Unit-2

Market Basket Analysis

Market Basket Analysis (MBA) is a data mining technique used to identify patterns and relationships between items frequently purchased together. It is widely used in retail, e-commerce, and recommendation systems to improve sales strategies, optimize product placement, and enhance customer experience.

Market Basket Analysis is based on the concept of association rule mining, where the goal is to find correlations between different items in a dataset of transactions. The most common application is in supermarkets and online stores, where it helps businesses understand what products are often bought together.

Example Scenario

If many customers who buy bread also purchase butter, a retailer might:

- Place bread and butter close together in the store.
- Offer discounts on butter when customers buy bread.
- Recommend butter to online shoppers who add bread to their cart.

Key Concepts in Market Basket Analysis

a) Transactional Dataset

A dataset where each row represents a transaction, and each transaction contains a set of items bought together.

Transaction ID	Items Bought
1	Milk, Bread, Butter
2	Milk, Bread
3	Bread, Butter, Jam
4	Milk, Butter

b) Association Rule Mining

Association rules are used to find relationships between items. A rule is written as:

$$A \Rightarrow B$$

Metrics in Market Basket Analysis

To measure the strength of associations, we use the following metrics:

1. **Support**
    - Measures how frequently an itemset appears in transactions.

- Formula:

  Support(A⇒B)=Transactions containing both A and B/ Total Transactions

2. **Confidence**
   - Measures how often **B** is bought when **A** is bought.
   - Formula:

     Confidence(A⇒B)=Support(A and B)/Support(A)

3. **Lift**
   - Measures how much more likely **B** is purchased when **A** is purchased, compared to its independent probability.
   - Formula:

     Lift(A⇒B)=Confidence(A → B)/Support(B)

   - **Lift > 1**: A and B are positively correlated.
   - **Lift = 1**: No correlation.
   - **Lift < 1**: A and B are negatively correlated.

Algorithms for Market Basket Analysis

Several data mining algorithms can be used for Market Basket Analysis:

a) Apriori Algorithm

- One of the most popular algorithms.
- Uses a "bottom-up" approach, generating frequent itemsets and rules by iteratively expanding itemsets.
- Prunes infrequent itemsets to improve efficiency.

Steps:

1. Identify **frequent individual items**.
2. Generate **item pairs** and check if they are frequent.
3. Extend to **triples, quadruples**, etc.
4. Stop when no more frequent itemsets are found.

The **downside** of the Apriori algorithm is that it requires **multiple scans** of the dataset, which makes it slow for large datasets.

b) FP-Growth Algorithm (Frequent Pattern Growth)

- More efficient than Apriori for large datasets.
- Uses a tree structure (FP-Tree) to mine frequent itemsets without generating candidate sets.

Applications of Market Basket Analysis

1. **Retail & Supermarkets**
    - Optimize product placement (e.g., placing chips near soft drinks).
    - Design combo offers (e.g., burger + fries + drink deals).
2. **E-commerce & Recommendation Systems**
    - Amazon, eBay, and Netflix use MBA to recommend products based on customer purchases.
    - Example: "Customers who bought this also bought..."
3. **Healthcare**
    - Analyze prescription patterns to recommend medications.
    - Detect potential drug interactions.
4. **Finance & Banking**
    - Identify fraudulent transactions based on unusual purchase patterns.
5. **Telecommunications**
    - Bundle mobile services based on customer preferences.

Apriori Algorithm

The **Apriori algorithm** is one of the most widely used data mining techniques for **association rule mining**. It is used to identify frequent itemsets in a transactional dataset and generate rules that describe relationships between items. The algorithm is commonly applied in **Market Basket Analysis**, where businesses analyze purchasing behavior to optimize sales and marketing strategies.

The Apriori algorithm is a **frequent pattern mining** technique that helps discover associations between items in large datasets. It follows the **"apriori property"**, which states:

"If an itemset is frequent, then all its subsets must also be frequent."

This property allows the algorithm to prune unpromising itemsets and improve efficiency.

Example Use Case

Consider a retail store analyzing customer purchases:

- Many customers who buy **Bread** also buy **Butter**.
- The store can use this insight to:
    - **Place Bread and Butter together** to increase sales.
    - **Offer discounts** on Butter when Bread is purchased.
    - **Recommend complementary products** online.

Steps in the Apriori Algorithm

The Apriori algorithm follows an **iterative** approach to find frequent itemsets and generate association rules.

Step 1: Set Minimum Support & Confidence

- Users define a **minimum support** and **minimum confidence** threshold.

- Example: Support ≥ **30%**, Confidence ≥ **60%**.

Step 2: Generate Frequent 1-itemsets

- Identify **individual items** that appear frequently (based on support).
- Example: {Milk, Bread, Butter}.

Step 3: Generate Frequent 2-itemsets

- Combine frequent 1-itemsets to form **2-itemsets**.
- Example: {Milk, Bread}, {Bread, Butter}, {Milk, Butter}.

Step 4: Generate Frequent 3-itemsets

- Combine **frequent 2-itemsets** to form **3-itemsets**.
- Example: {Milk, Bread, Butter}.

Step 5: Prune Infrequent Itemsets

- Any itemset that does not meet the **minimum support** is discarded.

Step 6: Generate Association Rules

- Calculate **Confidence & Lift** for remaining itemsets.
- Generate rules like:
  - {Milk} → {Bread}
  - {Bread} → {Butter}

Problem Statement

Given the following transactional dataset, determine **frequent itemsets** and generate **association rules** using the **Apriori Algorithm** with a **minimum support of 50%** and **minimum confidence of 60%**.

Transaction ID	Items Bought
1	Milk, Bread, Butter
2	Milk, Bread
3	Bread, Butter
4	Milk, Butter
5	Bread, Jam

Step 1: Calculate Support for 1-Itemsets

Support Formula:

Support(A)=Transactions containing A/ Total Transactions

Finding 1-Itemsets

Item	Transactions Containing Item	Support Calculation	Support Value
Milk	{1, 2, 4}	35\frac{3}{5}53	60%
Bread	{1, 2, 3, 5}	45\frac{4}{5}54	80%
Butter	{1, 3, 4}	35\frac{3}{5}53	60%
Jam	{5}	15\frac{1}{5}51	20%

Frequent 1-itemsets (Support ≥ 50%):

{Milk},{Bread},{Butter}

Jam is eliminated since its support is **below 50%**.

Step 2: Generate 2-Itemsets and Compute Support

Now, we generate **pairwise combinations** of frequent 1-itemsets.

Itemset	Transactions Containing Itemset	Support Calculation	Support Value
{Milk, Bread}	{1, 2}	25\frac{2}{5}52	40%
{Milk, Butter}	{1, 4}	25\frac{2}{5}52	40%
{Bread, Butter}	{1, 3}	25\frac{2}{5}52	40%

None of the 2-itemsets meet the **50% support threshold**, so no further itemsets are generated.

Step 3: Generate Association Rules

Since no **2-itemset** has support ≥ **50%**, we **cannot generate any valid association rules**.

Modified Example with Lower Support

If we lower the **minimum support threshold to 40%**, then we consider:

- **Frequent 2-itemsets**: {Milk, Bread}, {Milk, Butter}, {Bread, Butter}

Generating Association Rules

Confidence Formula:

$$\text{Confidence}(A \Rightarrow B) = \text{Support}(A,B) / \text{Support}(A)$$

Rule	Support(A, B)	Support(A)	Confidence Calculation	Confidence Value
{Milk} → {Bread}	40%	60%	4060\frac{40}{60}6040	66.7% ✅
{Milk} → {Butter}	40%	60%	4060\frac{40}{60}6040	66.7% ✅
{Bread} → {Butter}	40%	80%	4080\frac{40}{80}8040	50% ❌

Only **{Milk} → {Bread}** and **{Milk} → {Butter}** meet the **minimum confidence of 60%**.

Final Output

Frequent Itemsets

- **1-itemsets**: {Milk}, {Bread}, {Butter}
- **2-itemsets**: {Milk, Bread}, {Milk, Butter}, {Bread, Butter} (for 40% support)

Association Rules

- {Milk} → {Bread} (Confidence: **66.7%** ✅)
- {Milk} → {Butter} (Confidence: **66.7%** ✅)

Apriori Algorithm (Pruning)

The pruning step in the apriori algorithm is based on the concept that a subset of a frequent itemset must also be a frequent itemset. In other word**s, if we have an itemset having a subset that is not a frequent itemset, the itemset cannot be a frequent itemset.**

We use pruning to minimize the time taken in executing the apriori algorithm. After creating itemsets of K items, we use the following steps to prune the candidate set.

Apriori Algorithm Numerical example

By now, we have discussed the definition and procedure for the apriori algorithm. Let us now discuss a numerical example using the apriori algorithm to understand it in a better manner. For the numerical example, we will use the following dataset.

Transaction ID	Items
T1	I1, I3, I4
T2	I2, I3, I5, I6
T3	I1, I2, I3, I5
T4	I2, I5

T5	I1, I3, I5

Create Frequent Itemsets With One Item

The Apriori algorithm starts by creating candidate itemsets with one item. For this, let us calculate the support count of each item.

Itemset	Support Count
{I1}	3
{I2}	3
{I3}	4
{I4}	1
{I5}	4
{I6}	1

Candidate Itemsets with one item

The above table contains the support count of candidate itemsets with one item. Here, you can observe that the itemsets {I4} and {I6} have support count 1 which is less than the minimum support count 2. Hence, we will omit these itemsets from the above table. After this, we will get the table containing frequent itemsets with a single item as shown below.

Itemset	Support Count
{I1}	3
{I2}	3
{I3}	4
{I5}	4

Now that we have created frequent itemsets containing a single item, we will move to calculate the frequent itemsets with two items.

Create Frequent Itemsets With Two Items

To create frequent itemsets with two items, we will first create the candidate itemset with two items. For this, we will join all the frequent itemsets with one item with each other. After joining, we will get the following itemsets.

{I1,I2}, {I1,I3}, {I1,I5},{I2,I3},{I2,I5}, and {I3,I5}

After creating the itemsets with two items, we need to prune the itemsets having subsets that are not frequent itemsets. As the {I1}, {I2}, {I3}, and {I5} all are frequent itemsets, no itemsets will be removed from the above list while pruning.

As the next step, we will calculate the support count of each itemset having two items to create the candidate itemset. The result is tabulated below.

Itemset	Support Count
{I1,I2}	1
{I1,I3}	3
{I1,I5}	2
{I2,I3}	2
{I2,I5}	3
{I3,I5}	3

In the above candidate itemset, you can observe that the itemset {I1, I2} has the support count 1 which is less than the minimum support count. Hence, we will remove the above itemset from the table and obtain the table containing frequent itemsets with two items as shown below.

Itemset	Support Count
{I1,I3}	3
{I1,I5}	2
{I2,I3}	2
{I2,I5}	3
{I3,I5}	3

Calculate Frequent Itemsets With Three Items

To calculate the frequent itemsets with three items, we first need to calculate the candidate set. For this, let us first join the frequent itemsets with two items and create itemsets with three items as shown below.

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

On the above itemsets, we will perform pruning to remove any itemset that has a subset that is not a frequent itemset. For this, we will create subsets of 2 items for each itemset and check if they are frequent itemsets or not. All the subsets of the above itemsets are tabulated below

Itemset	Subsets	All the subsets are frequent itemsets?
{I1, I3, I5}	{I1, I3},{I1, I5},{I3, I5}	Yes
{I1, I2, I3}	{I1, I2}, {I1, I3}, {I2, I3}	No
{I1, I2, I5}	{I1, I2},{I1, I5},{I2, I5}	No
{I2, I3, I5}	{I2, I3}, {I2, I5}, {I3, I5}	Yes

In the table, you can observe that the itemset {I1,I2 , I3} and {I1, I2, I5} contain the itemset {I1, I2} which is not a frequent itemset. Hence, we will prune the itemsets {I1, I2, I3} and {I1, I2, I5}. After this, we will get the itemsets {I1, I3, I5} and {I2, I3, I5} as candidate itemsets for the itemsets having three items. Let us calculate their support count.

Itemset	Support Count
{I2, I3, I5}	2
{I1, I3, I5}	2

In the above table, both itemsets have a support count of 2 which is equal to the minimum support count. Hence, both itemsets will be considered frequent itemsets.

Itemset	Support Count
{I2, I3, I5}	2
{I1, I3, I5}	2

Calculate Frequent Itemsets With Four Items

Now, we will calculate the frequent itemsets with four items. For this, we will first join the items in the frequent itemsets with three items to create itemsets with four items. We will get only one itemset as shown below.

{I1, I2, I3, I5}

The above itemset has four subsets with three elements i.e. {I2, I3, I5},{I1, I3, I5}, {I1, I2, I5}, {I1, I2, I3}. In these itemsets, {I1, I2, I5} and {I1, I2, I3} are not frequent itemsets. Hence, we will prune the itemset {I1, I2, I3, I5}. Thus, we have no candidate set for itemsets with 4 items. Hence, the process of frequent itemset generation stops here.

Now, let us tabulate all the frequent itemsets created in this numerical example on the apriori algorithm.

Itemset	Support Count
{I1}	3
{I2}	3
{I3}	4
{I5}	4
{I1,I3}	3
{I1,I5}	2
{I2,I3}	2
{I2,I5}	3
{I3,I5}	3
{I2, I3, I5}	2
{I1, I3, I5}	2

Generate Association Rules From Frequent Itemsets

To create association rules from the frequent itemsets, we will select each itemset. Then, we will select a subset from the itemset and make it an antecedent of the association rule. We will make the rest of the items as the consequent of the association rule. All the association rules that can be generated from the frequent itemsets are shown in the following table.

Itemset	Association Rules
{I1}	x
{I2}	x
{I3}	x
{I5}	x
{I1,I3}	{I1}->{I3}, {I3}->{I1}
{I1,I5}	{I1}->{I5}, {I5}->{I1}

{I2,I3}	{I2}->{I3}, {I3}->{I2}
{I2,I5}	{I2}->{I5}, {I5}->{I2}
{I3,I5}	{I3}->{I5}, {I5}->{I3}
{I2, I3, I5}	{I2}->{I3, I5}, {I3}->{I2, I5}, {I5}->{I2, I3}, {I2, I3}->{I5}, {I2, I5}->{I3}, {I3, I5}->{I2}
{I1, I3, I5}	{I1}->{I3, I5}, {I3}->{I1, I5}, {I5}->{I1, I3}, {I1, I3}->{I5}, {I1, I5}->{I3}, {I3, I5}->{I1}

Association Rule	Confidence
{I1}->{I3}	100%
{I3}->{I1}	75%
{I2}->{I3}	66.67%
{I3}->{I2}	50%
{I1}->{I5}	66.67%
{I5}->{I1}	50%
{I2}->{I5}	100%
{I5}->{I2}	75%
{I3}->{I5}	75%
{I5}->{I3}	75%
{I2}->{I3, I5}	66.67%
{I3}->{I2, I5}	50%
{I5}->{I2, I3}	50%
{I2, I3}->{I5}	100%
{I2, I5}->{I3}	66.67%
{I3, I5}->{I2}	66.67%
{I1}->{I3, I5},	66.67%

{I3}->{I1, I5}, 50%

{I5}->{I1, I3}, 50%

{I1, I3}->{I5}, 66.67%

{I1, I5}->{I3}, 100%

{I3, I5}->{I1}, 66.67%

Mining Frequent Itemsets without Candidate Generation

The **Apriori algorithm** is a traditional method for mining **frequent itemsets**, but it suffers from high computational costs due to **candidate generation** and multiple database scans. To overcome these limitations, the **FP-Growth (Frequent Pattern Growth) algorithm** was introduced, which **mines frequent itemsets without candidate generation**.

Why Avoid Candidate Generation?

Candidate generation (as used in Apriori) involves:

- **Multiple database scans**, increasing time complexity.
- **Generating a large number of candidate itemsets**, leading to excessive memory usage.
- **Checking each candidate's support individually**, slowing down performance.

Instead, **FP-Growth compresses the database** using a compact data structure called an **FP-Tree (Frequent Pattern Tree)** and extracts frequent itemsets efficiently.

How FP-Growth Works

The FP-Growth algorithm follows these steps:

Step 1: Construct the FP-Tree

- **Scan the dataset once** to count the frequency of items.
- **Sort items in descending order of frequency**.
- **Build a tree structure**, where paths represent transactions.

Step 2: Extract Frequent Patterns

- **Start from the least frequent item** (bottom-up traversal).
- **Find all occurrences of the item** in the FP-tree.
- **Extract prefix paths** (conditional pattern bases).
- **Recursively mine frequent itemsets**.

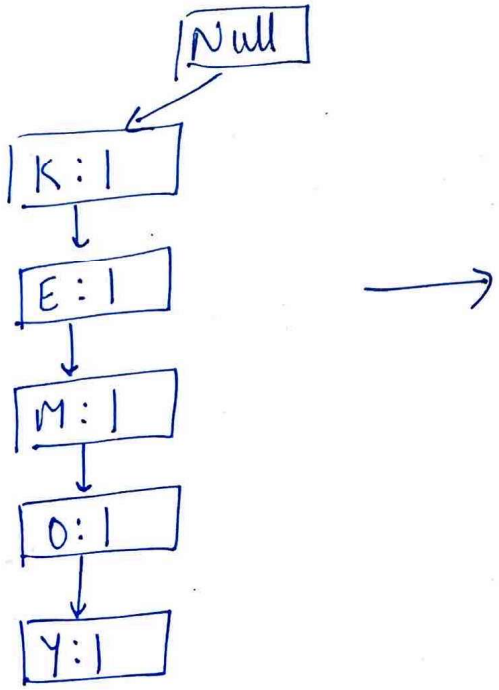
Frequent Pattern Growth Algo

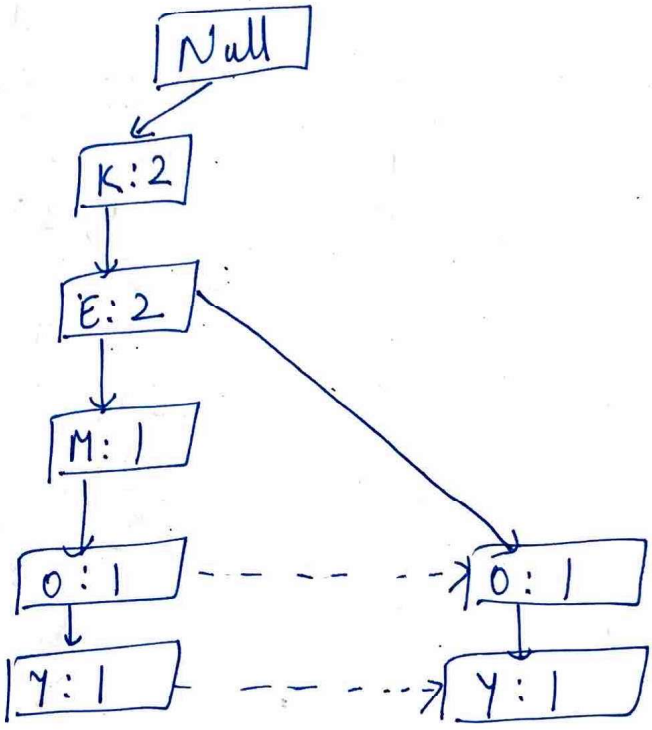
TID	Items
T1	{E, K, M, N, O, Y}
T2	{D, E, K, N, O, Y}
T3	{A, E, K, M}
T4	{C, K, M, U, Y}
T5	{C, E, I, K, O, O}

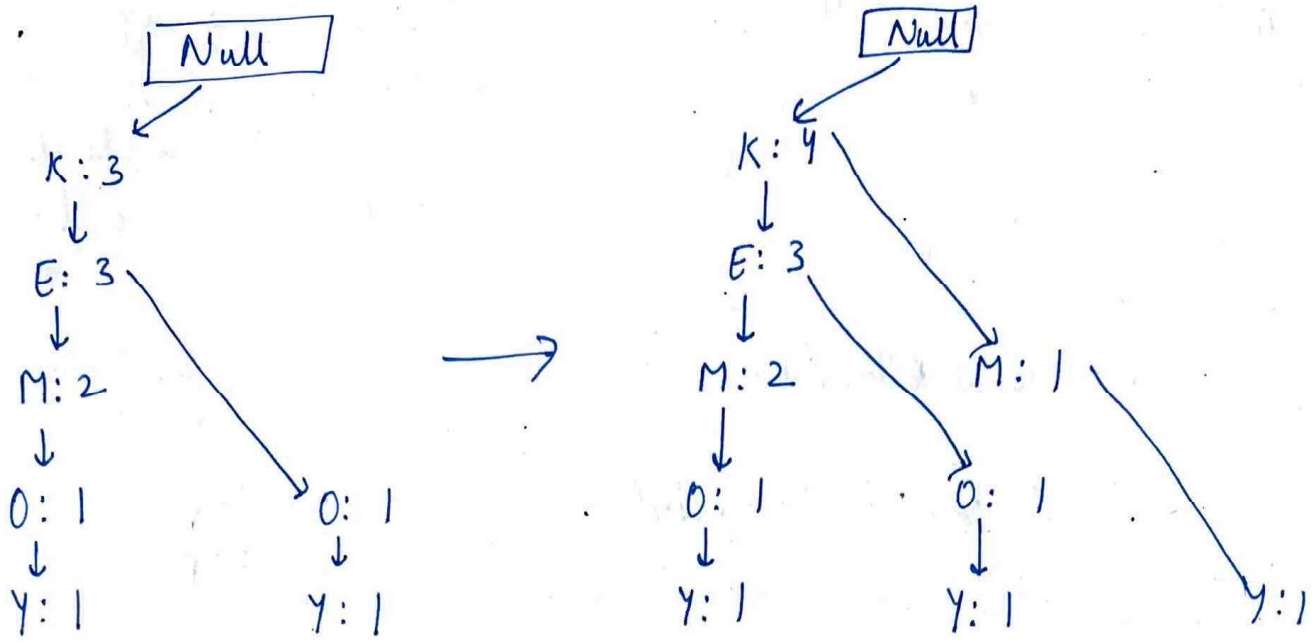
Item	Freq
A	1
C	2
D	1
E	4
I	1
K	5
M	3
N	2
O	3
U	1
Y	3

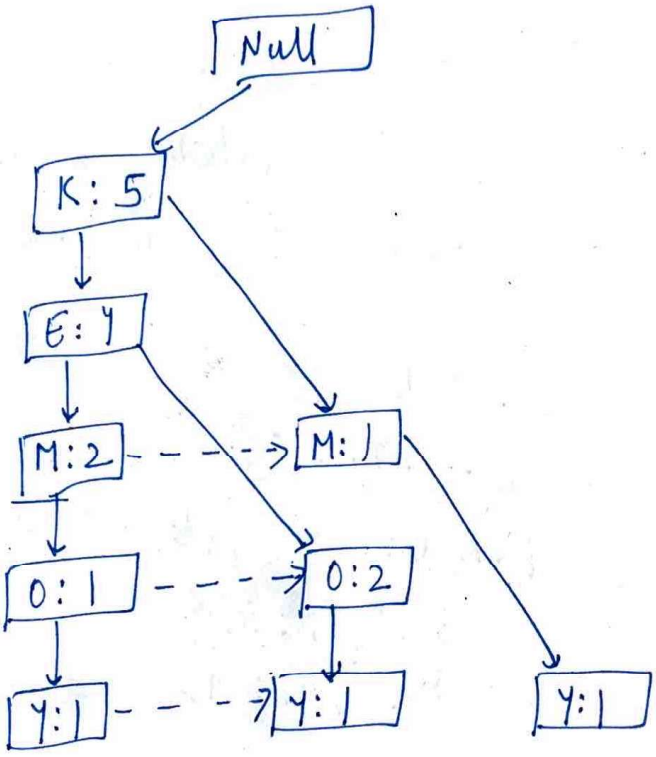
Min Supp count = 3

TID	Items	Ordered Item Set
T1	{E, K, M, N, O, Y}	{K, E, M, O Y}
T2	{D, E, K, N, O, Y}	{K, E, O, Y}
T3	{A, E, K, M}	{K, E, M}
T4	{C, K, M, U, Y}	{K, M, Y}
T5	{C, E, I, K, O, O}	{K, E, O}









For each item, conditional pattern base is computed which is path labels of all the paths which lead to any node of given item in the frequent pattern tree. The items are arranged in the ascending order of their frequencies.

Items	Conditional Pattern Base
Y	{K, E, M, O: 1}, {K, E, O: 1}, {K, M: 1}
O	{K, E, M: 1}, {K, E: 2}
M	{K, E: 2}, {K: 1}
E	{K: 4}
K	

Now, conditional frequent pattern tree is built. It is done by taking the set of elements that is common in all the paths in conditional pattern base of that item & cal. its support count by summing the support counts of all the paths in conditional pattern base.

Items	Conditional Pattern Base	Conditional Frequent Pattern Tree
Y	{K, E, M, O : 1}, {K, E, O : 1}, {K, M : 1}	{K : 3}
O	{K, E, M : 1}, {K, E : 2}	{K, E : 3}
M	{K, E : 2}, {K : 1}	{K : 3}
E	{K : 4}	{K : 4}

Now, frequent pattern rules are generated by pairing the items of conditional frequent pattern tree to the corresponding item

Items	Rules
Y	{<K, Y : 3>} —
O	{<K, O : 3>}, {<E, O : 3>, <E, K, O : 3>}
M	{<K, M : 3>}
E	{<E, K : 4>}

$K \rightarrow Y$ & $Y \rightarrow K$

↓

Cal. confidence values

If $K \rightarrow O$

$O \rightarrow K$

Association Rules – To generate association rules, a new table will be created with the possible rules from combination {A, B, C}

Rules	Supp	Confidence
$A \wedge B \rightarrow C$	2	$Sup((A \wedge B) \wedge C) / sup(A \wedge B) = 2/4 = 0.5 = 50\%$
$B \wedge C \rightarrow A$	2	$Sup\{(B \wedge C) \wedge A\} / sup(B \wedge C) = 2/4 = 0.5 = 50\%$
$A \wedge C \rightarrow B$	2	$Sup\{(A \wedge C) \wedge B\} / sup(A \wedge C) = 2/4 = 0.5 = 50\%$
$C \rightarrow A \wedge B$	2	$Sup((C \wedge (A \wedge B)) / sup(C) = 2/5 = 0.4 = 40\%$
$A \rightarrow B \wedge C$	2	$= 2/6 = 0.33 = 33.3\%$
$B \rightarrow B \wedge C$	2	$= 2/7 = 0.28 = 28\%$

Q.

TID	Items
1	I1, I2, I5
2	I2, I4
3	I2, I3
4	I1, I2, I4
5	I1, I3
6	I2, I3
7	I1, I3
8	I1, I2, I3, I5
9	I1, I2, I3

Mining Frequent Itemsets using Vertical Data Format

Mining frequent itemsets is a key step in association rule mining and market basket analysis. Traditional algorithms like Apriori use a horizontal database format, where transactions are stored as lists of items. However, another approach is the vertical data format, where each item is associated with the transactions in which it appears.

Differences Between Horizontal and Vertical Data Formats

Format	Representation	Example
Horizontal Format	**Transaction-based**	**{T1: {A, B, C}, T2: {B, C, D}}**
Vertical Format	**Item-based (TID sets)**	**{A: {T1}, B: {T1, T2}, C: {T1, T2}, D: {T2}}**

In the vertical format, each item is linked to a TID-list (transaction ID list), making it easier to compute support using set intersection.

Why Use Vertical Data Format?

Using a vertical format allows us to:

- Reduce database scans by storing transactions efficiently.
- Speed up frequent itemset mining by using set intersections.
- Process large and dense datasets more efficiently than Apriori.

Comparison of Itemset Mining Approaches

Algorithm	Data Representation	Candidate Generation?	Efficiency
Apriori	Horizontal (list of transactions)	Yes	Slow for large datasets
FP-Growth	Tree structure (FP-Tree)	No	Fast for large datasets
ECLAT	Vertical (TID lists)	No	Efficient for dense datasets

How ECLAT Works

The Equivalence Class Clustering and Bottom-Up Lattice Traversal (ECLAT) Algorithm is a depth-first search (DFS) method that:

1. Stores transactions in vertical format (each item → TID set).
2. Finds frequent 1-itemsets by counting the size of each TID set.

3. Generates frequent k-itemsets by intersecting TID lists of smaller itemsets.
4. Prunes infrequent itemsets based on minimum support.

Step-by-Step Example of ECLAT Using Vertical Format

Given Transactions (Horizontal Format)

Transaction ID	Items Bought
1	A, B, D, E
2	B, C, E
3	A, B, C, E
4	B, C
5	A, B, C, E

Step 1: Convert to Vertical Format (TID Sets)

Item	TID Set
A	{1, 3, 5}
B	{1, 2, 3, 4, 5}
C	{2, 3, 4, 5}
D	{1}
E	{1, 2, 3, 5}

Step 2: Find 1-Frequent Itemsets

The support of an item is the number of transactions it appears in:

- **{A} → {1, 3, 5} (Support = 3)**
- **{B} → {1, 2, 3, 4, 5} (Support = 5)**
- **{C} → {2, 3, 4, 5} (Support = 4)**
- **{D} → {1} (Support = 1, Discarded ❌)**
- **{E} → {1, 2, 3, 5} (Support = 4)**

If min support = 2, we remove {D} because its support is less than 2.

Step 3: Find 2-Frequent Itemsets (Intersection of TID Sets)

Now, we compute the intersection of TID sets to generate 2-itemsets.

Itemset	Intersection (TID Set)	Support
{A, B}	{1, 3, 5}	3 ✅
{A, C}	{3, 5}	2 ✅
{A, E}	{1, 3, 5}	3 ✅
{B, C}	{2, 3, 4, 5}	4 ✅
{B, E}	{1, 2, 3, 5}	4 ✅
{C, E}	{2, 3, 5}	3 ✅

Step 4: Find 3-Frequent Itemsets (Intersection of TID Sets)

Itemset	Intersection (TID Set)	Support
{A, B, E}	{1, 3, 5}	3 ✅
{B, C, E}	{2, 3, 5}	3 ✅

At this point, all frequent itemsets have been found.

Mining Closed Frequent Itemsets

In frequent itemset mining, a frequent itemset is an itemset that appears at least a certain number of times (defined by the minimum support threshold) in a transaction database. However, storing all frequent itemsets can be redundant and inefficient, leading to excessive computational costs.

- A **frequent itemset** is a set of items that appears in a dataset with a frequency (support) above a given threshold.
- A **closed itemset** is a frequent itemset for which no proper superset has the same support.
- A **closed frequent itemset (CFI)** is both frequent and closed.

To reduce redundancy, we focus on Closed Frequent Itemsets (CFIs).

Definition: Closed Frequent Itemset (CFI)

A frequent itemset X is closed if no superset of X has the same support as X.

- **In simple terms: A frequent itemset is "closed" if it cannot be extended without reducing its support.**
- Mining **Closed Frequent Itemsets** instead of **all frequent itemsets** offers several advantages:
- **Advantages of CFIs:**
- ✓ **Compact Representation** – Significantly reduces the number of itemsets stored.
  ✓ **No Information Loss** – Contains the same essential knowledge as all frequent itemsets.
  ✓ **Efficient Rule Generation** – Simplifies association rule mining.
  ✓ **Improved Performance** – Reduces computation time and memory usage.

1. Understanding Supersets

A **superset** of an itemset **X** is any itemset that **contains all elements of X**, plus at least one additional item.

🔹 Example:

If **X = {A, B}**, then some of its supersets are:

✅ **{A, B, C}**

✅ **{A, B, D}**

✅ **{A, B, C, D}**

Example Dataset

Transaction ID	Items Bought
1	A, B, C
2	A, B
3	A, B, C
4	B, C
5	A, C

Let's say we want to find the supersets of **{A, B}**:

1. Look for frequent itemsets that contain **A and B**.
2. In our dataset, these itemsets are:
   - {A, B, C} ✅ (Contains {A, B} and one extra item "C")
   - {A, B, D} ❌ (Does not exist in this dataset)
   - {A, B, C, D} ❌ (Does not exist)

So the **superset of {A, B}** in this dataset is **{A, B, C}**.

Algorithm for Mining Closed Itemsets :

Step 1: Scan the dataset to find frequent itemsets.
Step 2: For each frequent itemset, check if it is a closed itemset by comparing its support to the support of all its supersets. If no superset has the same support, the itemset is closed.
Step 3: Store the closed itemsets in a compact structure.
Step 4: Use the closed itemsets for further analysis or rule generation.

Example: Given a set of transactions:
T1: {A, B, C}T2: {A, B}T3: {B, C}T4: {A, C}

Let's say the minimum support threshold is 2.Frequent itemsets (with support ≥ 2) are:{A} (support = 3)
{B} (support = 3)
{C} (support = 3)
{A, B} (support = 2)
{B, C} (support = 2)
{A, C} (support = 2)

Now, let's find the closed itemsets:
{A, B} is a closed itemset because no superset of {A, B} (like {A, B, C}) has the same support (it has support 1, so it doesn't match).
Similarly, {B, C} and {A, C} are also closed because no supersets have the same support.
Thus, the closed itemsets are:{A}, {B}, {C}, {A, B}, {B, C}, {A, C}

Example: Finding Closed Frequent Itemsets

Consider a transaction database:

Transaction ID	Items Bought
1	A, B, C
2	A, B
3	A, B, C, D
4	B, C, D
5	B, C

Step 1: Find All Frequent Itemsets (Using min_support = 2)

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

Step 2: Identify Closed Frequent Itemsets

- **{A} is NOT closed because {A, B} has the same support.**
- **{B} is NOT closed because {B, C} has the same support.**
- **{A, B, C} is closed (no superset has the same support).**
- **{B, C} is closed (no superset has the same support).**
- **{B, C, D} is closed (no superset has the same support).**
- **Final Closed Frequent Itemsets (CFIs)**

Itemset	Support
{A, B, C}	2
{B, C}	4
{B, C, D}	2

Mining Multilevel Association Rules

In traditional association rule mining, we find relationships between items at a single level (e.g., "Milk → Bread"). However, in many real-world applications, data is organized in a hierarchical or multilevel structure.

If we only mine frequent itemsets at a single level, we might miss important patterns that exist at higher or lower levels. Multilevel association rule mining helps discover rules at different levels of abstraction.

Why Use Multilevel Association Rules?

✓ More Insightful Patterns – Discovers both general and specific relationships.
✓ Better Decision Making – Helps in inventory management, marketing, and recommendation systems.
✓ More Accurate Rules – Avoids missing important relationships hidden in hierarchical data.

Example: Single-Level vs. Multilevel Association Rules

Rule Type	Example
Single-Level Rule	{Milk} → {Bread}
Multilevel Rule	{Dairy Products} → {Bakery Products}
More Specific Rule	{Low-Fat Milk} → {Whole Wheat Bread}

Types of Multilevel Association Rule Approaches

There are two main approaches for mining multilevel association rules:

(a) Top-Down Approach (Apriori-Based)

- First mine frequent itemsets at higher levels, then go deeper if necessary.
- Uses different minimum support thresholds for different levels (MSD - Multiple Support Difference).

◆ Example:

- Min support for Dairy Products (general) = 5%
- Min support for Cheese (specific) = 2%

Higher levels may need higher support thresholds to prevent too many rules.

(b) Bottom-Up Approach (FP-Growth-Based)

- Start mining frequent itemsets at the lowest level (most detailed items).
- Use generalization techniques to merge them into higher-level patterns.
- Faster than Apriori-based methods because it avoids candidate generation.

◆ Example:

- {Whole Milk, Butter} → {Bread}

- Generalized to {Dairy Products} → {Bakery Products}

Problem Statement

A supermarket has recorded transactions, and products are organized in a hierarchical structure as follows:

```
Level 1: Category → Level 2: Subcategory → Level 3: Specific Items
-------------------------------------------------------------------
Beverages → Soft Drinks → {Coca-Cola, Pepsi}
Beverages → Juices → {Orange Juice, Apple Juice}
Snacks → Chips → {Lays, Pringles}
Snacks → Biscuits → {Oreo, Bourbon}
```

The store has the following transactions:

Transaction ID	Items Purchased
1	Coca-Cola, Lays, Oreo
2	Pepsi, Pringles, Oreo
3	Orange Juice, Lays, Oreo
4	Coca-Cola, Lays, Bourbon
5	Apple Juice, Pringles, Bourbon

The **minimum support threshold** is:

- **Level 1**: 60%
- **Level 2**: 50%
- **Level 3**: 40%

Step 1: Calculate Support for Each Itemset

Level 1: Categories

Category	Support Count	Support %
Beverages	5	100%
Snacks	5	100%

Both categories are **frequent (≥ 60%)**.

Level 2: Subcategories

Subcategory	Support Count	Support %
Soft Drinks	3	60% ✅
Juices	2	40% ❌
Chips	4	80% ✅
Biscuits	4	80% ✅

Level 3: Specific Items

Item	Support Count	Support %
Coca-Cola	2	40% ✅
Pepsi	1	20% ❌
Orange Juice	1	20% ❌
Apple Juice	1	20% ❌
Lays	3	60% ✅
Pringles	2	40% ✅
Oreo	3	60% ✅
Bourbon	2	40% ✅

- **Frequent items:** Coca-Cola, Lays, Pringles, Oreo, Bourbon.
- **Non-frequent items (pruned):** Pepsi, Orange Juice, Apple Juice.