Data Mining: Foundation, Techniques and Applications

Lesson 4: Pre-Computation

Anthony Tung(鄧锦浩)
School of Computing
National University of Singapore

Li Cuiping(李翠平)
School of Information
Renmin University of China

Outline

- Introduction
- Data Cubes
  - Concepts/Modeling
  - Views selection
  - Cube computation
  - High dimension cube
  - Other issues
- Data Reduction:
  - One-Dimensional Synopses: Histograms, Wavelets

Introduction

- Complex queries on raw data can be very expensive in both CPU and I/O cost
- Many of these queries are often repeated or share common intermediate result
- Solution: Pre-computed information which can be used to speed up the answering of queries or mining. Result returned can be exact or approximate.

Outline

- Introduction
- Data Cubes
  - Concepts/Modeling
  - Views selection
  - Cube computation
  - High dimension cube
  - Other issues
- Data Reduction:
  - One-Dimensional Synopses: Histograms, Wavelets

Data Cubes

- Two ways to look at them:
  - As a relational aggregation operator to generalize group-by and aggregate. Use to model data warehouse. Also known as OLAP(On-line Analytical Processing)
  - As the implementation for supporting the above operator
- Relational Operator

```
select  model, year, color, sum(sales)
from  car_sales
where  model in {“chevy”, “ford”}
and  year between 1990 and 1994
group by cube model, year, color
having sum(sales) > 0;
```

Data Cube: An Example

SALES			
Model	Year	Color	Sales
Chevy	1990	red	5
Chevy	1990	white	87
Chevy	1990	blue	62
Chevy	1991	red	54
Chevy	1991	white	95
Chevy	1991	blue	49
Chevy	1992	red	31
Chevy	1992	white	54
Chevy	1992	blue	71
Ford	1990	red	64
Ford	1990	white	62
Ford	1990	blue	63
Ford	1991	red	52
Ford	1991	white	9
Ford	1991	blue	55
Ford	1992	red	27
Ford	1992	white	62
Ford	1992	blue	39

CUBE

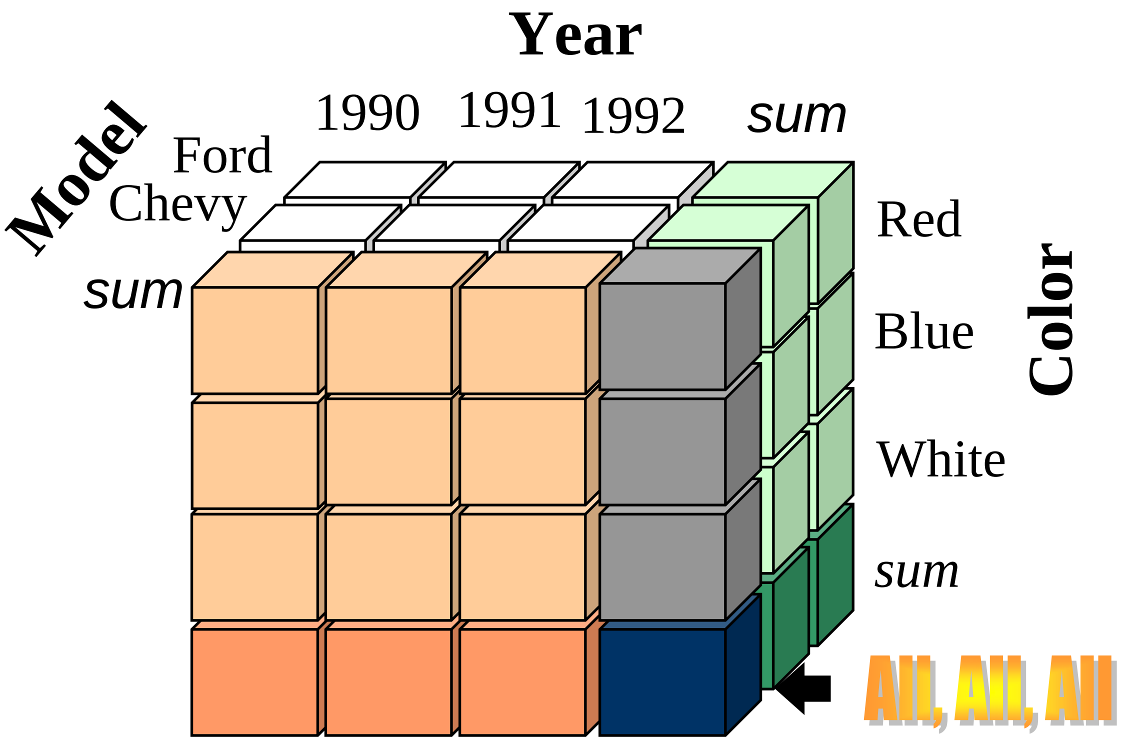
DATA CUBE			
Model	Year	Color	Sales
ALL	ALL	ALL	942
chevy	ALL	ALL	510
ford	ALL	ALL	432
ALL	1990	ALL	343
ALL	1991	ALL	314
ALL	1992	ALL	285
ALL	ALL	red	165
ALL	ALL	white	273
ALL	ALL	blue	339
chevy	1990	ALL	154
chevy	1991	ALL	199
chevy	1992	ALL	157
ford	1990	ALL	189
ford	1991	ALL	116
ford	1992	ALL	128
chevy	ALL	red	91
chevy	ALL	white	236
chevy	ALL	blue	183
ford	ALL	red	144
ford	ALL	white	133
ford	ALL	blue	156
ALL	1990	red	69
ALL	1990	white	149
ALL	1990	blue	125
ALL	1991	red	107
ALL	1991	white	104
ALL	1991	blue	104
ALL	1992	red	59
ALL	1992	white	116
ALL	1992	blue	110

Why the ALL Value?

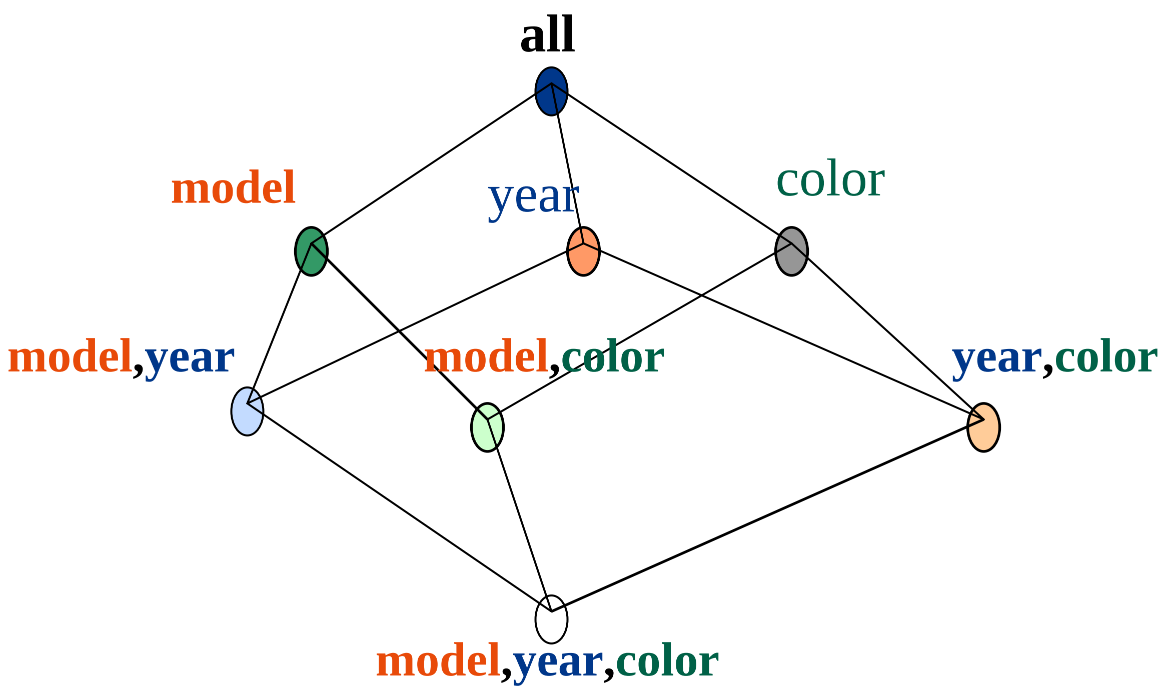
- Need a new "Null" value (overloads the null indicator)
- Value must not already be in the aggregated domain
- Can't use NULL since may aggregate on it.
- Think of ALL as a token representing the set
  - {red, white, blue}, {1990, 1991, 1992}, {Chevy, Ford}
- Rules for "ALL" in other areas not explored
  - assertions
  - insertion / deletion / ...
  - referential integrity
- Follow "set of values" semantics.

Data Cube: A Conceptual Views

- Provide a multidimensional view of data for easier data analysis



Cube: A Lattice of Cuboids



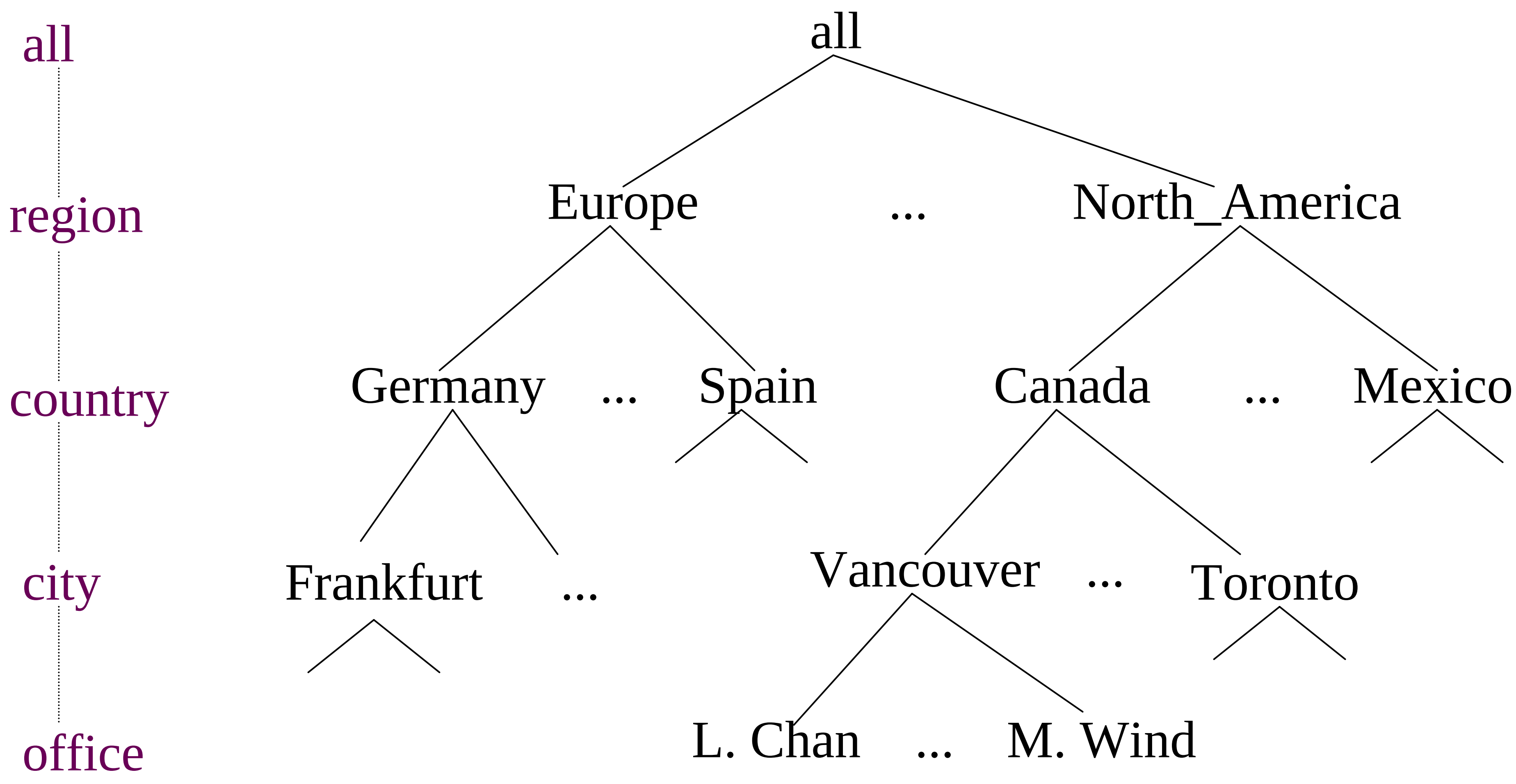
0-D(apex) cuboid

1-D cuboids

2-D cuboids

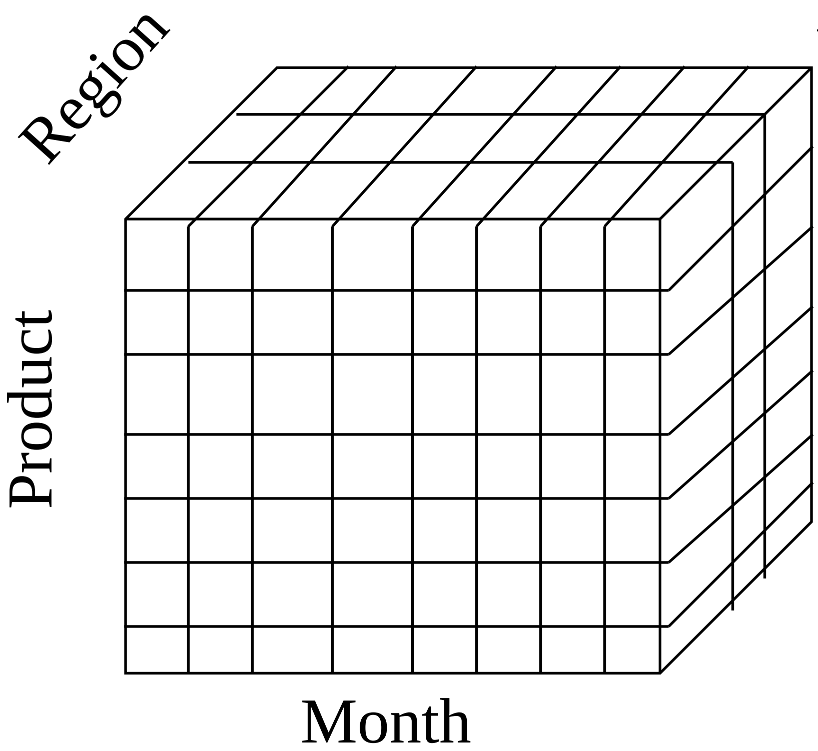
3-D(base) cuboid

A Concept Hierarchy: Dimension (location)

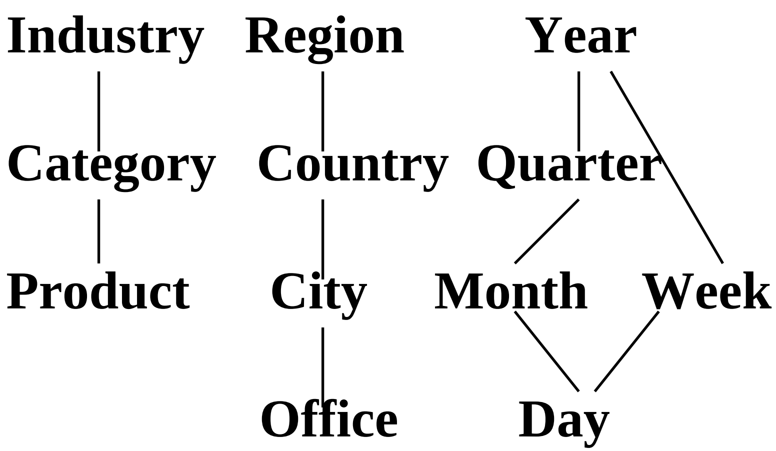


A Concept Hierarchy: More examples

- Sales volume as a function of product, month, and region



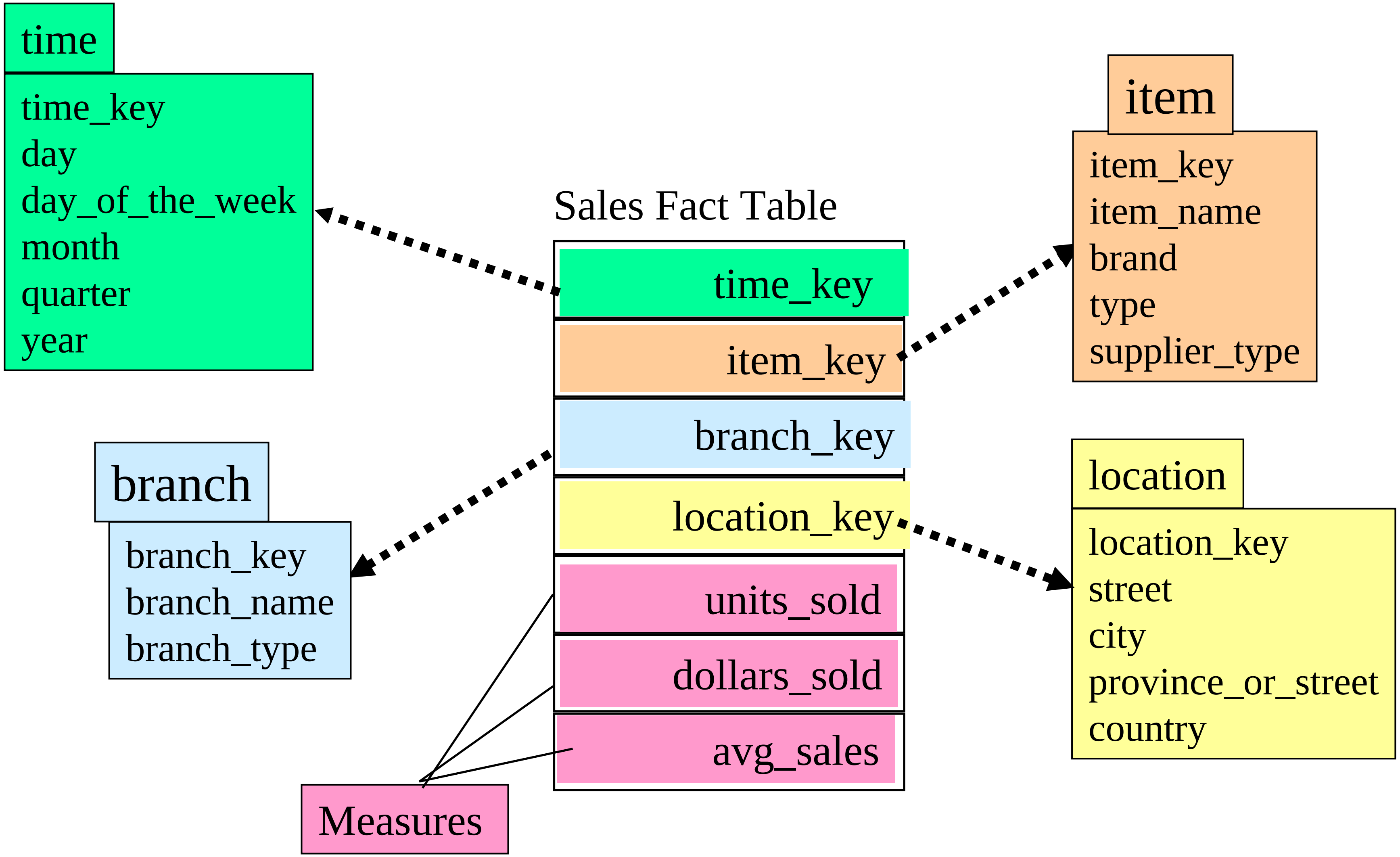
Dimensions: Product, Location, Time
Hierarchical summarization paths



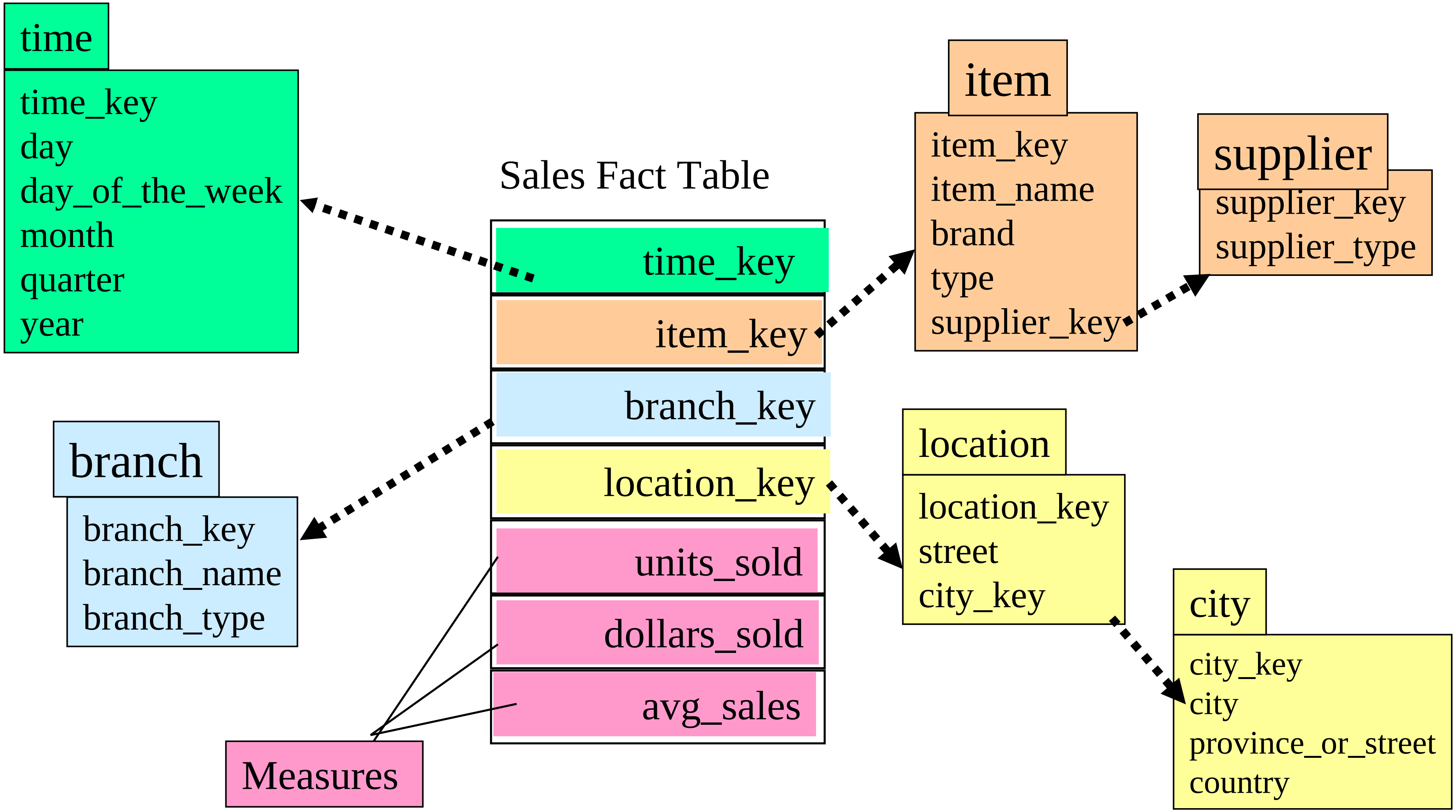
Conceptual Modeling of Data Cube

- Modeling data cubes: dimensions & measures
  - Star schema: A fact table in the middle connected to a set of dimension tables
  - Snowflake schema: A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape similar to snowflake
  - Fact constellations: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation

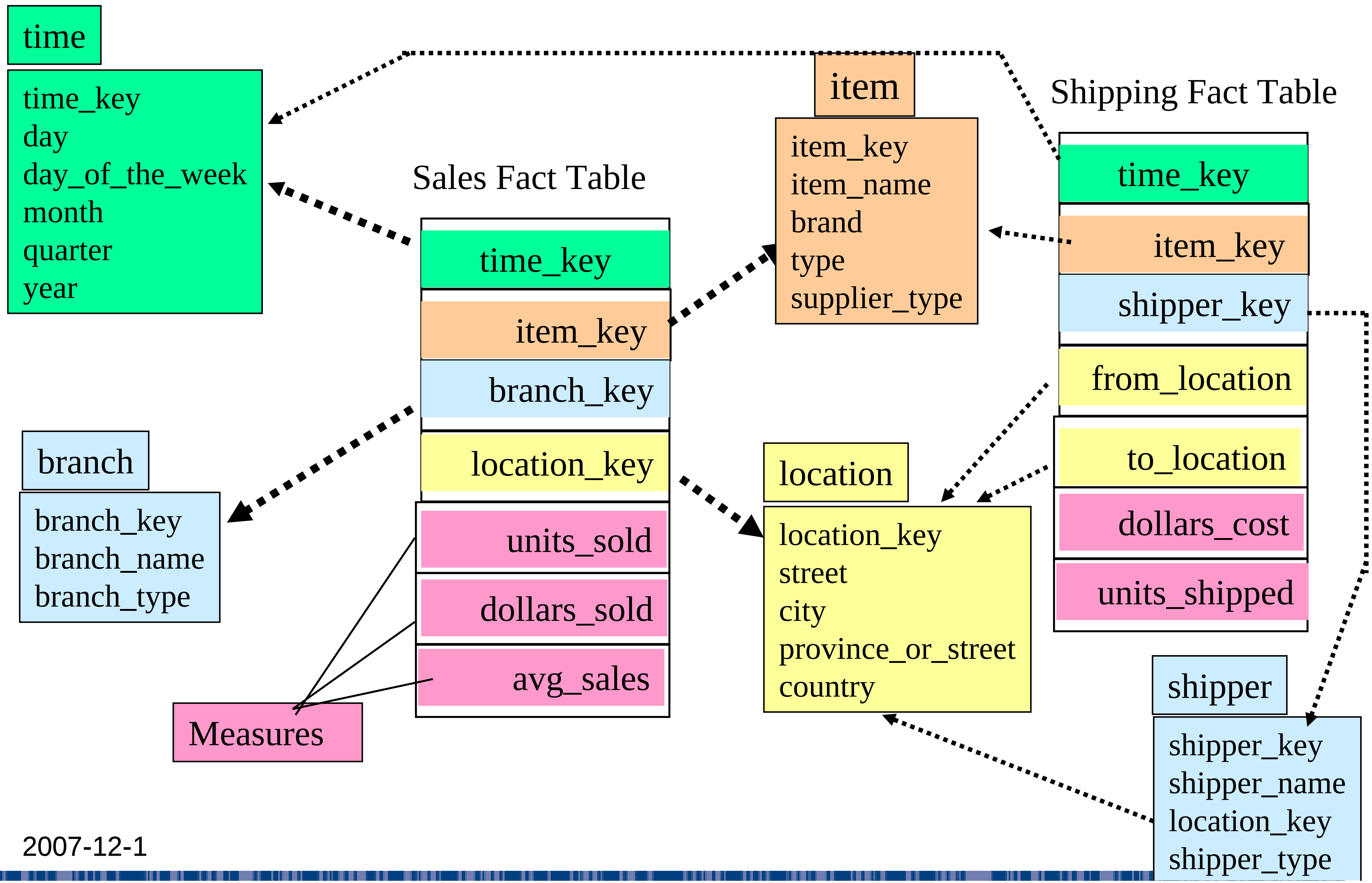
Example of Star Schema



Example of Snowflake Schema



Example of Fact Constellation



Measures: Three Categories

- distributive: if the result derived by applying the function to *n* aggregate values is the same as that derived by applying the function on all the data without partitioning.
  - E.g., count(), sum(), min(), max().
- algebraic: if it can be computed by an algebraic function with *M* arguments (where *M* is a bounded integer), each of which is obtained by applying a distributive aggregate function.
  - E.g., avg(), min_N(), standard_deviation().
- holistic: if there is no constant bound on the storage size needed to describe a subaggregate.
  - E.g., median(), mode(), rank().

