Chapter 5

กฎแห่งการเกี่ยวโยง

Mining frequent patterns and association rules

อ.ศศิภา พันธุวดีธร

2301456 Intro DW-DM

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

Topics

- Frequent patterns analysis
- Market basket analysis
- Measures of rule interestingness
- Basic concepts : frequent patterns and association rules
- Efficient and scalable frequent itemset mining methods – Apriori, FP-Growth, ECLAT-vertical format 3 Algor
- Mining various kinds of association rules
  - Mining multilevel association
  - Miming multidimensional association
  - Mining quantitative association
- From association mining to correlation analysis
- Constraint-based association mining

What Is Frequent Pattern Analysis?

กลุ่มของ item ที่เกิดขึ้นพร้อม ๆ กัน or sub-sequence → เกิดอันหนึ่งก่อนแล้วเกิดอีกอันตามมา

- Frequent pattern รูปแบบที่พบบ่อย ๆ

  : 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

  ค้นพบสิ่งที่ซ่อนอยู่ใน Data ได้

- Motivation: Finding inherent regularities in data
  - What products were often purchased together? ดูการซื้อสินค้า

    มักซื้อเบียร์คู่กับ ผ้าอ้อม

    — Beer and diapers?!

    คนที่ซื้อ PC มักซื้อกล้องกับ mem ต่อเพื่อเก็บภาพได้เยอะขึ้น

  - What are the subsequent purchases after buying a PC? — digital camera, memory card

    ยามีผลกับ DNA เรารึเปล่า ? ไวรัสทิ้งร่องรอยไว้ใน DNA → รู้ว่าเคยเป็นโรคอะไร

  - What kinds of DNA are sensitive to this new drug?

What Is Frequent Pattern Analysis?

- Applications
  - ดูตลาดซื้อ จับคู่สินค้าเพื่อนำเสนอ
  - Basket data analysis, frequent pattern analysis, cross-marketing, catalog design, แวะหน้าเว็บไหน sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.
- Association rule mining is the task of finding correlations among items in a dataset
- Initial research was largely motivated by the analysis of market basket data

What Is Frequent Pattern Analysis?

- In this chapter, we study methods of frequent itemset mining.
  - STEP 1: Find frequent itemsets
  - STEP 2: Generate strong association rules
- Interesting questions are
  - How can we find frequent itemsets from large amounts of data (transactional, relational)?
  - Which association rules are the most interesting?
  - How can we help or guide the mining procedure to discover interesting associations or correlations?

Market basket analysis

- A process to analyze customer buying habits by finding associations between the different items in "shopping baskets"
- The discovered pattern helps retailers develop marketing strategies

  ได้สิ่งที่รู้แล้ว (obvious) → ไม่ถือว่าทำ mining

  - Bread + milk, computer + antivirus software, computer + printer, beer + diapers!!!

    -> select brands, plan shelf space, plan proximity of placing products on shelf, discount, design a new catalog, campaign with credit cards

Market basket analysis

- Analyze buying transactions e.g. customer receipts that contain data about
  - Customer information ex 7-eleven
  - Product type, brand, price
  - Total items
  - Total cost
- Patterns can be represented in the form of association rules

ไม่ได้บอกว่าซื้ออะไรก่อนหลัง

buys(X, “Computer”) => buys(X, “antivirus_software”) [support=2%, confidence=60%]

มีค่าตัววัด บอกความถี่ที่เกิด ค. น่าสนใจ

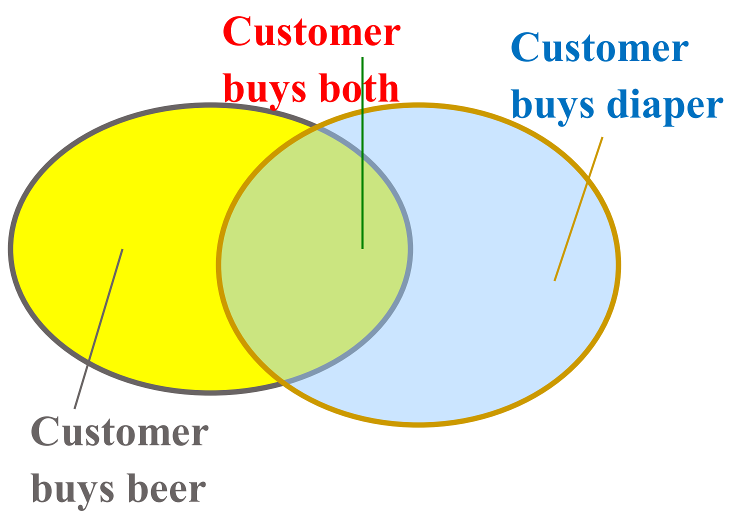
Measures of rule interestingness

- Two measures of rule interestingness
  - จน.ของสิ่งที่ซื้อพร้อมกันว่าพบกี่รายการของสิ่งที่ซื้อทั้งหมด
  - <u>Support</u> : 2% of all the transactions show that computer and antivirus software are purchased together
  - <u>Confidence</u> : 60% of the customers who purchased a computer also bought the software เชื่อมั่นได้ 60 %
- Ass. Rules are interesting if they satisfy both a minimum support threshold and a minimum confidence threshold (set by users/experts)

Basic Concepts: Frequent Patterns

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

3



1 - itemset → ประกอบด้วยสินค้า 1 ชิ้น
2 2
3 3

- itemset: A set of one or more items
- k-itemset $X = \{\overbrace{x_1, \ldots, x_k}^{k}\}$
- *(absolute) support*, or, *support count* of X: Frequency or occurrence of an itemset X (ค.ถี่ของก.เกิด item ที่เราสนใจ)

Beer: 3

Nuts: 3

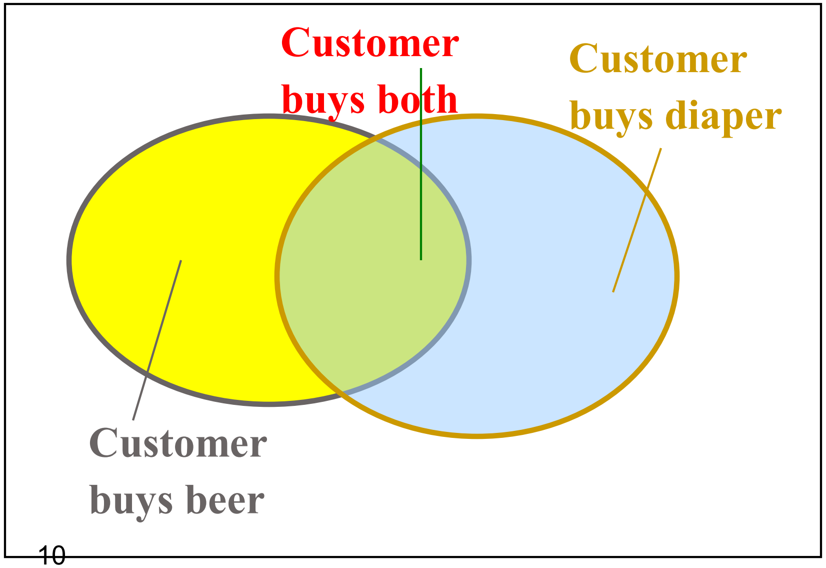
Diaper: 4

Eggs: 3

{Beer, Diaper} : 3 (คู่กัน)

Basic Concepts: Frequent Patterns

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- *(relative) support*, *s*, is the fraction of transactions that contains X (i.e., the probability that a transaction contains X)
- support(A=>B) = P(A ∪ B)
- support(Beer=>Diaper)

  → 5 รายการ

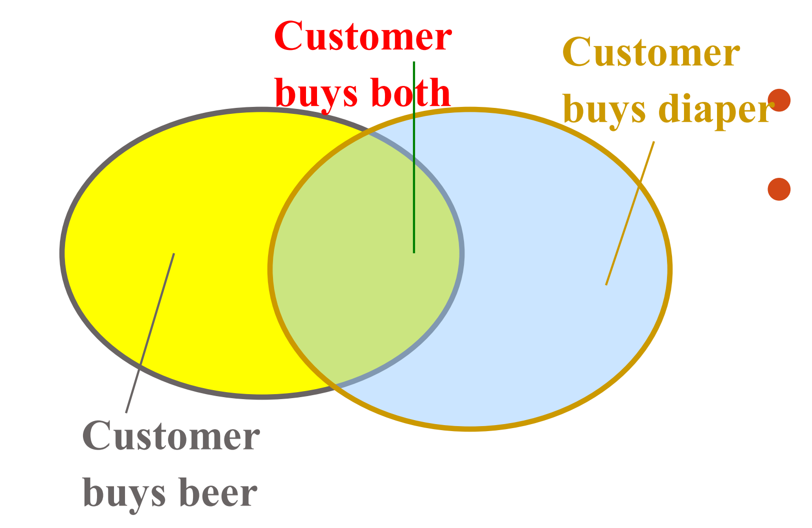
  = 3/5*100%

  = 60%
- An itemset X is *frequent* if X's support is no less than a *minsup* threshold

  ถ้าตั้ง minsup เป็น 2 แล้วได้ 3 ✓

Basic Concepts: Association Rules

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



หากฎของ x ไป y

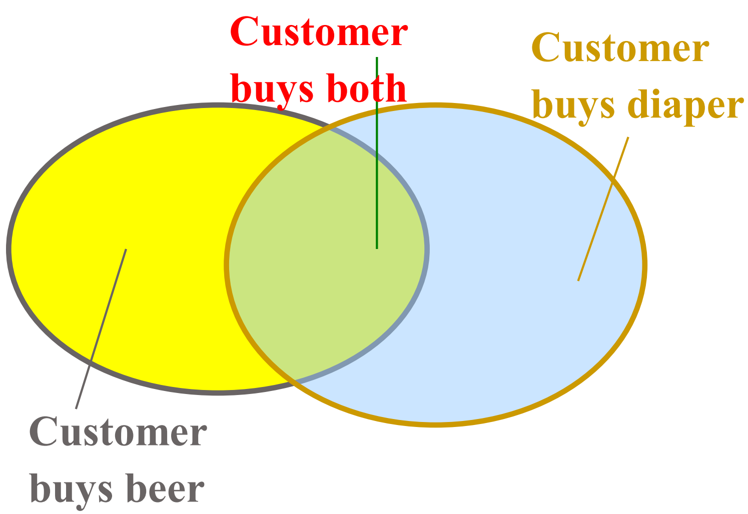
- Find all the rules $X \rightarrow Y$ with minimum support and confidence (strong/interesting rules)
  - support, *s*, probability that a transaction contains $X \cup Y$ ~~X or Y~~ ต้องมีพร้อมกัน
  - confidence, *c*, conditional probability that a transaction having X also contains *Y*
- support(A=>B) = $P(A \cup B)$
- confidence(A=>B) = $P(B|A)$

= support($A \cup B$) / support(A)

= sup_cnt($A \cup B$) / sup_cnt(A)

Basic Concepts: Association Rules

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



Let $minsup = 50\%$, $minconf = 50\%$

$\{\text{Beer, Diaper}\}:3$, support = $\frac{3}{5} \times 100 = 60$ %.

- Association rules: (many more!)
  - $Beer \rightarrow Diaper$ (60%, 100%) $\frac{3}{3} \times 100 = 100$ (beer + diaper / beer)
  - $Diaper \rightarrow Beer$ (60%, 75%) $\frac{3}{4} \times 100 = 75$ (beer + diaper / diaper)

เกิน 50 % ทั้ง 2 เป็น strong association rules

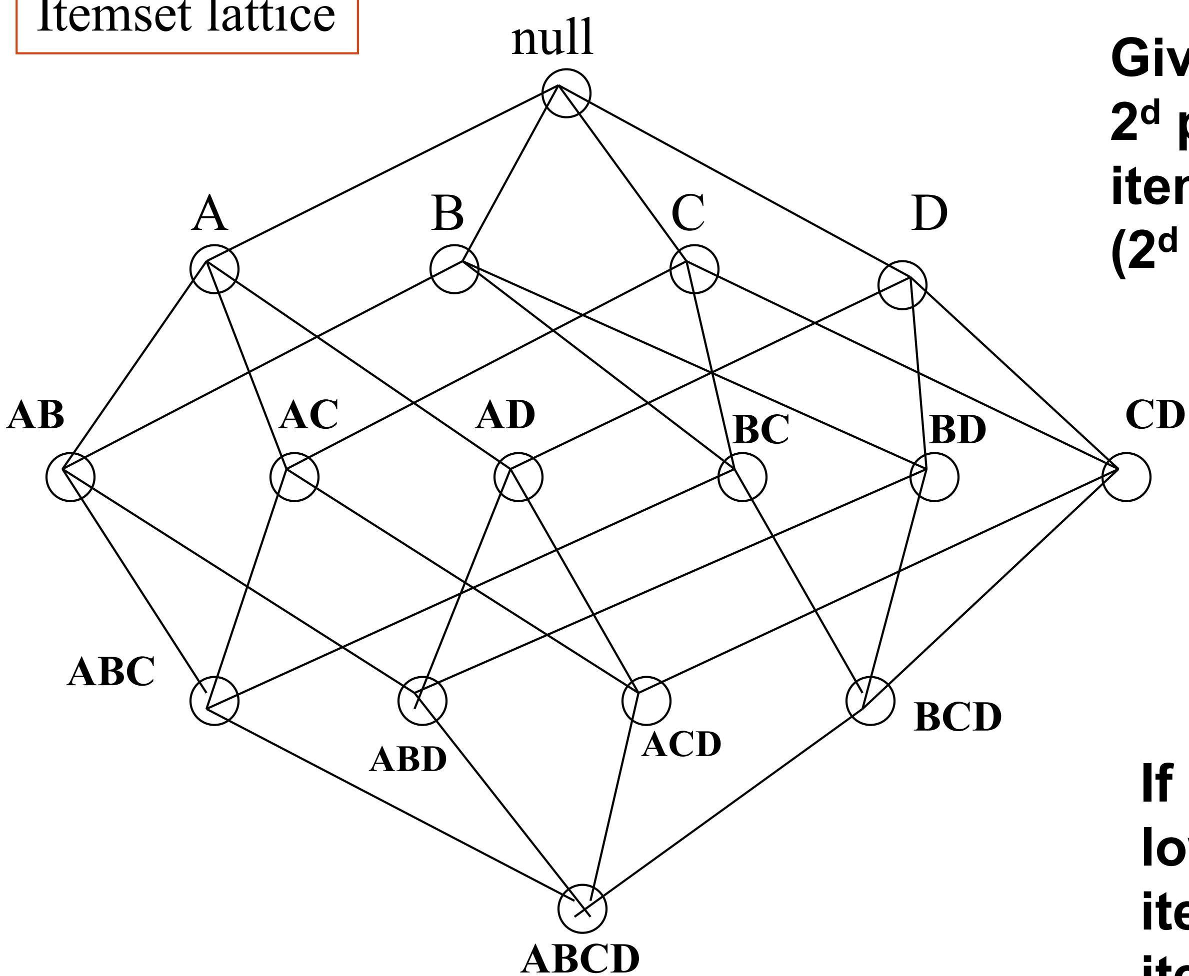
- support(A=>B) = P(A ∪ B)
- confidence(A=>B) = P(B|A)

= support(A ∪ B) / support(A)

= sup_cnt(A ∪ B) / sup_cnt(A)

Possible candidate itemsets

Itemset lattice



Given d items, there are 2^d possible candidate itemsets
(2^d –1) มีสินค้า d items

$$\binom{4}{1}+\binom{4}{2}+\binom{4}{3}+\binom{4}{4}$$

4+6+4+1=15 รูปแบบของ item set ที่เป็นไปได้ทั้งหมด

If minsup is set very low, these candidate itemsets will be frequent itemsets (too huge)

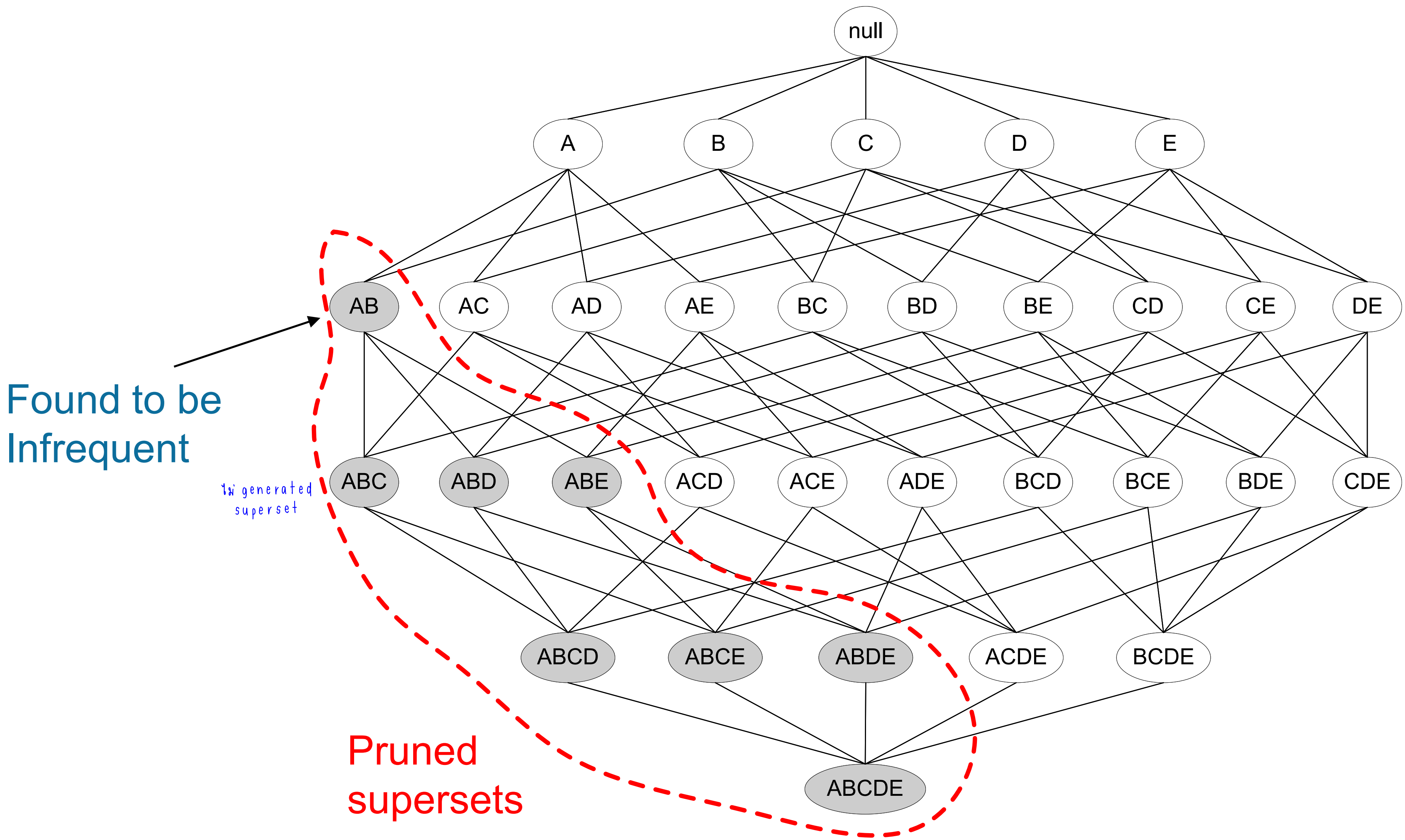
The Downward Closure Property and Scalable Mining Methods

- The downward closure property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If {**beer, diaper, nuts**} is frequent, so is {**beer, diaper**}
  - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- Scalable mining methods: Three major approaches
  1. Apriori (Agrawal & Srikant@VLDB'94)
  2. นิยม → rapid miner
     Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD'00)
  3. Vertical data format approach (Charm—Zaki & Hsiao @SDM'02)

1. Apriori: A Candidate Generation & Test Approach

- <u>Apriori pruning principle</u>: If there is any itemset which is
  ไม่ถูก generated
  infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Method: ต้องอ่าน DB หลายรอบ (ไม่ค่อยเวิร์ค)
  หา 1-items → ผ่าน minsup ไหม ผ่านไปต่อ
  - 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 (must scan DB again!!)
  - Terminate when no frequent or candidate set can be generated

Illustrating Apriori Principle



Slide from P. Tan, M. Steinbach, and V. Kumar

The Apriori Algorithm (Pseudo-Code)

1 - item
2 - item

C_k: Candidate itemset of size k

ผ่าน minsup = large (อดีต) = frequent ✓

L_k : frequent itemset of size k ***(used to call large itemset)***

---

frequent 1 - item → หา candidate → นับ เช็คกับ minsup

L_1 = {frequent items};

for ($k = 1$; L_k $!= \emptyset$; k++) **do begin**

  C_{k+1} = candidates generated from L_k;

  for each transaction t in database do

    increment the count of all candidates in C_{k+1} that are contained in t

  L_{k+1} = candidates in C_{k+1} with min_support

  end

return $\cup_k L_k$;

The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1^{st} scan

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

น้อยกว่า minsup ตกไป

L_1

เหลือ

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

candidate set

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2^{nd} scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

ไม่ผ่าน minsup

ไม่ผ่าน minsup

L_2

เหลือ

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

generate พร้อม purning (หาตัวหน้าเหมือนกัน แล้วลอกตัวหลัง)

C_3

Itemset
{B, C, E}

แล้วดู subset ด้วยว่าตัวไหนไม่ Frequent = ไม่ generated

3^{rd} scan

L_3

Itemset	sup
{B, C, E}	2

เกิน minsup

Implementation of Apriori

- 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 ดูต่อหน้าซ้ำ
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning: ดู frequent
    - *acde* is removed because *ade* is not in L_3
  - $C_4 = \{abcd\}$

Candidate Generation: An SQL Implementation

- SQL Implementation of candidate generation
  - Suppose the items in L_{k-1} are listed in an order
  - Step 1: self-joining L_{k-1}

    insert into $\boldsymbol{C_k}$

    select $\boldsymbol{p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}}$

    from $\boldsymbol{L_{k-1}\ p, L_{k-1}\ q}$

    where $\boldsymbol{p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}}$
  - Step 2: pruning

    forall ***itemsets c in*** $\boldsymbol{C_k}$ do

    forall ***(k-1)-subsets s of c*** do

    if *(s is not in* L_{k-1}*)* **then delete** *c* **from** C_k

All frequent itemsets

- $L_1 \cup L_2 \cup L_3$ เอาที่ผ่านมา ∪ กัน
- {A}, {B}, {C}, {E}, {A, C}, {B, C}, {B, E}, {C, E}, {B, C, E}

- Generate association rules
- We are not interested in ถ้า 1-item ไม่สน → ไม่สร้างเป็นกฎ
  - A -> A, B -> B, …

Interesting/Strong ass. rules

A → C
C → A

- From {A, C}, {B, C}, {B, E}, {C, E}, {B, C, E}
- Association rules from an itemset are binary partitions of the same itemset

4 รายการ confident

AC A
  - A -> C (2/4=0.5, 2/2 = 1) ✓

AC C
  - C -> A (2/4=0.5, 2/3 = 0.67)
  - B -> C (2/4=0.5, 2/3 = 0.67)
  - C -> B (2/4=0.5, 2/3 = 0.67)
  - B -> E (3/4=0.75, 3/3 = 1) ✓
  - E -> B (3/4=0.75, 3/3 = 1) ✓
  - C -> E (2/4=0.5, 2/3 = 0.67)
  - E -> C (2/4=0.5, 2/3 = 0.67)

Interesting/Strong ass. rules

- From {A, C}, {B, C}, {B, E}, {C, E}, {B, C, E}
- Association rules from an itemset are binary partitions of the same itemset
  - B -> C Λ E (2/4=0.5, 2/3 = 0.67)
  - C -> B Λ E (2/4=0.5, 2/3 = 0.67)
  - E -> B Λ C (2/4=0.5, 2/3 = 0.67)
  - B Λ C -> E (2/4=0.5, 2/2 = 1) ✓
  - B Λ E -> C (2/4=0.5, 2/3 = 0.67)
  - C Λ E -> B (2/4=0.5, 2/2 = 1) ✓
- Keep rules satisfying minimum confidence (because all rules are already satisfy minimum support)
- Let's say : minimum confidence = 75%
- IN THIS EXAMPLE, strong rules are rules that have confidence = 100%

example

minsup = $\frac{40}{100}$ = $\frac{2}{5}$

- Assume the user-specified minimum support is 40%, then generate all frequent itemsets.
- **Given:** The transaction database shown below
- **TID** | ***A*** | ***B*** | ***C*** | ***D*** | ***E***

TID	***A***	***B***	***C***	***D***	***E***
T1	1	1	1	0	0
T2	1	1	1	1	1
T3	1	0	1	1	0
T4	1	0	1	1	1
T5	1	1	1	1	0

ผ่านหมด

- **Pass 1**

C_1 *after scan DB*

Itemset	sup
{A}	5
{B}	3
{C}	5
{D}	4
{E}	2

$\xrightarrow{L_1}$

Itemset	sup
{A}	5
{B}	3
{C}	5
{D}	4
{E}	2

- **Pass 2**

C_2 *after scan DB*

Itemset	sup
{A, B}	3
{A, C}	5
{A, D}	4
{A, E}	2
{B, C}	3
{B, D}	2
~~{B, E}~~	~~1~~
{C, D}	4
{C, E}	2
{D, E}	2

$\xrightarrow{L_2}$

Itemset	sup
{A, B}	3
{A, C}	5
{A, D}	4
{A, E}	2
{B, C}	3
{B, D}	2
{C, D}	4
{C, E}	2
{D, E}	2

ต่อไป self join

Nothing pruned because every subset is frequent

- **Pass 3**
- To create C3 only look at items that have the same first item (in pass k, the first k - 2 items must match)

purning

	Itemset
Join AB with AC	{A, B, C}
Join AB with AD	{A, B, D}
~~Join AB with AE~~	~~{A, B, E}~~
Join AC with AD	{A, C, D}
Join AC with AE	{A, C, E}
Join AD with AE	{A, D, E}
Join BC with BD	{B, C, D}
Join CD with CE	{C, D, E}

C_3 after scan DB

And L_3 are the same

Itemset	sup
{A, B, C}	3
{A, B, D}	2
{A, C, D}	4
{A, C, E}	2
{A, D, E}	2
{B, C, D}	2
{C, D, E}	2

Pruning: Pruning eliminates ABE since BE is not frequent

- **Pass 4**

C_4 *after scan DB*

join & pruning

	Itemset
Join ABC with ABD	{A, B, C, D}
Join ACD with ACE	{A, C, D, E}

ไม่มี 3 ตัวหน้าเหมือนกัน เลยหยุดแค่นี้

Itemset	sup
{A, B, C, D}	2
{A, C, D, E}	2

And L_4 are the same

Pruning:

- For ABCD we check whether ABC, ABD, ACD, BCD are frequent. They are in all cases, so we do not prune ABCD.
- For ACDE we check whether ACD, ACE, ADE, CDE are frequent. Yes, in all cases, so we do not prune ACDE

Pass 5: For pass 5 we can't form any candidates because there aren't two frequent 4-itemsets beginning with the same 3 items.

Exercise

- Trace the results of using the Apriori algorithm on the grocery store example with support threshold *s*=33.33% and confidence threshold *c*=60%. Show the candidate and frequent itemsets for each database scan. Enumerate all the final frequent itemsets. Also indicate the association rules that are generated and highlight the strong ones, sort them by confidence.
-

Transaction ID	Items
T1	HotDogs, Buns, Ketchup
T2	HotDogs, Buns
T3	HotDogs, Coke, Chips
T4	Chips, Coke
T5	Chips, Ketchup
T6	HotDogs, Coke, Chips

Further Improvement of the Apriori Method

- Major computational challenges ข้อเสีย
  - Multiple scans of transaction database อ่าน database หลายรอบ
  - Huge number of candidates สร้าง candidate set จน.มาก
  - Tedious workload of support counting for candidates เสียเวลานับ support counting
- Improving Apriori: general ideas ปรับปรุง
  - Reduce passes of transaction database scans ลดการอ่าน DB
  - Shrink number of candidates ลด candidates
  - Facilitate support counting of candidates หาวิธีนับเร็ว ๆ

Transaction reduction

ลดจน. transaction ที่จะต้องอ่านในรอบถัดไป

- Reduce the number of transactions scanned in future iterations
- By removing the transaction that does not contain any frequent k-itemsets ลบไปเลย

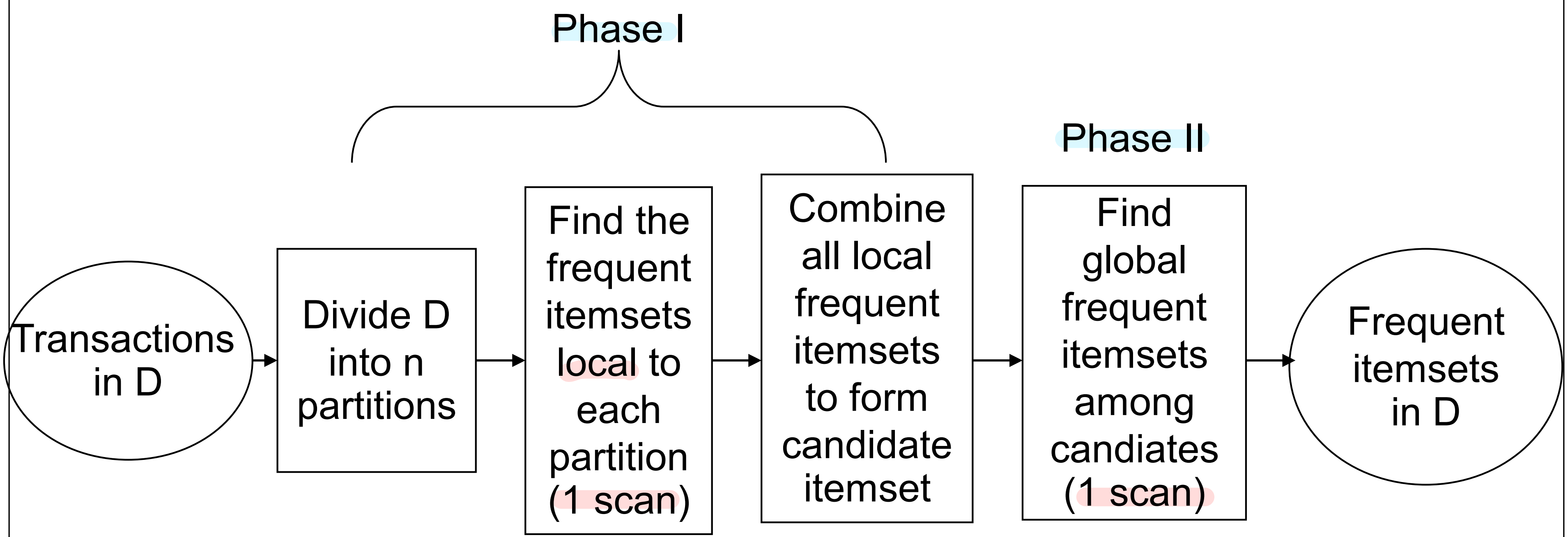
Partitioning:
Scan Database Only Twice

แสกน DB แค่ 2 ครั้ง

- 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

    มารวมกันเป็น global
  - Scan 2: consolidate global frequent patterns
  - See next figure
  - A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association in large databases. In *VLDB'95*

Mining by partitioning the data

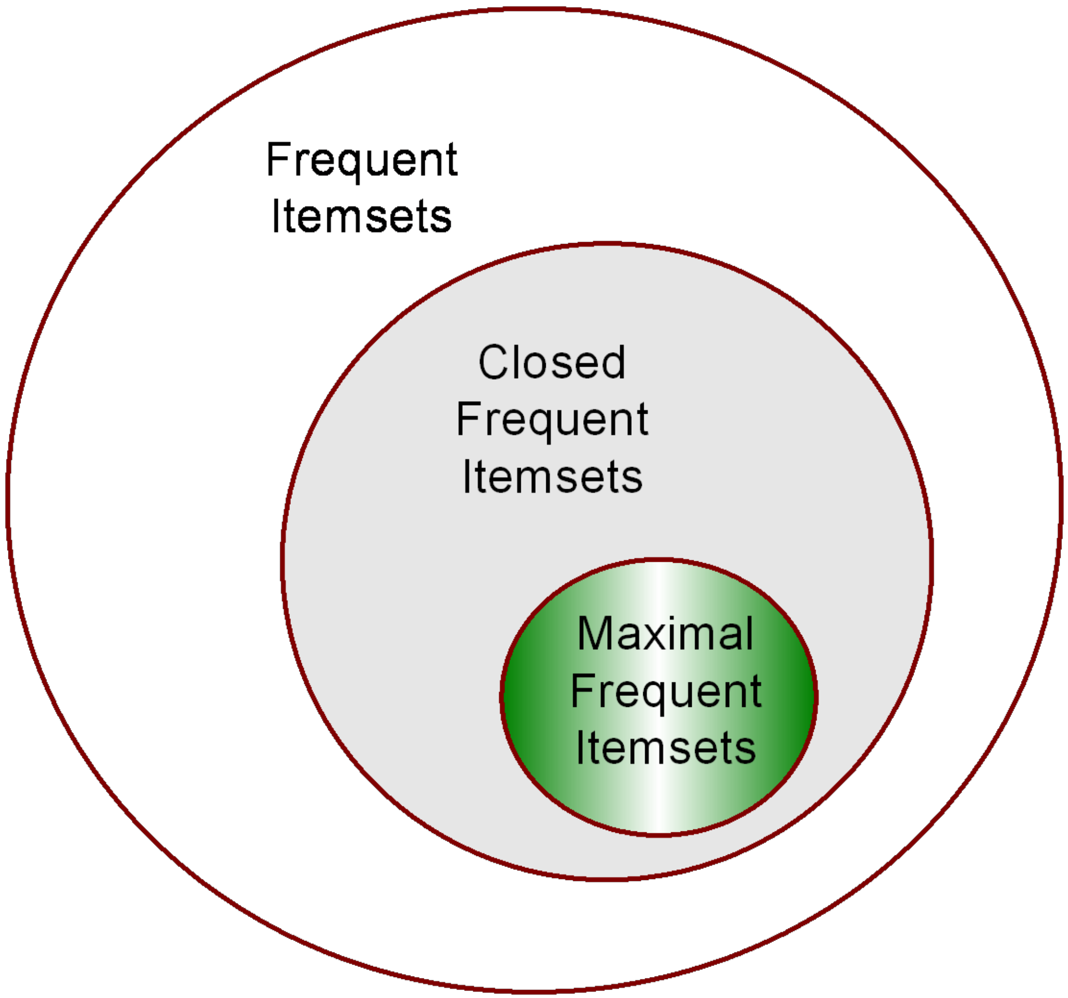


Sampling:mining a subset of data

สุ่มข้อมูล เอาแค่ sample มาใช้

- Select a sample of original database, mine frequent patterns within sample using Apriori
- The S sample size is such that the search for frequent itemsets in S can be done in main memory
  fit in
- Use lower support threshold than minimum support
- Scan database once to verify frequent itemsets found in sample.
  - If yes, finish.
  - If no, scan database again to find missed frequent patterns
- H. Toivonen. Sampling large databases for association rules. In *VLDB'96*

Reduce redundant ass rules : Maximal vs Closed Itemsets



ACE

$A \rightarrow CE$ $A \rightarrow C$ ลดก.ซ้ำซ้อน

$C \rightarrow AE$ $C \rightarrow E$

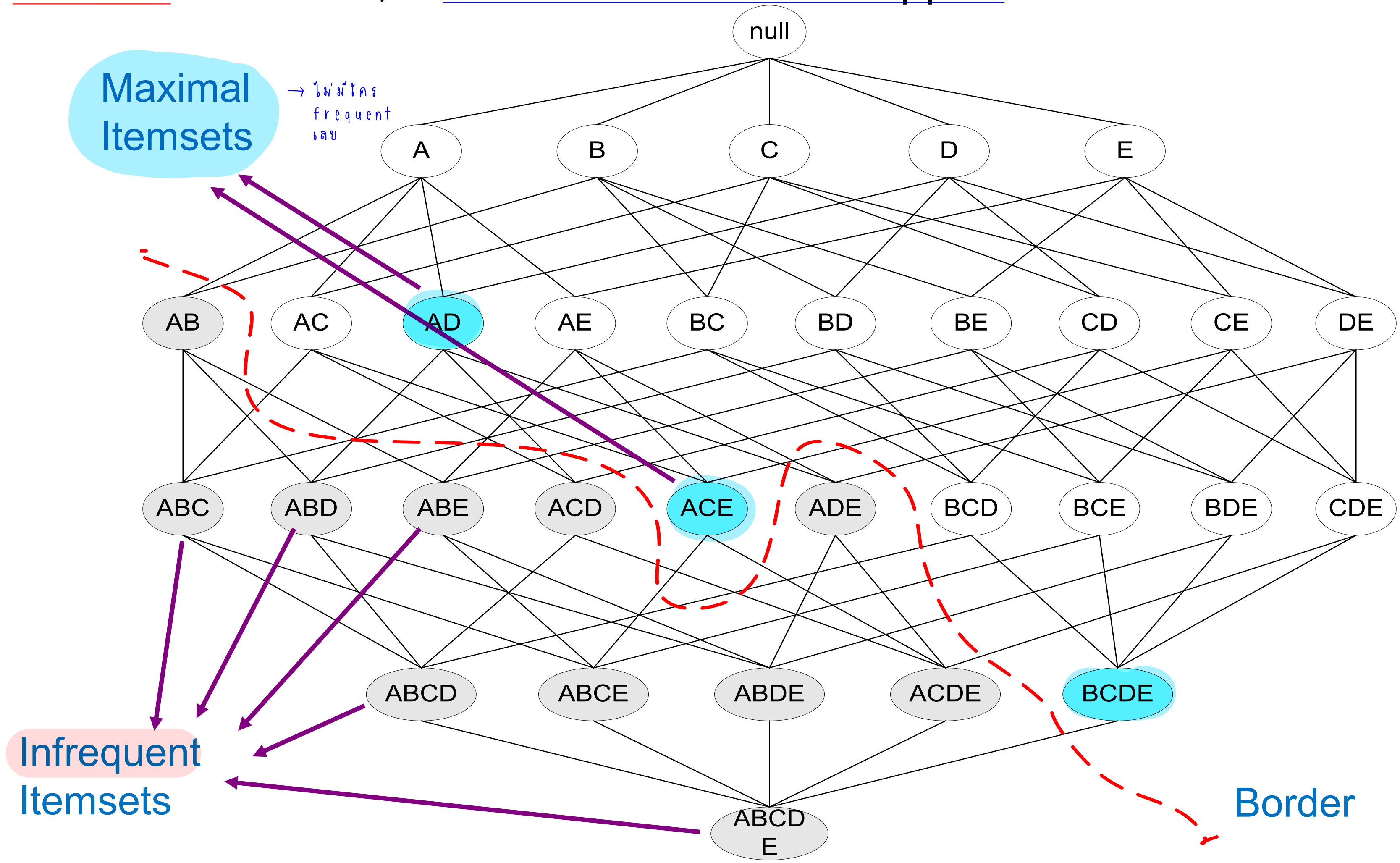
$E \rightarrow AC$

$\vdots$

Slide from P. Tan, M. Steinbach, and V. Kumar

Maximal Frequent Itemset

- An itemset is maximal frequent if none of its immediate supersets is frequent
- Maximal Frequent Itemset provides a compact representation of the frequent itemsets. However, it does not contain the support information of its subsets



Slide from P. Tan, M. Steinbach, and V. Kumar

Closed Itemset

มีค่า support count ไม่เหมือนกับมันเลย

- 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

เป็น closed

เป็น closed
ไม่มีตัวไหนเท่า

เป็น closed

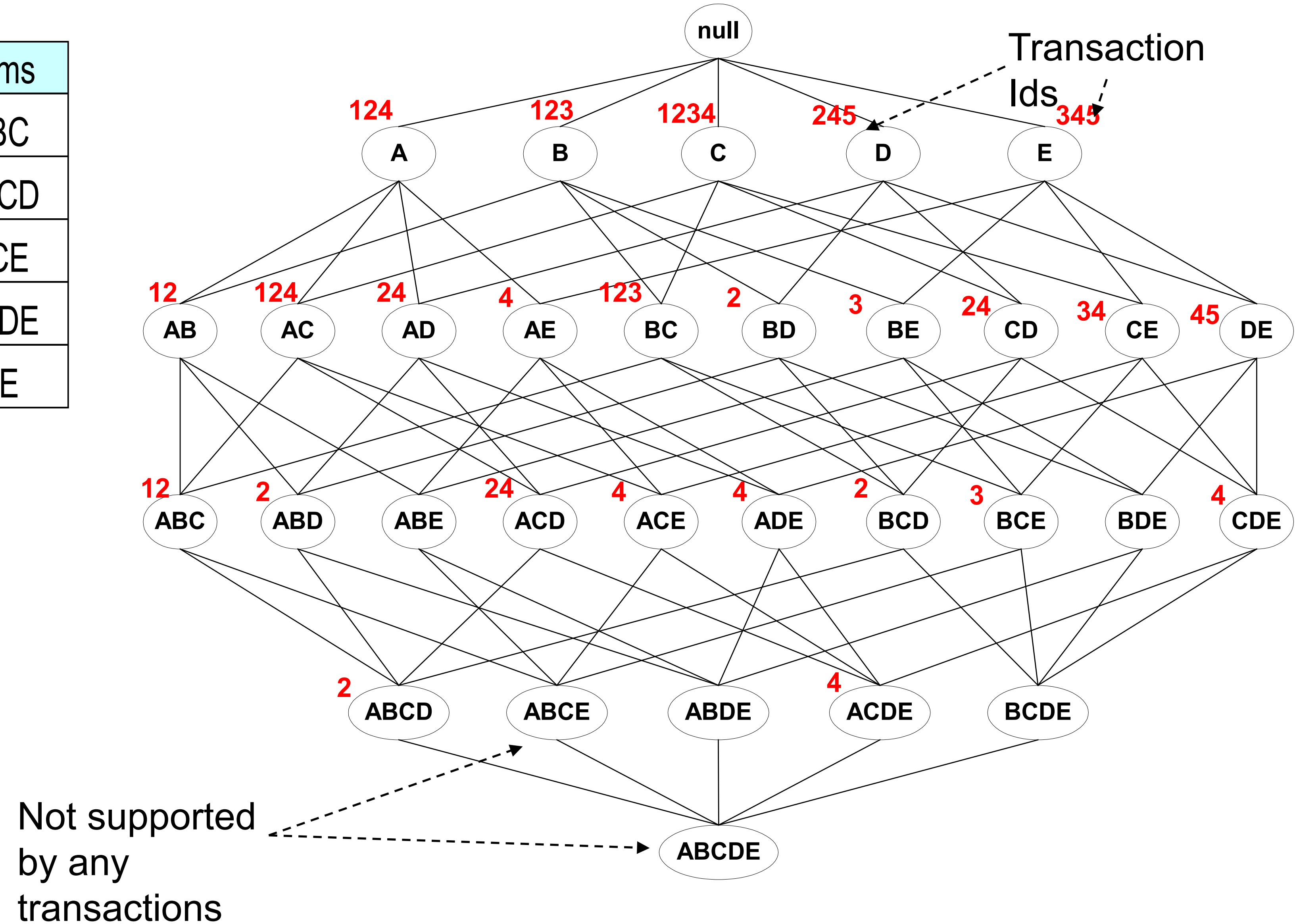
Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2

Closed frequent itemset

- It is a frequent itemset that is both closed and its support is greater than or equal to minsup.
- Provide a minimal representation of itemsets without losing their support infomation

Maximal vs Closed Itemsets

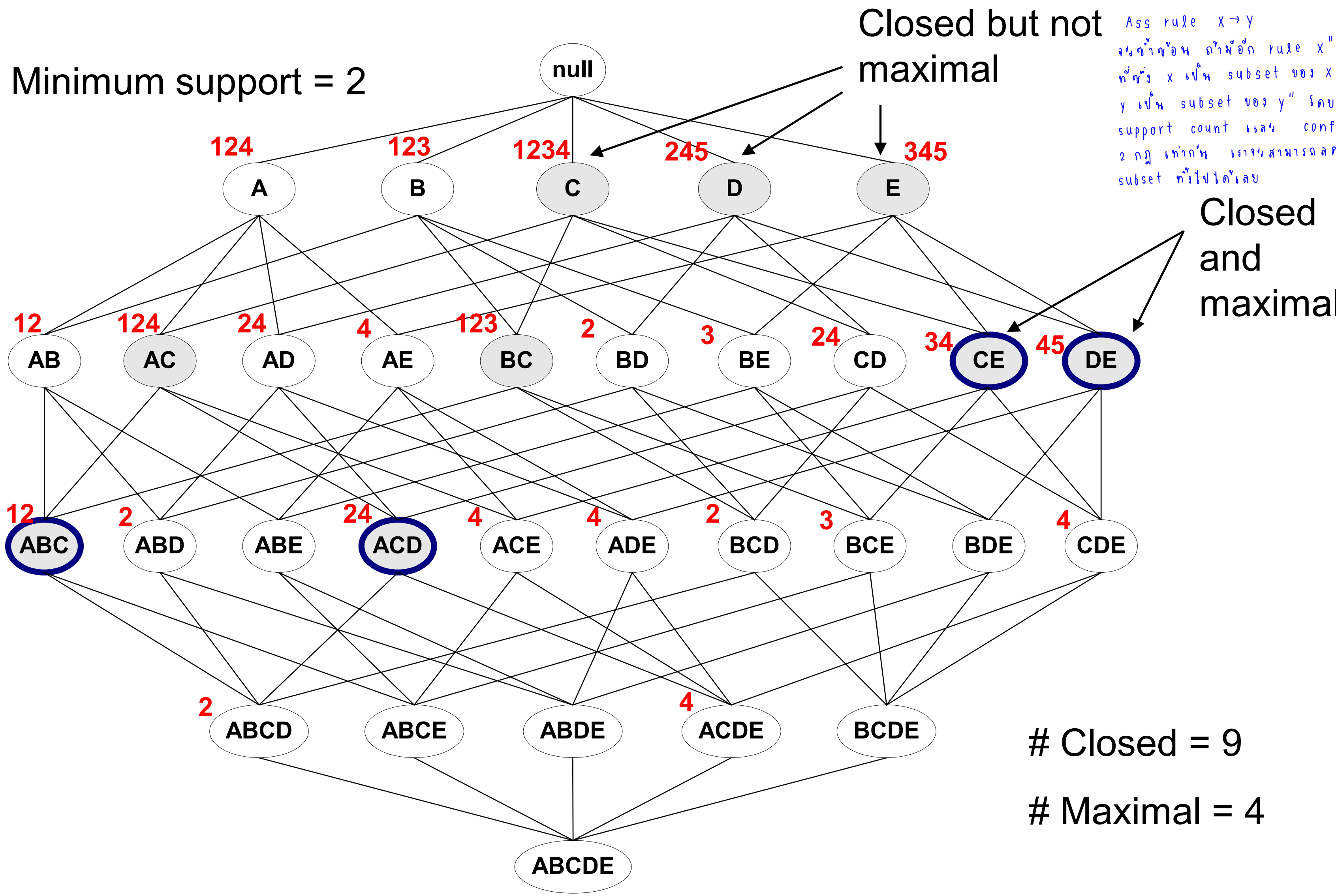
TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



Slide from P. Tan, M. Steinbach, and V. Kumar

Maximal vs Closed Frequent Itemsets

Ass rule x → y
จะซ้ำซ้อน ถ้ามีอีก rule x" → y"
ที่ซึ่ง x เป็น subset ของ x" และ
y เป็น subset ของ y" โดยที่มีค่า
support count และ conf. ของทั้ง
2 กฎ เท่ากัน เราจะสามารถลดกฎที่สร้างจาก
subset ทิ้งไปได้เลย



Slide from P. Tan, M. Steinbach, and V. Kumar

2. 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

2. Pattern-Growth Approach: Mining Frequent Patterns Without 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

Frequent Pattern-Growth Approach

- Adopts a divide-and-conquer strategy
- **First,** compress the database representing frequent items into a frequent-pattern tree (FP-tree), which retains the itemset association information
- **Then,** divide the compressed database into a set of conditional databases, each associated with one frequent item or "pattern fragment", and mine each such database separately

Construct FP-tree from a Transaction Database

TID	*Items bought*
100	*{f, a, c, d, g, i, m, p}*
200	*{a, b, c, f, l, m, o}*
300	*{b, f, h, j, o, w}*
400	*{b, c, k, s, p}*
500	*{a, f, c, e, l, p, m, n}*

min_support = 3

1. Scan DB once, find frequent 1-itemset (single item pattern)

   f = 4
   c = 4
   a = 3
   b = 3
   m = 3
   p = 3

ไม่เอาตัวที่ไม่เกิน minsup

เรียงลำดับ

2. Sort frequent items in frequency descending order, f-list

F-list = f-c-a-b-m-p

Construct FP-tree from a Transaction Database

เลือกใช้แต่ตัวที่เป็น frequent item set

3. Scan DB again, construct FP-tree

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

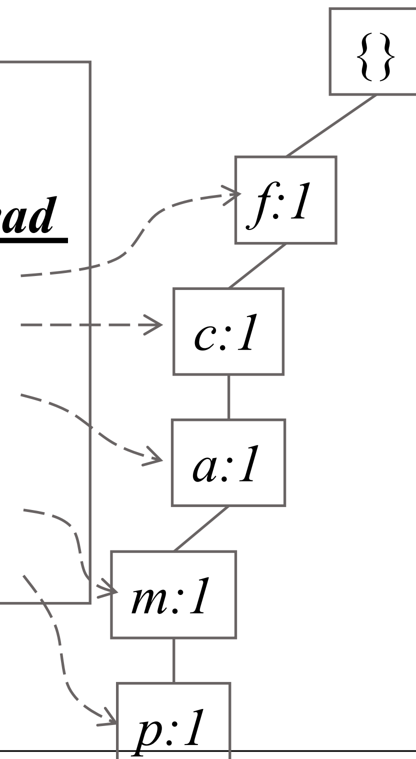
F-list = f-c-a-b-m-p

Construct FP-tree starting by the first transaction

Header Table

Item	*frequency*	*head*
f	*4*	
c	*4*	
a	*3*	
b	*3*	
m	*3*	
p	*3*	

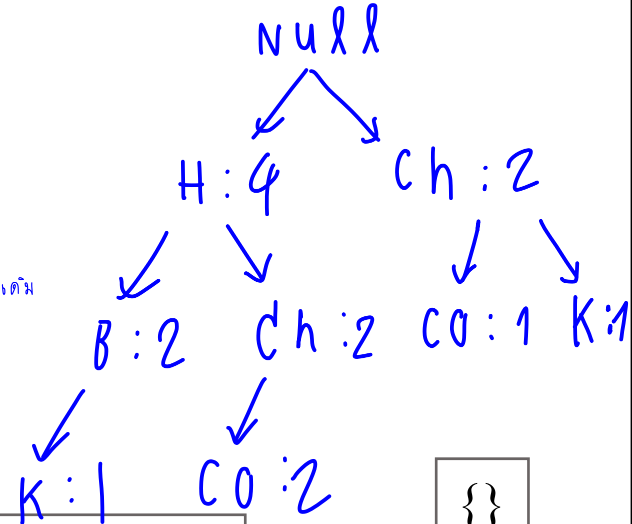
มาวาด tree



Construct FP-tree from a Transaction Database

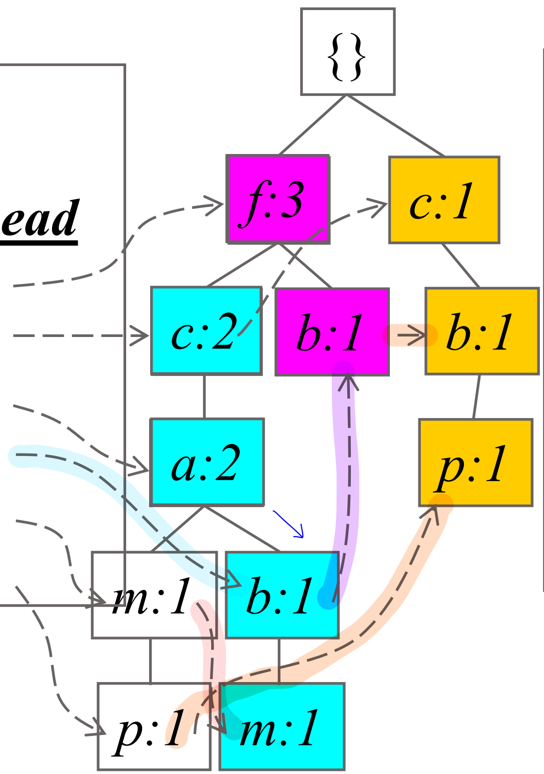
TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

ซ้ำกัน = ทางเดิม



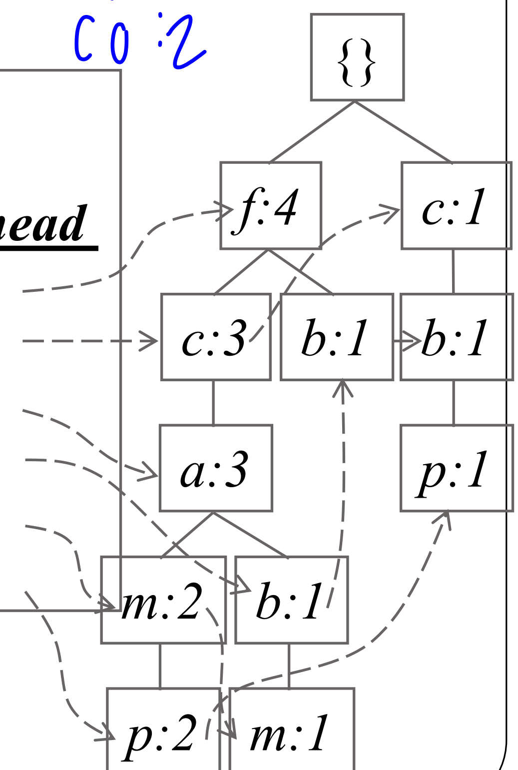
Header Table

Item	*frequency*	*head*
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



Header Table

Item	*frequency*	*head*
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



Partition Patterns and Databases

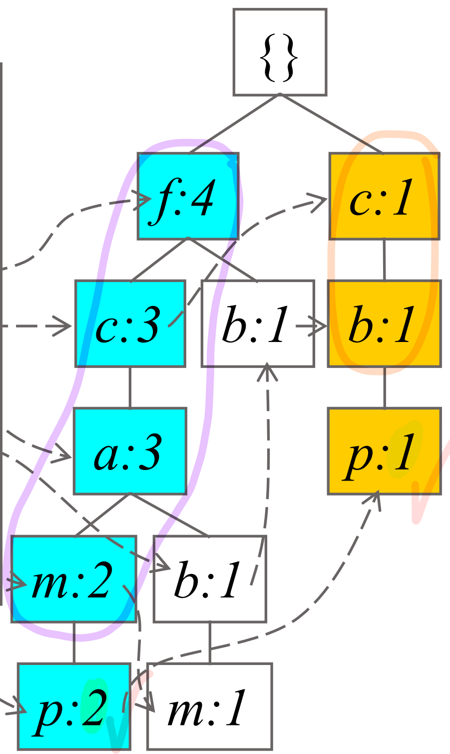
- Frequent patterns can be partitioned into subsets according to f-list
  - F-list = f-c-a-b-m-p
  - Patterns containing p
  - Patterns having m but no p
  - …
  - Patterns having c but no a nor b, m, p
  - Pattern f
- Completeness and non-redundancy

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

Header Table

Item	*frequency*	*head*
f	*4*	
c	*4*	
a	*3*	
b	*3*	
m	*3*	
p	*3*	



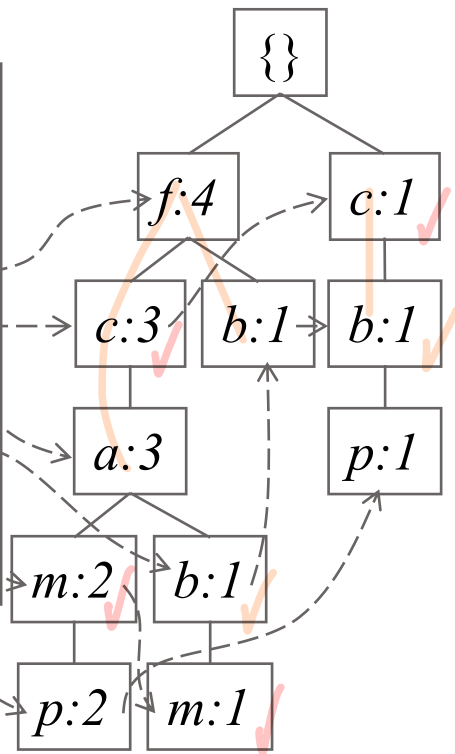
***Conditional* pattern bases**

item	*cond. pattern base*
p	*fcam:2, cb:1*

Find Patterns Having node n From node n-conditional Database

Header Table

Item	***frequency***	***head***
f	*4*	
c	*4*	
a	*3*	
b	*3*	
m	*3*	
p	*3*	



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
  - รวมข้อมูลที่เป็น count แล้วหยิบมาสร้างเป็น FP-tree
  - Accumulate the count for each item in the base
  - Construct the FP-tree for the frequent items of the pattern base

ตกไป

***m*-conditional pattern base:**

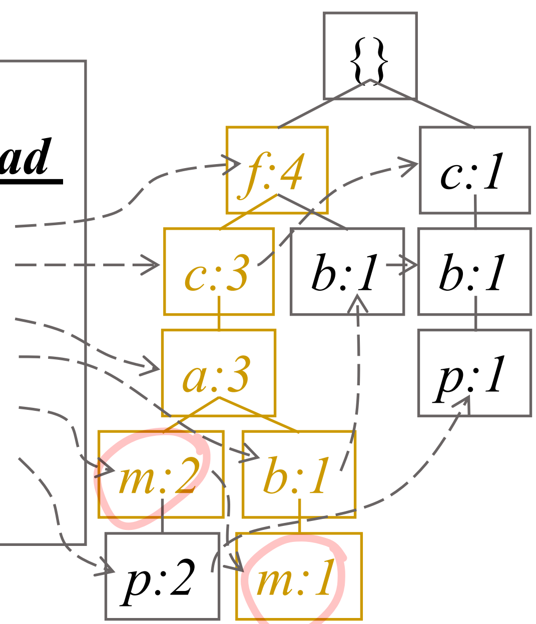
3 3 3 3 3 3 1

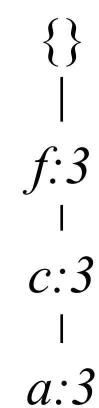
fca:2, fcab:1

Combine and leave b out

Header Table

Item	***frequency***	***head***
f	*4*	
c	*4*	
a	*3*	
b	*3*	
m	*3*	
p	*3*	





***m*-conditional FP-tree**

All frequent patterns relate to *m*

m,

fm, cm, am,

fcm, fam, cam,

fcam

pattern ทั้งหมดที่ relate กับ m

From Conditional Pattern-bases to Conditional FP-trees. Then, generate frequent patterns

Item	Conditional pattern-base	Conditional FP-tree
p	{(fcam:2), (cb:1)} c บวกกันได้ 3 ผ่าน	{(c:3)}|p
m	{(fca:2), (fcab:1)}	{(f:3, c:3, a:3)}|m
b	{(fca:1), (f:1), (c:1)}	Empty
a	{(fc:3)}	{(f:3, c:3)}|a
c	{(f:3)}	{(f:3)}|c
f	Empty	Empty

Recursion: Mining Each Conditional FP-tree

{}

|

f:3

|

c:3

|

a:3

m-conditional **FP-tree**

Cond. pattern base of "am": (fc:3)

{}

|

f:3

|

c:3

am-conditional **FP-tree**

Cond. pattern base of "cm": (f:3)

{}

|

f:3

cm-conditional **FP-tree**

Cond. pattern base of "cam": (f:3)

{}

|

f:3

cam-conditional **FP-tree**

Benefits of the FP-tree Structure

- Completeness ได้คำตอบครบ
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness คิดเฉพาะ frequent
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the *count* field)

The Frequent Pattern Growth Mining Method (conclusion)

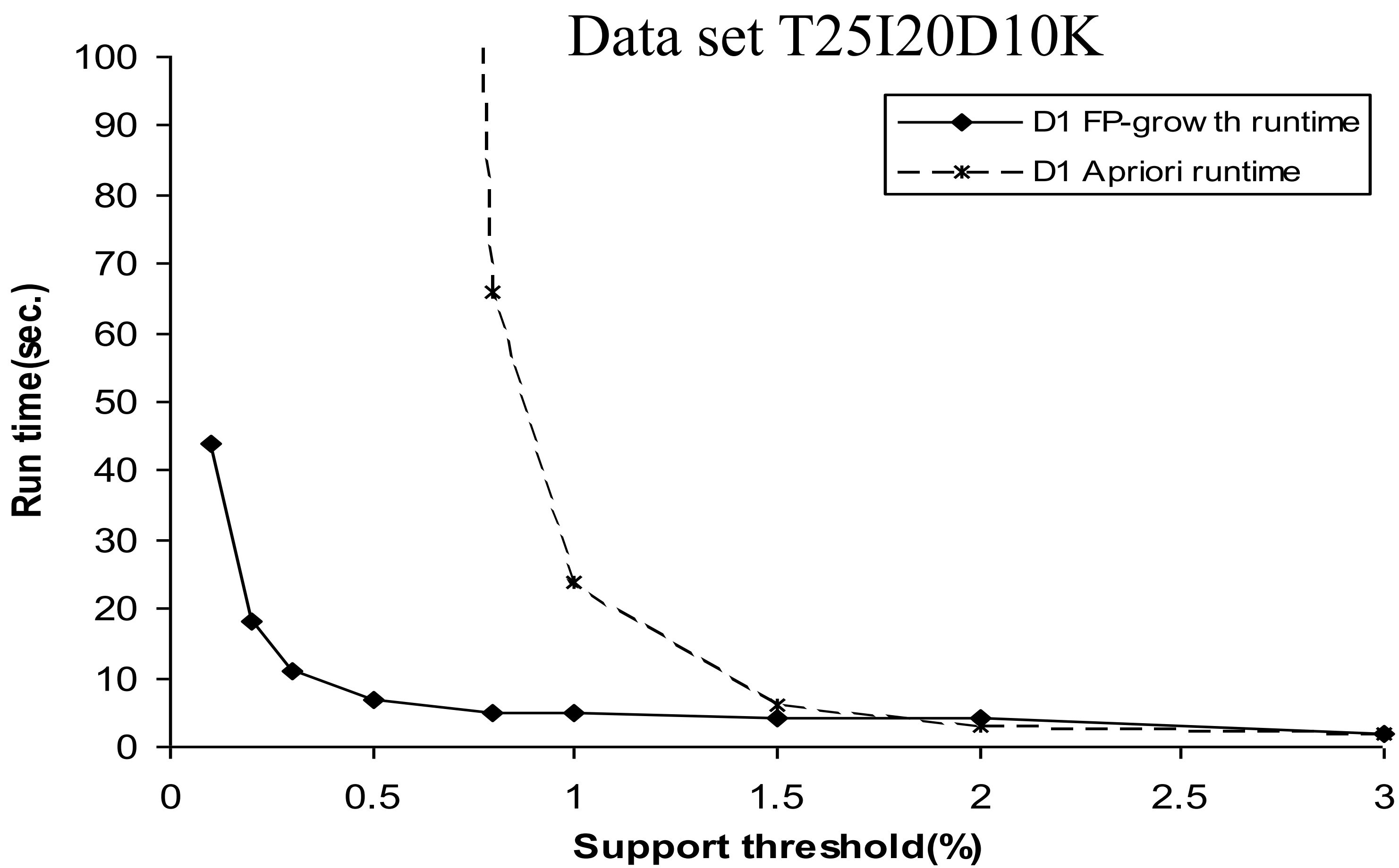
- Idea: Frequent pattern growth
  - Recursively grow frequent patterns by pattern and database partition
- Method
  - For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
  - Repeat the process on each newly created conditional FP-tree
  - Until the resulting FP-tree is empty, or it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

Scaling FP-growth by Database Projection

- What about if FP-tree cannot fit in memory?
  - DB projection   ข้อมูลเยอะ ไม่สามารถเก็บได้ใน mem
- First partition a database into a set of projected DBs
- Then construct and mine FP-tree for each projected DB   วิธีแก้
- Parallel projection vs. partition projection techniques
  - Parallel projection
    - Project the DB in parallel for each frequent item
    - Parallel projection is space costly
    - All the partitions can be processed in parallel
  - Partition projection
    - Partition the DB based on the ordered frequent items
    - Passing the unprocessed parts to the subsequent partitions

FP-Growth vs. Apriori: Scalability With the Support Threshold

ประสิทธิภาพดีกว่า



Advantages of the Pattern Growth Approach

- Divide-and-conquer: เป็นส่วนงานเป็นส่วนย่อย ๆ
  - Decompose both the mining task and DB according to the frequent patterns obtained so far
  - Lead to focused search of smaller databases
- Other factors
  - No candidate generation, no candidate test
  - Compressed database: FP-tree structure
  - No repeated scan of entire database
  - Basic ops: counting local freq items and building sub FP-tree, no pattern search and matching
- A good open-source implementation and refinement of FPGrowth
  - FPGrowth+ (Grahne and J. Zhu, FIMI'03)

3. ECLAT : Vertical data format approach

- ECLAT (Zaki et al. @KDD'97)
- Apriori and FP-growth methods mine frequent patterns from a set of transactions in TID-itemset format
  - Horizontal data format
    {TID:itemset}
- Vertical data format approach mine frequent patterns from a set of transactions in item-TIDset format
  - Vertical data format
    {item:TIDset}

Vertical data format approach

- Transform data from horizontal data format into vertical data format by scanning DB once and find frequent 1-itemsets
- Intersect the TIDsets of every pair of frequent 1-item and find frequent 2-itemsets
- Repeat the intersection steps until cannot find any more frequent itemsets
- Union all frequent 1-itemsets, 2-itemsets, … to get frequent itemsets of this dataset

Vertical data format approach

- The method is still 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
- To find a frequent itemset, the support count of each itemset must be greater than the minimum support
- The support count of an itemset is the length of the TIDset of the itemset. So, no need to scan DB to count the support count. However, the set may be quite long and take substantial memory space as well as computation time for intersecting the long sets

Example

- A set of transactions in TID-itemset format

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

Min_sup = 2

Example

- Transform to the vertical data format
- Find frequent 1-itemsets

itemset	TIDset
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}

- support count is the length of the TIDset
- all 1-itemsets are frequent because the support count of each itemset is greater than the min_sup (2)

Example

- Based on Apriori property, use k-itemsets to construct the candidate (k+1)-itemsets by intersecting TIDset of frequent k-itemsets to compute the TIDset of (k+1)-itemset

itemset	TIDset	
{I1, I2}	{T100, T400, T800, T900}	ดูจาก ค.ยาว
{I1, I3}	{T500, T700, T800, T900}	
~~{I1, I4}~~	~~{T400}~~	1 < minsup
{I1, I5}	{T100, T800}	
{I2, I3}	{T300, T600, T800, T900}	
{I2, I4}	{T200, T400}	
{I2, I5}	{T100, T800}	
~~{I3, I5}~~	~~{T800}~~	1 < minsup

- Count the support count to generate frequent 2-itemsets

Example

- Repeat until no more candidate itemsets can be generated

itemset	TIDset
{I1, I2}	{T100, T400, T800, T900}
{I1, I3}	{T500, T700, T800, T900}
{I1, I5}	{T100, T800}
{I2, I3}	{T300, T600, T800, T900}
{I2, I4}	{T200, T400}
{I2, I5}	{T100, T800}

Frequent 2-itemsets

itemset	TIDset	
{I1, I2, I3}	{T800, T900}	หา intersec
{I1, I2, I5}	{T100, T800}	

Frequent 3-itemsets

I2, I3, I4 X

I2, I3, I5 X ได้ T800

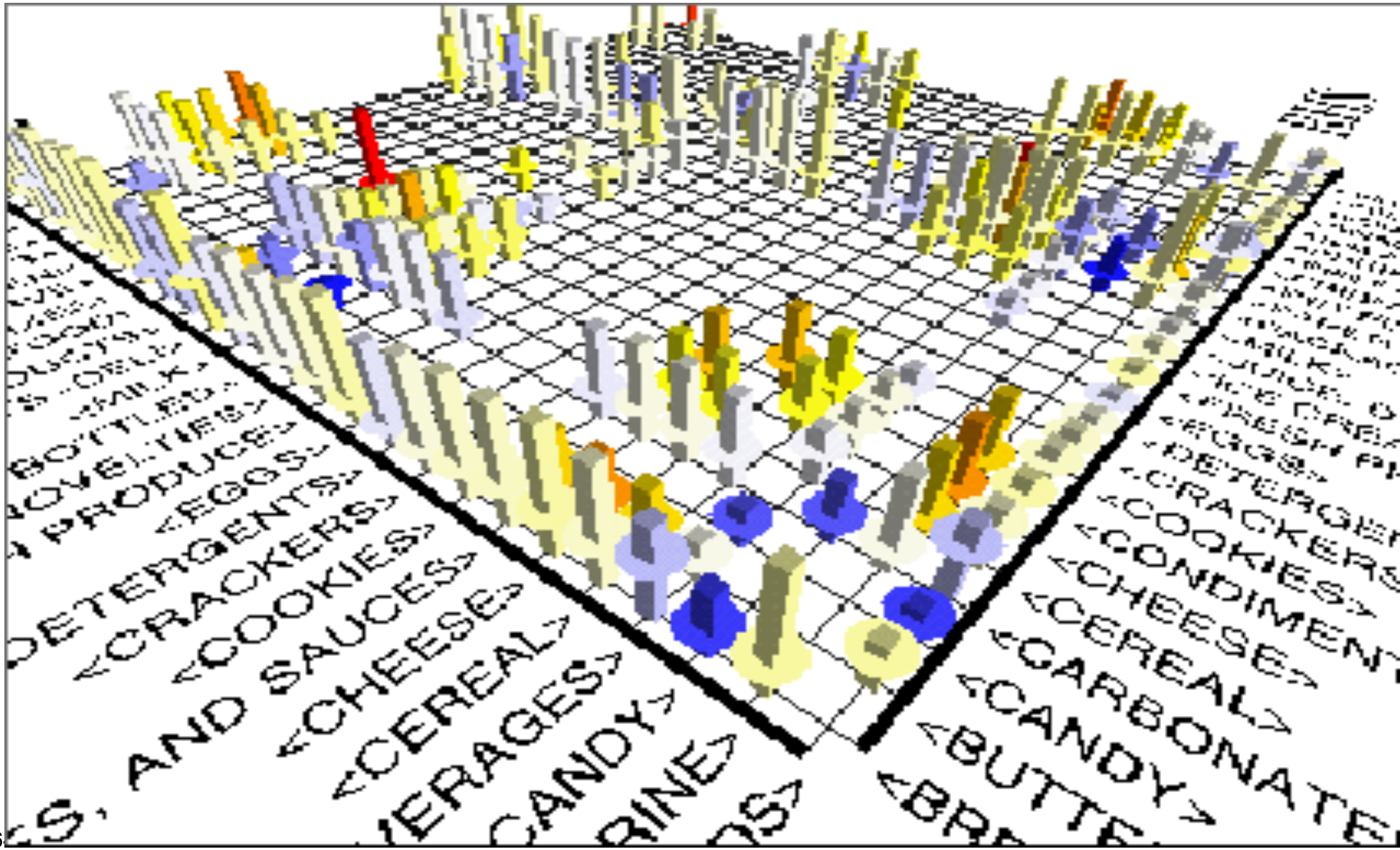
Example

- all frequent itemsets are Union

  {I1}, {I2}, {I3}, {I4}, {I5}, {I1, I2}, {I1, I3}, {I1, I5}, {I2, I3}, {I2, I4}, {I2, I5}, {I1, I2, I3}, {I1, I2, I5}

หา Ass rules

Visualization of Association Rules (SGI/MineSet 3.0)



Mining Various Kinds of Association Rules

- Mining multilevel association

level → subcategory ของสินค้าชิ้นนั้น ยังเกี่ยวกับการ buy

ตะกร้าซื้อ item 1, item 2 → item 3

พร้อมๆ

- Miming multidimensional association

ตาราง > 1 attr.

- Mining quantitative association

dynamic discretization

nominal

numeric

age 13

18

25

discretization

Mining Multiple-Level Association Rules

- Involve concepts at different levels of abstraction
- It is difficult to find strong associations among data items at low or primitive levels of abstraction due to the sparsity of data at those levels
- Strong associations discovered at high levels of abstraction may represent commonsense knowledge (or may seem novel to another)

Uniform support

minsup ค่าเดียว ไม่ได้ผล

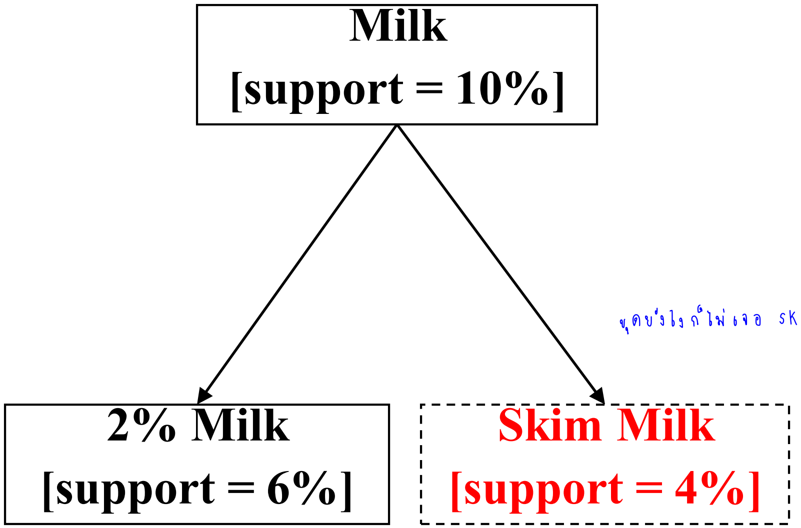
Adv : use only one value of minimum support -> simplified the search procedure

Disadv : items at low levels doesn’t occur as often as those at higher levels

- if min_sup is set too high, the strong associations at low levels cannot be found
- if min_sup is set too low, it may generate many uninteresting associations at high level ไม่ค่อยน่าสนใจ

Uniform support

Level 1
min_sup = 5%



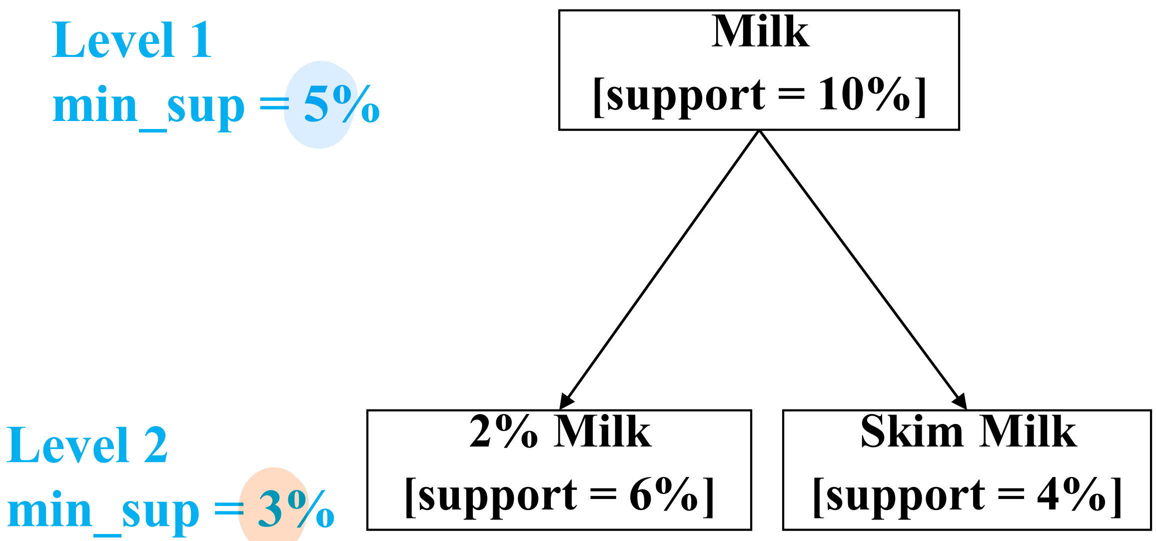
ขุดยังไงก็ไม่เจอ Skim Milk

Level 2
min_sup = 5%

Reduced support

วิธีแก้

- Flexible support settings
  - Items at the lower level are expected to have lower support



Disadvantage of mining multilevel association rules

- Generation of many redundant rules across multiple levels of abstraction due to the "ancestor" relationships among items
- milk -> wheat bread [sup=8%,conf=70%]
- 2%milk -> wheat bread [sup=2%, conf=72%] $\frac{1}{4} = 25\%$
- The first rule is an ancestor of the second rule
- Redundant if its support and confidence are close to their "expected" values, based on the rule's ancestor
- ex ¼ of sales of milk is 2% milk, the second rule is uninteresting!!!

Mining Multi-Dimensional Association

- Single-dimensional rules:

  buys(X, “milk”) ⇒ buys(X, “bread”)

  มากกว่า 1 dimension

- Multi-dimensional rules: ≥ 2 dimensions or predicates
  - Inter-dimension assoc. rules (*no repeated predicates*)

    นักเรียนอายุ 19-25 จะซื้อ coke กิน

age(X,”19-25”) ∧ occupation(X,“student”) ⇒ buys(X, “coke”)

  - hybrid-dimension assoc. rules (*repeated predicates*)

    อายุ 19-25 จะซื้อ popcorn คู่กับ coke

age(X,”19-25”) ∧ buys(X, “popcorn”) ⇒ buys(X, “coke”)

Mining Multi-Dimensional Association

- Categorical Attributes: finite number of possible values, no ordering among values

  Ex occupation, brand, color

  categorical attributes are sometimes called nominal attributes because their values are "names of things"

- Quantitative Attributes: Numeric, implicit ordering among values

  Ex age, income, price

Techniques for mining multi-dimensional association

- Techniques can be categorized by how numerical attributes, such as age or salary are treated

ดีไซน์ช่วงไว้อยู่แล้ว

1. Static discretization based on predefined concept hierarchies (data cube methods) -> this technique occurs before mining

   ex income : replace numeric values by interval labels “0..20K”, “21K..30K” (treated as categorical attributes)

2. Dynamic discretization based on data distribution -> this technique occurs during mining process

Mining multi-dimensional association using static Discretization of Quantitative Attributes

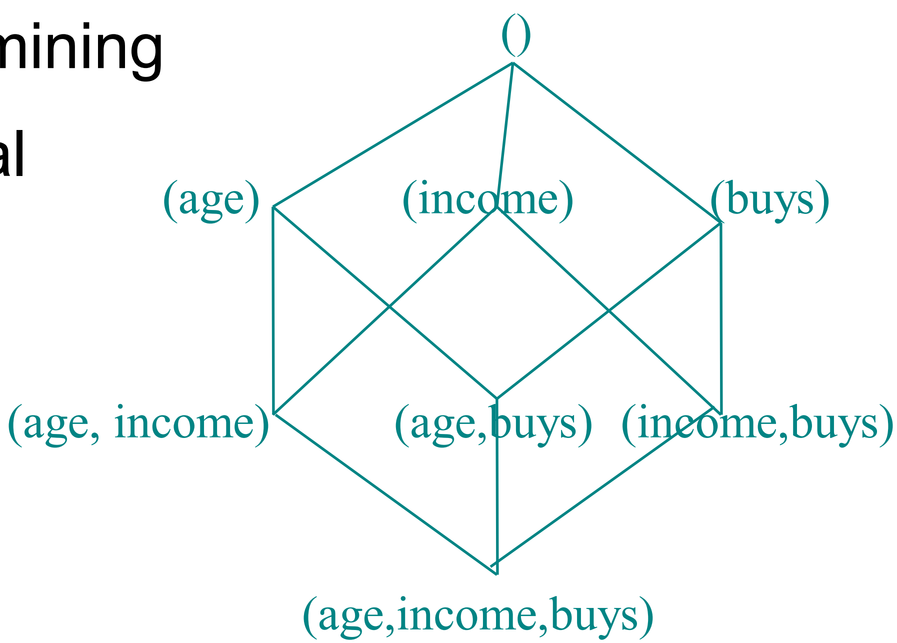
- Discretized prior to mining using concept hierarchy.
- Numeric values are replaced by ranges

age(X,”19-25”) $\wedge$ occupation(X,“student”) $\Rightarrow$ buys(X, “coke”)

- Find frequent k-predicate sets instead of frequent k-itemsets
- A k-predicate set is a set containing k conjuctive predicates ex 3-predicate set : {age, occupation, buys}
- If {age, occupation, buys} is a frequent 3-predicate set, {age, occupation}, {age, buys}, {occupation, buys} are also frequent

Mining multi-dimensional association using static Discretization of Quantitative Attributes

- Use Apriori algo. to find frequent k-predicate sets
- Every subset of frequent k-predicate sets must be frequent
- In relational database, finding all frequent k-predicate sets will require *k* or *k*+1 table scans
- Data cube is well suited for mining
- The cells of an n-dimensional cuboid correspond to the predicate sets
- Mining from data cubes can be much faster



Quantitative Association Rules

- Mining multi-dimensional association using dynamic discretization of quantitative attributes is mining for quantitative association rules
- Numeric attributes are *dynamically* discretized during mining process (using binning techniques) so as to satisfy some mining criteria
  - Maximizing the confidence or compactness of the rules mined
- The intervals are dynamic in that they may later be further combined during the mining process
- For example, 2-D quantitative association rules:

$$A_{quan1} \wedge A_{quan2} \Rightarrow A_{cat}$$

<u>Example</u>

age(X, "34-35") $\wedge$ income(X, "31-50K") $\Rightarrow$ buys(X, "high resolution TV")

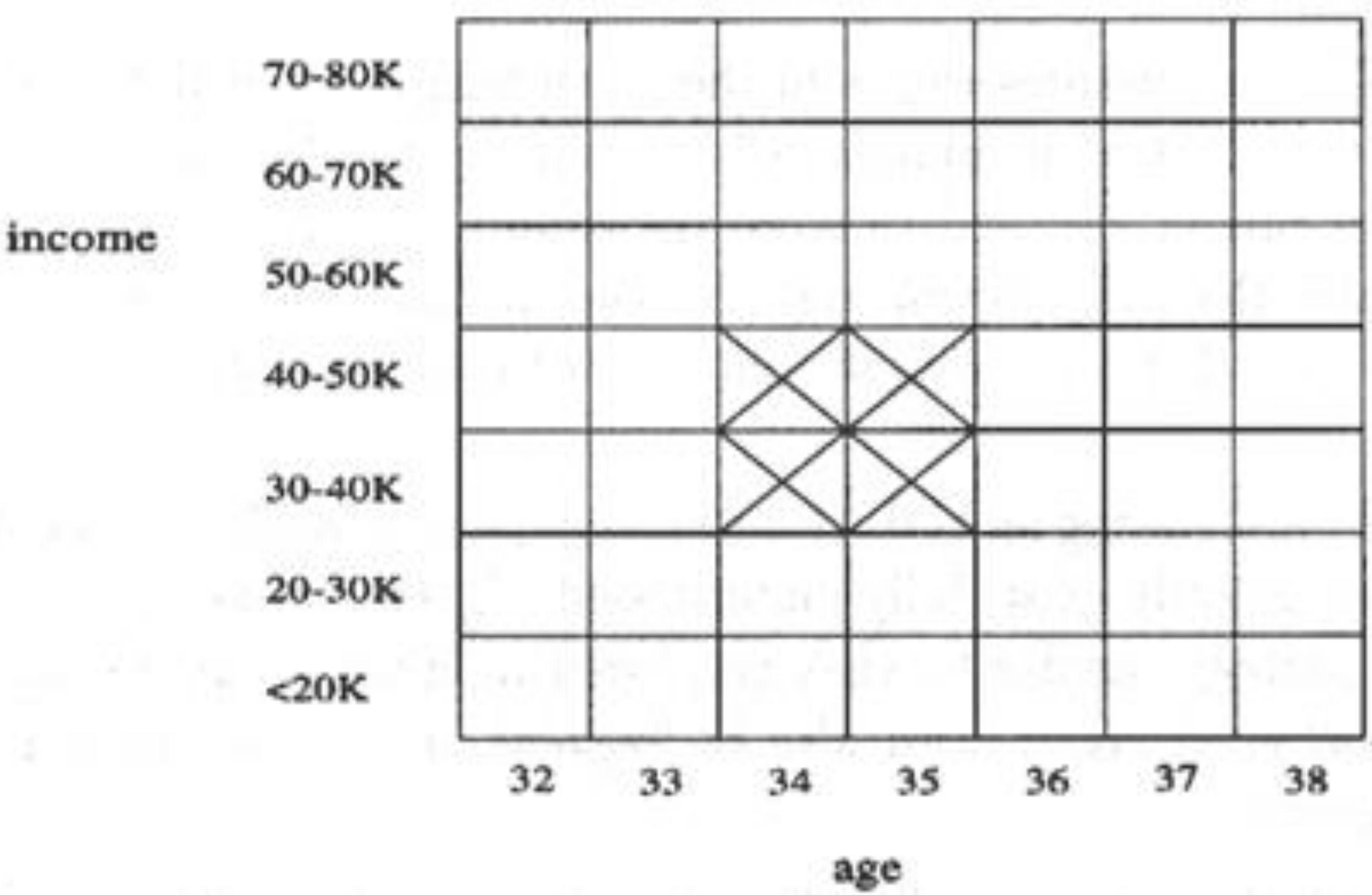
Quantitative Association Rules

- Cluster *adjacent* association rules to form general rules using a 2-D grid
- Finding frequent predicate sets : once the 2-D array containing the count distribution for each category is set up, it can be scanned to find the frequent predicate sets (those satisfying min_sup) that also satisfy min_conf. Then, generate strong ass. rule
- Clustering the association rules : the strong ass. rule are then mapped to a 2-D grid

<u>Example</u>

age(X, “34”) $\wedge$ income(X, “31-40K”) $\Rightarrow$ buys(X, “high resolution TV”)
age(X, “35”) $\wedge$ income(X, “31-40K”) $\Rightarrow$ buys(X, “high resolution TV”)
age(X, “34”) $\wedge$ income(X, “41-50K”) $\Rightarrow$ buys(X, “high resolution TV”)
age(X, “35”) $\wedge$ income(X, “41-50K”) $\Rightarrow$ buys(X, “high resolution TV”)

Quantitative Association Rules



Four rules can be combined or clustered to form a new rule

age(X, "34-35") $\wedge$ income(X, "31-50K") $\Rightarrow$ buys(X, "high resolution TV")

From association mining to correlation analysis

- How can we tell which strong ass. rules are really interesting?
- Support and confidence are not good to indicate correlations
- Measure of dependent/correlated events: lift

$$lift(A,B) = \frac{P(A \cup B)}{P(A)P(B)}$$

ดูว่ามันน่าสนใจจริง ๆ มั้ย

- If result < 1, the occurrence of A is negatively correlated with the occurrence of B
- If result > 1, positively correlated
- If result = 1, A and B are independent

Interestingness Measure: Correlations (Lift)

sup confi

- *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%] is misleading
  - The overall % of students eating cereal is 75% > 66.7%. ไม่น่าสนใจ เพราะปกติเค้ากิน cereal กันอยู่แล้ว
- *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%] is more accurate, although with lower support and confidence
- Measure of dependent/correlated events: lift

$$lift(A,B) = \frac{P(A \cup B)}{P(A)P(B)}$$

	Basketball	Not basketball	Sum (row)
Cereal	2000	1750	3750
Not cereal	1000	250	1250
Sum(col.)	3000	2000	5000

$$lift(A,B) = \frac{2000/5000}{3000/5000 * 3750/5000} = 0.89$$

negative

$$lift(A,\neg B) = \frac{1000/5000}{3000/5000 * 1250/5000} = 1.33$$

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
  - System 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: type of knowledge to be mined
  - classification, association, etc.
- Data constraint: specify the set of task-relevant data — using SQL-like queries
  - find product pairs sold together in stores in Chicago in Dec.'02
- Dimension/level constraint: specify dimension, level of concept hierarchies
  - in relevance to region, price, brand, customer category
- Rule (or pattern) constraint: specify rule template, value of attributes
  - small sales (price < $10) triggers big sales (sum > $200)
- Interestingness constraint: specify thresholds
  - strong rules: min_support ≥ 3%, min_confidence ≥ 60%

Frequent-Pattern Mining: Summary

- Frequent pattern mining—an important task in data mining
- Scalable frequent pattern mining methods
  - Apriori (Candidate generation & test)
  - Projection-based (FPgrowth)
  - Vertical format approach
- Mining a variety of rules and interesting patterns
- From association mining to correlation analysis
- Constraint-based mining

References

- J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 3rd ed., 2012.
- P. Tan, M. Steinbach, and V. Kumar, ***Introduction to Data Mining***, Addison-Wesley, 2018.