DATA SCIENCE

Market Basket Analysis

Association Rules

Market Basket Analysis

Can we really get insight from market baskets?

Market Basket Analysis

- Provides insight into which products tend to be purchased together and which are most amenable to promotion.

- Can return
  - Actionable rules
  - Trivial rules

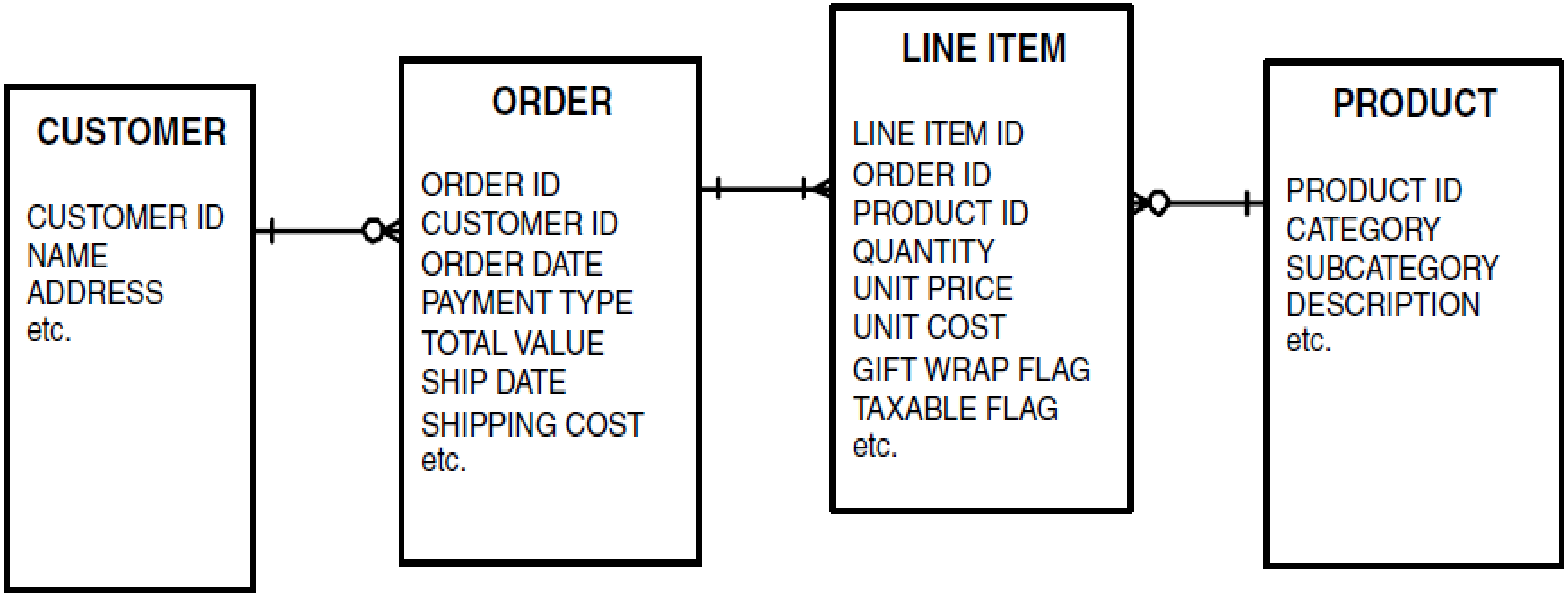
    -People who buy shoes also buy socks
  - Inexplicable

    -People who buy shirts also buy milk

Market Basket Analysis

- Cross Selling
  - Offer an associated item when the customer buys any product
- Product Placement
- Customer Behaviour
  - Based on Credit Card usage data, we may be able to detect certain purchase behaviour that can be associated with fraud
- Fraud Detection
- Pharma
  - Medical patient histories can give indications of likely complications based on certain combinations of treatments

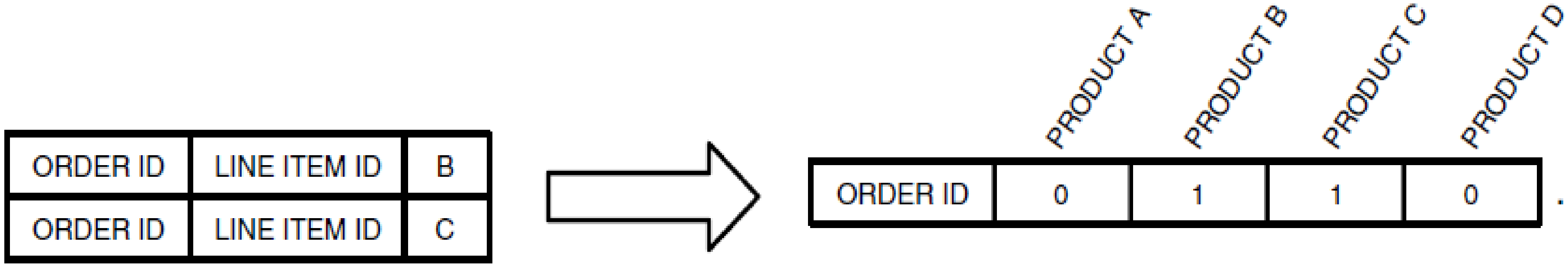
Database Structure



Frequent Market Basket Questions

- What is the average number of orders per customer?
- What is the most common item found in a one-item order?
- What is the average number of unique items per order?
- What is the average number of items per order?

Transform the Data and form a Co-Occurrence Table



	Product A	Product B	Product C	Product D
Product A				
Product B				
Product C				
Product D				

Use Case

ID	Product
1	Orange juice
2	Soda
3	Milk
4	Window cleaner
5	Detergent

Line item table

ID	Order ID	Product ID	Quantity
1	1	1	2
2	1	2	1
3	2	3	3
4	2	1	2
5	2	4	1
6	3	1	2
7	3	5	3
8	4	1	1
9	4	5	1
10	4	2	2
11	5	2	2
12	5	4	3

Order ID	Products
1	Orange juice, Soda
2	Milk, orange juice, window cleaner
3	Orange juice, detergent
4	Orange juice, detergent, soda
5	Window cleaner, soda

Find the Insights??

Product	OJ	Window Cleaner	Milk	Soda	Detergent
OJ	4	1	1	2	2
Window cleaner	1	2	1	1	0
Milk	1	1	1	0	0
Soda	2	1	0	3	1
Detergent	2	0	0	1	2

How do we generate these rules automatically on large data?

Apriori Algorithm

- FP Tree

Apriori Algorithm

Database D

Minsup = 0.5

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}

L_3

itemset	sup
{2 3 5}	2

Apriori Algorithm

- Apriori can be very slow as it needs to compute the support at every instance by looking at the original itemset

- We need a quicker implementation
  - FP Tree

Example – FP TREE

TID	*Items bought*
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*}

Example – FP TREE

Item	*frequency*
f	*4*
c	*4*
a	*3*
b	*3*
m	*3*
p	*3*

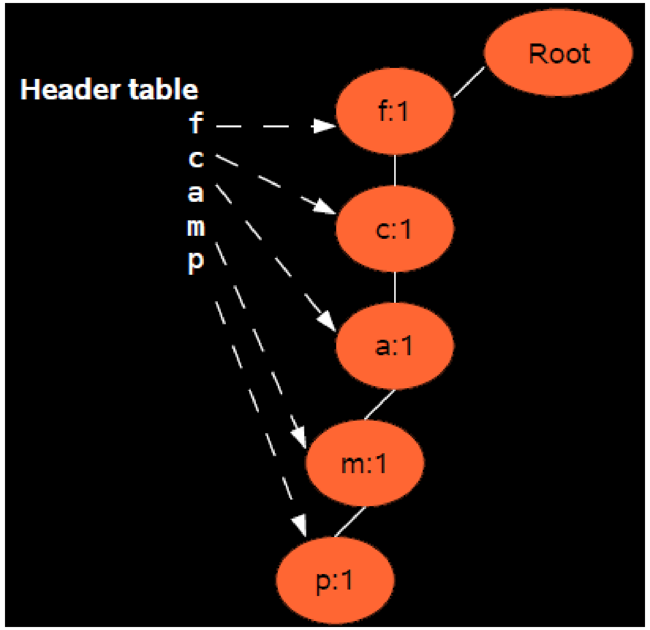
We avoided all those items that do not have the minimum support of 50% (so, a count of 3 in 5 transactions). So, d, e, g, h, I, j, k, l and n are dropped as their count is lower than 2.

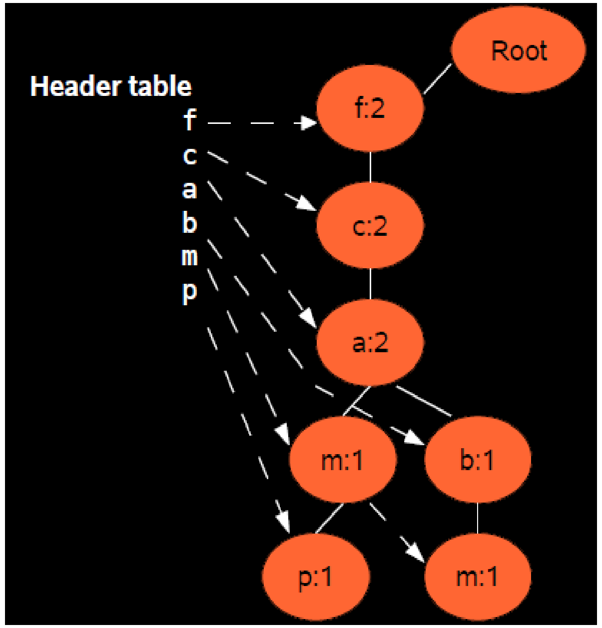
Re-Order the Item Sets

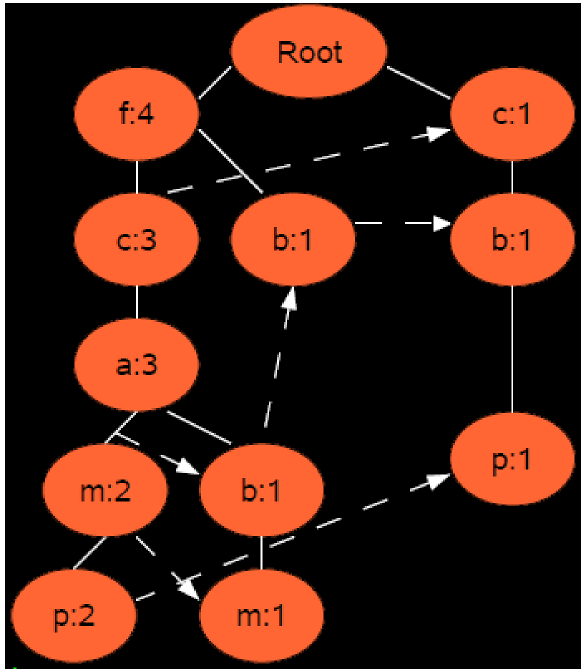
TID	*Items bought*	*(ordered) frequent items*
100	{*f, a, c, d, g, i, m, p*}	{*f, c, a, m, p*}
200	{*a, b, c, f, l, m, o*}	{*f, c, a, b, m*}
300	{*b, f, h, j, o*}	{*f, b*}
400	{*b, c, k, s, p*}	{*c, b, p*}
500	{*a, f, c, e, l, p, m, n*}	{*f, c, a, m, p*}

FP TREE

$\{f, c, a, m, p\}$
$\{f, c, a, b, m\}$
$\{f, b\}$
$\{c, b, p\}$
$\{f, c, a, m, p\}$







FP TREE

- It never breaks a long pattern of any transaction
- reduces irrelevant information—infrequent items are gone
- More frequent items are more likely to be shared and are at the top
- We keep a count at the nodes to compute support/confidence. So, no need to view the DB again.