Data Mining:

Concepts and Techniques

(3rd ed.)

— Chapter 5 —

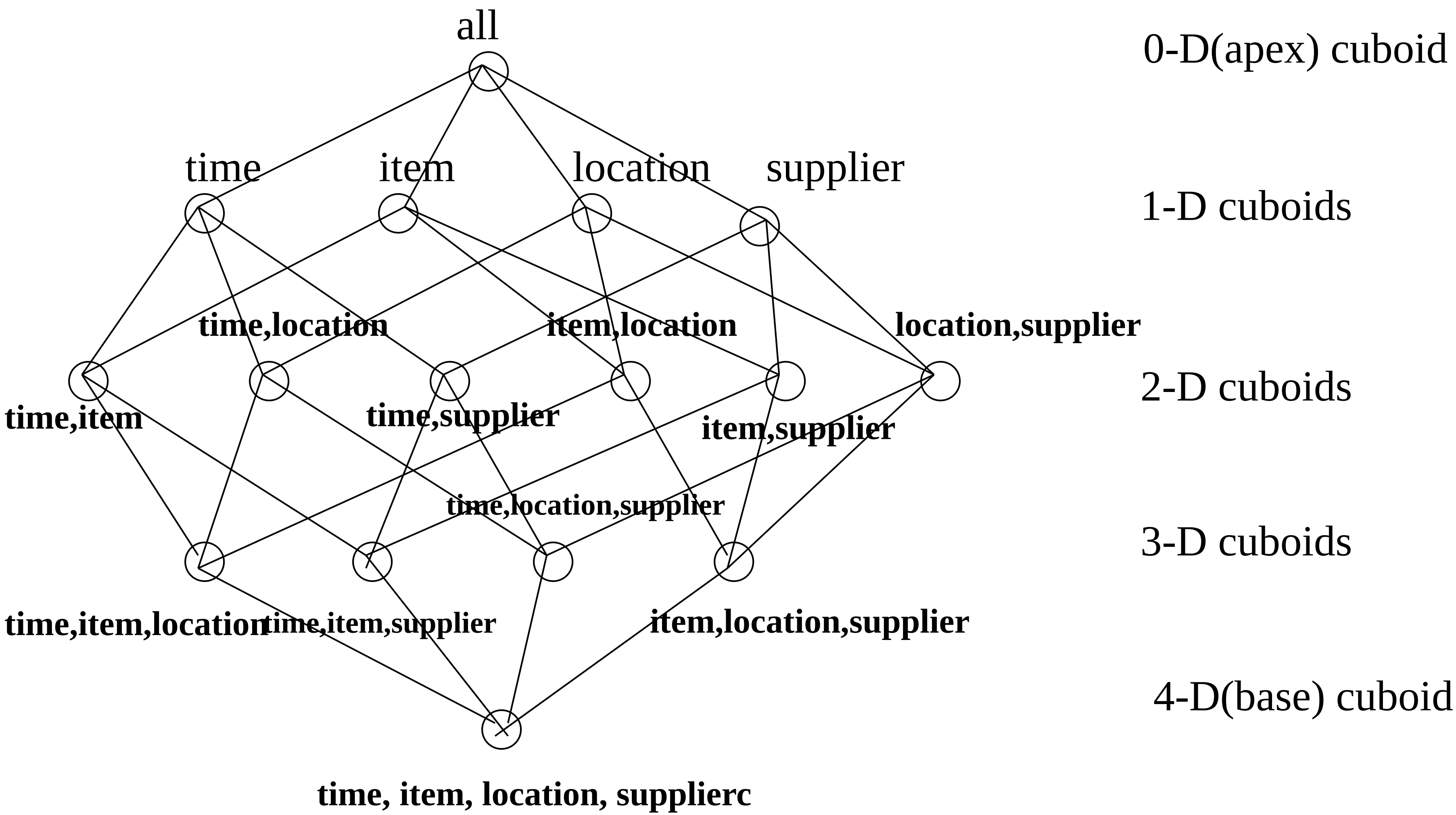
Xike Xie

Slides are based on Jiawei Han's work.

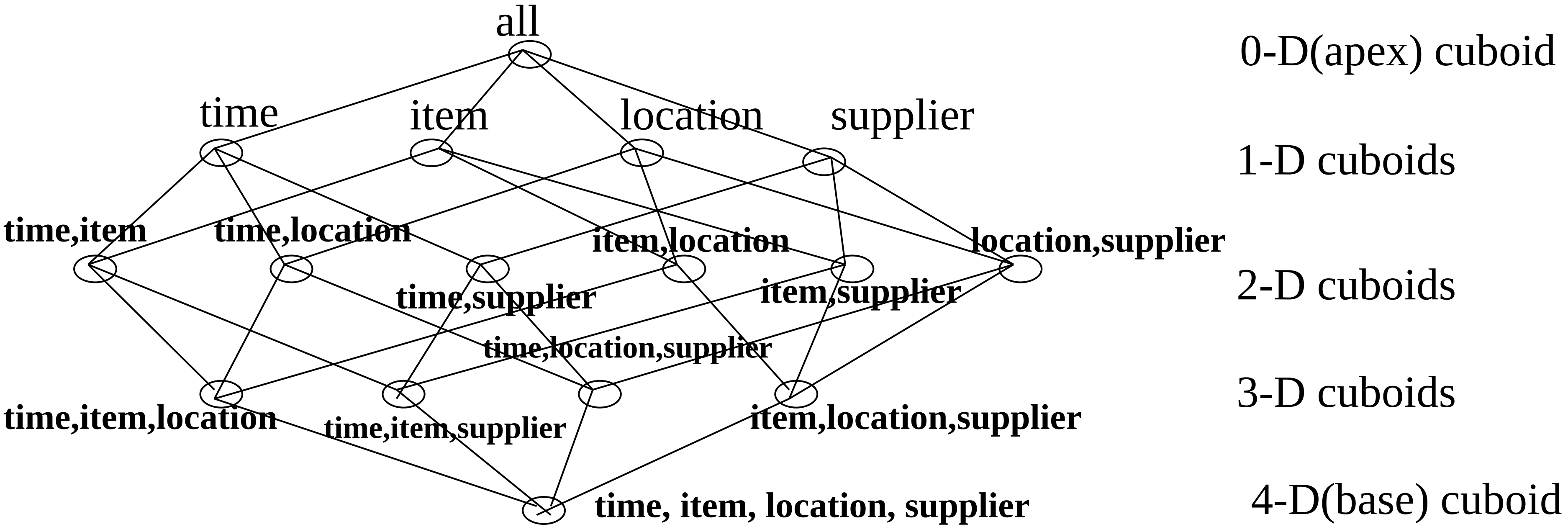
Chapter 5: Data Cube Technology

- Data Cube Computation: Preliminary Concepts
- Data Cube Computation Methods
- Processing Advanced Queries by Exploring Data Cube Technology
- Multidimensional Data Analysis in Cube Space
- Summary

Data Cube: A Lattice of Cuboids

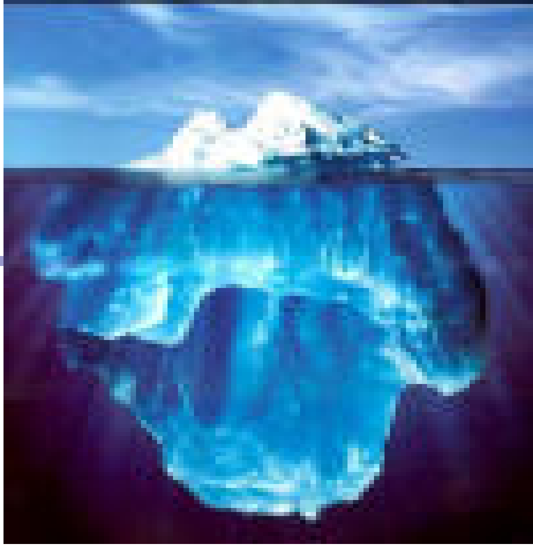


Data Cube: A Lattice of Cuboids



- Base vs. aggregate cells; ancestor vs. descendant cells; parent vs. child cells
  1. (9/15, milk, Urbana, Dairy_land)
  2. (9/15, milk, Urbana, *)
  3. (*, milk, Urbana, *)
  4. (*, milk, Urbana, *)
  5. (*, milk, Chicago, *)
  6. (*, milk, *, *)

Cube Materialization: Full Cube vs. Iceberg Cube



- Full cube vs. iceberg cube

```
compute cube sales iceberg as
select month, city, customer group, count(*)
from salesInfo
cube by month, city, customer group
having count(*) >= min support
```

iceberg condition

- Computing *only* the cuboid cells whose measure satisfies the iceberg condition
- Only a small portion of cells may be "above the water'' in a sparse cube
- Avoid explosive growth: A cube with 100 dimensions
  - 2 base cells: (a1, a2, …., a100), (b1, b2, …, b100)
  - How many aggregate cells if "having count >= 1"?
  - What about "having count >= 2"?