Typical OLAP Operations

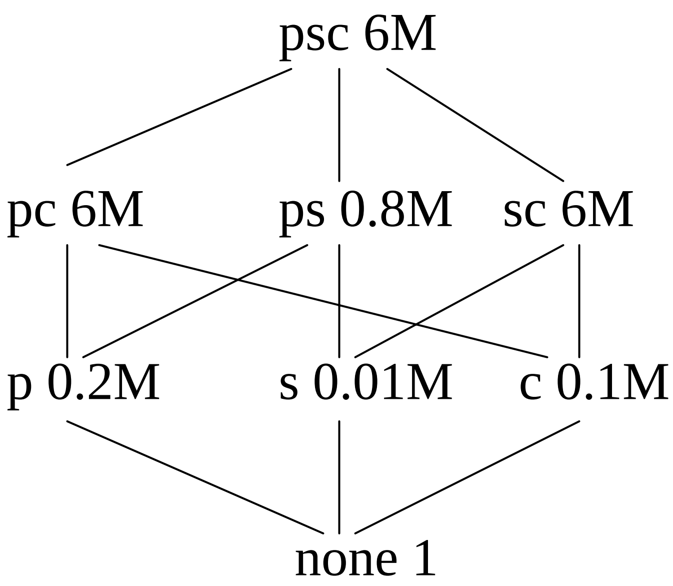
- Roll up (drill-up): summarize data
  - *by climbing up hierarchy or by dimension reduction*
- Drill down (roll down): reverse of roll-up
  - *from higher level summary to lower level summary or detailed data, or introducing new dimensions*
- Slice and dice:
  - *select on one or more dimensions*
- Pivot (rotate):
  - *reorient the cube, visualization, 3D to series of 2D planes.*
- Other operations
  - *drill across: involving (across) more than one fact table*
  - *drill through: through the bottom level of the cube to its back-end relational tables (using SQL)*

Cube Alternatives in RDBMS

- Physically materialize the whole data cube
  - Best query response
  - Heavy pre-computing, large storage space
- Materialize nothing
  - Worse query response
  - Dynamic query evaluation, less storage space
- Materialize only part of the data cube
  - Balance the storage space and response
  - Addressed in this paper

Motivating Example

- Parts (p) are bought from suppliers (s) and then sold to customers (c) at a sale price SP.



1. part,supplier,customer (6M, I.e., 6 million rows)
2. Part, customer (6M)
3. Part, supplier (0.8M)
4. Supplier, customer (6M)
5 part (0.2M)
6. Supplier (0.01M)
7. Customer (0.1M)
8. None (1)

Questions

- How many views must we materialize to get reasonable performance?
- Given that we have space S, what views do we materialize so that we minimize average query cost
- If we're willing to tolerate an X% degradation in average query cost from a fully materialize data cube, how much space can we save over the full materialized data cube?

Outline

- Introduction
- Data Cubes
  - Concepts/Modeling
  - Views selection
  - Cube computation
  - High dimension cube
  - Other issues
- Data Reduction:
  - One-Dimensional Synopses: Histograms, Wavelets

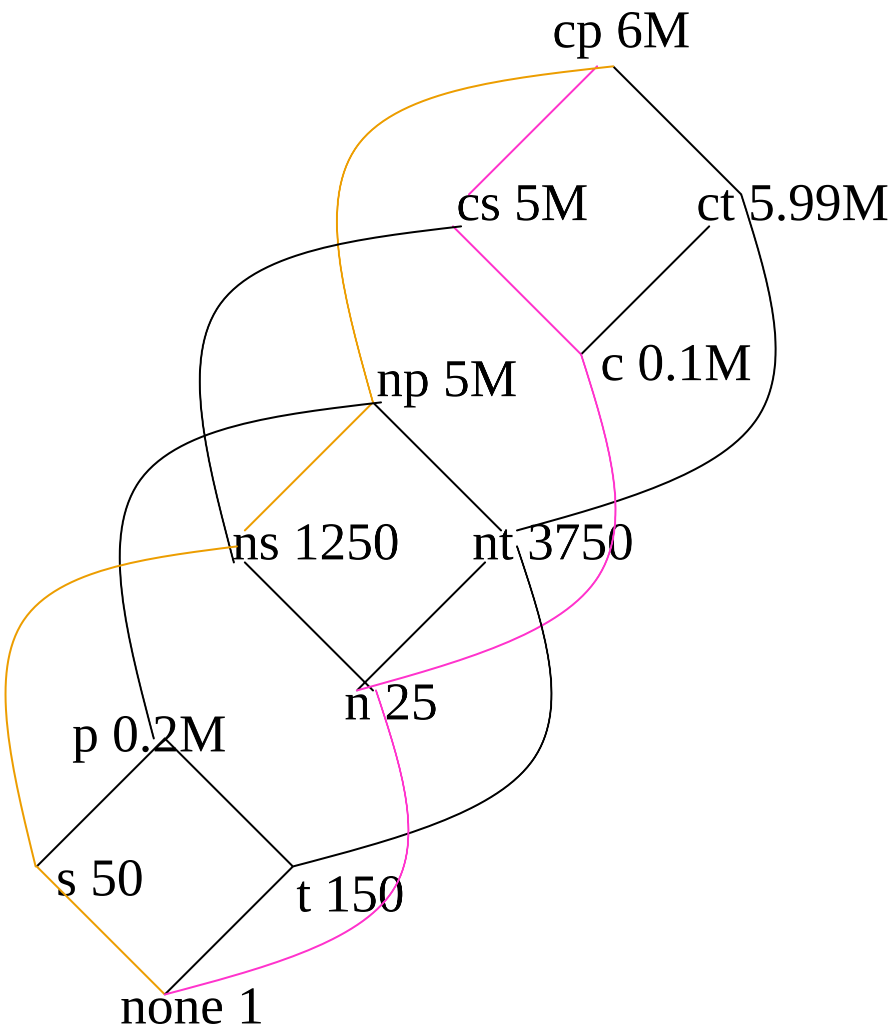
Dependence Relation on Queries

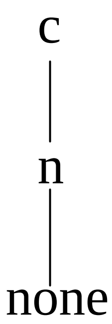
- Q1, Q2 are two queries, Q1≦Q2 if and only if Q1 can be answered using only the results of query Q2, or we may say Q1 is dependent on Q2.
  - E.g. (part) ≦(part, customer), (part) ≦(customer), (customer) ≦(part)
  - Here, relation '≦' is called partial order
  - All the views (queries) of a cube L and dependence relations '≦' is a lattice, denoted as <L, ≦>

Composite Lattices

- Two kinds of query dependence
  - Query dependencies caused by the interaction of the different dimensions with one another.
  - Query dependencies within a dimension caused by attributes hierarchies.
- Composite lattice
  - H_i is the hierarchy lattice of dimension_i, then $(a_1,a_2,\ldots,a_n) \leqq (b_1,b_2,\ldots,b_n)$ if and only if $a_i \leqq b_i$ for all i.

Example

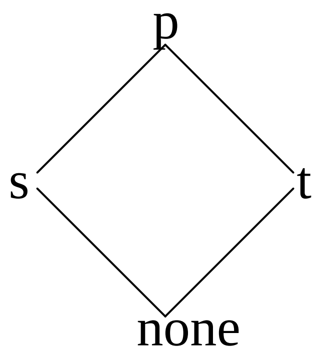




customer

c:individual customer

n: country



part

p:individual part

s: size

T: type

View Queries

- Types of view queries
  - Queries for the whole view
    - Scanning of the whole view, cost are proportional to the size of the view
  - Queries for a single or small numbers of cells.
    - The materialized view is indexed, about 1 (I/O)
    - No index, the cost almost equal to scanning the whole view
- Assumption
  - All the queries are identical to some element (view) in the given lattice.

* In fact, complex access path to single cells may be used

Cost Model

- Linear relationship between cost and size

  $T=m*S + c$

  T: running time of the query on the view

  m: ratio of query time to the size

  S: size of the view

  c: a fixed cost (overhead of running this query on a view of negligible size)

Source	size	Time(sec.)	Ratio
From cell itself	1	2.07	Not applicable
From view(supplier)	10,000	2.38	.000031
View (part, supplier)	800,000	20.77	.000023
View (part, supplier, customer)	6,000,000	226.23	.000037

Benefit to Materialize a View

$\langle L, \leqq \rangle$ is a cube,

$S = \{MV_1, MV_2, \ldots, MV_n\}$ is a subset of L,all MV_is are already materialized. S always includes the top view.

A view $V \in L$, the benefit of V relative to S is defined as follows.

1. For each $W \leqq V$, define the quantity B_w by
   (a) Let U be the view of least cost in S such that $W \leq U$. Note that since the top view is in S, there must be at least one such view in S.
   (b) If $C(V) < C(U)$, then $B_w = C(U) - C(V)$. Otherwise $B_w = 0$.
2. Define $B(V,S) = \sum_{W \leqq V} B_w$

*actually, here B(*V*,S) indicates how much will be saved if V is materialized.

Greedy Algorithm

Only k views are materialized in order to save the space.

S = {top view};

For I=1 to k do begin

select that view V not in S such that B(V,S) is maximized;

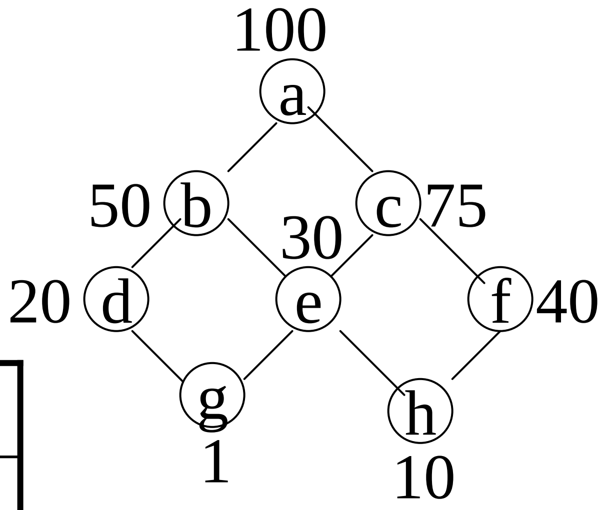
S=S union {V};

End;

Resulting S is the greedy selection;

Example for Greedy Selection

‘a’ is the top view; The first loop, ‘b’ is selected; the second loop, ‘f’ is selected, the third loop, ‘d’ is selected



	First Choice	Second Choice	Third Choice
b	50X5=250		
c	25X5=125	25X2=50	25X1=25
d	80X2=160	30X2=60	30X2=60
e	70X3=210	20X3=60	20+20+10=50
f	60X2=120	60+10=70	
g	99X1=99	49X1=49	49X1=49
h	90X1=90	40X1=40	30X1=30

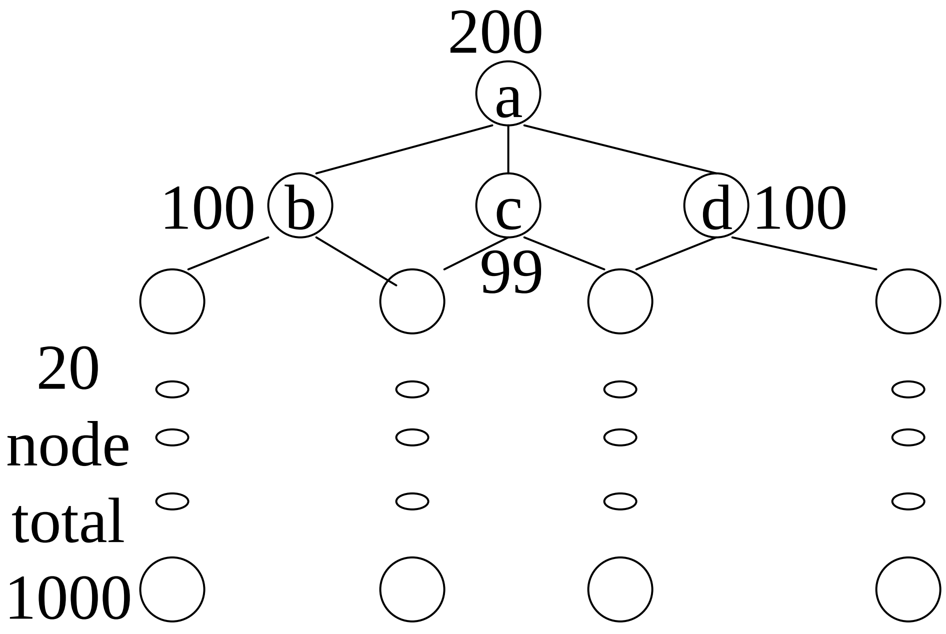
If only a is materialized, total cost is 800. If a,b,d,f materialized, total cost is 420

Not an Optimal

Suppose k = 2. The first loop, c is selected with benefit 41X101=4141. The second loop, b (or d) is selected with benefit 100X21=2100.

Total benefit=6241.

