+

Tutorial 5：
DW Implementation

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

+ Data Warehouse Review

- Why data warehouse?
  - Data analytic queries
    - Complicated queries: Group-By/Order-By/Aggregation
    - Flexible queries: Hierarchies/Measures
- Why materialized view?
  - Interactive queries (OLAP)
    - Query results should be pre-computed and stored
    - Convert aggregation queries to selective queries
- How to materialize the views?
  - Storage cost
  - Benefits on query performance

+ Question 1

- Consider a data warehouse with d dimensions. The fact table T contains $|T|$ records, and each dimension A_i contains $|A_i|$ distinct values
  - $d = 2$
  - $|T| = 12$
  - $A_1 = 3,\ A_2 = 4$

RID	***item***	***location***	***sales***
R1	computer	Chicago	882
R2	computer	New York	968
R3	computer	Toronto	746
R4	computer	Vancouver	825
R5	phone	Chicago	89
R6	phone	New York	38
R7	phone	Toronto	43
R8	phone	Vancouver	14
R9	security	Chicago	623
R10	security	New York	872
R11	security	Toronto	591
R12	security	Vancouver	400

+ Q1 Bitmap Index

- (a) Assume that we construct a bitmap index for each dimension. What is the total size (i.e., number of bits) of the bitmap indices?

RID	***item***	***location***	***sales***
R1	computer	Chicago	882
R2	computer	New York	968
R3	computer	Toronto	746
R4	computer	Vancouver	825
R5	phone	Chicago	89
R6	phone	New York	38
R7	phone	Toronto	43
R8	phone	Vancouver	14
R9	security	Chicago	623
R10	security	New York	872
R11	security	Toronto	591
R12	security	Vancouver	400

+ Q1 Bitmap Index

- Index on a particular column
  - Pick up all distinct values in the column
    - *Computer, phone* and *security*
  - Create a bit vector for each distinct value
    - Vector length = # of records
- Index on *item*

computer	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
phone	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**
security	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**

RID	***item***
R1	computer
R2	computer
R3	computer
R4	computer
R5	phone
R6	phone
R7	phone
R8	phone
R9	security
R10	security
R11	security
R12	security

+ Q1 Bitmap Index

- Advantages
  - Less space and I/O
    - Raw table: 16(varchar(16))*12(records) = 192 bytes
    - Index:
      - Dynamic part:
        - Index size: 3(distinct values)*12(records) = 36 bits ≈ 5 bytes
      - Fixed part:
        - Dictionary size: 3(distinct values)*16(value name) = 48 bytes
        - Other costs: Pointers, index length, etc.
    - Increase slowly when data scales
      - Fixed part becomes negligible when table is large
      - Only consider dynamic part when calculating size

item
computer
computer
computer
computer
phone
phone
phone
phone
security
security
security
security

computer	1	1	1	1	0	0	0	0	0	0	0	0
phone	0	0	0	0	1	1	1	1	0	0	0	0
security	0	0	0	0	0	0	0	0	1	1	1	1

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

+ Q1 Bitmap Index

- (a) Assume that we construct a bitmap index for each dimension. What is the total size (i.e., number of bits) of the bitmap indices?
  - Total number of indices
    - Count all distinct values in dimensions
      - $\sum_{i=1}^{d} |A_i|$
  - Each vertex size is $|T|$ bits
  - Total size is $|T| \sum_{i=1}^{d} |A_i|$ bits

RID	item	location	sales
R1	computer	Chicago	882
R2	computer	New York	968
R3	computer	Toronto	746
R4	computer	Vancouver	825
R5	phone	Chicago	89
R6	phone	New York	38
R7	phone	Toronto	43
R8	phone	Vancouver	14
R9	security	Chicago	623
R10	security	New York	872
R11	security	Toronto	591
R12	security	Vancouver	400

computer												
phone												
security												

Chicago												
NY												
Toronto												
Van												

+ Q1 Bitmap Index

- (b) Given the example below, please create bitmap indices for both dimensions.
  - *Item*: 3 distinct values
  - *Location*: 4 distinct values

RID	***item***	***location***	***sales***
R1	computer	Chicago	882
R2	computer	New York	968
R3	computer	Toronto	746
R4	computer	Vancouver	825
R5	phone	Chicago	89
R6	phone	New York	38
R7	phone	Toronto	43
R8	phone	Vancouver	14
R9	security	Chicago	623
R10	security	New York	872
R11	security	Toronto	591
R12	security	Vancouver	400

computer	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
phone	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**
security	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**

Chicago	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
New York	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**
Toronto	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**
Vancouver	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**

THE UNIVERSITY
OF QUEENSLAND
AUSTRALIA

Q1 Bitmap Index

- Advantages
  - Less query time
    - comparison/join/aggregation –> bit operations
      - Bit operations are very fast

computer	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
phone	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**
security	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**

RID	***item***	***location***	***sales***
R1	computer	Chicago	882
R2	computer	New York	968
R3	computer	Toronto	746
R4	computer	Vancouver	825
R5	phone	Chicago	89
R6	phone	New York	38
R7	phone	Toronto	43
R8	phone	Vancouver	14
R9	security	Chicago	623
R10	security	New York	872
R11	security	Toronto	591
R12	security	Vancouver	400

+ Q1 Bitmap Index

- (c) How can we use the bitmap indices to answer the following queries?
  - Find the total sales of each item type
    - Scan computer vector and select all records whose computer = 1
    - Summarize their sales
    - Perform the same to other items

