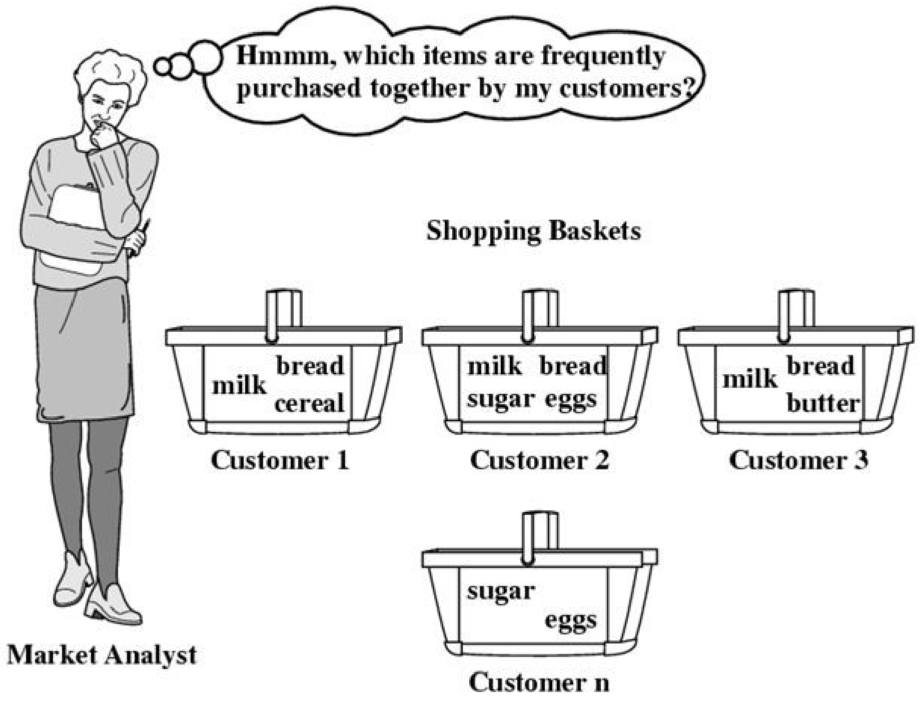
# What Is Association Mining?

- Association Rule Mining
  - Finding frequent patterns, associations, correlations, or causal structures among item sets in transaction databases, relational databases, and other information repositories
- Applications
  - Market basket analysis (marketing strategy: items to put on sale at reduced prices), cross-marketing, catalog design, shelf space layout design, etc
- Examples
  - Rule form: Body $\rightarrow$ Head [Support, Confidence].
  - buys(x, “Computer”) $\rightarrow$ buys(x, “Software”) [2%, 60%]
  - major(x, “CS”) ^ takes(x, "DB”) $\rightarrow$ grade(x, “A”) [1%, 75%]

# Market Basket Analysis



Typically, association rules are considered interesting if they satisfy both a minimum support threshold and a minimum confidence threshold.

## Assocaition Rule Mining

- Data Mining Task
- Find frequent itemset
- Find interesting association or correlation relationship between a large set of data item.
- Rule based association learning method

# Association Rule Mining

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

## Market-Basket transactions

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

**Example of Association Rules**

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

Implication means co-occurrence, not causality!

# Definition: Frequent Itemset

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count (📅)**
  - Frequency of occurrence of an itemset
  - E.g. 📅({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** |

## Definition: Association Rule

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

- **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

| *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:

