SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

SUBJECT NAME: Data Mining and Warehousing

SUBJECT CODE: SIT1301

ASSOCIATION RULE MINING

ASSOCIATION RULE MINING

3.1 Frequent patterns are patterns (e.g., itemsets, subsequences, or substructures) that appear frequently in a data set. For example, a set of items, such as milk and bread, that appear frequently together in a transaction data set is a ***frequent itemset***.

- A subsequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a (*frequent*) ***sequential pattern***.

- A *substructure* can refer to different structural forms, such as subgraphs, subtrees, or sublattices, which may be combined with itemsets or subsequences. If a substructure occurs frequently, it is called a **(*frequent*) *structured pattern***.

Association Mining

- **Association rule mining**
  - Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.

➢ **Applications**

- Basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, classification, etc.

➢ **Examples.**

- Rule form: —Body ® Head [support, confidence].
- buys(x, —diapers‖) ® buys(x, —beers‖) [0.5%, 60%]
- major(x, —CS‖) ^ takes(x, —DB‖) ® grade(x, —A‖) [1%, 75%]

3.2 Association and Correlations

➢ **Association Rule: Basic Concepts**

➢ Given: (1) database of transactions, (2) each transaction is a list of items (purchased by a customer in a visit)

➢ Find: all rules that correlate the presence of one set of items with that of another set of items

- E.g., *98% of people who purchase tires and auto accessories also get automotive services done*

- **Applications**
  - $* \Rightarrow$ *Maintenance Agreement* (What the store should do to boost Maintenance Agreement sales)
  - *Home Electronics* $* \Rightarrow$ (What other products should the store stocks up?)
  - Attached mailing in direct marketing
  - Detecting - ping-pongling of patients, faulty - collisions||

Rule Measures: Support and Confidence

- Find all the rules X & $Y \Rightarrow Z$ with minimum confidence and support
  - support, *s*, probability that a transaction contains {X 4 Y 4 Z}
  - confidence, *c,* conditional probability that a transaction having {X 4 Y} also contains *Z*

- Let minimum support 50%, and minimum confidence 50%, we have
  - $A \Rightarrow C$ (50%, 66.6%)
  - $C \Rightarrow A$ (50%, 100%)

