Data Warehousing
資料倉儲

Data Cube Computation and Data Generation

Min-Yuh Day
戴敏育
Assistant Professor
專任助理教授
Dept. of Information Management, Tamkang University
淡江大學 資訊管理學系
http://mail.im.tku.edu.tw/~myday/

2011-04-12

Syllabus

1   100/02/15   Introduction to Data Warehousing

2   100/02/22   Data Warehousing, Data Mining, and Business Intelligence

3   100/03/01   Data Preprocessing: Integration and the ETL process

4   100/03/08   Data Warehouse and OLAP Technology

5   100/03/15   Data Warehouse and OLAP Technology

6   100/03/22   Data Warehouse and OLAP Technology

7   100/03/29   Data Warehouse and OLAP Technology

8   100/04/05   (放假一天) (民族掃墓節)

9   100/04/12   Data Cube Computation and Data Generation

10   100/04/19   Mid-Term Exam (期中考試週)

11   100/04/26   Association Analysis

12   100/05/03   Classification and Prediction, Cluster Analysis

13   100/05/10   Social Network Analysis, Link Mining, Text and Web Mining

14   100/05/17   Project Presentation

15   100/05/24   Final Exam (畢業班考試)

Data Cube Computation and Data Generalization

- Efficient Computation of Data Cubes

- Exploration and Discovery in Multidimensional Databases

- Attribute-Oriented Induction ─ An Alternative Data Generalization Method

Data Mining: Concepts and Techniques

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 (Li, et al. VLDB'04)
- 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
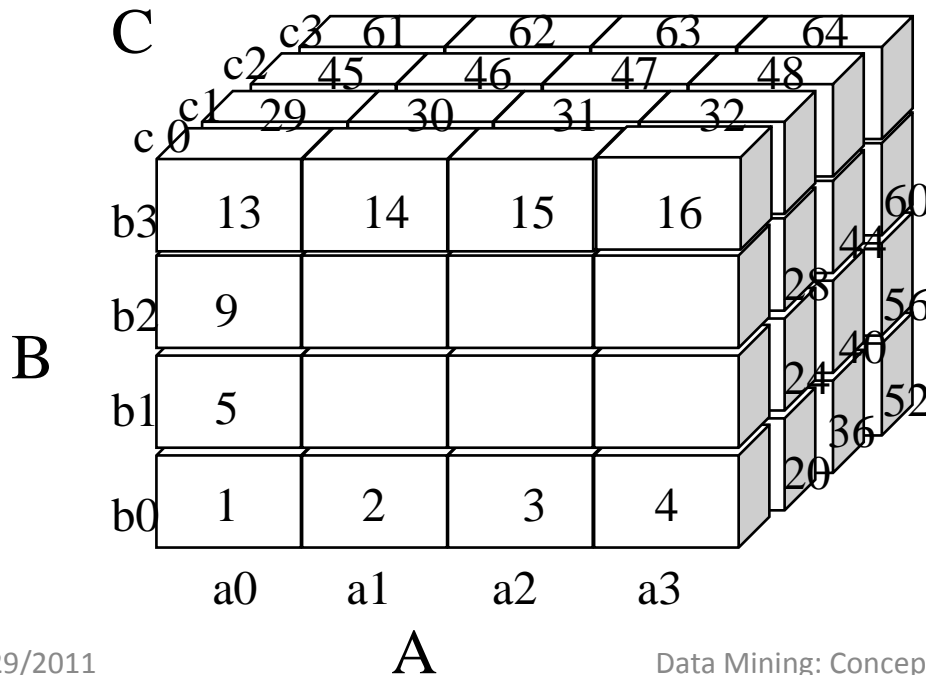
Multi-Way Array Aggregation

- 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

all

A   B   C

AB   AC   BC

ABC
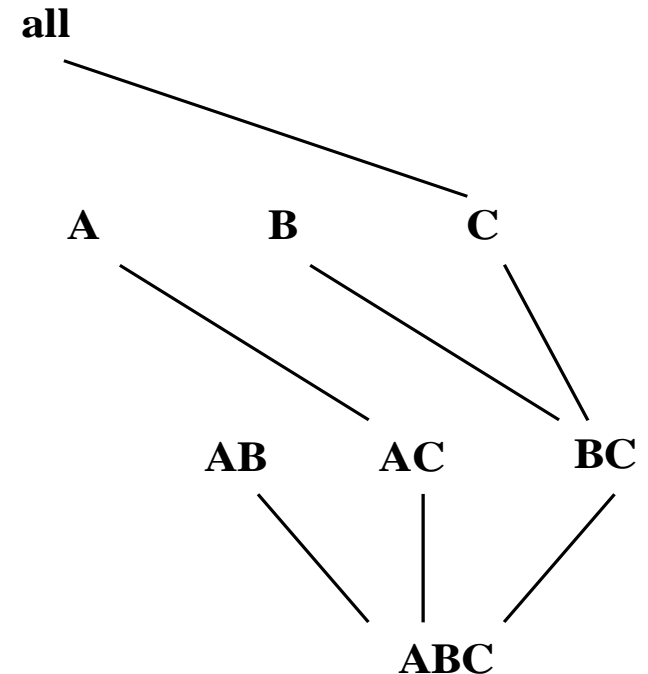
Multi-way Array Aggregation for Cube Computation (MOLAP)

- 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.

