CS4.405 : Data Analytics-I

P. Krishna Reddy

IIIT Hyderabad
E-mail: pkreddy@iiit.ac.in
http://www.iiit.net/~pkreddy

Prof. P. Krishna Reddy
Data Science and Analytics Center, IIIT Hyderabad, India
E-mail: pkreddy@iiit.ac.in; http://www.iiit.ac.in/~pkreddy
Phone: 040-66531322; +91-9849329324

Research Areas

•Data mining/ Web Mining/ Bio-mining

Diverse frequent patterns, Coverage Patterns, Periodic frequent patterns, Rare knowledge extraction, Classification/prediction, Clustering.

Internet monetization, Search engine advertisement, Banner advertisement placemen, Recommendation systems, Link-based community extraction, E-commerce, Information extraction

Protein function prediction.

•Database Systems/ Systems Building

Schema summarization, Transaction synchronization, Database recovery, Data replication, Speculative transaction management, Memory efficient mining, User interfaces, Weather-based decision support systems.

•IT for Agriculture

Personalized agro-advice advisory, Knowledge transfer, Systems for bridging lab to land gap, Virtual crop labs for agriculture education, Location-specific content development, Agriculture extension, Agriculture planning tools for a farmer.

•IT for law:

Finding similar judgements

Systems Built

•eSagu: An IT-based Personalized Agro-Advisory System
•eAquaSagu: An IT-based Personalized Aqua-Advisory System
•eAgromet: An IT-based agro-meteorological advisory system
•Village-level eSagu: A Scalable and Location-Specific Agro-advisory System

Key Ideas Developed

•Coverage patterns, Diverse Patterns
•Dense-bipartite graph based web communities
•Rare frequent /knowledge patterns.
•Periodic frequent patterns
•Weather condition, Coupled weather condition

•Deadlock prevention using data flow graphs.
•Replica synchronization using data flow graphs
•Backup commit protocol
•Speculative locking protocol for regular and read-only transactions
•Temporality-based user interface design
•Query by object based user interface design

•Personalized agro-advisory (eSagu and eAgromet)
•Post-graduate Diploma in applied agriculture and IT (PGDAAIT)
•Virtual crop labs.

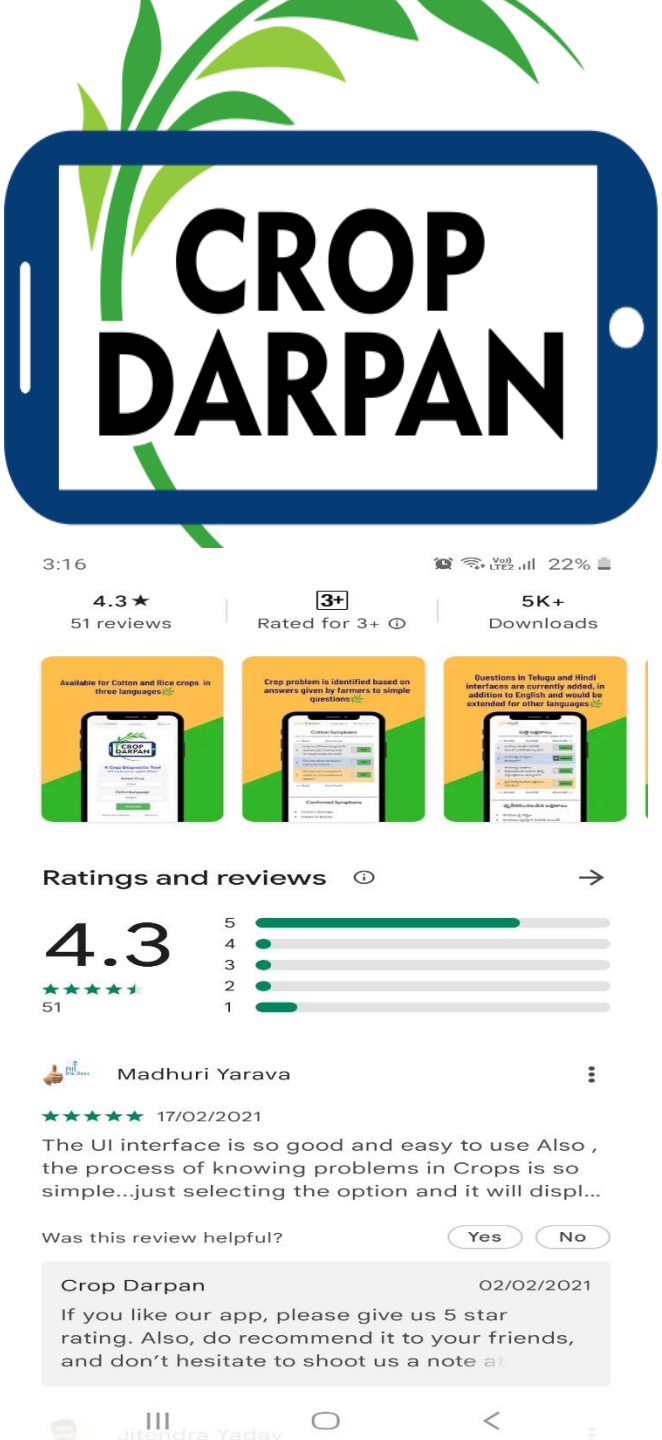
Future Research

•Diverse frequent patterns based recommendation and search systems.
•Coverage patterns based banner/search engine advertisement management
•Periodic frequent pattern based knowledge extraction system
•Efficient knowledge transfer systems
•Coverage pattern based subgraph discovery and visual mining
•Building query by object interface for non-SQL users
•Speculation-based synchronization for big data
•Protein function prediction
•Scalable decision support system for agriculture
•Building of resource planning tool for farmers
•Virtual crop labs for enhancing practical agricultural education.
•Decision support system for law

Farm diagnostic tool

Install Crop Darpan App in your smart phone

- For desktop, the system is available at www.cropdarpan.com
- Crop Darpan App is available (on Andoid OS) at play store. Download and give feedback. https://play.google.com/store/apps/details?id=in.iiit.cropdarpan
- App is live on Apple App Store. https://apps.apple.com/us/app/crop-darpan/id1556486922#?platform=iphone



Introduction: Presentation Outline

- **Why Data Analytics (or data mining)?**
  - **Courses you have Completed**
  - Content of human mind
  - Sample data mining problems
- Data mining definition and KDD process
- Multidimensional view of data mining
- Data mining functionalities
- Course outline
- Summary and History of data mining society

Courses You have Completed

- Digital Logic Design
  - Able to carry out arithmetic and logic operations
- Operating systems
  - Able to store the file and retrieve the file
- Programming language
  - Able to write a program, given the requirement
- Algorithms, data structures
  - Efficient computation given the requirement: sorting and searching
- Database systems
  - Able to store data and retrieve data based on SQL

- Questions
  - What will you do with 10,000 or 100,000 rows result?
  - Can we get some new insights? If yes, how?

Introduction: Presentation Outline

- **Why Data Analytics (or data mining)?**
  - Courses you have Completed
  - **Content of human mind**
  - Sample data mining problems
- Data mining definition and KDD process
- Multidimensional view of data mining
- Data mining functionalities
- Course outline
- Summary and History of data mining society

Data, Information, Knowledge, and Wisdom by Gene Bellinger, Durval Castro, Anthony Mills

- According to Russell Ackoff, content of human mind can be classified into five categories: Data, Information, Knowledge, Understanding and wisdom

- Data: Symbols
  - **Data represents a fact or a statement of event without relation to other things.**
  - Data is raw. It simply exists and has no significance beyond its existence (in and of itself). It can exist in any form, usable or not. It does not have meaning of itself. In computer parlance, a spreadsheet generally starts out by holding data.
  - Ex: It is raining.

Content of Human Mind

- **Information:** Data that are processed to be useful; provides answer to "who", "what", "where", and "when" questions.
  - Information is data that has been given meaning by way of relational connection. This "meaning" can be useful, but does not have to be.
  - Information embodies **the understanding** of a relationship of some sort, possibly cause and effect.
  - Example The temperature dropped 15 degrees and then it started raining.
  - In computer parlance, a relational database makes information from the data stored within it.

Content of Human Mind

- Knowledge: application of data and information; answers “how” questions.
  - Knowledge represents a pattern that connects and generally providing a high level of predictability as what is described or what will happen next.
  - Knowledge is the appropriate collection of information, such that it's intent is to be useful. Knowledge is a deterministic process.
  - When someone "memorizes" information (as less-aspiring test-bound students often do), then they have amassed knowledge. This knowledge has useful meaning to them, but it does not provide for, in and of itself, an integration such as would infer further knowledge.

Content of Human Mind

- Knowledge: application of data and information; answers "how" questions.
  - …
  - For example, elementary school children memorize, or amass knowledge of, the "times table". They can tell you that "2 x 2 = 4" because they have amassed that knowledge (it being included in the times table). But when asked what is "1267 x 300", they can not respond correctly because that entry is not in their times table.
  - To correctly answer such a question requires a true cognitive and analytical ability that is only encompassed in the next level... understanding.
  - **Ex: If the humidity is very high and the temperature drops suddenly, the atmosphere is often unlikely to be able to hold the moisture, so it rains.**
  - In computer parlance, most of the applications we use (modeling, simulation, **data mining** etc.) some type of stored knowledge.

Content of Human Mind

- Understanding: appreciation of why
  - Understanding is an interpolative and probabilistic process. It is cognitive and analytical.
  - It is the process by which one can take knowledge and synthesize new knowledge from previously held knowledge.
  - The difference between understanding and knowledge is between "learning" and "memorizing".
  - People who have understanding can undertake useful actions because they can synthesize new knowledge, or in some cases, at least new information, from what is previously known (and understood).
  - **That is, understanding can build upon currently held information, knowledge and understanding itself.**
  - In computer parlance, AI systems possess understanding in the sense that they are able to synthesize new knowledge from previously stored information and knowledge.

Content of human mind

- **Wisdom: evaluated understanding**
- Wisdom embodies more of an understanding of fundamental principles embodied within the knowledge that are essentially the basis for the knowledge being what it is. Wisdom is essentially systemic.
  - Ex: It rains because it rains. And this encompasses an understanding of all the interactions that happen between raining, evaporation, air currents, temperature gradients, changes, raining.
  - Wisdom is an extrapolative and non-deterministic, non-probabilistic process.
  - It calls upon all the previous levels of consciousness, and specifically upon special types of human programming (moral, ethical codes, etc.).
  - **It beckons to give us understanding about which there has previously been no understanding, and in doing so, goes far beyond understanding itself.**

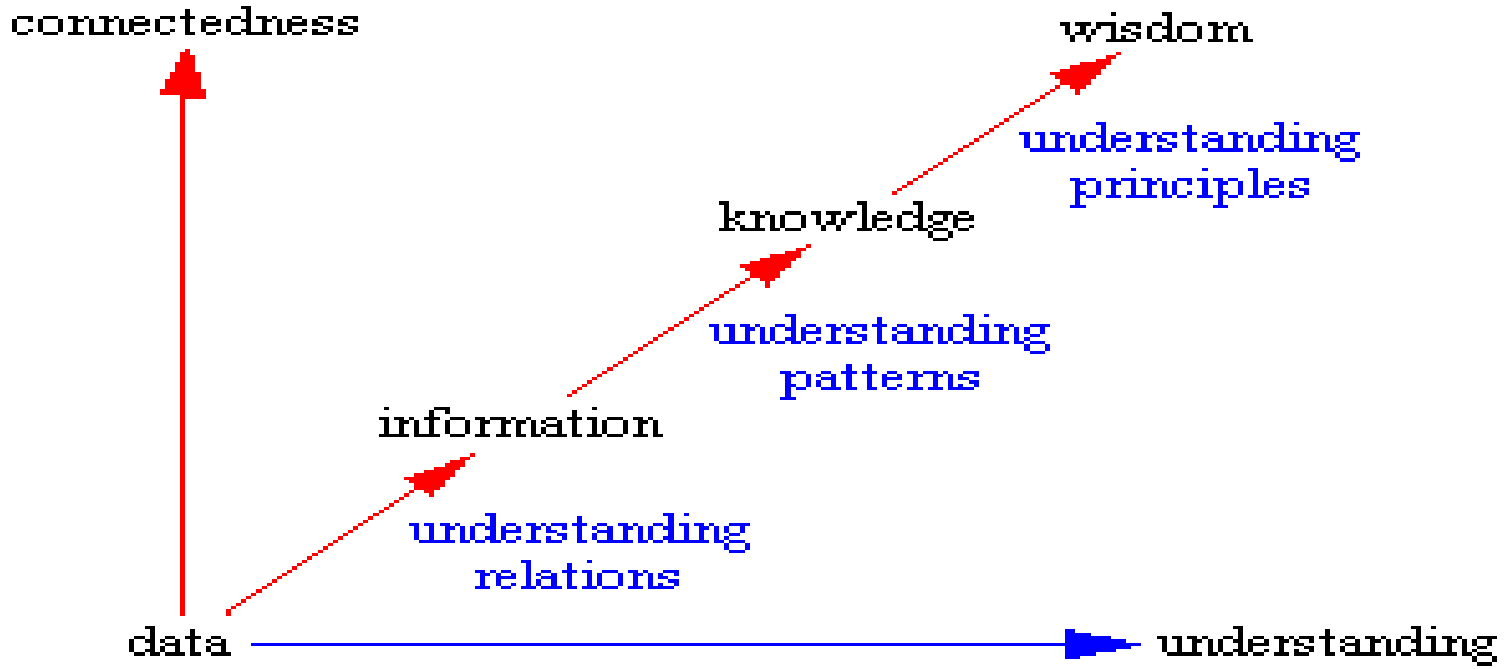
Content of human mind

- **Wisdom: evaluated understanding**
  - It is the essence of philosophical probing.
  - Unlike the previous four levels, it asks questions to which there is no (easily-achievable) answer, and in some cases, to which there can be no humanly-known answer period.
  - Wisdom is therefore, the process by which we also discern, or judge, between right and wrong, good and bad.

  - I personally believe that computers do not have, and will never have the ability to posses wisdom.
  - Wisdom is a uniquely human state, or as I see it, wisdom requires one to have a soul, for it resides as much in the heart as in the mind. And a soul is something machines will never possess (or perhaps I should reword that to ay, a soul is something that, in general, will never possess a machine).

Content of human mind

- Understanding that supports the transition from each stage to the next.
- Understanding is not a separate level of its own.
- Important: Ability to connect



Examples

- Example 1
  - Abugt dbesbt regtc uatn s uitrzt.
  - ubtxte pstye ysote anet sser extess
  - bxtedstes bet3 ibtes otesb tapbesct ehracts
- There is no foundation for you to connect with the pattern. If you know or **understand** the translation, these are Newton's 3 laws of motion

- Example 2
  - I have a box.
  - The box is 3' wide, 3' deep, and 6' high.
  - The box is very heavy.
  - The box has a door on the front of it.
  - When I open the box it has food in it.
  - It is colder inside the box than it is outside.
  - You usually find the box in the kitchen.
  - There is a smaller compartment inside the box with ice in it.
  - When you open the door the light comes on.
  - When you move this box you usually find lots of dirt underneath it.
  - Junk has a real habit of collecting on top of this box.
- At some point in the sequence, you connected **(understand)** with the pattern and understood it was a description of a refrigerator. From that point on each statement only added confirmation to **your understanding**.
- But, if you lived in a society that had never seen a refrigerator you might still be scratching your head as to what the sequence of statements referred to.

Introduction: Presentation Outline

- **Why Data Analytics (or data mining)?**
  - Courses you have Completed
  - Content of human mind
  - **Sample data mining problems**
- Data mining definition and KDD process
- Multidimensional view of data mining
- Data mining functionalities
- Course outline
- Summary and History of data mining society

Sample data mining problem # 1

I manage a supermarket (restaurant, video store, book store), and my cash register (or website) pumps transactions into my DB.

- Can you help me visualize my sales?
- Can you profile my customers?
- Tell me something interesting …
- I do not know statistics, and I do not want to hire statisticians.

Sample data mining problem #2

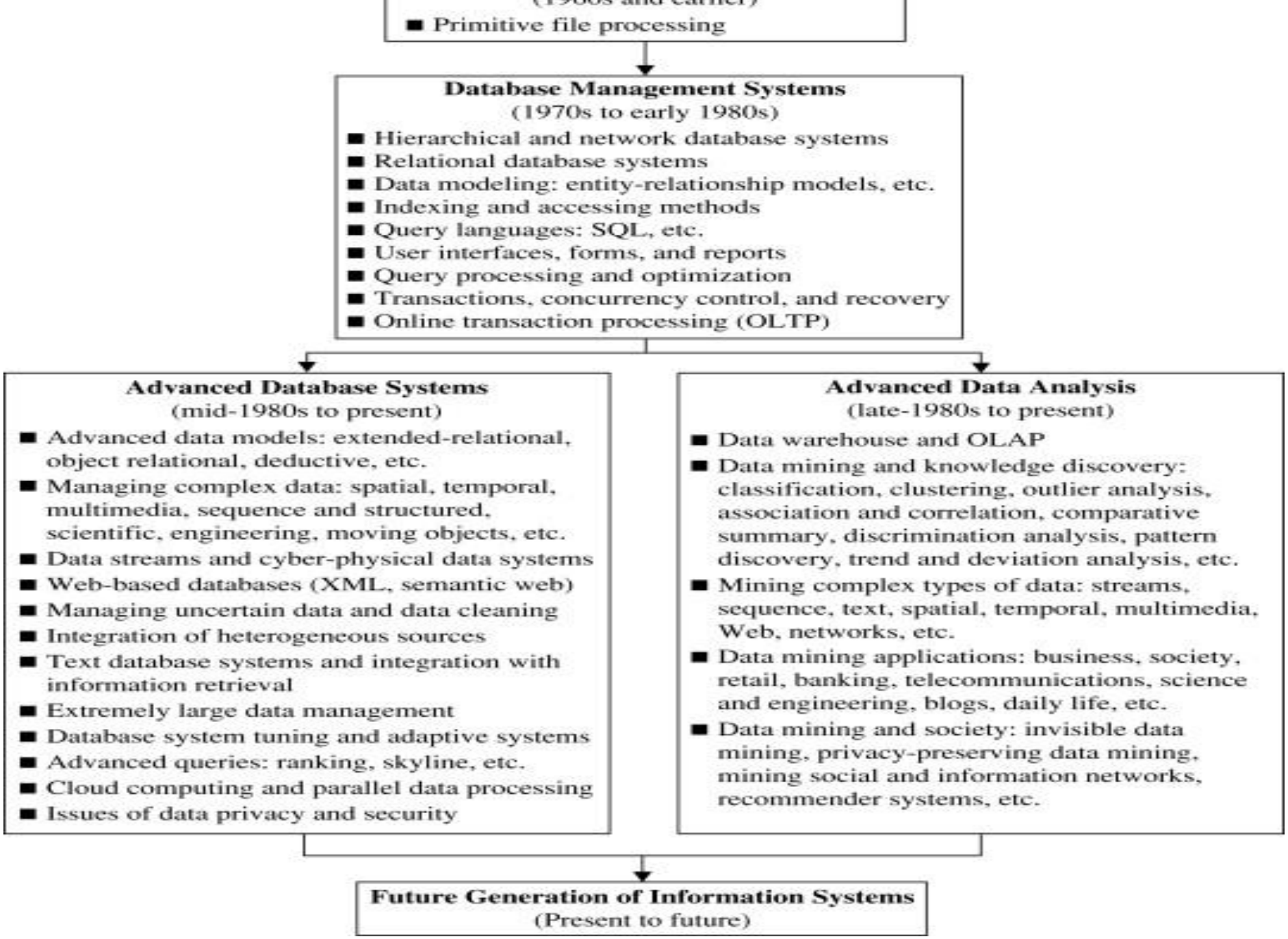
- I am an astronomer and I have sky survey 3 tera bytes of data, 2 billion objects.
  - Can you help to recognize the objects ?
  - Most of my data is beyond my reach.
  - Can you find new/unusual items in my data ?
  - Can you help me with basic manipulation, so I can focus on basic science ?
  - I know my data and statistics, but that is not enough …

Issue of Knowledge

- The Explosive Growth of Data: from terabytes to petabytes
  - Data collection and data availability
    - Automated data collection tools, database systems, Web, computerized society
  - Major sources of abundant data
    - Business: Web, e-commerce, transactions, stocks, …
    - Science: Remote sensing, bioinformatics, scientific simulation, …
    - Society and everyone: news, digital cameras, YouTube
- **We are drowning in data, but starving for knowledge!**
- "Necessity is the mother of invention"—Data mining—Automated analysis of massive data sets

Evolution of Sciences

- Before 1600, **empirical science**
- 1600-1950s, **theoretical science**
  - Each discipline has grown a *theoretical* component. Theoretical models often motivate experiments and generalize our understanding.
- 1950s-1990s, **computational science**
  - Over the last 50 years, most disciplines have grown a third, *computational* branch (e.g. empirical, theoretical, and computational ecology, or physics, or linguistics.)
  - Computational Science traditionally meant simulation. It grew out of our inability to find closed-form solutions for complex mathematical models.
- 1990-now, **data science**
  - The flood of data from new scientific instruments and simulations
  - The ability to economically store and manage petabytes of data online
  - The Internet and computing Grid that makes all these archives universally accessible
  - Scientific info. management, acquisition, organization, query, and visualization tasks scale almost linearly with data volumes. Data mining is a major new challenge!
- Jim Gray and Alex Szalay, *The World Wide Telescope: An Archetype for Online Science*, Comm. ACM, 45(11): 50-54, Nov. 2002


(1960s and earlier)
Primitive file processing
Database Management Systems
(1970s to early 1980s)
Hierarchical and network database systems
Relational database systems
Data modeling: entity-relationship models, etc.
Indexing and accessing methods
Query languages: SQL, etc.
User interfaces, forms, and reports
Query processing and optimization
Transactions, concurrency control, and recovery
Online transaction processing (OLTP)
Advanced Database Systems
(mid-1980s to present)
Advanced data models: extended-relational, object relational, deductive, etc.
Managing complex data: spatial, temporal, multimedia, sequence and structured, scientific, engineering, moving objects, etc.
Data streams and cyber-physical data systems
Web-based databases (XML, semantic web)
Managing uncertain data and data cleaning
Integration of heterogeneous sources
Text database systems and integration with information retrieval
Extremely large data management
Database system tuning and adaptive systems
Advanced queries: ranking, skyline, etc.
Cloud computing and parallel data processing
Issues of data privacy and security
Advanced Data Analysis
(late-1980s to present)
Data warehouse and OLAP
Data mining and knowledge discovery: classification, clustering, outlier analysis, association and correlation, comparative summary, discrimination analysis, pattern discovery, trend and deviation analysis, etc.
Mining complex types of data: streams, sequence, text, spatial, temporal, multimedia, Web, networks, etc.
Data mining applications: business, society, retail, banking, telecommunications, science and engineering, blogs, daily life, etc.
Data mining and society: invisible data mining, privacy-preserving data mining, mining social and information networks, recommender systems, etc.
Future Generation of Information Systems
(Present to future)

Introduction: Presentation Outline

- Why Data Analytics (or data mining)?
  - Courses you have Completed
  - Content of human mind
  - Sample data mining problems
- **Data mining definition and KDD process**
- Multidimensional view of data mining
- Data mining functionalities
- Course outline
- Summary and History of data mining society

What Is Data Mining?

- Data mining (knowledge discovery from data)
  - Extraction of interesting (non-trivial, implicit, previously unknown and potentially useful) patterns or knowledge from **huge** amount of data
  - Data mining: a misnomer?
- Alternative names
  - Knowledge discovery (mining) in databases (KDD), knowledge extraction, data/pattern analysis, data archeology, data dredging, information harvesting, business intelligence, etc.
- Watch out: Is everything "data mining"?
  - Simple search and query processing
  - (Deductive) expert systems

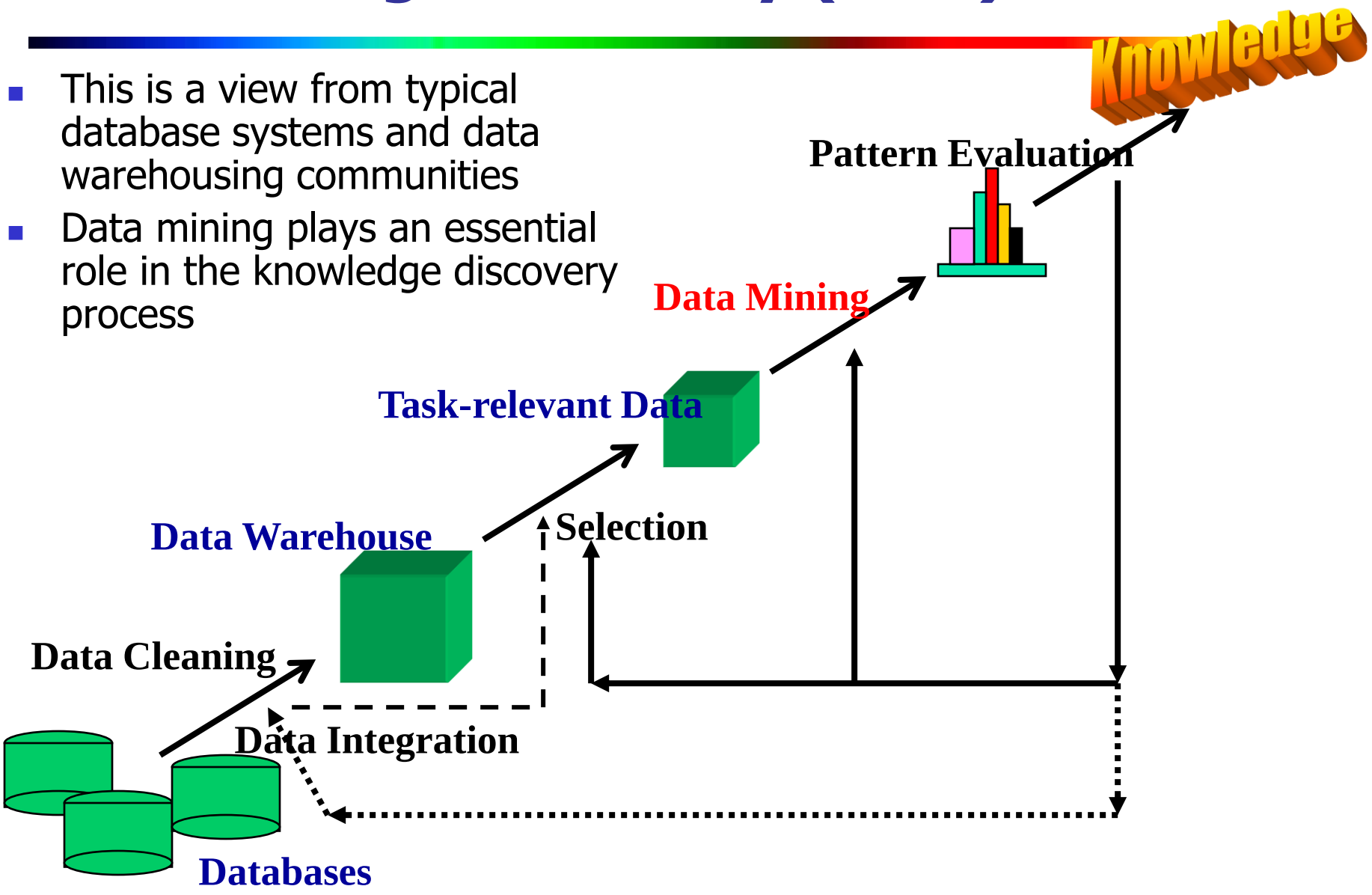
What is (not) Data Mining?

- **What is not Data Mining?**
  - Look up phone number in phone directory
  - Query a Web search engine for information about “Amazon”

- **What is Data Mining?**
  - Certain names are more prevalent in certain US locations (O’Brien, O’Rurke, O’Reilly… in Boston area)
  - Group together similar documents returned by search engine according to their context (e.g. Amazon rainforest, Amazon.com,)

Knowledge Discovery (KDD) Process

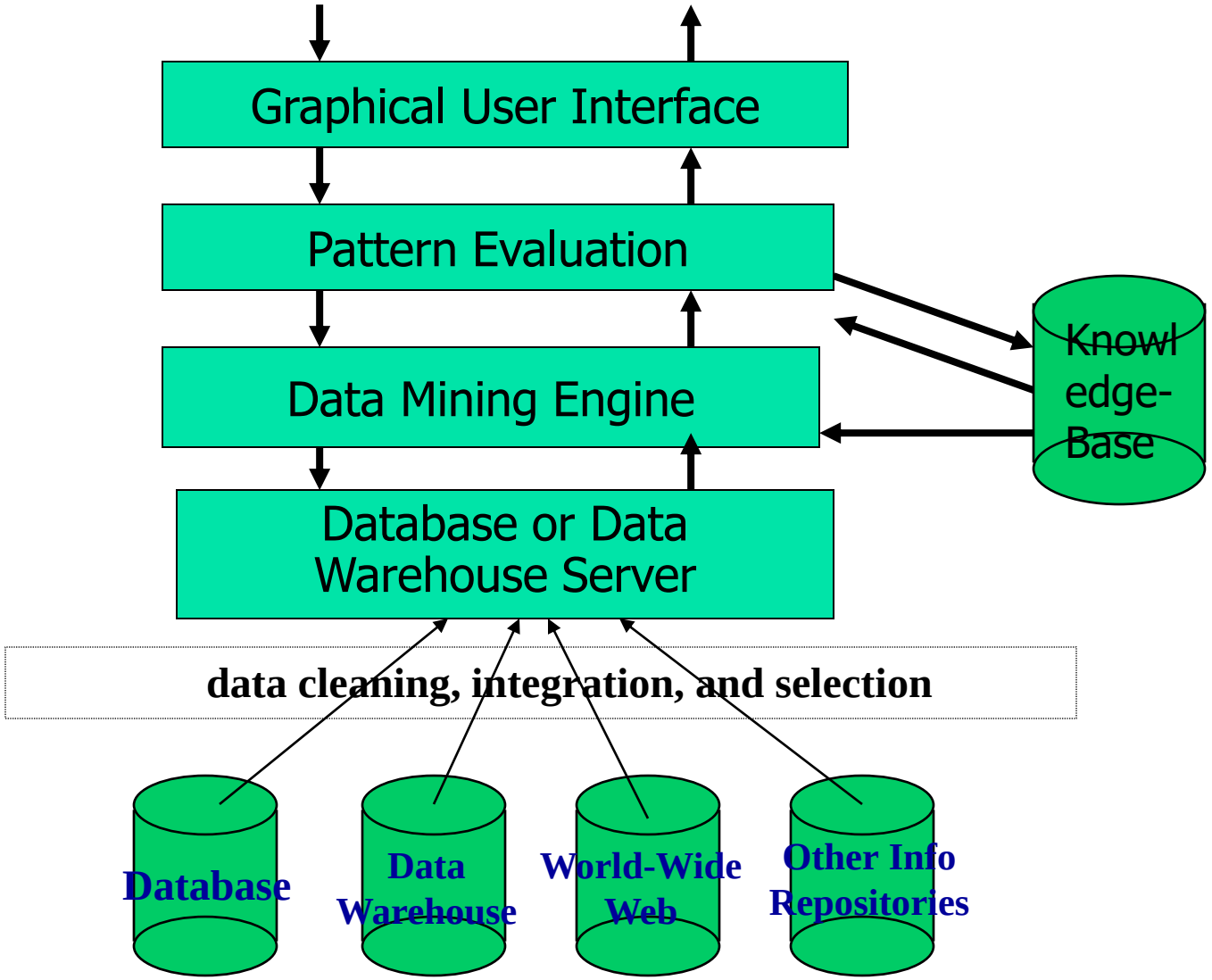
- This is a view from typical database systems and data warehousing communities
- Data mining plays an essential role in the knowledge discovery process



Steps of a KDD Process

- Learning the application domain:
  - relevant prior knowledge and goals of application
- Data cleaning: to remove noise and inconsistent data
- Data integration: Multiple data sources can be combined
- Creating a target data set: data selection
- Data cleaning and preprocessing: (may take 60% of effort!)
- Data reduction and transformation:
  - Find useful features, dimensionality/variable reduction, invariant representation.
- Choosing functions of data mining
  - summarization, association, classification, clustering.
- Choosing the mining algorithm(s)
- Data mining: search for patterns of interest
- Pattern evaluation and knowledge presentation
  - visualization, transformation, removing redundant patterns, etc.
- Use of discovered knowledge

Architecture: Typical Data Mining System



Components of data mining system

- **Database, Data warehouse, World Wide Web or other information Repository**
  - Data cleaning and data integration techniques are performed on this data
- **Database and data warehouse server:** Responsible for fetching the relevant data, based on the user's data mining request.
- **Knowledge-base:** Domain knowledge which is used to guide the data mining process.
  - Attribute levels, semantics, user beliefs, pattern interestingness, thrsholds, meta data
- **Data mining engine:** Set of functional modules for tasks such as characterization, summarization, association, classification, clustering, outlier extraction
- **Pattern evaluation:** Employees interestingness measures
  - Put the evaluation pattern as much deep as you can so that one can optimize.
- **User interface:** communication between users and the data mining system.

Data Mining in Business Intelligence

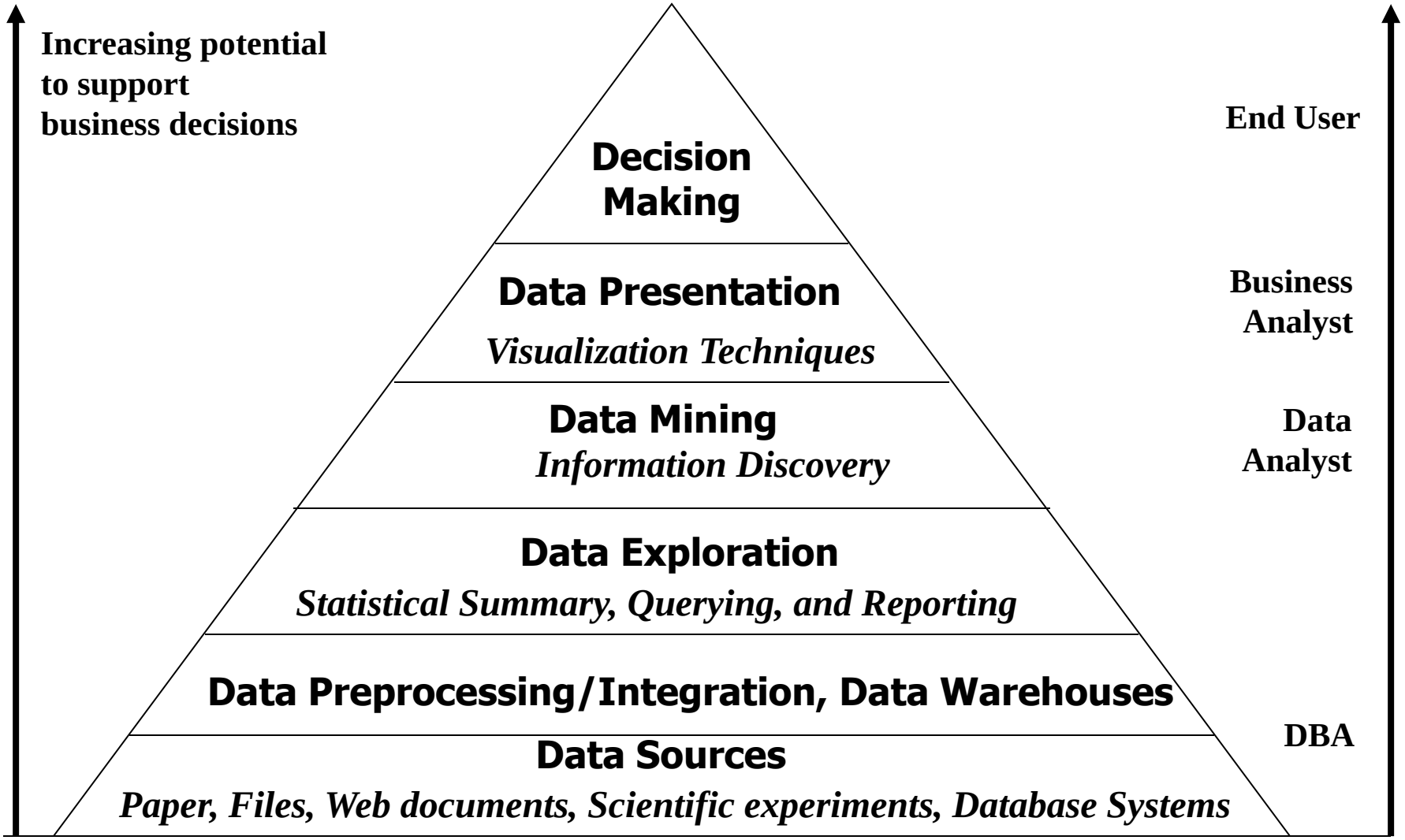
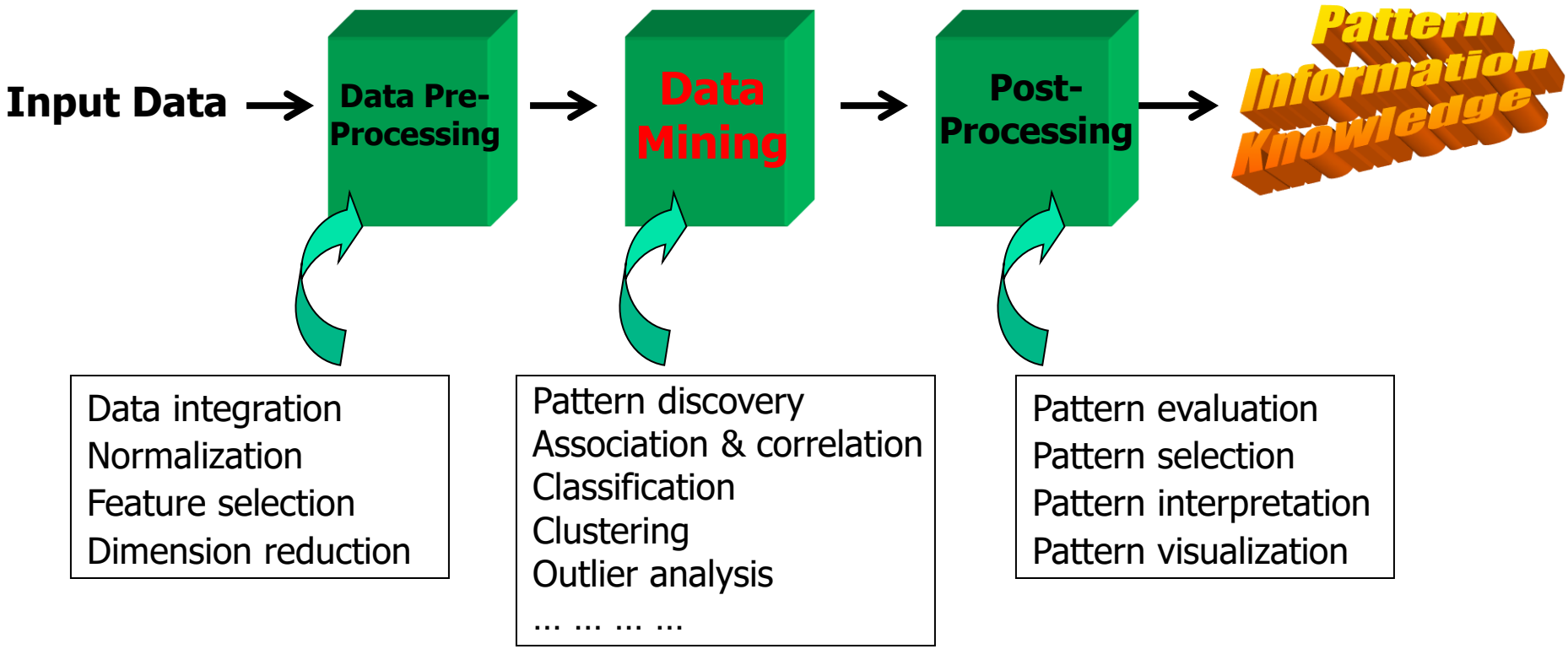


Fig. 3. Data-to-insight-to-decision whole-of-life analytics.

KDD Process: A Typical View from ML and Statistics



- This is a view from typical machine learning and statistics communities

Example: Medical Data Mining

- Health care & medical data mining – often adopted such a view in statistics and machine learning
- Preprocessing of the data (including feature extraction and dimension reduction)
- Classification or/and clustering processes
- Post-processing for presentation

Introduction: Presentation Outline

- Why Data Analytics (or data mining)?
  - Courses you have Completed
  - Content of human mind
  - Sample data mining problems
- Data mining definition and KDD process
- **Multidimensional view of data mining**
- Data mining functionalities
- Course outline
- Summary and History of data mining society

Multi-Dimensional View of Data Mining

- **Data to be mined**
  - Database data (extended-relational, object-oriented, heterogeneous, legacy), data warehouse, transactional data, stream, spatiotemporal, time-series, sequence, text and web, multi-media, graphs & social and information networks
- **Knowledge to be mined (or: Data mining functions)**
  - Characterization, discrimination, association, classification, clustering, trend/deviation, outlier analysis, etc.
  - Descriptive vs. predictive data mining
  - Multiple/integrated functions and mining at multiple levels
- **Techniques utilized**
  - Data-intensive, data warehouse (OLAP), machine learning, statistics, pattern recognition, visualization, high-performance, etc.
- **Applications adapted**
  - Retail, telecommunication, banking, fraud analysis, bio-data mining, stock market analysis, text mining, Web mining, etc.

Data Mining: On What Kinds of Data?

- Database-oriented data sets and applications
  - Relational database, data warehouse, transactional database
- Advanced data sets and advanced applications
  - Data streams and sensor data
  - Time-series data, temporal data, sequence data (incl. bio-sequences)
  - Structure data, graphs, social networks and multi-linked data
  - Object-relational databases
  - Heterogeneous databases and legacy databases
  - Spatial data and spatiotemporal data
  - Multimedia database
  - Text databases
  - The World-Wide Web

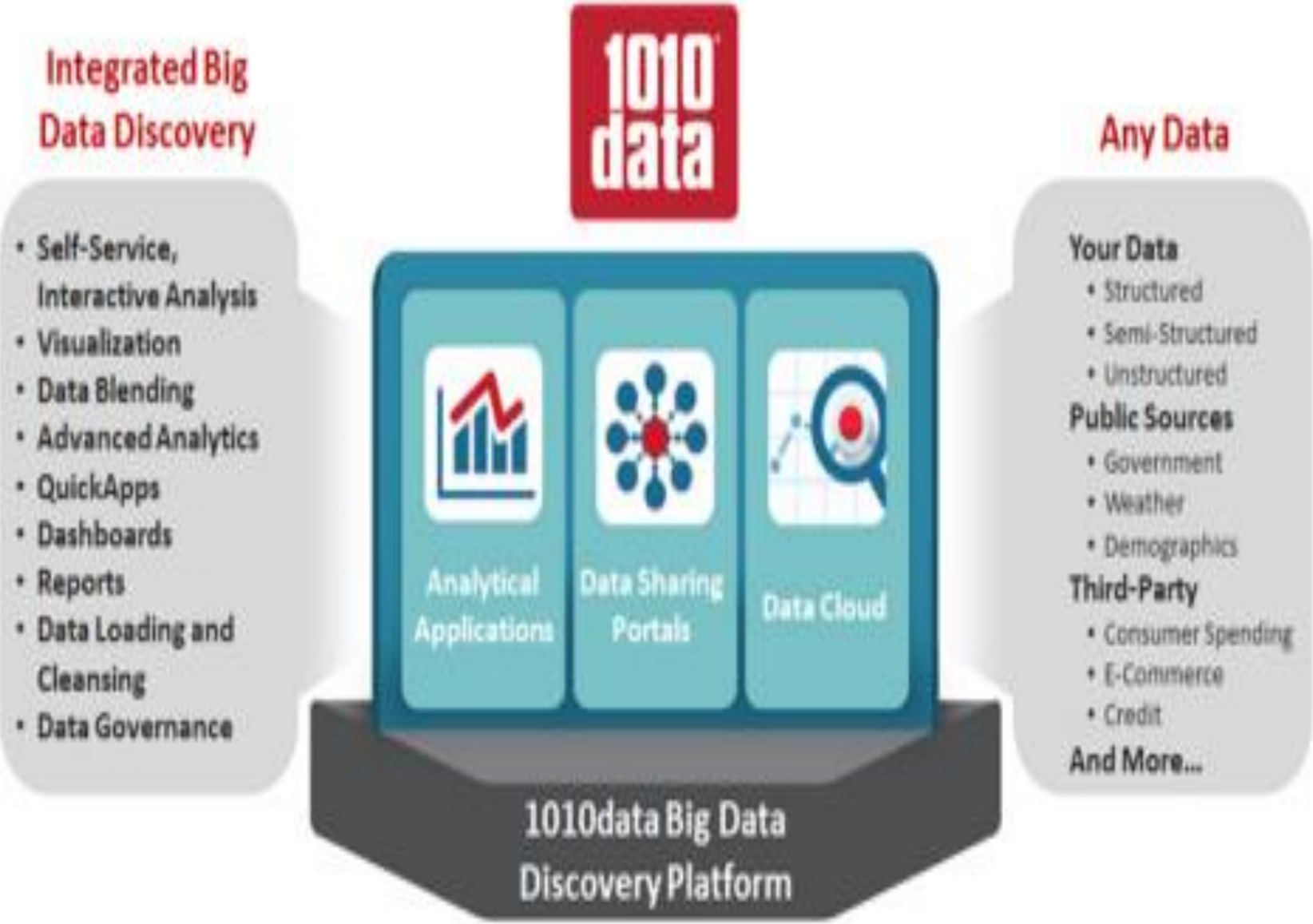
Introduction: Presentation Outline

- Why Data Analytics (or data mining)?
  - Courses you have Completed
  - Content of human mind
  - Sample data mining problems
- Data mining definition and KDD process
- Multidimensional view of data mining
- **Data mining functionalities**
- Course outline
- Summary and History of data mining society

Data Mining Functionalities

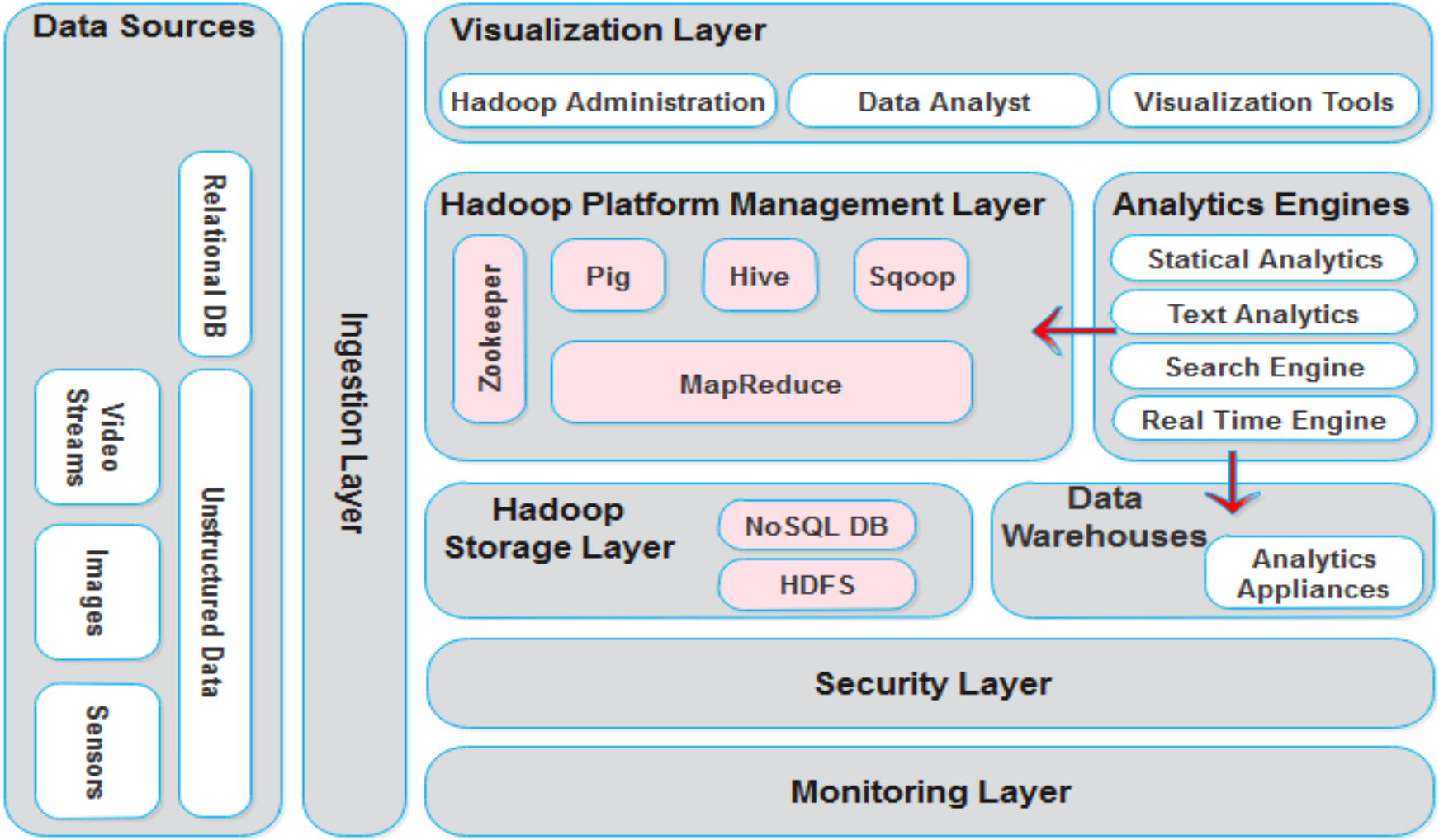
- Generalization/Summarization: characterization and discrimination
- Pattern mining, Association mining, correlation
- Classification
- Clustering
- Outlier analysis
- Sequential, trend and evolution analysis
- Structure and network analysis

16 Top Big Data Analytics Platforms


Integrated Big Data Discovery
• Self-Service, Interactive Analysis
• Visualization
• Data Blending
• Advanced Analytics
• QuickApps
• Dashboards
• Reports
• Data Loading and Cleansing
• Data Governance
1010 data
Analytical Applications
Data Sharing Portals
Data Cloud
1010data Big Data Discovery Platform
Any Data
Your Data
• Structured
• Semi-Structured
• Unstructured
Public Sources
• Government
• Weather
• Demographics
Third-Party
• Consumer Spending
• E-Commerce
• Credit
And More...

Top 10 Open Source Big Data Tools

Hadoop Big Data Architecture



Course Outline
(CS4.405: Data Analytics-I)

Prerequisite Course / Knowledge:

(i) Data and Applications, *or equivalent courses that cover Data modelling, normalization, SQL*

(ii) First courses on programming, data-structures and algorithms

(iii) Basics of Python language, to be able to use relevant libraries and toolkits for data analytics

Objective:

- In a computerized and networked society, vast amount of data is being collected every day in multiple domains.
- We are drowning in data, but starving for knowledge or actionable insights.
- Data mining or data analytics constitute a collection of concepts and algorithms, which are being developed to answer "how" questions by extracting interesting and useful knowledge of from large data.
- Data analytics based platforms are being operated in multiple domains to extract valuable and actionable insights from the data to improve the business performance.
- **The objective** of the course is to learn the important concepts and algorithms related to data analytics and data mining functionalities such as summarization, pattern mining, classification, clustering.
- **We will also briefly discuss the related research trends.**

Course Outcomes (COs)

- After completing the course successfully, the students are able to
  - CO-1. describe the concepts of data summarization, data warehousing, pattern mining, classification and clustering approaches
  - CO-2. perform the task of data summarization, pattern mining, classification and clustering based on the requirement.
  - CO-3. prescribe a single or a combination of data summarization, pattern mining, classification and clustering approaches for the problem scenario of a business/ organization.
  - CO-4. construct the improved data analytics methods for the existing services.
  - CO-5. formulate new data mining problems for creating new services and design the corresponding solutions

Detailed Syllabus

- Unit1:Introduction, data summarization through characterization, discrimination and data cube techniques (9 hours)
- Unit 2: Concepts and algorithms for mining patterns and associations (10 hours)
- Unit 3: Concepts and algorithms related to classification and regression (10 hours)
- Unit 4: Concepts and algorithms for clustering the data (10 hours)
- Unit 5: Outlier analysis and future trends. (3 hours)

Course Code: CS4.405: Data Analytics-I

Class Schedule (MONSOON 2024)

class	Date	Topic	Lab
1	1/8/2024 (THU)	Introduction to Data Mining	
2	5/8/2024 (MON)	Overview of data mining concepts, Knowing your data	Lab 1
3	8/8/2024 (THU)	Data preprocessing	
4	12/8/2024 (MON)	Attribute Oriented Induction	
5	16/8/2024 (FRI)	Data warehousing and OLAP technology	
6	19/8/2024 (MON)	DATA CUBE COMPUTATION ALGORITHMS 1. On the computation of multidimensional aggregates (Agarwal et al.'96) 2. Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)	Lab 2
7	22/8/2024 (THU)	DATA CUBE COMPUTATION ALGORITHMS 3.BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99) 4.H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)	
8	26/8/2024 (MON)	Quiz I	
9	29/8/2024 (THU)	Overview of other data cube algorithms: Star Cubing, Shell Fragments, Sampling Cubes, Ranking Cubes, Prediction Cubes, Discovery driven Cubes	
10	02/9/2024 (MON)	Introduction to Association mining	Lab 3
11	05/9/2024 (THU)	Apriori Algorithm and extensions	
12	09/9/2024 (MON)	FP growth algorithm	
13	12/9/2024 (THU)	ECLAT algorithm and pattern Evaluation	
14	19/9/2024 (THU)	Advances in Pattern Mining: Overview of Multi-level mining, Quantitative association rules, Utility mining, Diverse Patterns, Coverage Patterns	
		Midterm Exam	
15	26/9/2024 (THU)	Basic concepts	Lab 4
16	30/9/2024 (MON)	Decision Tree Induction	
17	3/10/2024 (THU)	Bayes Classification Methods	
18	7/10/2024 (MON)	Rule-Based Classification	
19	10/10/2024 (THU)	Model Evaluation and Selection, Techniques to Improve Accuracy	
20	14/10/2024 (MON)	Overview of Other classification methods: Bayesian belief network (probabilistic networks), Support Vector Machine (SVM), Pattern-based classification, lazy learners (KNN, case-based reasoning), genetic algorithms, rough set and fuzzy set approaches, Multiclass classification, Semi-supervised classification, Active learning, Transfer learning	
21	17/10/2024 (THU)	Quiz 2	
22	21/10/2024 (MON)	Introduction to clustering	
23	24/10/2024 (THU)	Partitioning methods	Lab 5
24	28/10/2024 (MON)	Hierarchical methods	
25	07/11/2024 (THU)	Chameleon algorithm and evaluation of clustering results	
26	11/11/2024 (TUE)	Overview: Advanced cluster analysis and outlier mining	
27	14/11/2024 (FRI)	Overview of deep learning	
28	18/11/2024(MON)	Data mining trends	

Lab

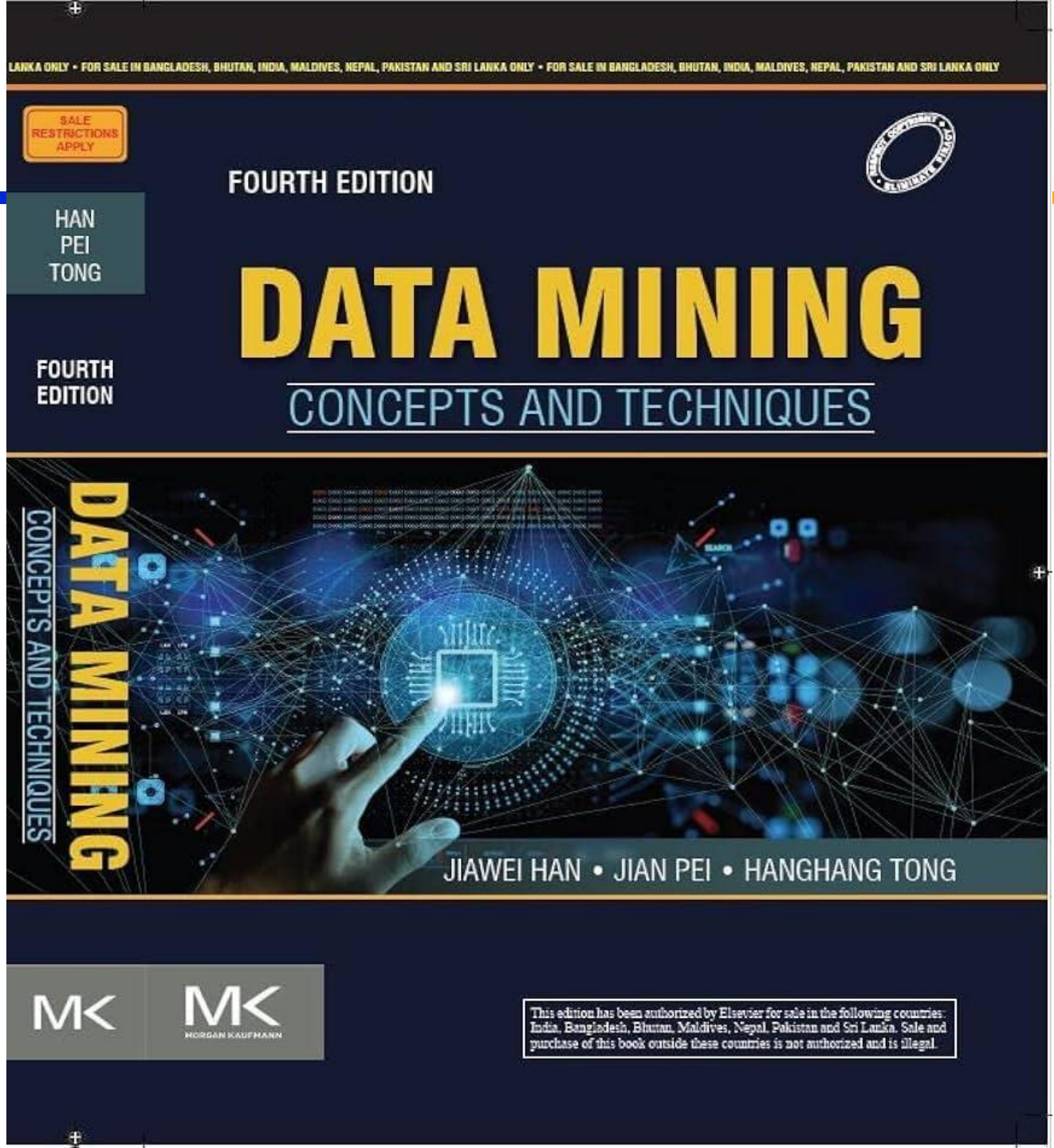
- Five mini projects related to the above syllabus will be done by students in the laboratory

Reference Books and materials:

1. **Book: Jiawei Han, Jian Pei, Hanghang Tong, Data Mining: Concepts and Techniques, Fourth edition, 2022, Elseiver Inc.**

2. Book: Pang-Nong Tan, Michael Steinbach and Vipin Kumar, Introduction to Data Mining, 2006, Pearson Education.

1. Research Papers: About 25 research papers from the proceeding of the conferences and journals related to data summarization, data warehousing, pattern mining, classification, clustering, outlier detection.


LANKA ONLY • FOR SALE IN BANGLADESH, BHUTAN, INDIA, MALDIVES, NEPAL, PAKISTAN AND SRI LANKA ONLY • FOR SALE IN BANGLADESH, BHUTAN, INDIA, MALDIVES, NEPAL, PAKISTAN AND SRI LANKA ONLY
SALE RESTRICTIONS APPLY
FOURTH EDITION
HAN
PEI
TONG
DATA MINING
CONCEPTS AND TECHNIQUES
FOURTH EDITION
DATA MINING
CONCEPTS AND TECHNIQUES
JIAWEI HAN • JIAN PEI • HANGHANG TONG
MK
MORGAN KAUFMANN
This edition has been authorized by Elsevier for sale in the following countries: India, Bangladesh, Bhutan, Maldives, Nepal, Pakistan and Sri Lanka. Sale and purchase of this book outside these countries is not authorized and is illegal.

Assessment methods and weightages

- Two Classroom tests: 10 marks (5+5)
- Mid Semester Examination in theory: 20 marks,
- End Semester Examination in Theory: 40 marks,
- Assessment of five mini projects in the Laboratory: 30 marks
- **New Idea**
  - **Based on the course contents, if you produce a new idea of publishable quality by doing additional work, additional weightage will be given.**

About new idea

- You can give two or three new ideas as per the below format (follow this
format for each idea).

  ****

  1. Title of the problem:
  2. Problem statement (about 50 words)
  3. Background (< 200 words)
  4. What is the state of art and gaps (<200 words)
  5. Description of the proposed solution (<500 words)
  6. Explain how the proposed solution will be different over existing approaches (<200 words)
  7. Explain how you will conduct the experiments and the corresponding performance metrics (<200 words)
  8. References (<10 references)

  ****

Lab Assignments

- You will be given the lab assignments in advance
  - The instructor will provide you with adequate background information to do the assignment
  - Some installations will be necessary. Install well in advance and if you encounter any problems with any installation, you can contact your TA
- Policies for lab
  - You should try to solve any programming issues by yourself first, remember that troubleshooting is how you would actually be learning a lot
  - Every time you encounter any minor programming problem, if you ask the TA or instructor, you will lose out on an excellent learning experience
  - Only if you are unable to solve the problem after putting in a reasonable amount of effort, you should contact your TA or your instructor
  - You should also look online for solutions to your problems, but do not copy code from anywhere
- Grading criteria
  - The overall quality of your lab assignment submission in terms of correctness, quality of code, system design etc.

Plagiarism

- This course has a zero-tolerance policy w.r.t. plagiarism
- Any instance of plagiarism will result in serious penalties (e.g., an F grade for the entire course, among other penalties)
- Forget about doing any kind of plagiarism

Deadlines

- Strict deadlines: You will not be able to submit after the deadline has already passed.

Accessing the Course Materials

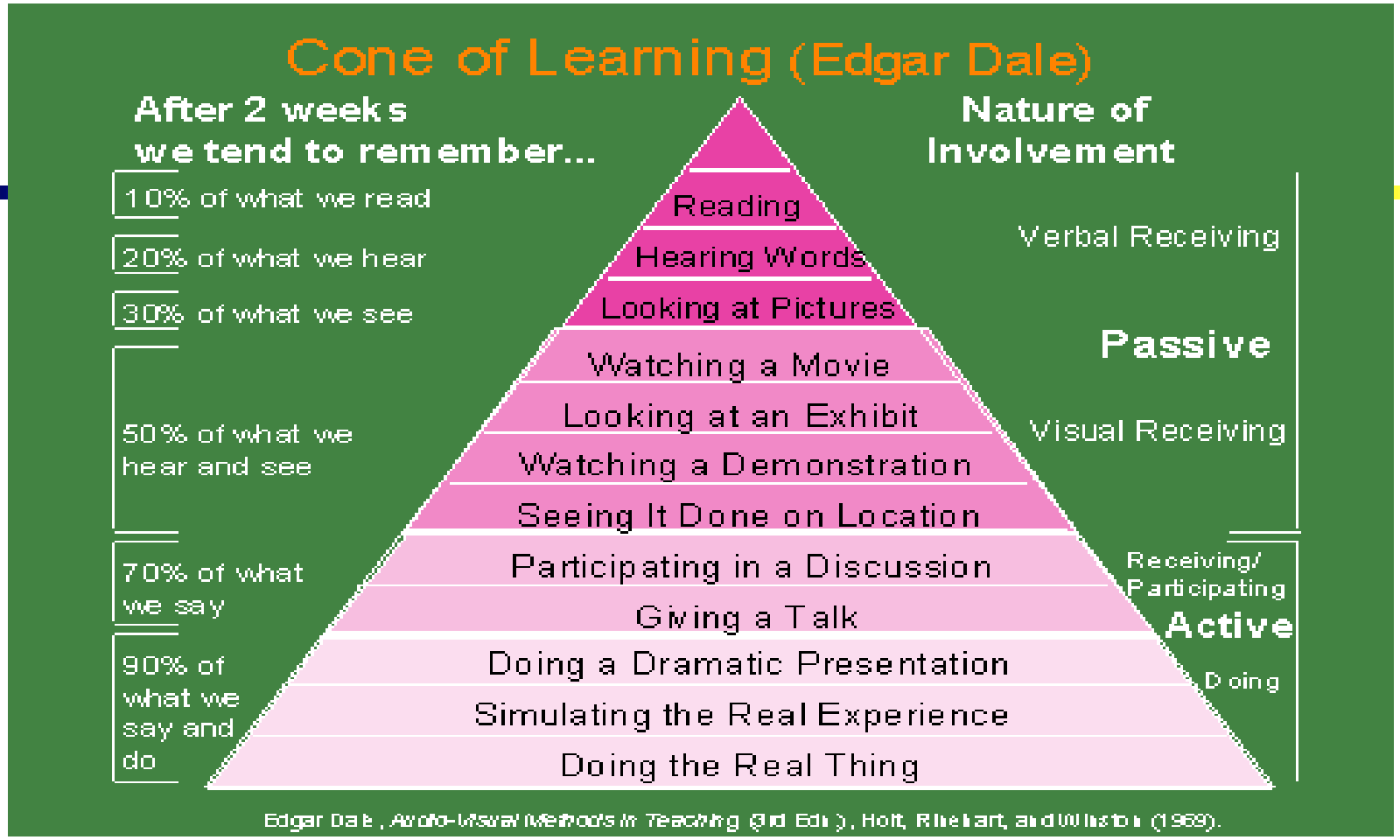
- The presentations, the materials, lab assignments are available on the course portal
  - Pls. access the course portal regularly
- All students should procure the book, as soon as possible

Asking Questions

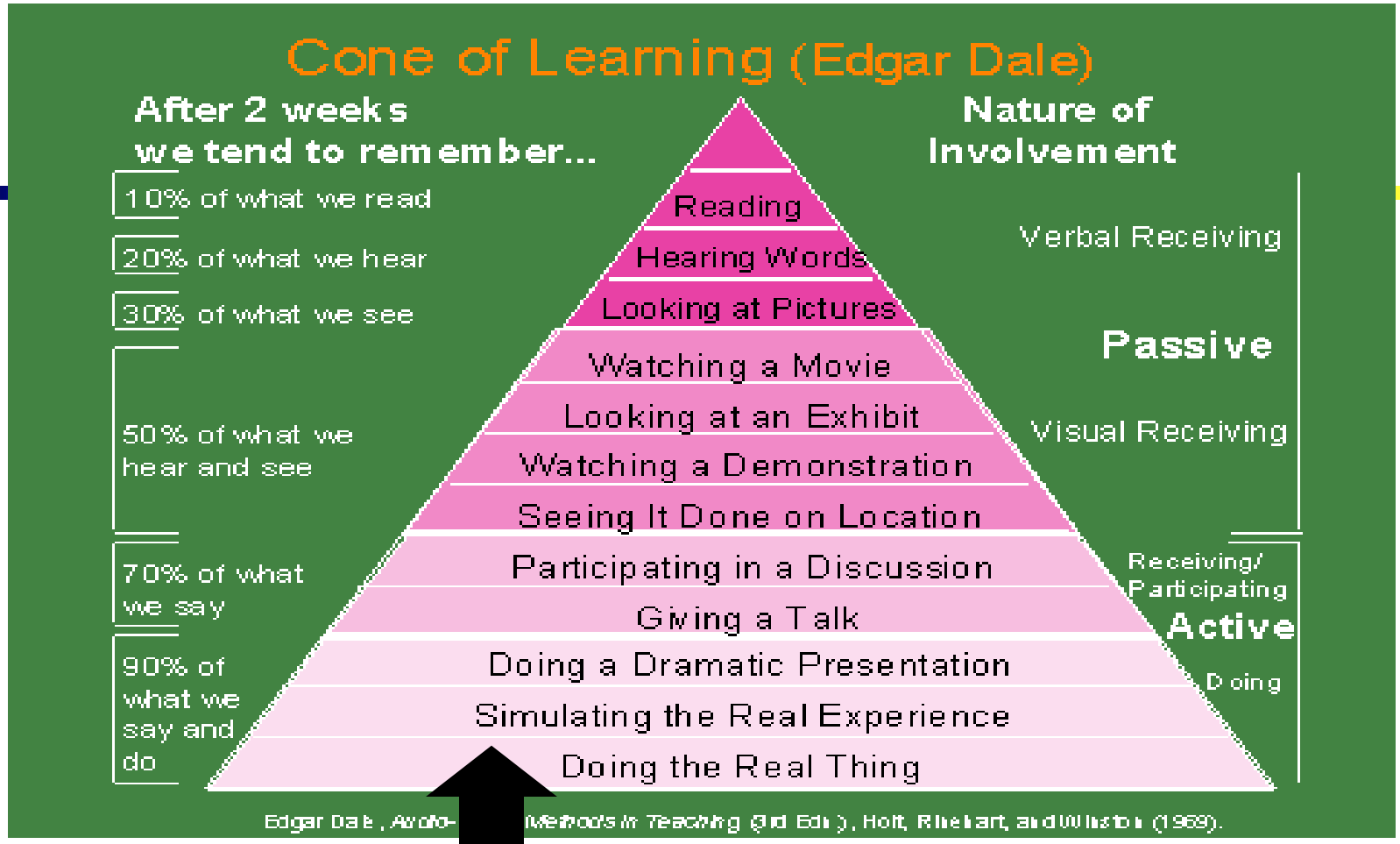
- Theory and lab related questions can be asked through the course portal
  - It is better, because all other students can gain the related knowledge
  - Try to ask as many subject/lab questions as possible.
  - Do not hesitate to ask silly/simple questions regarding lab or theory
    - And try to get the problem resolved as soon as possible

Maximize the benefit from the course

- In the course, we are going to study about an important technology
  - Data scientist or data analyst
- Focus on a thorough understanding of the concepts, not on memorizing
  - Try to understand every sentence of the book



Source: http://www.cals.ncsu.edu/agexed/sae/ppt1/sld012.htm



Bottomline: Do the assignments sincerely because it will facilitate you in **INTERNALIZING** the ideas/techniques you learnt in this course.

Source: http://www.cals.ncsu.edu/agexed/sae/ppt1/sld012.htm

Presentation Outline

- Why Data Analytics (or data mining)?
  - Courses you have Completed
  - Content of human mind
  - Sample data mining problems
- Data mining definition and KDD process
- Multidimensional view of data mining
- Overview of data mining functionalities
- Issues in data mining
- Course outline
- **Summary and History of data mining society**

Summary

- Data mining: Discovering interesting patterns and knowledge from massive amount of data
- A natural evolution of database technology, in great demand, with wide applications
- A KDD process includes data cleaning, data integration, data selection, transformation, data mining, pattern evaluation, and knowledge presentation
- Mining can be performed in a variety of data
- **Data mining functionalities:**
  - **characterization,**
  - **discrimination,**
  - **association,**
  - **classification,**
  - **clustering,**
  - **outlier and trend analysis, etc.**

A Brief History of Data Mining Society

- 1989 IJCAI Workshop on Knowledge Discovery in Databases
  - Knowledge Discovery in Databases (G. Piatetsky-Shapiro and W. Frawley, 1991)
- 1991-1994 Workshops on Knowledge Discovery in Databases
  - Advances in Knowledge Discovery and Data Mining (U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, 1996)
- 1995-1998 International Conferences on Knowledge Discovery in Databases and Data Mining (KDD'95-98)
  - Journal of Data Mining and Knowledge Discovery (1997)
- ACM SIGKDD conferences since 1998 and SIGKDD Explorations
- More conferences on data mining
  - PAKDD (1997), PKDD (1997), SIAM-Data Mining (2001), (IEEE) ICDM (2001), etc.
- ACM Transactions on KDD starting in 2007

Conferences and Journals on Data Mining

- KDD Conferences
  - ACM SIGKDD Int. Conf. on Knowledge Discovery in Databases and Data Mining (KDD)
  - SIAM Data Mining Conf. (SDM)
  - (IEEE) Int. Conf. on Data Mining (ICDM)
  - European Conf. on Machine Learning and Principles and practices of Knowledge Discovery and Data Mining (ECML-PKDD)
  - Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD)
  - Int. Conf. on Web Search and Data Mining (WSDM)
- Other related conferences
  - DB conferences: ACM SIGMOD, VLDB, ICDE, EDBT, ICDT, …
  - Web and IR conferences: WWW, SIGIR, WSDM
  - ML conferences: ICML, NIPS
  - PR conferences: CVPR,
- Journals
  - Data Mining and Knowledge Discovery (DAMI or DMKD)
  - IEEE Trans. On Knowledge and Data Eng. (TKDE)
  - KDD Explorations
  - ACM Trans. on KDD

Where to Find References? DBLP, CiteSeer, Google

- Data mining and KDD (SIGKDD: CDROM)
  - Conferences: ACM-SIGKDD, IEEE-ICDM, SIAM-DM, PKDD, PAKDD, etc.
  - Journal: Data Mining and Knowledge Discovery, KDD Explorations, ACM TKDD
- Database systems (SIGMOD: ACM SIGMOD Anthology—CD ROM)
  - Conferences: ACM-SIGMOD, ACM-PODS, VLDB, IEEE-ICDE, EDBT, ICDT, DASFAA
  - Journals: IEEE-TKDE, ACM-TODS/TOIS, JIIS, J. ACM, VLDB J., Info. Sys., etc.
- AI & Machine Learning
  - Conferences: Machine learning (ML), AAAI, IJCAI, COLT (Learning Theory), CVPR, NIPS, etc.
  - Journals: Machine Learning, Artificial Intelligence, Knowledge and Information Systems, IEEE-PAMI, etc.
- Web and IR
  - Conferences: SIGIR, WWW, CIKM, etc.
  - Journals: WWW: Internet and Web Information Systems,
- Statistics
  - Conferences: Joint Stat. Meeting, etc.
  - Journals: Annals of statistics, etc.
- Visualization
  - Conference proceedings: CHI, ACM-SIGGraph, etc.
  - Journals: IEEE Trans. visualization and computer graphics, etc.

OTHER SLIDES

Data Mining Function: (1) Generalization

- Information integration and data warehouse construction
  - Data cleaning, transformation, integration, and multidimensional data model
- Data cube technology
  - Scalable methods for computing (i.e., materializing) multidimensional aggregates
  - OLAP (online analytical processing)
- Multidimensional concept description: Characterization and discrimination
  - Generalize, summarize, and contrast data characteristics, e.g., dry vs. wet region

Data Mining Function: (2) Association and Correlation Analysis

- Frequent patterns (or frequent itemsets)
  - What items are frequently purchased together in your Walmart?
- Association, correlation vs. causality
  - A typical association rule
    - Diaper → Beer [0.5%, 75%] (support, confidence)
  - Are strongly associated items also strongly correlated?
- How to mine such patterns and rules efficiently in large datasets?
- How to use such patterns for classification, clustering, and other applications?

Data Mining Function: (3) Classification

- Classification and label prediction
  - Construct models (functions) based on some training examples
  - Describe and distinguish classes or concepts for future prediction
    - E.g., classify countries based on (climate), or classify cars based on (gas mileage)
  - Predict some unknown class labels
- Typical methods
  - Decision trees, naïve Bayesian classification, support vector machines, neural networks, rule-based classification, pattern-based classification, logistic regression, …
- Typical applications:
  - Credit card fraud detection, direct marketing, classifying stars, diseases, web-pages, …

Data Mining Function: (4) Cluster Analysis

- Unsupervised learning (i.e., Class label is unknown)
- Group data to form new categories (i.e., clusters), e.g., cluster houses to find distribution patterns
- Principle: Maximizing intra-class similarity & minimizing interclass similarity
- Many methods and applications

Data Mining Function: (5) Outlier Analysis

- Outlier analysis
  - Outlier: A data object that does not comply with the general behavior of the data
  - Noise or exception? — One person's garbage could be another person's treasure
  - Methods: by product of clustering or regression analysis, …
  - Useful in fraud detection, rare events analysis

Time and Ordering: Sequential Pattern, Trend and Evolution Analysis

- Sequence, trend and evolution analysis
  - Trend, time-series, and deviation analysis: e.g., regression and value prediction
  - Sequential pattern mining
    - e.g., first buy digital camera, then buy large SD memory cards
  - Periodicity analysis
  - Motifs and biological sequence analysis
    - Approximate and consecutive motifs
  - Similarity-based analysis
- Mining data streams
  - Ordered, time-varying, potentially infinite, data streams

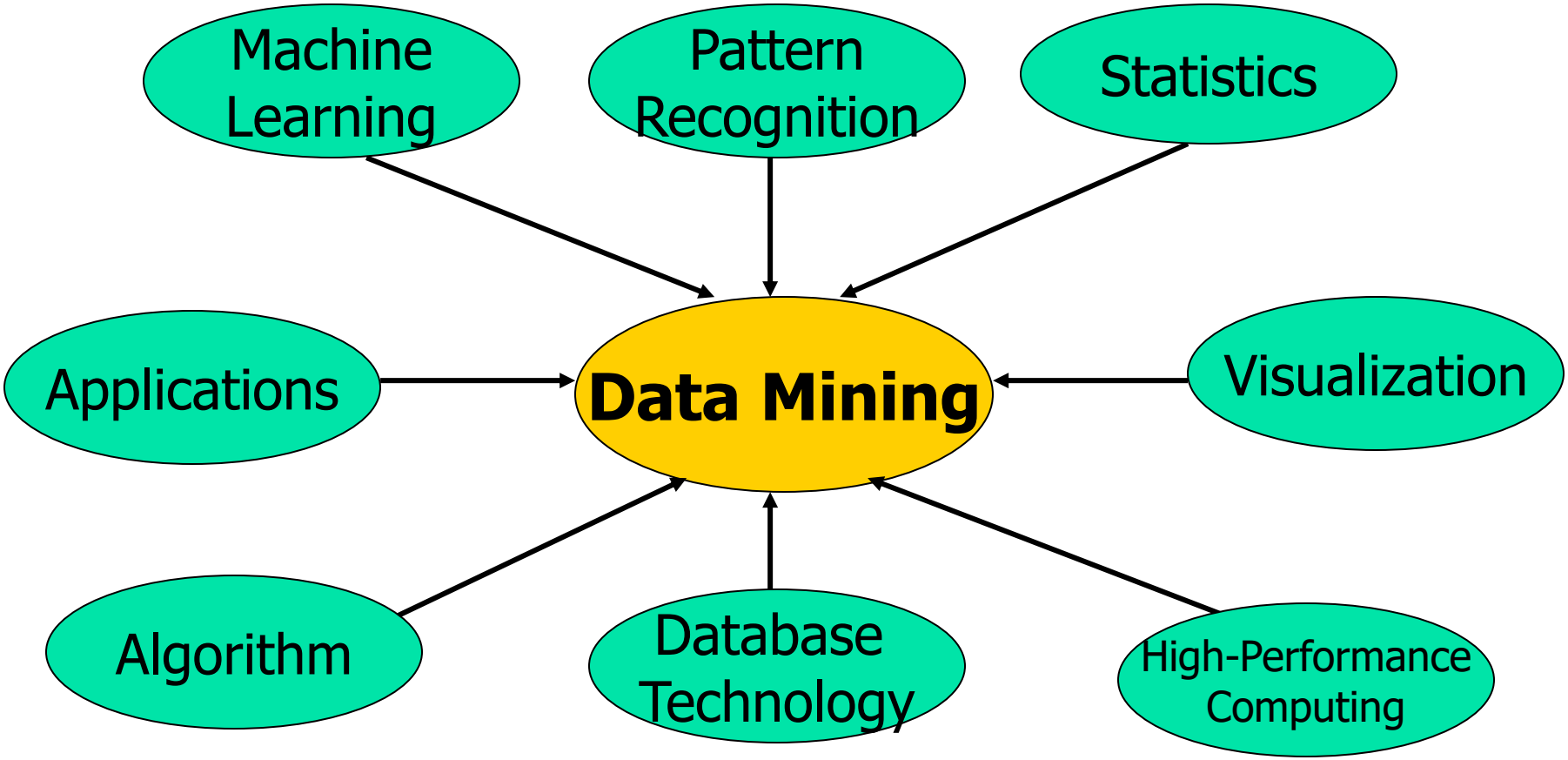
Structure and Network Analysis

- Graph mining
  - Finding frequent subgraphs (e.g., chemical compounds), trees (XML), substructures (web fragments)
- Information network analysis
  - Social networks: actors (objects, nodes) and relationships (edges)
    - e.g., author networks in CS, terrorist networks
  - Multiple heterogeneous networks
    - A person could be multiple information networks: friends, family, classmates, …
  - Links carry a lot of semantic information: Link mining
- Web mining
  - Web is a big information network: from PageRank to Google
  - Analysis of Web information networks
    - Web community discovery, opinion mining, usage mining, …

Evaluation of Knowledge

- Are all mined knowledge interesting?
  - One can mine tremendous amount of "patterns" and knowledge
  - Some may fit only certain dimension space (time, location, …)
  - Some may not be representative, may be transient, …
- Evaluation of mined knowledge → directly mine only interesting knowledge?
  - Descriptive vs. predictive
  - Coverage
  - Typicality vs. novelty
  - Accuracy
  - Timeliness
  - …

Data Mining: Confluence of Multiple Disciplines



Why Confluence of Multiple Disciplines?

- Tremendous amount of data
  - Algorithms must be highly scalable to handle such as tera-bytes of data
- High-dimensionality of data
  - Micro-array may have tens of thousands of dimensions
- High complexity of data
  - Data streams and sensor data
  - Time-series data, temporal data, sequence data
  - Structure data, graphs, social networks and multi-linked data
  - Heterogeneous databases and legacy databases
  - Spatial, spatiotemporal, multimedia, text and Web data
  - Software programs, scientific simulations
- New and sophisticated applications

Applications of Data Mining

- Web page analysis: from web page classification, clustering to PageRank & HITS algorithms
- Collaborative analysis & recommender systems
- Basket data analysis to targeted marketing
- Biological and medical data analysis: classification, cluster analysis (microarray data analysis), biological sequence analysis, biological network analysis
- Data mining and software engineering (e.g., IEEE Computer, Aug. 2009 issue)
- From major dedicated data mining systems/tools (e.g., SAS, MS SQL-Server Analysis Manager, Oracle Data Mining Tools) to invisible data mining

Introduction: Presentation Outline

- Why Data Analytics (or data mining)?
  - Courses you have Completed
  - Content of human mind
  - Sample data mining problems
- Data mining definition and KDD process
- Multidimensional view of data mining
- Overview of data mining functionalities
- **Issues in data mining**
- Course outline
- Summary and History of data mining society

Major Issues in Data Mining (1)

- Mining Methodology
  - Mining various and new kinds of knowledge
  - Mining knowledge in multi-dimensional space
  - Data mining: An interdisciplinary effort
  - Boosting the power of discovery in a networked environment
  - Handling noise, uncertainty, and incompleteness of data
  - Pattern evaluation and pattern- or constraint-guided mining
- User Interaction
  - Interactive mining
  - Incorporation of background knowledge
  - Presentation and visualization of data mining results

Major Issues in Data Mining (2)

- Efficiency and Scalability
  - Efficiency and scalability of data mining algorithms
  - Parallel, distributed, stream, and incremental mining methods
- Diversity of data types
  - Handling complex types of data
  - Mining dynamic, networked, and global data repositories
- Data mining and society
  - Social impacts of data mining
  - Privacy-preserving data mining
  - Invisible data mining

Introduction: Presentation Outline

- Why Data Analytics (or data mining)?
  - Courses you have Completed
  - Content of human mind
  - Sample data mining problems
- Data mining definition and KDD process
- Multidimensional view of data mining
- Overview of data mining functionalities
- Issues in data mining
- **Course outline**
- Summary and History of data mining society

Data Mining Concepts and Data Statistics

P. Krishna Reddy
IIIT Hyderabad

Documents Uploaded

- Data, Information, Knowledge, and Wisdom by Gene Bellinger, Durval Castro, Anthony Mills
- From Data to Wisdom: A Note by Russell Ackoff
- Data Science— A Systematic Treatment by M. TAMER ÖZSU, Communications of ACM, 2023
- Theoretical frameworks for data mining, Heikki Mannila, 2000.
- Data Science: A Comprehensive Overview, LONGBING CAO, ACM Computing Surveys, 2017

Reference

- Chapter 1 and Chapter 2 of the text book.

Detailed Syllabus

- **Unit1: Introduction, data and data preprocessing,** data summarization through characterization, discrimination and data cube techniques (9 hours)
- Unit 2: Concepts and algorithms for mining patterns and associations (10 hours)
- Unit 3: Concepts and algorithms related to classification and regression (10 hours)
- Unit 4: Concepts and algorithms for clustering the data (10 hours)
- Unit 5: Outlier analysis and future trends. (3 hours)

Presentation Outline

- **Data mining functionalities**
- Research Issues in Data mining
- Sources of data
- Data Objects and Attribute Types
- About Big Data
- Basic Statistical Descriptions of Data
- Data Visualization
- Case study
- Summary

Data Mining Functionalities

- Summarization
- Pattern mining, Association mining, correlation
- Classification
- Clustering
- Outlier analysis
- Sequential, trend and evolution analysis
- Structure and network analysis

Data Mining Function (1): Summarization

- Input: Large data
- Output: summarize/characterize an interested set of data and compare it with the contrasting sets at some high levels.
  - Data characterization: Summarizing the data of the class under study.
  - Data discrimination: Comparison of target class with one or a set of target class.
- Multidimensional concept description: Characterization and discrimination
  - Generalize, summarize, and contrast data characteristics, e.g., dry vs. wet region
- Output can be represented with pie charts, bar charts, curves, multidimensional data cubes, and multidimensional tables, including crosstabs

Concept description: Characterization and discrimination

- Generalize, summarize, and contrast data characteristics, e.g., dry vs. wet regions.
- 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

Tabulation of Data

- Tabulation facilitates the presentation of large information into concise way under different titles and sub-titles.

Table 1. Students

Class	Rural				Urban			
	Male		Female		Male		Female	
	Hostellers	Day Scholars	Hostellers	Day scholars	Hostellers	Day Scholars	Hostellers	Day scholars
Intermediate								
Graduate								
Post Graduate								

Summarization techniques

- Statistical measures
- Attribute oriented induction
- Data cube based OLAP methods
  - Multidimensional summarization

Data Mining Function: (2) Association and Correlation Analysis

- Frequent patterns (or frequent itemsets)
  - What items are frequently purchased together in your Walmart?
- Association, correlation vs. causality
  - A typical association rule
    - Diaper → Beer [0.5%, 75%] (support, confidence)
  - Are strongly associated items also strongly correlated?
- How to mine such patterns and rules efficiently in large datasets?
- How to use such patterns for classification, clustering, and other applications?

Approaches to mine Association Rules

- Apriori approach
- FP Tree approach
- Hash-based itemset counting: A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
- Transaction reduction: A transaction that does not contain any frequent k-itemset is useless in subsequent scans
- Partitioning: Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
- Sampling: mining on a subset of given data, lower support threshold + a method to determine the completeness
- Dynamic itemset counting: add new candidate itemsets only when all of their subsets are estimated to be frequent

Data Mining Function: (3) Classification

- Classification and label prediction
  - Construct models (functions) based on some training examples
  - Describe and distinguish classes or concepts for future prediction
    - E.g., classify countries based on (climate), or classify cars based on (gas mileage)
  - Predict some unknown class labels
- Typical applications:
  - Credit card fraud detection, direct marketing, classifying stars, diseases, web-pages, …

age(X, "youth") AND income(X, "high") ⟶ *class(X, "A")*

age(X, "youth") AND income(X, "low") ⟶ *class(X, "B")*

age(X, "middle_aged") ⟶ *class(X, "C")*

age(X, "senior") ⟶ *class(X, "C")*

(a)

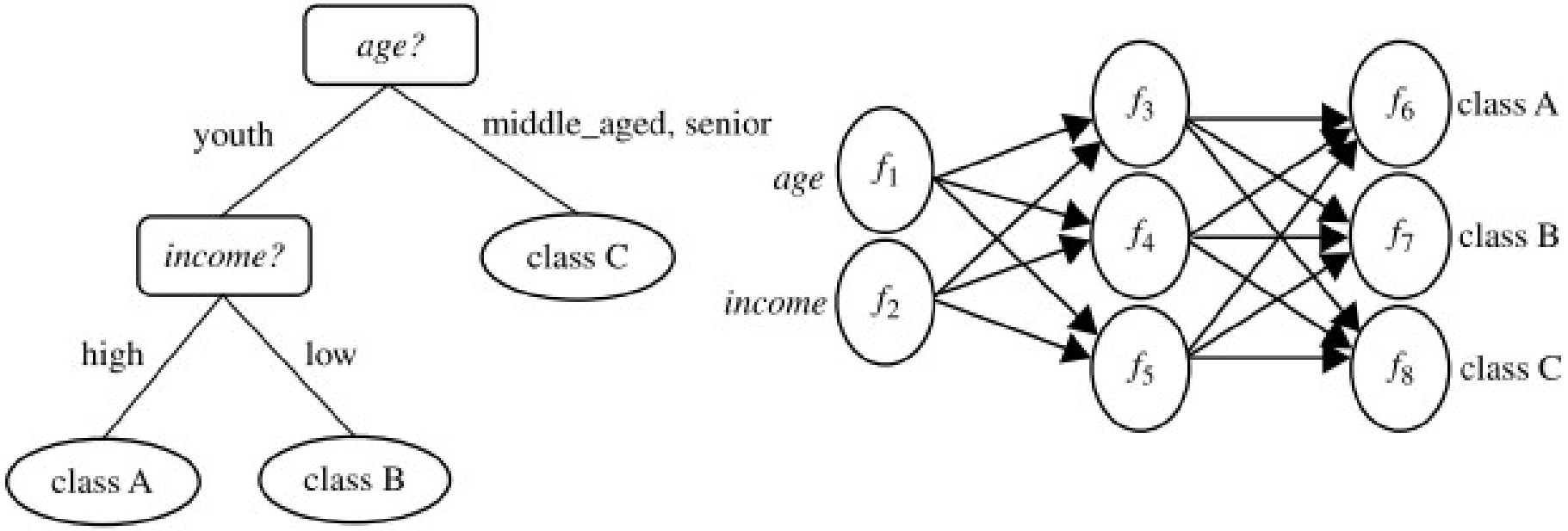


FIGURE 1.2

A classification model can be represented in various forms: (a) IF-THEN rules, (b) a decision tree, or (c) a neural network.

Approaches to classification

- Decision tree
- Bayesian Classification
- Neural networks
- Association based classification
- k-nearest neighbor classifier
- Ensemble classification

Data Mining Function: (4) Cluster Analysis

- Unsupervised learning (i.e., Class label is unknown)
- Group data to form new categories (i.e., clusters), e.g., cluster houses to find distribution patterns
- Principle: Maximizing intra-class similarity & minimizing interclass similarity
- Many methods and applications

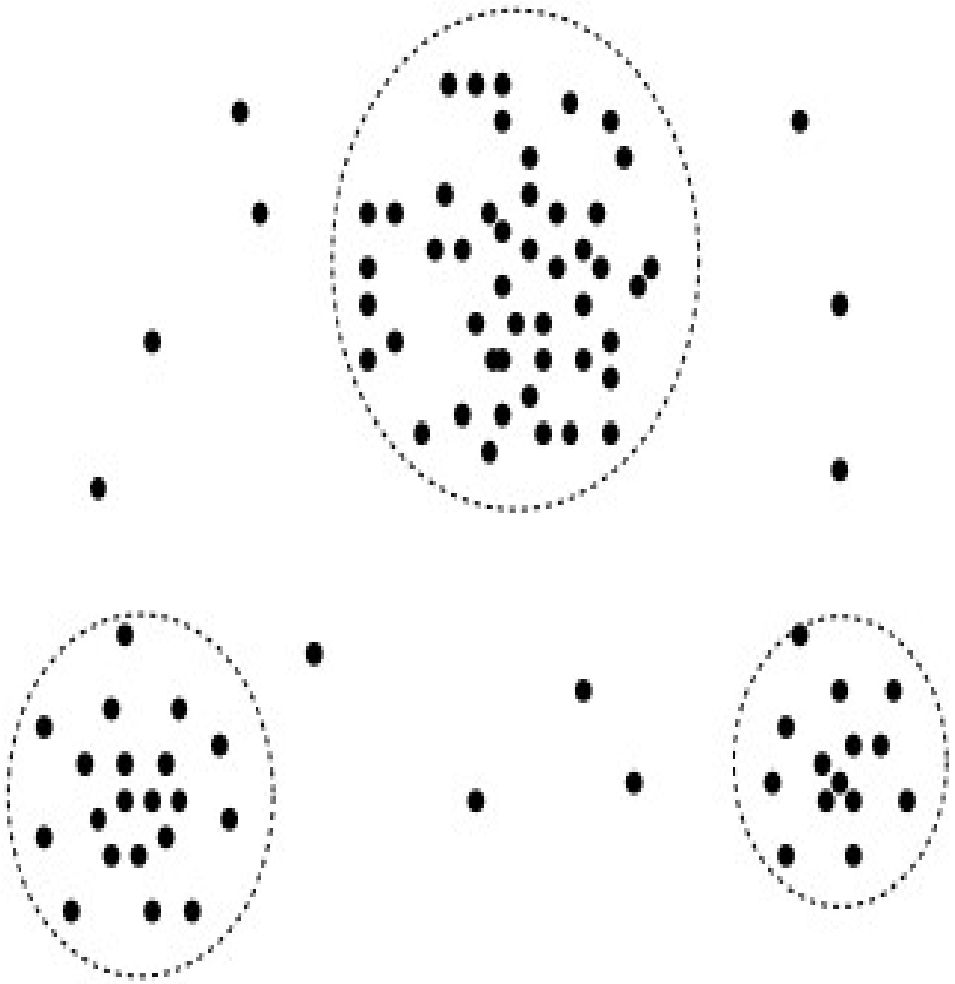


FIGURE 1.3

A 2-D plot of customer data with respect to customer locations in a city, showing three data clusters.

Major Clustering Approaches

- Partitioning algorithms: Construct various partitions and then evaluate them by some criterion
- Hierarchy algorithms: Create a hierarchical decomposition of the set of data (or objects) using some criterion
- Density-based: based on connectivity and density functions
- Grid-based: based on a multiple-level granularity structure
- Model-based: A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other

Data Mining Function: (5) Outlier Analysis

- Outlier analysis
  - Outlier: A data object that does not comply with the general behavior of the data
  - Noise or exception? — One person's garbage could be another person's treasure
  - Methods: by product of clustering or regression analysis, …
  - Useful in fraud detection, rare events analysis

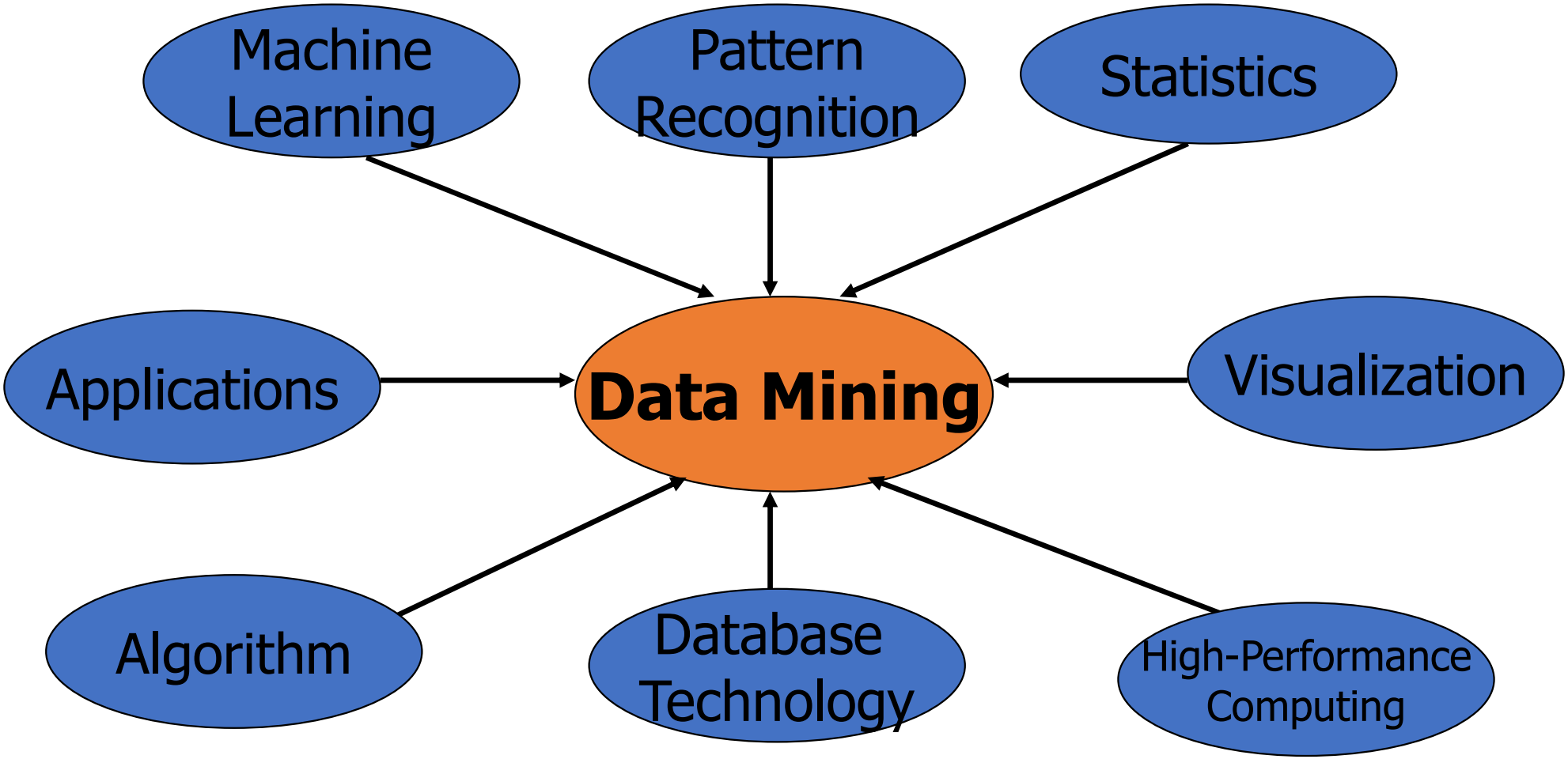
Time and Ordering: Sequential Pattern, Trend and Evolution Analysis

- Sequence, trend and evolution analysis
  - Trend, time-series, and deviation analysis: e.g., regression and value prediction
  - Sequential pattern mining
    - e.g., first buy digital camera, then buy large SD memory cards
  - Periodicity analysis
  - Motifs and biological sequence analysis
    - Approximate and consecutive motifs
  - Similarity-based analysis
- Mining data streams
  - Ordered, time-varying, potentially infinite, data streams

Structure and Network Analysis

- Graph mining
  - Finding frequent subgraphs (e.g., chemical compounds), trees (XML), substructures (web fragments)
- Information network analysis
  - Social networks: actors (objects, nodes) and relationships (edges)
    - e.g., author networks in CS, terrorist networks
  - Multiple heterogeneous networks
    - A person could be multiple information networks: friends, family, classmates, …
  - Links carry a lot of semantic information: Link mining
- Web mining
  - Web is a big information network: from PageRank to Google
  - Analysis of Web information networks
    - Web community discovery, opinion mining, usage mining, …

Data Mining: Confluence of Multiple Disciplines



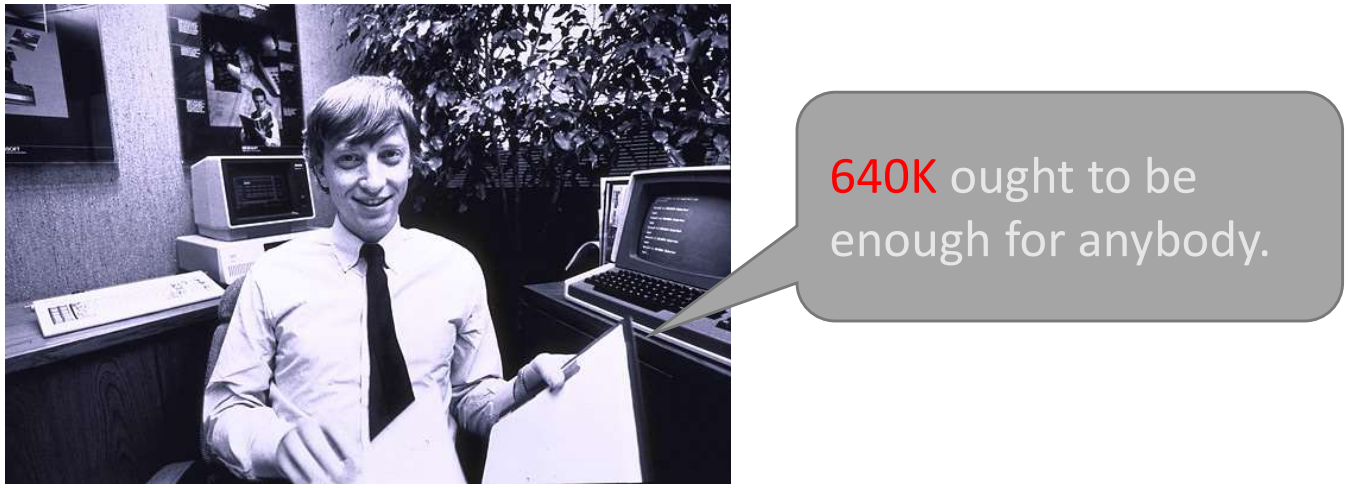
Presentation Outline

- Data mining functionalities
- **Research Issues in Data mining**
- Sources of data
- Data Objects and Attribute Types
- About Big Data
- Basic Statistical Descriptions of Data
- Data Visualization
- Case study
- Summary

About Big Data

How much data?

- Google processes 20 PB a day (2008)
- Facebook has 2.5 PB of user data + 15 TB/day (4/2009)
- eBay has 6.5 PB of user data + 50 TB/day (5/2009)
- CERN's Large Hydron Collider (LHC) generates 15 PB a year

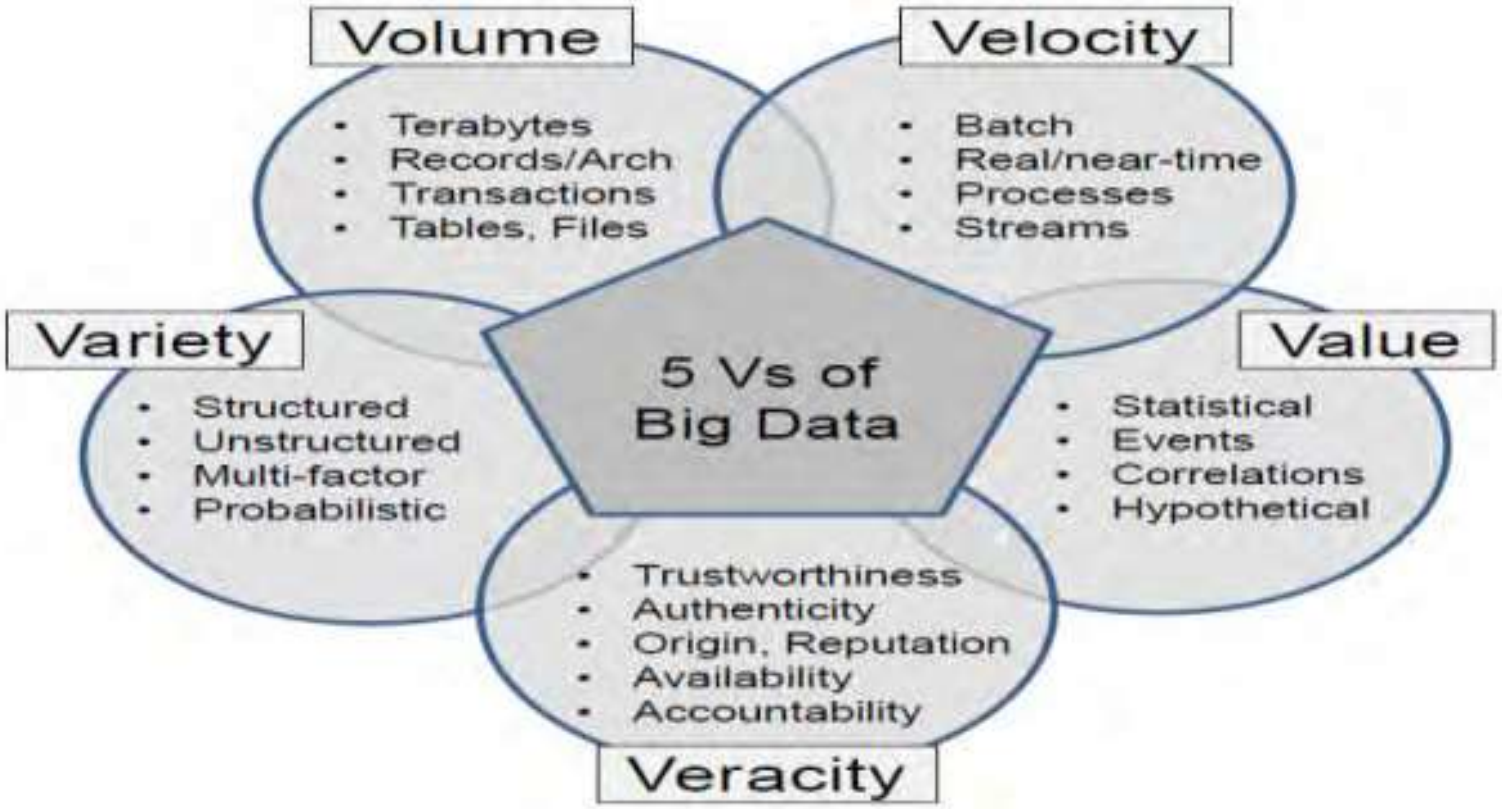


Big Data Definition

- No single standard definition…

"***Big Data***" is data whose scale, diversity, and complexity require new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it…

Characteristics of Big data



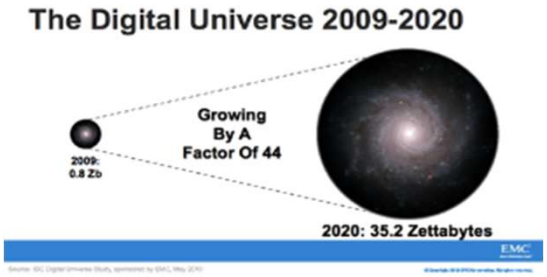
Volume

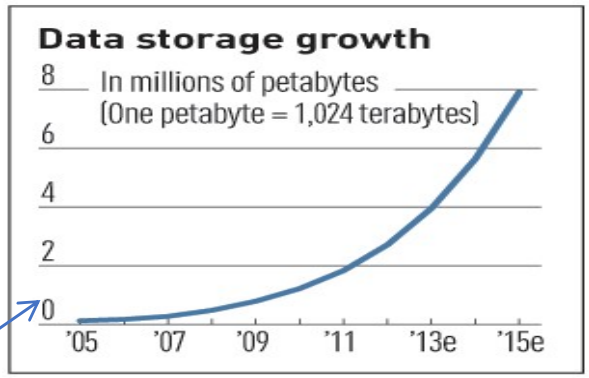
- The costs of computing, storage, and connectivity resources are plunging, and new technologies like scanners, smartphones, ubiquitous video, and other data-collectors mean we are awash in volumes of data that dwarf what was available even five to 10 years ago.

- We capture every mouse click, phone call, text message, Web search, transaction, and more. As the volume of data grows, we can learn more – but only if we uncover meaningful relationships and patterns.

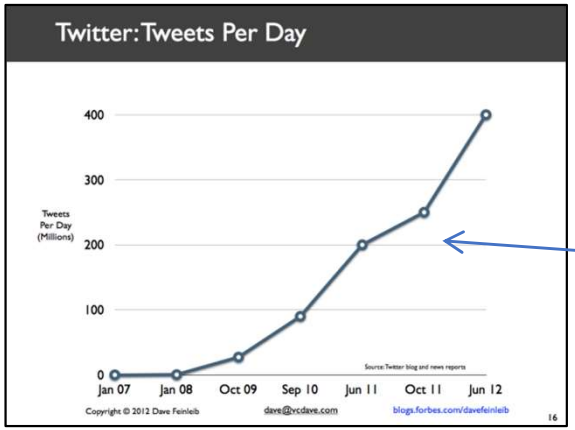
Volume..

- **Data Volume**
  - 44x increase from 2009 2020
  - From 0.8 zettabytes to 35zb
- Data volume is increasing exponentially





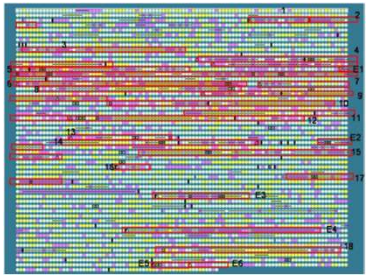
the amount of data stored by the average company today

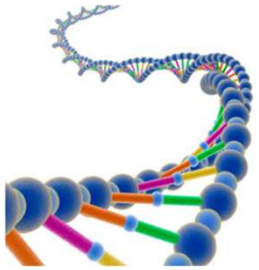


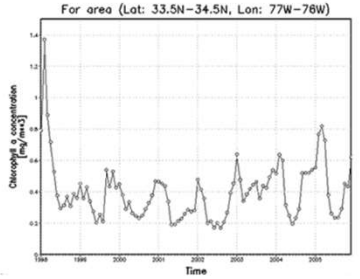
Exponential increase in collected/generated data

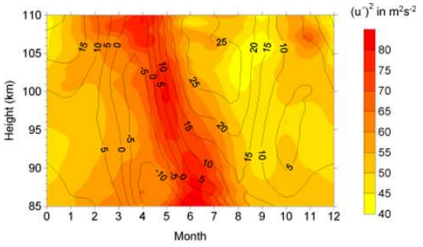
Variety

- Various formats, types, and structures
- Text, numerical, images, audio, video, sequences, time series, social media data, multi-dim arrays, etc…
- Static data vs. streaming data
- A single application can be generating/collecting many types of data









To extract knowledge➔ all these types of data need to linked together

Velocity

- Data is begin generated fast and need to be processed fast
- Online Data Analytics
- Late decisions ➔ missing opportunities

- **Examples**
  - **E-Promotions:** Based on your current location, your purchase history, what you like ➔ send promotions right now for store next to you
  - **Healthcare monitoring:** sensors monitoring your activities and body ➔ any abnormal measurements require immediate reaction

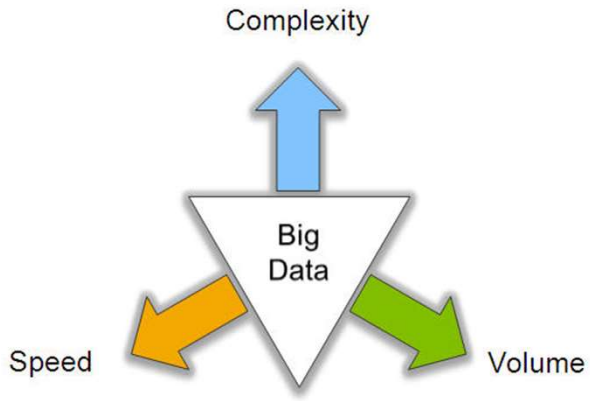
Veracity

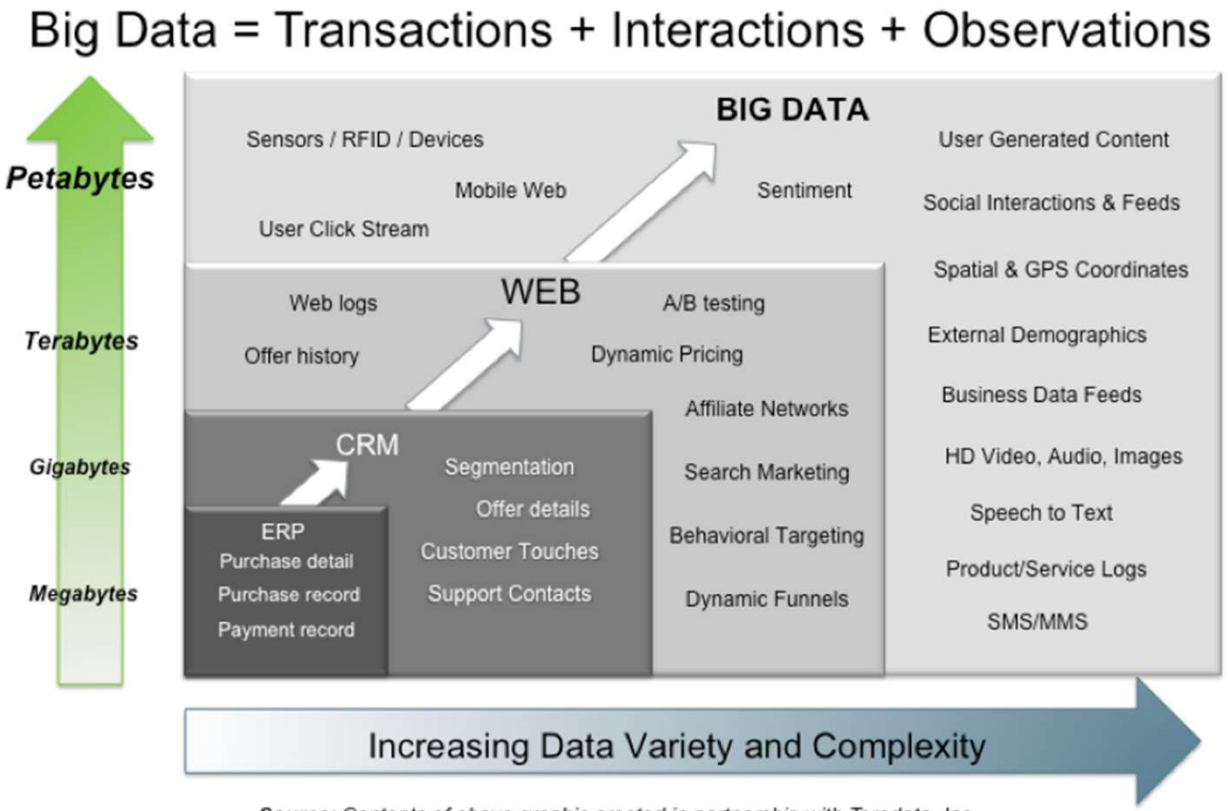
- Veracity indicates the inherent trustworthiness of data.
- The uncertainty about the consistency or completeness of data and other ambiguities can become major obstacles.
- As a result, basic principles as data quality, data cleansing, master data management, and data governance remain critical disciplines when working with Big Data.

Value

- Access to big data is no good, unless it turned into a good value.
- Return on investment

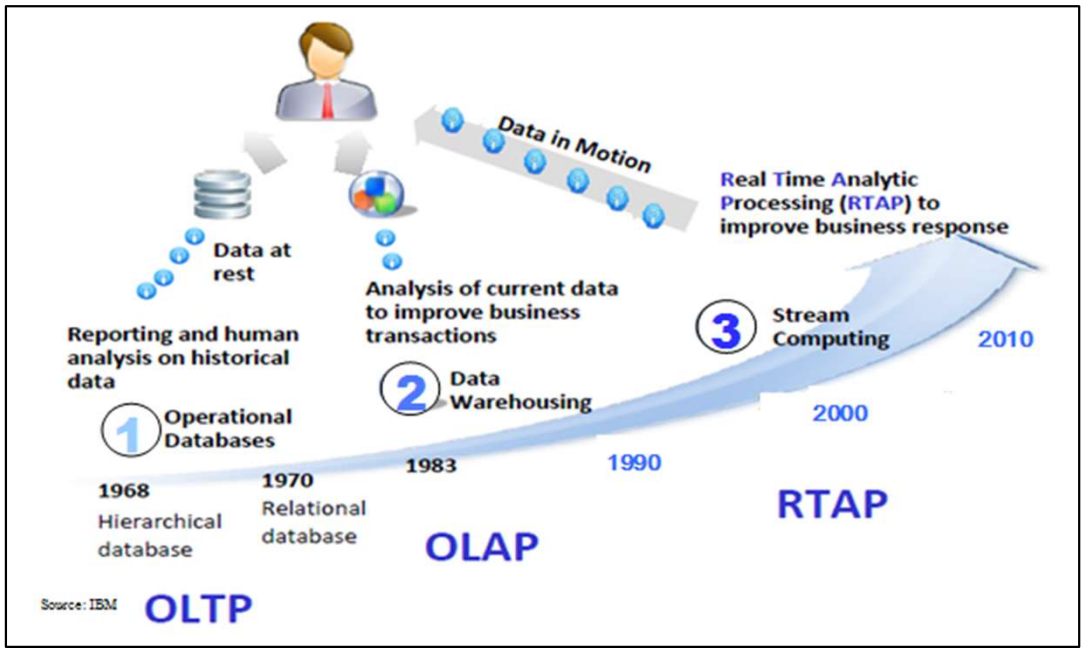
Big Data: 3V's





***Source**: Contents of above graphic created in partnership with Teradata, Inc.*

Harnessing Big Data



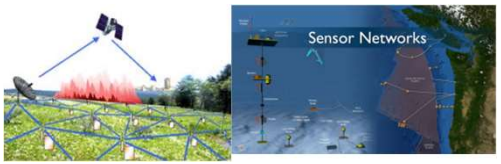
- **OLTP:** Online Transaction Processing (DBMSs)
- **OLAP:** Online Analytical Processing (Data Warehousing)
- **RTAP:** Real-Time Analytics Processing (Big Data Architecture & technology)

Who's Generating Big Data

Social media and networks
(all of us are generating data)

Scientific instruments
(collecting all sorts of data)

Mobile devices
(tracking all objects all the time)



Sensor technology and networks
(measuring all kinds of data)

- The progress and innovation is no longer hindered by the ability to collect data
- But, by the ability to manage, analyze, summarize, visualize, and discover knowledge from the collected data in a timely manner and in a scalable fashion

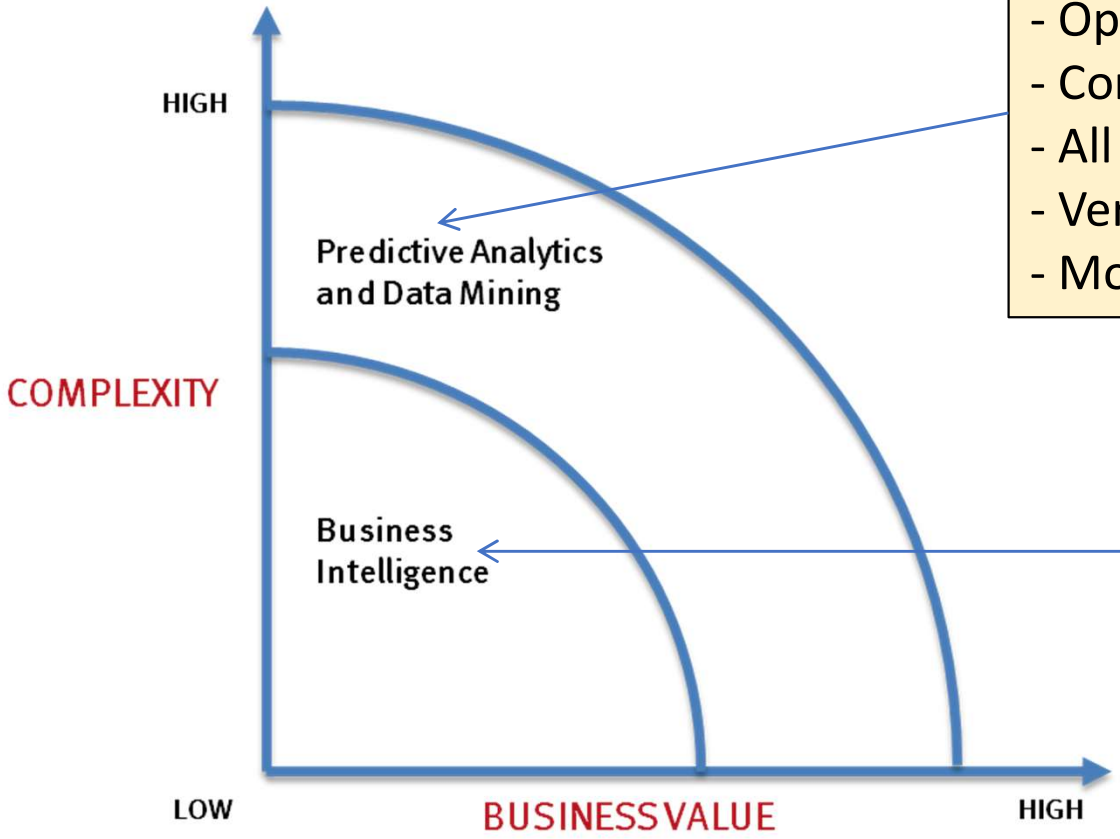
The Model Has Changed…

- **The Model of Generating/Consuming Data has Changed**

Old Model: Few companies are generating data, all others are consuming data

New Model: all of us are generating data, and all of us are consuming data

What's driving Big Data

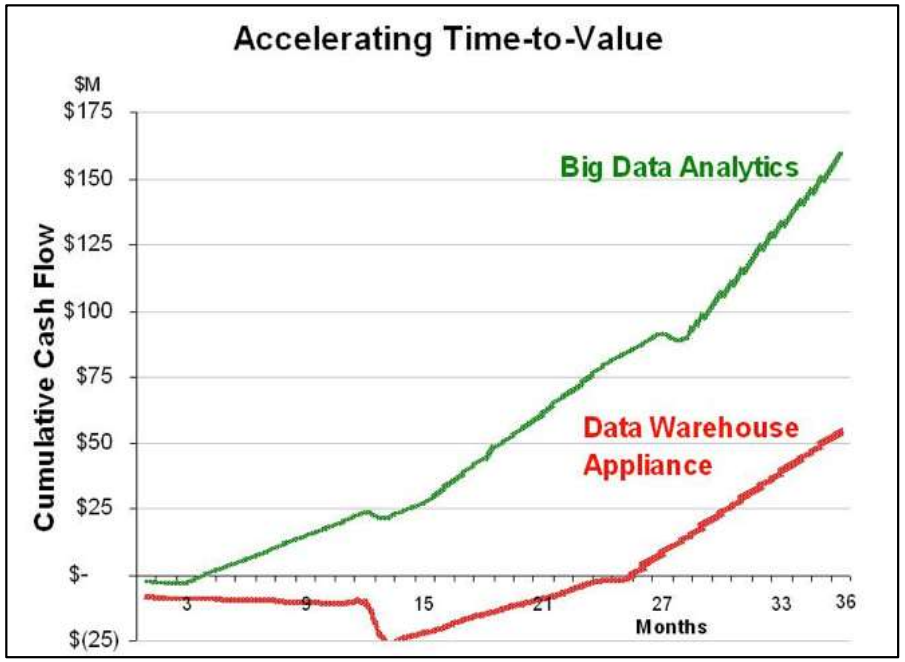


- Optimizations and predictive analytics
- Complex statistical analysis
- All types of data, and many sources
- Very large datasets
- More of a real-time

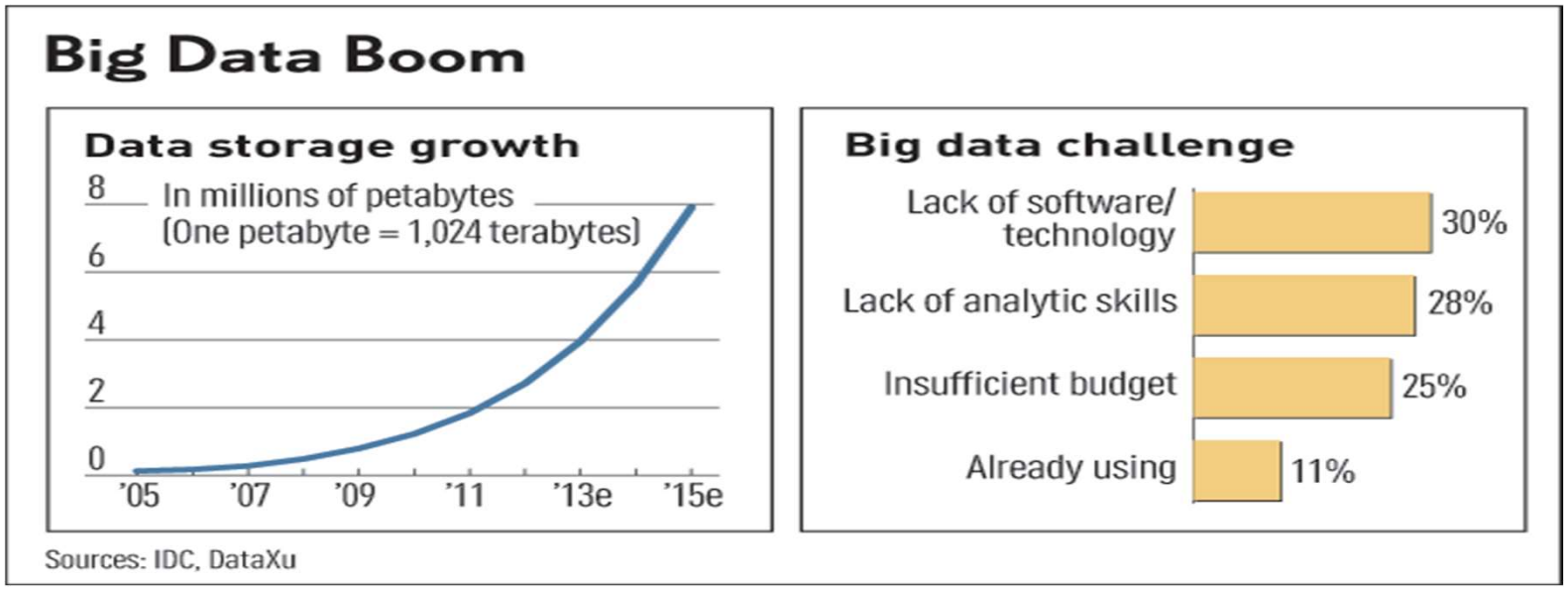
- Ad-hoc querying and reporting
- Data mining techniques
- Structured data, typical sources
- Small to mid-size datasets

Value of Big Data Analytics

- Big data is more real-time in nature than traditional DW applications
- Traditional DW architectures (e.g. Exadata, Teradata) are not well-suited for big data apps
- Shared nothing, massively parallel processing, scale out architectures are well-suited for big data apps

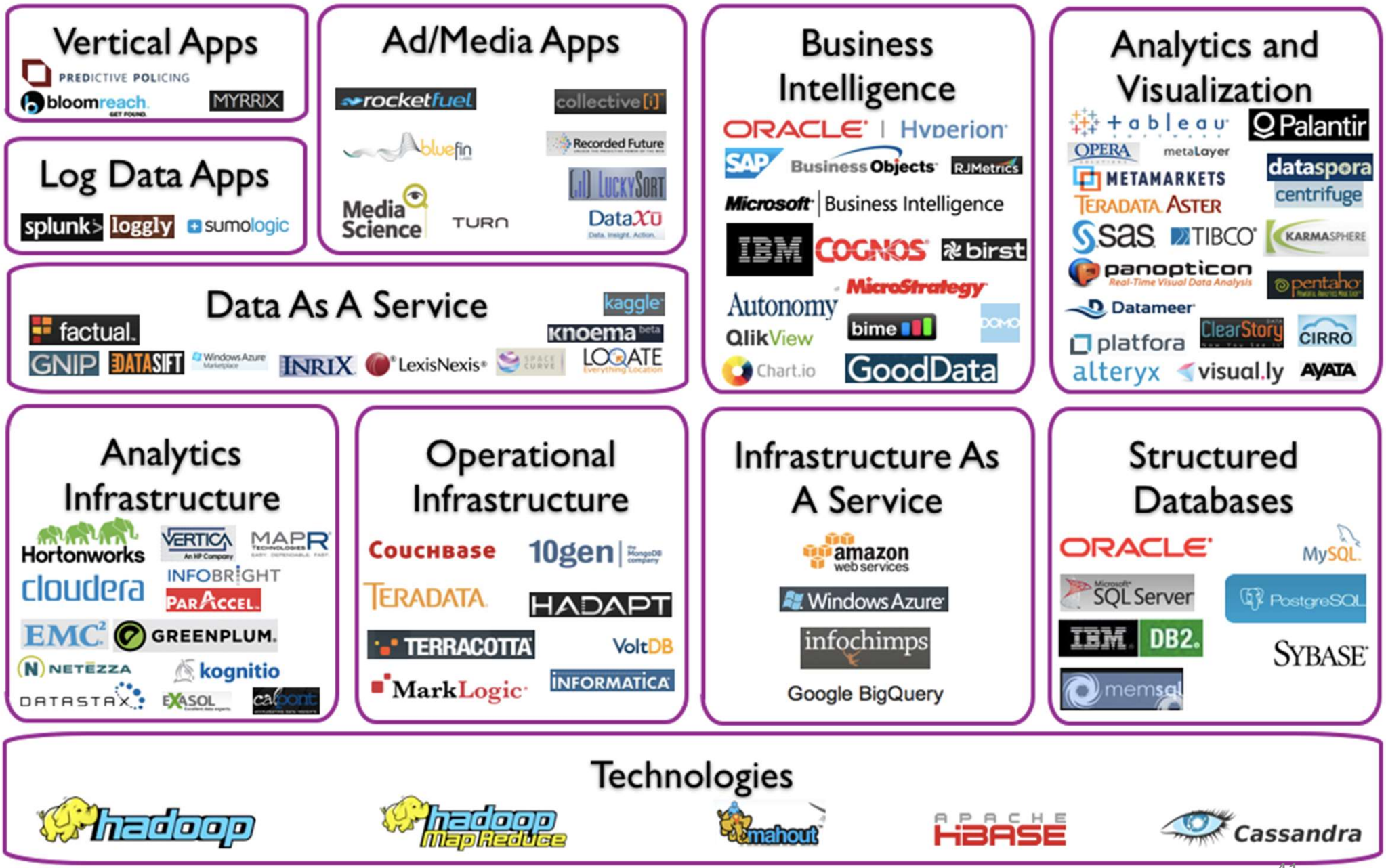


Challenges in Handling Big Data



- **The Bottleneck is in technology**
  - New architecture, algorithms, techniques are needed
- **Also in technical skills**
  - Experts in using the new technology and dealing with big data

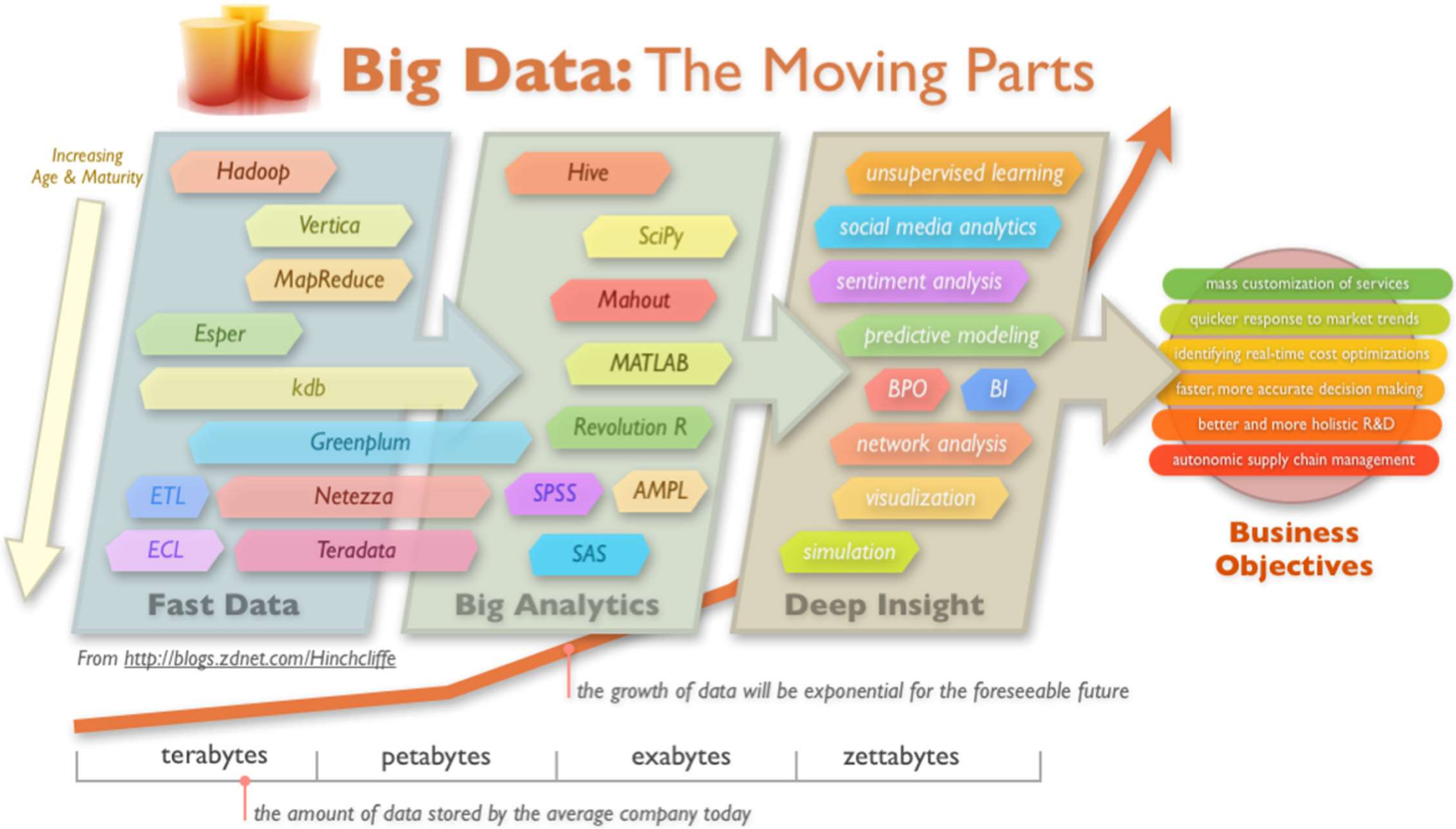
Big Data Landscape



dave@vcdave.com

blogs.forbes.com/davefeinleib

Big Data Technology



About Big Data: Conclusion

- Beginning of big data economy.
- Big data and data science will bring a major social change.
- Companies which fail to exploit big data runs the risk of left behind.
- Should be exploited for sustainable development and equitable society

Evaluation of Knowledge

- Are all mined knowledge interesting?
  - One can mine tremendous amount of "patterns" and knowledge
  - Some may fit only certain dimension space (time, location, …)
  - Some may not be representative, may be transient, …
- Evaluation of mined knowledge → directly mine only interesting knowledge?
  - Descriptive vs. predictive
  - Coverage
  - Typicality vs. novelty
  - Accuracy
  - Timeliness
  - …

Applications of Data Mining

- Web page analysis: from web page classification, clustering to PageRank & HITS algorithms
- Collaborative analysis & recommender systems
- Basket data analysis to targeted marketing
- Biological and medical data analysis: classification, cluster analysis (microarray data analysis), biological sequence analysis, biological network analysis
- Data mining and software engineering (e.g., IEEE Computer, Aug. 2009 issue)
- From major dedicated data mining systems/tools (e.g., SAS, MS SQL-Server Analysis Manager, Oracle Data Mining Tools) to invisible data mining

Why Confluence of Multiple Disciplines?

- Tremendous amount of data
  - Algorithms must be highly scalable to handle such as tera-bytes of data
- High-dimensionality of data
  - Micro-array may have tens of thousands of dimensions
- High complexity of data
  - Data streams and sensor data
  - Time-series data, temporal data, sequence data
  - Structure data, graphs, social networks and multi-linked data
  - Heterogeneous databases and legacy databases
  - Spatial, spatiotemporal, multimedia, text and Web data
  - Software programs, scientific simulations
- New and sophisticated applications

Major Issues in Data Mining (1)

- Mining Methodology
  - Mining various and new kinds of knowledge
  - Mining knowledge in multi-dimensional space
  - Data mining: An interdisciplinary effort
  - Boosting the power of discovery in a networked environment
  - Handling noise, uncertainty, and incompleteness of data
  - Pattern evaluation and pattern- or constraint-guided mining
- User Interaction
  - Interactive mining
  - Incorporation of background knowledge
  - Presentation and visualization of data mining results

Major Issues in Data Mining (2)

- Efficiency and Scalability
  - Efficiency and scalability of data mining algorithms
  - Parallel, distributed, stream, and incremental mining methods
- Diversity of data types
  - Handling complex types of data
  - Mining dynamic, networked, and global data repositories
- Data mining and society
  - Social impacts of data mining
  - Privacy-preserving data mining
  - Invisible data mining

Important Characteristics of Structured Data

- Dimensionality
  - Curse of dimensionality
- Sparsity
  - Only presence counts
  - Bag-of-words
- Resolution
  - Patterns depend on the scale
- Distribution
  - Centrality and dispersion

Multi-Dimensional View of Data Mining

- **Data to be mined**
  - Database data (extended-relational, object-oriented, heterogeneous, legacy), data warehouse, transactional data, stream, spatiotemporal, time-series, sequence, text and web, multi-media, graphs & social and information networks
- **Knowledge to be mined (or: Data mining functions)**
  - summarization association, classification, clustering, trend/deviation, outlier analysis, etc.
  - Descriptive vs. predictive data mining
  - Multiple/integrated functions and mining at multiple levels
- **Techniques utilized**
  - Data-intensive, data warehouse (OLAP), machine learning, statistics, pattern recognition, visualization, high-performance, etc.
- **Applications adapted**
  - Retail, telecommunication, banking, fraud analysis, bio-data mining, stock market analysis, text mining, Web mining, etc.

Research Issues in Data Mining

- Data mining (knowledge discovery in databases):
  - Extraction of interesting (non-trivial, implicit, previously unknown and potentially useful) information or patterns from data in large databases
- So far, data mining means
  - Summarization, Association rules, clustering, classification
- Research Issues
  - Finding new patterns
    - Market basket data, Complex data, stream data
  - Improving the performance of existing algorithms
    - Scalable algorithms
      - Data, features or dimensions
    - Complex data
  - Visualizing the patterns

Data Mining: On What Kinds of Data?

- Database-oriented data sets and applications
  - Relational database, data warehouse, transactional database
- Advanced data sets and advanced applications
  - Data streams and sensor data
  - Time-series data, temporal data, sequence data (incl. bio-sequences)
  - Structure data, graphs, social networks and multi-linked data
  - Object-relational databases
  - Heterogeneous databases and legacy databases
  - Spatial data and spatiotemporal data
  - Multimedia database
  - Text databases
  - The World-Wide Web

Presentation Outline

- Data mining functionalities
- Research Issues in Data mining
- **Sources of data**
- Data Objects and Attribute Types
- About Big Data
- Basic Statistical Descriptions of Data
- Data Visualization
- Case study
- Summary

Sources of data

- Primary data
  - Primary data or raw data is **a type of information that is obtained directly from first-hand sources through experiments, surveys, or observations**.

- Examples of primary data
  - Autobiographies and memoirs.
  - Diaries, personal letters, and correspondence.
  - Interviews, surveys, and fieldwork.
  - Internet communications on email, blogs, listservs, and newsgroups.
  - Photographs, drawings, and posters.
  - Works of art and literature.

- Secondary data
  - Secondary data means **data collected by someone else earlier**.

- Examples of secondary data
  - Tax records and social security data.
  - Census data.
  - Electoral statistics.
  - Health records.
  - Books, journals, or other print media.
  - Social media monitoring, internet searches, and other online data.
  - Sales figures or other reports from third-party companies.

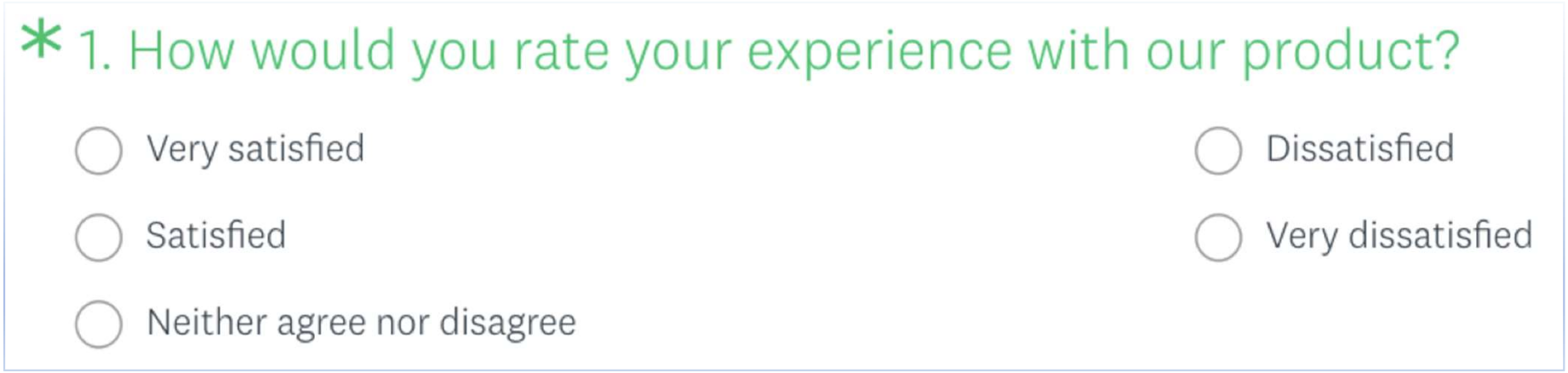
Example of primary data collection

- Quantitative methods
  - Quantitative research is the process of collecting and analyzing numerical data. It can be used to find patterns and averages, make predictions, test causal relationships, and generalize results to wider populations.
- Qualitative methods
  - Examples
  - One-on-one interviews.
    - Interviews are one of the most common qualitative data-collection methods, and they're a great approach when you need to gather highly personalized information
  - Open-ended surveys and questionnaires.
  - Focused groups
  - Observation
  - Case studies

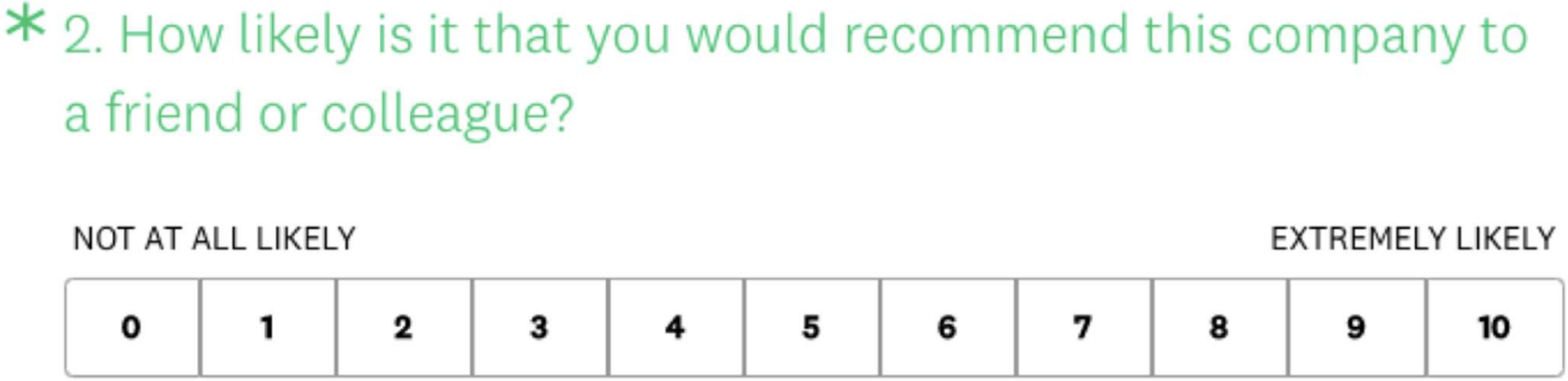
A few sample questions

- Next few slides contain a few sample survey questions

Single-answer multiple choice question.



Rating scales questions



Source: https://www.surveymonkey.com/mp/survey-question-types/

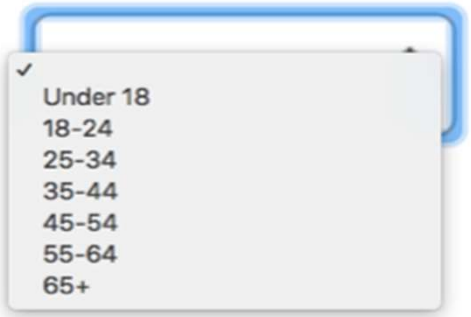
Likert scales: Do you agree or disagree

8. I'm satisfied with the investment my organization makes in education:

- ◯ Strongly agree
- ◯ Agree
- ◯ Neither agree nor disagree
- ◯ Disagree
- ◯ Strongly disagree

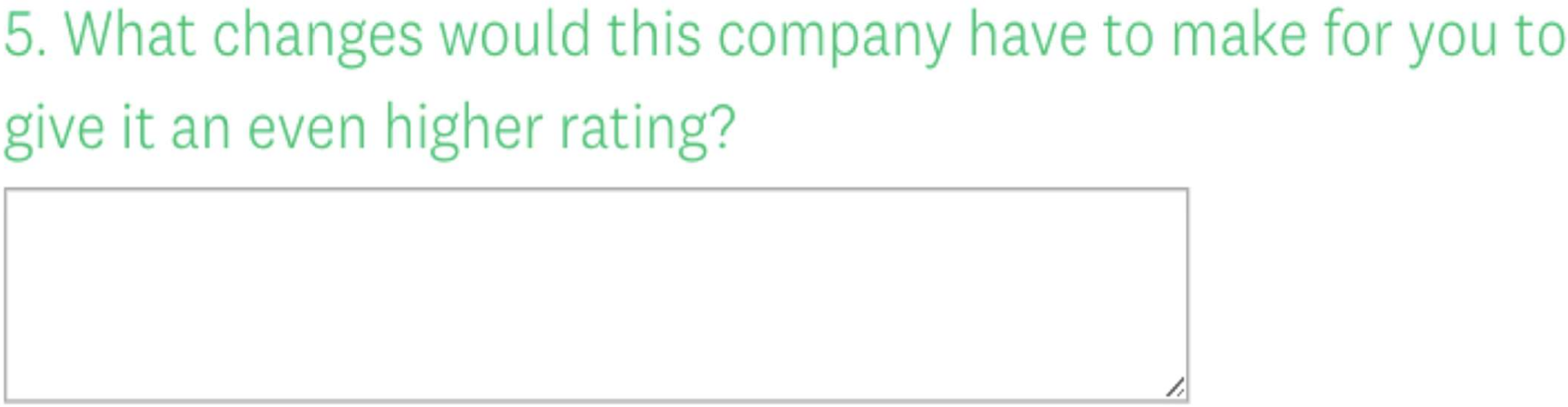
Dropdown questions

3. What's your age?



Open-ended questions

- Open-ended survey questions require respondents to type their answer into a comment box and don't provide specific pre-set answer options. Responses are then viewed individually or by text analysis tools.



Demographic questions

14. Which of the following best describes your current relationship status?

- ◯ Married
- ◯ Widowed
- ◯ Divorced
- ◯ Separated
- ◯ In a domestic partnership or civil union
- ◯ Single, but cohabiting with a significant other
- ◯ Single, never married

Ranking questions

5. Rank the following shows in order of preference—1 being your favorite and 5 being your least favorite.

- ⠿ ▲▼ The office
- ⠿ ▲▼ Parks and Recreation
- ⠿ ▲▼ Arrested Development
- ⠿ ▲▼ Orange is the New Black
- ⠿ ▲▼ New Girl

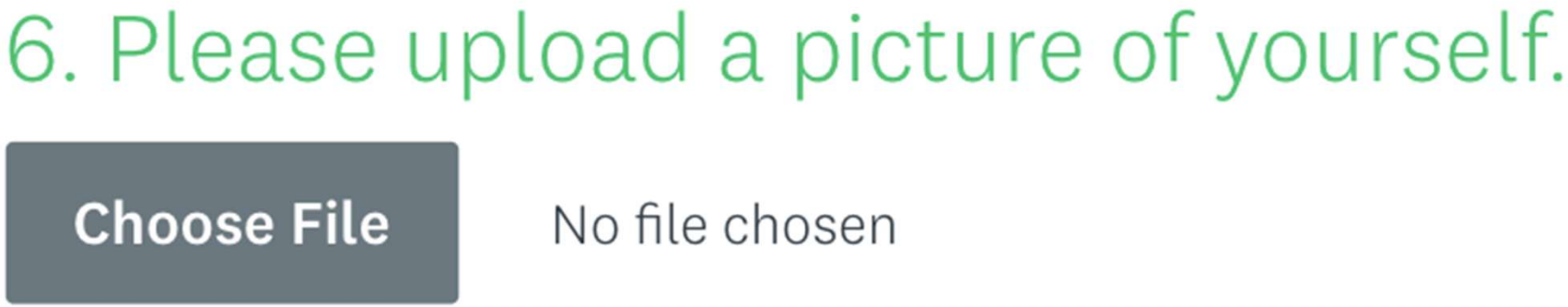
Image choice questions

7. Now that you've reviewed the logos, please pick your favorite.

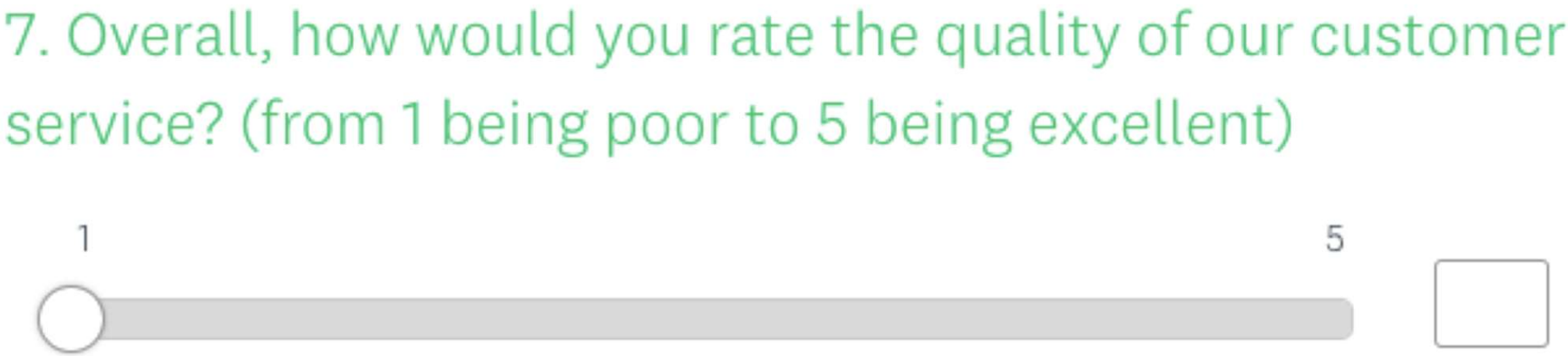
Click map questions

Click the part of the packaging that is the most appealing to you.

File upload questions: Uploading of resume or image



Slider questions



Benchmarkable questions

How likely is it that you would recommend

Acme to a friend or colleague?

Service Feedback | Product Feedback | Insurance

Brand Research | Show More

Form Number 2802 3739

Census of India 2021 Houselisting and Housing Census Schedule

Confidential when filled

Page No. SIDE- A

ion ulars | State/UT | District | Tahsil/Taluk/ P.S./Dev. Block Circle/Mandal | Town/ Village | Ward Code No. (only for Town) | Houselisting Block No.

Fill if the census house is used wholly or partly as a residence (If '1' or '2' in col. 7)

Fill columns 14 to 34 for normal households. Put dash (-) in case of institutional households and non-residential census houses

Amenities available to the household

Assets possessed by the household

Building number (Municipal / local authority / census number)	Census house number	Predominant material of floor, wall and roof of the census house (Give code from the respective lists below)			Ascertain use of Census house (Write the actual use and then choose the appropriate code from the list below and record the same in the box at the left hand side of the column)		If '1' or '2' in col. 7, condition of this census house: Good-1 / Livable- 2 / Dilapidated-3	Household number (Give separate serial number to each household and write '999' for every institutional household)	Total number of persons normally residing in this household	Information relating to the head of the household: Name of the head of the household	Do not fill columns 12 and 13 for institutional households		Ownership status of this house: Owned-1/ Rented but has own house elsewhere-2/ Rented and doesn't own any house-3/ Any other-4	Number of dwelling rooms # exclusively in possession of this household (Record 0,1,2,3..)	Number of married couple(s) living in this household (Record 0, 1, 2, 3....)	Main source of drinking water (Give code from the list below)	Availability of drinking water source: S. Within premises-1/ Near the premises-2/ Away-3	Main source of lighting: Electricity-1/ Kerosene-2 Solar-3/ Other oil-4/ Any other-5/ No lighting-6	Access to latrine: Yes: Exclusively for household use only -1/ Shared with other household-2/ Public latrine-3/ No: Open-4 If '1' or '2' in col. 20, then type of latrine (Give code from the list below)	Waste water outlet connected to: Closed drainage-1/Open drainage-2/ No drainage-3	Bathing facility available within the premises: Yes: Bathroom-1/ Enclosure without roof-2/ No-3	Availability of kitchen and LPG/PNG Connection: (Give code from the list below)	Main fuel used for cooking: (If '1' to '6' in col. 24) (Give code from the list below)	Radio/ Transistor: Yes: Traditional radio set-1/ On mobile/smartphone-2/ On any other device-3 /No-4	Television: Yes: Doordarshan free dish-1/ Other DTH/Dish-2/ Cable connection-3/ Any other-4/No -5	Access to internet: Yes: On laptop/Computer-1/ On mobile/smartphone-2/ On any other device-3/ No-4	Laptop/ Computer: Yes: 1/ No-2	Telephone and Mobile phone/Smartphone (Give code from the list below)	Bicycle and Scooter/ Motorcycle/ Moped (Give code from list below)	Car/ Jeep/ Van: Yes: 1/ No-2	Main cereal consumed in the household: Rice-1/ Wheat-2/ Jowar-3/ Bajra-4/ Maize-5/ Any other-6	Mobile Number (For Census related communications only)
		Floor	Wall	Roof	Code No.	Actual use						Sex: Male-1/ Female-2/ Transgender person-3	If SC* or ST* or Other: SC -1/ ST - 2 / Other - 3																					
2	3	4	5	6	7		8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34		

End of 2nd Lecture

Presentation Outline

- Data mining functionalities
- Research Issues in Data mining
- Sources of data
- **Data Objects and Attribute Types**
- Basic Statistical Descriptions of Data
- Data Visualization
- Case study
- Summary

Data Objects

- Data sets are made up of data objects.
- A **data object** represents an entity.
- Examples:
  - sales database: customers, store items, sales
  - medical database: patients, treatments
  - university database: students, professors, courses
- Also called *samples , examples, instances, data points, objects, tuples, records, vector, patten, event, case, observation or entity*
- Data objects are described by **attributes**.
- Database rows -> data objects; columns ->attributes.

Types of Data Sets

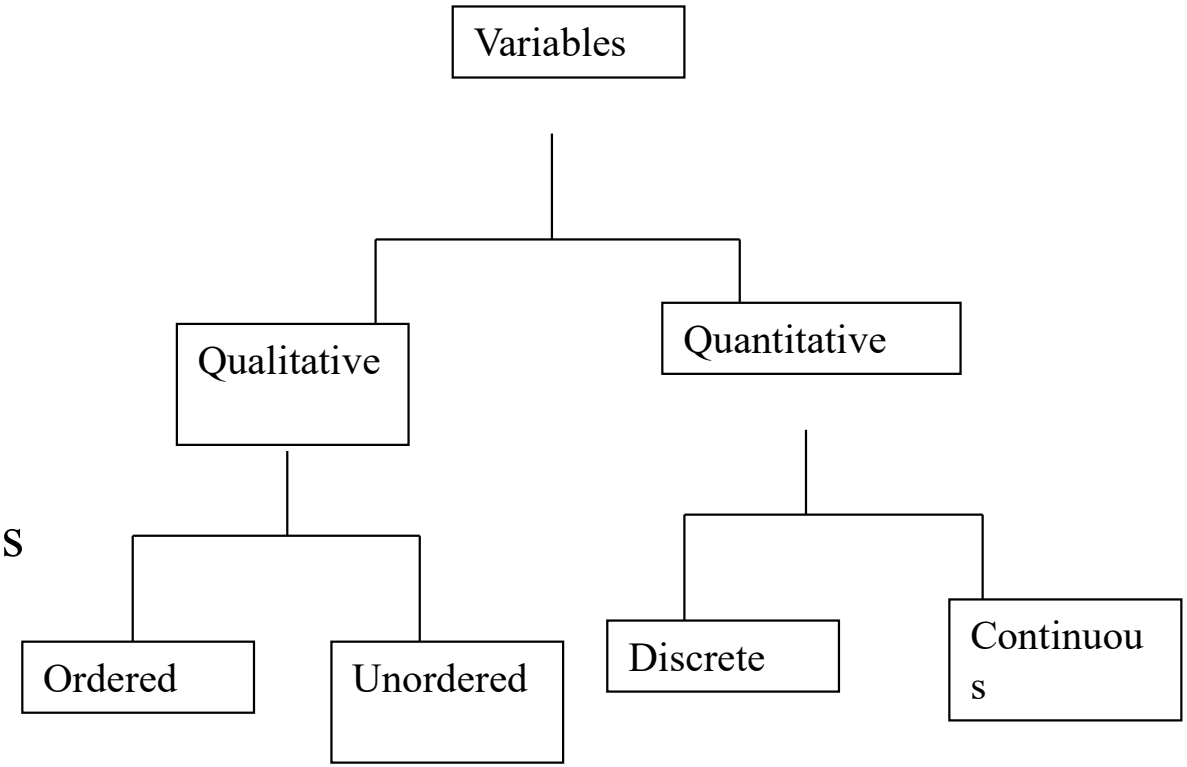
- Record
  - Relational records
  - Data matrix, e.g., numerical matrix, crosstabs
  - Document data: text documents: term-frequency vector
  - Transaction data
- Graph and network
  - World Wide Web
  - Social or information networks
  - Molecular Structures
- Ordered
  - Video data: sequence of images
  - Temporal data: time-series
  - Sequential Data: transaction sequences
  - Genetic sequence data
- Spatial, image and multimedia:
  - Spatial data: maps
  - Image data:
  - Video data:

	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

TID	*Items*
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Attributes

- **Attribute (or dimensions, features, variables):**
  - a data field, representing a characteristic or feature of a data object.
    - *E.g., customer _ID, name, address*
- **A measurement scale:**
  - Associates a numerical or symbolic value with an attribute of an object.
- Type of attribute: Type of measurement of scale
- Attribute Types:
  - Qualitative/categorical attributes
    - Nominal attributes
    - Binary attributes
    - Ordinal attributes
  - Quantitative attributes or numeric attributes
    - Interval-scaled attributes
    - Ratio-scaled attributes



Qualitative Attributes

- **Nominal attributes :** categories, states, or "names of things"
  - *Hair_color = {auburn, black, blond, brown, grey, red, white}*
  - marital status, occupation, ID numbers, zip codes
  - **No order**
  - **Mean/median can not be calculated. Mode is the option.**
- **Binary attributes:** Nominal attribute with only 2 states (0 and 1)
  - Symmetric binary attribute: both outcomes are equally important
    - e.g., gender
  - Asymmetric binary attribute: outcomes not equally important.
    - e.g., medical test (positive vs. negative)
    - Convention: assign 1 to most important outcome (e.g., HIV positive)

- **Ordinal**
  - Values have a meaningful order (ranking), but the magnitude between successive values is unknown.
  - *Example*
    - *Size = {small, medium, large},* grades, army rankings
    - Customer satisfaction: 0: very satisfied, 1: some dissatisfied, 2: neutral, 3: satisfied, and 4: very satisfied
    - Central tendency: Mode or median is the measure
  - New value=f(old value)

Quantitative Numeric Attribute Types

- Quantity (integer or real-valued)
- **Interval**
  - Measured on a scale of **equal-sized units**
  - Values have order
    - E.g., *temperature in C˚or F˚, calendar dates*
  - No true zero-point. New value= a*old value+b
- **Ratio**
  - Inherent **zero-point**
  - We can speak of values as being an order of magnitude larger than the unit of measurement (10 K˚ is twice as high as 5 K˚).
    - e.g., the *temperature in Kelvin, length, counts, monetary quantities*
    - *New value= a*old-value*

Discrete vs. Continuous Attributes

- **Discrete Attribute**
  - Has only a finite or countably infinite set of values
    - E.g., zip codes, profession, or the set of words in a collection of documents
  - Sometimes, represented as integer variables
  - Note: Binary attributes are a special case of discrete attributes
- **Continuous Attribute**
  - Has real numbers as attribute values
    - E.g., temperature, height, or weight
  - Practically, real values can only be measured and represented using a finite number of digits
  - Continuous attributes are typically represented as floating-point variables

Presentation Outline

- Data mining functionalities
- Research Issues in Data mining
- Sources of data
- Data Objects and Attribute Types
- **Basic Statistical Descriptions of Data**
- Data Visualization
- Summary

Basic Statistical Descriptions of Data

- Motivation
  - To better understand the data: central tendency, variation and spread
- Data dispersion characteristics
  - median, max, min, quantiles, outliers, variance, etc.
- Numerical dimensions correspond to sorted intervals
  - Data dispersion: analyzed with multiple granularities of precision
  - Boxplot or quantile analysis on sorted intervals
- Dispersion analysis on computed measures
  - Folding measures into numerical dimensions
  - Boxplot or quantile analysis on the transformed cube

Basic Statistical Descriptions of Data…

- Measures of Central Tendency (middle or center of the distribution)
  - Mean, Median, Mode, Geometric Mean, Harmonic Mean
- Variability measures or measuring the dispersion. How the data is spread out?
  - Range, Quartile Deviation, Mean Deviation, Standard Deviation, Coefficient of Variation
- Skewness
- Constructing a boxplot

Measures of Central Tendency

- A group of data is represented with a single number
- It brings very important information from it

Arithmetic Mean

- Arithmetic mean: Sum of observations divided by its number.

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

  Note: *n* is sample size and *N* is population size.

  - Weighted arithmetic mean:

$$\mu = \frac{\sum x}{N}$$

  - Trimmed mean: chopping extreme values

- It should be computable.
  - Mean from categorical data is not calculable

$$\overline{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

- Advantages: Can carry out algebraic manipulations.
- Demerits:
  - Gives more weight to extreme items (outliers).

Median

- Median: Middle value if odd number of values, or average of the middle two values otherwise
- Merits
  - It can be calculated even if extreme classes are not defined.
- Property
  - Sum of absolute deviations from is least when it is taken from median.
- Demerits
  - Not based on all observations
  - Not widely used in practice

Mode

- Mode: It is the value which occurs most frequently.
- Merits
  - It can be easily located
  - It can be calculated when extreme classes are not defined
  - It is used widely in the business.
  - It can be calculated easily except if maximum frequency occurs more than once.
- Demerits
  - Not based on the all observations
  - It is not having algebraic properties.
  - It is not stable
    - Different class intervals results into different mode value.
- The relationship between mean, median and mode
  - Mean - mode=3(mean-median)

Selecting among the mean, median and mode

- Common mistake: specifying the wrong index of central tendency
  - It is common to specify the mean
- If data is categorical, if yes use “mode”.
  - If no
    - If the total is of interest, use “mean”
    - If no
      - If the distribution is skewed, use “median”
      - Otherwise, use “mean”

- Type of micro processor → use mode
- Total CPU time → mean

- Skew: ratio of minimum and maximum is large, the data is skewed.

Summarizing the Variability

- It is a measure that can give the widespread or scattering of observations among themselves or from a center point.
  - Range, Quartile deviation, Mean deviation, and standard deviation (variance)
- Range:
  - The difference between the highest and lowest values in a series of observations.
  - Problem: it depends on two extreme values.
- Quartile deviation
  - Quartile deviation (Q.D)=$(Q_3-Q_1)/2$, where Q1 is first quartile, Q3 is third quartile. The first and third quartiles are called lower and upper quartiles, respectively.
  - First quartile: The value of the variate below which one-fourth of the values lie and above which three-fourths of the values lie, when the values are arranged in ascending order of magnitude.
  - Third quartile: The value of the variate below which three-fourths of the values lie and above which the remaining one-fourth of the values lie, when the values are arranged in ascending order of magnitude.
  - Merits:
    - The presence of abnormal values does not affect quartile deviation.

Mean deviation, Standard Deviation, Variance

- Mean Deviation: The mean deviation is the mean of the absolute values of the deviations taken from the average.

- Standard deviation: It is defined as the square root of the mean of the squares of the deviations taken from the arithmetic mean.

  - σ =square root($1/n \sum (x1 - X)^{2)}$ where X is arithmetic mean

- Variance: Square of standard deviation.

Coefficient of Variation

- CV is the percentage ratio of S.D to mean.
- CV= (SD/Mean)*100

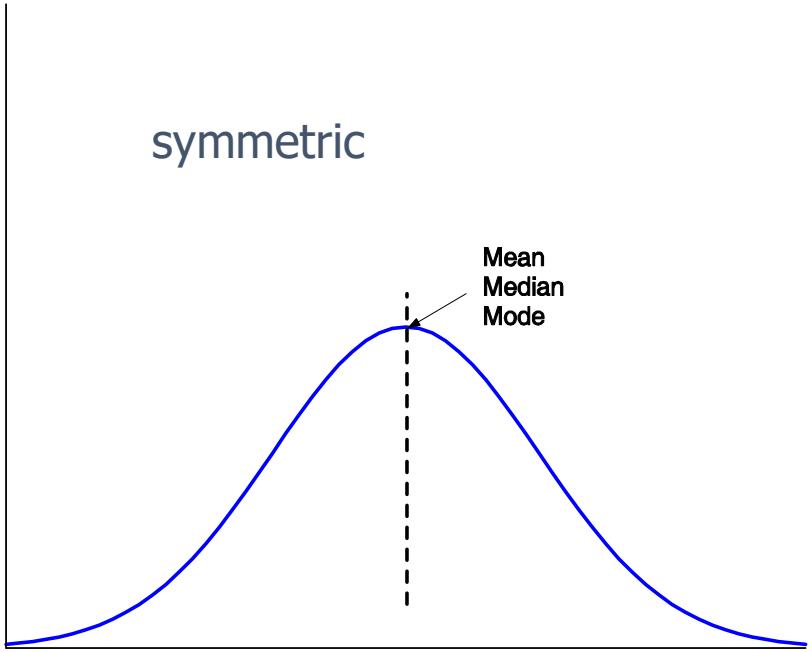
- For a player scores: if CV is high he/she is inconsistent,

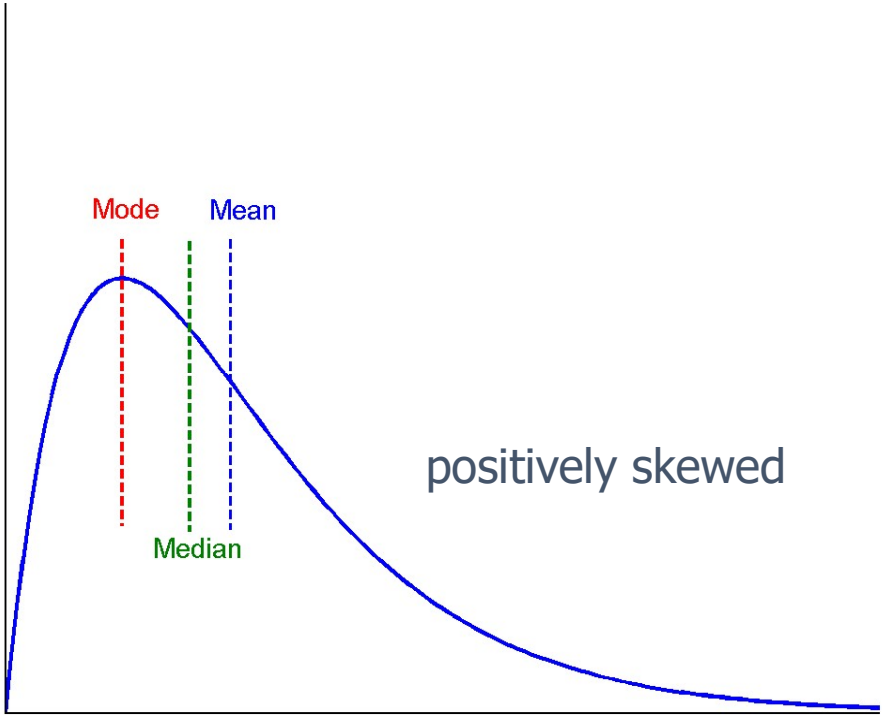
Statistical Population

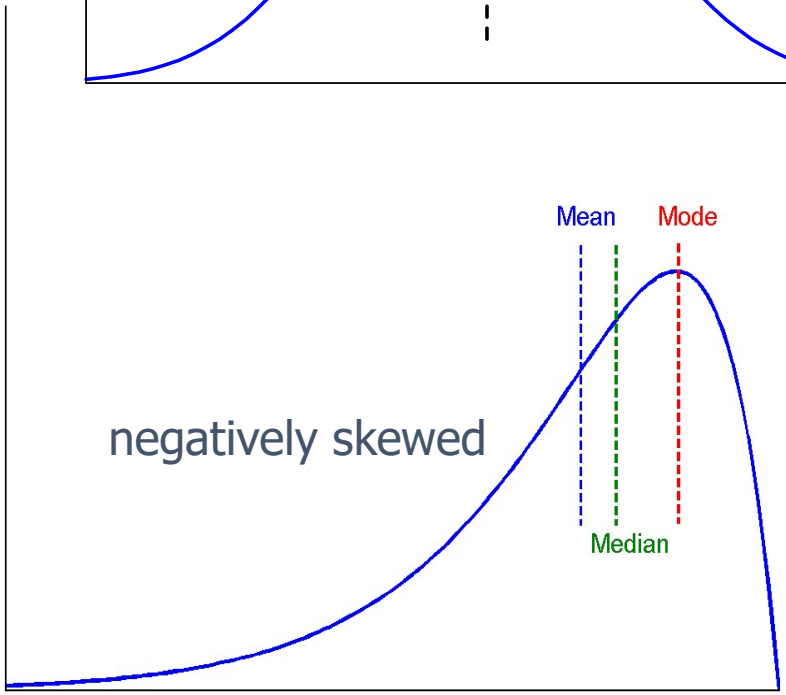
- Population: All the values/objects
- Sample: A part of population.
  - A sample should be a representative of population
  - Should be taken in a random manner.
  - Population is specified with parameters
  - Sample is specified with statistics
- Population error: Difference between the sample mean and population mean.

Symmetric vs. Skewed Data

- Even if two measures mean and standard deviations are same for the distributions, still the shape of two curves may differ.







Determining the Distribution of Data

- The simplest way is to plot the histogram of data
  - Requires dividing the range into a number of sub-ranges called cells or buckets.
- The count of observations that fall into each cell are determined.
- The counts are normalized to frequencies by dividing by the total number of observations. The cell frequencies are plotted as a column chart
- Key problem in determining the cell size.
  - Small cells lead to very few observations per cell and large variations in the number of observations.
  - Large cells result in fewer variations, but the details of observations are completely lost.
  - Guideline: if a cell has fewer than five observations, the cell size should be increased, or a variable cell histogram should be used.

Measures of Skewness

- Pearson's Coefficient of skewness
  - (Mean-mode)/SD
- Quartile coefficient of skewness
  - ((Q3-Q2)-(Q2-Q1))/(Q3-Q1)

Selecting Index of Dispersion

- If the variable is bounded, use “range”.
- If there are no natural bounds, and the distribution is symmetric use either SD, variance or CV.
- If the distribution is non-symmetric, percentiles are best choices.

Box Plot
or The 5 Number Summary

- A **box plot** or **boxplot** (also known as a **box-and-whisker diagram** or **plot**) is a convenient way of graphically depicting groups of numerical data through their five-number summaries:
  - the smallest observation (sample minimum),
  - lower quartile (Q1),
  - median (Q2),
  - upper quartile (Q3), and
  - largest observation (sample maximum).
  - A boxplot may also indicate which observations, if any, might be considered outliers.
    - usually, a value higher/lower than 1.5 x IQR

Constructing a box and whisker plot

- Step 1 - Find the median.
- Remember, the median is the middle value in a data set.

18, 27, 34, 52, 54, 59, 61, 68, 78, 82, 85, 87, 91, 93, 100

68 is the median of this data set.

Constructing a box and whisker plot

- Step 2 – Find the lower quartile.
- The lower quartile is the median of the data set to the left of 68.

(18, 27, 34, 52, 54, 59, 61,) 68, 78, 82, 85, 87, 91, 93, 100

52 is the lower quartile

Constructing a box and whisker plot

- Step 3 – Find the upper quartile.
- The upper quartile is the median of the data set to the right of 68.

18, 27, 34, 52, 54, 59, 61, 68, (78, 82, 85, 87, 91, 93, 100)

87 is the upper quartile

Constructing a box and whisker plot

- Step 4 – Find the maximum and minimum values in the set.
- The maximum is the greatest value in the data set.
- The minimum is the least value in the data set.

18, 27, 34, 52, 54, 59, 61, 68, 78, 82, 85, 87, 91, 93, 100

18 is the minimum and 100 is the maximum.

Constructing a box and whisker plot

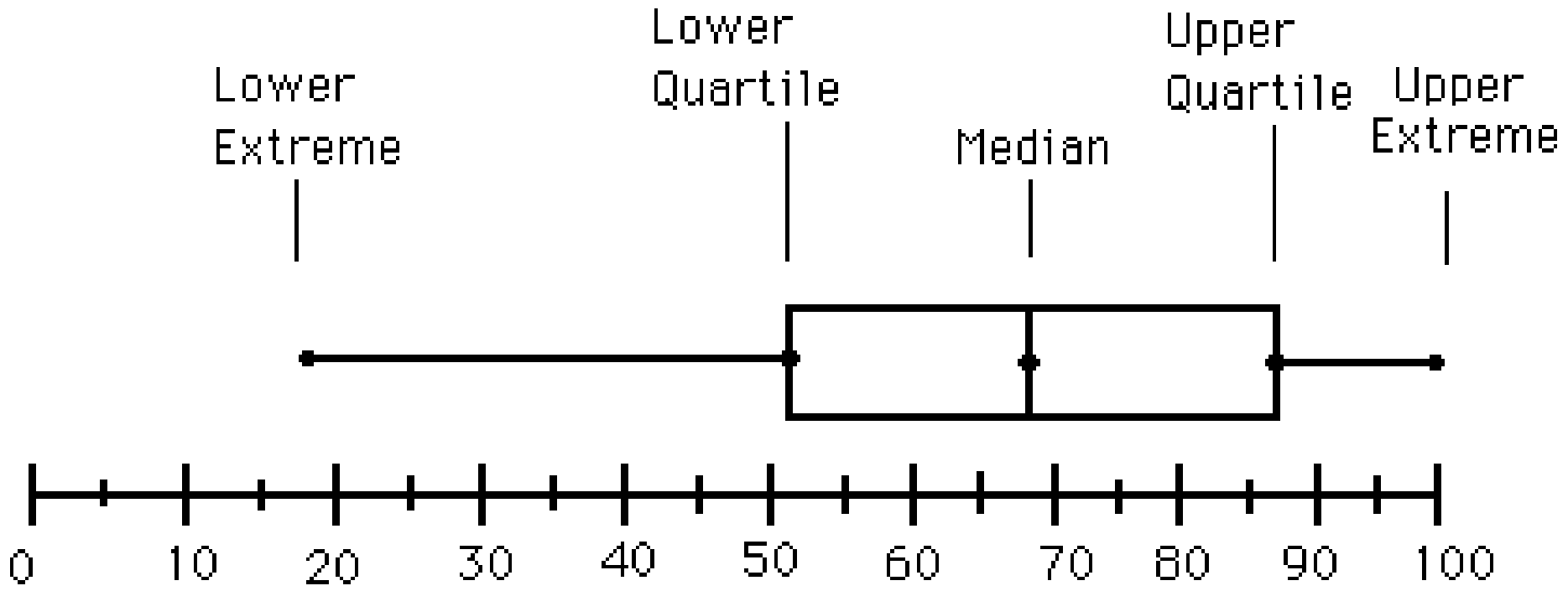
- Step 5 – Find the inter-quartile range (IQR).
- The inter-quartile (IQR) range is the difference between the upper and lower quartiles.
  - Upper Quartile = 87
  - Lower Quartile = 52
  - 87 – 52 = 35
  - 35 = IQR

The 5 Number Summary

- Organize the 5 number summary
  - Median – 68
  - Lower Quartile – 52
  - Upper Quartile – 87
  - Max – 100
  - Min – 18

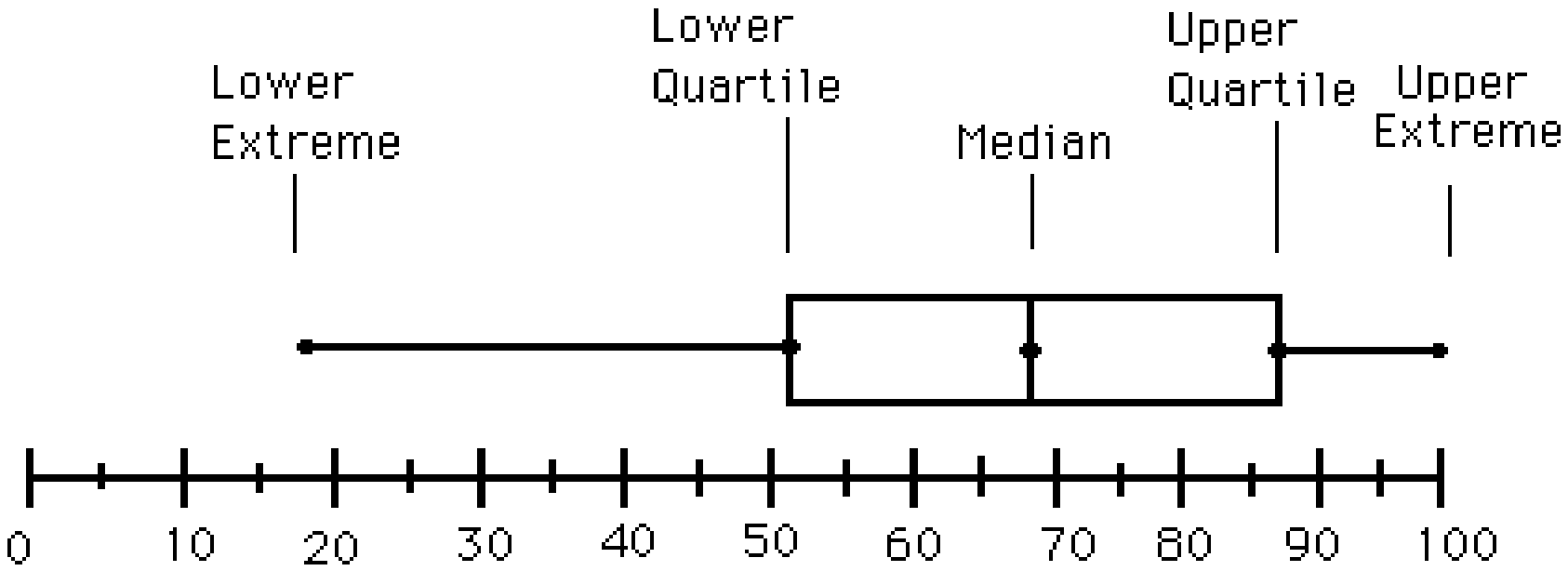
Graphing The Data

- Notice, the Box includes the lower quartile, median, and upper quartile.
- The Whiskers extend from the Box to the max and min.

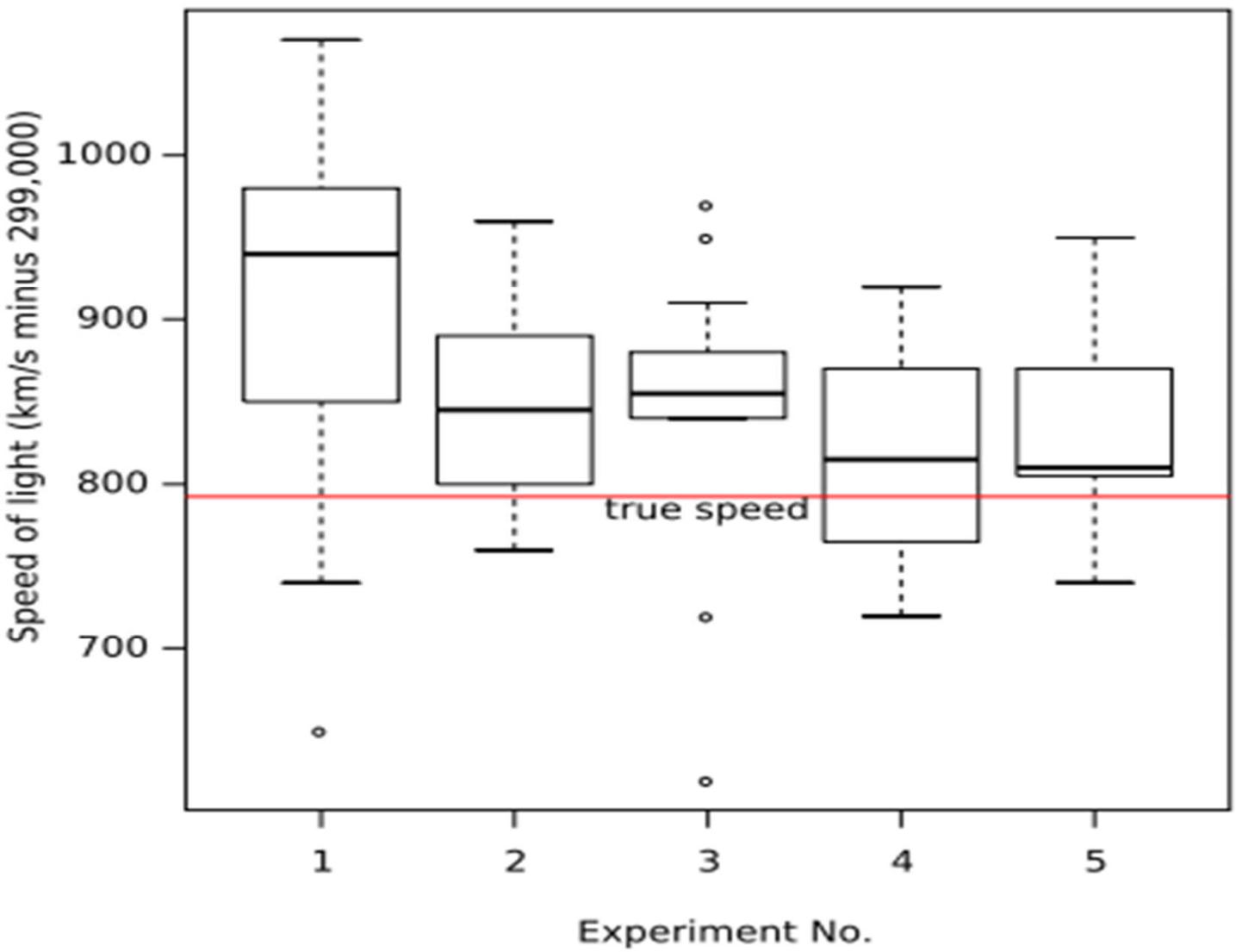


Analyzing The Graph

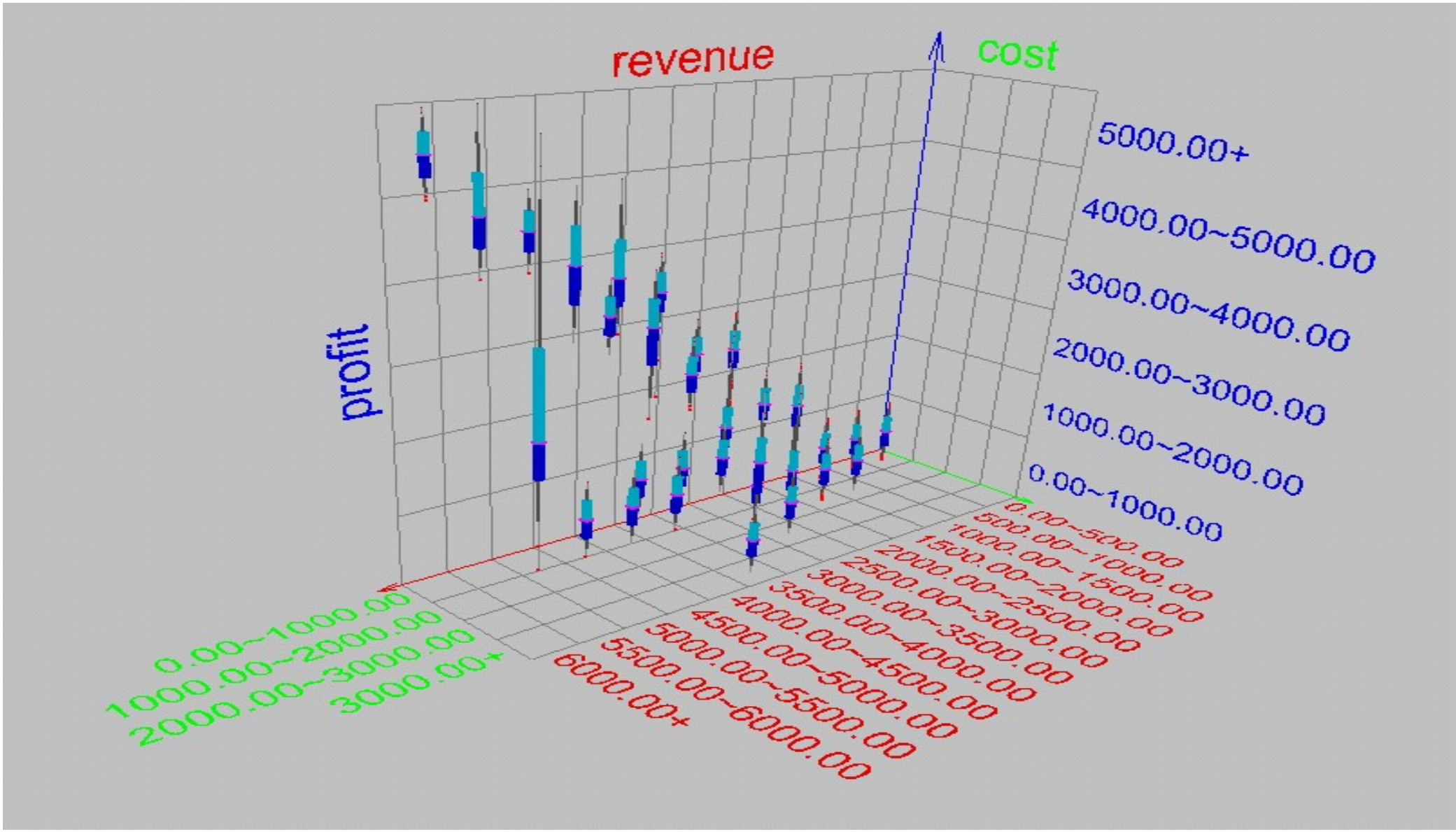
- The data values found inside the box represent the middle half (50%) of the data.
- The line segment inside the box represents the median



Example Box Plot



Visualization of Data Dispersion: 3-D Boxplots

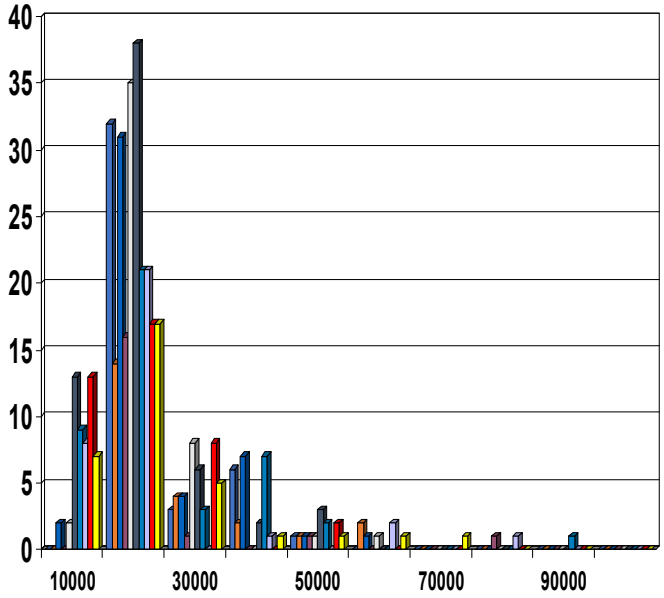


Graphic Displays of Basic Statistical Descriptions

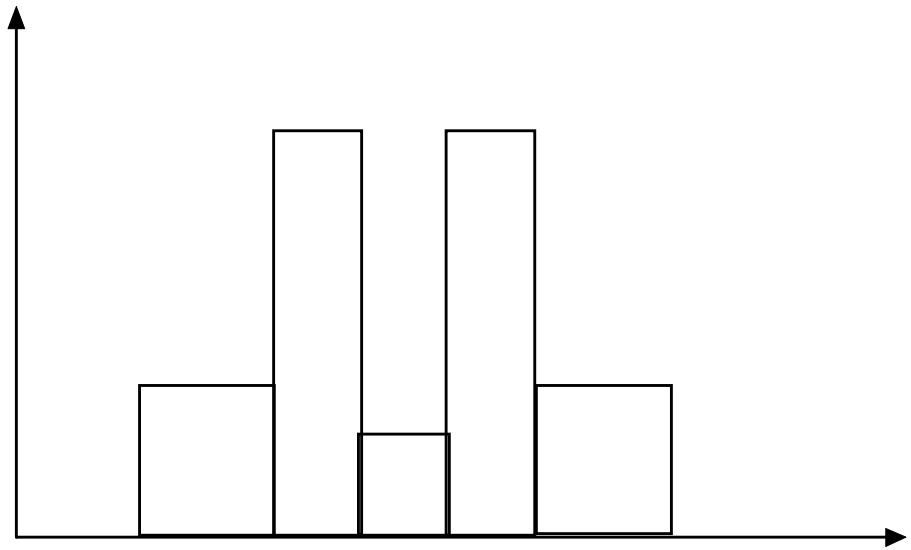
- **Boxplot**: graphic display of five-number summary
- **Histogram**: x-axis are values, y-axis repres. frequencies
- **Quantile plot**: each value x_i is paired with f_i indicating that approximately 100 f_i % of data are $\leq x_i$
- **Quantile-quantile (q-q) plot**: graphs the quantiles of one univariant distribution against the corresponding quantiles of another
- **Scatter plot**: each pair of values is a pair of coordinates and plotted as points in the plane

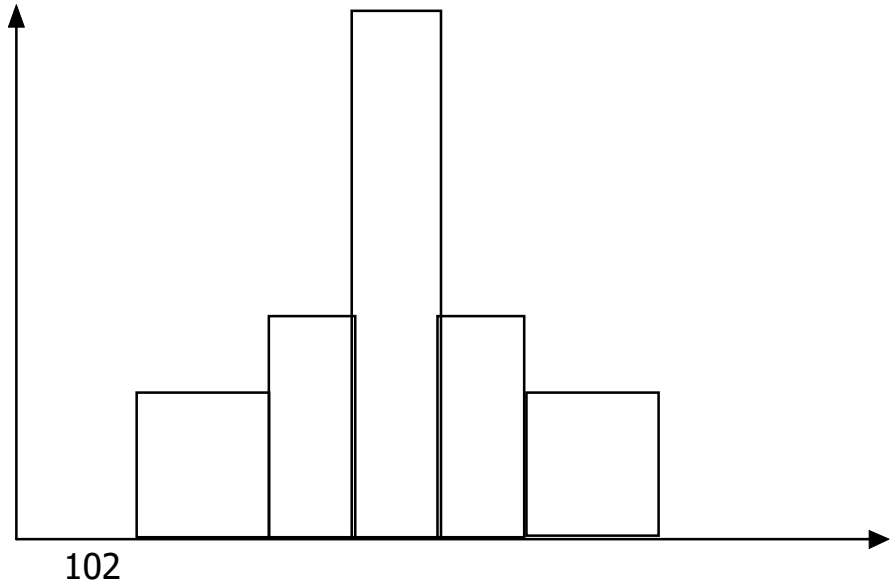
Histogram Analysis

- Histogram: Graph display of tabulated frequencies, shown as bars
- It shows what proportion of cases fall into each of several categories
- Differs from a bar chart in that it is the *area* of the bar that denotes the value, not the height as in bar charts, a crucial distinction when the categories are not of uniform width
- The categories are usually specified as non-overlapping intervals of some variable. The categories (bars) must be adjacent



Histograms Often Tell More than Boxplots

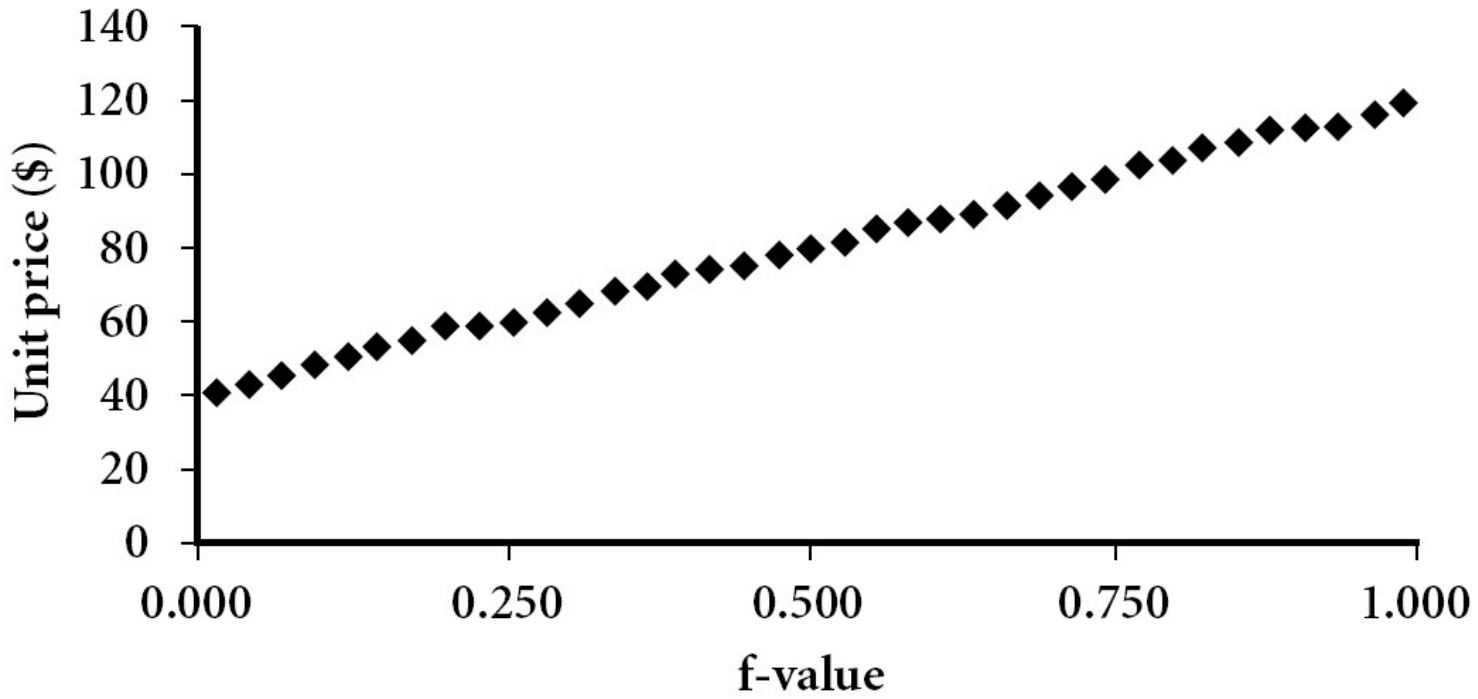




- The two histograms shown in the left may have the same boxplot representation
  - The same values for: min, Q1, median, Q3, max
- But they have rather different data distributions

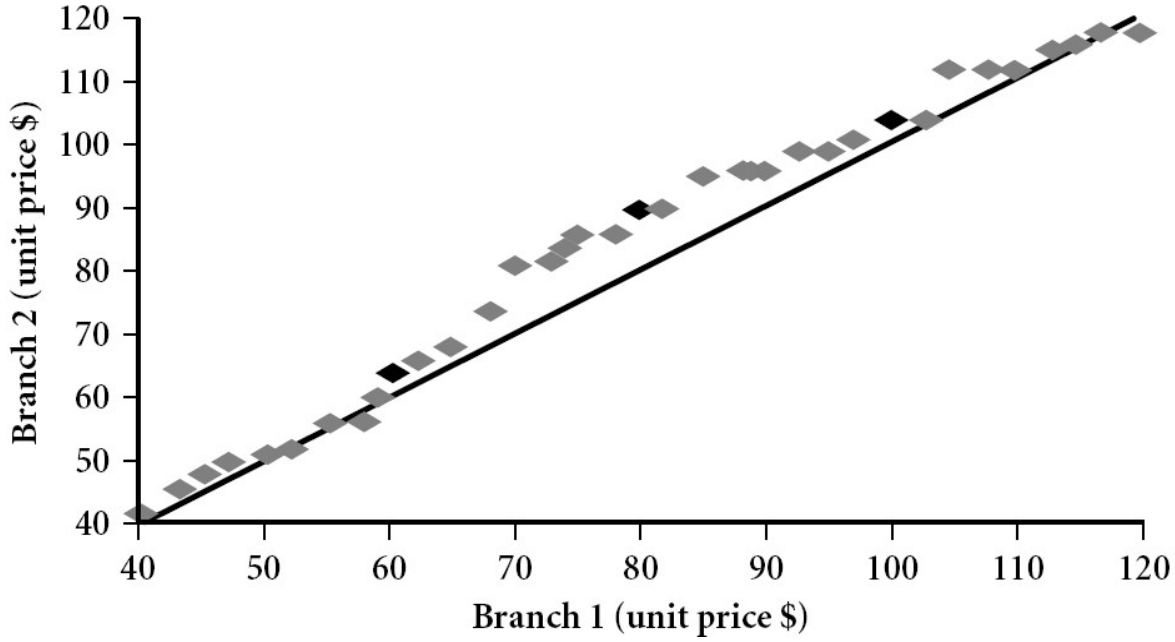
Quantile Plot

- Displays all of the data (allowing the user to assess both the overall behavior and unusual occurrences)
- Plots quantile information
  - For a data x_i data sorted in increasing order, f_i indicates that approximately 100 f_i% of the data are below or equal to the value x_i



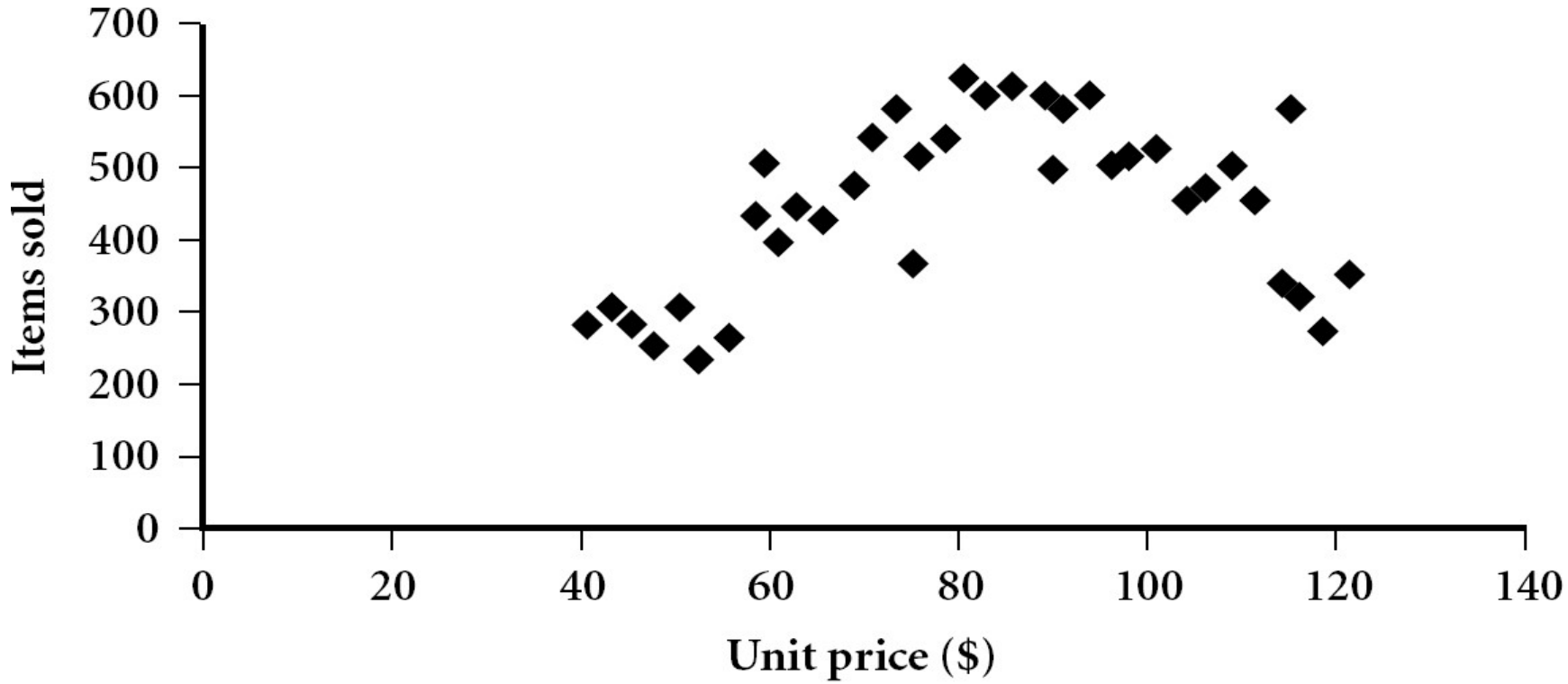
Quantile-Quantile (Q-Q) Plot

- Graphs the quantiles of one univariate distribution against the corresponding quantiles of another
- View: Is there is a shift in going from one distribution to another?
- Example shows unit price of items sold at Branch 1 vs. Branch 2 for each quantile. Unit prices of items sold at Branch 1 tend to be lower than those at Branch 2.

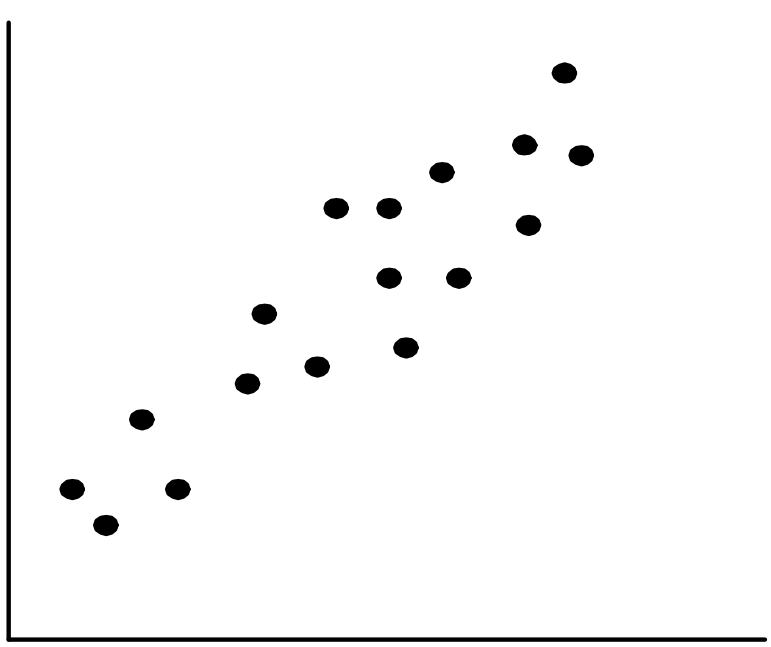


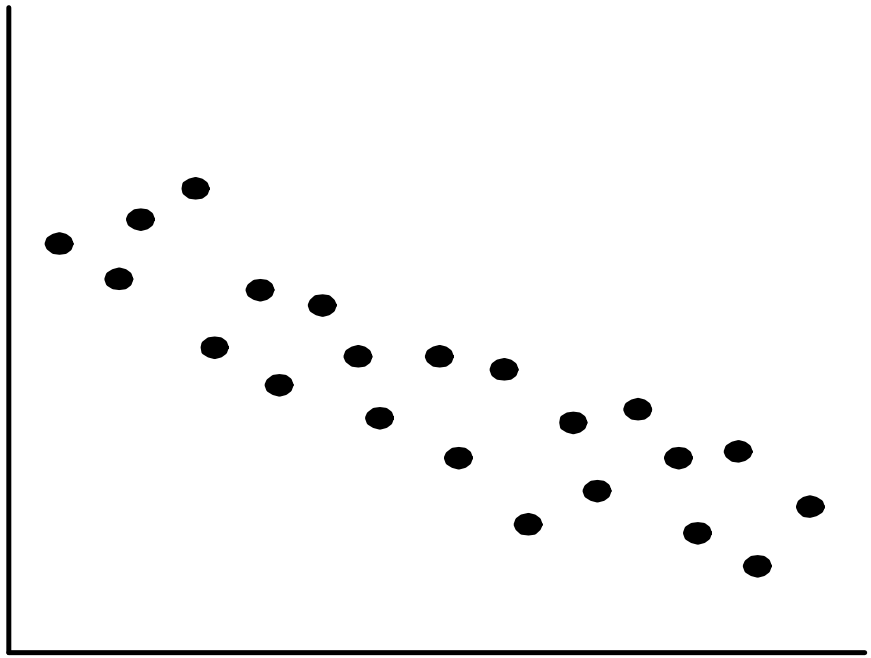
Scatter plot

- Provides a first look at bivariate data to see clusters of points, outliers, etc
- Each pair of values is treated as a pair of coordinates and plotted as points in the plane



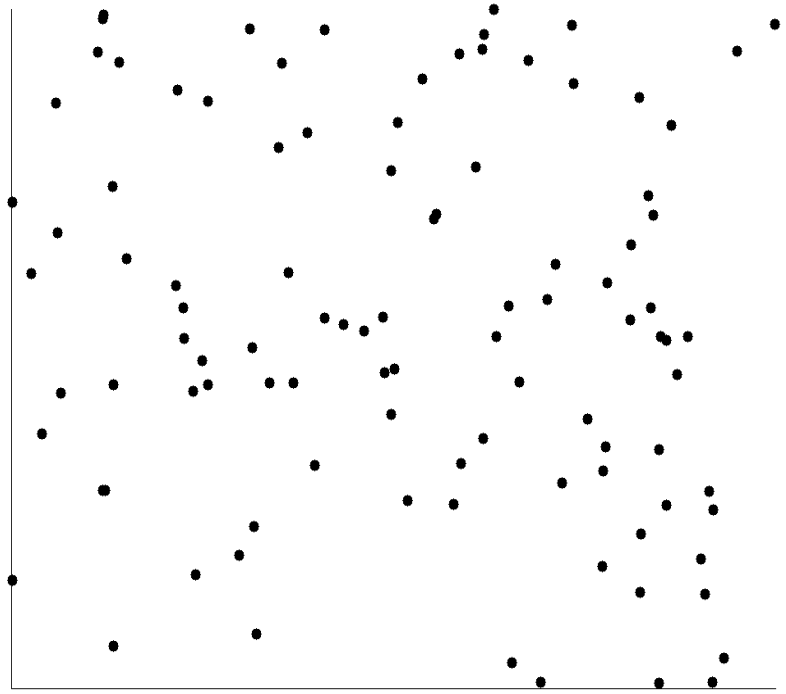
Positively and Negatively Correlated Data

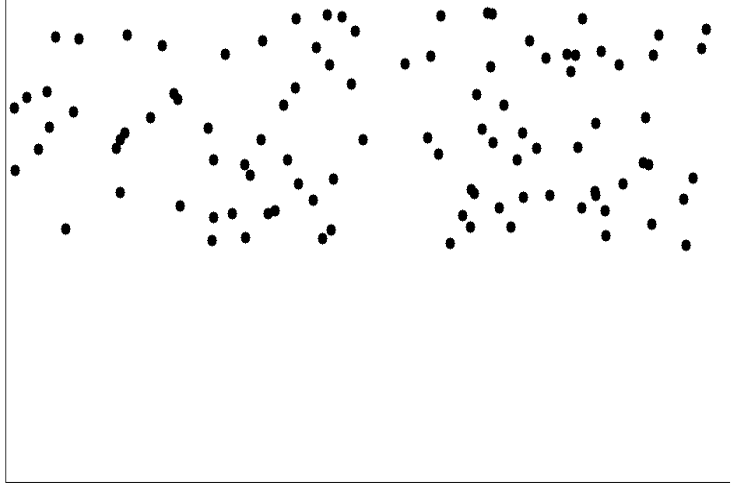


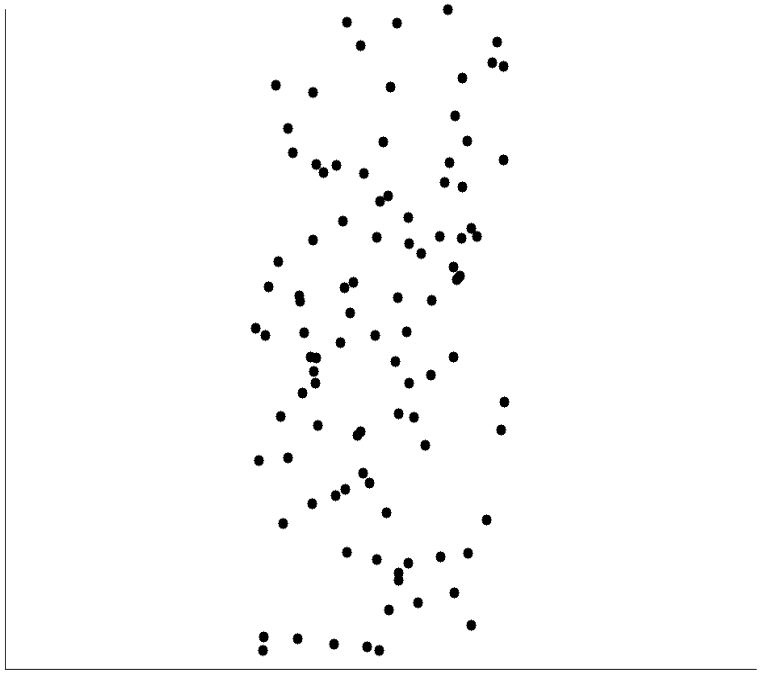


- The left half fragment is positively correlated
- The right half is negative correlated

Uncorrelated Data







Presentation Outline

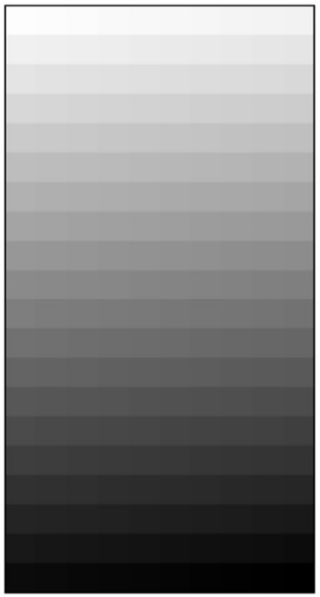
- Data mining functionalities
- Research Issues in Data mining
- Sources of data
- Data Objects and Attribute Types
- Basic Statistical Descriptions of Data
- **Data Visualization**
- Case study
- Summary

Data Visualization

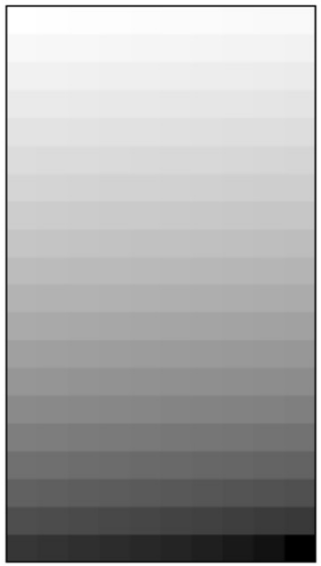
- Why data visualization?
  - Gain insight into an information space by mapping data onto graphical primitives
  - Provide qualitative overview of large data sets
  - Search for patterns, trends, structure, irregularities, relationships among data
  - Help find interesting regions and suitable parameters for further quantitative analysis
  - Provide a visual proof of computer representations derived
- Categorization of visualization methods:
  - Pixel-oriented visualization techniques
  - Geometric projection visualization techniques
  - Icon-based visualization techniques
  - Hierarchical visualization techniques
  - Visualizing complex data and relations

Pixel-Oriented Visualization Techniques

- For a data set of m dimensions, create m windows on the screen, one for each dimension
- The m dimension values of a record are mapped to m pixels at the corresponding positions in the windows
- The colors of the pixels reflect the corresponding values
- Example: Income ascending order of customer information table

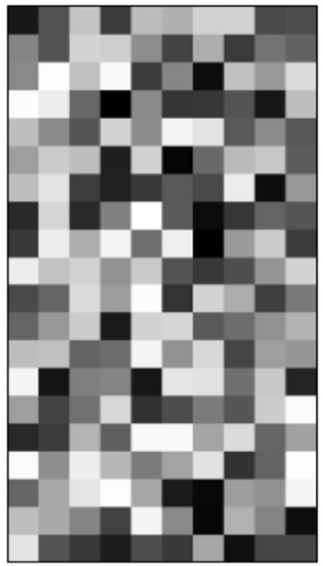


(a) Income



(b) Credit Limit

(c) transaction volume



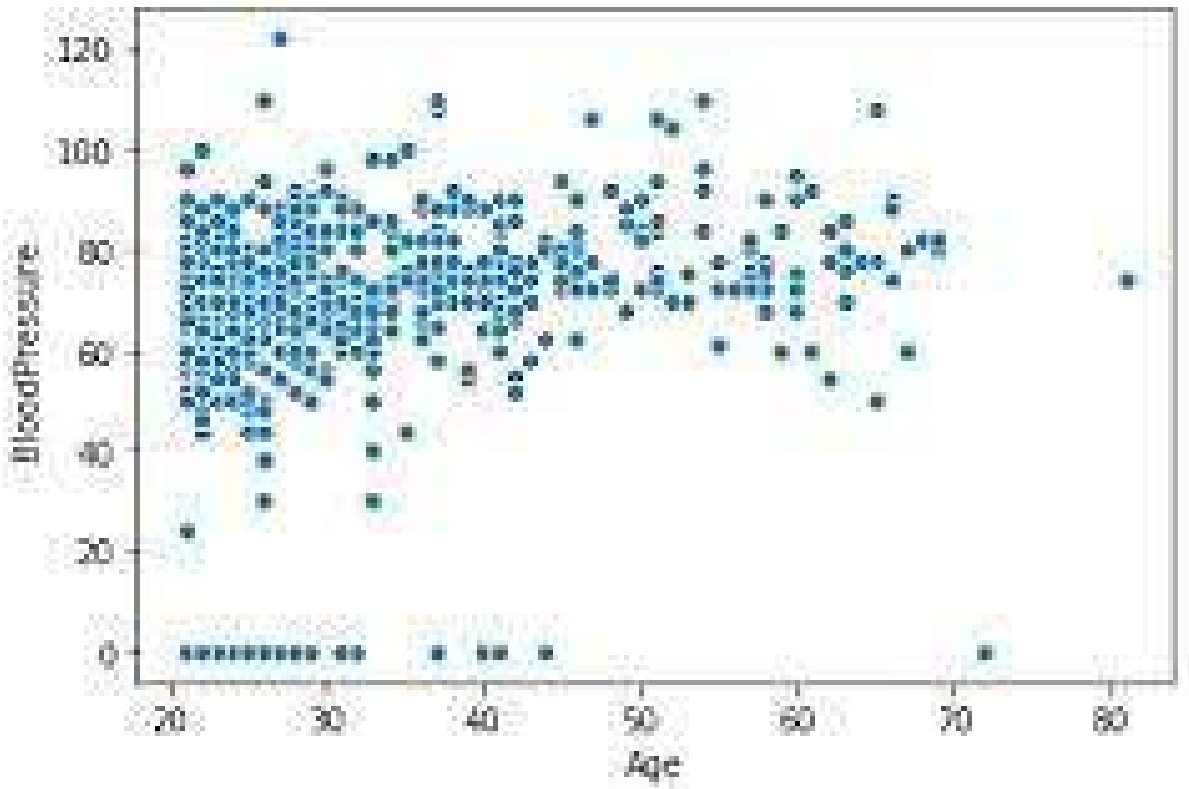
(d) age

Geometric Projection Visualization Techniques

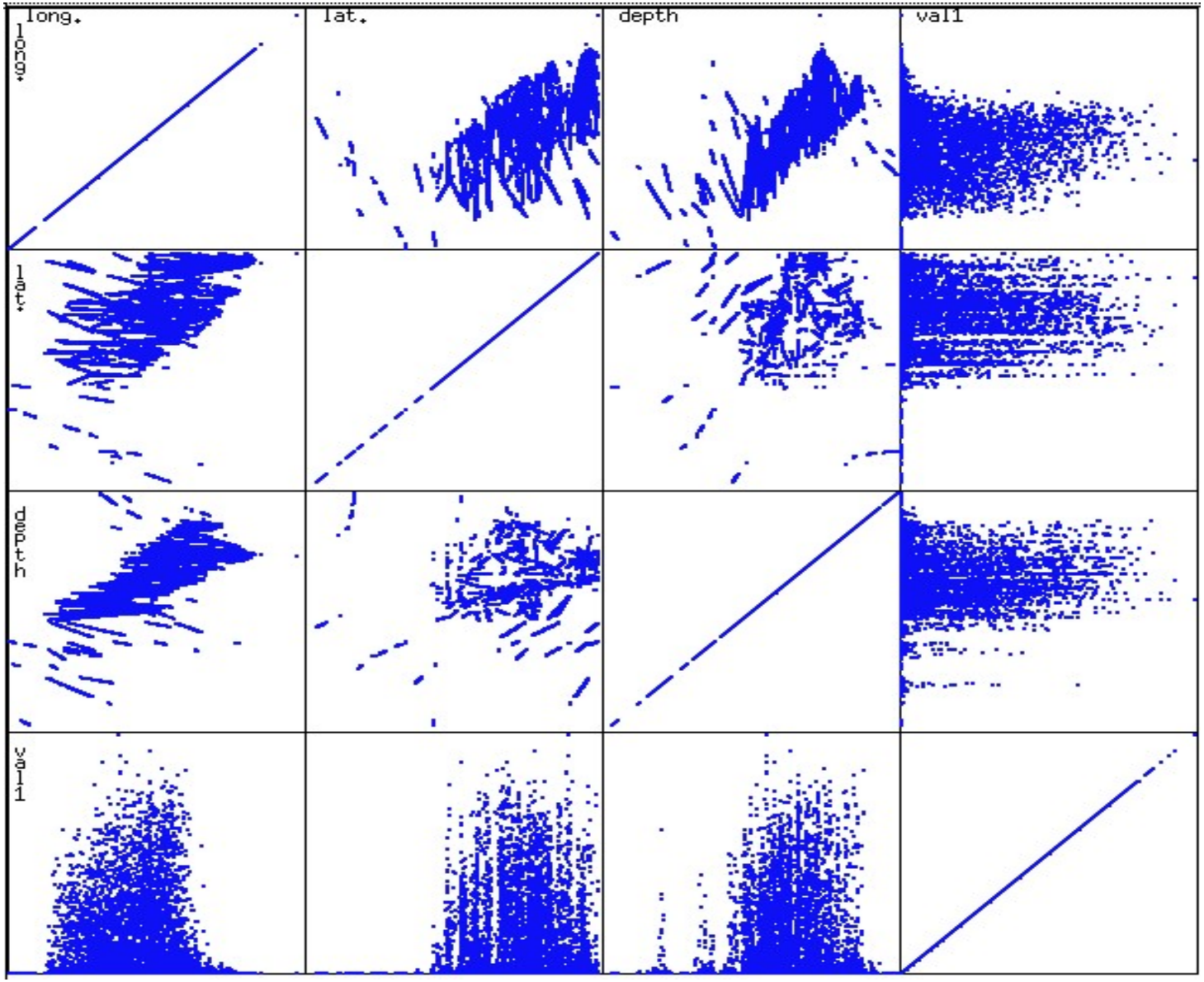
- Visualization of geometric transformations and projections of the data
- Methods
  - Scatterplot and scatterplot matrices
  - Landscapes
  - Parallel coordinates

Scatter plot

A scatter plot displays 2D data points using Cartesian coordinates.