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

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

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

# 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

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

### Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

### Observations:

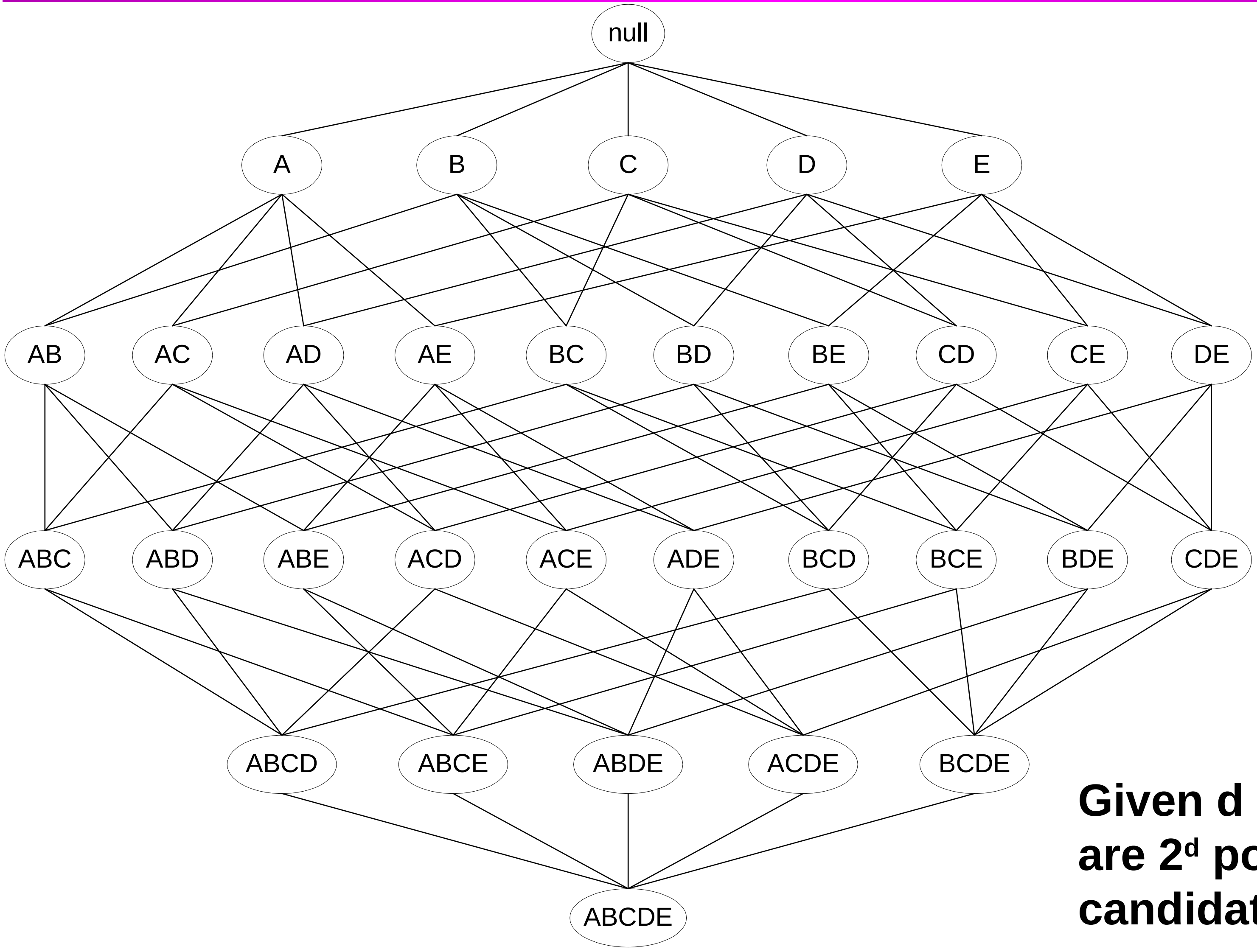
- ✂ All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Beer}
- ✂ Rules originating from the same itemset have identical support
  but
  can have different confidence
- ✂ Thus, we may decouple the support and confidence requirements

## Mining Association Rules

Find interesting association or correlation relationship between a large set of data item

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

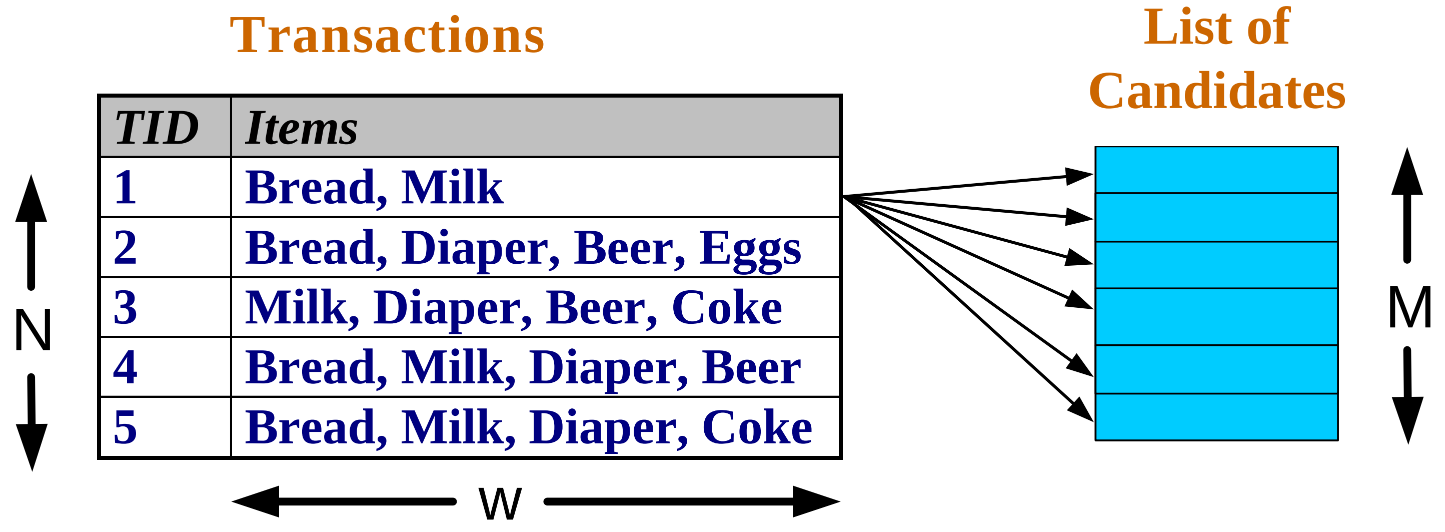
# Frequent Itemset Generation



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

## Frequent Itemset Generation

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



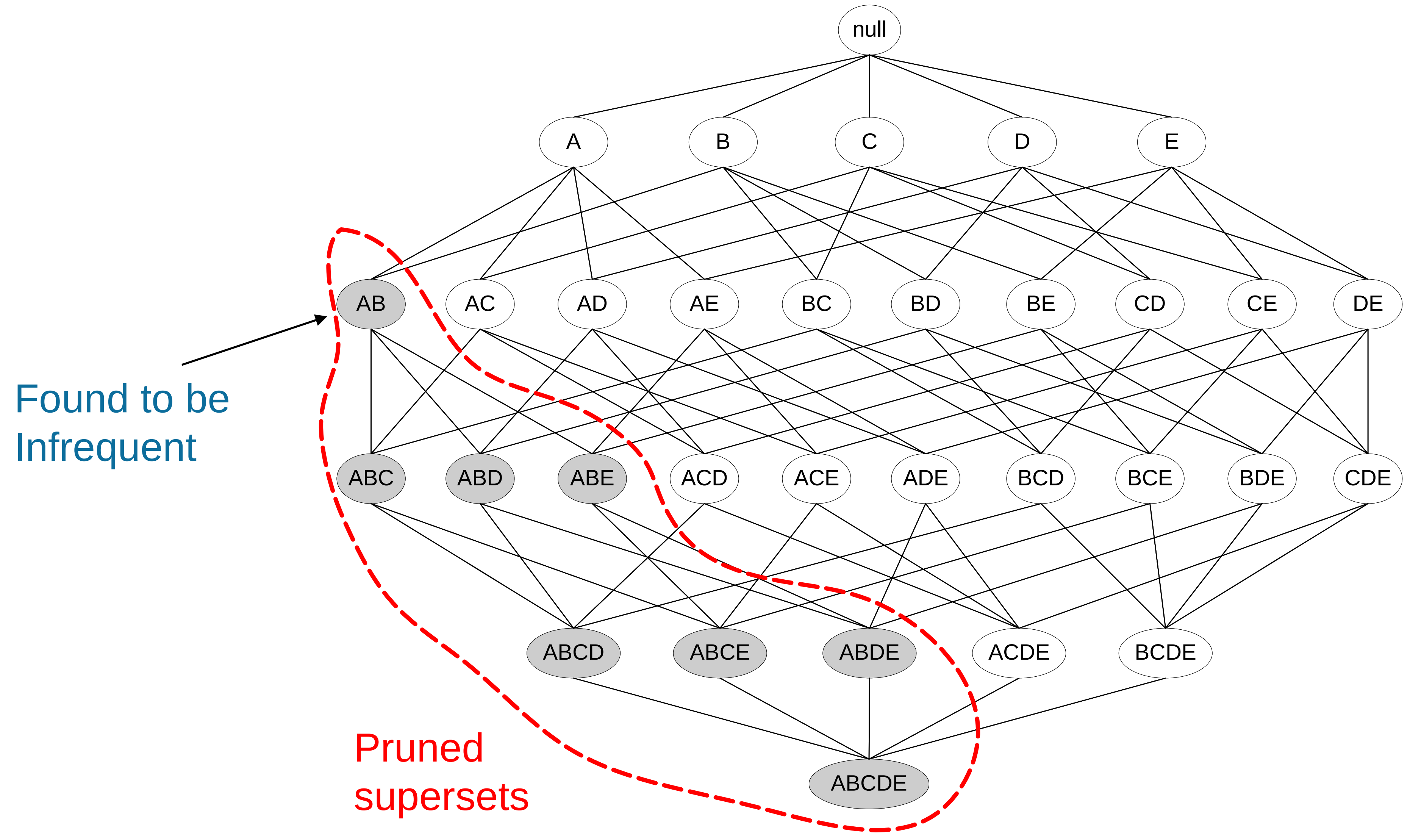
| *TID* | *Items* |
|---|---|
| 1 | Bread, Milk |
| 2 | Bread, Diaper, Beer, Eggs |
| 3 | Milk, Diaper, Beer, Coke |
| 4 | Bread, Milk, Diaper, Beer |
| 5 | Bread, Milk, Diaper, Coke |

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

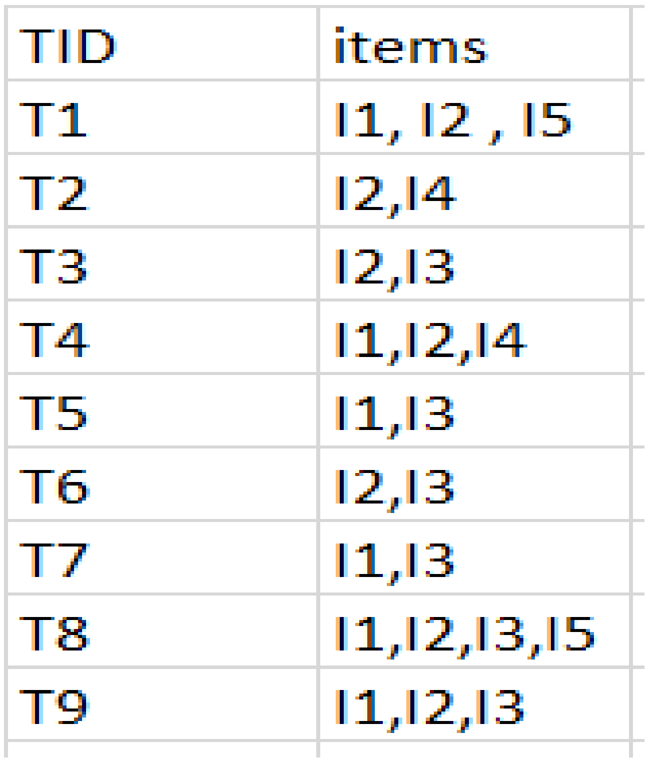
## Apriori Algorithm

- Algorithm for mining frequent item sets
- Uses prior knowledge of frequent itemset properties
- Iterative approach known as level-wise search
- K-itemset are used to explore (k+1)itemsets
- First,the set of frequent 1-itemsets is found and collect those items that support minimum support. This resulting set is denoted by L1.
- Next L1 is used to find L2 ,the set of frequent 2-itemsets,which is used to finf L3 and so on, until no more k-iremset can be found.

# Illustrating Apriori Principle



## Illustrating Apriori Principle

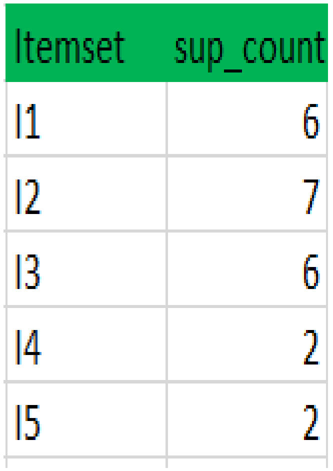


| TID | items |
|---|---|
| T1 | I1, I2 , I5 |
| T2 | I2,I4 |
| T3 | I2,I3 |
| T4 | I1,I2,I4 |
| T5 | I1,I3 |
| T6 | I2,I3 |
| T7 | I1,I3 |
| T8 | I1,I2,I3,I5 |
| T9 | I1,I2,I3 |

# Step1..

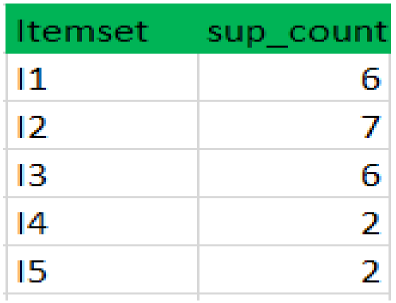
K=1

(I) Create a table containing support count of each item present in dataset – Called **C1(candidate set)**



| Itemset | sup_count |
|---|---|
| I1 | 6 |
| I2 | 7 |
| I3 | 6 |
| I4 | 2 |
| I5 | 2 |

(II) compare candidate set item's support count with minimum support count(here min_support=2
 if support_count of candidate set items is less than min_support then remove those items).
This gives us **itemset L1.**

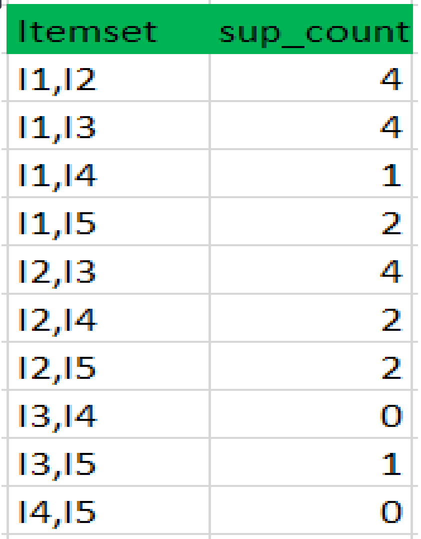


| Itemset | sup_count |
|---|---|
| I1 | 6 |
| I2 | 7 |
| I3 | 6 |
| I4 | 2 |
| I5 | 2 |

# Step2

K=2

- Generate candidate set C2 using L1 (this is called join step). Condition of joining $L_{k-1}$ and $L_{k-1}$ is that it should have (K-2) elements in common.
- Check all subsets of an itemset are frequent or not and if not frequent remove that itemset.(Example subset of{I1, I2} are {I1}, {I2} they are frequent. Check for each itemset)
- Now find support count of these itemsets by searching in dataset

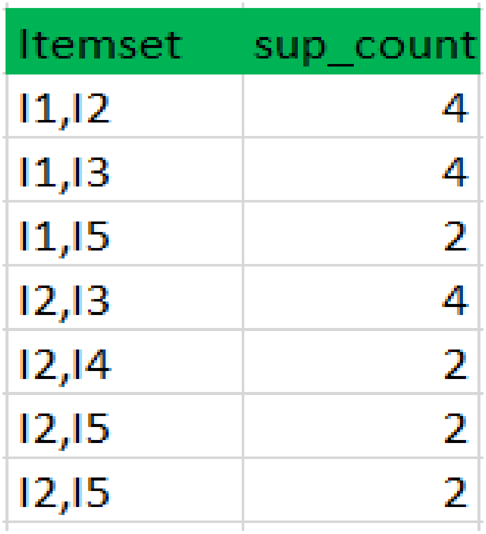


| Itemset | sup_count |
|---|---|
| I1,I2 | 4 |
| I1,I3 | 4 |
| I1,I4 | 1 |
| I1,I5 | 2 |
| I2,I3 | 4 |
| I2,I4 | 2 |
| I2,I5 | 2 |
| I3,I4 | 0 |
| I3,I5 | 1 |
| I4,I5 | 0 |

## Step2 Contd...

(II) compare candidate (C2) support count with minimum support count(here min_support=2)

if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L2.

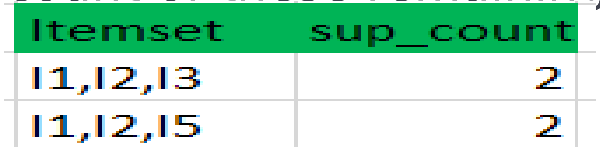


| Itemset | sup_count |
|---|---|
| I1,I2 | 4 |
| I1,I3 | 4 |
| I1,I5 | 2 |
| I2,I3 | 4 |
| I2,I4 | 2 |
| I2,I5 | 2 |
| I2,I5 | 2 |

# Step3

**Step-3:**

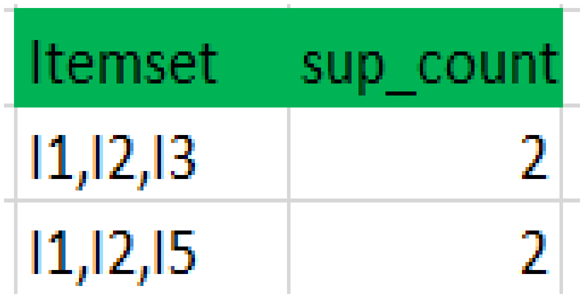
- Generate candidate set C3 using L2 (join step). Condition of joining $L_{k-1}$ and $L_{k-1}$ is that it should have (K-2) elements in common. So here, for L2, first element should match.
- So itemset generated by joining L2 is {I1, I2, I3}{I1, I2, I5}{I1, I3, i5}{I2, I3, I4}{I2, I4, I5}{I2, I3, I5}
- Check if all subsets of these itemsets are frequent or not and if not, then remove that itemset.(Here subset of {I1, I2, I3} are {I1, I2},{I2, I3},{I1, I3} which are frequent. For {I2, I3, I4}, subset {I3, I4} is not frequent so remove it. Similarly check for every itemset)
- find support count of these remaining itemset by searching in dataset.



| Itemset | sup_count |
|---|---|
| I1,I2,I3 | 2 |
| I1,I2,I5 | 2 |

## Step3 Contd..

(II) Compare candidate (C3) support count with minimum support count(here min_support=2 if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L3.



| Itemset | sup_count |
|---|---|
| I1,I2,I3 | 2 |
| I1,I2,I5 | 2 |

## Step 4

**Step-4:**

- Generate candidate set C4 using L3 (join step). Condition of joining $L_{k-1}$ and $L_{k-1}$ (K=4) is that, they should have (K-2) elements in common. So here, for L3, first 2 elements (items) should match.

- Check all subsets of these itemsets are frequent or not (Here itemset formed by joining L3 is {I1, I2, I3, I5} so its subset contains {I1, I3, I5}, which is not frequent). So no itemset in C4

- We stop here because no frequent itemsets are found further

## Apriori Property

## All non empty subsets of a frequent itemset must also be frequent

- If a itemset I doesnot satisfy the minimum support threshold, min sup ,then I is not frequent.

- If an item a is added to a itemset I, the resulting itemset appear more frequent than I

- Antimonotonicity- if a set cannot pass a test, all of its supersets will fall test as well

## Apriori Steps

- **1.** The join step: To find find $L_k$, a set of candidate *k*-itemsets is generated by joining $L_k$ with itself. This set of candidates is denoted $C_k$.

- **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 $C_k$

# Example

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} |

Scan $D$ for count of each candidate →

$C_3$

| Itemset | Sup. count |
|---|---|
| {I1, I2, I3} | 2 |

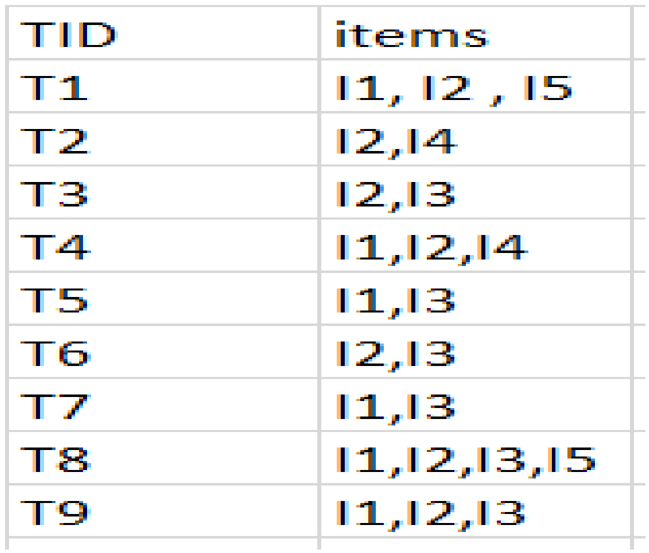
Compare candidate support count with minimum support count →

$L_3$

| Itemset | Sup. cou |
|---|---|
| {I1, I2, I3} | 2 |

## GENERATING ASSOCIATION RULES FROM FREQUENT ITEMSET

- Frequent Itemset X= {I1, I2, I3}
- Non empty subset of X={ I1, I2, I3, {I1,I2}, {I1,I3}, {I2,I3}}
- Confidence(A->B)=Support count(A∪B)/Support_count(A)



| TID | items |
|---|---|
| T1 | I1, I2 , I5 |
| T2 | I2,I4 |
| T3 | I2,I3 |
| T4 | I1,I2,I4 |
| T5 | I1,I3 |
| T6 | I2,I3 |
| T7 | I1,I3 |
| T8 | I1,I2,I3,I5 |
| T9 | I1,I2,I3 |

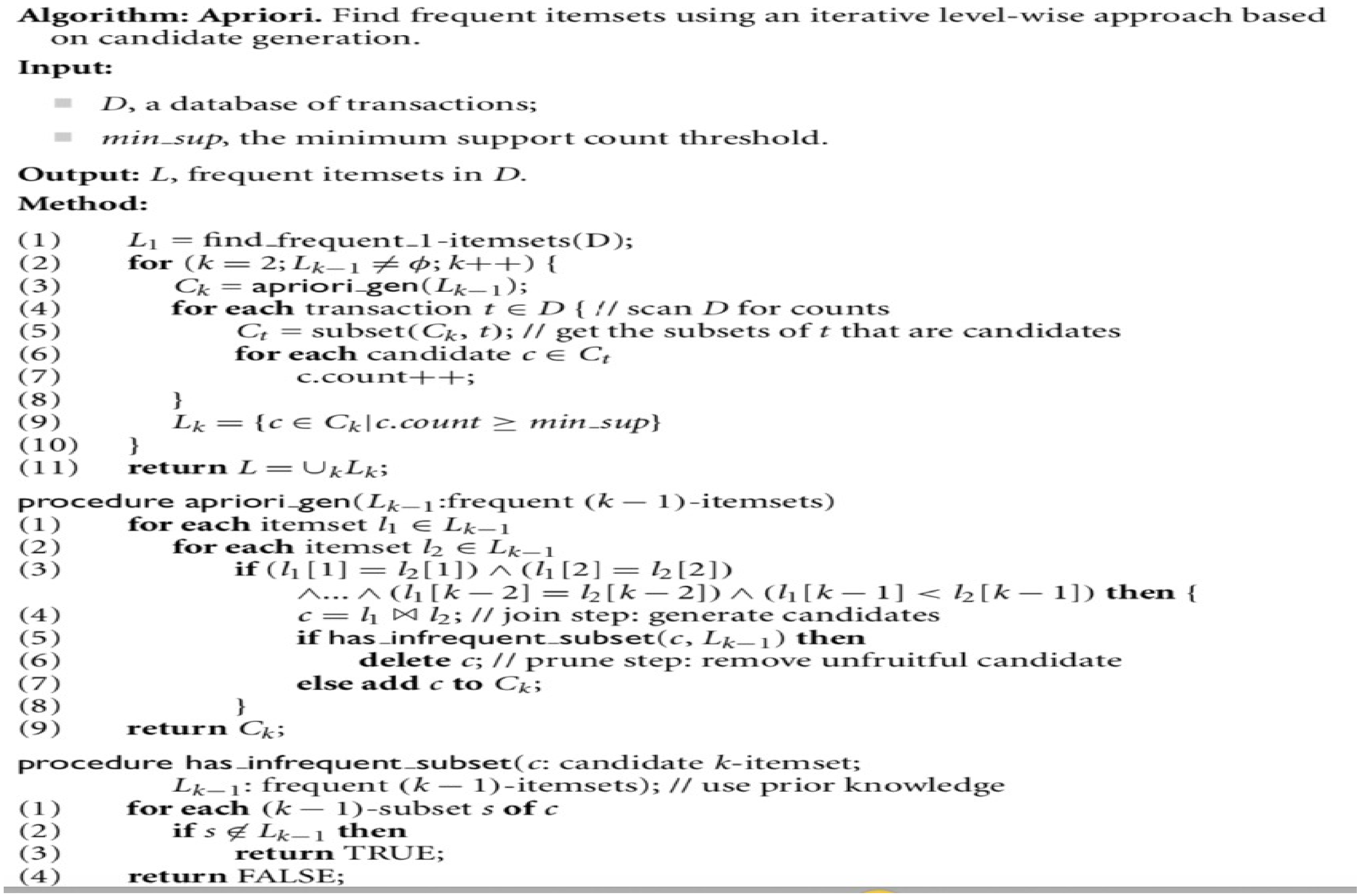
| | |
|---|---|
| I1,I2=>I3 | confidence = 2/4=50% |
| I1,I3=>I2 | confidence= 2/4=50% |
| [I2,I3]=>[I1] | confidence = 2/4=50% |
| [I1]=>[I2, I3] | confidence 2/6 =33% |
| [I2]=>[I1,I3] | confidence = 2/7=28% |
| [I3]=>[I1,I2] | confidence = 2/6 =33% |

So if minimum confidence is **50%**, then first 3 rules can be considered as strong associatio

# Apriori Algorithm

- $F_k$: frequent k-itemsets
- $L_k$: candidate k-itemsets

- Algorithm
  - Let k=1
  - Generate $F_1$ = {frequent 1-itemsets}
  - Repeat until $F_k$ is empty
    - **Candidate Generation**: Generate $L_{k+1}$ from $F_k$
    - **Candidate Pruning**: Prune candidate itemsets in $L_{k+1}$ containing subsets of length k that are infrequent
    - **Support Counting**: Count the support of each candidate in $L_{k+1}$ by scanning the DB
    - **Candidate Elimination**: Eliminate candidates in $L_{k+1}$ that are infrequent, leaving only those that are frequent => $F_{k+1}$



**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:**

```
(1)    $L_1$ = find_frequent_1-itemsets(D);
(2)    for ($k = 2; L_{k-1} \neq \phi; k{+}{+}$) {
(3)        $C_k$ = apriori_gen($L_{k-1}$);
(4)        for each transaction $t \in D$ { // scan $D$ for counts
(5)            $C_t$ = subset($C_k$, $t$); // get the subsets of $t$ that are candidates
(6)            for each candidate $c \in C_t$
(7)                c.count++;
(8)        }
(9)        $L_k = \{c \in C_k | c.count \geq min\_sup\}$
(10)   }
(11)   return $L = \cup_k L_k$;

procedure apriori_gen($L_{k-1}$:frequent $(k-1)$-itemsets)
(1)    for each itemset $l_1 \in L_{k-1}$
(2)        for each itemset $l_2 \in L_{k-1}$
(3)            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 {
(4)                $c = l_1 \bowtie l_2$; // join step: generate candidates
(5)                if has_infrequent_subset($c$, $L_{k-1}$) then
(6)                    delete $c$; // prune step: remove unfruitful candidate
(7)                else add $c$ to $C_k$;
(8)            }
(9)    return $C_k$;

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

## Factors Affecting Complexity of Apriori

- Choice of minimum support threshold
- Dimensionality (number of items) of the data set
- Size of database
- Average transaction width