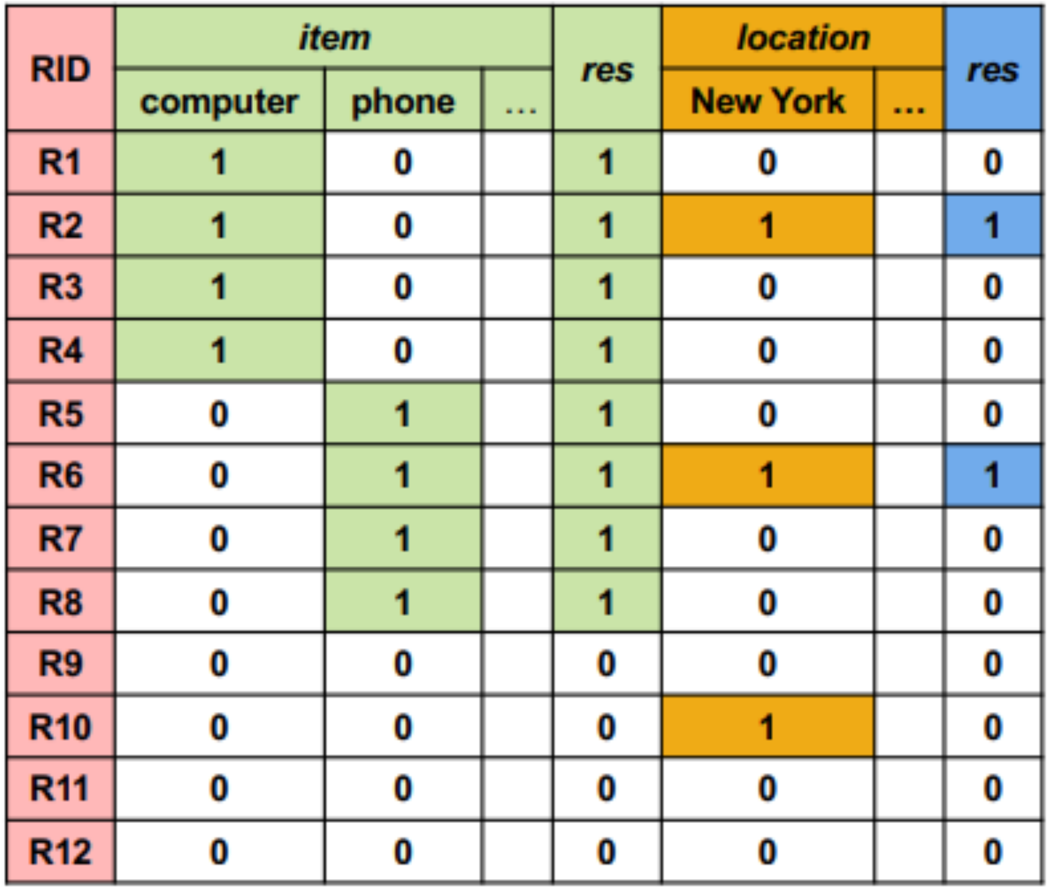
RID	*item*		
	computer	phone	security
R1	1	0	0
R2	1	0	0
R3	1	0	0
R4	1	0	0
R5	0	1	0
R6	0	1	0
R7	0	1	0
R8	0	1	0
R9	0	0	1
R10	0	0	1
R11	0	0	1
R12	0	0	1

RID	*sales*
R1	882
R2	968
R3	746
R4	825
R5	89
R6	38
R7	43
R8	14
R9	623
R10	872
R11	591
R12	400

+ Q1 Bitmap Index

- (c) How can we use the bitmap indices to answer the following queries?
  - Find the total sales of “computer” and “phone” in “New York”
    - “computer” OR “phone”
    - AND “New York”

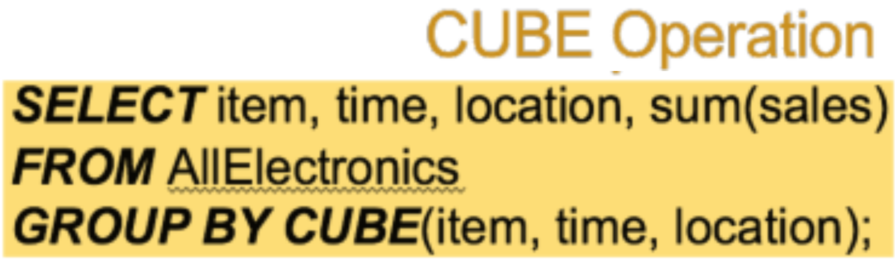
SELECT *
FROM bitmap
WHERE (item = ‘computer’
OR item = ‘phone’)
AND location = ‘New York’



RID	*item*			*res*	*location*		*res*
	computer	phone	…		New York	…	
R1	1	0		1	0		0
R2	1	0		1	1		1
R3	1	0		1	0		0
R4	1	0		1	0		0
R5	0	1		1	0		0
R6	0	1		1	1		1
R7	0	1		1	0		0
R8	0	1		1	0		0
R9	0	0		0	0		0
R10	0	0		0	1		0
R11	0	0		0	0		0
R12	0	0		0	0		0

+ Question 2

- Consider a data warehouse with d dimensions, and a data cube constructed on all these dimensions $\{A_1, \dots, A_d\}$
- (a) How many cuboids will be created if the dimensions have no hierarchies, and why?
  - Cuboid: the result of a parameterized group-by query on a data cube
    - Data cube: {item, time, location}
      - $2^3 = 8$ cuboids
  - $2 * 2 * \cdots * 2 = 2^d$ cuboids
    - Two choices for each dimension (parameterize it or not)
    - *d* dimensions in total



