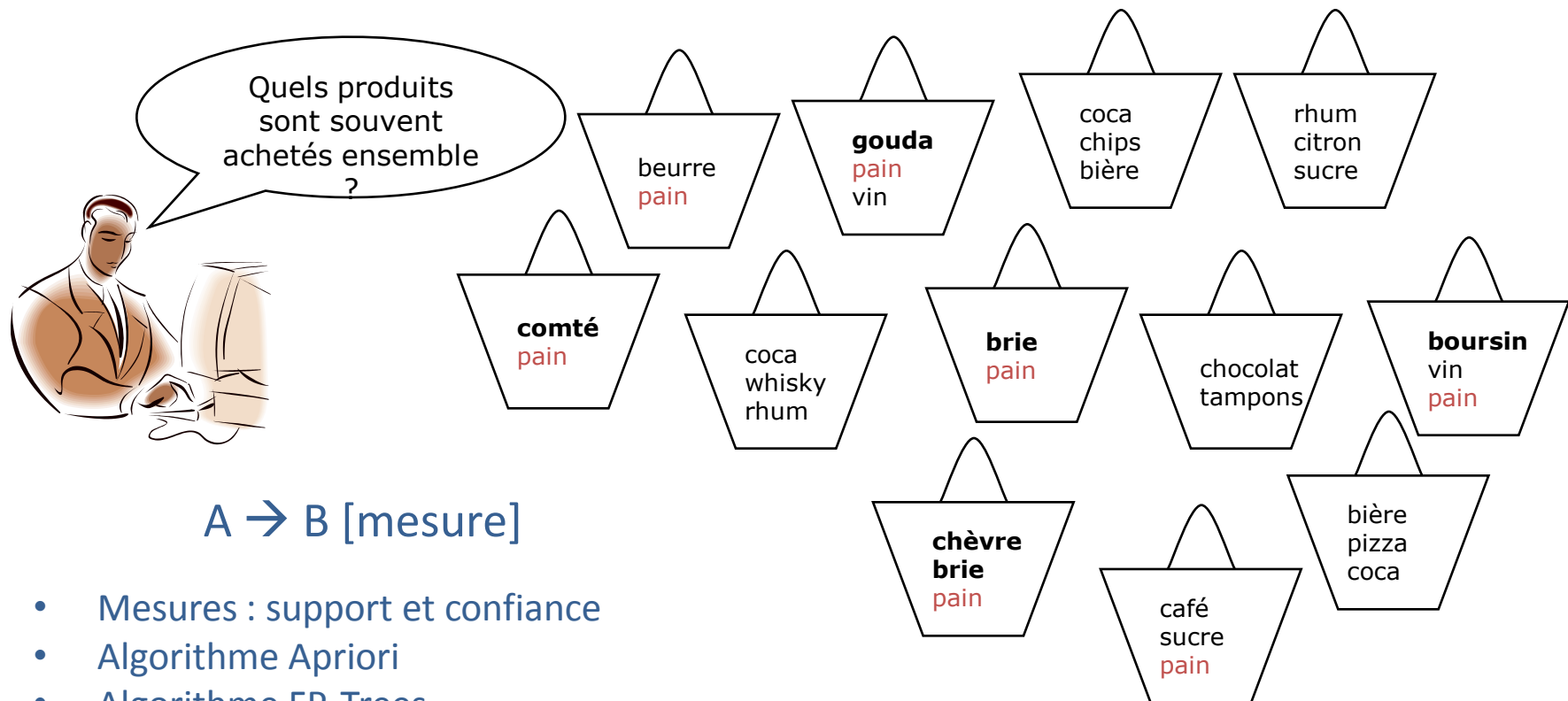


# Règles d'associations



$A \rightarrow B$  [mesure]

- Mesures : support et confiance
- Algorithme Apriori
- Algorithme FP-Trees
- Règles booléennes (présence/absence) / quantitatives ( $30 < \text{âge} < 40$ )
- Règles multi-niveaux ( $\text{brie} < \text{fromage} < \text{produits laitiers}$ )
- Associations multidimensionnelles (âge, revenus, achats)
- Autres mesures
- Exploration avec contraintes

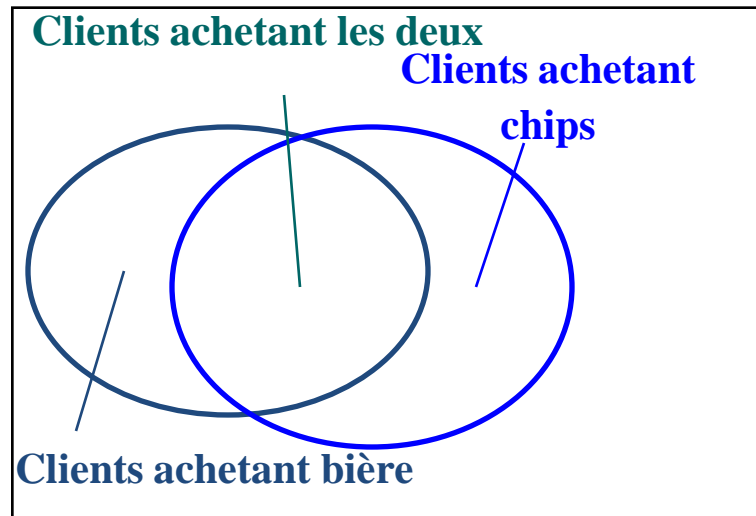
- Règles d'association :
  - ♦ motifs de la forme : Corps  $\rightarrow$  Tête
  - ♦ Exemple :  
 $\text{achète}(x, \text{"cacahuètes"}) \rightarrow \text{achète}(x, \text{"bière"})$
- Étant donnés :
  1. une base de transactions  $D, I = \{i_1, i_2, \dots, i_n\}$
  2. chaque transaction est décrite par un identifiant TID et une liste d'items  $T_{\text{TID}} = \{i_1, i_2, \dots, i_m\} \subseteq I$

une transaction contient  $A \Leftrightarrow A \subseteq T$

règle  $A \rightarrow B / A \subset I, \text{ et } B \subset I \text{ et } A \cap B = \emptyset$

$\rightarrow$  Trouver: toutes les règles qui expriment une corrélation entre la présence d'un item avec la présence d'un ensemble d'items

Ex : *98% des personnes qui achètent des chips achètent de la bière*



Trouver les règles  $X \& Y \Rightarrow Z$  avec un support  $> s$  et une confiance  $> c$

- ♦ support,  $s$ , probabilité qu'une transaction contienne  $\{X, Y, Z\}$
- ♦ confiance,  $c$ , probabilité conditionnelle qu'une transaction qui contient  $\{X, Y\}$  contienne aussi  $Z$

$$\text{Confiance} = \text{support}(X, Y, Z) / \text{support}(X, Y)$$

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

Soit support minimum 50%, et confiance minimum 50%,

$A \Rightarrow C$  (50%, 66.6%)

$C \Rightarrow A$  (50%, 100%)

Pour  $A \Rightarrow C$ :

support = support( $\{A, C\}$ ) = 50%

confiance = support( $\{A, C\}$ )/support( $\{A\}$ ) = 66.6%

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

Min. support 50%  
Min. confiance 50%

Itemsets fréquents	Support
{A}	75%
{B}	50%
{C}	50%
{A,C}	50%

## Extraction des associations : 1<sup>ère</sup> approche (naïve)

- D: une base de transactions
- I: ensemble de tous les items avec  $|I|=n$

**pour chaque** ensemble d'items possible

calculer son support

**si** support  $\geq$  min\_support

**pour chaque** règle  $r: A_1, \dots, A_{m-1} \rightarrow A_m$  t.q  $J = \{A_1, \dots, A_m\}$

**si** confiance(r)  $\geq$  min\_confiance

afficher r

Bien sûr, en pratique, il y a trop de sous-ensembles de I possibles :  $2^n$

- On suppose :
  - ♦  $10^5$  items
  - ♦  $10^7$  transactions
  - ♦ En moyenne chaque transaction concerne 10 items.
- Pour trouver les paires d'items fréquents (2-itemsets) :
  - ♦ Pour chaque transaction on a  $C_{10}^2 = 45$  paires à considérer :  $45 \times 10^7 = 450$  millions

- Principe : Si un ensemble est non fréquent, alors tous ses sur-ensembles ne sont pas fréquents
  - Si  $\{A\}$  n'est pas fréquent alors  $\{A,B\}$  ne peut pas l'être
  - si  $\{A,B\}$  est fréquent alors  $\{A\}$  et  $\{B\}$  le sont
- Itérativement, trouver les itemsets fréquents dont la cardinalité varie de 1 à k (k-itemset)
- Utiliser les itemsets fréquents pour générer les règles d'association

## L'algorithme Apriori

- Étape de jointure:  $C_k$  est généré en joignant  $L_{k-1}$  avec lui même
- Étape d'élimination: Chaque  $(k-1)$ -itemset qui n'est pas fréquent ne peut être un sous ensemble d'un  $k$ -itemset fréquent

**$C_k$ : Itemset candidat de taille  $k$ ,**

**$L_k$  : itemset fréquent de taille  $k$**

$L_1 = \{\text{items fréquents}\}$

**pour** ( $k = 1$ ;  $L_k \neq \emptyset$ ;  $k++$ )

$C_{k+1}$  = candidats générés à partir de  $L_k$  % jointure

**pour chaque** transaction  $t$  dans la base

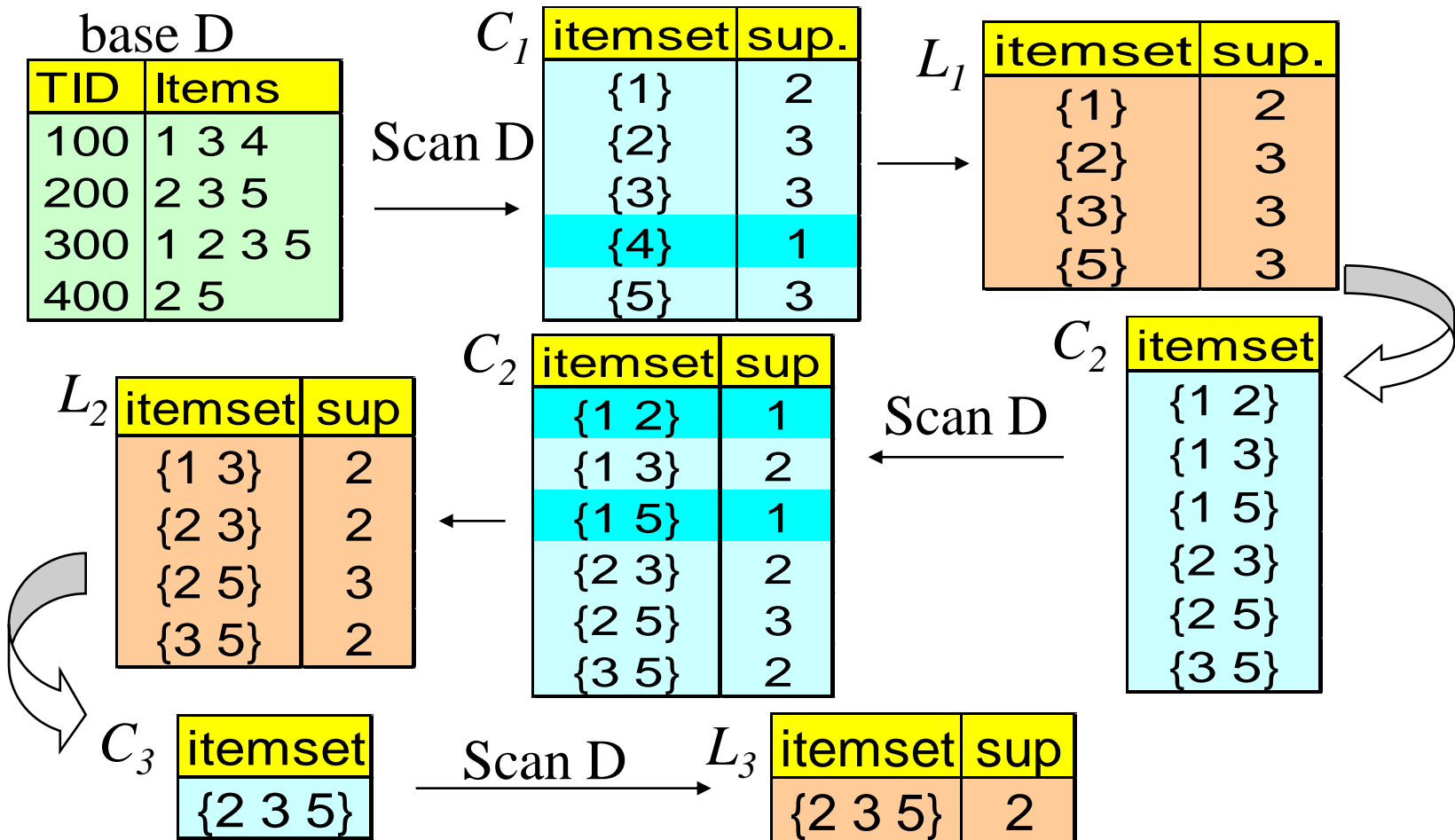
**incrémenter le COUNT des candidats de  $C_{k+1}$  qui sont dans  $t$**

