PURDUE
UNIVERSITY

CS590D: Data Mining

Chris Clifton

January 13, 2005

Data Preparation

Indiana Center for Database Systems

Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Why Data Preprocessing?

- Data in the real world is dirty
  - incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    - e.g., occupation=“”
  - noisy: containing errors or outliers
    - e.g., Salary=“-10”
  - inconsistent: containing discrepancies in codes or names
    - e.g., Age=“42” Birthday=“03/07/1997”
    - e.g., Was rating “1,2,3”, now rating “A, B, C”
    - e.g., discrepancy between duplicate records

Why Is Data Dirty?

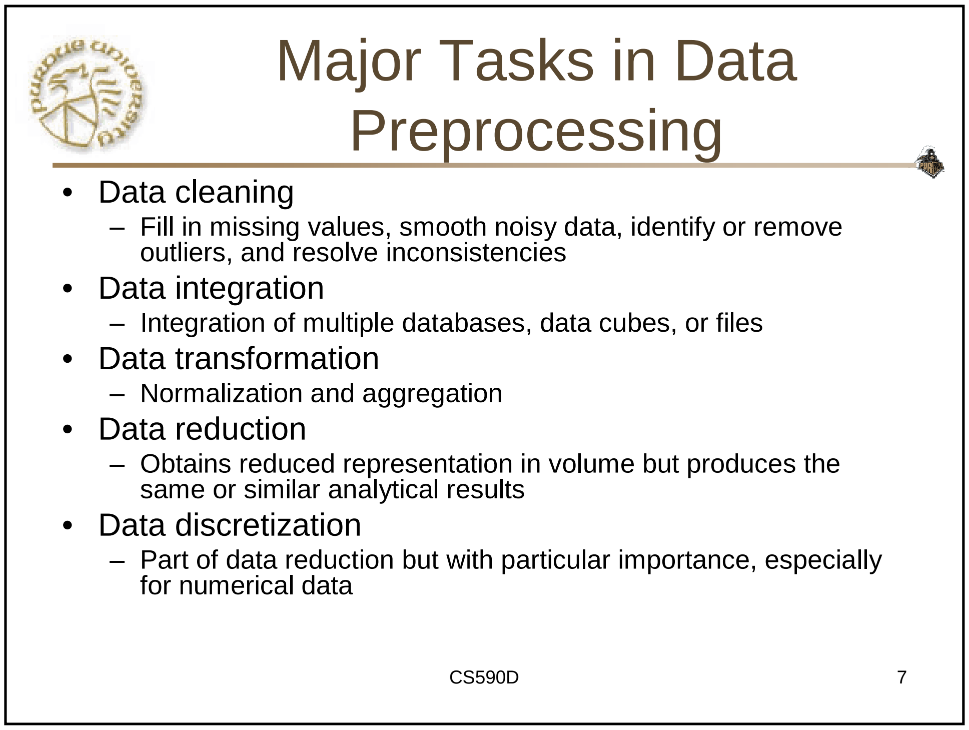
- Incomplete data comes from
  - n/a data value when collected
  - different consideration between the time when the data was collected and when it is analyzed.
  - human/hardware/software problems
- Noisy data comes from the process of data
  - collection
  - entry
  - transmission
- Inconsistent data comes from
  - Different data sources
  - Functional dependency violation

Why Is Data Preprocessing Important?

- No quality data, no quality mining results!
  - Quality decisions must be based on quality data
    - e.g., duplicate or missing data may cause incorrect or even misleading statistics.
  - Data warehouse needs consistent integration of quality data
- Data extraction, cleaning, and transformation comprises the majority of the work of building a data warehouse. —Bill Inmon

Multi-Dimensional Measure of Data Quality

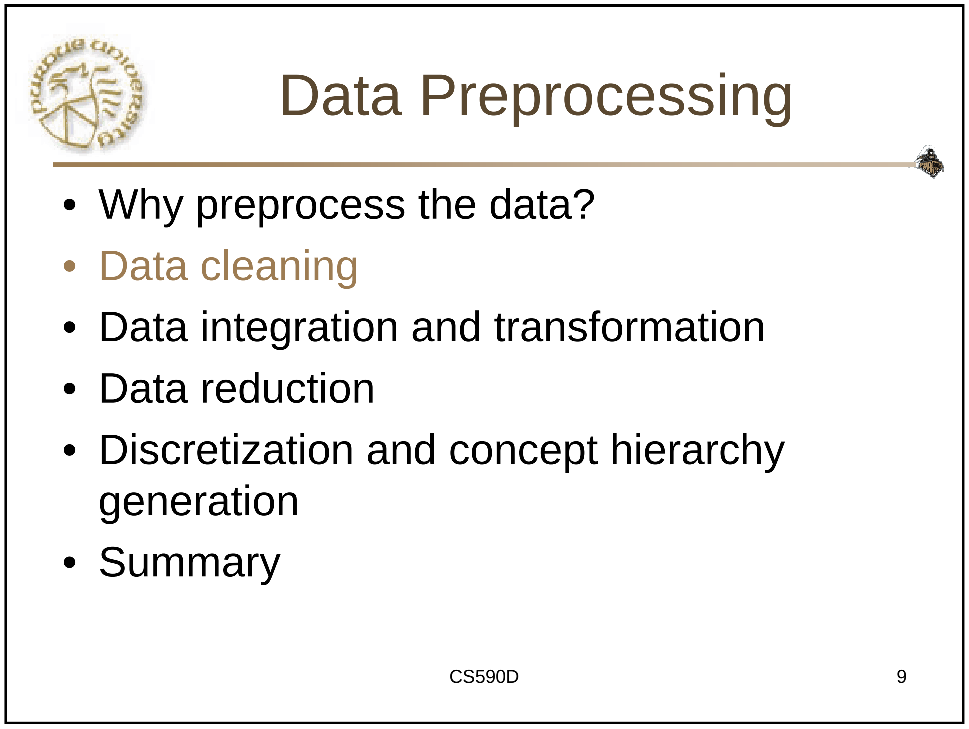
- A well-accepted multidimensional view:
  - Accuracy
  - Completeness
  - Consistency
  - Timeliness
  - Believability
  - Value added
  - Interpretability
  - Accessibility
- Broad categories:
  - intrinsic, contextual, representational, and accessibility.



Major Tasks in Data Preprocessing

- Data cleaning
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- Data integration
  - Integration of multiple databases, data cubes, or files
- Data transformation
  - Normalization and aggregation
- Data reduction
  - Obtains reduced representation in volume but produces the same or similar analytical results
- Data discretization
  - Part of data reduction but with particular importance, especially for numerical data

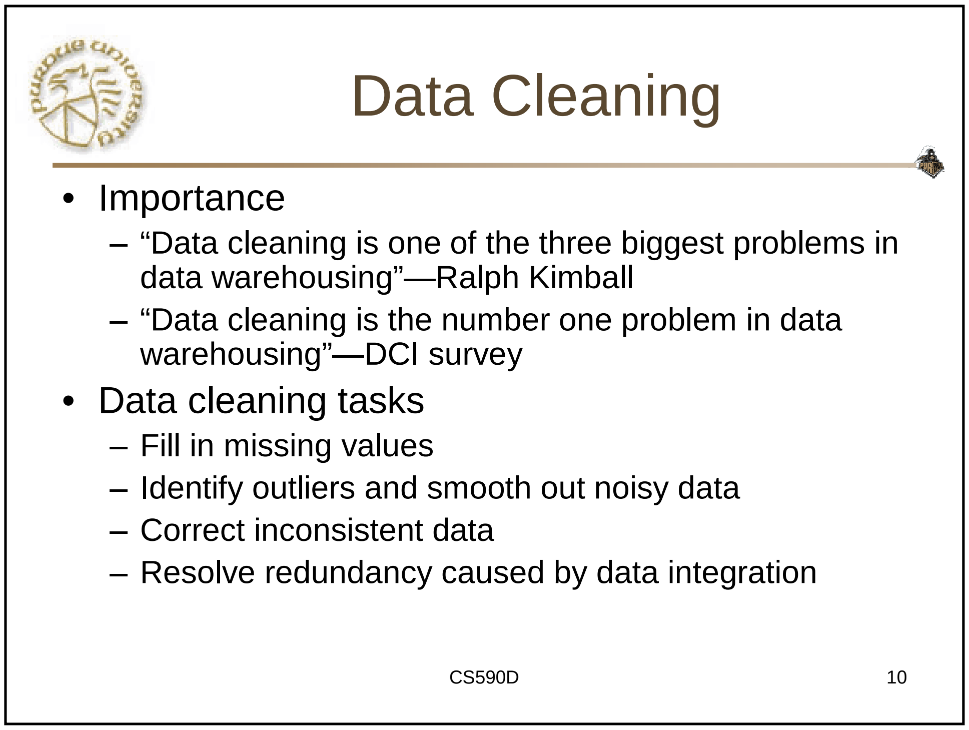
CS590D 7



Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

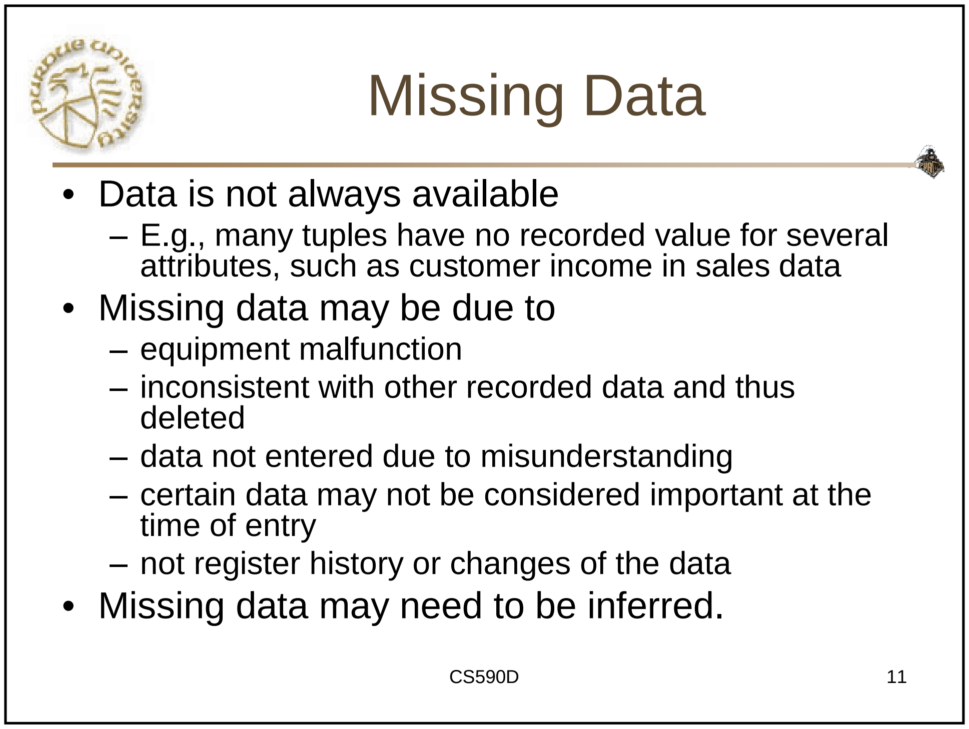
CS590D 9



Data Cleaning

- Importance
  - “Data cleaning is one of the three biggest problems in data warehousing”—Ralph Kimball
  - “Data cleaning is the number one problem in data warehousing”—DCI survey
- Data cleaning tasks
  - Fill in missing values
  - Identify outliers and smooth out noisy data
  - Correct inconsistent data
  - Resolve redundancy caused by data integration

CS590D 10



Missing Data

- Data is not always available
  - E.g., many tuples have no recorded value for several attributes, such as customer income in sales data
- Missing data may be due to
  - equipment malfunction
  - inconsistent with other recorded data and thus deleted
  - data not entered due to misunderstanding
  - certain data may not be considered important at the time of entry
  - not register history or changes of the data
- Missing data may need to be inferred.

CS590D 11

How to Handle Missing Data?

- Ignore the tuple: usually done when class label is missing (assuming the tasks in classification—not effective when the percentage of missing values per attribute varies considerably.
- Fill in the missing value manually: tedious + infeasible?
- Fill in it automatically with
  - a global constant : e.g., “unknown”, a new class?!
  - the attribute mean
  - the attribute mean for all samples belonging to the same class: smarter
  - the most probable value: inference-based such as Bayesian formula or decision tree

Noisy Data

- Noise: random error or variance in a measured variable
- Incorrect attribute values may due to
  - faulty data collection instruments
  - data entry problems
  - data transmission problems
  - technology limitation
  - inconsistency in naming convention
- Other data problems which requires data cleaning
  - duplicate records
  - incomplete data
  - inconsistent data

How to Handle Noisy Data?

- Binning method:
  - first sort data and partition into (equi-depth) bins
  - then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.
- Clustering
  - detect and remove outliers
- Combined computer and human inspection
  - detect suspicious values and check by human (e.g., deal with possible outliers)
- Regression
  - smooth by fitting the data into regression functions

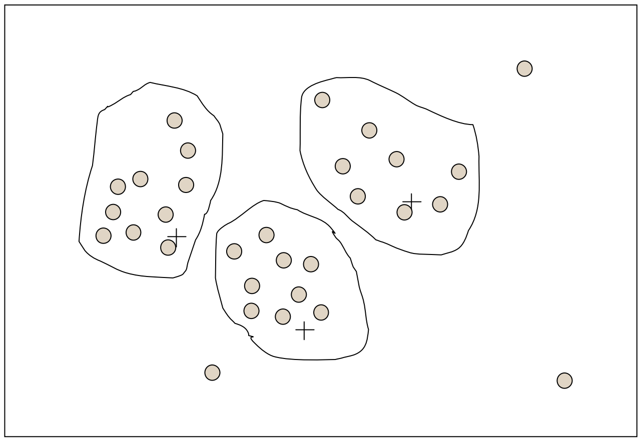
Simple Discretization Methods: Binning

- Equal-width (distance) partitioning:
  - Divides the range into N intervals of equal size: uniform grid
  - if A and B are the lowest and highest values of the attribute, the width of intervals will be: W = (B –A)/N.
  - The most straightforward, but outliers may dominate presentation
  - Skewed data is not handled well.
- Equal-depth (frequency) partitioning:
  - Divides the range into N intervals, each containing approximately same number of samples
  - Good data scaling
  - Managing categorical attributes can be tricky.

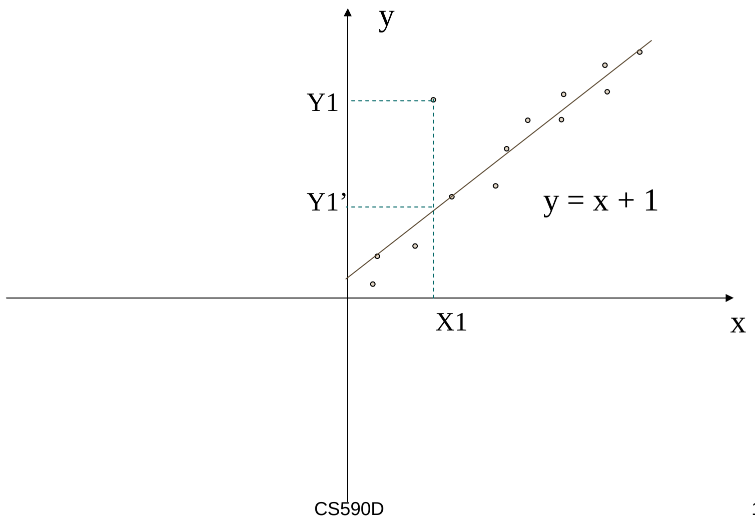
Binning Methods for Data Smoothing

- Sorted data (e.g., by price)
  - 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34
- Partition into (equi-depth) bins:
  - Bin 1: 4, 8, 9, 15
  - Bin 2: 21, 21, 24, 25
  - Bin 3: 26, 28, 29, 34
- Smoothing by bin means:
  - Bin 1: 9, 9, 9, 9
  - Bin 2: 23, 23, 23, 23
  - Bin 3: 29, 29, 29, 29
- Smoothing by bin boundaries:
  - Bin 1: 4, 4, 4, 15
  - Bin 2: 21, 21, 25, 25
  - Bin 3: 26, 26, 26, 34

Cluster Analysis



Regression



Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Data Integration

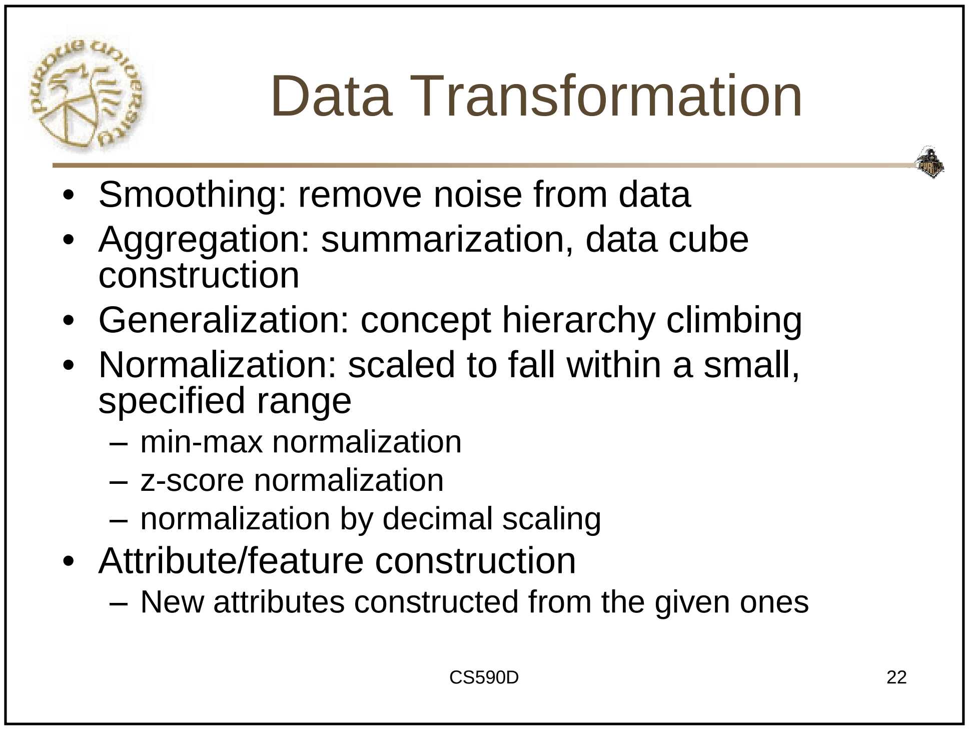
- Data integration:
  - combines data from multiple sources into a coherent store
- Schema integration
  - integrate metadata from different sources
  - Entity identification problem: identify real world entities from multiple data sources, e.g., A.cust-id ≡ B.cust-#
- Detecting and resolving data value conflicts
  - for the same real world entity, attribute values from different sources are different
  - possible reasons: different representations, different scales, e.g., metric vs. British units

CS590D 20

Handling Redundancy in Data Integration

- Redundant data occur often when integration of multiple databases
  - The same attribute may have different names in different databases
  - One attribute may be a “derived” attribute in another table, e.g., annual revenue
- Redundant data may be able to be detected by correlational analysis
- Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality

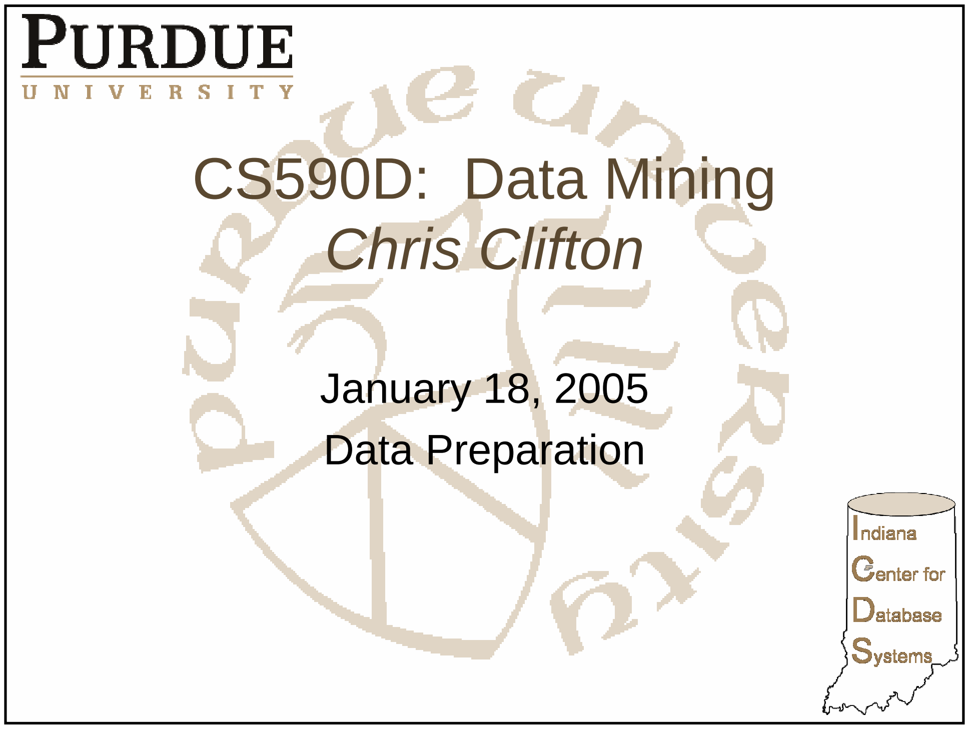
CS590D 21



Data Transformation

- Smoothing: remove noise from data
- Aggregation: summarization, data cube construction
- Generalization: concept hierarchy climbing
- Normalization: scaled to fall within a small, specified range
  - min-max normalization
  - z-score normalization
  - normalization by decimal scaling
- Attribute/feature construction
  - New attributes constructed from the given ones

CS590D 22



PURDUE UNIVERSITY

CS590D: Data Mining

Chris Clifton

January 18, 2005

Data Preparation

Indiana Center for Database Systems

Data Transformation: Normalization

- min-max normalization

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

- z-score normalization

$$v' = \frac{v - mean_A}{stand\_dev_A}$$

- normalization by decimal scaling

$$v' = \frac{v}{10^j}$$ Where j is the smallest integer such that Max(|v'|)<1

Z-Score (Example)

v	v'			v	v'		
0.18	-0.84	Avg	0.68	20	-.26	Avg	34.3
0.60	-0.14	sdev	0.59	40	.11	sdev	55.9
0.52	-0.27			5	.55		
0.25	-0.72			70	4		
0.80	0.20			32	-.05		
0.55	-0.22			8	-.48		
0.92	0.40			5	-.53		
0.21	-0.79			15	-.35		
0.64	-0.07			250	3.87		
0.20	-0.80			32	-.05		
0.63	-0.09			18	-.30		
0.70	0.04			10	-.44		
0.67	-0.02			-14	-.87		
0.58	-0.17			22	-.23		
0.98	0.50			45	.20		
0.81	0.22			60	.47		
0.10	-0.97			-5	-.71		
0.82	0.24			7	-.49		
0.50	-0.30			2	-.58		
3.00	3.87			4	-.55		

Data Preprocessing

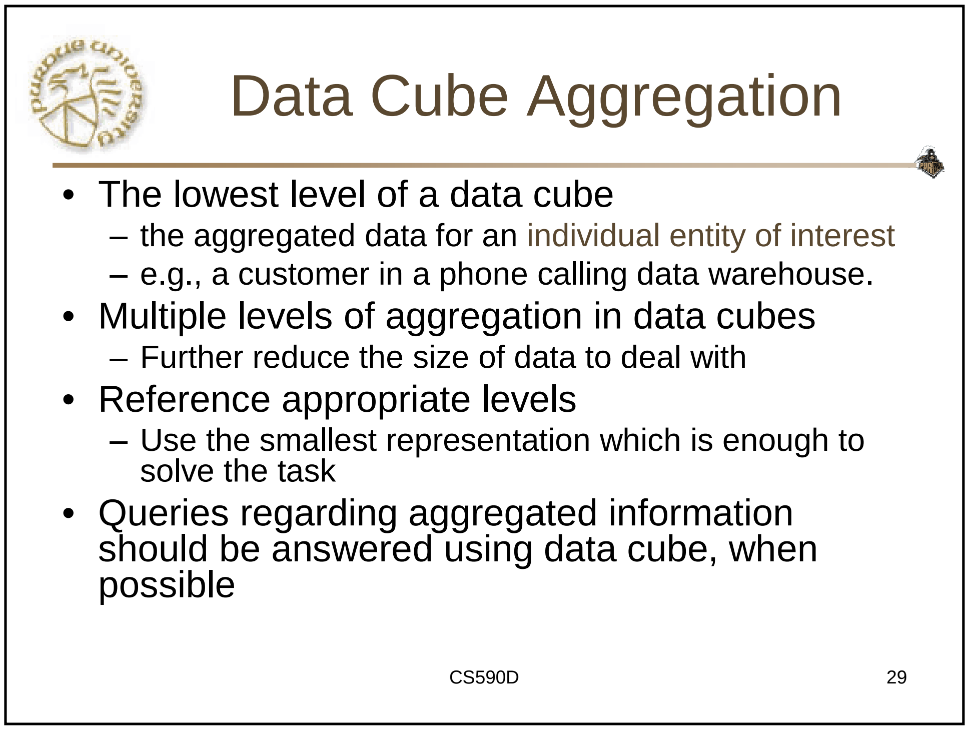
- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

CS590D 27

Data Reduction Strategies

- A data warehouse may store terabytes of data
  – Complex data analysis/mining may take a very long time to run on the complete data set
- Data reduction
  – Obtain a reduced representation of the data set that is much smaller in volume but yet produce the same (or almost the same) analytical results
- Data reduction strategies
  – Data cube aggregation
  – Dimensionality reduction — remove unimportant attributes
  – Data Compression
  – Numerosity reduction — fit data into models
  – Discretization and concept hierarchy generation

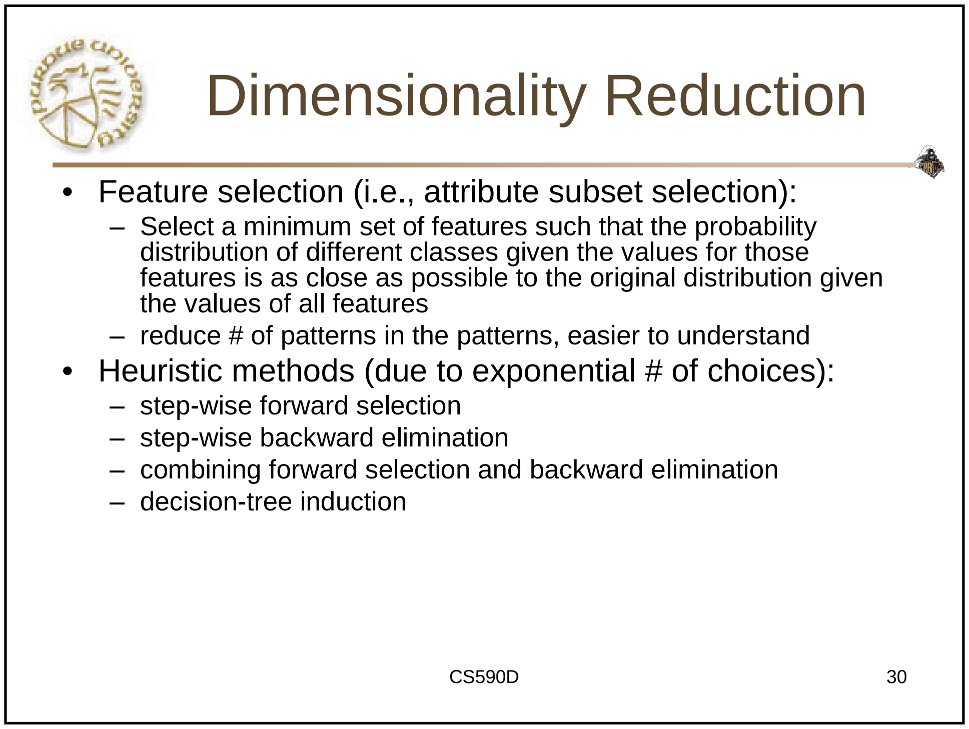
CS590D 28



Data Cube Aggregation

- The lowest level of a data cube
  - the aggregated data for an individual entity of interest
  - e.g., a customer in a phone calling data warehouse.
- Multiple levels of aggregation in data cubes
  - Further reduce the size of data to deal with
- Reference appropriate levels
  - Use the smallest representation which is enough to solve the task
- Queries regarding aggregated information should be answered using data cube, when possible

CS590D 29



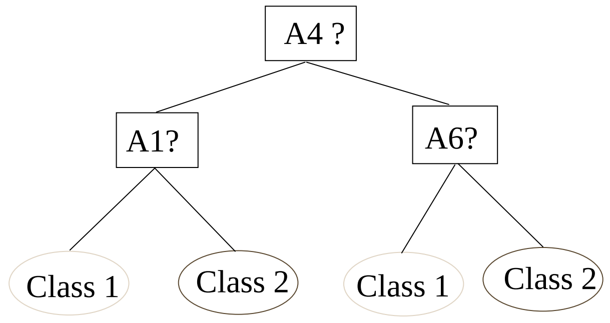
Dimensionality Reduction

- Feature selection (i.e., attribute subset selection):
  - Select a minimum set of features such that the probability distribution of different classes given the values for those features is as close as possible to the original distribution given the values of all features
  - reduce # of patterns in the patterns, easier to understand
- Heuristic methods (due to exponential # of choices):
  - step-wise forward selection
  - step-wise backward elimination
  - combining forward selection and backward elimination
  - decision-tree induction

CS590D 30

Example of Decision Tree Induction

Initial attribute set:
{A1, A2, A3, A4, A5, A6}

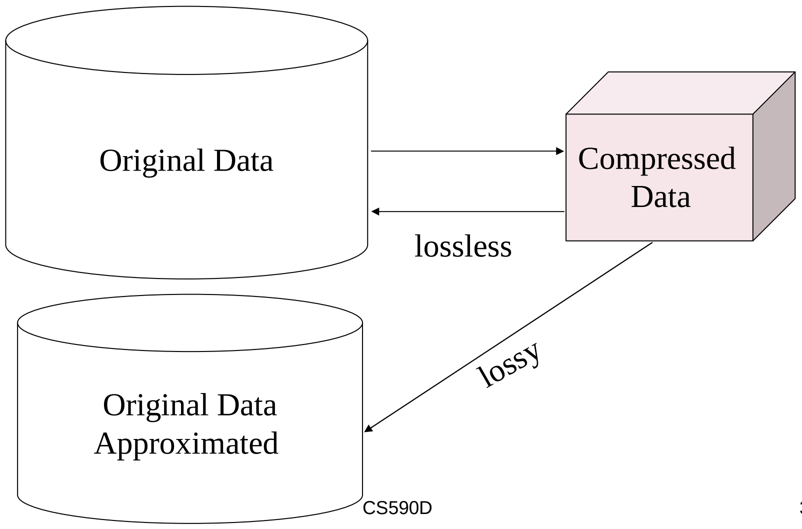


-------> Reduced attribute set: {A1, A4, A6}

Data Compression

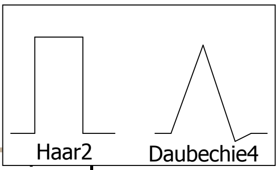
- String compression
  - There are extensive theories and well-tuned algorithms
  - Typically lossless
  - But only limited manipulation is possible without expansion
- Audio/video compression
  - Typically lossy compression, with progressive refinement
  - Sometimes small fragments of signal can be reconstructed without reconstructing the whole
- Time sequence is not audio
  - Typically short and vary slowly with time

Data Compression



CS590D 34

Wavelet Transformation

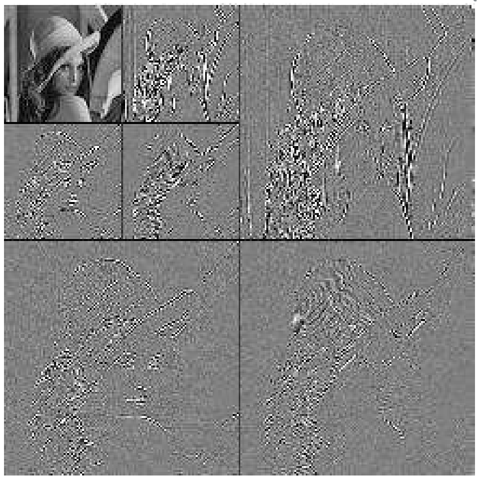


- Discrete wavelet transform (DWT): linear signal processing, multiresolutional analysis
- Compressed approximation: store only a small fraction of the strongest of the wavelet coefficients
- Similar to discrete Fourier transform (DFT), but better lossy compression, localized in space
- Method:
  - Length, L, must be an integer power of 2 (padding with 0s, when necessary)
  - Each transform has 2 functions: smoothing, difference
  - Applies to pairs of data, resulting in two set of data of length L/2
  - Applies two functions recursively, until reaches the desired length

CS590D 35

DWT for Image Compression

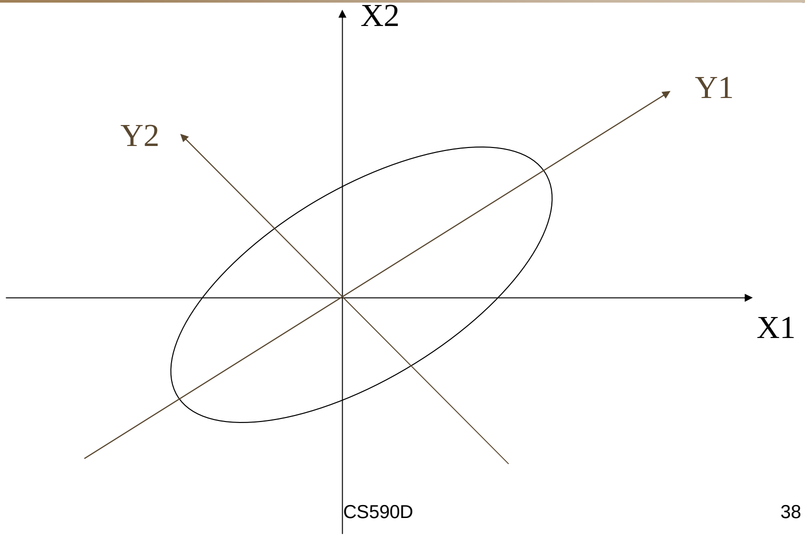
$$\begin{tikzcd}
 & & & \text{Image} \arrow[dl] \arrow[dr] & \\
 & & \text{Low Pass} \arrow[dl] \arrow[dr] & & \text{High Pass} \\
 & \text{Low Pass} \arrow[dl] \arrow[dr] & & \text{High Pass} & \\
\text{Low Pass} & & \text{High Pass} & &
\end{tikzcd}$$



Principal Component Analysis

- Given N data vectors from k-dimensions, find c ≤ k orthogonal vectors that can be best used to represent data
  - The original data set is reduced to one consisting of N data vectors on c principal components (reduced dimensions)
- Each data vector is a linear combination of the c principal component vectors
- Works for numeric data only
- Used when the number of dimensions is large

Principal Component Analysis



Numerosity Reduction

- Parametric methods
  - Assume the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers)
  - Log-linear models: obtain value at a point in m-D space as the product on appropriate marginal subspaces
- Non-parametric methods
  - Do not assume models
  - Major families: histograms, clustering, sampling

Regression and Log-Linear Models

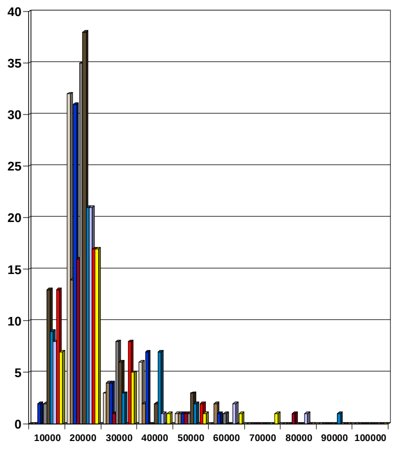
- Linear regression: Data are modeled to fit a straight line
  - Often uses the least-square method to fit the line
- Multiple regression: allows a response variable Y to be modeled as a linear function of multidimensional feature vector
- Log-linear model: approximates discrete multidimensional probability distributions

Regress Analysis and Log-Linear Models

- Linear regression: $Y = \alpha + \beta X$
  - Two parameters , α and β specify the line and are to be estimated by using the data at hand.
  - using the least squares criterion to the known values of *Y1, Y2, …, X1, X2, ….*
- Multiple regression: $Y = b0 + b1\ X1 + b2\ X2$.
  - Many nonlinear functions can be transformed into the above.
- Log-linear models:
  - The multi-way table of joint probabilities is approximated by a product of lower-order tables.
  - Probability: $p(a, b, c, d) = \alpha ab\ \beta ac \chi ad\ \delta bcd$

Histograms

- A popular data reduction technique
- Divide data into buckets and store average (sum) for each bucket
- Can be constructed optimally in one dimension using dynamic programming
- Related to quantization problems.



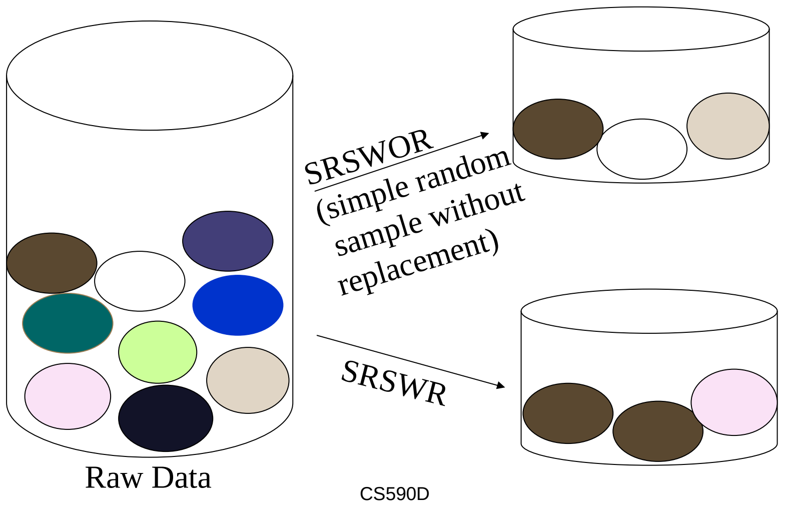
Clustering

- Partition data set into clusters, and one can store cluster representation only
- Can be very effective if data is clustered but not if data is “smeared”
- Can have hierarchical clustering and be stored in multi-dimensional index tree structures
- There are many choices of clustering definitions and clustering algorithms, further detailed in Chapter 8

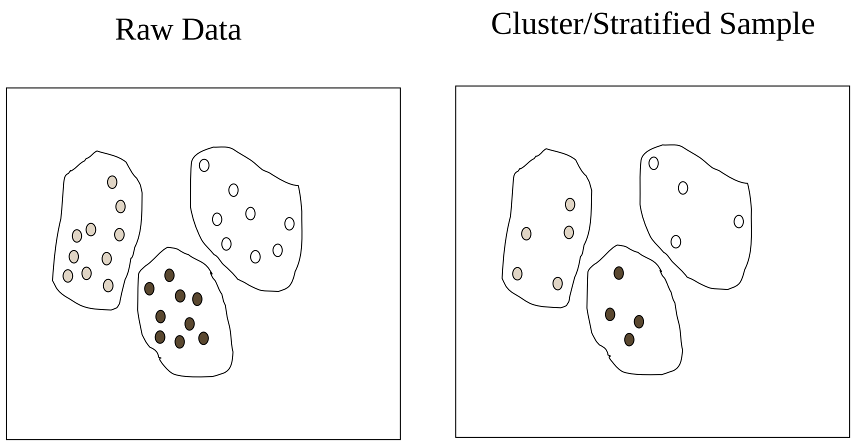
Sampling

- Allow a mining algorithm to run in complexity that is potentially sub-linear to the size of the data
- Choose a representative subset of the data
  - Simple random sampling may have very poor performance in the presence of skew
- Develop adaptive sampling methods
  - Stratified sampling:
    - Approximate the percentage of each class (or subpopulation of interest) in the overall database
    - Used in conjunction with skewed data
- Sampling may not reduce database I/Os (page at a time).