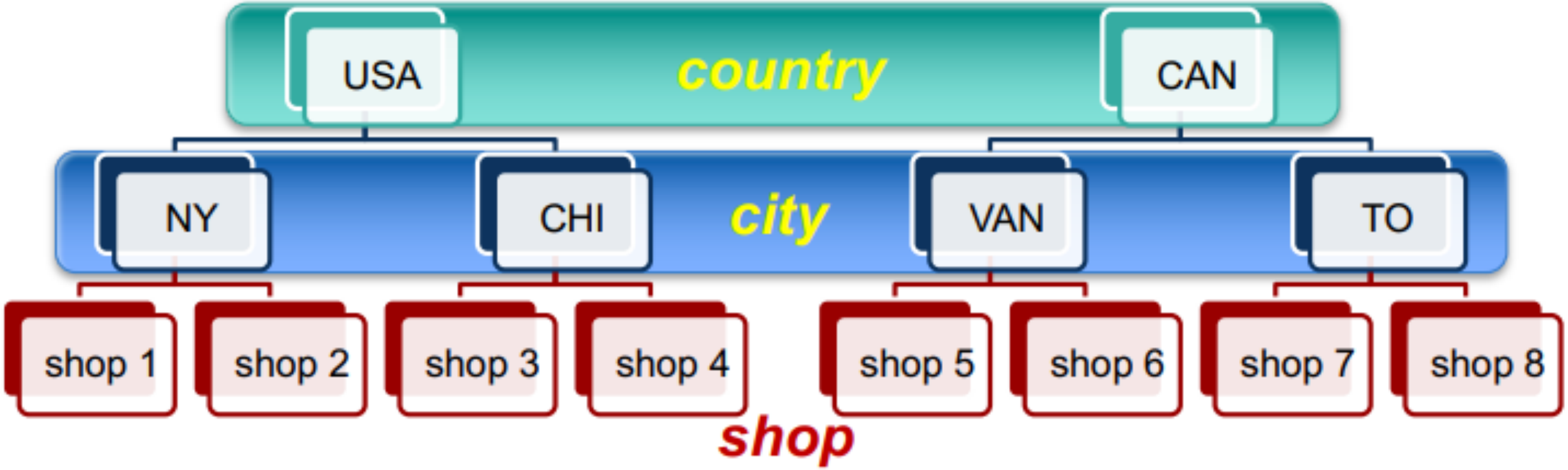
CUBE Operation

```
SELECT item, time, location, sum(sales)
FROM AllElectronics
GROUP BY CUBE(item, time, location);
```

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

+ Q2 Cuboid

- (b) Suppose that each dimension A_i contains L_i levels in the hierarchy. How many cuboids will be created, and why?
  - For dimension A_i, $L_i + 1$ choices
    - L_i: if chosen, it should parameterize on one of the levels
    - 1: group-by query doesn't contain this dimension
  - *d* dimensions in total
  - $\prod_{i=1}^{d}(L_i + 1)$ $2 * 2 * \cdots * 2 = 2^d \ (if\ L_i = 1)$



+ Q2 Cuboid

- (c) Consider the *AllElectronics* data warehouse
  - Three dimensions and one measure
    - Time: [day <month < quarter < year]
    - Item: [item name < brand < type]
    - Location: [street <city < state < country]
    - Measure: sales
  - Given a group-by query on {brand, state}, can we use each of the following cuboids to answer the query, and why?
    - Cuboid1: {year, item name, city}
    - Cuboid2: {year, brand, country}
    - Cuboid3: {year, brand, state}
    - Cuboid4: {item name, state}

+ Q2 Cuboid

- (c) Consider the *AllElectronics* data warehouse
  - Three dimensions and one measure
    - Time: [day <month < quarter < year]
    - Item: [item name < brand < type]
    - Location: [street <city < state < country]
    - Measure: sales
  - Query: {brand, state}
    - All the pre-aggregated results on lower levels can be used in answering queries on higher levels
      - Jan:45,Feb:55,Mar:40 –>1st quarter:140
    - Dimensions that are not parameterized in query is summarized to the most general level(all_years, all_types, all_countries)
      - All the pre-aggregated results in that dimension can be used

+ Q2 Cuboid

- (c) Consider the *AllElectronics* data warehouse
  - Three dimensions and one measure
    - Time: [day <month < quarter < year] ✓
    - Item: [item name < brand < type] ✓
    - Location: [street <city < state < country] ✓
    - Measure: sales
  - Query: {brand, state}
  - Given cuboid
    - Cuboid1: {year, item name, city} ✓

+ Q2 Cuboid

- (c) Consider the *AllElectronics* data warehouse
  - Three dimensions and one measure
    - Time: [day <month < quarter < year] ✓
    - Item: [item name < brand < type] ✓
    - Location: [street <city < state < country] ✗
    - Measure: sales
  - Query: {brand, state}
  - Given cuboid
    - Cuboid2: {year, brand, country} ✗