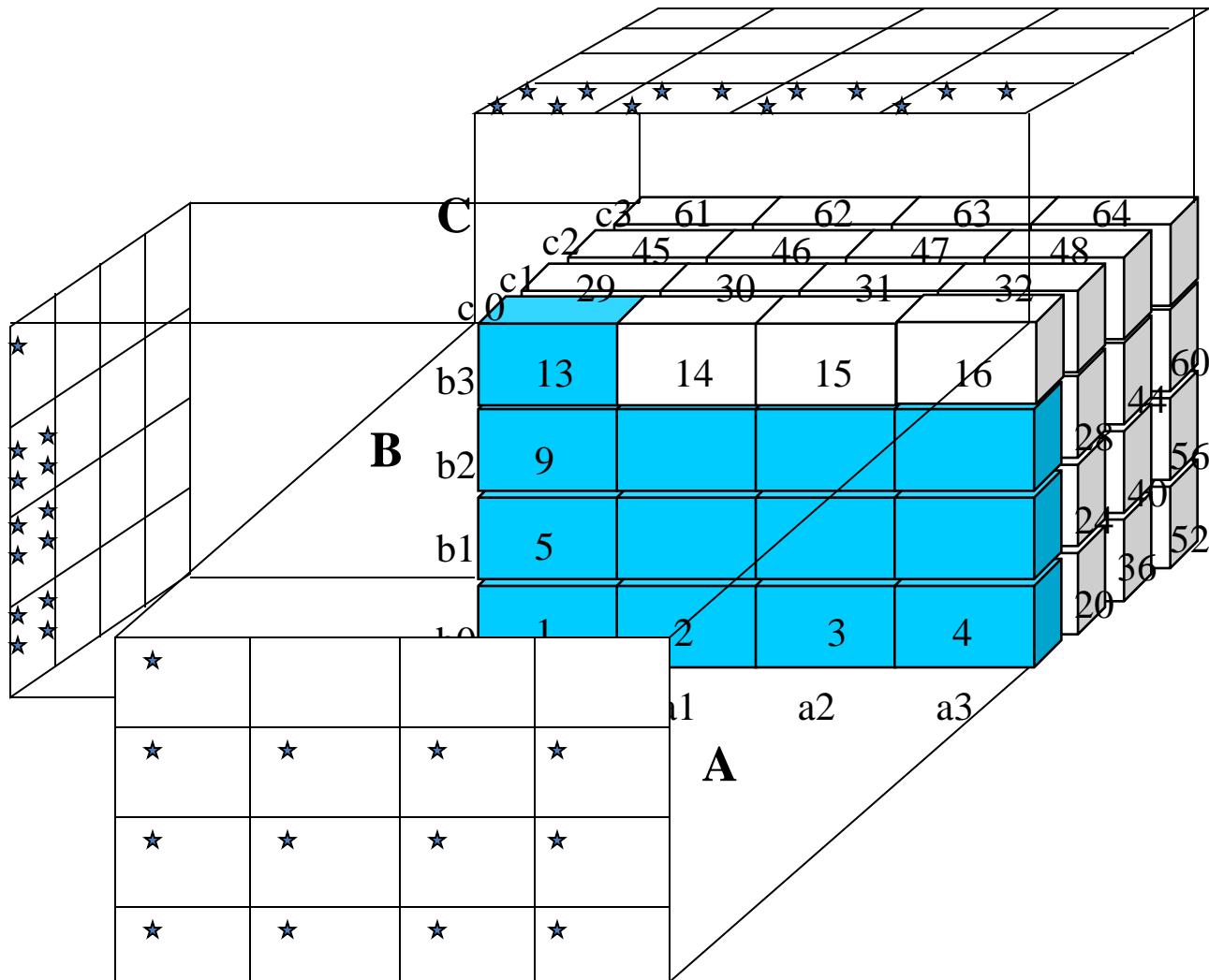
What is the best traversing order to do multi-way aggregation?

