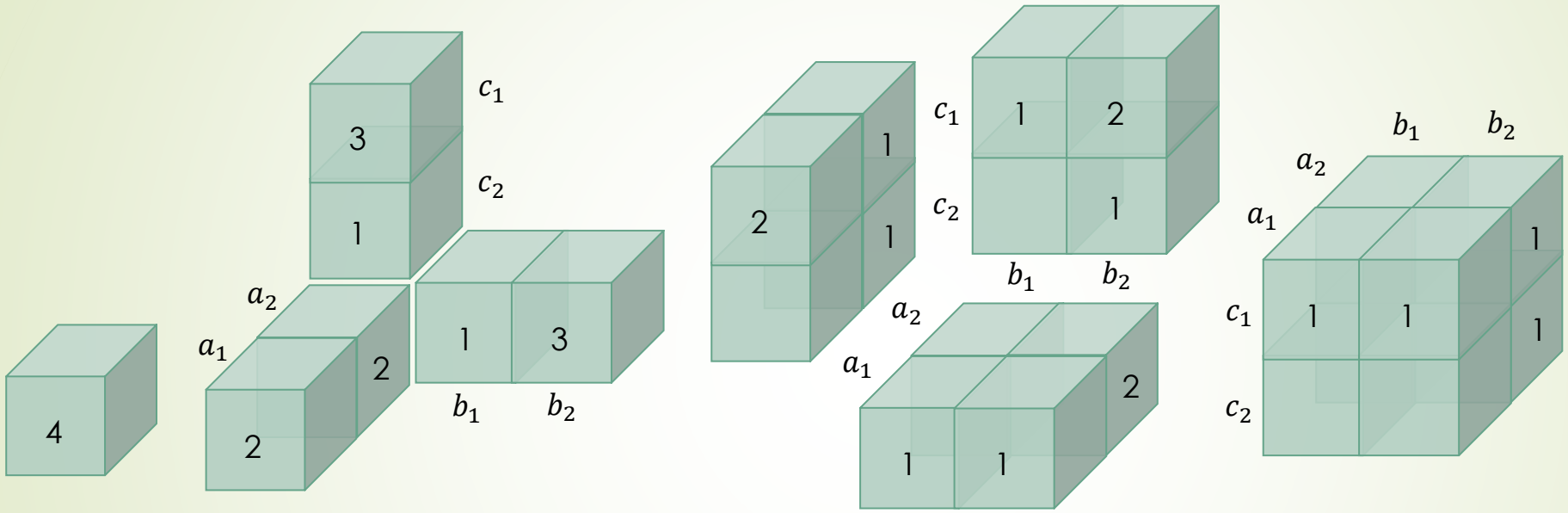
香港城市大學
City University of Hong Kong
專業 創新 胸懷全球
Professional·Creative
For The World

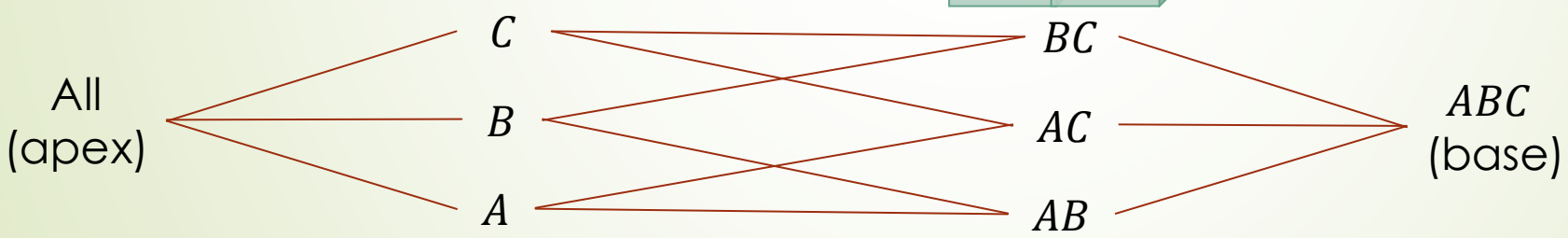
Data Cube Computation:
Bottom-up Construction