+ Q2 Cuboid

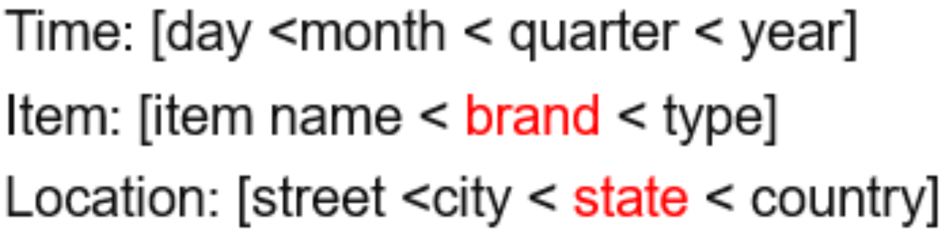
- (c) Consider the *AllElectronics* data warehouse
  - Three dimensions and one measure
    - Time: [day <month < quarter < year] ✓
    - Item: [item name < brand < type] ✓
    - Location: [street <city < state < country] ✓
    - Measure: sales
  - Query: {brand, state}
  - Given cuboid
    - Cuboid3: {year, brand, state} ✓

+ Q2 Cuboid

- (c) Consider the *AllElectronics* data warehouse
  - Three dimensions and one measure
    - Time: [day <month < quarter < year] ✓
    - Item: [item name < brand < type] ✓
    - Location: [street <city < state < country] ✓
    - Measure: sales
  - Query: {brand, state}
  - Given cuboid
    - Cuboid4: {item name, state} ✓

+ Q2 Cuboid

- (d) Which of the above cuboids is the best, in terms of query efficiency, to answer the group-by query on **{brand, state}**, and why?
  - Cuboid1: {year, item name, city} ✓
  - Cuboid2: {year, brand, country} ✗
  - Cuboid3: {year, brand, state}
  - Cuboid4: {item name, state}



Time: [day <month < quarter < year]

Item: [item name < brand < type]

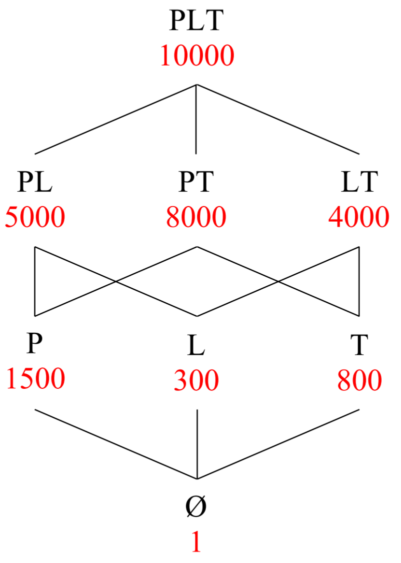
Location: [street <city < state < country]

+ Q2 Cuboid

- Compare the cost of aggregation
  - Cuboid3: {year, brand, state} (Year –> All_Years)
  - Cuboid4: {item name, state} (Item name –> brand)
    - The final results are the same
    - Compare the number of candidates to aggregate
      - # of years is usually less than # of items
    - Cuboid3 is better in this case

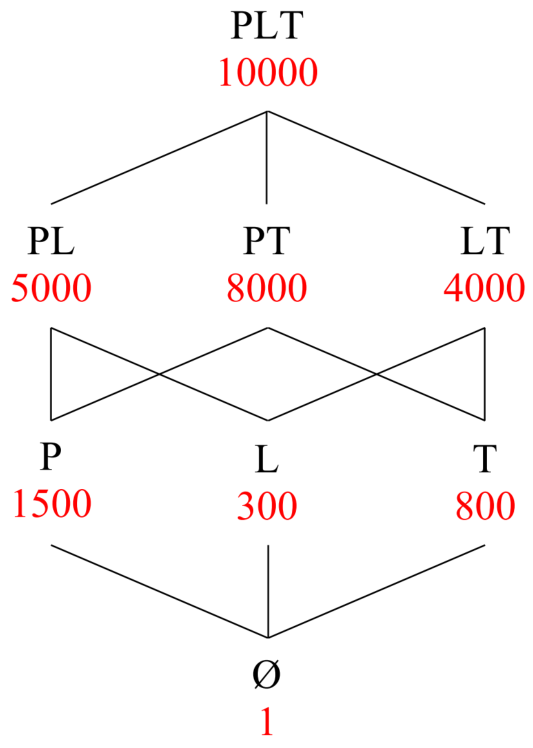
+ Question 3

- Lattice of a data warehouse
  - Three dimensions: *product(P)*, *location(L)* and *time(T)*
    - No hierarchy
  - Red number: the cost of using the corresponding cuboid, if materialized, to answer a group-by query
  - Frequency distribution of group-by queries
    - $\{PTL\ (0.05), PL\ (0.25), PT\ (0.15), LT\ (0.1), P\ (0.2), L\ (0.1), T\ (0.1), \emptyset\ (0.05)\}$



+ Q3 Lattice

- Dimensions
  - PLT: group-by{product, location, time}
    - Return the total sales of a particular product on a particular location on one day
  - Φ: no group-by parameter
    - Return the total sales of the entire table
    - Only one value is returned



Q3 Lattice

- Cuboid query cost
  - The cost of answering a group-by query using the given table
    - Scan the table and perform a further group-by
    - Proportionate to the size of the cuboid