Data Mining: Concepts and Techniques

Multi-way Array Aggregation for Cube Computation

Data Mining: Concepts and Techniques
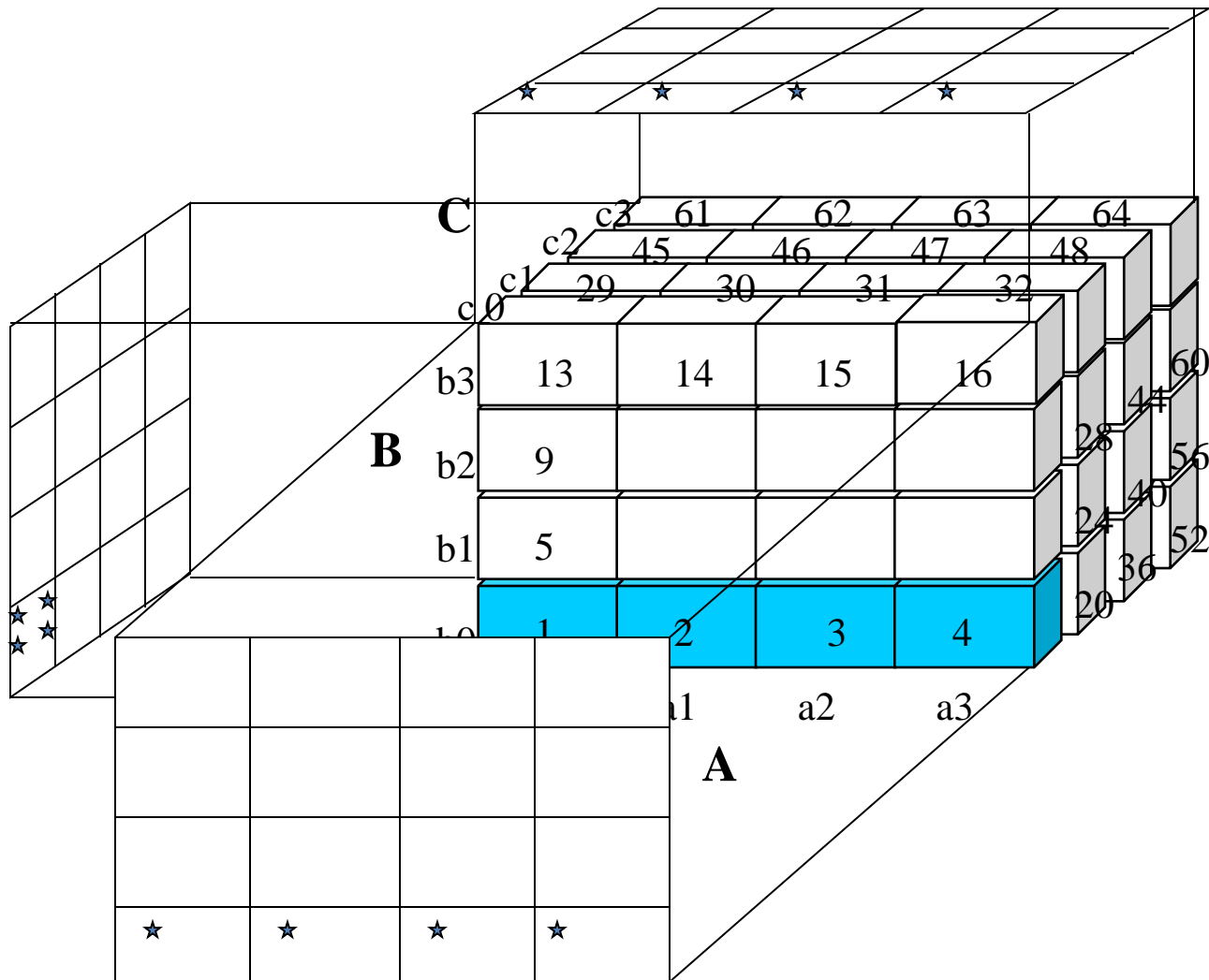
Multi-way Array Aggregation for Cube Computation

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

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 $\Rightarrow$ compute full CUBE!

- No simultaneous aggregation

```
              all
      A     B     C     D
  AB   AC   AD   BC   BD   CD
ABC   ABD   ACD   BCD
           ABCD
```

```
              1 all
      2 A    10 B   14 C   16 D
  3 AB  7 AC  9 AD  11 BC  13 BD  15 CD
  4 ABC   6 ABD   8 ACD   12 BCD
              5 ABCD
```

Data Mining: Concepts and Techniques
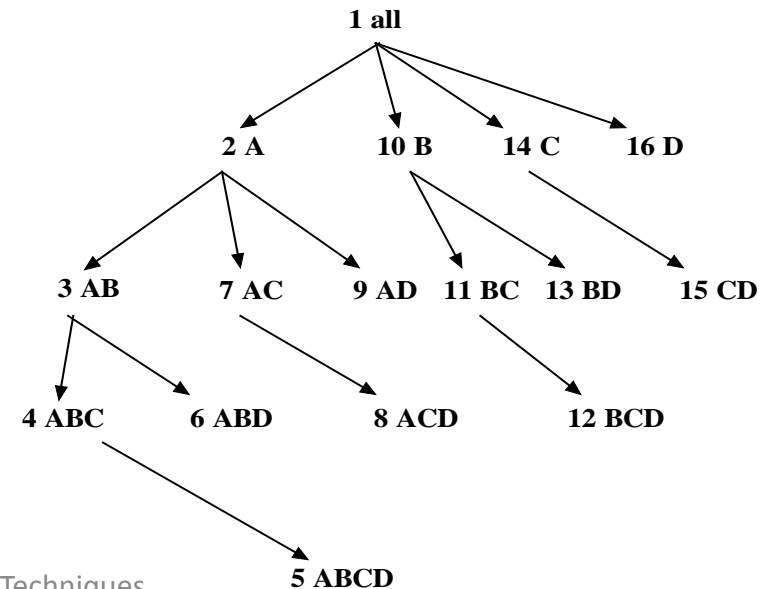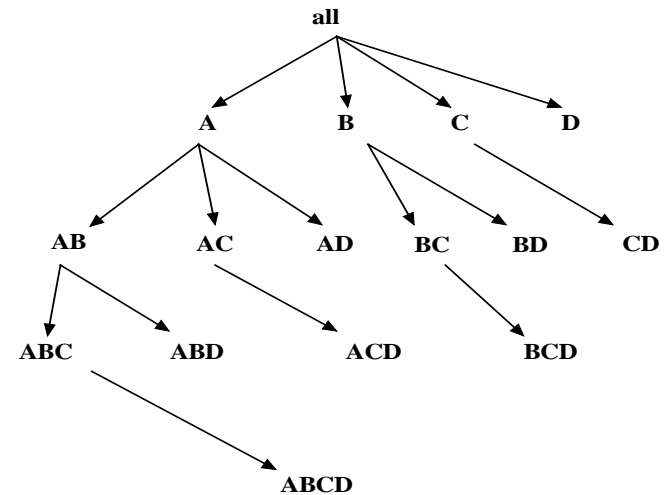
BUC: Partitioning

- 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!
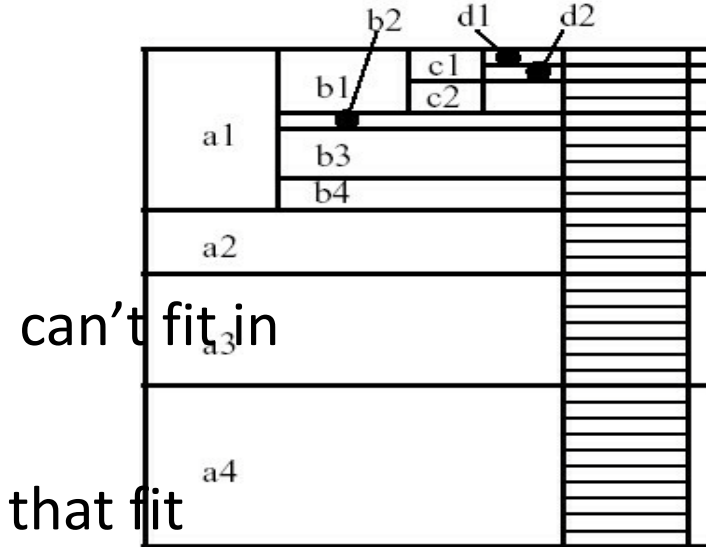
Star-Cubing: An Integrating Method

- Integrate the top-down and bottom-up methods
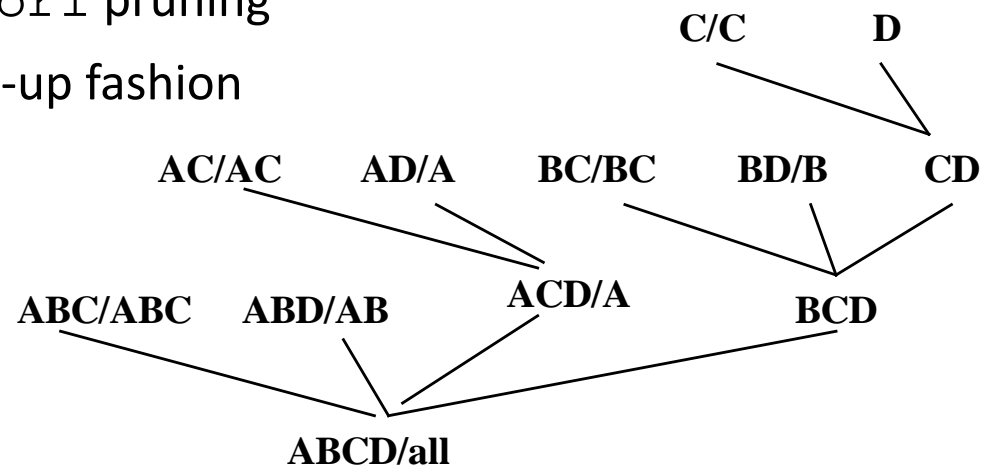
- ***Explore shared dimensions***

    - 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

```
                                              C/C        D

         AC/AC      AD/A      BC/BC      BD/B        CD

                              ACD/A             BCD
ABC/ABC   ABD/AB

             ABCD/all
```

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?

Cell Trees

- Use a tree structure similar to H-tree to represent cuboids

- Collapses common prefixes to save memory

- Keep count at node

- Traverse the tree to retrieve a particular tuple

root: 100

a1: 30   a2: 20   a3: 20   a4: 20

b1: 10   b2: 10   b3: 10

c1: 5   c2: 5

d1: 2   d2: 3
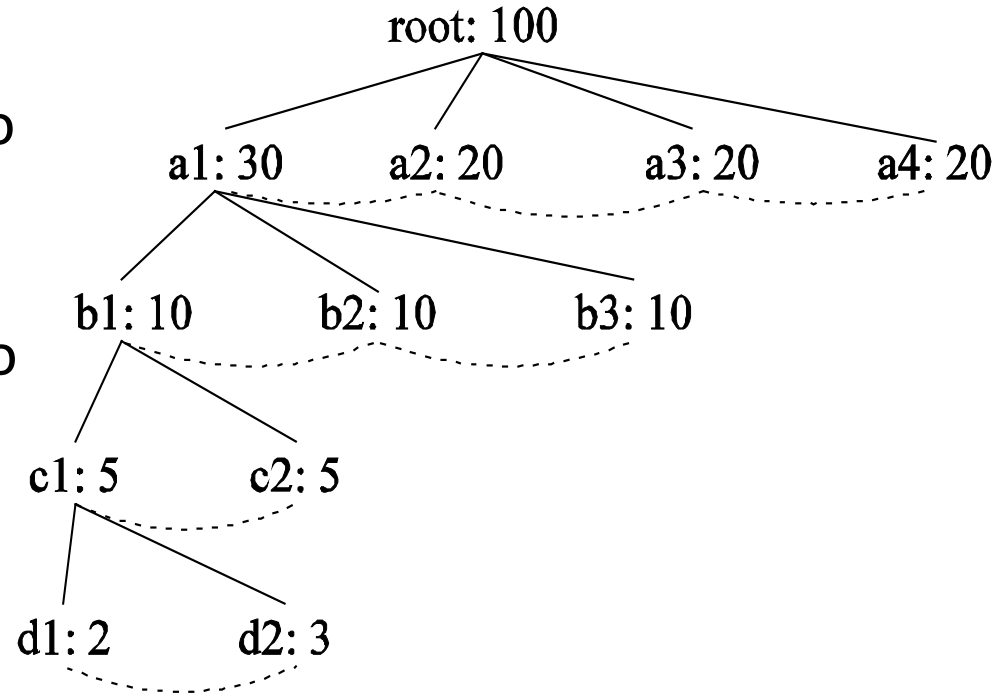
Star Attributes and Star Nodes

- Intuition: If a single-dimensional aggregate on an attribute value p does not satisfy the iceberg condition, it is useless to distinguish them during the iceberg computation
  - E.g., b_2, b_3, b_4, c_1, c_2, c_4, d_1, d_2, d_3

- Solution: Replace such attributes by a *. Such attributes are _star attributes,_ and the corresponding nodes in the cell tree are _star nodes_

A	B	C	D	Count
a1	b1	c1	d1	1
a1	b1	c4	d3	1
a1	b2	c2	d2	1
a2	b3	c3	d4	1
a2	b4	c3	d4	1

Data Mining: Concepts and Techniques

Example: Star Reduction

- Suppose minsup = 2

- Perform one-dimensional aggregation. Replace attribute values whose count < 2 with *. And collapse all *'s together

- Resulting table has all such attributes replaced with the star-attribute

- With regards to the iceberg computation, this new table is a *loseless compression* of the original table

A	B	C	D	Count
a1	b1	*	*	1
a1	b1	*	*	1
a1	*	*	*	1
a2	*	c3	d4	1
a2	*	c3	d4	1

↓

A	B	C	D	Count
a1	b1	*	*	2
a1	*	*	*	1
a2	*	c3	d4	2

Data Mining: Concepts and Techniques

- Efficient Computation of Data Cubes

- Exploration and Discovery in Multidimensional Databases

- Attribute-Oriented Induction — An Alternative Data Generalization Method

Computing Cubes with Non-Antimonotonic Iceberg Conditions

- Most cubing algorithms cannot compute cubes with non-antimonotonic iceberg conditions efficiently

- Example

    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

- Needs to study how to push constraint into the cubing process

Non-Anti-Monotonic Iceberg Condition

- Anti-monotonic: if a process fails a condition, continue processing will still fail

- The cubing query with avg is non-anti-monotonic!

  - (Mar, *, *, 600, 1800) fails the HAVING clause

  - (Mar, *, Bus, 1300, 360) passes the clause

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
...	...	...	...	...	...

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

From Average to 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

Computing Approximate top-k average

Suppose for (*, Van, *), we have

Range	Sum	Count
Over 800	28000	20
600~800	10600	15
400~600	15200	30
…	…	…

Top 50

Approximate $\text{avg}^{50}()=$
$(28000+10600+600*15)/50=952$

The cell may pass the HAVING clause

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

Weakened Conditions Facilitate Pushing

- 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

weakest ⟶ **strongest**

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

Computing Iceberg Cubes with Other Complex Measures

- Computing other complex measures

  - Key point: find a function which is weaker but ensures certain anti-monotonicity

- Examples

  - Avg() $\leq$ v: $avg_k(c) \leq$ v (bottom-k avg)

  - Avg() $\geq$ v only (no count): max(price) $\geq$ v

  - Sum(profit) (profit can be negative):

    - p_sum(c) $\geq$ v if p_count(c) $\geq$ k; or otherwise, $sum^k(c) \geq$ v

  - Others: conjunctions of multiple conditions

Compressed Cubes: Condensed or Closed Cubes

- W. Wang, H. Lu, J. Feng, J. X. Yu, Condensed Cube: An Effective Approach to Reducing Data Cube Size, ICDE'02.

- Icerberg 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

- Closed cube
  - Dong Xin, Jiawei Han, Zheng Shao, and Hongyan Liu, "C-Cubing: Efficient Computation of Closed Cubes by Aggregation-Based Checking", ICDE'06.

- Efficient Computation of Data Cubes

- Exploration and Discovery in Multidimensional Databases

- Attribute-Oriented Induction — An Alternative Data Generalization Method

Discovery-Driven Exploration of Data Cubes

- Hypothesis-driven
  - exploration by user, huge search space
- Discovery-driven (Sarawagi, et al.'98)
  - Effective navigation of large OLAP data cubes
  - pre-compute measures indicating exceptions, guide user in the data analysis, at all levels of aggregation
  - Exception: significantly different from the value anticipated, based on a statistical model
  - Visual cues such as background color are used to reflect the degree of exception of each cell

Kinds of Exceptions and their Computation

- Parameters
    - SelfExp: surprise of cell relative to other cells at same level of aggregation
    - InExp: surprise beneath the cell
    - PathExp: surprise beneath cell for each drill-down path
- Computation of exception indicator (modeling fitting and computing SelfExp, InExp, and PathExp values) can be overlapped with cube construction
- Exception themselves can be stored, indexed and retrieved like precomputed aggregates

Examples: Discovery-Driven Data Cubes

item	all
region	all

Sum of sales	month											
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Total		1%	-1%	0%	1%	3%	-1	-9%	-1%	2%	-4%	3%

Avg sales	month											
item	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Sony b/w printer		9%	-8%	2%	-5%	14%	-4%	0%	41%	-13%	-15%	-11%
Sony color printer		0%	0%	3%	2%	4%	-10%	-13%	0%	4%	-6%	4%
HP b/w printer		-2%	1%	2%	3%	8%	0%	-12%	-9%	3%	-3%	6%
HP color printer		0%	0%	-2%	1%	0%	-1%	-7%	-2%	1%	-5%	1%
IBM home computer		1%	-2%	-1%	-1%	3%	3%	-10%	4%	1%	-4%	-1%
IBM laptop computer		0%	0%	-1%	3%	4%	2%	-10%	-2%	0%	-9%	3%
Toshiba home computer		-2%	-5%	1%	1%	-1%	1%	5%	-3%	-5%	-1%	-1%
Toshiba laptop computer		1%	0%	3%	0%	-2%	-2%	-5%	3%	2%	-1%	0%
Logitech mouse		3%	-2%	-1%	0%	4%	6%	-11%	2%	1%	-4%	0%
Ergo-way mouse		0%	0%	2%	3%	1%	-2%	-2%	-5%	0%	-5%	8%

item	IBM home computer

Avg sales	month											
region	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
North		-1%	-3%	-1%	0%	3%	4%	-7%	1%	0%	-3%	-3%
South		-1%	1%	-9%	6%	-1%	-39%	9%	-34%	4%	1%	7%
East		-1%	-2%	2%	-3%	1%	18%	-2%	11%	-3%	-2%	-1%
West		4%	0%	-1%	-3%	5%	1%	-18%	8%	5%	-8%	1%

Complex Aggregation at Multiple Granularities: Multi-Feature Cubes

- Multi-feature cubes (Ross, et al. 1998): Compute complex queries involving multiple dependent aggregates at multiple granularities

- Ex. Grouping by all subsets of {item, region, month}, find the maximum price in 1997 for each group, and the total sales among all maximum price tuples

  select item, region, month, max(price), sum(R.sales)

  from purchases

  where year = 1997

  cube by item, region, month: R

  such that R.price = max(price)

- Continuing the last example, among the max price tuples, find the min and max shelf live, and find the fraction of the total sales due to tuple that have min shelf life within the set of all max price tuples

Cube-Gradient (Cubegrade)

- Analysis of changes of sophisticated measures in multi-dimensional spaces

  – Query: changes of average house price in Vancouver in '00 comparing against '99

  – Answer: Apts in West went down 20%, houses in Metrotown went up 10%

- Cubegrade problem by Imielinski et al.

  – Changes in dimensions → changes in measures

  – Drill-down, roll-up, and mutation

From Cubegrade to Multi-dimensional Constrained Gradients in Data Cubes

- Significantly more expressive than association rules

  - Capture trends in user-specified measures

- Serious challenges

  - Many trivial cells in a cube → "significance constraint" to prune trivial cells

  - Numerate pairs of cells → "probe constraint" to select a subset of cells to examine

  - Only interesting changes wanted → "gradient constraint" to capture significant changes

Data Mining: Concepts and Techniques

MD Constrained Gradient Mining

- Significance constraint C_{sig}: (cnt$\geq$100)

- Probe constraint $C_{prb:}$ (city="Van", cust_grp="busi", prod_grp="*")

- Gradient constraint $C_{grad}(c_g, c_p)$: (avg_price(c_g)/avg_price(c_p)$\geq$1.3)

Probe cell: satisfied C_{prb} (c4, c2) satisfies C_{grad}!

Base cell

Aggregated cell

Siblings

Ancestor

Dimensions					Measures	
cid	Yr	City	Cst_grp	Prd_grp	Cnt	Avg_price
c1	00	Van	Busi	PC	300	2100
c2	*	Van	Busi	PC	2800	1800
c3	*	Tor	Busi	PC	7900	2350
c4	*	*	busi	PC	58600	2250

Data Mining: Concepts and Techniques

Efficient Computing Cube-gradients

- Compute probe cells using C_{sig} and C_{prb}
  - The set of probe cells P is often very small

- Use probe P and constraints to find gradients
  - Pushing selection deeply
  - Set-oriented processing for probe cells
  - Iceberg growing from low to high dimensionalities
  - Dynamic pruning probe cells during growth
 - Incorporating efficient iceberg cubing method

- Efficient Computation of Data Cubes

- Exploration and Discovery in Multidimensional Databases

- Attribute-Oriented Induction ─ An Alternative Data Generalization Method

What is Concept Description?

- Descriptive vs. predictive data mining
  - Descriptive mining: describes concepts or task-relevant data sets in concise, summarative, informative, discriminative forms
  - Predictive mining: Based on data and analysis, constructs models for the database, and predicts the trend and properties of unknown data
- Concept description:
  - Characterization: provides a concise and succinct summarization of the given collection of data
  - Comparison: provides descriptions comparing two or more collections of data

Data Generalization and Summarization-based Characterization

- Data generalization
  - A process which abstracts a large set of task-relevant data in a database from a low conceptual levels to higher ones.
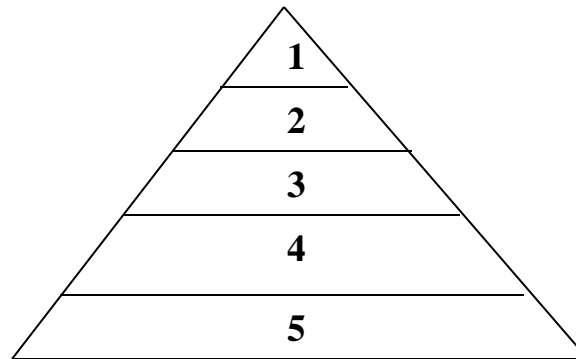
  - Approaches:

Conceptual levels

    - Data cube approach(OLAP approach)
    - Attribute-oriented induction approach

Concept Description vs. OLAP

- Similarity:

  - Data generalization

  - Presentation of data summarization at multiple levels of abstraction.

  - Interactive drilling, pivoting, slicing and dicing.

- Differences:

  - Can handle complex data types of the attributes and their aggregations

  - Automated desired level allocation.

  - Dimension relevance analysis and ranking when there are many relevant dimensions.

  - Sophisticated typing on dimensions and measures.

  - Analytical characterization: data dispersion analysis

Attribute-Oriented Induction

- Proposed in 1989 (KDD '89 workshop)

- Not confined to categorical data nor particular measures

- How it is done?

  - Collect the task-relevant data (*initial relation*) using a relational database query

  - Perform generalization by <u>attribute removal</u> or <u>attribute generalization</u>

  - Apply aggregation by merging identical, generalized tuples and accumulating their respective counts

  - Interactive presentation with users

Basic Principles of Attribute-Oriented Induction

- Data focusing: task-relevant data, including dimensions, and the result is the *initial relation*

- Attribute-removal: remove attribute *A* if there is a large set of distinct values for *A* but (1) there is no generalization operator on *A*, or (2) *A*'s higher level concepts are expressed in terms of other attributes

- Attribute-generalization: If there is a large set of distinct values for *A*, and there exists a set of generalization operators on *A*, then select an operator and generalize *A*

- Attribute-threshold control: typical 2-8, specified/default

- Generalized relation threshold control: control the final relation/rule size

Attribute-Oriented Induction: Basic Algorithm

- InitialRel: Query processing of task-relevant data, deriving the *initial relation*.

- PreGen: Based on the analysis of the number of distinct values in each attribute, determine generalization plan for each attribute: removal? or how high to generalize?

- PrimeGen: Based on the PreGen plan, perform generalization to the right level to derive a "prime generalized relation", accumulating the counts.

- Presentation: User interaction: (1) adjust levels by drilling, (2) pivoting, (3) mapping into rules, cross tabs, visualization presentations.