CS5483 Data Warehousing and Data Mining

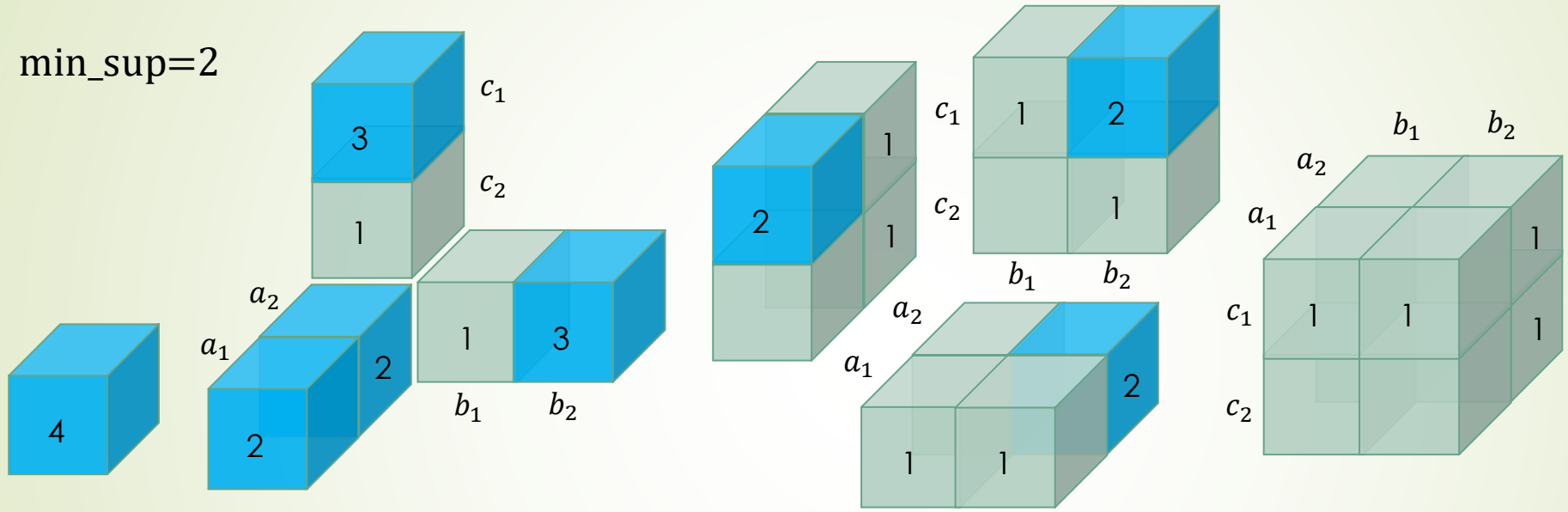
Data cube for count



A	B	C
a_2	b_2	c_2
a_1	b_2	c_1
a_2	b_2	c_1
a_1	b_1	c_1



Iceberg cube



A	B	C
a_2	b_2	c_2
a_1	b_2	c_1
a_2	b_2	c_1
a_1	b_1	c_1

- **Iceberg cube**: Keep only cells with the **iceberg condition** counts ≥ min_sup.
- How to compute the iceberg cube efficiently without computing the full cube?

Apriori algorithm
Initialization

A	B	C
a_2	b_2	c_2
a_1	b_2	c_1
a_2	b_2	c_1
a_1	b_1	c_1

min_sup=2

C_1

$(a_1,*,*)$	2
$(a_2,*,*)$	2
$(*,b_1,*)$	1
$(*,b_2,*)$	3
$(*,*,c_1)$	3
$(*,*,c_2)$	1

1-D cells in iceberg

$(a_1,*,*)$	2
$(a_2,*,*)$	2
$(*,b_2,*)$	3
$(*,*,c_1)$	3

- Create C_1 by computing the cells in 1-D cuboids.
- Create 1-D iceberg cuboids using cells in C_1 that satisfy the iceberg condition.

Apriori algorithm
Join and Prune

C_3?

A	B	C
a_2	b_2	c_2
a_1	b_2	c_1
a_2	b_2	c_1
a_1	b_1	c_1

min_sup=2

C_2

$(a_1, b_2, *)$	1
$(a_1, *, c_1)$	2
$(a_2, b_2, *)$	2
$(a_2, *, c_1)$	1
$(*, b_2, c_1)$	2

2-D cells in iceberg

$(a_1, *, c_1)$	2
$(a_2, b_2, *)$	2
$(*, b_2, c_1)$	2

1-D cells in iceberg

$(a_1, *, *)$	2
$(a_2, *, *)$	2
$(*, b_2, *)$	3
$(*, *, c_1)$	3

0-D apex cell

$(*, *, *)$	4

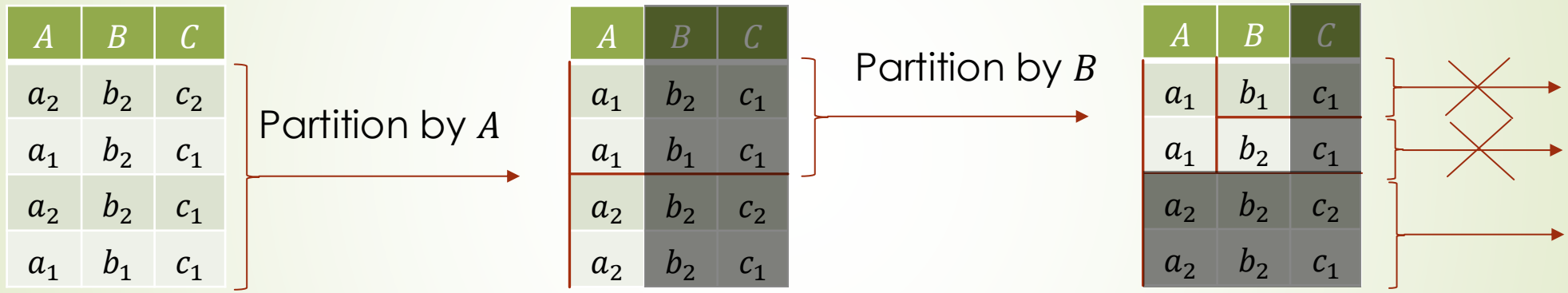
- **Join step:** Create C_k by Joining two cells in $(k-1)$-D iceberg cuboids differing only in the last non-star attributes.
  - The first _____non-star attributes and their values must be identical.
  - The last non-star attributes must be different. Why? _____
- **Prune step:** Remove a cell from C_k if any of its _____ is not in the iceberg cube. (Try to create an example.)