But if a,b,d are selected, total benefit = 20X100X4+2X100 = 8200



However

- Performance is guaranteed
  - A/B > 0.63 (A: optimal benefit; B: greedy benefit)
- The views in a lattice are unlikely to have the same probability of being requested in a query.
  - Weight each view by its probability
- Limited space may replace limited number of views
  - need further considerations

Outline

- Introduction
- Data Cubes
  - Concepts/Modeling
  - Views selection
  - Cube computation
  - High dimension cube
  - Other issues
- Data Reduction:
  - One-Dimensional Synopses: Histograms, Wavelets

Efficient Computation of Data Cubes

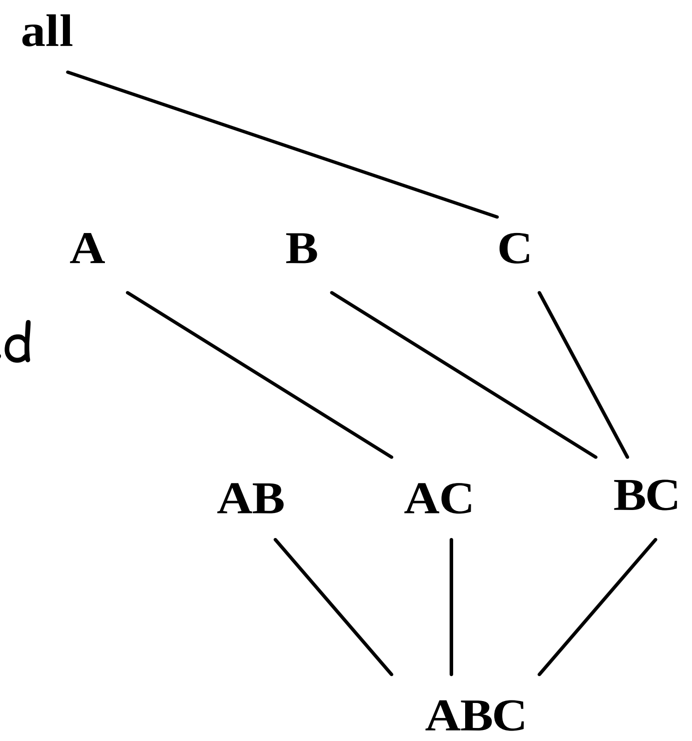
- Preliminary cube computation tricks (Agarwal et al'96)
- Computing full/iceberg cubes: 3 methodologies
  - Top-Down: Multi-Way array aggregation (Zhao,Deshpande & Naughton, SIGMOD'97)
  - Bottom-Up:
    - Bottom-up computation: BUC (Beyer & Ramarkrishnan, SIGMOD'99)
    - H-cubing technique (Han, Pei, 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
- Computing alternative kinds of cubes:
  - Partial cube, closed cube, approximate cube, etc.

Preliminary Tricks (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

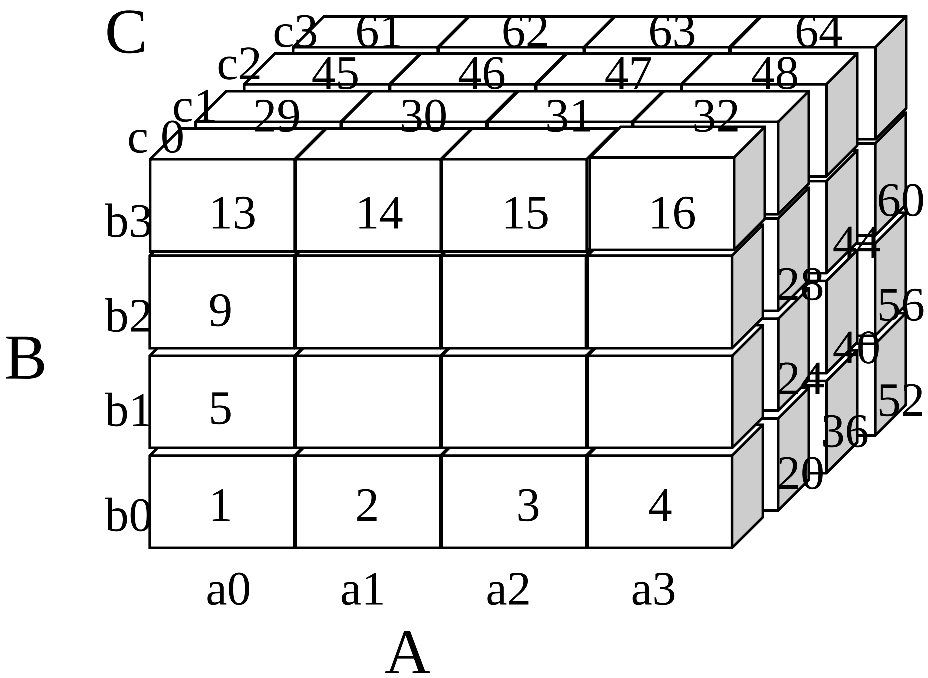
Multi-Way Array Aggregation

- 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)

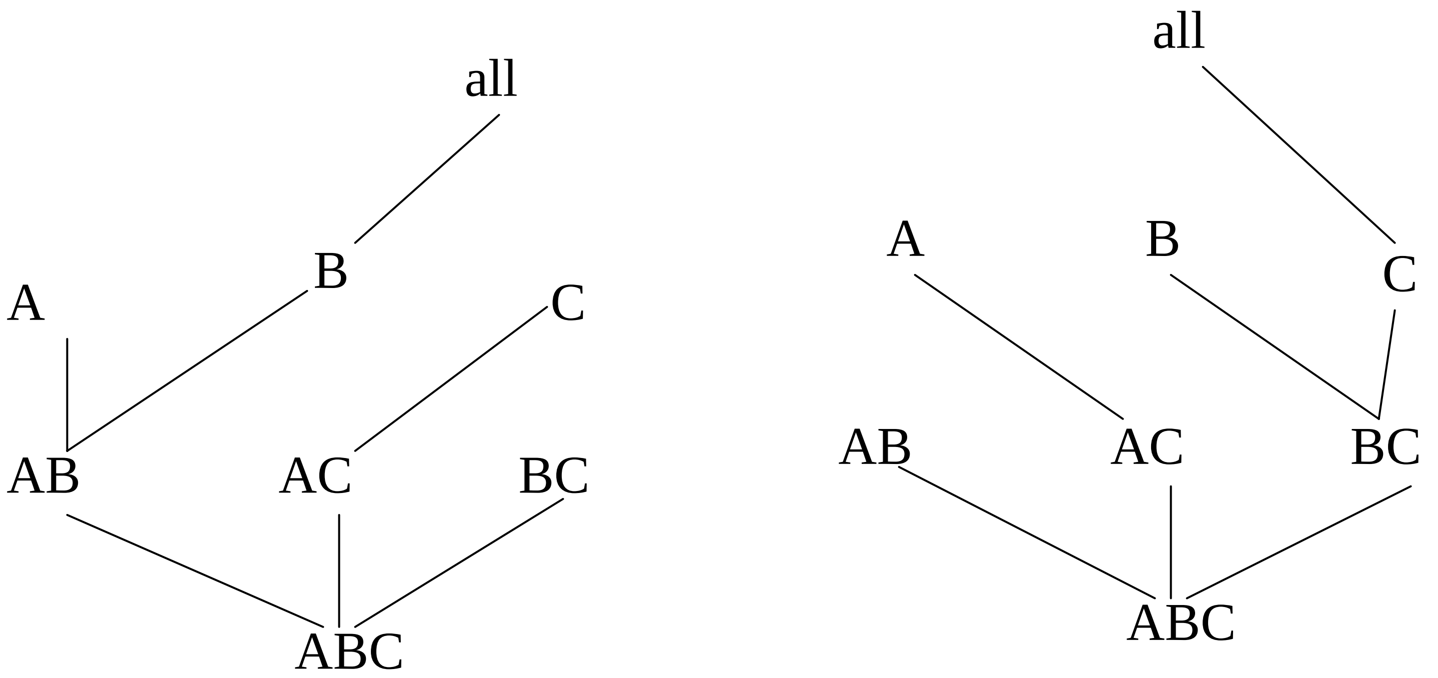
- Partition arrays into chunks (a small subcube which fits in memory).
- Compressed sparse array addressing: (chunk_id, offset)
- Compute aggregates in “multiway” by visiting cube cells in the order which minimizes the # of times to visit each cell, and reduces memory access and storage cost.



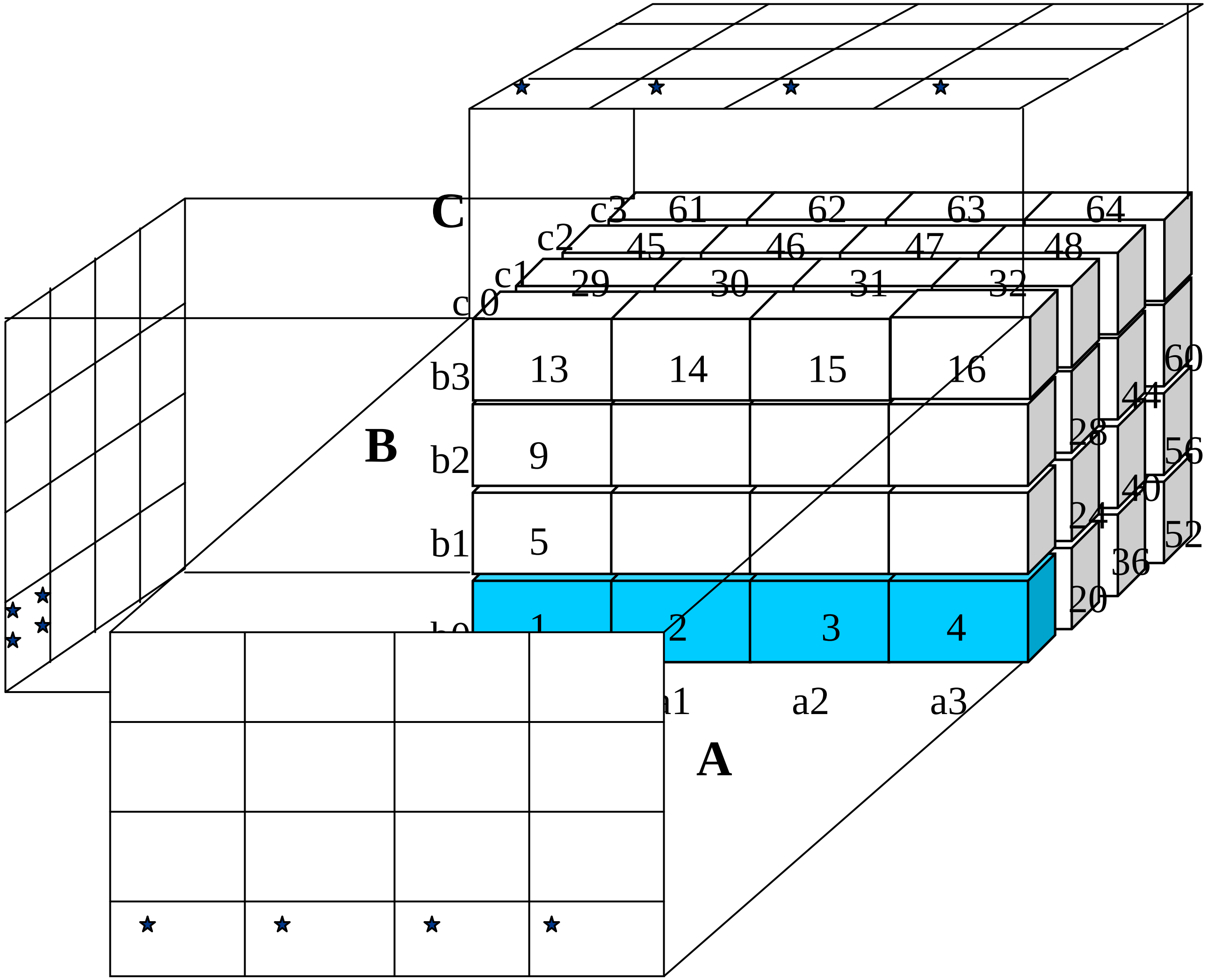
What is the best traversing order to do multi-way aggregation?

Order of computation

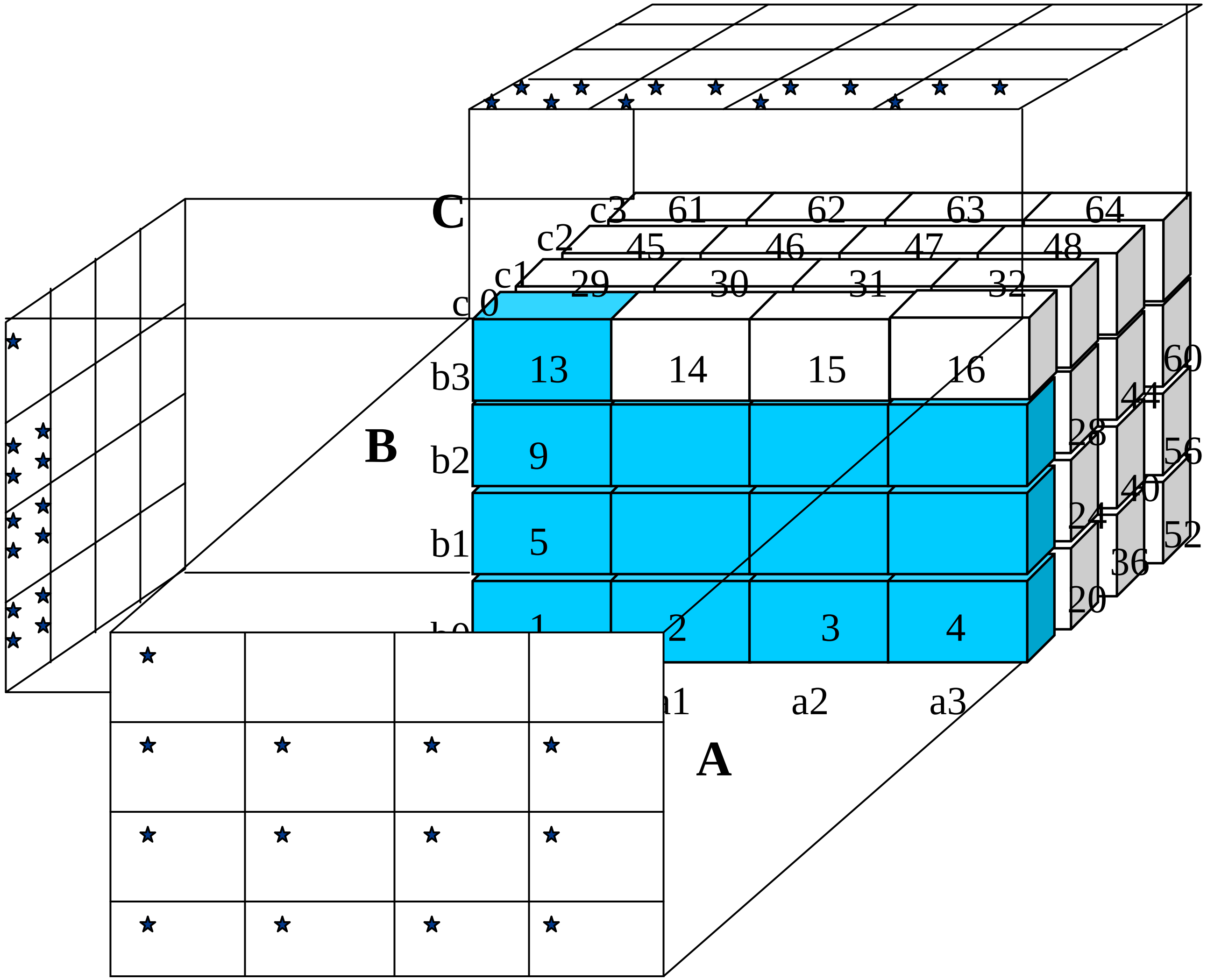
Two ways of ordering computation of a cube.
Which is more efficient ?



Multi-way Array Aggregation for Cube Computation



Multi-way Array Aggregation for Cube Computation

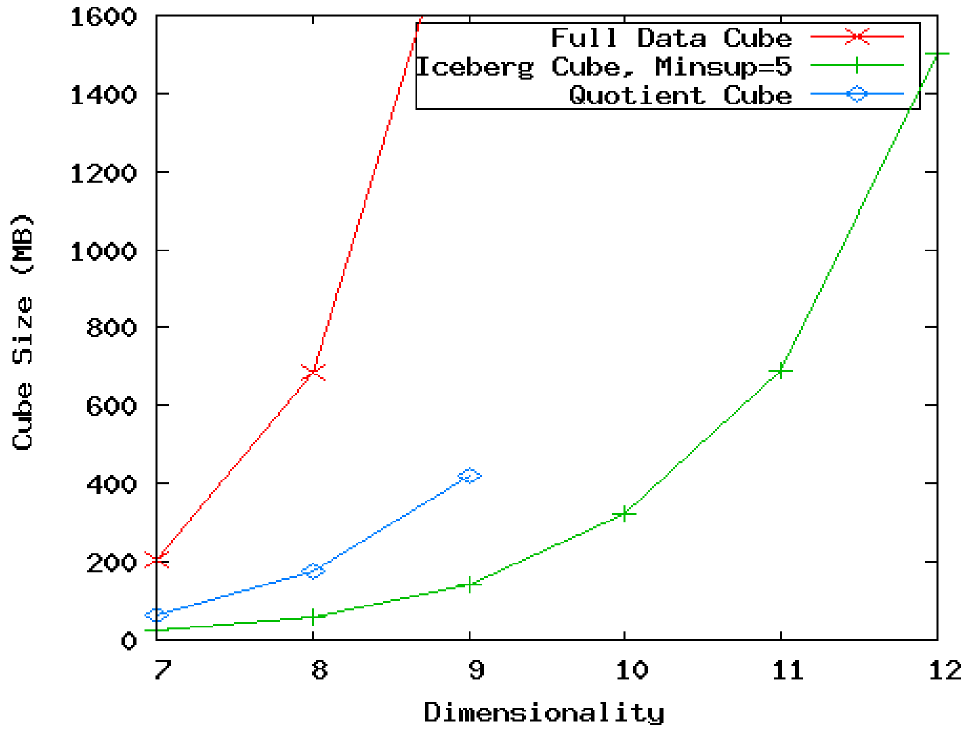


Multi-Way Array Aggregation for Cube Computation (Cont.)

- 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

The Curse of Dimensionality

- None of the previous cubing method can handle high dimensionality!
- A database of 600k tuples. Each dimension has cardinality of 100 and *zipf* of 2.



Outline

- Introduction
- Data Cubes
  - Concepts/Modeling
  - Views selection
  - Cube computation
  - High dimension cube
  - Other issues
- Data Reduction:
  - One-Dimensional Synopses: Histograms, Wavelets

Motivation of High-D OLAP

- Challenge to current cubing methods:
  - The “curse of dimensionality” problem
  - Iceberg cube and compressed cubes: only delay the inevitable explosion
  - Full materialization: still significant overhead in accessing results on disk
- High-D OLAP is needed in applications
  - Science and engineering analysis
  - Bio-data analysis: thousands of genes
  - Statistical surveys: hundreds of variables

Fast High-D OLAP with Minimal Cubing

- Observation: OLAP occurs only on a small subset of dimensions at a time
- Semi-Online Computational Model
  1. Partition the set of dimensions into **shell fragments**
  2. Compute data cubes for each shell fragment while retaining **inverted indices** or **value-list indices**
  3. Given the pre-computed **fragment cubes**, dynamically compute cube cells of the high-dimensional data cube *online*

Properties of Proposed Method

- Partitions the data vertically
- Reduces high-dimensional cube into a set of lower dimensional cubes
- Online re-construction of original high-dimensional space
- Lossless reduction
- Offers tradeoffs between the amount of pre-processing and the speed of online computation

Example Computation

- Let the cube aggregation function be `count`

tid	**A**	**B**	**C**	**D**	**E**
1	a1	b1	c1	d1	e1
2	a1	b2	c1	d2	e1
3	a1	b2	c1	d1	e2
4	a2	b1	c1	d1	e2
5	a2	b1	c1	d1	e3

- Divide the 5 dimensions into 2 shell fragments:
  - (`A, B, C`) and (`D, E`)

1-D Inverted Indices

- Build traditional invert index or RID list

Attribute Value	TID List	List Size
a1	1 2 3	3
a2	4 5	2
b1	1 4 5	3
b2	2 3	2
c1	1 2 3 4 5	5
d1	1 3 4 5	4
d2	2	1
e1	1 2	2
e2	3 4	2
e3	5	1

Shell Fragment Cubes

- Generalize the 1-D inverted indices to multi-dimensional ones in the data cube sense

Cell	Intersection	TID List	List Size
a1 b1	1 2 3 $\cap$ 1 4 5	1	1
a1 b2	1 2 3 $\cap$ 2 3	2 3	2
a2 b1	4 5 $\cap$ 1 4 5	4 5	2
a2 b2	4 5 $\cap$ 2 3	$\otimes$	0

Shell Fragment Cubes (2)

- Compute all cuboids for data cubes ABC and DE while retaining the inverted indices
- For example, shell fragment cube ABC contains 7 cuboids:
  - A, B, C
  - AB, AC, BC
  - ABC
- This completes the offline computation stage

Shell Fragment Cubes (4)

- Shell fragments do not have to be disjoint
- Fragment groupings can be arbitrary to allow for maximum online performance
  - Known common combinations (e.g.,<city, state>) should be grouped together.
- Shell fragment sizes can be adjusted for optimal balance between offline and online computation

ID_Measure Table

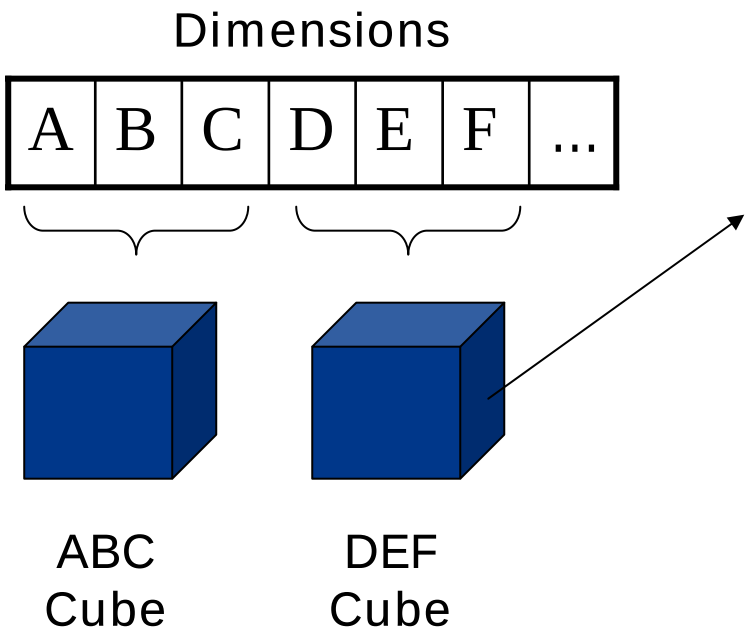
- If measures other than `count` are present, store in *ID_measure* table separate from the shell fragments

tid	count	sum
1	5	70
2	3	10
3	8	20
4	5	40
5	2	30

The Frag-Shells Algorithm

1. Partition set of dimension ($A_1,\ldots,A_n$) into a set of k fragments ($P_1,\ldots,P_k$).
2. Scan base table once and do the following
3. insert <tid, measure> into ID_measure table.
4. for each attribute value a_i of each dimension A_i
5. build inverted index entry <a_i, tidlist>
6. For each fragment partition P_i
7. build local fragment cube S_i by intersecting tid-lists in bottom-up fashion.

Frag-Shells (2)



D Cuboid

EF Cuboid

DE Cuboid

Cell	Tuple-ID List
d1 e1	{1, 3, 8, 9}
d1 e2	{2, 4, 6, 7}
d2 e1	{5, 10}
...	...

Online Query Computation

- For example, returns a 2-D data cube.

  (a1, b1, c1, * , e1)

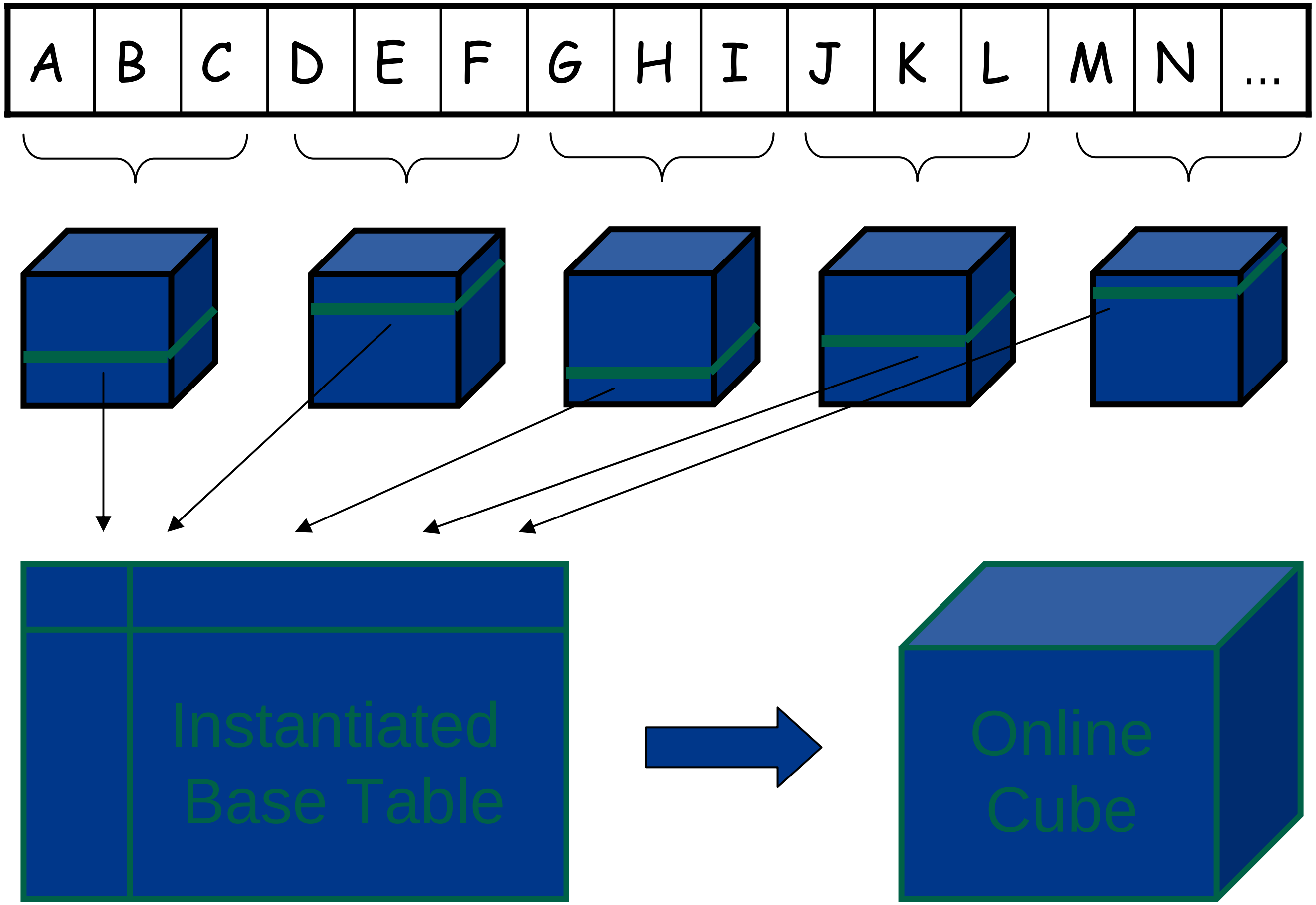
$$\langle a_1, a_2, \ldots, a_n : M \rangle$$

Online Query Computation (2)

- Given the fragment cubes, process a query as follows

1. Divide the query into fragment, same as the shell
2. Fetch the corresponding TID list for each fragment from the fragment cube
3. Intersect the TID lists from each fragment to construct **instantiated base table**
4. Compute the data cube using the base table with any cubing algorithm

Online Query Computation (3)