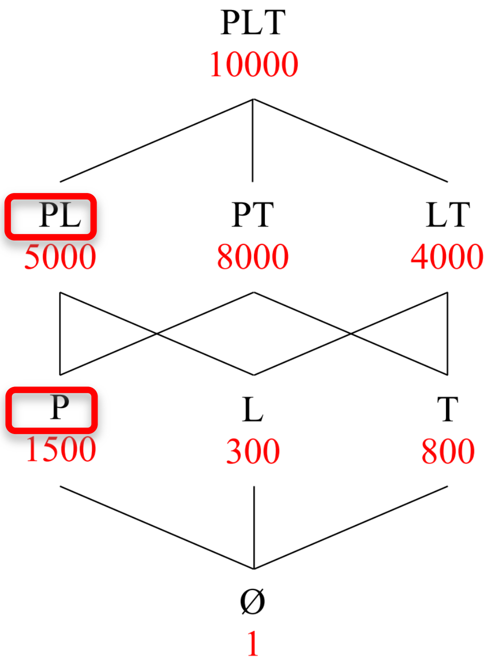
RID	product	time	location	sales
R1	computer	Q1	Chicago	441
R2	computer	Q2	Chicago	441
R3	phone	Q1	Chicago	89
R4	security	Q1	Chicago	623
R5	computer	Q4	New York	968
R6	phone	Q3	New York	38
R7	security	Q1	New York	872
R8	computer	Q1	Vancouver	825
R9	phone	Q1	Vancouver	14
R10	security	Q1	Vancouver	400

⇨

RID	product	location	sales
R1	computer	Chicago	882
R2	computer	New York	968
R3	computer	Vancouver	825
R4	phone	Chicago	89
R5	phone	New York	38
R6	phone	Vancouver	14
R7	security	Chicago	623
R8	security	New York	872
R9	security	Vancouver	400

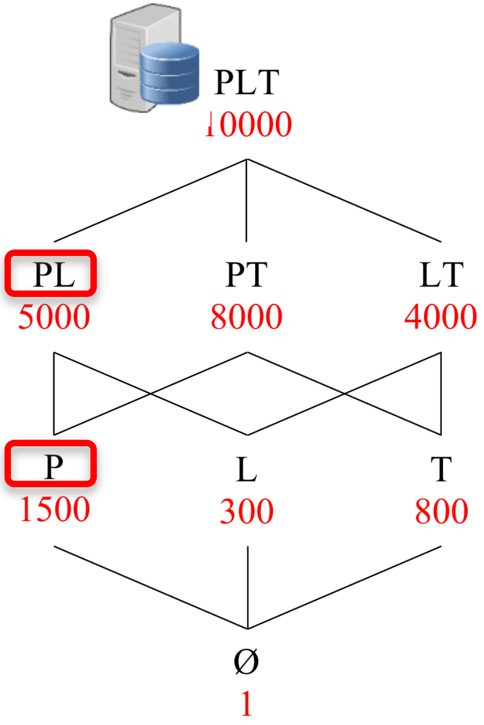
⇨

RID	product	sales
R1	computer	2675
R2	phone	141
R3	security	1895



+ Q3 Lattice

- Cuboid query cost
  - The cost of answering a group-by query using the given table
    - Scan the table and perform a further group-by
    - Proportionate to the size of the cuboid
  - Maintaining cuboid *PL* and answer query *P*
    - Previous cost: 10000 (using *PLT*)
    - Current cost: 5000 (using *PL*)
    - Benefit: 10000-5000
    - Queries that benefit: *PL, P, L, Φ*
    - Total benefit: (10000-5000)*4

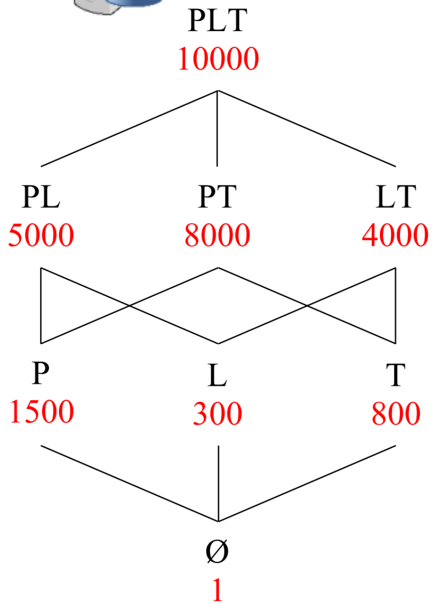


+ Q3 Lattice

- Frequency distribution of group-by queries
  - $\{PTL\ (0.05), PL\ (0.25), PT\ (0.15), LT\ (0.1), P\ (0.2), L\ (0.1), T\ (0.1), Ø\ (0.05)\}$
- What are the first two cuboids that should be materialized in order to minimize total query cost, and why?
  - The first cuboids should be PLT always
    - Only PLT can answer queries that group-by on PLT
  - Calculate the total benefit of materializing a second cuboid
    - Choose PL

Queries	Previous	Current	Weight	Benefit
PL	10000	5000	0.25	1250
P	10000	5000	0.2	1000
L	10000	5000	0.1	500
Φ	10000	5000	0.05	250

Cuboid	Benefit
PL	3000
PT	
LT	
P	
L	
T	
Φ	



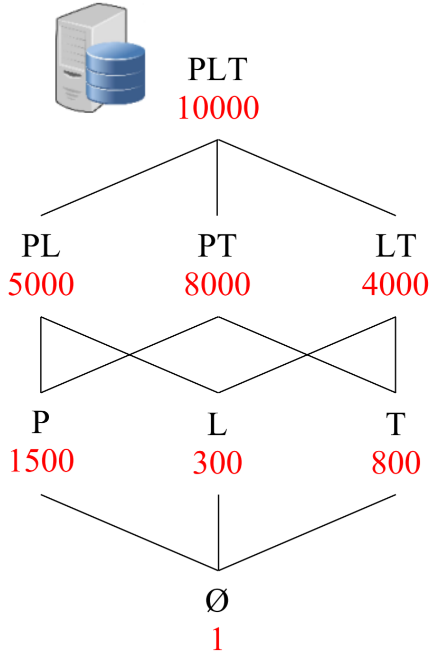
THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