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

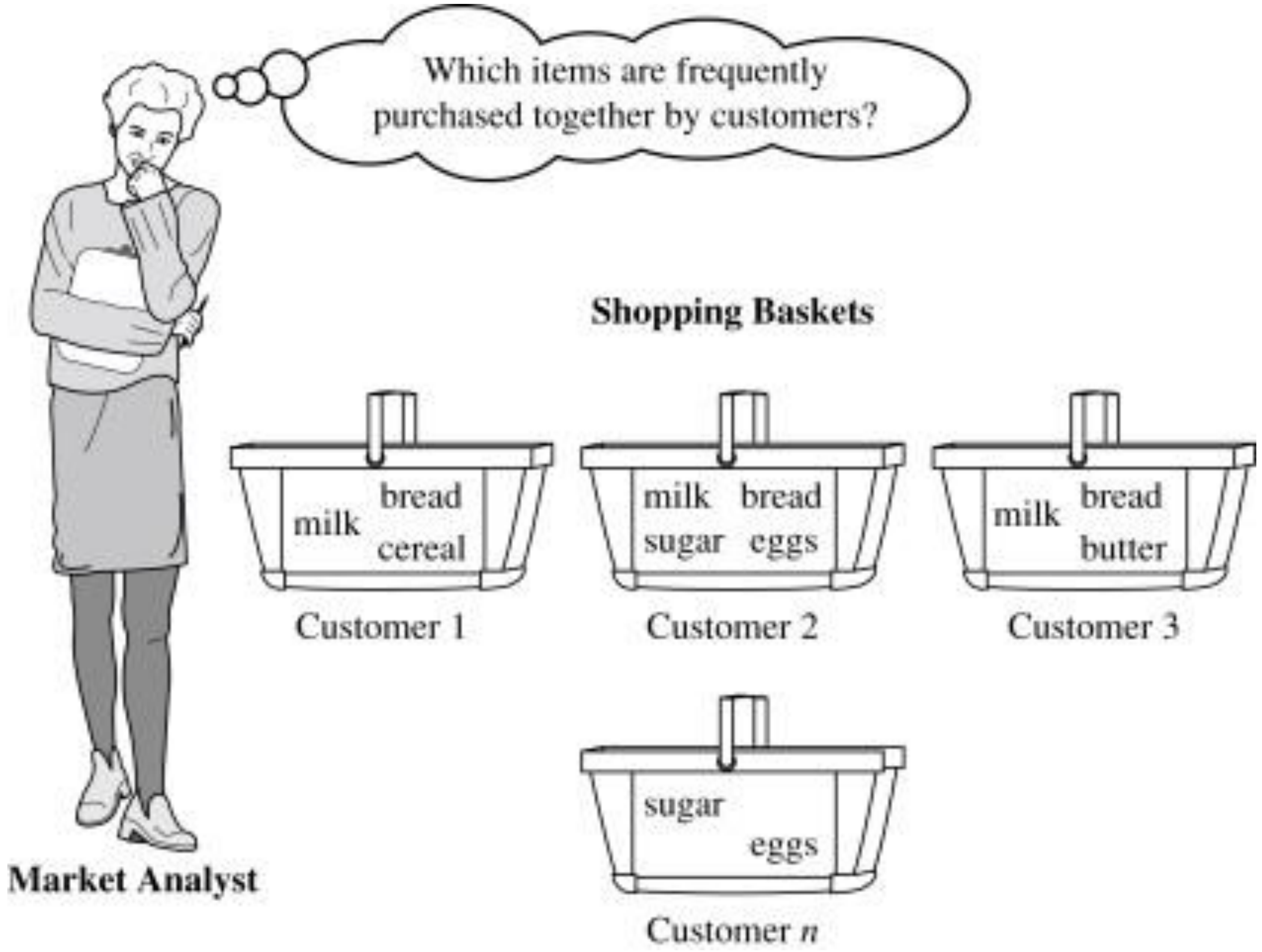
Association Rule Mining: A Road Map

- Boolean vs. quantitative association Based on the types of values handled)
  - buys(x, ―SQLServer||) ^ buys(x, ―DMBook|| ® buys(x, ―DBMiner) [0.2%, 60%]
  - age(x, ―30..39||) ^ income(x, ―42..48K||) ® buys(x, ―PC||) [1%, 75%]
- Single dimension vs. multiple dimensional associations (see ex. Above)
- Single level vs. multiple-level analysis
  - What brands of beers are associated with what brands of diapers?
- Various extensions
  - Correlation, causality analysis
    - Association does not necessarily imply correlation or causality
- Maxpatterns and closed itemsets
  - Constraints enforced
    - E.g., small sales (sum < 100) trigger big buys (sum > 1,000)?

Market – Basket analysis

- A market basket is a collection of items purchased by a customer in a single transaction, which is a well-defined business activity. For example, a customer's visits to a grocery store or an online purchase from a virtual store on the Web are typical customer transactions. Retailers accumulate huge collections of transactions by recording business activities over time. One common analysis run against a transactions database is to find sets of items, or *itemsets*, that appear together in many transactions. A business can use knowledge of these patterns to improve the Placement of these items in the store or the layout of mail-order catalog page and Web pages. **An itemset containing *i* items is called an *i- itemset*. The percentage of transactions that contain an itemset is called the itemset's *support*. For an itemset to be interesting, its support must be higher than a user-specified minimum. Such itemsets are said to be frequent.**

Figure : Market basket analysis



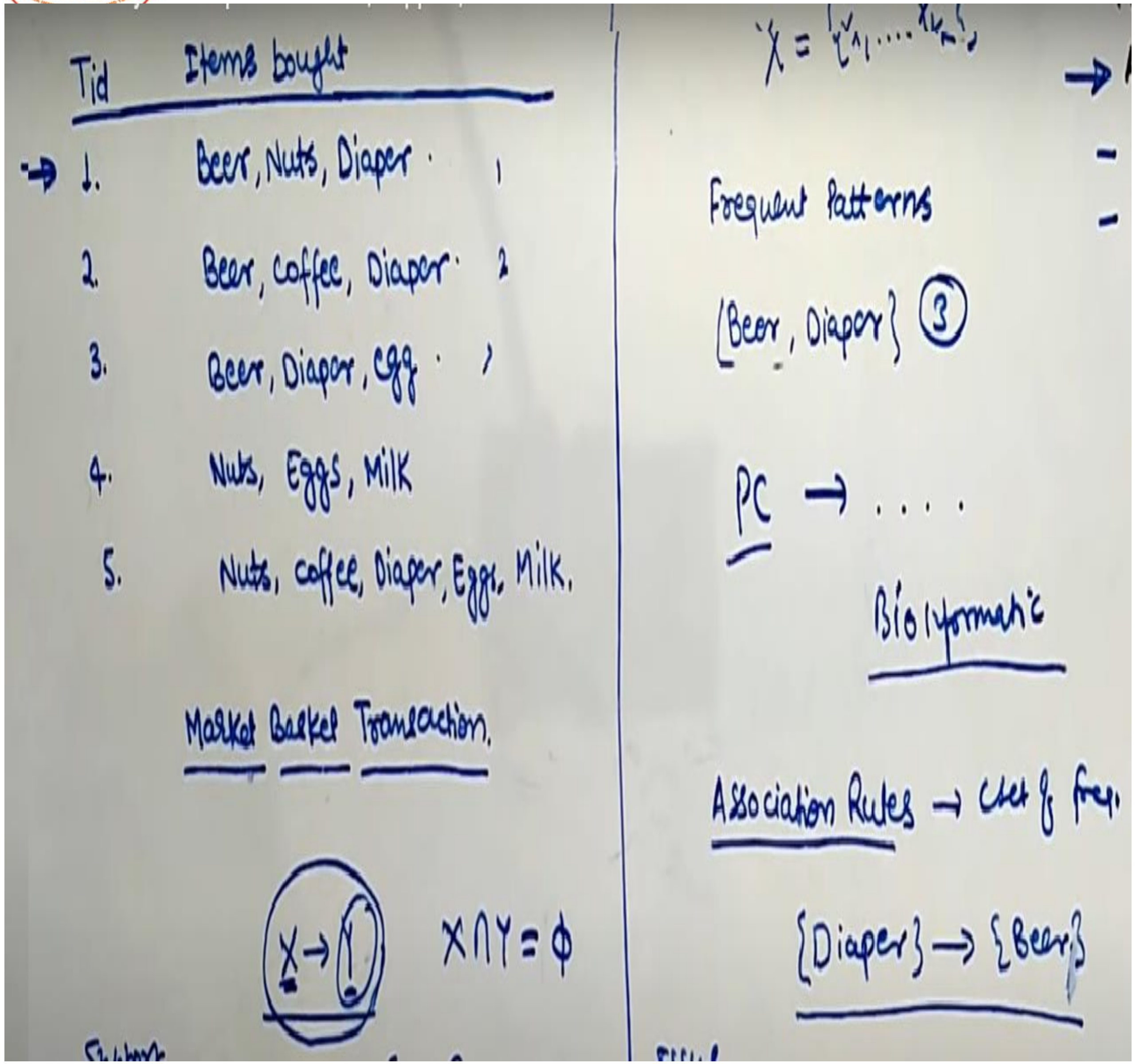
Computer ⇒ financial_management_ software [support = 2%, confidence = 60%]

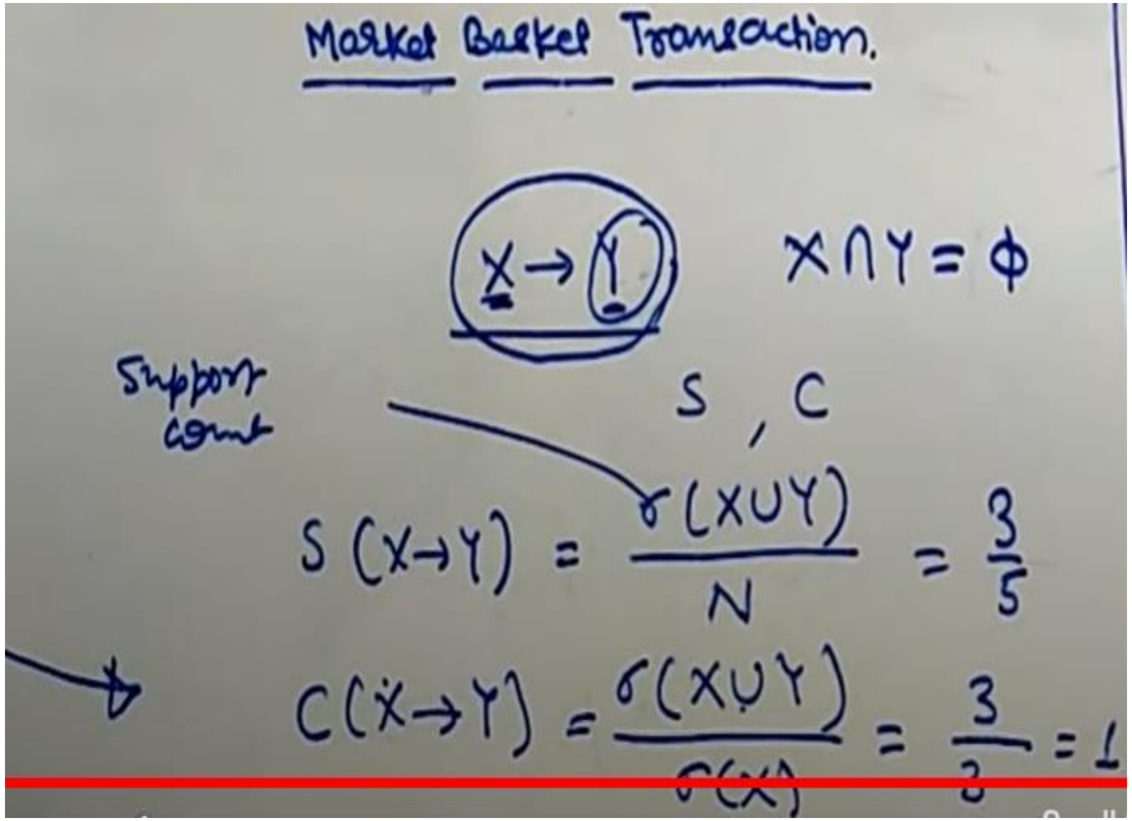
- Rule **support and confidence** are two measures of rule interestingness. They respectively reflect the usefulness and certainty of discovered rules. A support of 2% for association Rule means that 2% of all the transactions under analysis show that computer and financial management software are purchased together. **A confidence of 60% means that 60% of the customers who purchased a computer also bought the software.** Typically, association rules are considered interesting if they satisfy both a minimum support threshold and a minimum confidence threshold.

3.3 Mining Methods

- Mining Frequent Pattern with candidate generation
- Mining Frequent Pattern without candidate generation

Support and Confidence





Support and Confidence

$I = \{I_1, I_2, \ldots I_m\}$ set of items

Association rule $\quad T \subseteq I$

$A \Rightarrow B$. $A \subset I$, $B \subset I$, and $A \cap B = \phi$

<u>Support</u> (s) is the percentage of transactions in D that contain $A \cup B$ (ie the union of sets A and B or say both A and B).

<u>Confidence</u> (c)

Percentage of transactions in D containing A that also contain B

Apriori: A Candidate Generation & Test Approach

- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

Transaction	Itemset
I_1	A, B, C
I_2	A, C
I_3	A, D
I_4	B, E, F

Min Support = 50%.
Min Confidence = 50%.

$\frac{50}{100} \times 4 = 2$

C_1

Items	Support
{A}	3
{B}	2
{C}	2
{D}	1
{E}	1
{F}	1

L_1

Items	Support
{A}	3
{B}	2
{C}	2

C_2

Items	Support
{A,B}	1
{B,C}	1
{A,C}	2

$\Rightarrow L_2$

Items	Support
{A,C}	2

Transaction	Itemset
I_1	A, B, C
I_2	A, C
I_3	A, D
I_4	B, E, F

Association rule	support	Confid	Conf%
$A \to C$	2	$2/3 = 0.66$	66%
$C \to A$	2	$2/2 = 1$	100%

$$A \to C = \frac{\text{Support}}{\text{occ. of } A} = \frac{2}{3} = 0.66$$

$$C \to A = \frac{\text{Support}}{\text{occurence of } C} = \frac{2}{2} = 1$$

Both rules are accepted because $S > C$ - above 50%

Final Rule

$A \to C$

$C \to A$

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 !=$\emptyset$; 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 $\cup_k L_k$;

3.4 Mining Frequent Patterns With candidate Generation

- **The method that mines the complete set of frequent itemsets with candidate generation.**

Apriori property & The Apriori Algorithm. Apriori property

- All nonempty subsets of a frequent item set most also be frequent.
  - An item set I does not satisfy the minimum support threshold, min-sup, then I is not frequent, i.e., support(I) < min-sup
  - If an item A is added to the item set I then the resulting item set (I U A) can not occur more frequently than I.
- Monotonic functions are functions that move in only one direction.
- This property is called anti-monotonic.
- If a set can not pass a test, all its supersets will fail the same test as well.
- This property is monotonic in failing the test.

The Apriori Algorithm

- Join Step: Ck is generated by joining Lk-1with itself
- Prune Step: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k- itemset

Input: Database, D, of transactions; minimum support threshold, min_sup.

Output: L_1 frequent itemsets in D.

Method

```
L1 = find_frequent_1 itemsets(D);
for (k = 2; Lk-1 ≠ φ; k++) {
        Ck = apriori_gen (Lk-1, min_sup);
        For each transaction t ∈ D { // scan D for counts
                Ct = subset (Ck, t); // get the subsets of t that are candidates
                for each candidate c ∈ Ct
                    c.count++;
        }
        Lk = {c ∈ Ck|c.count ≥ min_sup}
}
return L = UkLL;
```

Apriori/FP Growth

- Bottlenecks of the Apriori approach
  - Breadth-first (i.e., level-wise) search
  - Candidate generation and test
    - Often generates a huge number of candidates
- The FPGrowth Approach (J. Han, J. Pei, and Y. Yin, SIGMOD' 00)
  - Depth-first search
  - Avoid explicit candidate generation

3.6 Mining Frequent Item set without Candidate Generation

Frequent Pattern Growth Tree Algorithm

It grows long patterns from short ones using local frequent items

- “abc” is a frequent pattern
- Get all transactions having “abc”: DB|abc
- “d” is a local frequent item in DB | abc € abcd is a frequent pattern

Construct FP-tree from a Transaction Database

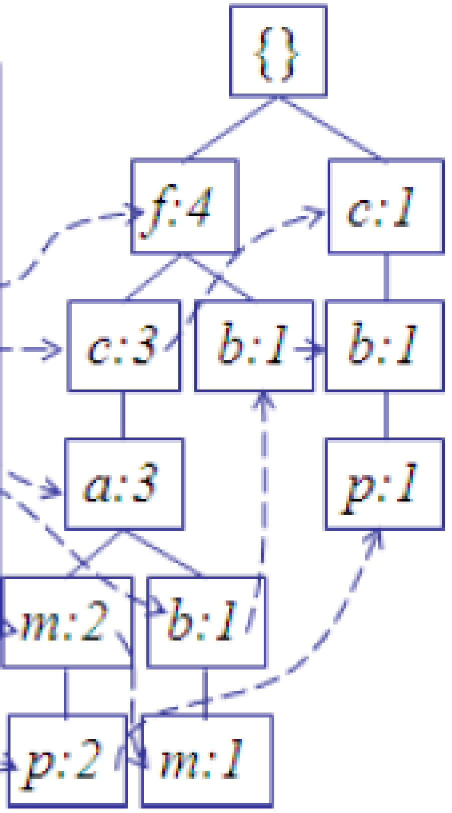
TID	*Items bought*	*(ordered) frequent items*
100	{*f, a, c, d, g, i, m, p*}	{*f, c, a, m, p*}
200	{*a, b, c, f, l, m, o*}	{*f, c, a, b, m*}
300	{*b, f, h, j, o, w*}	{*f, b*}
400	{*b, c, k, s, p*}	{*c, b, p*}
500	{*a, f, c, e, l, p, m, n*}	{*f, c, a, m, p*}

min_support = 3

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree

Header Table

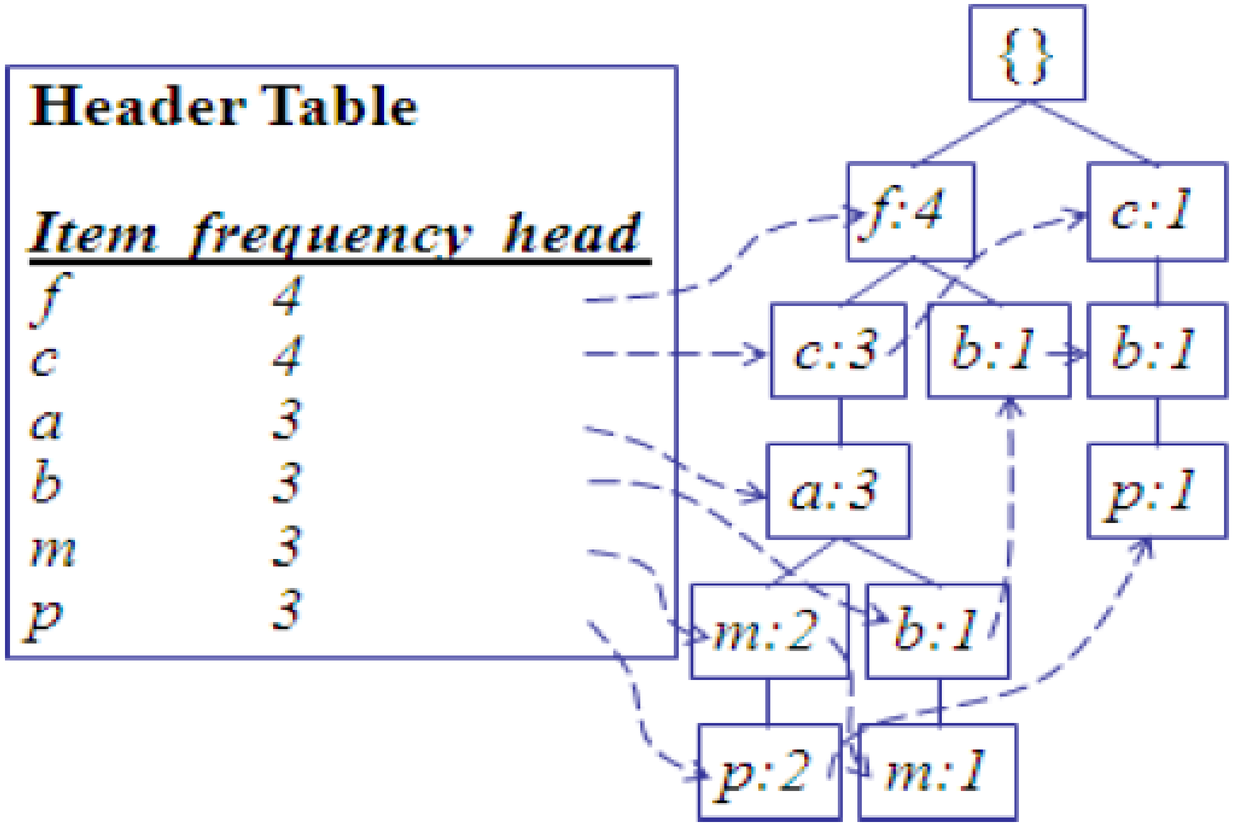
Item	*frequency*	*head*
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



F-list=f-c-a-b-m-p

Find Patterns Having P from P-conditional Database

- Starting at the frequent item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item P
- Accumulate all of transformed prefix paths of items *p* to form *P*'s conditional pattern base



Conditional pattern bases

item	cond. pattern base
c	f:3
a	fc:3
b	fca:1, f:1, c:1
m	fca:2, fcab:1
p	fcam:2, cb:1

Benefits of the FP-tree Structure

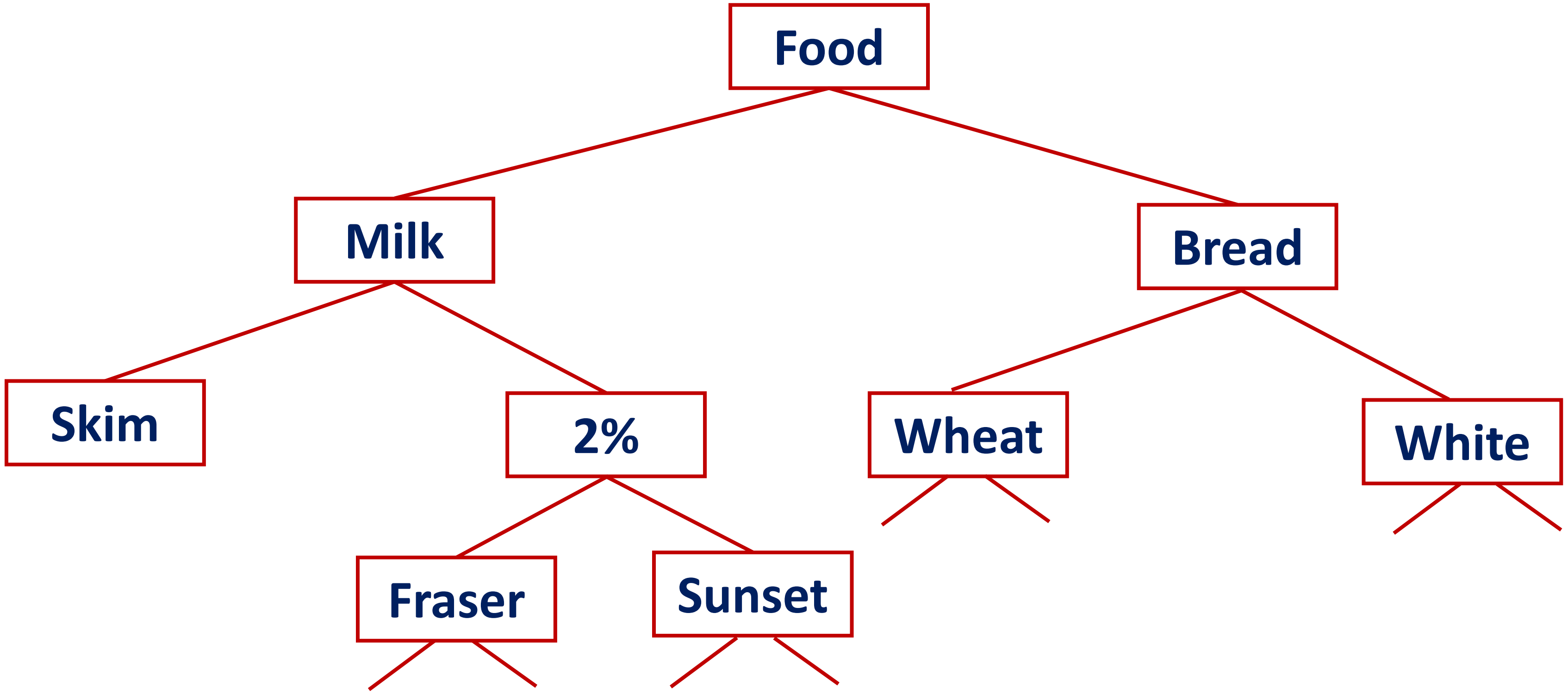
- **Completeness**
  - never breaks a long pattern of any transaction
  - preserves complete information for frequent pattern mining
- **Compactness**
  - reduce irrelevant information—infrequent items are gone
  - frequency descending ordering: more frequent items are more likely to be shared
  - never be larger than the original database (if not count node-links and counts)
  - Example: For Connect-4 DB, compression ratio could be over 100

3.7 Mining various kinds of Association Rule

- **Mining Multi-level association rule**
- **Mining Multi dimensional Association Rule**

Mining multilevel association rules from transactional databases.

- Items often form hierarchy.
- Items at the lower level are expected to have lower support.
- Rules regarding itemsets at appropriate levels could be quite useful.
- Transaction database can be encoded based on dimensions and levels
- We can explore shared multi-level mining



TID	Items
T1	{111, 121, 211, 221}
T2	{111, 211, 222, 323}
T3	{112, 122, 221, 411}
T4	{111, 121}
T5	{111, 122, 211, 221, 413}

3.8 Mining Multi-Level Associations

- A top_down, progressive deepening approach:
  - First find high-level strong rules:

    milk ® bread [20%, 60%]
  - Then find their lower-level —weaker| ||‖rules:

    2% milk ® wheat bread [6%, 50%].

- Variations at mining multiple-level association rules.
  - Level-crossed association rules:

    2% milk ® Wonder wheat bread
  - Association rules with multiple, alternative hierarchies:

    2% milk ® Wonder bread

Multi-level Association: Uniform Support vs. Reduced Support

- Uniform Support: the same minimum support for all levels
- + One minimum support threshold. No need to examine itemsets containing any item whose ancestors do not have minimum support
- Lower level items do not occur as frequently. If support threshold
  - too high $\Rightarrow$ miss low level associations
  - too low $\Rightarrow$ generate too many high level associations
- Reduced Support: reduced minimum support at lower levels
- There are 4 search strategies:
  - Level-by-level independent
  - Level-cross filtering by k-itemset
  - Level-cross filtering by single item
  - Controlled level-cross filtering by single item

Multi-level Association: Redundancy Filtering

- Some rules may be redundant due to —ancestor‖ relationships between items.

- Example
  - milk $\Rightarrow$ wheat bread [support = 8%, confidence = 70%]
  - 2% milk $\Rightarrow$ wheat bread [support = 2%, confidence = 72%]

- We say the first rule is an ancestor of the second rule.

- A rule is redundant if its support is close to the — expected‖ value, based on the rule's ancestor

Multi-Level Mining: Progressive Deepening

- A top-down, progressive deepening approach:
  - First mine high-level frequent items:

    milk (15%), bread (10%)
  - Then mine their lower-level —weaker‖ frequent itemsets:

    2% milk (5%), wheat bread (4%)

- Different min_support threshold across multi-levels lead to different algorithms:
  - If adopting the same *min_support* across multi-levels then toss *t* if any of *t*'s ancestors is infrequent.
  - If adopting reduced *min_support* at lower levels then examine only those descendents whose ancestor's support is frequent/non-negligible.

3.9 Mining Multidimensional Association mining

- Mining our *AllElectronics* database, we may discover the Boolean association rule

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

- Following the terminology used in multidimensional databases,

- **single- dimensional** or **intradimensional association rule** because it contains a single distinct predicate (e.g., *buys*) with multiple occurrences (i.e., the predicate occurs more than once within the rule). Such rules are commonly mined from transactional data.

- Considering each database attribute or warehouse dimension as a predicate, we can therefore mine association rules containing *multiple* predicates such as

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

- Association rules that involve two or more dimensions or predicates can be referred to as **multidimensional association rules**. Rule contains three predicates (*age, occupation*, and *buys*), each of which occurs *only once* in the rule. Hence, we say that it has **no repeated predicates**.

- Multidimensional association rules with no repeated predicates are called **interdimensional association rules**. We can also mine multidimensional association rules with repeated predicates, which contain multiple occurrences of some predicates. These rules are called **hybrid-dimensional association rules**.

- An example of such a rule is the following, where the predicate *buys* is repeated:

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

- Considering each database attribute or warehouse dimension as a predicate, we can therefore mine association rules containing *multiple* predicates such as

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

- Association rules that involve two or more dimensions or predicates can be referred to as **multidimensional association rules**. Rule contains three predicates (*age, occupation*, and *buys*), each of which occurs *only once* in the rule. Hence, we say that it has **no repeated predicates**.

- Multidimensional association rules with no repeated predicates are called **interdimensional association rules**. We can also mine multidimensional association rules with repeated predicates, which contain multiple occurrences of some predicates. These rules are called **hybrid-dimensional association rules**.

- An example of such a rule is the following, where the predicate *buys* is repeated:

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

- Database attributes can be nominal or quantitative. The values of **nominal** (or cate- gorical) attributes are "names of things." Nominal attributes have a finite number of possible values, with no ordering among the values (e.g., *occupation, brand*, *color*).

- **Quantitative** attributes are numeric and have an implicit ordering among values (e.g., *age, income*, *price*). Techniques for mining multidimensional association rules can be categorized into two basic approaches regarding the treatment of quantitative attributes.

- In the first approach, *quantitative attributes are discretized using predefined concept hierarchies*. This discretization occurs before mining. For instance, a concept hierarchy for *income* may be used to replace the original numeric values of this attribute by inter- val labels such as "0..20K," "21K..30K," "31K..40K," and so on.

- Here, discretization is *static* and predetermined. Chapter 3 on data preprocessing gave several techniques for discretizing numeric attributes. The discretized numeric attributes, with their interval labels, can then be treated as nominal attributes (where each interval is considered a category).

Mining Quantitative Association Rules

- Determine the number of partitions for each quantitative attribute
- Map values/ranges to consecutive integer values such that the order is preserved
- Find the support of each value of the attributes, and combine when support is less than MaxSup. Find frequent itemsets, whose support is larger than MinSup
- Use frequent set to generate association rules
- Pruning out uninteresting rules

Partial Completeness

- R : rules obtained before partition
- R': rules obtained after partition
- Partial Completeness measures the maximum distance between a rule in R and its closest generalization in R'

- $\hat{X}$ is a generalization of itemset X: if

$$\forall\, x \in \text{attributes}\ (X)\ [< x, l, u > \in X \wedge x, l', u' > \in \hat{X} \Rightarrow l' \leq l \leq u \leq u']$$

- The distance is defined by the ratio of support

K-Complete

- *C* : the set of frequent itemsets
- For any K ≥ 1, *P* is K-complete w.r.t *C* if:
    1. *P* *C*
    2. For any itemset *X* (or its subset) in *C*, there exists a generalization whose support is no more than *K* times that of *X* (or its subset)
- The smaller K is, the less the information lost

3.10 Correlation Analysis

- Interest (correlation, lift)
    - taking both P(A) and P(B) in consideration
    - P(A^B)=P(B)*P(A), if A and B are independent events
    - A and B negatively correlated, if the value is less than 1; otherwise A and B positively correlated

X2 Correlation

- X2 measures correlation between categorical attributes

X	1	1	1	1	0	0	0	0
Y	1	1	0	0	0	0	0	0
Z	0	1	1	1	1	1	1	1

Itemset	Support	Interest
X,Y	25%	2
X,Z	37.50%	0.9
Y,Z	12.50%	0.57

$$x = {}^2 = \sum \frac{(\text{observed\_expected})^2}{\text{expected}}$$

	game	not game	sum(row)
video	4000(4500)	3500(3000)	7500
not video	2000(1500)	500 (1000)	2500
sum(col.)	6000	4000	10000

- expected(i,j) = count(row i) * count(column j) / N
- X2 = (4000 - 4500)2 / 4500 - (3500 - 3000)2 / 3000 - (2000 - 1500)2 / 1500 - (500 - 1000)2 / 1000 = 555.6
- X2 > 1 and observed value of (game, video) < expected value, there is a negative correlation

Numeric correlation

- Correlation concept in statistics
  - Used to study the relationship existing between 2 or more numeric variables
  - A correlation is a measure of the linear relationship between variables Ex: number of hours spent studying in a class with grade received
  - Outcomes:
    - → positively related
    - → Not related
    - → negatively related
  - Statistical relationships
    - Covariance
    - Correlation coefficient

3.11 Constraint-Based Association Mining

- Interactive, exploratory mining giga-bytes of data?
  - Could it be real? — Making good use of constraints!

- What kinds of constraints can be used in mining?
  - Knowledge type constraint: classification, association, etc.
  - Data constraint: SQL-like queries
    - Find product pairs sold together in Vancouver in Dec.'98.

- Dimension/level constraints:
  - in relevance to region, price, brand, customer category.

- Rule constraints
  - small sales (price < \$10) triggers big sales (sum > \$200).

- Interestingness constraints:
  - strong rules (min_support ≥ 3%, min_confidence ≥ 60%).

Rule Constraints in Association Mining

- Two kind of rule constraints:
  - Rule form constraints: meta-rule guided mining.
  - P(x, y) ^ Q(x, w) ® takes(x, —database systems||).

- Rule (content) constraint: constraint-based query optimization (Ng, et al., SIGMOD'98).
  - sum(LHS) < 100 ^ min(LHS) > 20 ^ count(LHS) > 3 ^ sum(RHS) > 1000

- 1-variable vs. 2-variable constraints (Lakshmanan, et al. SIGMOD'99):
  - 1-var: A constraint confining only one side (L/R) of the rule, e.g., as shown above.
  - 2-var: A constraint confining both sides (L and R).
    - sum(LHS) < min(RHS) ^ max(RHS) < 5* sum(LHS)

Constrain-Based Association Query

- Database: (1) trans (TID, Itemset), (2) itemInfo (Item, Type, Price)
- A constrained asso. query (CAQ) is in the form of {(*S1, S2*)|*C* },
  - where C is a set of constraints on S1, S2 including frequency constraint

- A classification of (single-variable) constraints:
  - Class constraint: $S \subset A$. *e.g.* $S \subset Item$
  - Domain constraint:
    - $S\ \theta\ v,\ \theta \in \{ =, \neq, <, \leq, >, \geq \}$.
    - $v\ \theta\ S$, θ *is* $\in$ *or* $\notin$ *e.g. snacks* $\notin$ *S. Type*
    - $V\ \theta\ S$, *or* $S\ \theta\ V$, $\theta \in \{\subseteq, \subset, \not\subset, =, \neq)$

      - *e.g. {snacks, sodas}* $\subseteq$ *S.Type*
  - Aggregation constraint: *agg(S)* θ *v*, where *agg* is in {*min, max, sum, count, avg*}, and $\theta \in \{ =, \neq, <, \leq, >, \geq \}$.
    - *e.g. count(S1.Type) = 1 , avg(S2.Price) < 100*

Constrained Association Query Optimization Problem

- Given a CAQ = { (*S1, S2*) / *C* }, the algorithm should be :
  - sound: It only finds frequent sets that satisfy the given constraints C
  - complete: All frequent sets satisfy the given constraints C are found

- A naïve solution:
  - Apply Apriori for finding all frequent sets, and then to test them for constraint satisfaction one by one.

- Our approach:
  - Comprehensive analysis of the properties of constraints and try to push them as deeply as possible inside the frequent set computation.

Categories of Constraints.

1. Anti-monotone and Monotone Constraints

- constraint Ca is anti-monotone iff. for any pattern S not satisfying Ca, none of the super-patterns of S can satisfy Ca
- A constraint Cm is monotone iff. for any pattern S satisfying Cm, every super-pattern of S also satisfies it

2. Succinct Constraint

- A subset of item Is is a succinct set, if it can be expressed as σp(I) for some selection predicate p, where σ is a selection operator
- SP⊆2I is a succinct power set, if there is a fixed number of succinct set I1, …, Ik ⊆ I, s.t. SP can be expressed in terms of the strict power sets of I1, …, Ik using union and minus
- A constraint Cs is succinct provided SATCs(I) is a succinct power set

3. Convertible Constraint

- Suppose all items in patterns are listed in a total order R
- A constraint C is convertible anti-monotone iff a pattern S satisfying the constraint implies that each suffix of S w.r.t. R also satisfies C
- A constraint C is convertible monotone iff a pattern S satisfying the constraint implies that each pattern of which S is a suffix w.r.t. R also satisfies C

Property of Constraints: Anti-Monotone

- Anti-monotonicity: *If a set S violates the constraint, any superset of S violates the constraint.*
- Examples:
  - *sum(S.Price)* $\leq$ *v* is anti-monotone
  - *sum(S.Price)* $\geq$ *v* is not anti-monotone
  - *sum(S.Price)* = *v* is partly anti-monotone
- Application:
  - Push —*sum(S.price)* $\leq$ 1000| deeply into iterative frequent set computation.

Example of Convertible Constraints: Avg(S) θ V

- Let R be the value descending order over the set of items
  - E.g. I = {9, 8, 6, 4, 3, 1}
- Avg(S) $\geq$ v is convertible monotone w.r.t. R
  - If S is a suffix of S1, avg(S1) $\geq$ avg(S)
    - {8, 4, 3} is a suffix of {9, 8, 4, 3}
    - avg({9, 8, 4, 3})=6 $\geq$ avg({8, 4, 3})=5
  - If S satisfies avg(S) $\geq$ v, so does S1
    - {8, 4, 3} satisfies constraint avg(S) $\geq$ 4, so does {9, 8, 4, 3}

Property of Constraints: Succinctness

- Succinctness:
  - For any set S1 and S2 satisfying C, S1 ∪ S2 satisfies C
  - Given A1 is the sets of size 1 satisfying C, then any set S satisfying C are based on A1 , i.e., it contains a subset belongs to A1 ,
- Example :
  - $sum(S.Price) \geq v$ is not succinct
  - $min(S.Price) \leq v$ is succinct

Optimization

- If C is succinct, then C is pre-counting prunable. The satisfaction of the constraint alone is not affected by the iterative support counting.
  - ed based on the training set

    Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

THANK YOU