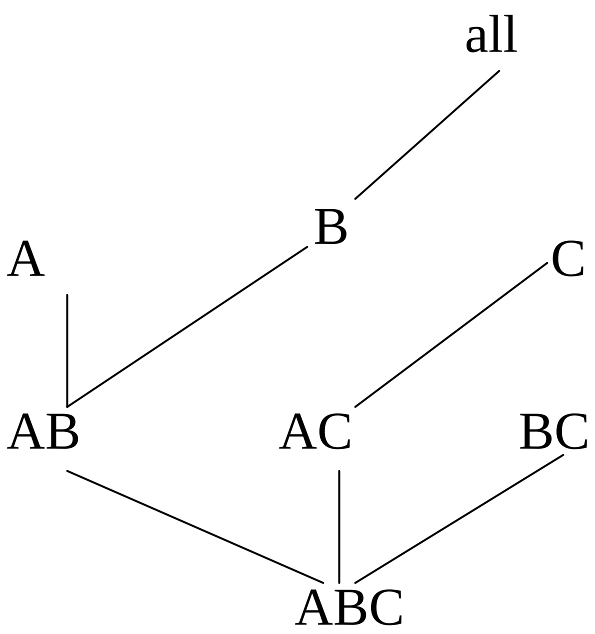


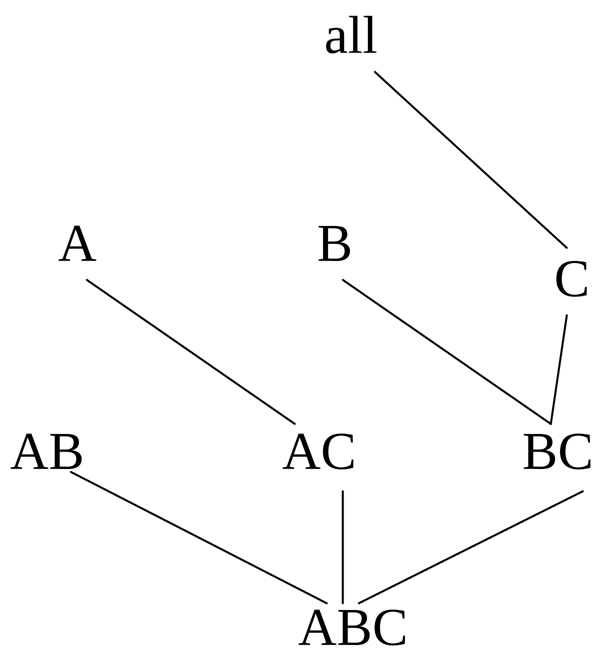
Outline

- Introduction
- Data Cubes
  - Concepts/Modeling
  - Views selection
  - Cube computation
  - High dimension cube
  - Other issues
- Data Reduction:
  - One-Dimensional Synopses: Histograms, Wavelets

Other Issues: Order of Materialization(II)

Two ways of ordering computation of a cube.
Which is more efficient ?





Other Issues: Efficient Range Query Processing

- What if we want to restrict our aggregation to certain range of values in each dimension ?
- Eg. Total number of people with (25<age<=35) and (160 < height <180) which is highlighted in blue.

age →

height ↓

3	5	1	2	2	3
7	3	2	6	8	2
2	4	2	3	3	5

Other Issues: Efficient Range Query Processing(II)

- Potential Solution: Pre-computed prefix-sum array

age →

height ↓

3	5	1	2	2	3
7	3	2	6	8	2
2	4	2	3	3	5

sum along row →

3	8	9	11	13	16
7	10	12	18	26	28
2	6	8	11	14	19

sum along column →

3	8	9	11	13	16
10	18	21	29	39	44
12	24	29	40	53	63

sum =(40-11)-(24-8)

= 40-11-24+8

= 13

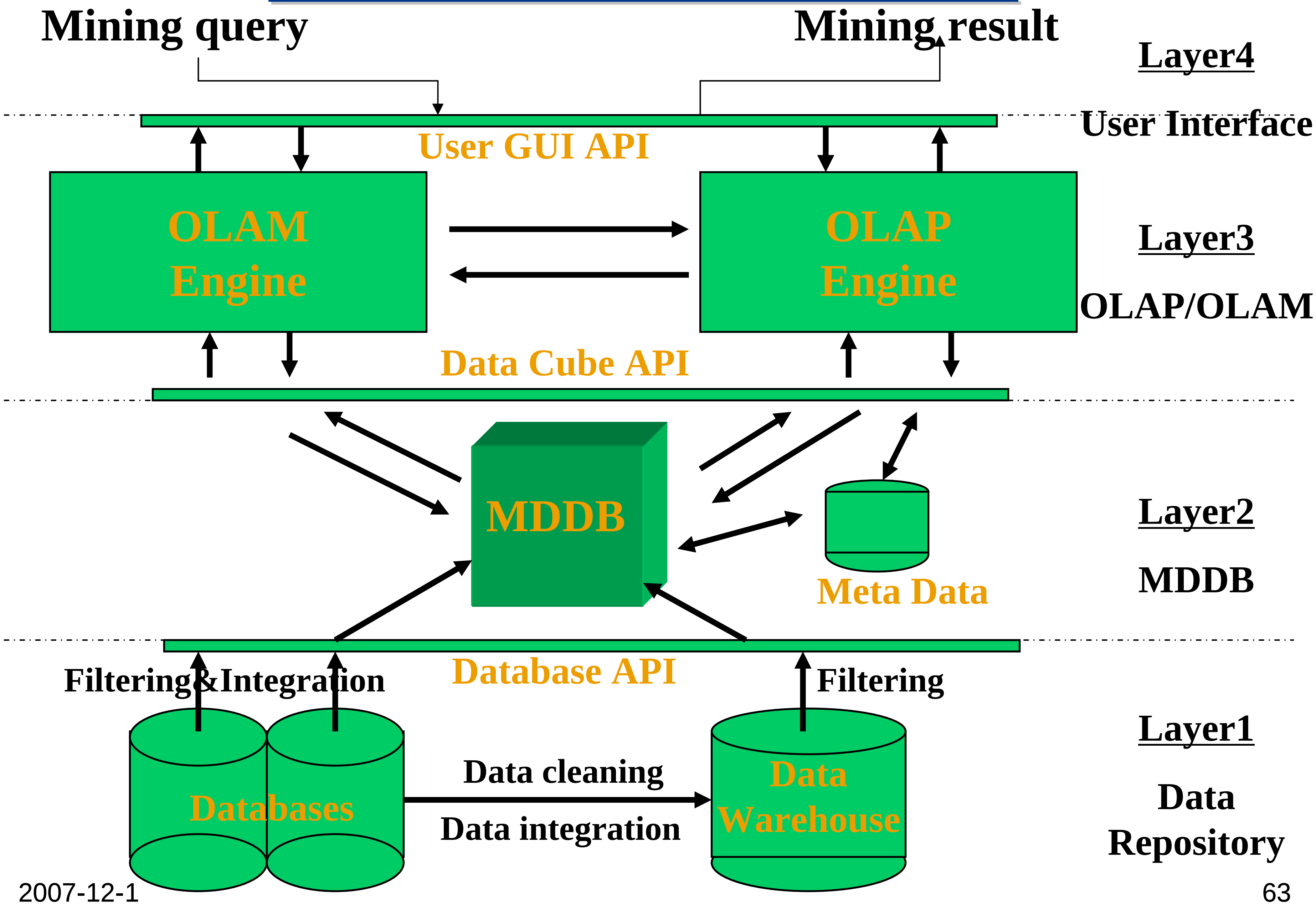
Other Issues: Efficient Range Query Processing(III)

- What about range max query ?
- What if the cube is sparse ?
- What happen if update is required ?

From On-Line Analytical Processing to On Line Analytical Mining (OLAM)

- Why online analytical mining?
  - High quality of data in data warehouses
    - DW contains integrated, consistent, cleaned data
  - Available information processing structure surrounding data warehouses
    - ODBC, OLEDB, Web accessing, service facilities, reporting and OLAP tools
  - OLAP-based exploratory data analysis
    - mining with drilling, dicing, pivoting, etc.
  - On-line selection of data mining functions
    - integration and swapping of multiple mining functions, algorithms, and tasks.
- Architecture of OLAM

An OLAM Architecture



Outline

- Introduction
- Data Cubes
  - Concepts/Modeling
  - Views selection
  - Cube computation
  - High dimension cube
  - Other issues
- Data Reduction:
  - One-Dimensional Synopses: Histograms, Wavelets

Data reduction

- Volume of data that must be handled in databases and data warehouses can be very large- terabytes of data are not uncommon
- Analyses and mining can be complex and can take a very long time to run on complete data set. There is also a need to do some estimation of the data distribution in order to formulate good query plan (or mining plan)
- Is it possible to have a reduced representation of the data set that is much smaller in data volume and yet produce the same analytical result ?

Measure of Performance

- Reduction Ratio
  - Size after reduction/Size before reduction
- Secondary Measure
  - Progressive Resolution
  - Incremental Computation
  - Speed of reduction - as long as not too slow
  - Speed of retrieval - more important

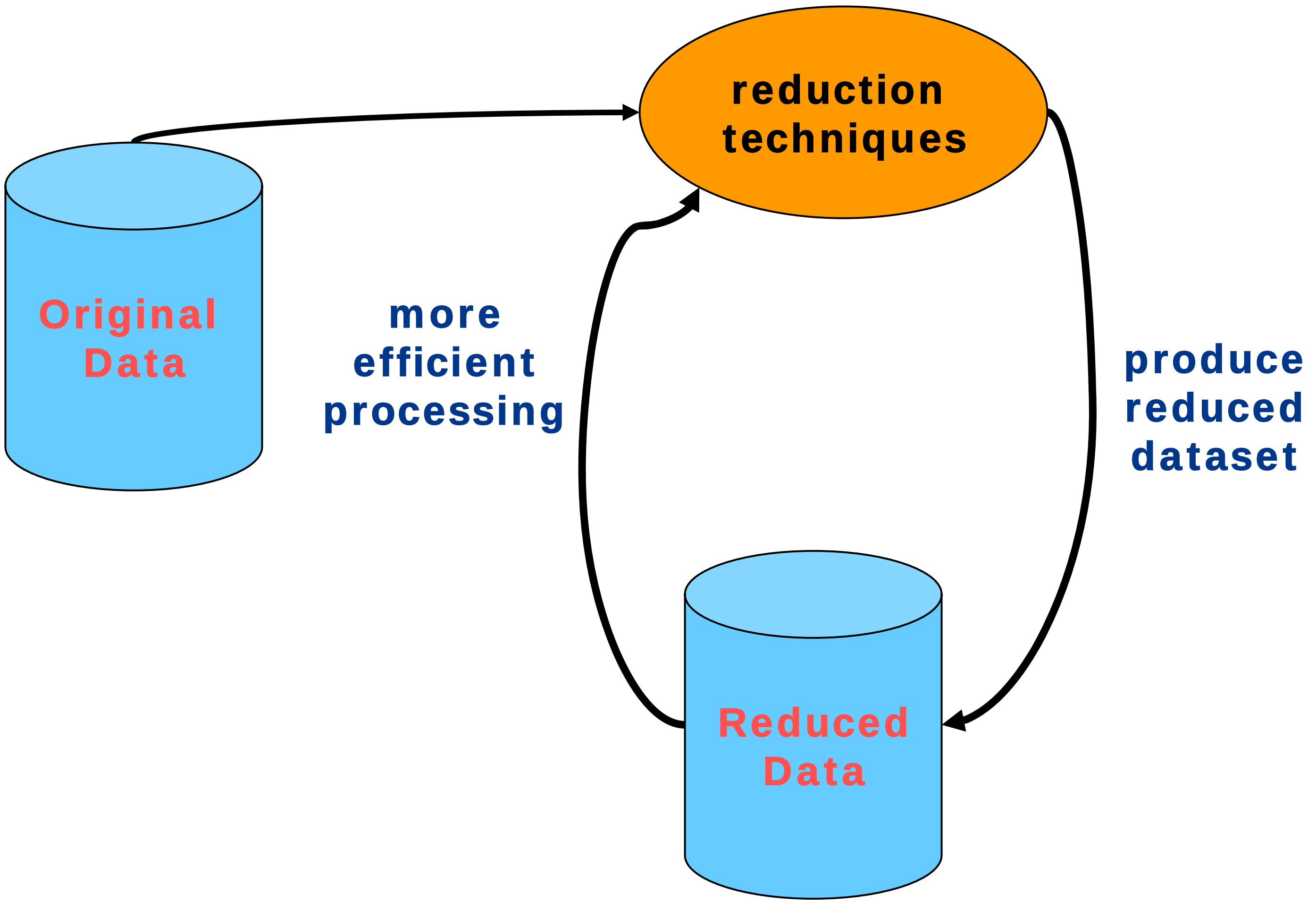
Numerosity vs Dimension Reduction

- Numerosity Reduction
  - Reduce the number of distinct values or tuples
  - Can involve one or more dimensions
  - Methods include
    - histograms, wavelet, sampling, clustering, index tree
- Dimension Reduction
  - Reduce the number of dimensions
  - Have to involve more than one dimensions
  - Methods include
    - Singular Value Decomposition (SVD), local dimensionality reduction
- Of course, you can try to do both

Parametric vs Non-Parametric Techniques

- Parametric
  - Assume a model for the data and the aim is to estimate the parameters for the model
  - Example: log-linear model, SVD, linear regression...
  - Advantage: Give good reduction if correct model and parameters are found
  - Disadvantage: Parameters are hard to estimate
- Non-Parametric
  - Opposite of parametric method, assume no model
  - Example: histogram, cluster, index tree
  - Advantage: No need to set parameters
  - Disadvantage: Less reduction
- Sampling: Neither parametric or non-parametric

The Data Reduction Cycle



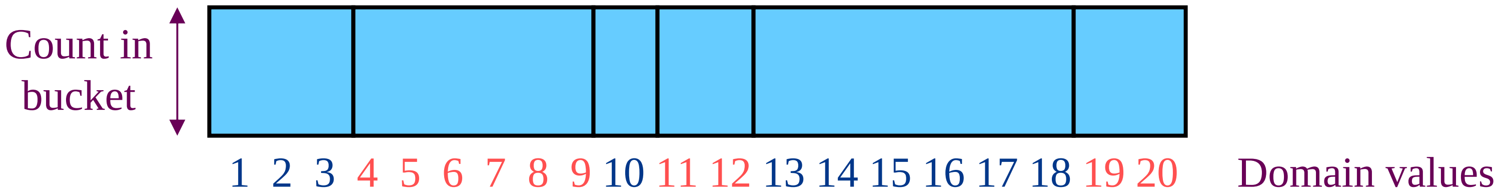
Our Focus of Studies

- One-Dimensional Synopses
  - **Histograms:** Equi-width, Equi-depth, Compressed, V-optimal
  - **Wavelets:** 1-D Haar-wavelet histogram construction

Histograms

- Partition attribute value(s) domain into a set of buckets
- Issues:
  - How to partition
  - What to store for each bucket
  - How to estimate an answer using the histogram
- Long history of use for selectivity estimation within a query optimizer [Koo80], [PSC84], etc.
- [PIH96] [Poo97] introduced a taxonomy, algorithms, etc.

1-D Histograms: Equi-Depth



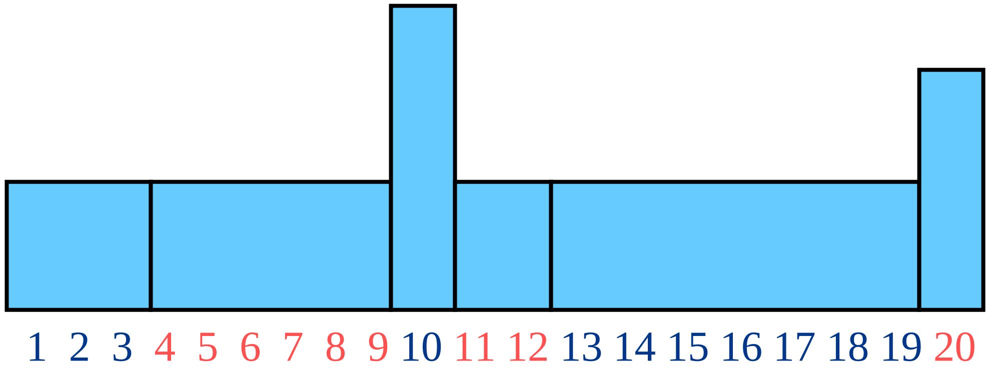
- Goal: Equal number of rows per bucket (B buckets in all)

- Can **construct** by first sorting then taking B-1 equally-spaced splits

  1 2 2 3 4 7 8 9 10 10 10 10 11 11 12 12 14 16 16 18 19 20 20 20

- **Faster construction:** Sample & take equally-spaced splits in sample
  - Nearly equal buckets
  - Can also use one-pass quantile algorithms (e.g., [GK01])

1-D Histograms: Compressed



[PIH96]

- Create singleton buckets for largest values, equi-depth over the rest

- Improvement over equi-depth since get exact info on largest values, e.g., join estimation in DB2 compares largest values in the relations

Construction: Sorting + O(B log B) + one pass; can use sample

1-D Histograms: V-Optimal

[IP95] defined V-optimal & showed it minimizes the average selectivity estimation error for equality-joins & selections

– Idea: Select buckets to minimize frequency variance within buckets

One-Dimensional Haar Wavelets

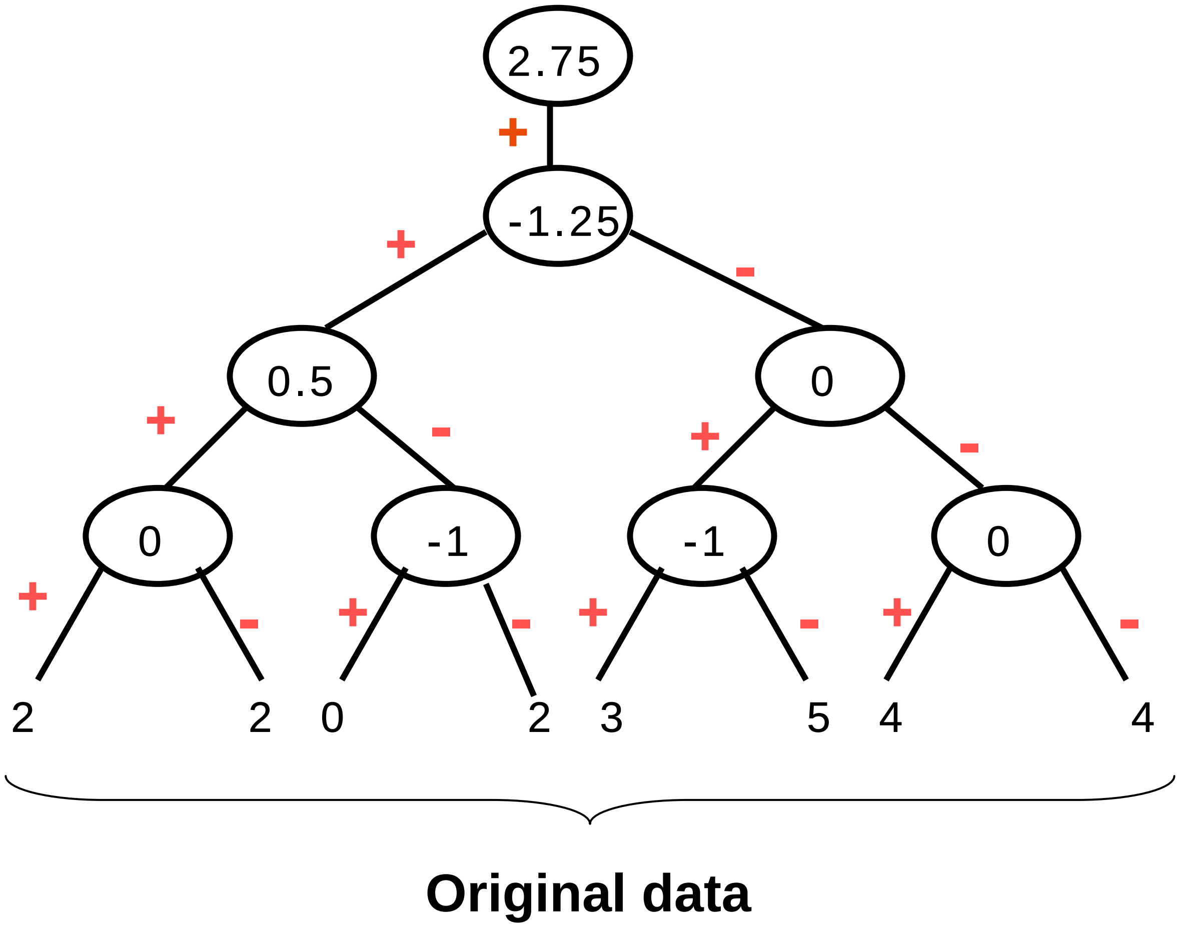
- Wavelets: mathematical tool for hierarchical decomposition of functions/signals
- Haar wavelets: simplest wavelet basis, easy to understand and implement
  - *Recursive pairwise averaging and differencing* at different resolutions

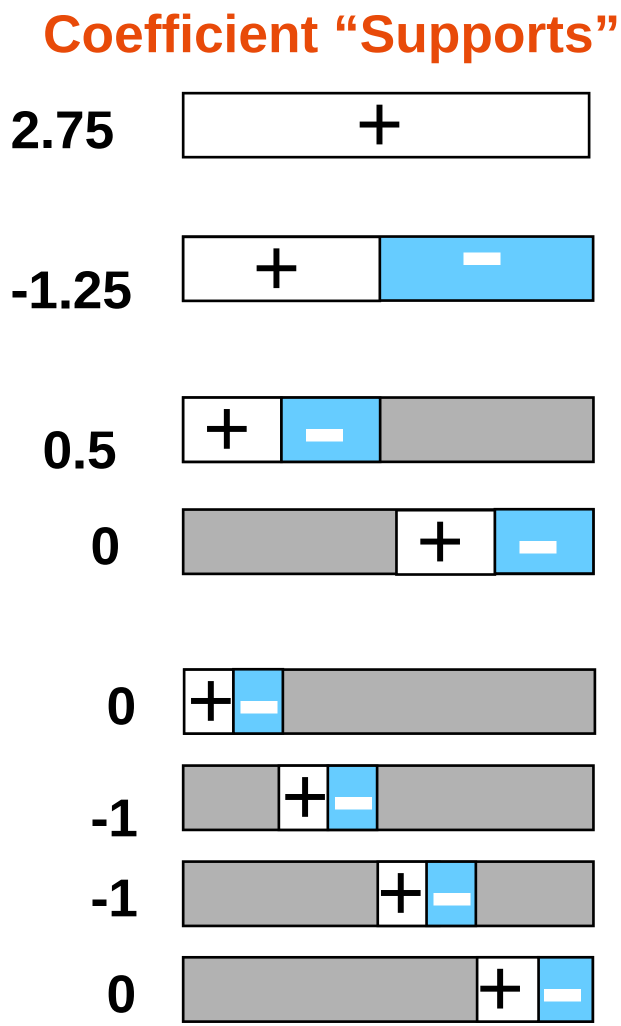
Resolution	Averages	Detail Coefficients
3	[2, 2, 0, 2, 3, 5, 4, 4]	----
2	[2, 1, 4, 4]	[0, -1, -1, 0]
1	[1.5, 4]	[0.5, 0]
0	[2.75]	[-1.25]

Haar wavelet decomposition: [2.75, -1.25, 0.5, 0, 0, -1, -1, 0]

Haar Wavelet Coefficients

- Hierarchical decomposition structure (a.k.a. "error tree")



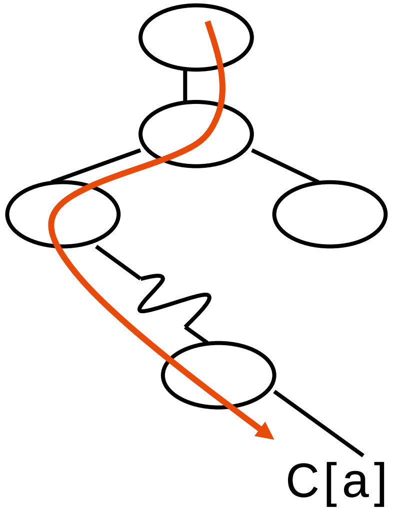


Wavelet-based Histograms [MVW98]

- Problem: range-query selectivity estimation
- Key idea: use a compact subset of Haar/linear wavelet coefficients for approximating the data distribution

Using Wavelet-based Histograms

- Selectivity estimation: sel(a<= X<= b) = C'[b] - C'[a-1]
  - C' is the (approximate) "reconstructed" cumulative distribution
  - Time: O(min{b, logN}), where b = size of wavelet synopsis (no. of coefficients), N= size of domain



- At most logN+1 coefficients are needed to reconstruct any C value

- Empirical results over synthetic data
  - Improvements over random sampling and histograms (MaxDiff)

Essential Readings

- **Data Mining Concepts and techniques. Chapter 2,3,4**
- **[Gray+96]** J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichaxt, and M. Venkatrao. "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals". In DMKD, pages 29-53. Kluwer Academic Publishers, 1997.
- **[JKM98]** H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. Sevcik, and T. Suel. "Optimal Histograms with Quality Guarantees". VLDB 1998

Optional Readings

- Bar+97] D. Barbara, W. DuMouchel, C. Faloutsos, P. J. Haas, J. M. Hellerstein, Y. Ioannidis, H. V. Jagadish, T. Johnson, R. Ng, , V. Poosala, K. A. Ross, and K. C. Sevcik. "The New Jersey data reduction report". Bulletin of the Technical Committee on Data Engineering, 20(4), 1997.
- [MVW98] Y. Matias, J.S. Vitter, and M. Wang. "Wavelet-based Histograms for Selectivity Estimation". ACM SIGMOD 1998
- [HG04]X. Li, J. Han, and H. Gonzalez, "High-Dimensional OLAP: A Minimal Cubing Approach", Proc. 2004 Int. Conf. on Very Large Data Bases (VLDB'04), Toronto, Canada, Aug. 2004.

Acknowledgment

- Part of these slides are adopted from the slides of the following tutorials
  - ***"Approximate Query Processing: Taming the Terabytes" (ppt)*** *(with Phillip Gibbons)*. 2001 International Conference on Very Large Databases (VLDB'2001), Roma, Italy, September 2001.

Reference

- [PIH96] V. Poosala, Y. Ioannidis, P. Haas, and E. Shekita. “Improved Histograms for Selectivity Estimation of Range Predicates”. ACM SIGMOD 1996.
- [MVW00] Y. Matias, J.S. Vitter, and M. Wang. “Dynamic Maintenance of Wavelet-based Histograms”. VLDB 2000

References (1)

- [AC99] A. Aboulnaga and S. Chaudhuri. "Self-Tuning Histograms: Building Histograms Without Looking at Data". ACM SIGMOD 1999.
- [AGM99] N. Alon, P. B. Gibbons, Y. Matias, M. Szegedy. "Tracking Join and Self-Join Sizes in Limited Storage". ACM PODS 1999.
- [AGP00] S. Acharya, P. B. Gibbons, and V. Poosala. "Congressional Samples for Approximate Answering of Group-By Queries". ACM SIGMOD 2000.
- [AGP99] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. "Join Synopses for Fast Approximate Query Answering". ACM SIGMOD 1999.
- [AMS96] N. Alon, Y. Matias, and M. Szegedy. "The Space Complexity of Approximating the Frequency Moments". ACM STOC 1996.
- [BCC00] A.L. Buchsbaum, D.F. Caldwell, K.W. Church, G.S. Fowler, and S. Muthukrishnan. "Engineering the Compression of Massive Tables: An Experimental Approach". SODA 2000.
  - Proposes exploiting simple (differential and combinational) data dependencies for effectively compressing data tables.
- [BCG01] N. Bruno, S. Chaudhuri, and L. Gravano. "STHoles: A Multidimensional Workload-Aware Histogram". ACM SIGMOD 2001.
- [BDF97] D. Barbara, W. DuMouchel, C. Faloutsos, P. J. Haas, J. M. Hellerstein, Y. Ioannidis, H. V. Jagadish, T. Johnson, R. Ng, V. Poosala, K. A. Ross, and K. C. Sevcik. "The New Jersey Data Reduction Report". IEEE Data Engineering bulletin, 1997.

