Chapter 5

Mining Frequent Patterns, Associations, and Correlations

頻繁樣式, 關聯, 相互關係的探勘

Outlines

1. Basic concepts and a road map
2. Efficient and scalable frequent itemset mining methods
3. Mining various kinds of association rules
4. From association mining to correlation analysis
5. Constraint-based association mining
6. Summary

1. **Basic concepts and a road map**
2. Efficient and scalable frequent itemset mining methods
3. Mining various kinds of association rules
4. From association mining to correlation analysis
5. Constraint-based association mining
6. Summary

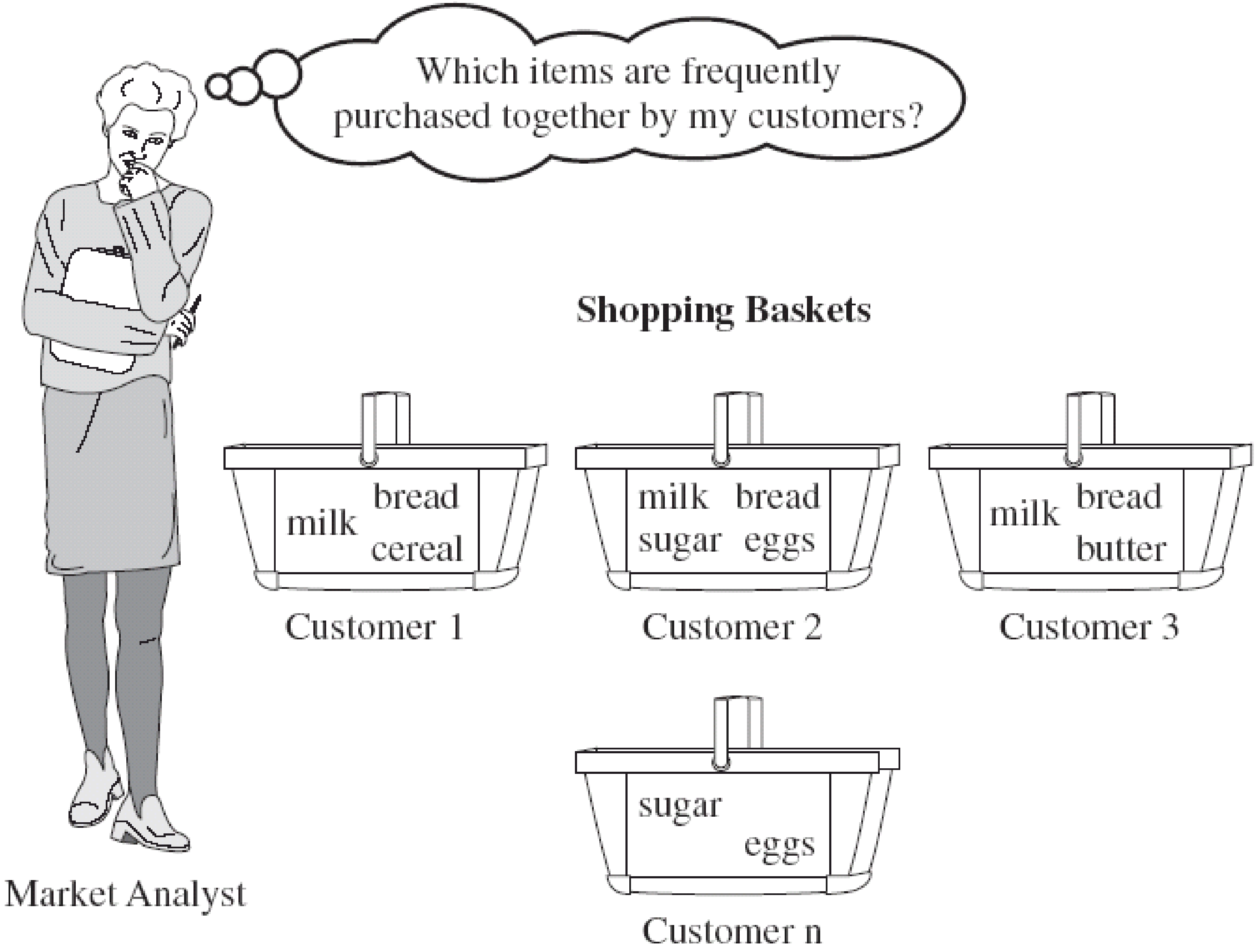
What Is Frequent Pattern Analysis?

- 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.

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

5.1.1 Market Basket Analysis (購物籃分析): A Motivating Example

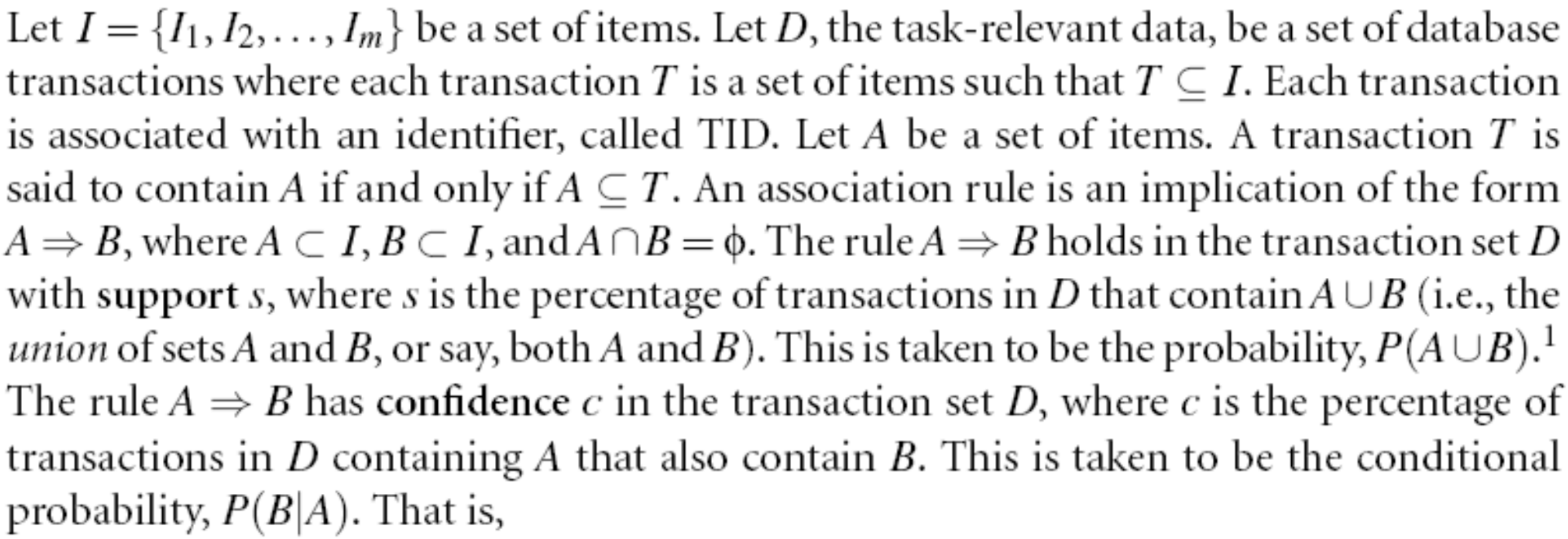


- If we think of the universe as the set of items available at the store, then each item has a Boolean variable representing the presence or absence of that item.
- Each basket can then be represented by a Boolean vector of values assigned to these variables.
- The Boolean vectors can be analyzed for buying patterns that reflect items that are frequently *associated* or purchased together.
- These patterns can be represented in the form of association rules. For example, the information that customers who purchase computers also tend to buy antivirus software at the same time is represented in
- Association Rule (5.1) below:

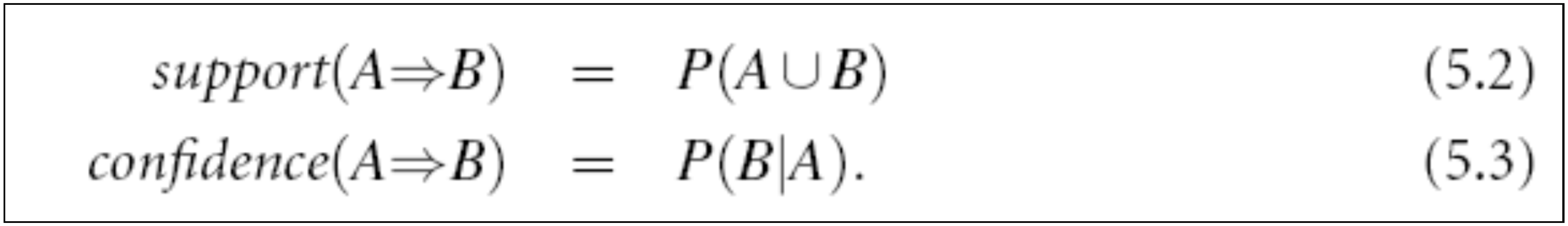
$$\boxed{computer \Rightarrow antivirus\_software \;\; [support = 2\%, confidence = 60\%] \qquad (5.1)}$$

- Rule support and confidence are two measures of rule interestingness. They respectively reflect the usefulness and certainty of discovered rules.
- A support of 2% for Association Rule (5.1) means that 2% of all the transactions under analysis show that computer and antivirus software are purchased together (一起購買).
- A confidence of 60% means that **60% of the customers who purchased a computer (先決條件是要買電腦)** also bought the software.
- Typically, association rules are considered interesting if they satisfy both a **minimum support threshold** (最小支持度) and a **minimum confidence threshold** (最小信賴度). Such thresholds can be set by users or domain experts.
- Additional analysis can be performed to uncover interesting statistical correlations between associated items.

5.1.2 Frequent Itemsets, Closed Itemsets, and Association Rules

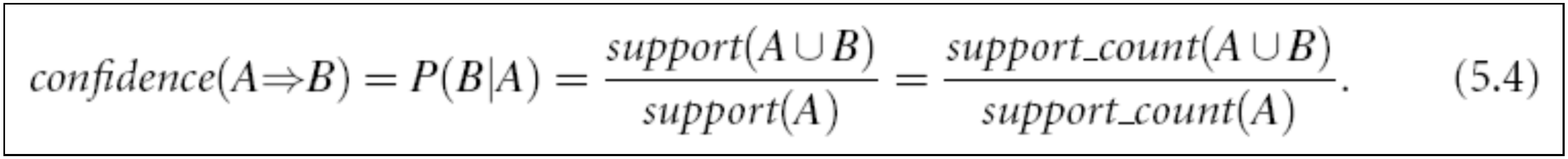


Let $I = \{I_1, I_2, \ldots, I_m\}$ be a set of items. Let D, the task-relevant data, be a set of database transactions where each transaction T is a set of items such that $T \subseteq I$. Each transaction is associated with an identifier, called TID. Let A be a set of items. A transaction T is said to contain A if and only if $A \subseteq T$. An association rule is an implication of the form $A \Rightarrow B$, where $A \subset I$, $B \subset I$, and $A \cap B = \phi$. The rule $A \Rightarrow B$ holds in the transaction set D with **support** s, where s is the percentage of transactions in D that contain $A \cup B$ (i.e., the *union* of sets A and B, or say, both A and B). This is taken to be the probability, $P(A \cup B)$.[1] The rule $A \Rightarrow B$ has **confidence** c in the transaction set D, where c is the percentage of transactions in D containing A that also contain B. This is taken to be the conditional probability, $P(B|A)$. That is,



$$support(A \Rightarrow B) = P(A \cup B) \tag{5.2}$$

$$confidence(A \Rightarrow B) = P(B|A). \tag{5.3}$$

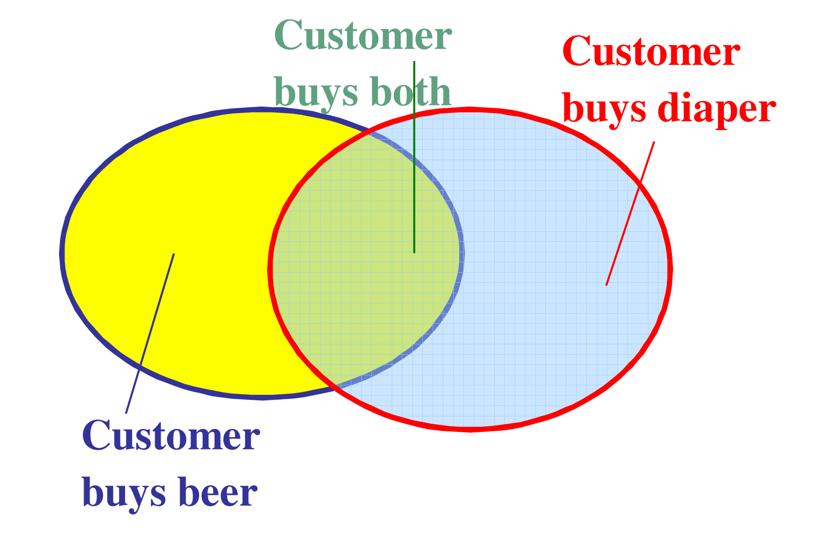


$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)} = \frac{support\_count(A \cup B)}{support\_count(A)}. \tag{5.4}$$

- A set of items is referred to as an ***itemset***.
- An itemset that contains k items is a **k-itemset**.
  - i.e., The set {computer, antivirus software} is a 2-itemset.
- The occurrence frequency of an itemset is the number of transactions that contain the itemset. This is also known, simply, as the frequency, support count, or count of the itemset.
- Note that the itemset support defined in Equation (5.2) is sometimes referred to as relative support (相對支持度), whereas the occurrence frequency is called the absolute support (絕對支持度).
- If the relative support of an itemset I satisfies a prespecified minimum support threshold (i.e., the absolute support of I satisfies the corresponding minimum support count threshold), then I is a **frequent itemset (頻繁項目集)**.
- The set of frequent k-itemsets is commonly denoted by L_k.

Example:

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F



- Itemset $X = \{x_1, \ldots, x_k\}$
- Find all the rules *X* *Y* with minimum support and confidence
  - support, *s*, probability that a transaction contains $X \cup Y$
  - confidence, *c*, conditional probability that a transaction having X also contains *Y*

Let $sup_{min} = 50\%$, $conf_{min} = 50\%$

Freq. Pat.: {*A:3, B:3, D:4, E:3, AD:3*}

Association rules:

A *D*

D *A*

association rule mining can be viewed as a two-step process:

p.231

1. Find all frequent itemsets
(找出所有的頻繁項目集):
By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, min sup.
2. Generate strong association rules from the frequent itemsets
(從頻繁項目集產生強關聯規則):
By definition, these rules must satisfy minimum support and minimum confidence.

Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., {a_1, …, a_{100}} contains:

$$\boxed{\binom{100}{1} + \binom{100}{2} + \cdots + \binom{100}{100} = 2^{100} - 1 \approx 1.27 \times 10^{30}}$$

- Solution: *Mine closed patterns and max-patterns instead*
- An itemset X is **closed** if X is *frequent* and there exists *no super-pattern* Y has *the same support* count as X in S.
- An itemset X is a **closed frequent itemset** (緊密頻繁項目集) in set S if X is both closed and frequent in S.
- An itemset X is a **maximal frequent itemset** (最大頻繁項目集) if X is frequent and there exists no frequent super-pattern Y s.t. $X \subset Y$and Y is frequent in S.
- Closed pattern is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules

Example 5.2 Closed Patterns and Max-Patterns

- 資料庫有二筆交易記錄,
  DB = $\{<a_1, \ldots, a_{100}>, < a_1, \ldots, a_{50}>\}$
  - 假設 Min_sup = 1.
- What is the set of closed frequent itemset?
  - $C=\{\{a_1, a_2, \ldots, a_{100}\}: 1\}:1 ; \{a_1, \ldots, a_{50}\}: 2\}$
- What is the set of maximal frequent itemset?
  - $M=\{\{a_1, \ldots, a_{100}\}: 1\}$
- What is the set of all patterns?
  - !!

5.1.3 Frequent Pattern Mining:
A Road Map

1. Based on the *completeness* of patterns to be mined
2. Based on the *levels of abstraction* involved in the rule set
3. Based on the *number of data dimensions* involved in the rule

$$buys(X, \text{“}computer\text{”}) \Rightarrow buys(X, \text{“}antivirus\_software\text{”}) \qquad (5.8)$$

一個維度 (buys)

$$age(X, \text{“}30\ldots39\text{”}) \wedge income(X, \text{“}42K\ldots48K\text{”}) \Rightarrow buys(X, \text{“}high\ resolution\ TV\text{”}). \qquad (5.9)$$

三個維度 (age, income, buys)

4. Based on the *types of values* handled in the rule
5. Based on the kinds of *rules* to be mined
6. Based on the *kinds of patterns* to be mined

1. Basic concepts and a road map
2. **Efficient and scalable frequent itemset mining methods**
3. Mining various kinds of association rules
4. From association mining to correlation analysis
5. Constraint-based association mining
6. Summary

5.2.1 The Apriori Algorithm: Finding Frequent Itemsets Using Candidate Generation (利用候選項目集尋找頻繁項目集)

- Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets for Boolean association rules.
- The name of the algorithm is based on the fact that the algorithm uses *prior knowledge* of frequent itemset properties
- Concept:
  1. An iterative approach known as a level-wise search(逐層搜尋法)
  2. k-itemsets are used to explore (k+1)-itemsets.

- Method:
  1. Initially, scan DB once to get frequent 1-itemset (L_1) and satisfy minimum support.
  2. Generate length (k+1) candidate itemsets from length k frequent itemsets $\boxed{L_1 \rightarrow L_2 \rightarrow L_3 \rightarrow \ldots}$
  3. Test the candidates against DB
  4. Terminate when no frequent or candidate set can be generated

Apriori property

- *All nonempty subsets of a frequent itemset must also be frequent.*
  - If an itemset I does not satisfy the minimum support threshold, *min_sup*, then I is not frequent; that is, $P(I) <$ *min_sup*.
  - If an item A is added to the itemset I, then the resulting itemset (i.e., $I \cup A$) cannot occur more frequently than I.
  - $I \cup A$ is not frequent and $P(I \cup A) <$ *min sup*.
- Antimonotone (反一致性):
  If a set cannot pass a test, all of its supersets (超集合) will fail the same test as well.

Example 5.3

- The *AllElectronics* transaction database, *D*, of Table 5.1.
- There are nine transactions in this database, that is, |*D*| = 9.
- We use Figure 5.2 to illustrate the Apriori algorithm for finding frequent itemsets in *D*.

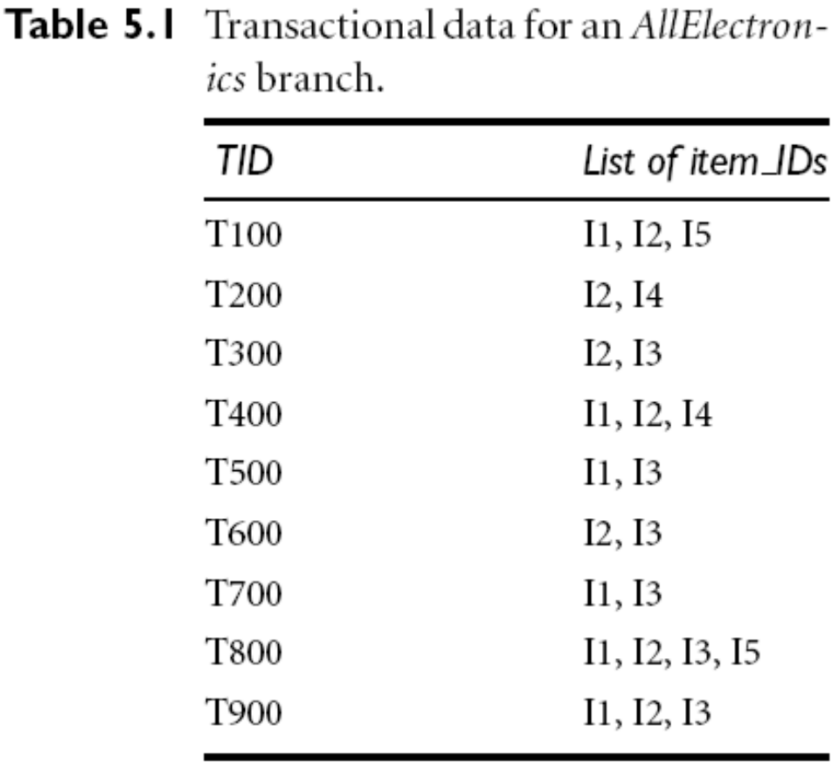
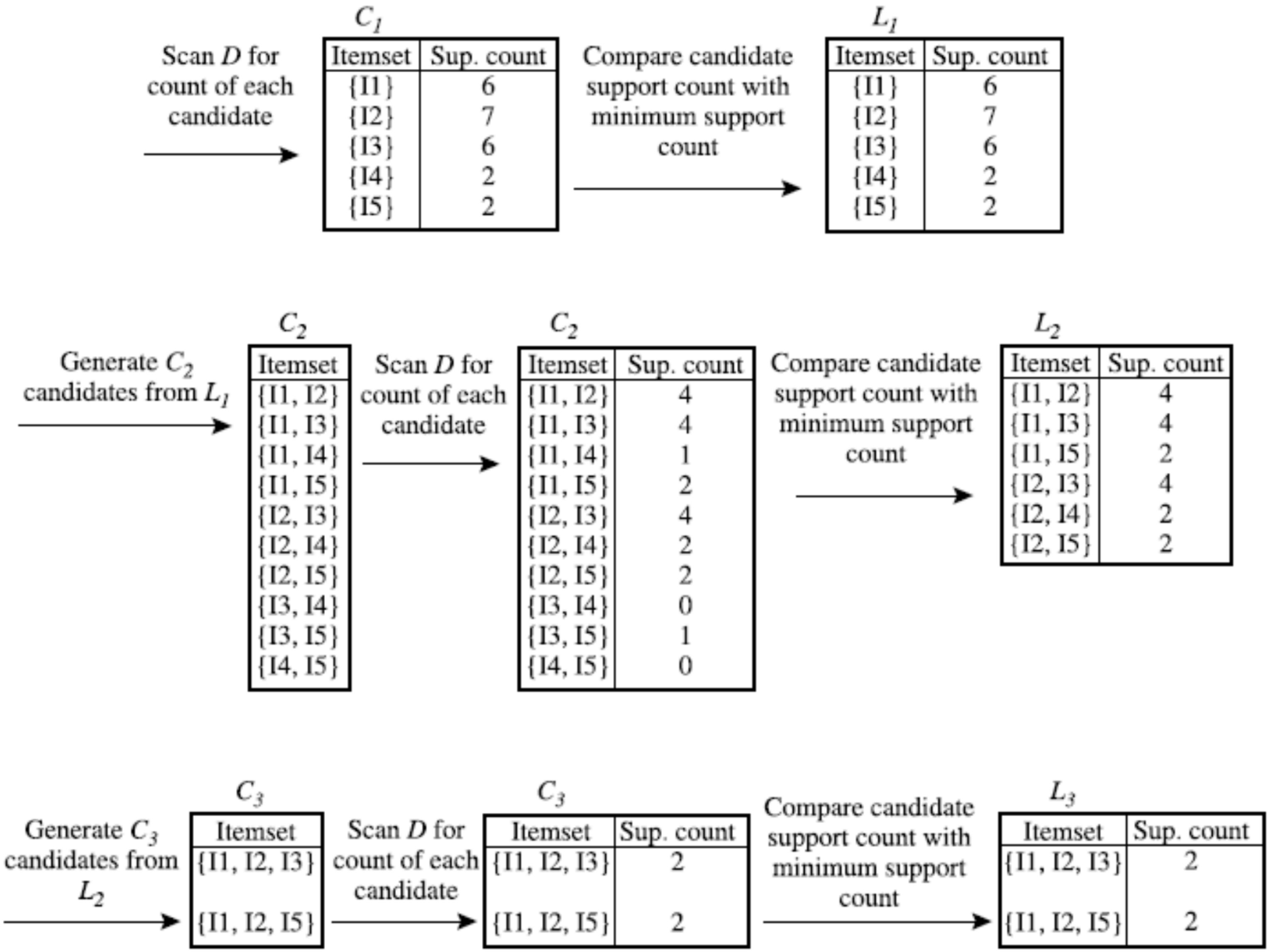


Table 5.1 Transactional data for an *AllElectronics* branch.

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



Scan *D* for count of each candidate →

C_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Compare candidate support count with minimum support count →

L_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Generate C_2 candidates from L_1 →

C_2

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

Scan *D* for count of each candidate →

C_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

Compare candidate support count with minimum support count →

L_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Generate C_3 candidates from L_2 →

C_3

Itemset
{I1, I2, I3}
{I1, I2, I5}

Scan *D* for count of each candidate →

C_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

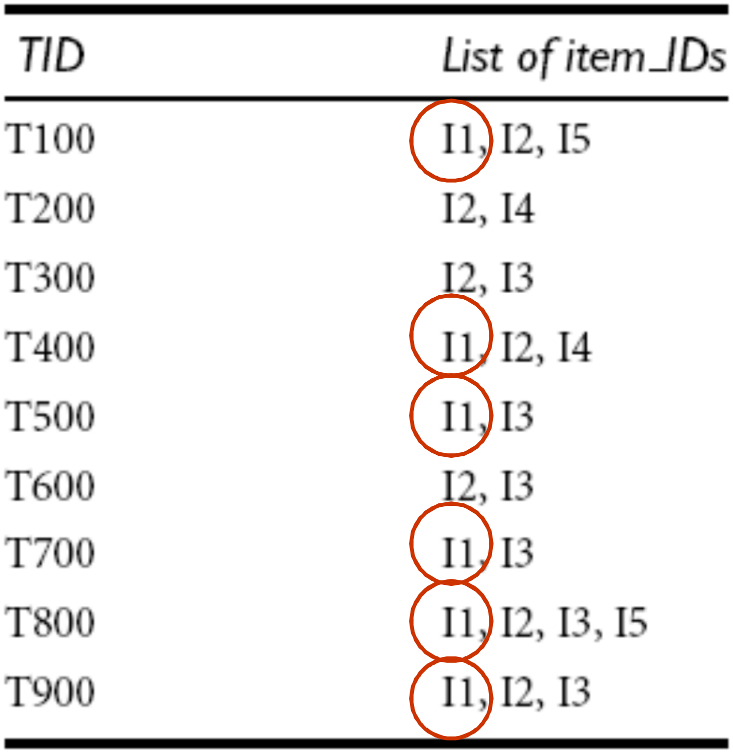
Compare candidate support count with minimum support count →

L_3

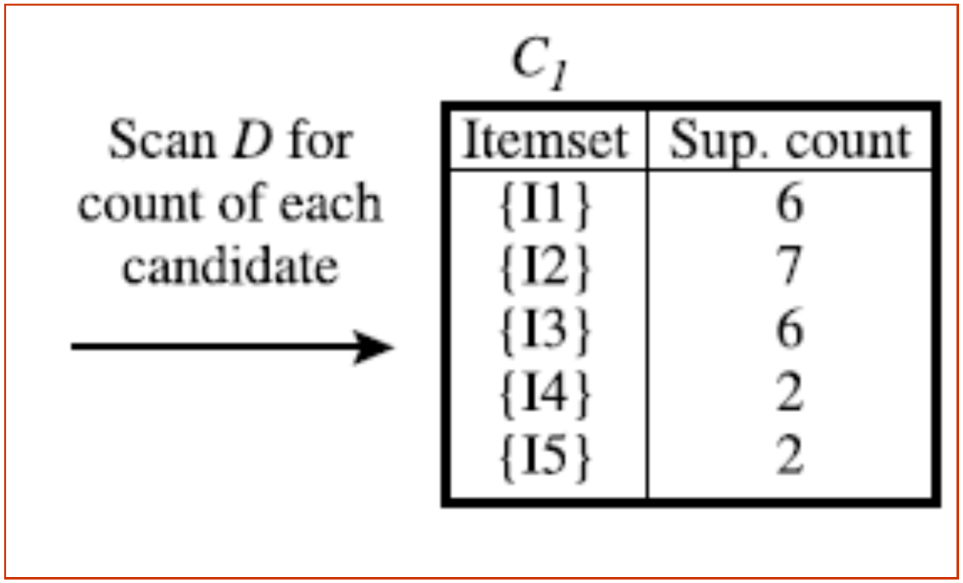
Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Figure 5.2 Generation of candidate itemsets and frequent itemsets, where the minimum support count is 2.

- [步驟1] In the first iteration of the algorithm, each item is a member of the set of candidate **1-itemsets**, C_1. The algorithm simply scans all of the transactions in order to count the number of occurrences of each item.



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



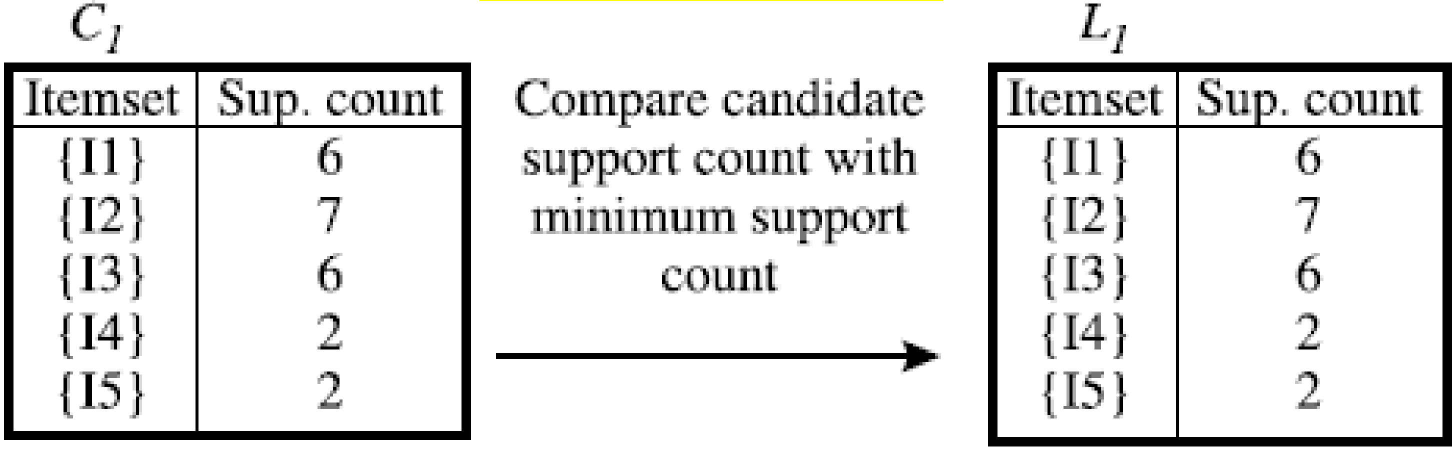
Scan D for count of each candidate

C_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

- [步驟2] Suppose that the minimum support count required is 2, that is, *min_sup* = 2. (Here, we are referring to absolute support because we are using a support count. The corresponding relative support is 2/9 = 22%). The set of frequent 1-itemsets, L_1, can then be determined. It consists of the candidate 1-itemsets satisfying minimum support.

利用 C_1 找出 L_1



C_1

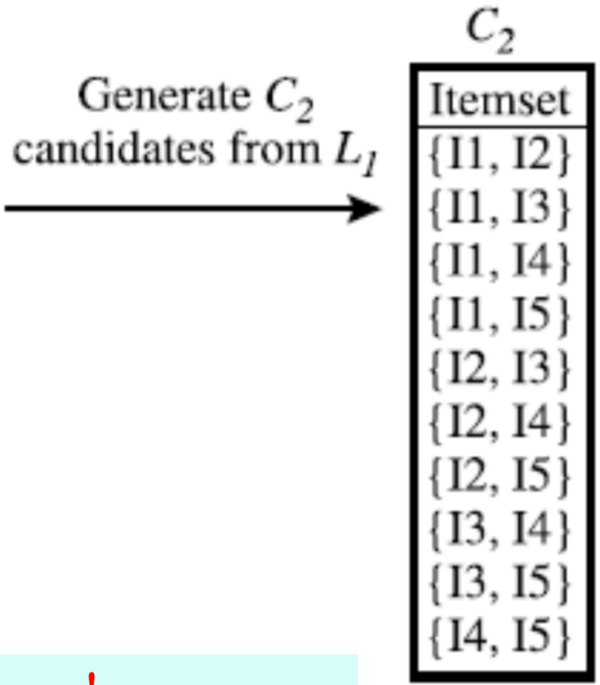
Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Compare candidate support count with minimum support count

L_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

- [步驟3] To discover the set of frequent 2-itemsets, L_2, the algorithm uses the join $L_1 \bowtie L_1$ to generate a candidate set of 2-itemsets, C2.
- C_2 consists of $\binom{|L_1|}{2}$ 2-itemsets.
- Note that no candidates are removed from C_2 during the prune step (修剪) because each subset of the candidates is also frequent.



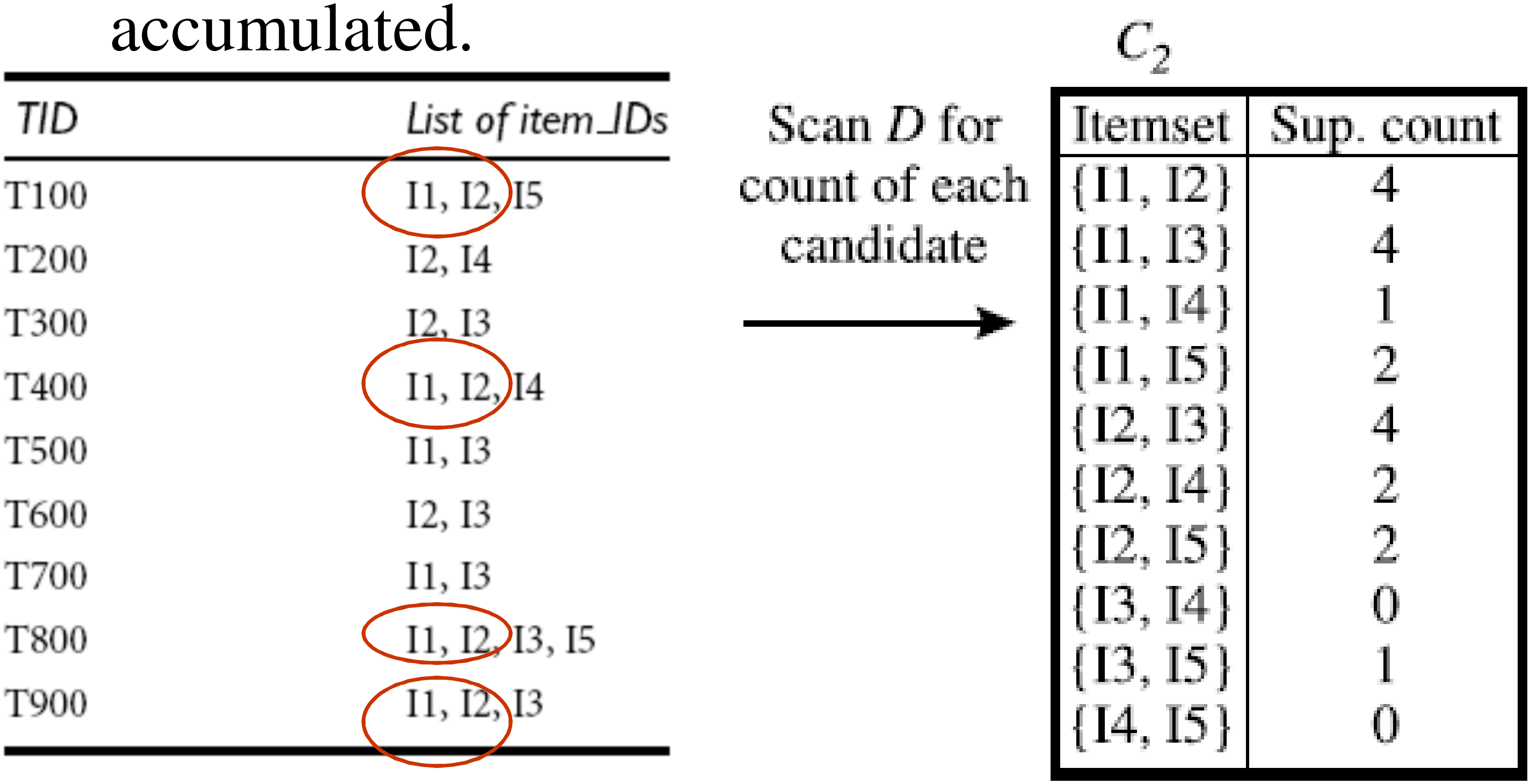
C_2

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

$$\text{排列}, Permutations: P_r^n = \frac{n!}{(n-r)!}, r \le n$$

$$\text{組合}, Combinations: C_r^n = \frac{n!}{r!(n-r)!}, r \le n$$

- [步驟4] Next, the transactions in *D* are scanned and the support count of each candidate itemset in C_2 is accumulated.



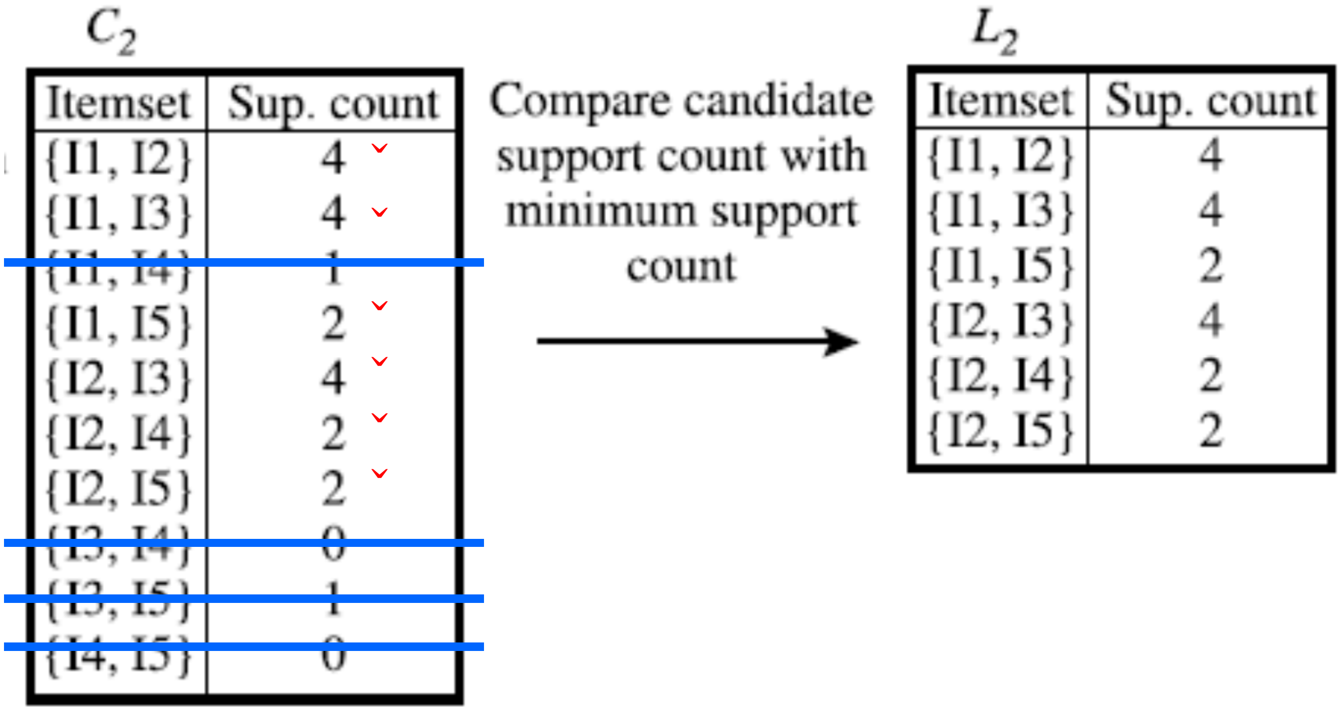
TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Scan *D* for count of each candidate

C_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

- [步驟5] The set of frequent 2-itemsets, L2, is then determined, consisting of those candidate 2-itemsets in C2 having minimum support. (找出 Sup.count ≧2)



C_2

Itemset	Sup. count
{I1, I2}	4 ✓
{I1, I3}	4 ✓
~~{I1, I4}~~	~~1~~
{I1, I5}	2 ✓
{I2, I3}	4 ✓
{I2, I4}	2 ✓
{I2, I5}	2 ✓
~~{I3, I4}~~	~~0~~
~~{I3, I5}~~	~~1~~
~~{I4, I5}~~	~~0~~

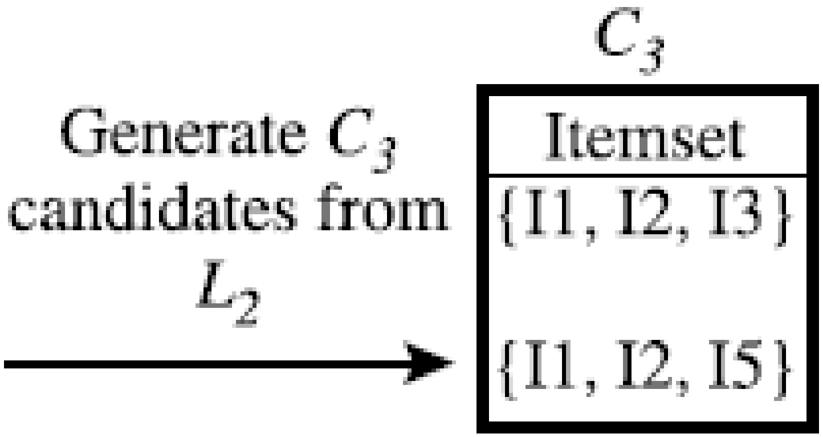
Compare candidate support count with minimum support count

L_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

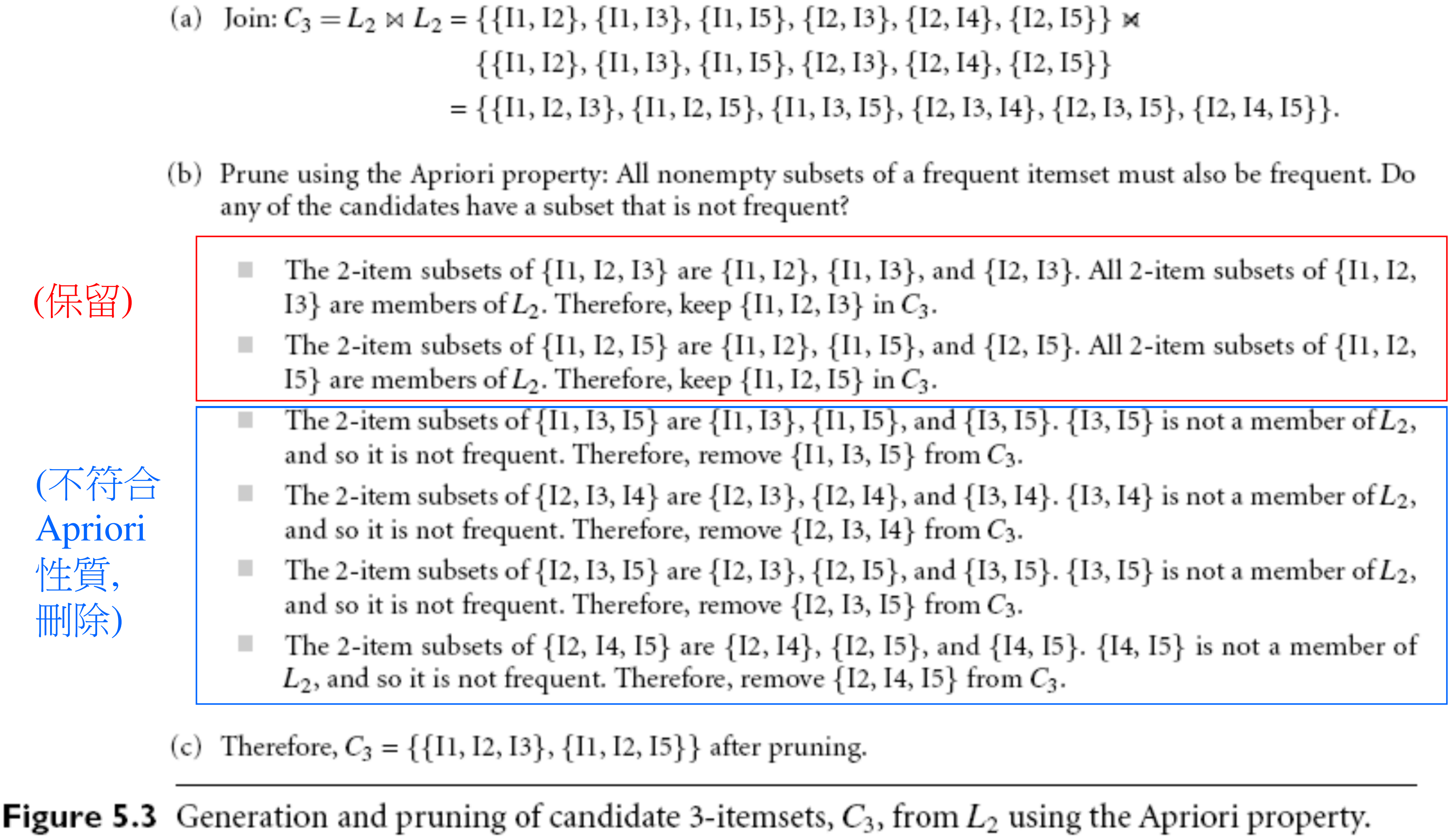
- [步驟6]

The generation of the set of candidate 3-itemsets, C_3, is detailed in Figure 5.3. From the join step, we first get $C_3 = L_2 \bowtie L_2 = \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}$. Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that the four latter candidates cannot possibly be frequent. We therefore remove them from C_3, thereby saving the effort of unnecessarily obtaining their counts during the subsequent scan of D to determine L_3.



C_3

Itemset
{I1, I2, I3}
{I1, I2, I5}



(a) Join: $C_3 = L_2 \bowtie L_2 = \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\} \bowtie$
$\{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
$= \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}$.

(b) Prune using the Apriori property: All nonempty subsets of a frequent itemset must also be frequent. Do any of the candidates have a subset that is not frequent?

(保留)

- The 2-item subsets of {I1, I2, I3} are {I1, I2}, {I1, I3}, and {I2, I3}. All 2-item subsets of {I1, I2, I3} are members of L_2. Therefore, keep {I1, I2, I3} in C_3.
- The 2-item subsets of {I1, I2, I5} are {I1, I2}, {I1, I5}, and {I2, I5}. All 2-item subsets of {I1, I2, I5} are members of L_2. Therefore, keep {I1, I2, I5} in C_3.

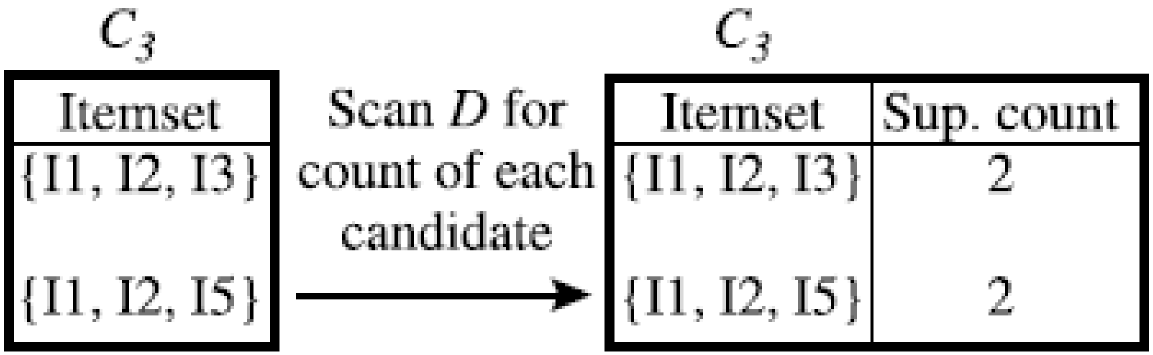
(不符合 Apriori 性質, 刪除)

- The 2-item subsets of {I1, I3, I5} are {I1, I3}, {I1, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I1, I3, I5} from C_3.
- The 2-item subsets of {I2, I3, I4} are {I2, I3}, {I2, I4}, and {I3, I4}. {I3, I4} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I4} from C_3.
- The 2-item subsets of {I2, I3, I5} are {I2, I3}, {I2, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I5} from C_3.
- The 2-item subsets of {I2, I4, I5} are {I2, I4}, {I2, I5}, and {I4, I5}. {I4, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I4, I5} from C_3.

(c) Therefore, $C_3 = \{\{I1, I2, I3\}, \{I1, I2, I5\}\}$ after pruning.

Figure 5.3 Generation and pruning of candidate 3-itemsets, C_3, from L_2 using the Apriori property.

- [步驟7] The transactions in D are scanned in order to determine L_3, consisting of those candidate 3-itemsets in C_3 having minimum support.



C_3

Itemset
{I1, I2, I3}
{I1, I2, I5}

Scan *D* for count of each candidate

C_3

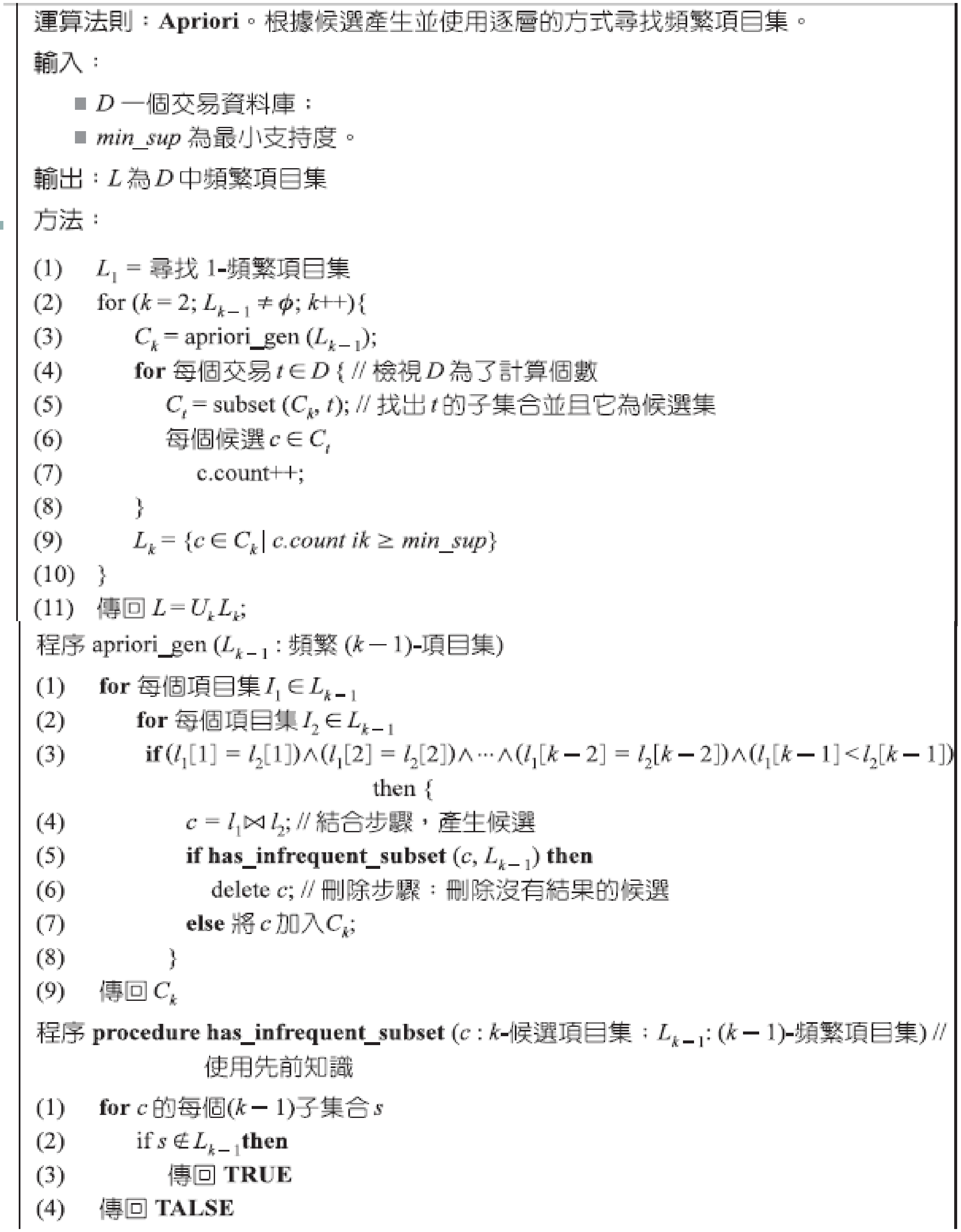
Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

- [步驟8]
  The algorithm uses $L_3 \bowtie L_3$ to generate a candidate set of 4-itemsets, C_4.
- Although the join results in {{I_1, I_2, I_3, I_5}}, this itemset is pruned because its subset {{I2, I3, I5}} **is not frequent.**
- Thus, $C_4 = \phi$, and the algorithm terminates, having found all of the frequent itemsets.

p.239 Fig. 5-4 Apriori Algorithm - Pseudo-code

p.239 Apriori Algorithm

Fig. 5-4



運算法則：**Apriori**。根據候選產生並使用逐層的方式尋找頻繁項目集。

輸入：

- *D* 一個交易資料庫；
- *min_sup* 為最小支持度。

輸出：*L* 為 *D* 中頻繁項目集

方法：

主程式

```
$L_1$ = 尋找 1-頻繁項目集
for ($k = 2; L_{k-1} \neq \phi; k$++){
    $C_k$ = apriori_gen ($L_{k-1}$);
    for 每個交易 $t \in D$ { // 檢視 $D$ 為了計算個數
        $C_t$ = subset ($C_k, t$); // 找出 $t$ 的子集合並且它為候選集
        每個候選 $c \in C_t$
            c.count++;
    }
    $L_k = \{c \in C_k \mid c.count\ ik \geq min\_sup\}$
}
傳回 $L = U_k L_k$;
```

•產生候選項目集

•JOIN 結合

•PRUNE 修剪

程序 apriori_gen (L_{k-1} : 頻繁 ($k-1$)-項目集)

```
for 每個項目集 $I_1 \in L_{k-1}$
    for 每個項目集 $I_2 \in L_{k-1}$
     if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \cdots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$
                            then {
          $c = l_1 \bowtie l_2$; // 結合步驟，產生候選
          if has_infrequent_subset ($c, L_{k-1}$) then
            delete $c$; // 刪除步驟：刪除沒有結果的候選
          else 將 $c$ 加入 $C_k$;
        }
傳回 $C_k$
```

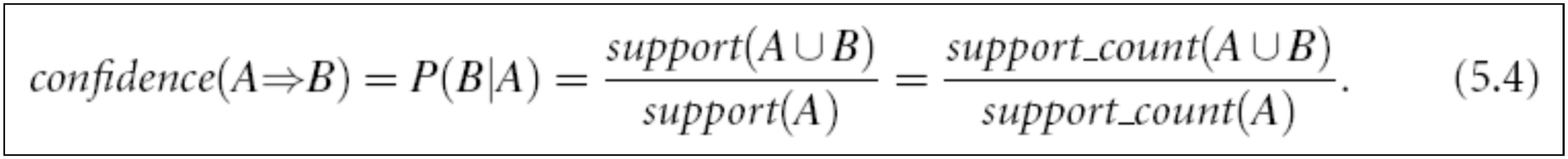
•利用 prior knowledge

程序 **procedure has_infrequent_subset** (c : k-候選項目集；L_{k-1}: ($k-1$)-頻繁項目集) // 使用先前知識

```
for $c$ 的每個($k-1$)子集合 $s$
    if $s \notin L_{k-1}$ then
        傳回 TRUE
傳回 TALSE
```

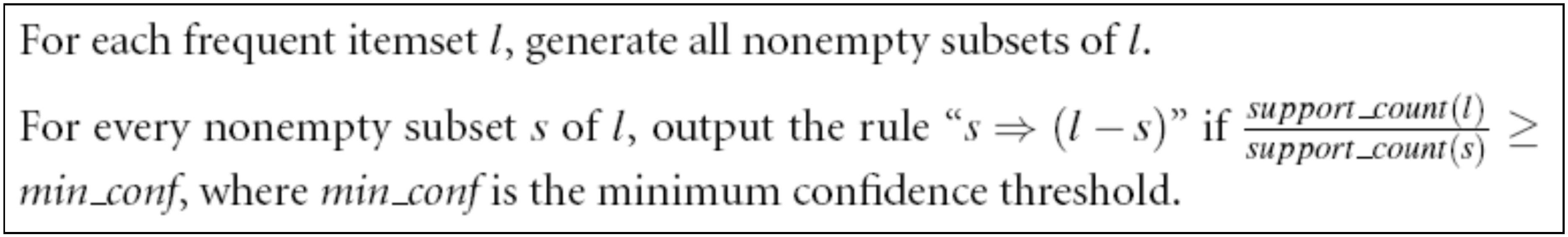
5.2.2 Generating Association Rules from Frequent Itemsets
從頻繁項目集產生關聯規則

- Strong association rules 必須滿足二個條件:
  - minimum support
  - minimum confidence).
- This can be done using Equation (5.4) for confidence



$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)} = \frac{support\_count(A \cup B)}{support\_count(A)}. \qquad (5.4)$$

  - support *count(A ∪B)* is the number of transactions containing the itemsets A∪B
  - support *count(A)* is the number of transactions containing the itemset *A*.

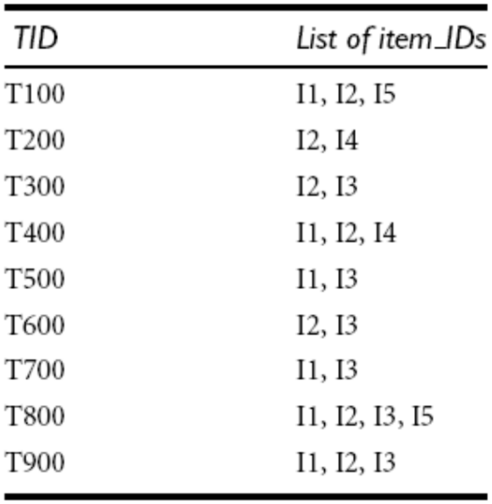


For each frequent itemset l, generate all nonempty subsets of l.

For every nonempty subset s of l, output the rule "$s \Rightarrow (l - s)$" if $\frac{support\_count(l)}{support\_count(s)} \geq$ min_conf, where min_conf is the minimum confidence threshold.

Example 5.4 Generating association rules.

- Suppose the data contain the frequent itemset *l* = {I1, I2, I5}. What are the association rules that can be generated from *l*?
- Nonempty subsets of *l* are {I1, I2}, {I1, I5}, {I2, I5}, {I1},{I2}, and {I5}.



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

$$I1 \wedge I2 \Rightarrow I5, \qquad confidence = 2/4 = 50\%$$

$$\boxed{\begin{aligned} &I1 \wedge I5 \Rightarrow I2, \qquad confidence = 2/2 = 100\% \\ &I2 \wedge I5 \Rightarrow I1, \qquad confidence = 2/2 = 100\% \end{aligned}}$$

$$I1 \Rightarrow I2 \wedge I5, \qquad confidence = 2/6 = 33\%$$

$$I2 \Rightarrow I1 \wedge I5, \qquad confidence = 2/7 = 29\%$$

$$\boxed{I5 \Rightarrow I1 \wedge I2, \qquad confidence = 2/2 = 100\%}$$

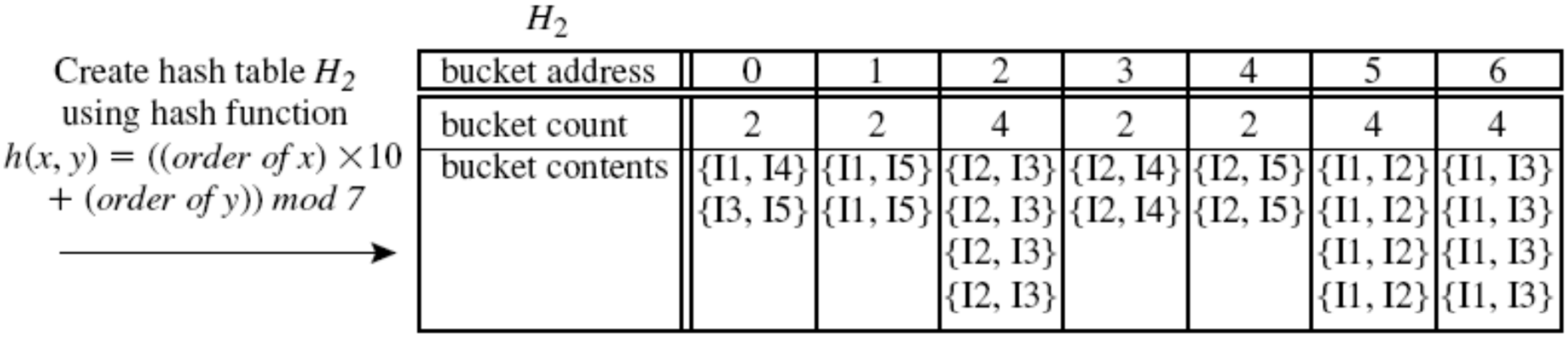
- If the minimum confidence threshold is, say, 70%, then only the second, third, and last rules above are output, because these are the only ones generated that are strong.
- Note that, unlike conventional classification rules, association rules can contain more than one conjunct in the right-hand side of the rule.

5.2.3 Improving the Efficiency of Apriori

- **Hash-based technique**:
  To reduce the size of the candidate *k*-itemsets, C_k, *k*>1.

$$I=\{I_1,I_2,I_3,I_4,I_5\}, i.e., 計算\{I_2,I_4\}:(\underset{\overset{\uparrow}{x}}{2}\times 10+\underset{\overset{\uparrow}{y}}{4}) \bmod 7=3$$

$$order=\{1,2,3,4,5\}$$



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 5.5 Hash table, H_2, for candidate 2-itemsets: This hash table was generated by scanning the transactions of Table 5.1 while determining L_1 from C_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.

5.2.3 Improving the Efficiency of Apriori (cont.)

2. **Transaction reduction**:
   reducing the number of transactions scanned in future iterations
3. **Partitioning:**
   partitioning the data to find candidate itemsets
4. **Sampling:**
   mining on a subset of the given data
5. **Dynamic itemset counting:**
   adding candidate itemsets at different points during a scan

5.2.4 Mining Frequent Itemsets without Candidate Generation

- the Apriori candidate generate-and-test method significantly reduces the size of candidate sets, leading to good performance gain.
- 問題:二種非明顯成本 or 隱藏成本(nontrivial costs):
  - A huge number of candidate sets.
  - It may need to repeatedly scan the database and check a large set of candidates by pattern matching.
- 方法:
  - Frequent-Pattern growth (頻繁樣式成長) or 簡稱 FP-growth (FP成長)

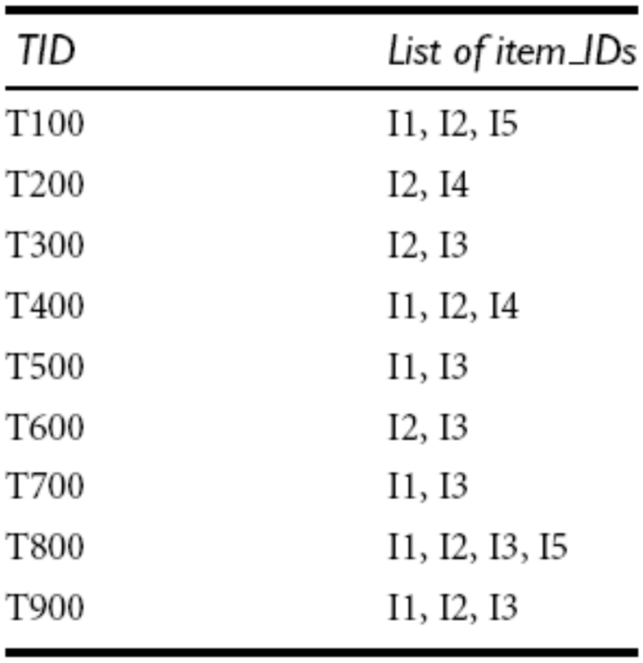
Frequent-Pattern growth

- Divide-and-conquer strategy 分割與克服的策略
  - [步驟1]:
    Compress the database representing frequent items into a frequent-pattern tree (FP-tree) 頻繁樣式樹 and retains the itemset association information.
  - [步驟2]:
    Divides the compressed database into a set of conditional databases (a special kind of projected database), each associated with one frequent item or "pattern fragment," and mines each such database separately.

p.243

Example 5.5 FP-growth (finding frequent itemsets without candidate generation)

- [步驟1]: Scan of the database and derives the set of frequent items (1-itemsets) and their support counts frequencies
  - Let the minimum support count be 2.
  - The set of frequent items is sorted in the order of descending (遞減排序) support count. This resulting set or list is denoted *L*. *L* ={{I2: **7**}, {I1: **6**}, {I3: **6**}, {I4: **2**}, {I5: **2**}}.

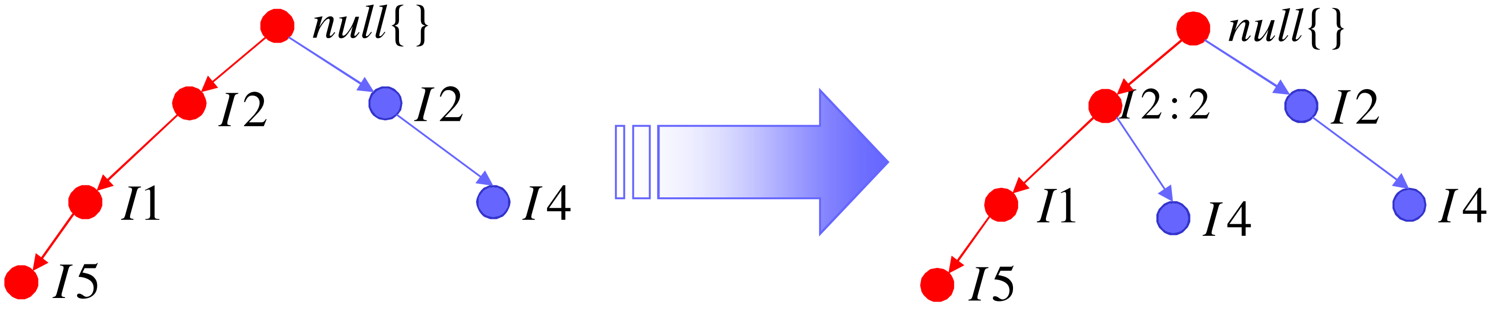


TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Example 5.5 FP-growth (cont.)

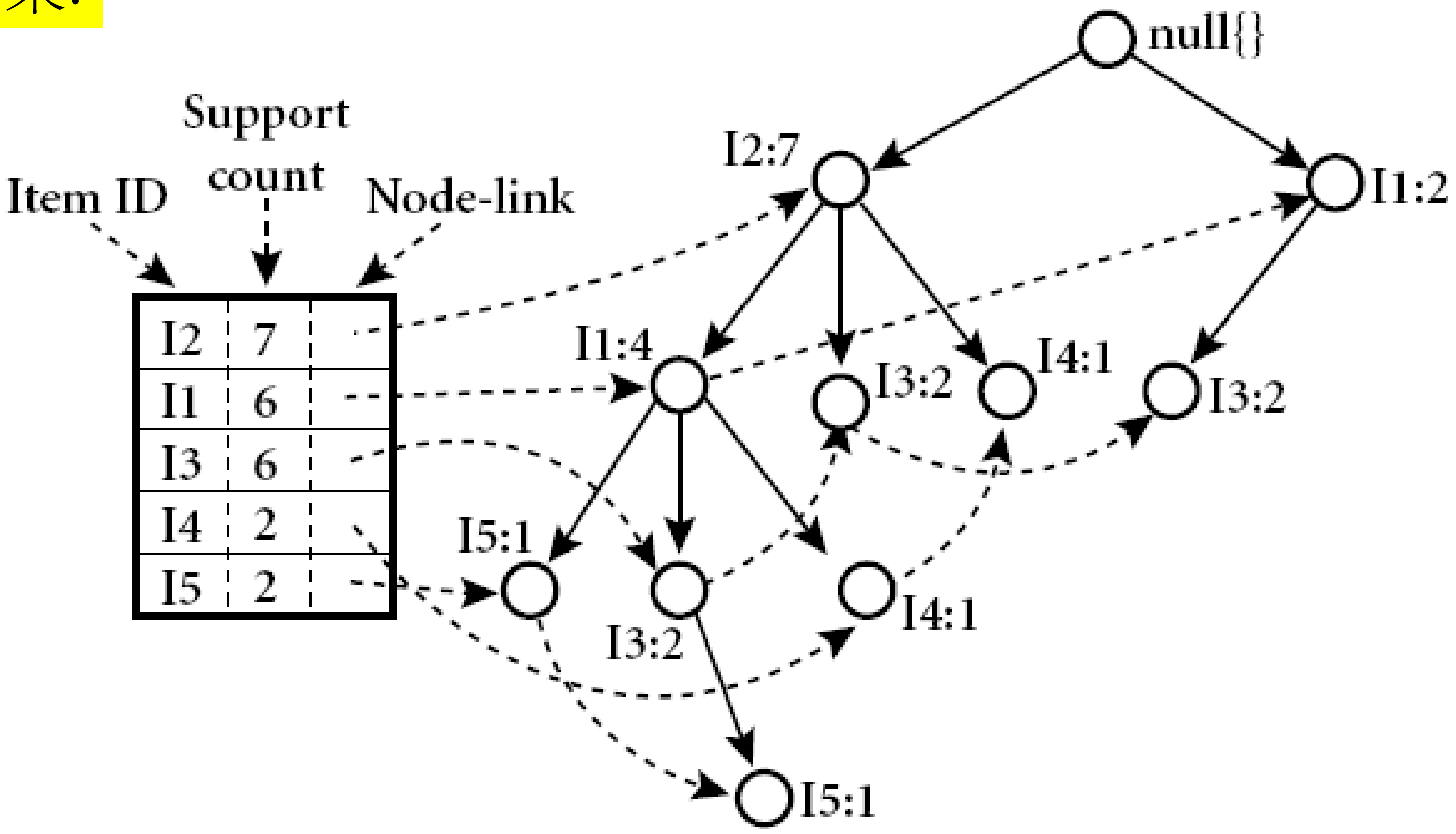
- [步驟2]: 建立 FP-trees
  1. First, create the root of the tree, labeled with “null.”
  2. Scan database D a second time. The items in each transaction are processed in *L* order (i.e., sorted according to descending support count), and a branch is created for each transaction.

L ={{I2: 7}, {I1: 6}, {I3: 6}, {I4: 2}, {I5: 2}}



- 第1筆記錄, “T100: I1, I2, I5” which contains three items (I2, I1, I5 in L order), construct of the first branch of the tree with three nodes, <I2: 1>, <I1:1>, and <I5: 1>, where I2 is linked as a child of the root, I1 is linked to I2, and I5 is linked to I1.
- 第2筆記錄, “T200: I2, I4”, which would result in a branch where I2 is linked to the root and I4 is linked to I2. However, this branch would share a common prefix (前置), I2, with the existing path for T100. Therefore, we instead increment the count of the I2 node by 1, and create a new node, <I4: 1>,which is linked as a child of <I2: 2>.
- In general, when considering the branch to be added for a transaction, the count of each node along a common prefix is incremented by 1, and nodes for the items following the prefix are created and linked accordingly.

步驟2之結果:



Item ID	Support count	Node-link
I2	7	
I1	6	
I3	6	
I4	2	
I5	2	

Figure 5.7 An FP-tree registers compressed, frequent pattern information.

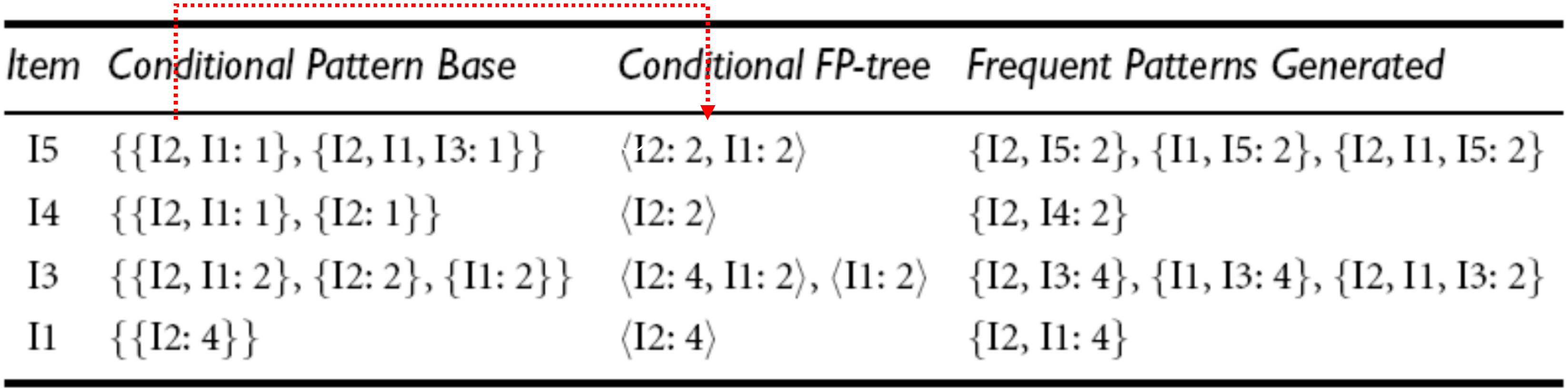
- [步驟3]: Mining FP-trees
  - Start from each frequent length-1 pattern (as an initial suffix pattern, 啟始的後置樣式)
  - Construct its conditional pattern base 條件基礎樣式 - consists of **the set of prefix paths(前置路徑) in the FP-tree co-occurring with the suffix pattern(後置樣式),** 包含後置樣式的前置路徑
  - Construct conditional FP-tree 條件FP樹, and perform mining recursively on such a tree.
  - The pattern growth is achieved by the concatenation(連鎖) of the suffix pattern with the frequent patterns generated from a conditional FP-tree.

p.243

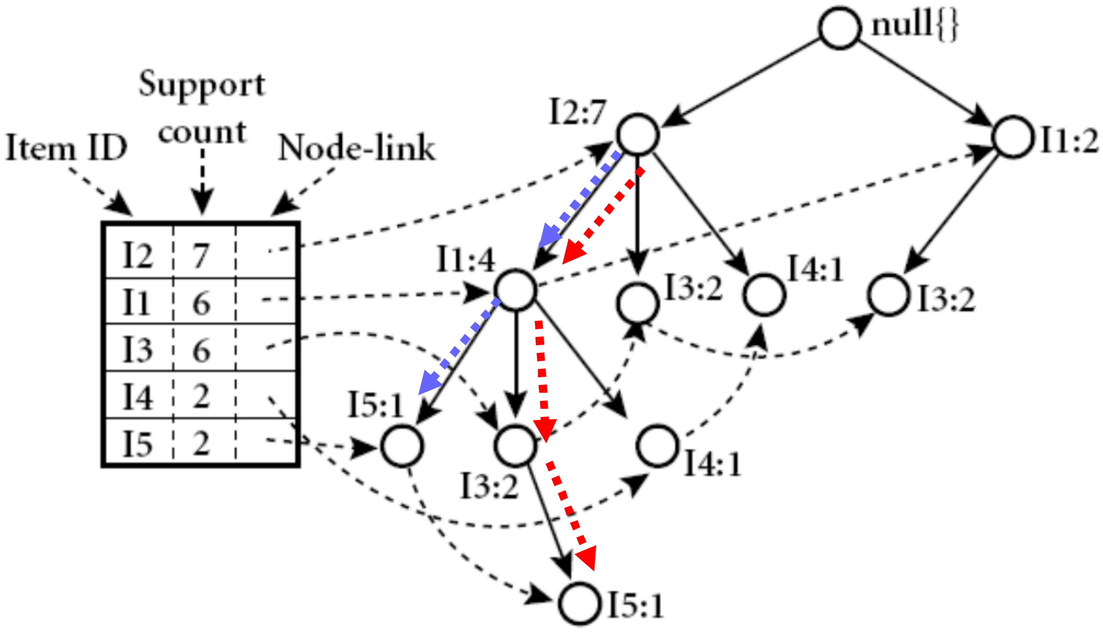
步驟3之結果:

L ={{I2: 7}, {I1: 6}, {I3: 6}, {I4: 2}, {I5: 2}}

Table 5.2, 從 L 之最後一個, I5 開始



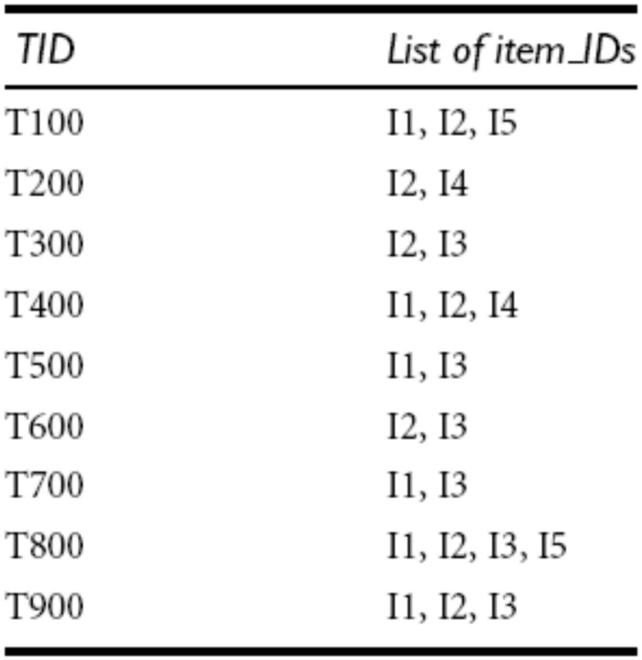
Item	*Conditional Pattern Base*	*Conditional FP-tree*	*Frequent Patterns Generated*
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}



I2	7	
I1	6	
I3	6	
I4	2	
I5	2	

5.2.5 Mining Frequent Itemsets Using Vertical Data Format

- 如果資料格式記錄項目所在交易 ({項目：交易編號})，則此種資料格式被稱為垂直資料格式



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

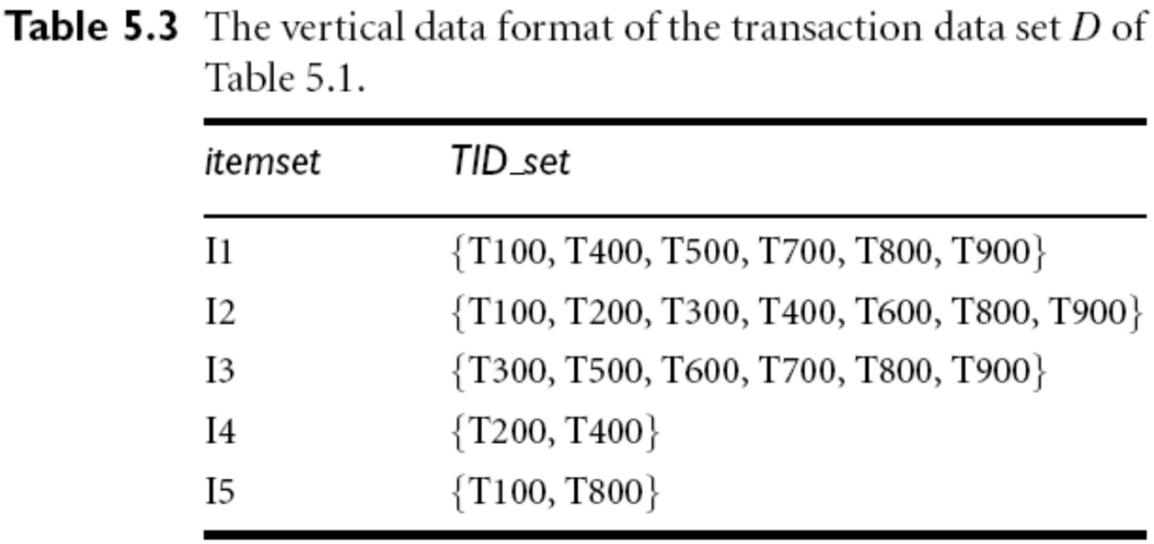
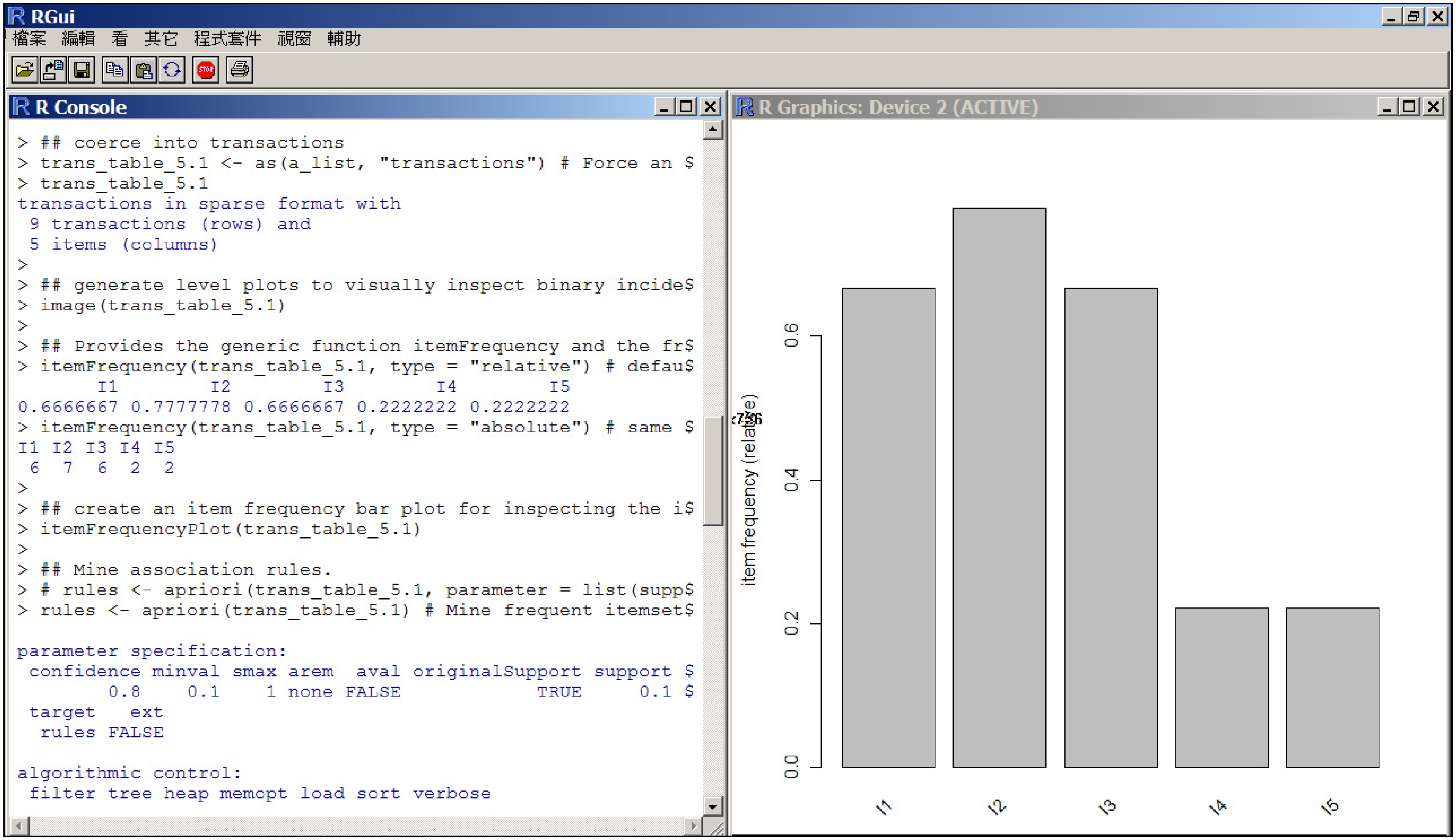


Table 5.3 The vertical data format of the transaction data set *D* of Table 5.1.

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}

R. **arules** package


RGui
檔案 編輯 看 其它 程式套件 視窗 輔助
R Console
R Graphics: Device 2 (ACTIVE)
item frequency (relative)
0.0
0.2
0.4
0.6
I1
I2
I3
I4
I5

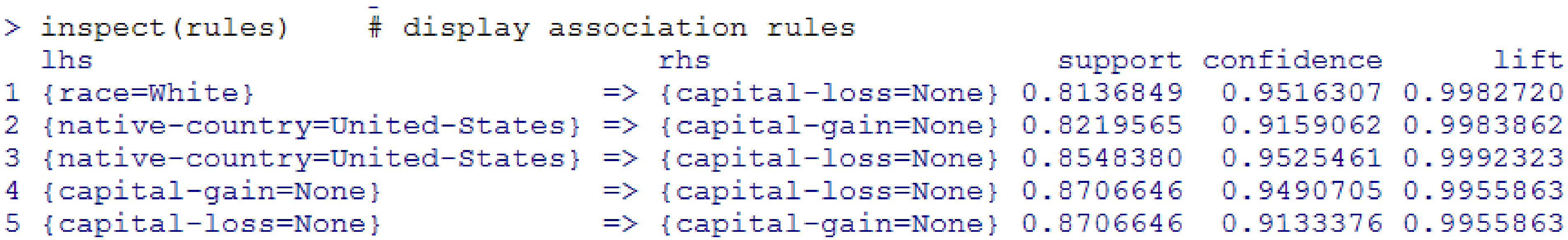
```
> ## coerce into transactions
> trans_table_5.1 <- as(a_list, "transactions") # Force an $
> trans_table_5.1
transactions in sparse format with
 9 transactions (rows) and
 5 items (columns)
>
> ## generate level plots to visually inspect binary incide$
> image(trans_table_5.1)
>
> ## Provides the generic function itemFrequency and the fr$
> itemFrequency(trans_table_5.1, type = "relative") # defau$
       I1        I2        I3        I4        I5
0.6666667 0.7777778 0.6666667 0.2222222 0.2222222
> itemFrequency(trans_table_5.1, type = "absolute") # same $
I1 I2 I3 I4 I5
 6  7  6  2  2
>
> ## create an item frequency bar plot for inspecting the i$
> itemFrequencyPlot(trans_table_5.1)
>
> ## Mine association rules.
> # rules <- apriori(trans_table_5.1, parameter = list(supp$
> rules <- apriori(trans_table_5.1) # Mine frequent itemset$

parameter specification:
 confidence minval smax arem  aval originalSupport support $
        0.8    0.1    1 none FALSE            TRUE     0.1 $
 target   ext
  rules FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
```

Example - Association rules

```
# R codes
# Example – Association rules
library(arules)
data("Adult")
# Mine association rules.
rules <- apriori(Adult, parameter = list(supp = 0.8,
   conf = 0.9, target = "rules", minlen=2))
inspect(Adult)     # display transactions
inspect(rules)     # display association rules
# end
```



```
> inspect(rules)    # display association rules
  lhs                                rhs                    support confidence      lift
1 {race=White}                    => {capital-loss=None} 0.8136849  0.9516307 0.9982720
2 {native-country=United-States} => {capital-gain=None} 0.8219565  0.9159062 0.9983862
3 {native-country=United-States} => {capital-loss=None} 0.8548380  0.9525461 0.9992323
4 {capital-gain=None}             => {capital-loss=None} 0.8706646  0.9490705 0.9955863
5 {capital-loss=None}             => {capital-gain=None} 0.8706646  0.9133376 0.9955863
```

1. Basic concepts and a road map
2. Efficient and scalable frequent itemset mining methods
3. **Mining various kinds of association rules**
4. From association mining to correlation analysis
5. Constraint-based association mining
6. Summary

5.3 Mining Various Kinds of Association Rules

- Mining multilevel association rules 多層次關聯規則
- Miming multidimensional association rules 多維度關聯規則
  - Single dimensional / Intradimensional association rule
  - Multidimensional association rules
  - Mining quantitative association rules 數量關聯規則

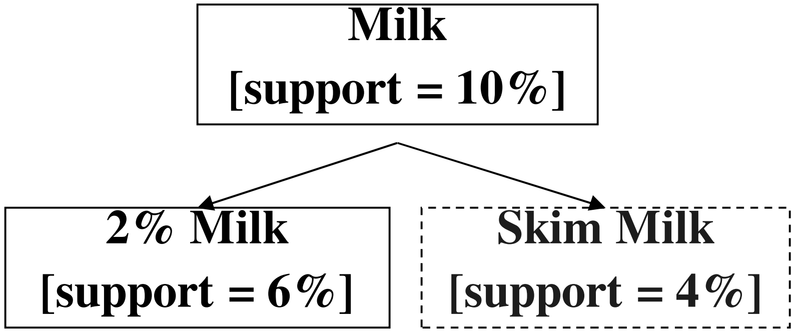
Mining Multiple-Level Association Rules

- Items often form hierarchies
- Flexible support settings
  - Items at the lower level are expected to have lower support
- Exploration of *shared* multi-level mining (Agrawal & Srikant@VLB'95, Han & Fu@VLDB'95)

uniform support

reduced support

Level 1
min_sup = 5%



Milk
[support = 10%]

2% Milk
[support = 6%]

Skim Milk
[support = 4%]

Level 1
min_sup = 5%

Level 2
min_sup = 5%

Level 2
min_sup = 3%

Multi-level Association: Redundancy Filtering

- Some rules may be redundant due to “ancestor” relationships between items.
- Example
  - milk ⇒ wheat bread [support = 8%, confidence = 70%]
  - 2% milk ⇒ wheat bread [support = 2%, confidence = 72%]
- We say 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.

Mining Multi-Dimensional Association

- Single-dimensional rules:

  buys(X, “milk”) ⇒ buys(X, “bread”)

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

    age(X,”19-25”) ∧ occupation(X,“student”) ⇒ buys(X, “coke”)
  - hybrid-dimension assoc. rules (*repeated predicates*)

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

Mining Quantitative Associations

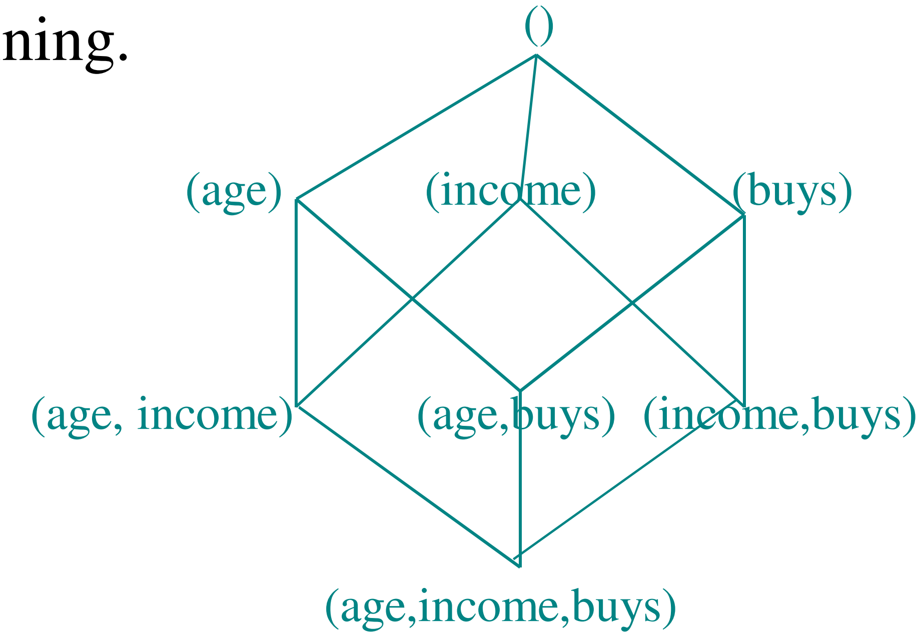
- Techniques can be categorized by how numerical attributes, such as age or salary are treated

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)

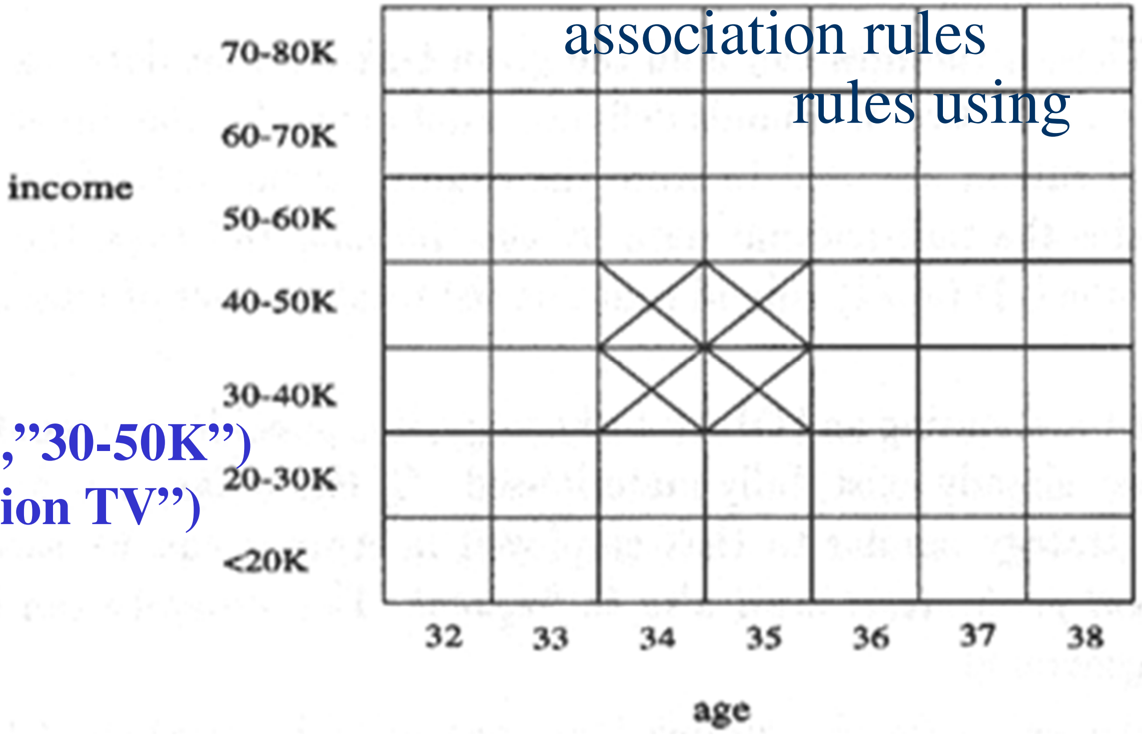
Static Discretization of Quantitative Attributes

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

- Proposed by Lent, Swami and Widom ICDE'97
- Numeric attributes are *dynamically* discretized
  - Such that the confidence or compactness of the rules mined is maximized
- 2-D quantitative association rules: $A_{quan1} \wedge A_{quan2} \Rightarrow A_{cat}$
- Cluster *adjacent* association rules to form general rules using a 2-D grid
- Example



age(X,"34-35") $\wedge$ income(X,"30-50K")
$\Rightarrow$ buys(X,"high resolution TV")

Mining Other Interesting Patterns

- Flexible support constraints (Wang et al. @ VLDB'02)
  - Some items (e.g., diamond) may occur rarely but are valuable
  - Customized sup_{min} specification and application
- Top-K closed frequent patterns (Han, et al. @ ICDM'02)
  - Hard to specify sup_{min}, but top-k with $length_{min}$ is more desirable
  - Dynamically raise sup_{min} in FP-tree construction and mining, and select most promising path to mine

Chapter 5: Mining Frequent Patterns, Association and Correlations

1. Basic concepts and a road map
2. Efficient and scalable frequent itemset mining methods
3. Mining various kinds of association rules
4. **From association mining to correlation analysis**
5. Constraint-based association mining
6. Summary

Interestingness Measure: Correlations (Lift)

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

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

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

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

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

Are *lift* and χ^2 Good Measures of Correlation?

- “*Buy walnuts* ⇒ *buy milk* [1%, 80%]” is misleading
  - if 85% of customers buy milk
- Support and confidence are not good to represent correlations
- So many interestingness measures? (Tan, Kumar, Sritastava @KDD’02)

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

$$all\_conf = \frac{\mathrm{sup}(X)}{\mathrm{max}\_item\_\mathrm{sup}(X)}$$

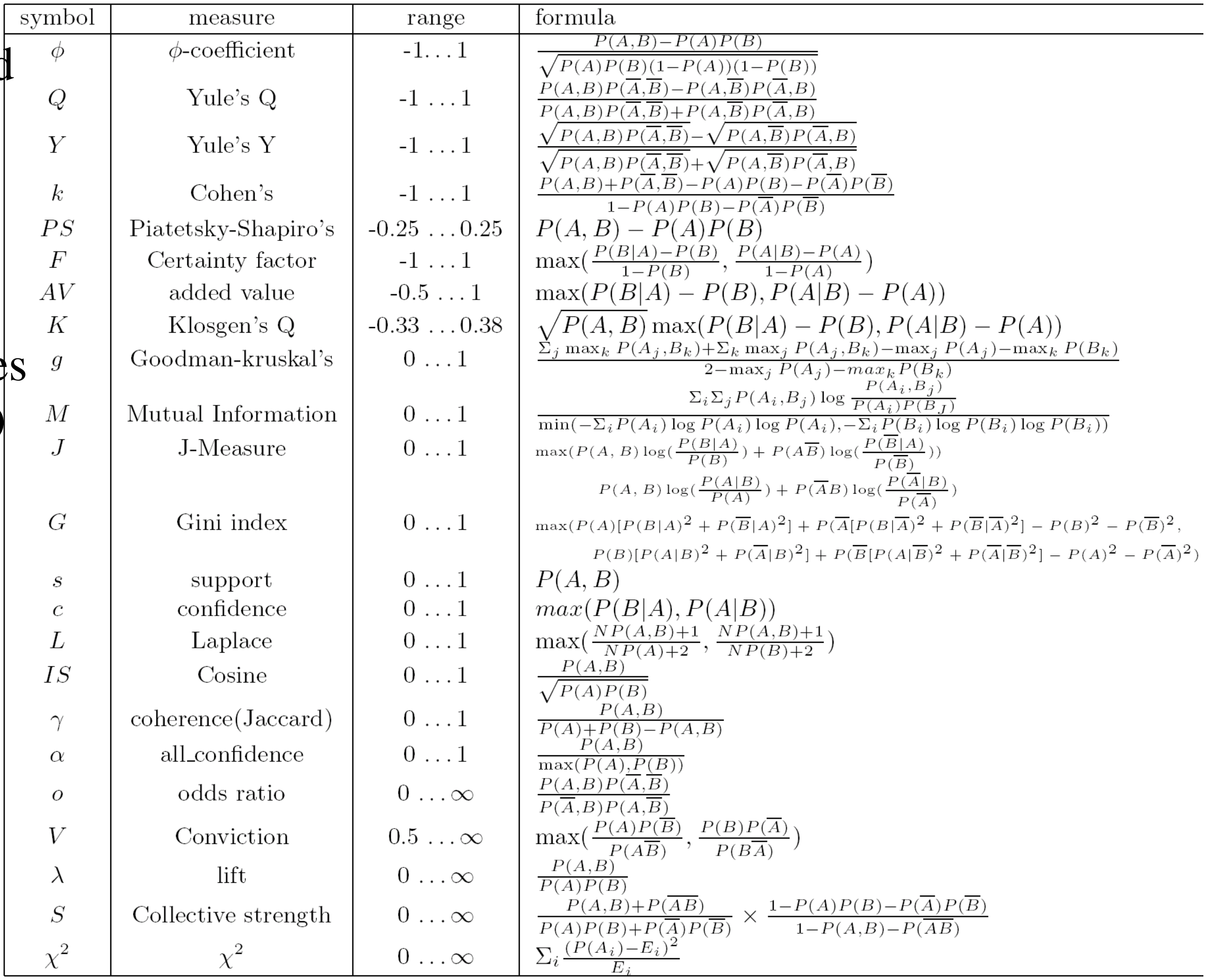
$$coh = \frac{\mathrm{sup}(X)}{|universe(X)|}$$

	Milk	No Milk	Sum (row)
Coffee	m, c	~m, c	c
No Coffee	m, ~c	~m, ~c	~c
Sum(col.)	m	~m	Σ

DB	m, c	~m, c	m~c	~m~c	lift	all-conf	coh	χ2
A1	1000	100	100	10,000	9.26	0.91	0.83	9055
A2	100	1000	1000	100,000	8.44	0.09	0.05	670
A3	1000	100	10000	100,000	9.18	0.09	0.09	8172
A4	1000	1000	1000	1000	1	0.5	0.33	0

Which Measures Should Be Used?

- **lift** *and* **χ^2** are not good measures for correlations in large transactional DBs
- **all-conf** or **coherence** could be good measures (Omiecinski@TKDE'03)
- Both **all-conf** and **coherence** have the downward closure property
- Efficient algorithms can be derived for mining (Lee et al. @ICDM'03sub)



symbol	measure	range	formula
ϕ	ϕ-coefficient	-1 . . . 1	$\frac{P(A,B)-P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
Q	Yule's Q	-1 . . . 1	$\frac{P(A,B)P(\overline{A},\overline{B})-P(A,\overline{B})P(\overline{A},B)}{P(A,B)P(\overline{A},\overline{B})+P(A,\overline{B})P(\overline{A},B)}$
Y	Yule's Y	-1 . . . 1	$\frac{\sqrt{P(A,B)P(\overline{A},\overline{B})}-\sqrt{P(A,\overline{B})P(\overline{A},B)}}{\sqrt{P(A,B)P(\overline{A},\overline{B})}+\sqrt{P(A,\overline{B})P(\overline{A},B)}}$
k	Cohen's	-1 . . . 1	$\frac{P(A,B)+P(\overline{A},\overline{B})-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A)P(B)-P(\overline{A})P(\overline{B})}$
PS	Piatetsky-Shapiro's	-0.25 . . . 0.25	$P(A,B)-P(A)P(B)$
F	Certainty factor	-1 . . . 1	$\max(\frac{P(B\mid A)-P(B)}{1-P(B)}, \frac{P(A\mid B)-P(A)}{1-P(A)})$
AV	added value	-0.5 . . . 1	$\max(P(B\mid A)-P(B), P(A\mid B)-P(A))$
K	Klosgen's Q	-0.33 . . . 0.38	$\sqrt{P(A,B)}\max(P(B\mid A)-P(B), P(A\mid B)-P(A))$
g	Goodman-kruskal's	0 . . . 1	$\frac{\Sigma_j \max_k P(A_j,B_k)+\Sigma_k \max_j P(A_j,B_k)-\max_j P(A_j)-\max_k P(B_k)}{2-\max_j P(A_j)-max_k P(B_k)}$
M	Mutual Information	0 . . . 1	$\frac{\Sigma_i \Sigma_j P(A_i,B_j)\log\frac{P(A_i,B_j)}{P(A_i)P(B_J)}}{\min(-\Sigma_i P(A_i)\log P(A_i)\log P(A_i), -\Sigma_i P(B_i)\log P(B_i)\log P(B_i))}$
J	J-Measure	0 . . . 1	$\max(P(A,B)\log(\frac{P(B\mid A)}{P(B)})+P(A\overline{B})\log(\frac{P(\overline{B}\mid A)}{P(\overline{B})}))$ $P(A,B)\log(\frac{P(A\mid B)}{P(A)})+P(\overline{A}B)\log(\frac{P(\overline{A}\mid B)}{P(\overline{A})})$
G	Gini index	0 . . . 1	$\max(P(A)[P(B\mid A)^2+P(\overline{B}\mid A)^2]+P(\overline{A}[P(B\mid\overline{A})^2+P(\overline{B}\mid\overline{A})^2]-P(B)^2-P(\overline{B})^2,$ $P(B)[P(A\mid B)^2+P(\overline{A}\mid B)^2]+P(\overline{B}[P(A\mid\overline{B})^2+P(\overline{A}\mid\overline{B})^2]-P(A)^2-P(\overline{A})^2)$
s	support	0 . . . 1	$P(A,B)$
c	confidence	0 . . . 1	$max(P(B\mid A), P(A\mid B))$
L	Laplace	0 . . . 1	$\max(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2})$
IS	Cosine	0 . . . 1	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
γ	coherence(Jaccard)	0 . . . 1	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
α	all_confidence	0 . . . 1	$\frac{P(A,B)}{\max(P(A),P(B))}$
o	odds ratio	0 . . . ∞	$\frac{P(A,B)P(\overline{A},\overline{B})}{P(\overline{A},B)P(A,\overline{B})}$
V	Conviction	0.5 . . . ∞	$\max(\frac{P(A)P(\overline{B})}{P(A\overline{B})}, \frac{P(B)P(\overline{A})}{P(B\overline{A})})$
λ	lift	0 . . . ∞	$\frac{P(A,B)}{P(A)P(B)}$
S	Collective strength	0 . . . ∞	$\frac{P(A,B)+P(\overline{AB})}{P(A)P(B)+P(\overline{A})P(\overline{B})} \times \frac{1-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A,B)-P(\overline{AB})}$
χ^2	χ^2	0 . . . ∞	$\Sigma_i \frac{(P(A_i)-E_i)^2}{E_i}$

Chapter 5: Mining Frequent Patterns, Association and Correlations

1. Basic concepts and a road map
2. Efficient and scalable frequent itemset mining methods
3. Mining various kinds of association rules
4. From association mining to correlation analysis
5. **Constraint-based association mining**
6. 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
  - System optimization: explores such constraints for efficient mining—constraint-based mining

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 in Dec.'02
- 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 $\geq 3\%$, min_confidence $\geq 60\%$

Constrained Mining vs. Constraint-Based Search

- Constrained mining vs. constraint-based search/reasoning
  - Both are aimed at reducing search space
  - Finding all patterns satisfying constraints vs. finding some (or one) answer in constraint-based search in AI
  - Constraint-pushing vs. heuristic search
  - It is an interesting research problem on how to integrate them
- Constrained mining vs. query processing in DBMS
  - Database query processing requires to find all
  - Constrained pattern mining shares a similar philosophy as pushing selections deeply in query processing

Anti-Monotonicity in Constraint Pushing

- Anti-monotonicity
  - *When an intemset S **violates** the constraint, so does any of its superset*
  - *sum(S.Price)* $\leq$ *v* is anti-monotone
  - *sum(S.Price)* $\geq$ *v* is not anti-monotone
- Example. C: range(S.profit) $\leq$ 15 is anti-monotone
  - Itemset *ab* violates C
  - So does every superset of *ab*

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Monotonicity for Constraint Pushing

- Monotonicity
  - *When an intemset S* ***satisfies*** *the constraint, so does any of its superset*
  - *sum(S.Price)* $\geq$ *v* is monotone
  - *min(S.Price)* $\leq$ *v* is monotone
- Example. C: range(S.profit) $\geq$ 15
  - Itemset *ab* satisfies C
  - So does every superset of *ab*

TDB (min_sup=2)

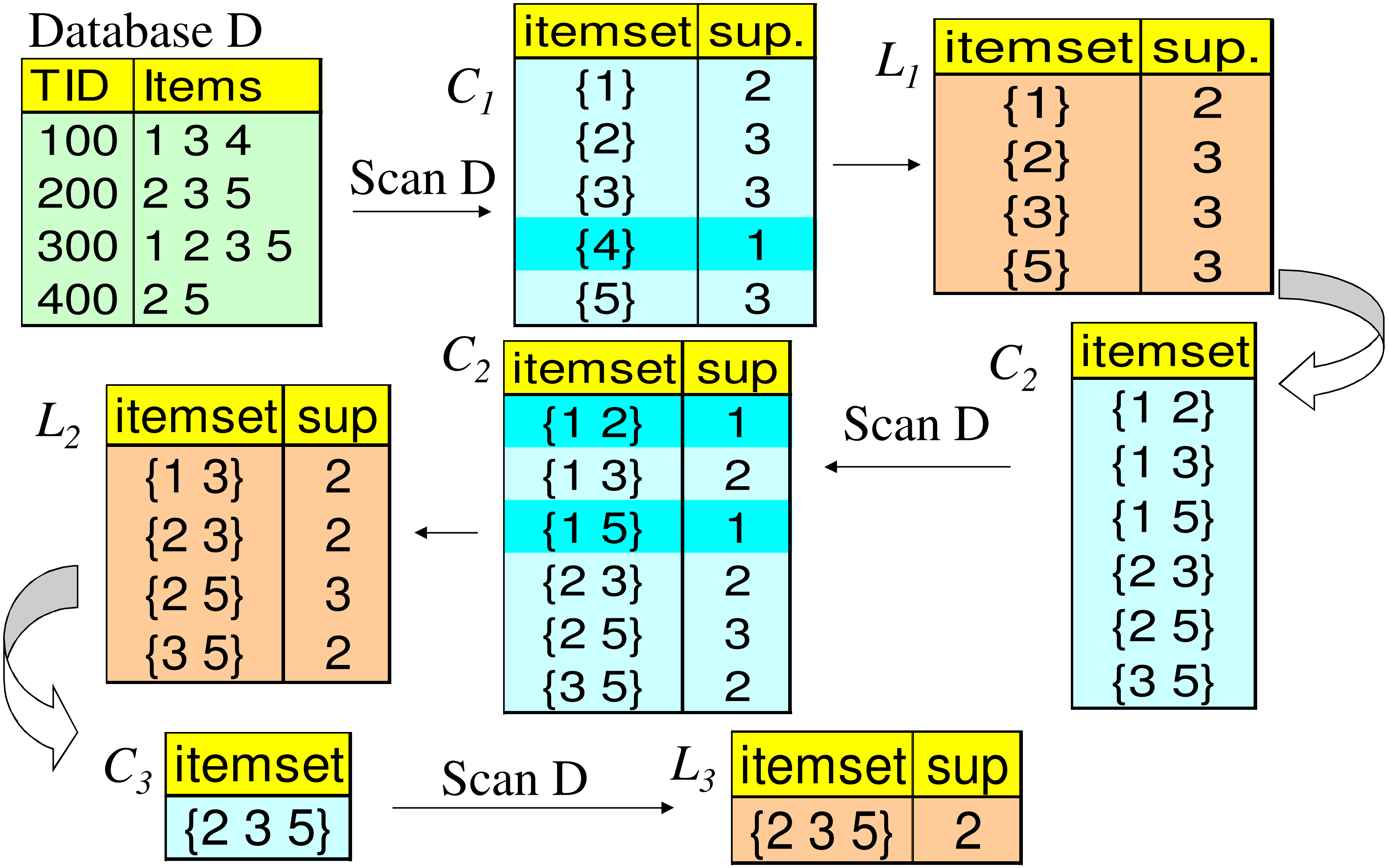
TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Succinctness

- Succinctness:
  - Given $A_{1,}$ the set of items satisfying a succinctness constraint C, then any set S satisfying C is based on A_1 , i.e., S contains a subset belonging to A_1
  - Idea: Without looking at the transaction database, whether an itemset S satisfies constraint C can be determined based on the selection of items
  - $min(S.Price) \leq v$ is succinct
  - $sum(S.Price) \geq v$ is not succinct
- Optimization: If C is succinct, C is pre-counting pushable

The Apriori Algorithm — Example



Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

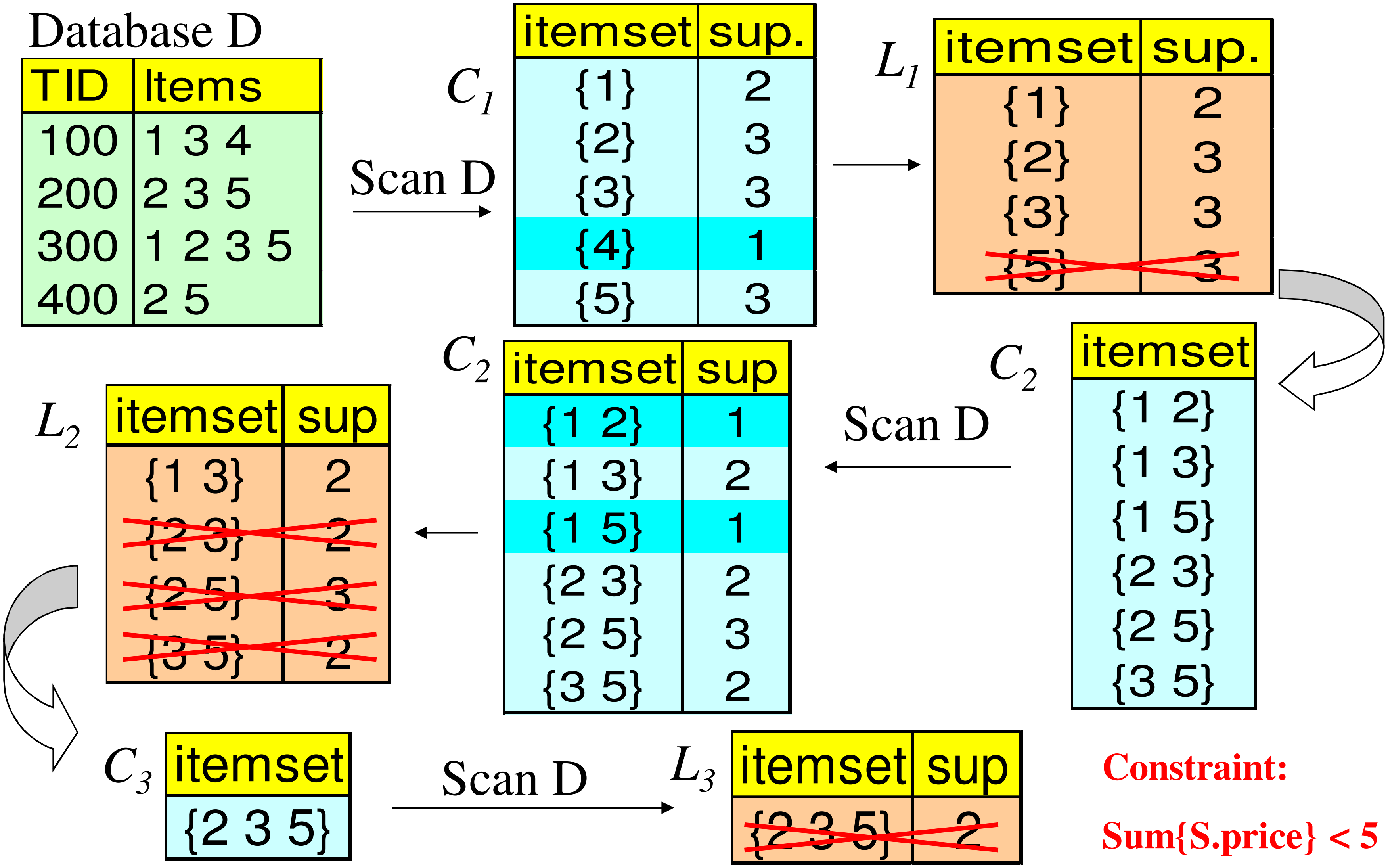
itemset
{2 3 5}

Scan D

L_3

itemset	sup
{2 3 5}	2

Naïve Algorithm: Apriori + Constraint



Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
~~{5}~~	~~3~~

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

C_3

itemset
{2 3 5}

Scan D

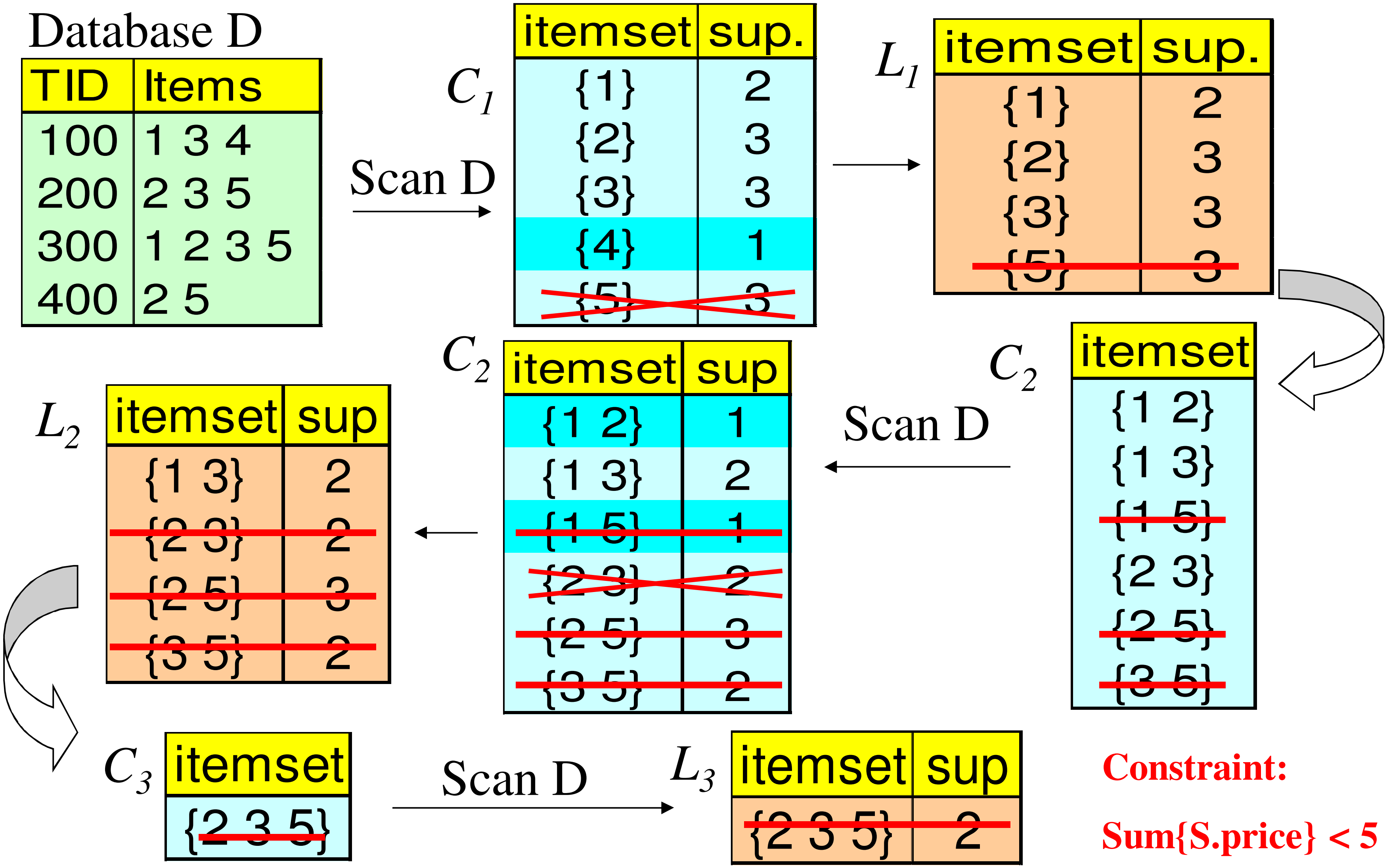
L_3

itemset	sup
~~{2 3 5}~~	~~2~~

Constraint:

Sum{S.price} < 5

The Constrained Apriori Algorithm: Push an Anti-monotone Constraint Deep



Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
~~{5}~~	~~3~~

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
~~{5}~~	~~3~~

C_2

itemset
{1 2}
{1 3}
~~{1 5}~~
{2 3}
~~{2 5}~~
~~{3 5}~~

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
~~{1 5}~~	~~1~~
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

C_3

itemset
~~{2 3 5}~~

Scan D

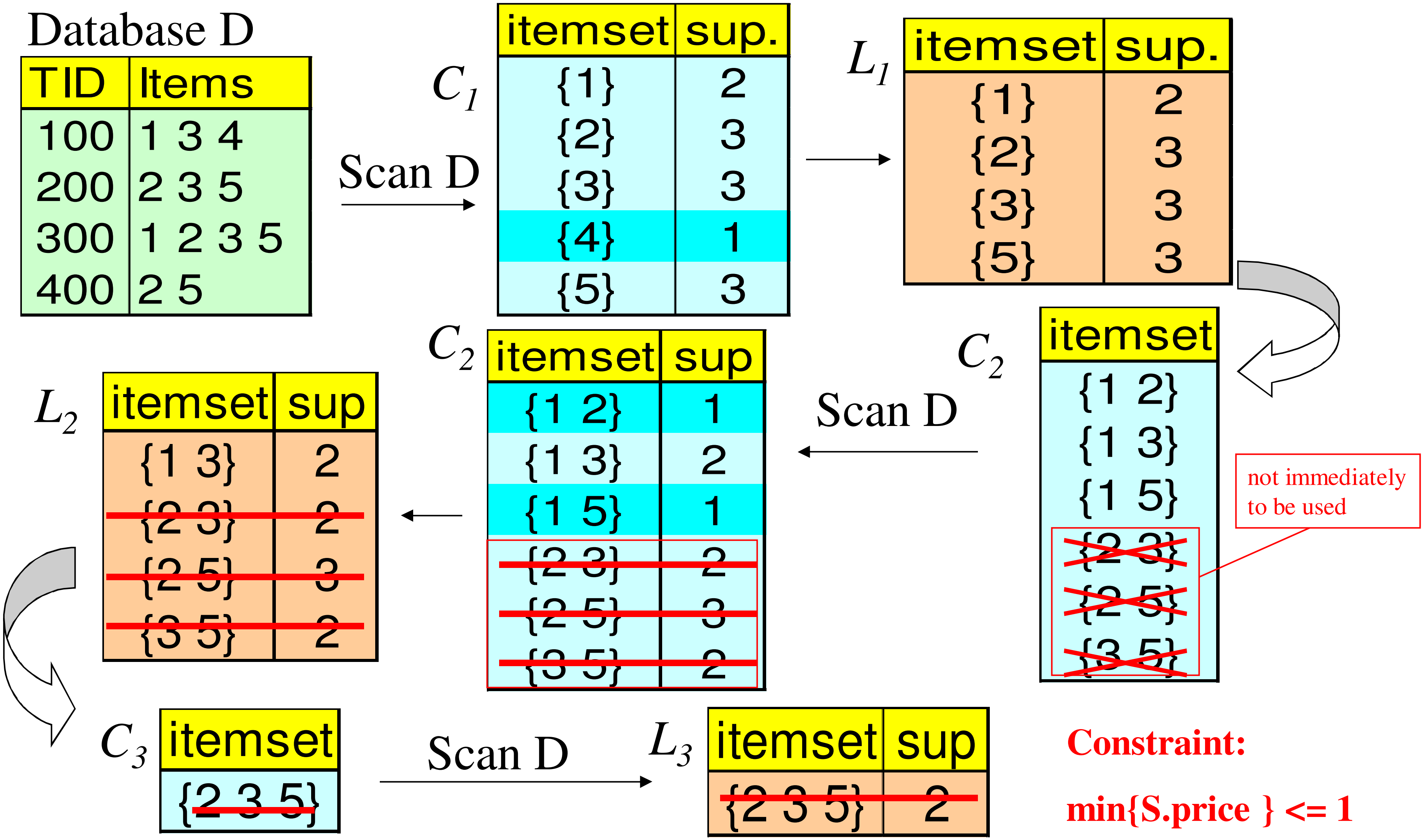
L_3

itemset	sup
~~{2 3 5}~~	~~2~~

Constraint:

Sum{S.price} < 5

The Constrained Apriori Algorithm: Push a Succinct Constraint Deep



Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
~~{2 3}~~
~~{2 5}~~
~~{3 5}~~

not immediately to be used

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

C_3

itemset
~~{2 3 5}~~

Scan D

L_3

itemset	sup
~~{2 3 5}~~	~~2~~

Constraint:

min{S.price } <= 1

Converting "Tough" Constraints

- Convert tough constraints into anti-monotone or monotone by properly ordering items
- Examine C: avg(*S*.profit) ≥ 25
  - Order items in value-descending order
    - ✓ <*a, f, g, d, b, h, c, e*>
  - If an itemset *afb* violates C
    - ✓ So does *afbh, afb**
    - ✓ It becomes anti-monotone!

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Strongly Convertible Constraints

- avg(X) ≥ 25 is convertible anti-monotone w.r.t. item value descending order R: <*a, f, g, d, b, h, c, e*>
  - If an itemset *af* violates a constraint C, so does every itemset with *af* as prefix, such as *afd*
- avg(X) ≥ 25 is convertible monotone w.r.t. item value ascending order R^{-1}: <*e, c, h, b, d, g, f, a*>
  - If an itemset *d* satisfies a constraint *C*, so does itemsets *df* and *dfa*, which having *d* as a prefix
- Thus, avg(X) ≥ 25 is strongly convertible

Item	**Profit**
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Can Apriori Handle Convertible Constraint?

- A convertible, not monotone nor anti-monotone nor succinct constraint cannot be pushed deep into the an Apriori mining algorithm
  - Within the level wise framework, no direct pruning based on the constraint can be made
  - Itemset df violates constraint C: avg(X)>=25
  - Since adf satisfies C, Apriori needs df to assemble adf, df cannot be pruned
- But it can be pushed into frequent-pattern growth framework!

Item	Value
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Mining With Convertible Constraints

- C: avg(X) >= 25, min_sup=2
- List items in every transaction in value descending order R: <a, f, g, d, b, h, c, e>
  - C is convertible anti-monotone w.r.t. R
- Scan TDB once
  - remove infrequent items
    - ✓ Item h is dropped
  - Itemsets a and f are good, …
- Projection-based mining
  - Imposing an appropriate order on item projection
  - Many tough constraints can be converted into (anti)-monotone

Item	**Value**
a	40
f	30
g	20
d	10
b	0
h	-10
c	-20
e	-30

TDB (min_sup=2)

TID	**Transaction**
10	a, f, d, b, c
20	f, g, d, b, c
30	a, f, d, c, e
40	f, g, h, c, e

Handling Multiple Constraints

- Different constraints may require different or even conflicting item-ordering
- If there exists an order *R* s.t. both C_1 and C_2 are convertible w.r.t. *R*, then there is no conflict between the two convertible constraints
- If there exists conflict on order of items
  - Try to satisfy one constraint first
  - Then using the order for the other constraint to mine frequent itemsets in the corresponding projected database

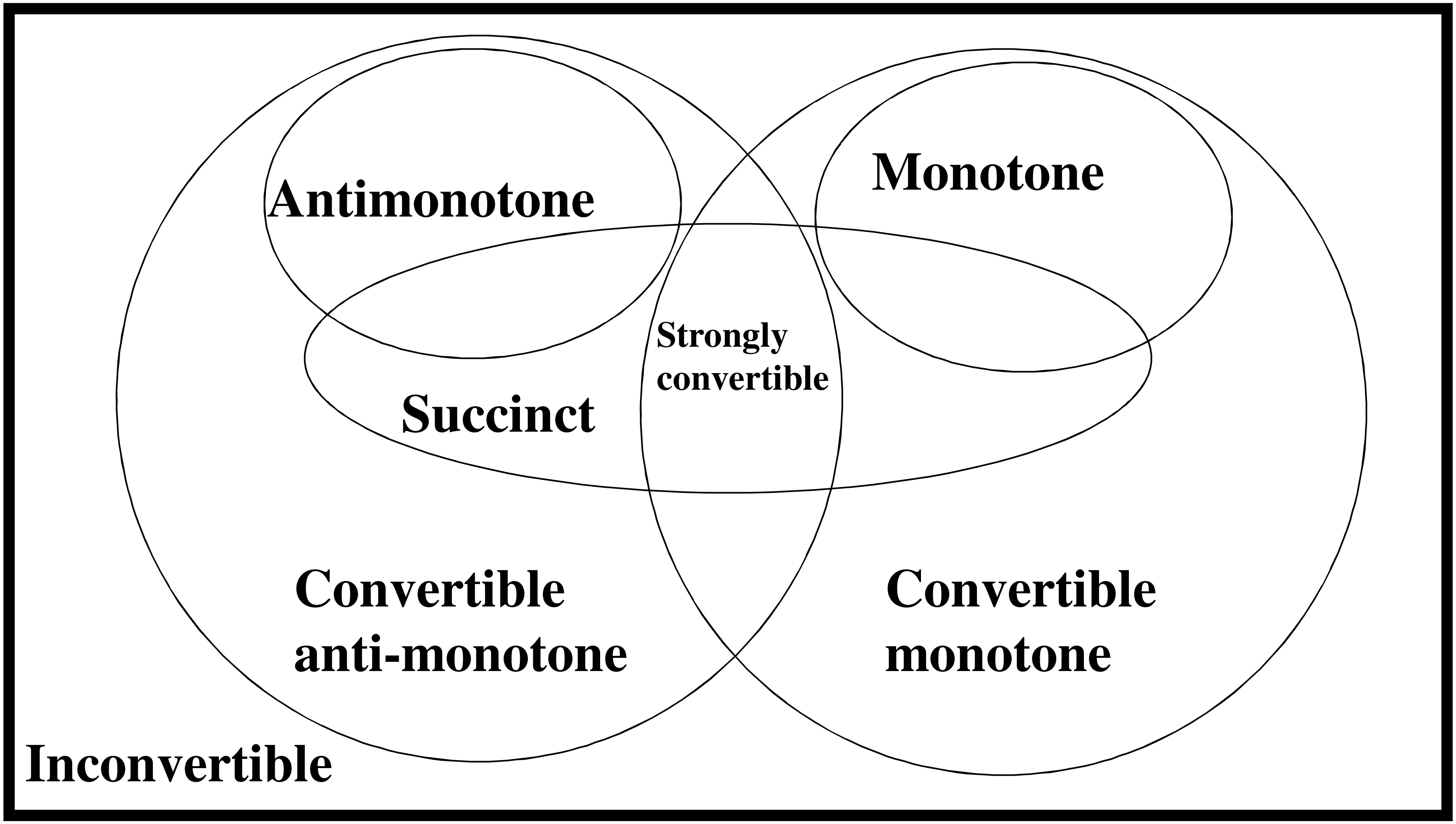
What Constraints Are Convertible?

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	Antimonotone	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
$\mathrm{count}(S) \leq v$	yes	no	weakly
$\mathrm{count}(S) \geq v$	no	yes	weakly
$\mathrm{sum}(S) \leq v \ (a \in S, a \geq 0)$	yes	no	no
$\mathrm{sum}(S) \geq v \ (a \in S, a \geq 0)$	no	yes	no
$\mathrm{range}(S) \leq v$	yes	no	no
$\mathrm{range}(S) \geq v$	no	yes	no
$\mathrm{avg}(S)\ \theta\ v, \theta \in \{=, \leq, \geq\}$	convertible	convertible	no
$\mathrm{support}(S) \geq \xi$	yes	no	no
$\mathrm{support}(S) \leq \xi$	no	yes	no

A Classification of Constraints



Chapter 5: Mining Frequent Patterns, Association and Correlations

1. Basic concepts and a road map
2. Efficient and scalable frequent itemset mining methods
3. Mining various kinds of association rules
4. From association mining to correlation analysis
5. Constraint-based association mining
6. **Summary**

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, CLOSET+, ...)
  - Vertical format approach (CHARM, ...)
- Mining a variety of rules and interesting patterns
- Constraint-based mining
- Mining sequential and structured patterns
- Extensions and applications

Frequent-Pattern Mining: Research Problems

- Mining fault-tolerant frequent, sequential and structured patterns
  - Patterns allows limited faults (insertion, deletion, mutation)
- Mining truly interesting patterns
  - Surprising, novel, concise, …
- Application exploration
  - E.g., DNA sequence analysis and bio-pattern classification
  - “Invisible” data mining

Ref: Basic Concepts of Frequent Pattern Mining

- (Association Rules) R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. SIGMOD'93.
- (Max-pattern) R. J. Bayardo. Efficiently mining long patterns from databases. SIGMOD'98.
- (Closed-pattern) N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. ICDT'99.
- (Sequential pattern) R. Agrawal and R. Srikant. Mining sequential patterns. ICDE'95