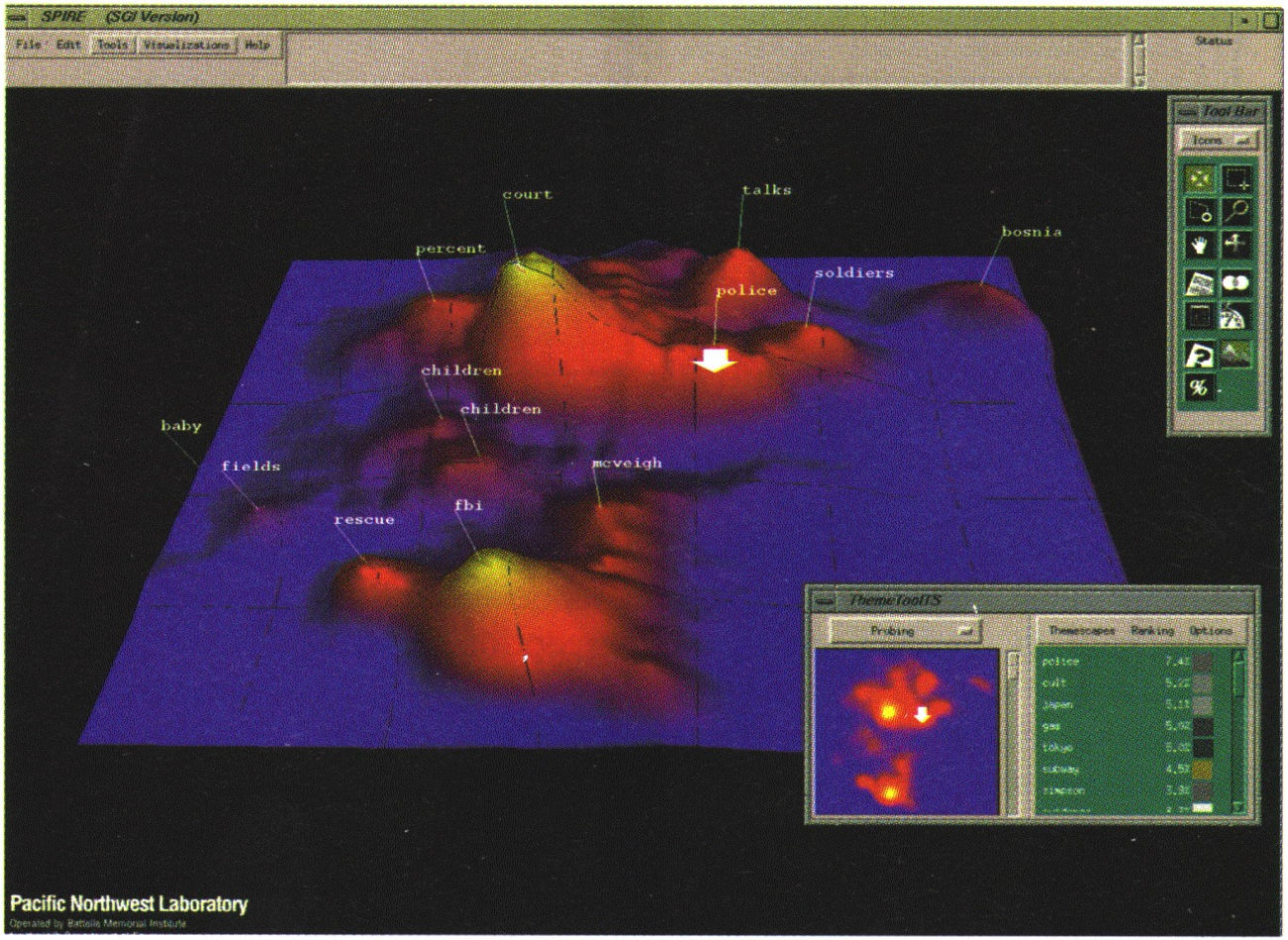


Scatterplot Matrices



Matrix of scatterplots (x-y-diagrams) of the k-dim. data [total of (k2/2-k) scatterplots]

Landscapes



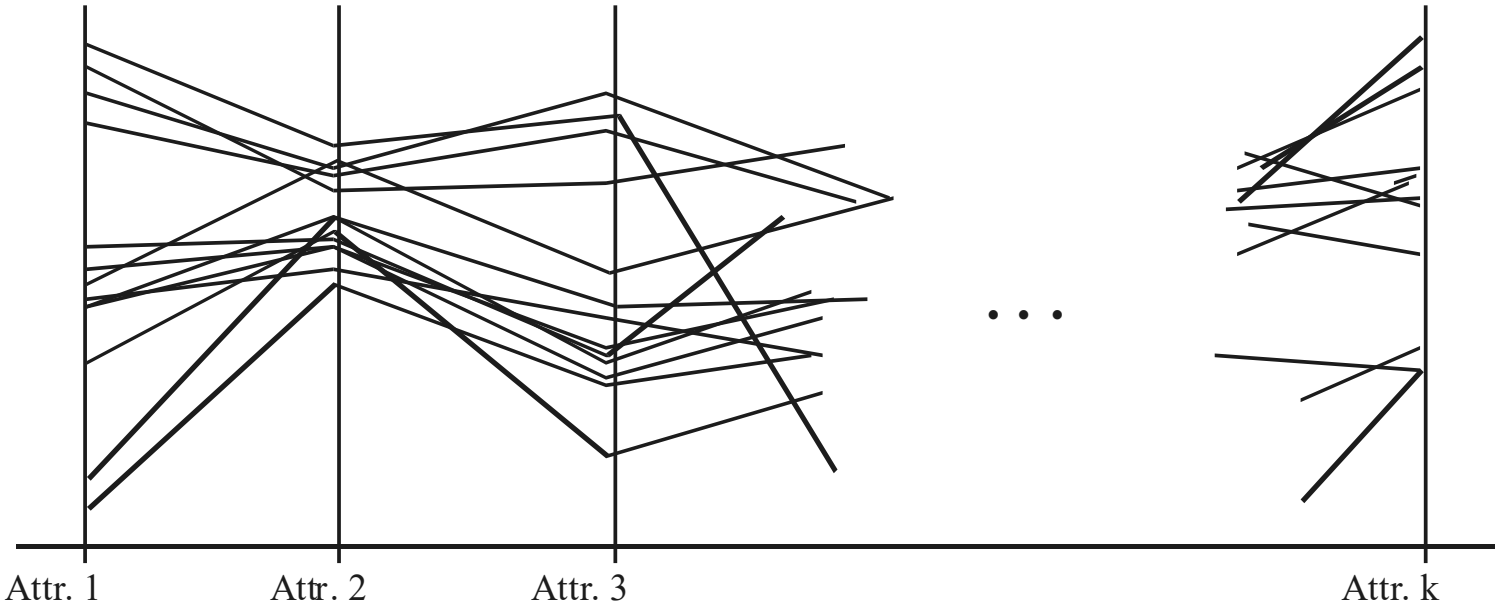
Used by permission of B. Wright, Visible Decisions Inc.

news articles visualized as a landscape

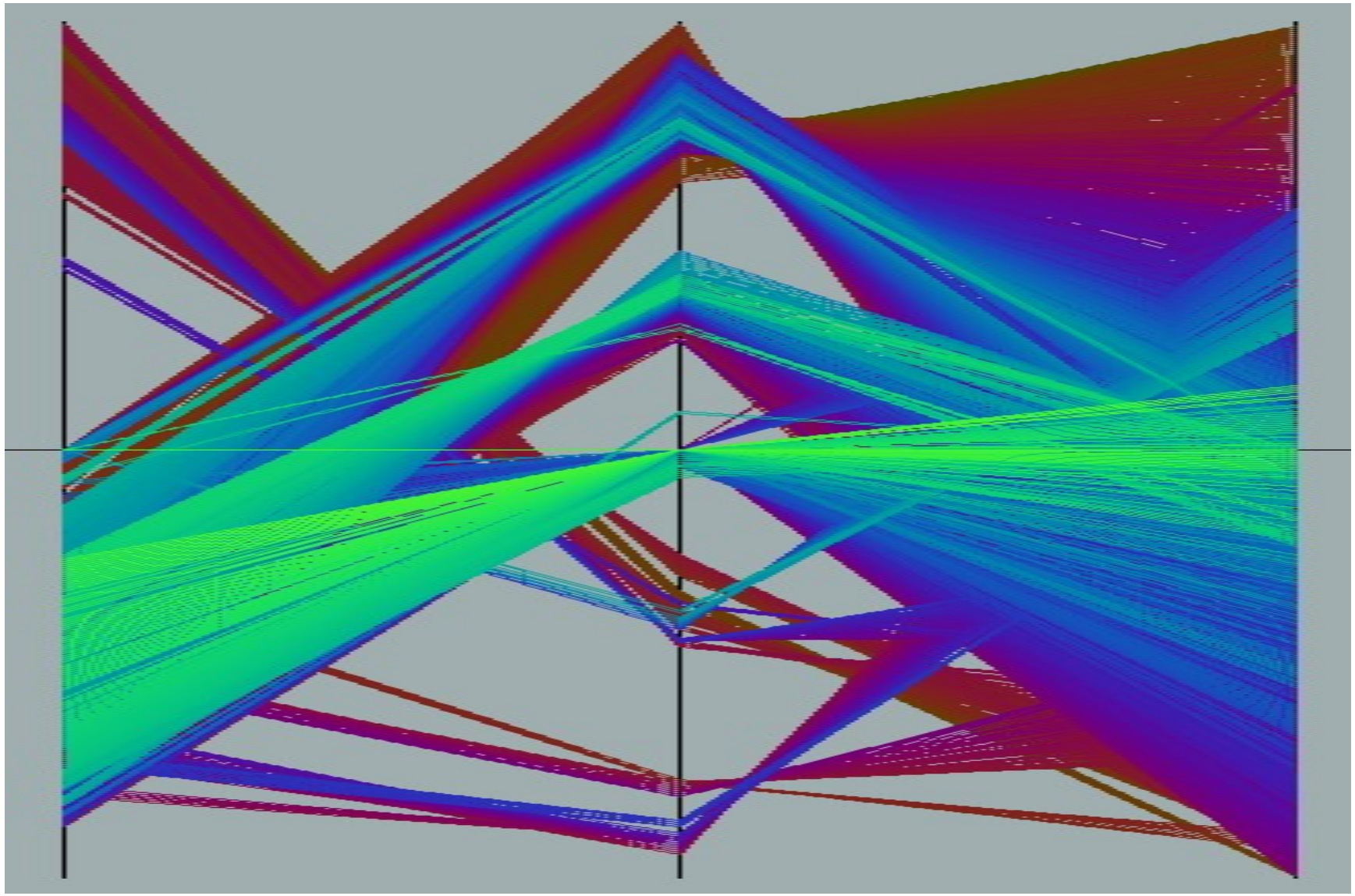
- Visualization of the data as perspective landscape
- The data needs to be transformed into a (possibly artificial) 2D spatial representation which preserves the characteristics of the data

Parallel Coordinates

- In equidistant axes which are parallel to one of the screen axes and correspond to the attributes
- The axes are scaled to the [minimum, maximum]: range of the corresponding attribute
- Every data item corresponds to a polygonal line which intersects each of the axes at the point which corresponds to the value for the attribute



Parallel Coordinates of a Data Set

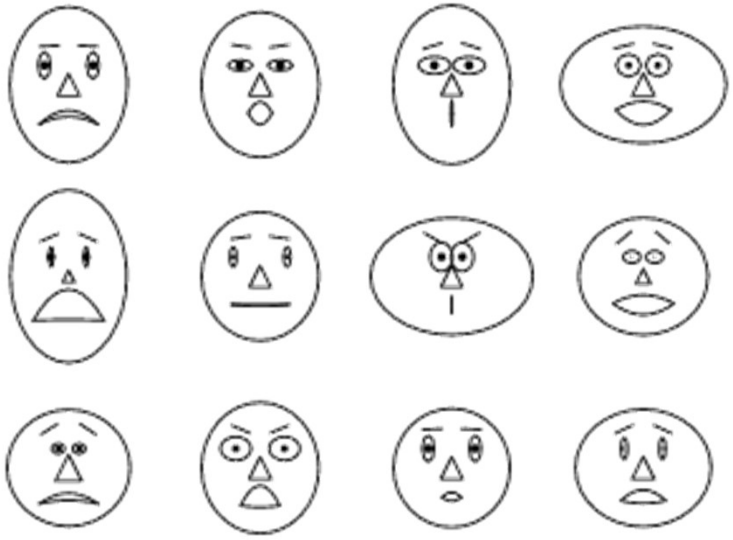


Icon-Based Visualization Techniques

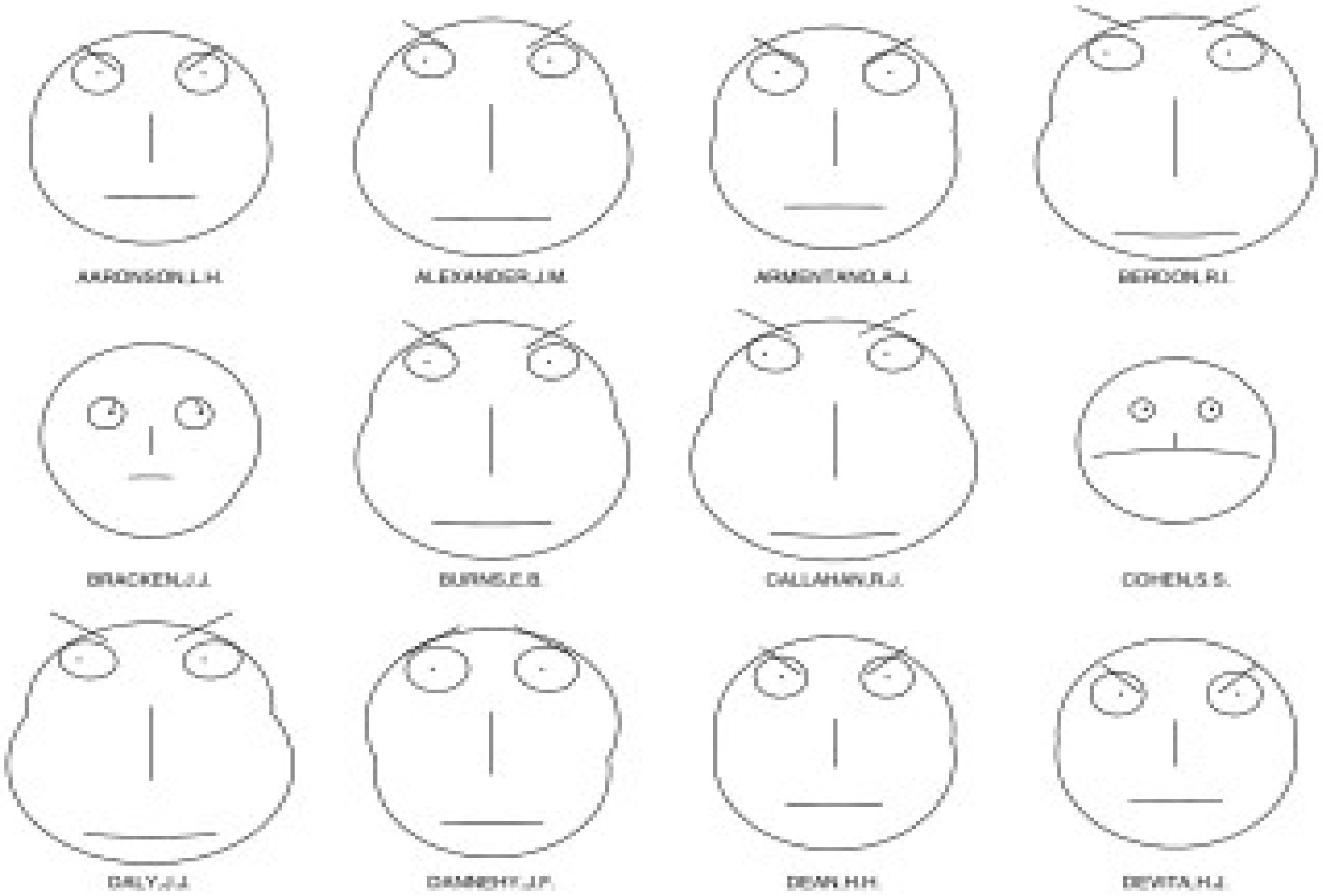
- Visualization of the data values as features of icons
- Typical visualization methods
  - Chernoff Faces
  - Stick Figures
- General techniques
  - Shape coding: Use shape to represent certain information encoding
  - Color icons: Use color icons to encode more information
  - Tile bars: Use small icons to represent the relevant feature vectors in document retrieval

Chernoff Faces

- A way to display variables on a two-dimensional surface, e.g., let x be eyebrow slant, y be eye size, z be nose length, etc.
- The figure shows faces produced using 10 characteristics--head eccentricity, eye size, eye spacing, eye eccentricity, pupil size, eyebrow slant, nose size, mouth shape, mouth size, and mouth opening): Each assigned one of 10 possible values, generated using *Mathematica* (S. Dickson)

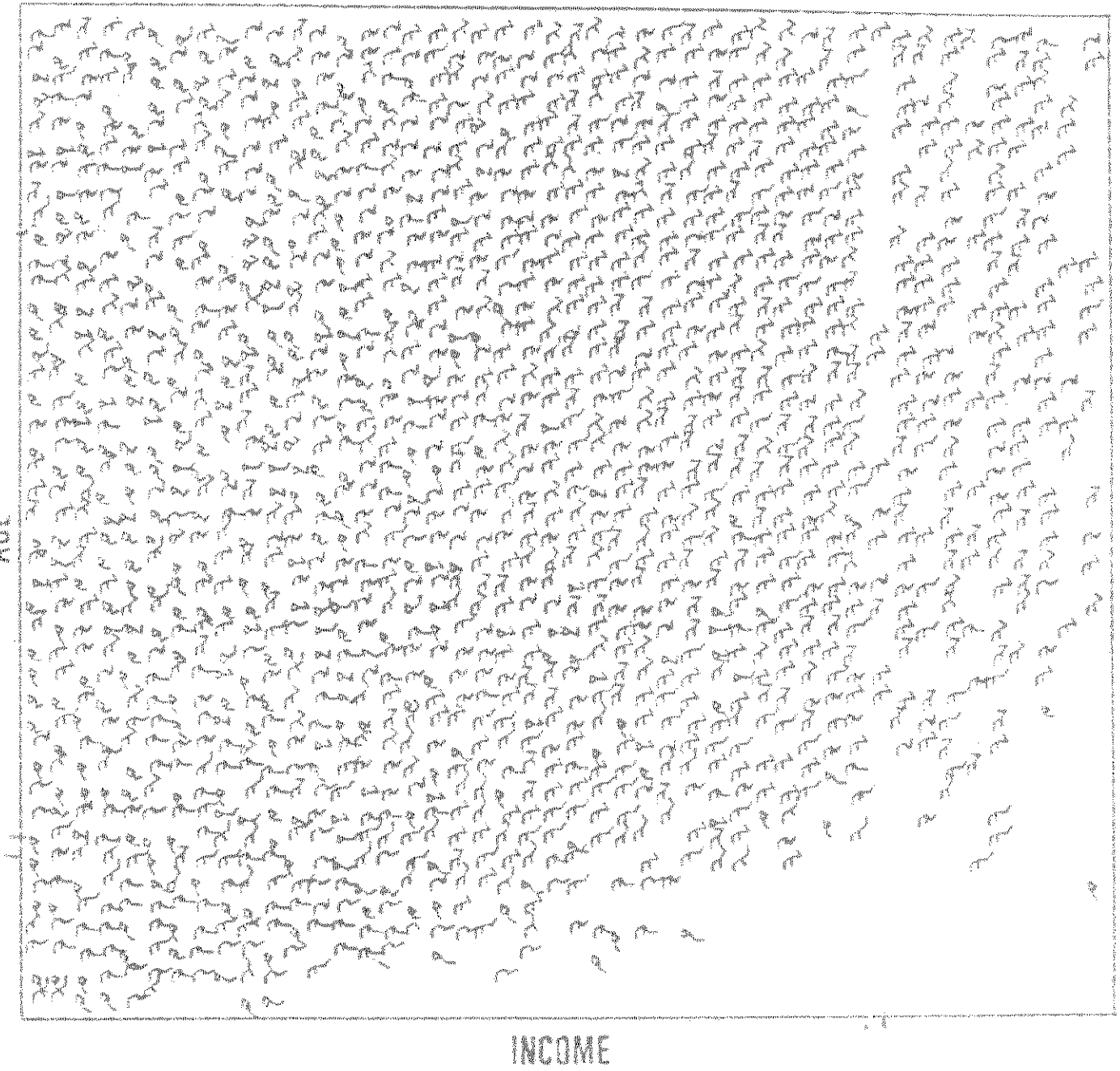


- REFERENCE: Gonick, L. and Smith, W. *The Cartoon Guide to Statistics.* New York: Harper Perennial, p. 212, 1993
- Weisstein, Eric W. "Chernoff Face." From *MathWorld*--A Wolfram Web Resource. mathworld.wolfram.com/ChernoffFace.html



This example shows Chernoff faces for lawyers' ratings of twelve judges

Stick Figure



A census data figure showing age, income, gender, education, etc.

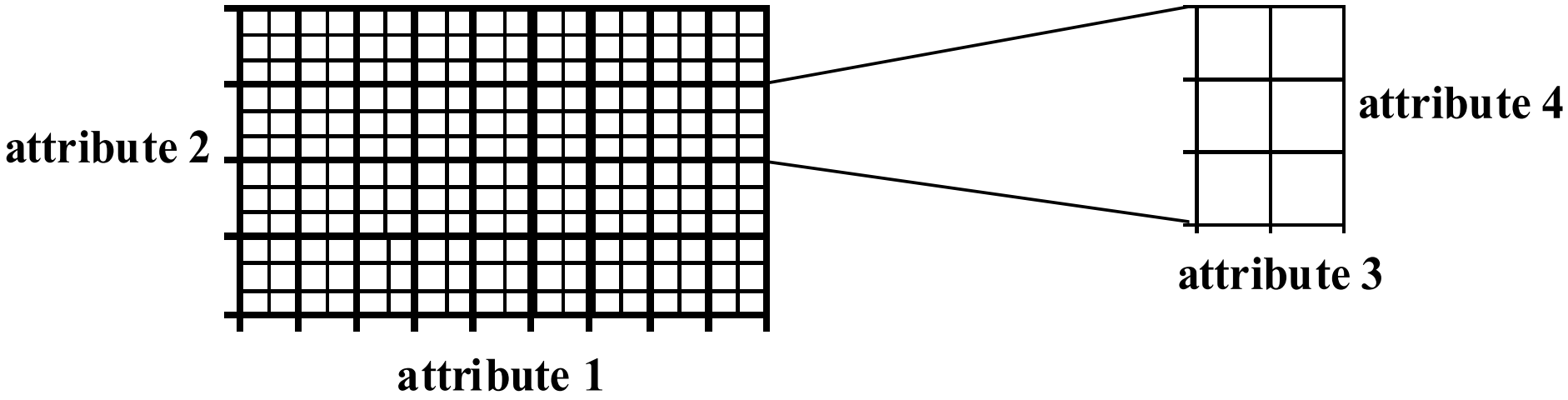
A 5-piece stick figure (1 body and 4 limbs w. different angle/length)

Two attributes mapped to axes, remaining attributes mapped to angle or length of limbs". Look at texture pattern

Hierarchical Visualization Techniques

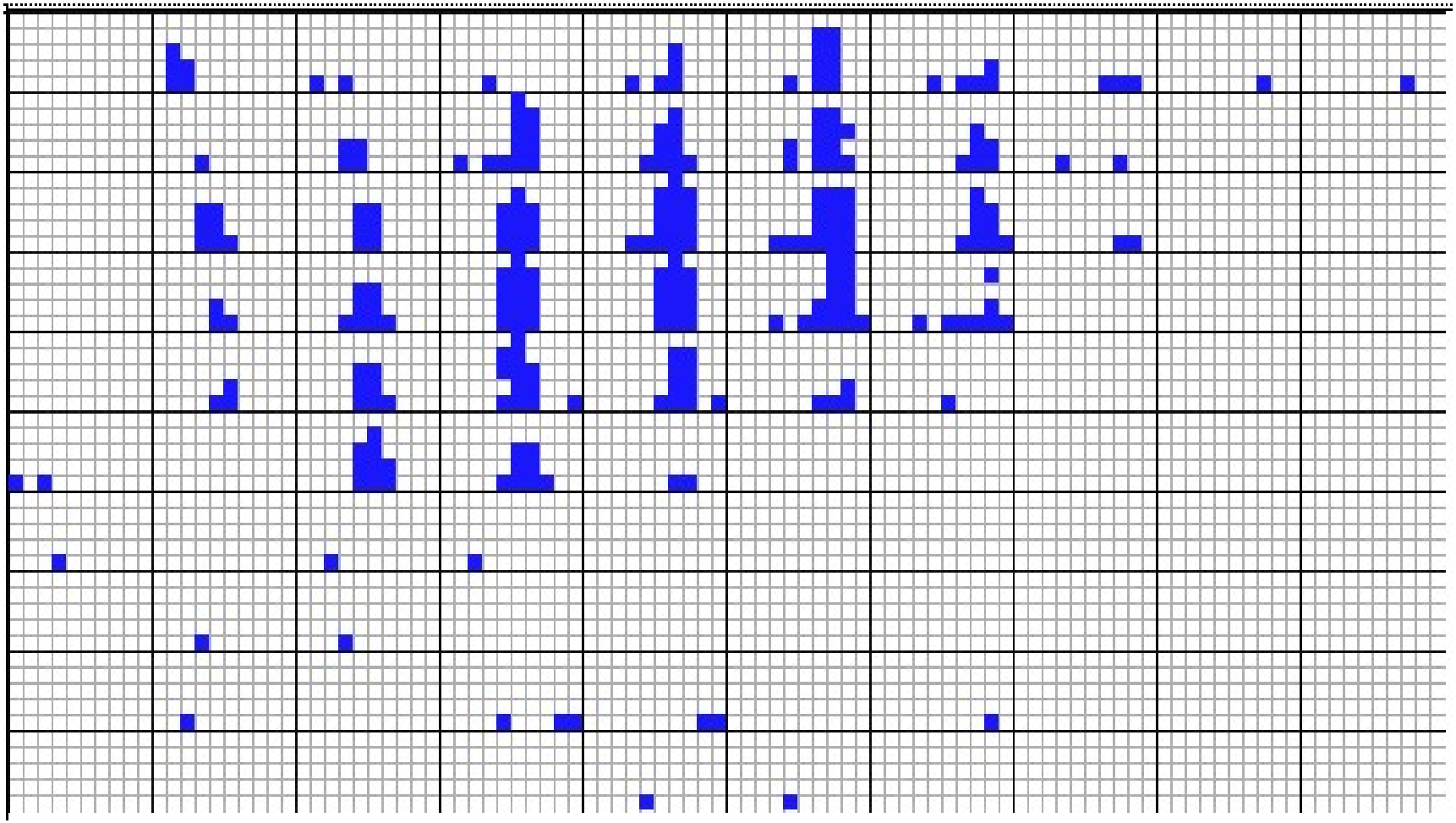
- Visualization of the data using a hierarchical partitioning into subspaces
  - Dimensionality stacking
  - Tag cloud

Dimensional Stacking



- Partitioning of the n-dimensional attribute space in 2-D subspaces, which are 'stacked' into each other
- Partitioning of the attribute value ranges into classes. The important attributes should be used on the outer levels.
- Adequate for data with ordinal attributes of low cardinality
- But, difficult to display more than nine dimensions
- Important to map dimensions appropriately

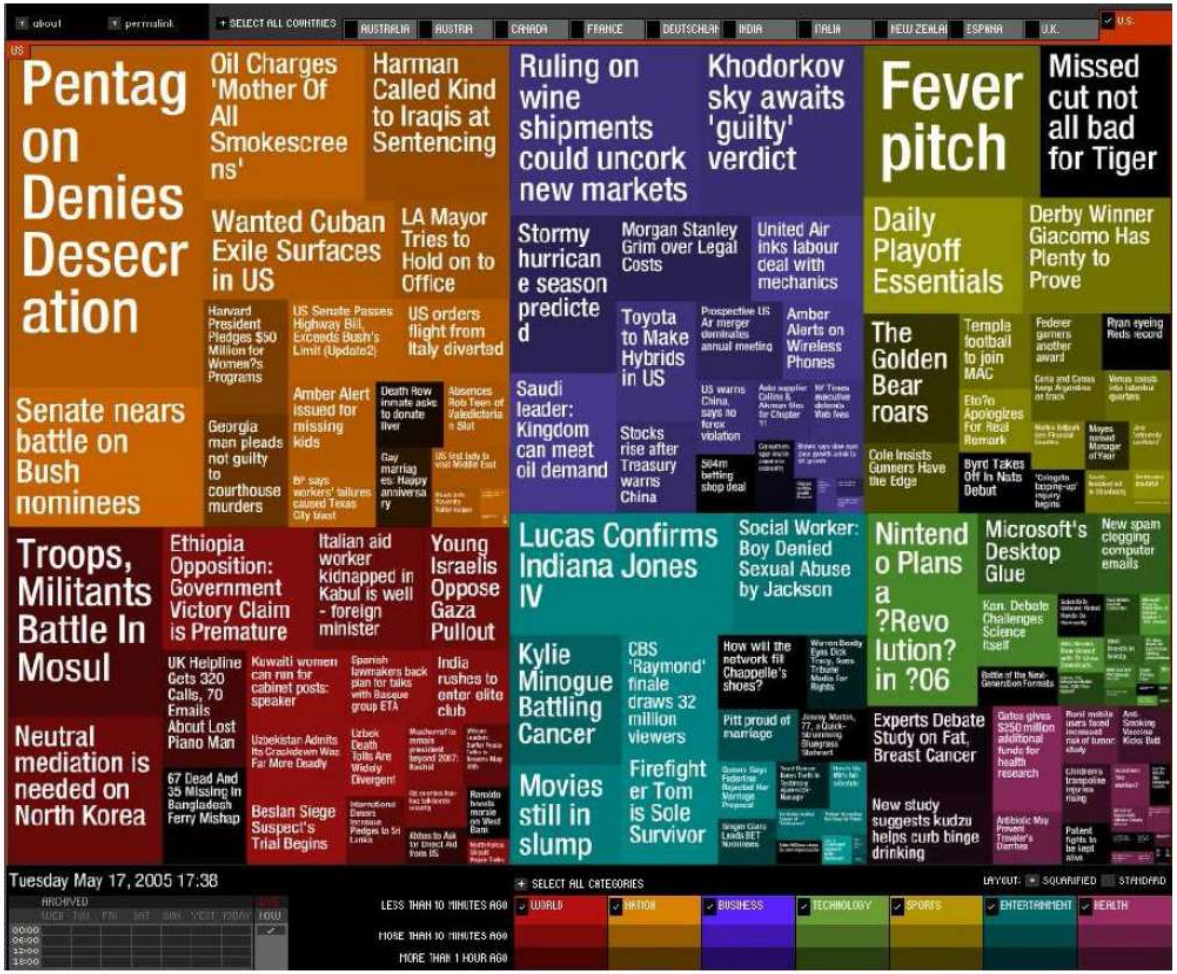
Dimensional Stacking



Visualization of oil mining data with longitude and latitude mapped to the outer x-, y-axes and ore grade and depth mapped to the inner x-, y-axes

Visualizing Complex Data and Relations

- Visualizing non-numerical data: text and social networks
- Tag cloud: visualizing user-generated tags
  - The importance of tag is represented by font size/color
- Besides text data, there are also methods to visualize relationships, such as visualizing social networks



Newsmap: Google News Stories in 2005

Presentation Outline

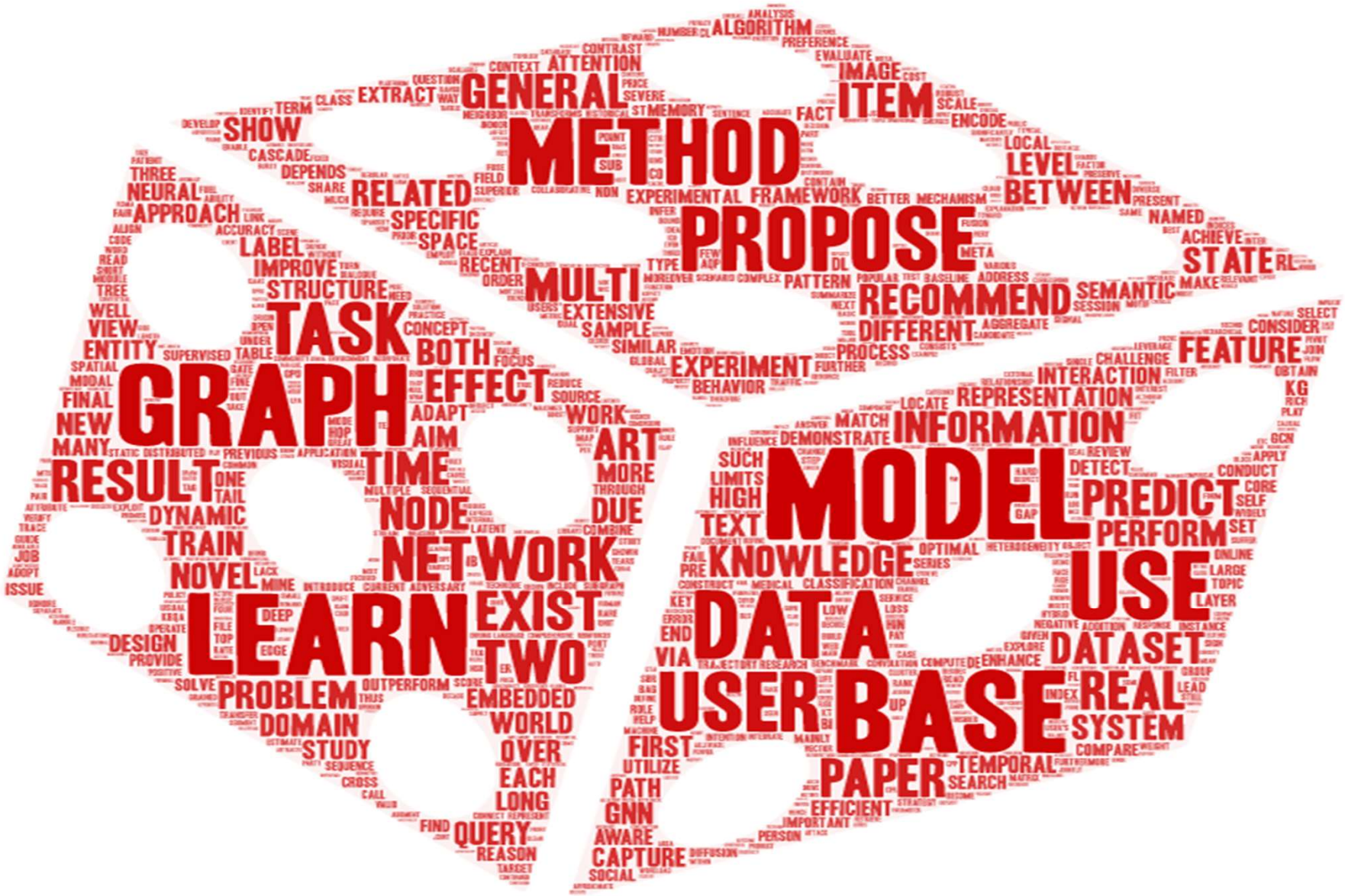
- Data mining functionalities
- Research Issues in Data mining
- Sources of data
- Data Objects and Attribute Types
- Basic Statistical Descriptions of Data
- Data Visualization
- **Case study**
- Summary

Case Study: Analysis of paper submissions to DASFAA-2022 conference

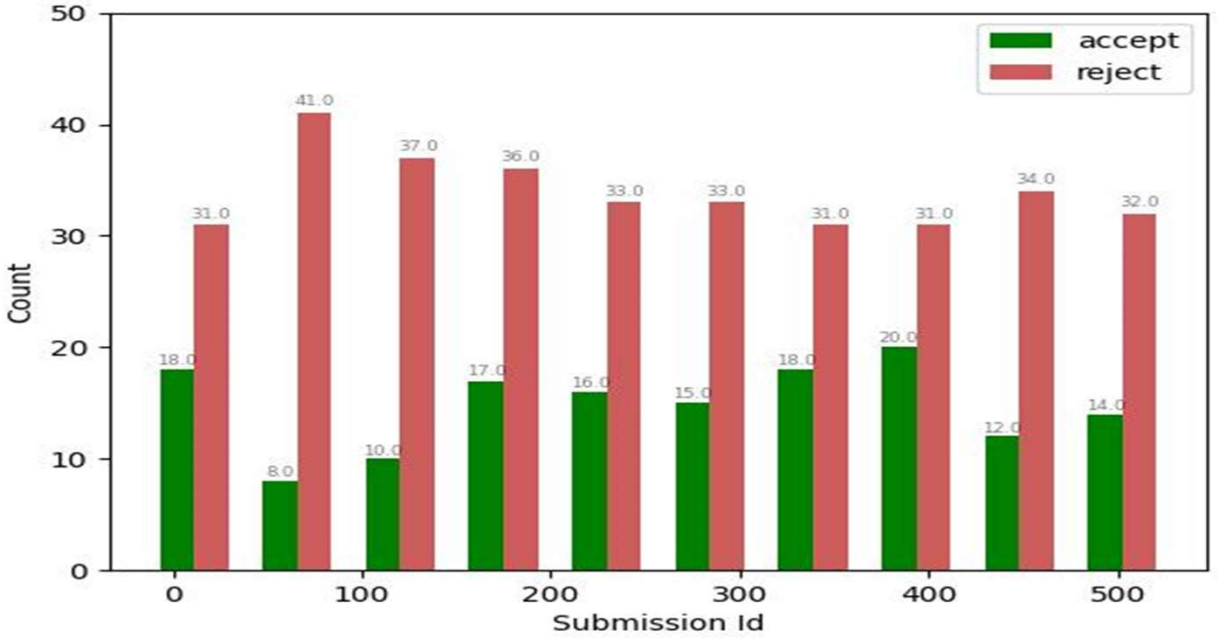
The 27th International Conference on Database Systems for Advanced Applications (DASFAA-2022), April 11-14, 2022, Hyderabad, India.

- Number of valid submissions (after desk-reject): 400
  - Yes, this is the number; no rounding off was done!
- Program committee
  - Reviewers: 205
  - Meta-Reviewers: 41
- Accept
  - Full: 72 (18%)
  - Short: 76 (19%)

Tag Cloud from Accepted Abstracts

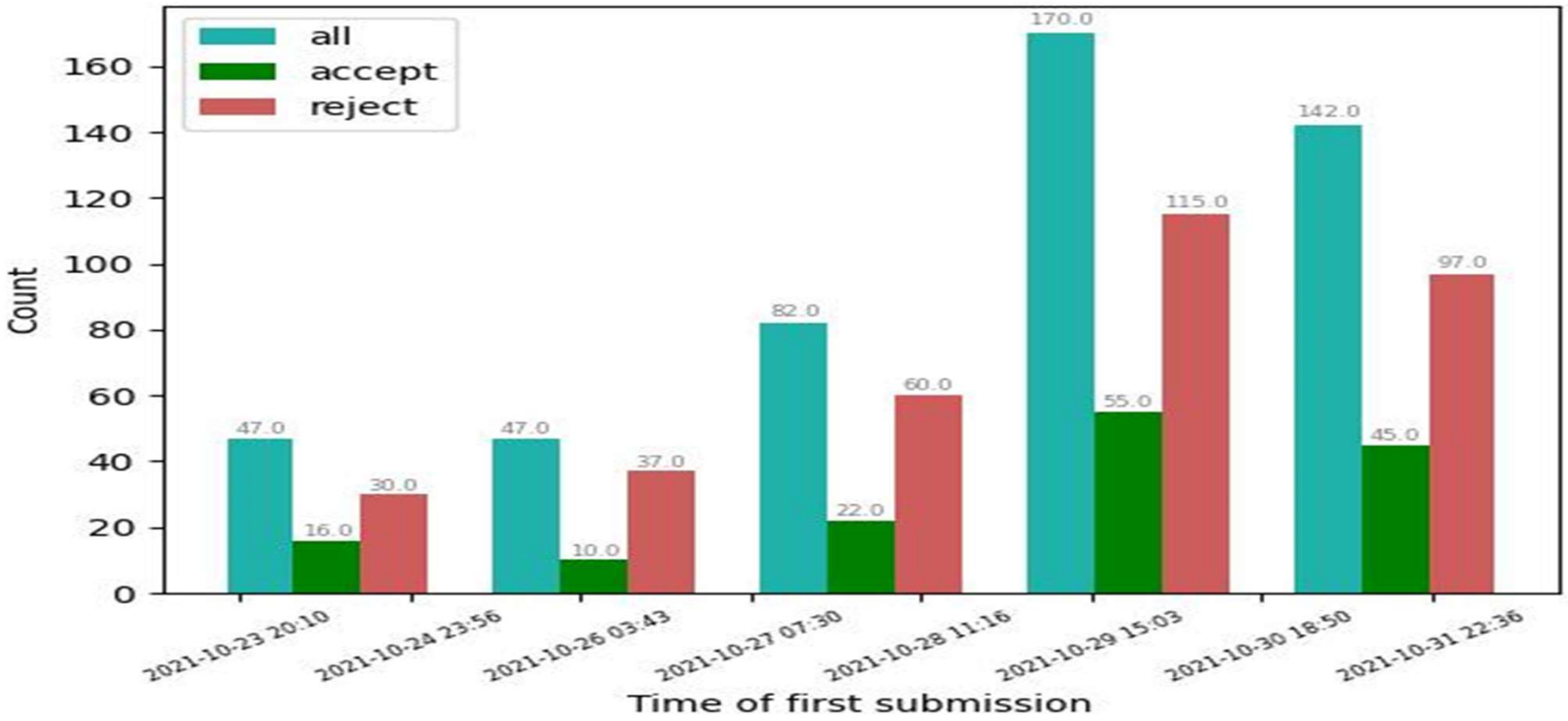


Does submission id matter?



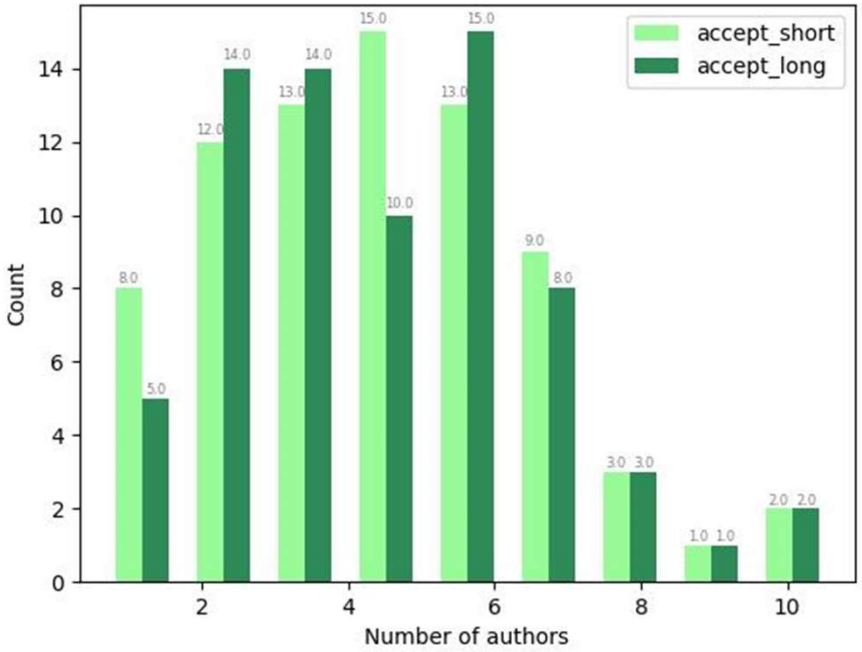
- Is there a sweet spot around 80 percentile?
- How will one know when to hit the 80 percentile?

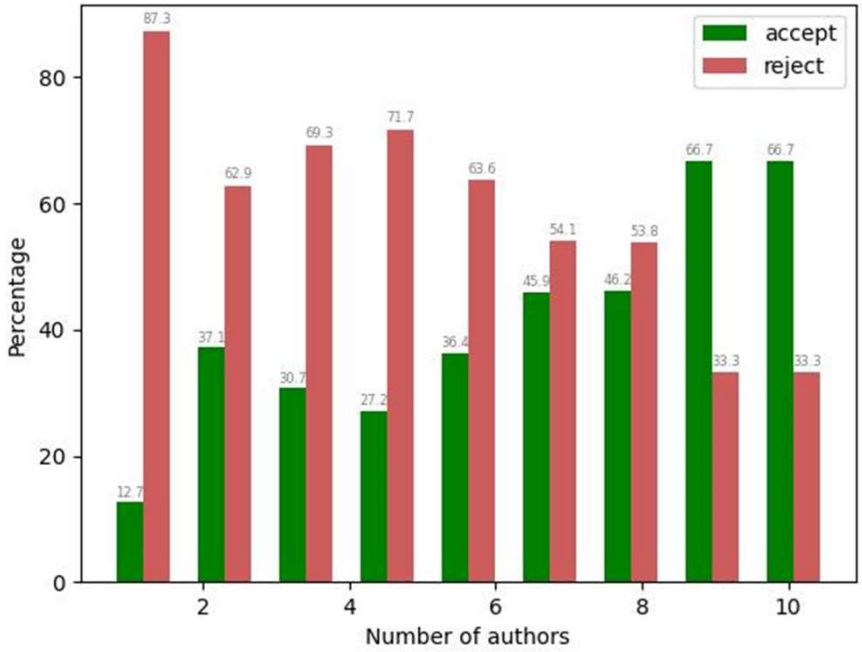
Time of first submission



- 1.5 to 2 days before the deadline!

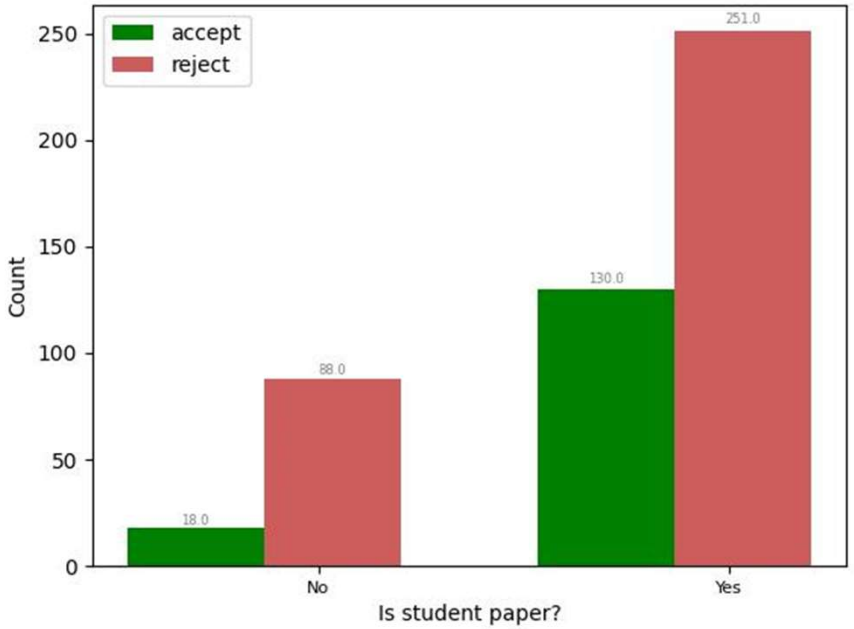
Does number of authors matter?

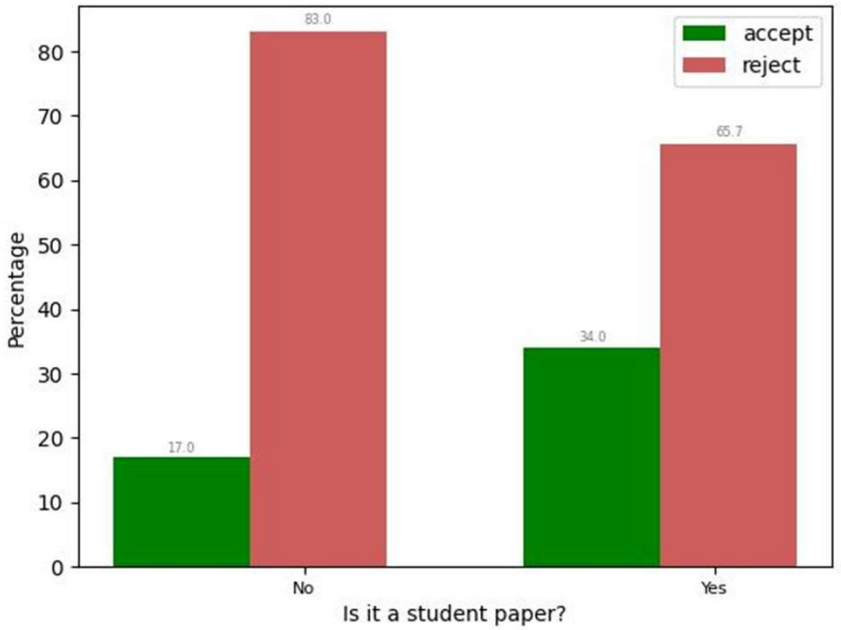




- The more the merrier – collaborations and multiple eyes matter!

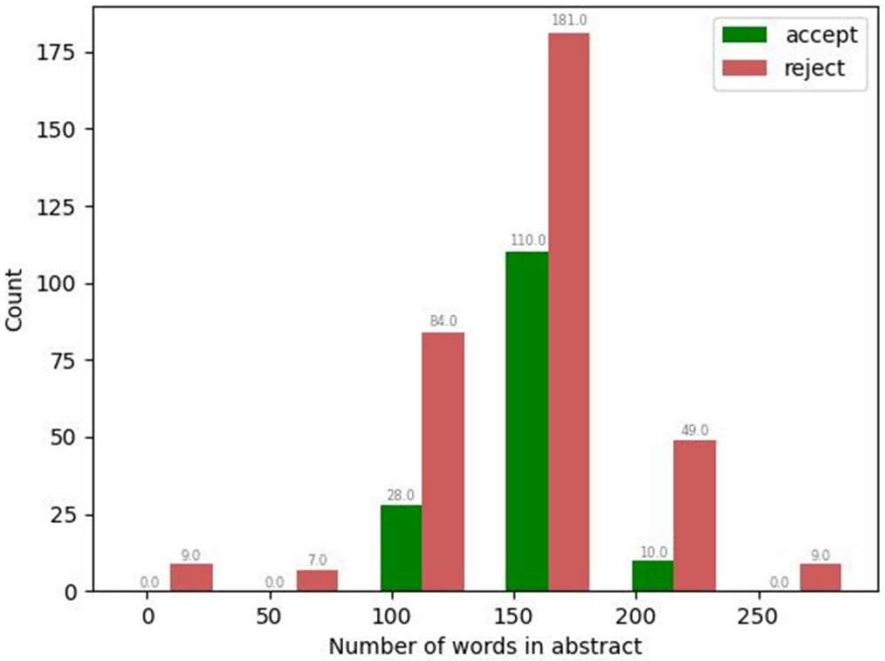
Student paper – first author is a student

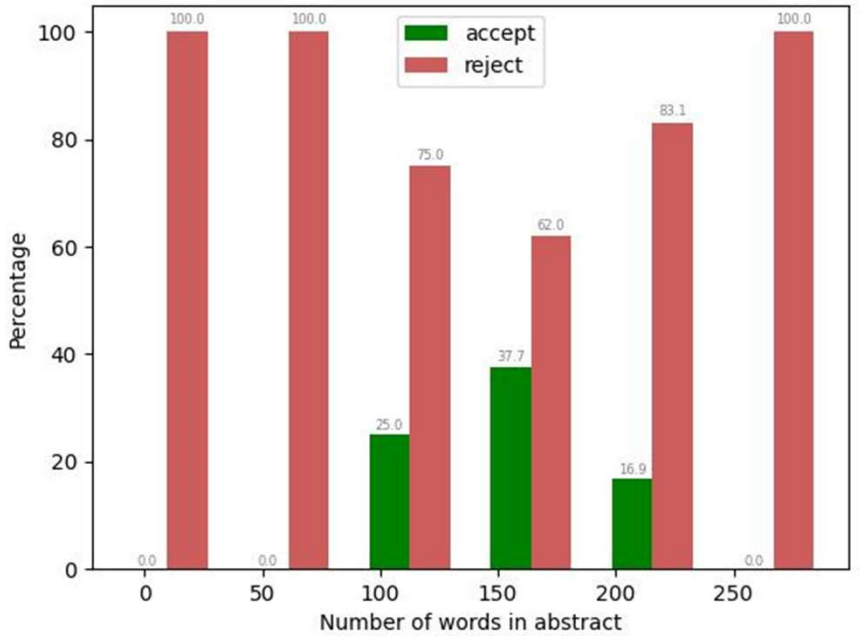




- Significant jump in acceptance percentage when a student is the first author – why?

Number of words in the abstract





- Too long or too short an abstract is not good

Other factors

- Other factors tried but no significant visual difference
- Number of words in the title
- Number of characters in the title
- ...
- ...

Summary

- Data attribute types: nominal, binary, ordinal, interval-scaled, ratio-scaled
- Many types of data sets, e.g., numerical, text, graph, Web, image.
- Gain insight into the data by:
  - Basic statistical data description: central tendency, dispersion, graphical displays
  - Data visualization: map data onto graphical primitives
  - Measure data similarity (not discussed)
- Above steps are the beginning of data preprocessing.
- Many methods have been developed but still an active area of research.

References

- W. Cleveland, Visualizing Data, Hobart Press, 1993
- T. Dasu and T. Johnson. Exploratory Data Mining and Data Cleaning. John Wiley, 2003
- U. Fayyad, G. Grinstein, and A. Wierse. Information Visualization in Data Mining and Knowledge Discovery, Morgan Kaufmann, 2001
- L. Kaufman and P. J. Rousseeuw. Finding Groups in Data: an Introduction to Cluster Analysis. John Wiley & Sons, 1990.
- H. V. Jagadish, et al., Special Issue on Data Reduction Techniques. Bulletin of the Tech. Committee on Data Eng., 20(4), Dec. 1997
- D. A. Keim. Information visualization and visual data mining, IEEE trans. on Visualization and Computer Graphics, 8(1), 2002
- D. Pyle. Data Preparation for Data Mining. Morgan Kaufmann, 1999
- S. Santini and R. Jain," Similarity measures", IEEE Trans. on Pattern Analysis and Machine Intelligence, 21(9), 1999
- E. R. Tufte. The Visual Display of Quantitative Information, 2nd ed., Graphics Press, 2001
- C. Yu , et al., Visual data mining of multimedia data for social and behavioral studies, Information Visualization, 8(1), 2009

Dissimilarity Computation and Data Preprocessing

P. Krishna Reddy

IIIT Hyderabad

Lecture #4

Detailed Syllabus

- **Introduction (1.5 hour): Definition, KDD framework, Issues in data mining.**
- **Data summarization (7.5 hrs): Data Types, Preprocessing,** Characterization, Discrimination, data warehousing techniques (Multidimensional data model, Data warehousing architecture, Data cube computation and OLAP technology)
- Concepts and algorithms for mining patterns and associations (9 hours) (Frequent item-set generation, A priori and FP-growth algorithm, Evaluation of Association patterns) and preprocessing
- Concepts and algorithms related to classification and regression (9hrs) (Overview, Decision tree induction, Over-fitting and under-fitting, Scalable decision tree algorithms, Bayesian Classification, Regression-based Prediction methods (9 hours)
- Concepts and algorithms for clustering the data (9 hours) (Overview, Types of Data, K-means, Aglomerative clustering, Clustering algorithms (DBSCAN, BIRCH, CURE, ROCK, CHAMELEON)).
- Outlier analysis and future trends (graph mining, spatio-temporal mining). (3 hours)

Outline

- Proximity computation measures

Proximity: Similarity or Dissimilarity

- So far we have analyzed the values of single attribute
  - Central tendency, dispersion, and spread of some attribute X.
- Objects are described by multiple attributes.
- **Similarity**
  - Numerical measure of how alike two data objects are
    - How to compare two persons or two objects?
  - Value is higher when objects are more alike
  - Often falls in the range [0,1]
- **Dissimilarity** (e.g., distance)
  - Numerical measure of how different two data objects are
  - Lower when objects are more alike
  - Minimum dissimilarity is often 0
  - Upper limit varies
- **Proximity** refers to a similarity or dissimilarity

Dissimilarity/Similarity metric

- Dissimilarity/Similarity metric
  - Similarity is expressed in terms of a distance function, typically metric: $d(i, j)$
  - The definitions of distance functions are usually rather different for interval-scaled, boolean, categorical, ordinal ratio, and vector variables
  - Weights should be associated with different variables based on applications and data semantics
- sim(i,j)= 1-d(i,j)

Metric

- A distance that satisfies the following properties is a metric
  - Non-negativity: $d(i, j) > 0$ if $i \neq j$, and $d(i, i) = 0$ (Positive definiteness)
    - Distance is non-negative number
  - Identify: $d(i,j) >= 0$
    - The distance of an object to itself is 0
  - $d(i, j) = d(j, i)$ (Symmetry)
    - Distance is a symmetric function
  - $d(i, j) \leq d(i, k) + d(k, j)$ (Triangle Inequality)

Data Matrix and Dissimilarity Matrix

- Data matrix
  - n data points with p dimensions

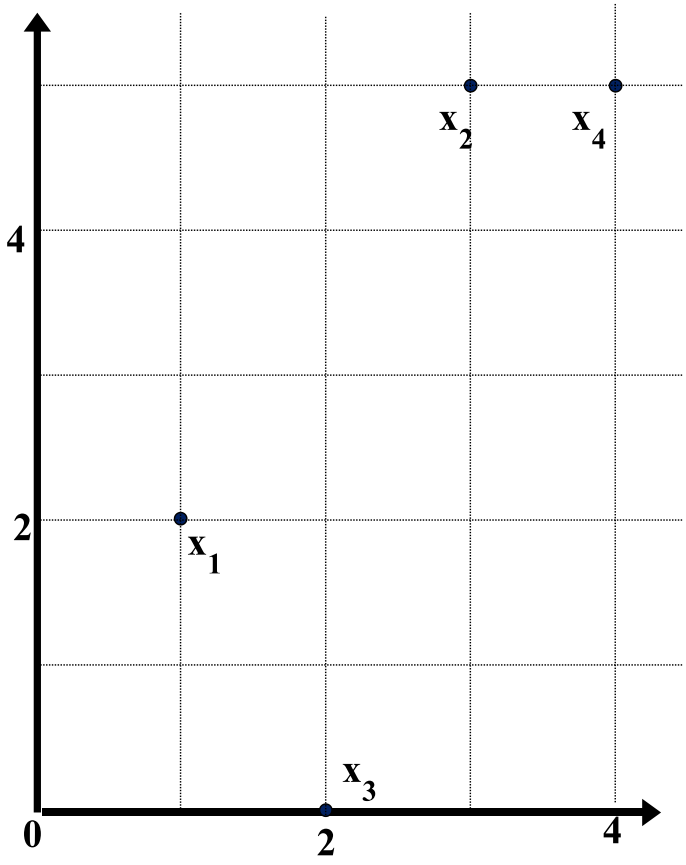
$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

- Dissimilarity matrix
  - n data points, but registers only the distance
  - A triangular matrix
  - Single mode

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

- Most clustering and nearest neighbor algorithms operate on dissimilarity matrix

Example:
Data Matrix and Dissimilarity Matrix



Data Matrix

point	attribute1	attribute2
x1	1	2
x2	3	5
x3	2	0
x4	4	5

Dissimilarity Matrix

(with Euclidean Distance)

	x1	*x2*	*x3*	*x4*
x1	0			
x2	3.61	0		
x3	5.1	5.1	0	
x4	4.24	1	5.39	0

Types of Attributes

- Nominal attributes
- Binary attributes
- Numeric data
- Ordinal data
- Mixed data

Proximity Measure for Nominal Attributes

- Can take 2 or more states
  - Example: Map colour (red, yellow, blue, green (generalization of a binary attribute))
- Let i and j are objects with p variables
- Method 1: Simple matching
  - *m*: # of matches, *p*: total # of variables
  - The dissimilarity

$$d(i, j) = \frac{p - m}{p}$$

- Method 2: Use a large number of binary attributes
  - creating a new binary attribute for each of the *M* nominal states

Proximity Measure for Binary Attributes

- A contingency table for binary data
  - q: # of attributes equal to 1 in i and j.
  - r: # of attributes equal to 1 in i but equal to 0 in j
  - s :# of attributes equal to 0 in i but equal to 1 in j
  - t: # of attributes that equal to 0 for both i and j
- Distance measure for symmetric binary variables:
- Distance measure for asymmetric binary variables:
- Jaccard coefficient (*similarity* measure for *asymmetric* binary variables): 1-d(i,j)

Object *i* \ Object *j*	1	0	sum
1	q	r	$q+r$
0	s	t	$s+t$
sum	$q+s$	$r+t$	p

$$d(i,j) = \frac{r+s}{q+r+s+t}$$

$$d(i,j) = \frac{r+s}{q+r+s}$$

$$sim_{Jaccard}(i,j) = \frac{q}{q+r+s}$$

- Note: Jaccard coefficient is the same as "coherence":

$$coherence(i,j) = \frac{sup(i,j)}{sup(i)+sup(j)-sup(i,j)} = \frac{q}{(q+r)+(q+s)-q}$$

Example: Dissimilarity between Asymmetric Binary Variables

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	P	N	N	N	N

- Gender is a symmetric attribute (not counted in)
- The remaining attributes are asymmetric binary
- Let the values Y and P be 1, and the value N be 0
- Distance: $d(i, j) = \frac{r + s}{q + r + s}$

$$d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$$

$$d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$$

$$d(jim, mary) = \frac{1+2}{1+1+2} = 0.75$$

Mary

Jack	1	0	$\sum_{row}$
1	2	0	2
0	1	3	4
$\sum_{col}$	3	3	6

Jim

Jack	1	0	$\sum_{row}$
1	1	1	2
0	1	3	4
$\sum_{col}$	2	4	6

Mary

Jim	1	0	$\sum_{row}$
1	1	1	2
0	2	2	4
$\sum_{col}$	3	3	6

Distance on Numeric Data: Minkowski Distance

- *Minkowski distance*: A popular distance measure

$$d(i, j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \cdots + |x_{ip} - x_{jp}|^h}$$

where $i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \ldots, x_{jp})$ are two p-dimensional data objects, and h is the order (the distance so defined is also called L-h norm)

Special Cases of Minkowski Distance

- h = 1: Manhattan (city block, L_1 norm) distance
  - E.g., the Hamming distance: the number of bits that are different between two binary vectors

$$d(i,j) = |x_{i_1} - x_{j_1}| + |x_{i_2} - x_{j_2}| + \ldots + |x_{i_p} - x_{j_p}|$$

- h = 2: (L_2 norm) Euclidean distance

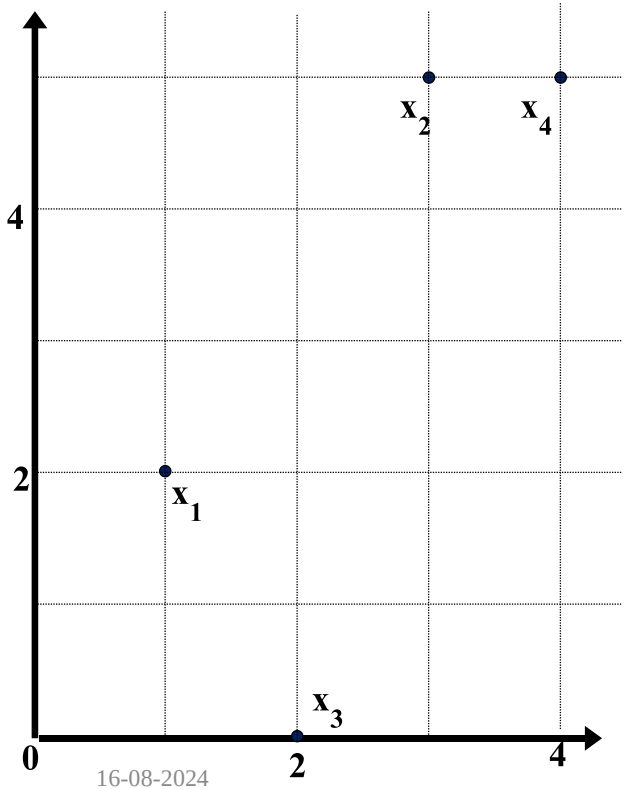
$$d(i,j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + \ldots + |x_{i_p} - x_{j_p}|^2)}$$

- $h \to \infty$. "supremum" (L_{max} norm, L_∞ norm) distance.
  - This is the maximum difference between any component (attribute) of the vectors

$$d(i,j) = \lim_{h \to \infty} \left(\sum_{f=1}^{p} |x_{if} - x_{jf}|^h \right)^{\frac{1}{h}} = \max_{f}^{p} |x_{if} - x_{jf}|$$

Example: Minkowski Distance

point	attribute 1	attribute 2
x1	1	2
x2	3	5
x3	2	0
x4	4	5



Dissimilarity Matrices

Manhattan (L_1)

L	x1	x2	x3	x4
x1	0			
x2	5	0		
x3	3	6	0	
x4	6	1	7	0

Euclidean (L_2)

L2	x1	x2	x3	x4
x1	0			
x2	3.61	0		
x3	2.24	5.1	0	
x4	4.24	1	5.39	0

Supremum

L_∞	x1	x2	x3	x4
x1	0			
x2	3	0		
x3	2	5	0	
x4	3	1	5	0

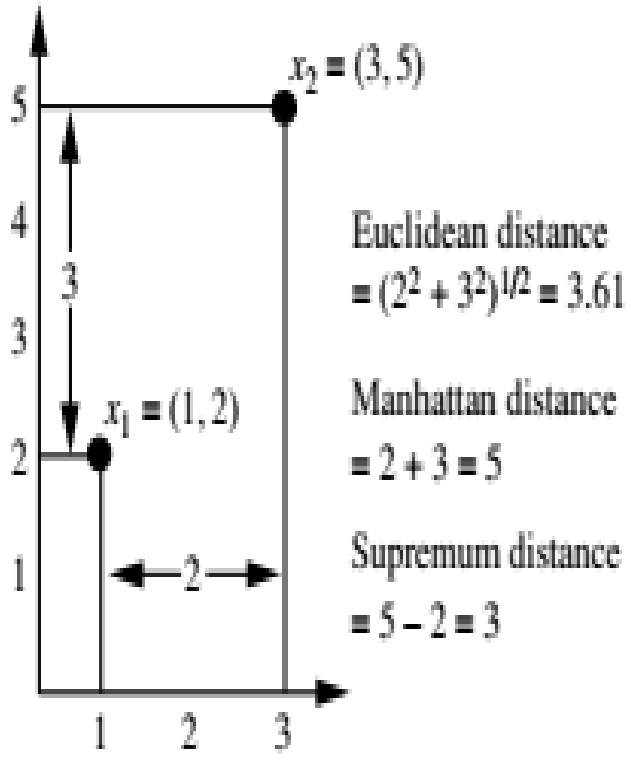


Figure 2.23 Euclidean, Manhattan, and supremum distances between two objects.

Standardizing Numeric Data

- Z-score: $z = \frac{x - \mu}{\sigma}$
  - x: raw score to be standardized, μ: mean of the population, σ: standard deviation
  - the distance between the raw score and the population mean in units of the standard deviation
  - negative when the raw score is below the mean, "+" when above
- An alternative way: Calculate the mean absolute deviation

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + \ldots + |x_{nf} - m_f|)$$

where

$$m_f = \frac{1}{n}(x_{1f} + x_{2f} + \ldots + x_{nf})$$

  - standardized measure (*z-score*): $z_{if} = \frac{x_{if} - m_f}{s_f}$
- Using mean absolute deviation is more robust than using standard deviation

Ordinal Variables

- An ordinal variable can be discrete or continuous
- Order is important, e.g., rank
- Can be treated like interval-scaled
  - replace x_{if} by their rank $r_{if} \in \{1, \ldots, M_f\}$
  - map the range of each variable onto [0, 1] by replacing *i*-th object in the *f*-th variable by

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

  - compute the dissimilarity using methods for interval-scaled variables

Attributes of Mixed Type

- A dataset may contain all attribute types
  - Nominal, symmetric binary, asymmetric binary, numeric, and ordinal
- One may use a weighted formula to combine their effects:

$$d(i, j) = \frac{\sum_{f=1}^{p} w_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} w_{ij}^{(f)}}$$

  - If f is numeric: Use the normalized distance
  - If f is binary or nominal: $d_{ij}^{(f)} = 0$ if $x_{if} = x_{jf}$; or $d_{ij}^{(f)} = 1$ otherwise
  - If f is ordinal
    - Compute ranks z_{if} (where $z_{if} = \frac{r_{if} - 1}{M_f - 1}$)
    - Treat z_{if} as interval-scaled

Cosine Similarity

- A **document** can be represented by thousands of attributes, each recording the *frequency* of a particular word (such as keywords) or phrase in the document.

Document	*team*	*coach*	*hockey*	*baseball*	*soccer*	*penalty*	*score*	*win*	*loss*	*season*
Document1	5	0	3	0	2	0	0	2	0	0
Document2	3	0	2	0	1	1	0	1	0	1
Document3	0	7	0	2	1	0	0	3	0	0
Document4	0	1	0	0	1	2	2	0	3	0

- Other vector objects: gene features in micro-arrays, …
- Applications: information retrieval, biologic taxonomy, gene feature mapping, ...
- Cosine measure: If d_1 and d_2 are two vectors (e.g., term-frequency vectors), then

  $\cos(d_1, d_2) = (d_1 \bullet d_2) / \|d_1\| \, \|d_2\|$,

  where $\bullet$ indicates vector dot product, $\|d\|$: the length of vector d

Example: Cosine Similarity

- $\cos(d_1, d_2) = (d_1 \bullet d_2) / \|d_1\| \|d_2\|$,
  where $\bullet$ indicates vector dot product, $\|d\|$: the length of vector d

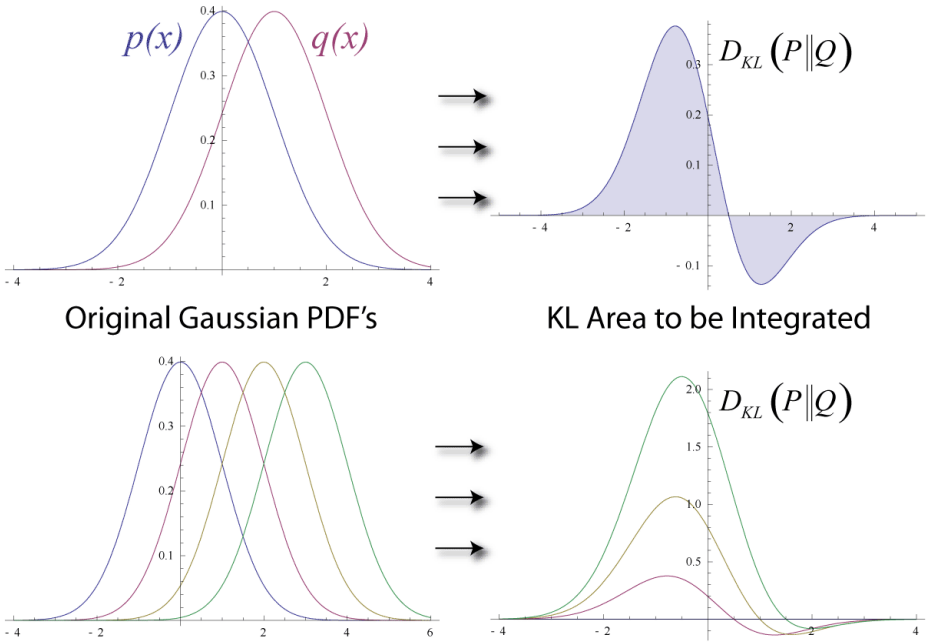
- Ex: Find the **similarity** between documents 1 and 2.

$d_1 = (5, 0, 3, 0, 2, 0, 0, 2, 0, 0)$
$d_2 = (3, 0, 2, 0, 1, 1, 0, 1, 0, 1)$

$d_1 \bullet d_2 = 5*3+0*0+3*2+0*0+2*1+0*1+0*1+2*1+0*0+0*1 = 25$
$\|d_1\| = (5*5+0*0+3*3+0*0+2*2+0*0+0*0+2*2+0*0+0*0)^{0.5} = (42)^{0.5} = 6.481$
$\|d_2\| = (3*3+0*0+2*2+0*0+1*1+1*1+0*0+1*1+0*0+1*1)^{0.5} = (17)^{0.5} = 4.12$
$\cos(d_1, d_2) = 0.94$

KL Divergence: Comparing Two Probability Distributions

- *The Kullback-Leibler (KL) divergence:* Measure the difference between two probability distributions over the same variable *x*
  - From information theory, closely related to *relative entropy*, *information divergence*, and *information for discrimination*
- $D_{KL}(p(x) \parallel q(x))$: divergence of $q(x)$ from $p(x)$, measuring the information lost when $q(x)$ is used to approximate $p(x)$



$$D_{KL}(p(x)||q(x)) = \sum_{x \in X} p(x) \ln \frac{p(x)}{q(x)}$$

Discrete form

$$D_{KL}(p(x)||q(x)) = \int_{-\infty}^{\infty} p(x) \ln \frac{p(x)}{q(x)} dx$$

Continuous form

Ack.: Wikipedia entry: *The Kullback-Leibler (KL) divergence*

More on KL Divergence

$$D_{KL}(p(x)||q(x)) = \sum_{x \in X} p(x) \ln \frac{p(x)}{q(x)}$$

- The KL divergence measures the expected number of extra bits required to code samples from $p(x)$ ("true" distribution) when using a code based on $q(x)$, which represents a theory, model, description, or approximation of $p(x)$
- The KL divergence is not a distance measure, not a metric: asymmetric, not satisfy triangular inequality ($D_{KL}(P\|Q)$ does not equal $D_{KL}(Q\|P)$)
- In applications, P typically represents the "true" distribution of data, observations, or a precisely calculated theoretical distribution, while Q typically represents a theory, model, description, or approximation of P.
- The Kullback–Leibler divergence from Q to P, denoted $D_{KL}(P\|Q)$, is a measure of the information gained when one revises one's beliefs from the prior probability distribution Q to the posterior probability distribution P. In other words, it is the amount of information lost when Q is used to approximate P.
- The KL divergence is sometimes also called the information gain achieved if P is used instead of Q. It is also called the relative entropy of P with respect to Q.

KDD process
Data preprocessing is a major step

- Learning the application domain:
  - relevant prior knowledge and goals of application
- Creating a target data set: data selection
- **Preprocessing: (may take 60% of effort!)**
- **Data reduction and transformation:**
  - **Find useful features, dimensionality/variable reduction, invariant representation.**
- Choosing functions of data mining
  - summarization, classification, association, clustering.
- Choosing the mining algorithm(s)
- Data mining: search for patterns of interest
- Pattern evaluation and knowledge presentation
  - visualization, transformation, removing redundant patterns, etc.
- Use of discovered knowledge

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 may come from
  - "Not applicable" data value when collected
  - Different considerations between the time when the data was collected and when it is analyzed.
  - Human/hardware/software problems
- Noisy data (incorrect values) may come from
  - Faulty data collection instruments
  - Human or computer error at data entry
  - Errors in data transmission
- Inconsistent data may come from
  - Different data sources
  - Functional dependency violation (e.g., modify some linked data)
- Duplicate records also need data cleaning

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

Data Preprocessing

- Data Preprocessing: An Overview
  - **Data Quality**
  - Major tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- Data Transformation
- Dimensionality Reduction
- Summary

Data Quality: Why Preprocess the Data?

- Measures for data quality: A multidimensional view
  - Accuracy: correct or wrong, accurate or not
  - Completeness: not recorded, unavailable, …
  - Consistency: some modified but some not, dangling, …
  - Timeliness: timely update?
  - Believability: how trustable the data are correct?
  - Interpretability: how easily the data can be understood?

Example

- Consider AllElectronics company
  - You want to analyze <item, price, units_sold>
  - Observations
    - Do not have values for several attributes
    - You want to know whether the item was advertised or not, but you find that it was not recorded
    - There are errors, unusual values, data inconsistencies
  - It means
    - Data is incomplete (lacking required attributes)
    - Inaccurate or noisy (containing errors or deviate from the expected values)
    - Inconsistent (containing discrepancies in the department codes used to categorize items)
      - *Age*="42", *Birthday*="03/07/2010"
      - Was rating "1, 2, 3", now rating "A, B, C"
      - discrepancy between duplicate records

- accuracy, completeness and consistency are common-place properties of real world databases
- Reasons
  - Inaccuracy
    - Data collection instruments may be faulty
    - Human or computer errors in data entry
    - Users purposefully submit incorrect data intentionally for mandatory fields
      - By choosing January 1 by default
    - Data transmission errors
    - Buffer size limitation
    - Naming conventions for data codes and different formats

- Reasons for incomplete data
  - Attributes of interest may not be available
    - They were not considered important at the time of entry.
    - Relevant data may not be recorded due to misunderstanding or because of equipment malfunctions
    - Data may be deleted
    - Missing values of the attributes may not be inferred.
- Data quality depends on the intended use of data
  - If the 80% of addresses are correct
    - It is OK for marketing manager
    - But, may not be Ok for sales manager

Timeliness, Believability, Interpretability

- Timeliness: Consider a giving monthly bonuses to top sales representatives
  - If the data is not updated in a monthly fashion has a negative impact on the data quality
- Believability
  - How much the data is trused by users
  - Several errors many lead to lack of trust
- Interpretability
  - How easy the data are understood
  - For example, the data may use several codes; difficult to interpret.

Data Preprocessing

- Data Preprocessing: An Overview
  - Data Quality
  - **Major tasks in Data Preprocessing**
- Data Cleaning
- Data Integration
- Data Transformation
- Dimensionality Reduction
- Summary

Major Tasks in Data Preprocessing

- **Data cleaning**
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
  - Apply data cleaning techniques
- **Data integration**
  - Integration of multiple databases, data cubes, or files
  - Examples:
  - 1. customer identification may be referred as customer-id in one database and cust_id in another database
  - 2. first name may be Bill on one database and William in another database
  - How to create a common database by integrating Facebook and LinkedIn users?

Major Tasks in Data Preprocessing

- **Data reduction**
  - Question: Data set is very huge. How to reduce the data set size without jeopardizing the data mining results?
  - Dimensionality reduction
    - Compressed representation
  - Numerosity reduction
    - Regression or log-linear models
  - Data compression
- **Data transformation and data discretization**
  - Better results if the data is transformed, i.e., normalized
  - Normalization: Example age and salary: transform all age and salary values between 0 and 1.
  - Discretization: mining data at different abstraction levels
  - Concept hierarchy generation

Data Preprocessing

- Data Preprocessing: An Overview
  - Data Quality
  - Major tasks in Data Preprocessing
- **Data Cleaning**
- Data Integration
- Data Transformation
- Dimensionality Reduction
- Summary

How to Handle Missing Data?

- Method 1: Ignore the tuple:
  - usually done when class label is missing (when doing classification)—not effective when the % of missing values per attribute varies considerably
- Method 2: Fill in the missing value manually: tedious + infeasible?
- Method 3: Use a global constant to fill the missing value
  - a global constant : e.g., "unknown", a new class?!
- Method 4: fill with the attribute mean/ median/mode
- Method 5: fill with the attribute mean/ median/mode belong to the same class: Smarter
- Method 6: the most probable value: inference-based such as Bayesian formula or decision tree-
  - Popular strategy

How to Handle Noisy Data?

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

How to Handle Noisy Data?

- Method 1: Binning: consult neighbourhood
  - first sort data and partition into (equal-frequency) bins
  - then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.
- Method 2: Regression
  - smooth by fitting the data into regression functions (will discuss later)
- Method 3: Clustering
  - detect and remove outliers
- Combined computer and human inspection
  - detect suspicious values and check by human (e.g., deal with possible outliers)

Simple Discretization Methods: Binning

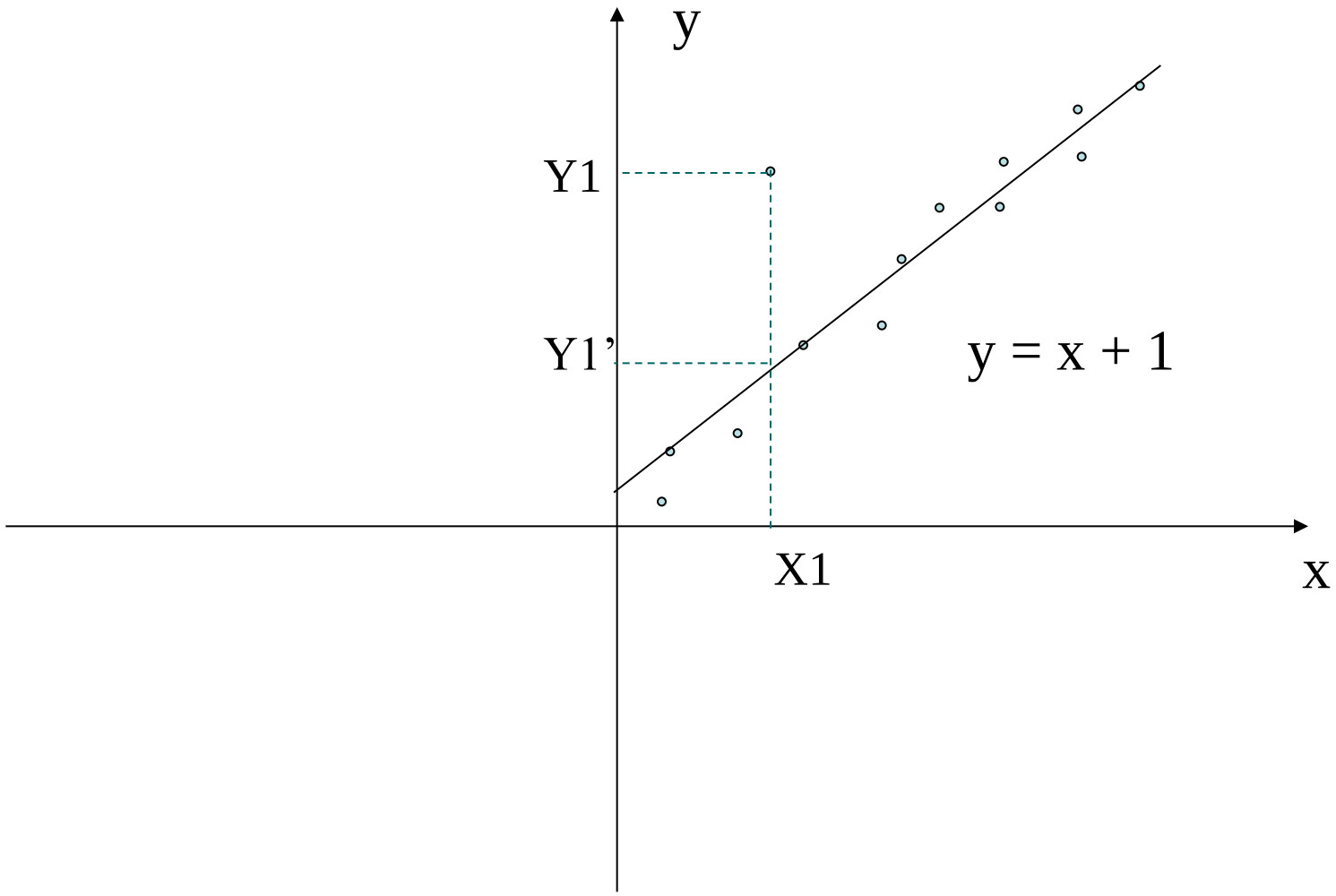
- Equal-width (distance) partitioning
  - Divides the range into *N* intervals of equal size: uniform grid
  - if *A* and *B* are the lowest and highest values of the attribute, the width of intervals will be: $W = (B - A)/N$.
  - The most straightforward, but outliers may dominate presentation
  - Skewed 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

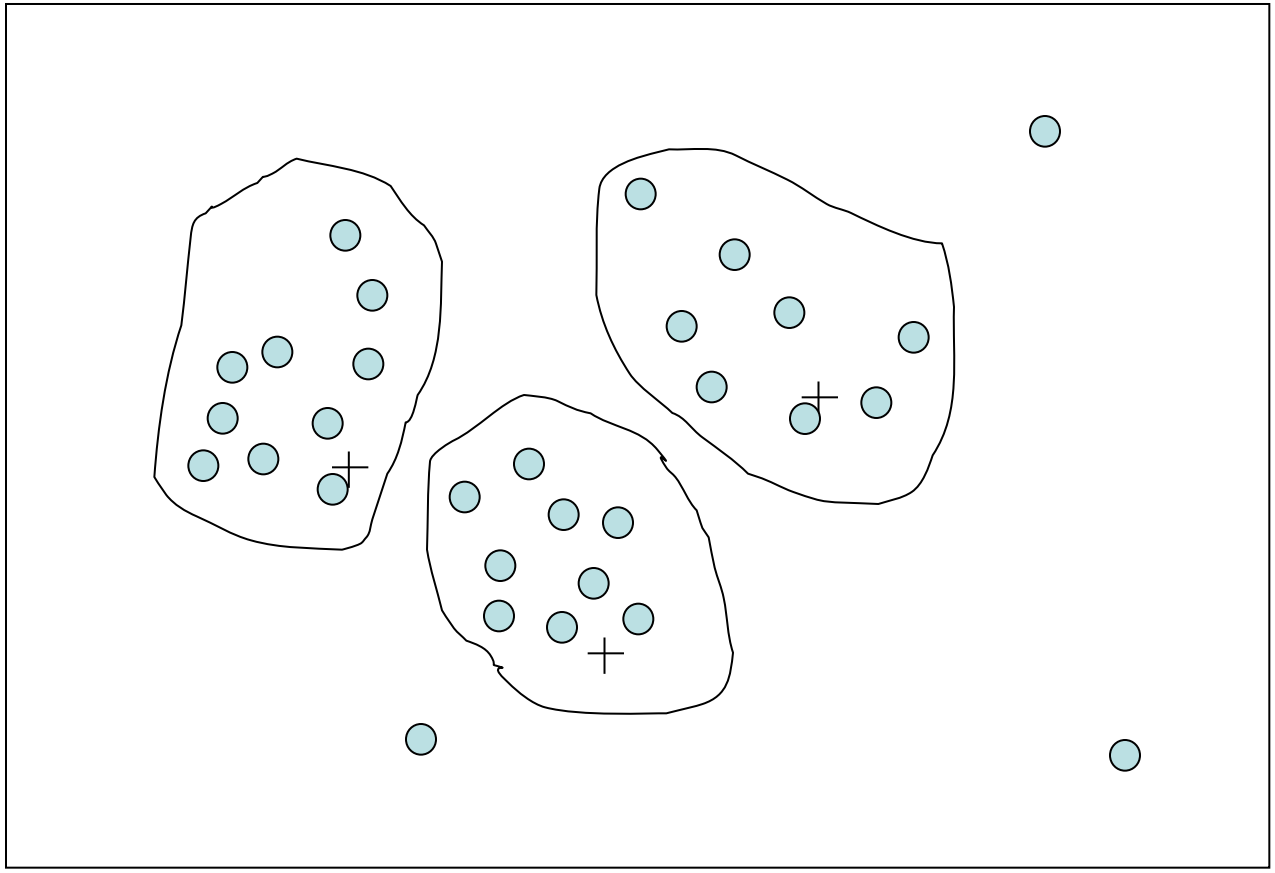
- Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34

* Partition into equal-frequency (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

Regression



Cluster Analysis



Lecture #5

Data Cleaning as a Process

- Step 1: Data discrepancy detection
- Step 2: Data transformation.

Data Cleaning as a Process
Step 1: data discrepancy detection

- Reasons
  - Poorly designed data entry forms
  - Deliberate errors
  - Data decay (outdated addresses)
  - Inconsistent data codes
  - The same attribute may have different names
- Use the following techniques
- Use metadata (e.g., domain, range, dependency, distribution)
  - Use any knowledge you have about the data
    - Acceptable values of each attribute, range of attribute
    - Write your script
      - Identify anomalies and trends using basic statistics
- Check Field overloading
    - Developers squeeze new attribute definitions to unused portions of already used attributes

Data Cleaning as a Process
Step 1: data discrepancy detection (contd.)

- Check the following rules
  - Uniqueness rule
    - Each value of the attribute is different from other value
  - Consecutive rule
    - There is no missing value between minimum and maximum
      - Example check numbers
  - Null rule
    - Use of blanks, question marks, special characters or strings
    - Example: License number is kept blank by non-drivers
- Use commercial tools
  - Data scrubbing: use simple domain knowledge (e.g., postal code, spell-check) to detect errors and make corrections
  - Data auditing: by analyzing data to discover rules and relationship to detect violators (e.g., correlation and clustering to find outliers)

Data Cleaning as a Process
Step 2: data transformation

- Data migration
  - Data migration tools: allow transformations to be specified
    - Example: replacing gender with sex
  - ETL (Extraction/Transformation/Loading) tools: allow users to specify transformations through a graphical user interface
- Integration of the two processes (discrepancy detection and transformation)
  - Iterative and interactive (e.g., Potter's Wheels tool)
  - https://control.cs.berkeley.edu/abc/
- Important aspect
  - Update the meta-data whenever you modify data

Data Preprocessing

- Data Preprocessing: An Overview
  - Data Quality
  - Major tasks in Data Preprocessing
- Data Cleaning
- **Data Integration**
- Data Transformation
- Dimensionality Reduction
- Summary

Data Integration

- **Data integration**:
  - Combines data from multiple sources into a coherent store
- Schema integration: e.g., A.cust-id ≡ B.cust-#
  - Integrate metadata from different sources
- *Sample research paper: Guoliang Li: Human-in-the-loop Data Integration. Proc. VLDB Endow. 10(12): 2006-2017 (2017)*

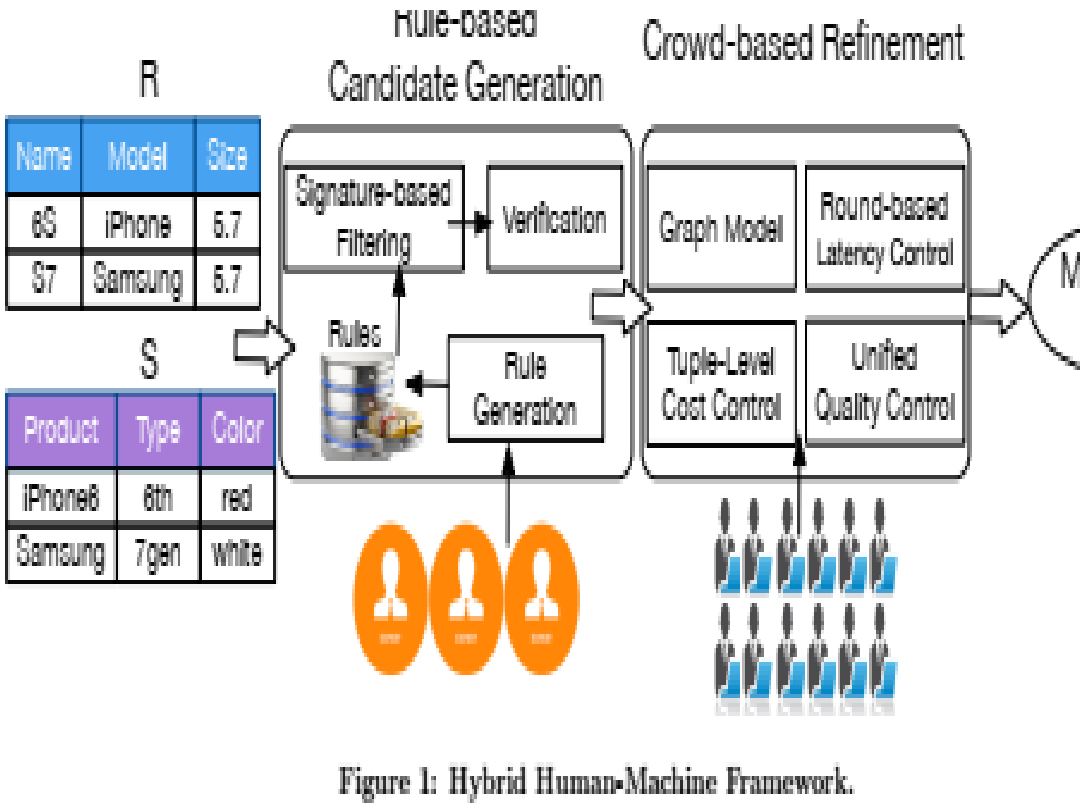


Figure 1: Hybrid Human-Machine Framework.

Data Integration: Entity identification problem:

- Identify real world entities from multiple data sources,
  - e.g., Bill Clinton = William Clinton
- Structure of the data
  - Functional dependencies and referential integrity constraints should match
  - Example: in one data, discount may be on entire order and in another data discount may be on line item
- 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

Data integration: Handling Redundancy in Data Integration

- Redundant data occur often when integration of multiple databases
  - *Object identification*: The same attribute or object may have different names in different databases
  - *Derivable data:* One attribute may be a "derived" attribute in another table, e.g., annual revenue
- Redundant attributes may be able to be detected by *correlation analysis* and *covariance analysis*
- Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality

Correlation Analysis (Nominal Data)

- Suppose *A* has *c* distinct values, namely *a*1,*a*2,..,*ac*. *B* has *r* distinct values, namely *b*1,*b*2,…,*br*.
- The data tuples described by *A* and *B* can be shown as a **contingency table**
- Let *(Ai,Bj)* denote the joint event that attribute *A* takes on value *ai* and attribute *B* takes on value *bj*, that is, where *(A = ai, B = bj)*.

- The $\chi 2$ value (also known as the *Pearson* $\chi 2$ *statistic*) is computed as

$$\chi^2 = \sum_{i=1}^{c} \sum_{j=1}^{r} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}, \qquad (3.1)$$

- Where oij is the observed frequency of the joint event (Ai, Bj) and eij is the expected frequency of the joint event (Ai,Bj)

$$e_{ij} = \frac{count(A = a_i) \times count(B = b_j)}{n}, \qquad (3.2)$$

Correlation Analysis (Nominal Data)

- The larger the X^2 value, the more likely the variables are related
- The cells that contribute the most to the X^2 value are those whose actual count is very different from the expected count
- Correlation does not imply causality
  - # of hospitals and # of car-theft in a city are correlated
  - Both are causally linked to the third variable: population

Chi-Square Calculation: An Example

	Play chess	Not play chess	Sum (row)
Like science fiction	250(90)	200(360)	450
Not like science fiction	50(210)	1000(840)	1050
Sum(col.)	300	1200	1500

- X^2 (chi-square) calculation (numbers in parenthesis are expected counts calculated based on the data distribution in the two categories)

$$\chi^2 = \frac{(250-90)^2}{90} + \frac{(50-210)^2}{210} + \frac{(200-360)^2}{360} + \frac{(1000-840)^2}{840} = 507.93$$

- It shows that like_science_fiction and play_chess are correlated in the group

Correlation Analysis (Numeric Data)

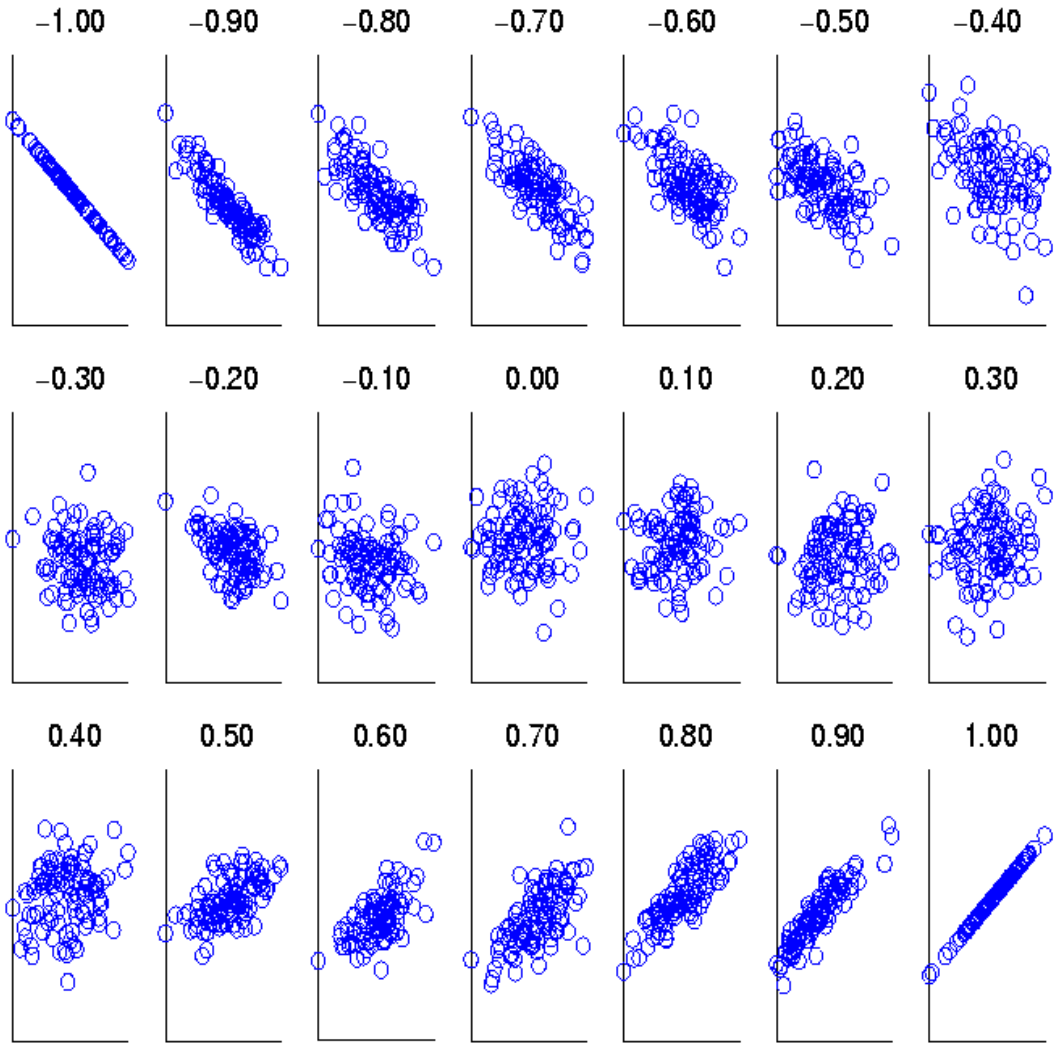
- Correlation coefficient (also called Pearson's product moment coefficient)

$$r_{A,B} = \frac{\sum_{i=1}^{n}(a_i - \bar{A})(b_i - \bar{B})}{n\sigma_A\sigma_B} = \frac{\sum_{i=1}^{n}(a_i b_i) - n\bar{A}\bar{B}}{n\sigma_A\sigma_B},$$

where n is the number of tuples, $\bar{A}$ and $\bar{B}$ are the respective means of A and B, σ_A and σ_B are the respective standard deviation of A and B, and $\Sigma(a_i b_i)$ is the sum of the AB cross-product.

- If $r_{A,B} > 0$, A and B are positively correlated (A's values increase as B's). The higher, the stronger correlation.
- $r_{A,B} = 0$: independent; $r_{AB} < 0$: negatively correlated

Visually Evaluating Correlation



Scatter plots showing the similarity from –1 to 1.

Covariance (Numeric Data)

- Covariance is similar to correlation

$$Cov(A, B) = E((A - \bar{A})(B - \bar{B})) = \frac{\sum_{i=1}^{n}(a_i - \bar{A})(b_i - \bar{B})}{n}$$

Correlation coefficient: $r_{A,B} = \frac{Cov(A, B)}{\sigma_A \sigma_B}$

where n is the number of tuples, $\bar{A}$ and $\bar{B}$ are the respective mean or **expected values** of A and B, σ_A and σ_B are the respective standard deviation of A and B.

- **Positive covariance**: If $Cov_{A,B} > 0$, then A and B both tend to be larger than their expected values.
- **Negative covariance**: If $Cov_{A,B} < 0$ then if A is larger than its expected value, B is likely to be smaller than its expected value.
- **Independence**: $Cov_{A,B} = 0$ but the converse is not true:
  - Some pairs of random variables may have a covariance of 0 but are not independent. Only under some additional assumptions (e.g., the data follow multivariate normal distributions) does a covariance of 0 imply independence

Co-Variance: An Example

$$Cov(A,B) = E((A-\bar{A})(B-\bar{B})) = \frac{\sum_{i=1}^{n}(a_i-\bar{A})(b_i-\bar{B})}{n}$$

- It can be simplified in computation as

$$Cov(A,B) = E(A \cdot B) - \bar{A}\bar{B}$$

- Suppose two stocks A and B have the following values in one week: (2, 5), (3, 8), (5, 10), (4, 11), (6, 14).
- Question: If the stocks are affected by the same industry trends, will their prices rise or fall together?
  - E(A) = (2 + 3 + 5 + 4 + 6)/ 5 = 20/5 = 4
  - E(B) = (5 + 8 + 10 + 11 + 14) /5 = 48/5 = 9.6
  - Cov(A,B) = (2×5+3×8+5×10+4×11+6×14)/5 − 4 × 9.6 = 4
- Thus, A and B rise together since Cov(A, B) > 0.

Data Integration: Tuple duplication

- Duplication should also be detected at tuple level
  - Denormalized tables
  - Same purchaser name is appearing in different addresses

Data Preprocessing

- Data Preprocessing: An Overview
  - Data Quality
  - Major tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- **Data Transformation**
- Dimensionality Reduction
- Summary

Data Transformation

- In *data transformation*, the data are transformed or consolidated into forms appropriate for mining.
- Through appropriate data transformation, the resulting mining process may be more efficient, and the patterns found may be easier to understand.
- Approaches
  - Normalization
  - Discretization
  - Data compression
  - Sampling

Normalization

- The measurement unit used can affect the data analysis.
  - For example, changing measurement units from meters to inches for *height* or from kilograms to pounds for weight may lead to very different results.
  - Salary attribute may dominate over age attribute
- To help avoid dependence on the choice of measurement units, the data should be *normalized* or *standardized*.
- Normalizing the data attempts to give all attributes an equal weight.

Normalization

- **Min-max normalization**: to [new_min$_A$, new_max$_A$]

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

  - Ex. Let income range \$12,000 to \$98,000 normalized to [0.0, 1.0]. Then \$73,000 is mapped to $\frac{73{,}600 - 12{,}000}{98{,}000 - 12{,}000}(1.0 - 0) + 0 = 0.716$

- **Z-score normalization** (μ: mean, σ: standard deviation):

$$v' = \frac{v - \mu_A}{\sigma_A}$$

  - Ex. Let μ = 54,000, σ = 16,000. Then $\frac{73{,}600 - 54{,}000}{16{,}000} = 1.225$

- **Normalization by decimal scaling**

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

Discretization

- Data discretization is a common data transformation technique where the raw values of a numeric attribute (e.g., *age*) are replaced by interval labels (e.g., 0–10, 11–20, etc.) or conceptual labels (e.g., *youth*, *adult*, *senior*).
  - The labels, in turn, can be recursively organized into higher-level concepts, resulting in a *concept hierarchy* for the numeric attribute.

Discretization

- Three types of attributes
  - Nominal—values from an unordered set, e.g., color, profession
  - Ordinal—values from an ordered set, e.g., military or academic rank
  - Numeric—real numbers, e.g., integer or real numbers
- Discretization: Divide the range of a continuous attribute into intervals
  - Interval labels can then be used to replace actual data values
  - Reduce data size by discretization
  - Supervised vs. unsupervised
  - Split (top-down) vs. merge (bottom-up)
  - Discretization can be performed recursively on an attribute
  - Prepare for further analysis, e.g., classification

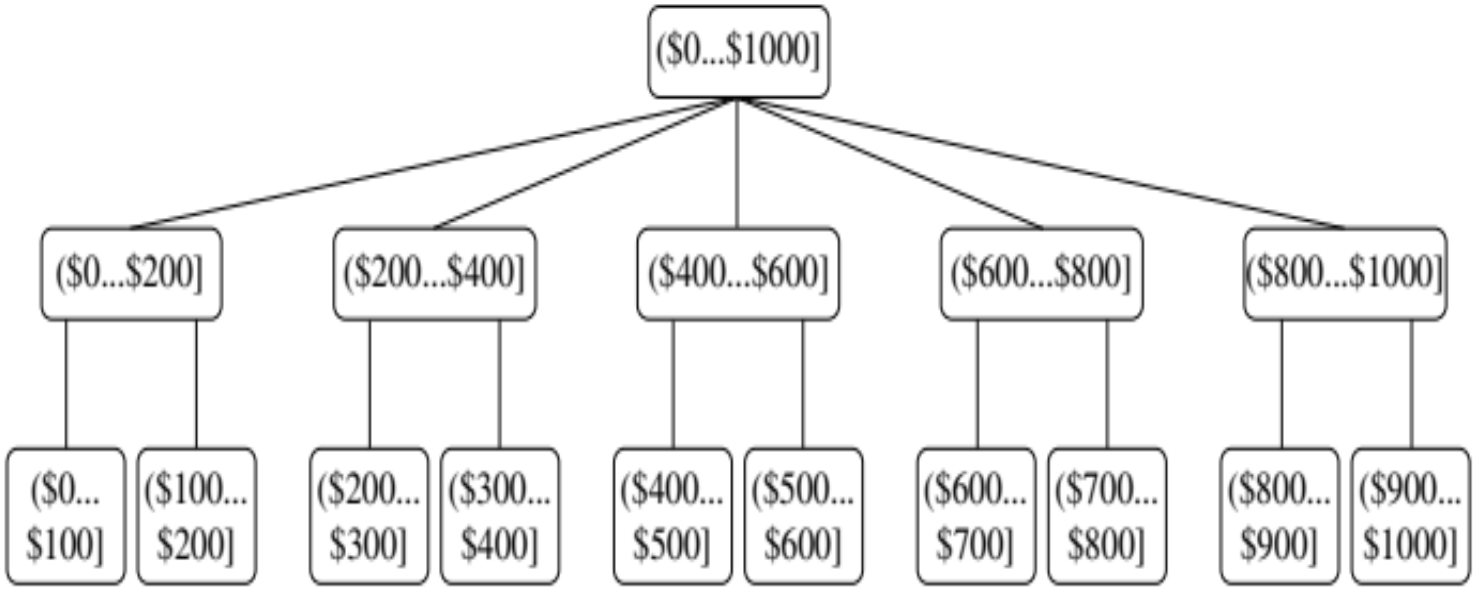


Figure 3.12 A concept hierarchy for the attribute *price*, where an interval ($*X*...$*Y*] denotes the range from $*X* (exclusive) to $*Y* (inclusive).

Data Discretization Methods

- Typical methods: All the methods can be applied recursively
  - Binning
    - Top-down split, unsupervised
  - Histogram analysis
    - Top-down split, unsupervised
  - Clustering analysis (unsupervised, top-down split or bottom-up merge)
  - Decision-tree analysis (supervised, top-down split)
  - Correlation (e.g., χ^2) analysis (unsupervised, bottom-up merge)

Simple Discretization: Binning

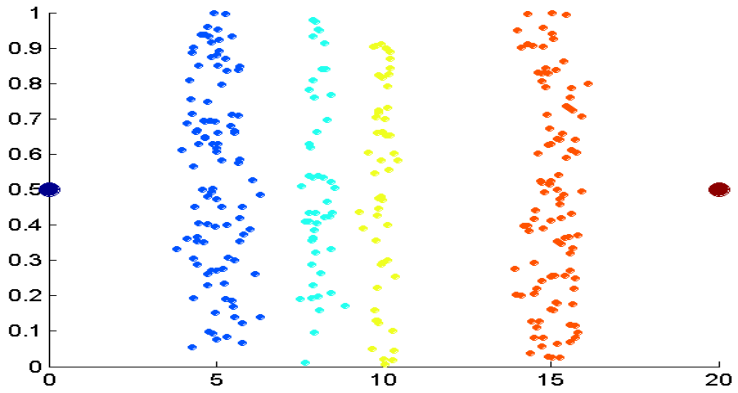
- Equal-width (distance) partitioning
  - Divides the range into *N* intervals of equal size: uniform grid
  - if *A* and *B* are the lowest and highest values of the attribute, the width of intervals will be: $W = (B - A)/N$.
  - The most straightforward, but outliers may dominate the presentation
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

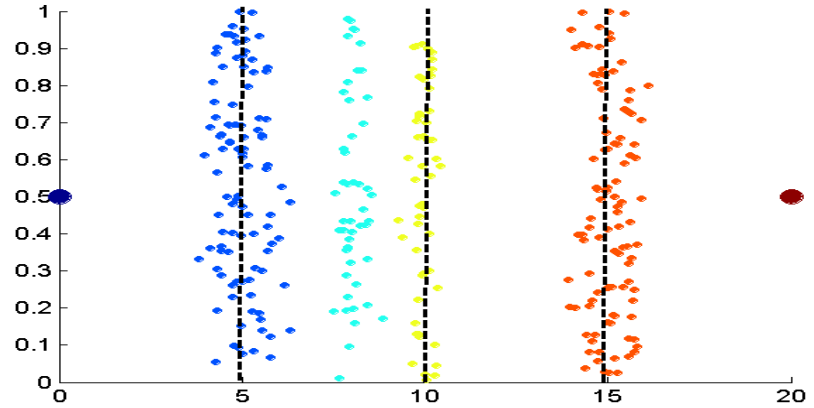
Binning Methods for Data Smoothing

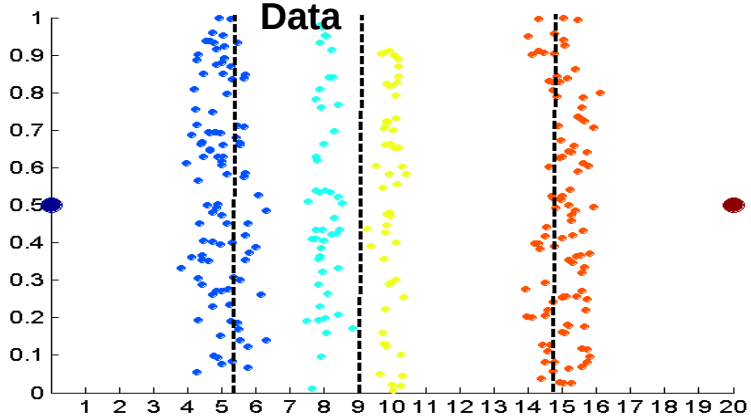
- Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34

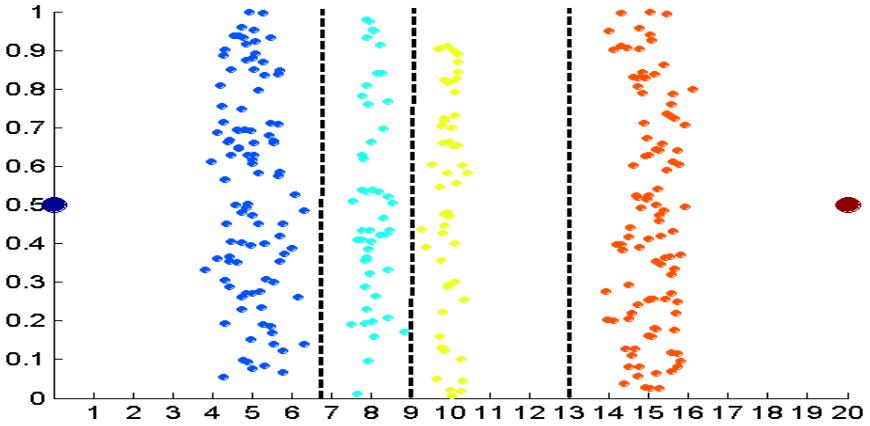
* Partition into equal-frequency (**equi-depth**) bins:
  - Bin 1: 4, 8, 9, 15
  - Bin 2: 21, 21, 24, 25
  - Bin 3: 26, 28, 29, 34
* Smoothing by **bin means**:
  - Bin 1: 9, 9, 9, 9
  - Bin 2: 23, 23, 23, 23
  - Bin 3: 29, 29, 29, 29
* Smoothing by **bin boundaries**:
  - Bin 1: 4, 4, 4, 15
  - Bin 2: 21, 21, 25, 25
  - Bin 3: 26, 26, 26, 34

Discretization Without Using Class Labels (Binning vs. Clustering)









Equal frequency (binning)

K-means clustering leads to better results

Discretization by Classification & Correlation Analysis

- Classification (e.g., decision tree analysis)
  - Supervised: Given class labels, e.g., cancerous vs. benign
  - Using *entropy* to determine split point (discretization point)
  - Top-down, recursive split
  - Details to be covered in Chapter 7
- Correlation analysis (e.g., Chi-merge: χ^2-based discretization)
  - Supervised: use class information
  - Bottom-up merge: find the best neighboring intervals (those having similar distributions of classes, i.e., low χ^2 values) to merge
  - Merge performed recursively, until a predefined stopping condition

Concept Hierarchy Generation

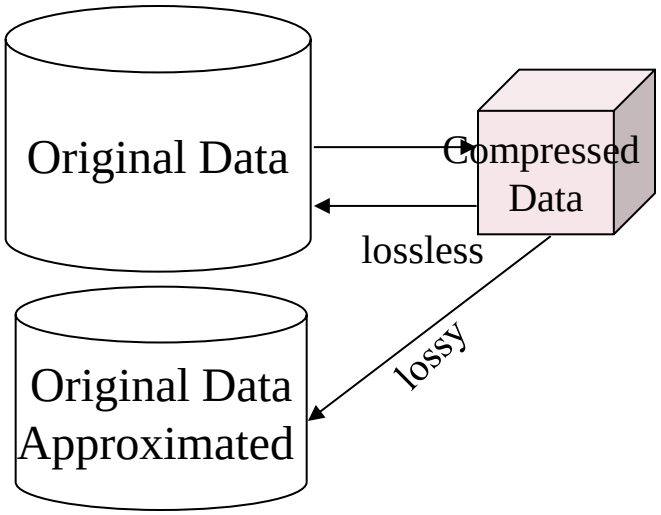
- **Concept hierarchy** organizes concepts (i.e., attribute values) hierarchically and is usually associated with each dimension in a data warehouse
- Concept hierarchies facilitate drilling and rolling in data warehouses to view data in multiple granularity
- Concept hierarchy formation: Recursively reduce the data by collecting and replacing low level concepts (such as numeric values for *age*) by higher level concepts (such as *youth*, *adult*, or *senior*)
- Concept hierarchies can be explicitly specified by domain experts and/or data warehouse designers
- Concept hierarchy can be automatically formed for both numeric and nominal data. For numeric data, use discretization methods shown.

Concept Hierarchy Generation for Nominal Data

- Specification of a partial/total ordering of attributes explicitly at the schema level by users or experts
  - *street* < *city* < *state* < *country*
- Specification of a hierarchy for a set of values by explicit data grouping
  - {Urbana, Champaign, Chicago} < Illinois
- Specification of only a partial set of attributes
  - E.g., only *street* < *city*, not others
- Automatic generation of hierarchies (or attribute levels) by the analysis of the number of distinct values
  - E.g., for a set of attributes: {*street*, *city*, *state*, *country*}

Data Compression

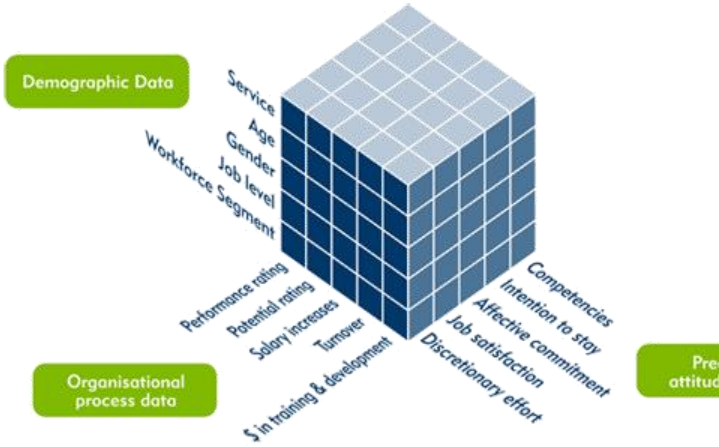
- String compression
  - There are extensive theories and well-tuned algorithms
  - Typically lossless, but only limited manipulation is possible without expansion
- Audio/video compression
  - Typically lossy compression, with progressive refinement
  - Sometimes, small fragments of the signal can be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



Lossy vs. lossless compression

Data Cube Aggregation

- The lowest level of a data cube (base cuboid)
  - 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



Automatic Concept Hierarchy Generation

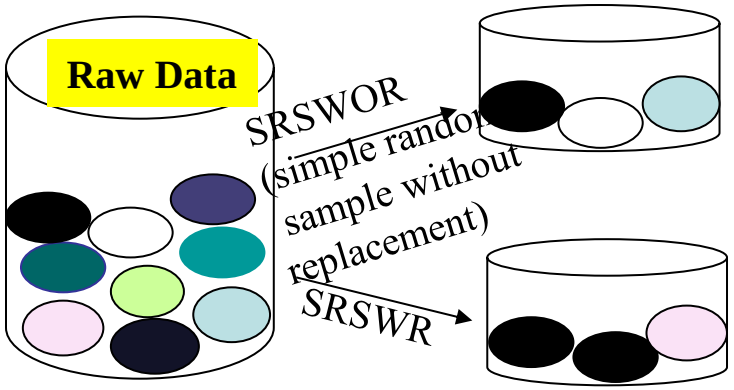
- Some hierarchies can be automatically generated based on the analysis of the number of distinct values per attribute in the data set
  - The attribute with the most distinct values is placed at the lowest level of the hierarchy
  - Exceptions, e.g., weekday, month, quarter, year

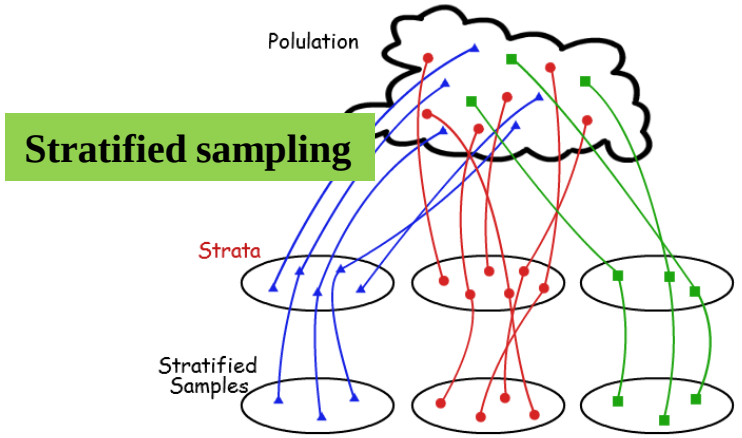
Sampling

- Sampling: obtaining a small sample *s* to represent the whole data set *N*
- Allow a mining algorithm to run in complexity that is potentially sub-linear to the size of the data
- Key principle: Choose a **representative** subset of the data
  - Simple random sampling may have very poor performance in the presence of skew
  - Develop adaptive sampling methods, e.g., stratified sampling:
- Note: Sampling may not reduce database I/Os (page at a time)

Types of Sampling

- **Simple random sampling:** equal probability of selecting any particular item
- **Sampling without replacement**
  - Once an object is selected, it is removed from the population
- **Sampling with replacement**
  - A selected object is not removed from the population
- **Stratified sampling**
  - Partition (or cluster) the data set, and draw samples from each partition (proportionally, i.e., approximately the same percentage of the data)





Data Preprocessing

- Data Preprocessing: An Overview
  - Data Quality
  - Major tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- Data Transformation
- **Dimensionality Reduction**
- Summary

Data Reduction Strategies

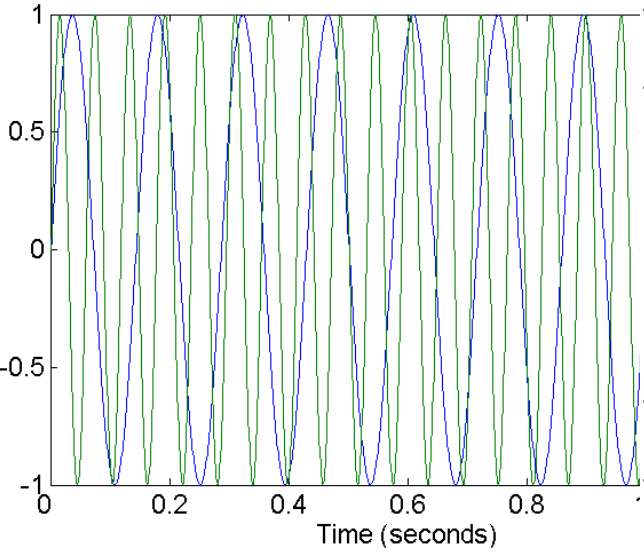
- **Data reduction**: Obtain a reduced representation of the data set that is much smaller in volume but yet produces the same (or almost the same) analytical results
- Why data reduction? — A database/data warehouse may store terabytes of data. Complex data analysis may take a very long time to run on the complete data set.
- Data reduction strategies
  - Dimensionality reduction, e.g., remove unimportant attributes
    - Wavelet transforms
    - Principal Components Analysis (PCA)
    - Feature subset selection, feature creation
  - Numerosity reduction (some simply call it: Data Reduction)
    - Regression and Log-Linear Models
    - Histograms, clustering, sampling
    - Data cube aggregation
  - Data compression

Data Reduction 1: Dimensionality Reduction

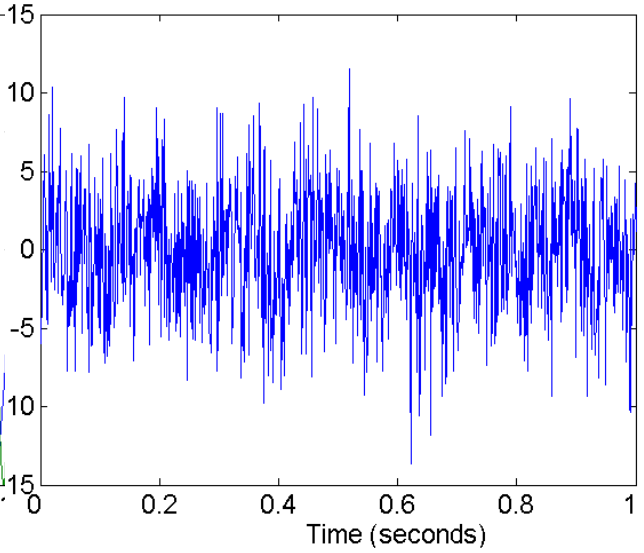
- **Curse of dimensionality**
  - When dimensionality increases, data becomes increasingly sparse
  - Density and distance between points, which is critical to clustering, outlier analysis, becomes less meaningful
  - The possible combinations of subspaces will grow exponentially
- **Dimensionality reduction**
  - Avoid the curse of dimensionality
  - Help eliminate irrelevant features and reduce noise
  - Reduce time and space required in data mining
  - Allow easier visualization
- **Dimensionality reduction techniques**
  - Wavelet transforms
  - Principal Component Analysis
  - Supervised and nonlinear techniques (e.g., feature selection)

Mapping Data to a New Space

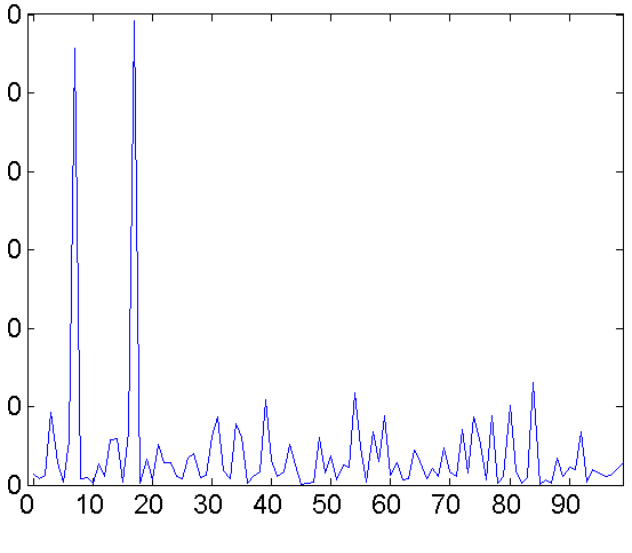
- **Fourier transform**
- **Wavelet transform**



Two Sine Waves



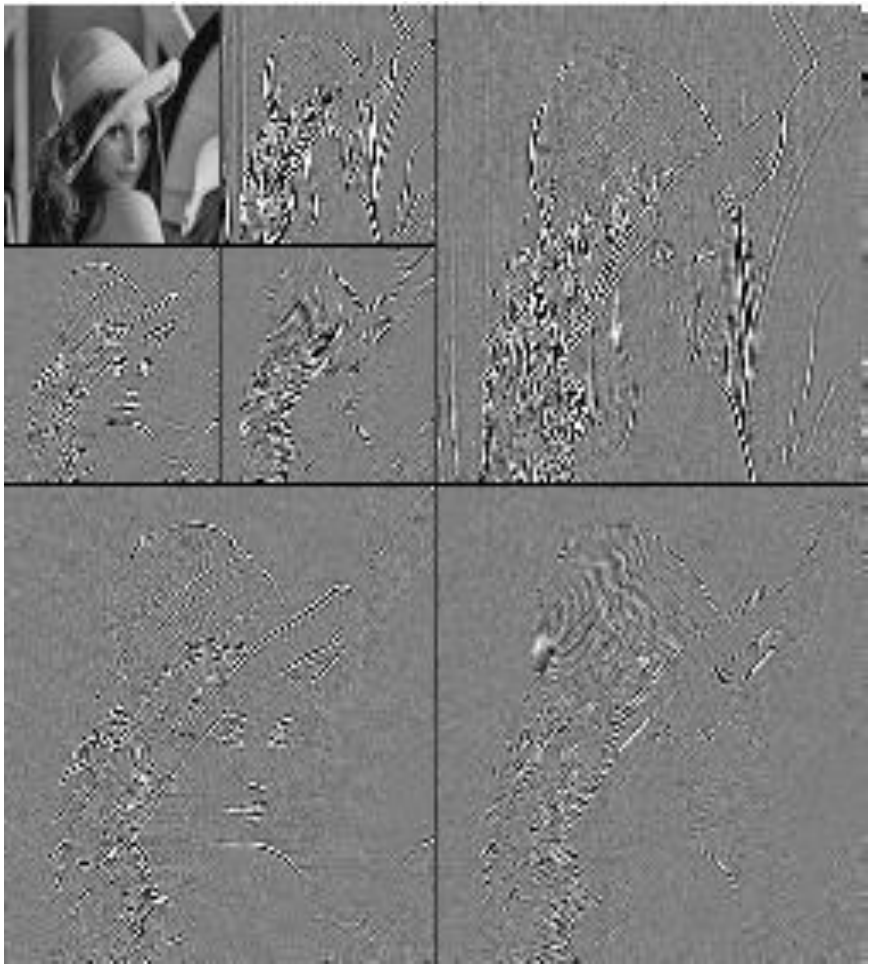
Two Sine Waves + Noise



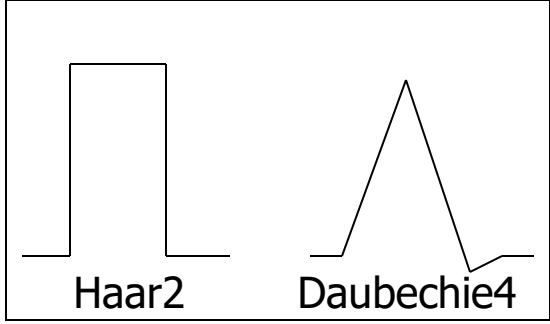
Frequency

What Is Wavelet Transform?

- Decomposes a signal into different frequency subbands
  - Applicable to n-dimensional signals
- Data are transformed to preserve relative distance between objects at different levels of resolution
- Allow natural clusters to become more distinguishable
- Used for image compression



Wavelet Transformation



- Discrete wavelet transform (DWT) for linear signal processing, multi-resolution 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 0's, 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

Wavelet Decomposition

- Wavelets: A math tool for space-efficient hierarchical decomposition of functions
- S = [2, 2, 0, 2, 3, 5, 4, 4] can be transformed to $S_{\wedge}$ = [$2^{3}/_{4}$, $-1^{1}/_{4}$, $^{1}/_{2}$, 0, 0, -1, -1, 0]
- Compression: many small detail coefficients can be replaced by 0's, and only the significant coefficients are retained

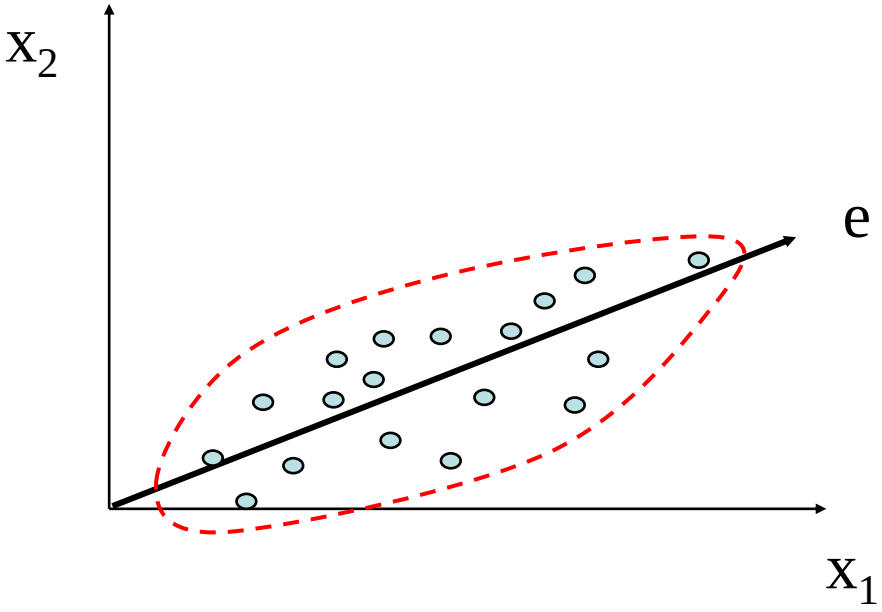
Resolution	Averages	Detail Coefficients
8	$[2, 2, 0, 2, 3, 5, 4, 4]$	
4	$[2, 1, 4, 4]$	$[0, -1, -1, 0]$
2	$[1\frac{1}{2}, 4]$	$[\frac{1}{2}, 0]$
1	$[2\frac{3}{4}]$	$[-1\frac{1}{4}]$

Why Wavelet Transform?

- Use hat-shape filters
  - Emphasize region where points cluster
  - Suppress weaker information in their boundaries
- Effective removal of outliers
  - Insensitive to noise, insensitive to input order
- Multi-resolution
  - Detect arbitrary shaped clusters at different scales
- Efficient
  - Complexity O(N)
- Only applicable to low dimensional data

Principal Component Analysis (PCA)

- Find a projection that captures the largest amount of variation in data
- The original data are projected onto a much smaller space, resulting in dimensionality reduction. We find the eigenvectors of the covariance matrix, and these eigenvectors define the new space



Principal Component Analysis (Steps)

- Given *N* data vectors from *n*-dimensions, find $k \leq n$ orthogonal vectors (*principal components*) that can be best used to represent data
  - Normalize input data: Each attribute falls within the same range
  - Compute *k* orthonormal (unit) vectors, i.e., *principal components*
  - Each input data (vector) is a linear combination of the *k* principal component vectors
  - The principal components are sorted in order of decreasing "significance" or strength
  - Since the components are sorted, the size of the data can be reduced by eliminating the *weak components*, i.e., those with low variance (i.e., using the strongest principal components, it is possible to reconstruct a good approximation of the original data)
- Works for numeric data only
- Reference: https://www.turing.com/kb/guide-to-principal-component-analysis

Attribute Subset Selection

- Another way to reduce dimensionality of data
- Redundant attributes
  - Duplicate much or all of the information contained in one or more other attributes
  - E.g., purchase price of a product and the amount of sales tax paid
- Irrelevant attributes
  - Contain no information that is useful for the data mining task at hand
  - E.g., students' ID is often irrelevant to the task of predicting students' GPA

Heuristic Search in Attribute Selection

- There are 2^d possible attribute combinations of d attributes
- Typical heuristic attribute selection methods:
  - Best single attribute under the attribute independence assumption: choose by significance tests
  - Best step-wise feature selection:
    - The best single-attribute is picked first
    - Then next best attribute condition to the first, ...
  - Step-wise attribute elimination:
    - Repeatedly eliminate the worst attribute
  - Best combined attribute selection and elimination
  - Optimal branch and bound:
    - Use attribute elimination and backtracking
  - Decision tree induction

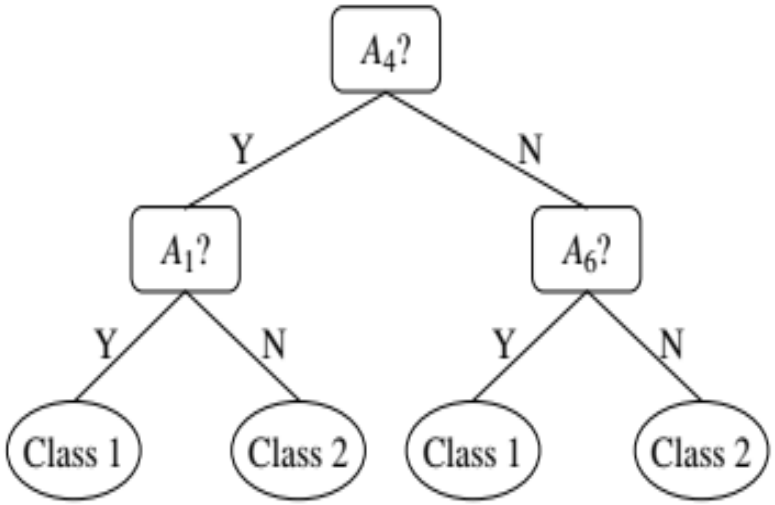
Forward selection	Backward elimination	Decision tree induction
Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ Initial reduced set: $\{\}$ $\Rightarrow \{A_1\}$ $\Rightarrow \{A_1, A_4\}$ $\Rightarrow$ Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ $\Rightarrow \{A_1, A_3, A_4, A_5, A_6\}$ $\Rightarrow \{A_1, A_4, A_5, A_6\}$ $\Rightarrow$ Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$  $\Rightarrow$ Reduced attribute set: $\{A_1, A_4, A_6\}$

Figure 3.6 Greedy (heuristic) methods for attribute subset selection.

Attribute Creation (Feature Generation)

- Create new attributes (features) that can capture the important information in a data set more effectively than the original ones
- Three general methodologies
  - Attribute extraction
    - Domain-specific
  - Mapping data to new space (see: data reduction)
    - E.g., Fourier transformation, wavelet transformation, manifold approaches (not covered)
  - Attribute construction
    - Combining features (see: discriminative frequent patterns in Chapter 7)
    - Data discretization

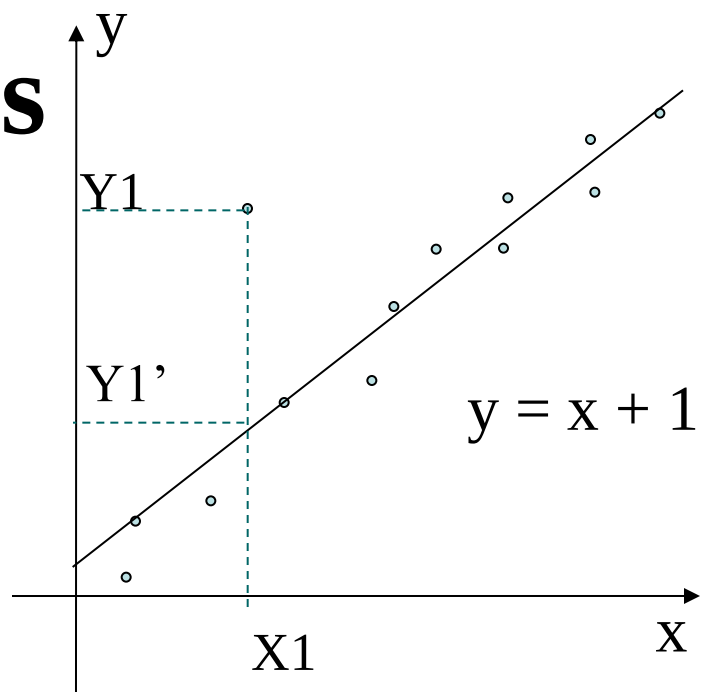
Data Reduction 2: Numerosity Reduction

- Reduce data volume by choosing alternative, *smaller forms* of data representation
- **Parametric methods** (e.g., regression)
  - Assume the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers)
  - Ex.: 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, …

Numerosity reduction
Regression and Log-Linear Models

- **Linear regression**
  - Data 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

Regression Analysis



- Regression analysis: A collective name for techniques for the modeling and analysis of numerical data consisting of values of a ***dependent variable*** (also called ***response variable*** or *measurement*) and of one or more *independent variables* (aka. ***explanatory variables*** or ***predictors***)
- The parameters are estimated so as to give a "**best fit**" of the data
- Most commonly the best fit is evaluated by using the ***least squares method***, but other criteria have also been used

- Used for prediction (including forecasting of time-series data), inference, hypothesis testing, and modeling of causal relationships

Regress Analysis and Log-Linear Models

- Linear regression: $Y = w X + b$
  - Two regression coefficients, w and b, specify the line and are to be estimated by using the data at hand
  - Using the least squares criterion to the known values of Y_1, Y_2, ..., X_1, X_2, ...
- Multiple regression: $Y = b_0 + b_1 X_1 + b_2 X_2$
  - Many nonlinear functions can be transformed into the above
- Log-linear models:
  - Approximate discrete multidimensional probability distributions
  - Estimate the probability of each point (tuple) in a multi-dimensional space for a set of discretized attributes, based on a smaller subset of dimensional combinations
  - Useful for dimensionality reduction and data smoothing

Histogram Analysis

- Divide data into buckets and store average (sum) for each bucket
- Partitioning rules:
  - Equal-width: equal bucket range
  - Equal-frequency (or equal-depth)

Example 3.3 **Histograms.** The following data are a list of *AllElectronics* prices for commonly sold items (rounded to the nearest dollar). The numbers have been sorted: 1, 1, 5, 5, 5, 5, 5, 8, 8, 10, 10, 10, 10, 12, 14, 14, 14, 15, 15, 15, 15, 15, 15, 18, 18, 18, 18, 18, 18, 18, 18, 20, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 25, 25, 25, 25, 25, 28, 28, 30, 30, 30.

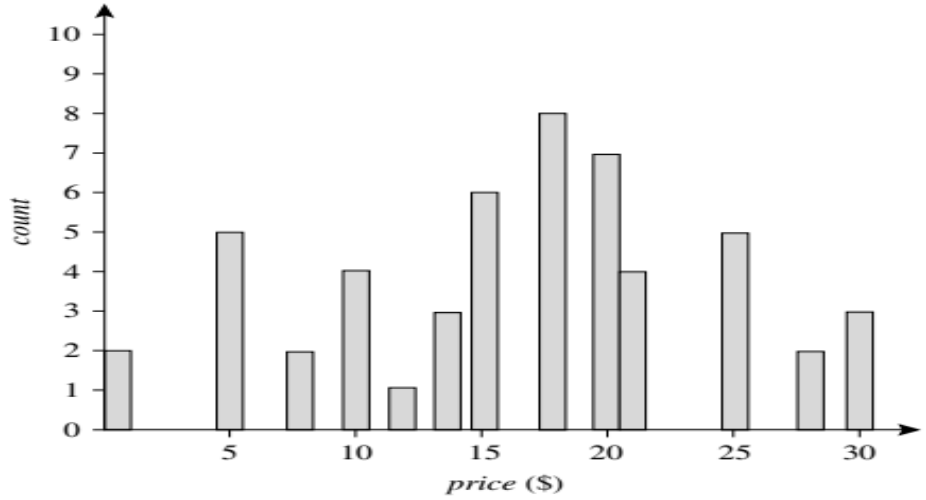


Figure 3.7 A histogram for *price* using singleton buckets—each bucket represents one price–value/frequency pair.

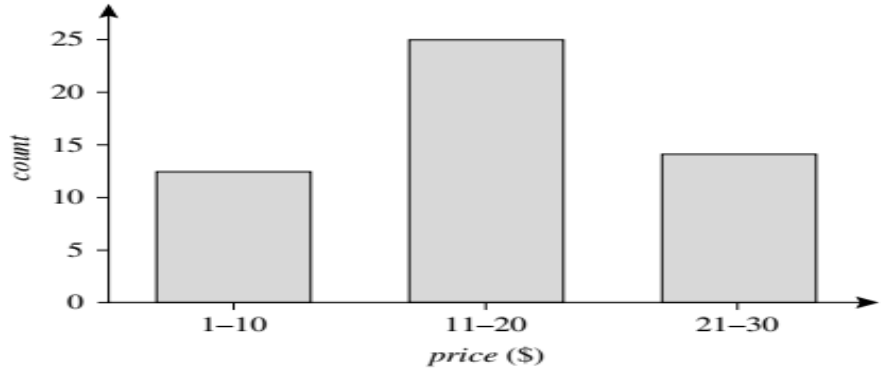
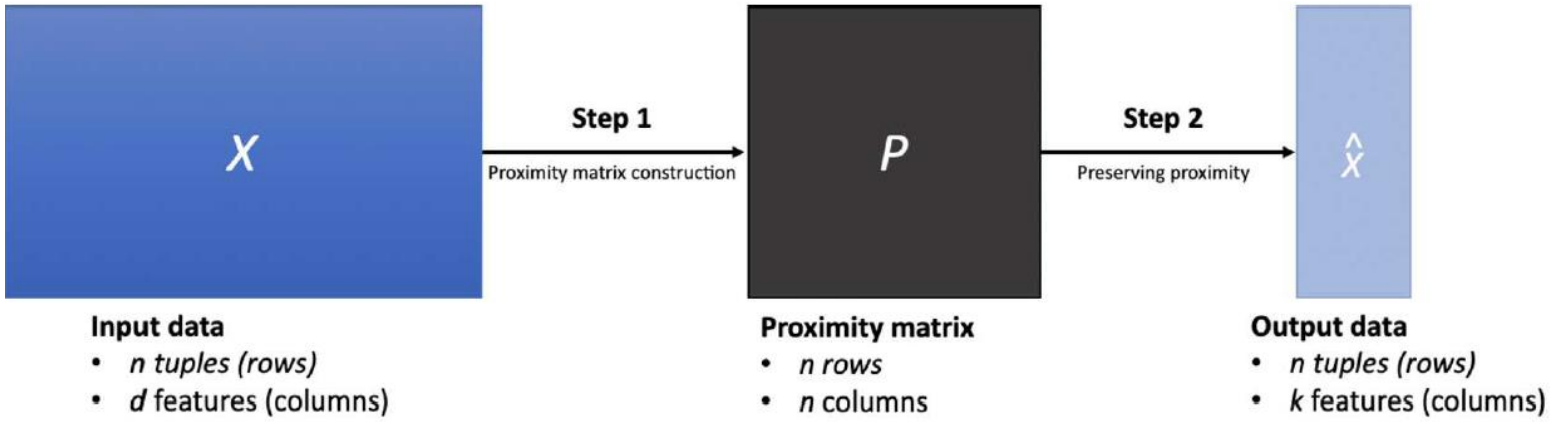


Figure 3.8 An equal-width histogram for *price*, where values are aggregated so that each bucket has a uniform width of $10.

Nonlinear Dimensionality Reduction Methods

- PCA is a linear method for dimensionality reduction
  - Each principal component is a linear combination of the original input attributes
  - It works well if the input data approximately follows a Gaussian distribution or forms a few linearly separable clusters
- When the input data is **linearly inseparable**, we need to construct a proximity matrix (P) and learn a new matrix with *k* features (*k* << *d*) that preserves the proximity



Nonlinear Dimensionality Reduction (I): Kernel PCA (KPCA)

- Use a kernel function $\kappa(\cdot)$ to construct the kernel matrix: $P(i, j) = \kappa(x_i, x_j)$, and learn the best low-dimensional representations so that the estimated proximity matrix $\hat{P}$ is as close as possible to the kernel matrix P
- This can be obtained by using top-*k* eigenvectors and eigenvalues of the kernel matrix P
- Typical kernel functions:
  - (1) polynomial kernel: $\kappa(x_i, x_j) = (1 + x_i \cdot x_j)^p$
  - (2) radial basis function (RBF): $\kappa(x_i, x_j) = e^{\frac{-\|x_i - x_j\|^2}{2\sigma^2}}$
  - If we choose a linear kernel: $\kappa(x_i, x_j) = x_i \cdot x_j$, KPCA degenerates to the standard PCA
- Major formulas of Kernel PCA vs. SNE (Stochastic neighborhood embedding)

	Step 1: Proximity Construction	Step 2: Preserving Proximity
KPCA	$P(i, j) = \kappa(x_i, x_j)$	$\min \sum_{i,j=1}^{n} (P(i, j) - \hat{P}(i, j))^2 = \|P - \hat{P}\|_{fro}^2$
SNE	$P(i, j) = \frac{e^{-d_{ij}^2}}{\sum_{l=1, l \neq i}^{n} e^{-d_{il}^2}}$	$\min \sum_{i=1}^{n} \mathrm{KL}(P_i \| \hat{P}_i)$

Nonlinear Dimensionality Reduction (II): SNE

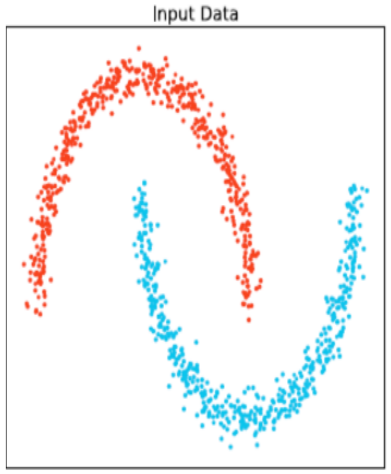
- SNE (Stochastic neighborhood embedding)
  - Construct a proximity matrix P using the formula: $P(i,j) = \frac{e^{-d_{ij}^2}}{\sum_{l=1,l\neq i}^{n} e^{-d_{il}^2}}$ where $d_{ij}^2 = \frac{\|x_i - x_j\|^2}{2\sigma^2}$
    - rep. the probability that x_j is the neighbor of x_i
  - Suppose we have learned the low-dimensional representations x_i^, we can compute anther estimated proximity matrix in the similar way: *P^ (i, j)*
  - We want to make the estimated proximity matrix P^ to be as close as possible to P
  - That is, we want to minimize the overall K-L divergence, that is,

$$\hat{x}_i = \arg\min_{\hat{x}_i,(i=1,\ldots n)} \sum_{i=1}^{n} D_{KL}(P_i||\hat{P}_i)$$

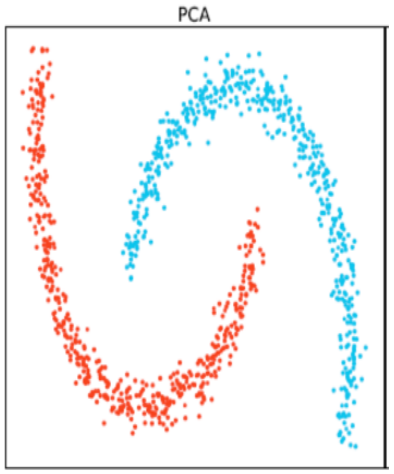
	Step 1: Proximity Construction	Step 2: Preserving Proximity
KPCA	$P(i,j) = \kappa(x_i, x_j)$	$\min \sum_{i,j=1}^{n} (P(i,j) - \hat{P}(i,j))^2 = \|P - \hat{P}\|_{fro}^2$
SNE	$P(i,j) = \frac{e^{-d_{ij}^2}}{\sum_{l=1,l\neq i}^{n} e^{-d_{il}^2}}$	$\min \sum_{i=1}^{n} \mathrm{KL}(P_i\|\|\hat{P}_i)$

Example: Comparison on Nonlinear Data Points: Linear vs. Nonlinear Dimensional Reduction Methods

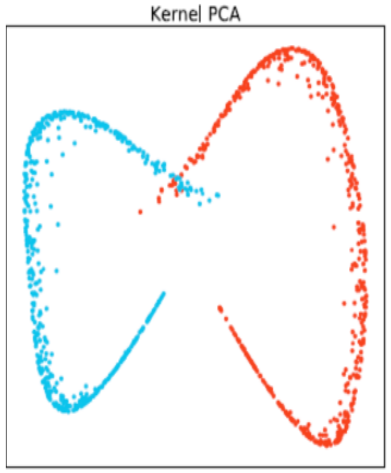
- Visualization: An example of linear vs. nonlinear dimensionality reduction methods
  - Given a collection of input data in 2-D space (Fig. (a)): Red and blue data points are not linearly separable
  - PCA transformation can not make it linearly separable
  - KPCA can make the points linearly separable
  - t-SNE (t-distributional NSE) can make them linearly separable



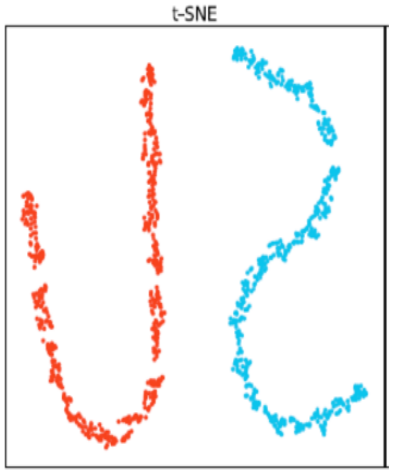
(a) Input data



(b) PCA



(c) KPCA



(d) t-SNE

Summary

- **Data quality**: accuracy, completeness, consistency, timeliness, believability, interpretability
- **Data cleaning**: e.g. missing/noisy values, outliers
- **Data integration** from multiple sources:
  - Entity identification problem
  - Remove redundancies
  - Detect inconsistencies
- **Data reduction**
  - Dimensionality reduction
  - Numerosity reduction
  - Data compression
- **Data transformation and data discretization**
  - Normalization
  - Concept hierarchy generation

References

- Refer to Bibliographic notes

- **Book: Jiawei Han, Jian Pei, Hanghang Tong, Data Mining: Concepts and Techniques, Fourth edition, 2022, Elseiver Inc.**

ATTRIBUTE-ORIENTED INDUCTION IN RELATIONAL DATABASES

by

Yandong Cai

Presented by

P. Krishna Reddy
IIIT Hyderabad

- Reference 1: Jiawei Han, Yandong Cai, Nick Cercone: Knowledge Discovery in Databases: An Attribute-Oriented Approach. VLDB 1992: 547-559

- Reference 2: Attribute Oriented Induction in Relational Databases, MS thesis, 1989, Simon Fraser University, https://summit.sfu.ca/item/4542

Outline

- Introduction
- Learning from examples
- Concepts of learning from databases
- Attribute oriented induction in relational databases
- Variations of the learning algorithms
- Discussion
- Implementation and experiments
- Conclusions and future research

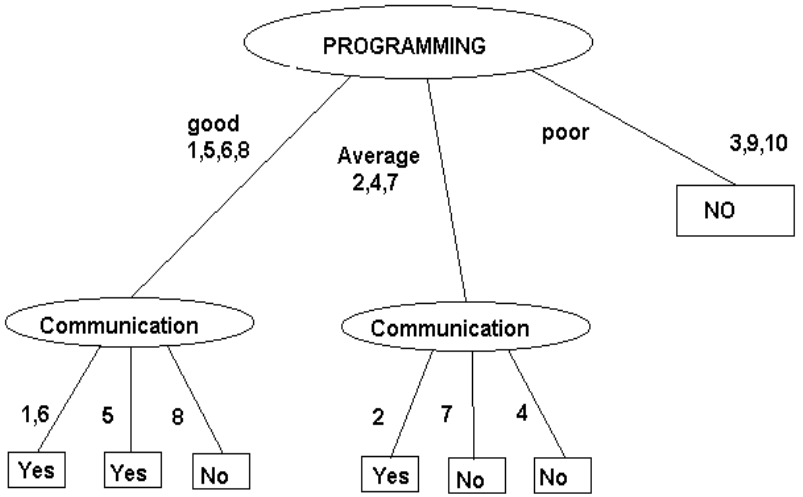
Introduction

- It is advantageous to learn characteristics of data in relational databases.
- By learning from databases, knowledge rules can be extracted from the large amount of data and interesting relationships among data can be discovered automatically.
- Relational databases store a large amount of information in a structured and organized manner.
- Each tuple in the database can be viewed as a typed logical formula in the conjunctive normal form.
- Such uniformity facilitates the application of well developed database implementation techniques and the development of efficient learning algorithms in large databases

Paradigm: Learning from Examples

- The paradigm of "learning from examples"
  - Learning by generalizing specific facts or observations has been adopted in many existing induction learning algorithms
  - **Example:** A recruitment agency which specialises in recruiting trainee programmers wants to develop a decision tree to filter out unsuitable applicants.
  - The agency we three criteria to make this decision: programming ability(Prog.), communication ability*Com.) & presentation(Pres.). Suppose the Agency provide the examples in the following Table:

NO	Prog.	Com	Pres	Decision
1.	good	ok	neat	yes
2.	average	ok	fair	yes
3.	poor	liar	scruffy	no
4.	average	liar	fair	no
5.	good	boring	scruffy	yes
6.	good	ok	scruffy	yes
7.	average	boring	scruffy	no
8.	good	liar	scruffy	no
9.	poor	boring	fair	no
10.	poor	boring	scruffy	no



Motivation

- The facts (large) in the database can be viewed as examples for learning.
  - The paradigm "learning from examples" should be a strategy for learning
- However, (limitations)
  - Current learning systems do not integrate well with relational database systems
    - Due to large number of possible combinations in such testing, the tuple-oriented approach is quite inefficient to perform learning from large databases
    - For example, if there are 100 training examples (tuples), each tuple has 5 attributes, and each attribute value can be one of the three concepts on three different generalization levels, such coverage testing will be invoked up to 100 * (5*5*5) = 12500 times in the worst case.
  - Moreover, most existing algorithms do not take the features and implementation techniques provided by database systems
    - They do not have negative examples

Introduction

- Following the learning from examples paradigm, the proposed attribute-oriented concept tree ascending technique, integrates database operations with the learning process.
- There are two types of knowledge rules
  - characteristic rules and classification rules
- Two algorithms are proposed
  - LCHR algorithm for Learning CHaracteristic Rules
  - LCLR algorithm for Learning CLassification Rules.
- These two algorithms can learn both conjunctive rules and restricted forms of disjunctive rules, and learning can be performed with databases containing exceptions and noisy data using database statistics.
- The algorithms show that attribute-oriented induction substantially reduces the complexity of the database learning processes

Outline

- Introduction
- **Learning from examples (Related Work)**
- Concepts of learning from databases
- Attribute oriented induction in relational databases
- Variations of the learning algorithms
- Discussion
- Implementation and experiments
- Conclusions and future research

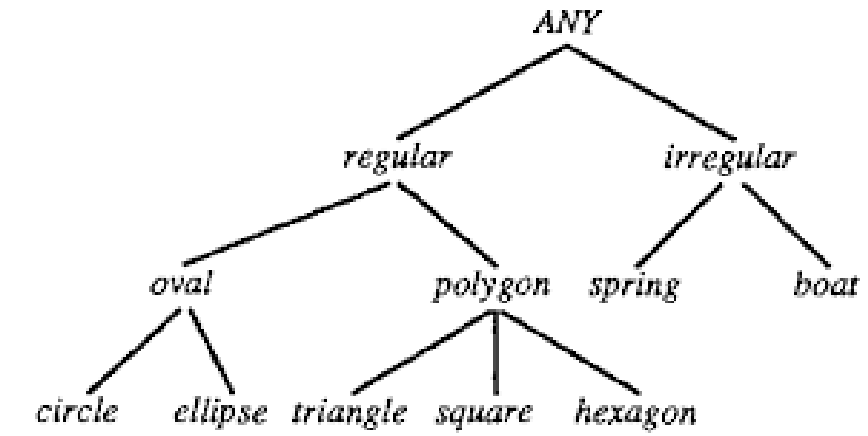
Learning from Examples: An AI approach

- Learning from examples can be viewed as a reasoning process from specific instances to general concepts.
- Learning from examples can be characterized by a tuple <P, N, C, A>, [GeN87]
  - where P is a set of positive examples of a concept,
  - N is a set of negative examples of a concept,
  - C is the conceptual bias which consists of a set of concepts to be used in defining learning rules and results,
  - and A is the logical bias which captures particular logic forms

- Approach 1 [Mic83]:
  - The training examples are classified in advance by the teacher into two disjoint sets, the positive example set and the negative example set. The learning task is to generalize these low-level concepts to general rules.
- Approach 2 [GeN87]:
  - There could be numerous inductive conclusions derived fiom a set of training examples. For instance, the concept "red" can be generalized in several ways: "red or black", "dark color", "warm color", etc.
  - To cope with this multiplicity of possibilities, it is necessary to use some additional information, problem background knowledge, to constrain the space of possible inductive conclusions and locate the most desired one(s).
  - The conceptual bias and the logical bias provide the desired concepts and logic forms which serve as this kind of background knowledge.
  - These biases restrict the candidates to formulas with a particular vocabulary and logic forms. Only those concepts which can be written in terms of this fixed vocabulary and logic forms are considered in the learning process.

- Depending on the structure of the attribute domains, we can distinguish among three basic types of attributes [Mic83]:
  - Nominal attributes:
    - Consists of independent symbols or names
    - No structure is assumed to relate the values in the domain.
    - Example: computer ID
  - Numeric attributes:
    - The value set of such attributes is a totally odered set. For example, weight, salary and gpa are numeric attributes.
  - Structured attributes:
    - The value set of such attributes has a tree structure which forms a generalization hierarchy.
    - A parent node in such a structure represents a more general concept than the concepts represented by its children nodes.

- The domain of structured attributes is defined by the problem background knowledge.
  - For example, the attribute shape could be a structured attribute whose domain is a tree structure with a set of leaves:
  - {triangle, circle, ellipse, hexagon, square, boat, spring), and
  - the non-leaf nodes are defined by rules:
    {circle, ellipse) c oval
    {triangle, square, hexagon) ⊂ polygon
    {oval, polygon] c regular
    {spring, boat) c irregular



About generalization rules

- The following generalization rules are particularly useful in learning systems [CoF83, Mic831]
  - If the concept F(v) holds for v when v is a constant a, or a constant b, and so on, then these concepts can be generalized into a statement that F(v) holds for every value of v.
  - F(a) AND F(b) .... GENERALIZED TO (|<) F(v) .
- Dropping conditions
  - red (v) AND apple (v) |< apple(v)
- Adding options
  - red(v) |< red(v) OR blue(v).
- Turning conjunction into disjunction.
  - red AND circle |< red OR circle
- Climbing a generalization tree
  - By ascending the generalization tree, the lower level concept is substituted for by the higher level concept.

Control Strategies in Learning from Examples

- Induction methods can be
  - bottom-up or top-down or combined
- In the data-driven methods, the presentation of the training examples drives the search.
  - These methods process the input examples one at a time, gradually generalizing the current set of concepts until a final conjunctive generalization is computed.
- In the model-driven methods, an a priori model is used to constrain the search.
  - These methods search a set of possible generalizations in an attempt to find a few "best" hypotheses that satisfy certain requirements.

- Data-driven techniques generally have the advantage of supporting incremental learning.
  - The learning process can start not only from the specific training examples, but also from the rules which have been discovered.
  - The learning systems are capable of updating the existing hypotheses to account for each new example.
- In contrast, the model-driven methods, which test and reject hypotheses based on an examination of the whole body of data, are difficult to be used in incremental learning situations.
  - When new training examples become available, model-driven methods must either backtrack or restart the learning process from the very beginning.
- Model-driven methods tend to have good noise immunity
- Noise impacts data driven methods.

- Several systems are being built (refer research paper)
- Example
  - If the weather is cloudy or raining, and the temperature is 40 to 60 degrees, the means of transportation should be a car.
  - [Transport = car] <= [Weather-type = cloudy V rain] A [Temp = 40..60]
- There are metrics to measure the strength of each rule (refer research paper)

Knowledge Discovery in Large Databases and Knowledges-Base

- There are efforts to build systems [KMK89]
- Suffers from computational inefficiency
- Just use database for retrieval purpose
- Follow tuple oriented approach.

Outline

- Introduction
- Learning from examples (Related Work)
- **Concepts of learning from databases**
- Attribute oriented induction in relational databases
- Variations of the learning algorithms
- Discussion
- Implementation and experiments
- Conclusions and future research

Primitives of learning from databases

- Primitives of Learning from Databases are characterized by a triple <D, C, A>, where
  - D represents the set of data in the database relevant to a specific learning task,
  - C represents a set of "concept biases" (generalization hierarchy, etc.) useful for defining particular concepts, and
  - A is a language used to phrase definition

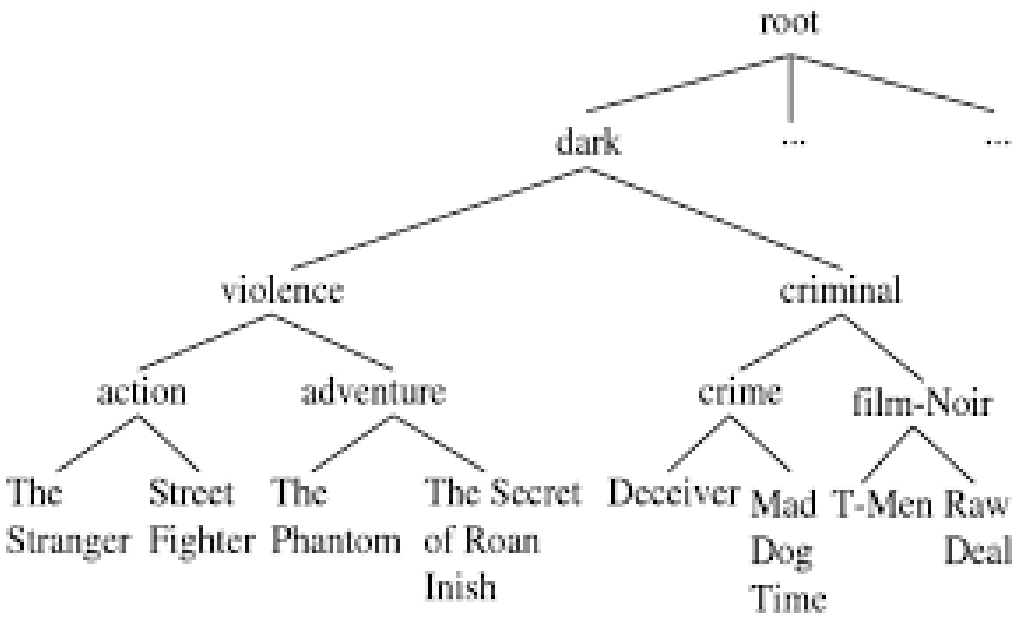
Data Relevant to Learning Task

- Characterize the specific learning task
  - For example, to characterize the features of graduate students, only the data relevant to graduate students are useful in the learning process.
- Data (D) should be obtained by performing relational operations, such as selection, projection and join.
- We assume that appropriate **preprocessing steps** are performed

- Most learning from examples algorithms partition the set of examples into positive and negative sets.
  - The positive examples are used to generalize the learning concepts, and the negative examples are used to specialize the learning concepts
- However, all of the data stored in the database which characterize the features of a property are positive data.
- However, most database applications assume that all of the information about a property is stored in the database.
  - For example, to distinguish graduate students from undergraduate students, the properties which belong to undergraduate students can be viewed as negative data.

Conceptual Bias Useful for Defining Concepts

- Assumption:
  - The conceptual bias is given for each attribute which is represented as a concept tree.
  - Such a concept tree is specified using an IS-A hierarchy and stored in a relation table, the concept hierarchy table.



Language Used to Phrase Definition

- We employ First-order predicate calculus

Name	Category	Major	Birth_Place	GPA
Jackson	senior	computing	Vancouver	3.5

can be viewed as representing the logic formula:

$$\exists\, t\ (\,(Name\,(t) = \text{Jackson}) \wedge (Category\,(t) = \text{senior}) \wedge (Major\,(t) = \text{computing})$$
$$\wedge\ (Birth\_Place\,(t) = \text{Vancouver}) \wedge (GPA\,(t) = 3.5)\,).$$

- The intermediate and final learning results can also be represented using relational tables.
  - Definition: A generalized relation is a relation obtained by substituting the specific concept(s) by the general concept(s) in some attribute(s)
- The maximum number of tuples in a final generalized relation, can be specified by users as a threshold value of the learning process.

Two Types of Rules

- Characteristic rules and Classification rules can be learned
- Characteristic rules
  - Definition: A characteristic rule is an assertion which characterizes the concepts satisfied by all of the data stored in the database.
    - Example: For example, the symptoms of a specific disease can be summarized as a characteristic rule.
- Classification rules
  - Definition: A classification rule is an assertion which discriminates the concepts of one class from other classes.
    - Distinguish one disease from others, a classification rule should summarize the symptoms that discriminate this disease from others.
- A characteristic rule provides generalized concepts about a property which can help people recognize the common features of the data in a class.
- The classification rule gives a discriminant criterion which can be used to predict the class membership of new data.

- The characteristic rules only concern the characteristics of the data.
  - Positive examples alone are enough to furnish the learning task.
- For learning classification rules, the negative examples must be incorporated into the learning process to derive the concepts which have the discriminant property
  - For example, the data about students may be classified into graduate students and undergraduate students based on the value of the attribute "Category".
- About target class and contrasting class.
  - Definition: **A target class** is a class in which the data are tuples in the database consistent with the learning concepts.
  - Definition: **A contrasting class** is a class in which the data do not belong to the target class.
    - For instance, to distinguish graduate students from undergraduate students, the class of graduate students is the target class, and the class of undergraduate students is the contrasting class

Example

A relation Student in a sample university database

Name	Category	Major	Birth_Place	GPA
Anderson	M.A.	history	Vancouver	3.5
Bach	junior	math	Calgary	3.7
Carey	junior	liberal arts	Edmonton	2.6
Fraser	M.S.	physics	Ottawa	3.9
Gupta	Ph.D.	math	Bombay	3.3
Hart	sophomore	chemistry	Richmond	2.7
Jackson	senior	computing	Victoria	3.5
Liu	Ph.D.	biology	Shanghai	3.4
Meyer	sophomore	music	Burnaby	3.0
Monk	Ph.D.	computing	Victoria	3.8
Wang	M.S.	statistics	Nanjing	3.2
Wise	freshman	literature	Toronto	3.9

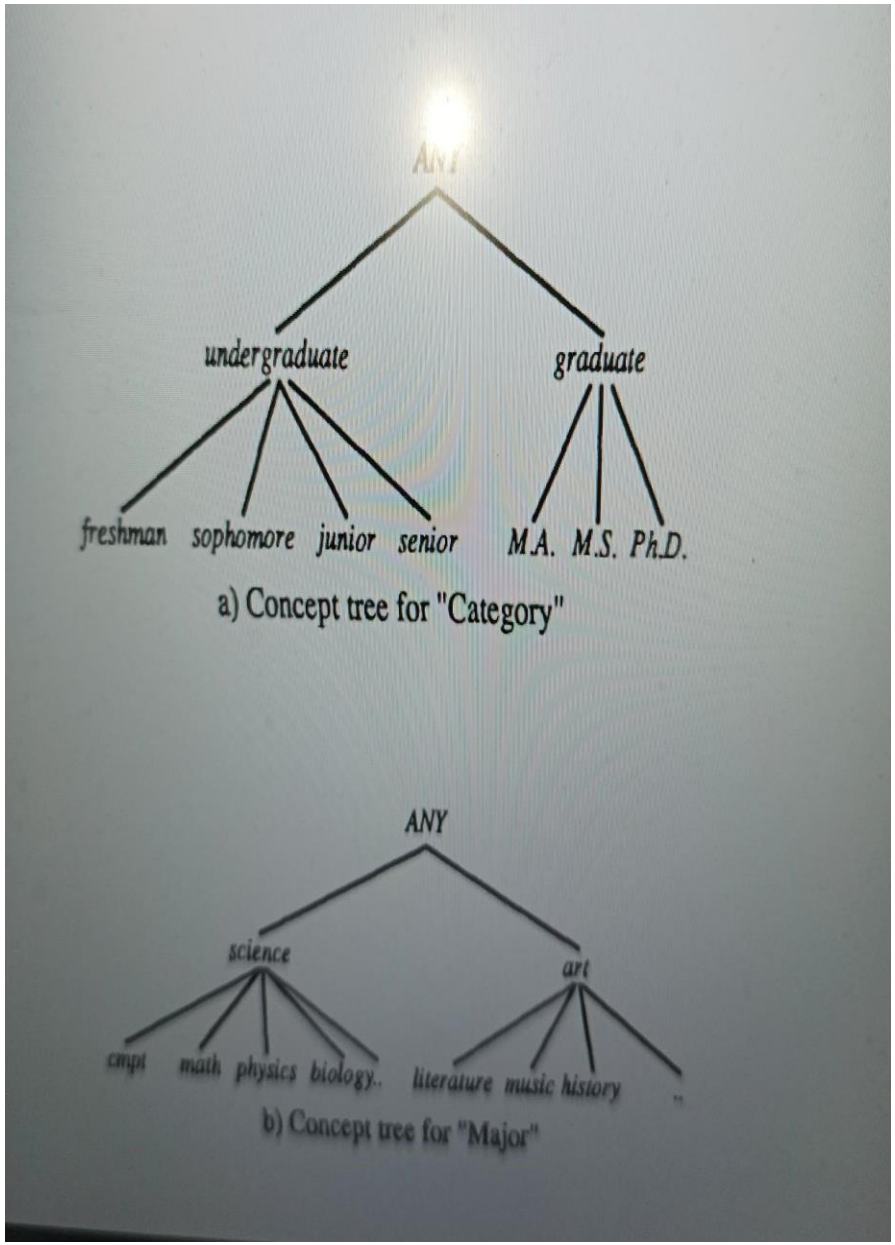
Learning Concept	Major	Birth_Place	GPA	Mark
graduate	art	B.C.	excellent	
	science	Ontario	excellent	
	science	B.C.	excellent	*
	science	India	good	
	science	China	good	
undergraduate	science	Alberta	excellent	
	art	Alberta	average	
	science	B.C.	average	
	science	B.C.	excellent	*
	art	B.C.	average	
	art	Ontario	excellent	

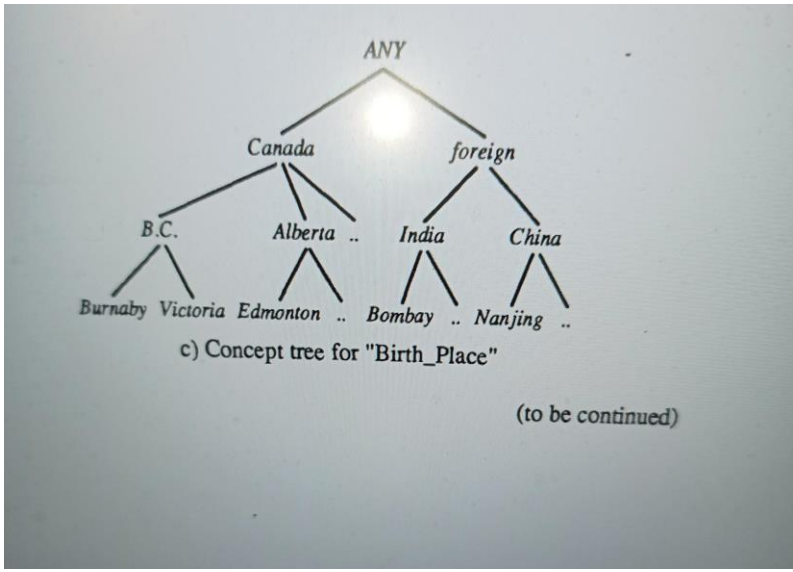
Table 4.4. A generalized relation

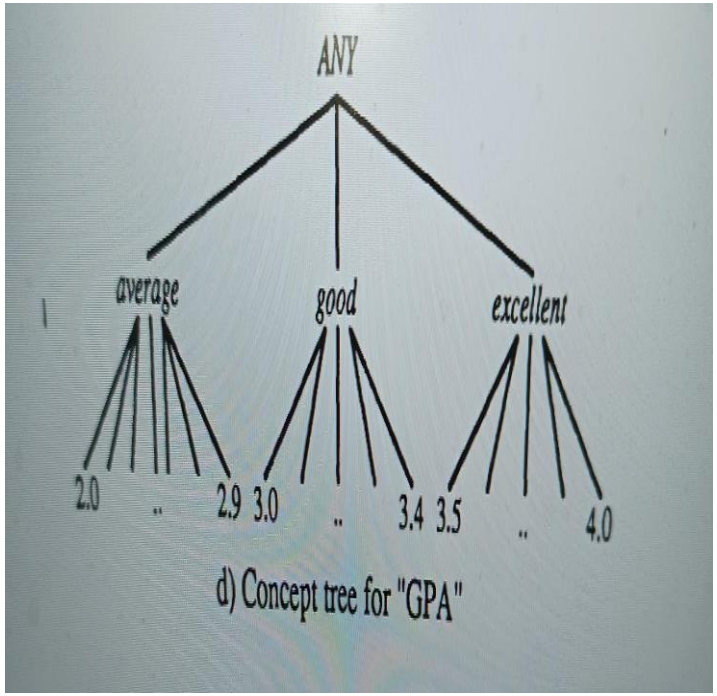
Concept Hierarchy C

- Relationship between the values in category and "graduates" should be consulted in order to extract the relevant set of data.
- We then examine C, the conceptual bias. Consider the following C, where A c B indicates that B is a"generalization" of A.

- { computing, math, biology, chemistry, statistics,physics } c science
  (music, history, liberal arts, literature) c art
  { freshman, sophomore, junior, senior) c undergraduate
  M.S., M.A., Ph.D.) c graduate
  (Bumaby, Richmond, Vancouver,
  Victoria) c British Columbia
  { Calgary, Edmonton) c Alberta
  { Ottawa, Toronto) c Ontario
  (Bombay } c India
  [Shanghai, Nanjing) c China
  (China, India) c Foreign
  [British Columbia, Alberta, Ontario) c Canada
  { 2.0 - 2.9) c average
  [3.0 -3.4) c good
  (3.5 - 4.0 } c excellent


ANY
undergraduate
graduate
freshman
sophomore
junior
senior
M.A.
M.S.
Ph.D.
a) Concept tree for "Category"
ANY
science
art
cmpt
math
physics
biology..
literature
music
history
..
b) Concept tree for "Major"


ANY
Canada
foreign
B.C.
Alberta ..
India
China
Burnaby
Victoria
Edmonton ..
Bombay ..
Nanjing ..
c) Concept tree for "Birth_Place"
(to be continued)


ANY
average
good
excellent
2.0
..
2.9
3.0
..
3.4
3.5
..
4.0
d) Concept tree for "GPA"

The set of data relevant to "graduates

Name	Category	Major	Birth_Place	GPA
Anderson	M.A.	history	Vancouver	3.5
Fraser	M.S.	physics	Ottawa	3.9
Gupta	Ph.D.	math	Bombay	3.3
Liu	Ph.D.	biology	Shanghai	3.4
Monk	Ph.D.	computing	Victoria	3.8
Wang	M.S.	statistics	Nanjing	3.2

Major	Birth_Place	GPA
ANY	Canada	excellent
science	Foreign	good

Table 3.3. The learned rule in relation table form

Outline

- Introduction
- Learning from examples (Related Work)
- Concepts of learning from databases
- **Attribute oriented induction in relational databases**
- Variations of the learning algorithms
- Discussion
- Implementation and experiments
- Conclusions and future research

Two algorithms: the LCHR algorithm and the LCLR algorithm

- Learning Characteristic Rules
  - Generalization Strategy 1 (Attribute Removal)
    - If there is a large set of distinct values for an attribute but there is no higher level concept provided for the attribute, the attribute should be removed during generalization.
    - Example: Name is the key of the relation and represents a large set of values so it should be removed.
    - Generalize to ALL or NULL
  - Generalization Strategy 2. (Generalization on the smallest decomposable components)
    - Generalization should be performed on the smallest decomposable components of a data relation.
    - Least commitment principle
      - Commitment to minimally generalized concepts
      - By generalizing lease decomposable attributes, we may discover interesting information about that components.

Two algorithms: the LCHR algorithm and the LCLR algorithm

- Learning Characteristic Rules
  - Generalization Strategy 3. (Ascending the concept tree)
    - If there are many distinct values for an attribute and there exists a higher level concept in the concept tree for the attribute, each value in the attribute of the relation should be substituted by a higher level concept in the learning process
  - Generalization Strategy 4. (Threshold control)
    - If the number of distinct values in a resulting relation is larger than the specified threshold value, further generalization on this attribute should be performed

Algorithm 4. LCHR -Learning characteristic rules from relational databases.

- Input: (i) a relational database,(ii) a concept hierarchy table,(iii) the learning task, and (iv) the threshold value (T).
- Output: A characteristic rule for the target class learned from the database
- Step 1. Select the task-relevant data relation P.
- Step 2. Perform attribute-oriented induction, which is described by the following procedure
  - Generalization is performed on each attribute of P.
- Step 3: Simplify the generalized relation
  - If only one attribute of several tuples contains distinct values, the several tuples can be reduced into one by taking the distinct values of that attribute as a set.
- Step 4: Transform the final relation into logic formulas

Learning Classification Rule

- Similar to the LCHR algorithm, the LCLR algorithms also applies the attribute-oriented induction technique.

- The difference is that in the extraction of classification rules, the facts which support the target class serve as positive examples, while the facts which support the other classes serve as negative examples.

- Since the learning task is to discover the concepts that have discriminant properties**, the portion of facts in the target class that overlaps with other classes should be detected and removed from the description of classification rules**

Overlapping rules

- Overlapping tuples
  Definition: A set of overlapping tuples is a set of tuples which are shared by different classes.

- Generalization Strategy 5. (Handling overlapping tuples)
  - If there are overlapping tuples in both target and contrasting classes, these tuples should be marked and eliminated from the final generalized relation

LCLR -Learning classification rules from relational database

- Input: (i) a relational database,(ii) a concept hierarchy table,(iii) the learning task, and (iv) the threshold value (T).
- Output: A classification rule for the target class learned from the database
- Step 1. Select the task-relevant data of the target class and the contrasting class to form relation P and cluster the data by classes.
- Step 2. Perform attribute-oriented induction, which is described by the following procedure
  - Generalization is performed on each attribute Ai of each class.
  - Perform intersection of both classes and mark the overlapping tuples;
- Step 3: Remove the overlapping tuples
- Step 4: Transform the final relation into logic formula

Outline

- Introduction
- Learning from examples (Related Work)
- Concepts of learning from databases
- Attribute oriented induction in relational databases
- **Variations of the learning algorithms**
- **Discussion**
- **Implementation and experiments**
- **Conclusions and future research**

Variations of the algorithms

- Adjusting Thresholds for Different Learning Results
  - The output vary based on the thresholds
- Dealing with Different Kinds of Concept Hierarchies
  - Balanced and imbalanced concept hierarchies
- Generalization of Concepts in the Hierarchies with Lattice
- Incorporating Quantitative Information
  - Addition of attribute “votes”, how many tuples have contributed to this rule?
- Handling Noisy Data and Exceptional Cases

DISCUSSION

- Limitations
  - Assumption:
    - availability of concept hierarchy data
    - Applicable on only formatted data

Experiments

- Refer research paper

Conclusion and Future work

- Conclusion
  - Our analysis of the algorithms demonstrates that the attribute-oriented induction approach substantially reduces the complexity of the database learning process.
- Future Work
  - Applications of Knowledge Rules Discovered from Relational Database
    - Discovery of knowledge rules for knowledge-base systems and expert systems
    - Processing of queries which involve abstract concepts.
    - Semantic query optimization using the learned rules
    - Construction of an Interactive Learning System
  - Discovery of Concept Hierarchies

Data Warehousing and OLAP Technology
Class #6

- Chapter 3 of the textbook

Detailed Syllabus

- Introduction (1.5 hour): Definition, KDD framework, Issues in data mining.
- Data summarization (7.5 hrs): Data Types, Preprocessing, Characterization, Discrimination, **data warehousing techniques (Data warehousing technology,** Data cube computation**)**
- Concepts and algorithms for mining patterns and associations (9 hours) (Frequent item-set generation, A priori and FP-growth algorithm, Evaluation of Association patterns) and preprocessing
- Concepts and algorithms related to classification and regression (9hrs) (Overview, Decision tree induction, Over-fitting and under-fitting, Scalable decision tree algorithms, Bayesian Classification, Regression-based Prediction methods (9 hours)
- Concepts and algorithms for clustering the data (9 hours) (Overview, Types of Data, K-means, Aglomerative clustering, Clustering algorithms (DBSCAN, BIRCH, CURE, ROCK, CHAMELEON)).
- Outlier analysis and future trends (graph mining, spatio-temporal mining). (3 hours)

- Data warehousing is the framework to generalize huge multi-dimensional databases

Data Warehousing and OLAP Technology

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary

What is Data Warehouse?

- Defined in many different ways, but not rigorously.
  - A decision support database that is maintained separately from the organization's operational database
  - Support information processing by providing a solid platform of consolidated, historical data for analysis.
- "A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision-making process."—W. H. Inmon
- Data warehousing:
  - The process of constructing and using data warehouses

Data Warehouse—Subject-Oriented

- Organized around major subjects, such as customer, product, sales
- Focusing on the modeling and analysis of data for decision makers, not on daily operations or transaction processing
- Provide a simple and concise view around particular subject issues by excluding data that are not useful in the decision support process

Data Warehouse—Integrated

- Constructed by integrating multiple, heterogeneous data sources
  - relational databases, flat files, on-line transaction records
- **Data cleaning and data integration techniques** are applied.
  - Ensure consistency in naming conventions, encoding structures, attribute measures, etc. among different data sources
    - E.g., Hotel price: currency, tax, breakfast covered, etc.
  - When data is moved to the warehouse, it is converted.

Data Warehouse—Time Variant

- The time horizon for the data warehouse is significantly longer than that of operational systems
  - Operational database: current value data
  - Data warehouse data: provide information from a historical perspective (e.g., past 5-10 years)
- Every key structure in the data warehouse
  - Contains an element of time, explicitly or implicitly
  - But the key of operational data may or may not contain "time element"

Data Warehouse—Nonvolatile

- A physically separate store of data transformed from the operational environment
- Operational update of data does not occur in the data warehouse environment
  - Does not require transaction processing, recovery, and concurrency control mechanisms
  - Requires only two operations in data accessing:
    - *initial loading of data* and *access of data*

Data Warehouse vs. Heterogeneous DBMS

- Traditional heterogeneous DB integration: A query driven approach
  - Build wrappers/mediators on top of heterogeneous databases
  - When a query is posed to a client site, a meta-dictionary is used to translate the query into queries appropriate for individual heterogeneous sites involved, and the results are integrated into a global answer set
  - Complex information filtering, compete for resources
- Data warehouse: update-driven, high performance
  - Information from heterogeneous sources is integrated in advance and stored in warehouses for direct query and analysis

Data Warehouse vs. Operational DBMS

- OLTP (on-line transaction processing)
  - Major task of traditional relational DBMS
  - Day-to-day operations: purchasing, inventory, banking, manufacturing, payroll, registration, accounting, etc.
- OLAP (on-line analytical processing)
  - Major task of data warehouse system
  - Data analysis and decision making
- Distinct features (OLTP vs. OLAP):
  - User and system orientation: customer vs. market
  - Data contents: current, detailed vs. historical, consolidated
  - Database design: ER + application vs. star + subject
  - View: current, local vs. evolutionary, integrated
  - Access patterns: update vs. read-only but complex queries

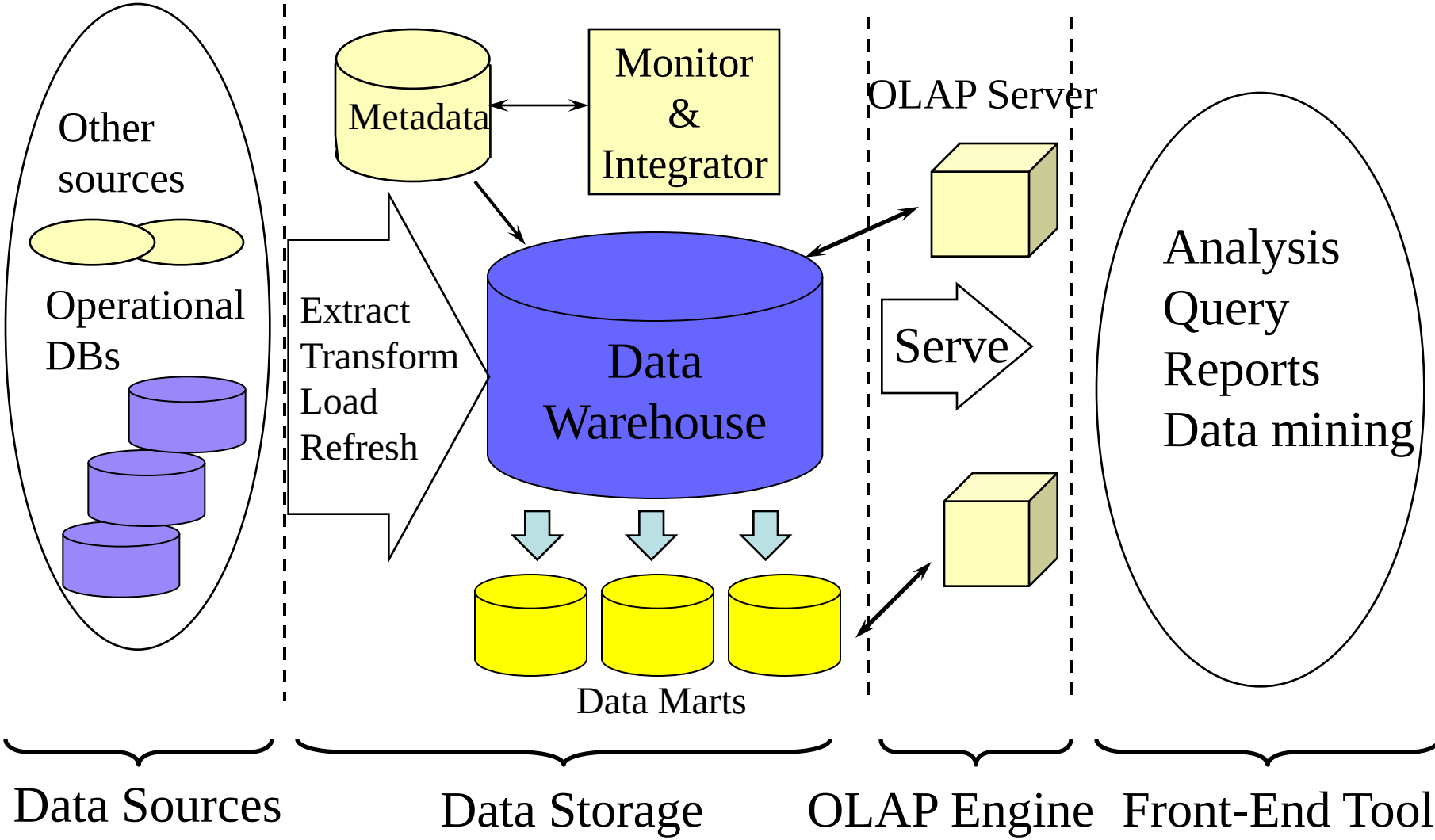
OLTP vs. OLAP

	OLTP	**OLAP**
users	clerk, IT professional	knowledge worker
function	day to day operations	decision support
DB design	application-oriented	subject-oriented
data	current, up-to-date detailed, flat relational isolated	historical, summarized, multidimensional integrated, consolidated
usage	repetitive	ad-hoc
access	read/write index/hash on prim. key	lots of scans
unit of work	short, simple transaction	complex query
# records accessed	tens	millions
#users	thousands	hundreds
DB size	100MB-GB	100GB-TB
metric	transaction throughput	query throughput, response

Why Separate Data Warehouse?

- High performance for both systems
  - DBMS— tuned for OLTP: access methods, indexing, concurrency control, recovery
  - Warehouse—tuned for OLAP: complex OLAP queries, multidimensional view, consolidation
- Different functions and different data:
  - missing data: Decision support requires historical data which operational DBs do not typically maintain
  - data consolidation: DS requires consolidation (aggregation, summarization) of data from heterogeneous sources
  - data quality: different sources typically use inconsistent data representations, codes and formats which have to be reconciled
- Note: There are more and more systems which perform OLAP analysis directly on relational databases

Data Warehouse: A Multi-Tiered Architecture



Three Data Warehouse Models

- Enterprise warehouse
  - collects all of the information about subjects spanning the entire organization
- Data Mart
  - a subset of corporate-wide data that is of value to a specific groups of users. Its scope is confined to specific, selected groups, such as marketing data mart
    - Independent vs. dependent (directly from warehouse) data mart
- Virtual warehouse
  - A set of views over operational databases
  - Only some of the possible summary views may be materialized

Extraction, Transformation, and Loading (ETL)

- **Data extraction**
  - get data from multiple, heterogeneous, and external sources
- **Data cleaning**
  - detect errors in the data and rectify them when possible
- **Data transformation**
  - convert data from legacy or host format to warehouse format
- **Load**
  - sort, summarize, consolidate, compute views, check integrity, and build indices and partitions
- **Refresh**
  - propagate the updates from the data sources to the warehouse

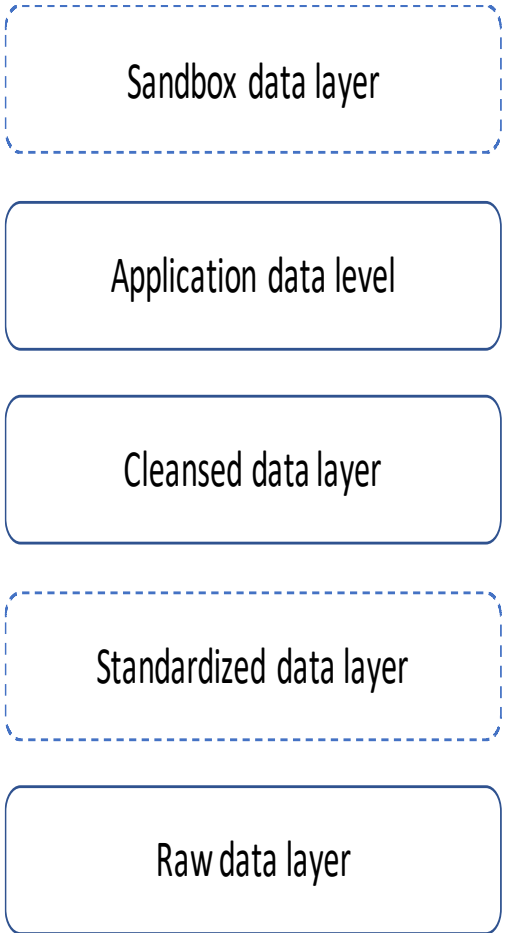
Metadata Repository

- **Meta data** is the data defining warehouse objects. It stores:
  - Description of the structure of the data warehouse
    - schema, view, dimensions, hierarchies, derived data defn, data mart locations and contents
  - Operational meta-data
    - data lineage (history of migrated data and transformation path), currency of data (active, archived, or purged), monitoring information (warehouse usage statistics, error reports, audit trails)
  - The algorithms used for summarization
  - The mapping from operational environment to the data warehouse
  - Data related to system performance
    - warehouse schema, view and derived data definitions
  - Business data
    - business terms and definitions, ownership of data, charging policies

Data Lake

- A data lake is a centralized repository storing all structured and unstructured data at any scale in an organization
- Data is stored as is,
  - without having first to structure the data, and run different types of analytics—from dashboards and visualizations to big data processing, real-time analytics, and machine learning to guide better decision
- https://azure.microsoft.com/en-in/resources/cloud-computing-dictionary/what-is-a-data-lake

Layers of Storage



Enriching data by data scientists and analysts

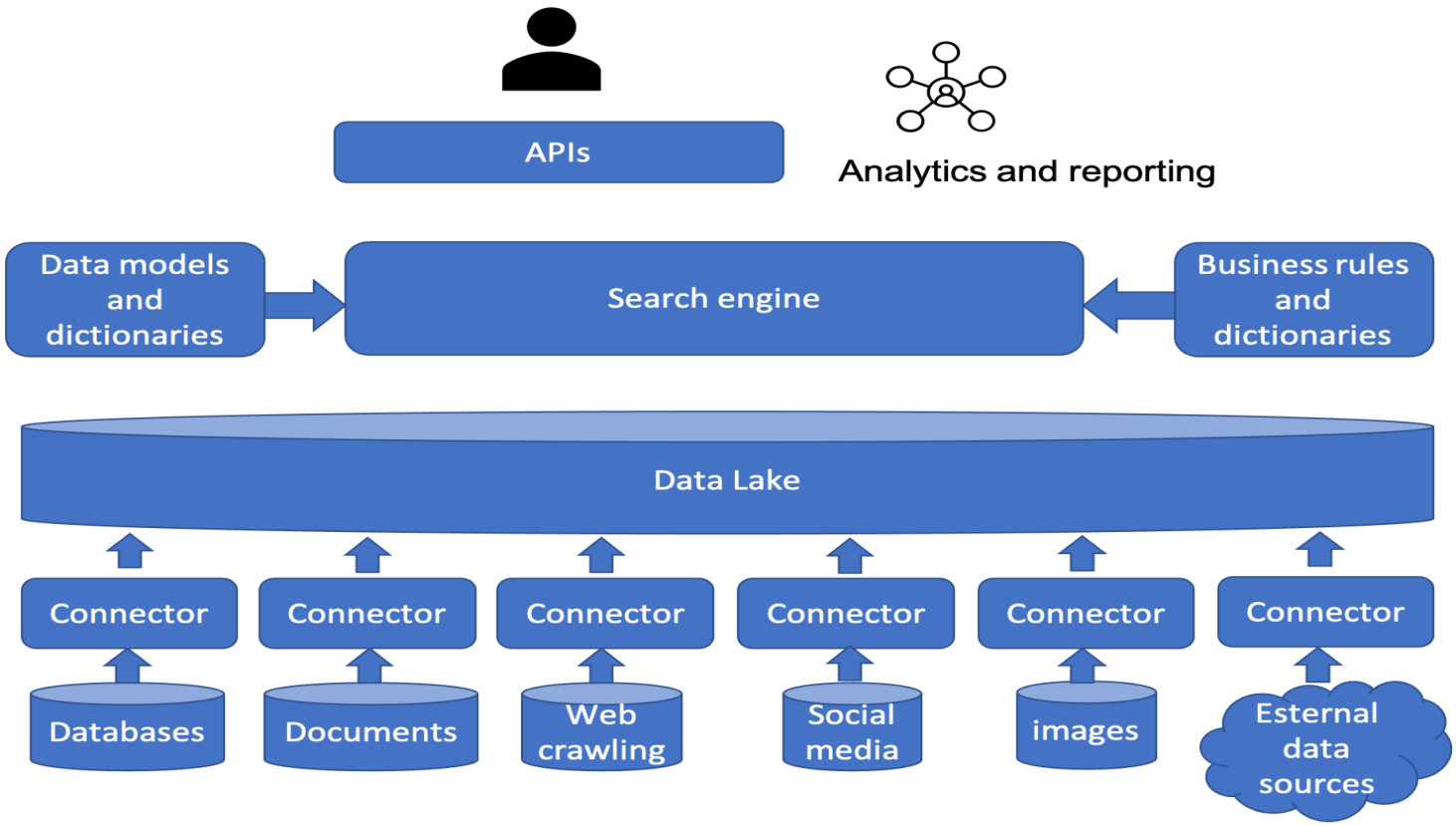
Implementing business logics

Cleansing and transforming data

Facilitating efficient data transferring and cleansing

Loading raw data

Conceptual Architecture



Two Purposes and Ecosystems

- Analytic use cases: decision support based on data
  - Data warehouses
  - Structured data, SQL/Python
- Operational use cases: data intelligence in applications
  - Data lakes
  - Raw data, Java/Scala, Python, R, and SQL
- Potential convergence of data warehouses and data lakes

Data Lakehouse

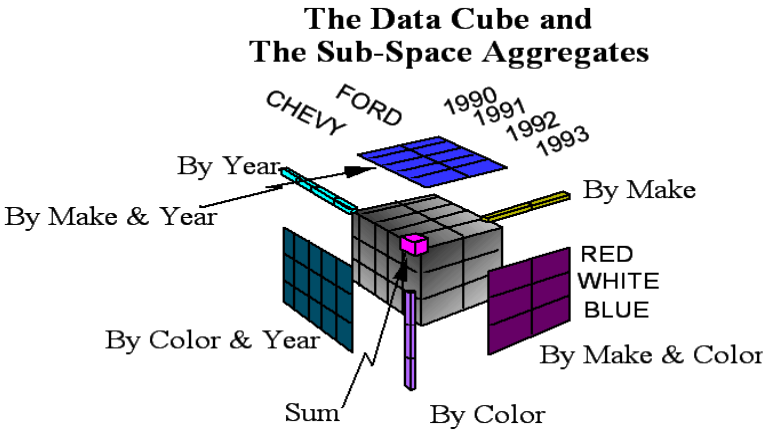
- A data lakehouse is an open standards-based storage solution that is multifaceted in nature.
  - It can address the needs of data scientists and engineers who conduct deep data analysis and processing and traditional data warehouse professionals who curate and publish data for business intelligence and reporting purposes.
  - The beauty of the lakehouse is that each workload can seamlessly operate on top of the data lake without having to duplicate the data into another structurally predefined database.
  - This ensures that everyone is working on the most up-to-date data while also reducing redundancies.

Data Warehousing and OLAP Technology: An Overview

- Data Warehouse: Basic Concepts
- **Data Warehouse Modeling: Data Cube and OLAP**
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary

CUBE Operator

- Problems with GROUP BY
  - GROUP BY cannot directly construct:
    - Histograms
    - Roll-Up Reports
    - Cross-Tabs
- CUBE Operator
  - Generalize GROUP BY and Roll-Up and Cross-Tabs!!

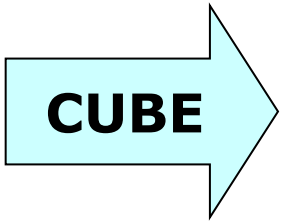


CUBE Operator
Think of the N-dimensional Cube

- N-dimensional Aggregate [sum(), max(),...]
  - Fits relational model exactly**:**
    - **a_1, a_2, ..., a_N, f()**
- Super-aggregate over *N-1* Dimensional sub-cubes
    - **ALL, a_2, ..., a_N , f()**
    - **a_3 , ALL, a_3, ..., a_N , f()**
    - **...**
    - **a_1, a_2, ..., ALL, f()**
  - This is the *N-1* Dimensional cross-tab**.**
- Super-aggregate over *N-2* Dimensional sub-cubes
    - **ALL, ALL, a_3, ..., a_N , f()**
    - **...**
    - **a_1, a_2 ,...., ALL, ALL, f()**
- …

CUBE Operator
An Example

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



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

- Think of ALL as a token representing the set
  - **{red, white, blue}**
  - **{1990, 1991, 1992}**
  - **{Chevy, Ford}**

From Tables and Spreadsheets to Data Cubes

- A data warehouse is based on a multidimensional data model which views data in the form of a data cube
- A data cube, such as sales, allows data to be modeled and viewed in multiple dimensions
  - Dimension tables, such as item (item_name, brand, type), or time(day, week, month, quarter, year)
  - Fact table contains measures (such as dollars_sold) and keys to each of the related dimension tables
- In data warehousing literature, an n-D base cube is called a base cuboid. The top most 0-D cuboid, which holds the highest-level of summarization, is called the apex cuboid. The lattice of cuboids forms a data cube.

Table 4.2 2-D View of Sales Data for *AllElectronics* According to *time* and *item*

location = "Vancouver"				
	item (type)			
time (quarter)	home entertainment	computer	phone	security
Q1	605	825	14	400
Q2	680	952	31	512
Q3	812	1023	30	501
Q4	927	1038	38	580

Note: The sales are from branches located in the city of Vancouver. The measure displayed is *dollars_sold* (in thousands).

Table 4.3 3-D View of Sales Data for *AllElectronics* According to *time*, *item*, and *location*

	location = "Chicago"				**location** = "New York"				**location** = "Toronto"				**location** = "Vancouver"			
	item				**item**				**item**				**item**			
time	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

Note: The measure displayed is *dollars_sold* (in thousands).

in this way, we may display any n-dimensional data as a series of $(n-1)$-dimensional

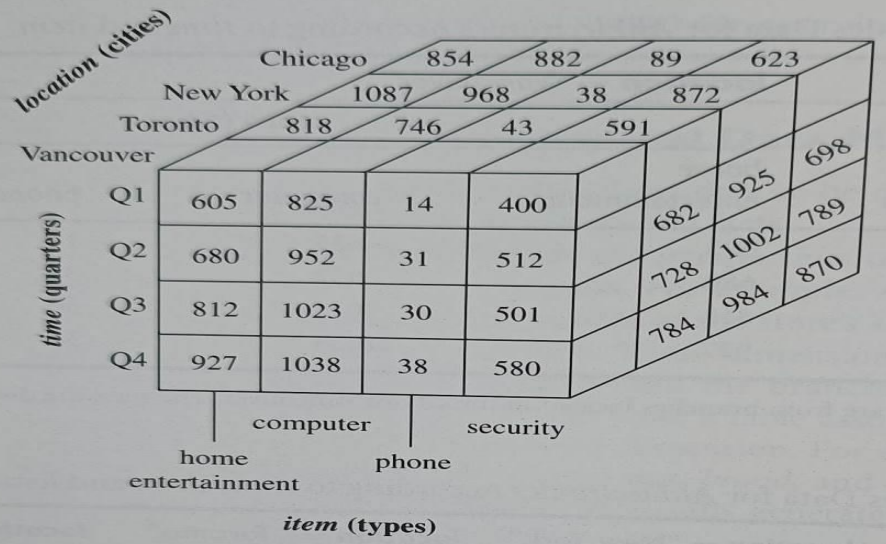


Figure 4.3 A 3-D data cube representation of the data in Table 4.3, according to *time*, *item*, and *location*. The measure displayed is *dollars_sold* (in thousands).

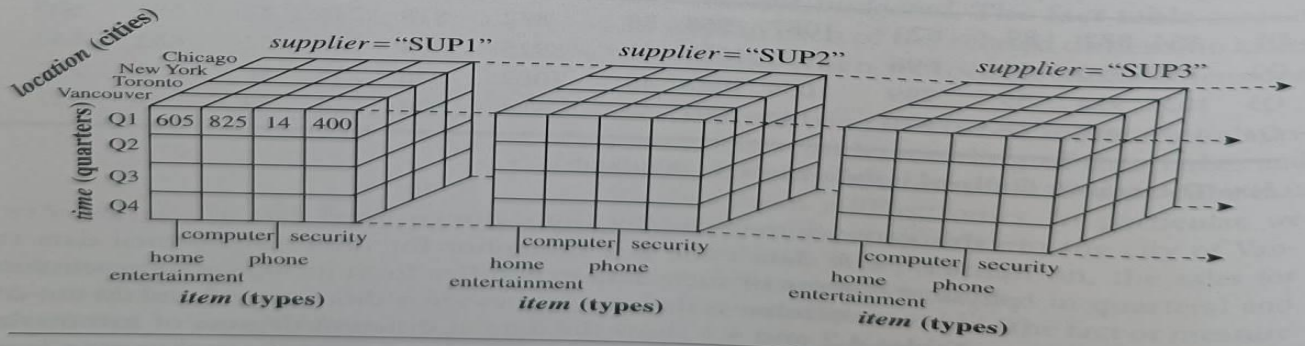
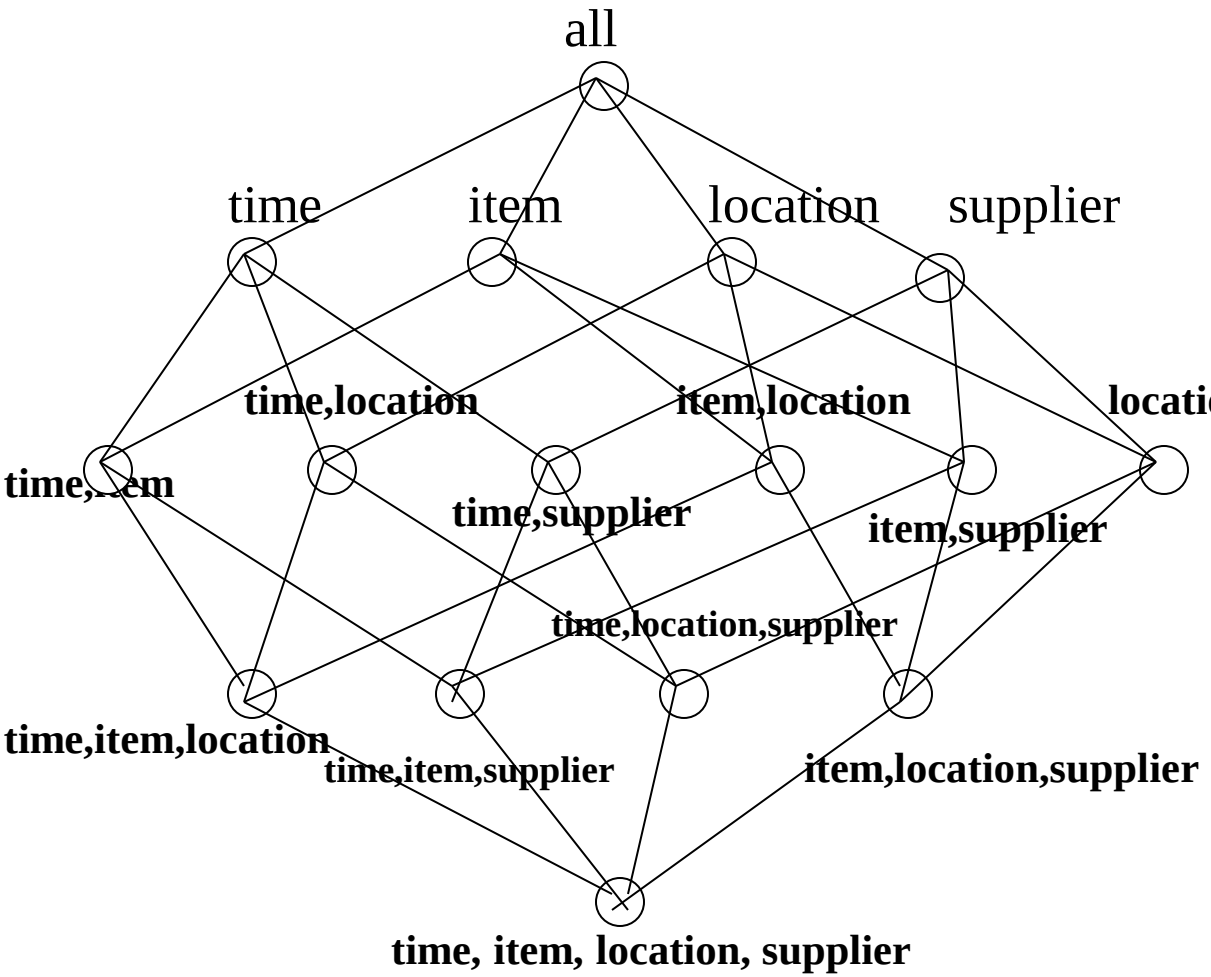


Figure 4.4 A 4-D data cube representation of sales data, according to *time*, *item*, *location*, and *supplier*. The measure displayed is *dollars_sold* (in thousands). For improved readability, only some of the cube values are shown.

and *location*, summarized for all suppliers. The 0-D cuboid, which

Cube: A Lattice of Cuboids



0-D(apex) cuboid

1-D cuboids

2-D cuboids

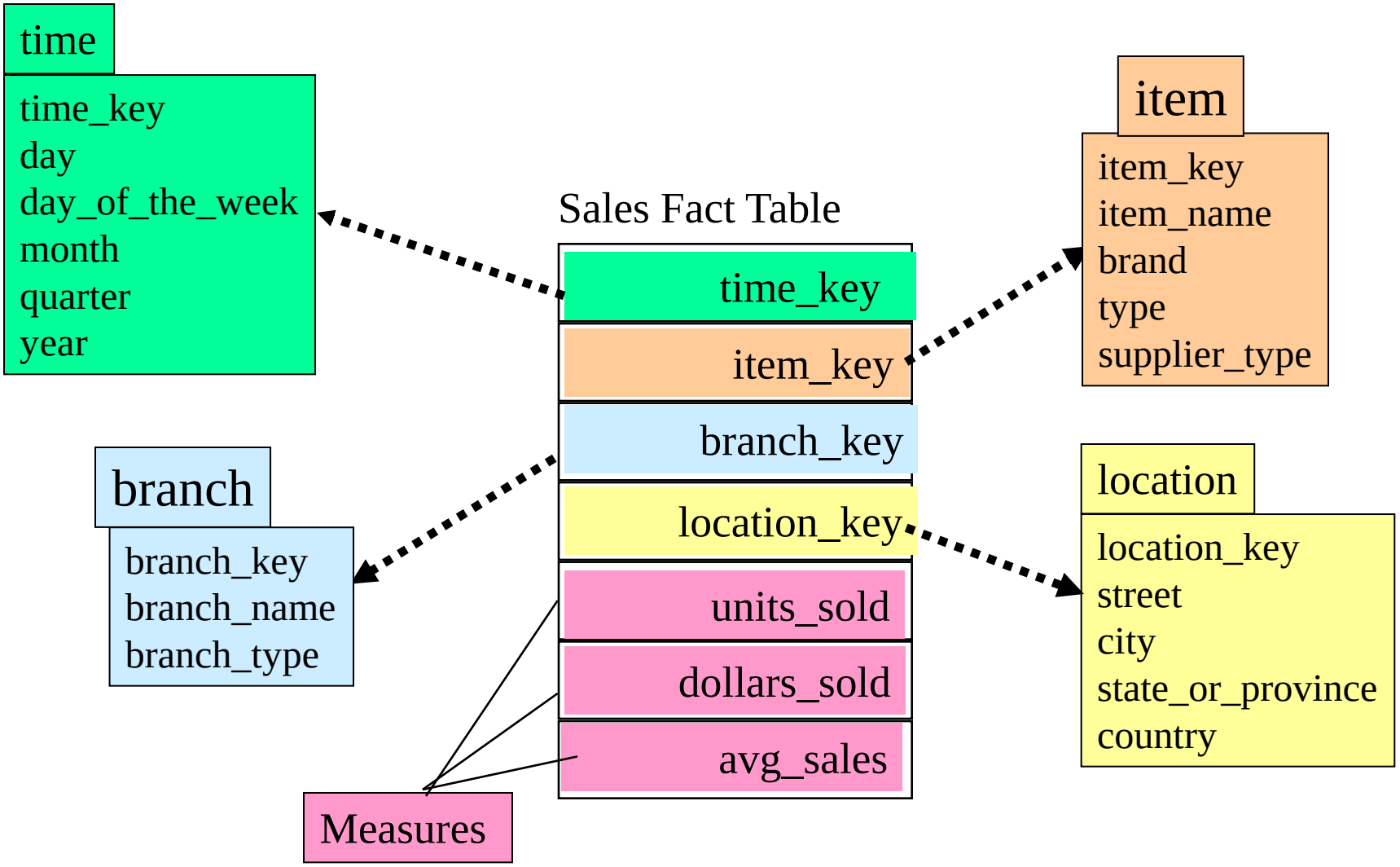
3-D cuboids

4-D(base) cuboid

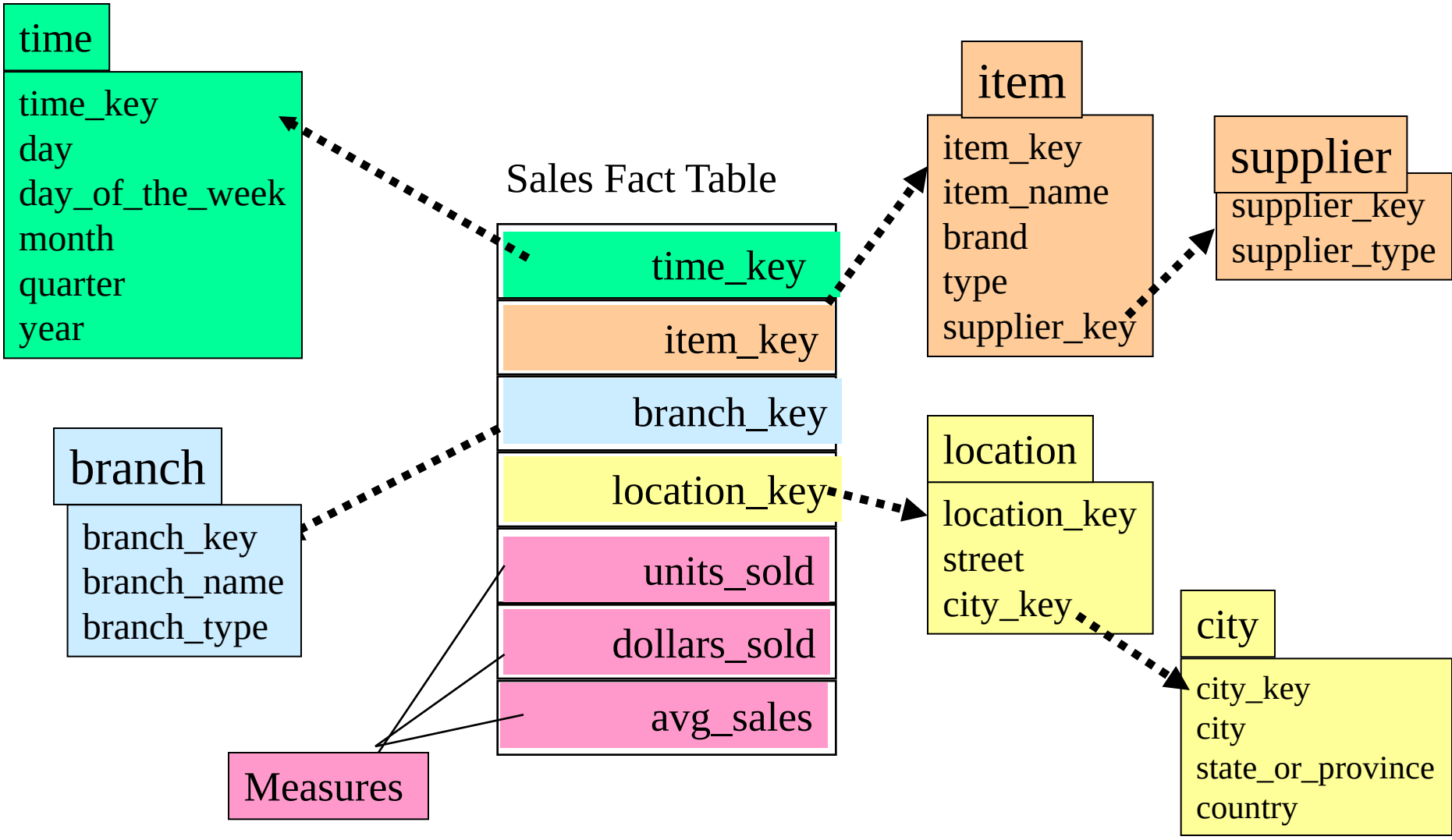
Conceptual Modeling of Data Warehouses

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

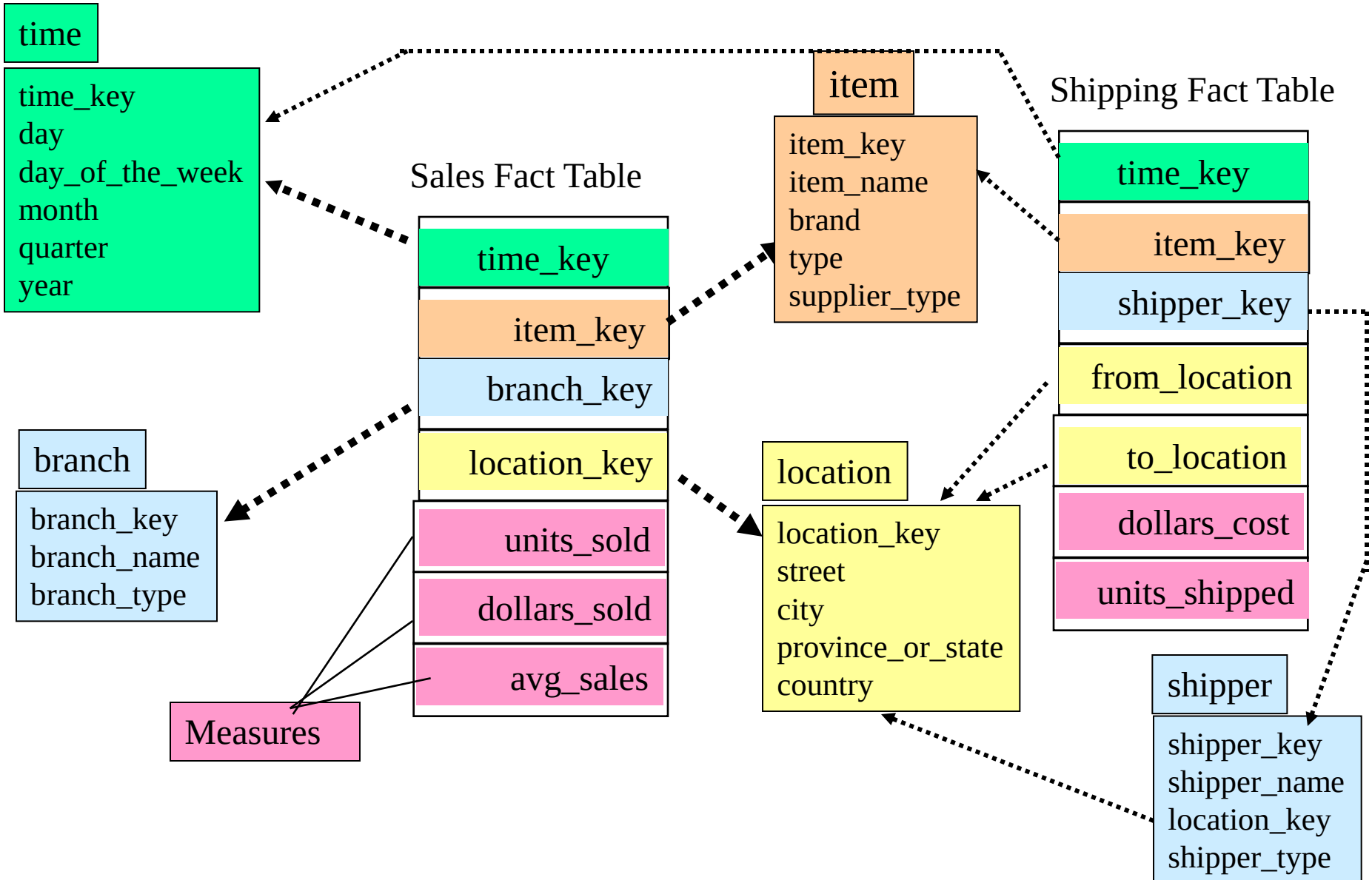
Example of Star Schema



Example of Snowflake Schema



Example of Fact Constellation



Cube Definition Syntax (BNF) in Data Mining Query Language (DMQL)

- Cube Definition (Fact Table)

  define cube <cube_name> [<dimension_list>]:
  <measure_list>

- Dimension Definition (Dimension Table)

  define dimension <dimension_name> as
  (<attribute_or_subdimension_list>)

- Special Case (Shared Dimension Tables)
  - First time as “cube definition”
  - define dimension <dimension_name> as <dimension_name_first_time> in cube <cube_name_first_time>

Defining Star Schema in DMQL

define cube sales_star [time, item, branch, location]:
dollars_sold = sum(sales_in_dollars), avg_sales =
avg(sales_in_dollars), units_sold = count(*)

define dimension time as (time_key, day, day_of_week, month, quarter, year)

define dimension item as (item_key, item_name, brand, type, supplier_type)

define dimension branch as (branch_key, branch_name, branch_type)

define dimension location as (location_key, street, city, province_or_state, country)

Defining Snowflake Schema in DMQL

define cube sales_snowflake [time, item, branch, location]:
dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars),
units_sold = count(*)

define dimension time as (time_key, day, day_of_week, month, quarter, year)

define dimension item as (item_key, item_name, brand, type,
supplier(supplier_key, supplier_type))

define dimension branch as (branch_key, branch_name, branch_type)

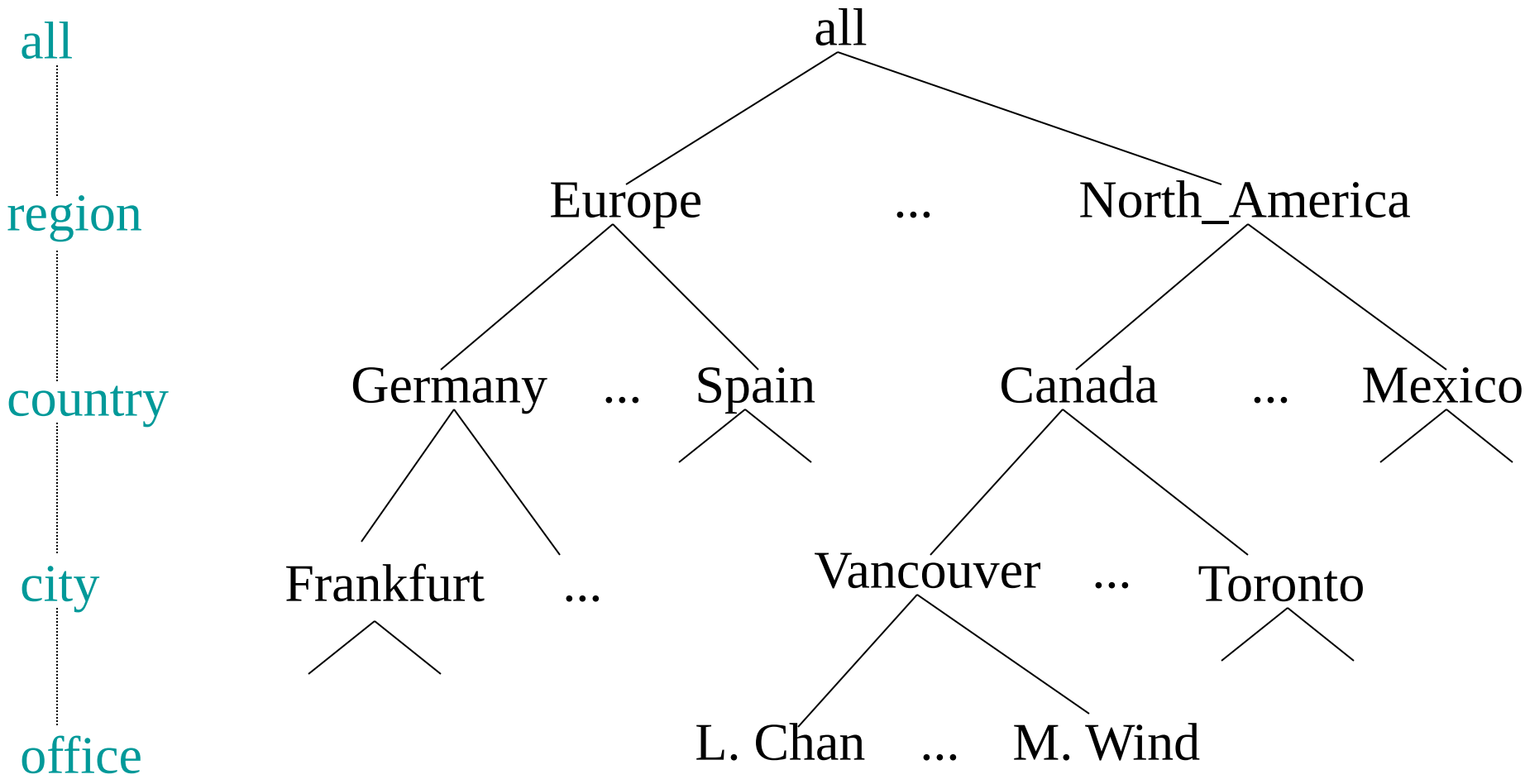
define dimension location as (location_key, street, city(city_key,
province_or_state, country))

Defining Fact Constellation in DMQL

```
define cube sales [time, item, branch, location]:
        dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars), units_sold = count(*)
define dimension time as (time_key, day, day_of_week, month, quarter, year)
define dimension item as (item_key, item_name, brand, type, supplier_type)
define dimension branch as (branch_key, branch_name, branch_type)
define dimension location as (location_key, street, city, province_or_state, country)
define cube shipping [time, item, shipper, from_location, to_location]:
        dollar_cost = sum(cost_in_dollars), unit_shipped = count(*)
define dimension time as time in cube sales
define dimension item as item in cube sales
define dimension shipper as (shipper_key, shipper_name, location as location in cube sales,
    shipper_type)
define dimension from_location as location in cube sales
define dimension to_location as location in cube sales
```

A Concept Hierarchy: Dimension (location)

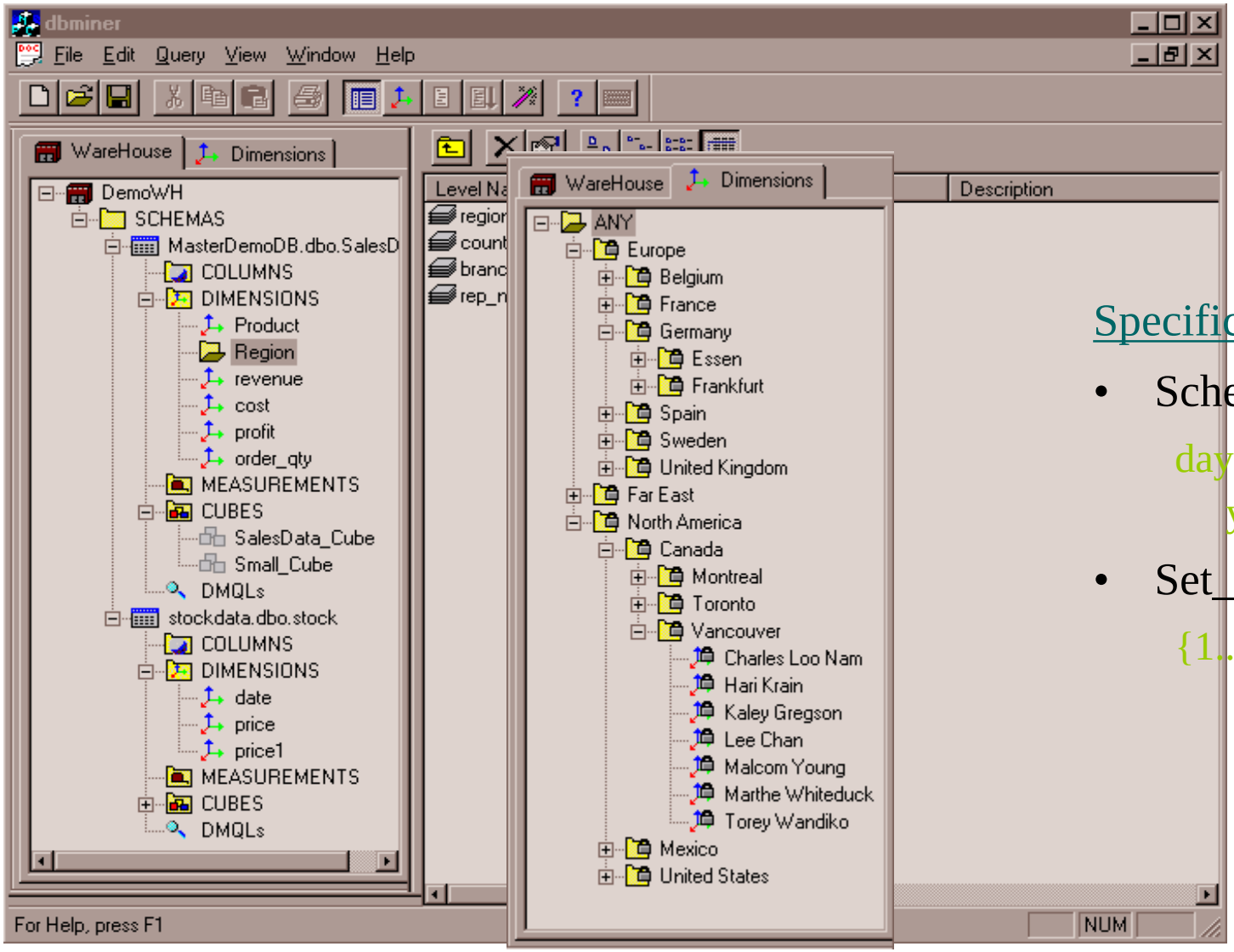
Definition: A concept hierarchy defines a sequence of mappings from a set of low-level concepts to higher-level, more general concepts.



Measures of Data Cube: Three Categories

- A measure in data cube is a numeric function which can be computed at each point in data space.
- Distributive: if the result derived by applying the function to *n* aggregate values is the same as that derived by applying the function on all the data without partitioning
    - E.g., count(), sum(), min(), max()
- Algebraic: if it can be computed by an algebraic function with *M* arguments (where *M* is a bounded integer), each of which is obtained by applying a distributive aggregate function
    - E.g., avg(), min_N(), standard_deviation()
    - Avg() can be computed by sum()/count() with two arguments, both sum() and count() are distributed aggregate functions.
- Holistic: if there is no constant bound on the storage size needed to describe a sub-aggregate.
    - E.g., median(), mode(), rank()

View of Warehouses and Hierarchies



Specification of hierarchies

- Schema hierarchy

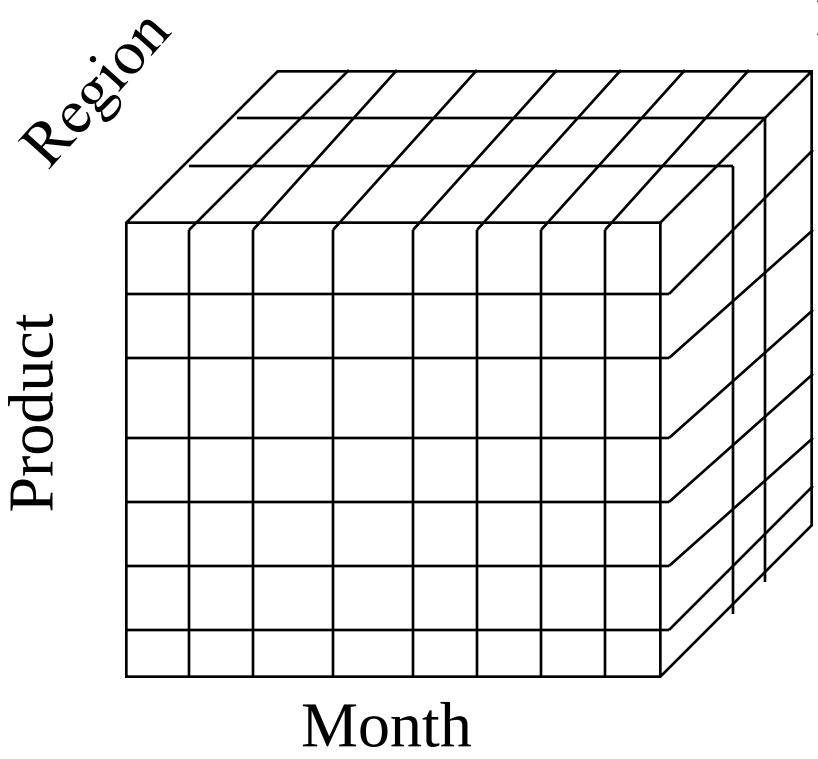
  day < {month < quarter; w
  year

- Set_grouping hierarchy

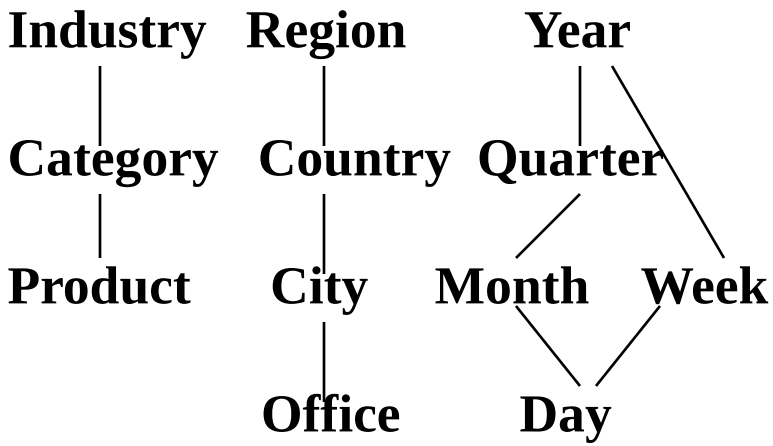
  {1..10} < inexpensive

Multidimensional Data

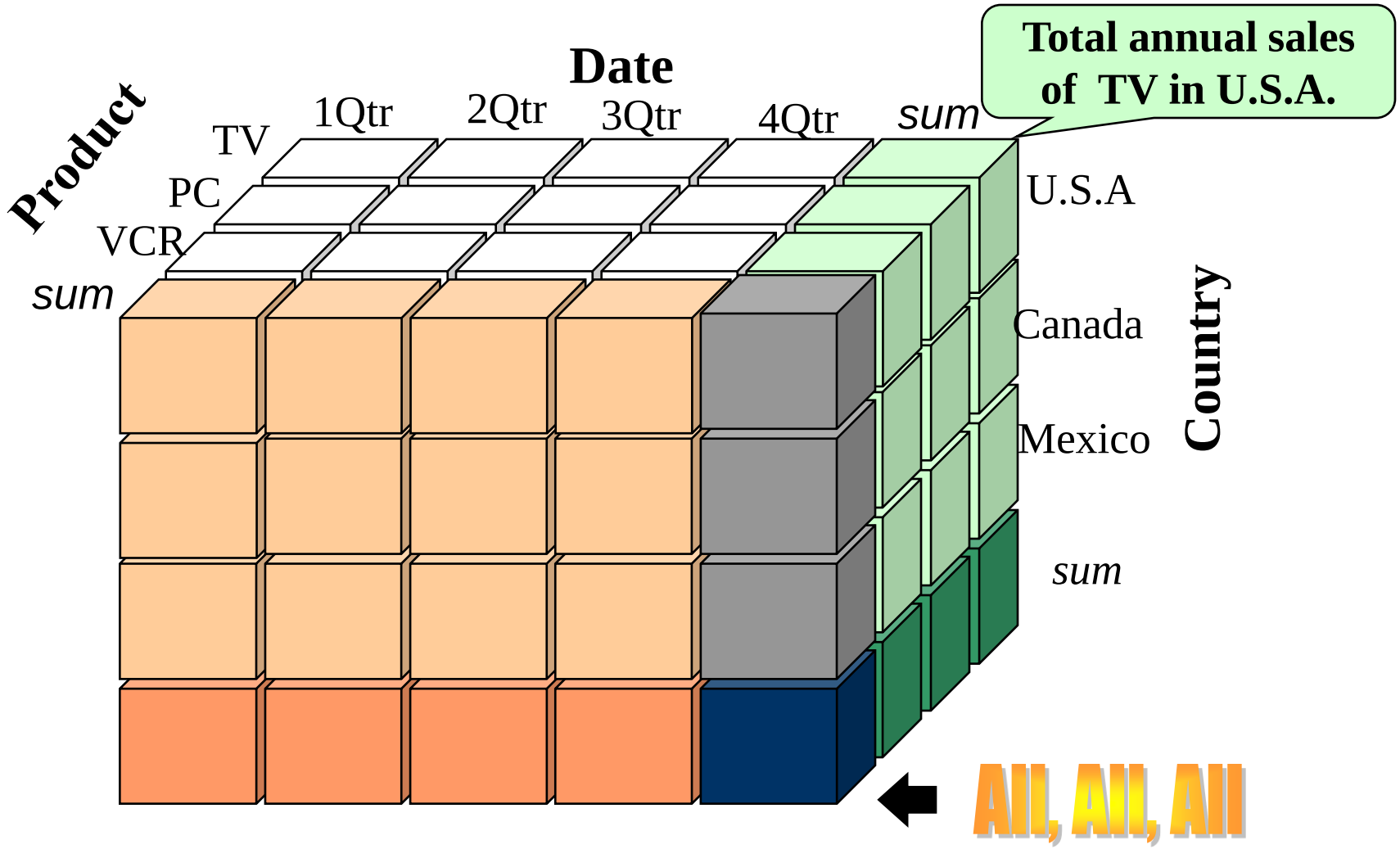
- Sales volume as a function of product, month, and region



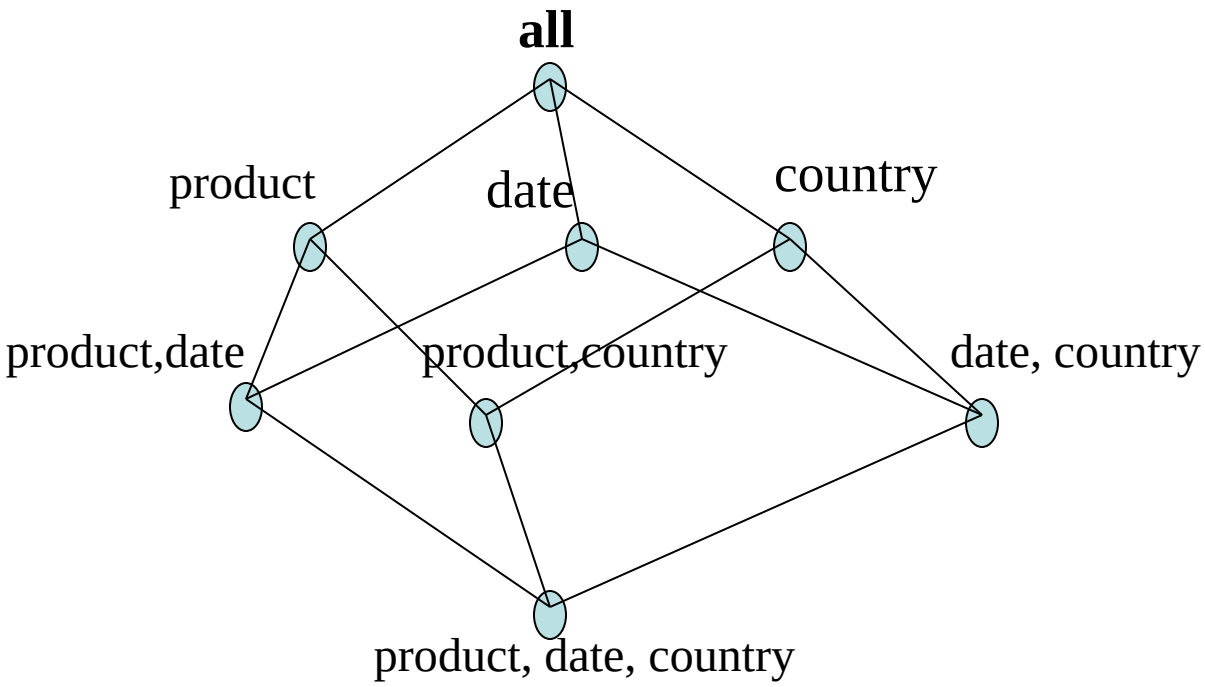
Dimensions: Product, Location, Time
Hierarchical summarization paths



A Sample Data Cube



Cuboids Corresponding to the Cube



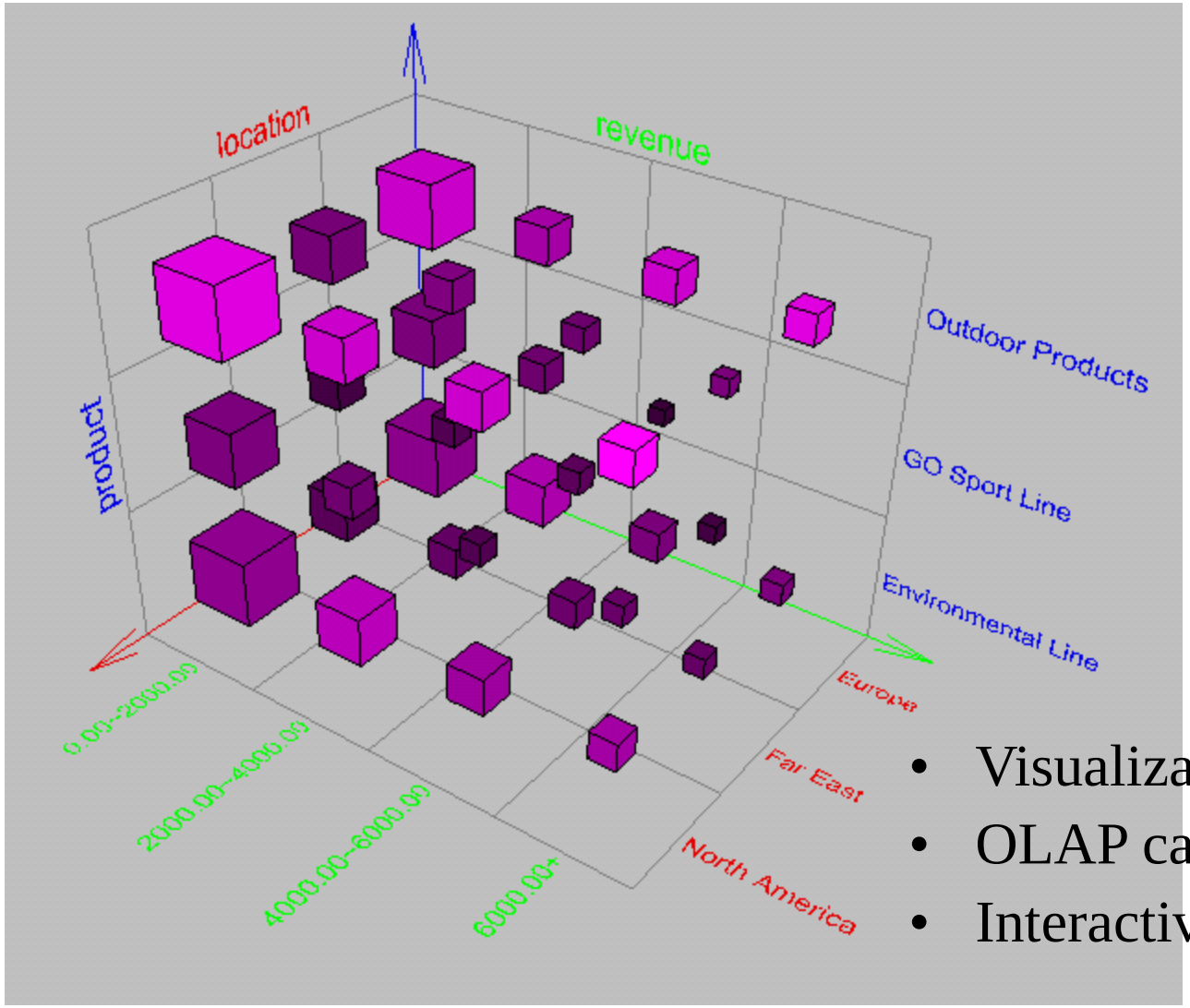
0-D(apex) cuboid

1-D cuboids

2-D cuboids

3-D(base) cuboid

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)

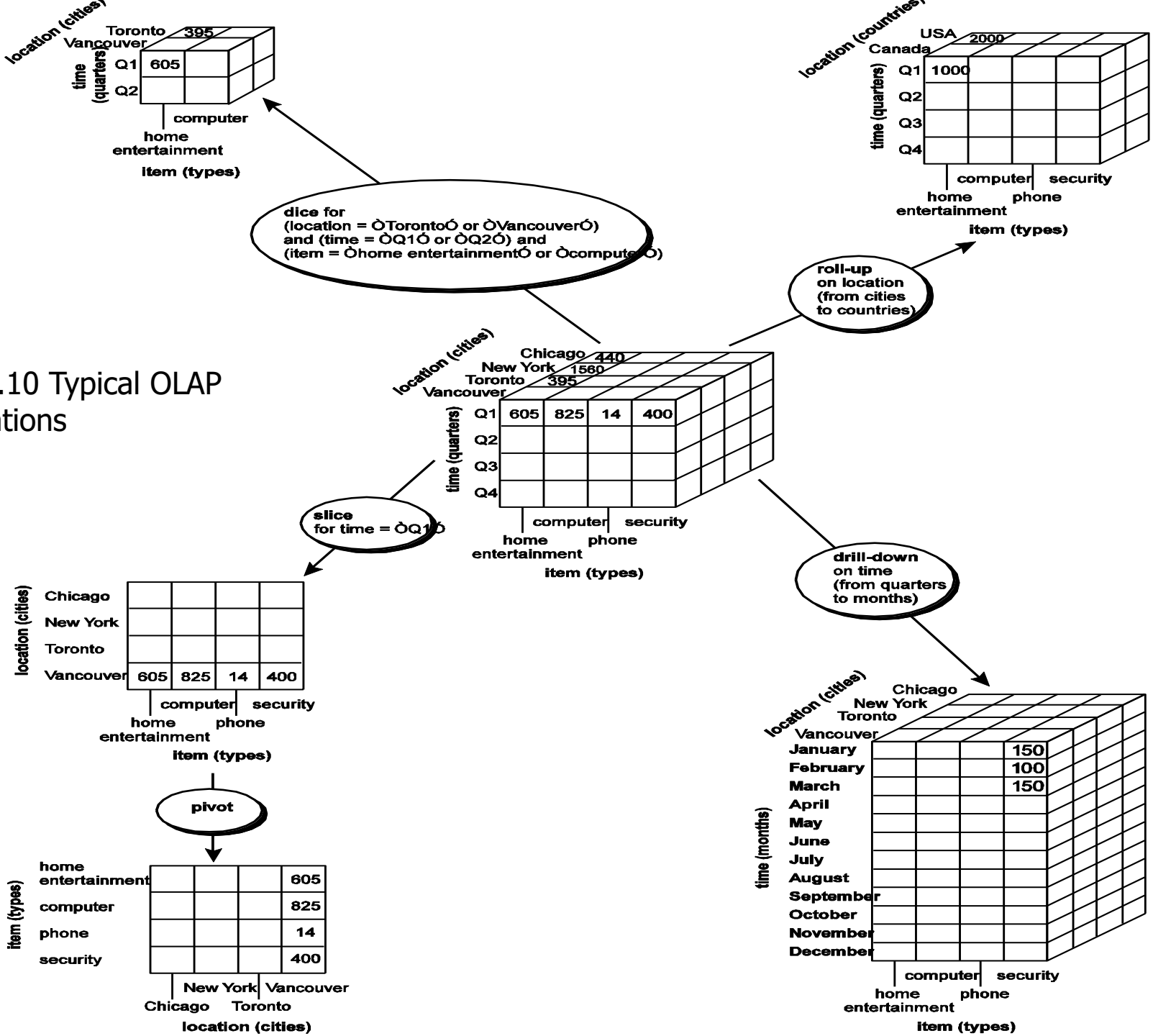
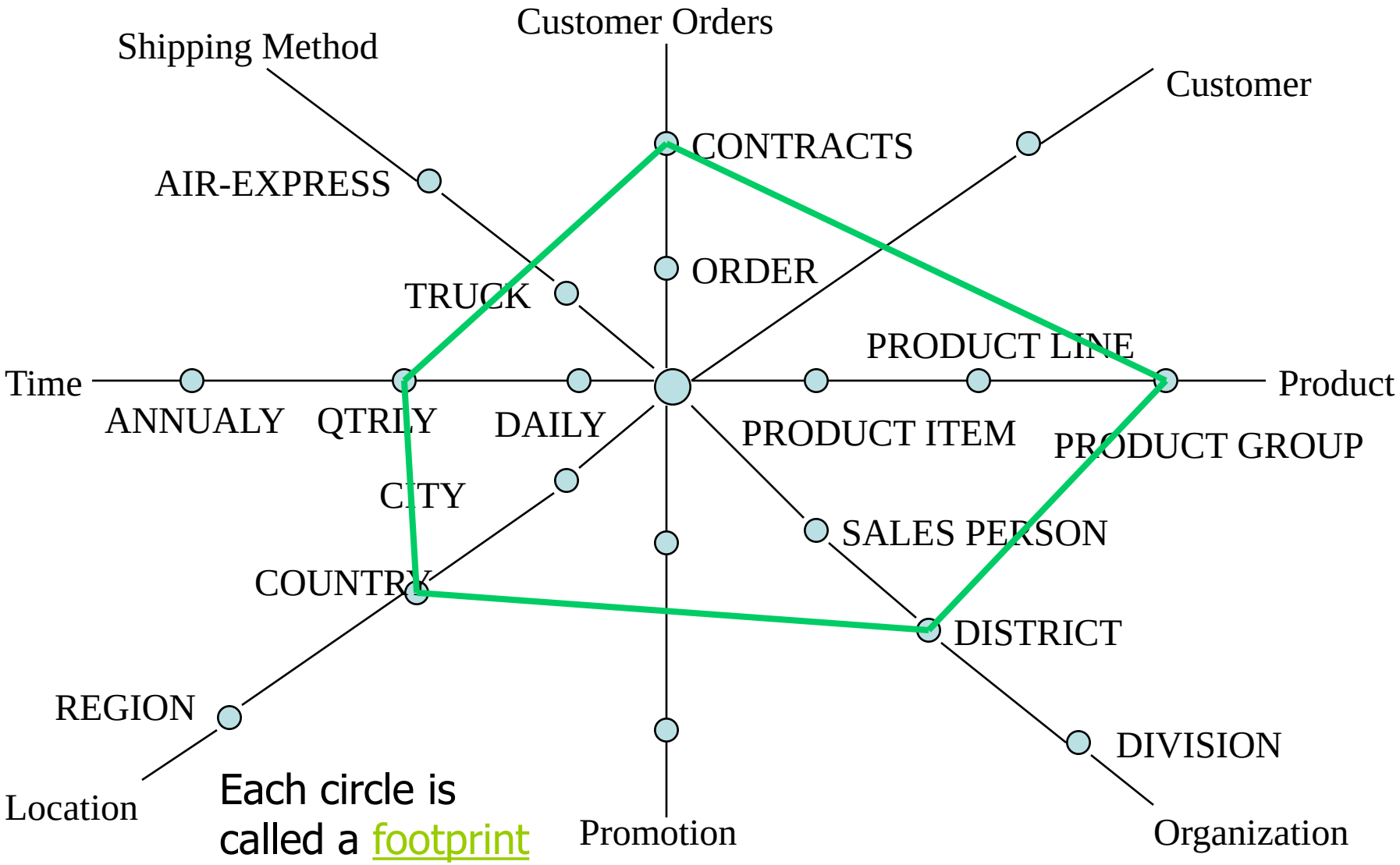


Fig. 3.10 Typical OLAP Operations

A Star-Net Query Model

- A Star-Net model consists of radial lines emanating from a central point, where each line represents a concept hierarchy of a dimension.
- Each abstraction level or a circle in the hierarchy is called **footprint**.
- It represents the granularities available for use by OLAP operations such as drill-down and roll-up.

A Star-Net Query Model



Each circle is called a footprint

Data Warehousing and OLAP Technology: An Overview

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- **Data Warehouse Design and Usage**
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary

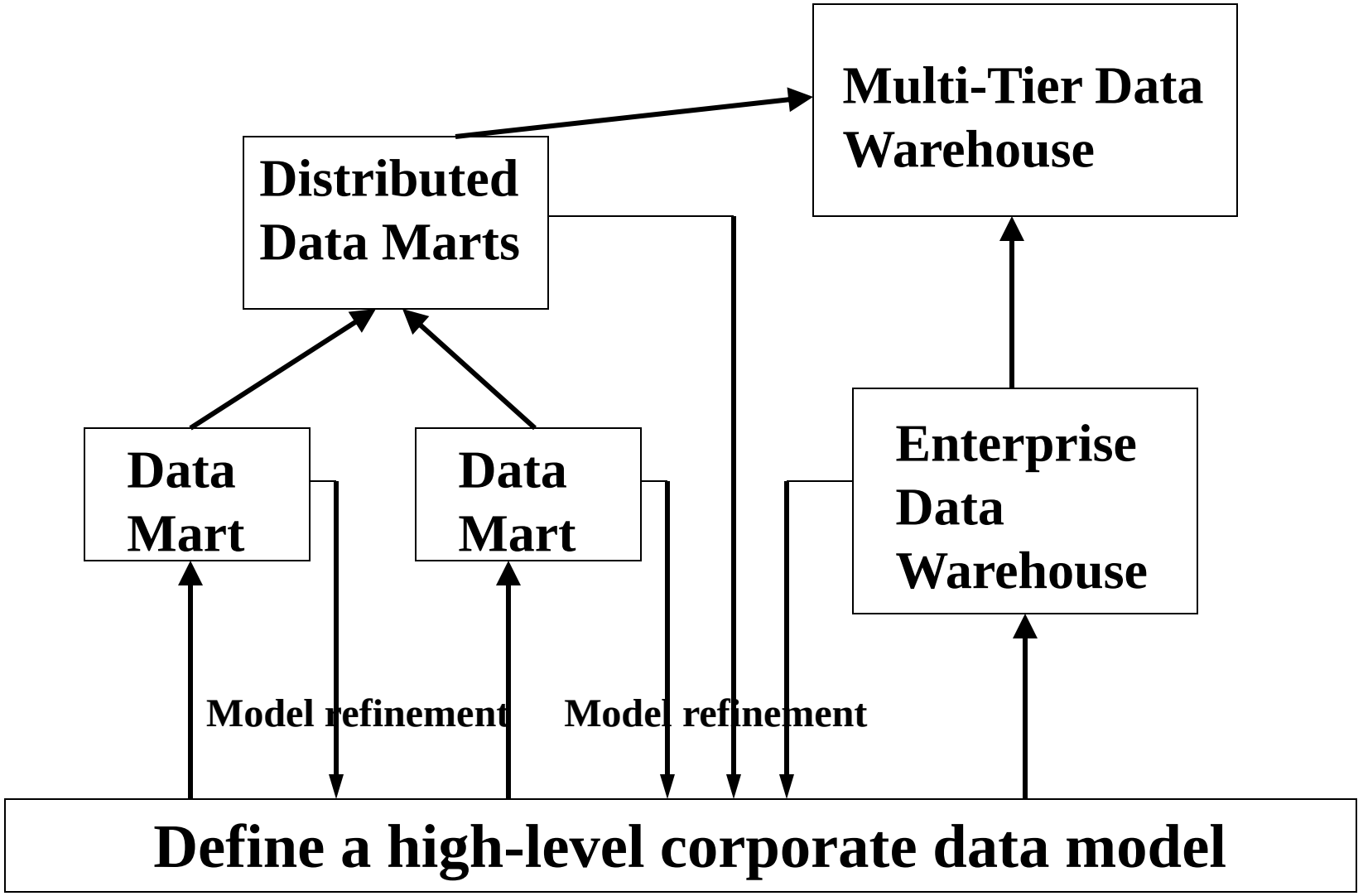
Design of Data Warehouse: A Business Analysis Framework

- Four views regarding the design of a data warehouse
  - Top-down view
    - allows selection of the relevant information necessary for the data warehouse
  - Data source view
    - exposes the information being captured, stored, and managed by operational systems
  - Data warehouse view
    - consists of fact tables and dimension tables
  - Business query view
    - sees the perspectives of data in the warehouse from the view of end-user

Data Warehouse Design Process

- **Top-down, bottom-up approaches or a combination** of both
  - Top-down: Starts with overall design and planning (mature)
  - Bottom-up: Starts with experiments and prototypes (rapid)
- **From software engineering point of view**
  - Waterfall: structured and systematic analysis at each step before proceeding to the next
  - Spiral: rapid generation of increasingly functional systems, short turn around time, quick turn around
- **Typical data warehouse design process**
  - Choose a business process to model, e.g., orders, invoices, etc.
  - Choose the *grain* (*atomic level of data*) of the business process
  - Choose the dimensions that will apply to each fact table record
  - Choose the measure that will populate each fact table record

Data Warehouse Development: A Recommended Approach



Data Warehouse Usage

- Three kinds of data warehouse applications
  - Information processing
    - supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts and graphs
  - Analytical processing
    - multidimensional analysis of data warehouse data
    - supports basic OLAP operations, slice-dice, drilling, pivoting
  - Data mining
    - knowledge discovery from hidden patterns
    - supports associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools

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

- Why online analytical mining?
  - High quality of data in data warehouses
    - DW contains integrated, consistent, cleaned data
  - Available information processing structure surrounding data warehouses
    - ODBC (Open Database Connectivity), OLEDB (**Object Linking and Embedding, Database)**, Web accessing, service facilities, reporting and OLAP tools
  - OLAP-based exploratory data analysis
    - Mining with drilling, dicing, pivoting, etc.
  - On-line selection of data mining functions
    - Integration and swapping of multiple mining functions, algorithms, and tasks

Data Warehousing and OLAP Technology: An Overview

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- **Data Warehouse Implementation**
- Data Generalization by Attribute-Oriented Induction
- Summary

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

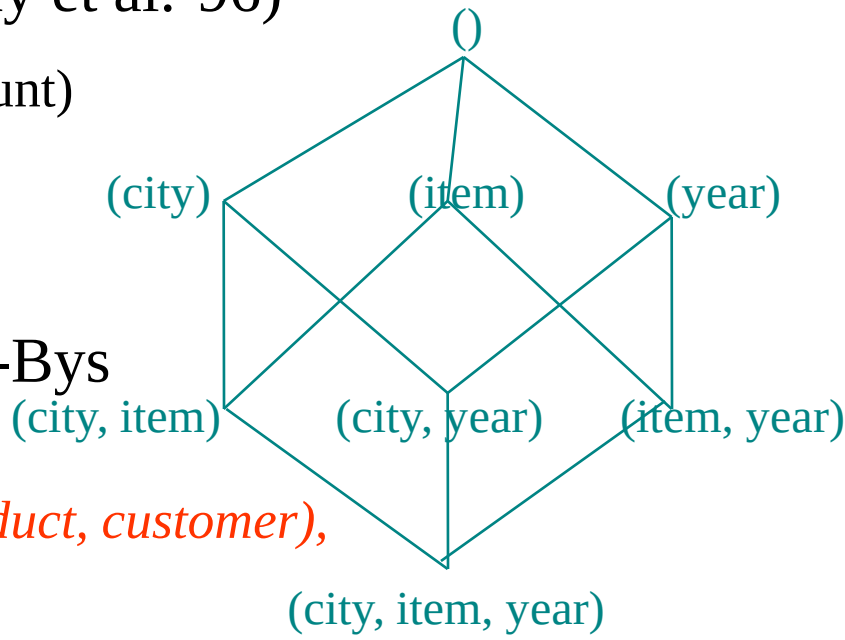
- Need compute the following Group-Bys

  (date, product, customer),

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

  (date), (product), (customer)

  ()



Bitmap indexes

- Assume that records have permanent numbers
- A bit-map index is a collection of bit vectors of length n, one for each value may appear in the field F.
- The vector for value v has 1 in position i if the i'th record has v in field F, and it has 0 there if not.
- Example for F and G fields:
  - (30, foo), (30,bar), (40, baz), (50, foo), (40, bar), (30, baz)
  - Bit index for F, each of 6 bits. For 30, it is 110001, for 40, it is 001010, and for 50, it is 000100.
  - Bit index for G also have three vectors. For "foo" it is, 100100, for "bar" it is 010010, and for "baz" it is 001001.

Bit map indexes: Partial match

- Bit maps allow answering of partial match queries quickly and efficiently.
- Example:
  - Movie(title, year, length, studioName)
  - SELECT title
  - FROM Movie
  - WHERE studioName= ‘Disney’ and Year=1965;
- If we have bitmap for studioName and year, then intersection or AND operation will give the result.
- Bit vectors do not occupy much space.

Bitmap indexes: range queries

- Example: consider the gold jewelry data of twelve points
  - 1 (25,60), 2(45,60), 3(50,75), 4(50,100), 5(50,120), 6(70, 110), 7(85,140), 8(30,260), 9(25,400), 10(45,350), 11(50,275), 12(60,260)
- Age has seven different values
  - 25(100000001000), 30(000000010000), 45(010000000100), 50(001110000010), 60(000000000001), 70(000001000000)
  - 85(000000100000)
- Salary has 10 different values
  - 60(000000000000), 75(001000000000), 100(000100000000)
  - 110(000001000000), 120(000010000000), 140(000000100000)
  - 260(000000010001), 275(000000000010), 350(000000000100)
  - 400(000000001000),

Example continued

- Find the jewelry buyers with an age range 45-55 and salary in the range 100-200
- Find the bit vectors of for the age values in the range and take OR
  - 010000000100 (for 45) and 001110000010 (for 50)
  - Result: 011110000110
- Find the bit vectors of salaries between 100 and 200 thousand.
  - There are four: 100,110,120, and 140, their bitwaise OR is 000111100000
- Take AND of both bit vectors
  - 000110000000
  - Find two records (50,100) and (50,120) are in the range.

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
  - A bit compression technique, Word-Aligned Hybrid (WAH), makes it work for high cardinality domain as well [Wu, et al. TODS'06]

Base table

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

Index on Region

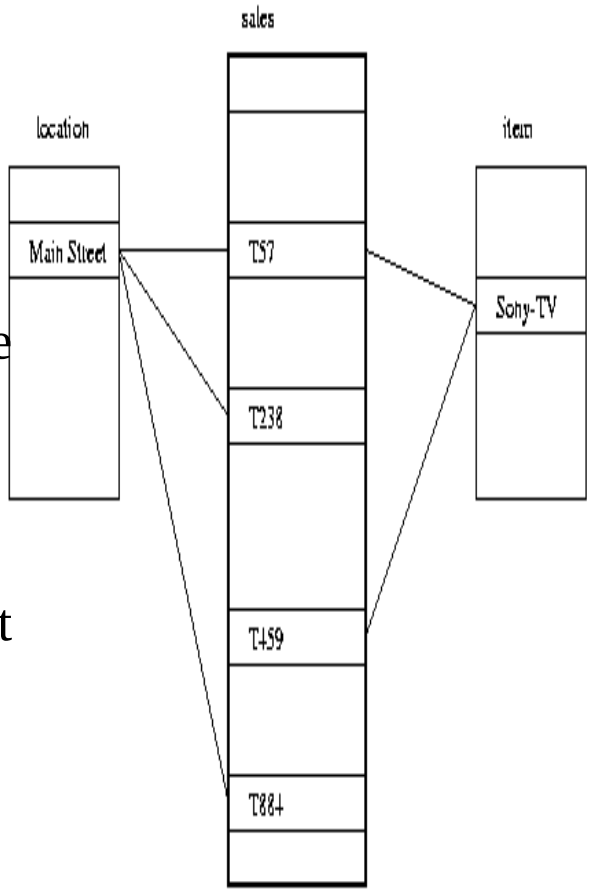
RecID	Asia	Europe	America
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1
5	0	1	0

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
- 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



Example of the join index

Fact: **store_sales**

Row id	...	ss_item_sk
1		1
2		1
3		2
4		5
5		3
6		1
7		2
8		4
9		6
10		5

Dimension: **item**

Row id	i_item_sk	i_item_name	...	i_item_color
1	1	TV		Black
2	2	Car		Yellow
3	3	IPhone		Black
4	4	Laptop		Green
5	5	Shirt		Black
6	6	Lamp		Brown

Index over store_sales.ss_item_sk:

Value	Row ids
1	1,2,6
2	3,7
3	5
4	8
5	4,10
6	9

Join index over item.i_item_color join key:
(store_sale.ss_item_sk=item.i_item_sk)

Value	Item sk	Row ids
Black	1,3,5 →	1,2,4,5,6,10
Yellow	2 →	3,7
Green	4 →	8
Brown	6 →	9

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 which materialized cuboid(s)** should be selected for OLAP op.
  - Let the query to be processed be on {*brand, province_or_state*} with the condition "*year = 2004*", and there are 4 materialized cuboids available:

    1) {*year, item_name, city*}

    2) {*year, brand, country*}

    3) {*year, brand, province_or_state*}

    4) {*item_name, province_or_state*} where *year = 2004*

    Which should be selected to process the query?
- Explore indexing structures and compressed vs. dense array structs in MOLAP

OLAP Server Architectures

- Relational OLAP (ROLAP)
  - Use relational or extended-relational DBMS to store and manage warehouse data and OLAP middle ware
  - Include optimization of DBMS backend, implementation of aggregation navigation logic, and additional tools and services
  - Greater scalability
- Multidimensional OLAP (MOLAP)
  - Sparse array-based multidimensional storage engine
  - Fast indexing to pre-computed summarized data
- Hybrid OLAP (HOLAP) (e.g., Microsoft SQLServer)
  - Flexibility, e.g., low level: relational, high-level: array
- Specialized SQL servers (e.g., Redbricks)
  - Specialized support for SQL queries over star/snowflake schemas

Data Warehousing and OLAP Technology: An Overview

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- **Data Generalization by Attribute-Oriented Induction**
- Summary

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 attribute removal or attribute generalization
  - Apply aggregation by merging identical, generalized tuples and accumulating their respective counts
  - Interaction with users for knowledge presentation

Attribute-Oriented Induction: An Example

Example: Describe general characteristics of graduate students in the University database

- Step 1. Fetch relevant set of data using an SQL statement, e.g.,

  Select * (i.e., name, gender, major, birth_place, birth_date, residence, phone#, gpa)

  from student

  where student_status in {“Msc”, “MBA”, “PhD” }

- Step 2. Perform attribute-oriented induction
- Step 3. Present results in generalized relation, cross-tab, or rule forms

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

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.

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.,

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

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

Concept Description vs. Cube-Based OLAP

- **Similarity**:
  - Data generalization
  - Presentation of data summarization at multiple levels of abstraction
  - Interactive drilling, pivoting, slicing and dicing
- **Differences**:
  - OLAP has systematic preprocessing, query independent, and can drill down to rather low level
  - AOI has automated desired level allocation, and may perform dimension relevance analysis/ranking when there are many relevant dimensions
  - AOI works on the data which are not in relational forms

Data Warehousing and OLAP Technology: An Overview

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- **Summary**

Summary

- Data warehousing: A multi-dimensional model of a data warehouse
  - A data cube consists of *dimensions* & *measures*
  - Star schema, snowflake schema, fact constellations
  - OLAP operations: drilling, rolling, slicing, dicing and pivoting
- Data Warehouse Architecture, Design, and Usage
  - Multi-tiered architecture
  - Business analysis design framework
  - Information processing, analytical processing, data mining, OLAM (Online Analytical Mining)
- Implementation: Efficient computation of data cubes
  - Partial vs. full vs. no materialization
  - Indexing OALP data: Bitmap index and join index
  - OLAP query processing
  - OLAP servers: ROLAP, MOLAP, HOLAP
- Data generalization: Attribute-oriented induction

References

- Refer to the Bibliographic notes of Chapter 3

- **Book: Jiawei Han, Jian Pei, Hanghang Tong, Data Mining: Concepts and Techniques, Fourth edition, 2022, Elseiver Inc.**

Surplus Slides

Compression of Bitmap Indices

- Bitmap indexes must be compressed to reduce I/O costs and minimize CPU usage—majority of the bits are 0's
- Two compression schemes:
  - Byte-aligned Bitmap Code (BBC)
  - Word-Aligned Hybrid (WAH) code
- Time and space required to operate on compressed bitmap is proportional to the total size of the bitmap
- Optimal on attributes of low cardinality as well as those of high cardinality.
- WAH out performs BBC by about a factor of two

Chapter 4: Data Cube Technology

P. Krishna Reddy, IIIT Hyderabad

Detailed Syllabus

- Introduction (1.5 hour): Definition, KDD framework, Issues in data mining.
- Data summarization (7.5 hrs): Data Types, Preprocessing, Characterization, Discrimination, data warehousing techniques (Data warehousing technology, **Data cube computation)**
- Concepts and algorithms for mining patterns and associations (9 hours) (Frequent item-set generation, A priori and FP-growth algorithm, Evaluation of Association patterns) and preprocessing
- Concepts and algorithms related to classification and regression (9hrs) (Overview, Decision tree induction, Over-fitting and under-fitting, Scalable decision tree algorithms, Bayesian Classification, Regression-based Prediction methods (9 hours)
- Concepts and algorithms for clustering the data (9 hours) (Overview, Types of Data, K-means, Aglomerative clustering, Clustering algorithms (DBSCAN, BIRCH, CURE, ROCK, CHAMELEON)).
- Outlier analysis and future trends (graph mining, spatio-temporal mining). (3 hours)

About OLAP and OLAM

- OLAP: Online Analytical Processing (OLAP)
  - Interactive analysis of multidimensional data at varied granularity levels
  - OLAP tools employ data cube and multidimensional data model to provide flexible access to summarized data
    - Drill-down (to see more specialized data) or roll-up (to see total sales)
- OLAM: exploratory multidimensional data mining or online analytical data mining
  - Searches for interesting patterns by exploring the data in multidimensional space.
  - Users can focus on a subset of dimensions by varying abstraction levels to find classification models, clusters, predictive rules and outliers.

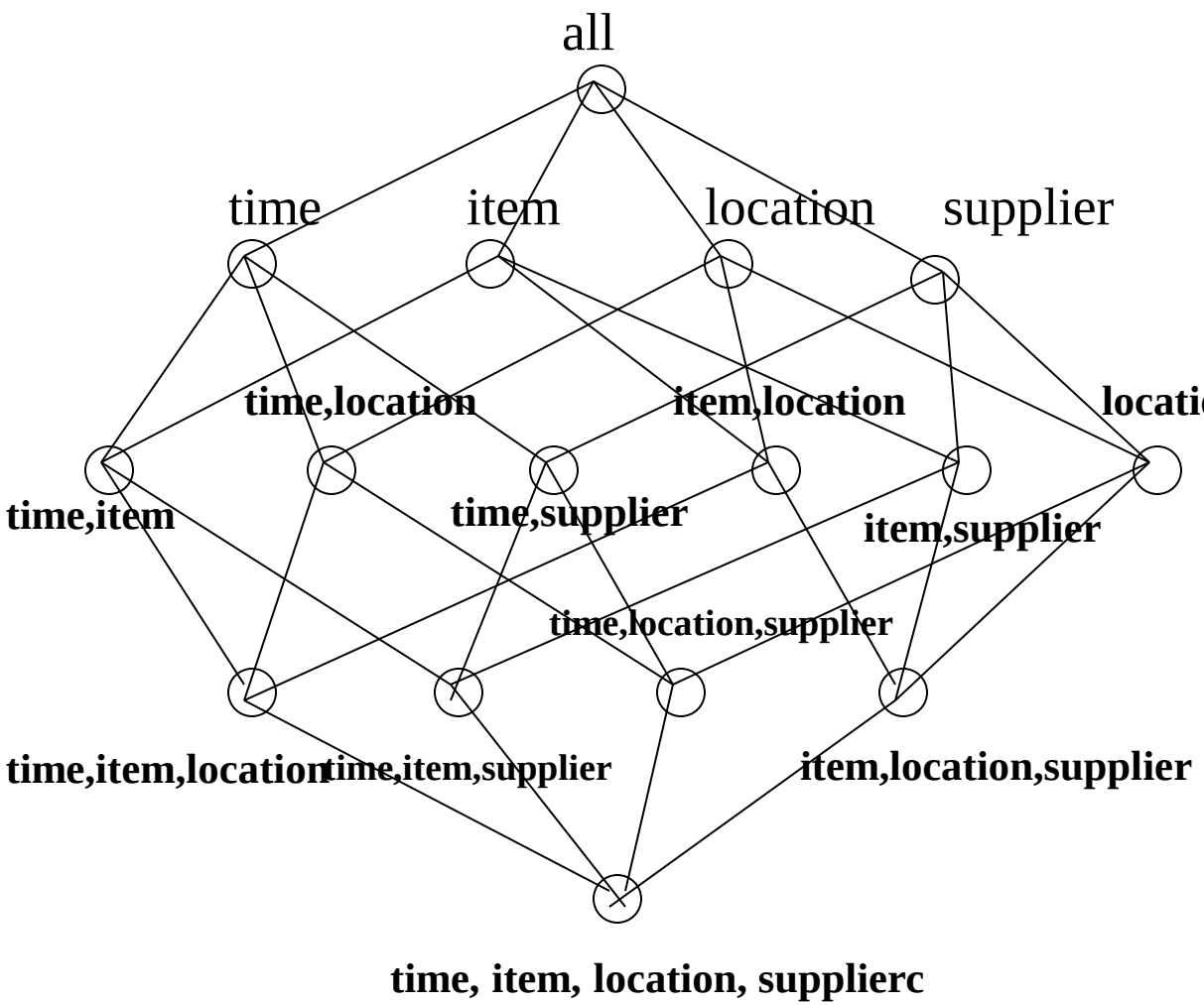
Outline

- **Data Cube Computation: Preliminary Concepts**
- Computing full/iceberg cubes: four methodologies
  - **1.**General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - **3**. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4**.** Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
    - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
    - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - 7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

Cube Materialization: Full Cube, Iceberg Cube, Closed Cube and Shell Cube

- Base Cell: A cell in the base cuboid.
- Aggregate cell: A cell from non-based cuboid is an aggregate.
- An aggregate cell aggregates over one or more dimensions.
  - Example: Let a=(a1,a2,...,an, measure) be a cell from one of the cuboids making-up a data cube. We say, "a" is a multi dimensional cell (from m-dimensional cuboid) if m (m<=n) values among {a1,a2,...,an} are not "*"
  - If m=n, then "a" is a base cell, otherwise it is an aggregate cell.
- Ancestor/descendant
  - 1-D cell is an ancestor of 2-D cell. 2-D cell is an ancestor of 3-D cell.

Data Cube: A Lattice of Cuboids



0-D(apex) cuboid

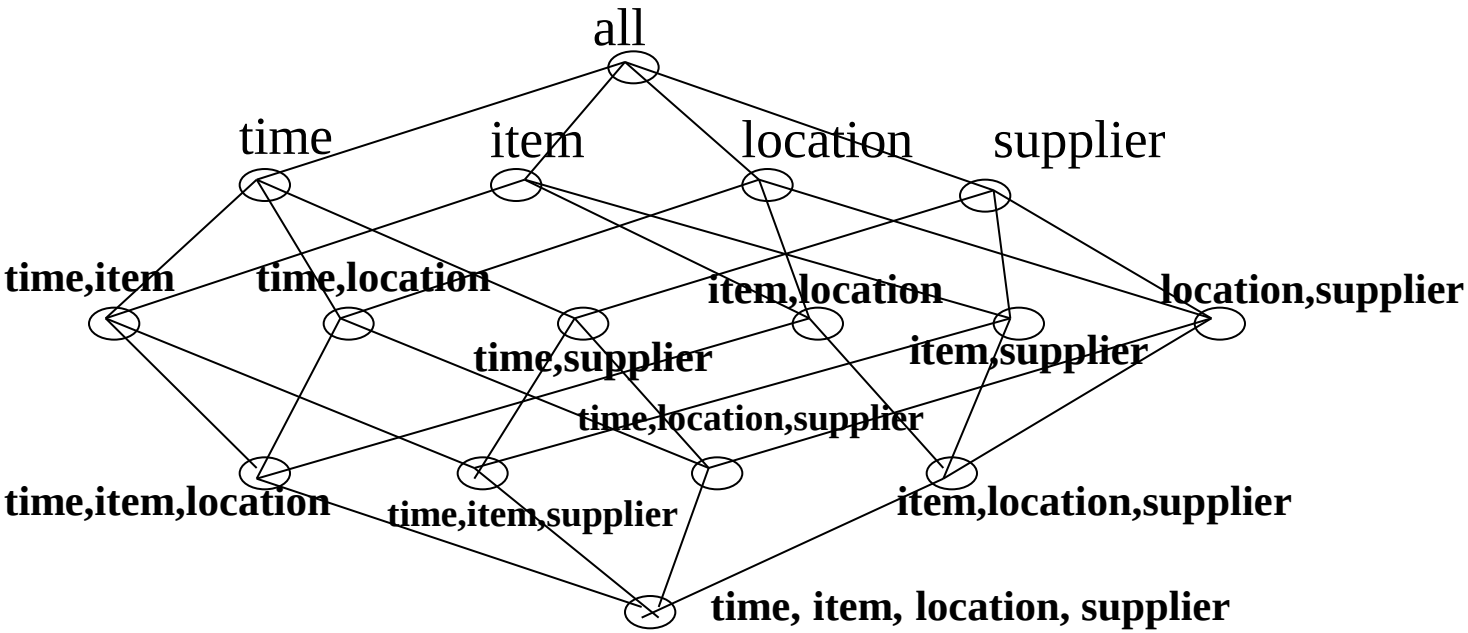
1-D cuboids

2-D cuboids

3-D cuboids

4-D(base) cuboid

Data Cube: A Lattice of Cuboids



0-D(apex) cuboid

1-D cuboids

2-D cuboids

3-D cuboids

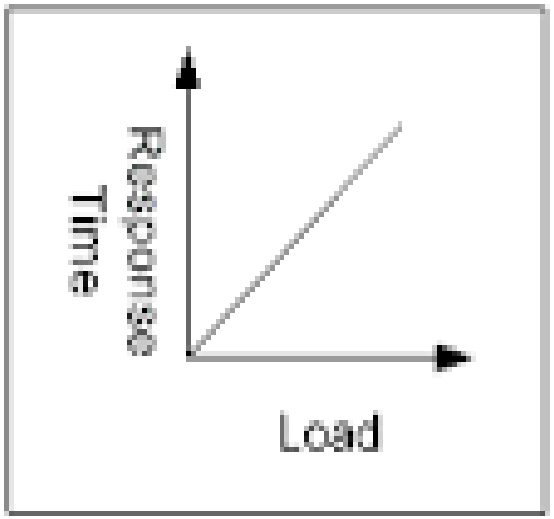
4-D(base) cuboid

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

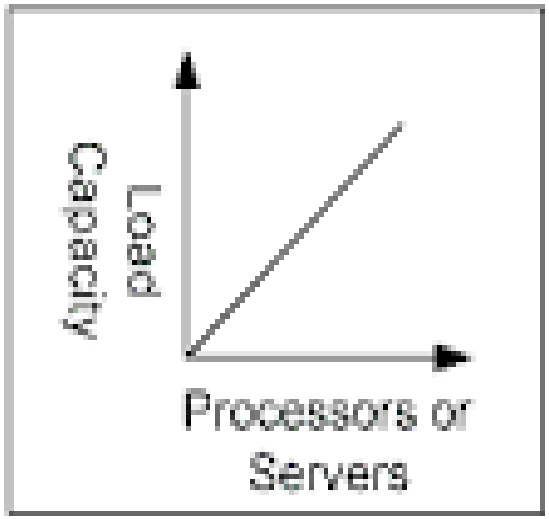
About the computation of data cube

- **Full cube**: all the cells of all cuboids for a given data cube
- Computation of full cuboid is of exponential complexity.
  - A data cube of n dimensions contains 2^n cuboids
  - More cuboids, if we consider concept hierarchy
  - The size of each cuboid depends on the cardinality of the dimension.
  - Precomputation of each cube requires huge and excessive amount of memory.
- It is important to explore the scalable methods.
  - memory consumed
  - Size of the computed data
  - time required for computation

About scalability



Linear Scalability Relative to Load



Linear Scalability Relative to System Resources (e.g., hardware)

Issue: Developing scalable algorithms

About Iceberg cubes

- Several cubes may not be interesting
  - For may cuboids, the aggregate might be zero or of less value.
- Sparse cuboid: the product of cardinalities of dimensions is large relative to the number of non-zero valued tuples.
- Sparse cube: if the cube contains many sparse cuboids.
- So, it is better to materialize only those cells above some minimum threshold.
- Such partially materialized cubes are known as iceberg cubes.
- By materializing only few cells in the data cube, the result is seen as a tip of iceberg.

Cube Materialization: Full Cube vs. Iceberg Cube

- Full cube vs. iceberg cube

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

iceberg condition

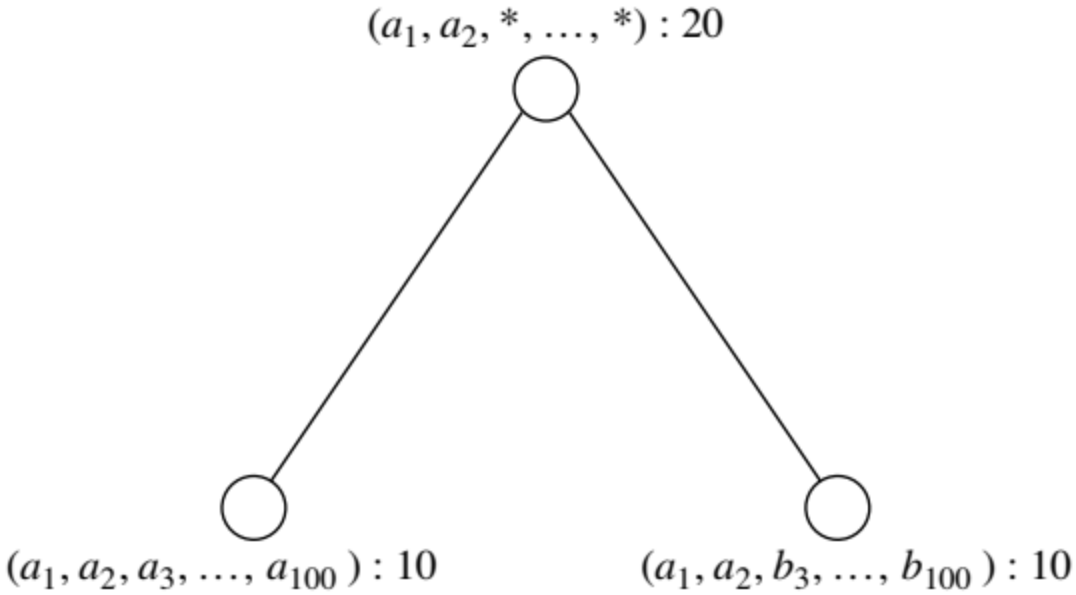
- Computing *only* the cuboid cells whose measure satisfies the iceberg condition
- Only a small portion of cells may be "above the water'' in a sparse cube
- Avoid explosive growth: A cube with 100 dimensions
  - 2 base cells: (a1, a2, …., a100), (b1, b2, …, b100)
  - How many aggregate cells if "having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

Iceberg Cube, Closed Cube & Cube Shell

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

For (A_1, A_2, … A_{10}), how many combinations to compute?

Three closed cells forming the lattice of a closed cube.



Example problem

6. When computing a cube of high dimensionality, we encounter the inherent *curse of dimensionality* problem: there exists a huge number of subsets of combinations of dimensions.

   (a) Suppose that there are only two base cells, $\{(a_1, a_2, a_3, \ldots, a_{100}), (a_1, a_2, b_3, \ldots, b_{100})\}$, in a 100-dimensional base cuboid. Compute the number of nonempty aggregate cells. Comment on the storage space and time required to compute these cells.

   (b) Suppose we are to compute an iceberg cube from the above. If the minimum support count in the iceberg condition is two, how many aggregate cells will there be in the iceberg cube? Show the cells.

   (c) Introducing iceberg cubes will lessen the burden of computing trivial aggregate cells in a data cube. However, even with iceberg cubes, we could still end up having to compute a large number of trivial uninteresting cells (i.e., with small counts). Suppose that a database has 20 tuples that map to (or cover) the two following base cells in a 100-dimensional base cuboid, each with a cell count of 10: $\{(a_1, a_2, a_3, \ldots, a_{100}) : 10, (a_1, a_2, b_3, \ldots, b_{100}) : 10\}$.

      i. Let the minimum support be 10. How many distinct aggregate cells will there be like the following: $\{(a_1, a_2, a_3, a_4, \ldots, a_{99}, *) : 10, \ldots, (a_1, a_2, *, a_4, \ldots, a_{99}, a_{100}) : 10, \ldots, (a_1, a_2, a_3, *, \ldots, *, *) : 10\}$?

      ii. If we ignore all the aggregate cells that can be obtained by replacing some constants with $*$'s while keeping the same measure value, how many distinct cells are left? What are the cells?

Answer:

(a) Suppose that there are only two base cells, $\{(a_1, a_2, a_3, \ldots, a_{100}), (a_1, a_2, b_3, \ldots, b_{100})\}$, in a 100-dimensional base cuboid. Compute the number of nonempty aggregate cells. Comment on the storage space and time required to compute these cells.

Each base cell generates $2^{100}-1$ aggregate cells. (We subtract 1 because, for example, $(a_1, a_2, a_3, \ldots, a_{100})$ is not an aggregate cell.) Thus, the two base cells generate $2 \times (2^{100}-1) = 2^{101}-2$ aggregate cells, however, four of these cells are counted twice. These four cells are: $(a_1, a_2, *, \ldots, *)$, $(a_1, *, \ldots, *)$, $(*, a_2, *, \ldots, *)$, and $(*, *, \ldots, *)$. Therefore, the total number of cells generated is $2^{101} - 6$.

(b) Suppose we are to compute an iceberg cube from the above. If the minimum support count in the iceberg condition is two, how many aggregate cells will there be in the iceberg cube? Show the cells.

They are 4: $\{(a_1, a_2, *, \ldots, *), (a_1, *, *, \ldots, *), (*, a_2, *, \ldots, *), (*, *, *, \ldots, *)\}$.

(c) Introducing iceberg cubes will lessen the burden of computing trivial aggregate cells in a data cube. However, even with iceberg cubes, we could still end up having to compute a large number of trivial uninteresting cells (i.e., with small counts). Suppose that a database has 20 tuples that map to (or cover) the two following base cells in a 100-dimensional base cuboid, each with a cell count of 10: $\{(a_1, a_2, a_3, \ldots, a_{100}) : 10, (a_1, a_2, b_3, \ldots, b_{100}) : 10\}$.

i. Let the minimum support be 10. How many distinct aggregate cells will there be like the following: $\{(a_1, a_2, a_3, a_4, \ldots, a_{99}, *) : 10, \ldots, (a_1, a_2, *, a_4, \ldots, a_{99}, a_{100}) : 10, \ldots, (a_1, a_2, a_3, *, \ldots, *, *) : 10\}$?
There will be $2^{101} - 6$, as shown above.

ii. If we ignore all the aggregate cells that can be obtained by replacing some constants by $*$'s while keeping the same measure value, how many distinct cells are left? What are the cells?
There are only three distinct cells left: $\{(a_1, a_2, a_3, \ldots, a_{100}) : 10, (a_1, a_2, b_3, \ldots, b_{100}) : 10, (a_1, a_2, *, \ldots, *) : 20\}$.

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - **1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)**
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - **3**. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4**.** Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - 7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

On the Computation of Multidimensional Aggregates

Sameet Agarwal Rakesh Agrawal Prasad M. Deshpande Ashish Gupta
Jeffrey F. Naughton Raghu Ramakrishnan Sunita Sarawagi

Proceedings of the 22nd VLDB Conference
Mumbai(Bombay), India, 1996

Optimizations Possible

- **Smallest-parent:** Computing a group-by from the smallest previously computed group-by. For instance, AB can be computed from ABC, ABD or ABCD. ABC or ABD are clearly better choices for computing AB.
- **Cache-results:** Caching (in memory) the results of a group-by from which other group-bys are computed to reduce disk I/O. For instance, for the cube in Figure 1, having computed ABC, we compute AB
- **Amortize-scans:** This optimization aims at amortizing disk reads by computing as many group-bys as possible, together in memory. For instance, if the group-by ABCD is stored on disk, we could reduce disk read costs if all of ABC, ACD, ABD and BCD were computed in one scan of ABCD.
- **Share-sorts:** This optimization is specific to the sort-based algorithms and aims at sharing sorting cost across multiple group-bys.
- **Share-partitions:** When the hash table is too large to fit in memory, data is partitioned and aggregation is done for each partition that fits in memory. We can save on partitioning cost by sharing this cost across multiple groupbys

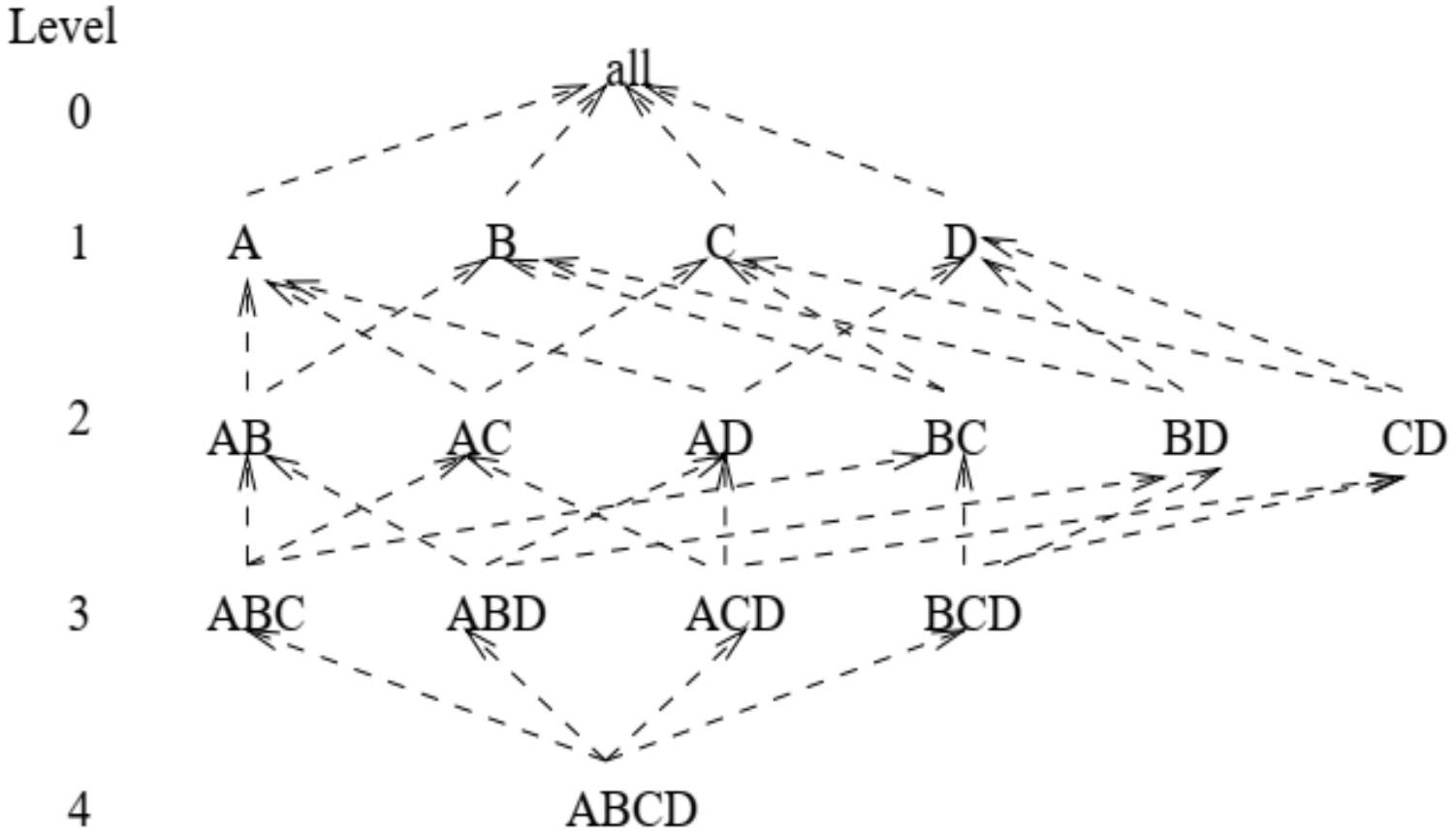


Figure 1: A search lattice for the cube operator

Algorithm PipeSort

- Employs the optimizations **cache-results and amortize-scans** to reduce disk scan cost by executing multiple group-bys in a pipelined fashion.
- Example
  - For instance, consider the example of using data sorted in the order ABCD to compute prefixes ABCD, ABC, AB and A.
  - Instead of computing each of these group-bys separately, we can compute them in a pipelined fashion.
    - Sort the raw data in the attribute order ABCD, we scan the sorted data to compute group-by ABCD.
      - Every time a tuple of ABCD is computed, it is propagated up the pipeline to compute ABC; every time a tuple of ABC is computed, it is propagated up to compute AB, and so on.
    - Thus, each pipeline is a list of group-bys all of which are computed in a single scan of the sort input stream. During the course of execution of a pipeline we need to keep only one tuple per group-by in the pipeline in memory.

Algorithm

- For each group-by we have estimate of the number of distinct values.
- Input is search lattice (Figure 1):
- Output: subgraph of the search lattice where each group-by is connected to a single parent from which it will be computed and is associated with attribute order in which it will be sorted.
- Generate plan incrementally.

(a) Transformed search lattice

(b) Minimum cost matching

Figure 2: Computing level 1 group-bys from level 2 group-bys in a 3 attribute cube

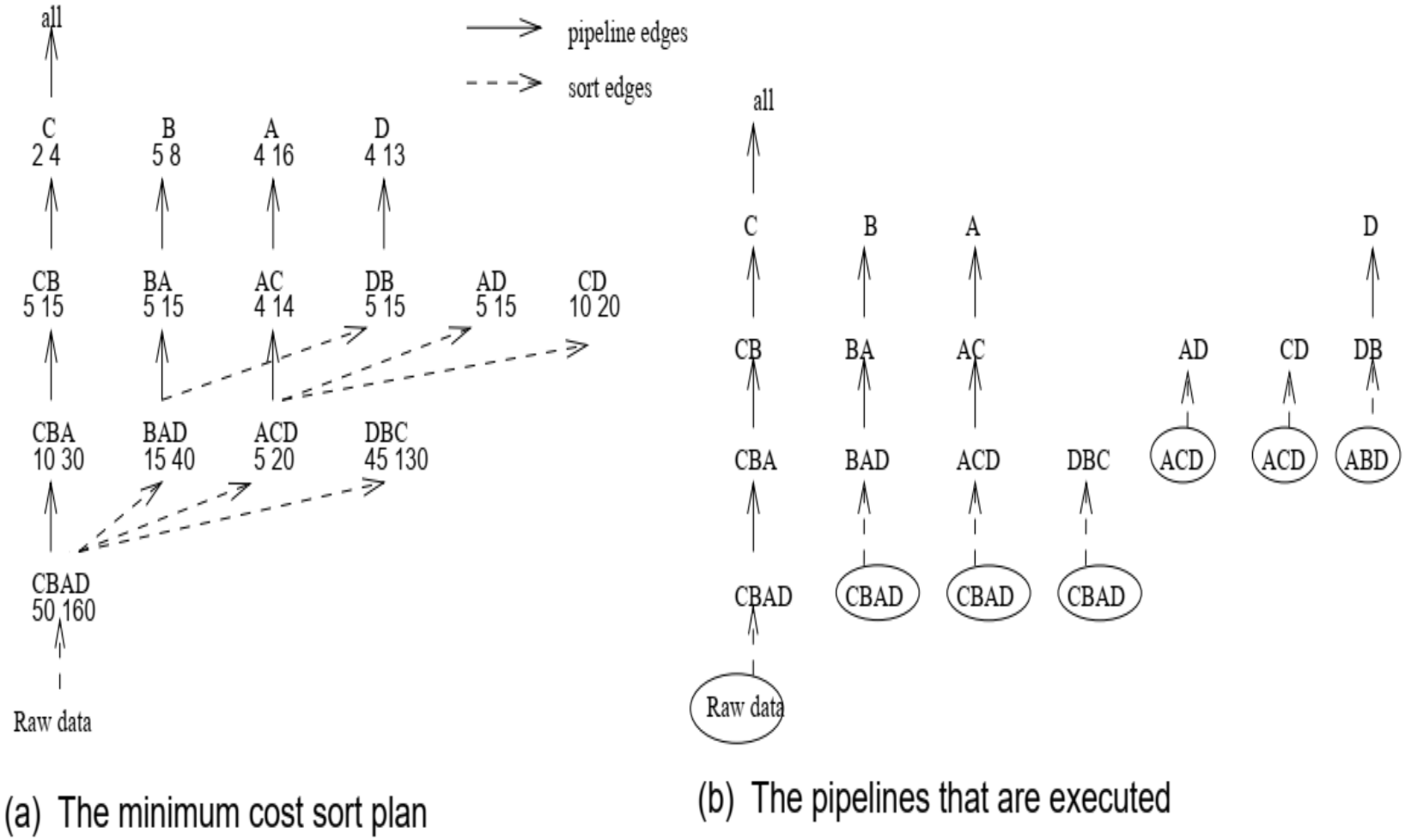


Figure 3: Sort-based method for computing a four attribute cube

Algorithm PipeHash

- The input is the search lattice.
- **Careful memory allocation**
  - **Optimizations: cache results and amortize scans**
- The PipeHash algorithm chooses for each group-by, the parent group-by with the smallest estimated total size.
- A minimum spanning tree (MST) is formed
  - where each vertex is a group-by and an edge from group-by a to b shows that a is the smallest parent of b.
- Approach: Divide the MST into smaller subtrees each of which can be computed in one scan of the group-by at the root of the MST such that the cost of scanning (from disk) the root group-by is minimized.
  - The problem is NP-complete
  - Heuristic: Optimizations cache-results and amortize-scans are favored by choosing as large a subtree of the MST as possible so that we can use the method above to compute together the group-bys in the subtree.

In the figure, we show the MST for a four attribute search lattice (the size of each group-by is indicated below the group-by).

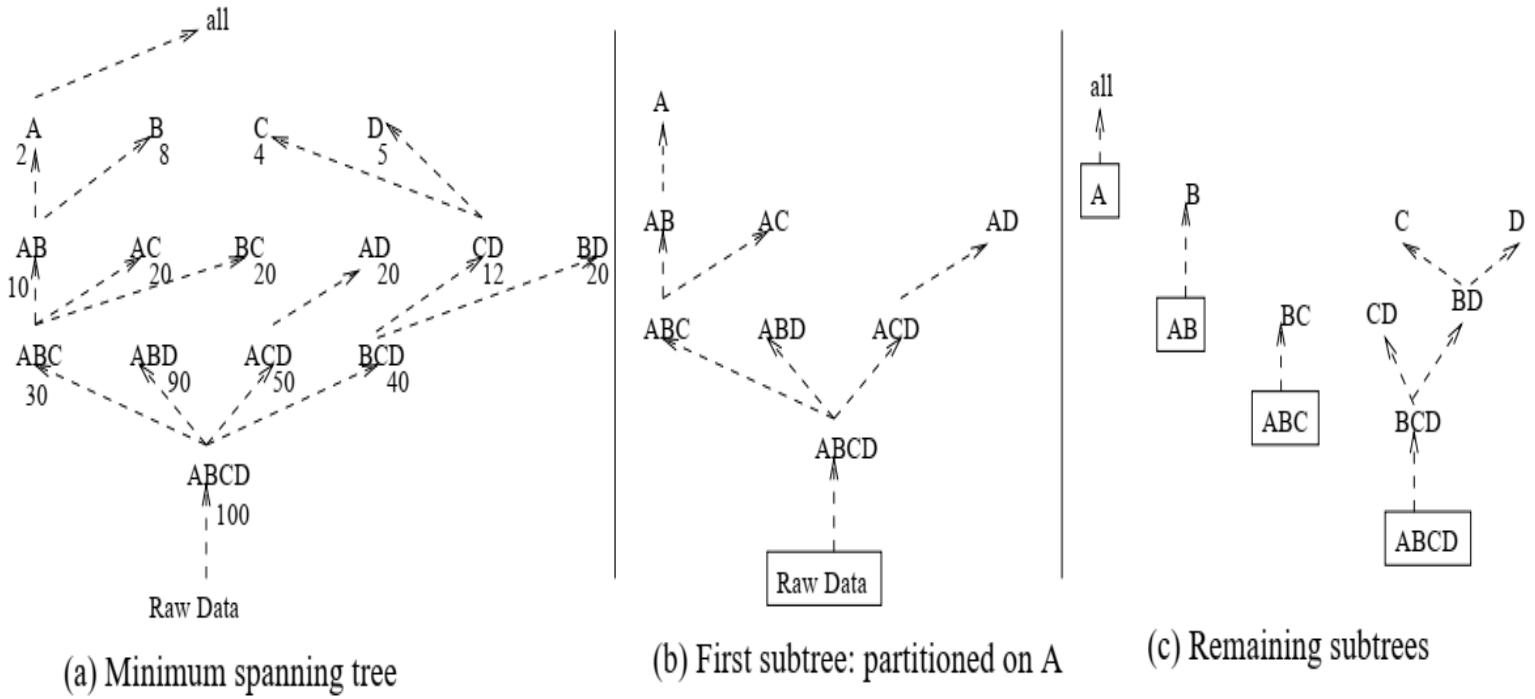


Figure 4: PipeHash on a four attribute group-by

Hierarchy of Group-By Queries (Parent Method)

- Consider the computation of the CUBE with {A; B ; C; D}.
  - The cuboid {A; C} can be computed from the cuboid {A; B ; C} or the cuboid {A; C; D}
  - In general, a cuboid on attribute set X can be computed from a cuboid on attribute set {Y} iff X subset of Y.
- Optimization
  - Choose Y to be as small as possible.
  - Heuristic: computing a cuboid with k-1 attributes from a cuboid with k attributes
  - Example: it is better to compute sum of sales by (product) using sum of sales by (product, customer) rather than sum of sales by (product, year, customer).
- We can view this hierarchy as a DAG, where the nodes are cuboids and there is an edge from a k attribute cuboid to a k-1 attribute cuboid iff the k-1 attribute set is a subset of the k attribute set.
- The DAG captures the consider-computing-from" relationship between the cuboids. The DAG for the CUBE on {A; B ; C; D} is shown in Figure 7.

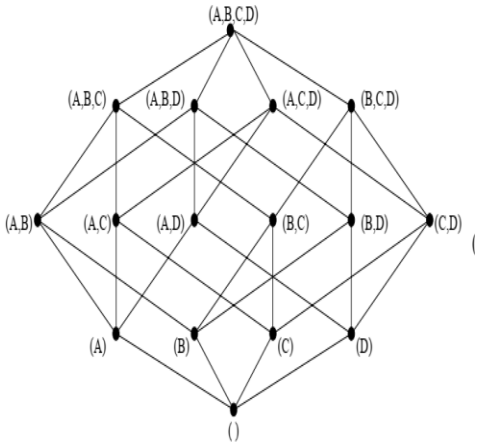


Figure 7: Sort orders enforced on the cuboids

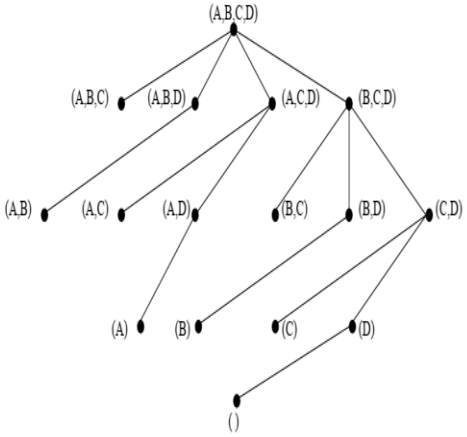


Figure 8: Cuboid Tree obtained from the cuboid DAG

Overlap Method

- As in the Parent method, the Overlap method computes each cuboid from one of its parents in the cuboid tree.
- Main Concept**: partially matching sort orders.**
- It tries to do better than the Parent method by overlapping the computation of different cuboids and using partially matching sort orders. This can significantly reduce the number of I/Os required.
- Sorting combined with Overlap seems to be a good option due to the following observations which help in reducing the number of sorting steps.
  - Cuboids can be computed from a sorted cuboid in sorted order.
  - An existing sort order on a cuboid can be used while computing other cuboids from it. For example, consider a cuboid X = {A; B ; D} to be computed from Y = {A; B ; C; D}.
  - Let Y be sorted in ABCD order which is not the same as ABD order needed to compute X. But Y need not be resorted to compute X. The existing order on Y can be used.

Main concept of overlap method: Sorted Runs

Sorted Runs :

Consider a cuboid on j attributes $\{A_1, A_2, \ldots, A_j\}$. We use $(A_1, A_2, \ldots, A_j)$ to denote the cuboid sorted on the attributes A_1, A_2, ..., A_j in that order. Consider the cuboid $S = (A_1, A_2, \ldots, A_{l-1}, A_{l+1}, \ldots, A_j)$ computed using $B = (A_1, A_2, \ldots, A_j)$. A *sorted run* R of S in B is defined as follows: $R = \Pi_{A_1, A_2, \ldots, A_{l-1}, A_{l+1}, \ldots, A_j}(Q)$ where Q is a **maximal** sequence of tuples τ of B such that for each tuple in Q, the first l columns have the same value. Informally a sorted run of S in B is a maximal run of tuples in B whose ordering is consistent with their ordering in the sort order associated with S.

For example, consider $B = [(a,1,2),(a,1,3),(a,2,2),(b,1,3),(b,3,2),(c,3,1)]$. Let S be the cuboid on the first and third attribute. i.e., $S = [(a,2),(a,3),(b,3),(b,2),(c,1)]$. The sorted runs for S are $[(a,2),(a,3)]$, $[(a,2)]$, $[(b,3)]$, $[(b,2)]$ and $[(c,1)]$.

Partitions :

B and S have a common prefix of $A_1, A_2, \ldots, A_{l-1}$. A *partition* of the cuboid S in B is a union of sorted runs such that the first $l-1$ columns (the common prefix) of all the tuples of the sorted runs have the same value. In the above example, the partitions for S in B will be $[(a,2),(a,3)]$, $[(b,2),(b,3)]$ and $[(c,1)]$.

This definition implies that all tuples of one partition are either less or greater than all tuples of any other partition. Tuples from different partitions will not merge for aggregation. Thus partition becomes a unit of computation and each partition can be computed independently of the others.

Outline of the Overlap Method

- The available memory may be insufficient for large CUBEs. We could try to reduce the amount of memory needed to compute a particular cuboid.
- Since partition can be a unit of computation, while computing a cuboid from another sorted cuboid, we just need memory sufficient to hold a partition of the cuboid.
- Basic idea
  - As soon as a partition is completed, the tuples can be pipelined into the computation of descendant cuboids, or written out to disk; the same memory can then be used to start computation of the next partition.
  - This is a significant reduction since for most cuboids the partition size is much less than the size of the cuboid.
- Example:
  - While computing (A; B ; C) and (A; B ; D) from (A; B ; C; D) the partition size for (A; B ; C) is 1 tuple (since (A; B ; C) sort order matches (A; B ; C; D) sort order) whereas the partition size for (A; B ; D) is bounded by the number of distinct values of D. So for computing these we just need space sufficient to hold a partition.
- So, with the overlap method, computation of many cuboids can be overlapped in the available memory effectively reducing the number of scans.

Summary

- Pipeline: Reduce scanning
- PipeHash: Create smallest size parent
- Parent method: Compute higher level cuboids from smallest cubiods
- Overlap: Create sorted orders

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - **2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)**
  - **3**. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4**.** Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - 7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

An Array-Based Algorithm for Simultaneous Multidimensional Aggregates *

Yihong Zhao
Computer Sciences Department
University of Wisconsin-Madison
zhao@cs.wisc.edu

Prasad M. Deshpande
Computer Sciences Department
University of Wisconsin-Madison
pmd@cs.wisc.edu

Jeffrey F. Naughton
Computer Sciences Department
University of Wisconsin-Madison
naughton@cs.wisc.edu

SIGMOD 97

Introduction

- Several efficient algorithms for Relational OnLine Analytical Processing (ROLAP) have been developed to compute the Cube.
- This paper: how to compute the Cube for Multidimensional OLAP (MOLAP) systems
- MOLAP systems present a different sort of challenge in computing the cube than do ROLAP systems.
- Reason: the data structures to store data are different

ROLAP systems

- ROLAP systems store in relational tables
  - A “cell” in a logically multidimensional space is represented as a tuple, with some attributes that identify the location of the tuple in the multidimensional space, and other attributes that contain the data value corresponding to that data cell.
  - Example:
    - A cell of the array might be represented by the tuple (shoes, WestTown, 3-July-96, $34. 00).
    - Computing the cube over such a table requires a generalization of standard relational aggregation operators.
- In prior work, three main ideas have been used to make ROLAP computation efficient:
  - Using some sort of grouping operation on the dimension attributes to bring together related tuples (e.g., sorting or hashing),
  - Using the grouping performed on behalf of one of the sub-aggregates as a partial grouping to speed the computation another sub-aggregate, and
  - To compute an aggregate from another aggregate, rather than from larger table.

MOLAP systems

- Store data in arrays
  - Example: For storing the tuple (shoes, WestToun, 3-JuIY-1996, $34.00), a MOLAP system would just store the data value $34. 00;
  - The position within the sparse array would encode the fact that this is a sales volume for shoes in the West Town store on July 3, 1996.
- For computing the cube on data stored in arrays, one can once again use the ROLAP trick of computing one aggregate from another.
- However, none of the other techniques of ROLAP cube can be applied.
  - No equivalent of "reordering to bring together related tuples" based upon dimension values.
  - There is no concept of using an order generated by one subaggregate in the computation of another;
- Should visit those values in the right order so that the computation is efficient.
- Basic idea
  - is to simultaneously compute spatially-delimited partial aggregates so that a cell does not have to be revisited for each sub-aggregate.
  - one must "chunk" them into small memory-sized pieces, and perform some sort of "compression" to avoid wasting space on cells that contain no valid data

- In this paper, we present a MOLAP algorithm incorporating of these ideas.
- The algorithm succeeds in overlapping the computation of multiple subaggregates, and makes good use of available main memory.
- It contains several mathematical proofs and theorems.

- One can always use our MOLAP algorithm in a relational system by the following three-step procedure:
  - Scan the table, and load it into an array.
  - Compute the cube on the resulting array
  - Dump the resulting cubed array into tables.
- Surprise:
  - that this three-step approach was actually faster than the direct approach of cubing the table.

Array Storage Issues

- There are three main issues to resolve.
- First,
  - it is highly likely in a multidimensional application that the array itself is far too large to fit in memory.
  - In this case, the array must be split up into “chunks”, each of which is small enough to fit comfortably in memory.
- Second,
  - even with this ‘chunking”, it is likely that many of the cells in the array are empty, meaning that there is no data for that combination of coordinates.
  - To efficiently store this sort of data we need to compress these chunks.
- Third,
  - in many cases an array may need to be loaded from data that is not in array format (e.g., from a relational table or from an external load file.)

1. Chunking Arrays

- Arrays should be stored in chunks, bot how?
- Standard programming language technique of storing the array in a row major or column major order is not very efficient.
  - Consider a row major representation of a two-dimensional array, with dimensions Store and Date, where Store forms the row and Date forms the column.
  - Accessing the array in the row order (order of ***Stores)*** is efficient with this representation, since each disk page that we read will contain several Stores. However, accessing in the order of columns ***(Date)*** is inefficient.
  - If the Store dimension is big, each disk page read will only contain data for one ***Date. T***o get data for the next ***Date*** will require another disk access;
    - In fact there will be one disk access for each ***Date*** required.
  - The simple row major layout creates an asymmetry among the dimensions, favouring one over the other. This is because data is accessed from disk in units of pages
- Solution: Chunking is a way to divide an n-dimensional array into small sizen-dimensional chunks and store each chunk as one object on disk. Each array chunk has `n' dimensions and will correspond to the blocking size on the disk.

Compressing Sparse Arrays

- For dense chunks,
  - Greater than 40% of the array cells have a valid value,
  - we do not compress the array, simply storing all cells of the array as is but assigning a null value to invalid array cells.
  - We will have uniform array chunks.
- Sparse chunks
  - Storing the data without compression is a waste of storage, as most of the cells do not contain valid data.
  - In this case we use what we call “chunk-offset compression”.
  - In chunk-offset compression, for each valid array entry, we store a pair, (offsetInChunk, data). The offsetlnChunk integer can be computed as follows: consider the chunk as a normal (uncompressed) array. Each cell c in the chunk is defined by a set of indices
  - For example, if we are working with a three-dimensional chunk, a given cell will have an “address” (i,j,k) in the chunk.

Loading Arrays from Table

- The table size and number of chunks are known.
- If the memory is less than the table size, create partitions of the main memory size.
- Next, read the partitions and create chunks consists of hash buckets.

A Basic Array Cubing Algorithm

- Use minimum memory, no overlap
- First consider how to compute a group-by from a simple non-chunked array.
  - Consider three dimensional array A, B, and C. and we want to compute the aggregate AB?
  - This can be seen as projecting onto the AB plane; logically, this can be done by sweeping a plane through the C dimension, aggregating as we go, until the whole array has been swept.
- Consider that ***ABC*** array is stored in a number of chunks.
  - Now instead of sweeping an entire plane of size |***A***|****B**|**.** where IA| and IBI are the sizes of the ***A*** and B dimensions,
    - we do it on a chunk by chunk basis.
  - With this, each chunk can be read by once
- This generalization of this algorithm to higher dimensions is straight-forward; instead of sweeping planes through arrays, in higher dimensions, say ***k*** dimensional arrays, one sweeps k-1 dimensional arrays.

Computing all aggregates of an array

- If the array has dimensions ABC, we need to compute AB, BC, AC, and A, B, C, as well as the overall total aggregate.
- Naive approach
  - compute all of these aggregates from the initial ABC array.
- It is far more efficient to compute A from AB than computing A from ABC.
  - This idea has been explored in the ROLAP cube computation literature.
  - the aggregates to be computed can be viewed as a lattice, with ***ABC as*** the root
  - *ABC* has children AB, BC, AC and AC had children A and C and so forth.
  - To compute the cube efficiently we embed a tree in this lattice, and compute each aggregate from its parent in this tree.
- For ROLAP cube computations this is a difficult question, since the sizes of the tables corresponding to the nodes in the lattice are not known until they are computed, so heuristics must be used.
- For our chunk-based array algorithm
  - We know about the storage requirement
  - The "minimum size spanning tree" can be denied for the lattice.
  - For each node n in the lattice, its parent in the minimum size spanning tree is the node n' which has the minimum size. The "n" can be computed.

Example

- The array ***ABC' is*** a 16 x 16 x 16 array with 4 x 4 x 4 array chunks laid out in the dimension order ABC.
- The chunks are numbered from 1 to 64.
- For example, to computing the BC group-by, we read in the chunk number order from 1 to 64, aggregate each four ABC chunks to a BC chunk, output the BC chunk to disk, and reuse the memory for the next BC chunk

- **So far we have not used the hierarchy of aggregates**
- **Th above basic algorithm will compute *AB* from *ABC*, then will re-scan *ABC* to compute *AC*, then will scan it a third time to compute *BC*.**

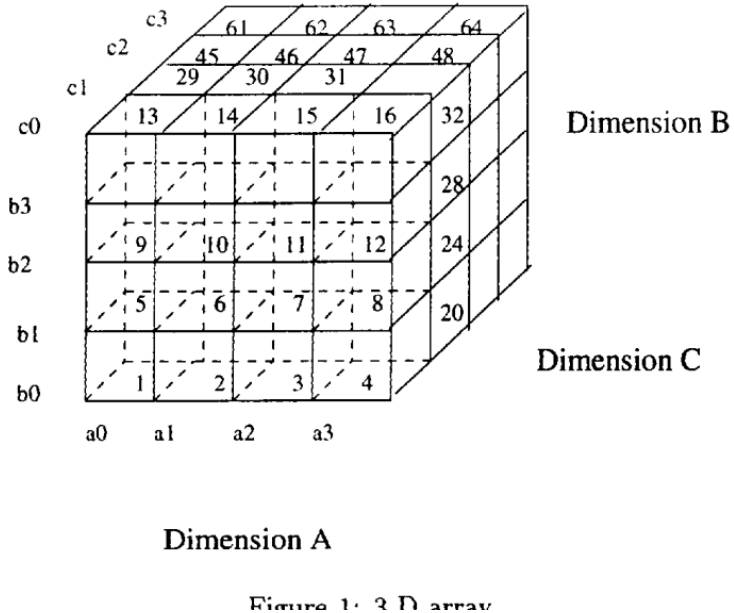


Figure 1: 3 D array

Multi-Way Array Aggregation

- Determine the dimensional order
  - When computing multiple group-bys simultaneously, the total memory required depends critically on the order in which the input array is scanned.
  - Basic idea: exploit a special logical order called "dimension order".
- Naïve algorithm:
  - We can formulate a general rule to determine what chunks of each group-by of the cube need to stay in memory in order to avoid rescanning a chunk of the input array.

- Example:
  - The array chunks are read in the dimension order *ABC,* i.e.. from chunk 1 to chunk 64. Suppose chunk 1 is read in.
  - For group-by *AB,* this chunk is aggregated along the C dimension to get a chunk of *AB.*
  - Similarly for *AC* and *BC,* this chunk is aggregated along *B* and *A* dimensions respectively.
  - Thus the first chunk's *AB* group by is aggregated to the chunk (a_0, b_0) of *AB;* the first chunk's *AC'* is aggregated to the chunk (a_0, c_0) of *AC;* the first chunk's *BC* is aggregated to the chunk (b_0, c_0) of *BC.*
  - As we read in new chunks, we aggregate the chunk's AB, *AC* and *BC* group-by to the corresponding chunks of group-bys *A B,* AC and *BC.*

Multi-way aggregation

- To compute each chunk of ***AB, AC,* and *BC*** group-by, we may naively allocate memory to each chunk of those group-bys in memory.
- Basic idea:
  - However, **we can exploit the order** in which each chunk is brought in memory to reduce the memory required by all group-by (AB,AC,BC, A,B,C)
- Example:
  - Read the chunks in dimension order (A, ***B, C)*** layout, which is a linear order from chunk 1 to chunk 64.
  - For the chunk 1 to chunk 4, complete the aggregation for the chunk (b_0, c_0) of *BC* after *aggre*gating each chunk's *BC* group-by to the chunk (b_0, c_0) of *BC*.
  - Once the (b_0, c_0) chunk is completed, we write out the chunk and reassign the chunk memory to the chunk (b_1, c_0) which is computed from the next 4 chunks of *ABC i.e.* the chunk 4 to chunk 8. **So we allot only one chunk of *BC* in memory to compute the entire *BC* group-by.** Similarly, we can carry out group by for AC group-by and AB group-by
- Notice that we generate each ***BC*** chunk in the dimension order ***(B, C).***
- **Before we write each *BC* chunk to disk,** we use the *BC* chunks to compute the chunks of *B* or C as if we read in each *BC* chunk in the dimension order *(B, C)*.

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

- Method: the chunks 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

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - **3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)**
  - 4**.** Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - 7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

Bottom-Up Computation of Sparse and Iceberg CUBEs

Kevin Beyer
Computer Sciences Department
University of Wisconsin – Madison
beyer@cs.wisc.edu

Raghu Ramakrishnan
Computer Sciences Department
University of Wisconsin – Madison
raghu@cs.wisc.edu

Presentation made by

P. Krishna Reddy, IIIT Hyderabad

Outline

- Introduction
- Motivation
- The Iceberg-CUBE Problem
- Previous CUBE Algorithms
- Algorithm Bottom-Up Cube
- Performance Analysis
- Conclusions

Introduction

- The basic CUBE problem computes all aggregates as efficiently as possible.
- The chief difficulty is that
  - The CUBE problem is exponential in the number of dimensions: for d dimensions, 2^d group-bys are computed.
  - The size of each group-by depends upon the cardinality of its dimensions.
  - If every store sold every product, then the (Product, Store) group-by would have |Product| x |Store| result tuples.
  - However, as the number of dimensions or the cardinalities increase, the product of the cardinalities grossly exceeds the (fixed) size of the input relation for many of the group-bys.
  - Even in our small example, if the data comes from a large department store it is highly unlikely that a given customer purchased even 5% of of the products, or shopped in more than 1% of the stores.

Contributions

- Introduced a variant of CUBE problem called Iceberg-CUBE
- We present a simple and efficient algorithm, called Bottom Up CUBE (BUC), for the Iceberg-CUBE problem
- We present a simple and efficient algorithm, called BUC, for the Iceberg-CUBE problem

Motivation

- A full CUBE on a real nine-dimensional dataset containing weather conditions at various weather stations on land for September 1985.
  - The dataset had 1,015,367 tuples (~39MB).
  - The CUBE on this dataset produces 210,343,580 tuples (~8GB)-more than 200 times the input size!
- Two alternatives to improve the performance
  - Compute a subset of CUBEs
  - Iceberg CUBE: proposed approach as a new alternative

The Iceberg-CUBE Problem

•CUBE

SELECT A,B,C,COUNT(•),SUM(X)
FROM R
CUBE BY A,B,C

•Iceberg CUBE

SELECT A,B,C,COUNT(•),SUM(X)
FROM R
CUBE BY A,B,C
HAVING COUNT(•) > N

•An Iceberg query computes a single group-by and eliminates all tuples with an aggregate value below some threshold. For example:

```
SELECT  A,B,C,COUNT(*)
FROM    R
GROUP BY A,B,C
HAVING COUNT(*) >= N
```

Previous Approaches

- Pipesort
  - Converts a CUBE lattice into processing tree
- PipeHash
  - Estimate
  - Compute by starting from smallest parent
- Overlap
  - Sorting order of Prefix attributes is shared
  - Overlap as much sorting as possible by computing a group-by from a parent with the maximum sort-order overlap.
- ArrayCube
  - it uses in-memory arrays to store the partitions and to avoid sorting.
- PartitionedCube and MemoryCube
  - PartitionedCube partitions the data on some attribute into memory-sized units
  - and MemoryCube computes the CUBE on each in-memory partition.

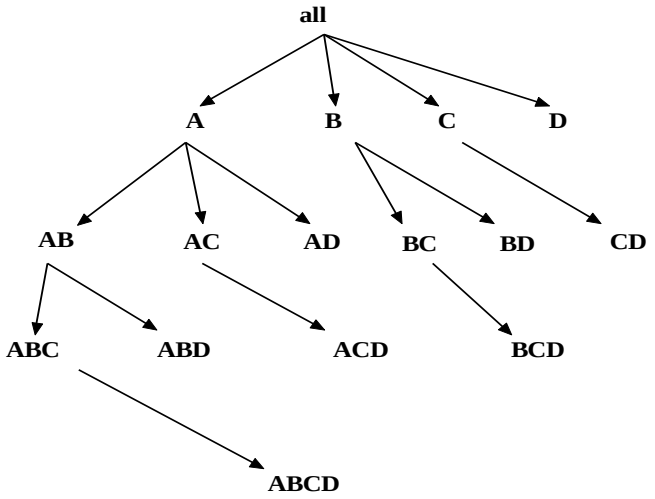
Algorithm Bottom-Up Cube (BUC)

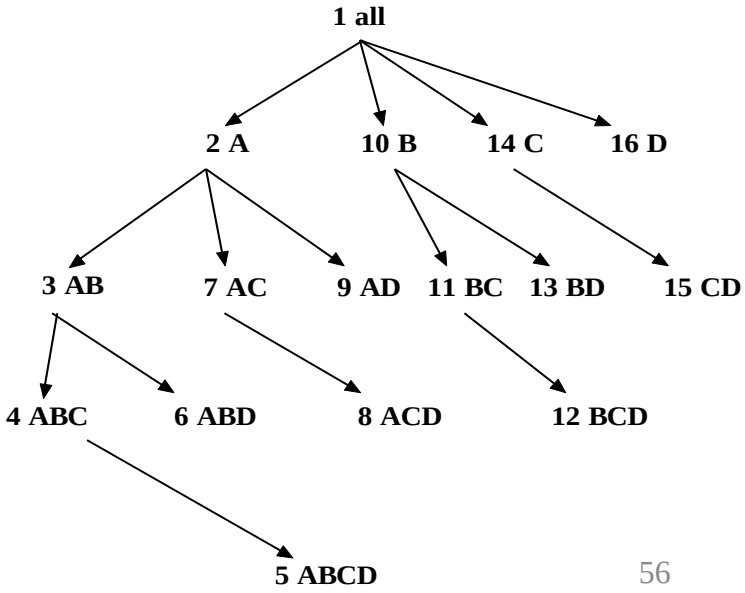
- BUC is proposed for sparse CUBE and Iceberg-CUBE computation.
- Basic idea
  - The idea in BUC is to combine the 1/0 efficiency of PartitionedCube/ MemoryCube, but to take advantage of minimum support pruning like Apriori.
- To achieve pruning, BUC proceeds from the bottom of the lattice (i.e., the smallest / most aggregated group-bys), and works its way up towards the larger, less aggregated group bys.
- All of the previous algorithms compute in the opposite direction. Since parent group-bys are used to compute child group-bys, the algorithms cannot avoid computing the parents.

Bottom-Up Computation (BUC)

- Bottom-up cube computation

  (Note: top-down in our view!)
- Divides dimensions into partitions and facilitates iceberg pruning
  - If *minsup* = 1 $\Rightarrow$ compute full CUBE!
- No simultaneous aggregation

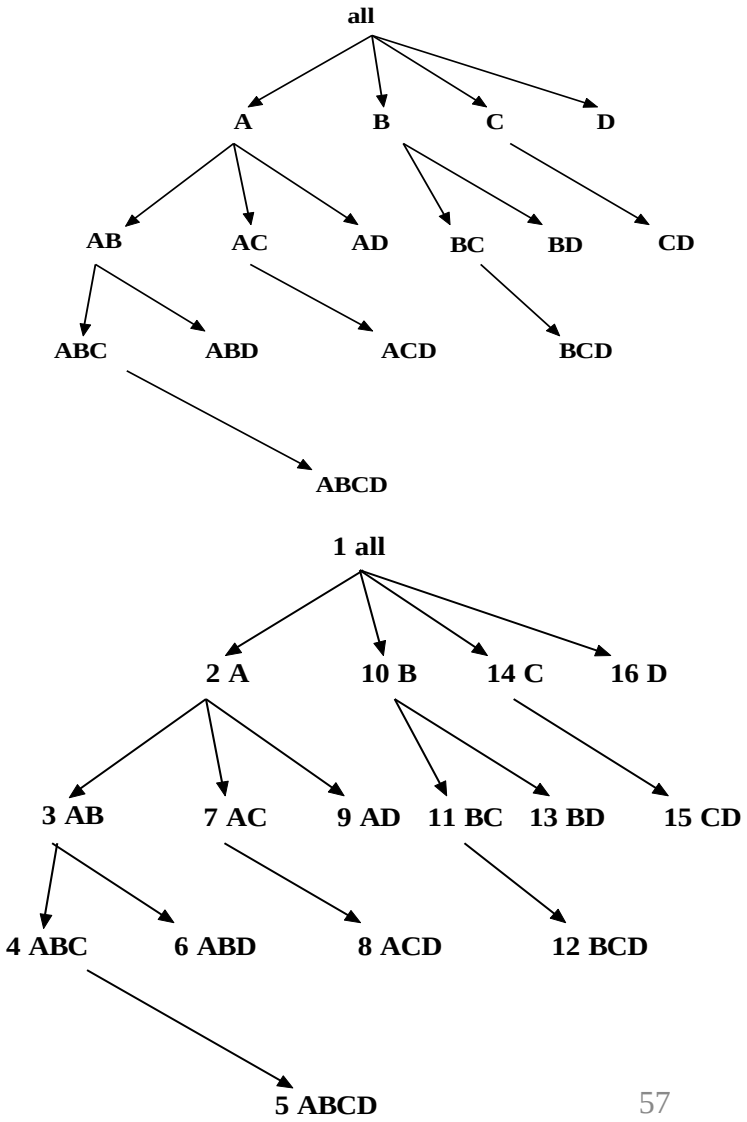




56

Iceberg CUBE with BUC

- **If a partition does not satisfy *min_sup*, its descendants can be pruned**
- No simultaneous aggregation
- When a small partition is found, instead of writing for all of the group-bys, BUC simply skips the partition and does not consider any of the partition's ancestors.
- The pruning is correct because the partition sizes are always decreasing when BUC recurses, and therefore none of the ancestors can have minimum support.

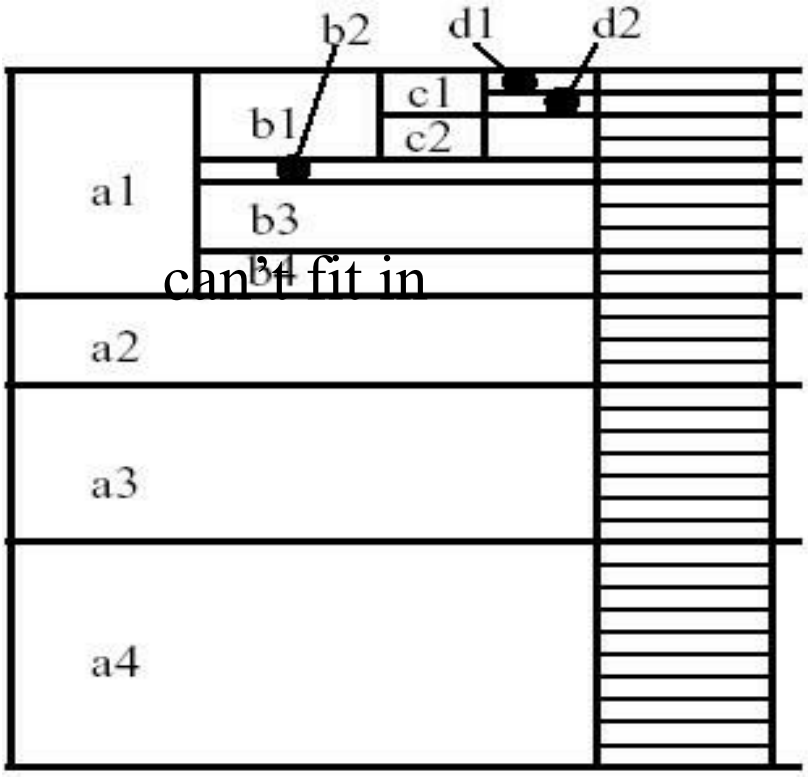


57

Additional Pruning Functions

- For any two sets (of tuples, in our case) S and *T* such that *S is a subset of T*, a function *f* is **monotonically decreasing** if *f(S) <= f(T)*.
  - Counting
- Average is not monotonic
- Consider HAVING COUNT(*) < X.
  - In this case, the user wants the groups with little aggregation, which occur towards the top of the lattice.
  - BUC can not help

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!

- **Partitioning**
  - When 'input' does not fit in main memory, the data must be partitioned to disk. This can be done with hash-partitioning, followed by a partitioning within each bucket (which hopefully now fits in main memory), or external-sort can be used.
  - When performing an external partitioning, the aggregation step can be combined with the partitioning.
- **Dimension Ordering**
  - The performance of BUC is sensitive to the ordering of the dimensions.
  - The goal of BUC is to prune as early as possible; i.e., BUG wants to find partitions that do not meet minimum support
  - For best performance, the most **discriminating** dimensions should be used first
  - **Cardinality**
    - **Higher the cardinality and the smaller the partitions**
  - **Skew**
    - **Skewed dimensions will have less cardinality and so should be considered later**
  - **Correlation**
    - **If the dimension is correlated with earlier dimension, it decreases pruning.**

Experimental Results

- See the research paper
- Results on real data
  - Weather data
  - Mail order data

Conclusion

- Major contribution
  - Sharing of partition costs
  - Like multiway, it can not share the computation of aggregates between parent and child group-bys.
  - Because, computation cuboid AB does not help ABC.

Topics till this slide are for QUIZ 1

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - 3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - **4. Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)**
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - 7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

Efficient Computation of Iceberg Cubes with Complex Measures *

Jiawei Han† Jian Pei† Guozhu Dong‡ Ke Wang†

† School of Computing Science, Simon Fraser University, B.C., Canada, {han, peijian, wangk}@cs.sfu.ca

‡ Department of Computer Science, Wright State University, Dayton, OH, U.S.A., gdong@cs.wright.edu

SIGMOD 2001

Presented by P. Krishna Reddy
IIIT Hyderabad

Outline

- Introduction
- Iceberg cube with the average measures
- Exploration of Weaker, Anti-monotonic Conditions
- Extension of Apriori and BUC for Iceberg Cube with Average
- Top-k H-Cubing: Top-k Cubing Using a Hyper-Tree Structure
- Performance Analysis
- Conclusions

Introduction

- Materializing iceberg cubes is one of existing method.
  - where an iceberg cube is a subset of a cube containing only those cells whose measure, such as count, satisfies certain constraints, such as minimal support threshold
- Advantage of Iceberg cube
  - Most of the cells at the low level cuboids are likely to contain trivial aggregate values and may not pass certain threshold, and therefore, do not need to be computed in an iceberg cube.
  - This not only saves processing time and disk space but also makes the analysis focused only on interesting data

Introduction

- Previous studies on efficient computation iceberg cubes have been conned to with simple measures, such as count and sum, by exploring the anti-monotonic property of such icebergs.
  - For example, if the count of a cell c in a cuboid C is no higher than v, then the count of any of c's descendant cells in the lower level cuboids can never be higher than v, and thus can be pruned by the Apriori-like method.
- Unfortunately, not all the measures have such antimonotonic property.
  - For example, even if the average value in a cell c of a cuboid C is no higher than v, the average value of some of c's descendant cells in the lower level cuboids may still be higher than v.
- **We study how to efficiently compute iceberg cubes with non-anti-monotonic measures**

Example queries

Example 1 Suppose a sales database has four dimensions: *time, location, customer*, and *product*, and two measures: *price* and *cost* (note: *profit* = *price* − *cost*). The following queries require the computation of iceberg cubes with such *complex* measures.

Q_1: Find groups of sales which contain at least 50 items and whose average item price is at least \$800, grouped by month, city, and/or customer groups.

Q_2: Find groups of sales which contain at least 200 items and whose *total profit*^{2} is more than \$6000, grouped by month, city, and/or customer groups.

Q_3: For sales grouped by month, city, and/or customer groups, containing at least 20 items, with an average item price of no less than \$800, find those customer groups on which one can make at least 10% more profit than the average of all the customers. □

Contributions

- It develops a mapping which transforms some non-antimonotonic testing conditions to somewhat weaker but anti-monotonic testing conditions.
  - For example, test on average can be mapped to an anti-monotonic, top-k average test. Mappings for several other measures are worked out as well.
- It extends two previously studied methods, Apriori and BUC, to Top-k Apriori and Top-k BUC, for computing iceberg cubes with the average measure.
- To further improve the performance for computing iceberg cubes, a hypertree structure, called H-tree, is designed, and a new iceberg cubing method, called Top-k H-Cubing, is developed.
- The method explores several efficient processing techniques, including
  - tree-based data compression
  - dynamic link adjustment, and
  - quantitative information merge

Outline

- Introduction
- **Iceberg cube with the average measures**
- Exploration of Weaker, Anti-monotonic Conditions
- Extension of Apriori and BUC for Iceberg Cube with Average
- Top-k H-Cubing: Top-k Cubing Using a Hyper-Tree Structure
- Performance Analysis
- Conclusions

Example 2 (Iceberg cubes on average) Consider a *Sales_Info* table, given in Table 1, which registers sales related to month, day, city, customer group, product, cost, and price.

Mon.	Day	City	Cust_group	Product	Cost	Price
Jan	10	Toronto	Edu.	HP Printer	500	485
Jan	15	Toronto	Household	Sony TV	800	1200
Jan	20	Toronto	Edu.	Canon Camera	1160	1280
Feb	20	Montreal	Busi.	IBM Laptop	1500	2500
Mar.	4	Vancouver	Edu.	Seagate HD	540	520
...	...	...	...	...	...	...

Table 1: A *Sales_Info* table.

An iceberg cube, *Sales_Iceberg*, which computes Q_1 is presented as follows.

```
CREATE CUBE Sales_Iceberg AS
SELECT month, city, customer-group, AVG(price),
       COUNT(*)
FROM Sales_Info
CUBEBY month, city, customer-group
HAVING AVG(price) >= 800 AND COUNT(*) >= 50
```

- Sales Iceberg is the restriction of the data cube
  - Sales Iceberg excludes all the cells of the latter whose average price is less than $800 or whose count is less than 50.
- It is also different from its corresponding Iceberg query,
  - Iceberg CUBE query contains the qualified cells of all the possible group-bys of the three dimensions
  - Sales_Iceberg contains only the qualified cells with the three dimensions grouped together

Definition

- Given an iceberg cube ICube, the (whole) data cube formed by the same specification of ICube without the HAVING clause is called the background cube of ICube, and is denoted as B(ICube).

$\mathcal{B}(Sales\_Iceberg)$, the *background cube* of the iceberg cube, *Sales_Iceberg*, of Example 2 is defined as,

```
CREATE CUBE Sales_Cube AS
SELECT month, city, customer-group, AVG(price),
       COUNT(*)
FROM Sales_Info
CUBEBY month, city, customer-group
```
□

Definition 2 An iceberg cube, *ICube*, is **anti-monotonic** if and only if for each cell c in $\mathcal{B}(ICube)$, if c *violates* the constraint specified by *ICube*'s `HAVING` clause, so does every descendant of c. □

Example 4 Given the sales table in Example 2, *Count_Iceberg*, shown below, is *anti-monotonic*.

```
CREATE CUBE Count_Iceberg AS
SELECT month, city, customer-group, COUNT(*)
FROM Sales_Info
CUBEBY month, city, customer-group
HAVING COUNT(*) >= 100
```

- If a cell c count is less than 100, then every descendant of c will violate the constraint since the count of each subcube of c must be no larger than that of c.
- Sales Iceberg in Example 2 is, however, not antimonotonic.
  - For example, even when the average price of all the items sold in March is less than $800, e.g., (March; *; *; 600; 1800), the average price for a subset containing only the sales to business people, e.g., (March;* ; Busi:; 1300; 360), may still satisfy the constraint specified in the HAVING clause.

Outline

- Introduction
- Iceberg cube with the average measures
- **Exploration of Weaker, Anti-monotonic Conditions**
- Extension of Apriori and BUC for Iceberg Cube with Average
- Top-k H-Cubing: Top-k Cubing Using a Hyper-Tree Structure
- Performance Analysis
- Conclusions

- **Question:** As iceberg cube involves the non-anti-monotonic measure average, it does not have the Apriori property. Can we find a weaker but anti-monotonic auxiliary condition that may help us compute iceberg cubes efficiently?
- **Solution:** Yes, Top-k average: An anti-monotonic condition for testing average

3.1 Top-k average: An anti-monotonic condition for testing average

Let us examine the following iceberg cube $AvgI$, a generalization of Example 2, defined on a relational table T with i dimensions and one measure M.

CREATE CUBE $AvgI$ AS
SELECT $A_1, A_2, \ldots, A_m$, AVG(M), COUNT($*$)
FROM T
CUBEBY $A_1, A_2, \ldots, A_m$
HAVING AVG(M) >= v AND COUNT($*$) >= k

Definition 3 A cell c is said to have n base cells if it covers n nonempty descendant base cells. The **top-k average** of c, denoted as $avg^k(c)$, is the average value of the *top-k base cells* of c (i.e., the first k cells when all the base cells in c are sorted in value-descending order) if $k \leq n$; or $-\infty$ if $k > n$.[3] □

Lemma 3.1 (Top-k Apriori) *Let c be an m-d cell which fails to satisfy $avg^k(c) \geq v$ in cube $\mathcal{B}(AvgI)$. If a cell c' is a descendant of c, then c' cannot satisfy $avg^k(c') \geq v$ in cube $\mathcal{B}(AvgI)$.* □

This lemma stimulates us to explore the utilization of the auxiliary condition $avg^k(c) \geq v$ as a looser bound for computing iceberg cubes with the `HAVING` clause "$avg(c) \geq v$ `AND` $count(c) \geq k$". The effectiveness of the search space pruning by top-k average is demonstrated in our performance study in section 6.

Optimization: Binning technique for top-k average

- Issue: How to keep track of top-k average values? If k is large, it is substantial.
- Solution: Binning technique can be used to reduce
  - the cost of storage and
  - computation of top-k average.
- Two ideas: Large value collapsing and small value binning
- Large value collapsing
  - For any measure value v' which is no less than v (i.e., v' >= v) in $avg^k(c)$ >= v, where v' is called a large value, there is no need to store it explicitly.
  - Instead, it is sufficient to store only two measures: (1) count, the number of large values, and (2) sum, the sum of all large values.
- Small value binning:
  - If the large values registered can make $avg^k(c)$ >= v, there is no need to store small ones (a value v' is small if v' < v).
  - Otherwise, we can set up a small set of bins and register two measures, count and sum, for each bin.
- The large-value group can be considered as a special bin, bin_1.
- Let the upper value boundary of bin_i be $\max(bin_i)$ and the lower one be $\min(bin_i)$.
  - For all (1 <= i < j), we have $\min(bin_i) > \max(bin_j)$.
- To make binning more effective, denser bins are used for the region closer to v, and sparser bins for the region relatively far away from v.
  - For example, suppose v >= 0, one can set up the ranges of five bins.
    - range(bin[1]) = [v; ∞),
    - range(bin[2]) = [0:95v; v),
    - range(bin[3]) = [0:85v; 0:95v),
    - range(bin[4]) = [0:70v; 0:85v), and
    - range(bin[5]) = [0:50v; 0:70v).
- Notice since we have count and sum of all the cells, that for the remaining range [- ∞; 0:50v) can be derived easily.

The set of bins for a cell c can be used to judge whether $avg^k(c) \geq v$ is false as follows. Let m be the smallest number such that the sum of counts of the upper m bins is no less than k, i.e., $count_m = \Sigma_{i=1}^{m} count(bin_i) \geq k$. We approximate $avg^k(c)$ using,

$avg'^k(c) = (\Sigma_{i=1}^{m-1} sum(bin_i) + max(bin_m) \times n_k)/k$,

where $n_k = k - \Sigma_{i=1}^{m-1} count(bin_i)$.

Lemma 3.2 *$avg^k(c) \leq avg'^k(c)$. Consequently, if $avg'^k(c) < v$, then no descendant of c can satisfy the* Having-*condition in AvgI.* □

- We denote three pieces of information sum, count, and top-k bins as quant-info, which often need to be accumulated with each cell for efficient computation of average iceberg cubes.

Outline

- Introduction
- Iceberg cube with the average measures
- Exploration of Weaker, Anti-monotonic Conditions
- **Extension of Apriori and BUC for Iceberg Cube with Average**
- Top-k H-Cubing: Top-k Cubing Using a Hyper-Tree Structure
- Performance Analysis
- Conclusions

Top-k Apriori

- Scan DB once to accumulate quant-info (i.e., count, sum, and top-k bin measures) for the 0-d cell c0 of the 0-d cuboid.
- Output the 0-d cuboid,
  - $R_0 = \{c_0 \ / \ \text{count}(c_0) >= 50 \ \wedge \ \text{avg}(c_0) = \frac{\text{sum}(c_0)}{\text{count}(c_0)} >= 800\}$, and
  - Keep the 0-d live set, $L_0 = \{c_0 \ / \ avg^{l50}(c_0) >= 800\}$. If L_0 =Φ;, the computation terminates.
- Otherwise, compute 1-d cells as follows.
  - All the 1-d cells are candidate cells, i.e., forming the candidate set C_1, such as (Jan; ; ; : : :), (Feb; ; ; : : :), . . . , (; T oronto; ; : : :), (; Vancouver; ; : : :), ...
  - Then scan DB, accumulate quant-info for each c_1 in C_1, output R_1, and keep the live set L_1:
    - $R_1 = \{c_1 \ / \ c_1 \ \varepsilon \ C_1 \ \wedge \ \text{count}(c_1) >= 50 \ \wedge \ \text{avg}(c_1) = \frac{\text{sum}(c_1)}{\text{count}(c_1)} >= 800\}$
    - $L_1 = \{c_1 \ / \ c_1 \ \varepsilon \ C_1 \wedge avg^{l50}(c_1) >= 800\}$
- This process continues level-by-level, until the live set L_k or the candidate set C_k for some k is empty
- Top-k Apriori computes average iceberg cubes by exploring candidate generation and level-wise computation.
- However, it still involves costly processing:
  - (1) it takes m scans of DB where m is the maximum number of dimensions containing nonempty candidate set, and
  - (2) it may generate a huge number of candidate sets.

Top-k BUC

- An efficient iceberg cube computation method BottomUp Cube (BUC) builds the cube from lower number of dimension combinations to higher ones.
- It explores the dimension ordering by putting the most discriminating dimensions rst and then recursively partitioning DB according to the ordering.
- At each step of recursive partition,
  - one can push in the iceberg constraint, such as min count, to remove those that cannot satisfy it.
  - This can be applied to computing iceberg cubes with the average measure.
    - For example, for computing AvgI, one can use $avg^k(c) > v$ to test the partitions generated: any partition that cannot pass the test will not need to be considered further

Outline

- Introduction
- Iceberg cube with the average measures
- Exploration of Weaker, Anti-monotonic Conditions
- Extension of Apriori and BUC for Iceberg Cube with Average
- **Top-k H-Cubing: Top-k Cubing Using a Hyper-Tree Structure**
- Performance Analysis
- Conclusions

Top-k H-Cubing: Top-k Cubing Using a Hyper-Tree Structure

- Example: a tree structure HT structure
- Tree HT has a root node null. and dimensions are in cardinality-ascending order, i.e. R: G, M, C.
- A header table is created, in which each entry records the quant-info for an attribute-value pair.
  - The rst tuple, t1 = (Edu:; J an; T oronto; 485), is inserted into HT , with three nodes, Edu:, Jan and Toronto inserted in sequence to form the rst branch, and quant-info in the leaf(Toronto).
  - Also, price 485 is used to update quant-info for Edu:, Jan and Toronto in the header table.
  - Similarly, t2 = (Household; Jan; Toronto; 1200), is inserted. Since the two leaf nodes have the same label, they are linked by a side-link.
  - Since t3 = (Edu:; Jan; Toronto; 1280) has the same attribute values as t1, t3 shares the path as t1, with quant-info in the leaf and header updated.
- The remaining tuples can be inserted similarly (Fig. 2, next slide).
- The tree so formed is called an H-tree. Its construction requires only one scan of the database.

Mon.	Day	City	Cust_group	Product	Cost	Price
Jan	10	Toronto	Edu.	HP Printer	500	485
Jan	15	Toronto	Household	Sony TV	800	1200
Jan	20	Toronto	Edu.	Canon Camera	1160	1280
Feb	20	Montreal	Busi.	IBM Laptop	1500	2500
Mar.	4	Vancouver	Edu.	Seagate HD	540	520
...	...	...	...	...	...	...

Table 1: A *Sales_Info* table.

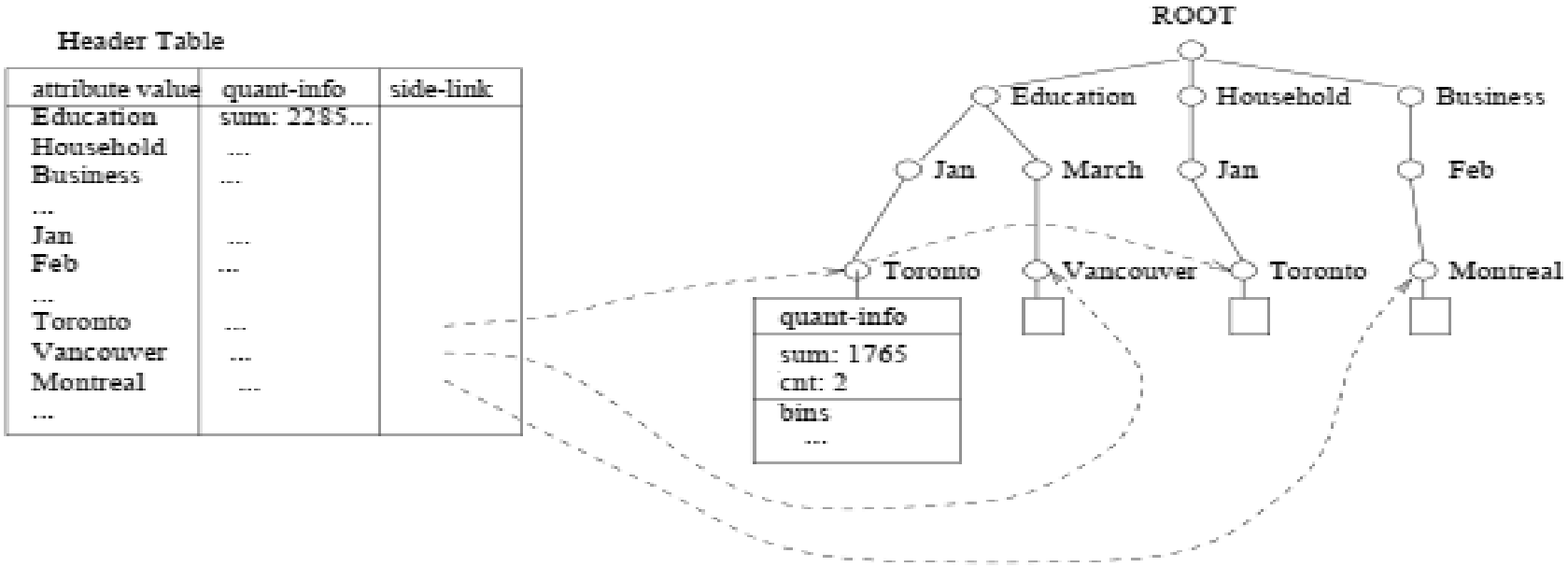


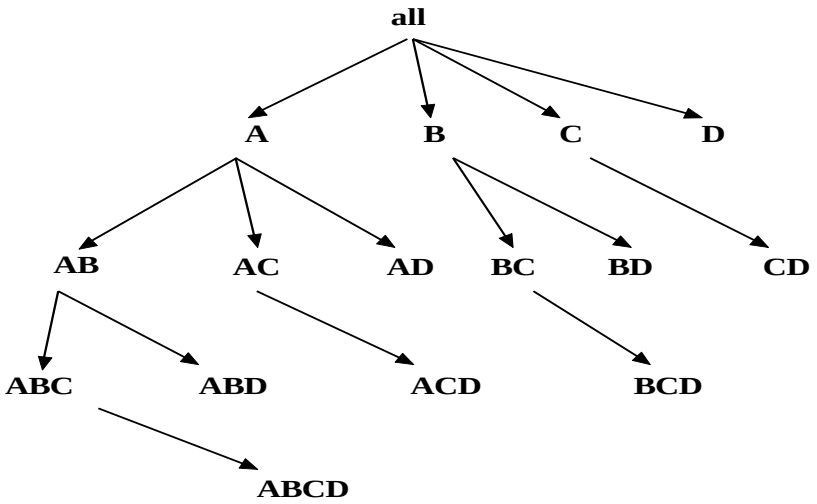
Figure 2: An H-tree.

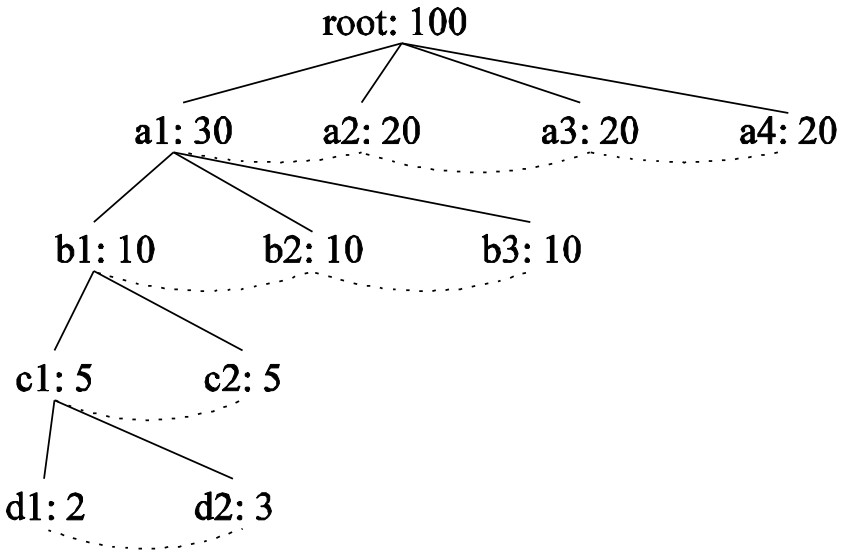
Properties of H-tree

- The H-tree HT has the following properties.
- 1. (Construction cost) The H-tree can be constructed by scanning the database only once.
- 2. (Completeness) The H-tree and its header table H contain the complete information needed for computing iceberg cube AvgI.
- 3. (Compactness) Let there be n tuples in table T and m attributes involved in AvgI. The number of nodes in H-tree cannot exceed n*m + 1.

H-Cubing: Using H-Tree Structure

- Bottom-up computation
- Exploring an H-tree structure
- If the current computation of an H-tree cannot pass min_sup, do not proceed further (pruning)
- No simultaneous aggregation





Top-k H-Cubing: Computing iceberg cubes using H-tree

- Step 1. Compute cells involving dimension C.
- The quant-info in the H-tree tells whether a cell in the form of (; ; c), where c is a city, passes the top-k average and average tests.
- For example, the entry Toronto in the header table contains $avg^k(c)$, $avg^k(price)$ and avg(price) for (*; *; Toronto).
- If avg(price) passes the average price threshold, output the cell.
- If $avg^k(price)$ passes it, the descendants of the cell (*; *;Toronto) should be examined as shown below.
  - The sub-graph of HT containing only the paths related to Toronto, denoted as $HT_{Toronto}$, is an H-tree for sub-cube (*;*; Toronto). $HT_{Toronto}$ is sufficient to compute the iceberg sub-cube w.r.t. Toronto.
  - Traverse the side-link Toronto in the header table H and
    - (1) make a copy of quant-info in every leaf-node labeled Toronto to its parent node in the tree,
    - (2) build a new header table $H_{Toronto}$, which collects quant-info for every attribute-value w.r.t. Toronto, and
    - (3) link all the parent nodes of the leaf-nodes labeled Toronto. (Fig. 3 is the updated tree)
  - Based on the header table $H_{Toronto}$, output all the cells of the form (*; m; Toronto) or (g; *; Toronto), which pass the average price test, where m ε M and g ε G. Also, explore recursively the descendants of the cells of the form (*; m; Toronto) or (g; *; Toronto) which pass the top-k average test.

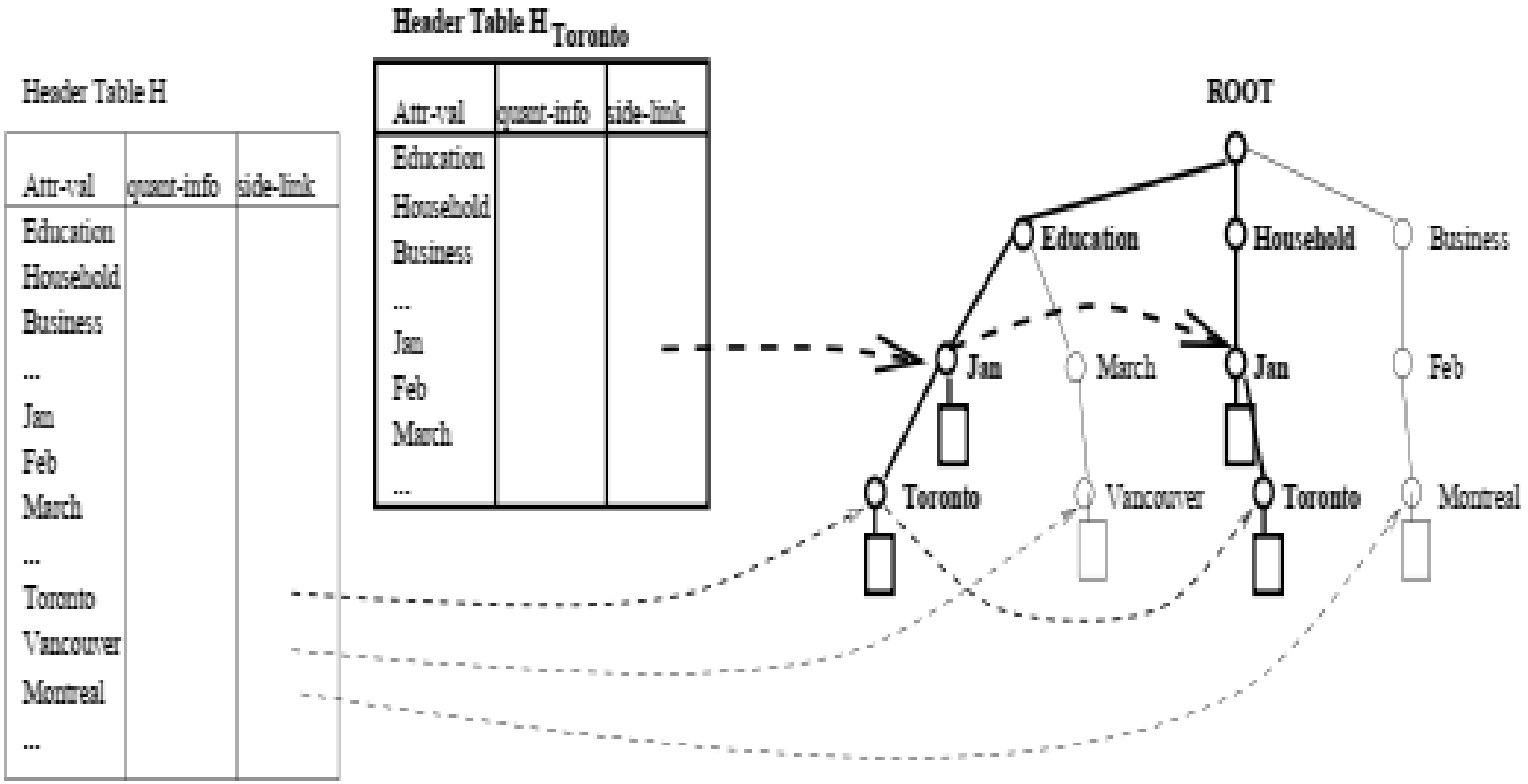


Figure 3: Updated H-tree for computing descendants of $(*, *, Toronto)$.

Step 2. Compute cells involving dimension M but no C

- 1. Roll-up quant-info to dimension M.
  - Every leaf node in H-tree merges its quant-info into that of its parent node. All nodes labeled by a common month, should be linked by side-links and also linked to the corresponding row in the header table H.
- 2. Compute cells involving M but no C. This is similar to Step 1 demonstrated before.

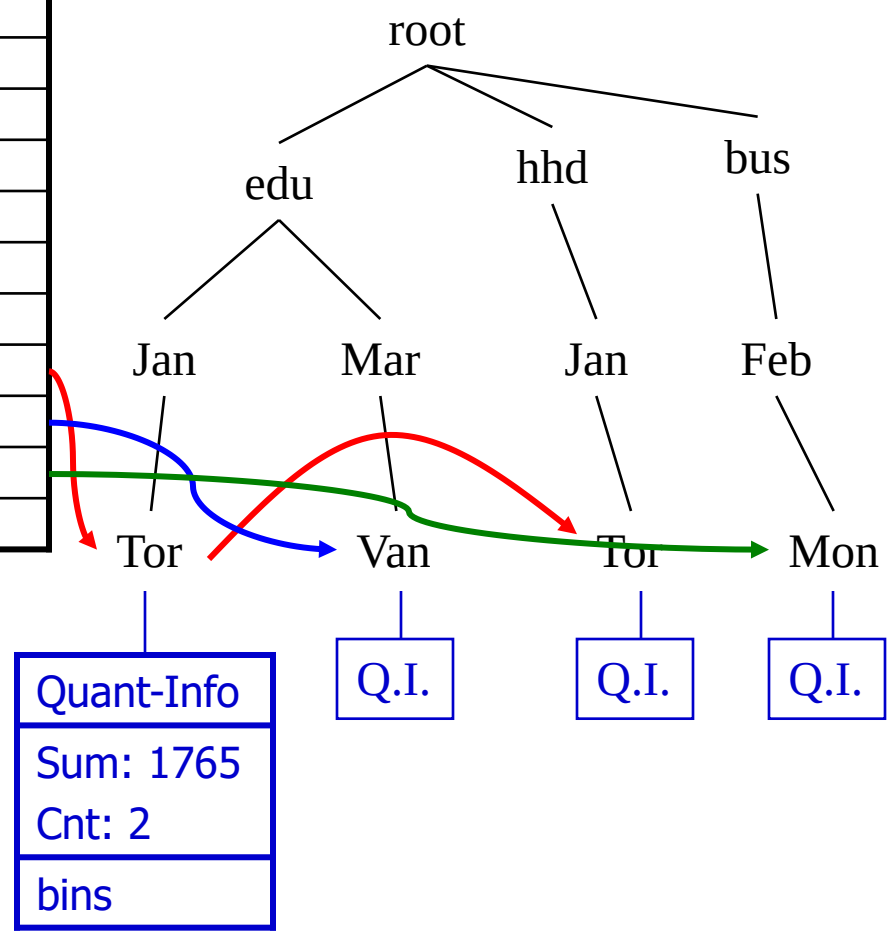
Reduction of Database Scans

- When the H-tree for a partition can not fit in memory, we turn to H-tree –based mining.
- We can use a a dual projection method, which requires less memory and disk space but scans DB and each partitioned database only once.

H-tree: A Prefix Hyper-tree

Header table

Attr. Val.	Quant-Info	Side-link
Edu	Sum:2285 …	
Hhd	…	
Bus	…	
…	…	
Jan	…	
Feb	…	
…	…	
Tor	**…**	
Van	**…**	
Mon	**…**	
…	…	



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

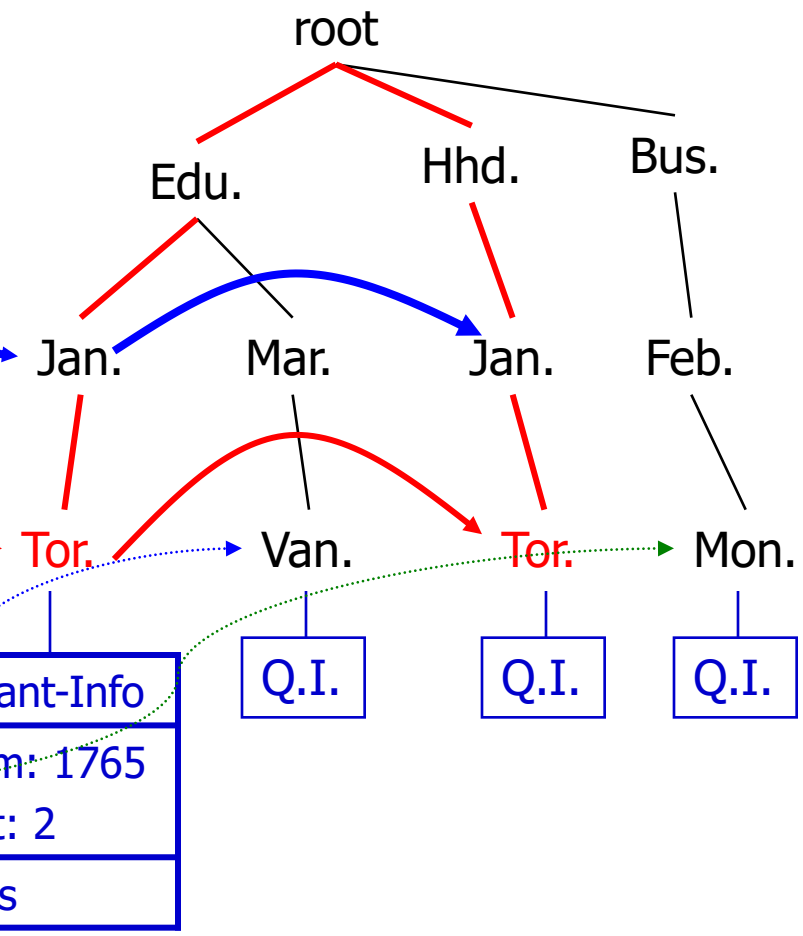
Computing Cells Involving "City"

Header Table H_{Tor}

Attr. Val.	Q.I.	Side-link
Edu	…	
Hhd	…	
Bus	…	
…	…	
Jan	**…**	
Feb	…	
…	…	

From (*, *, Tor) to (*, Jan, Tor)

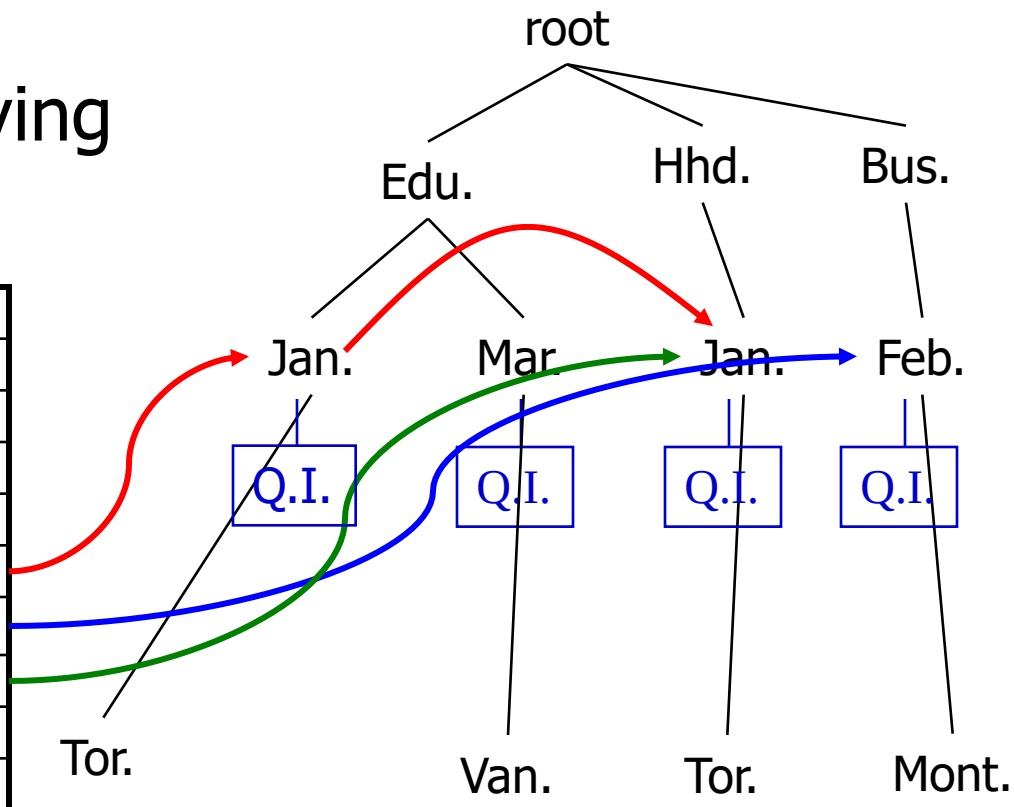
Attr. Val.	Quant-Info	Side-link
Edu	Sum:2285 …	
Hhd	…	
Bus	…	
…	…	
Jan	…	
Feb	…	
…	…	
Tor	**…**	
Van	…	
Mon	…	
…	…	



Computing Cells Involving Month But No City

1. Roll up quant-info
2. Compute cells involving month but no city

Attr. Val.	Quant-Info	Side-link
Edu.	Sum:2285 …	
Hhd.	…	
Bus.	…	
…	…	
Jan.	**…**	
Feb.	**…**	
Mar.	**…**	
…	…	
Tor.	…	
Van.	…	
Mont.	…	
…	…	

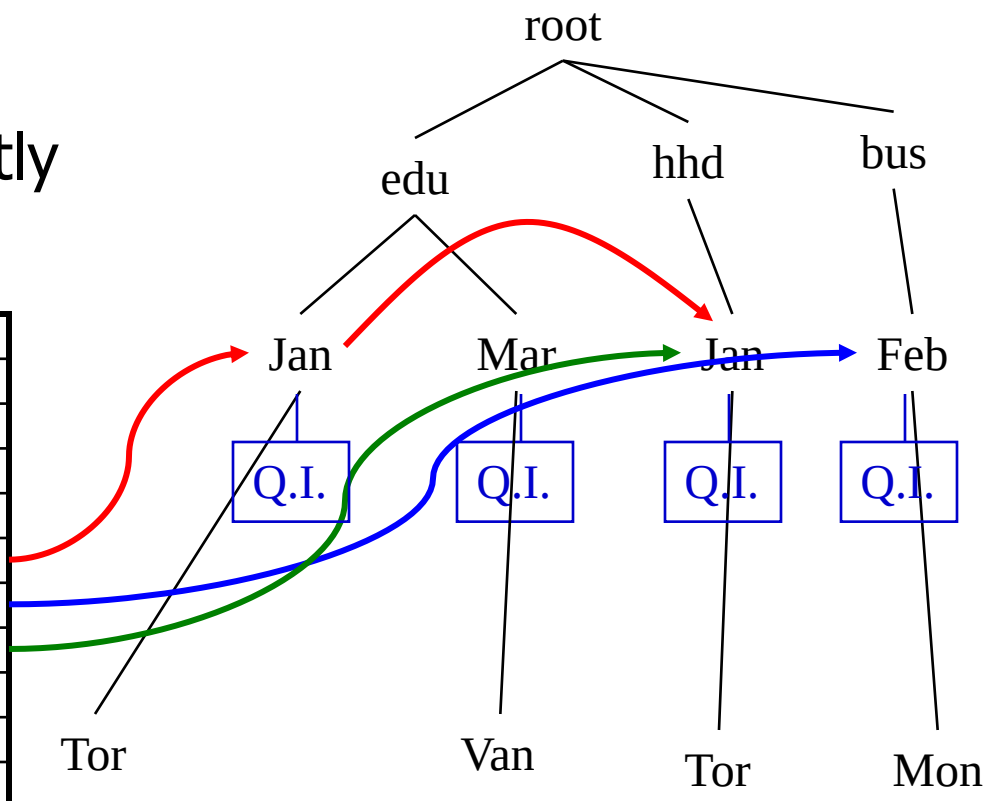


Top-k OK mark: if Q.I. in a child passes top-k avg threshold, so does its parents. No binning is needed!

Computing Cells Involving Only Cust_grp

Check header table directly

Attr. Val.	Quant-Info	Side-link
Edu	**Sum:2285 …**	
Hhd	**…**	
Bus	**…**	
…	…	
Jan	…	
Feb	…	
Mar	…	
…	…	
Tor	…	
Van	…	
Mon	…	
…	…	



Performance Analysis

- See the paper

Discussion

- The proposed approach can be extended to other measures, in addition to average.

Summary

- Studies issues on efficient computation of iceberg cubes with some popularly encountered complex measures.
- It contributes to iceberg cube computation in two aspects:
  - A methodology is developed that derives a weaker but antimonotonic condition for testing and pruning search space
  - Instead of simple extension of two previously studied methods, Apriori and BUC, to Top-k Apriori and Top-k BUC, an interesting hypertree structure, called H-tree, is designed and a new iceberg cubing method, is developed.
- Top-k H-Cubing performs better than the existing method.
- Remark
  - Interesting progress has been made for efficient computation of iceberg cubes with some complex measures.
  - Efficient computation of iceberg cubes with some other complex measures is still an open problem.
  - Moreover, the application of the H-tree structure and its computation method to other OLAP and data mining tasks may deserve further attention.

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - 3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4. Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
    - **5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)**
    - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - 7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

Star-Cubing: Computing Iceberg Cubes by Top-Down and Bottom-Up Integration*

Dong Xin Jiawei Han Xiaolei Li Benjamin W. Wah

University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

VLDB 2003

Presented by

P. Krishna Reddy, IIIT Hyderabad

Outline

- Introduction
- Overview of cubing algorithms
- Start-Cubing: An Integration of top-down and bottom-up approach
- Performance Analysis
- Conclusion

Introduction

- Data cube computation is one of the most essential but expensive operations in data warehousing.
- Previous studies have developed two major approaches, top-down vs. bottom-up, for efficient cube computation.
  - MultiWay Array Cube (called MultiWay) algorithm aggregates simultaneously on multiple dimensions;
    - however, it cannot take advantage of Apriori pruning when computing iceberg cubes.
  - BUC and H-Cubing, computes the iceberg cube bottom-up and facilitates Apriori pruning.
    - BUC explores fast sorting and partitioning techniques; whereas
    - H-Cubing explores a data structure, H-Tree, for shared computation.
- In this paper, we present a new method, StarCubing, that integrates the strengths of the previous three algorithms and performs aggregations on multiple dimensions simultaneously.
- It utilizes a star-tree structure, extends the simultaneous aggregation methods, and enables the pruning of the group-by's that do not satisfy the iceberg condition.

Outline

- Introduction
- **Overview of cubing algorithms**
- Start-Cubing: An Integration of top-down and bottom-up approach
- Performance Analysis
- Conclusion

MultiWay

- MultiWay is an array-based top-down cubing algorithm.
- It uses a compressed sparse array structure to load the base cuboid and compute the cube.
- In order to save memory usage, the array structure is partitioned into chunks. It is unnecessary to keep all the chunks in memory since only parts of the group-by arrays are needed at any time.
- By carefully arranging the chunk computation order, multiple cuboids can be computed simultaneously in one pass
- Taking ABCD as the base cuboid, the results of computing cuboid ACD can be used to compute AD, which in turn can be used to compute A.
- This shared computation makes MultiWay perform aggregations simultaneously on multiple dimensions, which leads to the computation order
- Intermediate aggregate values can be re-used for the computation of successive descendant cuboids.

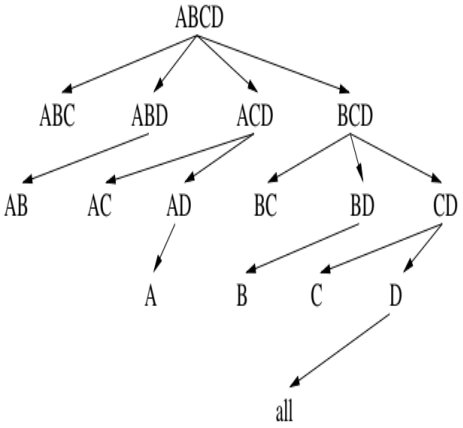


Figure 1: Top-Down Computation

About Multyway

- +ves
  - Good for moderate number of dimensions
- -ves
  - If the dimensionality is high and the data is too sparse, the method becomes infeasible because the arrays and intermediate results become too large to fit in memory.
  - The top-down algorithm cannot take advantage of Apriori pruning during iceberg cubing, i.e., the iceberg condition can only be used after the whole cube is computed.
    - This is because the successive computation does not have the anti-monotonic property .
      - If a cell in ABD does not satisfy min sup, one cannot assert that its children cell" in the cuboid AB does not satisfy min sup either since a cell in AB is likely to contain more base cells than that in ABD.

BUC

- BUC employs a bottom-up computation by starting at the apex cuboid
- BUC starts by reading the first dimension and partitioning it based on its distinct values. Then for each partition in the first dimension, it recursively computes the remaining dimensions.

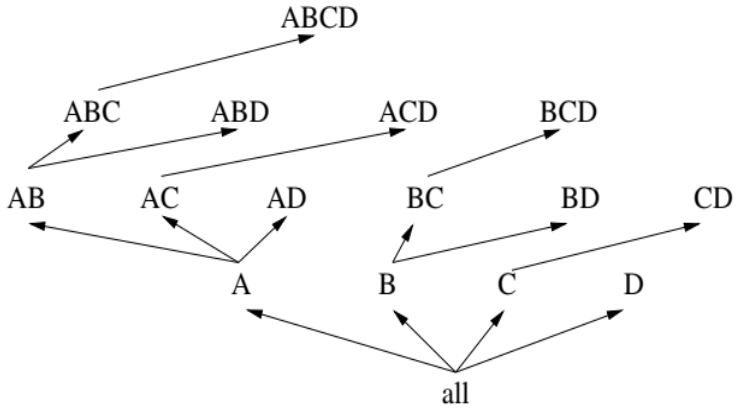


Figure 2: **Bottom-Up Computation**

About BUC

- +ves
  - Apriori pruning is used to reduce unnecessary computation based on the anti-monotonic property, which was not possible in the top-down computation.
    - For example, if the count of a cell c in a cuboid A is smaller than min sup, then the count of any descendant cells of c (with more dimensions, e.g., AC, ACD) can never be higher than min sup. Thus the descendant cells of c can be pruned.
- -ves
  - Partitioning and sorting are the major costs in BUC's cube computation. Since recursive partitioning in BUC does not reduce the input size, both partition and aggregation are costly.
  - Moreover, BUC is sensitive to skew in the data: the performance of BUC degrades as skew increases.
  - Unlike MultiWay, the results of a parent cuboid does not help compute that of its children in BUC.
    - For example, the computation of cuboid AB does not help that of ABC. The latter needs to be computed essentially from scratch.

H-Cubing

- H-Cubing uses a hyper-tree structure, called HTree, to facilitate cube computation.
  - Each level in the tree represents a dimension in the base cuboid. A base tuple (cell) of d-dimensions forms one path of length d (i.e., d nodes) in the tree.
- Nodes at the same level of tree that hold the same value are linked together via side-links.
- In addition, a Header Table is associated with each level to record the count of every distinct value in all the dimensions not below the current one, and provides links to the first node of the corresponding values in the H-Tree.
- With this data structure, there are two methods to compute the cube: bottom-up (BOT) vs. top-down (TOP) tree traversal.
- In both methods, the algorithm starts at a particular level of the H-Tree, i.e., a particular dimension, and examines the group-by's that include that level and levels above it in the H-Tree.
- The aggregation is facilitated by the Header Table constructed in the initial pass of the data and Header Tables local to the current group-by.
- During the aggregation, if the count of a particular node is below min sup, it skips itself and jumps to the next node via the side-link.
- The difference between the two traversal methods, H-Cubing-BOT and H-Cubing-TOP, is that the former starts the process at the bottom of the HTree, whereas the latter starts at the top

About H-cube

- +ves
  - One advantage of H-Cubing is that since the internal nodes of the H-Tree structure collapse duplicated data, shared processing and some simultaneous aggregation can be explored.
  - Also, it computes less dimension combinations before proceedings to more dimensions, and thus leads to Apriori pruning for iceberg cube computation.
- -ves
  - However, similar to BUC, it cannot use the intermediate results at computing low dimensions to facilitate the computation of high dimensional cuboids.

Outline

- Introduction
- Overview of cubing algorithms
- **Start-Cubing: An Integration of top-down and bottom-up approach**
- Performance Analysis
- Conclusion

Algorithm	Simultaneous Aggregation	Partition & Prune
MultiWay	Yes	No
BUC	No	Yes
H-Cubing	Weak	Yes
Star-Cubing	Yes	Yes

Table 1: **Summary of Four Algorithms**

Start Cubing

- The Star-Cubing algorithm explores both the topdown and bottom-up models:
- On the global computation order, it uses the top-down model similar to Figure 1.
- However, it has a sub-layer underneath based on the bottom-up model by exploring the notion of shared dimension.
- Main concepts
  - Shared dimensions
  - Star Nodes and Star Trees
  - Cuboid Trees

Shared Dimensions

- Observe Figure 1
  - all the cuboids in the left-most sub-tree of the root include dimensions ABC,
  - all those in the second sub-tree include dimensions AB,
  - and all those in the third include dimension A.
- We call these common dimensions the shared dimensions of those particular sub-trees.
- Based on this concept, Figure 1 is extended to Figure 3, which shows the spanning tree marked with the shared dimensions.
  - For example, ABD/AB means cuboid ABD has shared dimension AB, ACD/A means cuboids ACD has shared dimension A, and so on

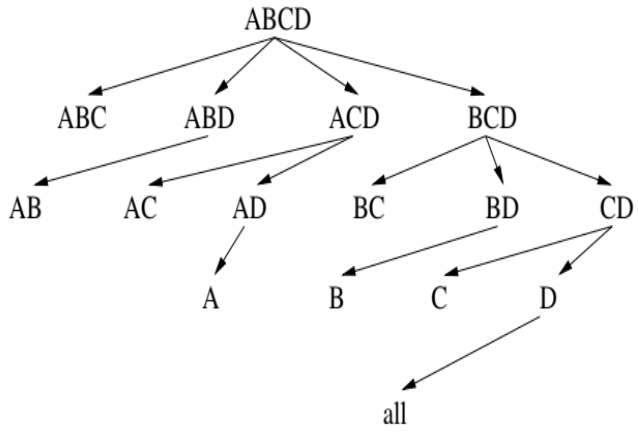


Figure 1: **Top-Down Computation**

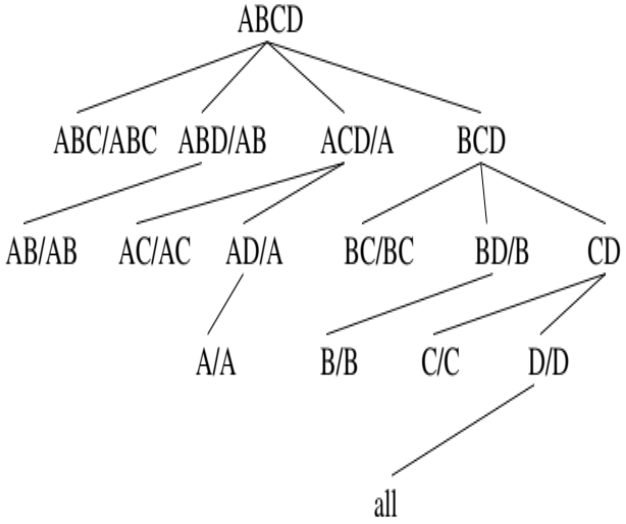


Figure 3: **Star-Cubing: Top-down computation with bottom-up growing shared dimensions**

Apriori Pruning

- 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
  - As an example, if the value in the shared dimension A is a1 and it fails to satisfy the iceberg condition, the whole sub-tree rooted at a1CD/a1 (including a1C/a1C, a1D/a1, a1/a1) can be pruned

- ***This is nothing but the Exploitation of BUC***

Cuboid Trees

- We use trees to represent individual cuboids.
- Figure 4 shows a fragment of the cuboid tree of the base cuboid ABCD
- Each node has four flelds:
  - the attribute value, aggregate value, pointer(s) to possible descendant(s), and pointer to possible sibling.
  - Tuples in the cuboid are inserted one by one into the tree. A path from the root to a node represents a tuple. For example, node c2 in the tree has aggregate (count) value of 5, which indicates that there are flve cells of value {a1 b1 c2 *}

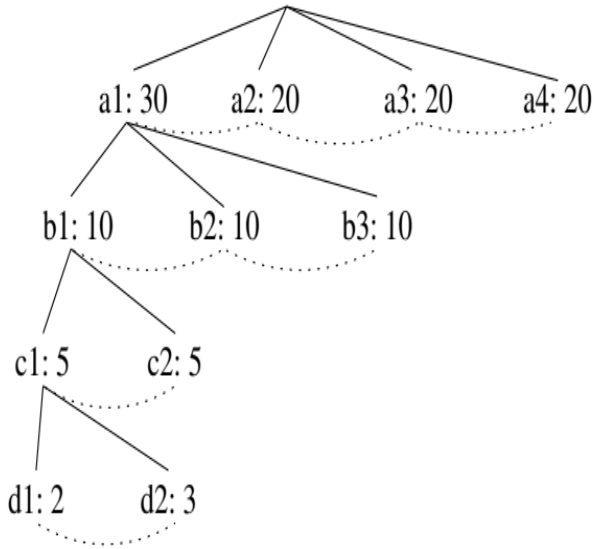


Figure 4: A fragment of the base cuboid tree

- ***This is nothing but the Exploiting of H-cube***

Star Nodes and Star Trees

- If the single dimensional aggregate on an attribute value p does not satisfy the iceberg condition, it is useless to distinguish such nodes in the iceberg cube computation.
- Thus the node p can be replaced by * so that the cuboid tree can be further compressed.
- This motivates us to introduce the concepts of star node and star tree.
- The node p in an attribute A is a star node if the single dimensional aggregate on p does not satisfy the iceberg condition; otherwise, p is a non-star node.
- A cuboid tree that consists of only non-star nodes and star(-replaced) nodes is called a star-tree.

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

Table 2: **Base (Cuboid) Table**: Before star reductior

Dimension	Count $= 1$	Count ≥ 2
A	-	$a1(3), a2(2)$
B	$b2, b3, b4$	$b1(2)$
C	$c1, c2$	$c3(3)$
D	$d1, d2, d3$	$d4(2)$

Table 3: **One-Dimensional Aggregates**

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

Table 4: **Compressed Base Table**: After star reduction.

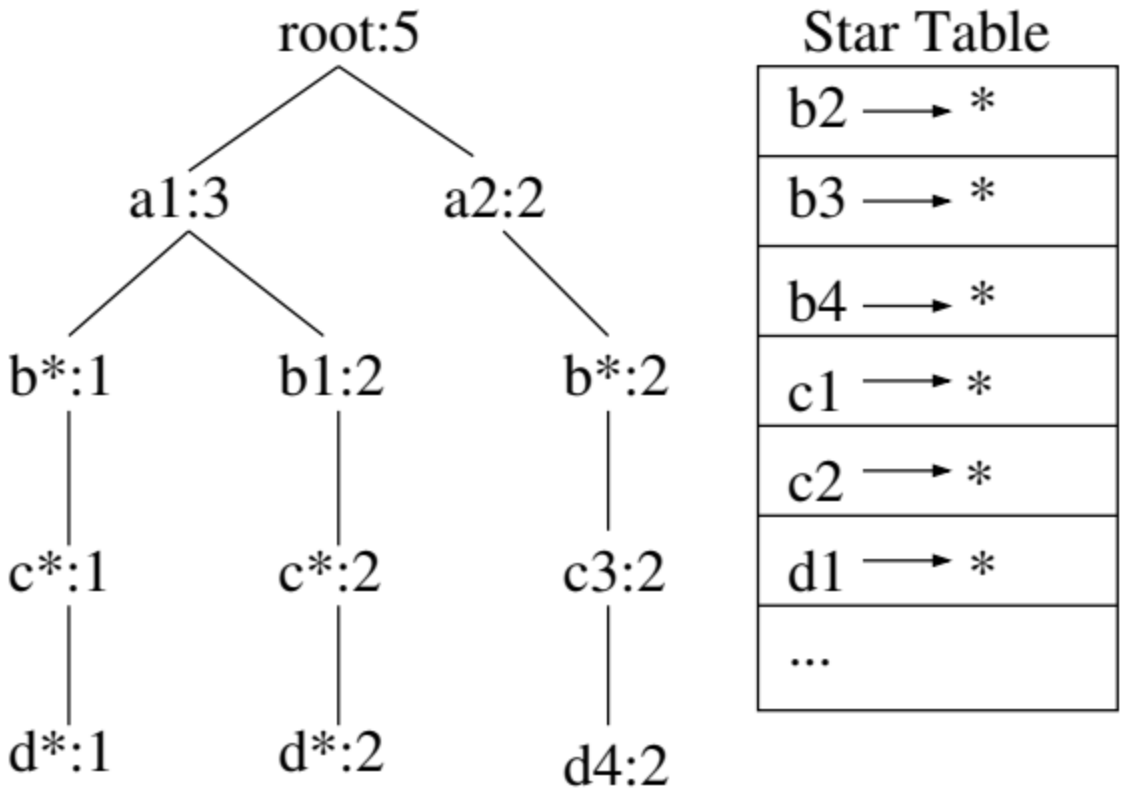


Figure 5: **Star Tree and Star Table**

Performance

- See the paper

Summary of Star CUBE

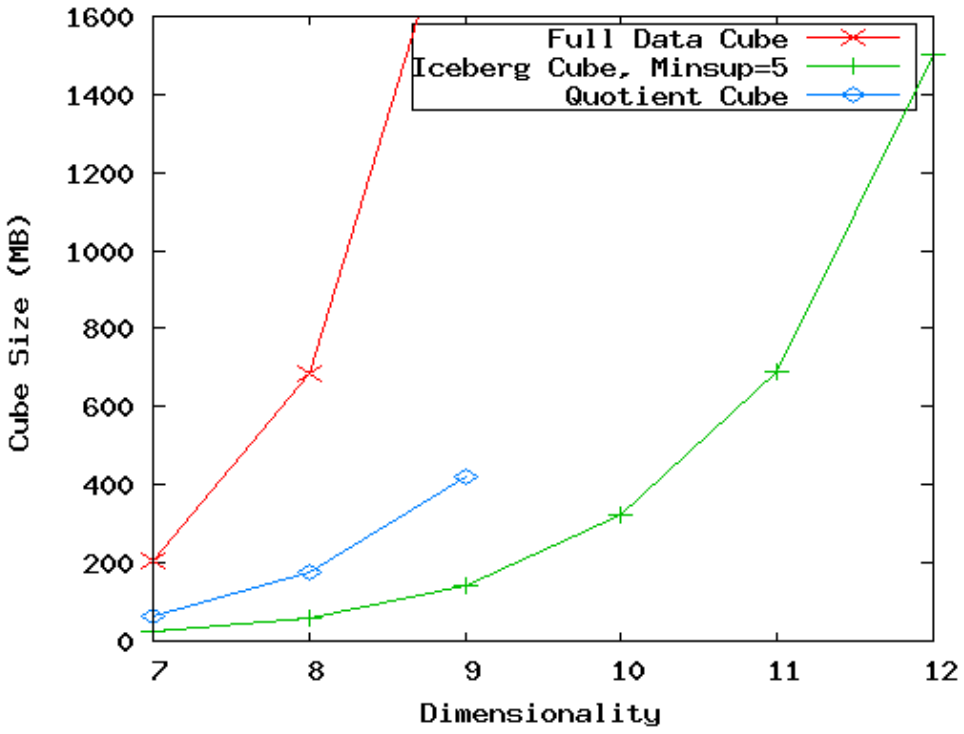
- An interesting cube computation method, Star-Cubing, is proposed
- Integrates the strength of both top-down and bottom-up cube computation
- Exploits
  - shared aggregation by taking advantage of shared dimensions among the current cuboid and its descendant cuboids; and
  - prune as soon as possible the unpromising cells during the cube computation using the anti-monotonic property of the iceberg cube measure.
- Moreover, a new compressed data structure, star-tree is proposed using star nodes. And a few other optimization techniques also contribute to the high performance of the method

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - 3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4. Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
    - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
    - **6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)**
- Overview of Data Cubes for Advanced Applications
  - 7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

The Curse of Dimensionality

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



Motivation of High-D OLAP

- X. Li, J. Han, and H. Gonzalez, High-Dimensional OLAP: A Minimal Cubing Approach, VLDB'04
- Challenge to current cubing methods:
  - The “curse of dimensionality” problem
  - Iceberg cube and compressed cubes: only delay the inevitable explosion
  - Full materialization: still significant overhead in accessing results on disk
- High-D OLAP is needed in applications
  - Science and engineering analysis
  - Bio-data analysis: thousands of genes
  - Statistical surveys: hundreds of variables

Fast High-D OLAP with Minimal Cubing

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

Properties of Proposed Method

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

Example Computation

- Let the cube aggregation function be `count`

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

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

1-D Inverted Indices

- Build traditional invert index or RID list

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

Shell Fragment Cubes: Ideas

- Generalize the 1-D inverted indices to multi-dimensional ones in the data cube sense
- Compute all cuboids for data cubes ABC and DE while retaining the inverted indices
- For example, shell fragment cube ABC contains 7 cuboids:
  - A, B, C
  - AB, AC, BC
  - ABC
- This completes the offline computation stage

Cell	Intersection	TID List	List Size
a1 b1	1 2 3 ∩1 4 5	1	1
a1 b2	1 2 3 ∩2 3	2 3	2
a2 b1	4 5 ∩1 4 5	4 5	2
a2 b2	4 5 ∩2 3	⊗	0

Shell Fragment Cubes: Size and Design

- Given a database of T tuples, D dimensions, and F shell fragment size, the fragment cubes' space requirement is:
  - For F < 5, the growth is sub-linear $O\left(T\left\lceil\frac{D}{F}\right\rceil(2^F-1)\right)$
- Shell fragments do not have to be disjoint
- Fragment groupings can be arbitrary to allow for maximum online performance
  - Known common combinations (e.g.,<city, state>) should be grouped together.
- Shell fragment sizes can be adjusted for optimal balance between offline and online computation

ID_Measure Table

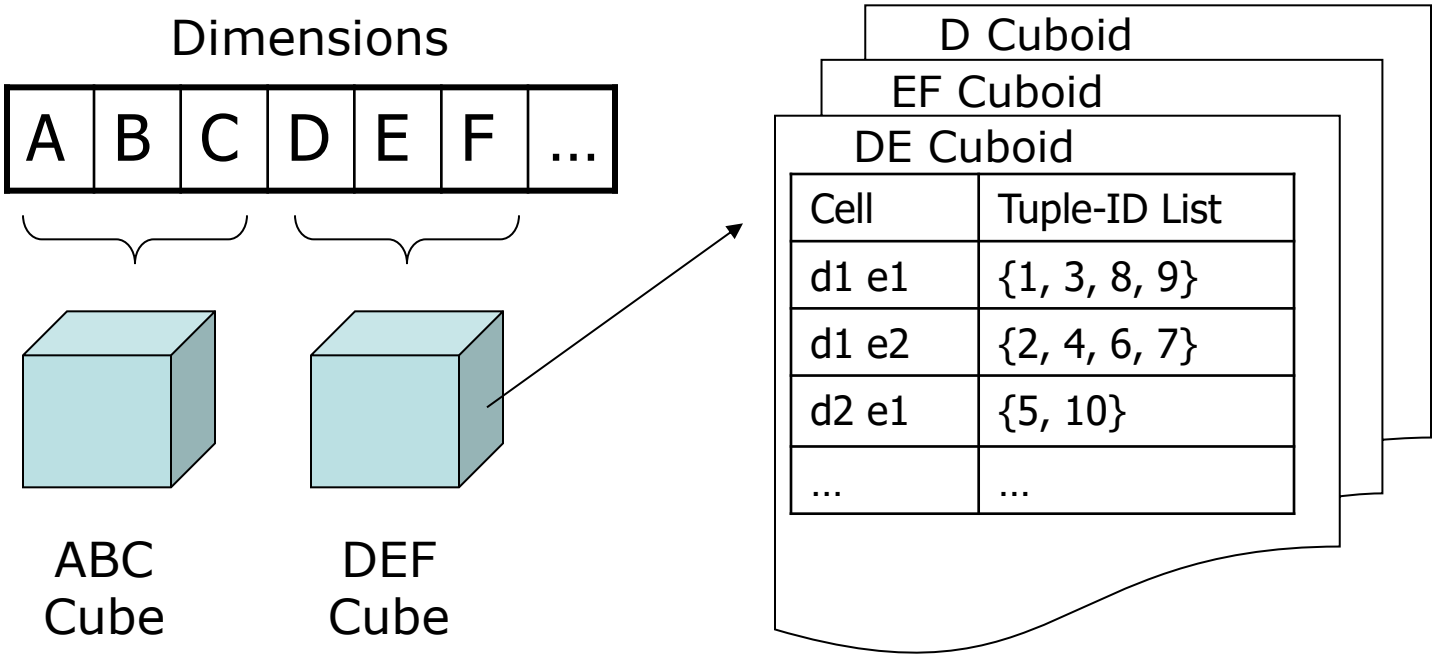
- If measures other than `count` are present, store in *ID_measure* table separate from the shell fragments

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

The Frag-Shells Algorithm

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

Frag-Shells (2)



Online Query Computation: Query

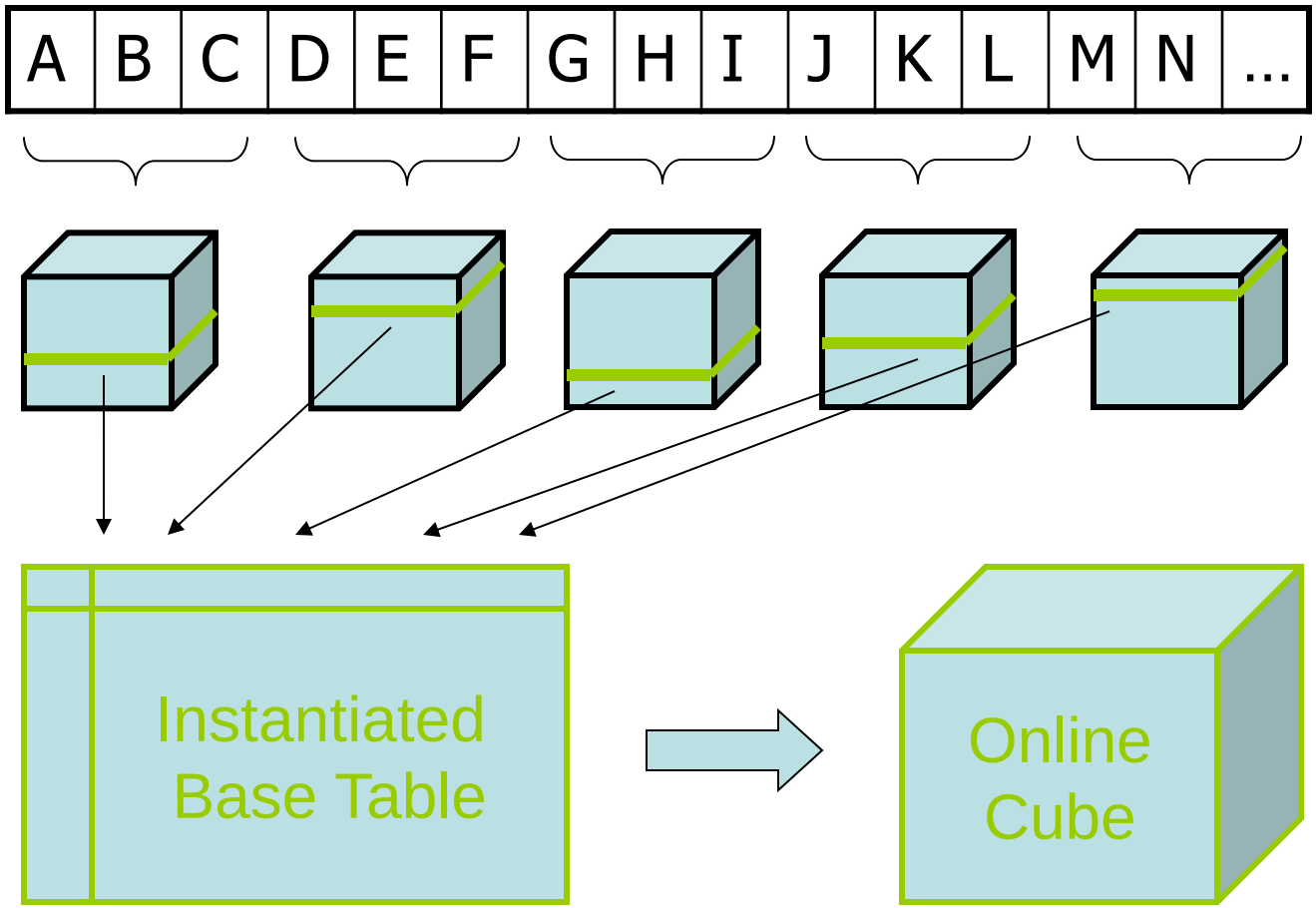
- A query has the general form $\langle a_1, a_2, \ldots, a_n : M \rangle$
- Each a_i has 3 possible values
  1. Instantiated value
  2. Aggregate * function
  3. Inquire ? function

■ For example, $\langle 3 \quad ? \quad ? \quad * \quad 1 : count \rangle$ returns a 2-D data cube.

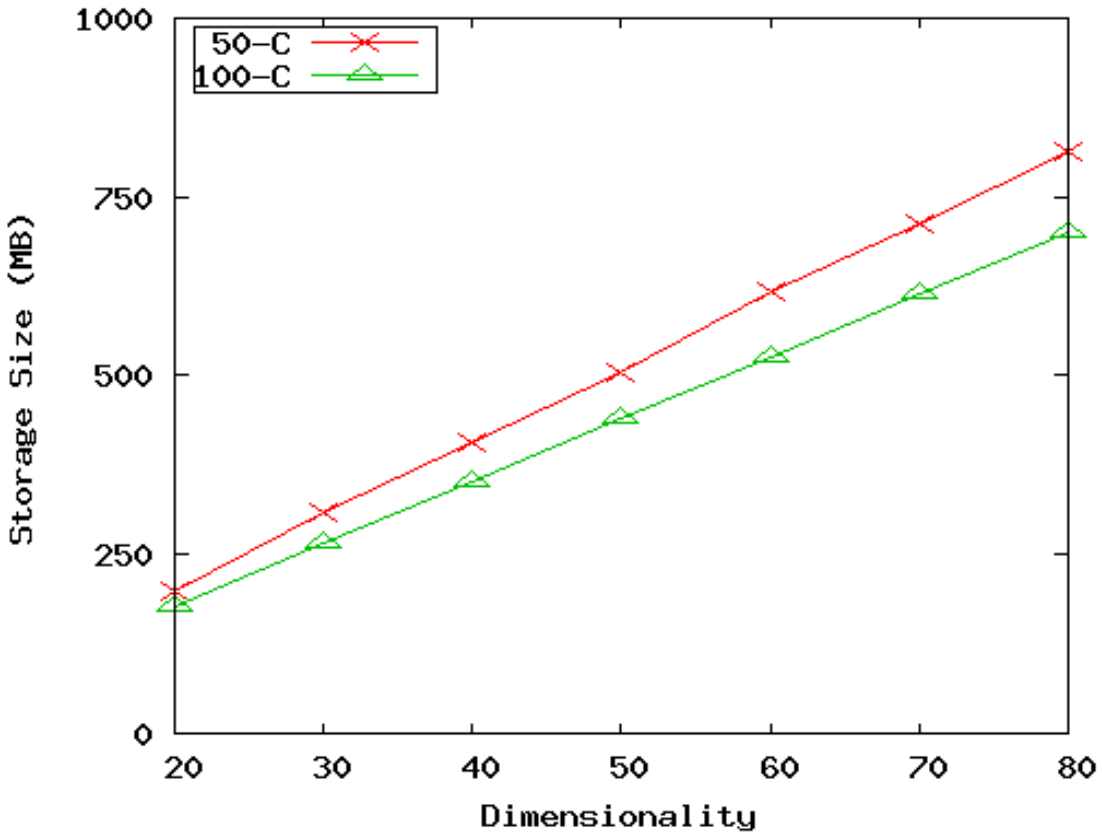
Online Query Computation: Method

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

Online Query Computation: Sketch



Experiment: Size vs. Dimensionality (50 and 100 cardinality)



- (50-C): 10^6 tuples, 0 skew, 50 cardinality, fragment size 3.
- (100-C): 10^6 tuples, 2 skew, 100 cardinality, fragment size 2.

Experiments on Real World Data

- UCI Forest CoverType data set
  - 54 dimensions, 581K tuples
  - Shell fragments of size 2 took 33 seconds and 325MB to compute
  - 3-D subquery with 1 instantiate D: 85ms~1.4 sec.
- Longitudinal Study of Vocational Rehab. Data
  - 24 dimensions, 8818 tuples
  - Shell fragments of size 3 took 0.9 seconds and 60MB to compute
  - 5-D query with 0 instantiated D: 227ms~2.6 sec.

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - 3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4. Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - **7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)**
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

Processing Advanced Queries by Exploring Data Cube Technology

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

Statistical Surveys and OLAP

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

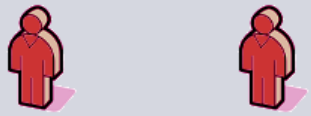
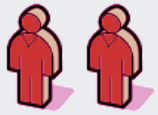
Surveys: Sample vs. Whole Population

Data is only a sample of **population**

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

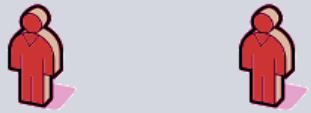
Problems for Drilling in Multidim. Space

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

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

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

- Semantics of query is unchanged
- Input data has changed

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

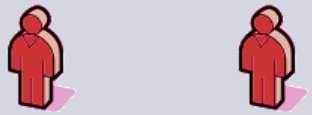
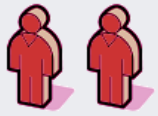
Challenges for OLAP on Sampling Data

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

Example 1: Confidence Interval

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

Return not only query result but also confidence interval

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

Confidence Interval

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

Efficient Computing Confidence Interval Measures

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

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

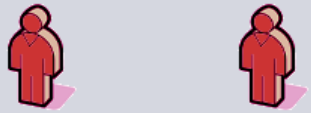
$\bar{x}$ is algebraic

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

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

Example 2: Query Expansion

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

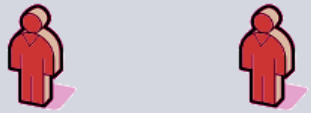
Age/Education	High-school	College	Graduate
18			
19			
20			
...			

Boosting Confidence by Query Expansion

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

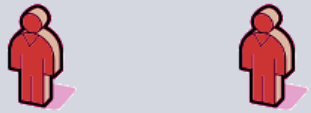
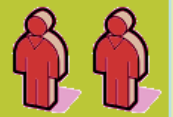
Intra-Cuboid Expansion: Choice 1

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

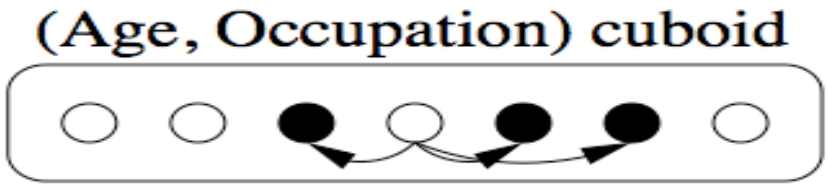
Age/Education	High-school	College	Graduate
18			
19			
20			
...			

Intra-Cuboid Expansion: Choice 2

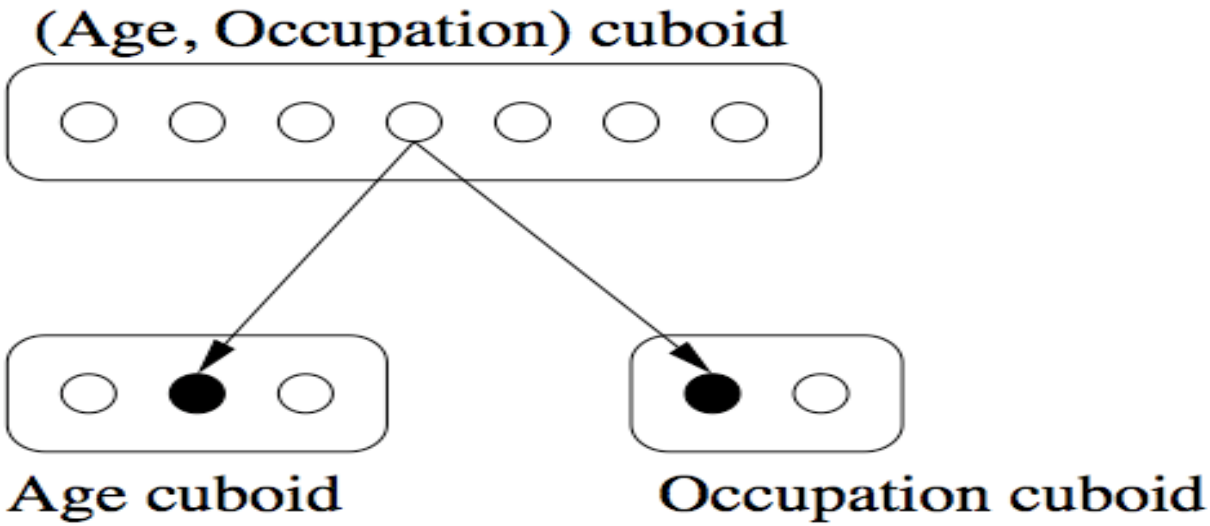
Expand query to include **high-school** and **graduate** students?

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

Query Expansion



(a) Intra-Cuboid Expansion



(b) Inter-Cuboid Expansion

Intra-Cuboid Expansion

- Combine other cells' data into own to "boost" confidence
  - If share semantic and cube similarity
  - Use only if necessary
  - Bigger sample size will decrease confidence interval
- Cell segment similarity
  - Some dimensions are clear: **Age**
  - Some are fuzzy: **Occupation**
  - May need domain knowledge
- Cell value similarity
  - How to determine if two cells' samples come from the same population?
  - Two-sample t-test (confidence-based)

Inter-Cuboid Expansion

- If a query dimension is
  - Not correlated with cube value
  - But is causing small sample size by drilling down too much
- Remove dimension (i.e., generalize to *) and move to a more general cuboid
- Can use two-sample t-test to determine similarity between two cells across cuboids
- Can also use a different method to be shown later

Query Expansion Experiments

- Real world sample data: 600 dimensions and 750,000 tuples
- 0.05% to simulate "sample" (allows error checking)

(a) Intra-Cuboid Expansion with **Age** dimension and **Average Number of Children** cube measure

Query		Average Query Answer Error			Sampling Sizes		
Gender	Marital	No Expand	Expand	% Improve	Population	Sample	Expanded
FEMALE	MARRIED	0.48	0.32	33%	2473.0	2.2	28.3
FEMALE	SINGLE	0.31	0.21	30%	612.6	0.6	6.4
FEMALE	DIVORCED	0.49	0.43	11%	321.1	0.3	3.4
MALE	MARRIED	0.42	0.21	49%	4296.8	4.4	37.6
MALE	SINGLE	0.26	0.21	16%	571.8	0.5	3.6
MALE	DIVORCED	0.33	0.27	19%	224.7	0.2	1.2
	Average	0.38	0.27	26%	1416.7	1.4	13.4

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - 3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4. Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - **7.** Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - **8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)**
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

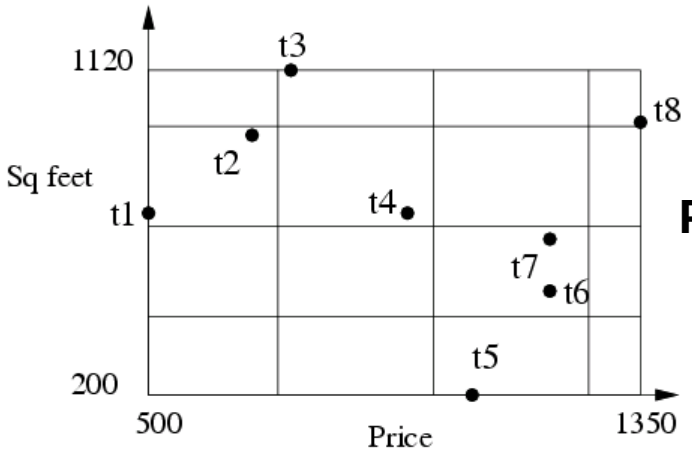
Ranking Cubes – Efficient Computation of Ranking queries

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

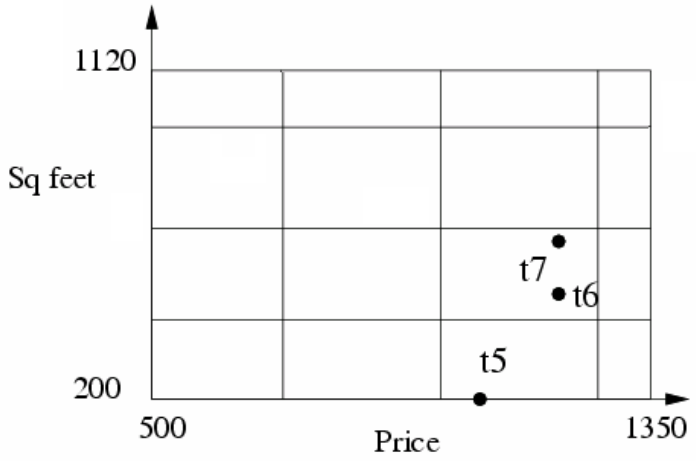
Ranking Cube: Partition Data on Both Selection and Ranking Dimensions

One single data partition as the template

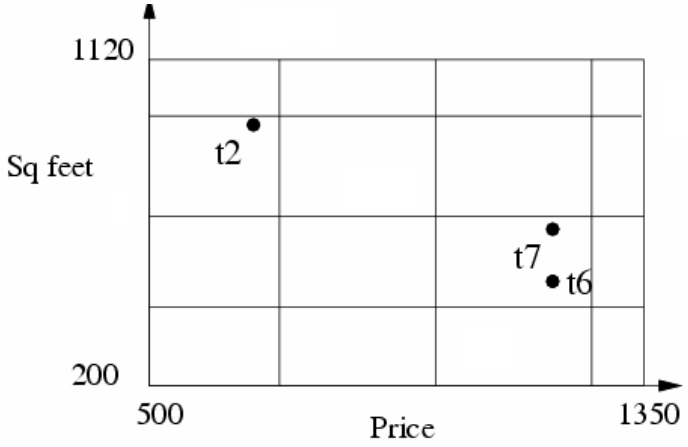
Slice the data partition by selection conditions



Partition for all data



Sliced Partition for city="LA"

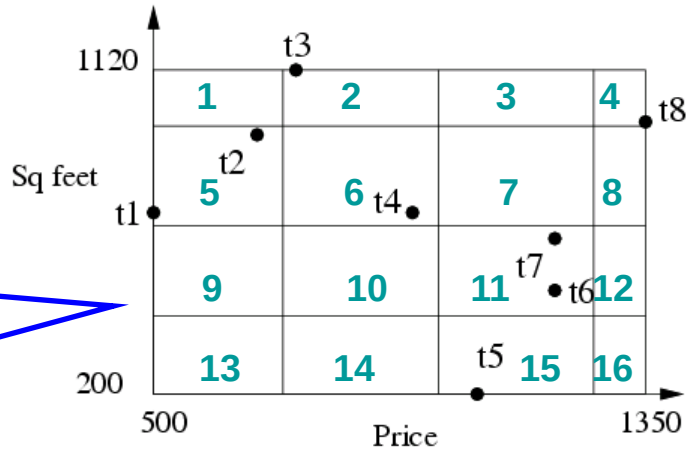


Sliced Partition for BR=2

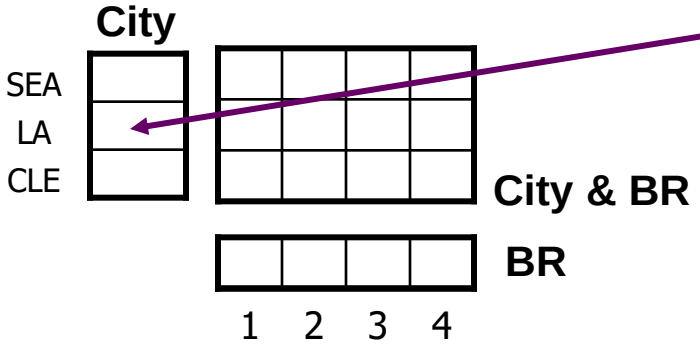
Materialize Ranking-Cube

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

Step 1: Partition Data on Ranking Dimensions



Step 2: Group data by Selection Dimensions

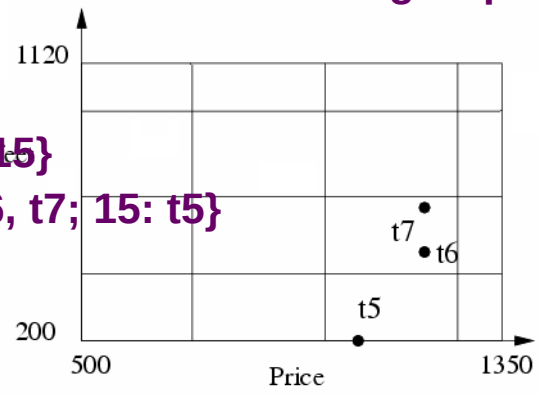


Step 3: Compute Measures for each group

For the cell (LA)

Block-level: {11, 15}

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

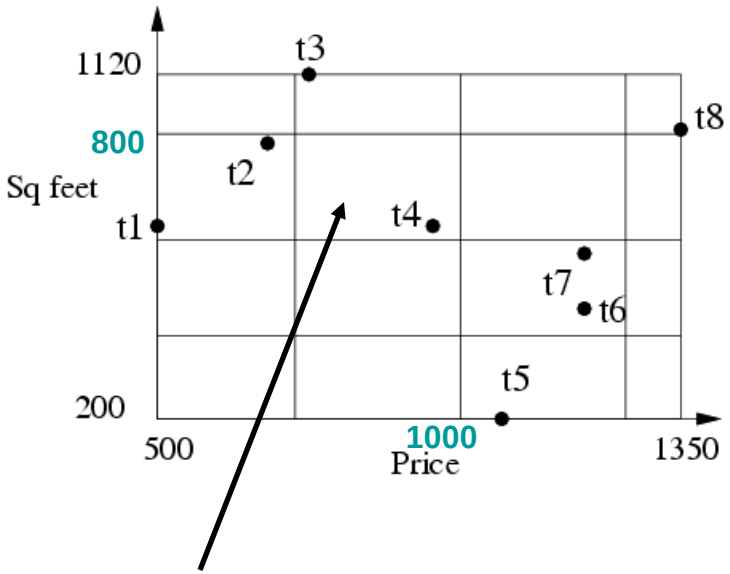


Search with Ranking-Cube: Simultaneously Push Selection and Ranking

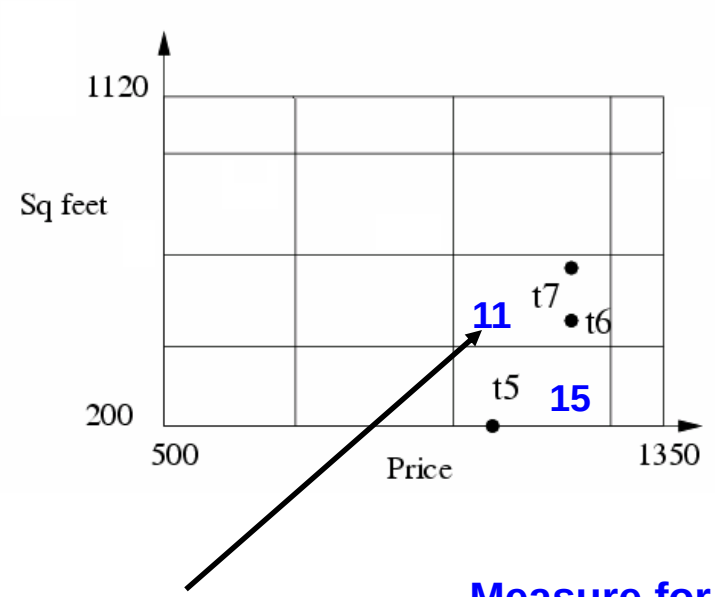
Select top 1 from Apartment
where city = "LA"
order by [price – 1000]^2 + [sq feet - 800]^2 asc

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

Given the bin boundaries, locate the block with top score



Without ranking-cube: start search from here



With ranking-cube: start search from here

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

Ranking Cube: Methodology and Extension

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

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - 3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4. Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - **7.** Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - **8**. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. **Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)**
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

Data Mining in Cube Space

- Data cube greatly increases the analysis bandwidth
- Four ways to interact OLAP-styled analysis and data mining
  - Using cube space to define data space for mining
  - Using OLAP queries to generate features and targets for mining, e.g., multi-feature cube
  - Using data-mining models as building blocks in a multi-step mining process
  - Using data-cube computation techniques to speed up repeated model construction
    - Cube-space data mining may require building a model for each candidate data space
    - Sharing computation across model-construction for different candidates may lead to efficient mining

Prediction Cubes

- **Prediction cube**: A cube structure that stores prediction models in multidimensional data space and supports prediction in OLAP manner

- Prediction models are used as building blocks to define the interestingness of subsets of data, i.e., to answer which subsets of data indicate better prediction

Reference: B.-C. Chen, L. Chen, Y. Lin, and R. Ramakrishnan. Prediction cubes. VLDB'05

How to Determine the Prediction Power of an Attribute?

- Ex. A customer table **D:**
  - Two dimensions **Z**: *Time* (*Month*, *Year*) and *Location* (*State*, *Country*)
  - Two features **X**: *Gender* and *Salary*
  - One class-label attribute Y: *Valued Customer*
- Q: "Are there times and locations in which the value of a customer depended greatly on the customers gender (i.e., Gender: predictiveness attribute **V**)?"
- Idea:
  - Compute the difference between the model built on that using **X** to predict Y and that built on using **X – V** to predict Y
    - X is a set of attributes and V is an attribute
  - If the difference is large, **V** must play an important role at predicting Y

Efficient Computation of Prediction Cubes

- Naïve method: Fully materialize the prediction cube, i.e., exhaustively build models and evaluate them for each cell and for each granularity

- Better approach: Explore score function decomposition that reduces prediction cube computation to data cube computation

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - 3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4. Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - **7.** Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - **8**. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - 9. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**. Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)**
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

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 2010 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 = 2010
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

Reference Paper: Ross, K.A., Srivastava, D., Chatziantoniou, D. (1998). Complex aggregation at multiple granularities. EDBT 1998. Lecture Notes in Computer Science, vol 1377. Springer, Berlin, Heidelberg.

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - 3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4. Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - **7.** Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - **8**. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - 9. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**. Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)**
- Summary

Discovery-Driven Exploration of Data Cubes

- Hypothesis-driven
  - exploration by user, huge search space
- Discovery-driven
  - 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

Sarawagi, S., Agrawal, R., Megiddo, N. (1998). Discovery-driven exploration of OLAP data cubes. EDBT 1998. Lecture Notes in Computer Science, vol 1377.

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%

Summary

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - **1.**General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - **3**. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4**.** Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
    - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
    - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - 7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

Other cubes

- Stream cubes for multi dimensional stream analysis
  - Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang, Multi-Dimensional Regression Analysis of Time-Series Data Streams, VLDB'02
- Map cube for visualizing spatial data
  - Map Cube: A Visualization Tool for Spatial Data Warehouses
- Multimedia cube
  - **MultiMediaMiner: A System Prototype for MultiMedia Data Mining**
- Text Cube for analyzing multi dimensional text databases
  - C. X. Lin, B. Ding, J. Han, F. Zhu, and B. Zhao. Text Cube: Computing IR measures for multidimensional text database analysis. ICDM'08
- Topic cube based on topic modelling
  - D. Zhang, C. Zhai, and J. Han. Topic cube: Topic modeling for OLAP on multi-dimensional text databases. SDM'09
- Flow cube for analysing RFID data
  - **FlowCube: Constructing RFID FlowCubes for Multi-Dimensional Analysis of Commodity Flows, VLDB** *06*

What Is Data Mining?

- Data mining (knowledge discovery from data)
  - Extraction of interesting (non-trivial, implicit, previously unknown and potentially useful) patterns or knowledge from **huge** amount of data
- Data cube provides opportunity to extract interesting summarization knowledge from multi-dimensional data
- Potential opportunity
  - for research for new technologies

Final Word

Ref.(I) Data Cube Computation Methods

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

Ref. (II) Advanced Applications with Data Cubes

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

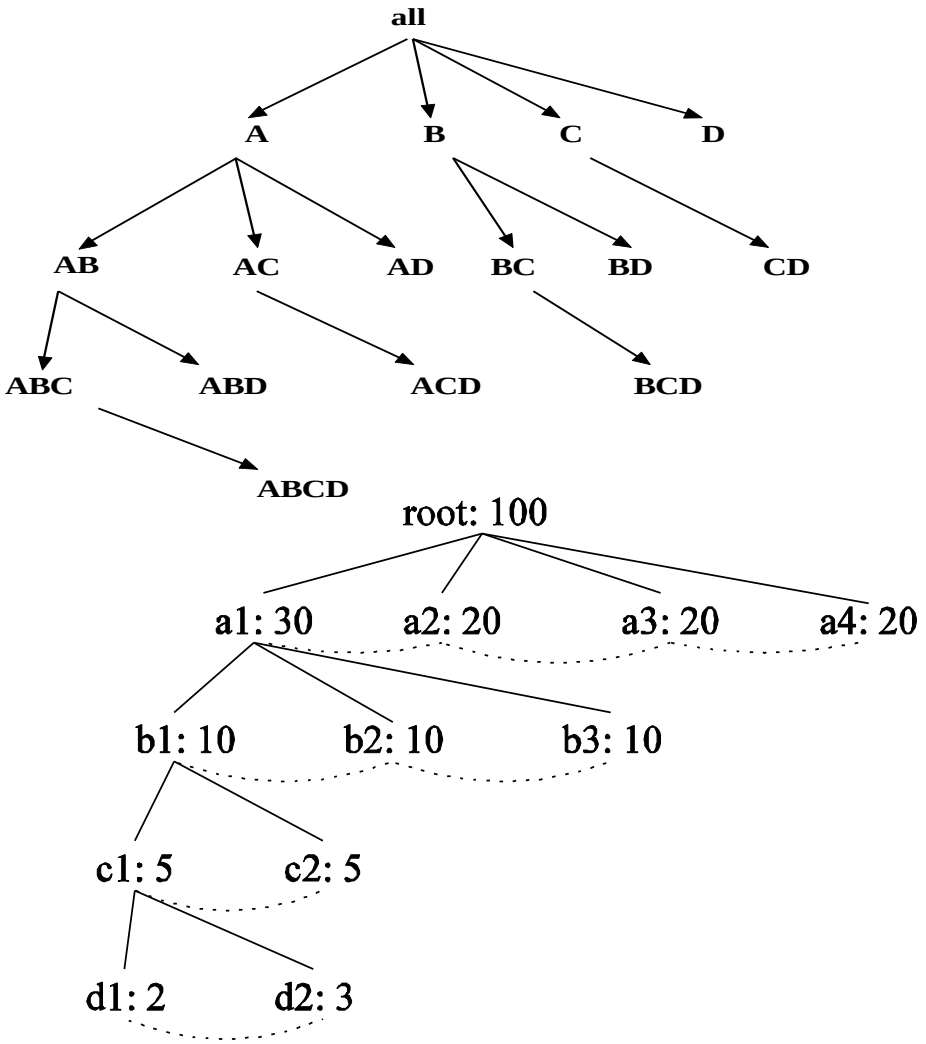
Ref. (III) Knowledge Discovery with Data Cubes

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

Surplus Slides

H-Cubing: Using H-Tree Structure

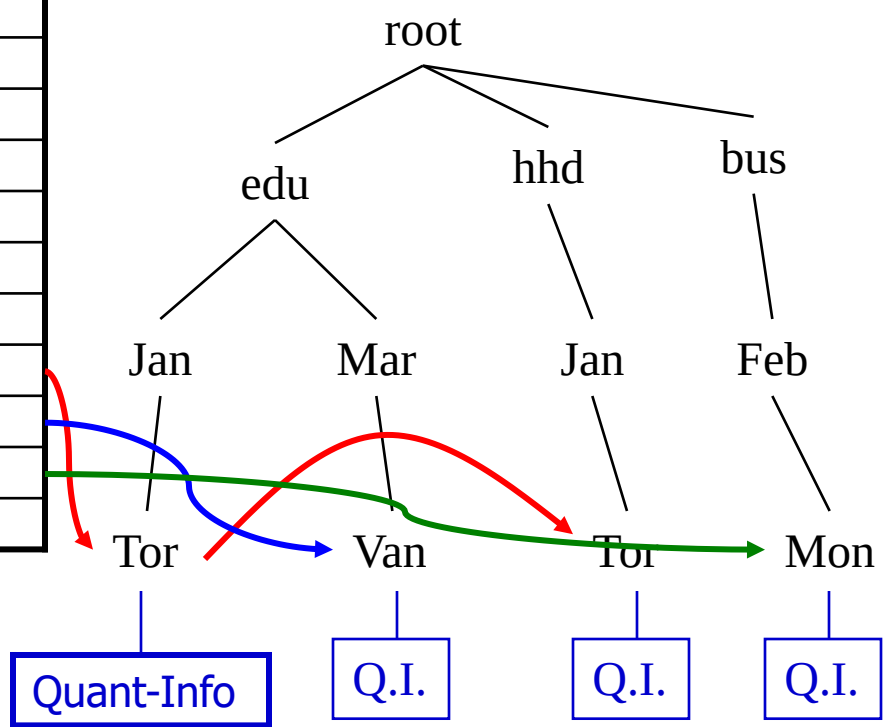
- Bottom-up computation
- Exploring an H-tree structure
- If the current computation of an H-tree cannot pass min_sup, do not proceed further (pruning)
- No simultaneous aggregation



H-tree: A Prefix Hyper-tree

Header table

Attr. Val.	Quant-Info	Side-link
Edu	Sum:2285 …	
Hhd	…	
Bus	…	
…	…	
Jan	…	
Feb	…	
…	…	
Tor	**…**	
Van	**…**	
Mon	**…**	
…	…	



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

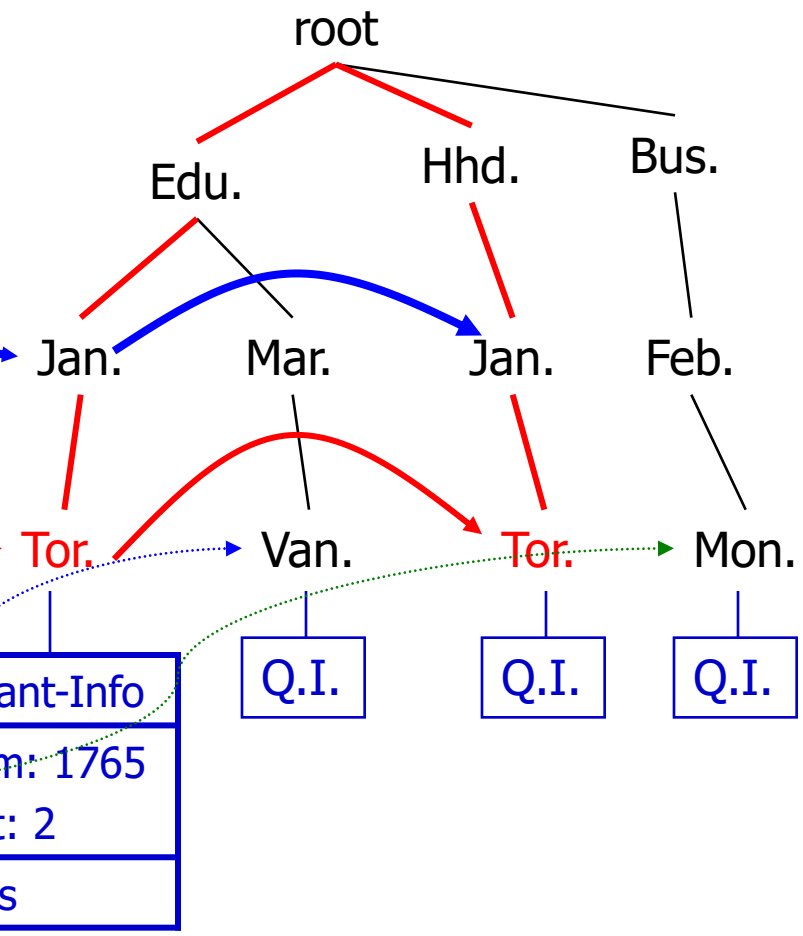
Computing Cells Involving "City"

Header Table H_{Tor}

Attr. Val.	Q.I.	Side-link
Edu	…	
Hhd	…	
Bus	…	
…	…	
Jan	**…**	
Feb	…	
…	…	

From (*, *, Tor) to (*, Jan, Tor)

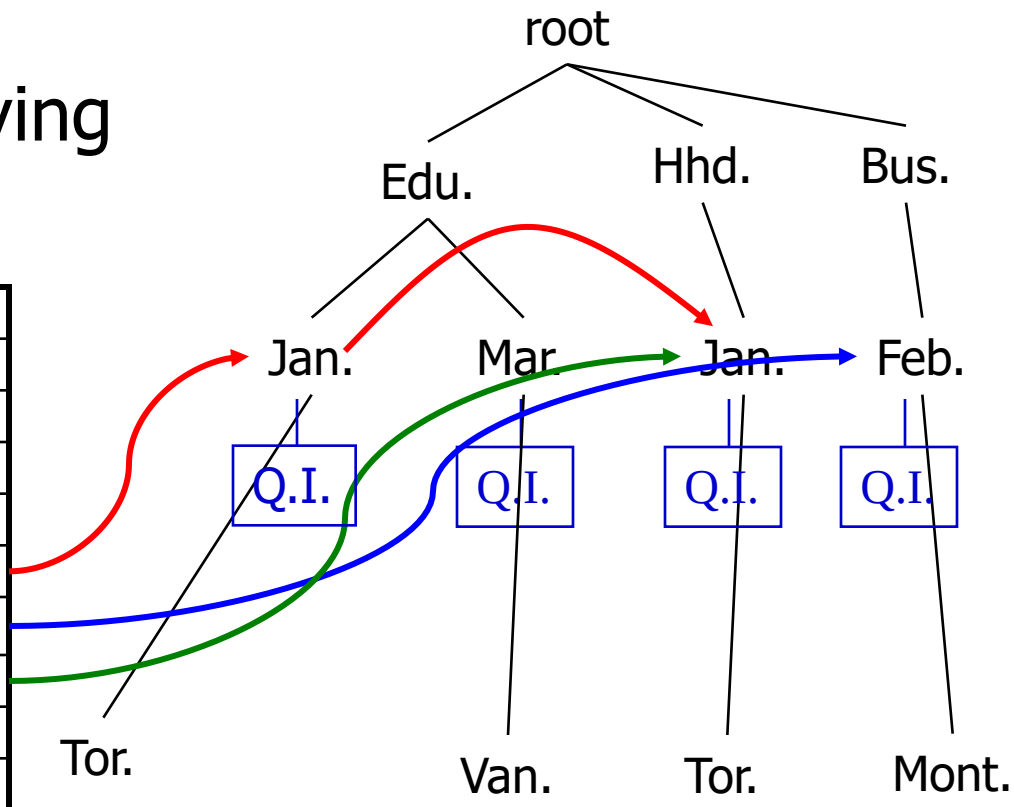
Attr. Val.	Quant-Info	Side-link
Edu	Sum:2285 …	
Hhd	…	
Bus	…	
…	…	
Jan	…	
Feb	…	
…	…	
Tor	**…**	
Van	…	
Mon	…	
…	…	



Computing Cells Involving Month But No City

1. Roll up quant-info
2. Compute cells involving month but no city

Attr. Val.	Quant-Info	Side-link
Edu.	Sum:2285 …	
Hhd.	…	
Bus.	…	
…	…	
Jan.	**…**	
Feb.	**…**	
Mar.	**…**	
…	…	
Tor.	…	
Van.	…	
Mont.	…	
…	…	

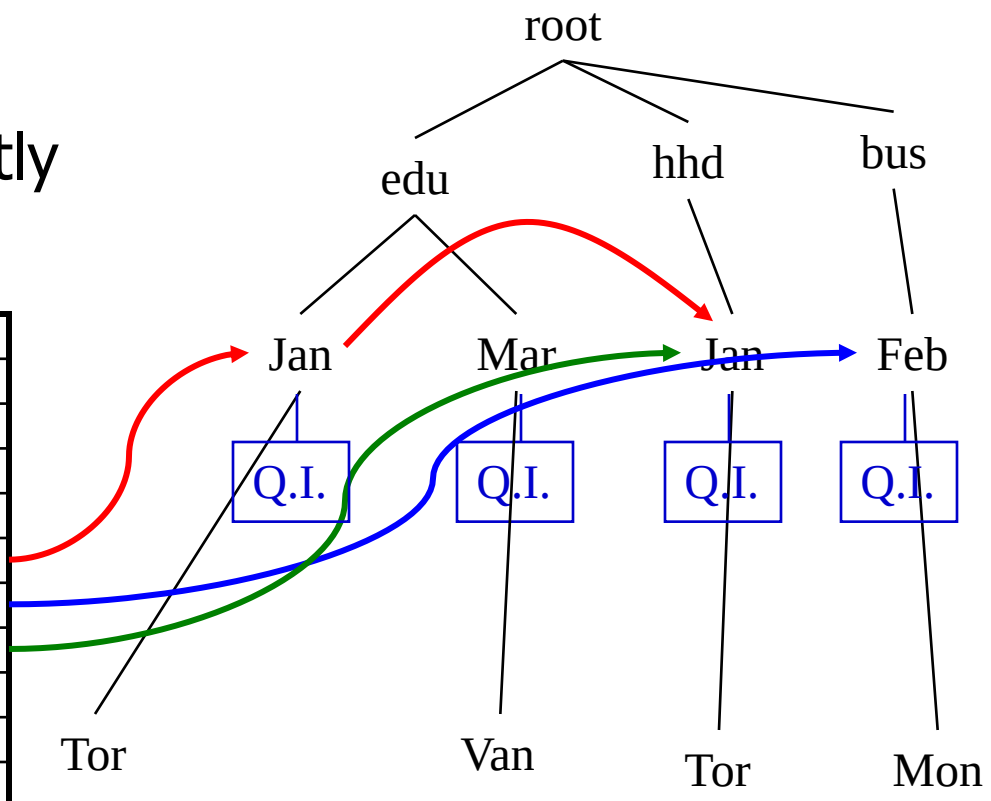


Top-k OK mark: if Q.I. in a child passes top-k avg threshold, so does its parents. No binning is needed!

Computing Cells Involving Only Cust_grp

Check header table directly

Attr. Val.	Quant-Info	Side-link
Edu	Sum:2285 …	
Hhd	…	
Bus	…	
…	…	
Jan	…	
Feb	…	
Mar	…	
…	…	
Tor	…	
Van	…	
Mon	…	
…	…	



Concepts and algorithms for mining patterns and associations (Class #9)

P.Krishna Reddy

IIIT, Hyderabad

Detailed Syllabus

- Introduction (1.5 (3) hour): Definition, KDD framework, Issues in data mining.
- Preprocessing and Data summarization (7.5 (7.5) hrs): Data Types, Preprocessing, Characterization, Discrimination, data warehousing techniques (Data warehousing technology, Data cube computation**)**
- **Concepts and algorithms for mining patterns and associations (9 hours) (Frequent item-set generation, A priori and FP-growth algorithm, Evaluation of Association patterns)**
- Concepts and algorithms related to classification and regression (9hrs) (Overview, Decision tree induction, Over-fitting and under-fitting, Scalable decision tree algorithms, Bayesian Classification, Regression-based Prediction methods (9 hours)
- Concepts and algorithms for clustering the data (9 hours) (Overview, Types of Data, K-means, Aglomerative clustering, Clustering algorithms (DBSCAN, BIRCH, CURE, ROCK, CHAMELEON)).
- Outlier analysis and future trends (graph mining, spatio-temporal mining). (3 hours)

Presentation Outline

- Basic concepts
  - Frequent itemssets, Closed Itemsets, and Association Rules
- Frequent itemset mining methods
  - Apriori Algorithm
    - Generating Association Rules from Frequent Itemsets
    - Improving the Efficiency of Apriori
  - A Pattern-Growth Approach
  - Mining Frequent Itemsets Using the Vertical Data Format
  - Mining Closed and Max Patterns
- Which Patterns are interesting? Pattern Evaluation Methods
- Summary

What Is Frequent Pattern Analysis?

- Use case: You are talking to a customer who recently bought a PC and a digital camera from the store. What should you recommend to her next?
  - Information about which products are frequently purchased by your customers following their purchases of a PC and a digital camera in sequence would be very helpful
  - Frequent patterns and association rules are the knowledge that you want to mine in such a scenario.

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
  - First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining

- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?

- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

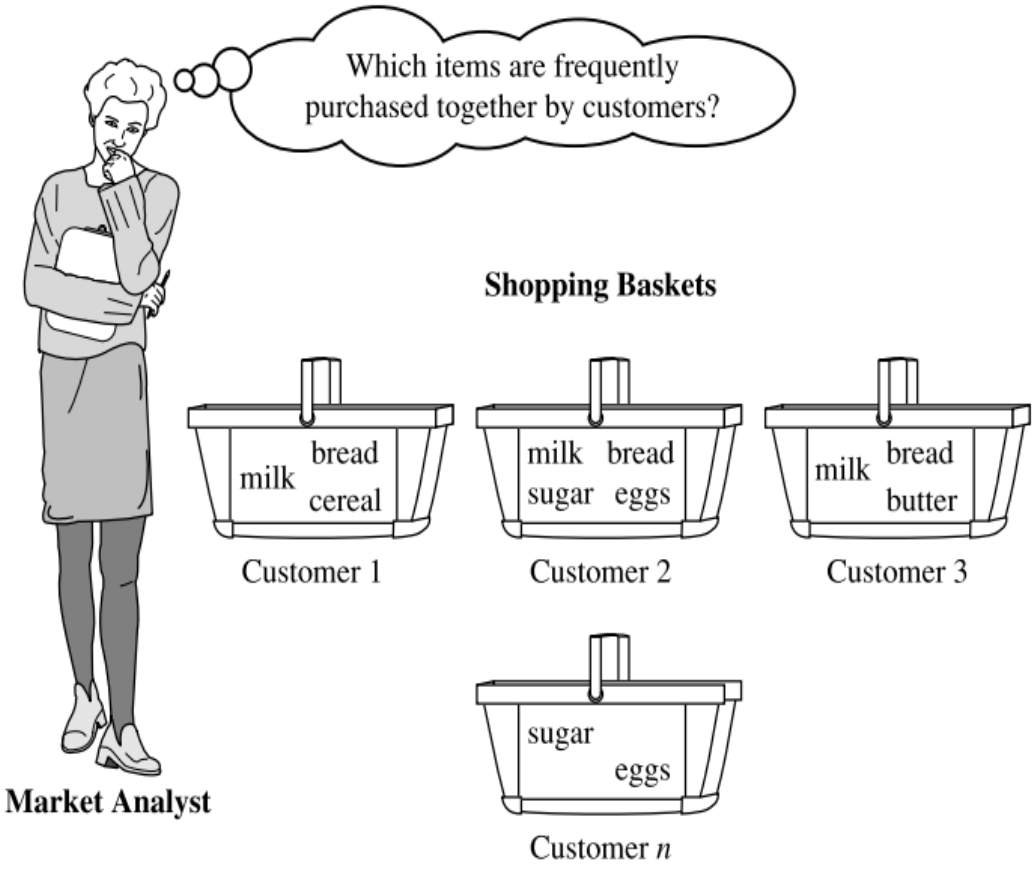


FIGURE 4.1

Market basket analysis.

Why Is Freq. Pattern Mining Important?

- Discloses an intrinsic and important property of data sets
- Forms the foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: associative classification
  - Cluster analysis: frequent pattern-based clustering
  - Data warehousing: iceberg cube and cube-gradient
  - Semantic data compression: fascicles
  - Broad applications

What Is Association Mining?

- Association rule mining:
  - Association rule mining searches for interesting relationships among items in a given data set.
  - *Example: 98% of people who purchase tires and auto accessories also get automotive services done*
- Applications:
  - Basket data analysis, sales, catalog design, storage layout, customer segmentation based on buying pattern.

Association Rule: Basic Concepts

- Given: (1) database of transactions, (2) each transaction is a list of items (purchased by a customer in a visit)
- Find: all rules that correlate the presence of one set of items with that of another set of items
  - E.g., *98% of people who purchase tires and auto accessories also get automotive services done*
- Applications
  - * $\Rightarrow$ *Maintenance Agreement* (What the store should do to boost Maintenance Agreement sales)
  - *Home Electronics* $\Rightarrow$ * (What other products should the store stocks up?)
  - Attached mailing in direct marketing

Formal Definition

- Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items. Let D be a set of transactions, where each transaction T is a set of items (itemsets).
- An association rule is an implication of the form $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \varnothing$.
- X is called antecedent, Y is called consequence.
- The rule $X \rightarrow Y$ has support *s* if *s*% of transactions in D contains $X \cup Y$.
  - support, *s*, probability that a transaction contains X U Y= % of transactions that contain X U Y

$$\textbf{Support}(X \rightarrow Y) = \frac{\textbf{Number of transactions that contain } X \cup Y}{\textbf{Total number of transactions in D}}$$

- The rule $X \rightarrow Y$ has confidence *c* if *c*% of transactions in D that contains X also contains Y.
  - confidence, *c*, conditional probability that a transaction having X also contains Y

$$\textbf{Confidence}\ (X \rightarrow Y) = \frac{\textbf{Support}\ (X \cup Y)}{\textbf{Support}\ (X)}$$

- Support indicates the frequencies of the occurring patterns while confidence measures the rule's strength.

Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Association Rules

{Diaper} → {Beer},
{Milk, Bread} → {Eggs,Coke},
{Beer, Bread} → {Milk},

Implication means co-occurrence, not causality!

Definition: Frequent Itemset

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - E.g. σ({Milk, Bread,Diaper}) = 2
- **Support**
  - Fraction of transactions that contain an itemset
  - E.g. s({Milk, Bread, Diaper}) = 2/5
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a *minsup* threshold

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Association Rule: Example

- **Association Rule**
  - An implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
  - Example:
    {Milk, Diaper} → {Beer}

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

- **Rule Evaluation Metrics**
  - Support (s)
    - Fraction of transactions that contain both X and Y
  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Confidence of a rule

- The confidence of a rule provides an accurate prediction on the association of the items in the rule.
- The support of a rule indicates how frequent the rule is in the transactions.
- Rules that have small support are uninteresting, since they do not describe large populations.

Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support $\geq$ *minsup* threshold
  - confidence $\geq$ *minconf* threshold

- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds

  $\Rightarrow$ Computationally prohibitive!

Mining Association Rules

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

Observations:

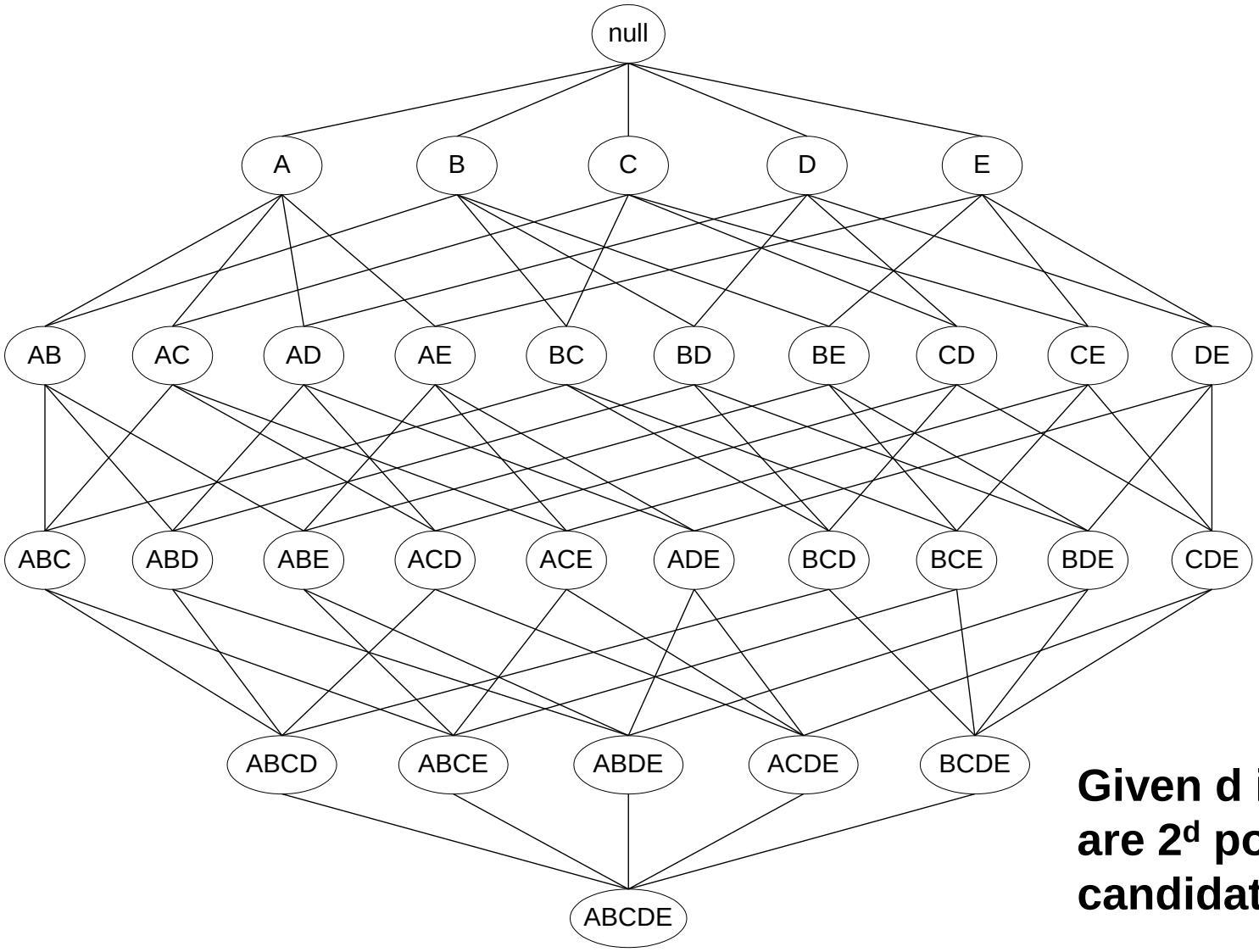
- All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

Mining Association Rules

- Two-step approach:
  1. Frequent Itemset Generation
     - Generate all itemsets whose support ≥ minsup
  2. Rule Generation
     - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

- Frequent itemset generation is still computationally expensive

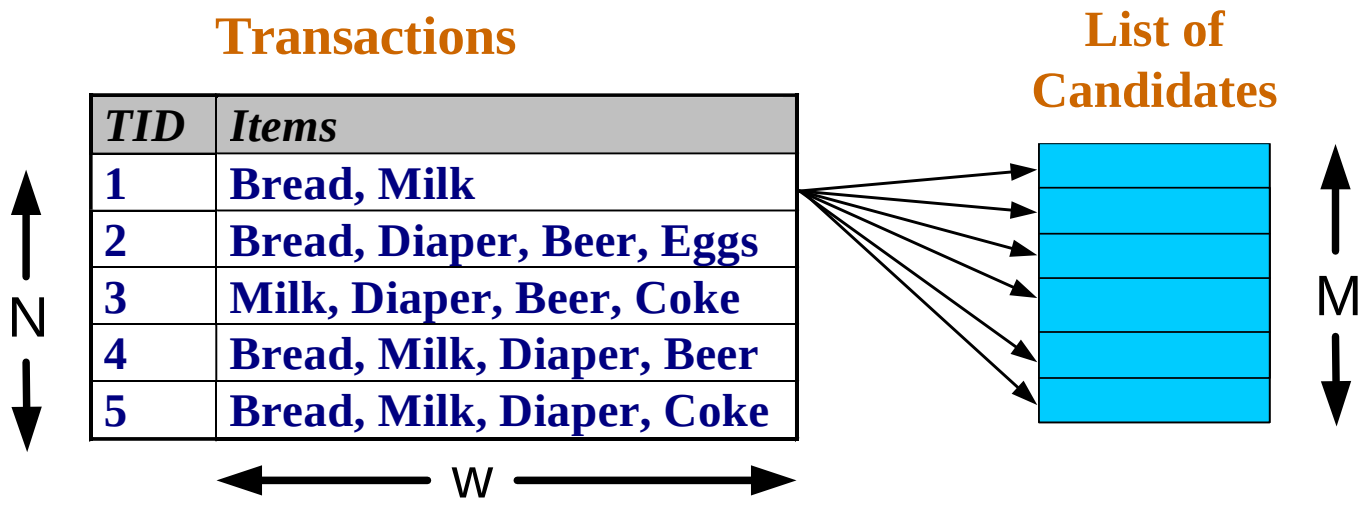
Frequent Itemset Generation



Given d items, there are 2^d possible candidate itemsets

Frequent Itemset Generation

- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database



  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => Expensive since $M = 2^d$!!!

Reducing Number of Candidates

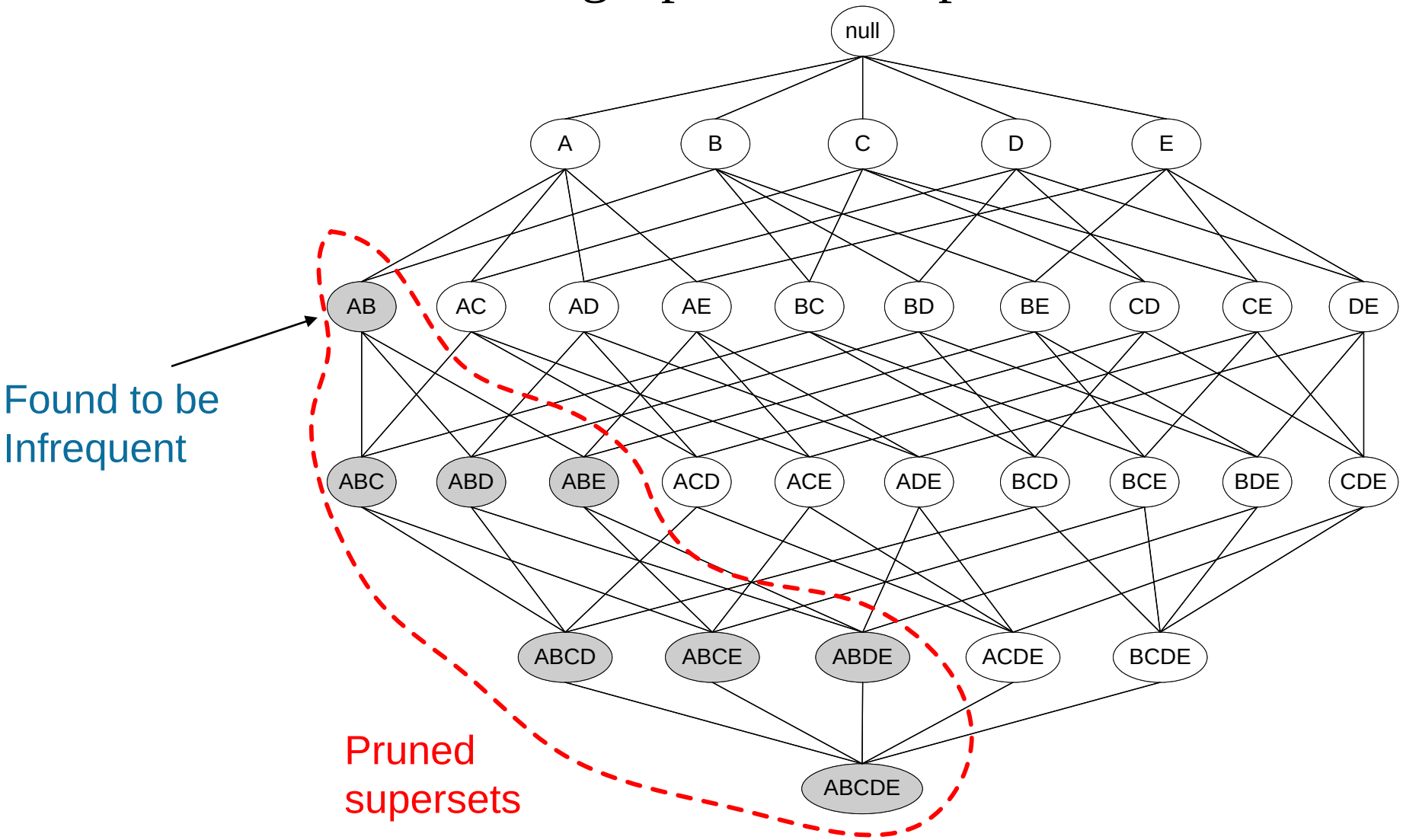
- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent

- Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

  - Support of an itemset never exceeds the support of its subsets
  - This is known as the anti-monotone property of support

Illustrating Apriori Principle



Closed Patterns and Max-Patterns

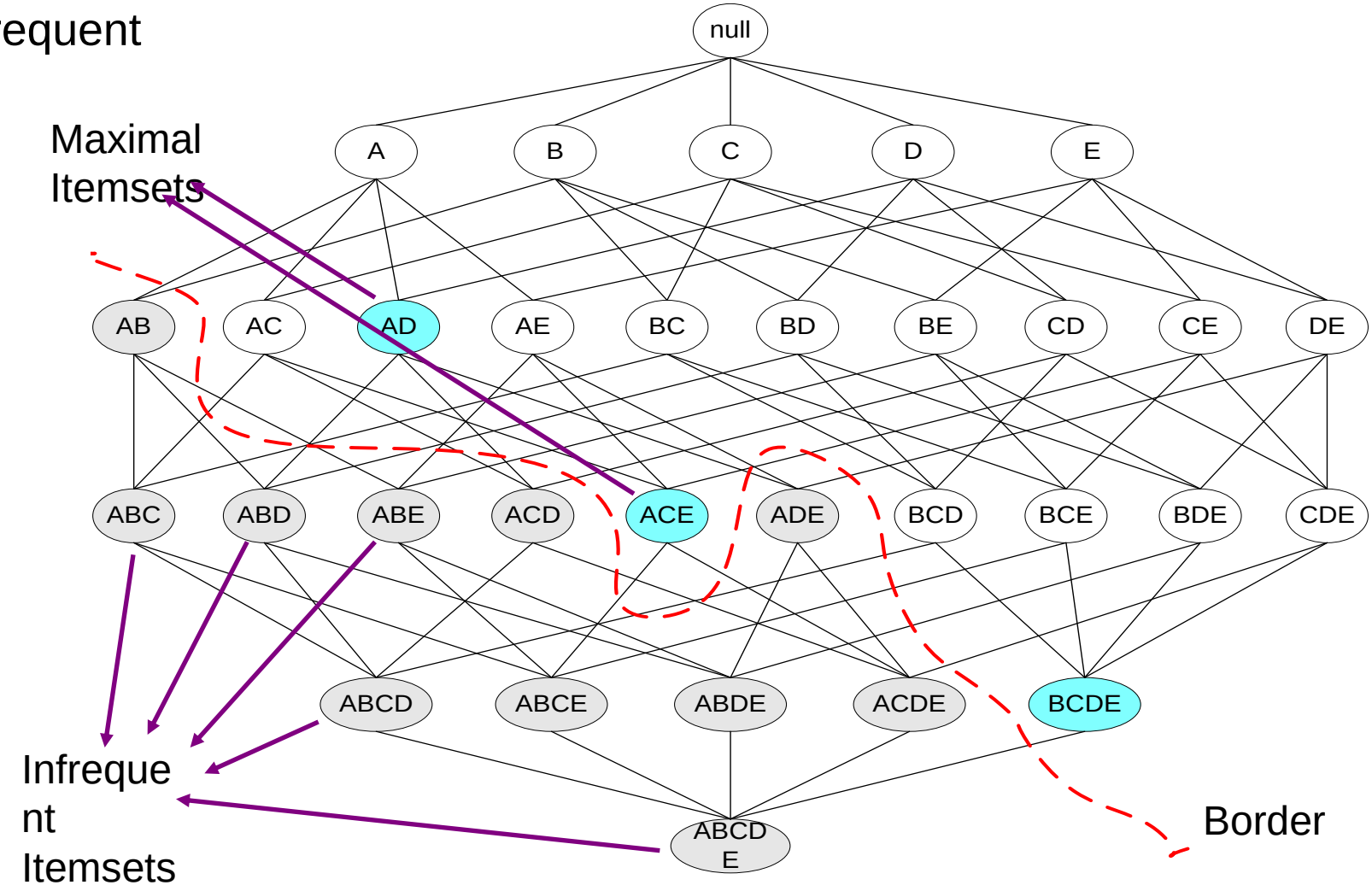
- A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \ldots, a_{100}\}$ contains $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{1}{1}\binom{0}{0}\binom{0}{0} = 2^{100} - 1 = 1.27*10^{30}$ sub-patterns!
- Solution: *Mine closed patterns and max-patterns instead*
- Closed Pattern:
  - An itemset *X* is **closed** in a data set *D* if there exists no proper super-itemset *Y* such that *Y* has the same support count as *X* in *D*.
    (proposed by Pasquier, et al. @ ICDT'99)
- Closed Frequent Pattern:
  - An itemset *X* is a **closed frequent pattern** in set *D* if *X* is both closed and frequent in *D*.
- Maximal Frequent Pattern:
  - An itemset X is a max-frequent-pattern if X is frequent and there exists no frequent super-pattern Y ⊃ X (proposed by Bayardo @ SIGMOD'98)
- Closed pattern is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules

Closed Patterns and Max-Patterns

- Exercise. DB = {<a_1, …, a_{100}>, < a_1, …, a_{50}>}
  - Min_sup = 1.
- What is the set of closed itemset?
  - <a_1, …, a_{100}>: 1
  - < a_1, …, a_{50}>: 2
- What is the set of max-pattern?
  - <a_1, …, a_{100}>: 1
- What is the set of all patterns?
  - !!

Maximal Frequent Itemset

An itemset is maximal frequent if none of its immediate supersets is frequent



Closed Itemset

- An itemset is closed if none of its immediate supersets has the same support as the itemset

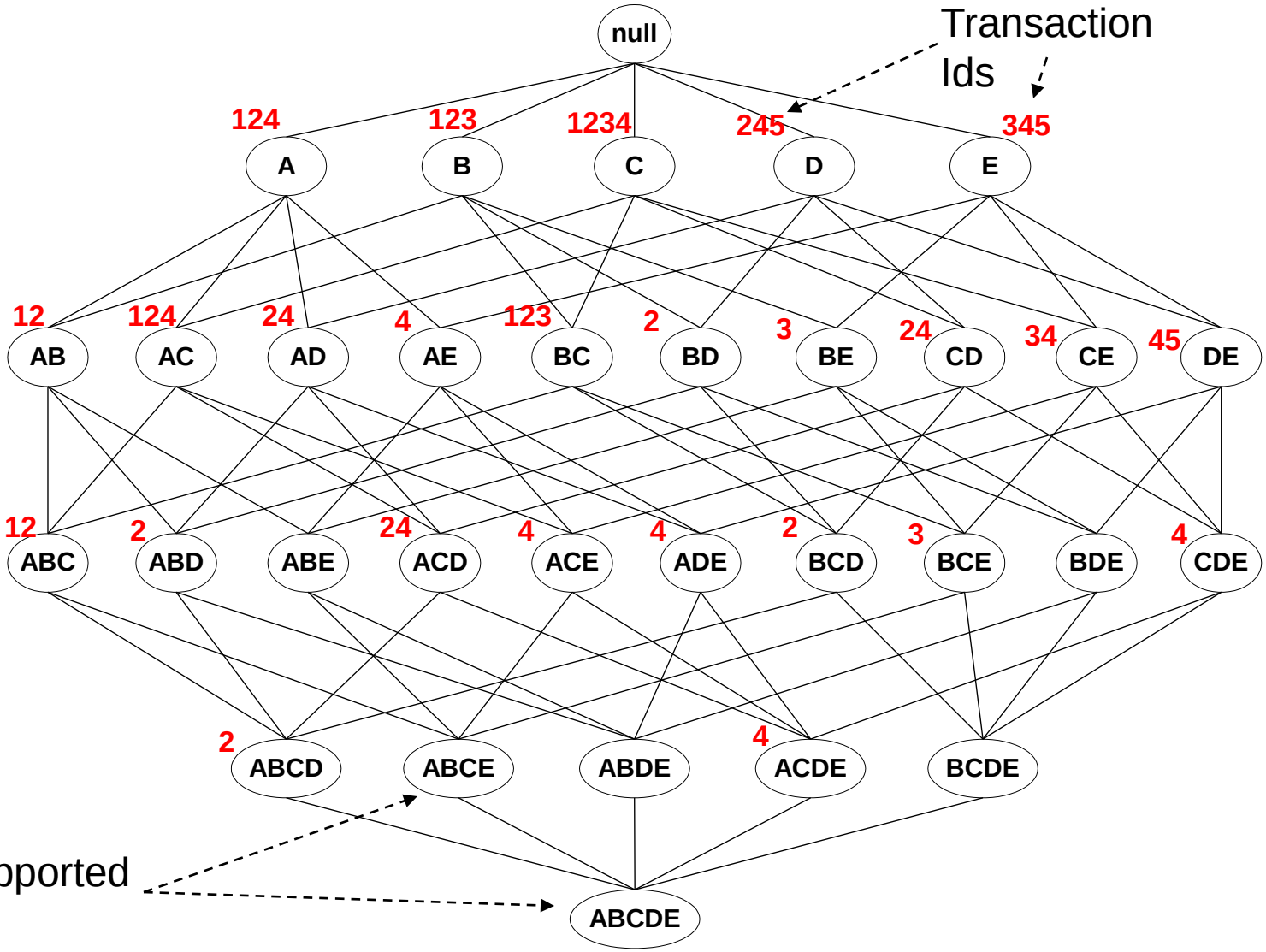
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	2
{A,D}	3
{B,C}	3
{B,D}	4
{C,D}	3

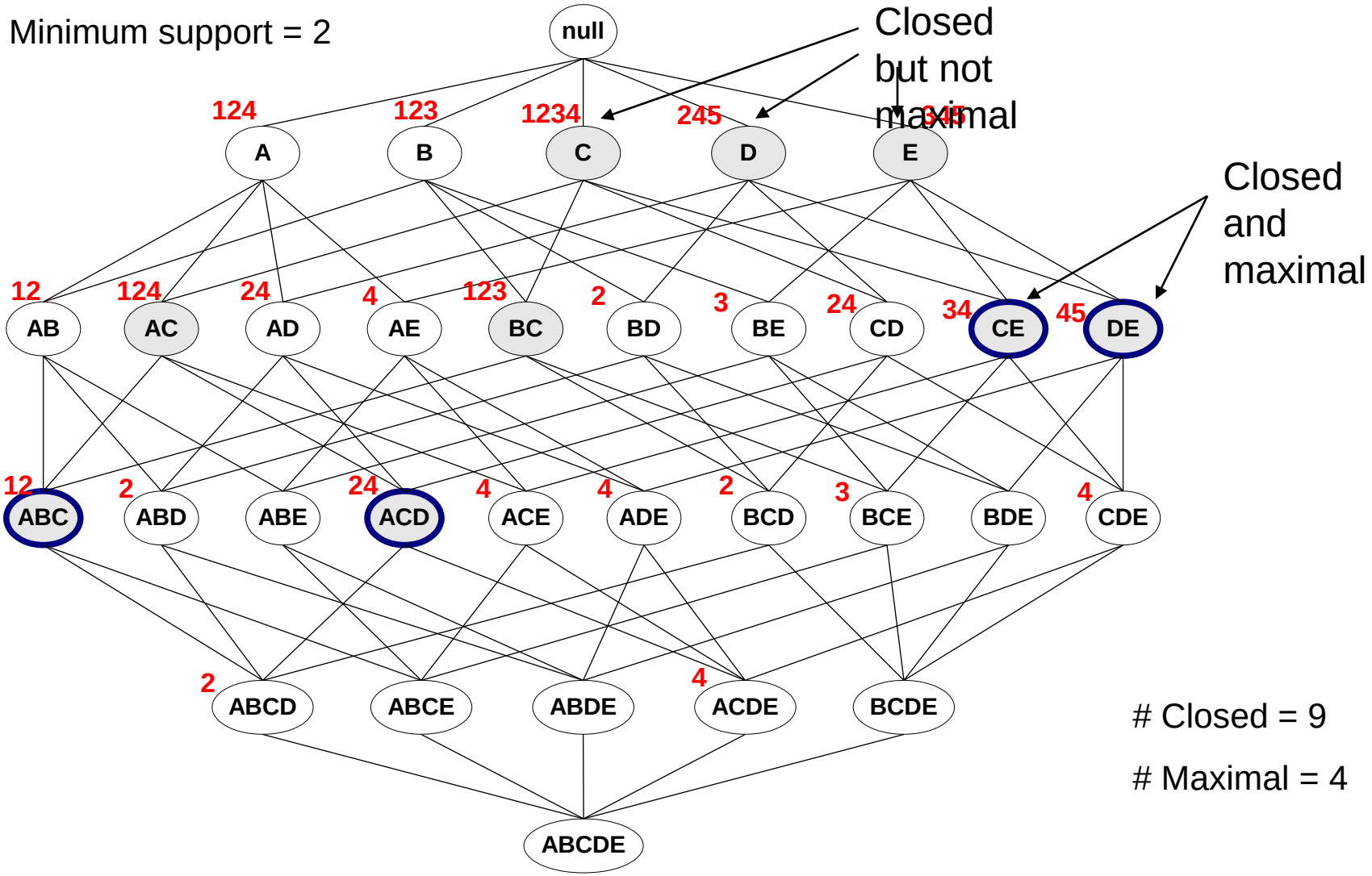
Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2

Maximal vs Closed Itemsets

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



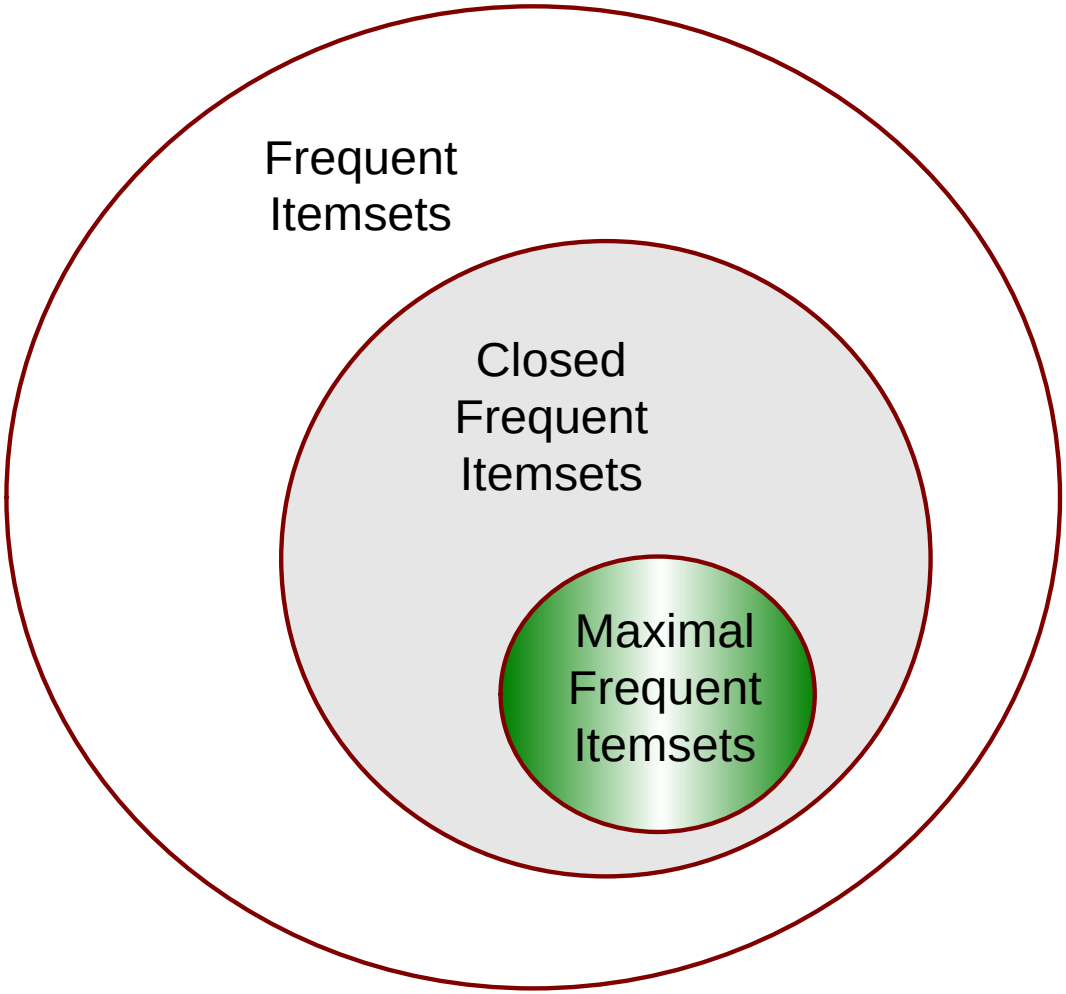
Maximal vs Closed Frequent Itemsets



About Closed frequent itemsets and Maximal frequent itemsets

- Let *C* be the set of closed frequent itemsets for a data set *D* satisfying a minimum support threshold, *min sup*.
- Let *M* be the set of maximal frequent itemsets for *D* satisfying *min sup*.
- Suppose that we have the support count of each itemset in *C* and *M*.
- Notice that C and its count information can be used to derive the whole set of frequent itemsets
  - Thus, we say that C contains complete information regarding its corresponding frequent itemsets.
  - On the other hand, M registers only the support of the maximal itemsets.
    - We can only assert about the support, but exact count is not known

Maximal vs Closed Itemsets



Computational Complexity of Frequent Itemset Mining

- How many itemsets are potentially to be generated in the worst case?
  - The number of frequent itemsets to be generated is senstive to the minsup threshold
  - When minsup is low, there exist potentially an exponential number of frequent itemsets
  - The worst case: M^N where M: # distinct items, and N: max length of transactions

Frequent Itemset Generation Strategies

- Reduce the number of candidates (M)
  - Complete search: $M=2^d$
  - Use pruning techniques to reduce M
- Reduce the number of transactions (N)
  - Reduce size of N as the size of itemset increases
  - Used by DHP and vertical-based mining algorithms
- Reduce the number of comparisons (NM)
  - Use efficient data structures to store the candidates or transactions
  - No need to match every candidate against every transaction

Presentation Outline

- Basic concepts
  - Frequent itemssets, Closed Itemsets, and Association Rules
- Frequent itemset mining methods
  - **Apriori Algorithm**
    - **Generating Association Rules from Frequent Itemsets**
    - Improving the Efficiency of Apriori
  - A Pattern-Growth Approach
  - Mining Frequent Itemsets Using the Vertical Data Format
  - Mining Closed and Max Patterns
- Which Patterns are interesting? Pattern Evaluation Methods
- Summary

Fast Algorithms for Mining Association Rules

Rakesh Agrawal Ramakrishnan Srikant*

IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120

Proceedings of the 20th VLDB Conference
Santiago, Chile, 1994

Apriori: A Candidate Generation & Test Approach

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!
- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

The Trick

Any subset of large itemset is large.

Therefore

To find large k-itemset

- Create candidates by combining large k-1 itemsets.
- Delete those that contain any subset that is not large.

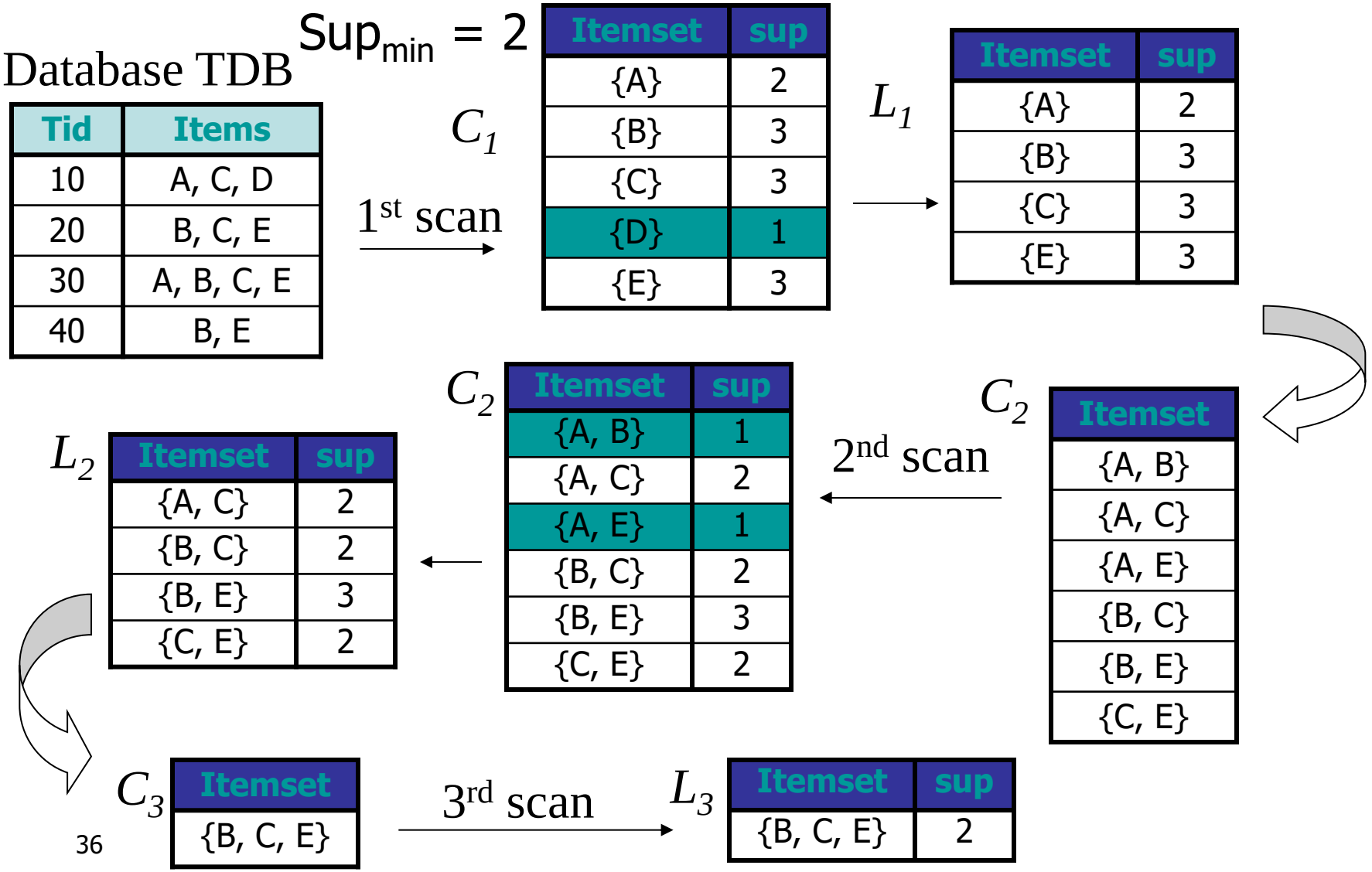
Apriori Algorithm

Notations

- k-itemset: An itemset having k items
- L_k: Set of large k-itemsets (those with minimum support). Each member of this set has two fields: i) itemset and ii) support count
- C_k: Set of candidate k-itemsets (potentially large itemsets). Each member of this set has two fields: i) itemset and ii) support count

- *Candidate* itemsets are generated using only the *large* itemsets of the previous pass without considering the transactions in the database.
- The *large* itemset of the previous pass is joined with itself to generate all itemsets whose size is higher by 1.
- Each generated itemset, that has a subset which is not *large*, is deleted. The remaining itemsets are the *candidate* ones.

The Apriori Algorithm—An Example



The Apriori Algorithm (Pseudo-Code)

1) L_1 = {large 1-itemsets};
2) **for** ($k = 2$; $L_{k-1} \neq \emptyset$; k++) **do begin**
3) C_k = apriori-gen(L_{k-1}); // New candidates
4) **forall transactions** $t \in \mathcal{D}$ **do begin**
5) C_t = subset(C_k, t); // Candidates contained in t
6) **forall candidates** $c \in C_t$ **do**
7) c.count++;
8) **end**
9) $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$
10) **end**
11) Answer = $\bigcup_k L_k$;

Figure 1: Algorithm Apriori

Notations

- k-itemset: An itemset having k items
- L_k: Set of large k-itemsets (those with minimum support). Each member of this set has two fields: i) itemset and ii) support count
- C_k: Set of candidate k-itemsets (potentially large itemsets). Each member of this set has two fields: i) itemset and ii) support count

Apriori-gen(L_{k-1})

The **apriori-gen** function takes as argument L_{k-1}, the set of all large $(k-1)$-itemsets. It returns a superset of the set of all large k-itemsets. The function works as follows. [1] First, in the *join* step, we join L_{k-1} with L_{k-1}:

insert into C_k
select $p.\text{item}_1$, $p.\text{item}_2$, ..., $p.\text{item}_{k-1}$, $q.\text{item}_{k-1}$
from L_{k-1} p, L_{k-1} q
where $p.\text{item}_1 = q.\text{item}_1$, ..., $p.\text{item}_{k-2} = q.\text{item}_{k-2}$,
$p.\text{item}_{k-1} < q.\text{item}_{k-1}$;

Prune Step

- In the Prune step we delete all itemsets c ε C_k Ck such that some (k-1)-subset of c is not in L_{k-1}

forall itemsets $c \in C_k$ **do**
  forall $(k-1)$-subsets s of c **do**
    if $(s \notin L_{k-1})$ **then**
      delete c from C_k;

Self-joining and Pruning: Example

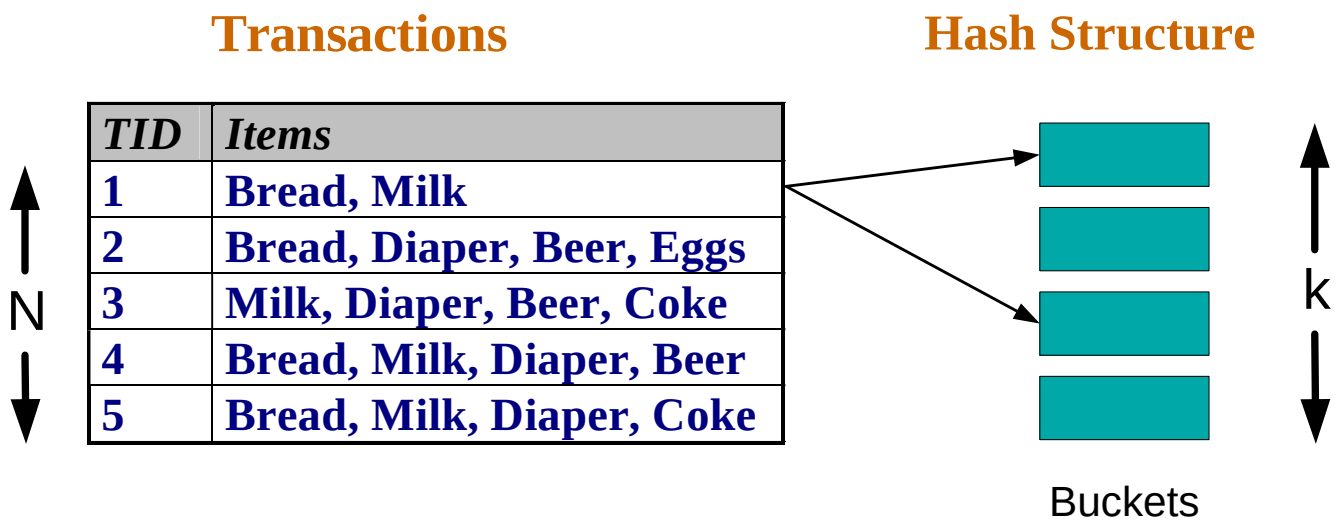
- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- Example of Candidate-generation
  - L_3={*abc*, *abd*, *acd*, *ace*, *bcd*}
  - Self-joining: L_3*L_3
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in L_3
  - C_4 = {*abcd*}

How to Count Supports of Candidates?

- Why counting supports of candidates a problem?
  - The total number of candidates can be very huge
  - One transaction may contain many candidates
- Method:
  - Candidate itemsets are stored in a *hash-tree*
  - *Leaf* node of hash-tree contains a list of itemsets and counts
  - *Interior* node contains a hash table
  - *Subset function*: finds all the candidates contained in a transaction

Reducing Number of Comparisons

- Candidate counting:
  - Scan the database of transactions to determine the support of each candidate itemset
  - To reduce the number of comparisons, store the candidates in a hash structure
    - Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets



How to Count Supports of Candidates?

- Candidate itemsets Ck are stored in a hash-tree.
- A node of the hash-tree either contains a list of itemsets (a leaf node) or a hash table (an interior node).
- In an interior node, each bucket of the hash table points to another node. The root of the hash-tree is defined to be at depth 1.
- An interior node at depth d points to nodes at depth d+1.
- Itemsets are stored in the leaves.
- When we add an itemset c, we start from the root and go down the tree until we reach a leaf. At an interior node at depth d, we decide which branch to follow by applying a hash function to the d'th item of the itemset.
- All nodes are initially created as leaf nodes. When the number of itemsets in a leaf node exceeds a specified threshold, the leaf node is converted to an interior node.

How to Count Supports of Candidates?

- Starting from the root node, the subset function finds all the candidates contained in a transaction t as follows.
  - If we are at a leaf, we finds which of the itemsets in the leaf are contained in t and add references to them to the answer set.
  - If we are at an interior node and we have reached it by hashing the item i, we hash on each item that comes after i in t and recursively apply this procedure to the node in the corresponding bucket. For the root node, we hash on every item in t.

How to Count Supports of Candidates?

- To see why the subset function returns the desired set of references, consider what happens at the root node.
  - For any itemset c contained in transaction t, the first item of c must be in t.
  - At the root, by hashing on every item in t, we ensure that we only ignore itemsets that start with an item not in t. Similar arguments apply at lower depths.
  - The only additional factor is that, since the items in any itemset are ordered, if we reach the current node by hashing the item i, we only need to consider the items in t that occur after i.
  - If k is the size of a candidate itemset in the hash-tree, we can find in O(k) time whether the itemset is contained in a transaction by using a temporary bitmap. Each bit of the bitmap corresponds an item. The bitmap is created once for the data structure, and reinitialized for each transaction. This initialization takes O(size(transaction)) time for each transaction.

Subset Function

- Candidate itemsets - C_k are stored in a hash-tree
- Finds in O(k) time whether a candidate itemset of size k is contained in transaction t.
- Total time O(max(k,size(t))

$L_1 = \{large\ 1\text{-}itemsets\}$
For $(k = 2;\ L_{k-1} \neq \phi\ ;\ k++)$ do begin
    C_k = apriori-gen (L_{k-1});
    forall transactions $t \in D$ do begin
        C_t = subset (C_k, t)
        forall candidates $c \in C_t$ do
            $c.count++$;
        end
    end
    $L_k = \{ c \in C_k | c.count \geq minsup\}$
end
$Answer = \bigcup_k L_k$;

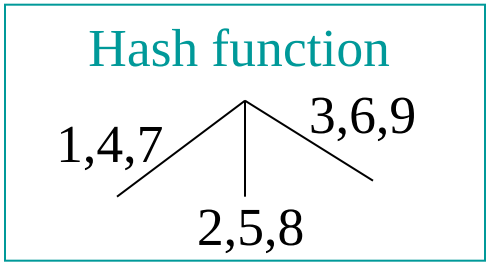
Generate Hash Tree

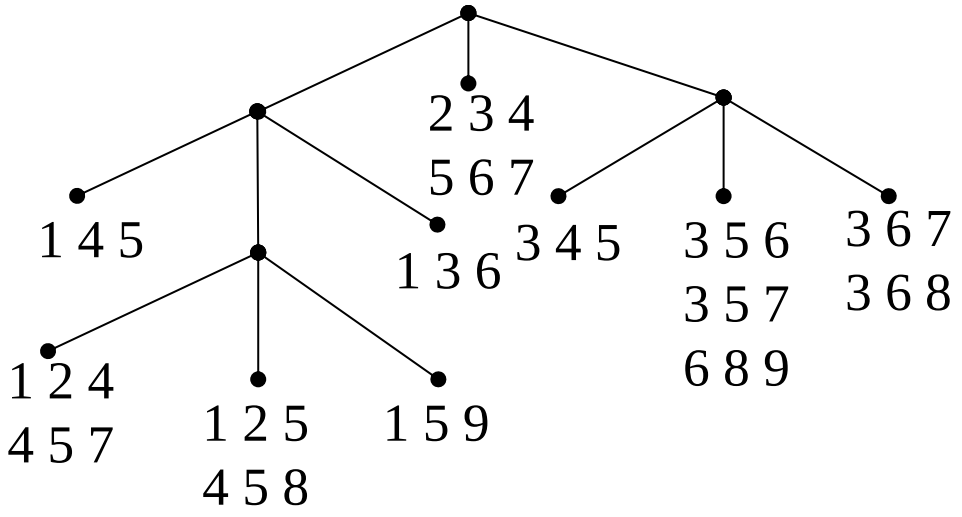
Suppose you have 15 candidate itemsets of length 3:

{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

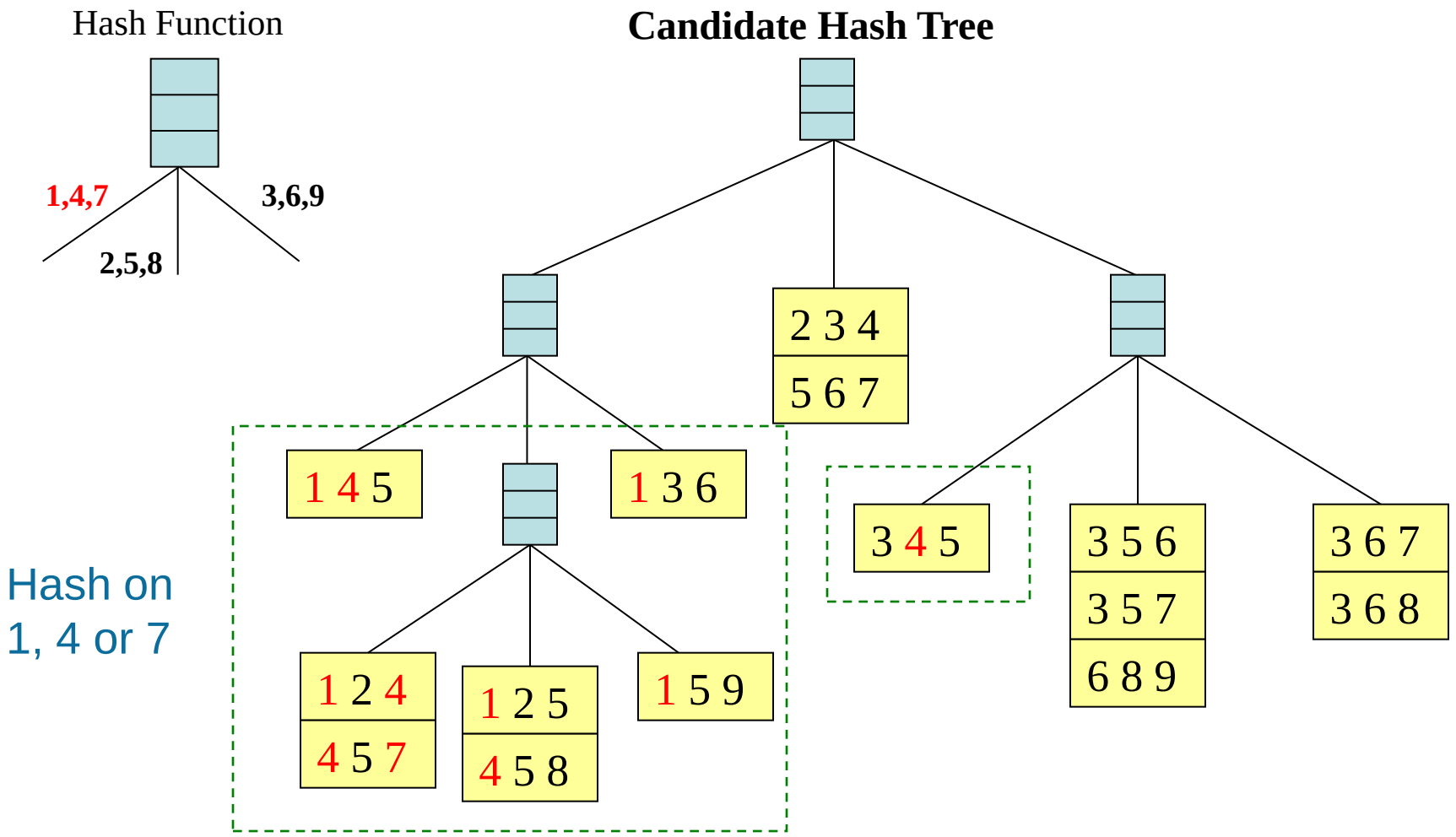
You need:

- **Hash function**
- **Max leaf size: max number of itemsets stored in a leaf node (if number of candidate itemsets exceeds max leaf size, split the node)**

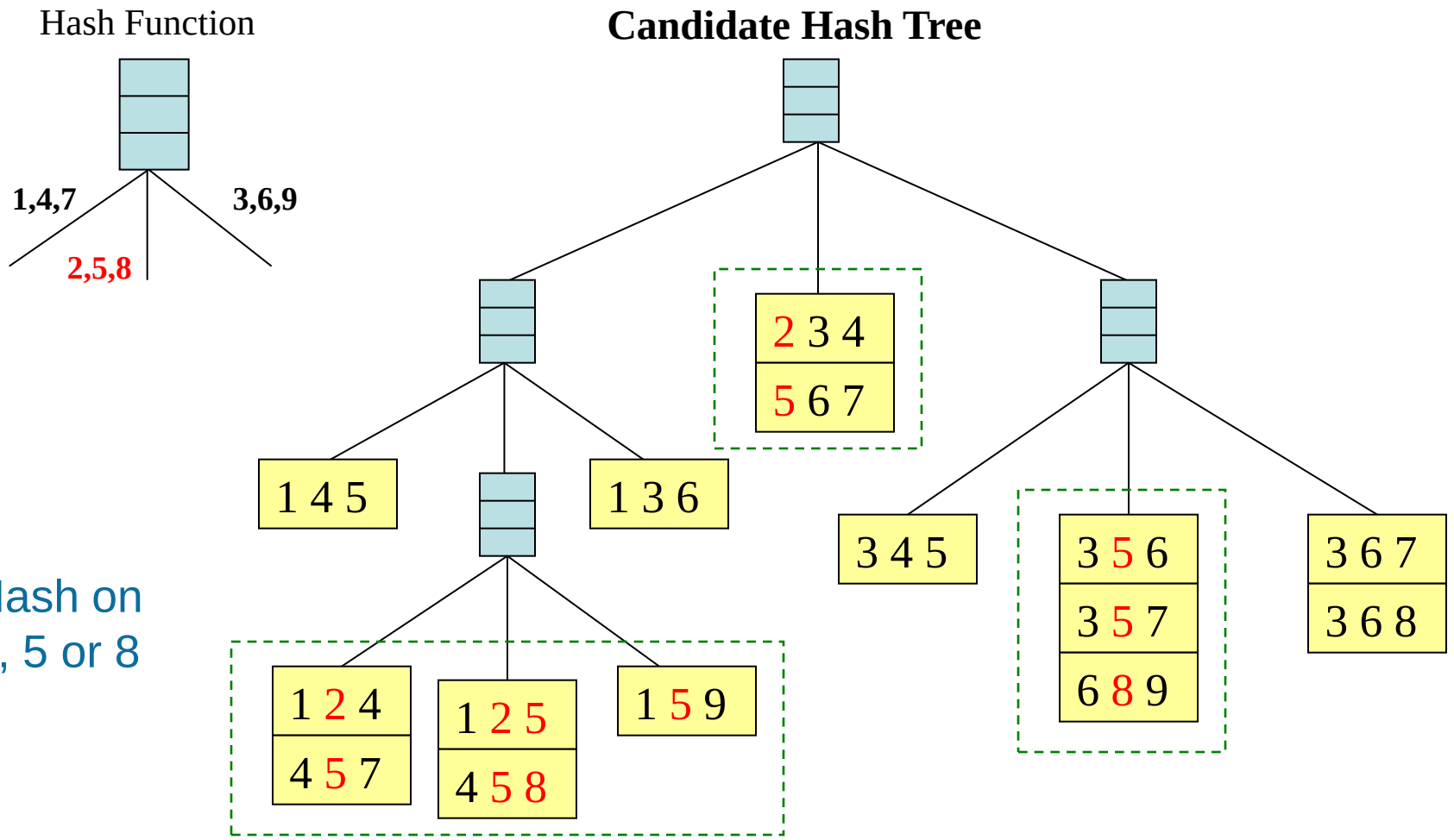




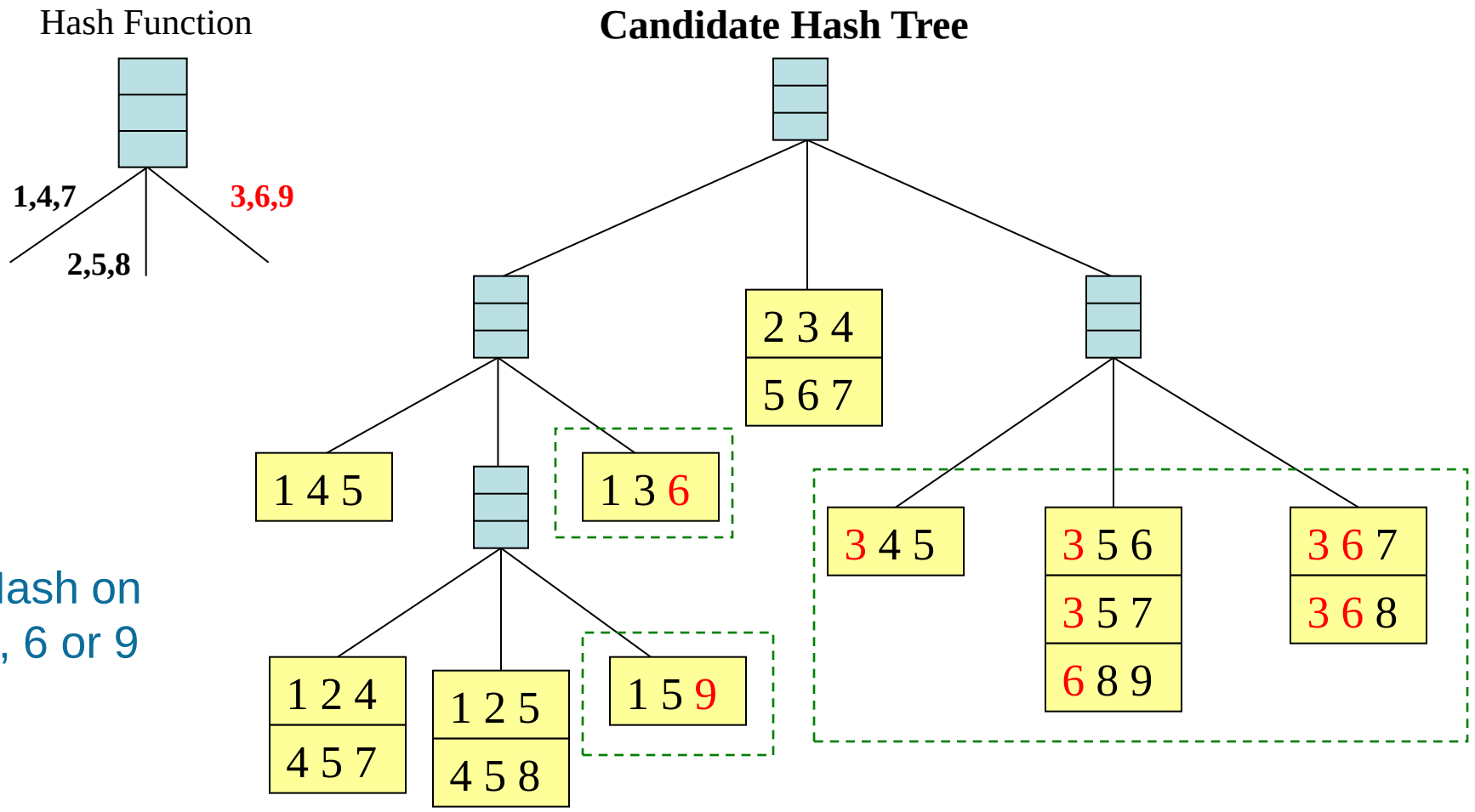
Association Rule Discovery: Hash tree



Association Rule Discovery: Hash tree

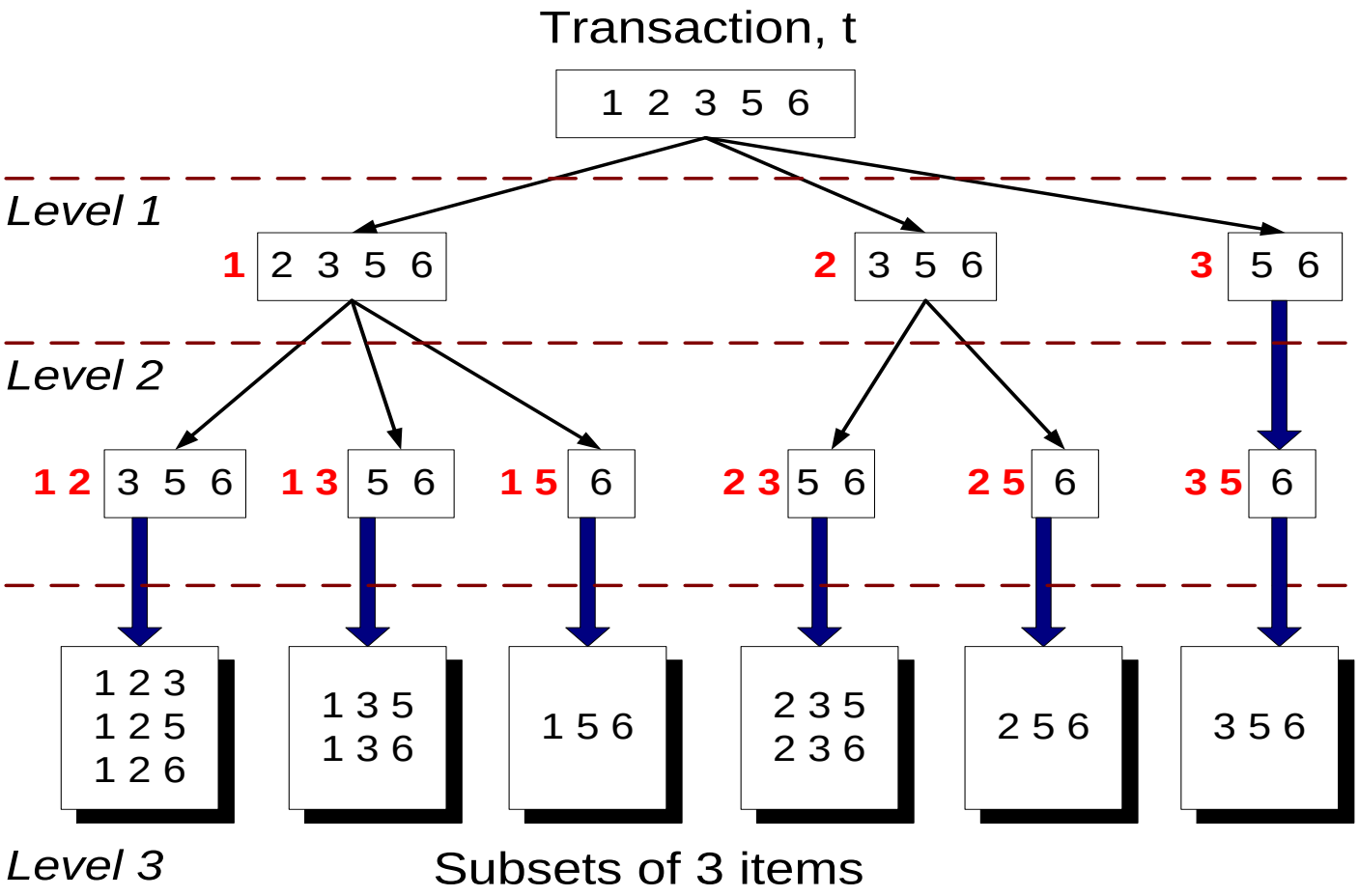


Association Rule Discovery: Hash tree

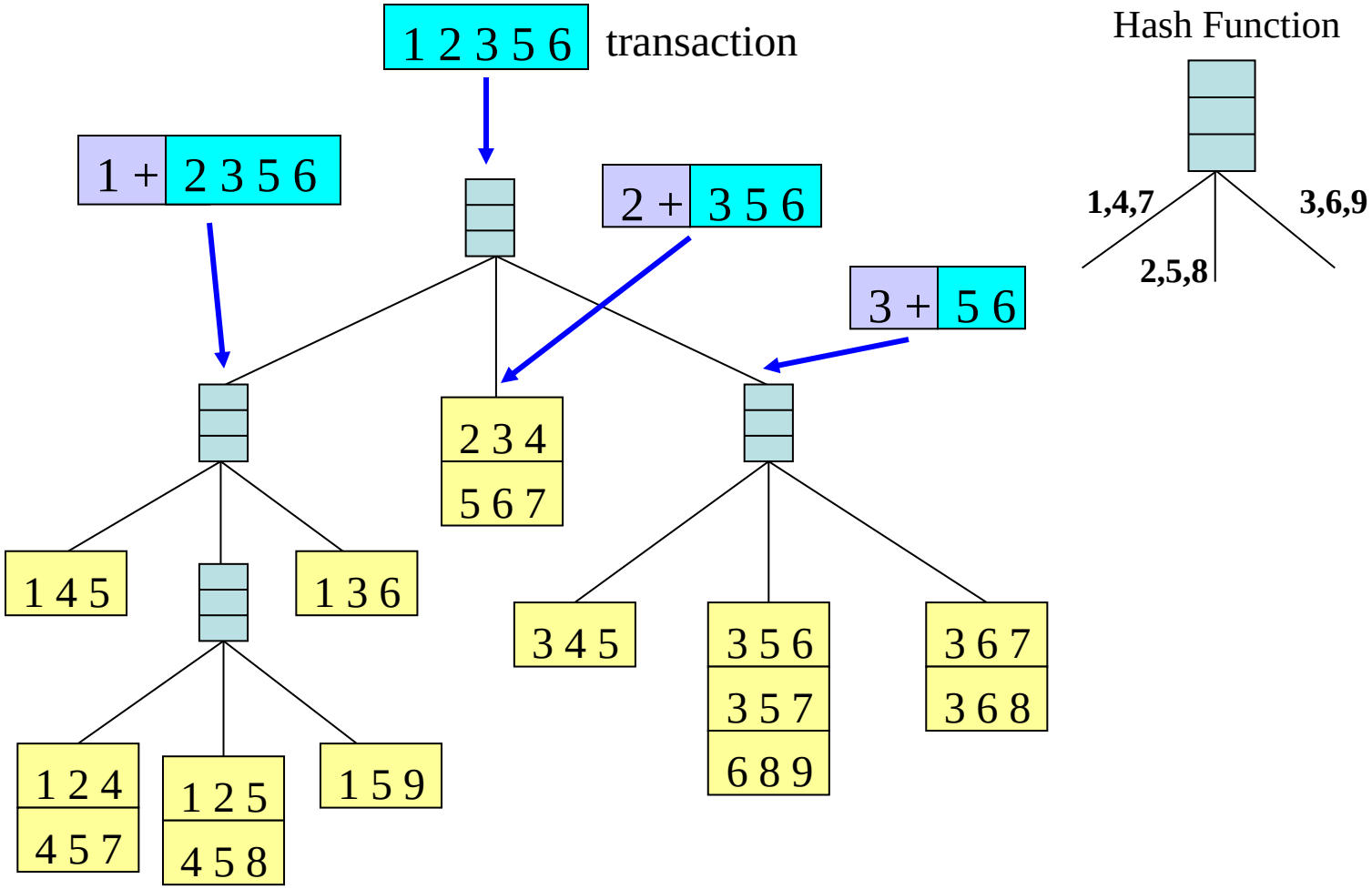


Subset Operation

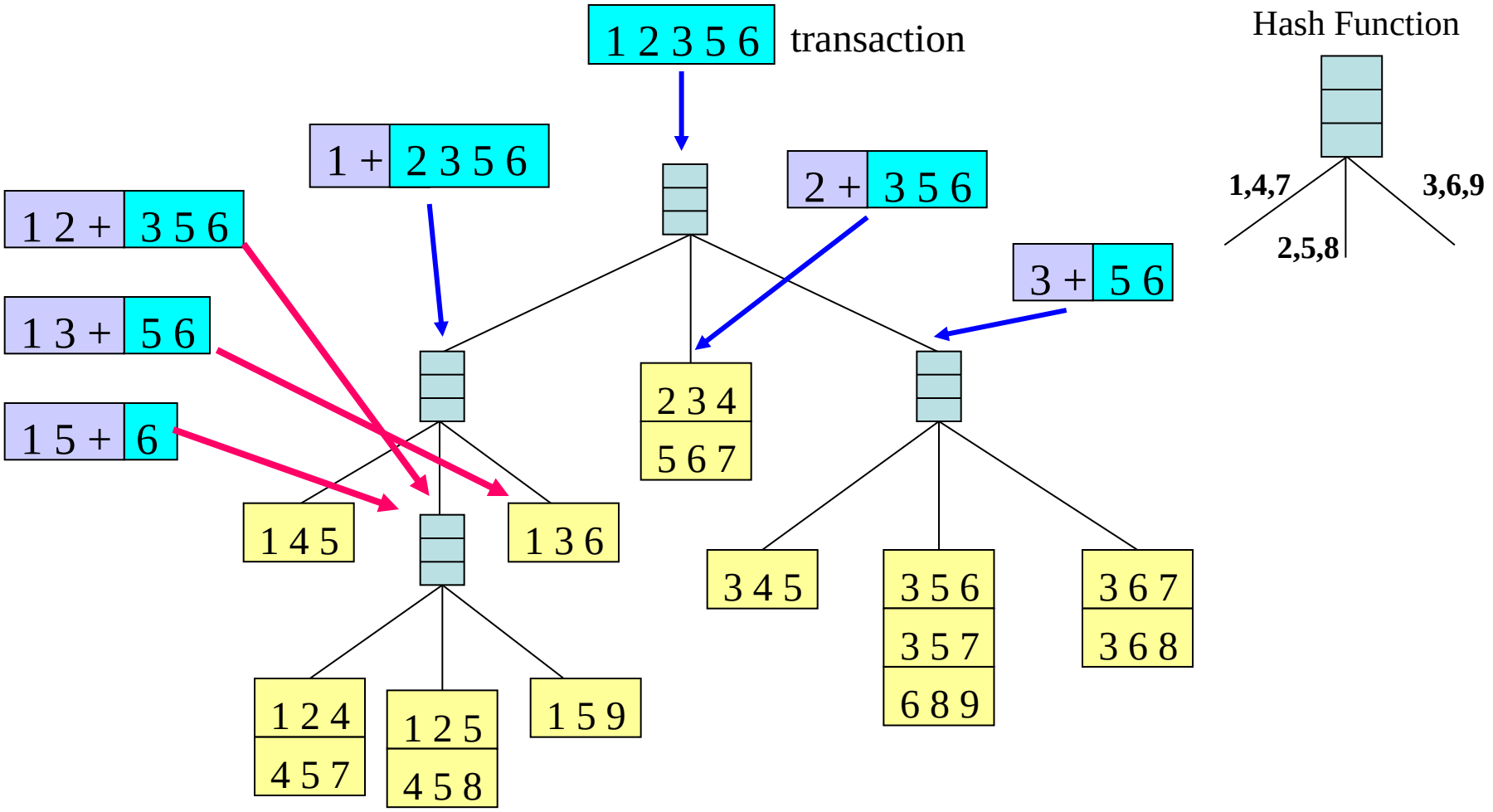
Given a transaction t, what are the possible subsets of size 3?



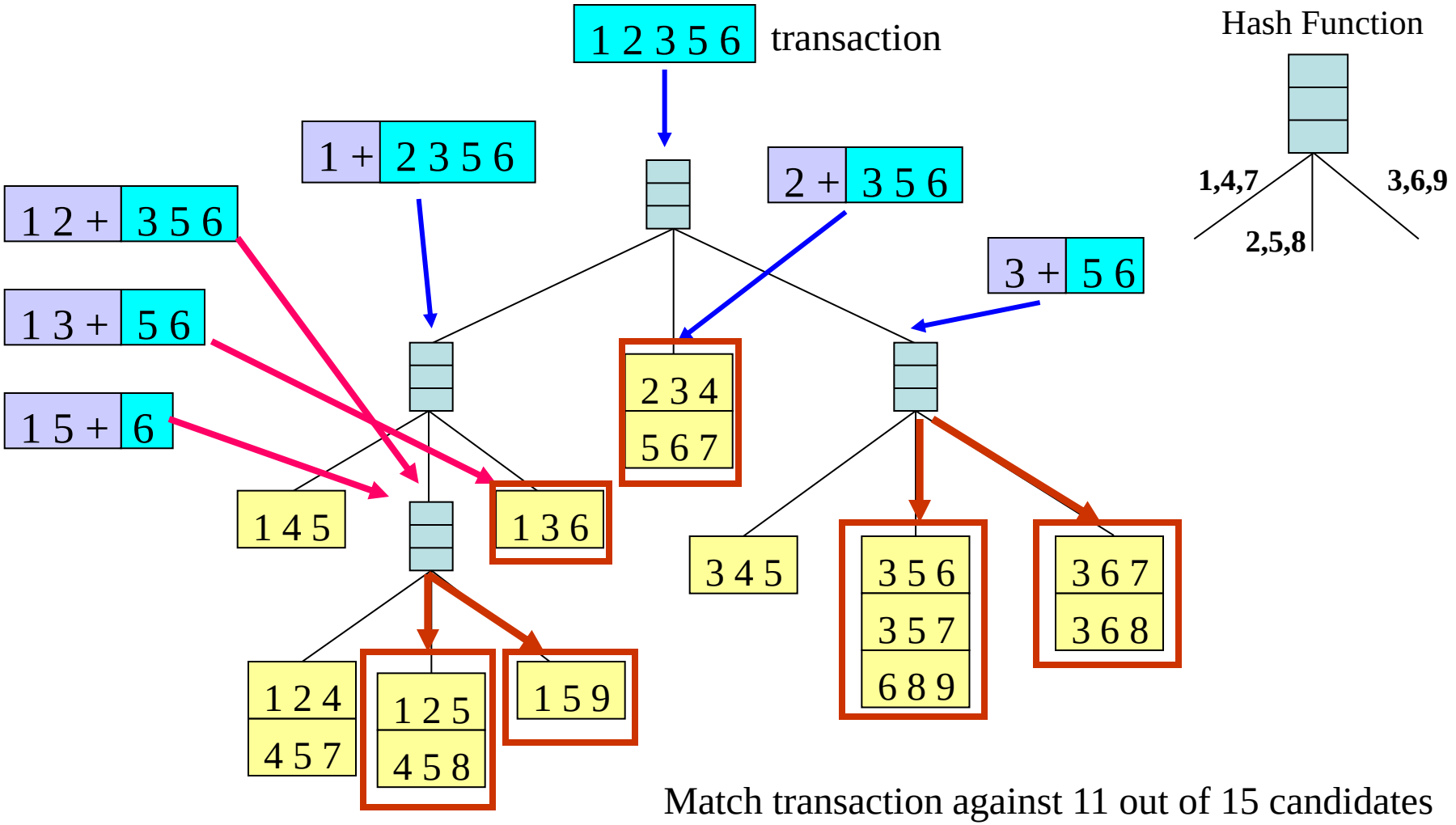
Subset Operation Using Hash Tree



Subset Operation Using Hash Tree

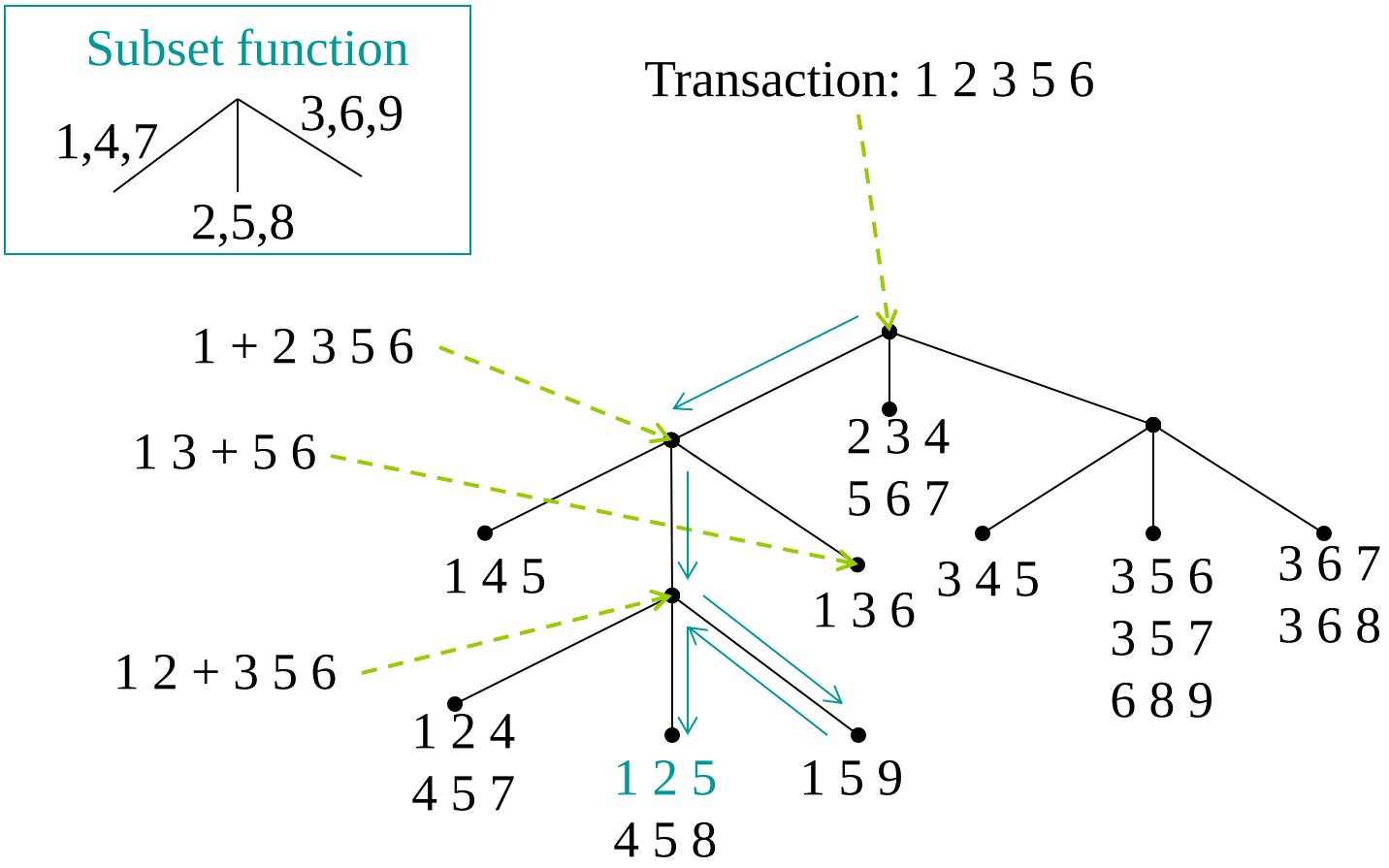


Subset Operation Using Hash Tree



Match transaction against 11 out of 15 candidates

Example: Counting Supports of Candidates



Generating Association Rules

- Once the frequent itemsets are found, it is straightforward to generate strong association rules from them (where strong association rules satisfy both minimum support and minimum confidence).

$$\textbf{Confidence } (X \rightarrow Y) = \frac{\textbf{Support } (X \cup Y)}{\textbf{Support } (X)}$$

- For each frequent itemset *l*, generate all nonempty subsets of *l*.
- For every nonempty subset *s* of *l*, output the rule $l \rightarrow (l-s)$,

  if $\frac{\textit{support-count } (l)}{\textit{support-count } (s)} >= \textit{min\_conf}$, where *min conf* is the minimum confidence threshold.
- Because the rules are generated from frequent itemsets, each one automatically satisfies the minimum support.
- Frequent itemsets can be stored ahead of time in hash tables along with their counts so that they can be accessed quickly

Example

- The data contain frequent itemset *X* (I1, I2, I5). What are the association rules that can be generated from *X*?
- Generate the nonempty subsets of *X* are {I1, I2}, {I1, I5}, {I2, I5}, {I1}, {I2}, and {I5}.
- The resulting association rules are
  - {I1, I2}→{I5}= confidence=2/4=50%
  - {I1, I5}→{I2}=2/2=100%
  - {I2, I5}→ {I1}=2/2=100%
  - {I1}→{I2,I5}= 2/6=33%
- ….
- …
- If minimum confidence =70%, second and third are potential rules

Table 6.1 Transactional Data for an *AllElectronics* Branch

TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Presentation Outline

- Basic concepts
  - Frequent itemssets, Closed Itemsets, and Association Rules
- Frequent itemset mining methods
  - Apriori Algorithm
    - Generating Association Rules from Frequent Itemsets
    - **Improving the Efficiency of Apriori**
  - A Pattern-Growth Approach
  - Mining Frequent Itemsets Using the Vertical Data Format
  - Mining Closed and Max Patterns
- Which Patterns are interesting? Pattern Evaluation Methods
- Summary

Further Improvement of the Apriori Method

- Apriori: Major computational challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

DHP (Direct Hashing and Pruning): Reduce the Number of Candidates

- Use of hash to reduce the size of candidate *k*-itemsets, *Ck*, for *k* > 1.
- For example, when scanning each transaction in the database to generate the frequent 1-itemsets, *L*1, we can generate all the 2-itemsets for each transaction, hash (i.e., map) them into the different *buckets* of a *hash table* structure, and increase the corresponding bucket counts.
- A 2-itemset with a corresponding bucket count in the hash table that is below the support threshold cannot be frequent (especially when k=2)
- J. Park, M. Chen, and P. Yu. An effective hash-based algorithm for mining association rules. *SIGMOD'95*

Create hash table H_2 using hash function $h(x, y) = ((order\ of\ x) \times 10 + (order\ of\ y))\ mod\ 7$

H_2

bucket address	0	1	2	3	4	5	6
bucket count	2	2	4	2	2	4	4
bucket contents	{I1, I4} {I3, I5}	{I1, I5} {I1, I5}	{I2, I3} {I2, I3} {I2, I3} {I2, I3}	{I2, I4} {I2, I4}	{I2, I5} {I2, I5}	{I1, I2} {I1, I2} {I1, I2} {I1, I2}	{I1, I3} {I1, I3} {I1, I3} {I1, I3}

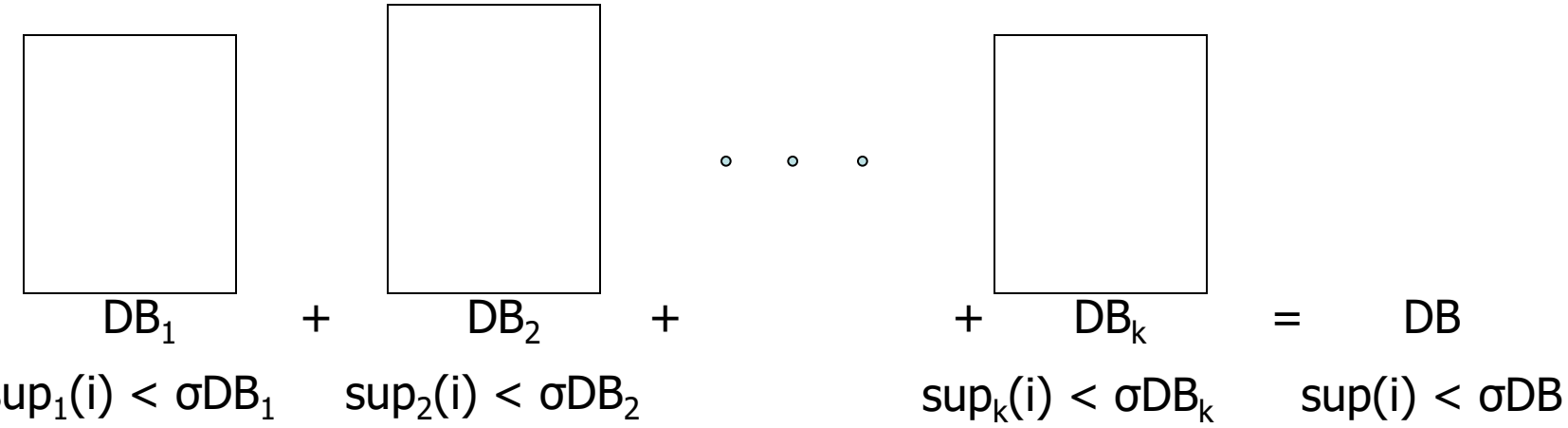
Figure 6.5 Hash table, H_2, for candidate 2-itemsets. This hash table was generated by scanning Table 6.1's transactions while determining L_1. If the minimum support count is, say, 3, then the itemsets in buckets 0, 1, 3, and 4 cannot be frequent and so they should not be included in C_2.

Transaction Reduction

- Reducing the number of transactions scanned in future iterations):
- A transaction that does not contain any frequent *k*-itemsets cannot contain any
  frequent (k+1)-itemsets.
- Therefore, such a transaction can be marked or removed
  from further consideration because subsequent database scans for *j*-itemsets, where

Partition: Scan Database Only Twice

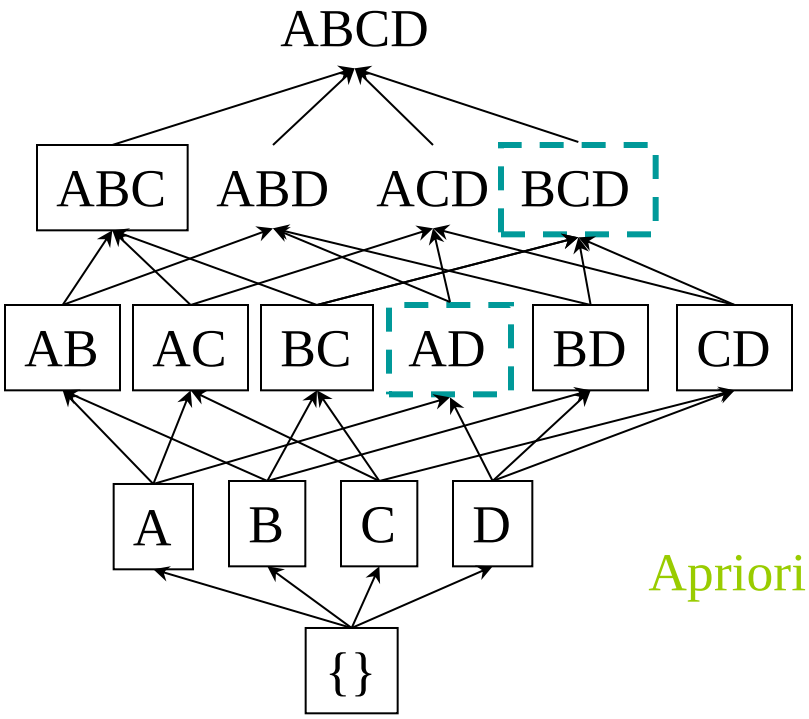
- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: partition database and find local frequent patterns
  - Scan 2: consolidate global frequent patterns
- A. Savasere, E. Omiecinski and S. Navathe, *VLDB'95*



Sampling for Frequent Patterns

- Select a sample of original database, mine frequent patterns within sample using Apriori
- Scan database once to verify frequent itemsets found in sample, only *borders* of closure of frequent patterns are checked
  - Example: check *abcd* instead of *ab, ac, …, etc.*
- Scan database again to find missed frequent patterns
- H. Toivonen. Sampling large databases for association rules. In *VLDB'96*

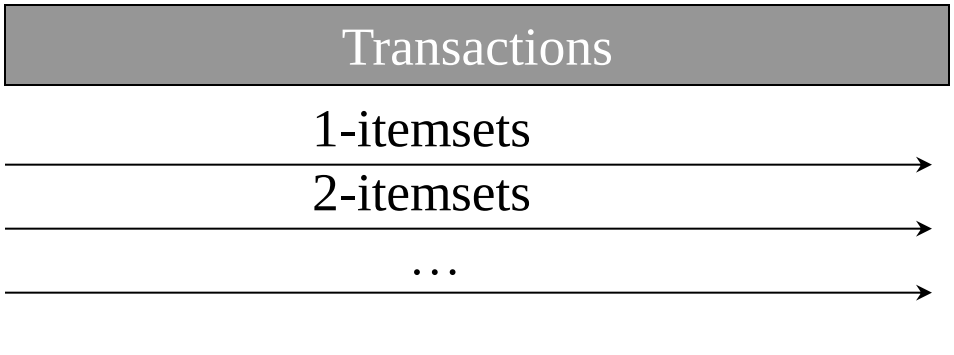
Dynamic Itemset Counting (DIC): Reduce Number of Scans



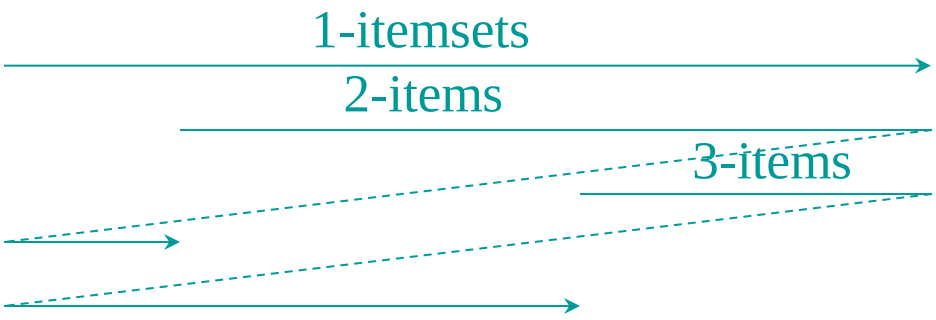
Itemset lattice

- Once both A and D are determined frequent, the counting of AD begins
- Once all length-2 subsets of BCD are determined frequent, the counting of BCD begins

Apriori



DIC



S. Brin R. Motwani, J. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. *SIGMOD'97*

Basic Idea of DIC

If we consider Apriori in this metaphor, all itemsets must get on at the start of a pass and get off at the end. The 1-itemsets take the first pass, the 2-itemsets take the second pass, and so on (see Figure 1). In DIC, we have the added flexibility of allowing itemsets to get on at any stop as long as they get off at the same stop the next time the train goes around. Therefore, the itemset has "seen" all the transactions in the file. This means that we can start counting an itemset as soon as we suspect it may be necessary to count it instead of waiting until the end of the previous pass.

For example, if we are mining 40,000 transactions and $M = 10,000$, we will count all the 1-itemsets in the first 40,000 transactions we will read. However, we will begin counting 2-itemsets after the first 10,000 transactions have been read. We will begin counting 3-itemsets after 20,000 transactions. For now, we assume there are no 4-itemsets we need to count. Once we get to the end of the file, we will stop counting the 1-itemsets and go back to the start of the file to count the 2 and 3-itemsets. After the first 10,000 transactions, we will finish counting the 2-itemsets and after 20,000 transactions, we will finish counting the 3-itemsets. In total, we have made 1.5 passes over the data instead of the 3 passes a level-wise algorithm would make.[3]

Presentation Outline

- Basic concepts
  - Frequent itemssets, Closed Itemsets, and Association Rules
- Frequent itemset mining methods
  - Apriori Algorithm
    - Generating Association Rules from Frequent Itemsets
    - Improving the Efficiency of Apriori
  - **A Pattern-Growth Approach**
  - Mining Frequent Itemsets Using the Vertical Data Format
  - Mining Closed and Max Patterns
- Which Patterns are interesting? Pattern Evaluation Methods
- Summary

Mining Frequent Patterns without Candidate Generation

Proc. 2000 ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD'00), Dallas, TX, May 2000.

Jiawei Han, Jian Pei, and Yiwen Yin

(FP Tree method)

Pattern-Growth Approach: Mining Frequent Patterns Without Candidate Generation

- Bottlenecks of the Apriori approach
  - Breadth-first (i.e., level-wise) search
  - Candidate generation and test
    - Often generates a huge number of candidates
- The FPGrowth Approach (J. Han, J. Pei, and Y. Yin, SIGMOD' 00)
  - Depth-first search
  - Avoid explicit candidate generation
- Major philosophy: Grow long patterns from short ones using local frequent items only
  - "abc" is a frequent pattern
  - Get all transactions having "abc", i.e., project DB on abc: DB|abc
  - "d" is a local frequent item in DB|abc → abcd is a frequent pattern

Main Concepts of FP-growth

- F-list
- FP tree
- Header table
- Conditional pattern bases
- Conditional FP-trees

Construct FP-tree from a Transaction Database

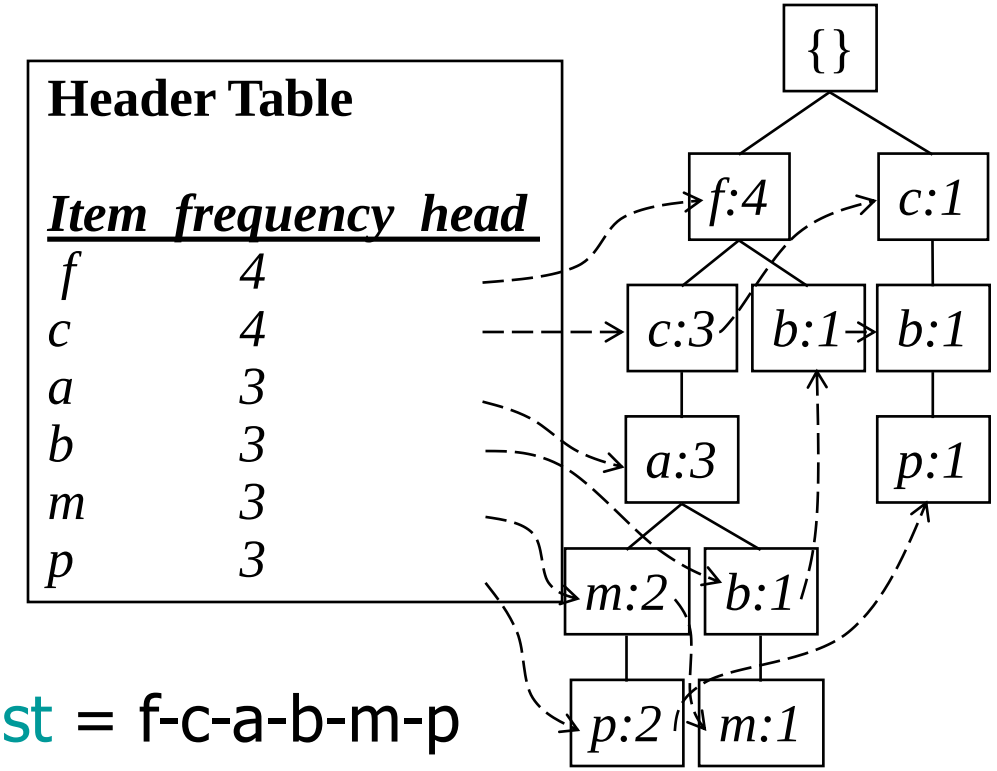
TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

min_support = 3

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree

Header Table

Item	frequency	head
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



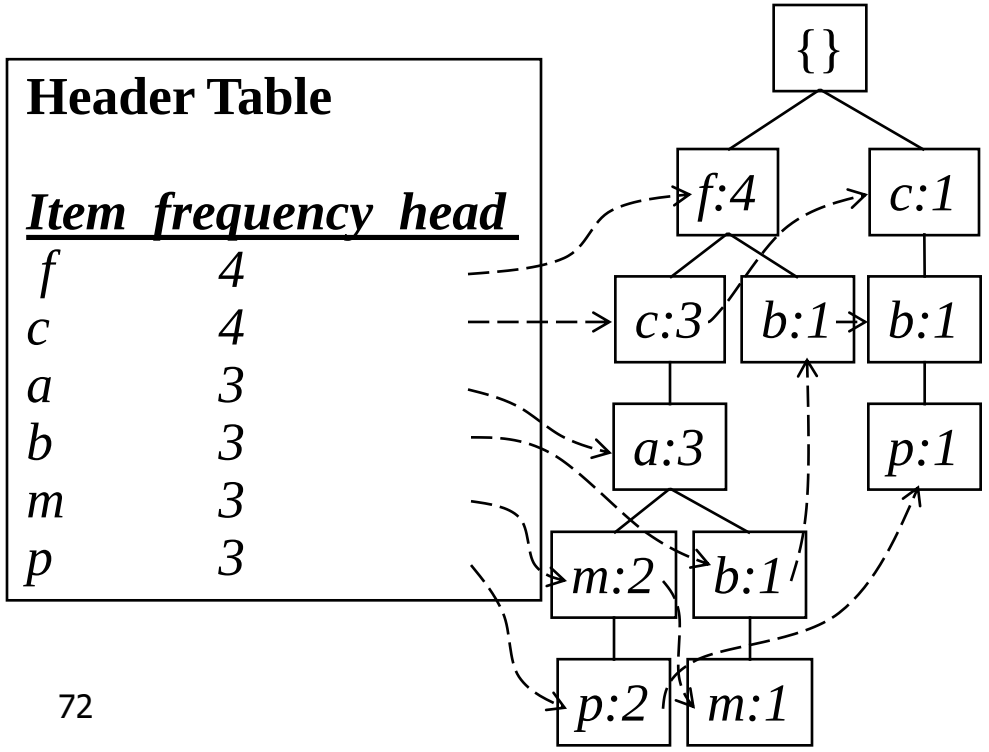
F-list = f-c-a-b-m-p

Partition Patterns and Databases

- Frequent patterns can be partitioned into subsets according to f-list
  - F-list = f-c-a-b-m-p
  - Patterns containing p
  - Patterns having m but no p
  - …
  - Patterns having c but no a nor b, m, p
  - Pattern f
- Completeness and non-redundency

Find Patterns Having P From P-conditional Database

- Starting at the frequent item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item *p*
- Accumulate all of *transformed prefix paths* of item *p* to form *p*'s conditional pattern base

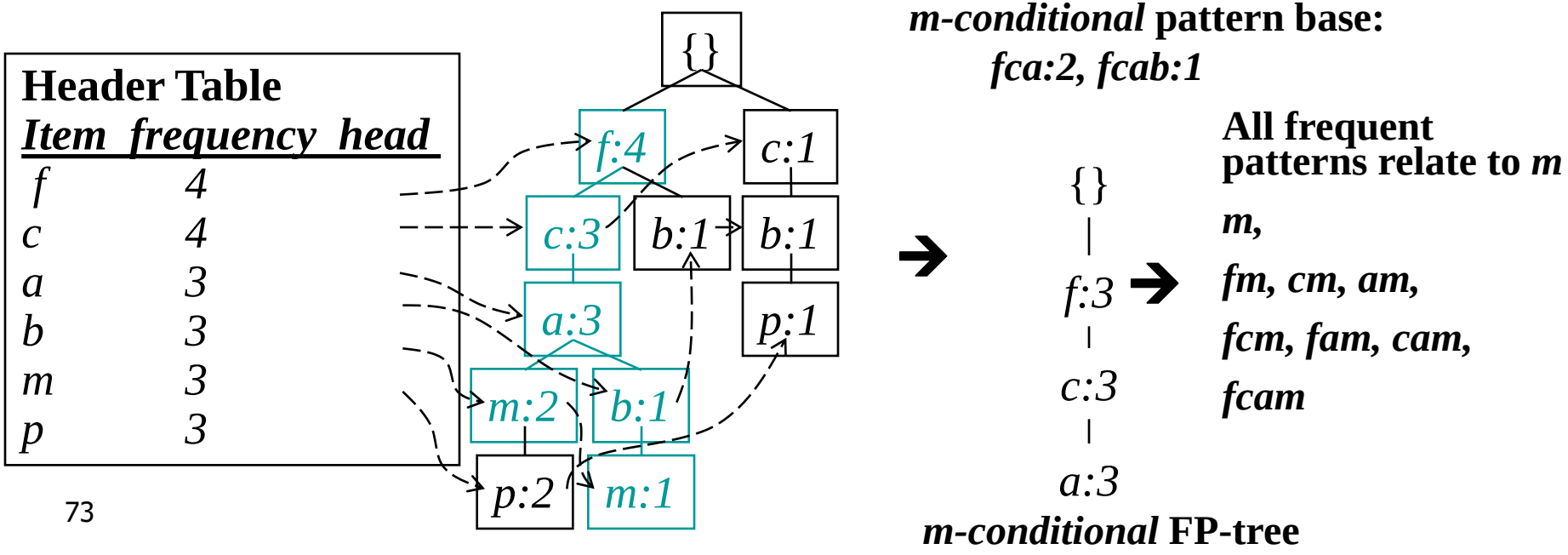


***Conditional* pattern bases**

item	*cond. pattern base*
c	***f:3***
a	***fc:3***
b	***fca:1, f:1, c:1***
m	***fca:2, fcab:1***
p	***fcam:2, cb:1***

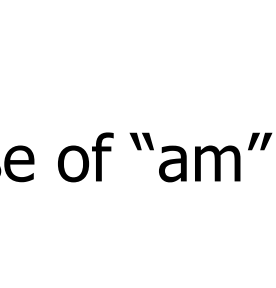
From Conditional Pattern-bases to Conditional FP-trees

- For each pattern-base
  - Accumulate the count for each item in the base
  - Construct the FP-tree for the frequent items of the pattern base



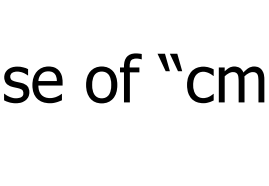
Recursion: Mining Each Conditional FP-tree

Cond. pattern base of "am": (fc:3)

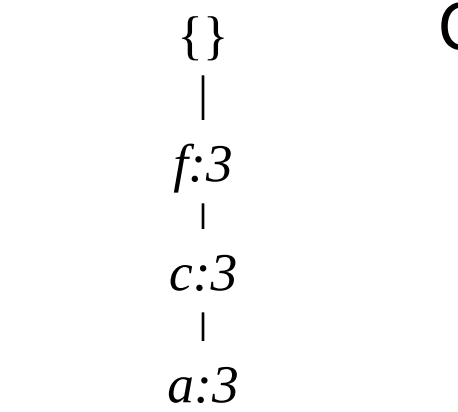


***am-conditional* FP-tree**

Cond. pattern base of "cm": (f:3)



***cm-conditional* FP-tree**



***m-conditional* FP-tree**

Cond. pattern base of "cam": (f:3)

***cam-conditional* FP-tree**

Presentation Outline

- Basic concepts
  - Frequent itemssets, Closed Itemsets, and Association Rules
- Frequent itemset mining methods
  - Apriori Algorithm
    - Generating Association Rules from Frequent Itemsets
    - Improving the Efficiency of Apriori
  - A Pattern-Growth Approach
  - **Mining Frequent Itemsets Using the Vertical Data Format**
  - Mining Closed and Max Patterns
- Which Patterns are interesting? Pattern Evaluation Methods
- Summary

IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 12, NO. 3, MAY/JUNE 2000

Scalable Algorithms for Association Mining

Mohammed J. Zaki, *Member*, *IEEE*

ECLAT: Mining by Exploring Vertical Data Format

(ECLAT: Equivalence CLAss Transformation)

- Transform the horizontally formatted data into the vertical format by scanning the data set once.
- The support count of an itemset is simply the length of the TID set of the itemset.
  - Starting with $k = 1$, the frequent k-itemsets can be used to construct the candidate $(k+1)$-itemsets based on the Apriori property.

Example:

- The minimum support count is 2.

- 10 intersections performed in total, which lead to eight nonempty 2-itemsets.
- Notice that because the itemsets {I1, I4} and {I3, I5} each contain only one transaction, they do not belong to the set of frequent 2-itemsets.
- Based on the Apriori property, a given 3-itemset is a candidate 3-itemset only if every one of its 2-itemset subsets is frequent. The candidate generation process here will generate only two 3-itemsets: {I1, I2, I3} and {I1, I2, I5}.
- By intersecting the TID sets of any two corresponding 2-itemsets of these candidate 3-itemsets, we can extract three itemsets.
- There are only two frequent 3-itemsets: {I1, I2, I3: 2} and {I1, I2, I5: 2}.

Table 6.3 The Vertical Data Format of the Transaction Data Set *D* of Table 6.1

itemset	*TID_set*
I1	{T100, T400, T500, T700, T800, T900}
I2	{T100, T200, T300, T400, T600, T800, T900}
I3	{T300, T500, T600, T700, T800, T900}
I4	{T200, T400}
I5	{T100, T800}

Table 6.4 2-Itemsets in Vertical Data Format

itemset	*TID_set*
{I1, I2}	{T100, T400, T800, T900}
{I1, I3}	{T500, T700, T800, T900}
{I1, I4}	{T400}
{I1, I5}	{T100, T800}
{I2, I3}	{T300, T600, T800, T900}
{I2, I4}	{T200, T400}
{I2, I5}	{T100, T800}
{I3, I5}	{T800}

Table 6.5 3-Itemsets in Vertical Data Format

itemset	*TID_set*
{I1, I2, I3}	{T800, T900}
{I1, I2, I5}	{T100, T800}

About ECLAT

- Advantages
  - Exploits apriori property
  - Scan the database only once
- Negative
  - the TID sets can be quite long, taking substantial memory space as well as computation time for intersecting the long sets.
- Notion of Diffset is proposed
  - Keep track of only Diffset
  - keeps track of only the differences of the TID sets of a *(k*-1*)*-itemset and a corresponding *k*-itemset.
    - For instance, in Example 6.6 we have {I1}={T100, T400, T500, T700, T800, T900} and {I1, I2}= {T100, T400, T800, T900}. The *diffset* between the two is *diffset*({I1, I2}, {I1}})= {T500, T700}.

Presentation Outline

- Basic concepts
  - Frequent itemssets, Closed Itemsets, and Association Rules
- Frequent itemset mining methods
  - Apriori Algorithm
    - Generating Association Rules from Frequent Itemsets
    - Improving the Efficiency of Apriori
  - A Pattern-Growth Approach
  - Mining Frequent Itemsets Using the Vertical Data Format
  - **Mining Closed and Max Patterns**
- Which Patterns are interesting? Pattern Evaluation Methods
- Summary

Basic Idea

- Naïve approach
  - Mine complete set of frequent item sets
  - Filter each frequent itemset which is a subset of another frequent itemset or caries the same support of the existing frequent itemset.
  - High complexity
- Recommended methodology
  - Search for closed itemsets directly during the mining process.
  - Prune the search space as soon as we find the closed itemsets.
- List of papers
  - J. Pei, J. Han & R. Mao. “CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets", DMKD’00.
  - R. Bayardo. Efficiently mining long patterns from databases. *SIGMOD’98*

Presentation Outline

- Basic concepts
  - Frequent itemssets, Closed Itemsets, and Association Rules
- Frequent itemset mining methods
  - Apriori Algorithm
    - Generating Association Rules from Frequent Itemsets
    - Improving the Efficiency of Apriori
  - A Pattern-Growth Approach
  - Mining Frequent Itemsets Using the Vertical Data Format
  - Mining Closed and Max Patterns
- **Which Patterns are interesting? Pattern Evaluation Methods**
- Summary

About Interesting patterns

- Association rule mining algorithms employ a support-confidence framework.
- Support-confidence
  - Extracts interesting rules
  - However, the framework also extracts uninteresting rules.
    - Low support and long patterns
- Support-confidence framework can supplemented with additional measures.

Strong Rules always may not be interesting

- A misleading association rule
- 10,000 transactions, 6000 include games, 7500 include videos, 4000 include games and videos
  - Buys(X, computer game) →buys(X, videos)
  - Support=40%, confidence=66%
  - Strong rule
- This rule is uninteresting, because the probability of purchasing videos is 75%
- Normally, computer games and videos are negatively correlated.
  - Purchase of one decreases the chances of purchase of the other.
- So support/confidence framework may be misleading.
- We need measures to measure real strength of the correlation and implication.

Lift: A correlation rule

- A→B (support, confidence, correlation)
- There are several correlation measures. One can choose any rule
- Lift is the simple correlation measure
- Lift(A,B)=P(AUB)/P(A)P(B)
  - < 1 negatively correlated. It means occurrence of one will lead to absence of the other
  - >1 Positively correlated. It means occurrence of one will lead to presence of other
  - =1 A and B are independent. There is no relationship
- Chi-square test is another correlation measure.
  - It tests the hypothesis that both are independent

Example

Table 6.6 2 × 2 Contingency Table Summarizing the Transactions with Respect to Game and Video Purchases

	game	$\overline{game}$	Σ_{row}
video	4000	3500	7500
$\overline{video}$	2000	500	2500
Σ_{col}	6000	4000	10,000

- Lift (game → videos)=0.40/0.60*0.75=0.89
- = <1 negatively correlated.
- Numerator: likelihood of purchasing both
- Denominator: likelihood of independent purchases.

Other interestingness measures

- All confidence
- Max confidence
- Kulczynski
- Cosine
- Each measure is only influenced by the supports of A,B, and AUB, but not by the total number of transactions.
- The value ranges between 0 to 1

Given two itemsets, A and B, the **all_confidence** measure of A and B is defined as

$$all\_conf(A, B) = \frac{sup(A \cup B)}{max\{sup(A), sup(B)\}} = min\{P(A|B), P(B|A)\}, \tag{6.9}$$

where $max\{sup(A), sup(B)\}$ is the maximum support of the itemsets A and B. Thus, $all\_conf(A, B)$ is also the minimum confidence of the two association rules related to A and B, namely, "$A \Rightarrow B$" and "$B \Rightarrow A$."

Given two itemsets, A and B, the **max_confidence** measure of A and B is defined as

$$max\_conf(A, B) = max\{P(A \mid B), P(B \mid A)\}. \tag{6.10}$$

The *max_conf* measure is the maximum confidence of the two association rules, "$A \Rightarrow B$" and "$B \Rightarrow A$."

Given two itemsets, A and B, the **Kulczynski** measure of A and B (abbreviated as **Kulc**) is defined as

$$Kulc(A, B) = \frac{1}{2}(P(A|B) + P(B|A)). \tag{6.11}$$

It was proposed in 1927 by Polish mathematician S. Kulczynski. It can be viewed as an average of two confidence measures. That is, it is the average of two conditional probabilities: the probability of itemset B given itemset A, and the probability of itemset A given itemset B.

Finally, given two itemsets, A and B, the **cosine** measure of A and B is defined as

$$\begin{aligned} cosine(A, B) &= \frac{P(A \cup B)}{\sqrt{P(A) \times P(B)}} = \frac{sup(A \cup B)}{\sqrt{sup(A) \times sup(B)}} \\ &= \sqrt{P(A|B) \times P(B|A)}. \end{aligned} \tag{6.12}$$

Which one is better?

- We introduced six measures? Which one is best in assessing the pattern relationship?

Table 6.8 2 × 2 Contingency Table for Two Items

	milk	$\overline{milk}$	Σ_{row}
coffee	mc	$\overline{m}c$	c
$\overline{coffee}$	$m\overline{c}$	$\overline{mc}$	$\overline{c}$
Σ_{col}	m	$\overline{m}$	Σ

Table 6.9 Comparison of Six Pattern Evaluation Measures Using Contingency Tables for a Variety of Data Sets

Data Set	mc	$\overline{m}c$	$m\overline{c}$	$\overline{mc}$	χ^2	*lift*	*all_conf.*	*max_conf.*	*Kulc.*	*cosine*
D_1	10,000	1000	1000	100,000	90557	9.26	0.91	0.91	0.91	0.91
D_2	10,000	1000	1000	100	0	1	0.91	0.91	0.91	0.91
D_3	100	1000	1000	100,000	670	8.44	0.09	0.09	0.09	0.09
D_4	1000	1000	1000	100,000	24740	25.75	0.5	0.5	0.5	0.5
D_5	1000	100	10,000	100,000	8173	9.18	0.09	0.91	0.5	0.29
D_6	1000	10	100,000	100,000	965	1.97	0.01	0.99	0.5	0.10

- M and C
  - are positively correlated in D1 and D2
  - Are negatively associated in D3
  - Neutral in D4
- Lift and chi-square produce different results for D1 and D2 because of $\overline{mc}$
- In real world applications $\overline{mc}$ is huge and unstable.
- So a good measure should not be effected by $\overline{mc}$
- D3: other measures are OK but lift and chisquare contradict
- D4, lift and chisquare show high correlation, others indicate neutral

- Reason for bad performance of lift and chi square
  - Null transactions: A null transaction does not contain the items being examined
  - The measure is strongly influenced by the transactions which do not contain m and c.
- A measure should be independent of null transactions

- Null-invariant measure: A measure is null-invariant if it value is free from the influence of null transactions

Which Null-Invariant Measure Is Better?

- IR (Imbalance Ratio): to assess the imbalance of two itemsets A and B in rule implications

$$IR(A,B) = \frac{|sup(A) - sup(B)|}{sup(A) + sup(B) - sup(A \cup B)}$$

- Suggestion: Kulczynski and Imbalance Ratio (IR) together present a clear picture for all the three datasets D_4 through D_6
  - D_4 is balanced & neutral
  - D_5 is imbalanced & neutral
  - D_6 is very imbalanced & neutral

Data	mc	$\overline{m}c$	$m\overline{c}$	$\overline{mc}$	*all_conf.*	*max_conf.*	*Kulc.*	*cosine*	IR
D_1	10,000	1,000	1,000	100,000	0.91	0.91	0.91	0.91	0.0
D_2	10,000	1,000	1,000	100	0.91	0.91	0.91	0.91	0.0
D_3	100	1,000	1,000	100,000	0.09	0.09	0.09	0.09	0.0
D_4	1,000	1,000	1,000	100,000	0.5	0.5	0.5	0.5	0.0
D_5	1,000	100	10,000	100,000	0.09	0.91	0.5	0.29	0.89
D_6	1,000	10	100,000	100,000	0.01	0.99	0.5	0.10	0.99

Presentation Outline

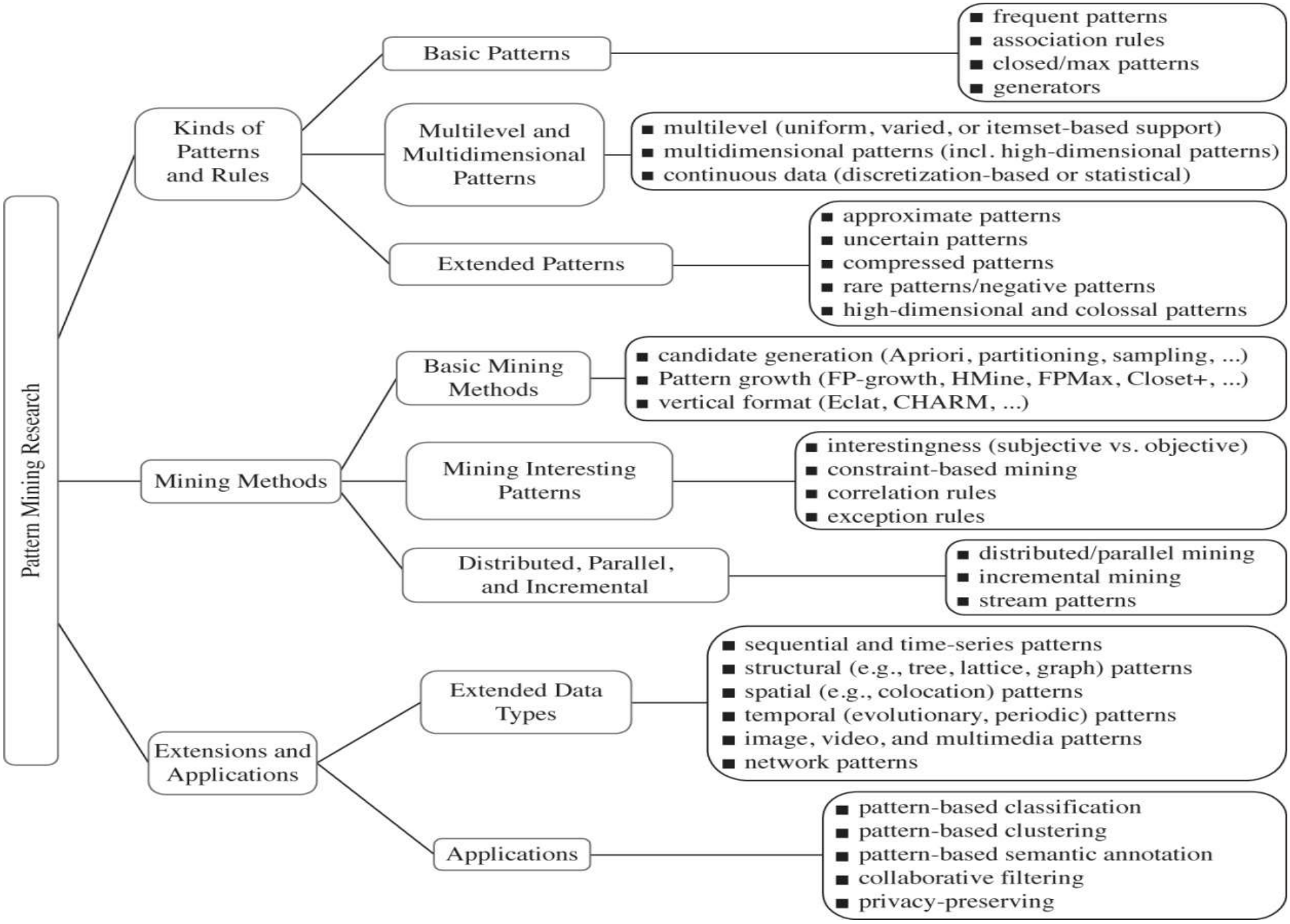
- Basic concepts
  - Frequent itemssets, Closed Itemsets, and Association Rules
- Frequent itemset mining methods
  - Apriori Algorithm
    - Generating Association Rules from Frequent Itemsets
    - **Improving the Efficiency of Apriori**
  - A Pattern-Growth Approach
  - Mining Frequent Itemsets Using the Vertical Data Format
  - Mining Closed and Max Patterns
- Which Patterns are interesting? Pattern Evaluation Methods
- Summary

Summary

- Frequent patterns: association between items
  - Application: DSSs, Market basket analysis
- Exponential problem
- Apriori algorithm
- Frequent pattern growth algorithm
- Eclat: extracting with vertical format
- Null-invariant measures

Advanced Pattern Mining

P. Krishna Reddy, IIIT Hyderabad



Pattern Mining: A Road Map

Outline

- Pattern Mining: A Road Map
- **Coverage Pattern Mining**
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- Multidimensional mining
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary

Discovering Coverage Patterns for Banner Advertisement Placement

P. Gowtham Srinivas, P. Krishna Reddy, S. Bhargav,

R. Uday Kiran, and D. Satheesh Kumar

International Institute of Information Technology

Hyderabad (IIIT-H) , India

Presented by P.Krishna Reddy

E-mail: pkreddy@iiit.ac.in

Outline

- **Introduction**
- Coverage patterns
- Mining Coverage Patterns
- Experimental Results
- Conclusions and future work

Introduction

- **Online advertising** is a form of promotion that uses the Internet to deliver marketing messages to attract customers.
  - provides sustainability for the e-commerce enterprises
  - a key factor in the growth of internet economy.
- There are three modes of online advertising:
  - Banner Advertising.
    - hyperlinked to the advertisers primary page or one with more information about the specific product or service advertised.
  - Contextual advertising
    - Relevant advertisements are assigned to a web page based on the content of the web page.
  - Sponsored search advertising
    - Advertisements are placed alongside with the search results.

About Banner Advertising

- Banner advertising is one of the dominant modes of online advertising.
- In this paper, we proposed a notion of coverage patterns to help to place the banner advertisements in a better manner.
- Three entities are involved in banner advertising: advertiser, publisher and visitor.
  - An advertiser is interested in endorsing products through banner advertisements.
  - A publisher manages a web site or an advertisement network that sells banner advertisement space.
  - Finally, a visitor visits the web pages of a web site which contains banners.

Banner advertisements on www.buy.com

Problem Description

- Goal of advertiser
  - spreading advertisement to a certain percentage of people visiting a web site.
- Goal of the publisher
  - efficiently sell the advertising space available in the web pages of a web site and meeting the demands of multiple advertisers.
- Opportunity:
  - For a given web site and period, one can analyse the visitors' behaviour by processing the transactions generated based on click stream dataset and identify the sets of web pages that cover a given percentage of visitors' population.
- Research Issue:
  - Investigate the approaches for discovering the sets of web pages which can cover a given percentage of visitors' population by analyzing the click stream transactions

Contributions

- We have proposed methodology to discover coverage patterns from transactional databases.
  - Given a set of data items, a coverage pattern is a set of non-overlapping data items covered by a certain percentage of transactions.
  - An Apriori-like algorithm called CMine is proposed for mining coverage patterns.

Related Work

- Coverage patterns
  - The notion of coverage is being used for solving the set cover problem [2] in set theory and node cover problem [3] in graphs respectively.
  - In [4], the notion of coverage and overlap is used to examine the creation of a tag cloud for exploring and understanding a set of objects.
  - In [5], the notion of coverage and overlap is used to solve the problem of topical query decomposition.
- Online Advertisement
  - Most of the research work on online advertisement has been focused on auction models [7], keyword or phrase identification based on user queries [8], contextual advertising [9] and allocation and scheduling of advertisements [10].
- To our knowledge, not much amount of research work has been carried out on improving the options offered by the publisher to the advertisers

Advantages

- We have proposed a different kind of knowledge patterns.
- The proposed patterns can be employed in improving the performance of several applications such as banner advertisements.
  - help the advertiser by making his advertisement visible to a certain percentage of web site visitors.
  - Ensuring the publisher to meet the demands of multiple advertisers by considering several groups of potential pages.

Outline

- Introduction
- **Coverage patterns**
- Mining Coverage Patterns
- Experimental Results
- Conclusions and future work

Coverage Patterns and Banner Advertisement

- We identify the issue of banner advertisement placement as one of the potential application of coverage patterns.
  - For a given e-commerce web site, transactions generated from click stream dataset can be used to identify the sets of web pages that cover a given percentage of visitors' population.
  - We consider transactions generated from click stream data of a web site. However, the model can be extended to any transactional data set.

Basic Terminology

- Let W = {w1,w2, · · · , wn} be a set of identifiers of web pages and D be a set of transactions, where each transaction T is a set of web pages such that $T \subseteq W$.
- Associated with each transaction is a unique transactional identifier called TID.
- A set of web pages in W i.e., X = {wp, ….. ,wq, wr},

  $1 \leq p \leq q \leq r \leq n$, is called a pattern.
- A pattern containing ‘k’ number of web pages is called a k-pattern. In other words, the length of k-pattern is k.
- The percentage of transactions in D that contain the web page $wi \in W$ is known as the “relative frequency of a web page $wi \in W$” and denoted as RF(wi).
- Let $|T^{wi}|$ indicates the total number of transactions that contain wi. The relative frequency of wi is denoted as RF(wi). That is, $RF(wi) = (|T^{wi}|)/|D|$.

Proposed Model…

TRANSACTIONAL DATABASE

Table 1.

TID	Web pages	TID	Web pages
1	a, b, c	6	b, d
2	a, c, e	7	b, d
3	a, c, e	8	b, e
4	a, c, d	9	b, e
5	b, d, f	10	a, b

Item	Relative Frequency (RF)
a	0.5
b	0.7
c	0.4
d	0.4
e	0.4
f	0.1

- {a, b} is a pattern. Since there are two web pages in this pattern it is a 2-pattern.

Frequent Page

- Note that from the advertisement point of view the pages that are visited by more number of users are interesting. We capture this aspect with the notion of frequent page.

- The frequent web pages are web pages which have relative frequency no less than the user-specified threshold value, called minimum relative frequency(*minRF*) . That is, $RF(w_i) \geq minRF.$

- Example 2: Continuing with the example, the relative frequency of 'a' i.e., $RF(a) = |T^a|/|D| = 5/10 = 0.5$. If the user-specified minRF = 0.5, then 'a' is called a frequent web page because $RF(a) \geq minRF$.

Coverage Set

- The coverage set is a set of transactions.
  - Notion: How many users visit at least one web page in the set.
- Coverage set of a pattern X = {w_p, ….. ,w_q, w_r}, $1 \leq p \leq q \leq r \leq n$:
  - The set of distinct TIDs containing at least one web page of X is called the coverage set of pattern X and is denoted as CSet(X). Therefore, $CSet(X) = \{T^{w_p} \cup \cdots \cup T^{w_q} \cup T^{w_r}\}$.

TID	Web pages	TID	Web pages
1	a, b, c	6	b, d
2	a, c, e	7	b, d
3	a, c, e	8	b, e
4	a, c, d	9	b, e
5	b, d, f	10	a, b

Example: The set of tids containing the web page `a' i.e., $T^a = \{1,2,3,4,10\}$. Similarly, $T^b = \{1,5,6,7,8,9,10\}$.

The coverage set of {a,b} i.e., $CSet(\{a,b\}) = \{1,2,3,4,5,6,7,8,9,10\}$.

Note: In case of frequent patterns, the support of {a,b} ={1,10}=2

Coverage Support

- A pattern will be interesting if its coverage set contains more than a threshold number of transactions. This aspect is captured through the notion of coverage support.
- ***Coverage-support of a pattern X*** *: The ratio* of size of coverage set of *X* to the transactional database size is called the coverage-support of pattern *X and denoted as CS(X).*
  - $CS(X) = |CSet(X)| / |D|$.
- For a pattern *X,* $CS(X) \in [0, 1]$.
  - If CS(X) = 0, no single web page of X has appeared in the entire transactional database.
  - If CS(X) = 1, every transaction in T contains at least one web page wj $\in$ X.
- Example: Coverage support of {a,b} i.e., CS({a,b}) = |CSet({a,b})|/|D| = 10/10=1.

Overlap ratio

- Adding other web pages in particular web pages co-occurring with any of the web pages belonging to pattern X may not increase the coverage support, significantly.
  - Due to co-occurrence, there is an overlap of coverage set of X and coverage set of new single web page.
  - The same users visit the new web page.
  - Such a pattern can be uninteresting to the advertiser because, the advertisement will be displayed to the same user multiple times.
- Example: T^a={1,2,3,4,10}. T^b={1,5,6,7,8,9,10}, and T^c={1,2,3,4}. CS(a)=0.5, CS(b)=0.7, CS(c)=0.4.
  - We can note that CS(a,c)=5/10=0.5. However, this pattern is uninteresting as the pattern `c' has not increased coverage-support of the pattern `a'.
  - We can note that CS(a,b)=1.As compared to {a,c}, the pattern {a,b} is interesting because, the web page `b' has increased the coverage support of a pattern.

Overlap Ratio

- The set of pages will have more coverage support if there is minimum intersection of coverage set of individual pages in the set. We capture this aspect with overlap ratio.
- ***Overlap ratio of a pattern OR(X)*** : Overlap ratio of a pattern X = $\{w_p, \ldots\ldots, w_q, w_r\}$, where $1 \leq p \leq q \leq r \leq n$ and $T^{w_p} >= \cdots >= T^{w_q} >= T^{w_r}$, is ratio of the number of transactions common in $X - \{w_r\}$ and $\{w_r\}$ to the number of transactions in w_r.
- $OR(X) = \dfrac{|CSet(X-\{w_r\}) \cap CSet(\{w_r\})|}{CSet(\{wr\})}$

…Overlap Ratio

- For a pattern X, OR(X) ∈ [0, 1]. If OR(X) = 0, there exists no common transactions between $X - \{w_r\}$ and $\{w_r\}$.
- If OR(X) = 1, then w_r has occurred in the transactions where at least one web page $w_j \in (X - \{w_r\})$ has occurred.
- Continuing with the example,
  - The OR({a,b}) = |CSet(b) ∩ CSet(a)|/|CSet(a)| = 2/5 = 0.4.
- Note that a coverage pattern is interesting if it has high coverage support and low overlap ratio.
  - As a result an advertisement is exposed to more number of users by reducing repetitive display of the advertisement. The definition of coverage pattern is as follows

Coverage Pattern

- ***Coverage pattern X*** : A pattern X ={w_p, …. , w_q, w_r}, where 1 ≤ p ≤ q ≤ r ≤ n, is said to be a coverage pattern if CS(x) ≥ minCS and OR(X) ≤ maxOR and RF(w_i) ≥ minRF ∀ w_i ∈ X.

  A coverage pattern *X* having CS(X) = a% and OR(X) = b% is expressed as

  X [CS = a%, OR = b%]

- If minRF=0.4, minCS=0.7 and maxOR=0.5, then the pattern {a,b} is a coverage pattern. It is because RF(a) >= minRF, RF(b) >= minRF, CS({a,b}) >= minCS and OR({a,b}) <= maxOR. This pattern is written as follows:
  - {a,b} [CS=1 (=100%), OR= 0.4 (=40%)]

Problem Statement

- Given a transactional database *D,* set of web pages *W* and user-specified minimum relatively frequency (*minRF),* minimum coverage support *(minCS)* and maximum overlap ratio (*maxOR),* discover complete set of coverage patterns such that
  i. If X is a coverage 1-pattern (i.e., k = 1), then $RF(w_i) \geq minRF$, $w_i \in X$.
  ii. Otherwise (i.e., when k > 1), each coverage pattern X

  must have $CS(X) \geq minCS$, $OR(X) \leq maxOR$ and

  $RF(w_i) \geq minRF$, where $w_i \in X$.

Outline

- Introduction
- Coverage patterns
- **Mining Coverage Patterns**
- Experimental Results
- Conclusions and future work

Extracting Coverage Patterns

- Naïve approach
  - Extract all possible patterns (leads to combinatorial explosion)
  - Each pattern in CP is added to the coverage pattern set if it satisfies minCS; minRF and maxOR constraints.
- Opportunity
  - The search space can be reduced if the coverage pattern satisfies downward closure property on either coverage support or overlap ratio.

Extracting Coverage Patterns

- Coverage support does not satisfy downward closure property. That is, although a pattern satisfies minCS, it is not necessary that all its non-empty subsets will also satisfy minCS value.
  - Consider the patterns {a}, {e} and {a,e}. The coverage supports of these patterns are 0.5, 0.4 and 0.7, respectively. If the user-specified minCS=0.7, then the pattern {a,e} satisfies minCS value. However, its non-empty subsets do not satisfy minCS value.
- The parameter overlap ratio also does not satisfy downward closure property if a pattern is considered as an unordered set of web pages.
- The Overlap ratio satisfies downward closure property if a pattern is an ordered set, where web pages are sorted in descending order of their frequencies. This property is known as the sorted closure property.

Extracting Coverage Patterns

- If $X \subset Y$, then $CSet(X) \subset= CSet(Y)$.
- Sorted closure property: Let $X = \{w_p, \ldots, w_q, w_r\}$, where $1 \leq p \leq q \leq r \leq n$ be a pattern such that $T^{w_p} >= \cdots >= T^{w_q} >= T^{w_r}$. If $OR(X) <= maxOR$, all its non-empty subsets containing w_r and having size $k >= 2$ will also have overlap ratio less than or equal to maxOR.
- Rationale: Let w_a, w_b and w_c be the web pages having $RF(w_a) \geq RF(w_b) \geq RF(w_c)$. If $OR(w_a \cup w_c) > maxOR$, then $OR(\{w_a \cup w_b\} \cup w_c) > maxOR$ because from above property

  $|CSet(w_a) \cap CSet(w_c)| / |CSet(w_c)| <= CSet(\{w_a \cup w_b\}) \cap CSet(w_c)/CSet(w_c)$.
- (Non-overlap pattern X.) A pattern X is said to be *non-overlap* if $OR(X) \leq maxOR$ and $RF(w_i) \geq minRF \ \forall w_i \in X$.

Extracting Coverage Patterns

- Every coverage pattern is a non-overlap pattern
- However, every non-overlap pattern may not be a coverage pattern.
- The sorted closure property of non-overlap patterns is used for minimizing the search space while mining complete set of coverage patterns by designing an algorithm similar to the Apriori algorithm .

Proposed Algorithm

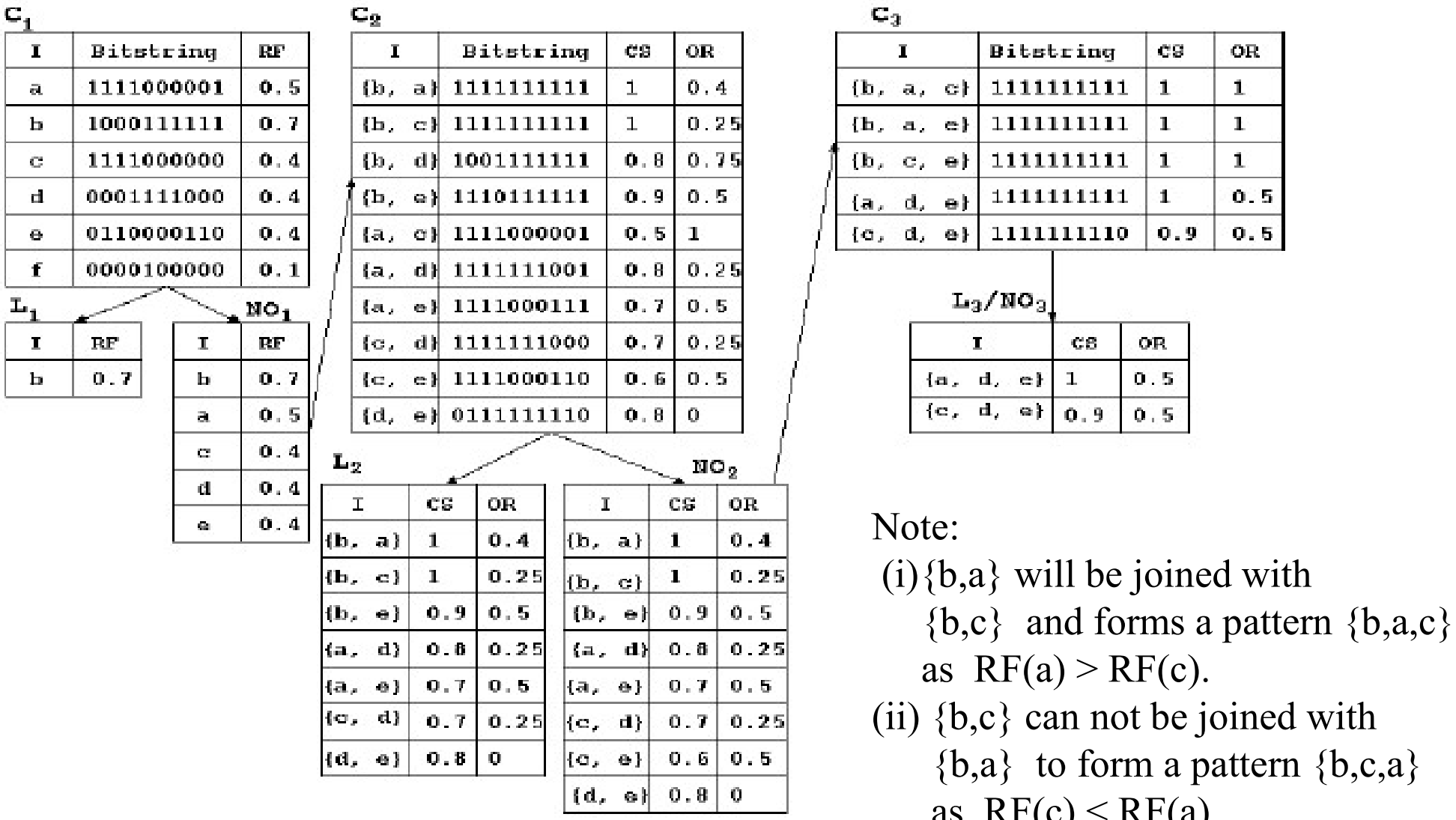
- The proposed algorithm *CMine* employs *level-wise* search to discover the complete set of coverage patterns.
- In *level-wise* search*, k*-patterns are used to explore *(k + 1)*-patterns.
- Notations

F	Set of Frequent items
C_k	Set of Candidate K-patterns
L_k	Set of Coverage K-patterns
NO_k	Set of non-overlap K-patterns

CMine Algorithm

- Step 1: In the first scan of the database, ***CMine*** discovers set of all frequent items (denoted as *F)* and coverage 1-patterns (denoted as L_1). Next, items in NO_1 are sorted in descending order of their frequencies.
- Step 2: Using NO_1 as a seed set, candidate patterns C_2 are generated.
- Step 3: From C_2, the patterns that satisfy *minCS* and *maxOR* are generated as *coverage* 2-patterns, L_2.
- **Simultaneously, all candidate 2-patterns that satisfy *maxOR* are generated as *non-overlap* 2-patterns NO_2.**
- Step 4 : C_3 *is* generated by combining $NO_2 \bowtie NO_2$.
- Step 5 : From C_3, L_3 *and* NO_3 are discovered.
- The above process is repeated until no new coverage pattern is found, or no new candidate pattern can be generated.

CMine Algorithm Example



C_1

I	Bitstring	RF
a	1111000001	0.5
b	1000111111	0.7
c	1111000000	0.4
d	0001111000	0.4
e	0110000110	0.4
f	0000100000	0.1

L_1

I	RF
b	0.7

NO_1

I	RF
b	0.7
a	0.5
c	0.4
d	0.4
e	0.4

C_2

I	Bitstring	CS	OR
{b, a}	1111111111	1	0.4
{b, c}	1111111111	1	0.25
{b, d}	1001111111	0.8	0.75
{b, e}	1110111111	0.9	0.5
{a, c}	1111000001	0.5	1
{a, d}	1111111001	0.8	0.25
{a, e}	1111000111	0.7	0.5
{c, d}	1111111000	0.7	0.25
{c, e}	1111000110	0.6	0.5
{d, e}	0111111110	0.8	0

L_2

I	CS	OR
{b, a}	1	0.4
{b, c}	1	0.25
{b, e}	0.9	0.5
{a, d}	0.8	0.25
{a, e}	0.7	0.5
{c, d}	0.7	0.25
{d, e}	0.8	0

NO_2

I	CS	OR
{b, a}	1	0.4
{b, c}	1	0.25
{b, e}	0.9	0.5
{a, d}	0.8	0.25
{a, e}	0.7	0.5
{c, d}	0.7	0.25
{c, e}	0.6	0.5
{d, e}	0.8	0

C_3

I	Bitstring	CS	OR
{b, a, c}	1111111111	1	1
{b, a, e}	1111111111	1	1
{b, c, e}	1111111111	1	1
{a, d, e}	1111111111	1	0.5
{c, d, e}	1111111110	0.9	0.5

L_3/NO_3

I	CS	OR
{a, d, e}	1	0.5
{c, d, e}	0.9	0.5

Note:

(i) {b,a} will be joined with {b,c} and forms a pattern {b,a,c} as RF(a) > RF(c).

(ii) {b,c} can not be joined with {b,a} to form a pattern {b,c,a} as RF(c) < RF(a).

minRF = 0.4, minCS = 0.7, maxOR = 0.5 for database given in Table 1.

Outline

- Introduction
- Coverage patterns
- Mining Coverage Patterns
- **Experimental Results**
- Conclusions and future work

Experimental Results

- For experimental purposes we have chosen two kinds of datasets: (i)real world dataset and (ii)synthetic-dataset.
- The four real world datasets considered for the experiments are described below.
  - Kosarak dataset is a sparse dataset with 9,90,002 number of transactions containing 41,270 distinct items.
  - MSNBC dataset contains data from Internet Information Server (IIS) logs for msnbc.com and news-related portions of msn.com for the entire day of September, 28, 1999 . Requests are at the level of page category. The number of categories are 17 and the number of transactions are 989,818
  - Mushroom dataset is a dense dataset containing 8,124 transactions and 119 distinct items.
  - BMS-POS dataset contains click stream data of a dotcom company . The dataset contains 515,597 number of transactions and 1656 distinct items.
- The synthetic dataset T40I10D100K dataset which is generated by the dataset generator [9]. The dataset contains 100000 transactions and 941 distinct items.
- The CMine algorithm was written in Java and run with Windows XP on a 2.66 GHz machine with 2GB memory.

Usefulness of coverage patterns

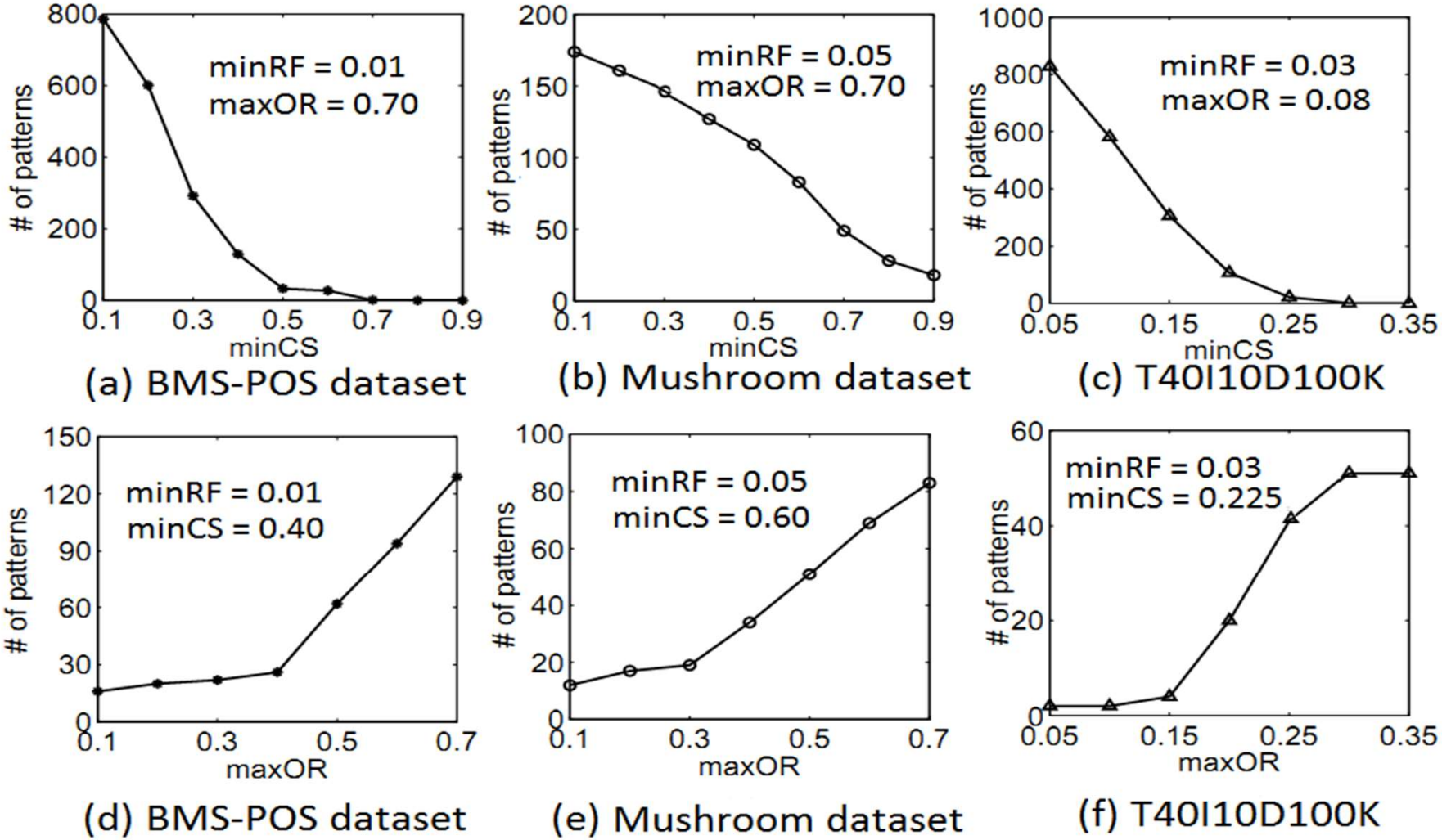
- The names of web page categories involved in MSNBC are
  - "frontpage", "news", "tech", "local", "opinion", "on-air", "misc", "weather", "health", "living", "business", "sports", "summary", "bbs" (bulletin board service), "travel", "msn-news", and "msn-sports".

S.No	Coverage Pattern	CS
1	{local, misc, front page}	0.42
2	{news, health, front page}	0.43
3	{tech, opinion, front page}	0.41
4	{on-air, news, misc}	0.40
5	{tech, weather, on-air}	0.41
6	{sports, misc, opinion}	0.43

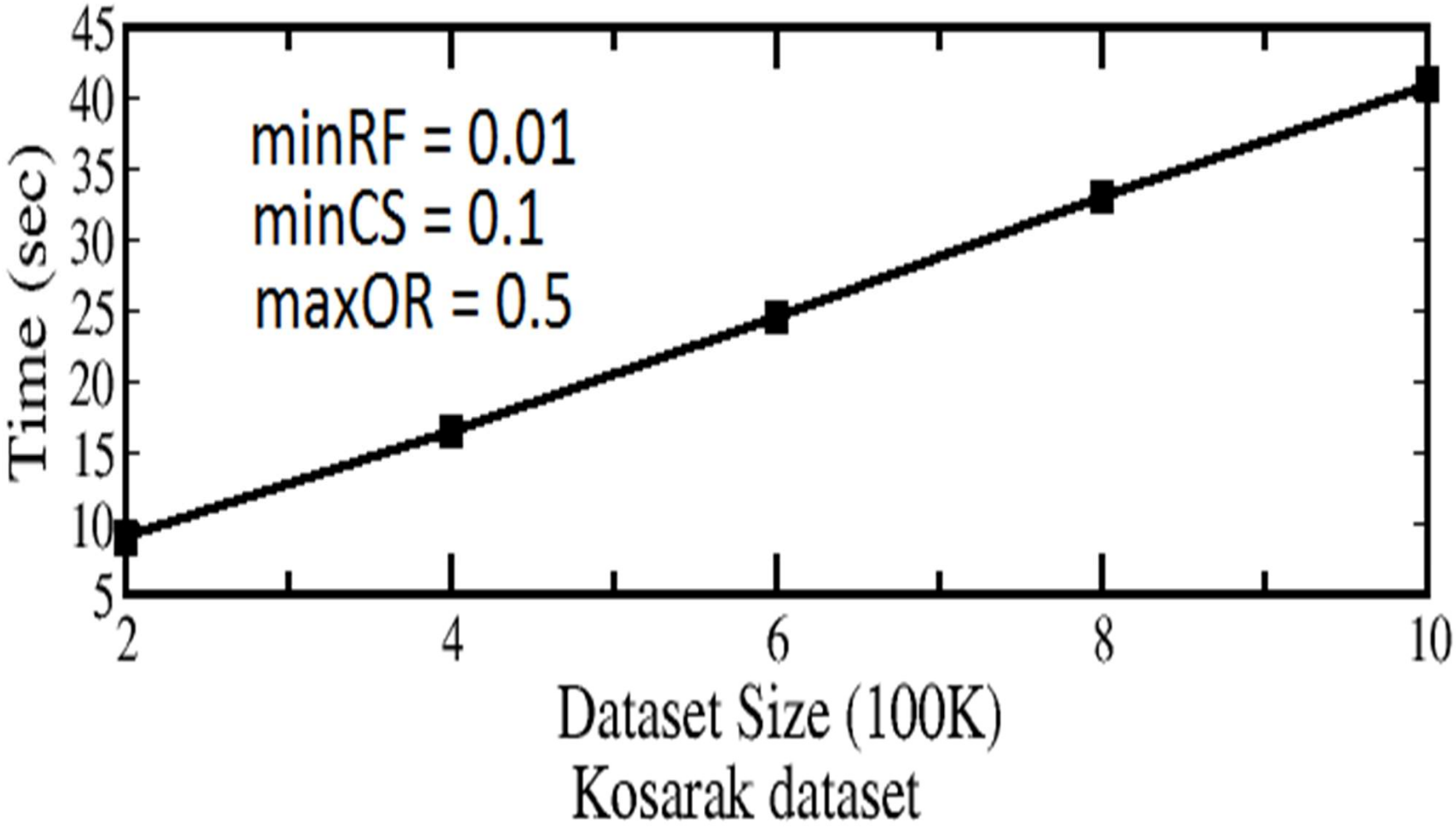
- Advantage:

Proposed approach provides flexibility to the publisher to meet the demands of multiple advertisers by considering different sets of web pages.

Coverage pattern generation



Scalability of CMine Algorithm



Conclusions and Future Work

- Proposed a framework of data mining pattern called "coverage pattern"
- Proposed a methodology to extract the same from transactional databases.
- Through experimental results, we have shown that the proposed model and methodology can effectively discover coverage patterns.
- The coverage patterns help the publisher to meet the demands of multiple advertisers.

Conclusions and Future Work

- We are going to investigate how both frequent and coverage pattern knowledge can be used for efficient banner advertisement placement.
- How the content of the web page and search query can be exploited to explore content specific coverage patterns.
- The notion of coverage patterns can be extended to other domains like bio-informatics for extracting potential knowledge patterns.

Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- **Multilevel Pattern Mining**
- Quantitative Pattern Mining
- Mining Multidimensional Associations
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary

About Multilevel Pattern Mining

- Applications require association rules at many levels of abstraction
  - “80 percent of customers that purchase milk may also purchase Bread”
  - “75 percent of people buy wheat bread if they buy “milk(2%)”
- The association relationship in the latter statement is expressed at a lower level of abstraction but carries more specific and concrete information than that in the former.
- Therefore, a data mining system should provide efficient methods for mining multiple-level association rules.

Why Multiple-Level Association Rules?

TID	items
T1	{m1, b2}
T2	{m2, b1}
T3	{b2}

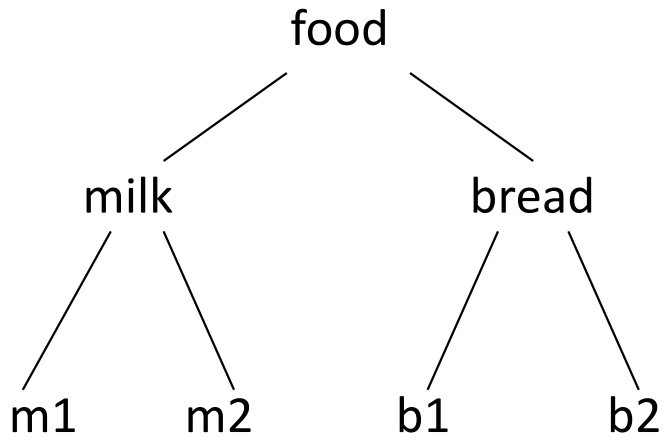
Frequent itemset: {b2}

Association rules: none

Is this database useless?

Why Multiple-Level Association Rules?

What if we have this abstraction tree?



TID	items
T1	{milk, bread}
T2	{milk, bread}
T3	{bread}

minisup = 50% miniconf = 50%

Frequent itemset: {milk, bread}

A.A rules: milk <=> bread

Why Multiple-Level Association Rules?

- Sometimes, at primitive data level, data does not show any significant pattern. But there are useful information hiding behind.

- The goal of Multiple-Level Association Analysis is to find the hidden information in or between levels of abstraction

Input

- Input to Multi-Level Association Rule Mining
  - 1) data at multiple levels of abstraction, and
  - 2) efficient methods for multiple-level rule mining.
- The first requirement can be satisfied by providing concept taxonomies from the primitive level concepts to higher levels.
- In many applications, the taxonomy information is either
  - stored implicitly in the database
  - provided by experts or users,
  - or computed by applying some cluster analysis methods
- We assume that concept taxonomies exist.

Requirements in Multiple-Level Association Analysis

Two general requirements to do multiple-level association rule mining:

1) Provide data at multiple levels of abstraction. (a common practice now)

2) Find efficient methods for multiple-level rule mining.

Observation

- One choice
  - Direct application of the existing single-level association rule mining methods to multiple-level association mining.
  - For example, one may apply the Apriori algorithm to examine data items at multiple levels of abstraction under the same minimum support and minimum confidence thresholds.
- Apriori with single minimum support and minimum confidence may lead to some undesirable results.

Observation..

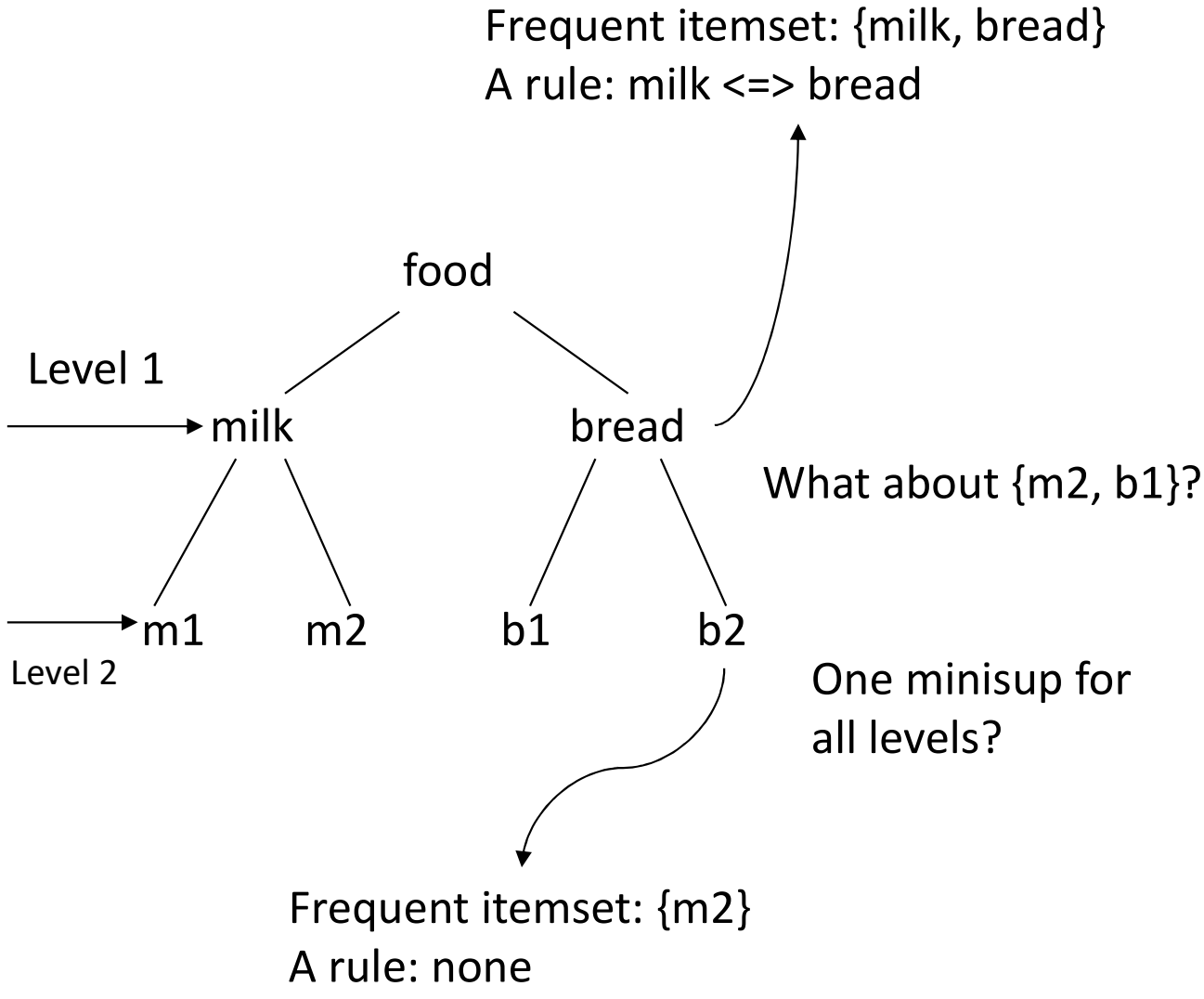
- Undesirable results with apriori
  - First, large support is more likely to exist at high levels of abstraction.
    - If one wants to find strong associations at relatively low levels of abstraction, the minimum support threshold must be reduced substantially, which may lead to the generation of many uninteresting associations at high or intermediate levels.

  - Second, since it is unlikely to find many strong association rules at a primitive concept level, mining strong associations should be performed at a rather high concept level. But, rules at high concept levels,
    - may often lead to the rules corresponding to prior knowledge and expectations, such as “milk->bread”, (which could be common sense),
    - or lead to some uninteresting attribute combinations if the minimum support is allowed to be rather small, such as **“toy -> milk**”,

Algorithm : observation

TID	items
T1	{milk, bread}
T2	{milk, bread}
T3	{bread}
T4	{milk, bread}
T5	{milk}

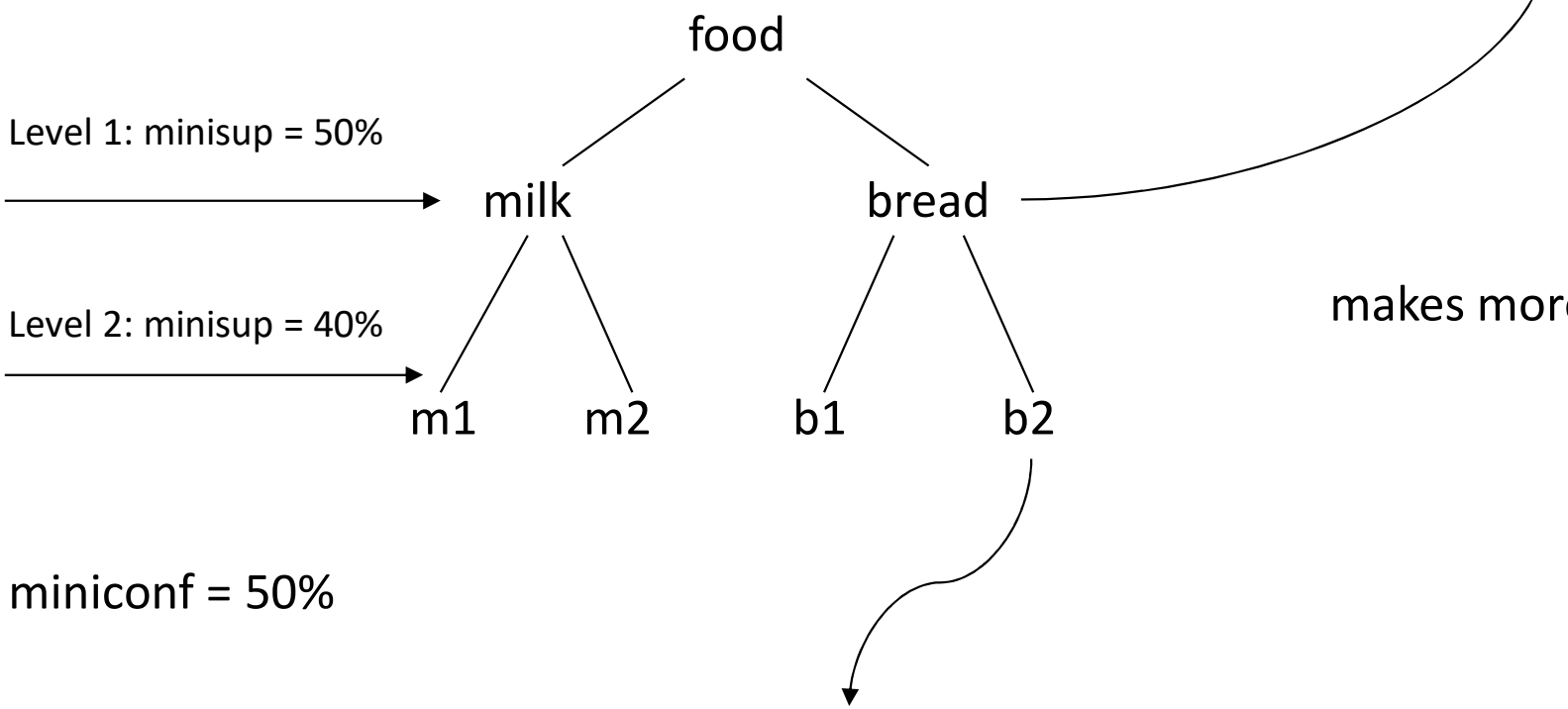
TID	items
T1	{m1, b2}
T2	{m2, b1}
T3	{b2}
T4	{m2, b1}
T5	{m2}



minisup = 50% miniconf = 50%

Algorithm : observation

Frequent itemset: {milk, bread}
A rule: milk <=> bread



makes more sense now

miniconf = 50%

Frequent itemset: { m2, b1, b2}
A rule: m2 <=> b1

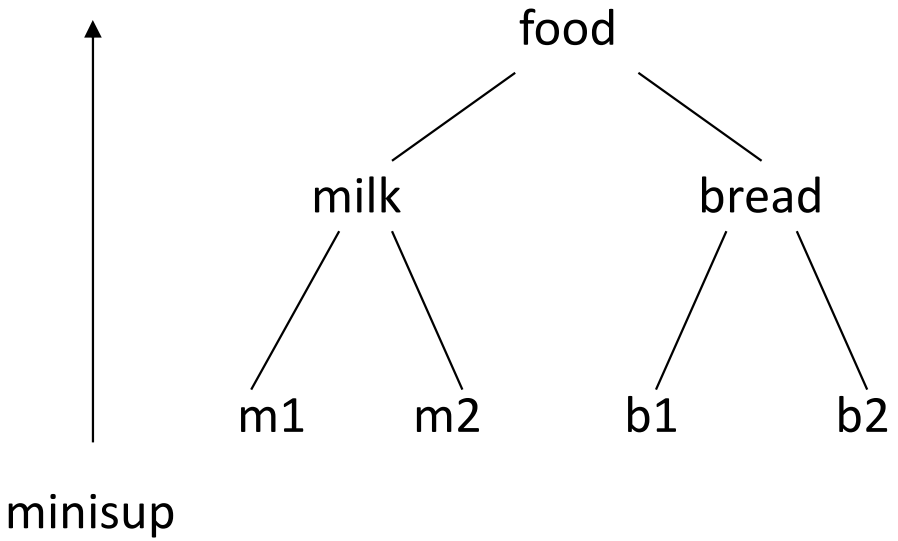
Algorithm: observation

Drawbacks to use only one minisup:

- If the minisup is too high, we are losing information from lower levels

- If the minisup is too low, we are gaining too many rules from higher levels, many of them are useless

Approach: ascending minisup on each level



Research Paper: Mining Multiple-level Association Rules in Large Databases

IEEE Transactions on Knowledge and Data Engineering, 1999

JIAWEI HAN and YONGJIAN FU

Multi-level Association Rules

We assume that the database contains: 1) an item data set which contains the description of each item in $\mathcal{I}$ in the form of $\langle A_i, description_i \rangle$, where $A_i \in \mathcal{I}$, and 2) a transaction data set, $\mathcal{T}$, which consists of a set of transactions $\langle T_i, \{A_p, \ldots, A_q\} \rangle$, where T_i is a transaction identifier and $A_i \in \mathcal{I}$ (for $i = p, \ldots, q$).

Definition 2.2. *A pattern A is **frequent** in set S at level l if the support of A is no less than its corresponding minimum support threshold σ'_l. A rule "$A \Rightarrow B/S$" is **strong** if, for a set S, each ancestor (i.e., the corresponding high-level item) of every item in A and B, if any, is frequent at its corresponding level, "$A \wedge B/S$" is frequent (at the current level), and the confidence of "$A \Rightarrow B/S$" is no less than minimum confidence threshold at the current level.*

Multi-level Association Rules..

- The definition 2.2
  - patterns to be examined at lower levels to be only those with large supports at their corresponding high levels (and thus avoids the generation of many meaningless combinations formed by the descendants of the infrequent patterns).
- For example, in a sales transaction data set,
  - if milk is a frequent pattern, its lower level patterns, such as 2 percent milk, will be examined;
  - whereas if fish is an infrequent pattern, its descendants, such as salmon, will not be examined further.

About the Method

- Used hierarchy information encoded transaction table
- Collect task relevant data and mine
- Encoding can be done during the collection

Encoding Method

- Purpose: To find multiple-level frequent item sets for mining strong association rules in a transaction database
- Input
  - T[1]: a hierarchy-information encoded transaction table of form <TID,Item-set>
  - minisup threshold for each level L in the form: (minsup[L])
- Output: Multiple-level frequent item sets
- Method: A top-down, progressively deepening process which collects frequent item sets at different concept levels as follows.

Algorithm: An Example

An entry of **sales_transaction** Table

Transaction_id	Bar_code_set
351428	{17325,92108,55349,88157,…}

A **sales_item** Description Relation

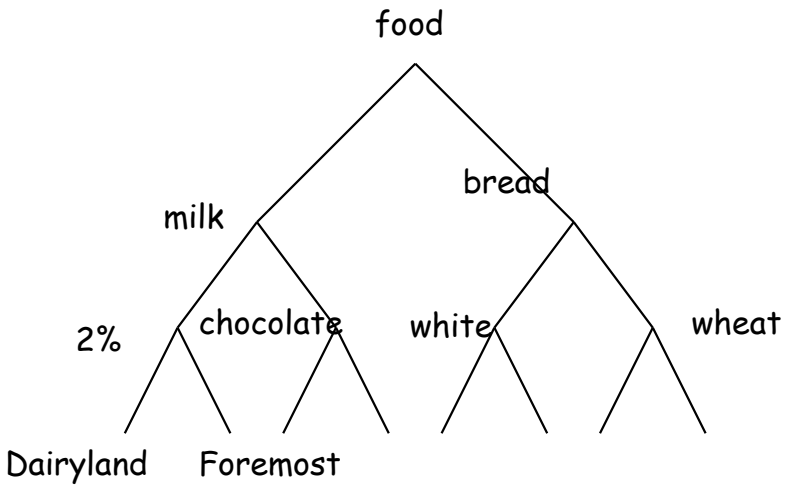
Bar_code	category	Brand	Content	Size	Storage_pd	price
17325	Milk	Foremost	2%	1ga.	14(days)	$3.89

Example 2.1. Let the query be to find multiple-level strong associations in the database in Table 1 for the purchase patterns related to category, content, and brand of the foods which can only be stored for less than three weeks.

Algorithm: An Example

Encode the database with layer information

GID	bar_code	category	content	brand
112	17325	Milk	2%	Foremost



First 1: implies milk

Second 1: implies 2% content

2: implies Foremost brand

Algorithm: An Example

Encoded Transaction Table:T[1]

TID	Items
T1	{111,121,211,221}
T2	{111,211,222,323}
T3	{112,122,221,411}
T4	{111,121}
T5	{111,122,211,221,413}
T6	{211,323,524}
T7	{323,411,524,713}

Algorithm: An Example

The frequent 1-itemset on level 1

L[1,1]

Level-1 minsup = 4

T[2]

Itemset	Support
{1**}	5
{2**}	5

only keep items
in L[1,1] from T[1]

TID	Items
T1	{111,121,211,221}
T2	{111,211,222}
T3	{112,122,221}
T4	{111,121}
T5	{111,122,211,221}
T6	{211}

L[1,2]

Itemset	Support
{1**,2**}	4

Use Apriori on each level

Algorithm: An Example

Level-2 minsup = 3

L[2,1]

Itemset	Support
{11*}	5
{12*}	4
{21*}	4
{22*}	4

L[2,2]

Itemset	Support
{11*,12*}	4
{11*,21*}	3
{11*,22*}	4
{12*,22*}	3
{21*,22*}	3

L[2,3]

Itemset	Support
{11*,12*,22*}	3
{11*,21*,22*}	3

Frequent Item Sets at Level 3

Level-3 minsup = 3

L[3,1]

Itemset	Support
{111}	4
{211}	4
{221}	3

L[3,2]

Itemset	Support
{111,211}	3

Only generate T[1] & T[2], all frequent itemsets after level 2 is generated from T[2]

E.g.

Level-1:

80% of customers that purchase milk also purchase bread.

milk ➔ bread with Confidence= 80%

Level-2:

75% of people who buy 2% milk also buy wheat bread.

2% milk ➔ wheat bread with Confidence= 75%

Multi-level Association: Flexible Support and Redundancy filtering

- Flexible min-support thresholds: Some items are more valuable but less frequent
  - Use non-uniform, group-based min-support
  - E.g., {diamond, watch, camera}: 0.05%; {bread, milk}: 5%; …
- Redundancy Filtering: Some rules may be redundant due to "ancestor" relationships between items
  - milk $\Rightarrow$ wheat bread [support = 8%, confidence = 70%]
  - 2% milk $\Rightarrow$ wheat bread [support = 2%, confidence = 72%]

  The first rule is an ancestor of the second rule
- A rule is *redundant* if its support is close to the "expected" value, based on the rule's ancestor

Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- **Quantitative Pattern Mining**
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary

Boolean Association Rules

Trans-ID	Items
1	A C D
2	B C E
3	A B C E
4	B E
5	A B C E

- The table has an attribute correspond to each item. A record correspond to each transaction.
- Value of attribute is “1” if it exists on record.
- All the attributes are Boolean.

Boolean Association Rules

TID	A	B	C	D	E
1	1	0	1	1	0
2	0	1	1	0	1
3	1	1	1	0	1
4	0	1	0	0	1
5	1	1	1	0	1

- Attribute has a value of “1” if the transaction contains the corresponding item; “0” otherwise.

Large Relational Tables in real-life.

- Relational tables in most business and scientific domains have richer attribute types.
  - Quantitative attributes (e.g. age, income)
  - Categorical attributes (e.g. zip code, marriage status, make of car)
- Can not apply existing methods to extract Boolean association rules to mine association rules over quantitative and categorical attributes.
  - This paper present techniques for discovering such rules.

Quantitative Association Rules: Example

RecordID	Age	Married	NumCars
100	23	No	1
200	25	Yes	1
300	29	No	0
400	34	Yes	2
500	38	Yes	2

Sample Rules	Support	Confidence
<age:30..39> and <married: yes> ==> <numCars:2>	40%	100%
<NumCars: 0..1> ==> <Married: No>	40%	66.70%

Mapping to Boolean Association Rules Problem

- Using <attribute: value> as new attribute, which has only Boolean values
- Categorical: The value of Boolean field corresponding to <attribute1, value1> would be “1”, if atribute1 had value 1 in the original record, and “0” otherwise.
- If the domain of values is large, partition the values into intervals and map each pair to Boolean attribute.

- Use any exiting algorithm a priori or FP Tree.

RecordID	Age: 20..29	Age: 30..39	Married: Yes	Married: No	NumCars: 0	NumCars: 1
100	1	0	0	1	0	1
200	1	0	1	0	0	1
300	1	0	0	1	1	0
400	0	1	1	0	0	0
500	0	1	1	0	0	0

First Problem with Direct Mapping

- **MinSup**: If the number of intervals for a quantitative attribute is large, the support for any single interval can be low. Some rules involving this attribute may not be found because they lack minimum support.

Second Problem with Direct Mapping

- **MinConf.** There is some information lost whenever we partition values into intervals. This information loss increases as the interval sizes become larger.

Example "*MinConf*" problem.

Rec-ID	Age: 20..29	Age: 30..39	Married: Yes	Married: No	NumCars: 0	NumCars: 1	NumCars: 2
100	1	0	0	1	0	1	0
200	1	0	1	0	0	1	0
300	1	0	0	1	1	0	0
400	0	1	1	0	0	0	1
500	0	1	1	0	0	0	1

- The rule

  < NumCars: 0> => <Married: No> has 100% confidence.

Example "*MinConf*" problem. (con't)

Rec-ID	Age: 20..29	Age: 30..39	Married: Yes	Married: No	NumCars: 0..1	NumCars: 2..3
100	1	0	0	1	1	1
200	1	0	1	0	1	1
300	1	0	0	1	1	0
400	0	1	1	0	0	0

- Closest rule is

  <NumCars: 0..1> => <Married: No>

  has only 66.66% confidence.

“catch-22” Situation.

- Mapping problems create “catch-22” situation.
  - If the intervals are too large, some rules may not have minimum confidence.
  - If the intervals are too small, some rules may not have minimum support.

Mining Quantitative Associations

Techniques can be categorized by how numerical attributes, such as age or salary are treated

1. Static discretization based on predefined concept hierarchies (data cube methods)
2. Dynamic discretization based on data distribution (quantitative rules, e.g., Agrawal & Srikant@SIGMOD96)
3. Clustering: Distance-based association (e.g., Yang & Miller@SIGMOD97)
   - One dimensional clustering then association
4. Deviation: (such as Aumann and Lindell@KDD99)

   Sex = female => Wage: mean=$7/hr (overall mean = $9)

Mining Multi-Dimensional Association

- Single-dimensional rules:

  buys(X, “milk”) $\Rightarrow$ buys(X, “bread”)

- Multi-dimensional rules: $\geq$ 2 dimensions or predicates
  - Inter-dimension assoc. rules (*no repeated predicates*)

    age(X,”19-25”) $\wedge$ occupation(X,“student”) $\Rightarrow$ buys(X, “coke”)
  - hybrid-dimension assoc. rules (*repeated predicates*)

    age(X,”19-25”) $\wedge$ buys(X, “popcorn”) $\Rightarrow$ buys(X, “coke”)

- Categorical Attributes: finite number of possible values, no ordering among values—data cube approach
- Quantitative Attributes: Numeric, implicit ordering among values—discretization, clustering, and gradient approaches

Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- **Diverse Frequent Pattern Mining**
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary

Model of Diverse Frequent Patterns

International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-019-00203-2

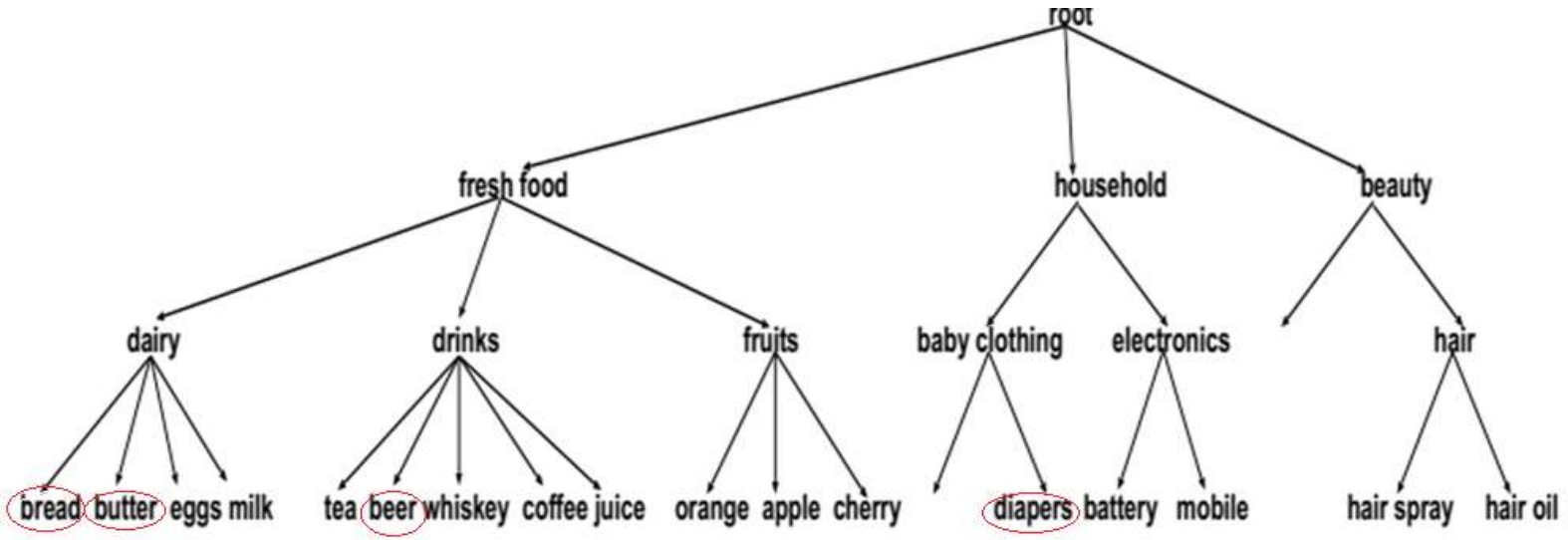
REGULAR PAPER

A model of concept hierarchy-based diverse patterns with applications to recommender system

M. Kumara Swamy[1] · P. Krishna Reddy[1]

Notion of Diversity

- Consider patterns, {***bread***, ***butter***} and {***beer***, ***diaper***} with the same *support*
- Normally {***beer***, ***diaper***} is more interesting than {***bread***, ***butter***}
- Reason
  - The items ***bread*** and ***butter*** are belong to same category ***Dairy*** (***Food Product***)
  - The items ***beer*** and ***diaper*** belong to different categories ***drinks*** and ***baby clothing***
- We say that {***beer***, ***diaper***} is more ***diverse*** than {***bread***, ***butter***}



Notion of Diversity …

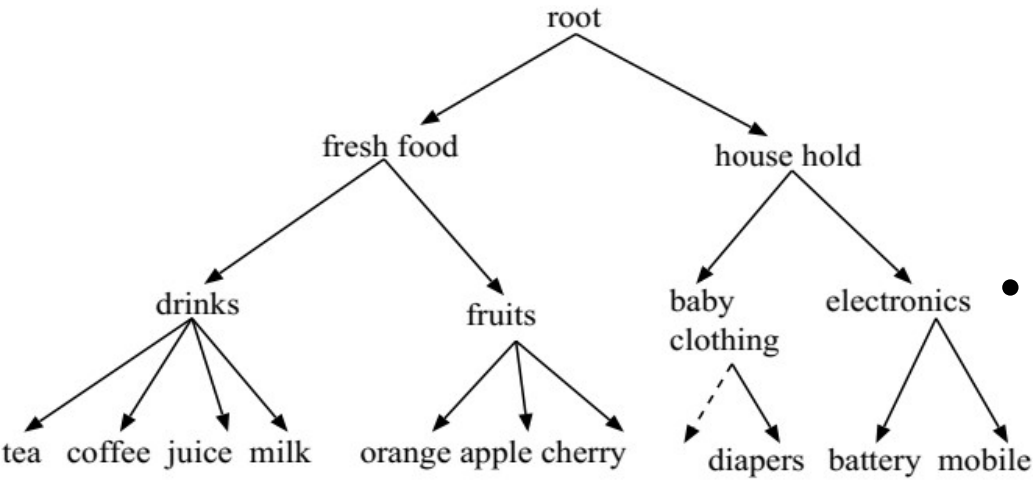
- For certain types of applications, it may be useful to distinguish between the pattern with items belonging to different categories and the pattern with items belonging to few categories.
- The existing pattern extraction approaches (frequent, sequential, periodic, etc.) fail to make such distinction.
- It is possible to rank the patterns by analysing the extent to which the items in the patterns belong to different categories.

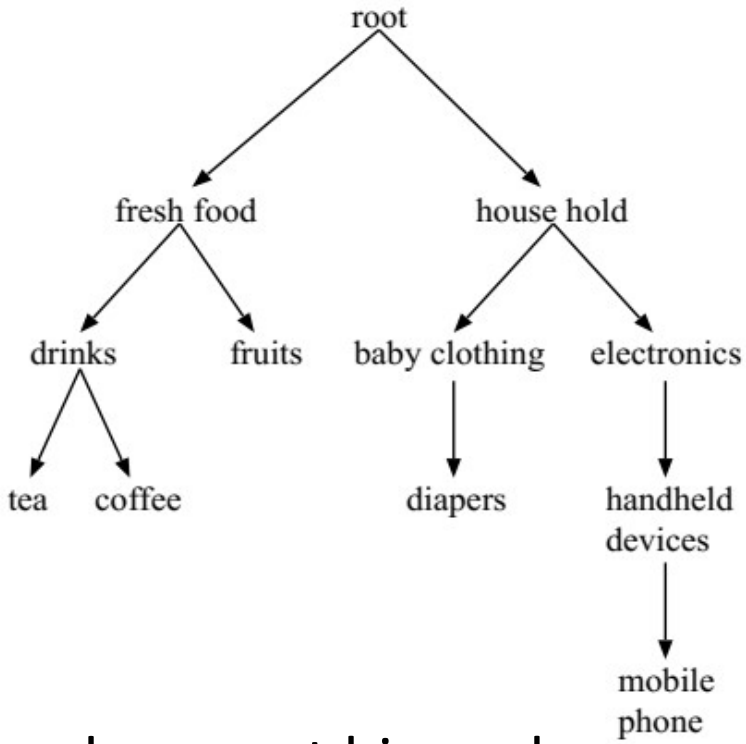
About Computation of Diversity

- A pattern is an itemset (set of items)
- The diversity value of a pattern is called diverse rank (*drank*)
- To compute the *drank* of a pattern, the following issues are to be resolved
  - To determine the category of items, category level relationship among the items is needed
    - We employ *concept hierarchy* to find the relationship of the items at higher level categories
  - Given a concept hierarchy, a framework is required to compute the *drank* value of the pattern
  - Given a set of items, concept hierarchy, and transactional database, an approach has to be developed to extract *diverse patterns* and *diverse frequent patterns*

About Balanced and Unbalanced Concept Hierarchy

- Balanced concept hierarchy
  - The depth of all leaf-level items are equal





- Unbalanced concept hierarchy
  - The depth of all leaf-level items are not equal

Model of Diverse Pattern

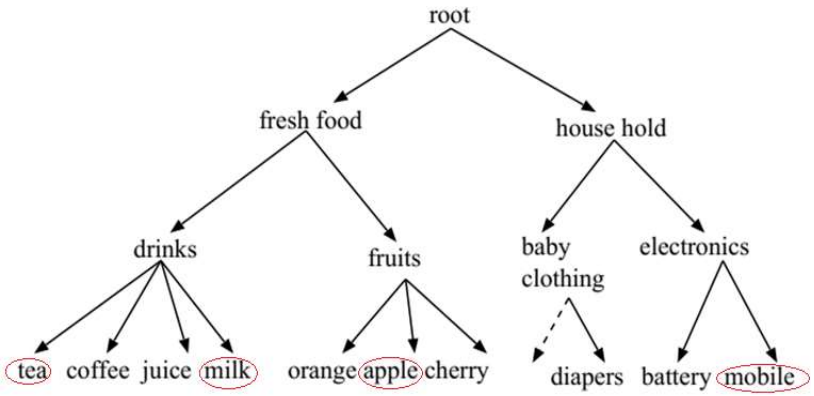
- The diversity of a pattern measures the extent of the items belong to multiple higher level categories.
  - The items of a pattern are mapped to the same/few categories in a concept hierarchy, the pattern has low diversity
  - Relatively, the items of a pattern are mapped to multiple categories, the pattern has more diversity

- Merging behavior: The speed of mapping of items from low level items to high level categories

- The question: how to measure the merging behavior?
  - The pattern merges into one/few higher level categories quickly, and ultimately all paths join at the *root*, the pattern has low diversity value
  - The pattern merges directly at *root* slowly by crossing several intermediary categories, it has relatively high diversity values

Model of Diverse of Pattern …

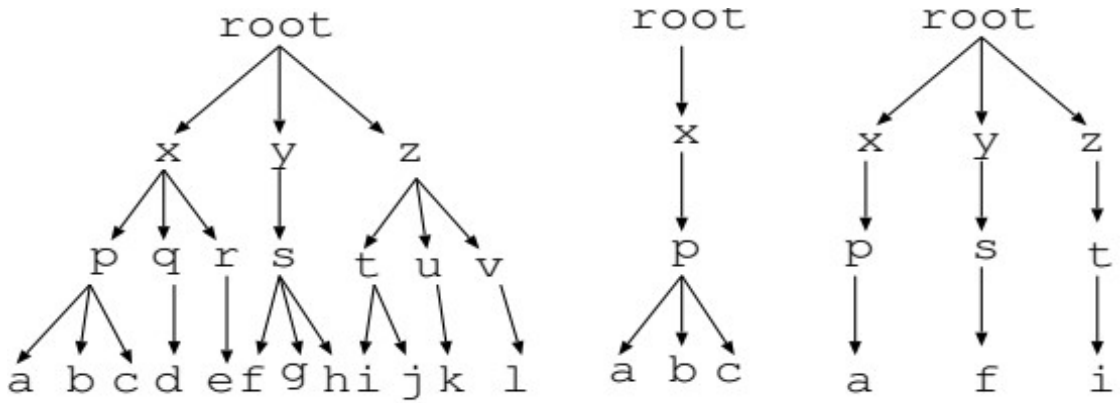
- Example
  - Consider the patterns {*tea*, *milk*}, {*milk*, *apple*} and {*milk*, *mobile*}
  - The pattern {*tea*, *milk*} quickly merges to parent, the diversity of the pattern is low.
  - The pattern {*milk*, *apple*} merges slowly as compared to the pattern {*tea*, *milk*}, the diversity of the pattern is medium.
  - Similarly, the pattern {*milk*, *mobile*} merges slowly at *root* crossing several internal nodes, the diversity of the pattern is high



A balanced concept hierarchy

Computing the Diversity of a Pattern

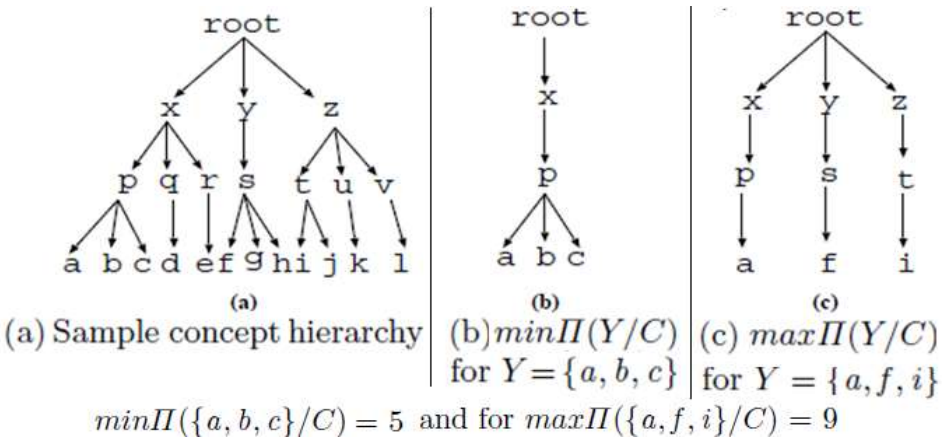
- Terminology
  - Balanced patterns (BP): Let $I = \{i_1, i_2 \cdots, i_n\}$ be a set of items, D be a transactional database on I, and C be a concept hierarchy of I with height h. A pattern $Y = \{i_1, i_2 \cdots, i_m\} \subseteq I$ with m items is called balanced pattern, if the height of all the items in Y is equal to h.
  - Projection of Pattern (Y) on concept hierarchy (C): It is denoted by $\pi(Y/C)$. It is a sub-tree which contains the portion of C concerning to the pattern Y. All the nodes and edges exist in the paths of the items of Y to the *root.*



Sample concept hierarchy $\Pi(Y/C)$ for $Y = \{a,b,c\}$ $\Pi(Y/C)$ for $Y = \{a,f,i\}$

Framework to Compute *drank* using Balanced Concept Hierarchy

- Given the pattern *Y* of certain size and concept hierarchy *C*, two extreme projections are possible
- Minimal projection
  - All the items of *Y* merge at immediate higher level
  - The number of edges $|min\ \pi(Y/C)|$ is equal to $(|Y| + h - 1)$
- Maximal projection
  - All the leaf level nodes merges at *root.*
  - The number of edges $|max\ \pi(Y/C)|$ is equal to $(|Y|$ x $h)$
- Observation: Given a pattern Y of certain size
  - Its *minimal projection* contains minimum number of edges and *maximal project* contains the maximum number of edges



(a) Sample concept hierarchy (b) $minΠ(Y/C)$ for $Y = \{a, b, c\}$ (c) $maxΠ(Y/C)$ for $Y = \{a, f, i\}$

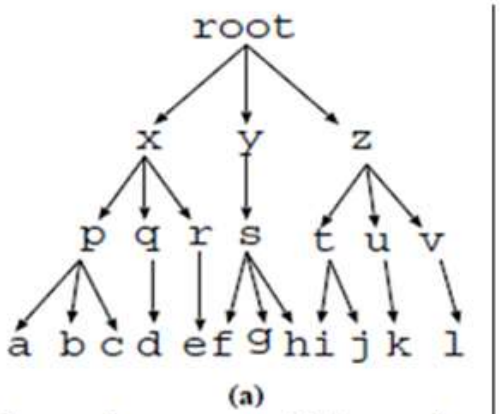
$minΠ(\{a, b, c\}/C) = 5$ and for $maxΠ(\{a, f, i\}/C) = 9$

Framework to Compute *drank* using Balanced Concept Hierarchy …

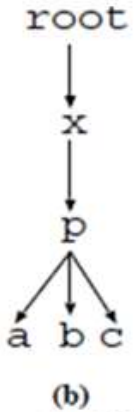
- Given a pattern Y, the *drank*(*Y*) is the ratio of number of edges in its projection ($\pi(Y/C)$) and the maximal projection ($max\ \pi(Y/C)$) which is equal to $\frac{|\Pi(Y/C)|}{|max\Pi(Y/C)|}$
- The minimum value of this ratio is equal to $\frac{|min\Pi(Y/C)|}{|max\Pi(Y/C)|}$ which is equal to 0.
- The maximum value of this ratio is equal to $\frac{|max\Pi(Y/C)|}{|max\Pi(Y/C)|}$ which is equal to 1.
- On applying the min-max normalization, we get following formula. $drank(Y) = \frac{(|\Pi(Y/C)|) - (|Y| + h - 1)}{(h-1)(|Y| - 1)}$

$$drank(Y) = \left(\frac{\left(\frac{|\Pi(Y/C)|}{|Y| \times h}\right) - \left(\frac{|Y|+h-1}{|Y| \times h}\right)}{1 - \left(\frac{|Y|+h-1}{|Y| \times h}\right)} \right) (1-0) + 0$$

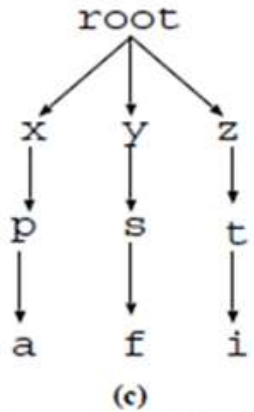
Example



(a) Sample concept hierarchy



(b) $minΠ(Y/C)$ for $Y=\{a,b,c\}$



(c) $maxΠ(Y/C)$ for $Y=\{a,f,i\}$

$minΠ(\{a,b,c\}/C)=5$ and for $maxΠ(\{a,f,i\}/C)=9$

$$drank(Y)=\frac{(|Π(Y/C)|)-(|Y|+h-1)}{(h-1)(|Y|-1)}$$

$drank(\{a,b,c\})=\frac{5-5}{2\times 2}=0.0$

$drank(\{a,b,e\})=0.25$

$drank(\{a,b,f\})=0.5$

$drank(\{a,e,f\})=0.75$

$drank(\{a,f,i\})=1.0$

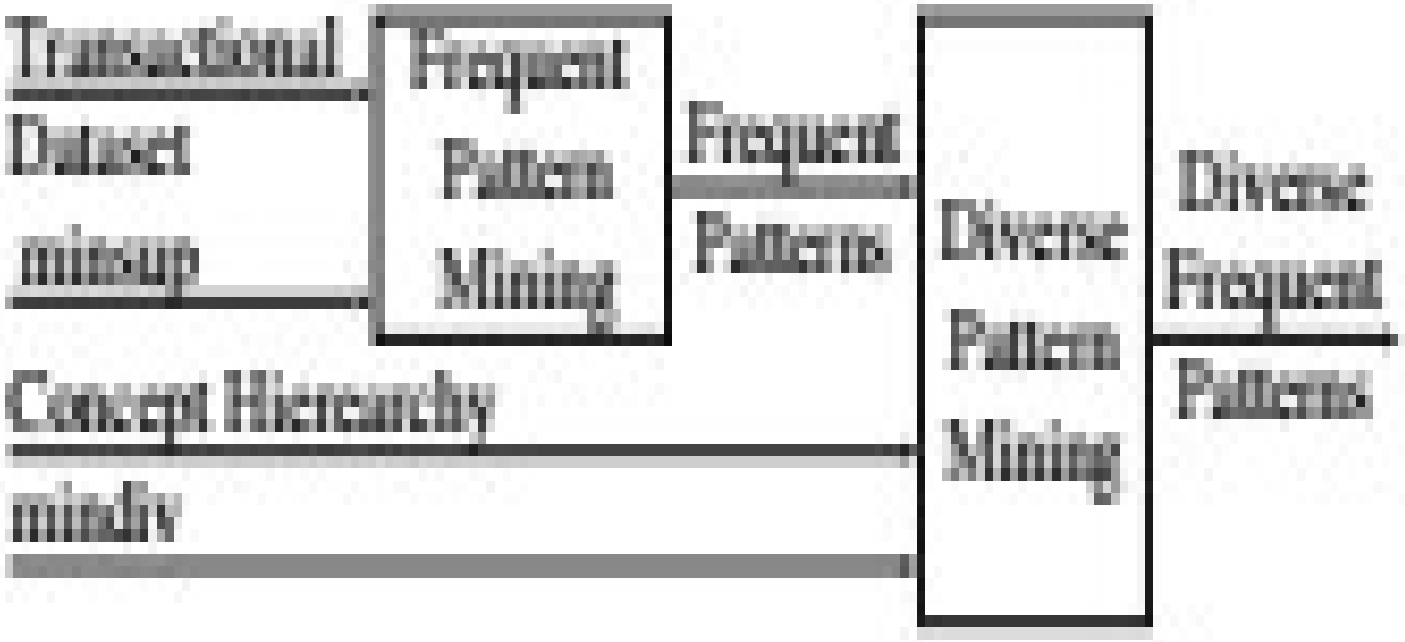


Fig. 3 Extraction of diverse frequent patterns

Experiments

- Dataset from MovieLens project (http://www.grouplens.org)
  : contains 100,000 ratings of 943 users on 1682 movies

Table 1 Top-10 patterns of size 3 sorted w.r.t. *drank*

Top 10 diverse patterns	*drank*	*Support* (%)
{G.I. Jane, The Game, Contact}	1	5.5
{G.I. Jane, Titanic, Contact}	1	5.3
{Amistad, L.A. Confidential, Titanic}	1	5.3
{One Fine Day, Return of the Jedi, Star Wars}	1	5.1
{One Fine Day, Star Wars, Return of the Jedi}	1	5.1
{The First Wives Club, Return of the Jedi, Star Wars}	1	5.1
{Volcano, Star Wars, Return of the Jedi}	1	5.0
{Welcome to the Dollhouse, Trainspotting, Star Wars}	1	4.8
{Austin Powers: International Man of Mystery, Fargo, Star Wars}	1	4.6
{Austin Powers: Int'L Man of Mystery, Return of the Jedi, Star Wars}	1	4.3

Table 2 Top 10 patterns of size 3 sorted w.r.t. support value

Top 10 frequent patterns	*Support* (%)	*drank*
{Raiders of the Lost Ark, Return of the Jedi, Star Wars}	23.2	0.33
{The Empire Strikes Back, Raiders of the Lost Ark, Star Wars}	20.6	0.33
{The Rock, Return of the Jedi, Star Wars}	17.7	0.33
{Independence Day, Return of the Jedi, Star Wars}	18.2	0.33
{Aliens, Star Wars, Raiders of the Lost Ark}	16.1	0.75
{Braveheart, Indiana Jones and the Last Crusade, Return of the Jedi}	14.9	0.33
{Aliens, Return of the Jedi, Star Wars}	14.8	0.33
{The Terminator, Pulp Fiction, Raiders of the Lost Ark}	14.8	0.33
{The Princess Bride, The Empire Strikes Back, Raiders of the Lost Ark}	14.5	0.33
{Independence Day, Raiders of the Lost Ark, Return of the Jedi}	14.4	0.33

Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- Diverse Frequent Pattern Mining
- **High Utility Pattern mining**
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary

HIGH UTILITY MINING

A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets

Ying Liu, Wei-keng Liao, and Alok Choudhary

PAKDD 2005

Limitations of Frequent Itemset Mining

- Purchase quantities are not taken into account.
  - Thus, an item may only appear once or zero time in a transaction. Thus, if a customer has bought five breads, ten breads or twenty breads, it is viewed as the same.
- All items are viewed as having the same importance, utility of weight.
  - For example, if a customer buys a very expensive bottle of wine or just a piece of bread, it is viewed as being equally important.
- Frequent pattern mining may find many frequent patterns that are not interesting.
  - For example, one may find that {bread, milk} is a frequent pattern. However, from a business perspective, this pattern may be uninteresting because it does not generate much profit.
  - Moreover, frequent pattern mining algorithms may miss the rare patterns that generate a high profit such as perhaps {caviar, wine}

High-utility itemset mining

- **High-utility itemset mining addresses the limitations** .
- A transaction database contains transactions where purchase quantities are taken into account as well as the unit profit of each item. For example, consider the following transaction database.
- Local transaction utility of an item
- External utility of an item
- Utility of the item in a transaction= Local utility * external utility

Table 1. A transaction database

(a) Transaction table. Each row is a transaction. The columns represent the number of items in a particular transaction. TID is the transaction identification number

TID \ ITEM	A	B	C	D	E
T_1	0	0	18	0	1
T_2	0	6	0	1	1
T_3	2	0	1	0	1
T_4	1	0	0	1	1
T_5	0	0	4	0	2
T_6	1	1	0	0	0
T_7	0	10	0	1	1
T_8	3	0	25	3	1
T_9	1	1	0	0	0
T_{10}	0	6	2	0	2

(b) The utility table. The right column displays the profit of each item per unit in dollars

ITEM	PROFIT ($) (per unit)
A	3
B	10
C	1
D	6
E	5

(c) Transaction utility (TU) of the transaction database

TID	TU	TID	TU
T_1	23	T_6	13
T_2	71	T_7	111
T_3	12	T_8	57
T_4	14	T_9	13
T_5	14	T_{10}	72

Two-Phase algorithm

- Transaction utility= sum of all utilities of items
- Transaction weighted utility of itemset X, twu(X)= Summation of utilities all transactions Ti such that X is belongs to Ti.
- High Transaction-weighted Utilization Itemset: Transaction weighted Utility of the itemset should be greater than the user given threshold.
- Transaction-weighted Downward Closure Property: If k-itemset is high transaction-weighted utilization itemset, k-1 itemset will be high transaction-weighted utilization itemset.
- Theorem: Let HTWU be the collection of all high transaction-weighted utilization itemsets in a transaction database D, and HU be the collection of high utility itemsets in D. Then HU is the subset of HTWU.

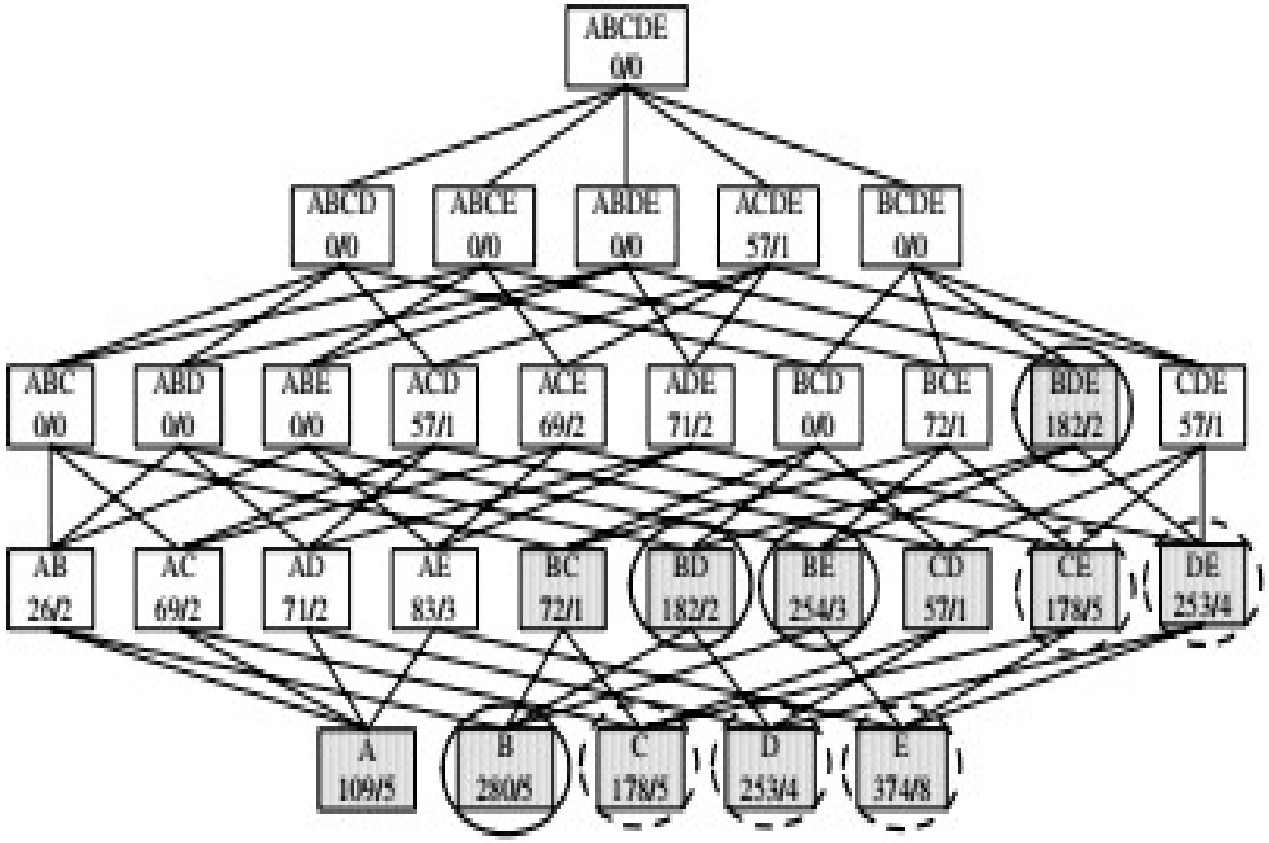


Fig. 1. Itemsets lattice related to the example in Table 1. ε' = 120. Itemsets in circles (solid and dashed) are the high transaction-weighted utilization itemsets in *transaction-weighted utilization mining model*. Gray-shaded boxes denote the search space. Itemsets in solid circles are high utility itemsets. Numbers in each box are transaction-weighted utilization / number of occurrence

Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- **Mining Rare Patterns and Negative Patterns**
- Constraint-Based Frequent Pattern Mining
- Summary

Negative and Rare Patterns

- Rare patterns: Very low support but interesting
  - E.g., buying Rolex watches
  - Mining: Setting individual-based or special group-based support threshold for valuable items
- Negative patterns
  - Since it is unlikely that one buys Ford Expedition (an SUV car) and Toyota Prius (a hybrid car) together, Ford Expedition and Toyota Prius are likely negatively correlated patterns
- Negatively correlated patterns that are infrequent tend to be more interesting than those that are frequent

Defining Negative Correlated Patterns (I)

- Definition 1 (support-based)
  - If itemsets X and Y are both frequent but rarely occur together, i.e.,

    $sup(X \cup Y) < sup(X) * sup(Y)$
  - Then X and Y are negatively correlated
- Problem: A store sold two needle 100 packages A and B, only one transaction containing both A and B.
  - When there are in total 200 transactions, we have

    $s(A \cup B) = 0.005$, $s(A) * s(B) = 0.25$, $s(A \cup B) < s(A) * s(B)$
  - When there are 10^5 transactions, we have

    $s(A \cup B) = 1/10^5$, $s(A) * s(B) = 1/10^3 * 1/10^3$, $s(A \cup B) > s(A) * s(B)$
  - Where is the problem? —Null transactions, i.e., the support-based definition is not null-invariant!

Defining Negative Correlated Patterns (II)

- Definition 2 (negative itemset-based)
  - X is a *negative itemset* if (1) X = $\bar{A}$ U B, where B is a set of positive items, and $\bar{A}$ is a set of negative items, $|\bar{A}| \geq 1$, and (2) $s(X) \geq \mu$
  - Itemsets X is negatively correlated, if

$$s(X) < \prod_{i=1}^{k} s(x_i), where\ x_i \in X,\ and\ s(x_i)\ is\ the\ support\ of\ x_i$$

- This definition suffers a similar null-invariant problem
- Definition 3 (Kulzynski measure-based) If itemsets X and Y are frequent, but $(P(X|Y) + P(Y|X))/2 < \epsilon$, where ϵ is a negative pattern threshold, then X and Y are negatively correlated.
- Ex. For the same needle package problem, when no matter there are 200 or 10^5 transactions, if $\epsilon = 0.01$, we have

$$(P(A|B) + P(B|A))/2 = (0.01 + 0.01)/2 < \epsilon$$

Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- **Constraint-Based Frequent Pattern Mining**
- Summary

Constraint-based (Query-Directed) Mining

- Finding all the patterns in a database autonomously? — unrealistic!
  - The patterns could be too many but not focused!
- Data mining should be an interactive process
  - User directs what to be mined using a data mining query language (or a graphical user interface)
- Constraint-based mining
  - User flexibility: provides constraints on what to be mined
  - Optimization: explores such constraints for efficient mining — constraint-based mining: constraint-pushing, similar to push selection first in DB query processing
  - Note: still find all the answers satisfying constraints, not finding some answers in “heuristic search”

Constraints in Data Mining

- Knowledge type constraint:
  - classification, association, etc.
- Data constraint — using SQL-like queries
  - find product pairs sold together in stores in Chicago this year
- Dimension/level constraint
  - in relevance to region, price, brand, customer category
- Rule (or pattern) constraint
  - small sales (price < $10) triggers big sales (sum > $200)
- Interestingness constraint
  - strong rules: min_support ≥ 3%, min_confidence ≥ 60%

Meta-Rule Guided Mining

- Meta-rule can be in the rule form with partially instantiated predicates and constants

  $P_1(X, Y) \wedge P_2(X, W) \Rightarrow buys(X, \text{“iPad”})$

- The resulting rule derived can be

  $age(X, \text{“15-25”}) \wedge profession(X, \text{“student”}) \Rightarrow buys(X, \text{“iPad”})$

- In general, it can be in the form of

  $P_1 \wedge P_2 \wedge \ldots \wedge P_l \Rightarrow Q_1 \wedge Q_2 \wedge \ldots \wedge Q_r$

- Method to find meta-rules
  - Find frequent (l+r) predicates (based on min-support threshold)
  - Push constants deeply when possible into the mining process (see the remaining discussions on constraint-push techniques)
  - Use confidence, correlation, and other measures when possible

Constraint-Based Frequent Pattern Mining

- Pattern space pruning constraints
  - Anti-monotonic: If constraint c is violated, its further mining can be terminated
  - Monotonic: If c is satisfied, no need to check c again
  - Succinct: c must be satisfied, so one can start with the data sets satisfying c
  - Convertible: c is not monotonic nor anti-monotonic, but it can be converted into it if items in the transaction can be properly ordered
- Data space pruning constraint
  - Data succinct: Data space can be pruned at the initial pattern mining process
  - Data anti-monotonic: If a transaction t does not satisfy c, t can be pruned from its further mining

What Constraints Are Convertible?

Constraint	Convertible anti-monotone	Convertible monotone	Strongly convertible
avg(S) ≤ , ≥ v	Yes	Yes	Yes
median(S) ≤ , ≥ v	Yes	Yes	Yes
sum(S) ≤ v (items could be of any value, v ≥ 0)	Yes	No	No
sum(S) ≤ v (items could be of any value, v ≤ 0)	No	Yes	No
sum(S) ≥ v (items could be of any value, v ≥ 0)	No	Yes	No
sum(S) ≥ v (items could be of any value, v ≤ 0)	Yes	No	No
……			

Constraint-Based Mining — A General Picture

Constraint	Anti-monotone	Monotone	Succinct
$v \in S$	no	yes	yes
$S \supseteq V$	no	yes	yes
$S \subseteq V$	yes	no	yes
$\min(S) \leq v$	no	yes	yes
$\min(S) \geq v$	yes	no	yes
$\max(S) \leq v$	yes	no	yes
$\max(S) \geq v$	no	yes	yes
$\text{count}(S) \leq v$	yes	no	weakly
$\text{count}(S) \geq v$	no	yes	weakly
$\text{sum}(S) \leq v$ ($a \in S, a \geq 0$)	yes	no	no
$\text{sum}(S) \geq v$ ($a \in S, a \geq 0$)	no	yes	no
$\text{range}(S) \leq v$	yes	no	no
$\text{range}(S) \geq v$	no	yes	no
$\text{avg}(S)\ \theta\ v, \theta \in \{=, \leq, \geq\}$	convertible	convertible	no
$\text{support}(S) \geq \xi$	yes	no	no
$\text{support}(S) \leq \xi$	no	yes	no

Other topics

- Mining sequential patterns
- Mining time series data
- Mining subgraph patterns

Summary

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary