Data Warehousing
資料倉儲

Data Cube Computation and Data Generation

1001DW05
MI4
Tue. 6,7 (13:10-15:00) B427

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/

2011-10-11

Syllabus

週次	日期	內容（Subject/Topics）
1	100/09/06	Introduction to Data Warehousing
2	100/09/13	Data Warehousing, Data Mining, and Business Intelligence
3	100/09/20	Data Preprocessing: Integration and the ETL process
4	100/09/27	Data Warehouse and OLAP Technology
5	100/10/04	Data Warehouse and OLAP Technology
6	100/10/11	Data Cube Computation and Data Generation
7	100/10/18	Data Cube Computation and Data Generation
8	100/10/25	Project Proposal
9	100/11/01	期中考試週

Syllabus

週次	日期	內容（Subject/Topics）
10	100/11/08	Association Analysis
11	100/11/15	Classification and Prediction
12	100/11/22	Cluster Analysis
13	100/11/29	Sequence Data Mining
14	100/12/06	Social Network Analysis
15	100/12/13	Link Mining
16	100/12/20	Text Mining and Web Mining
17	100/12/27	Project Presentation
18	101/01/03	期末考試週

Data Warehouse Development

- Data warehouse development approaches
  - Inmon Model: EDW approach (top-down)
  - Kimball Model: Data mart approach (bottom-up)
  - Which model is best?
    - There is no one-size-fits-all strategy to DW
  - One alternative is the hosted warehouse

- Data warehouse structure:
  - The Star Schema vs. Relational

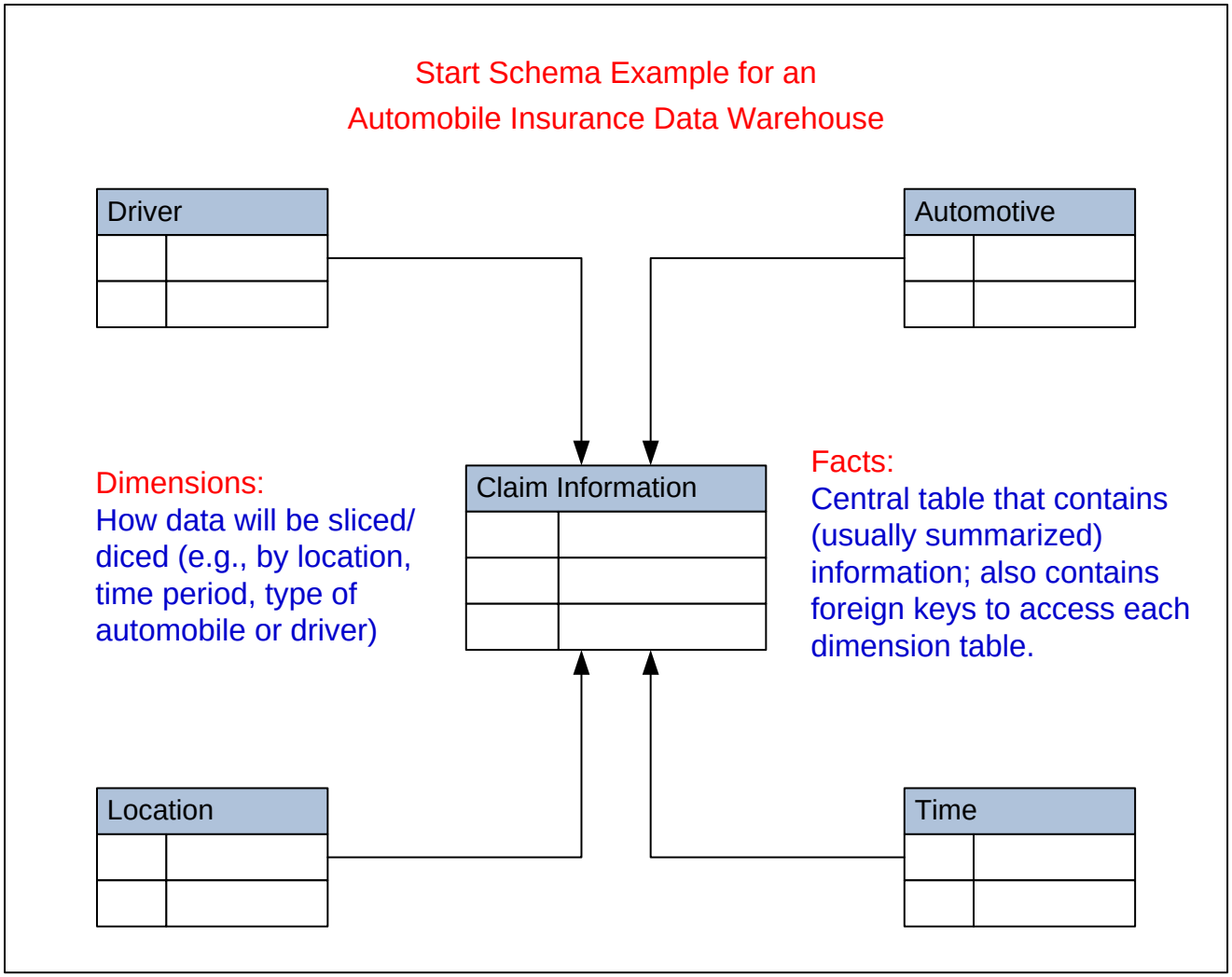
- Real-time data warehousing?