Sampling

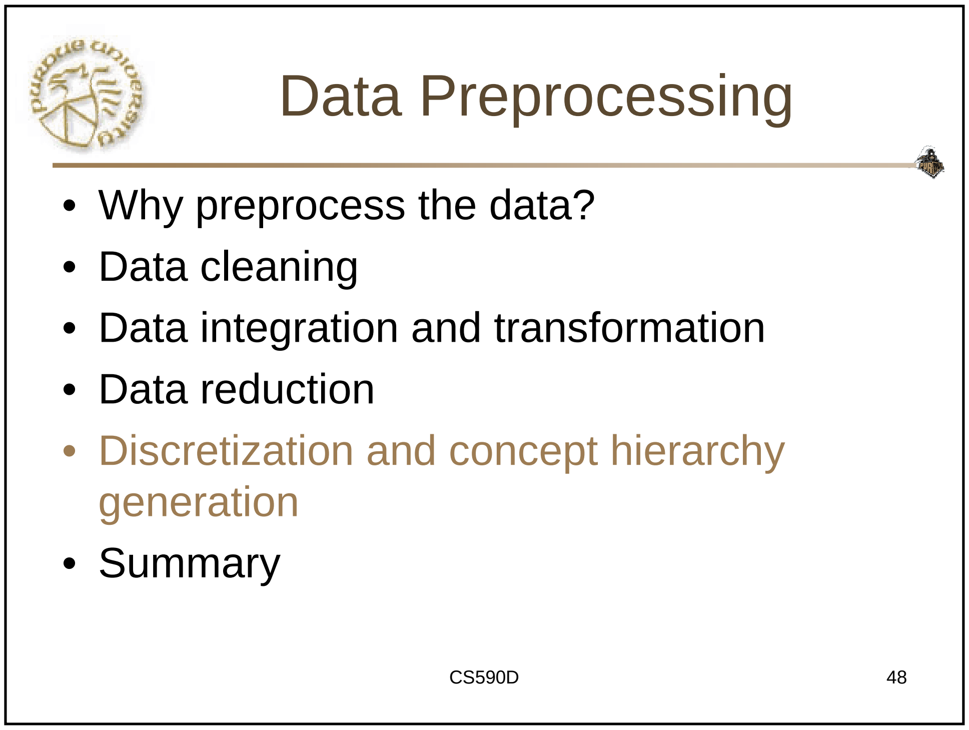


Sampling



Hierarchical Reduction

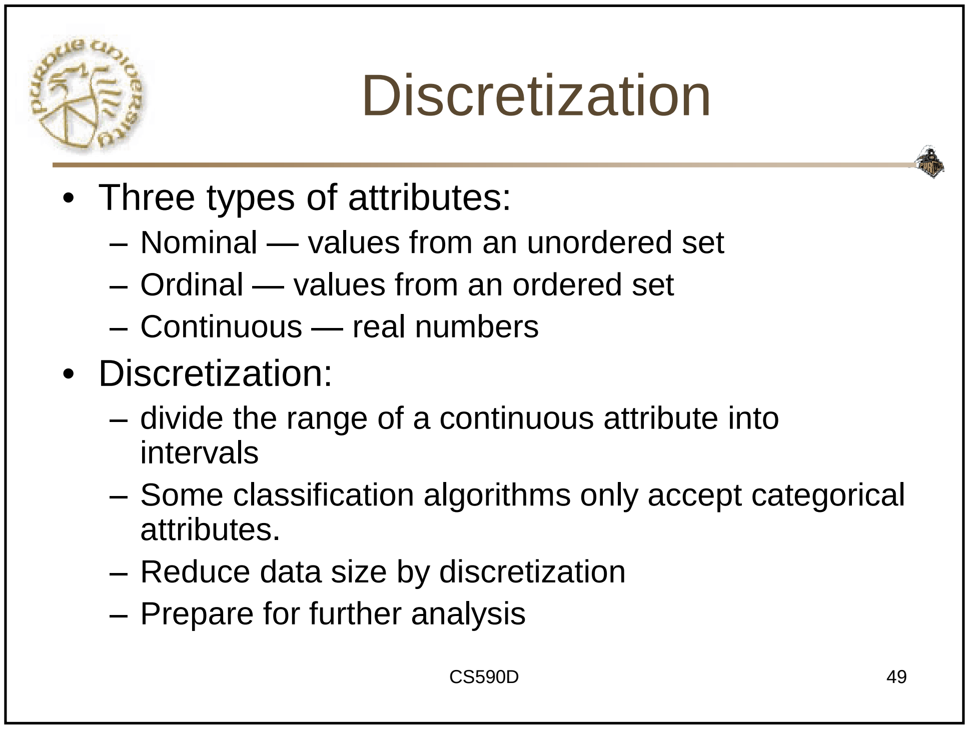
- Use multi-resolution structure with different degrees of reduction
- Hierarchical clustering is often performed but tends to define partitions of data sets rather than “clusters”
- Parametric methods are usually not amenable to hierarchical representation
- Hierarchical aggregation
  - An index tree hierarchically divides a data set into partitions by value range of some attributes
  - Each partition can be considered as a bucket
  - Thus an index tree with aggregates stored at each node is a hierarchical histogram



Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

CS590D 48



Discretization

- Three types of attributes:
  - Nominal — values from an unordered set
  - Ordinal — values from an ordered set
  - Continuous — real numbers
- Discretization:
  - divide the range of a continuous attribute into intervals
  - Some classification algorithms only accept categorical attributes.
  - Reduce data size by discretization
  - Prepare for further analysis

CS590D 49

Discretization and Concept hierachy

- Discretization
  - reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals. Interval labels can then be used to replace actual data values
- Concept hierarchies
  - reduce the data by collecting and replacing low level concepts (such as numeric values for the attribute age) by higher level concepts (such as young, middle-aged, or senior)

Discretization and Concept Hierarchy Generation for Numeric Data

- Binning (see sections before)
- Histogram analysis (see sections before)
- Clustering analysis (see sections before)
- Entropy-based discretization
- Segmentation by natural partitioning

Definition of Entropy

- Entropy $H(X) = \sum_{x \in A_X} -P(x) \log_2 P(x)$
- Example: Coin Flip
  - *A_X = {heads, tails}*
  - *P(heads) = P(tails) = ½*
  - *½ log$_2$(½) = ½ * - 1*
  - *H(X) = 1*
- What about a two-headed coin?
- Conditional Entropy: $H(X \mid Y) = \sum_{y \in A_Y} P(y) H(X \mid y)$

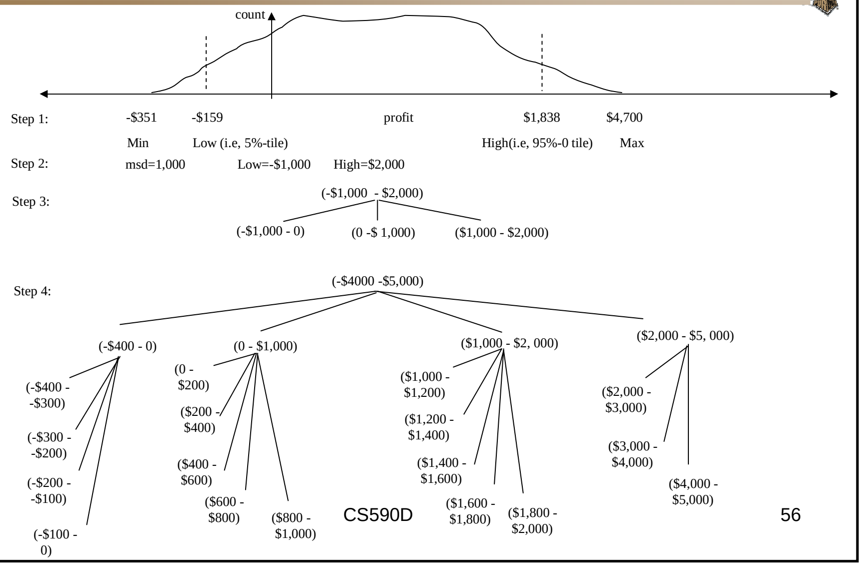
Entropy-Based Discretization

- Given a set of samples S, if S is partitioned into two intervals S1 and S2 using boundary T, the entropy after partitioning is
$$H(S,T) = \frac{|S_1|}{|S|} H(S_1) + \frac{|S_2|}{|S|} H(S_2)$$
- The boundary that minimizes the entropy function over all possible boundaries is selected as a binary discretization.
- The process is recursively applied to partitions obtained until some stopping criterion is met, e.g.,
$$H(S) - H(T,S) > \delta$$
- Experiments show that it may reduce data size and improve classification accuracy

Segmentation by Natural Partitioning

- A simply 3-4-5 rule can be used to segment numeric data into relatively uniform, “natural” intervals.
  - If an interval covers 3, 6, 7 or 9 distinct values at the most significant digit, partition the range into 3 equi-width intervals
  - If it covers 2, 4, or 8 distinct values at the most significant digit, partition the range into 4 intervals
  - If it covers 1, 5, or 10 distinct values at the most significant digit, partition the range into 5 intervals

Example of 3-4-5 Rule

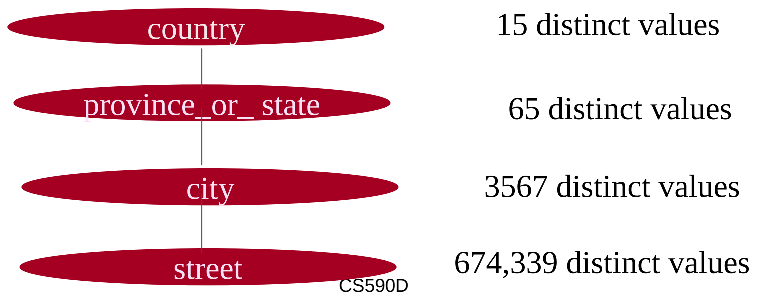


Concept Hierarchy Generation for Categorical Data

- Specification of a partial ordering of attributes explicitly at the schema level by users or experts
  - street<city<state<country
- Specification of a portion of a hierarchy by explicit data grouping
  - {Urbana, Champaign, Chicago}<Illinois
- Specification of a set of attributes.
  - System automatically generates partial ordering by analysis of the number of distinct values
  - E.g., street < city <state < country
- Specification of only a partial set of attributes
  - E.g., only street < city, not others

Automatic Concept Hierarchy Generation

- Some concept hierarchies can be automatically generated based on the analysis of the number of distinct values per attribute in the given data set
  - The attribute with the most distinct values is placed at the lowest level of the hierarchy
  - Note: Exception—weekday, month, quarter, year



Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

CS590D 59

Summary

- Data preparation is a big issue for both warehousing and mining
- Data preparation includes
  - Data cleaning and data integration
  - Data reduction and feature selection
  - Discretization
- A lot a methods have been developed but still an active area of research