Apriori algorithm
Limitations

- Computing the counts can be slow with a large database.
- The candidate list can be long and may not fit in memory.

Database Partition

- Partition the data tuples by the different attribute values of a dimension.



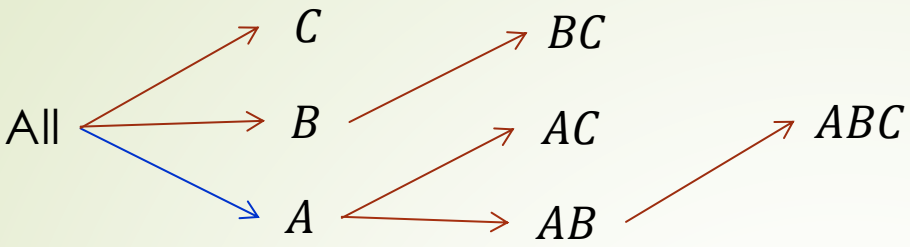
$\text{count}(a_1,*,*) = \text{count}(a_2,*,*) = 2$ $\text{count}(a_1, b_1,*) = \text{count}(a_1, b_2,*) = 1$

- (Optional) Can be Implemented using counting sort: $O(n + k_i)$ time where
  - n: Number of tuples to partition
  - k_i: Cardinality of dimension i to partition.

Bottom-Up Construction (BUC)

- BUC(all_data,first_dim)
- See Han11, p.202, Figure 5.6.

```
Input: input data (input) corresponding to a cell,
and starting dimension (dim) for the search.
Output: All cells in the iceberg cube along
dimensions starting from dim.
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
    write the cell to output.
    For each dimension from dim onwards
        Partition input by the dimension
        For each partition
            BUC(partition,next_dimension)
```



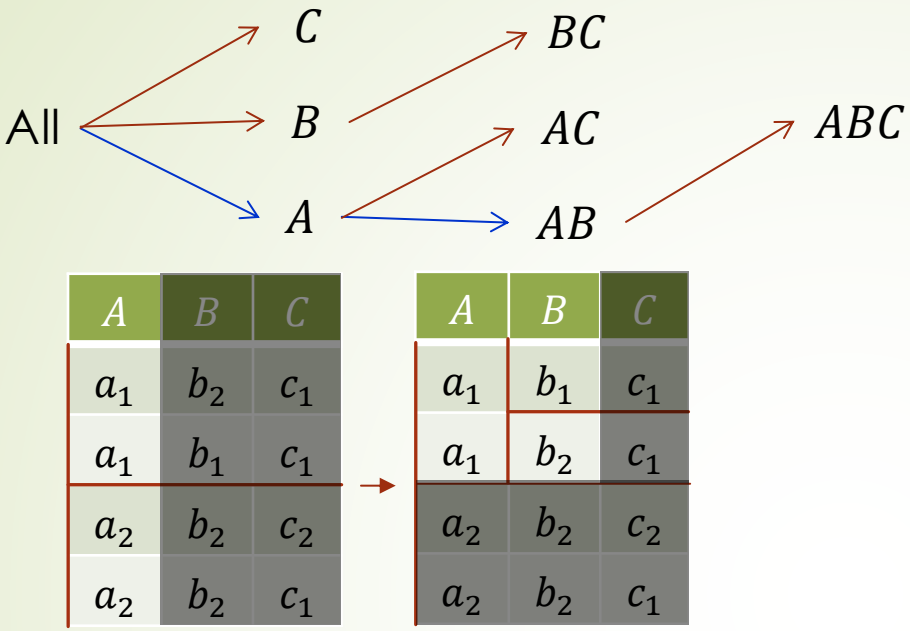
A	B	C
a_2	b_2	c_2
a_1	b_2	c_1
a_2	b_2	c_1
a_1	b_1	c_1

→

A	B	C
a_1	b_2	c_1
a_1	b_1	c_1
a_2	b_2	c_2
a_2	b_2	c_1

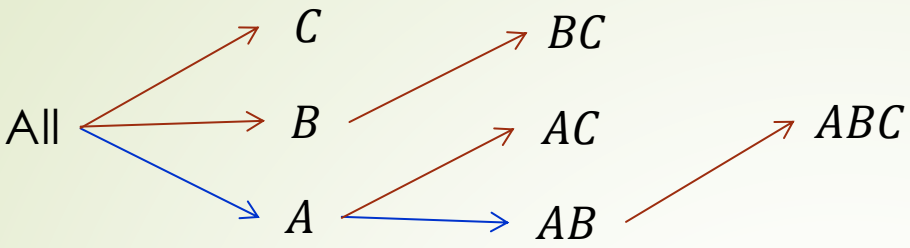
Cell	Count	In iceberg?
(*,*,*)	4	Y

```
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
    write the cell to output.
    For each dimension from dim onwards
        Partition input by the dimension
        For each partition
            BUC(partition,next_dimension)
```



Cell	Count	In iceberg?
$(*,*,*)$	4	Y
$(a_1,*,*)$	2	Y

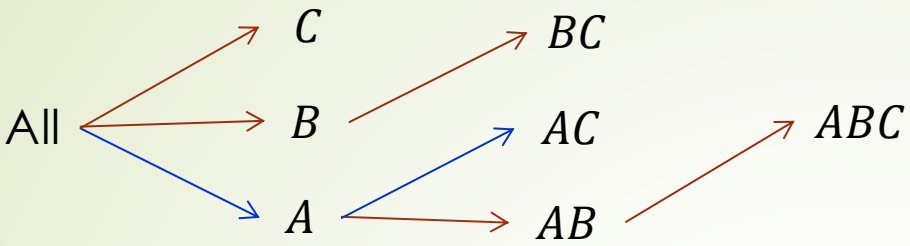
```
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
    write the cell to output.
    For each dimension from dim onwards
        Partition input by the dimension
        For each partition
            BUC(partition,next_dimension)
```



A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_2	c_2
a_2	b_2	c_1

Cell	Count	In iceberg?
$(*,*,*)$	4	Y
$(a_1,*,*)$	2	Y
$(a_1,b_1,*)$	1	N
$(a_1,b_2,*)$	1	N

```
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
    write the cell to output.
    For each dimension from dim onwards
        Partition input by the dimension
        For each partition
            BUC(partition,next_dimension)
```



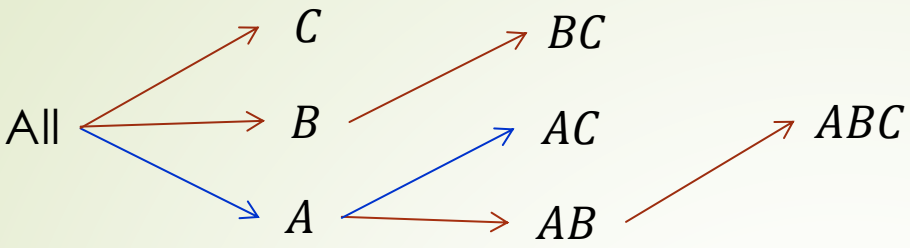
A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_2	c_2
a_2	b_2	c_1

→

A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_2	c_2
a_2	b_2	c_1

Cell	Count	In iceberg?
$(*,*,*)$	4	Y
$(a_1,*,*)$	2	Y
$(a_1,b_1,*)$	1	N
$(a_1,b_2,*)$	1	N

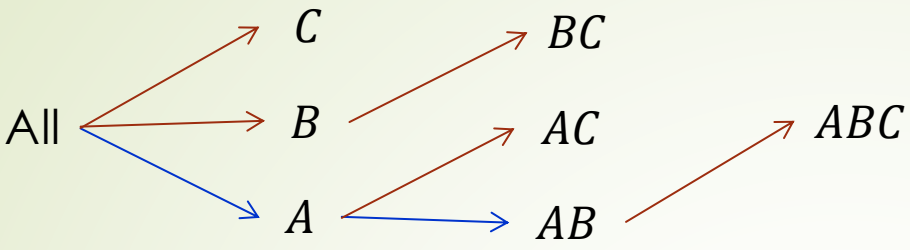
```
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
    write the cell to output.
    For each dimension from dim onwards
        Partition input by the dimension
        For each partition
            BUC(partition,next_dimension)
```



A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_2	c_2
a_2	b_2	c_1

Cell	Count	In iceberg?
$(*,*,*)$	4	Y
$(a_1,*,*)$	2	Y
$(a_1,b_1,*)$	1	N
$(a_1,b_2,*)$	1	N
$(a_1,*,c_1)$	2	Y
$(a_1,*,c_2)$	0	N

```
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
    write the cell to output.
    For each dimension from dim onwards
        Partition input by the dimension
        For each partition
            BUC(partition,next_dimension)
```



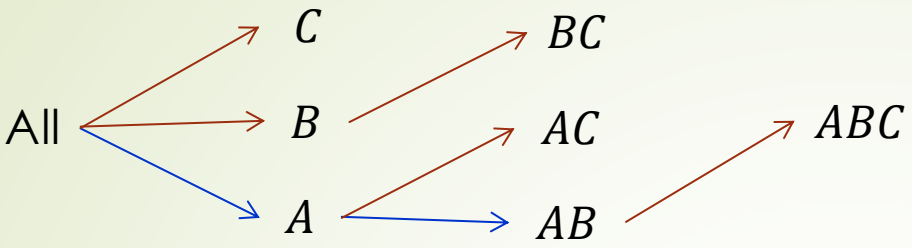
A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_2	c_2
a_2	b_2	c_1

→

A	B	C
a_1	b_2	c_1
a_1	b_1	c_1
a_2	b_2	c_2
a_2	b_2	c_1

Cell	Count	In iceberg?
$(*,*,*)$	4	Y
$(a_1,*,*)$	2	Y
$(a_1,b_1,*)$	1	N
$(a_1,b_2,*)$	1	N
$(a_1,*,c_1)$	2	Y
$(a_1,*,c_2)$	0	N
$(a_2,*,*)$	2	Y

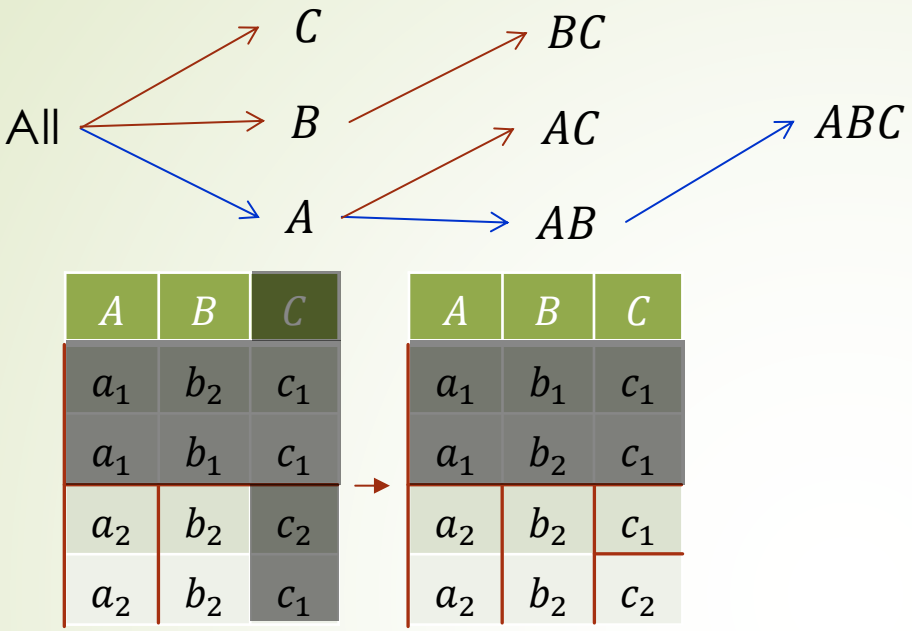
```
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
    write the cell to output.
    For each dimension from dim onwards
        Partition input by the dimension
        For each partition
            BUC(partition,next_dimension)
```



A	B	C
a_1	b_2	c_1
a_1	b_1	c_1
a_2	b_2	c_2
a_2	b_2	c_1

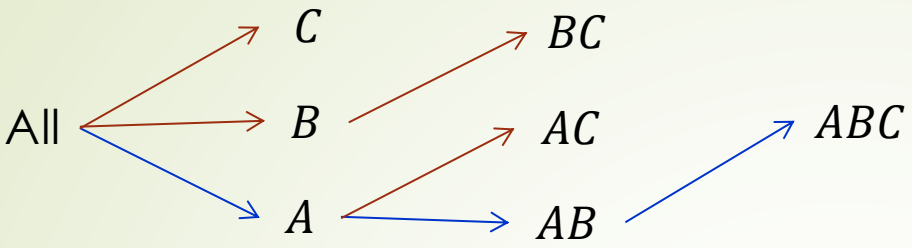
Cell	Count	In iceberg?
$(*,*,*)$	4	Y
$(a_1,*,*)$	2	Y
$(a_1,b_1,*)$	1	N
$(a_1,b_2,*)$	1	N
$(a_1,*,c_1)$	2	Y
$(a_1,*,c_2)$	0	N
$(a_2,*,*)$	2	Y
$(a_2,b_1,*)$	0	N
$(a_2,b_2,*)$	2	Y

```
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
    write the cell to output.
    For each dimension from dim onwards
        Partition input by the dimension
        For each partition
            BUC(partition,next_dimension)
```




```
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
     write the cell to output.
    For each dimension from dim onwards
         Partition input by the dimension
         For each partition
              BUC(partition,next_dimension)
```