Example

- DMQL: Describe general characteristics of graduate students in the Big-University database

    use Big_University_DB

    mine characteristics as "Science_Students"

    in relevance to name, gender, major, birth_place, birth_date, residence, phone#, gpa

    from student

    where status in "graduate"

- Corresponding SQL statement:

    Select name, gender, major, birth_place, birth_date, residence, phone#, gpa

    from student

    where status in {"Msc", "MBA", "PhD" }

Class Characterization: An Example

Initial Relation

Name	Gender	Major	Birth-Place	Birth_date	Residence	Phone #	GPA
Jim Woodman	M	CS	Vancouver,BC, Canada	8-12-76	3511 Main St., Richmond	687-4598	3.67
Scott Lachance	M	CS	Montreal, Que, Canada	28-7-75	345 1st Ave., Richmond	253-9106	3.70
Laura Lee	F	Physics	Seattle, WA, USA	25-8-70	125 Austin Ave., Burnaby	420-5232	3.83
…	…	…	…	…	…	…	…
Removed	**Retained**	**Sci,Eng, Bus**	**Country**	**Age range**	**City**	**Removed**	**Excl, VG,..**

Prime Generalized Relation

Gender	Major	Birth_region	Age_range	Residence	GPA	Count
M	Science	Canada	20-25	Richmond	Very-good	16
F	Science	Foreign	25-30	Burnaby	Excellent	22
…	…	…	…	…	…	…

Birth_Region / Gender	Canada	Foreign	Total
M	16	14	30
F	10	22	32
Total	26	36	62

Presentation of Generalized Results

- Generalized relation:

  - Relations where some or all attributes are generalized, with counts or other aggregation values accumulated.

- Cross tabulation:

  - Mapping results into cross tabulation form (similar to contingency tables).

  - Visualization techniques:

  - Pie charts, bar charts, curves, cubes, and other visual forms.

- Quantitative characteristic rules:

  - Mapping generalized result into characteristic rules with quantitative information associated with it, e.g.,

$$grad(x) \wedge male(x) \Rightarrow$$
$$birth\_region(x) = "Canada"[t:53\%] \vee birth\_region(x) = "foreign"[t:47\%].$$

Mining Class Comparisons

- <u>Comparison:</u> Comparing two or more classes

- <u>Method:</u>
    - Partition the set of relevant data into the target class and the contrasting class(es)
    - Generalize both classes to the same high level concepts
    - Compare tuples with the same high level descriptions
    - Present for every tuple its description and two measures
        - support - distribution within single class
        - comparison - distribution between classes
    - Highlight the tuples with strong discriminant features

- <u>Relevance Analysis:</u>
    - Find attributes (features) which best distinguish different classes

Quantitative Discriminant Rules

- Cj = target class
- q_a = a generalized tuple covers some tuples of class
  - but can also cover some tuples of contrasting class
- d-weight
  - range: [0, 1]

$$d-weight = \frac{count(q_a \in C_j\ )}{\sum_{i=1}^{m} count(q_a \in C_i\ )}$$

- quantitative discriminant rule form

$$\forall\ X,\ target\_class(X) \Leftarrow condition(X)\ \ [d:d\_weight]$$

Example: Quantitative Discriminant Rule

Status	Birth_country	Age_range	Gpa	Count
Graduate	Canada	25-30	Good	90
Undergraduate	Canada	25-30	Good	210

Count distribution between graduate and undergraduate students for a generalized tuple

- Quantitative discriminant rule

$$\forall X,\ graduate\_student(X) \Leftarrow$$

$$birth\_country(X) = "Canada" \wedge age\_range(X) = "25-30" \wedge gpa(X) = "good"\ [d:30\%]$$

  - where 90/(90 + 210) = 30%

Class Description

- Quantitative characteristic rule

  $$\forall X, \ target\_class(X) \Rightarrow condition(X) \ [t : t\_weight]$$
  - necessary

- Quantitative discriminant rule

  $$\forall X, \ target\_class(X) \Leftarrow condition(X) \ [d : d\_weight]$$
  - sufficient

- Quantitative description rule

  $$\forall X, \ target\_class(X) \Leftrightarrow$$

  $$condition_1(X)[t : w_1, d : w'_1] \vee ... \vee condition_n(X)[t : w_n, d : w'_n]$$
  - necessary and sufficient

Example: Quantitative Description Rule

Location/item	TV			Computer			Both_items		
	Count	*t-wt*	*d-wt*	*Count*	*t-wt*	*d-wt*	*Count*	*t-wt*	*d-wt*
Europe	80	25%	40%	240	75%	30%	320	100%	32%
N_Am	120	17.65%	60%	560	82.35%	70%	680	100%	68%
Both_ regions	200	20%	100%	800	80%	100%	1000	100%	100%

Crosstab showing associated t-weight, d-weight values and total number (in thousands) of TVs and computers sold at AllElectronics in 1998

- Quantitative description rule for target class *Europe*

$$\forall X, Europe(X) \Leftrightarrow$$

$$(item(X) = "TV")[t:25\%, d:40\%] \lor (item(X) = "computer")[t:75\%, d:30\%]$$

Summary

- Efficient algorithms for computing data cubes
- Further development of data cube technology
  - Discovery-drive cube
  - Multi-feature cubes
  - Cube-gradient analysis
- Anther generalization approach: Attribute-Oriented Induction

References

- Jiawei Han and Micheline Kamber, Data Mining: Concepts and Techniques, Second Edition, 2006, Elsevier