Iceberg Cube, Closed Cube & Cube Shell

- Is iceberg cube good enough?
  - 2 base cells: $\{(a_1, a_2, a_3 \ldots, a_{100}):10, (a_1, a_2, b_3, \ldots, b_{100}):10\}$
  - How many cells will the iceberg cube have if having count(*) >= 10? Hint: A huge but tricky number!
- Close cube:
  - Closed cell c: if there exists no cell d, s.t. d is a descendant of c, and d has the same measure value as c.
  - Closed cube: a cube consisting of only closed cells
  - What is the closed cube of the above base cuboid? Hint: only 3 cells
- Cube Shell
  - Precompute only the cuboids involving a small # of dimensions, e.g., 3

    For $(A_1, A_2, \ldots A_{10})$, how many combinations to compute?
  - More dimension combinations will need to be computed on the fly

Roadmap for Efficient Computation

- General cube computation heuristics (Agarwal et al.'96)
- Computing full/iceberg cubes: 3 methodologies
  - Bottom-Up: Multi-Way array aggregation (Zhao, Deshpande & Naughton, SIGMOD'97)
  - Top-down:
    - BUC (Beyer & Ramarkrishnan, SIGMOD'99)
    - H-cubing technique (Han, Pei, Dong & Wang: SIGMOD'01)
  - Integrating Top-Down and Bottom-Up:
    - Star-cubing algorithm (Xin, Han, Li & Wah: VLDB'03)
- High-dimensional OLAP: A Minimal Cubing Approach (Li, et al. VLDB'04)
- Computing alternative kinds of cubes:
  - Partial cube, closed cube, approximate cube, etc.

General Heuristics (Agarwal et al. VLDB'96)

- Sorting, hashing, and grouping operations are applied to the dimension attributes in order to reorder and cluster related tuples
- Aggregates may be computed from previously computed aggregates, rather than from the base fact table
  - **Smallest-child:** computing a cuboid from the smallest, previously computed cuboid
  - **Cache-results:** caching results of a cuboid from which other cuboids are computed to reduce disk I/Os
  - **Amortize-scans:** computing as many as possible cuboids at the same time to amortize disk reads
  - **Share-sorts:** sharing sorting costs cross multiple cuboids when sort-based method is used
  - **Share-partitions:** sharing the partitioning cost across multiple cuboids when hash-based algorithms are used

Chapter 5: Data Cube Technology

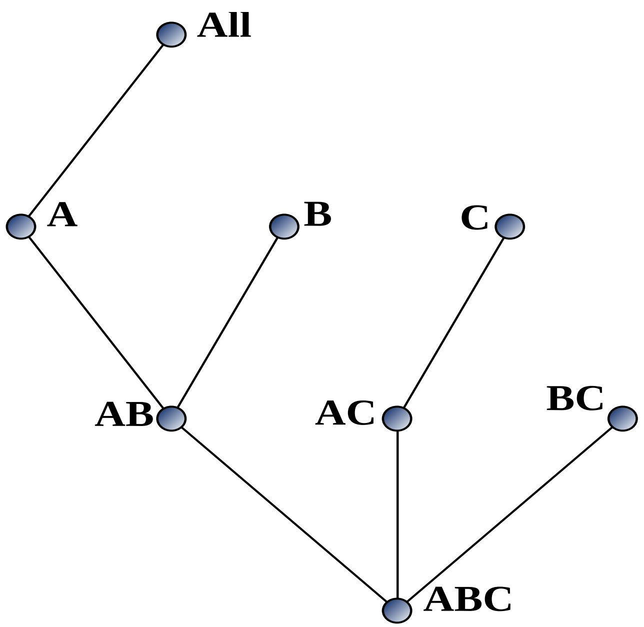
- Data Cube Computation: Preliminary Concepts
- Data Cube Computation Methods
- Processing Advanced Queries by Exploring Data Cube Technology
- Multidimensional Data Analysis in Cube Space
- Summary

Data Cube Computation Methods

- Multi-Way Array Aggregation
- BUC
- Star-Cubing
- High-Dimensional OLAP

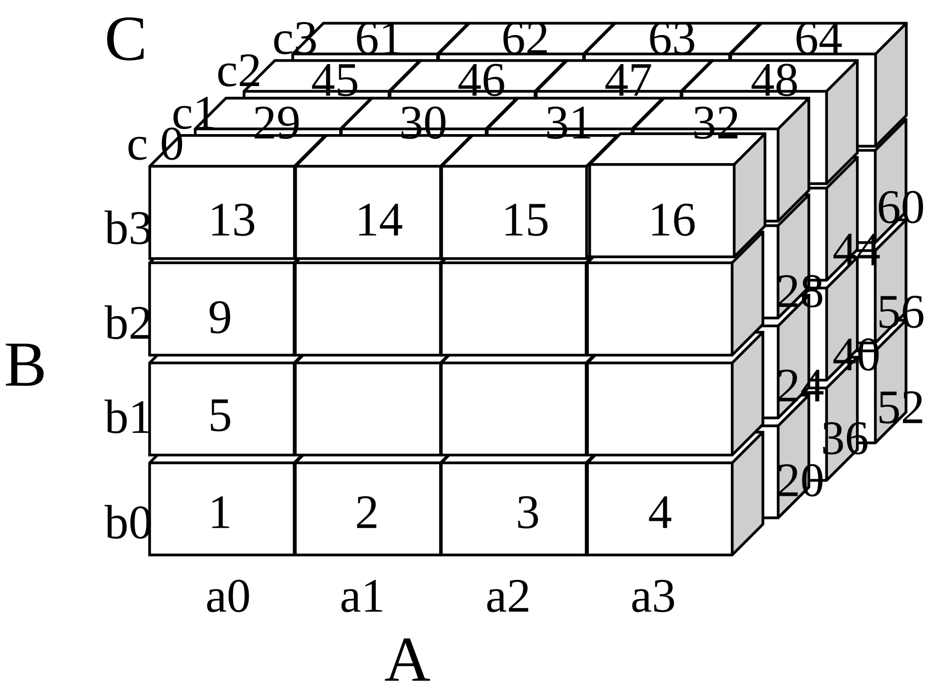
Multi-Way Array Aggregation

- Array-based "bottom-up" algorithm
- Using multi-dimensional chunks
- No direct tuple comparisons
- Simultaneous aggregation on multiple dimensions
- Intermediate aggregate values are re-used for computing ancestor cuboids
- Cannot do *Apriori* pruning: No iceberg optimization