$L_{k+1}$  = candidats dans  $C_{k+1}$  dont COUNT > support\_min

**renvoyer**  $\cup_k L_k$



Avec  $\text{min\_support}=2$



- $L_3 = \{abc, abd, acd, ace, bcd\}$
- jointure :  $C_4 = L_3 * L_3$ 
  - ♦  $abcd$  à partir de  $abc$  et  $abd$
  - ♦  $acde$  à partir  $acd$  et  $ace$
  - ♦  $C_4 = \{abcd, acde\}$
- Elagage :
  - ♦  $acde$  est supprimé car  $ade$  n'est pas dans  $L_3$
- $C_4 = \{abcd\}$

Confiance ( $A \rightarrow B$ ) =  $\text{support}(AB) / \text{support}(A)$

Algorithme :

**pour** chaque itemset fréquent  $f$   
**pour** chaque sous ensemble  $s \subset f$ , avec  $s \neq \emptyset$   
     **si**  $\text{confiance}(s \rightarrow f \setminus s) > \text{min\_conf}$   
         **alors** afficher( $s \rightarrow f \setminus s$ )  
**fin pour**  
**fin pour**

Items	count
1	2
2	3
3	3
5	3

Items	count
1, 3	2
2, 3	2
2, 5	3
3, 5	2
2, 3, 5	2

$\text{min\_sup}=50\%$   
 $\text{min\_conf}=75\%$

Règles	Conf.
$1 \rightarrow 3$	100%
$3 \rightarrow 1$	66%
$2 \rightarrow 3$	66%
$3 \rightarrow 2$	66%
$2 \rightarrow 5$	100%
$5 \rightarrow 2$	100%
$3 \rightarrow 5$	66%
$5 \rightarrow 3$	66%
$2, 3 \rightarrow 5$	100%
$2, 5 \rightarrow 3$	66%
$3, 5 \rightarrow 2$	100%
$2 \rightarrow 3, 5$	66%
$3 \rightarrow 2, 5$	100%
$5 \rightarrow 2, 3$	66%

- Le principe de l'algorithme:
  - ♦ Utiliser les  $(k - 1)$ -itemsets fréquents pour générer les k-itemsets candidats
  - ♦ Scanner la base pour tester le support des candidats
- Point faible : génération des candidats
  - ♦ Beaucoup :
    - $10^4$  1-itemsets fréquents générant  $\sim 5 \cdot 10^7$  paires d'items candidates
    - Pour trouver les 100-itemsets, on doit générer  $2^{100} \approx 10^{30}$  candidats.
  - ♦ Plusieurs scans de la base:
    - On doit faire  $(n + 1)$  scans, pour trouver les n-itemsets fréquents

- Compresser la base, Frequent-Pattern tree (FP-tree)
  - ♦ Une représentation condensée
  - ♦ Évite les scans coûteux de la base
- Développer une méthode efficace pour l'exploration basée sur une approche
  - ♦ diviser-pour-régner: décompose le problèmes en sous-problèmes
  - ♦ Pas de génération de candidats : test de la "sous-base" seulement !