CS590D 60

References

- E. Rahm and H. H. Do. Data Cleaning: Problems and Current Approaches. IEEE Bulletin of the Technical Committee on Data Engineering. Vol.23, No.4
- D. P. Ballou and G. K. Tayi. Enhancing data quality in data warehouse environments. Communications of ACM, 42:73-78, 1999.
- H.V. Jagadish et al., Special Issue on Data Reduction Techniques. Bulletin of the Technical Committee on Data Engineering, 20(4), December 1997.
- A. Maydanchik, Challenges of Efficient Data Cleansing (DM Review - Data Quality resource portal)
- D. Pyle. Data Preparation for Data Mining. Morgan Kaufmann, 1999.
- D. Quass. A Framework for research in Data Cleaning. (Draft 1999)
- V. Raman and J. Hellerstein. Potters Wheel: An Interactive Framework for Data Cleaning and Transformation, VLDB'2001.
- T. Redman. Data Quality: Management and Technology. Bantam Books, New York, 1992.
- Y. Wand and R. Wang. Anchoring data quality dimensions ontological foundations. Communications of ACM, 39:86-95, 1996.
- R. Wang, V. Storey, and C. Firth. A framework for analysis of data quality research. IEEE Trans. Knowledge and Data Engineering, 7:623-640, 1995.
- http://www.cs.ucla.edu/classes/spring01/cs240b/notes/data-integration1.pdf

PURDUE UNIVERSITY

CS590D: Data Mining

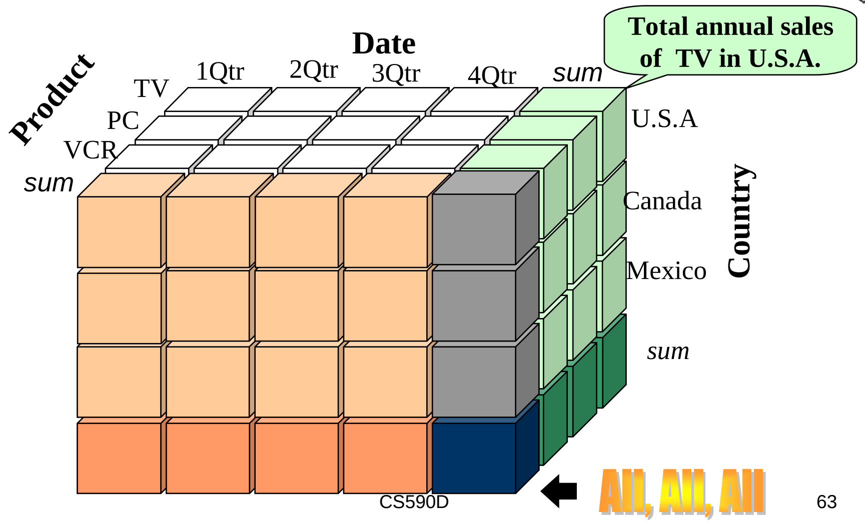
Chris Clifton

January 20, 2005

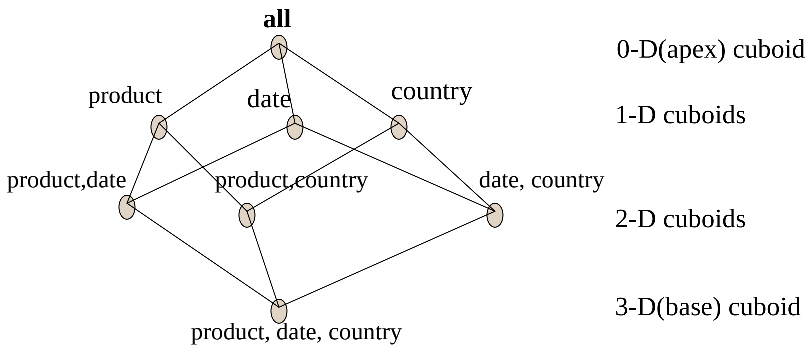
Data Cubes

Indiana Center for Database Systems

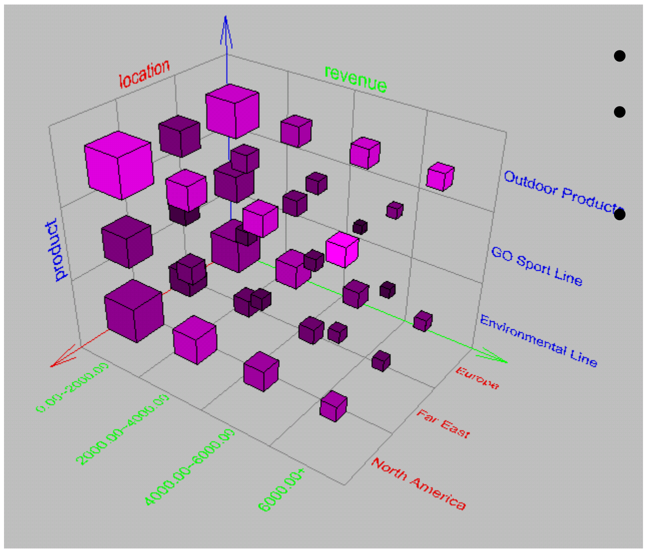
A Sample Data Cube



Cuboids Corresponding to the Cube



Browsing a Data Cube



- Visualization
- OLAP capabilities
- Interactive manipulation

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:
  - *project and select*
- 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)*

Efficient Data Cube Computation

- Data cube can be viewed as a lattice of cuboids
  - The bottom-most cuboid is the base cuboid
  - The top-most cuboid (apex) contains only one cell
  - How many cuboids in an n-dimensional cube with L levels? $T = \prod_{i=1}^{n} (L_i + 1)$
- Materialization of data cube
  - Materialize every (cuboid) (full materialization), none (no materialization), or some (partial materialization)
  - Selection of which cuboids to materialize
    - Based on size, sharing, access frequency, etc.

Cube Operation

- Cube definition and computation in DMQL

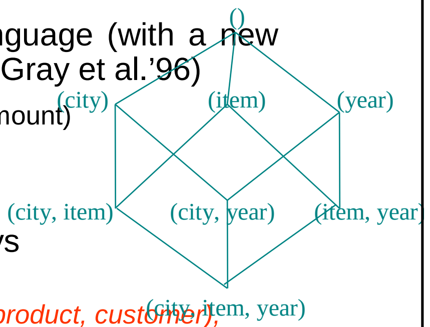
  define cube sales[item, city, year]: sum(sales_in_dollars)

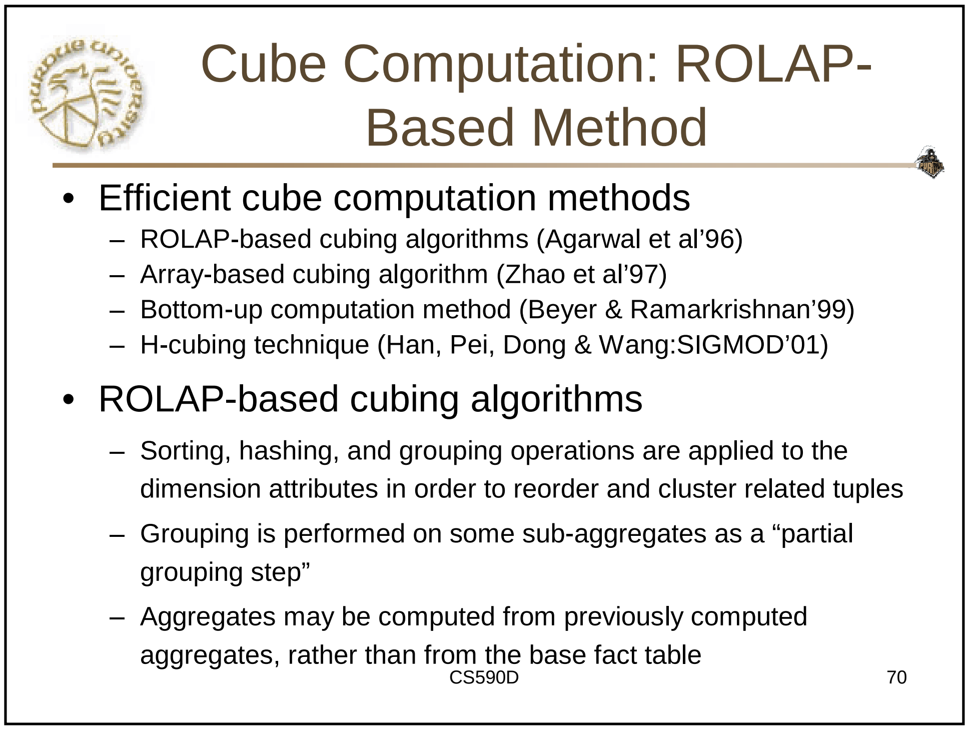
  compute cube sales
- Transform it into a SQL-like language (with a new operator cube by, introduced by Gray et al.'96)

```
SELECT item, city, year, SUM (amount)
FROM SALES
CUBE BY item, city, year
```

- Compute the following Group-Bys

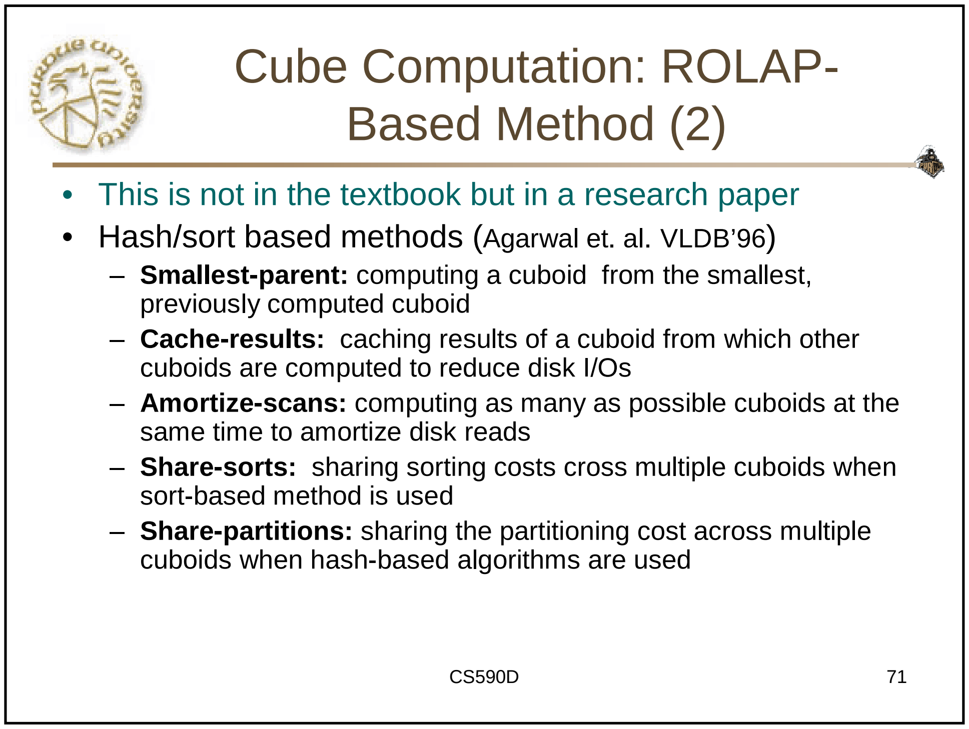
  (date, product, customer),
  (date,product),(date, customer), (product, customer),
  (date), (product), (customer)
  ()