DW Development Approaches

Effort	Data Mart Approach (Kimball Approach)	EDW Approach (Inmon Approach)
Scope	One subject area	Several subject areas
Development time	Months	Years
Development cost	$10,000 to $100,000+	$1,000,000+
Development difficulty	Low to medium	High
Data prerequisite for sharing	Common (within business area)	Common (across enterprise)
Sources	Only some operational and external systems	Many operational and external systems
Size	Megabytes to several gigabytes	Gigabytes to petabytes
Time horizon	Near-current and historical data	Historical data
Data transformations	Low to medium	High
Update frequency	Hourly, daily, weekly	Weekly, monthly
Technology		
Hardware	Workstations and departmental servers	Enterprise servers and mainframe computers
Operating system	Windows and Linux	Unix, Z/OS, OS/390
Databases	Workgroup or standard database servers	Enterprise database servers

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

DW Structure: Star Schema (a.k.a. Dimensional Modeling)



Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

Dimensional Modeling

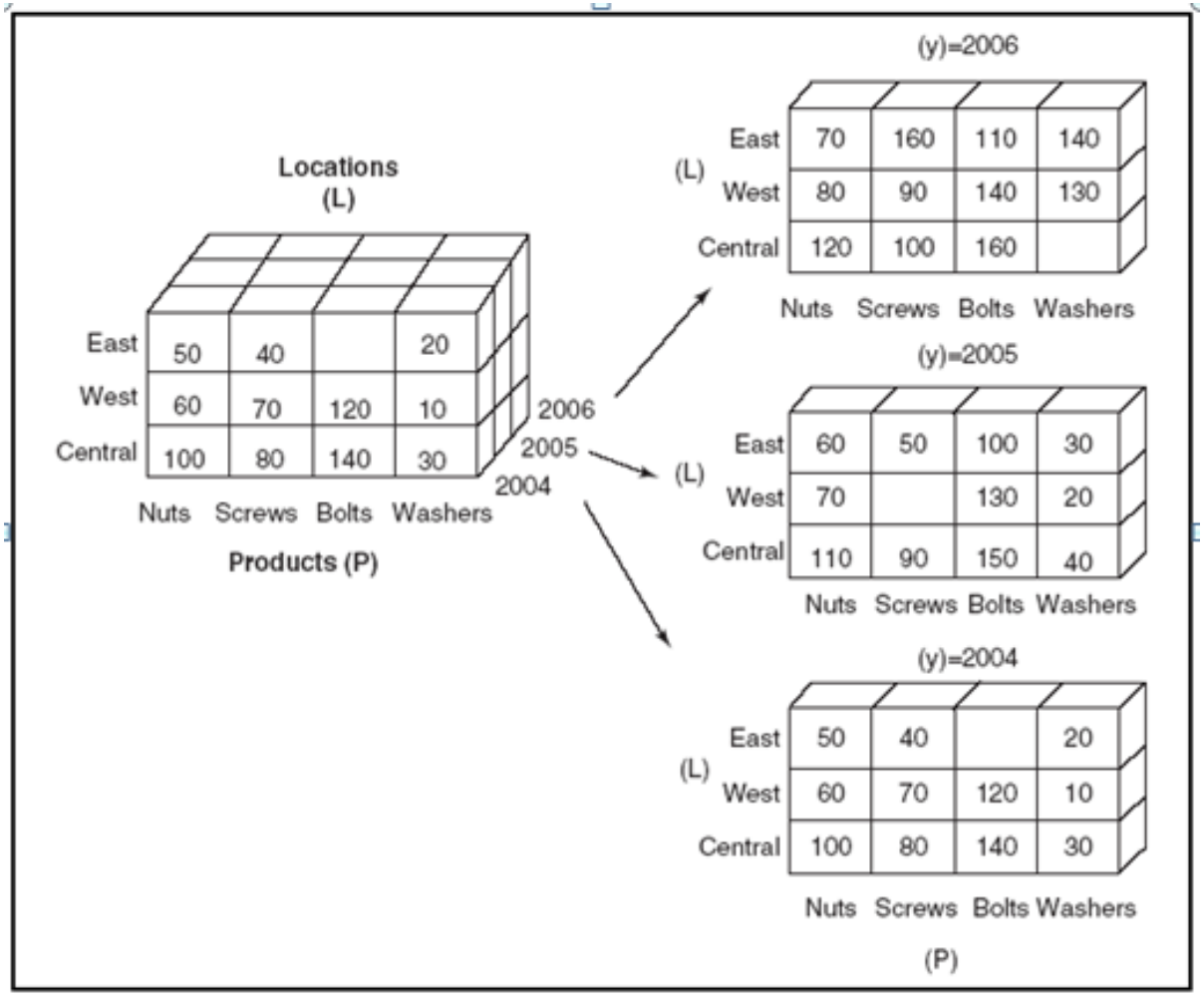
Data cube

A two-dimensional, three-dimensional, or higher-dimensional object in which each dimension of the data represents a measure of interest

-Grain

-Drill-down

-Slicing



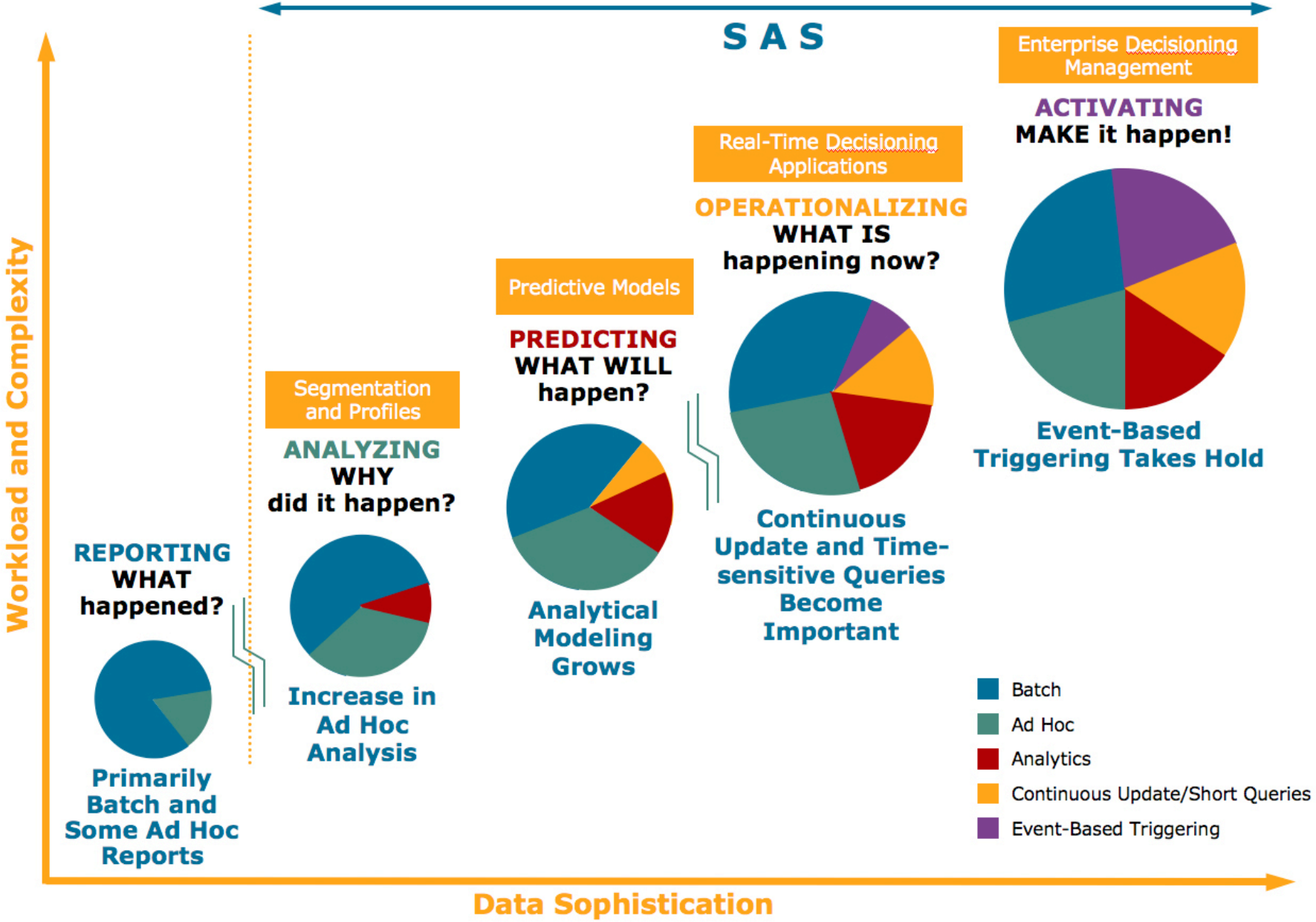
Best Practices for Implementing DW

- The project must fit with corporate strategy
- There must be complete buy-in to the project
- It is important to manage user expectations
- The data warehouse must be built incrementally
- Adaptability must be built in from the start
- The project must be managed by both IT and business professionals (a business–supplier relationship must be developed)
- Only load data that have been cleansed/high quality
- Do not overlook training requirements
- Be politically aware.

Real-time DW (a.k.a. Active Data Warehousing)

- Enabling real-time data updates for real-time analysis and real-time decision making is growing rapidly
  - Push vs. Pull (of data)
- Concerns about real-time BI
  - Not all data should be updated continuously
  - Mismatch of reports generated minutes apart
  - May be cost prohibitive
  - May also be infeasible

Evolution of DSS & DW



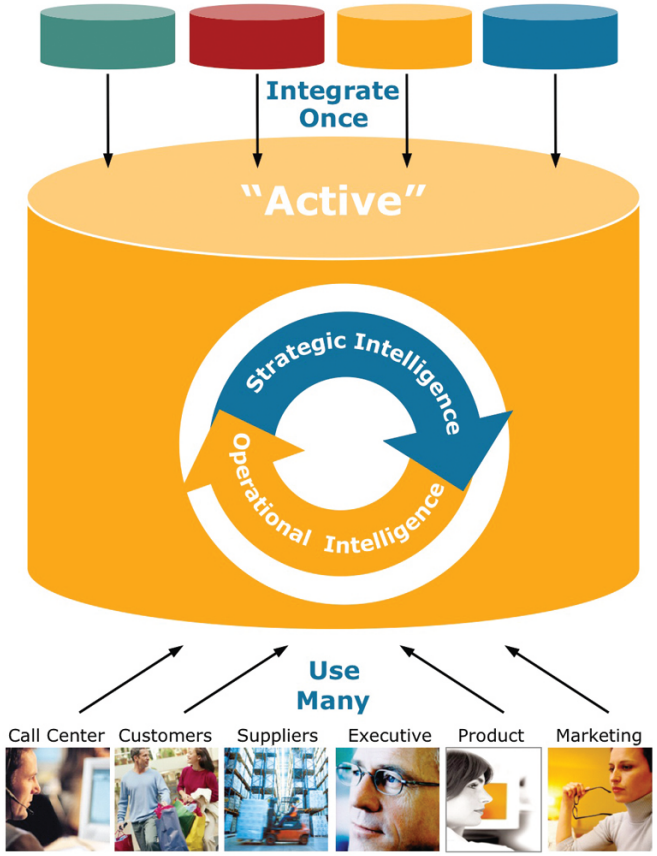
Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

Active Data Warehousing (by Teradata Corporation)

Active Access
Front-Line operational decisions or services supported by near-real-time (NRT) access; Service Level Agreements of 5 seconds or less

Active Load
Intra-day data acquisition; Mini-batch to NRT trickle data feeds measured in minutes or seconds

Active Events
Proactive monitoring of business activity initiating intelligent actions based on rules and context; to sys-tems or users supporting an operational business process



Active Workload Management
Dynamically manage system resources for optimum performance and resource utilization supporting a mixed-workload environment