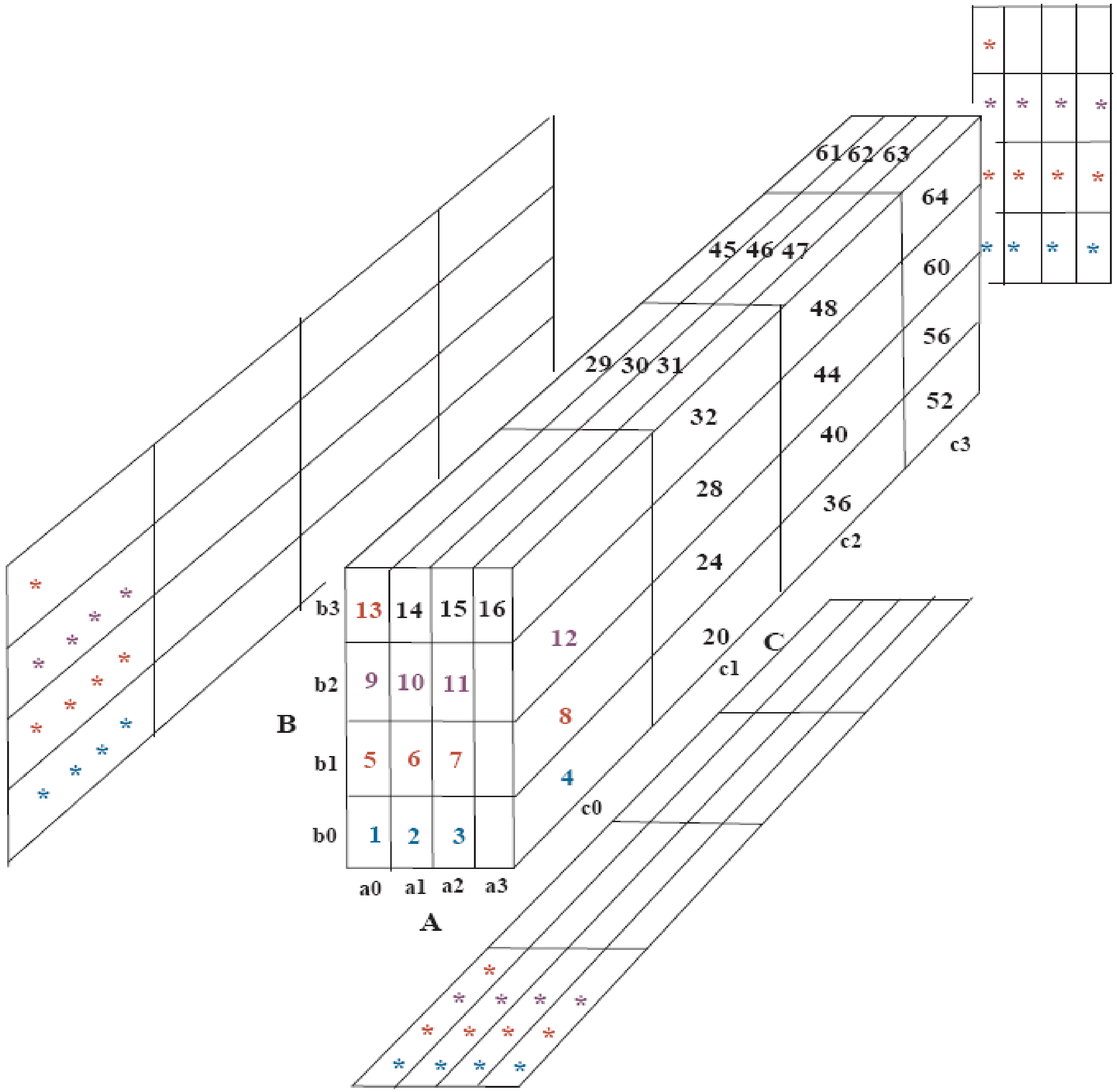
Multi-way Array Aggregation for Cube Computation (MOLAP)

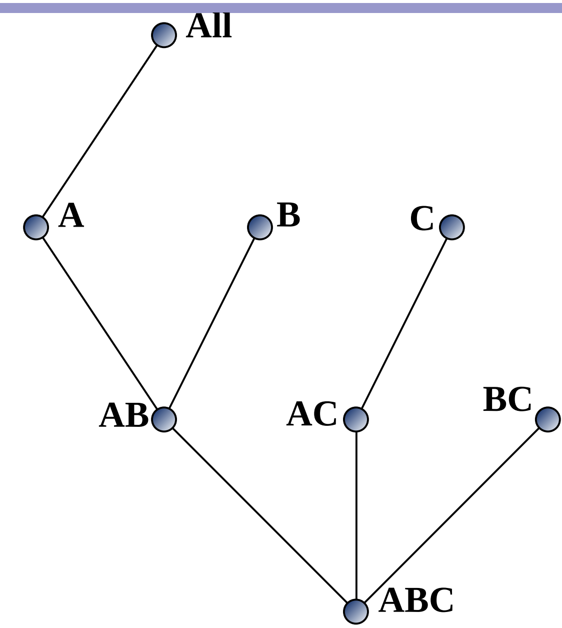
- Partition arrays into chunks (a small subcube which fits in memory).
- Compressed sparse array addressing: (chunk_id, offset)
- Compute aggregates in “multiway” by visiting cube cells in the order which minimizes the # of times to visit each cell, and reduces memory access and storage cost.



What is the best traversing order to do multi-way aggregation?

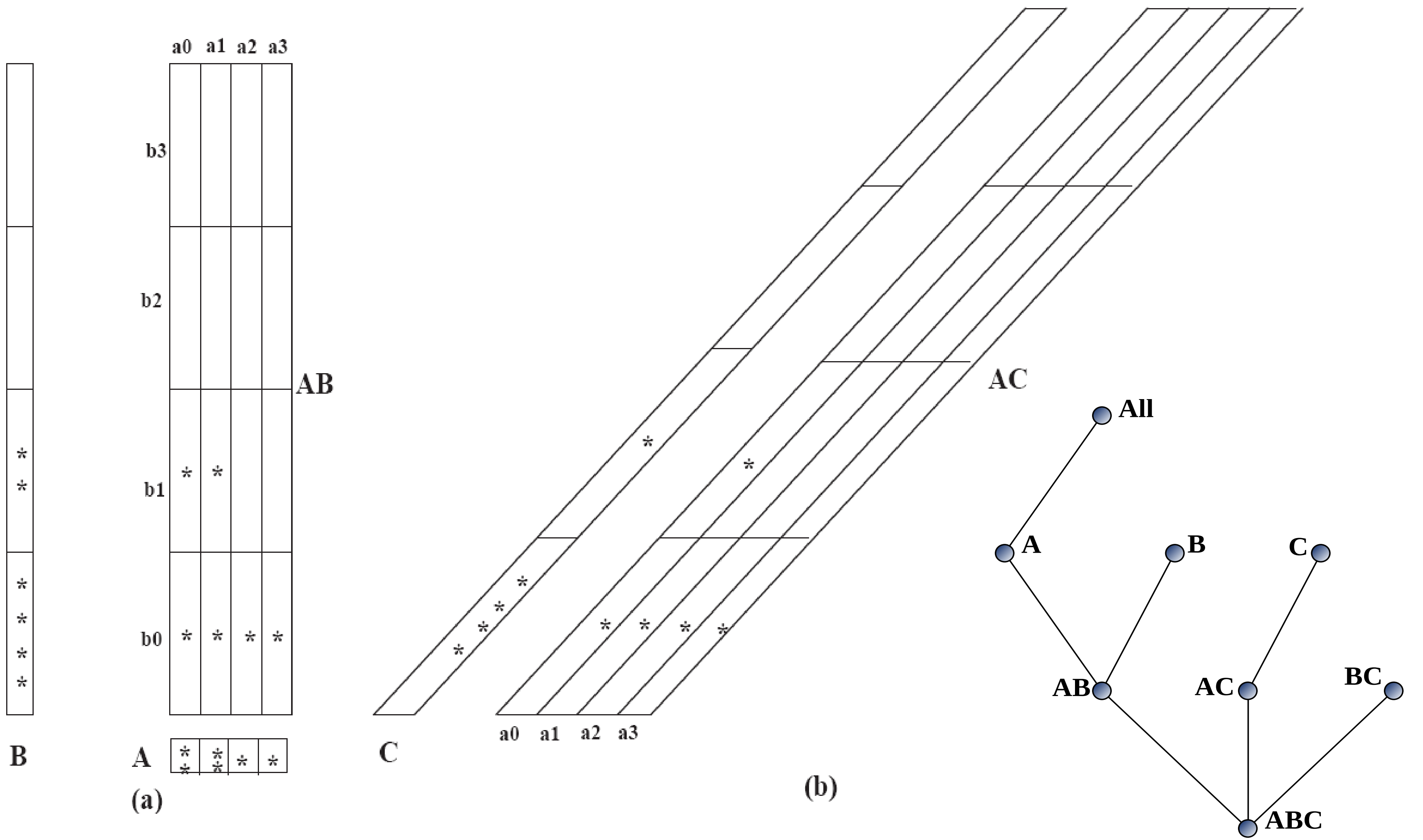
Multi-way Array Aggregation for Cube Computation (3-D to 2-D)