Cube Computation: ROLAP-Based Method

- Efficient cube computation methods
  - ROLAP-based cubing algorithms (Agarwal et al’96)
  - Array-based cubing algorithm (Zhao et al’97)
  - Bottom-up computation method (Beyer & Ramarkrishnan’99)
  - H-cubing technique (Han, Pei, Dong & Wang:SIGMOD’01)
- ROLAP-based cubing algorithms
  - Sorting, hashing, and grouping operations are applied to the dimension attributes in order to reorder and cluster related tuples
  - Grouping is performed on some sub-aggregates as a “partial grouping step”
  - Aggregates may be computed from previously computed aggregates, rather than from the base fact table

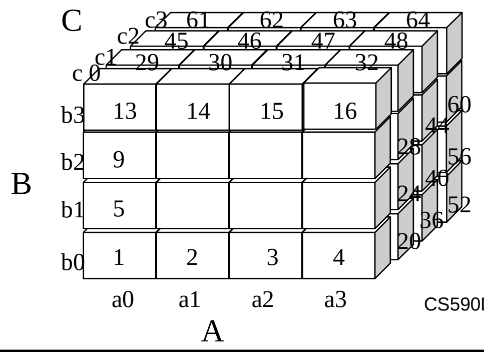


Cube Computation: ROLAP-Based Method (2)

- This is not in the textbook but in a research paper
- Hash/sort based methods (Agarwal et. al. VLDB’96)
  - **Smallest-parent:** 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 for Cube Computation

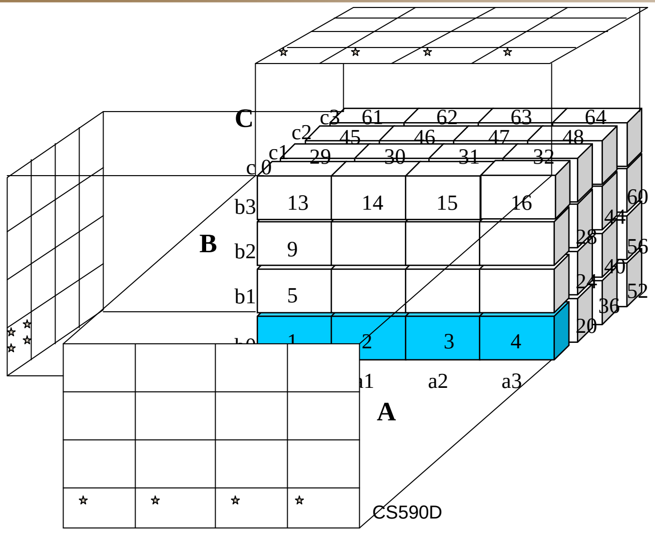
- Partition arrays into chunks (a small subcube which fits in memory).
- Compressed sparse array addressing: (chunk_id, offset)
- Compute aggregates in "multiway" by visiting cube cells in the order which minimizes the # of times to visit each cell, and reduces memory access and storage cost.



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

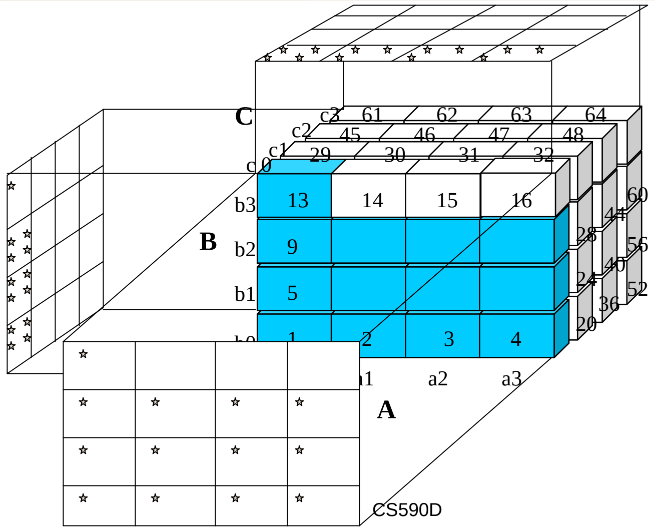
CS590D 72

Multi-way Array Aggregation for Cube Computation



CS590D 73

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.
  - See the details of Example 2.12 (pp. 75-78)
  - 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, “bottom-up computation” and iceberg cube computation methods can be explored

Indexing OLAP Data: Bitmap Index

- Index on a particular column
- Each value in the column has a bit vector: bit-op is fast
- The length of the bit vector: # of records in the base table
- The *i*-th bit is set if the *i*-th row of the base table has the value for the indexed column
- not suitable for high cardinality domains

Base table

Cust	Region	Type
C1	Asia	Retail
C2	Europe	Dealer
C3	Asia	Dealer
C4	America	Retail
C5	Europe	Dealer

Index on Region

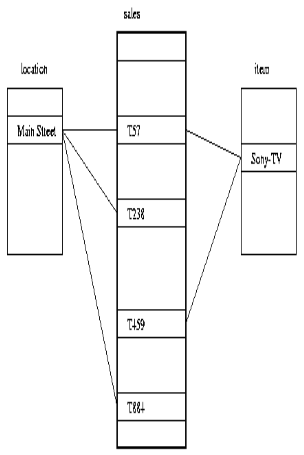
RecID	Asia	Europe	America
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1
5	0	1	0

Index on Type

RecID	Retail	Dealer
1	1	0
2	0	1
3	0	1
4	1	0
5	0	1

Indexing OLAP Data: Join Indices

- Join index: JI(R-id, S-id) where R (R-id, …) ⊳⊲ S (S-id, …)
- Traditional indices map the values to a list of record ids
  - It materializes relational join in JI file and speeds up relational join — a rather costly operation
- In data warehouses, join index relates the values of the dimensions of a start schema to rows in the fact table.
  - E.g. fact table: *Sales* and two dimensions *city* and *product*
    - A join index on *city* maintains for each distinct city a list of R-IDs of the tuples recording the Sales in the city
  - Join indices can span multiple dimensions



Efficient Processing OLAP Queries

- Determine which operations should be performed on the available cuboids:
  - transform drill, roll, etc. into corresponding SQL and/or OLAP operations, e.g, dice = selection + projection
- Determine to which materialized cuboid(s) the relevant operations should be applied.
- Exploring indexing structures and compressed vs. dense array structures in MOLAP

CS590D 78

Iceberg Cube

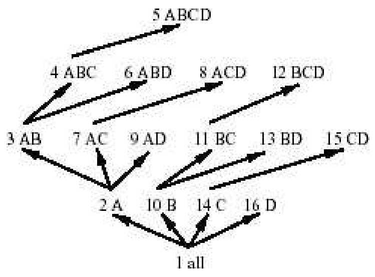
- Computing only the cuboid cells whose count or other aggregates satisfying the condition:

  HAVING COUNT(*) >= *minsup*
- Motivation
  - Only a small portion of cube cells may be “above the water’’ in a sparse cube
  - Only calculate “interesting” data—data above certain threshold
  - Suppose 100 dimensions, only 1 base cell. How many aggregate (non-base) cells if count >= 1? What about count >= 2?

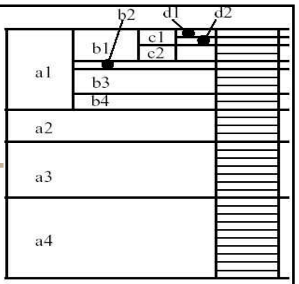
CS590D 79

Bottom-Up Computation (BUC)

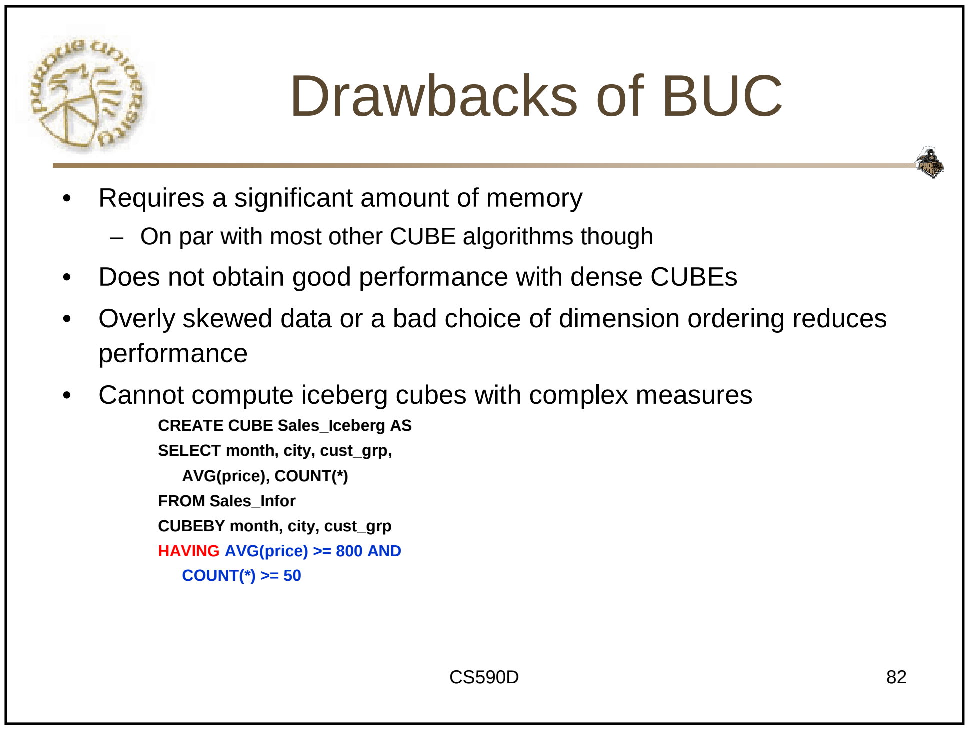
- BUC (Beyer & Ramakrishnan, SIGMOD'99)
- Bottom-up vs. top-down?—depending on how you view it!
- Apriori property:
  - Aggregate the data, then move to the next level
  - If *minsup* is not met, stop!
- If *minsup* = 1 ⇒ compute full CUBE!