+ Q3 Lattice

- Frequency distribution of group-by queries
  - $\{PTL\ (0.05), PL\ (0.25), PT\ (0.15), LT\ (0.1), P\ (0.2), L\ (0.1), T\ (0.1), Ø\ (0.05)\}$
- What are the first two cuboids that should be materialized in order to minimize total query cost, and why?
  - The first cuboids should be PLT always
    - Only PLT can answer queries that group-by on PLT
  - Calculate the total benefit of materializing a second cuboid
    - Choose PT

Queries	Previous	Current	Weight	Benefit
PT	10000	8000	0.15	300
P	10000	8000	0.2	400
T	10000	8000	0.1	200
Φ	10000	8000	0.05	100

Cuboid	Benefit
PL	3000
PT	1000
LT	
P	
L	
T	
Φ	

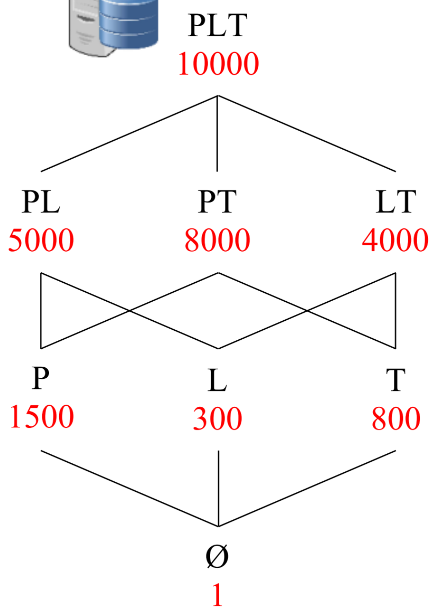


+ Q3 Lattice

- Frequency distribution of group-by queries
  - $\{PTL\ (0.05), PL\ (0.25), PT\ (0.15), LT\ (0.1), P\ (0.2), L\ (0.1), T\ (0.1), Ø\ (0.05)\}$
- What are the first two cuboids that should be materialized in order to minimize total query cost, and why?
  - The first cuboids should be PLT always
    - Only PLT can answer queries that group-by on PLT
  - Calculate the total benefit of materializing a second cuboid
    - Choose LT

Queries	Previous	Current	Weight	Benefit
LT	10000	4000	0.1	600
L	10000	4000	0.1	600
T	10000	4000	0.1	600
Φ	10000	4000	0.05	300

Cuboid	Benefit
PL	3000
PT	1000
LT	2100
P	
L	
T	
Φ	

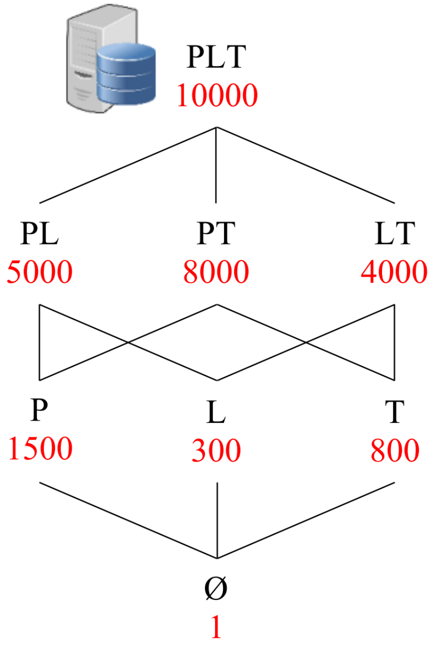


+ Q3 Lattice

- Frequency distribution of group-by queries
  - $\{PTL\ (0.05), PL\ (0.25), PT\ (0.15), LT\ (0.1), P\ (0.2), L\ (0.1), T\ (0.1), Ø\ (0.05)\}$
- What are the first two cuboids that should be materialized in order to minimize total query cost, and why?
  - The first cuboids should be PLT always
    - Only PLT can answer queries that group-by on PLT
  - Calculate the total benefit of materializing a second cuboid
    - Choose P

Queries	Previous	Current	Weight	Benefit
P	10000	1500	0.2	1700
Φ	10000	1500	0.05	425

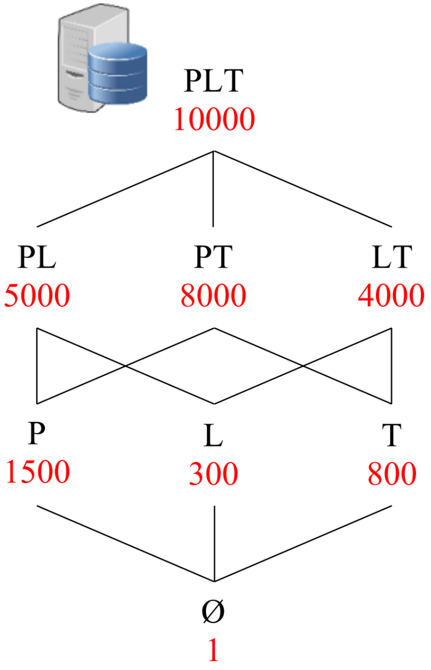
Cuboid	Benefit
PL	3000
PT	1000
LT	2100
P	2125
L	
T	
Φ	



