Data Mining
Association Analysis: Basic Concepts and Algorithms

Lecture Notes for Chapter 6

Introduction to Data Mining

by

Tan, Steinbach, Kumar

Why Mining Association Rules?

- Objective:
  - Finding interesting co-occurring items (or objects, events) in a given data set.
- Examples:
  - Given a database of transactions, each transaction is a list of items (purchased by a customer in a visit), you may find:

    computer → financial_management_software
    [support=2%, confidence=60%]
  - From a student database, you may find
    - major(x, "CS") ^ gpa(x, "A") → has_taken(x, "DB") [1%, 75%]

What Kind of Databases?

Transaction database TDB

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

- Itemset: a set of items
- A *transaction* is a tuple (tid, X)
  - Transaction ID tid
  - Itemset X
- A *transaction database* is a set of transactions
  - In many cases, a transaction database can be treated as a set of itemsets (ignore TIDs)
- Association rule from TDB (relates two itemsets):
  - {a, c, m} → {l} [support=40%, confidence=66.7%]

Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

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

Example of Association Rules

{Diaper} → {Beer},
{Milk, Bread} → {Eggs,Coke},
{Beer, Bread} → {Milk},

Implication means co-occurrence, not causality!

Why Mining Association Rules? (Cont'd)

Popular application: *Basket data analysis*

- place items frequently bought together close to each other to increase sales of these items.
- *?* → *a particular product* (What the store should do to boost sales of the particular product)
- *Home Electronics* → *?* (What other products should the store stock up?)

Definition: Association Rule

- **Association Rule**
  - An implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
  - Example: {Milk, Diaper} → {Beer}

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

- **Rule Evaluation Metrics**
  - Support (s)
    - Fraction of transactions that contain both X and Y
  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Mining Association Rules

- Problem statement

  Given a *minimum support* (*min_sup*), also called *support threshold*, and a *minimum confidence* (*min_conf*), also called *confidence threshold*, find all association rules that satisfy both *min_sup* and *min_conf* from a data set *D*.

Definition: Frequent Itemset

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - E.g. σ({Milk, Bread,Diaper}) = 2
- **Support**
  - Fraction of transactions that contain an itemset
  - E.g. s({Milk, Bread, Diaper}) = 2/5
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a *minsup* threshold

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

Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds

  ⇒ Computationally prohibitive!

Mining Association Rules

- Two-step approach:
  1. Frequent Itemset Generation
     - Generate all itemsets whose support ≥ minsup
  2. Rule Generation
     - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

- Frequent itemset generation is still computationally expensive

Support and Confidence: Example

$$\text{support}(X \rightarrow Y) = P(X \cup Y)$$

$$\text{confidence}(X \rightarrow Y) = P(Y|X)$$

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Relative frequency is used to estimate the Probability.

- $A \rightarrow C$
- $C \rightarrow A$
- $A \,\&\, C \rightarrow B$
- $A \,\&\, B \rightarrow E$

Support and Confidence: Example

$\text{support}(A \rightarrow B) = \text{P}(A \cup B)$

$\text{confidence}(A \rightarrow B) = \text{P}(B|A)$

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

- $A \rightarrow C$ (50%, 66.7%)
- $C \rightarrow A$ (50%, 100%)
- $A \,\&\, C \rightarrow B$ (25%, 50%)
- $A \,\&\, B \rightarrow E$ (0%, 0%)

How to Mine Association Rules

- A two-step process:
  - Find all frequent itemsets ---- the key step
  - Generate strong association rules from frequent itemsets.
- Example: given min_sup=50% and min_conf=50%

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Frequent Itemset	Support
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

- Generate strong rules:
  - {A} → {C} [support=50%, confidence=66.6%]
  - {C} → {A} [support=50%, confidence=100%]

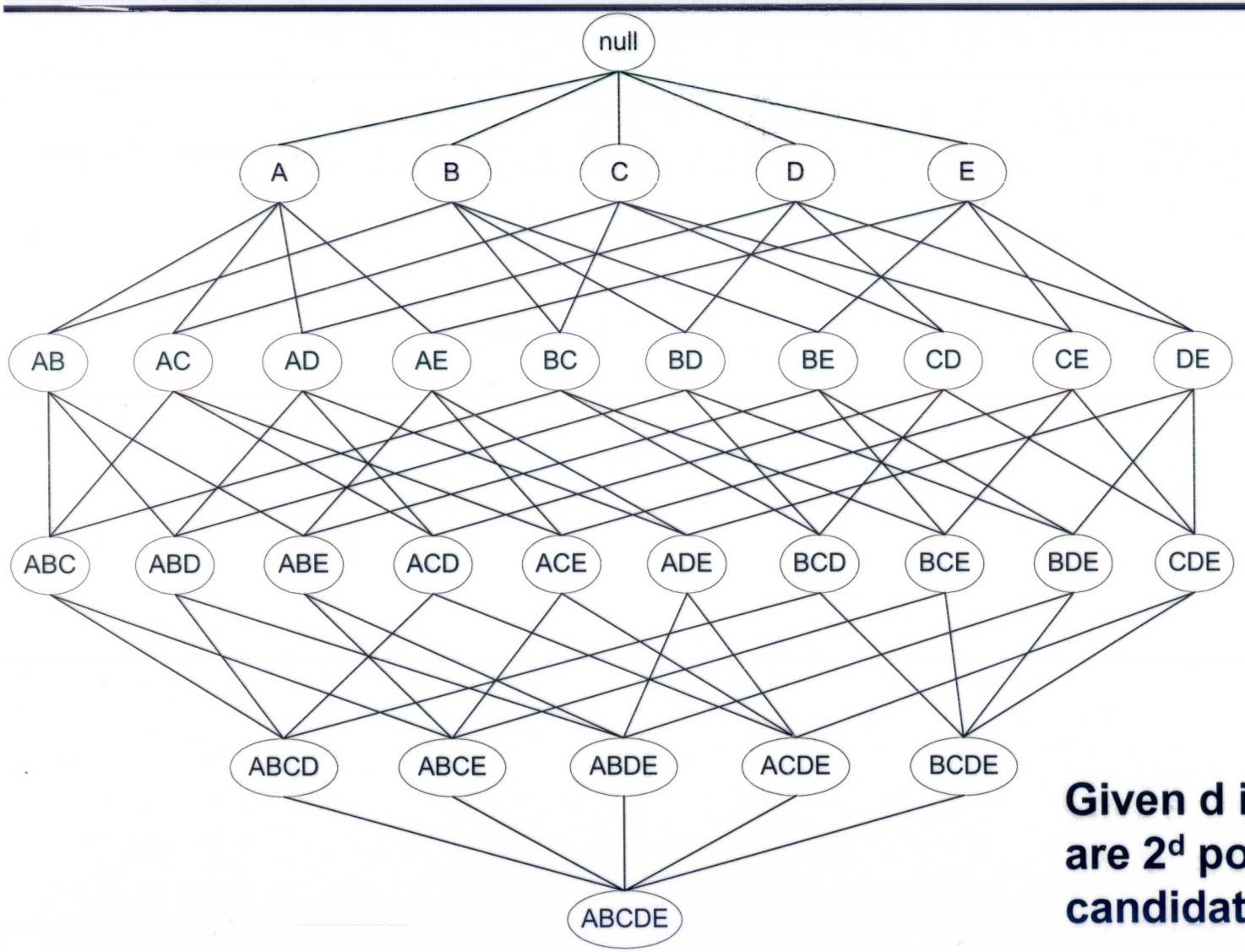
Finding Frequent Itemsets

- Objective: find the itemsets that satisfy the minimum support count.
- Algorithms
  - Apriori
  - FP-Growth
  - H-Mine
  - Partition
  - CLOSET
  - CHARM
  - etc.

Frequent Itemset Generation Strategies

- Reduce the number of candidates (M)
  - Complete search: $M=2^d$
  - Use pruning techniques to reduce M
- Reduce the number of transactions (N)
  - Reduce size of N as the size of itemset increases
  - Used by DHP and vertical-based mining algorithms
- Reduce the number of comparisons (NM)
  - Use efficient data structures to store the candidates or transactions
  - No need to match every candidate against every transaction

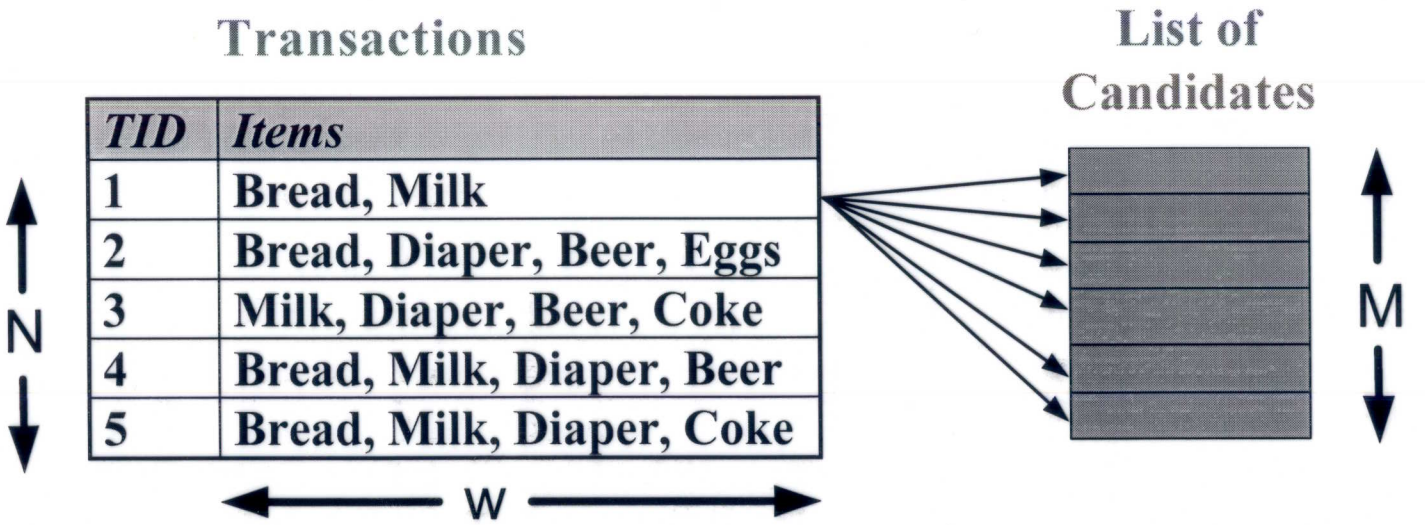
Frequent Itemset Generation



Given d items, there are 2^d possible candidate itemsets

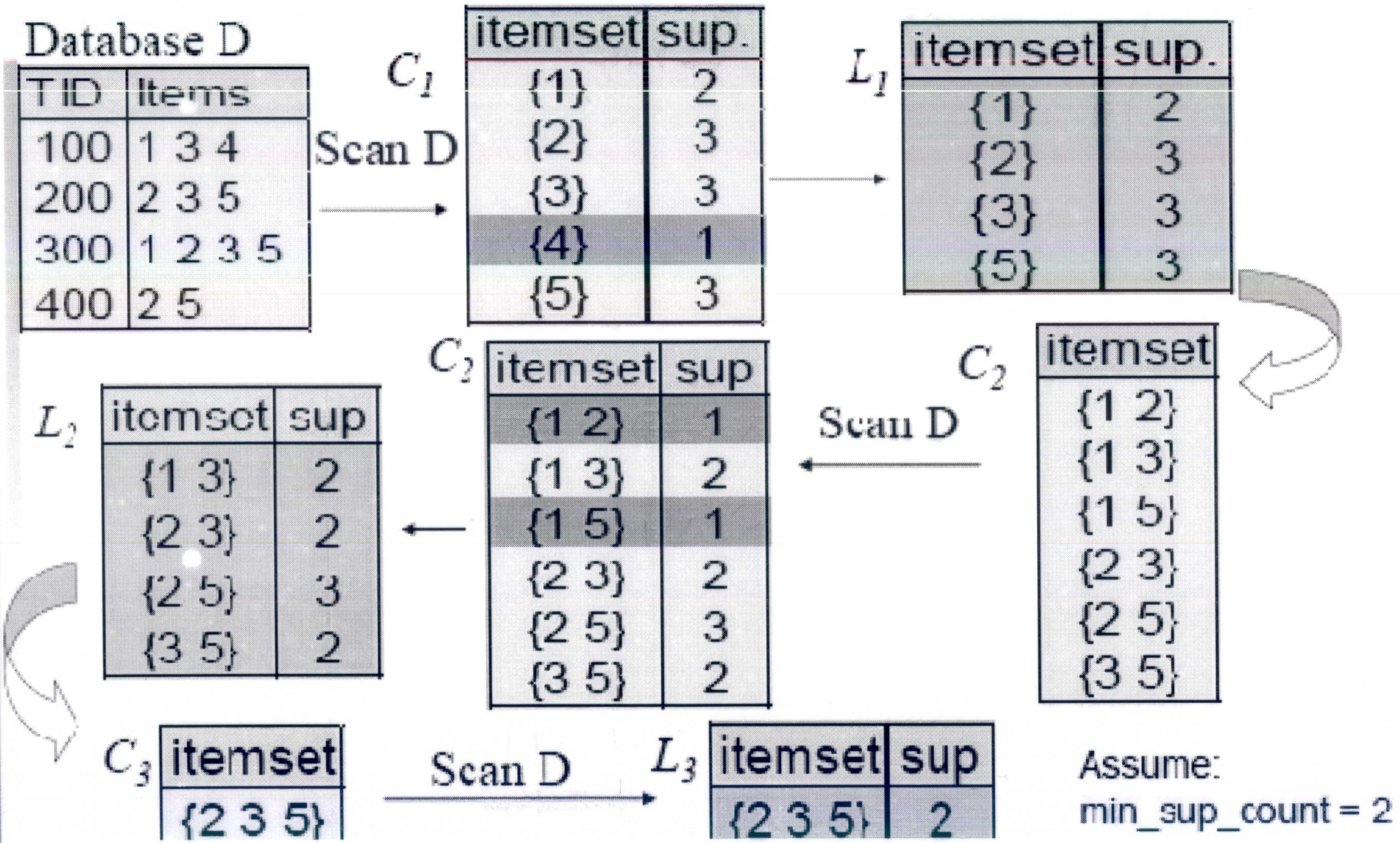
Frequent Itemset Generation

- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database



  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => Expensive since M = 2^d !!!

The Apriori Algorithm — Example



Apriori

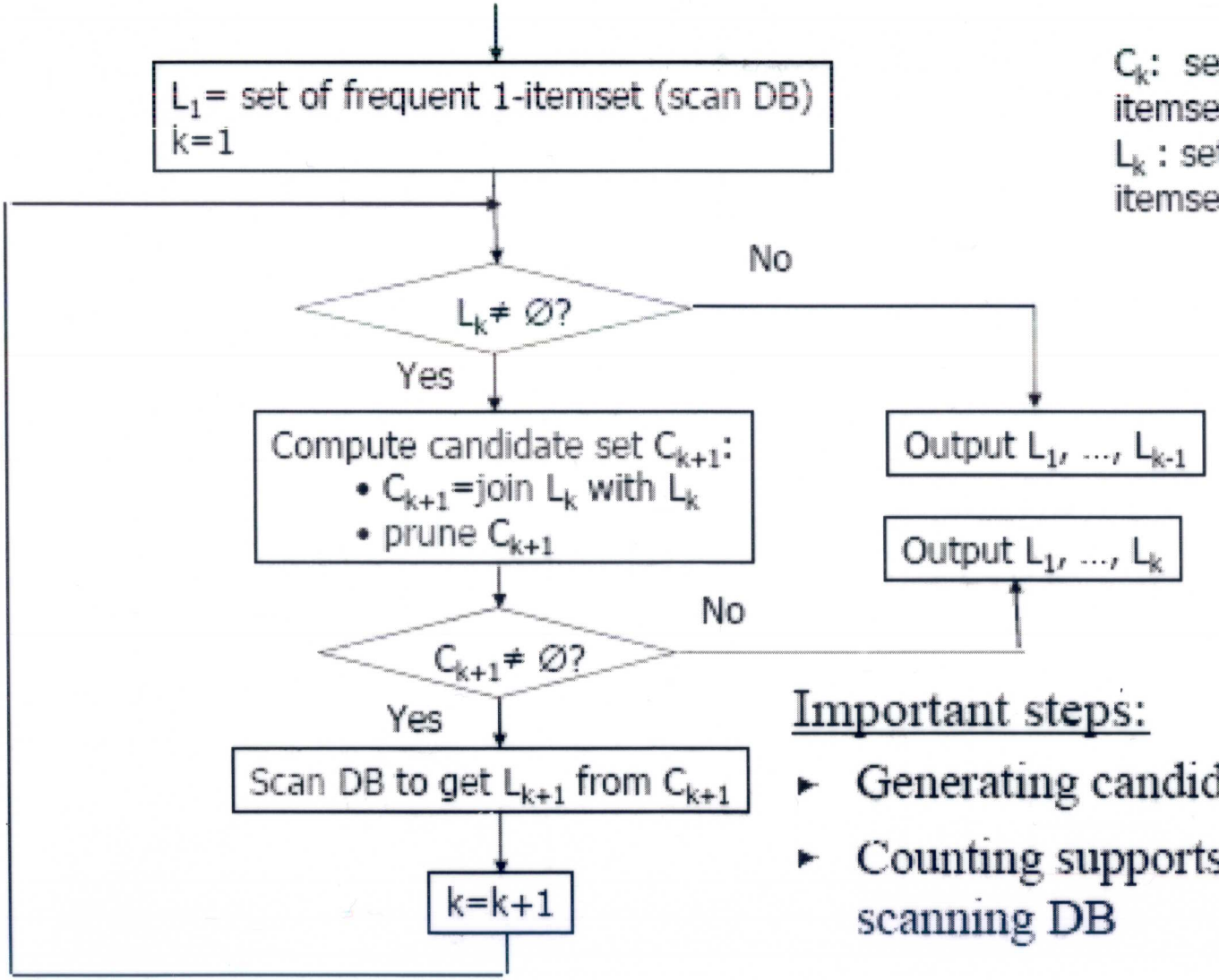
- The Apriori property (an anti-monotone property), also called *downward closure* property:

  Any nonempty subset of a frequent itemset must be frequent

  - i.e., if {*A*,*B*,*C*} is a frequent itemset, {A,B}, {A,C}, {B,C}, {*A*}, {*B*} and {C} must also be frequent.
  - This is because a transaction containing {A, B, C} also contains {A,B}, {B,C}, …

- No superset of any infrequent itemset should be checked.
  - Many item combinations can be pruned from the search space.
- Apriori-based mining (level-wise iterations)
  - Generate length (k+1) candidate itemsets from frequent k-itemsets
  - Test the candidates against DB

Apriori Algorithm (Flow Chart)



C_k: set of candidate k-itemsets

L_k : set of frequent k-itemsets

Important steps:

- Generating candidates
- Counting supports of candidates by scanning DB

The Apriori Algorithm

- Based on the Apriori property, use *iterative level-wise approach* and *candidate generation-and-test*
- Pseudo-code:

C_k: a set of candidate itemsets of size k
L_k : the set of frequent itemsets of size k

L_1 = {frequent items}; (*Scan database to find all frequent 1-itemsets*)
for (k = 1; L_k !=∅; 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} satisfying min_support
  end
return $\cup_k L_{k-1}$;

Scan database to calculate support for each itemset in C_{k+1}

Level-wise Generation process: $L_k \rightarrow C_{k+1} \rightarrow L_{k+1}$

(20)

How to Generate Candidates?
(i.e., How to Generate C_{k+1} from L_k)

- Given L_k = the set of frequent k-itemsets
- List the items in each itemset of L_k in an order

L_3 =

{1 2 3}
{1 2 4}
{1 3 4}
{1 3 5}
{2 3 4}

- Given L_k, generate C_{k+1} in two steps:
  - Join Step: Join L_k with L_k by joining two k-itemsets in L_k. Two k-itemsets are joinable if their first (k-1) items are the same ~~and the last item in the first itemset is smaller than the last item in the second itemset~~ (the condition for joining two members of L_k).

    Now, C_4={{1 2 3 4}, {1 3 4 5}}
  - Prune Step: Delete all candidates in C_{k+1} that have a non-frequent subset by checking all length-k subsets of a candidate
    - Now, C_4={{1 2 3 4}}

(20

Example of Candidate-generation

- L_4={*abcd, abcg, abdg, abef, abeh, acdg, bcdg*}
- Self-joining: L_4*L_4
  - *abcdg* from *abcd* and *abcg*
  - *abefh* from *abef* and *aceh*
- Pruning:
  - *abefh* is removed because *abfh or aefh or befh* is not in L_4
- C_5={*abcdg*}

Generate Association Rules from Frequent Itemsets

- Naïve algorithm:

```
for each frequent itemset l do
    for each nonempty proper subset s of l do
        if (support(l)/support(s) >= min_conf)
            output the rule s → l-s , with
                    support =support(l) and
                    confidence=support(l)/support(s)
```

Generate Association Rules from Frequent Itemsets

- Example:
  - Given a frequent itemset: l={I1, I2, I5}
  - nonempty subsets of l are {I1,I2}, {I1,I5}, {I2,I5}, {I1}, {I2}, {I5}
  - resulting association rules:

$$I1 \wedge I2 \rightarrow I5, \quad confidence = 50\%$$
$$I1 \wedge I5 \rightarrow I2, \quad confidence = 100\%$$
$$I2 \wedge I5 \rightarrow I1, \quad confidence = 100\%$$
$$I1 \rightarrow I2 \wedge I5, \quad confidence = 33\%$$
$$I2 \rightarrow I1 \wedge I5, \quad confidence = 29\%$$
$$I5 \rightarrow I1 \wedge I2, \quad confidence = 100\%$$

If minimum confidence threshold is 70%, only 3 rules are output.

Is Apriori Fast Enough? — Performance Bottlenecks

- The core of the Apriori algorithm:
  - Use frequent *k*-itemsets to generate candidate *(k+1)*-itemsets
  - Use database scan and pattern matching to collect counts for the candidate itemsets to generate frequent (k+1)-itemsets from k+1 candidate set
- The bottleneck of *Apriori*: candidate generation and test
  - Huge candidate sets:
    - 10^4 frequent 1-itemsets will generate more than 10^7 candidate 2-itemsets
    - To discover a frequent pattern of size 100, e.g., $\{a_1, a_2, \ldots, a_{100}\}$, one needs to generate $2^{100} \approx 10^{30}$ candidates.
  - Multiple scans of database:
    - Needs *n* or *n+1* scans, *n* is the length of the longest frequent pattern

Q7: For the following TDB, find all assoc. rules with 50 %. support and 75 %. Confidence.

TID	Items
100	Bread, cheese, Eggs, Juice
200	Bread, cheese, Juice
300	Bread, Milk, Yogurt
400	Bread, Juice, Milk
500	cheese, Juice, Milk

Ans: Min-support count = 3

C_1

Item	sup
Bread	4
cheese	3
Juice	4
Milk	3
× Eggs	1
× Yogurt	1

$\Rightarrow$ L_1

Item	sup
Bread	4
cheese	3
Juice	4
Milk	3

C_2 :

9temsets	sup. count	
(Bread, Cheese)	2	X
(Bread, Juice)	3	
(Bread, Milk)	2	X
(Cheese, Juice)	3	
(Cheese, Milk)	1	X
(Juice, Milk)	2	X

L_2 :

9temset	Sup.
⟹ (Bread, Juice)	3
(Cheese, Juice)	3

No C_3.

Rule Generation :-

		Conf.
L_2 : (Bread, Juice)	Bread → Juice	$\frac{3}{4} = 75\%$
	Juice → Bread	$\frac{3}{4} = 75\%$
(Cheese, Juice)	cheese → Juice	$3/3 = 100\%$
	Juice → cheese	$3/4 = 75\%$

∵ All have minimum 75% Confidence, all qualify.

Class Association Rules

- T --- Transaction data set with n transactions
- I --- set of all items in T
- Y --- set of all class labels
- A CAR is

$$X \rightarrow y, \text{ where } X \subseteq I, \text{ and } y \in Y.$$