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!

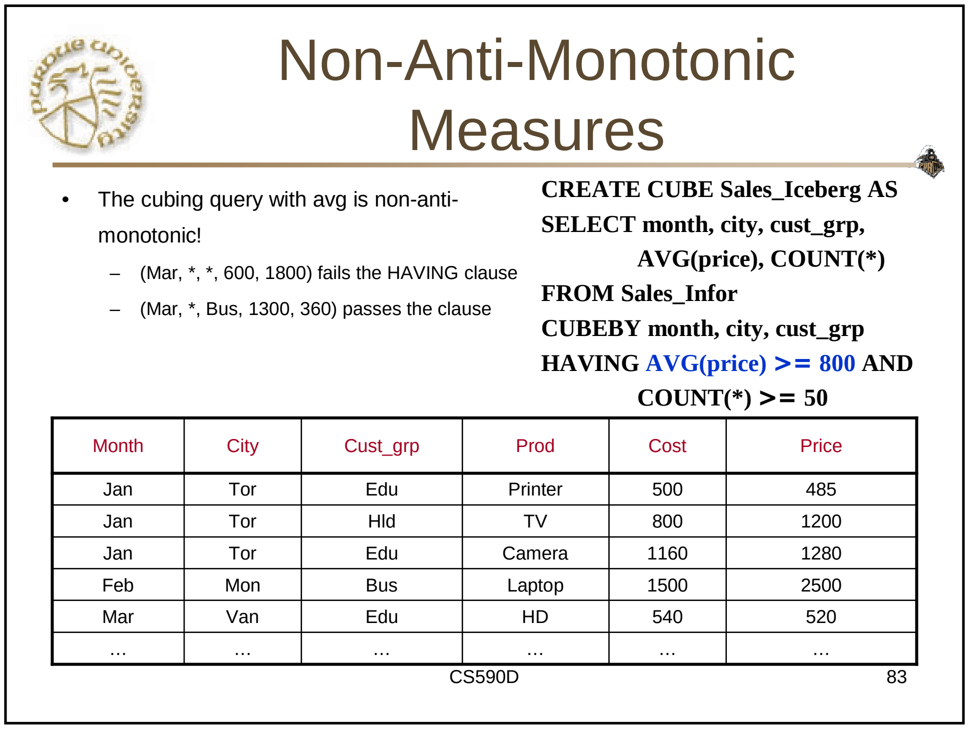


Drawbacks of BUC

- Requires a significant amount of memory
  - On par with most other CUBE algorithms though
- Does not obtain good performance with dense CUBEs
- Overly skewed data or a bad choice of dimension ordering reduces performance
- Cannot compute iceberg cubes with complex measures

```
CREATE CUBE Sales_Iceberg AS
SELECT month, city, cust_grp,
    AVG(price), COUNT(*)
FROM Sales_Infor
CUBEBY month, city, cust_grp
HAVING AVG(price) >= 800 AND
    COUNT(*) >= 50
```

CS590D 82



Non-Anti-Monotonic Measures

- The cubing query with avg is non-anti-monotonic!
  - (Mar, *, *, 600, 1800) fails the HAVING clause
  - (Mar, *, Bus, 1300, 360) passes the clause

```
CREATE CUBE Sales_Iceberg AS
SELECT month, city, cust_grp,
        AVG(price), COUNT(*)
FROM Sales_Infor
CUBEBY month, city, cust_grp
HAVING AVG(price) >= 800 AND
        COUNT(*) >= 50
```

Month	City	Cust_grp	Prod	Cost	Price
Jan	Tor	Edu	Printer	500	485
Jan	Tor	Hld	TV	800	1200
Jan	Tor	Edu	Camera	1160	1280
Feb	Mon	Bus	Laptop	1500	2500
Mar	Van	Edu	HD	540	520
…	…	…	…	…	…

CS590D 83

Top-k Average

- Let (*, Van, *) cover 1,000 records
  - Avg(price) is the average price of those 1000 sales
  - Avg^{50}(price) is the average price of the top-50 sales (top-50 according to the sales price
- Top-k average is anti-monotonic
  - The top 50 sales in Van. is with avg(price) <= 800 → the top 50 deals in Van. during Feb. must be with avg(price) <= 800

Month	City	Cust_grp	Prod	Cost	Price
…	…	…	…	…	…

Binning for Top-k Average

- Computing top-k avg is costly with large k
- Binning idea
  - Avg^{50}(c) >= 800
  - Large value collapsing: use a sum and a count to summarize records with measure >= 800
    - If count>=800, no need to check "small" records
  - Small value binning: a group of bins
    - One bin covers a range, e.g., 600~800, 400~600, etc.
    - Register a sum and a count for each bin

Quant-info for Top-k Average Binning

- Accumulate quant-info for cells to compute average iceberg cubes efficiently
  - Three pieces: sum, count, top-k bins
  - Use top-k bins to estimate/prune descendants
  - Use sum and count to consolidate current cell

$$\text{weakest} \longrightarrow \text{strongest}$$

Approximate avg^{50}()	real avg^{50}()	avg()
Anti-monotonic, can be computed efficiently	Anti-monotonic, but computationally costly	Not anti-monotonic

Discussion: Other Issues

- Computing iceberg cubes with more complex measures?
  - No general answer for holistic measures, e.g., median, mode, rank
  - A research theme even for complex algebraic functions, e.g., standard_dev, variance
- Dynamic vs . static computation of iceberg cubes
  - v and k are only available at query time
  - Setting reasonably low parameters for most nontrivial cases
- Memory-hog? what if the cubing is too big to fit in memory?—projection and then cubing

Condensed Cube

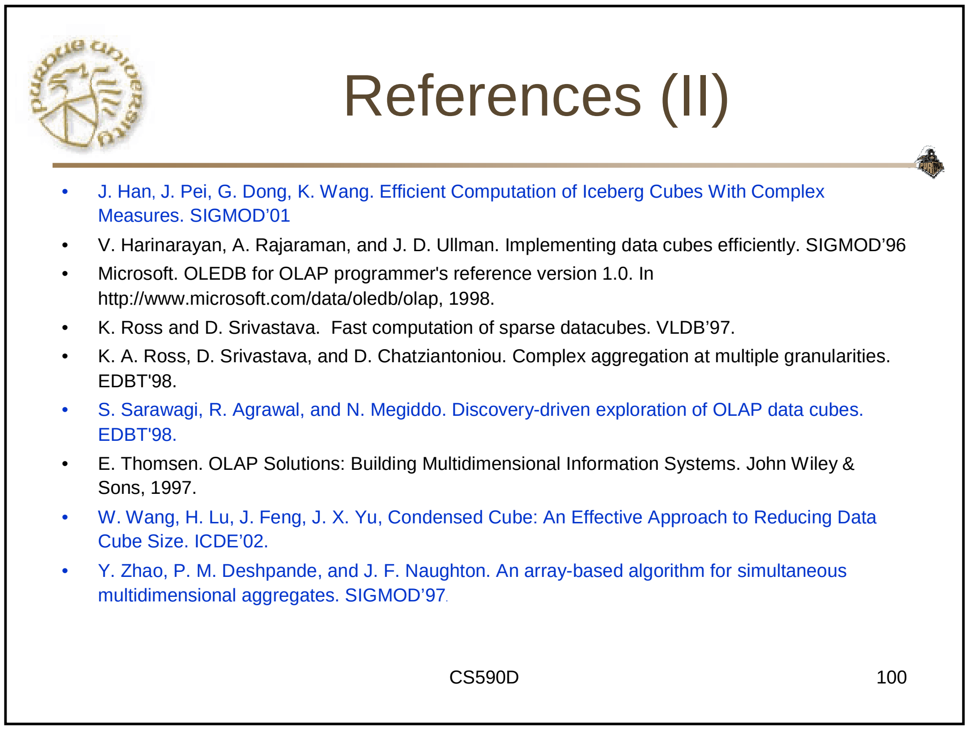
- W. Wang, H. Lu, J. Feng, J. X. Yu, Condensed Cube: An Effective Approach to Reducing Data Cube Size. ICDE'02.
- Iceberg cube cannot solve all the problems
  - Suppose 100 dimensions, only 1 base cell with count = 10. How many aggregate (non-base) cells if count >= 10?
- Condensed cube
  - Only need to store one cell (a_1, a_2, …, a_{100}, 10), which represents all the corresponding aggregate cells
  - Adv.
    - Fully precomputed cube without compression
  - Efficient computation of the minimal condensed cube

CS590D 98

References (I)

- 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.
- R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. ICDE'97
- K. Beyer and R. Ramakrishnan. Bottom-Up Computation of Sparse and Iceberg CUBEs.. SIGMOD'99.
- S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 26:65-74, 1997.
- OLAP council. MDAPI specification version 2.0. In http://www.olapcouncil.org/research/apily.htm, 1998.
- G. Dong, J. Han, J. Lam, J. Pei, K. Wang. Mining Multi-dimensional Constrained Gradients in Data Cubes. VLDB'2001
- 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.

CS590D 99



References (II)

- J. Han, J. Pei, G. Dong, K. Wang. Efficient Computation of Iceberg Cubes With Complex Measures. SIGMOD’01
- V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. SIGMOD’96
- Microsoft. OLEDB for OLAP programmer's reference version 1.0. In http://www.microsoft.com/data/oledb/olap, 1998.
- K. Ross and D. Srivastava. Fast computation of sparse datacubes. VLDB’97.
- 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.
- E. Thomsen. OLAP Solutions: Building Multidimensional Information Systems. John Wiley & Sons, 1997.
- W. Wang, H. Lu, J. Feng, J. X. Yu, Condensed Cube: An Effective Approach to Reducing Data Cube Size. ICDE’02.
- Y. Zhao, P. M. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. SIGMOD’97.