+ Q3 Lattice

- Frequency distribution of group-by queries
  - $\{PTL\ (0.05), PL\ (0.25), PT\ (0.15), LT\ (0.1), P\ (0.2), L\ (0.1), T\ (0.1), Ø\ (0.05)\}$
- What are the first two cuboids that should be materialized in order to minimize total query cost, and why?
  - The first cuboids should be PLT always
    - Only PLT can answer queries that group-by on PLT
  - Calculate the total benefit of materializing a second cuboid
  - Choose the one with highest benefit

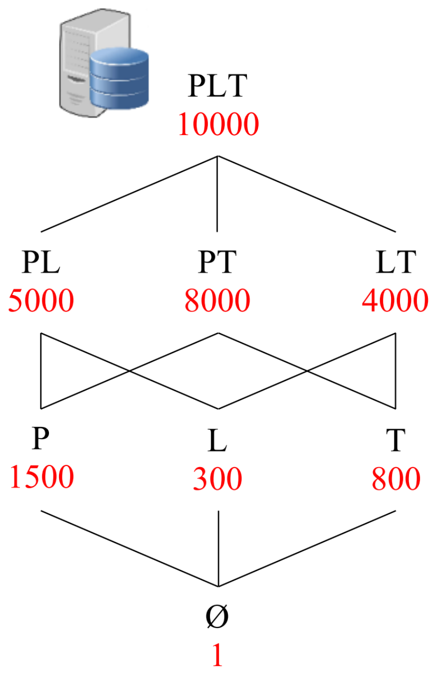
Cuboid	Benefit
PL	3000
PT	1000
LT	2100
P	2125
L	1455
T	1380
Φ	499.95



Q3 Lattice

- Frequency distribution of group-by queries
  - $\{PTL\ (0.05), PL\ (0.25), PT\ (0.15), LT\ (0.1), P\ (0.2), L\ (0.1), T\ (0.1), Ø\ (0.05)\}$
- What are the first two cuboids that should be materialized in order to minimize total query cost, and why?
  - The first cuboids should be PLT always
    - Only PLT can answer queries that group-by on PLT
  - Calculate the total benefit of materializing a second cuboid
  - Choose the one with highest benefit

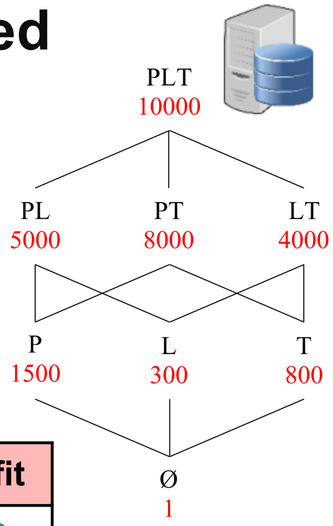
Cuboid	Benefit
PL	3000
PT	1000
LT	2100
P	2125
L	1455
T	1380
Φ	499.95



+ Q3 Lattice

- Frequency distribution of group-by queries
  - $\{PTL\ (0.05), PL\ (0.25), PT\ (0.15), LT\ (0.1), P\ (0.2), L\ (0.1), T\ (0.1), Ø\ (0.05)\}$

- **What are the first two cuboids that should be materialized in order to minimize total query cost, and why?**
  - The first cuboids should be PLT always
    - Only PLT can answer queries that group-by on PLT
  - Calculate the total benefit of materializing a second cuboid
  - Choose the one with highest benefit



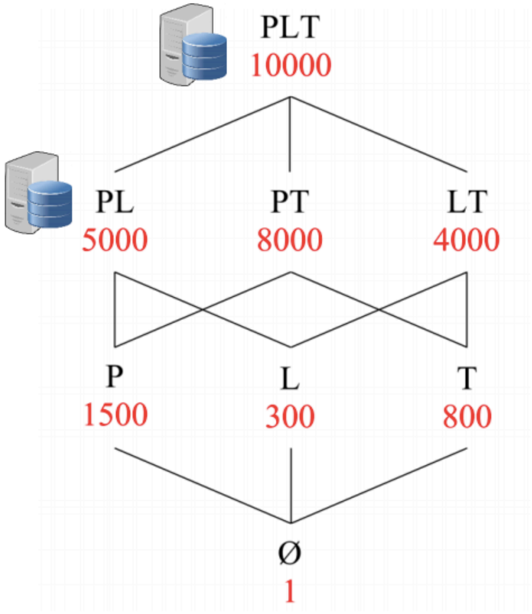
- **Better way?**
  - Same benefit for all queries under the cuboid
  - Benefit*total weight
    - Benefit(PL) = (10000-5000)*(0.25+0.2+0.1+0.05) = 3000
    - Benefit(P) = (10000-1500)*(0.2+0.05) = 2125

Cuboid	Benefit
PL	3000
PT	1000
LT	2100
P	2125
L	1455
T	1380
Φ	499.95

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

+ Q3 Lattice

- Frequency distribution of group-by queries
  - $\{PTL\ (0.05), PL\ (0.25), PT\ (0.15), LT\ (0.1), P\ (0.2), L\ (0.1), T\ (0.1), Ø\ (0.05)\}$
- **What are the first three cuboids that should be materialized in order to minimize total query cost, and why?**
  - The first cuboids should be PLT always
    - Only PLT can answer queries that group-by on PLT
  - $\{PTL, PL\}$
  - How to choose the third one?
    - Consider both PL and PLT when calculating the benefit of materializing a new cuboid.



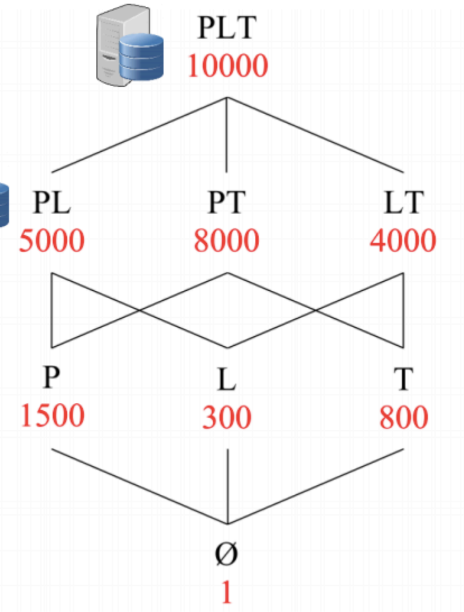
+ Q3 Lattice

- Frequency distribution of group-by queries
  - $\{PTL\ (0.05), PL\ (0.25), PT\ (0.15), LT\ (0.1), P\ (0.2), L\ (0.1), T\ (0.1), Ø\ (0.05)\}$
- **What are the first three cuboids that should be materialized in order to minimize total query cost, and why?**
  - $\{PTL,\ PL\}$
  - How to choose the third one?
    - Consider both PL and PLT when calculating the benefit of materializing a new cuboid.

Choose PT

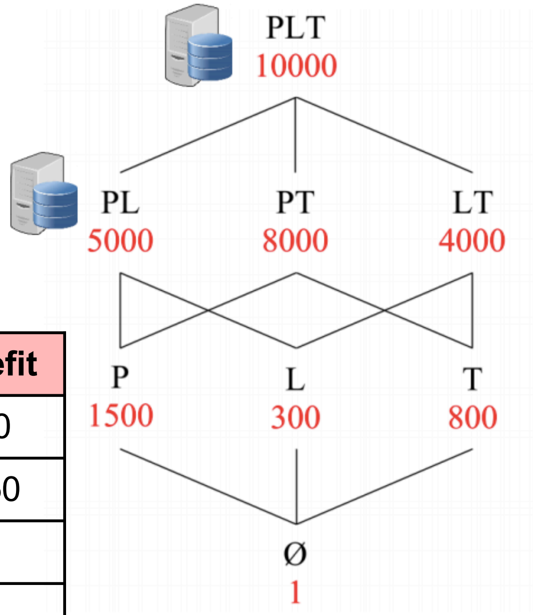
Queries	Previous	Current	Weight	Benefit
PT	10000	8000	0.15	300
P	5000	8000	0.2	0
T	10000	8000	0.1	200
Φ	5000	8000	0.05	0

Cuboid	Benefit
PT	500
LT	
P	
L	
T	
Φ	



+ Q3 Lattice

- Frequency distribution of group-by queries
  - $\{PTL\ (0.05), PL\ (0.25), PT\ (0.15), LT\ (0.1), P\ (0.2), L\ (0.1), T\ (0.1), Ø\ (0.05)\}$
- **What are the first three cuboids that should be materialized in order to minimize total query cost, and why?**
  - $\{PTL,\ PL\}$
  - How to choose the third one?
    - Consider both PL and PLT when calculating the benefit of materializing a new cuboid.



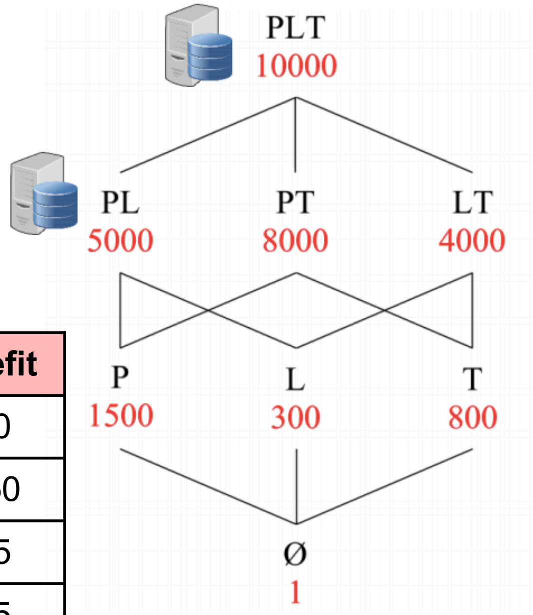
Choose LT

Queries	Previous	Current	Weight	Benefit
LT	10000	4000	0.1	600
L	5000	4000	0.1	100
T	10000	4000	0.1	600
Φ	5000	4000	0.05	50

Cuboid	Benefit
PT	500
LT	1350
P	
L	
T	
Φ	

+ Q3 Lattice

- Frequency distribution of group-by queries
  - $\{PTL\ (0.05), PL\ (0.25), PT\ (0.15), LT\ (0.1), P\ (0.2), L\ (0.1), T\ (0.1), Ø\ (0.05)\}$
- **What are the first three cuboids that should be materialized in order to minimize total query cost, and why?**
  - $\{PTL, PL\}$
  - How to choose the third one?
    - Consider both PL and PLT when calculating the benefit of materializing a new cuboid.



Choose P

Queries	Previous	Current	Weight	Benefit
P	5000	1500	0.2	700
Φ	5000	1500	0.05	175

Cuboid	Benefit
PT	500
LT	1350
P	875
L	705
T	1130
Φ	249.95

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA