Data Warehousing
資料倉儲

Association Analysis

992DW06
MI4
Tue. 8,9 (15:10-17:00) L413

Min-Yuh Day
戴敏育
Assistant Professor
專任助理教授
Dept. of Information Management, Tamkang University
淡江大學 資訊管理學系

http://mail.im.tku.edu.tw/~myday/

2011-04-26

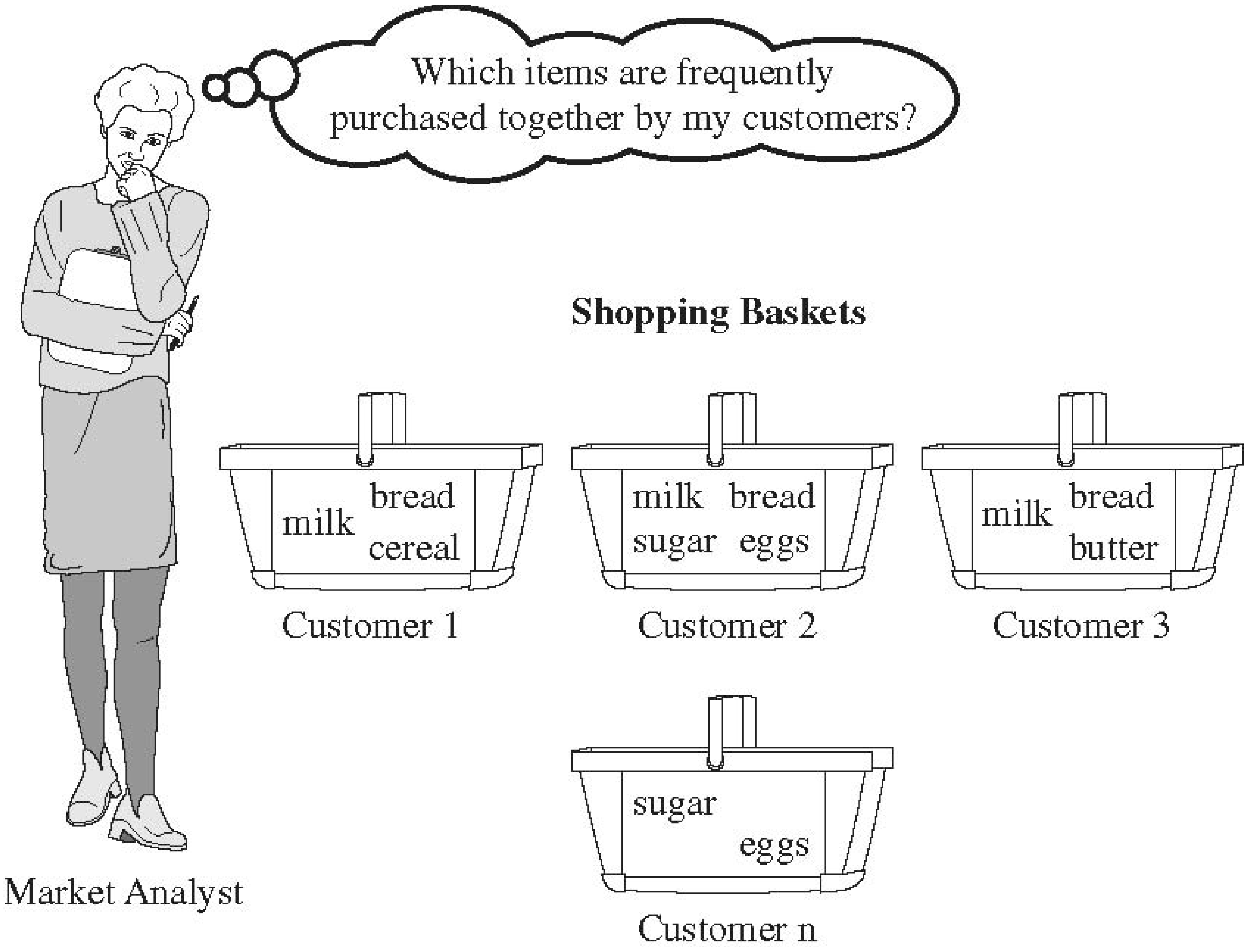
Syllabus

1	100/02/15	Introduction to Data Warehousing
2	100/02/22	Data Warehousing, Data Mining, and Business Intelligence
3	100/03/01	Data Preprocessing: Integration and the ETL process
4	100/03/08	Data Warehouse and OLAP Technology
5	100/03/15	Data Warehouse and OLAP Technology
6	100/03/22	Data Warehouse and OLAP Technology
7	100/03/29	Data Warehouse and OLAP Technology
8	100/04/05	(放假一天) (民族掃墓節)
9	100/04/12	Data Cube Computation and Data Generation
10	100/04/19	Mid-Term Exam (期中考試週)
11	100/04/26	Association Analysis
12	100/05/03	Classification and Prediction, Cluster Analysis
13	100/05/10	Social Network Analysis, Link Mining, Text and Web Mining
14	100/05/17	Project Presentation
15	100/05/24	Final Exam (畢業班考試)

Association Analysis: Mining Frequent Patterns, Association and Correlations

- Association Analysis
- Mining Frequent Patterns
- Association and Correlations
- Apriori Algorithm
- Mining Multilevel Association Rules

Market Basket Analysis



What Is Frequent Pattern Analysis?

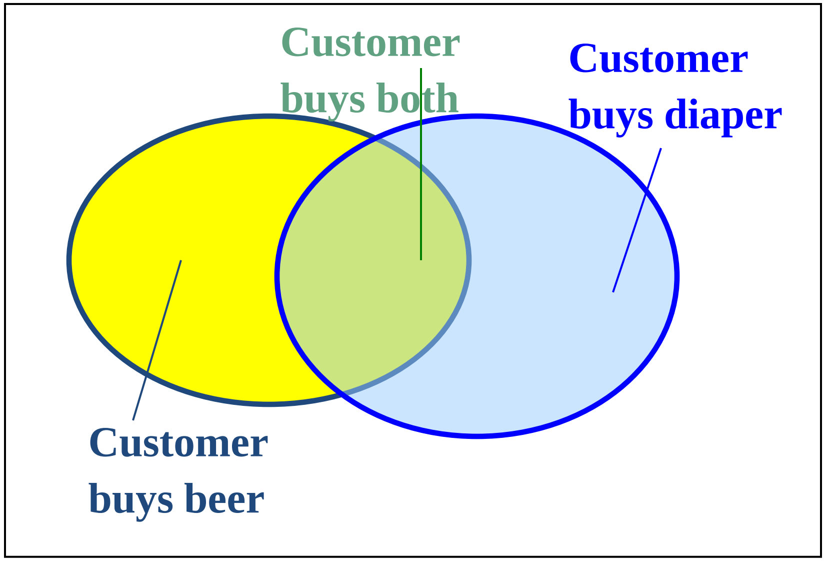
- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
  - What 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

Basic Concepts: Frequent Patterns and Association Rules

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 $X \rightarrow Y$ 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* (60%, 100%)

D → *A* (60%, 75%)

A → *D* (support = 3/5 = 60%, confidence = 3/3 =100%)

D → *A* (support = 3/5 = 60%, confidence = 3/4 = 75%)

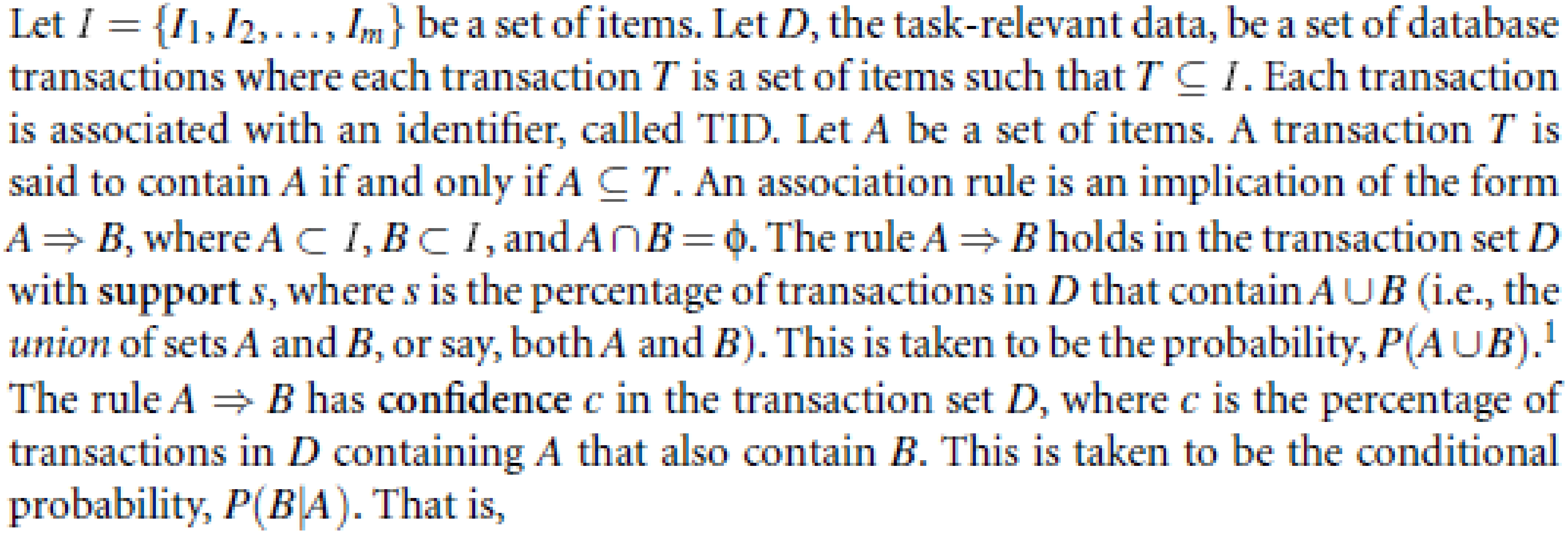
Market basket analysis

- Example
  - Which groups or sets of items are customers likely to purchase on a given trip to the store?
- Association Rule
  - *Computer → antivirus_software [support = 2%; confidence = 60%]*
    - A support of 2% means that 2% of all the transactions under analysis show that computer and antivirus software are purchased together.
    - A confidence of 60% means that 60% of the customers who purchased a computer also bought the software.

Association rules

- Association rules are considered interesting if they satisfy both
  - a minimum support threshold and
  - a minimum confidence threshold.

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 → B) = P(A ∪ B)

Confidence (A → B) = P(B|A)

Support $(A \rightarrow B) = P(A \cup B)$
Confidence $(A \rightarrow B) = P(B|A)$

- The notation *$P(A \cup B)$ indicates the probability that a transaction contains the union of set A and set B*
  - *(i.e., it contains every item in A and in B).*
- *This should not be confused with P(A or B),* which indicates the probability that a transaction contains either *A or B.*

- Rules that satisfy both a minimum support threshold (*min_sup*) and a minimum confidence threshold (*min_conf*) are called **strong**.
- By convention, we write support and confidence values so as to occur between 0% and 100%, rather than 0 to 1.0.

- itemset
  - A set of items is referred to as an itemset.
- K-itemset
  - An itemset that contains *k items is a k*-itemset*.*
- Example:
  - The set {*computer, antivirus software*} is a 2-itemset.

Absolute Support and Relative Support

- Absolute Support
  - The occurrence frequency of an itemset is the number of transactions that contain the itemset
    - frequency, support count, or count of the itemset
  - Ex: 3
- Relative support
  - Ex: 60%

- If the relative support of an itemset *I satisfies a prespecified minimum support threshold, then I is a* frequent itemset.
  - *i.e., the absolute* support of *I satisfies the corresponding minimum support count threshold*
- The set of frequent *k-itemsets is commonly denoted by* L_K

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

- the confidence of rule *A→ B can be easily derived from the* support counts of *A and A ∪ B.*
- once the support counts of *A, B, and A ∪ B are* found, it is straightforward to derive the corresponding association rules *A→B and B→*A and check whether they are strong.
- Thus the problem of mining association rules can be reduced to that of mining frequent itemsets.

Association rule mining: Two-step process

1. Find all frequent itemsets
   - 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
   - By definition, these rules must satisfy minimum support and minimum confidence.

Closed frequent itemsets and maximal frequent itemsets

- Suppose that a transaction database has only two transactions:
  - {(*a1, a2, …, a100*); (*a1, a2,… , a50*)}
- Let the minimum support count threshold be *min_sup*=1.
- We find two closed frequent itemsets and their support counts, that is,
  - *C* = {{*a1, a2, …, a100*}:1; {*a1, a2,… , a50*}: 2}
- There is one maximal frequent itemset:
  - *M* = {{*a1, a2, …, a100*}:1}
    - (We cannot include {*a1, a2,… , a50*} as a maximal frequent itemset because it has a frequent super-set, {*a1, a2, …, a100*})

Frequent Pattern Mining

- Based on the *completeness of patterns to be mined*
- Based on the *levels of abstraction involved in the rule set*
- Based on the *number of data dimensions involved in the rule*
- Based on the *types of values handled in the rule*
- Based on the *kinds of rules to be mined*
- Based on the *kinds of patterns to be mined*

Based on the *levels of abstraction involved in the rule set*

- *buys(X, “computer”))→buys(X, “HP printer”)*
- *buys(X, “laptop computer”)) →buys(X, “HP printer”)*

Based on the *number of data dimensions involved in the rule*

- Single-dimensional association rule
  - *buys(X, “computer”)) → buys(X, “antivirus software”)*
- Multidimensional association rule
  - *age(X, “30,…,39”) ^ income (X, “42K,…,48K”)) → buys (X, “high resolution TV”)*

Efficient and Scalable Frequent Itemset Mining Methods

- The Apriori Algorithm
  - Finding Frequent Itemsets Using Candidate Generation

Apriori Algorithm

- **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, as we shall* see following.

Apriori Algorithm

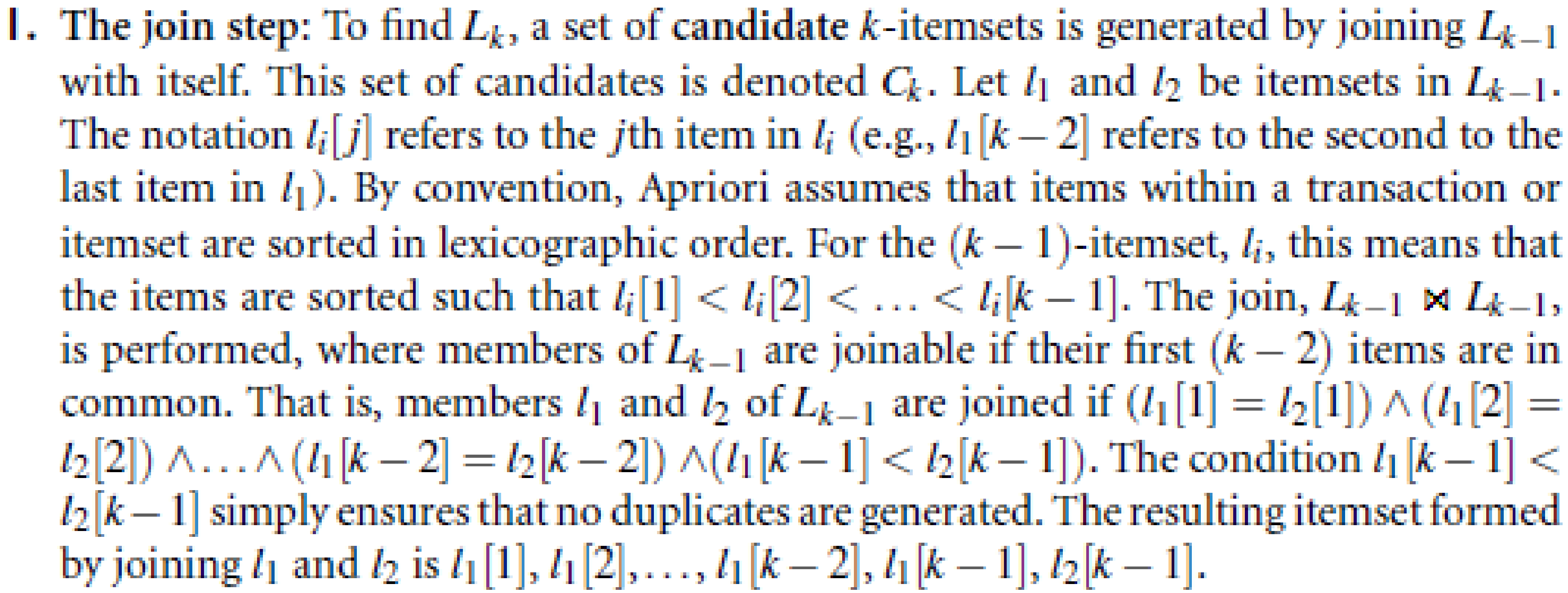
- Apriori employs an iterative approach known as a *level-wise search, where k-itemsets are used to explore (k+1)-itemsets.*
- *First, the set of frequent 1-itemsets is found* by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support. The resulting set is denoted L_1.
- *Next,* L_1 *is used to find* L_2, the set of frequent 2-itemsets, which is used to find L_3*, and so on, until no more frequent k-itemsets can be found.*
- *The finding of each* L_k *requires one full scan of the database.*

Apriori Algorithm

- To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the Apriori property.
- Apriori property
  - *All nonempty subsets of a frequent itemset must also be frequent.*

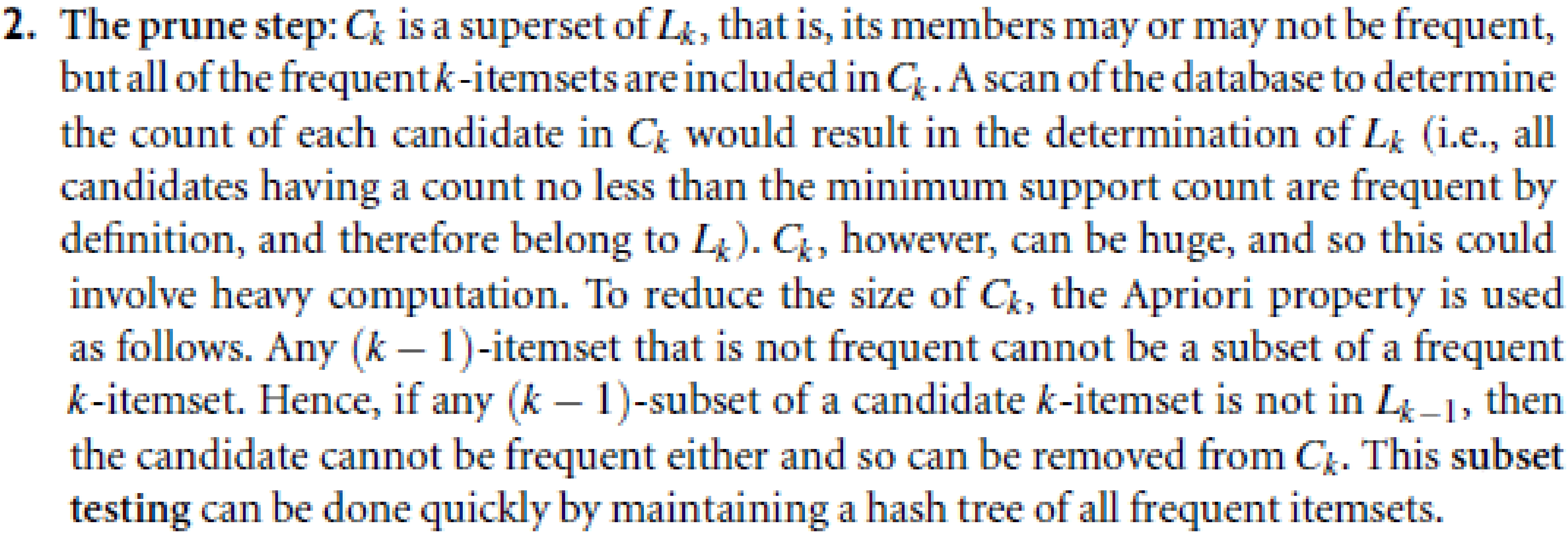
- *How is the Apriori property used in the algorithm?*
  - How L_{k-1} *is used to find* L_k *for k >= 2.*
  - A two-step process is followed, consisting of join and prune actions.

Apriori property used in algorithm 1. The join step



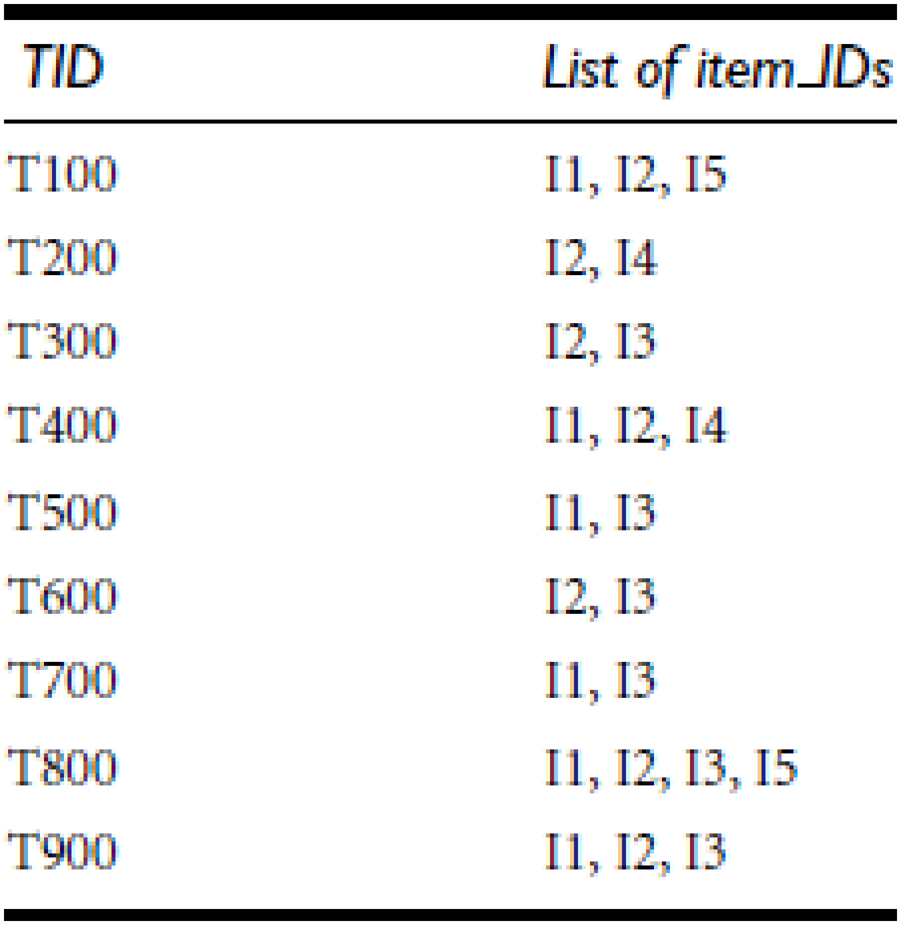
1. **The join step:** To find L_k, a set of **candidate** k-itemsets is generated by joining L_{k-1} with itself. This set of candidates is denoted C_k. Let l_1 and l_2 be itemsets in L_{k-1}. The notation $l_i[j]$ refers to the jth item in l_i (e.g., $l_1[k-2]$ refers to the second to the last item in l_1). By convention, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order. For the $(k-1)$-itemset, l_i, this means that the items are sorted such that $l_i[1] < l_i[2] < \ldots < l_i[k-1]$. The join, $L_{k-1} \bowtie L_{k-1}$, is performed, where members of L_{k-1} are joinable if their first $(k-2)$ items are in common. That is, members l_1 and l_2 of L_{k-1} are joined if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \ldots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$. The condition $l_1[k-1] < l_2[k-1]$ simply ensures that no duplicates are generated. The resulting itemset formed by joining l_1 and l_2 is $l_1[1], l_1[2], \ldots, l_1[k-2], l_1[k-1], l_2[k-1]$.

Apriori property used in algorithm
2. The prune step



2. **The prune step:** C_k is a superset of L_k, that is, its members may or may not be frequent, but all of the frequent k-itemsets are included in C_k. A scan of the database to determine the count of each candidate in C_k would result in the determination of L_k (i.e., all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to L_k). C_k, however, can be huge, and so this could involve heavy computation. To reduce the size of C_k, the Apriori property is used as follows. Any $(k-1)$-itemset that is not frequent cannot be a subset of a frequent k-itemset. Hence, if any $(k-1)$-subset of a candidate k-itemset is not in L_{k-1}, then the candidate cannot be frequent either and so can be removed from C_k. This **subset testing** can be done quickly by maintaining a hash tree of all frequent itemsets.

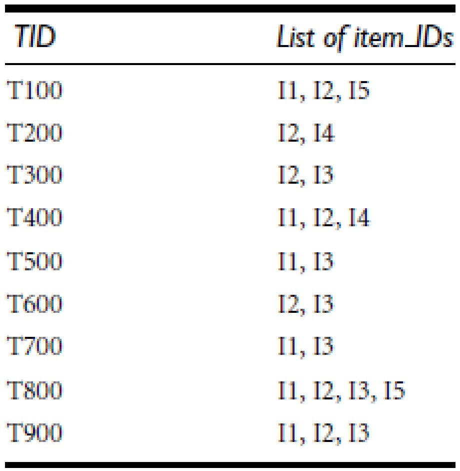
Transactional data for an *AllElectronics* branch



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

Example: Apriori

- Let’s look at a concrete example, based on the *AllElectronics transaction database, D.*
- *There are nine transactions in this database, that is, |D| = 9.*
- Apriori algorithm for finding frequent itemsets in *D*

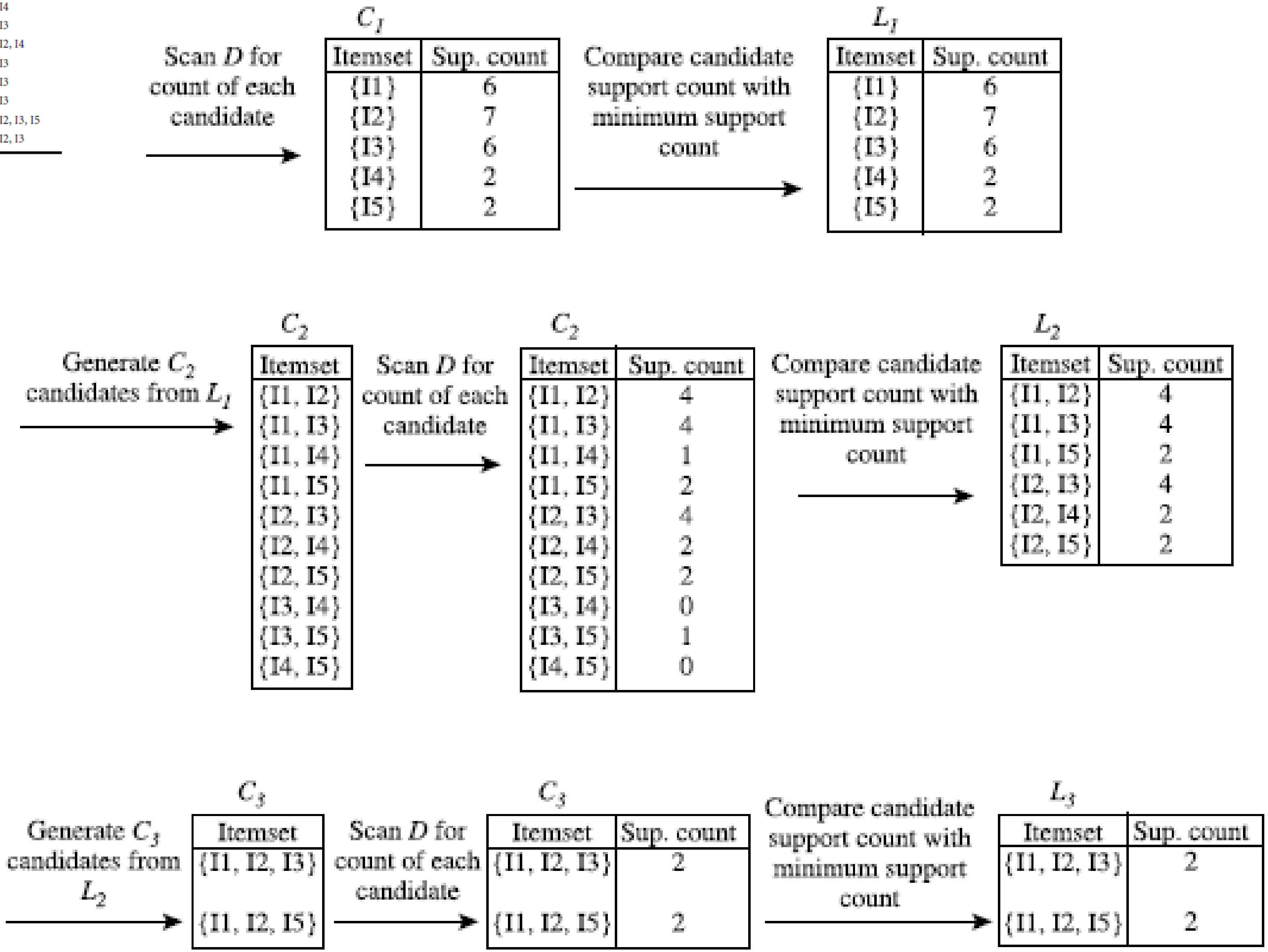


TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Example: Apriori Algorithm

Generation of candidate itemsets and frequent itemsets, where the minimum support count is 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



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

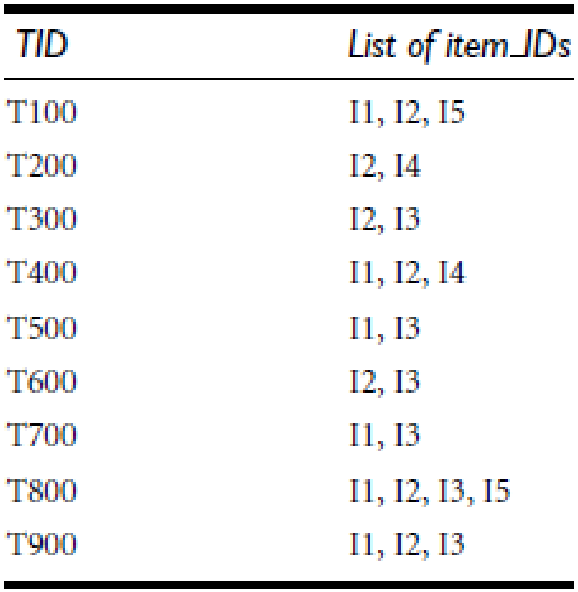
Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Compare candidate support count with minimum support count

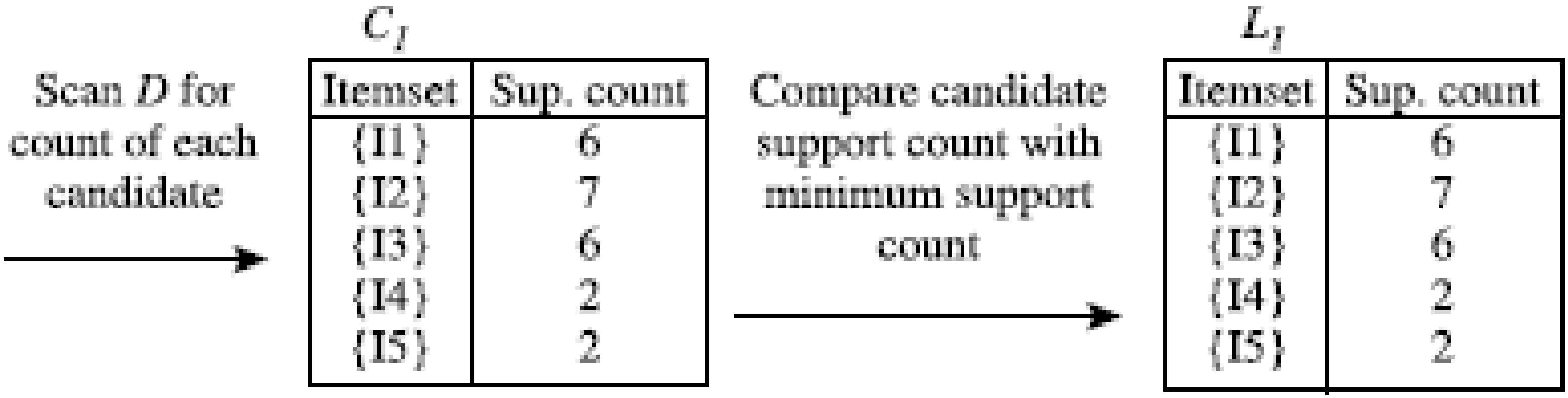
L_3

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

Example: Apriori Algorithm
$C_1 \rightarrow L_1$



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

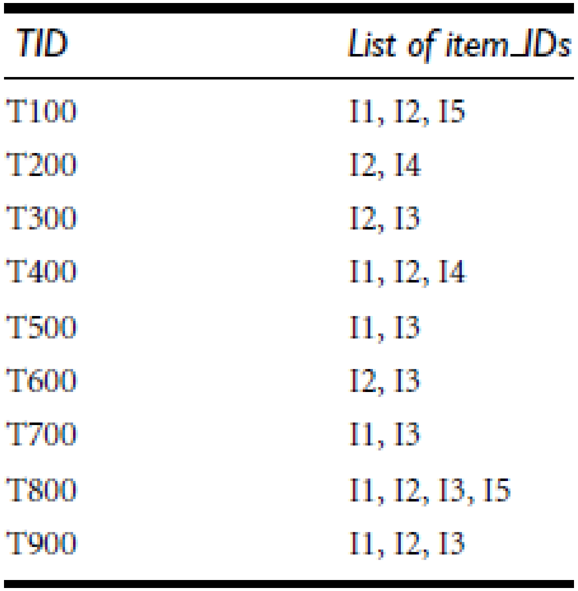
C_1

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

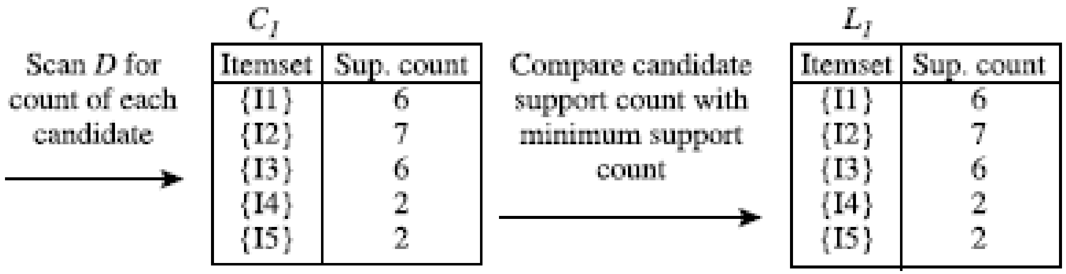
Compare candidate support count with minimum support count

L_1

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



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

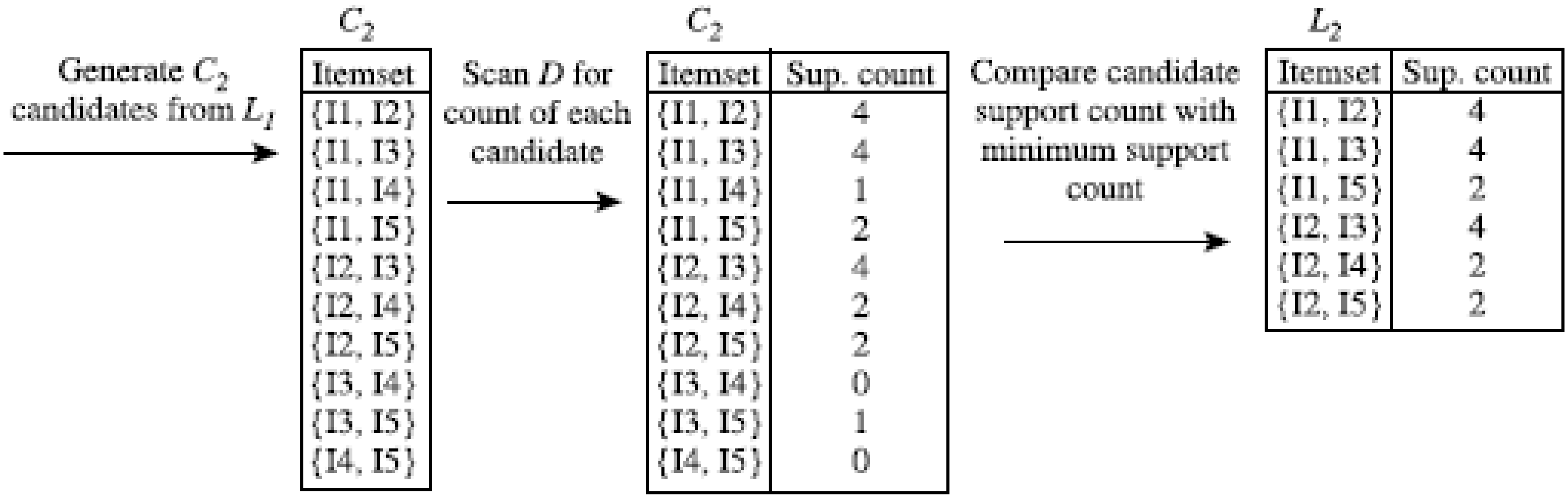
Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Compare candidate support count with minimum support count

L_1

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

Example: Apriori Algorithm $C_2 \rightarrow L_2$



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

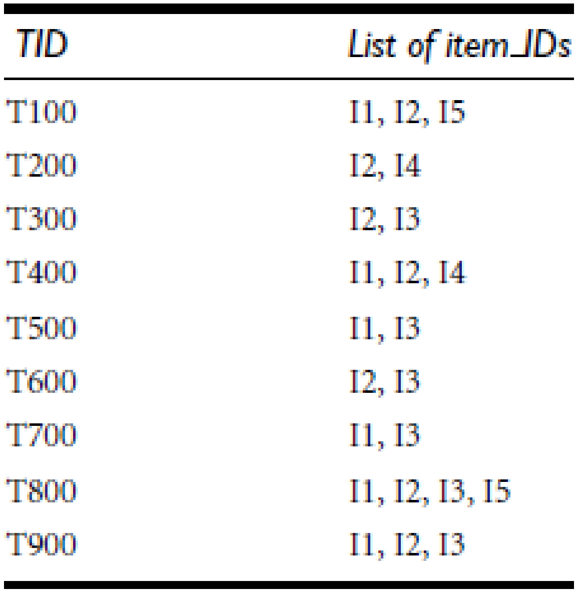
C_2

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

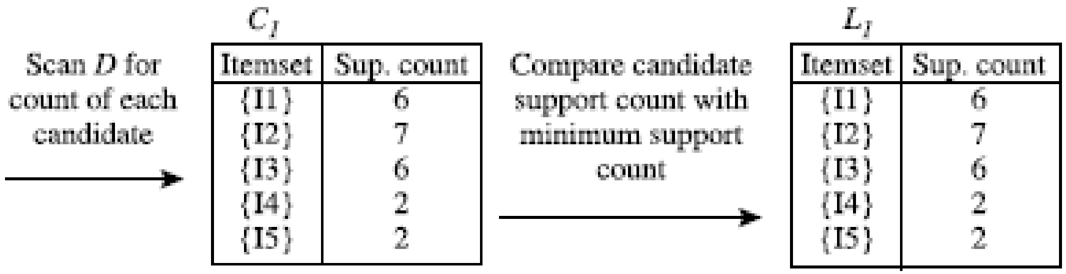
Compare candidate support count with minimum support count

L_2

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



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

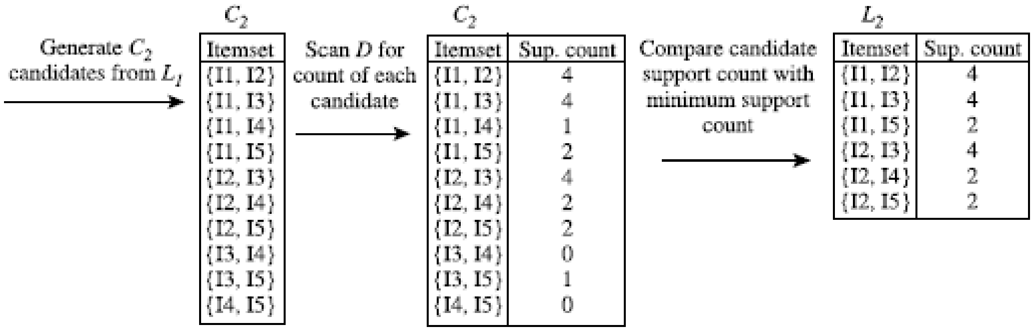
C_1

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

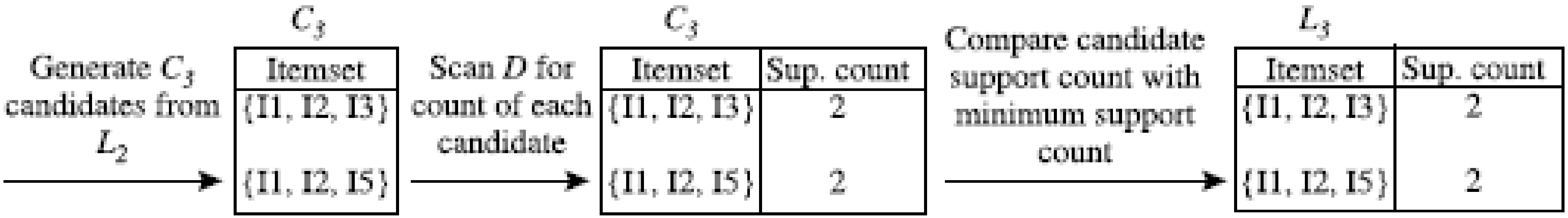
Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

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

Example: Apriori Algorithm
C_3 → L_3



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

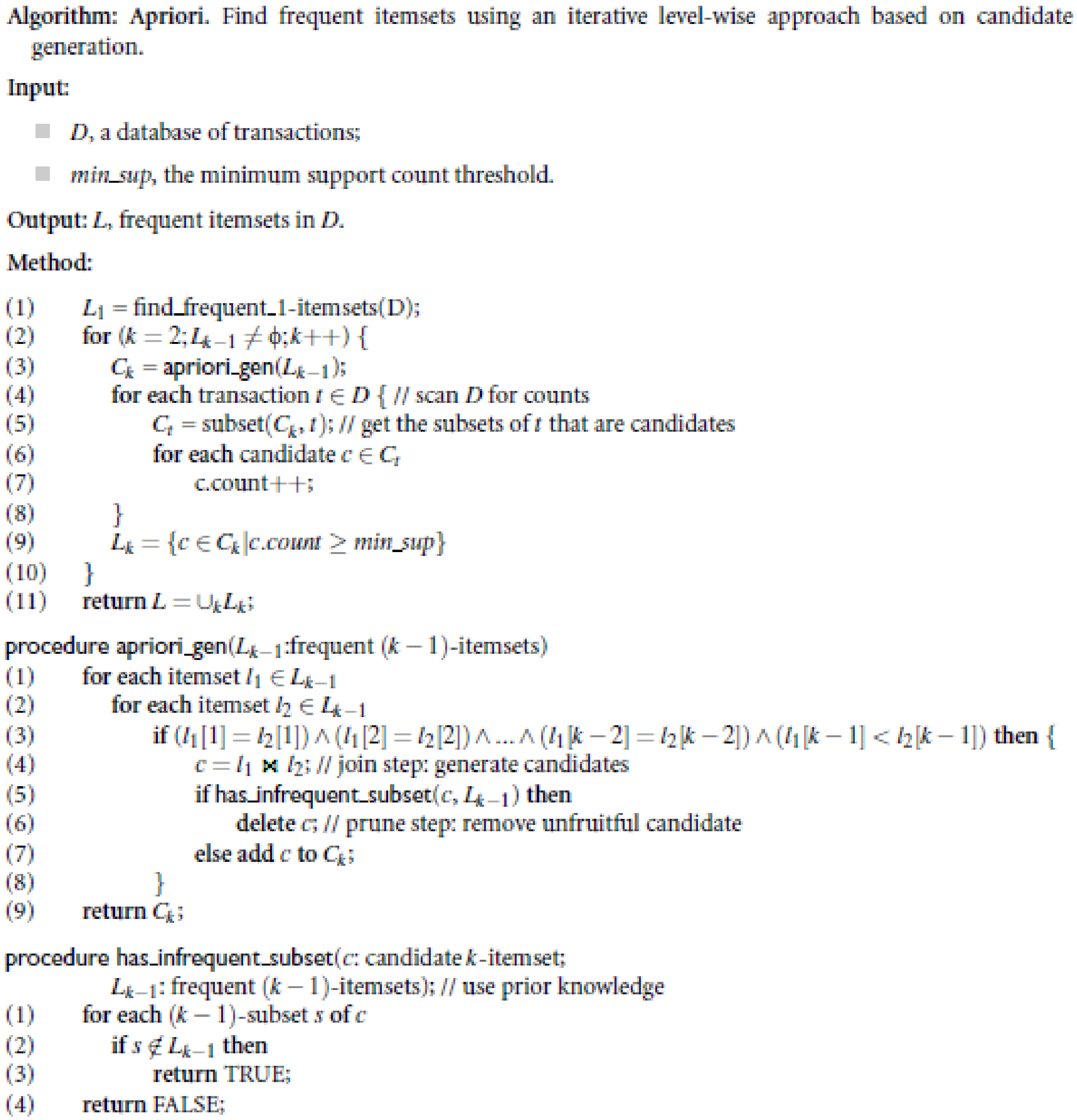
Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Compare candidate support count with minimum support count

L_3

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

The Apriori algorithm for discovering frequent itemsets for mining Boolean association rules.



Algorithm: Apriori. Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input:

- D, a database of transactions;
- min_sup, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

```
$L_1$ = find_frequent_1-itemsets(D);
for ($k = 2; L_{k-1} \neq \phi; k{+}{+}$) {
    $C_k$ = apriori_gen($L_{k-1}$);
    for each transaction $t \in D$ { // scan $D$ for counts
        $C_t$ = subset($C_k, t$); // get the subsets of $t$ that are candidates
        for each candidate $c \in C_t$
            c.count++;
    }
    $L_k = \{c \in C_k | c.count \geq min\_sup\}$
}
return $L = \cup_k L_k$;

procedure apriori_gen($L_{k-1}$:frequent $(k-1)$-itemsets)
for each itemset $l_1 \in L_{k-1}$
    for each itemset $l_2 \in L_{k-1}$
        if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge ... \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ then {
            $c = l_1 \bowtie l_2$; // join step: generate candidates
            if has_infrequent_subset($c, L_{k-1}$) then
                delete $c$; // prune step: remove unfruitful candidate
            else add $c$ to $C_k$;
        }
return $C_k$;

procedure has_infrequent_subset($c$: candidate $k$-itemset;
            $L_{k-1}$: frequent $(k-1)$-itemsets); // use prior knowledge
for each $(k-1)$-subset $s$ of $c$
    if $s \notin L_{k-1}$ then
        return TRUE;
return FALSE;
```

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

1st scan

C_1

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

L_1

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

C_2

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

2nd scan

C_2

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

L_2

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

C_3

Itemset
{B, C, E}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

The Apriori Algorithm

- Pseudo-code:

```
$C_k$: Candidate itemset of size k
$L_k$ : frequent itemset of size k

$L_1$ = {frequent items};
for ($k$ = 1; $L_k$ !=$\varnothing$; $k$++) 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 $\cup_k L_k$;
```

Important Details of Apriori

- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- How to count supports of candidates?
- Example of Candidate-generation
  - L_3={*abc, abd, acd, ace, bcd*}
  - Self-joining: L_3*L_3
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in L_3
  - C_4={*abcd*}

How to Generate Candidates?

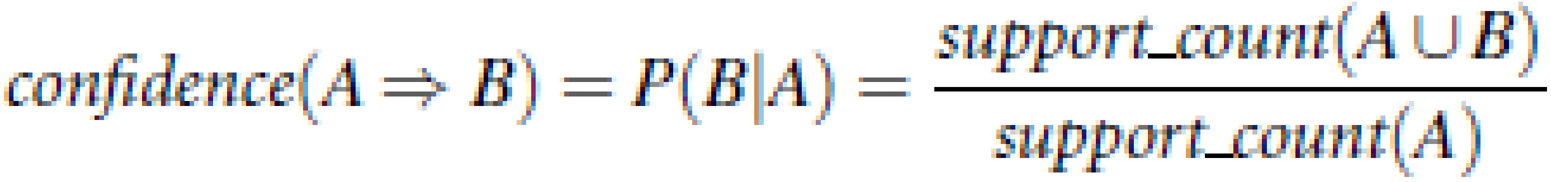
- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}

```
insert into $C_k$
select $p.item_1, p.item_2, \dots, p.item_{k-1}, q.item_{k-1}$
from $L_{k-1}\ p, L_{k-1}\ q$
where $p.item_1=q.item_1, \dots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
```

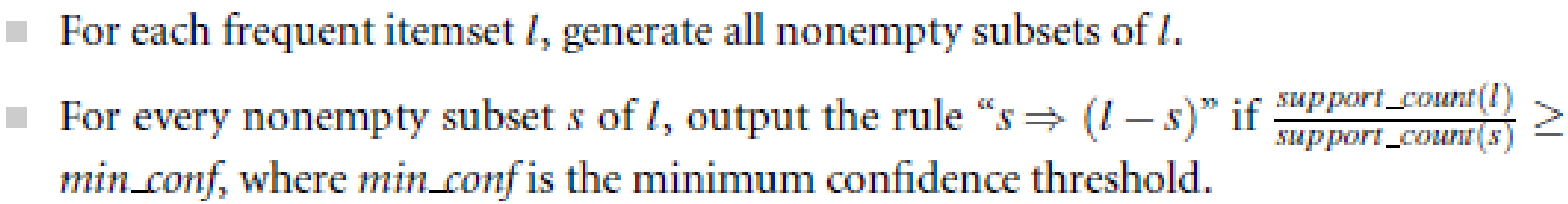
- Step 2: pruning

```
forall itemsets c in $C_k$ do
    forall (k-1)-subsets s of c do
        if (s is not in $L_{k-1}$) then delete c from $C_k$
```

Generating Association Rules from Frequent Itemsets



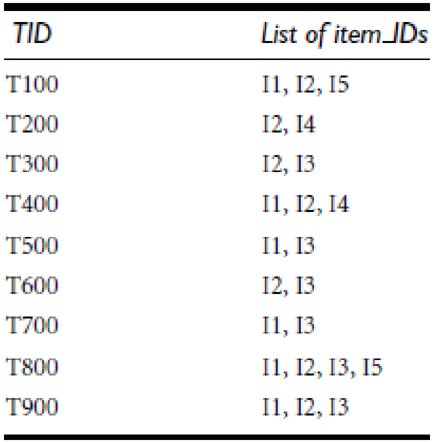
$$confidence(A \Rightarrow B) = P(B|A) = \frac{support\_count(A \cup B)}{support\_count(A)}$$



- 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:
Generating association rules

- frequent itemset *l = {I1, I2, 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

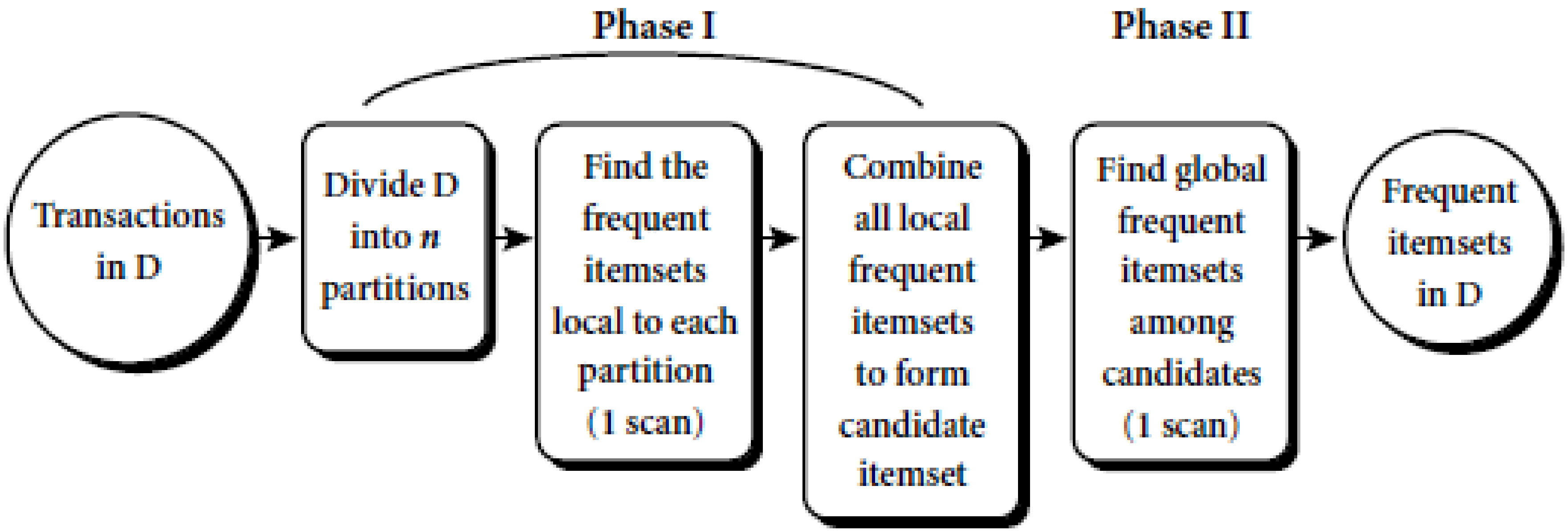
$$\begin{array}{ll}
I1 \wedge I2 \Rightarrow I5, & confidence = 2/4 = 50\% \\
I1 \wedge I5 \Rightarrow I2, & confidence = 2/2 = 100\% \\
I2 \wedge I5 \Rightarrow I1, & confidence = 2/2 = 100\% \\
I1 \Rightarrow I2 \wedge I5, & confidence = 2/6 = 33\% \\
I2 \Rightarrow I1 \wedge I5, & confidence = 2/7 = 29\% \\
I5 \Rightarrow I1 \wedge I2, & confidence = 2/2 = 100\%
\end{array}$$

- 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.

Improving the Efficiency of Apriori

- Hash-based technique
  - Hashing itemsets into corresponding buckets
- Transaction reduction
  - reducing the number of transactions scanned in future iterations
- Partitioning
  - partitioning the data to find candidate itemsets
- Sampling
  - mining on a subset of the given data
- Dynamic itemset counting
  - adding candidate itemsets at different points during a scan

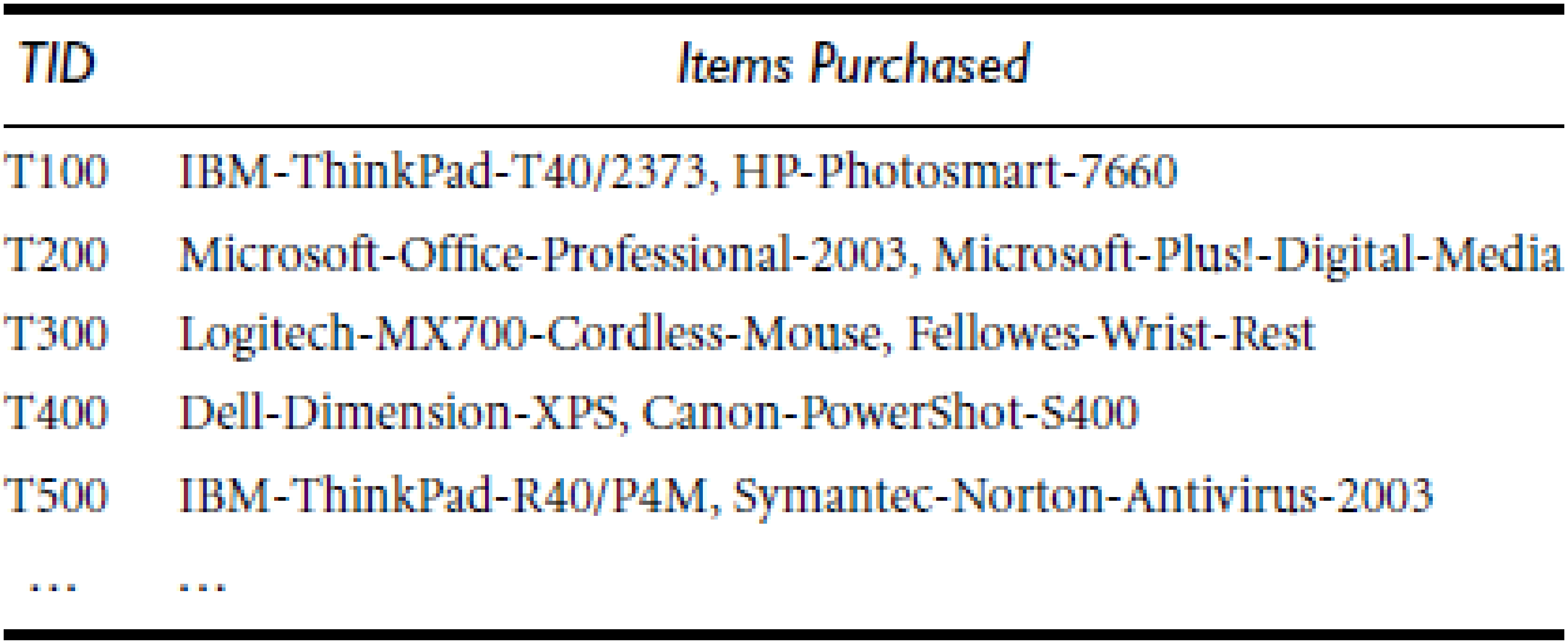
Mining by partitioning the data


Phase I
Phase II
Transactions in D
Divide D into *n* partitions
Find the frequent itemsets local to each partition (1 scan)
Combine all local frequent itemsets to form candidate itemset
Find global frequent itemsets among candidates (1 scan)
Frequent itemsets in D

Mining Various Kinds of Association Rules

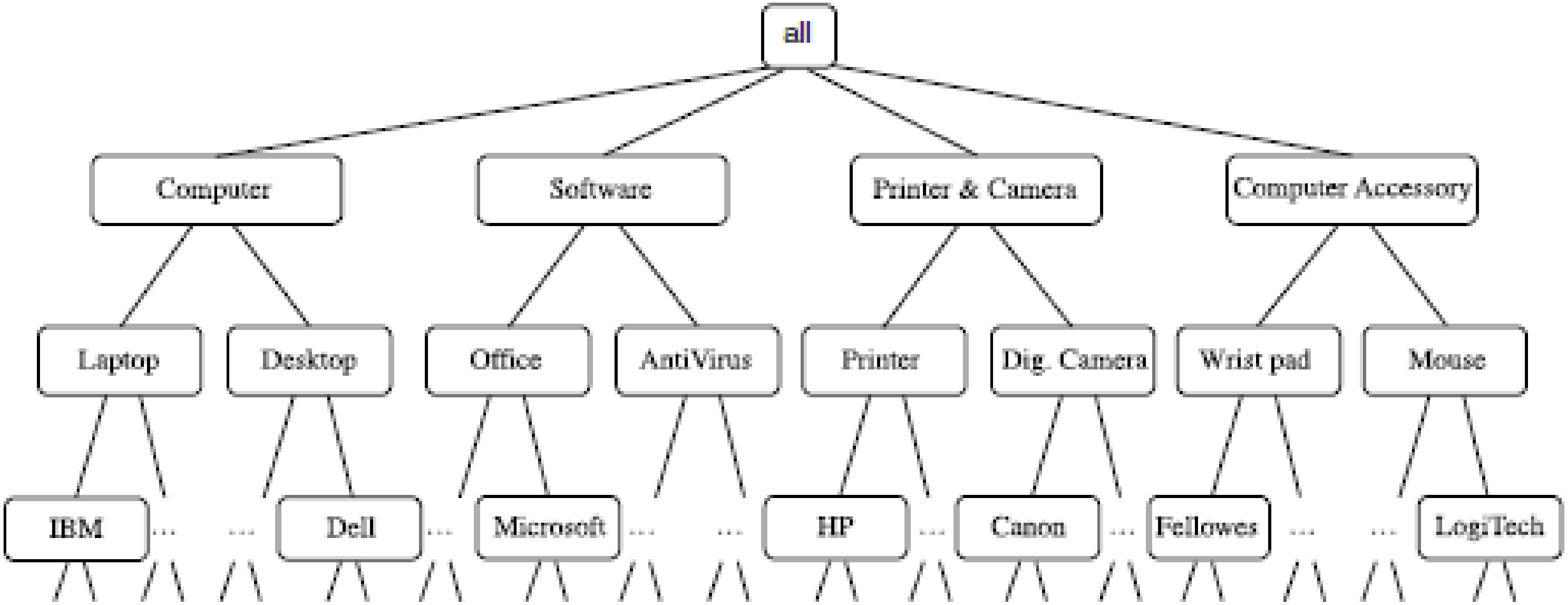
- Mining Multilevel Association Rules

Task-relevant data, *D*

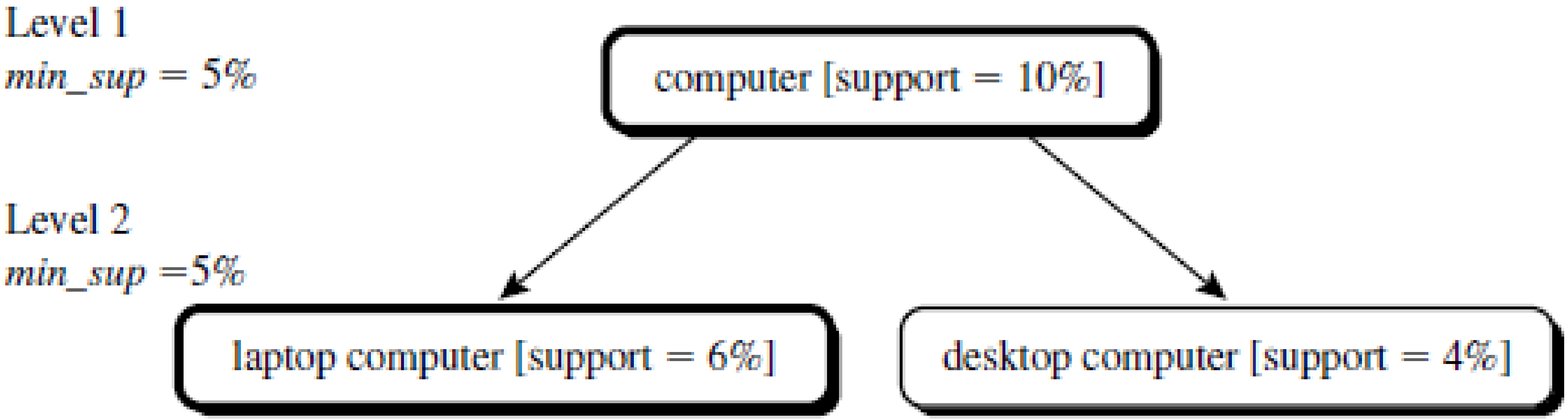


TID	*Items Purchased*
T100	IBM-ThinkPad-T40/2373, HP-Photosmart-7660
T200	Microsoft-Office-Professional-2003, Microsoft-Plus!-Digital-Media
T300	Logitech-MX700-Cordless-Mouse, Fellowes-Wrist-Rest
T400	Dell-Dimension-XPS, Canon-PowerShot-S400
T500	IBM-ThinkPad-R40/P4M, Symantec-Norton-Antivirus-2003
…	…

A concept hierarchy for *AllElectronics computer items*



Multilevel mining with uniform support



Level 1
min_sup = 5%

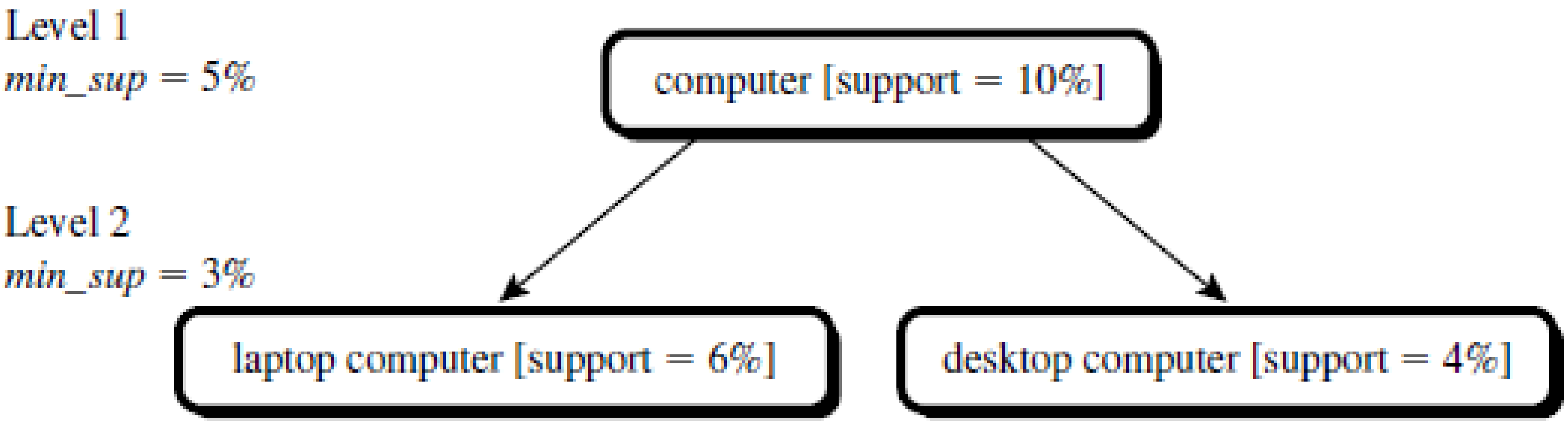
computer [support = 10%]

Level 2
min_sup =5%

laptop computer [support = 6%]

desktop computer [support = 4%]

Multilevel mining with reduced support



$$buys(X, \text{“}laptop\ computer\text{”}) \Rightarrow buys(X, \text{“}HP\ printer\text{”})$$
$$[support = 8\%, confidence = 70\%]$$

$$buys(X, \text{“}IBM\ laptop\ computer\text{”}) \Rightarrow buys(X, \text{“}HP\ printer\text{”})$$
$$[support = 2\%, confidence = 72\%]$$

Mining Multidimensional Association Rules from Relational Databases and DataWarehouses

Sigle-dimensional association rules

$$buys(X, \text{“}digital\ camera\text{”}) \Rightarrow buys(X, \text{“}HP\ printer\text{”})$$

Multidimensional association rules

$$age(X, \text{“}20...29\text{”}) \wedge occupation(X, \text{“}student\text{”}) \Rightarrow buys(X, \text{“}laptop\text{”})$$

Hybrid-dimensional association rules

$$age(X, \text{“}20...29\text{”}) \wedge buys(X, \text{“}laptop\text{”}) \Rightarrow buys(X, \text{“}HP\ printer\text{”})$$

Summary

- Association Analysis
- Mining Frequent Patterns
- Association and Correlations
- Apriori Algorithm
- Mining Multilevel Association Rules

References

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