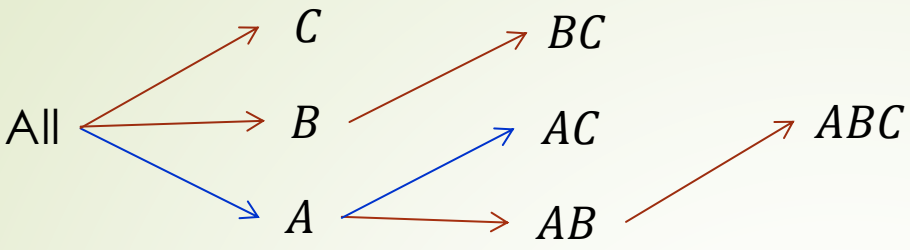
Cell	Count	In iceberg?
$(*,*,*)$	4	Y
$(a_1,*,*)$	2	Y
$(a_1,b_1,*)$	1	N
$(a_1,b_2,*)$	1	N
$(a_1,*,c_1)$	2	Y
$(a_1,*,c_2)$	0	N
$(a_2,*,*)$	2	Y
$(a_2,b_1,*)$	0	N
$(a_2,b_2,*)$	2	Y



A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_2	c_1
a_2	b_2	c_2

```
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
    write the cell to output.
    For each dimension from dim onwards
        Partition input by the dimension
        For each partition
            BUC(partition,next_dimension)
```

Cell	Count	In iceberg?
$(*,*,*)$	4	Y
$(a_1,*,*)$	2	Y
$(a_1,b_1,*)$	1	N
$(a_1,b_2,*)$	1	N
$(a_1,*,c_1)$	2	Y
$(a_1,*,c_2)$	0	N
$(a_2,*,*)$	2	Y
$(a_2,b_1,*)$	0	N
$(a_2,b_2,*)$	2	Y
(a_2,b_2,c_1)	1	N
(a_2,b_2,c_2)	1	N



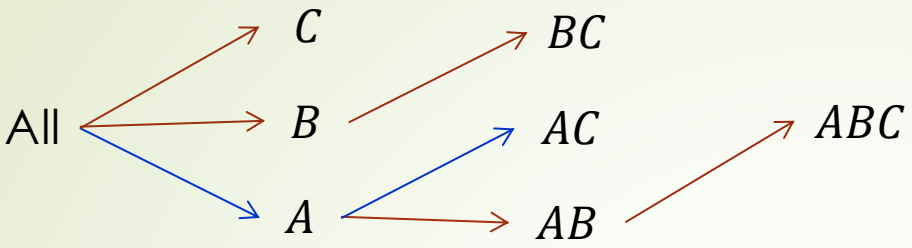
A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_2	c_2
a_2	b_2	c_1

→

A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_2	c_2
a_2	b_2	c_1

```
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
    write the cell to output.
    For each dimension from dim onwards
        Partition input by the dimension
        For each partition
            BUC(partition,next_dimension)
```

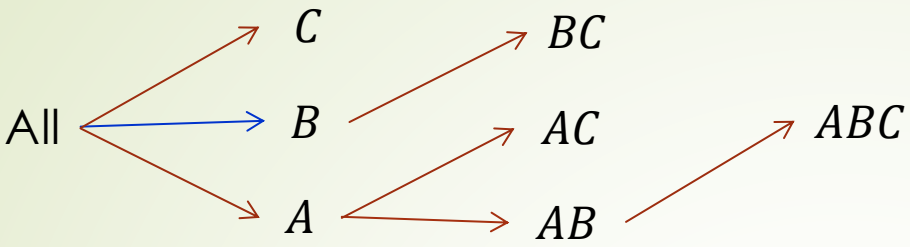
Cell	Count	In iceberg?
$(*,*,*)$	4	Y
$(a_1,*,*)$	2	Y
$(a_1,b_1,*)$	1	N
$(a_1,b_2,*)$	1	N
$(a_1,*,c_1)$	2	Y
$(a_1,*,c_2)$	0	N
$(a_2,*,*)$	2	Y
$(a_2,b_1,*)$	0	N
$(a_2,b_2,*)$	2	Y
(a_2,b_2,c_1)	1	N
(a_2,b_2,c_2)	1	N



A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_2	c_2
a_2	b_2	c_1

```
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
    write the cell to output.
    For each dimension from dim onwards
        Partition input by the dimension
        For each partition
            BUC(partition,next_dimension)
```

Cell	Count	In iceberg?
$(*,*,*)$	4	Y
$(a_1,*,*)$	2	Y
$(a_1,b_1,*)$	1	N
$(a_1,b_2,*)$	1	N
$(a_1,*,c_1)$	2	Y
$(a_1,*,c_2)$	0	N
$(a_2,*,*)$	2	Y
$(a_2,b_1,*)$	0	N
$(a_2,b_2,*)$	2	Y
(a_2,b_2,c_1)	1	N
(a_2,b_2,c_2)	1	N
$(a_2,*,c_1)$	1	N
$(a_2,*,c_2)$	1	N



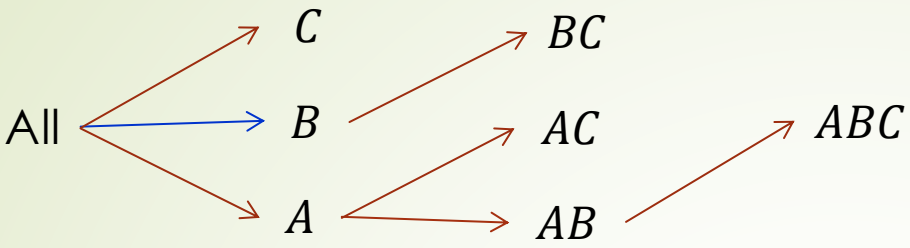
Cell	Count	In iceberg?
⋮	⋮	⋮

A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_2	c_1
a_2	b_2	c_2

→

A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_2	c_1
a_2	b_2	c_2

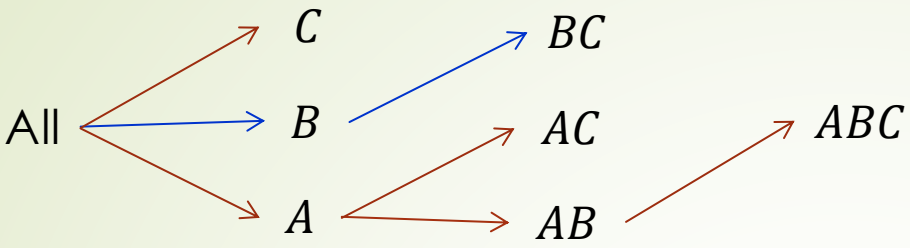
```
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
    write the cell to output.
    For each dimension from dim onwards
        Partition input by the dimension
        For each partition
            BUC(partition,next_dimension)
```



A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_2	c_1
a_2	b_2	c_2

Cell	Count	In iceberg?
⋮	⋮	⋮
$(*, b_1, *)$	1	N
$(*, b_2, *)$	3	Y

```
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
    write the cell to output.
    For each dimension from dim onwards
        Partition input by the dimension
        For each partition
            BUC(partition,next_dimension)
```



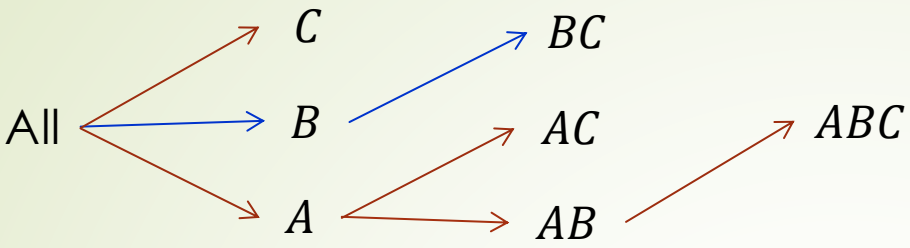
A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_2	c_1
a_2	b_2	c_2

→

A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_2	c_1
a_2	b_2	c_2

Cell	Count	In iceberg?
⋮	⋮	⋮
$(*, b_1, *)$	1	N
$(*, b_2, *)$	3	Y

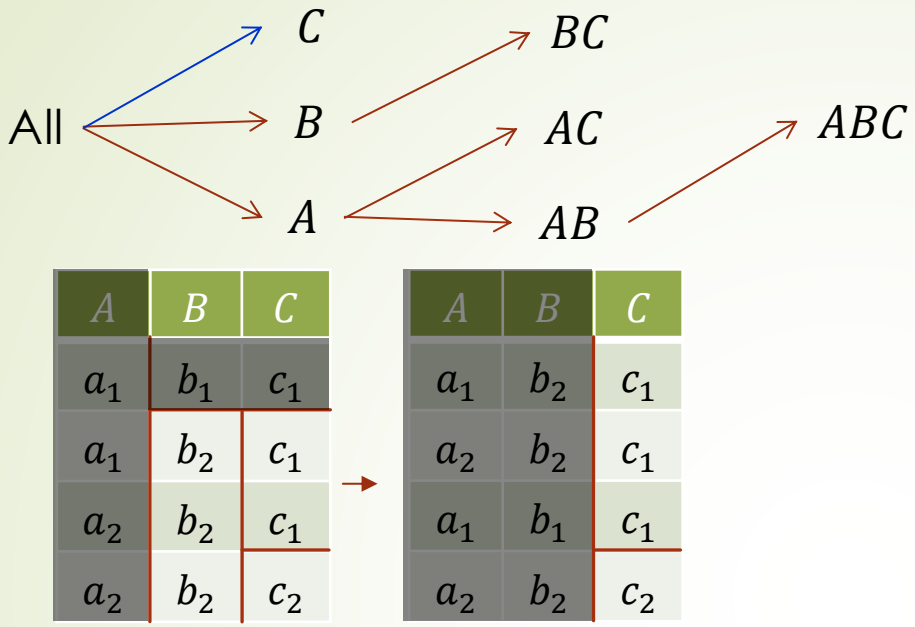
```
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
    write the cell to output.
    For each dimension from dim onwards
        Partition input by the dimension
        For each partition
            BUC(partition,next_dimension)
```



A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_2	c_1
a_2	b_2	c_2

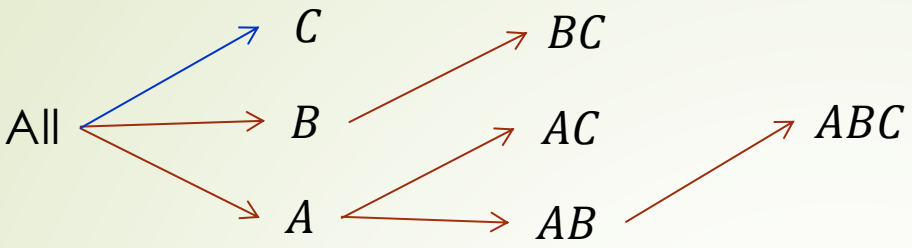
Cell	Count	In iceberg?
⋮	⋮	⋮
$(*, b_1, *)$	1	N
$(*, b_2, *)$	3	Y
$(*, b_2, c_1)$	2	Y
$(*, b_2, c_2)$	1	N

```
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
    write the cell to output.
    For each dimension from dim onwards
        Partition input by the dimension
        For each partition
            BUC(partition,next_dimension)
```



Cell	Count	In iceberg?
⋮	⋮	⋮
$(*, b_1, *)$	1	N
$(*, b_2, *)$	3	Y
$(*, b_2, c_1)$	2	Y
$(*, b_2, c_2)$	1	N

```
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
    write the cell to output.
    For each dimension from dim onwards
        Partition input by the dimension
        For each partition
            BUC(partition,next_dimension)
```

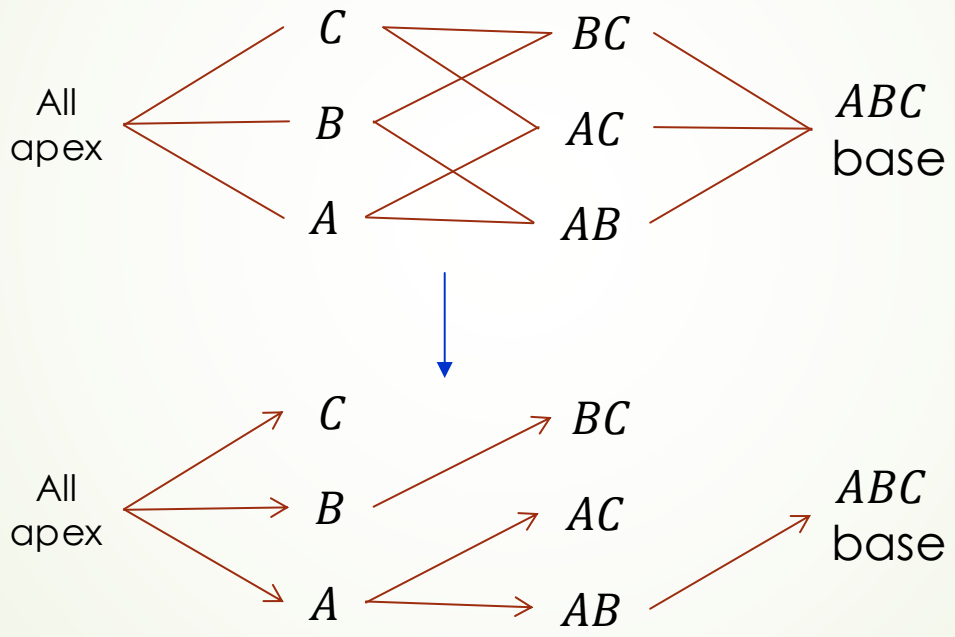


A	B	C
a_1	b_2	c_1
a_2	b_2	c_1
a_1	b_1	c_1
a_2	b_2	c_2

Cell	Count	In iceberg?
⋮	⋮	⋮
$(*, b_1, *)$	1	N
$(*, b_2, *)$	3	Y
$(*, b_2, c_1)$	2	Y
$(*, b_2, c_2)$	1	N
$(*, *, c_1)$	3	Y
$(*, *, c_2)$	1	N

```
Procedure BUC(input,dim)
Compute count of input
If iceberg condition holds
    write the cell to output.
    For each dimension from dim onwards
        Partition input by the dimension
        For each partition
            BUC(partition,next_dimension)
```

Final question
Why called bottom-up?



References

- 5.1 Data Cube Computation: Preliminary Concepts
- 5.2.2 BUC: Computing Iceberg Cubes from the Apex Cuboid Downward
- Optional:
  - Xin, Dong, et al. "Star-cubing: Computing iceberg cubes by top-down and bottom-up integration." Proceedings 2003 VLDB Conference. Morgan Kaufmann, 2003.
  - D. Xin, J. Han, X. Li, Z. Shao and B. W. Wah, "Computing Iceberg Cubes by Top-Down and Bottom-Up Integration: The StarCubing Approach," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 1, pp. 111-126, Jan. 2007.