Active Enterprise Integration
Integration into the Enterprise Architecture for delivery of intelligent decisioning services

Active Availability
Business Continuity to support the requirements of the business (up to 7X24)

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

Comparing Traditional and Active DW

Traditional Data Warehouse Environment	Active Data Warehouse Environment
Strategic decisions only	Strategic and tactical decisions
Results sometimes hard to measure	Results measured with operations
Daily, weekly, monthly data currency acceptable; summaries often appropriate	Only comprehensive detailed data available within minutes is acceptable
Moderate user concurrency	High number (1,000 or more) of users accessing and querying the system simultaneously
Highly restrictive reporting used to confirm or check existing processes and patterns; often uses predeveloped summary tables or data marts	Flexible ad hoc reporting, as well as machine-assisted modeling (e.g., data mining) to discover new hypotheses and relationships
Power users, knowledge workers, internal users	Operational staffs, call centers, external users

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

Data Warehouse Administration

- Due to its huge size and its intrinsic nature, a DW requires especially strong monitoring in order to sustain its efficiency, productivity and security.
- The successful administration and management of a data warehouse entails skills and proficiency that go past what is required of a traditional database administrator.
  - Requires expertise in high-performance software, hardware, and networking technologies

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

Efficient Computation of Data Cubes

- Preliminary cube computation tricks
- Computing full/iceberg cubes: 3 methodologies
  - Top-Down: Multi-Way array aggregation
  - Bottom-Up:
    - Bottom-up computation: BUC
    - H-cubing technique
    - Integrating Top-Down and Bottom-Up:
    - Star-cubing algorithm
    - High-dimensional OLAP: A Minimal Cubing Approach
- Computing alternative kinds of cubes:
  - Partial cube, closed cube, approximate cube, etc.

Source: Han & Kamber (2006)

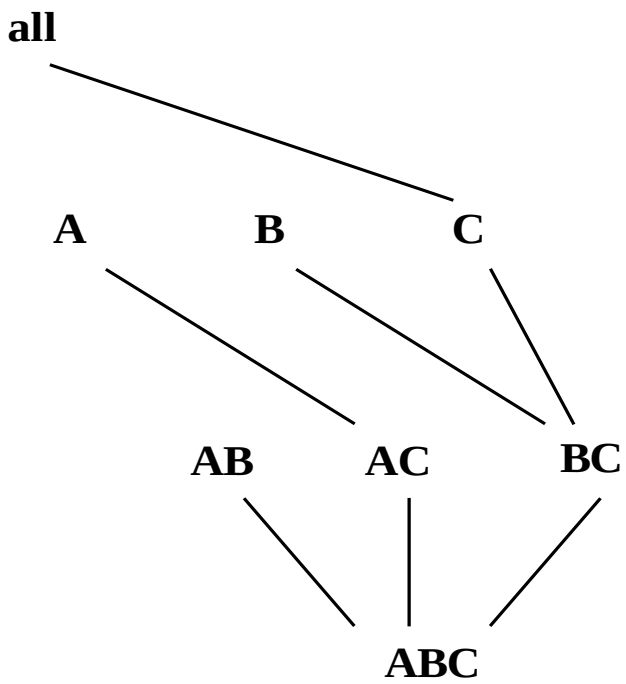
Preliminary Tricks (Agarwal et al. VLDB'96)

- 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

Source: Han & Kamber (2006)

Multi-Way Array Aggregation

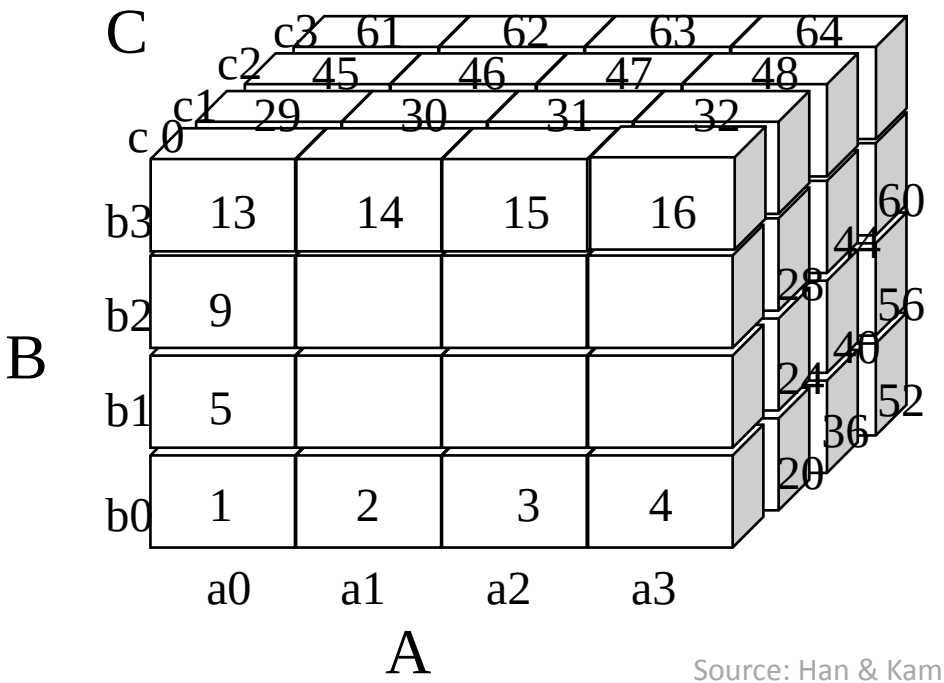
- Array-based "bottom-up" algorithm
- Using multi-dimensional chunks
- No direct tuple comparisons
- Simultaneous aggregation on multiple dimensions
- Intermediate aggregate values are re-used for computing ancestor cuboids
- Cannot do *Apriori* pruning: No iceberg optimization



Source: Han & Kamber (2006)

Multi-way Array Aggregation for Cube Computation (MOLAP)

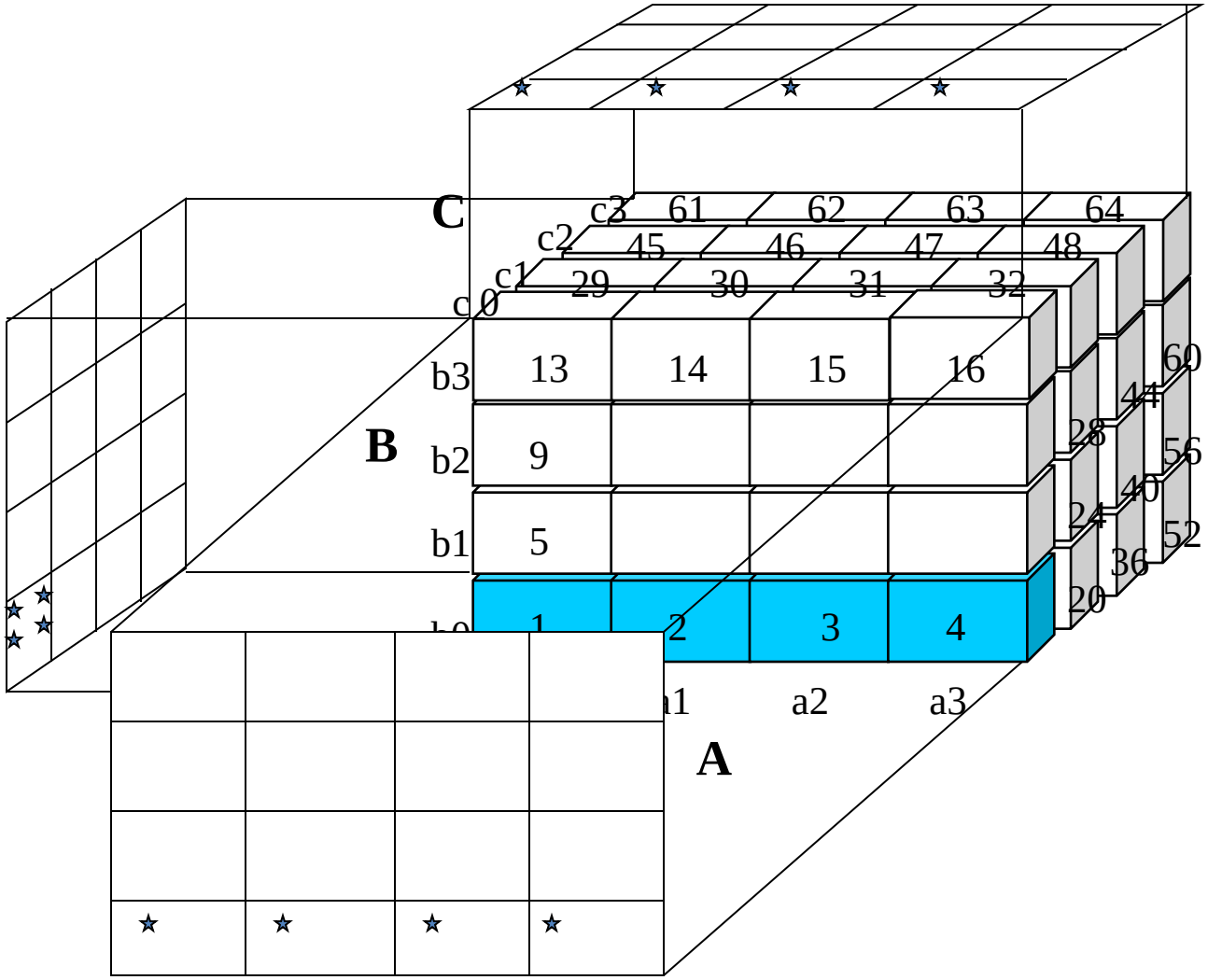
- Partition arrays into chunks (a small subcube which fits in memory).
- Compressed sparse array addressing: (chunk_id, offset)
- Compute aggregates in "multiway" by visiting cube cells in the order which minimizes the # of times to visit each cell, and reduces memory access and storage cost.



