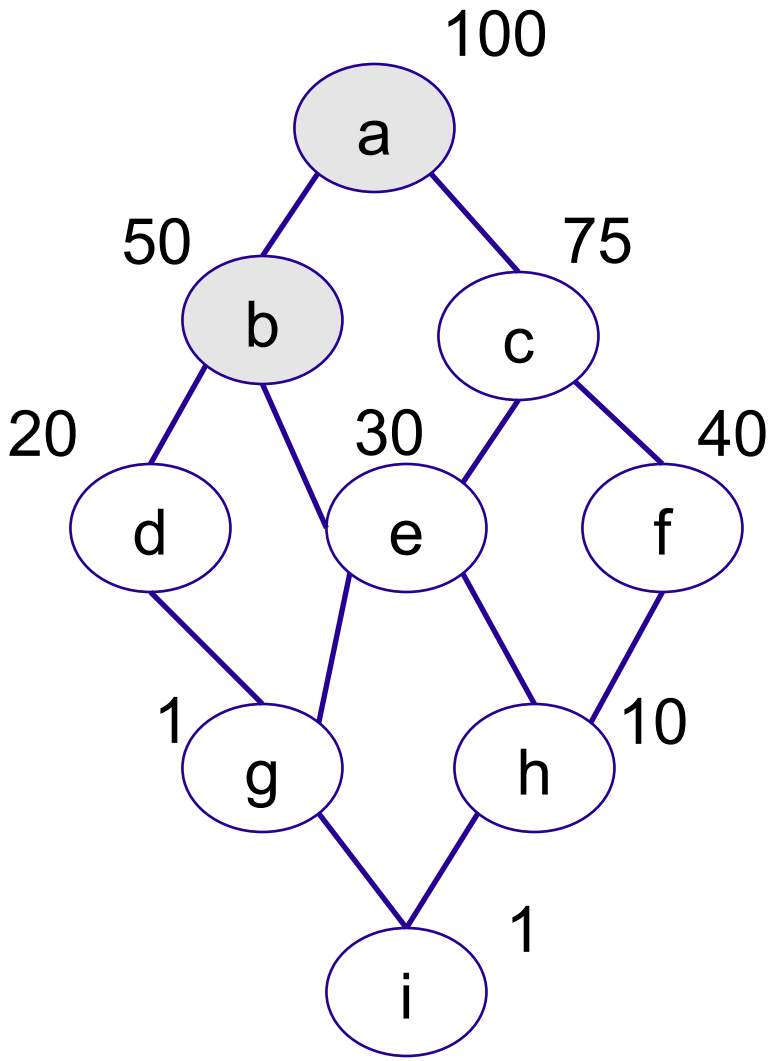
View	1st choice
b	**50*6=300**
c	25*6=150
d	80*3=240
e	70*4=280
f	60*3=180
g	99*2=198
h	90*2=180
i	99



In-class Exercise

Assuming 'a' is already materialized, what are the best 3 other views that we should materialize?

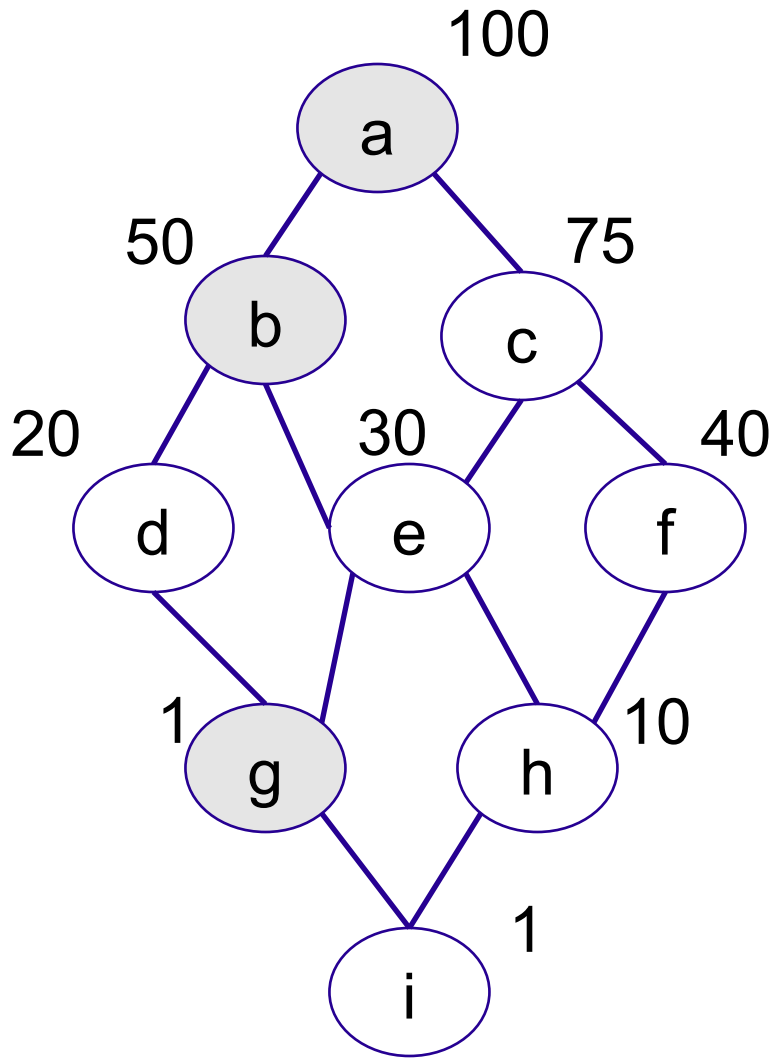
View	1st choice	2nd choice
b	**50*6=300**	
c	25*6=150	25*2=50
d	80*3=240	30*3=90
e	70*4=280	20*4=80
f	60*3=180	60+10*2=80
g	99*2=198	49*2=98
h	90*2=180	40*2=80
i	99	49



In-class Exercise

Assuming 'a' is already materialized, what are the best 3 other views that we should materialize?

View	1st choice	2nd choice
b	**50*6=300**	
c	25*6=150	25*2=50
d	80*3=240	30*3=90
e	70*4=280	20*4=80
f	60*3=180	60+10*2=80
g	99*2=198	**49*2=98**
h	90*2=180	40*2=80
i	99	49



In-class Exercise

Assuming 'a' is already materialized, what are the best 3 other views that we should materialize?

View	1st choice	2nd choice	3rd choice
b	**50*6=300**		
c	25*6=150	25*2=50	25*2=50
d	80*3=240	30*3=90	30
e	70*4=280	20*4=80	20*2=40
f	60*3=180	60+10*2=80	60+10=70
g	99*2=198	**49*2=98**	
h	90*2=180	40*2=80	40
i	99	49	0