References (2)

- [BFH75] Y.M.M. Bishop, S.E. Fienberg, and P.W. Holland. "Discrete Multivariate Analysis". The MIT Press, 1975.
- [BGR01] S. Babu, M. Garofalakis, and R. Rastogi. "SPARTAN: A Model-Based Semantic Compression System for Massive Data Tables". ACM SIGMOD 2001.
  - Proposes a novel, "model-based semantic compression" methodology that exploits mining models (like CaRT trees and clusters) to build compact, guaranteed-error synopses of massive data tables.
- [BKS99] B. Blohsfeld, D. Korus, and B. Seeger. "A Comparison of Selectivity Estimators for Range Queries on Metric Attributes". ACM SIGMOD 1999.
  - Studies the effectiveness of histograms, kernel-density estimators, and their hybrids for estimating the selectivity of range queries over metric attributes with large domains.
- [CCM00] M. Charlikar, S. Chaudhuri, R. Motwani, and V. Narasayya. "Towards Estimation Error Guarantees for Distinct Values". ACM PODS 2000.
- [CDD01] S. Chaudhuri, G. Das, M. Datar, R. Motwani, and V. Narasayya. "Overcoming Limitations of Sampling for Aggregation Queries". IEEE ICDE 2001.
  - Precursor to [CDN01]. Proposes a method for reducing sampling variance by collecting outliers to a separate "outlier index" and using a weighted sampling scheme for the remaining data.
- [CDN01] S. Chaudhuri, G. Das, and V. Narasayya. "A Robust, Optimization-Based Approach for Approximate Answering of Aggregate Queries". ACM SIGMOD 2001.
- [CGR00] K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim. "Approximate Query Processing Using Wavelets". VLDB 2000. (Full version to appear in The VLDB Journal)

References (3)

- [Chr84] S. Christodoulakis. “Implications of Certain Assumptions on Database Performance Evaluation”. ACM TODS 9(2), 1984.
- [CMN98] S. Chaudhuri, R. Motwani, and V. Narasayya. “Random Sampling for Histogram Construction: How much is enough?”. ACM SIGMOD 1998.
- [CMN99] S. Chaudhuri, R. Motwani, and V. Narasayya. “On Random Sampling over Joins”. ACM SIGMOD 1999.
- [CN97] S. Chaudhuri and V. Narasayya. “An Efficient, Cost-Driven Index Selection Tool for Microsoft SQL Server”. VLDB 1997.
- [CN98] S. Chaudhuri and V. Narasayya. “AutoAdmin “What-if” Index Analysis Utility”. ACM SIGMOD 1998.
- [Coc77] W.G. Cochran. “Sampling Techniques”. John Wiley & Sons, 1977.
- [Coh97] E. Cohen. “Size-Estimation Framework with Applications to Transitive Closure and Reachability”. JCSS, 1997.
- [CR94] C.M. Chen and N. Roussopoulos. “Adaptive Selectivity Estimation Using Query Feedback”. ACM SIGMOD 1994.
  - Presents a parametric, curve-fitting technique for approximating an attribute’s distribution based on query feedback.
- [DGR01] A. Deshpande, M. Garofalakis, and R. Rastogi. “Independence is Good: Dependency-Based Histogram Synopses for High-Dimensional Data”. ACM SIGMOD 2001.

References (4)

- [FK97] C. Faloutsos and I. Kamel. “Relaxing the Uniformity and Independence Assumptions Using the Concept of Fractal Dimension”. JCSS 55(2), 1997.
- [FM85] P. Flajolet and G.N. Martin. “Probabilistic counting algorithms for data base applications”. JCSS 31(2), 1985.
- [FMS96] C. Faloutsos, Y. Matias, and A. Silbershcatz. “Modeling Skewed Distributions Using Multifractals and the `80-20’ Law”. VLDB 1996.
  - Proposes the use of “multifractals” (i.e., 80/20 laws) to more accurately approximate the frequency distribution within histogram buckets.
- [GGM96] S. Ganguly, P.B. Gibbons, Y. Matias, and A. Silberschatz. “Bifocal Sampling for Skew-Resistant Join Size Estimation”. ACM SIGMOD 1996.
- [Gib01] P. B. Gibbons. “Distinct Sampling for Highly-Accurate Answers to Distinct Values Queries and Event Reports”. VLDB 2001.
- [GK01] M. Greenwald and S. Khanna. “Space-Efficient Online Computation of Quantile Summaries”. ACM SIGMOD 2001.
- [GKM01a] A.C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M.J. Strauss. “Optimal and Approximate Computation of Summary Statistics for Range Aggregates”. ACM PODS 2001.
  - Presents algorithms for building “range-optimal” histogram and wavelet synopses; that is, synopses that try to minimize the total error over all possible range queries in the data domain.

References (5)

- [GKM01b] A.C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M.J. Strauss. "Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries". VLDB 2001.
- [GKT00] D. Gunopulos, G. Kollios, V.J. Tsotras, and C. Domeniconi. "Approximating Multi-Dimensional Aggregate Range Queries over Real Attributes". ACM SIGMOD 2000.
- [GKS01a] J. Gehrke, F. Korn, and D. Srivastava. "On Computing Correlated Aggregates over Continual Data Streams". ACM SIGMOD 2001.
- [GKS01b] S. Guha, N. Koudas, and K. Shim. "Data Streams and Histograms". ACM STOC 2001.
- [GLR00] V. Ganti, M.L. Lee, and R. Ramakrishnan. "ICICLES: Self-Tuning Samples for Approximate Query Answering". VLDB 2000.
- [GM98] P. B. Gibbons and Y. Matias. "New Sampling-Based Summary Statistics for Improving Approximate Query Answers". ACM SIGMOD 1998.
  - Proposes the "concise sample" and "counting sample" techniques for improving the accuracy of sampling-based estimation for a given amount of space for the sample synopsis.
- [GMP97a] P. B. Gibbons, Y. Matias, and V. Poosala. "The Aqua Project White Paper". Bell Labs tech report, 1997.
- [GMP97b] P. B. Gibbons, Y. Matias, and V. Poosala. "Fast Incremental Maintenance of Approximate Histograms". VLDB 1997.

References (6)

- [GTK01] L. Getoor, B. Taskar, and D. Koller. "Selectivity Estimation using Probabilistic Relational Models". ACM SIGMOD 2001.
  - Proposes novel, Bayesian-network-based techniques for approximating joint data distributions in relational database systems.
- [HAR00] J. M. Hellerstein, R. Avnur, and V. Raman. "Informix under CONTROL: Online Query Processing". Data Mining and Knowledge Discovery Journal, 2000.
- [HH99] P. J. Haas and J. M. Hellerstein. "Ripple Joins for Online Aggregation". ACM SIGMOD 1999.
- [HHW97] J. M. Hellerstein, P. J. Haas, and H. J. Wang. "Online Aggregation". ACM SIGMOD 1997.
- [HNS95] P.J. Haas, J.F. Naughton, S. Seshadri, and L. Stokes. "Sampling-Based Estimation of the Number of Distinct Values of an Attribute". VLDB 1995.
  - Proposes and evaluates several sampling-based estimators for the number of distinct values in an attribute column.
- [HNS96] P.J. Haas, J.F. Naughton, S. Seshadri, and A. Swami. "Selectivity and Cost Estimation for Joins Based on Random Sampling". JCSS 52(3), 1996.
- [HOT88] W.C. Hou, Ozsoyoglu, and B.K. Taneja. "Statistical Estimators for Relational Algebra Expressions". ACM PODS 1988.
- [HOT89] W.C. Hou, Ozsoyoglu, and B.K. Taneja. "Processing Aggregate Relational Queries with Hard Time Constraints". ACM SIGMOD 1989.

References (7)

- [IC91] Y. Ioannidis and S. Christodoulakis. "On the Propagation of Errors in the Size of Join Results". ACM SIGMOD 1991.
- [IC93] Y. Ioannidis and S. Christodoulakis. "Optimal Histograms for Limiting Worst-Case Error Propagation in the Size of join Results". ACM TODS 18(4), 1993.
- [Ioa93] Y.E. Ioannidis. "Universality of Serial Histograms". VLDB 1993.
  - The above three papers propose and study serial histograms (i.e., histograms that bucket "neighboring" frequency values, and exploit results from majorization theory to establish their optimality wrt minimizing (extreme cases of) the error in multi-join queries.
- [IP95] Y. Ioannidis and V. Poosala. "Balancing Histogram Optimality and Practicality for Query Result Size Estimation". ACM SIGMOD 1995.
- [IP99] Y.E. Ioannidis and V. Poosala. "Histogram-Based Approximation of Set-Valued Query Answers". VLDB 1999.
- [JKM98] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. Sevcik, and T. Suel. "Optimal Histograms with Quality Guarantees". VLDB 1998.
- [JMN99] H. V. Jagadish, J. Madar, and R.T. Ng. "Semantic Compression and Pattern Extraction with Fascicles". VLDB 1999.
  - Discusses the use of "fascicles" (i.e., approximate data clusters) for the semantic compression of relational data.
- [KJF97] F. Korn, H.V. Jagadish, and C. Faloutsos. "Efficiently Supporting Ad-Hoc Queries in Large Datasets of Time Sequences". ACM SIGMOD 1997.

References (8)

- Proposes the use of SVD techniques for obtaining fast approximate answers from large time-series databases.

- [Koo80] R. P. Kooi. "The Optimization of Queries in Relational Databases". PhD thesis, Case Western Reserve University, 1980.
- [KW99] A.C. Konig and G. Weikum. "Combining Histograms and Parametric Curve Fitting for Feedback-Driven Query Result-Size Estimation". VLDB 1999.
  - Proposes the use of linear splines to better approximate the data and frequency distribution within histogram buckets.
- [Lau96] S.L. Lauritzen. "Graphical Models". Oxford Science, 1996.
- [LKC99] J.H. Lee, D.H. Kim, and C.W. Chung. "Multi-dimensional Selectivity Estimation Using Compressed Histogram Information". ACM SIGMOD 1999.
  - Proposes the use of the Discrete Cosine Transform (DCT) for compressing the information in multi-dimensional histogram buckets.
- [LM01] I. Lazaridis and S. Mehrotra. "Progressive Approximate Aggregate Queries with a Multi-Resolution Tree Structure". ACM SIGMOD 2001.
  - Proposes techniques for enhancing hierarchical multi-dimensional index structures to enable approximate answering of aggregate queries with progressively improving accuracy.
- [LNS90] R.J. Lipton, J.F. Naughton, and D.A. Schneider. "Practical Selectivity Estimation through Adaptive Sampling". ACM SIGMOD 1990.
  - Presents an adaptive, sequential sampling scheme for estimating the selectivity of relational equi-join operators.

References (9)

- [LNS93] R.J. Lipton, J.F. Naughton, D.A. Schneider, and S. Seshadri. "Efficient sampling strategies for relational database operators", Theoretical Comp. Science, 1993.
- [MD88] M. Muralikrishna and D.J. DeWitt. "Equi-Depth Histograms for Estimating Selectivity Factors for Multi-Dimensional Queries". ACM SIGMOD 1988.
- [MPS99] S. Muthukrishnan, V. Poosala, and T. Suel. "On Rectangular Partitionings in Two Dimensions: Algorithms, Complexity, and Applications". ICDT 1999.
- [MVW98] Y. Matias, J.S. Vitter, and M. Wang. "Wavelet-based Histograms for Selectivity Estimation". ACM SIGMOD 1998.
- [MVW00] Y. Matias, J.S. Vitter, and M. Wang. "Dynamic Maintenance of Wavelet-based Histograms". VLDB 2000.
- [NS90] J.F. Naughton and S. Seshadri. "On Estimating the Size of Projections". ICDT 1990.
  - Presents adaptive-sampling-based techniques and estimators for approximating the result size of a relational projection operation.
- [Olk93] F. Olken. "Random Sampling from Databases". PhD thesis, U.C. Berkeley, 1993.
- [OR92] F. Olken and D. Rotem. "Maintenance of Materialized Views of Sampling Queries". IEEE ICDE 1992.
- [PI97] V. Poosala and Y. Ioannidis. "Selectivity Estimation Without the Attribute Value Independence Assumption". VLDB 1997.

References (10)

- [PIH96] V. Poosala, Y. Ioannidis, P. Haas, and E. Shekita. "Improved Histograms for Selectivity Estimation of Range Predicates". ACM SIGMOD 1996.
- [PSC84] G. Piatetsky-Shapiro and C. Connell. "Accurate Estimation of the Number of Tuples Satisfying a Condition". ACM SIGMOD 1984.
- [Poo97] V. Poosala. "Histogram-Based Estimation Techniques in Database Systems". PhD Thesis, Univ. of Wisconsin, 1997.
- [RTG98] Y. Rubner, C. Tomasi, and L. Guibas. "A Metric for Distributions with Applications to Image Databases". IEEE Intl. Conf. On Computer Vision 1998.
- [SAC79] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. T. Price. "Access Path Selection in a Relational Database Management System". ACM SIGMOD 1979.
- [SDS96] E.J. Stollnitz, T.D. DeRose, and D.H. Salesin. "Wavelets for Computer Graphics, A Primer". Morgan-Kauffman Publishers Inc., 1996.
- [SFB99] J. Shanmugasundaram, U. Fayyad, and P.S. Bradley. "Compressed Data Cubes for OLAP Aggregate Query Approximation on Continuous Dimensions". KDD 1999.
  - Discusses the use of mixture models composed of multi-variate Gaussians for building compact models of OLAP data cubes and approximating range-sum query answers.
- [V85] J. S. Vitter. "Random Sampling with a Reservoir". ACM TOMS, 1985.

References (11)

- [VL93] S. V. Vrbsky and J. W. S. Liu. "Approximate—A Query Processor that Produces Monotonically Improving Approximate Answers". IEEE TKDE, 1993.
  - Uses class hierarchies on the data to iteratively fetch blocks relevant to the answer, producing tuples certain to be in the answer while narrowing the possible classes containing the answer.
- [VW99] J.S. Vitter and M. Wang. "Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets". ACM SIGMOD 1999.