- The best order is the one that minimizes the memory requirement and reduced I/Os

Multi-way Array Aggregation for Cube Computation (2-D to 1-D)



Multi-Way Array Aggregation for Cube Computation (Method Summary)

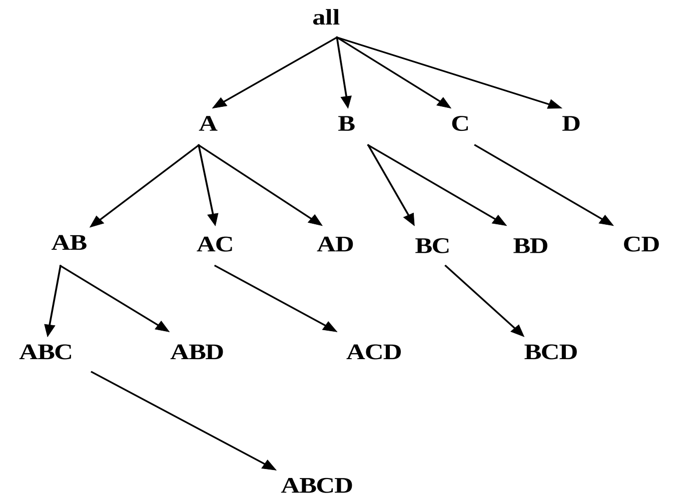
- Method: the planes should be sorted and computed according to their size in ascending order
  - Idea: keep the smallest plane in the main memory, fetch and compute only one chunk at a time for the largest plane
- Limitation of the method: computing well only for a small number of dimensions
  - If there are a large number of dimensions, "top-down" computation and iceberg cube computation methods can be explored

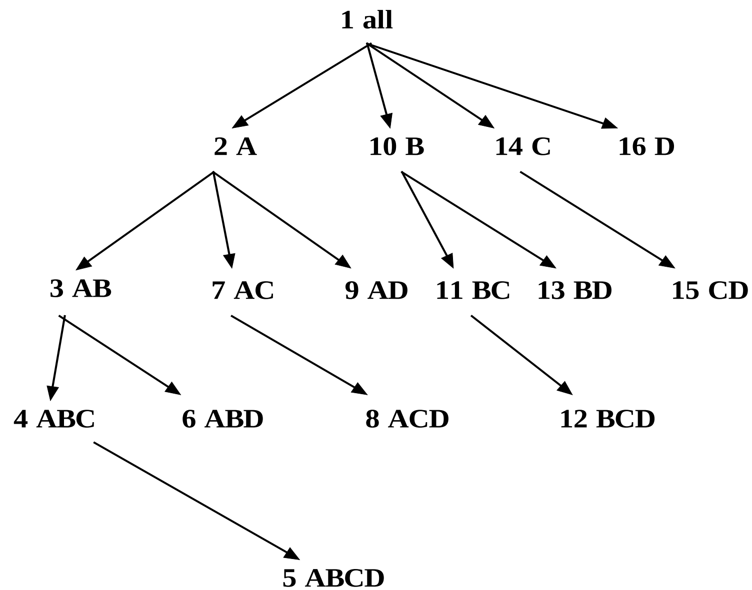
Data Cube Computation Methods

- Multi-Way Array Aggregation
- BUC
- Star-Cubing
- High-Dimensional OLAP

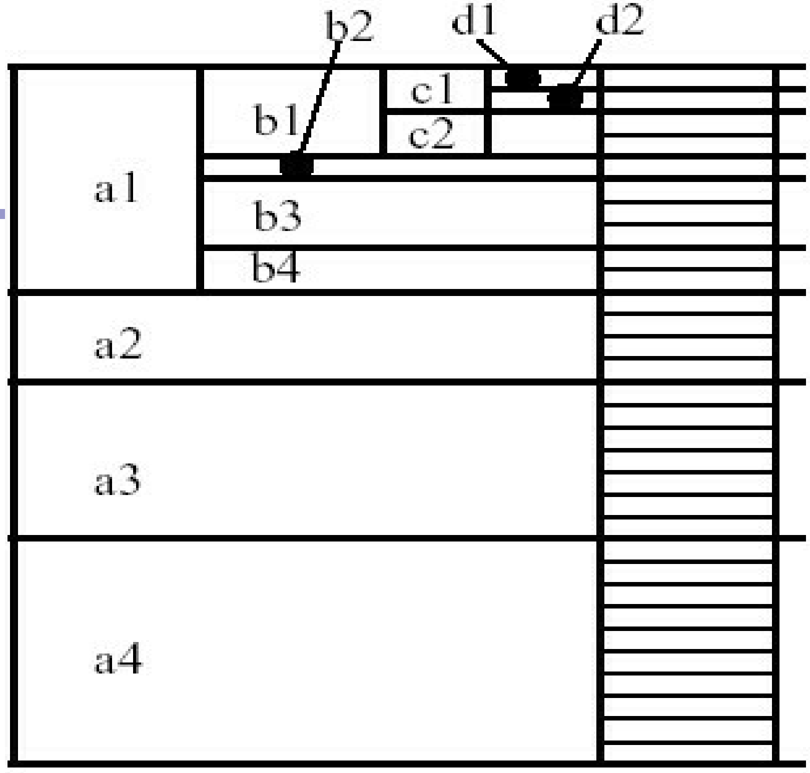
Bottom-Up Computation (BUC)

- BUC (Beyer & Ramakrishnan, SIGMOD'99)
- Bottom-up cube computation
  (Note: top-down in our view!)
- Divides dimensions into partitions and facilitates iceberg pruning
  - If a partition does not satisfy *min_sup*, its descendants can be pruned
  - If *minsup* = 1 ⇒ compute full CUBE!
- No simultaneous aggregation





BUC: Partitioning



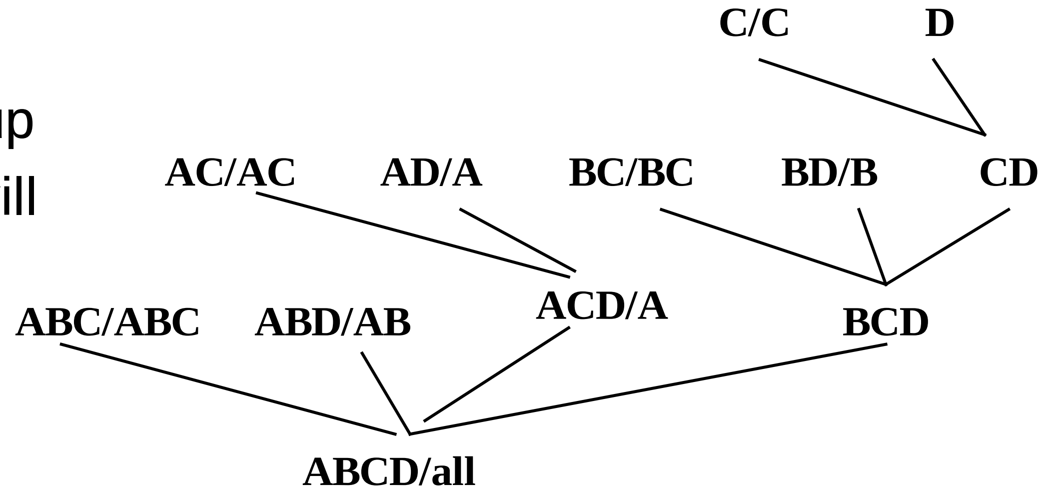
- Usually, entire data set can't fit in main memory
- Sort *distinct* values
  - partition into blocks that fit
- Continue processing
- Optimizations
  - Partitioning
    - External Sorting, Hashing, Counting Sort
  - Ordering dimensions to encourage pruning
    - Cardinality, Skew, Correlation
  - Collapsing duplicates
    - Can't do holistic aggregates anymore!

Data Cube Computation Methods

- Multi-Way Array Aggregation
- BUC
- Star-Cubing
- High-Dimensional OLAP

Star-Cubing: An Integrating Method

- D. Xin, J. Han, X. Li, B. W. Wah, Star-Cubing: Computing Iceberg Cubes by Top-Down and Bottom-Up Integration, VLDB'03
- ***Explore shared dimensions***
  - E.g., dimension A is the shared dimension of ACD and AD
  - ABD/AB means cuboid ABD has shared dimensions AB
- Allows for shared computations
  - e.g., cuboid AB is computed simultaneously as ABD
- Aggregate in a top-down manner but with the bottom-up sub-layer underneath which will allow Apriori pruning
- Shared dimensions grow in bottom-up fashion



Iceberg Pruning in Shared Dimensions

- Anti-monotonic property of shared dimensions
  - If the measure is *anti-monotonic*, and if the aggregate value on a shared dimension does not satisfy the *iceberg condition*, then all the cells extended from this shared dimension cannot satisfy the condition either
- Intuition: if we can compute the shared dimensions before the actual cuboid, we can use them to do `Apriori` pruning
- Problem: how to prune while still aggregate simultaneously on multiple dimensions?

Exercise 1

Assume a base cuboid of 10 dimensions contains only three base cells: (1) $(a_1, d_2, d_3, d_4, \ldots, d_9, d_{10})$, (2) $(d_1, b_2, d_3, d_4, \ldots, d_9, d_{10})$, and (3) $(d_1, d_2, c_3, d_4, \ldots, d_9, d_{10})$, where $a_1 \neq d_1$, $b_2 \neq d_2$, and $c_3 \neq d_3$. The measure of the cube is *count*.

(a) How many *nonempty* cuboids will a full data cube contain?

(b) How many *nonempty* aggregate (i.e., nonbase) cells will a full cube contain?

(c) How many *nonempty* aggregate cells will an iceberg cube contain if the condition of the iceberg cube is "$count \geq 2$"?

(d) A cell, c, is a *closed cell* if there exists no cell, d, such that d is a specialization of cell c (i.e., d is obtained by replacing a $*$ in c by a non-$*$ value) and d has the same measure value as c. A *closed cube* is a data cube consisting of only closed cells. How many closed cells are in the full cube?

Exercise 1 - Answer

(a) How many *nonempty* cuboids will a complete data cube contain?
2^{10}.

(b) How many *nonempty* aggregated (i.e., nonbase) cells a complete cube will contain?
(1) Each cell generates $2^{10} - 1$ nonempty aggregated cells, thus in total we should have $3 \times 2^{10} - 3$ cells with overlaps removed.
(2) We have 3×2^7 cells overlapped once (thus count 2) and 1×2^7 (which is $(*, *, *, d_4, ..., d_{10})$) overlapped twice (thus count 3). Thus we should remove in total $1 \times 3 \times 2^7 + 2 \times 1 \times 2^7 = 5 \times 2^7$ overlapped cells.
(3) Thus we have: $3 \times 8 \times 2^7 - 5 \times 2^7 - 3 = 19 \times 2^7 - 3$.

(c) How many *nonempty* aggregated cells will an iceberg cube contain if the condition of the iceberg cube is "*count* ≥ 2"?
Analysis: (1) $(*, *, d_3, d_4, \ldots, d_9, d_{10})$ has count 2 since it is generated by both cell 1 and cell 2; similarly, we have (2) $(*, d_2, *, d_4, \ldots, d_9, d_{10})$:2, (3) $(*, *, d_3, d_4, \ldots, d_9, d_{10})$:2; and (4) $(*, *, *, d_4, \ldots, d_9, d_{10})$:3. Therefore we have, $4 \times 2^7 = 2^9$.

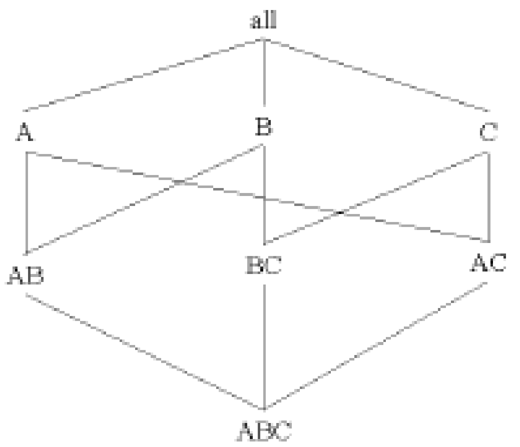
(d) A cell, c, is a *closed cell* if there exists no cell, d, such that d is a specialization of cell c (i.e., d is obtained by replacing a $*$ in c by a non-$*$ value) and d has the same measure value as c. A *closed cube* is a data cube consisting of only closed cells. How many closed cells are in the full cube?
There are seven cells, as follows
(1) $(a_1, d_2, d_3, d_4, \ldots, d_9, d_{10}) : 1$,
(2) $(d_1, b_2, d_3, d_4, \ldots, d_9, d_{10}) : 1$,
(3) $(d_1, d_2, c_3, d_4, \ldots, d_9, d_{10}) : 1$,
(4) $(*, *, d_3, d_4, \ldots, d_9, d_{10}) : 2$,
(5) $(*, d_2, *, d_4, \ldots, d_9, d_{10}) : 2$,
(6) $(d_1, *, *, d_4, \ldots, d_9, d_{10}) : 2$, and
(7) $(*, *, *, d_4, \ldots, d_9, d_{10}) : 3$.

Exercise 2

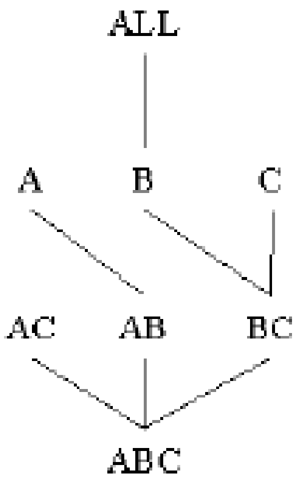
- Suppose that a base cuboid has three dimensions, *A*, *B*, *C*, with the following number of cells: /*A*/ =1, 000, 000, /*B*/ = 100, and /*C*/ = 1000.
- (a) Assuming each dimension has only one level, draw the complete lattice of the cube.
- (b) If each cube cell stores one measure with 4 bytes, what is the total size of the computed cube if the cube is *dense (each cell is non-empty)*?

Exercise 2 - Answer

(a)



(c)



(b)

The total size of the computed cube is as follows.

- all: 1
- *A*: 1,000,000; *B*: 100; *C*: 1, 000; subtotal: 1,001,100
- *AB*: 100,000,000; *BC*: 100,000; *AC*: 1,000,000,000; subtotal: 1,100,100,000
- *ABC*:100,000,000,000
- Total: 101,101,101,101 cells $\times$ 4 bytes = 404,404,404,404 bytes

Chapter 5: Data Cube Technology

- Data Cube Computation: Preliminary Concepts
- Data Cube Computation Methods
- Processing Advanced Queries by Exploring Data Cube Technology
  - Sampling Cube
  - Ranking Cube
- Multidimensional Data Analysis in Cube Space
- Summary

Processing Advanced Queries by Exploring Data Cube Technology

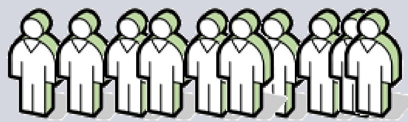
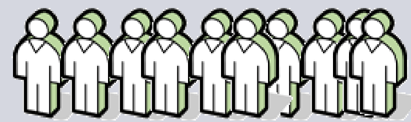
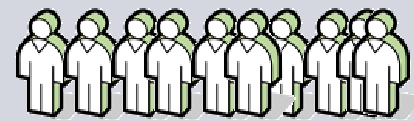
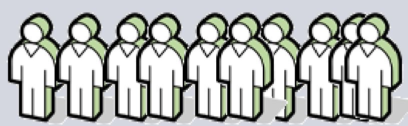
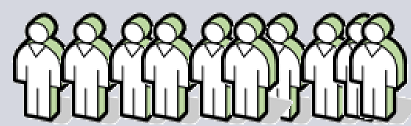
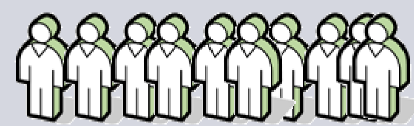
- Sampling Cube
  - X. Li, J. Han, Z. Yin, J.-G. Lee, Y. Sun, “Sampling Cube: A Framework for Statistical OLAP over Sampling Data”, SIGMOD’08
- Ranking Cube
  - D. Xin, J. Han, H. Cheng, and X. Li. Answering top-k queries with multi-dimensional selections: The ranking cube approach. VLDB’06
- Other advanced cubes for processing data and queries
  - Stream cube, spatial cube, multimedia cube, text cube, RFID cube, etc. — to be studied in volume 2

Statistical Surveys and OLAP

- Statistical survey: A popular tool to collect information about a **population** based on a **sample**
  - Ex.: TV ratings, US Census, election polls
- A common tool in politics, health, market research, science, and many more
- An efficient way of collecting information (Data collection is expensive)
- Many **statistical tools** available, to determine validity
  - Confidence intervals
  - Hypothesis tests
- OLAP (multidimensional analysis) on survey data
  - highly desirable but can it be done well?

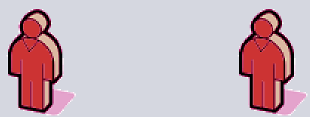
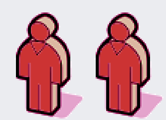
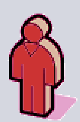
Surveys: Sample vs. Whole Population

Data is only a sample of **population**

Age\Education	High-school	College	Graduate
18			
19			
20			
...			

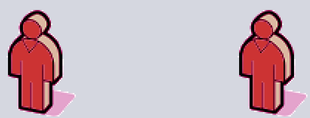
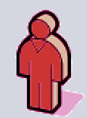
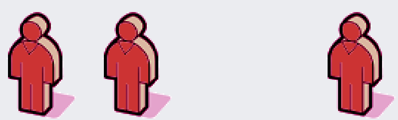
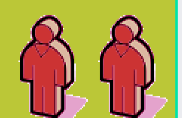
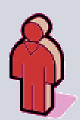
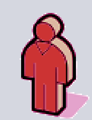
Problems for Drilling in Multidim. Space

Data is only a **sample** of population but samples could be small when drilling to certain multidimensional space

Age\Education	High-school	College	Graduate
18			
19			
20			
…			

OLAP on Survey (i.e., Sampling) Data

- Semantics of query is unchanged
- Input data has changed

Age/Education	High-school	College	Graduate
18			
19			
20			
…			

Challenges for OLAP on Sampling Data

- Computing confidence intervals in OLAP context
- No data?
  - Not exactly. No data in subspaces in cube
  - Sparse data
  - Causes include sampling bias and query selection bias
- Curse of dimensionality
  - Survey data can be high dimensional
  - Over 600 dimensions in real world example
  - Impossible to fully materialize

Example 1: Confidence Interval

What is the average income of 19-year-old high-school students?

Return not only query result but also confidence interval

Age / Education	High-school	College	Graduate
18			
19			
20			
...			

Confidence Interval

- *Confidence interval at* $\bar{x}$*:* $\bar{x} \pm t_c \hat{\sigma}_{\bar{x}}$
  - *x is a sample of data set;* $\bar{x}$ *is the mean of sample*
  - t_c *is the critical t-value, calculated by a look-up*
  - $\hat{\sigma}_{\bar{x}} = \frac{s}{\sqrt{l}}$ *is* the estimated standard error of the mean
- *Example: $50,000* $\pm$ *$3,000 with 95% confidence*
  - Treat points in cube cell as samples
  - Compute confidence interval as traditional sample set
- Return answer in the form of confidence interval
  - Indicates **quality** of query answer
  - User selects desired confidence interval

Efficient Computing Confidence Interval Measures

- Efficient computation in all cells in data cube
  - Both mean and confidence interval are **algebraic**
  - Why confidence interval measure is algebraic?

$$\bar{x} \pm t_c \hat{\sigma}_{\bar{x}}$$

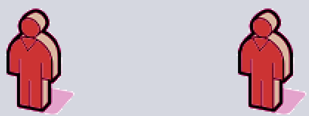
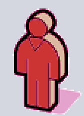
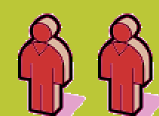
$\bar{x}$ is algebraic

$\hat{\sigma}_{\bar{x}} = \frac{s}{\sqrt{l}}$ where both *s* and *l (count)* are algebraic

- Thus one can calculate cells efficiently at more general cuboids without having to start at the base cuboid each time

Example 2: Query Expansion

What is the average income of 19-year-old college students?

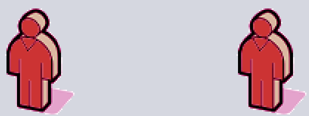
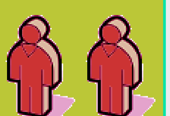
Age / Education	High-school	College	Graduate
18			
19			
20			
...			

Boosting Confidence by Query Expansion

- From the example: The queried cell “19-year-old college students” contains only 2 samples
- Confidence interval is large (i.e., low confidence). why?
  - Small sample size
  - High standard deviation with samples
- Small sample sizes can occur at relatively low dimensional selections
  - Collect more data?— expensive!
  - Use data in other cells? Maybe, but have to be careful

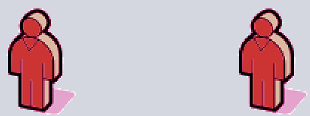
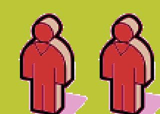
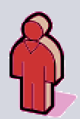
Intra-Cuboid Expansion: Choice 1

Expand query to include **18** and **20** year olds?

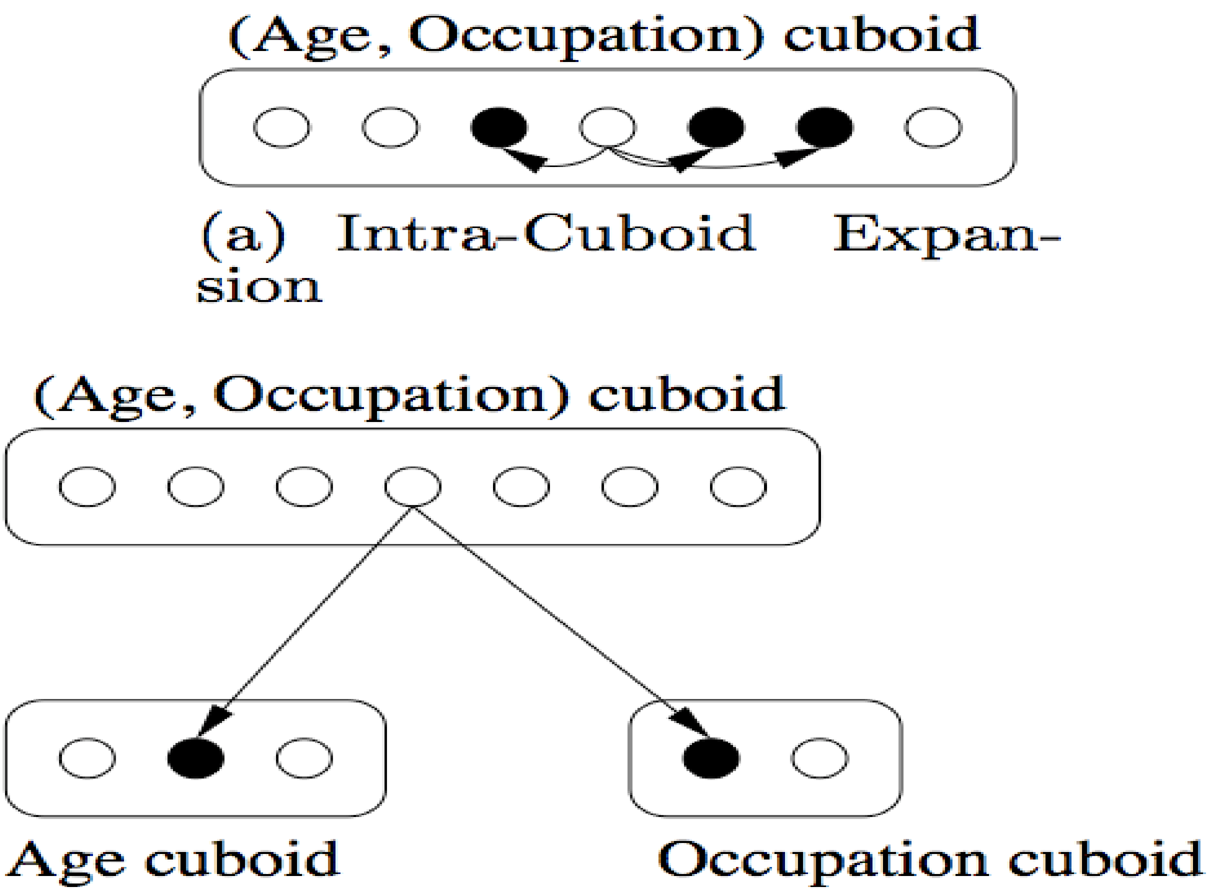
Age / Education	**High-school**	**College**	**Graduate**
18			
19			
20			
...			

Intra-Cuboid Expansion: Choice 2

Expand query to include **high-school** and **graduate** students?

Age / Education	**High-school**	**College**	**Graduate**
18			
19			
20			
...			

Query Expansion



(b) Inter-Cuboid Expansion

Chapter 5: Data Cube Technology

- Data Cube Computation: Preliminary Concepts
- Data Cube Computation Methods
- Processing Advanced Queries by Exploring Data Cube Technology
  - Sampling Cube
  - Ranking Cube
- Multidimensional Data Analysis in Cube Space
- Summary

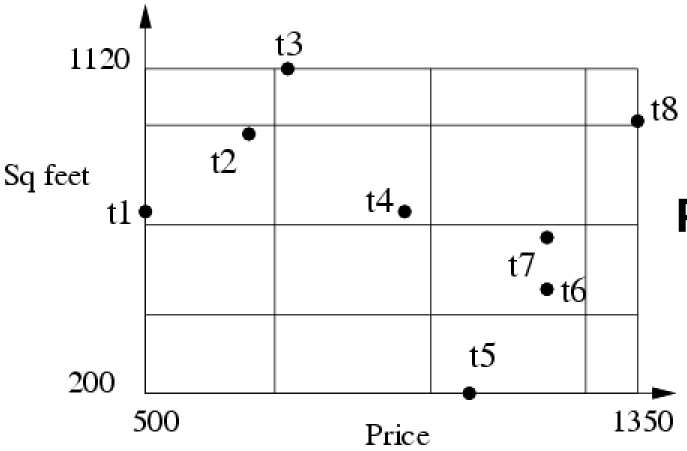
Ranking Cubes – Efficient Computation of Ranking queries

- Data cube helps not only OLAP but also ranked search
- **(top-k) ranking query**: only returns the best k results according to a user-specified preference, consisting of (1) a *selection condition* and (2) a *ranking function*
- Ex.: Search for apartments with expected price 1000 and expected square feet 800
  - Select top 1 from Apartment
  - where City = "LA" and Num_Bedroom = 2
  - order by [price – 1000]^2 + [sq feet - 800]^2 asc
- Efficiency question: Can we only search what we need?
  - Build a ranking cube on both selection dimensions and ranking dimensions

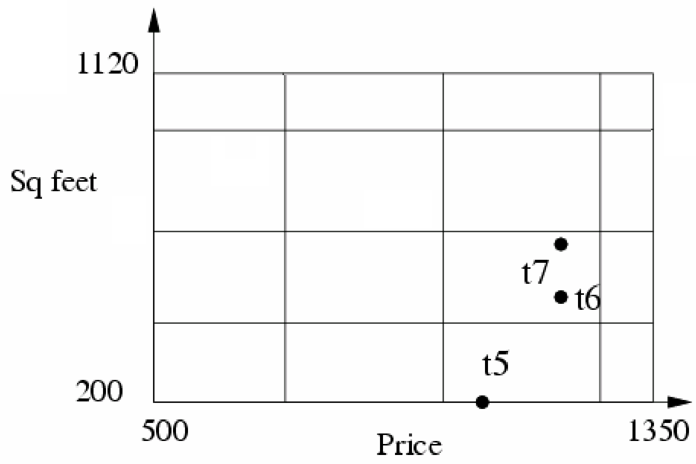
Ranking Cube: Partition Data on Both Selection and Ranking Dimensions

One single data partition as the template

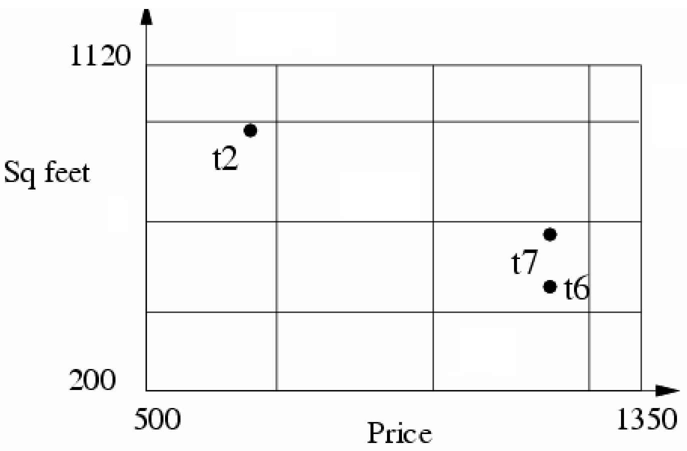
Slice the data partition by selection conditions



Partition for all data



Sliced Partition for city="LA"

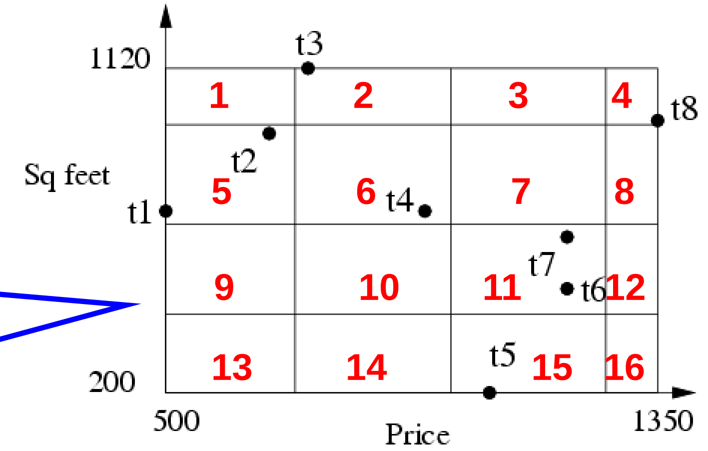


Sliced Partition for BR=2

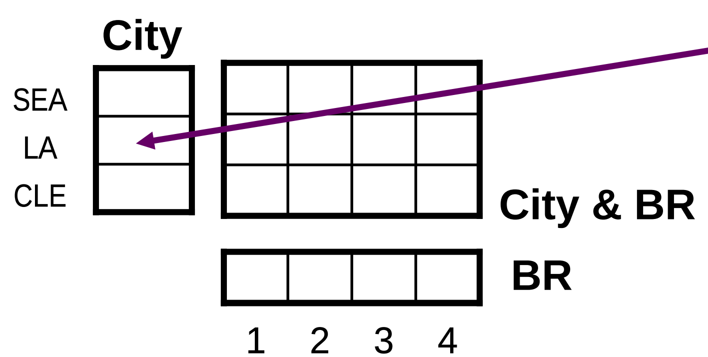
Materialize Ranking-Cube

tid	City	BR	Price	Sq feet
t1	SEA	1	500	600
t2	CLE	2	700	800
t3	SEA	1	800	900
t4	CLE	3	1000	1000
t5	LA	1	1100	200
t6	LA	2	1200	500
t7	LA	2	1200	560
t8	CLE	3	1350	1120

Step 1: Partition Data on Ranking Dimensions



Step 2: Group data by Selection Dimensions

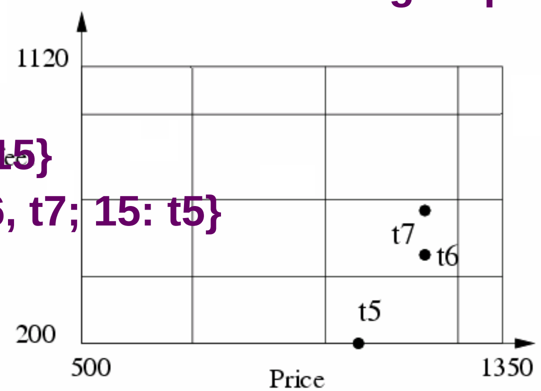


Step 3: Compute Measures for each group

For the cell (LA)

Block-level: {11, 15}

Data-level: {11: t6, t7; 15: t5}

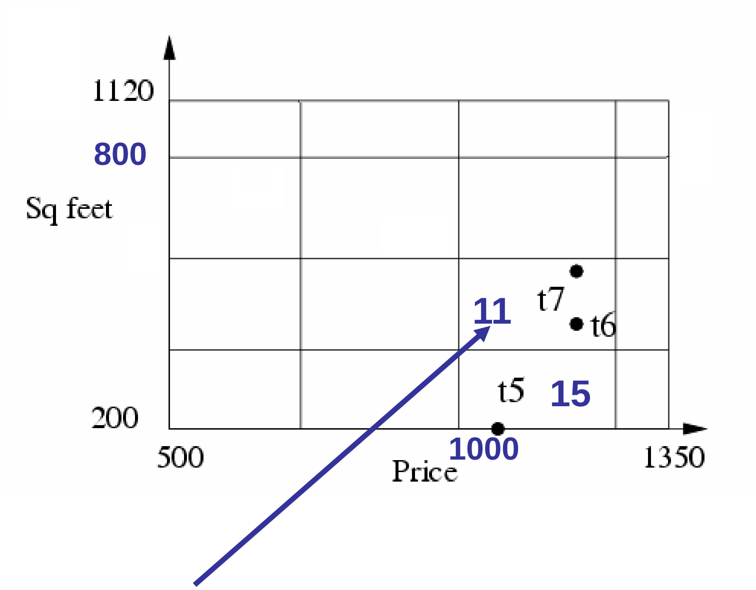


Processing Ranking Query: Execution Trace

Select top 1 from Apartment
where city = "LA"
order by [price – 1000]^2 + [sq feet - 800]^2 asc

Bin boundary for price	**[500, 600, 800, 1100,1350]**
Bin boundary for sq feet	**[200, 400, 600, 800, 1120]**

f=[price-1000]^2 + [sq feet – 800]^2



With ranking-cube: start search from here

Measure for LA: {11, 15}
{11: t6,t7; 15:t5}

Execution Trace:

1. **Retrieve High-level measure for LA {11, 15}**
2. **Estimate *lower bound score* for block 11, 15**

   f(block 11) = 40,000, f(block 15) = 160,000
3. **Retrieve block 11**
4. **Retrieve low-level measure for block 11**
5. **f(t6) = 130,000, f(t7) = 97,600**

   Output t7, done!

Ranking Cube: Methodology and Extension

- Ranking cube methodology
  - Push selection and ranking simultaneously
  - It works for many sophisticated ranking functions
- How to support high-dimensional data?
  - Materialize only those *atomic* cuboids that contain single selection dimensions
    - Uses the idea similar to high-dimensional OLAP
    - Achieves low space overhead and high performance in answering ranking queries with a high number of selection dimensions

Data Cube Technology: Summary

- Data Cube Computation: Preliminary Concepts
- Data Cube Computation Methods
  - MultiWay Array Aggregation
  - BUC
  - Star-Cubing
- Processing Advanced Queries by Exploring Data Cube Technology
  - Sampling Cubes
  - Ranking Cubes

Ref.(I) Data Cube Computation Methods

- S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. VLDB’96
- D. Agrawal, A. E. Abbadi, A. Singh, and T. Yurek. Efficient view maintenance in data warehouses. SIGMOD’97
- K. Beyer and R. Ramakrishnan. Bottom-Up Computation of Sparse and Iceberg CUBEs.. SIGMOD’99
- M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D. Ullman. Computing iceberg queries efficiently. VLDB’98
- J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. Data Mining and Knowledge Discovery, 1:29–54, 1997.
- J. Han, J. Pei, G. Dong, K. Wang. Efficient Computation of Iceberg Cubes With Complex Measures. SIGMOD’01
- L. V. S. Lakshmanan, J. Pei, and J. Han, Quotient Cube: How to Summarize the Semantics of a Data Cube, VLDB'02
- X. Li, J. Han, and H. Gonzalez, High-Dimensional OLAP: A Minimal Cubing Approach, VLDB'04
- Y. Zhao, P. M. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. SIGMOD’97
- K. Ross and D. Srivastava. Fast computation of sparse datacubes. VLDB’97
- D. Xin, J. Han, X. Li, B. W. Wah, Star-Cubing: Computing Iceberg Cubes by Top-Down and Bottom-Up Integration, VLDB'03
- D. Xin, J. Han, Z. Shao, H. Liu, C-Cubing: Efficient Computation of Closed Cubes by Aggregation-Based Checking, ICDE'06

Ref. (II) Advanced Applications with Data Cubes

- D. Burdick, P. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan. OLAP over uncertain and imprecise data. VLDB'05
- X. Li, J. Han, Z. Yin, J.-G. Lee, Y. Sun, "Sampling Cube: A Framework for Statistical OLAP over Sampling Data", SIGMOD'08
- C. X. Lin, B. Ding, J. Han, F. Zhu, and B. Zhao. Text Cube: Computing IR measures for multidimensional text database analysis. ICDM'08
- D. Papadias, P. Kalnis, J. Zhang, and Y. Tao. Efficient OLAP operations in spatial data warehouses. SSTD'01
- N. Stefanovic, J. Han, and K. Koperski. Object-based selective materialization for efficient implementation of spatial data cubes. IEEE Trans. Knowledge and Data Engineering, 12:938–958, 2000.
- T. Wu, D. Xin, Q. Mei, and J. Han. Promotion analysis in multidimensional space. VLDB'09
- T. Wu, D. Xin, and J. Han. ARCube: Supporting ranking aggregate queries in partially materialized data cubes. SIGMOD'08
- D. Xin, J. Han, H. Cheng, and X. Li. Answering top-k queries with multi-dimensional selections: The ranking cube approach. VLDB'06
- J. S. Vitter, M. Wang, and B. R. Iyer. Data cube approximation and histograms via wavelets. CIKM'98
- D. Zhang, C. Zhai, and J. Han. Topic cube: Topic modeling for OLAP on multi-dimensional text databases. SDM'09

Ref. (III) Knowledge Discovery with Data Cubes

- R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. ICDE'97
- B.-C. Chen, L. Chen, Y. Lin, and R. Ramakrishnan. Prediction cubes. VLDB'05
- B.-C. Chen, R. Ramakrishnan, J.W. Shavlik, and P. Tamma. Bellwether analysis: Predicting global aggregates from local regions. VLDB'06
- Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang, Multi-Dimensional Regression Analysis of Time-Series Data Streams, VLDB'02
- G. Dong, J. Han, J. Lam, J. Pei, K. Wang. Mining Multi-dimensional Constrained Gradients in Data Cubes. VLDB' 01
- R. Fagin, R. V. Guha, R. Kumar, J. Novak, D. Sivakumar, and A. Tomkins. Multi-structural databases. PODS'05
- J. Han. Towards on-line analytical mining in large databases. SIGMOD Record, 27:97–107, 1998
- T. Imielinski, L. Khachiyan, and A. Abdulghani. Cubegrades: Generalizing association rules. Data Mining & Knowledge Discovery, 6:219–258, 2002.
- R. Ramakrishnan and B.-C. Chen. Exploratory mining in cube space. Data Mining and Knowledge Discovery, 15:29–54, 2007.
- K. A. Ross, D. Srivastava, and D. Chatziantoniou. Complex aggregation at multiple granularities. EDBT'98
- S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. EDBT'98
- G. Sathe and S. Sarawagi. Intelligent Rollups in Multidimensional OLAP Data. *VLDB'01*