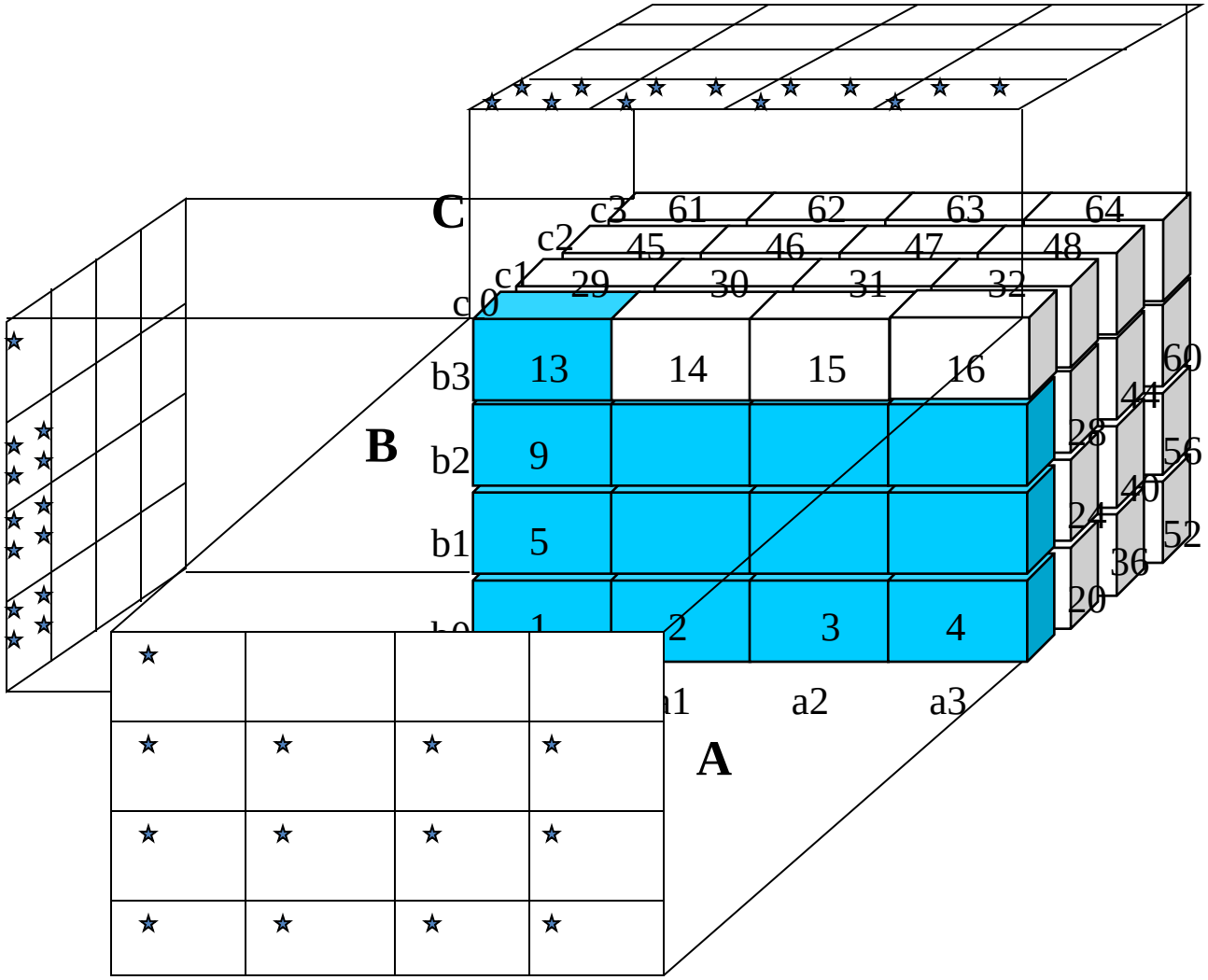
What is the best traversing order to do multi-way aggregation?

Source: Han & Kamber (2006)

Multi-way Array Aggregation for Cube Computation



Multi-way Array Aggregation for Cube Computation



Source: Han & Kamber (2006)

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

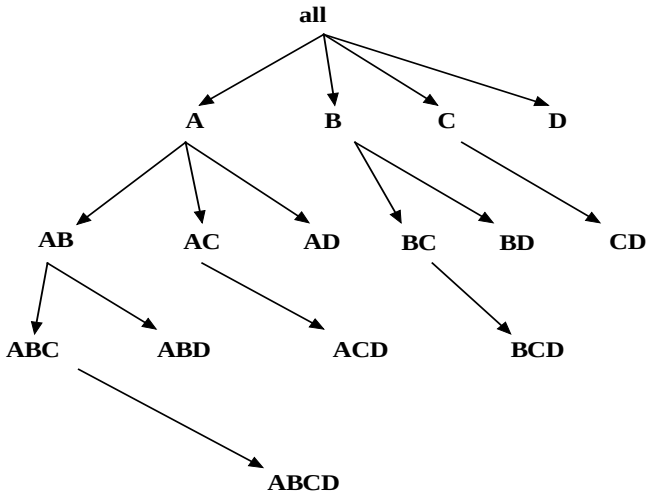
- Method: the planes should be sorted and computed according to their size in ascending order
  - Idea: keep the smallest plane in the main memory, fetch and compute only one chunk at a time for the largest plane
- Limitation of the method: computing well only for a small number of dimensions
  - If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

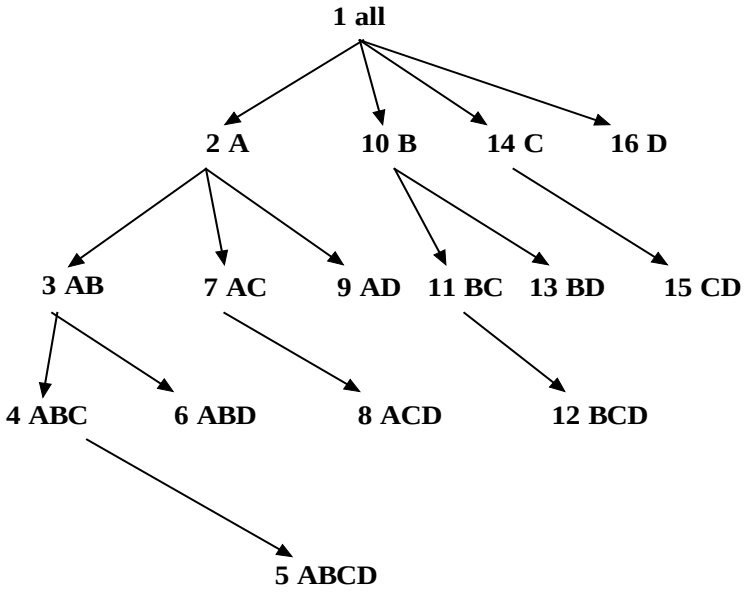
Source: Han & Kamber (2006)

Bottom-Up Computation (BUC)

- BUC (Beyer & Ramakrishnan, SIGMOD'99)
- Bottom-up cube computation

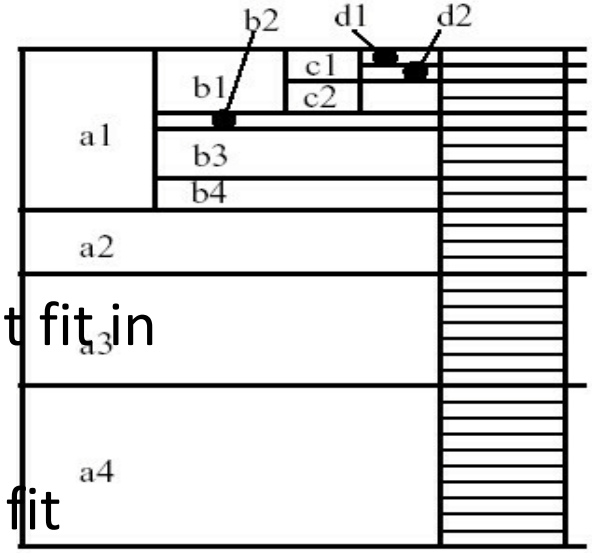
  (Note: top-down in our view!)
- Divides dimensions into partitions and facilitates iceberg pruning
  - If a partition does not satisfy *min_sup*, its descendants can be pruned
  - If *minsup* = 1 $\Rightarrow$ compute full CUBE!
- No simultaneous aggregation





Source: Han & Kamber (2006)

BUC: Partitioning

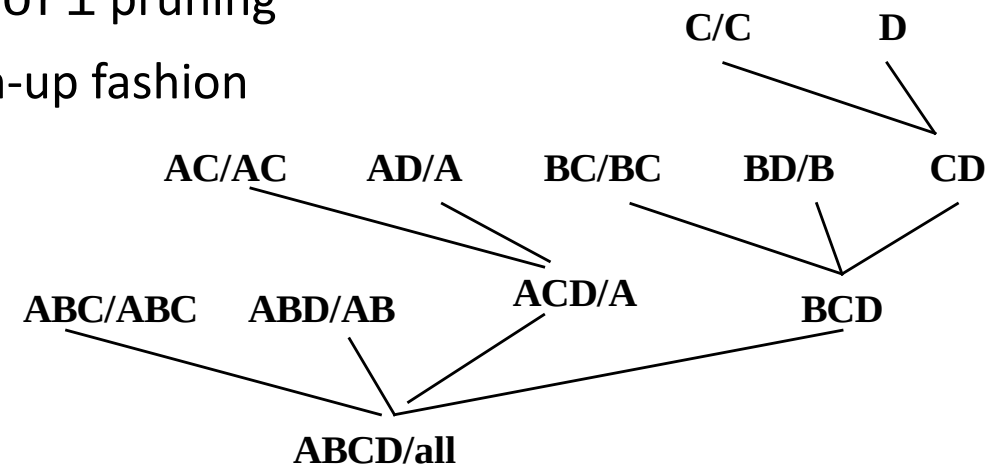


- Usually, entire data set can't fit in main memory
- Sort *distinct* values, partition into blocks that fit
- Continue processing
- Optimizations
  - Partitioning
    - External Sorting, Hashing, Counting Sort
  - Ordering dimensions to encourage pruning
    - Cardinality, Skew, Correlation
  - Collapsing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

Source: Han & Kamber (2006)

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



Source: Han & Kamber (2006)

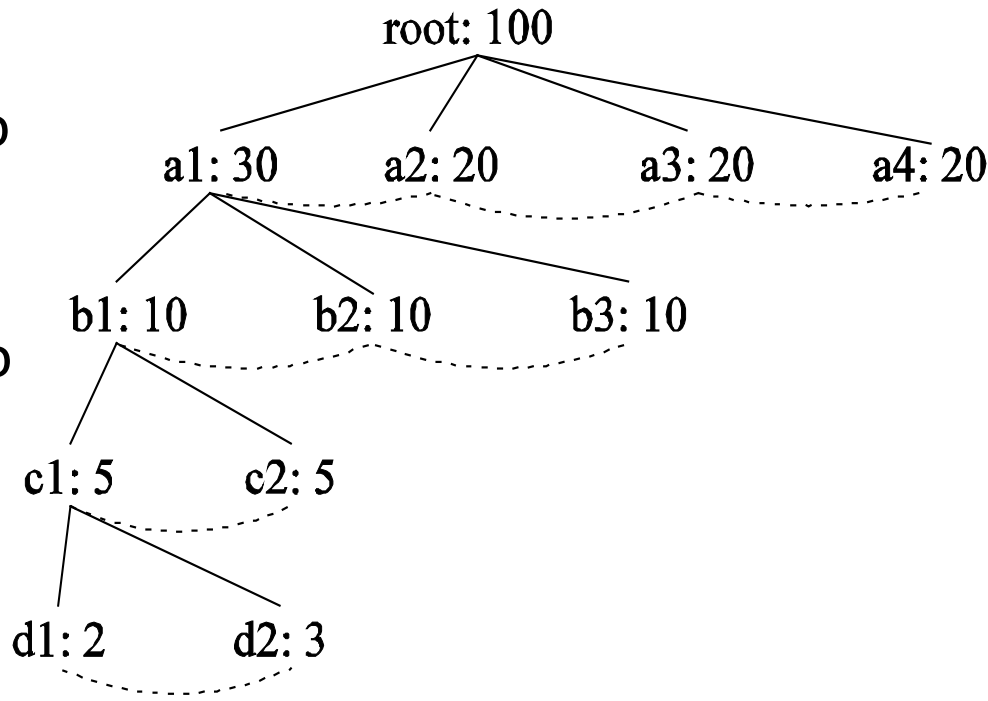
Iceberg Pruning in Shared Dimensions

- 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?

Source: Han & Kamber (2006)

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



Source: Han & Kamber (2006)

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

Source: Han & Kamber (2006)

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

Source: Han & Kamber (2006)

- 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

Source: Han & Kamber (2006)

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
```

Source: Han & Kamber (2006)

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
…	…	…	…	…	…

Source: Han & Kamber (2006)

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

Source: Han & Kamber (2006)

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

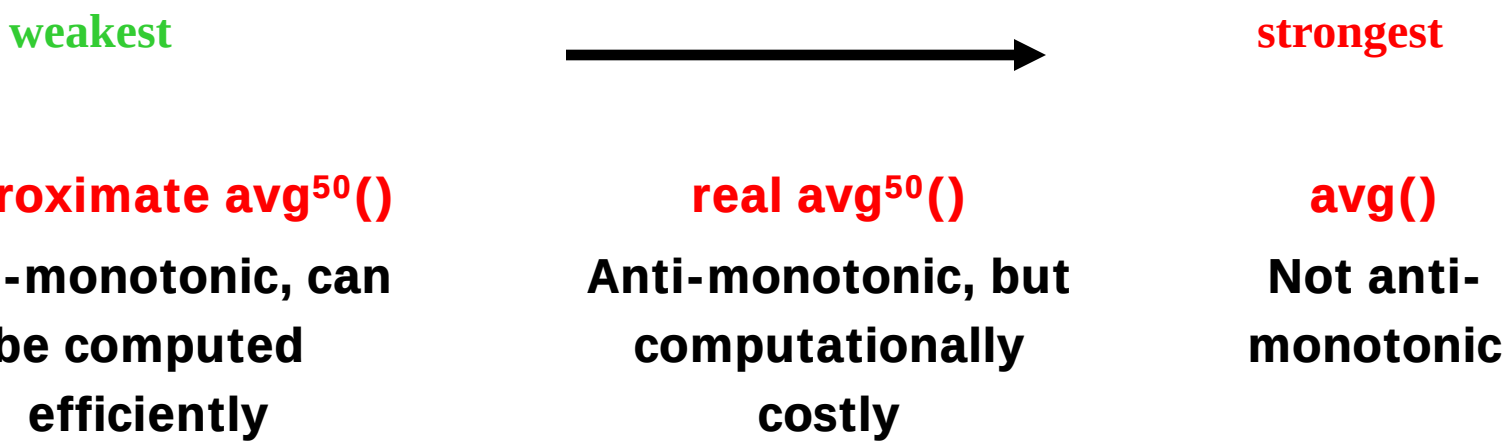
Approximate $avg^{50}()$=
(28000+10600+600*15)/50=952

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



Source: Han & Kamber (2006)

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

Source: Han & Kamber (2006)

- 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

Source: Han & Kamber (2006)

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

Source: Han & Kamber (2006)

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%

Source: Han & Kamber (2006)

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

Source: Han & Kamber (2006)

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

Source: Han & Kamber (2006)

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

Source: Han & Kamber (2006)

MD Constrained Gradient Mining

- Significance constraint C_{sig}: (cnt≥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)≥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

Source: Han & Kamber (2006)

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

Source: Han & Kamber (2006)

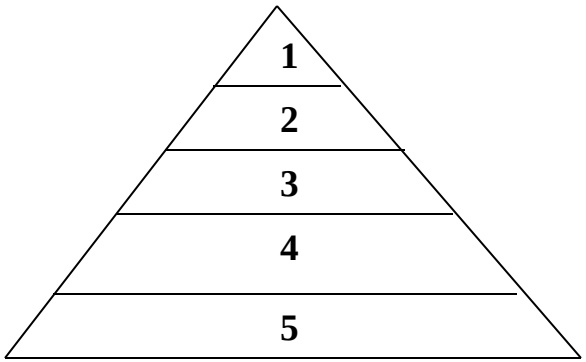
- Efficient Computation of Data Cubes
- Exploration and Discovery in Multidimensional Databases
- Attribute-Oriented Induction – An Alternative Data Generalization Method

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.

Higher level:
Young, Old

Lower level:
18 , 70 in Age



Conceptual levels

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

Source: Han & Kamber (2006)

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

Source: Han & Kamber (2006)

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

Source: Han & Kamber (2006)

Attribute-Oriented Induction

- Collect the task-relevant data (*initial relation*) using a relational database query
- Perform generalization by attribute removal or attribute generalization
- Apply aggregation by merging identical, generalized tuples and accumulating their respective counts
- Interactive presentation with users

Source: Han & Kamber (2006)

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" }

Source: Han & Kamber (2006)

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
…	…	…	…	…	…	…

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

Source: Han & Kamber (2006)

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.,

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

Source: Han & Kamber (2006)

Mining Class Comparisons

- Comparison: Comparing two or more classes
- Method:
  - 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
- Relevance Analysis:
  - Find attributes (features) which best distinguish different classes

Source: Han & Kamber (2006)

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

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

Source: Han & Kamber (2006)

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

$$\forall X,\ graduate\_student(X) \Leftarrow$$
$$birth\_country(X) = "Canada" \wedge age\_range(X) = "25-30" \wedge gpa(X) = "good" \quad [d:30\%]$$

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

Source: Han & Kamber (2006)

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

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

$$condition_1(X)[t : w_1, d : w'_1] \vee \ldots \vee condition_n(X)[t : w_n, d : w'_n]$$

  - necessary and sufficient

Source: Han & Kamber (2006)

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*

$$\forall X, Europe(X) \Leftrightarrow$$
$$(item(X) = "TV") [\mathrm{t}: 25\%, \mathrm{d}: 40\%] \vee (item(X) = "computer") [\mathrm{t}: 75\%, \mathrm{d}: 30\%]$$

Source: Han & Kamber (2006)

Summary

- Efficient algorithms for computing data cubes
- Further development of data cube technology
  - Discovery-drive cube
  - Multi-feature cubes
  - Cube-gradient analysis
- Alternative Data Generalization Method : Attribute-Oriented Induction

References

- Jiawei Han and Micheline Kamber, Data Mining: Concepts and Techniques, Second Edition, 2006, Elsevier
- Efraim Turban, Ramesh Sharda, Dursun Delen, Decision Support and Business Intelligence Systems, Ninth Edition, 2011, Pearson.