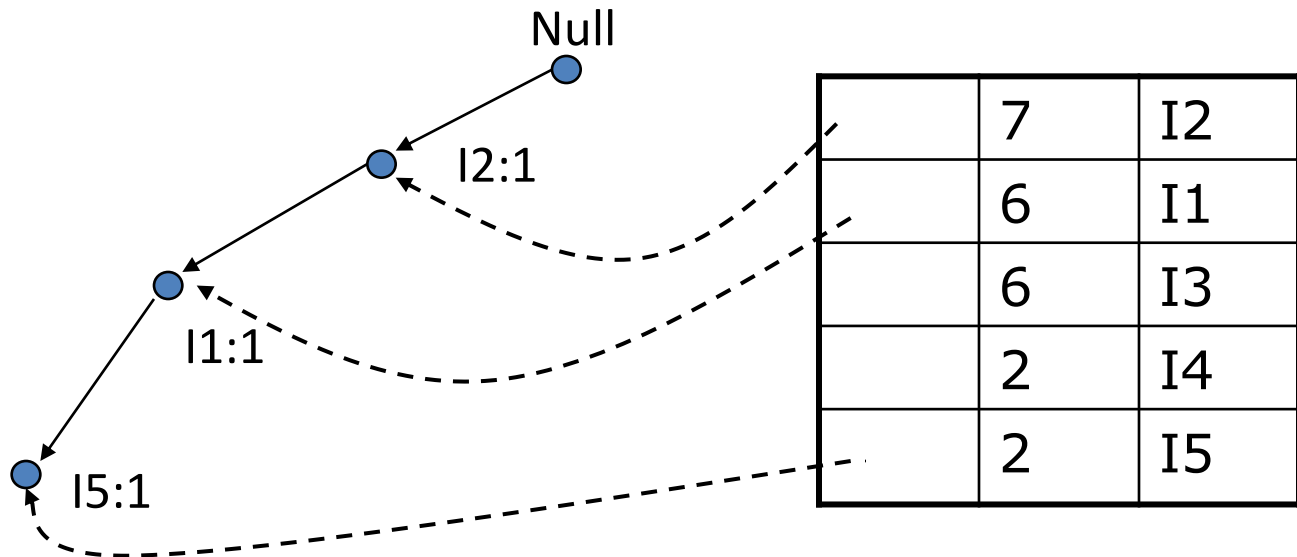
# FP-Trees : exemple

TID	T100	T200	T300	T400	T500	T600	T700	T800	T900
Liste items	I1, I2, I5	I2, I4, I6	I1, I3	I1, I2, I4	I2, I3, I8	I2, I3	I1, I3, I7	I1, I2, I3, I5	I1, I2, I3

Supposons que min-support=2. On construit la liste « triée »:

$L = [I2:7, I1:6, I3:6, I4:2, I5:2]$  Min support = 2, on ne s'intéresse qu'aux  $I \leq 2$  apparitions dans le tableau

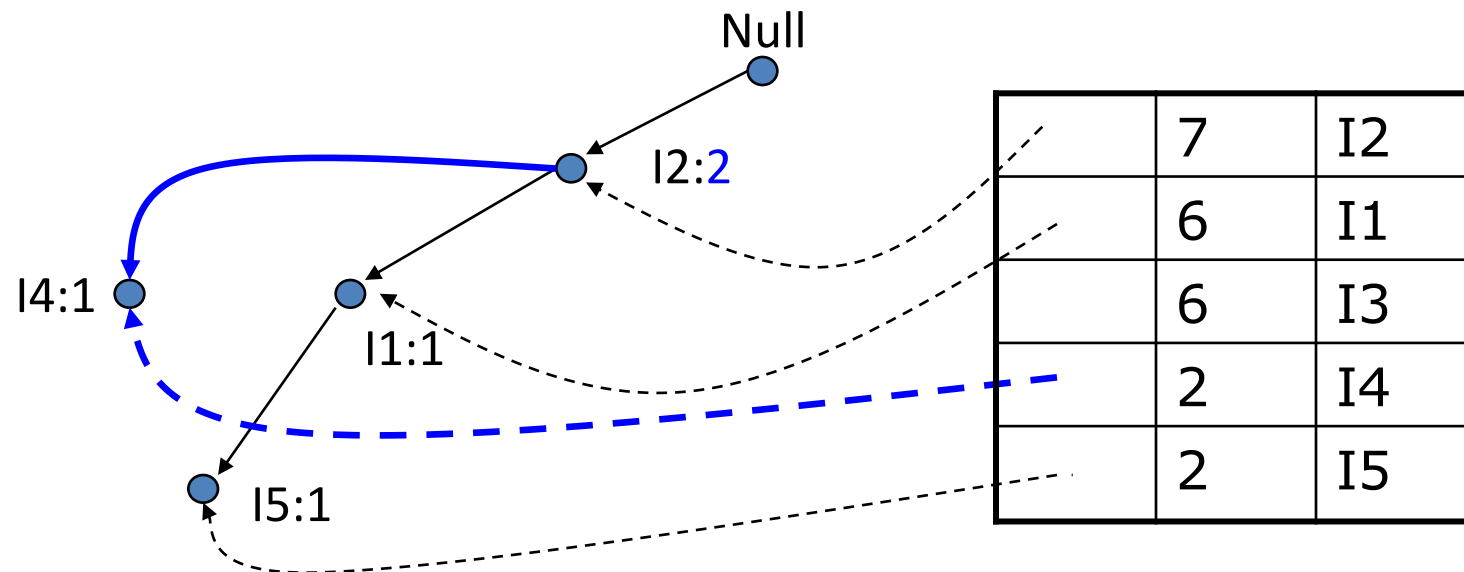
On parcourt une 2<sup>ème</sup> fois la base. On lit les transactions selon l'ordre des items dans L : pour T100 on a I2, I1, I5. La lecture de T100 donne



# FP-Trees : exemple

TID	T100	T200	T300	T400	T500	T600	T700	T800	T900
Liste items	I1, I2, I5	I2, I4, I6	I1, I3	I1, I2, I4	I2, I3, I8	I2, I3	I1, I3, I7	I1, I2, I3, I5	I1, I2, I3

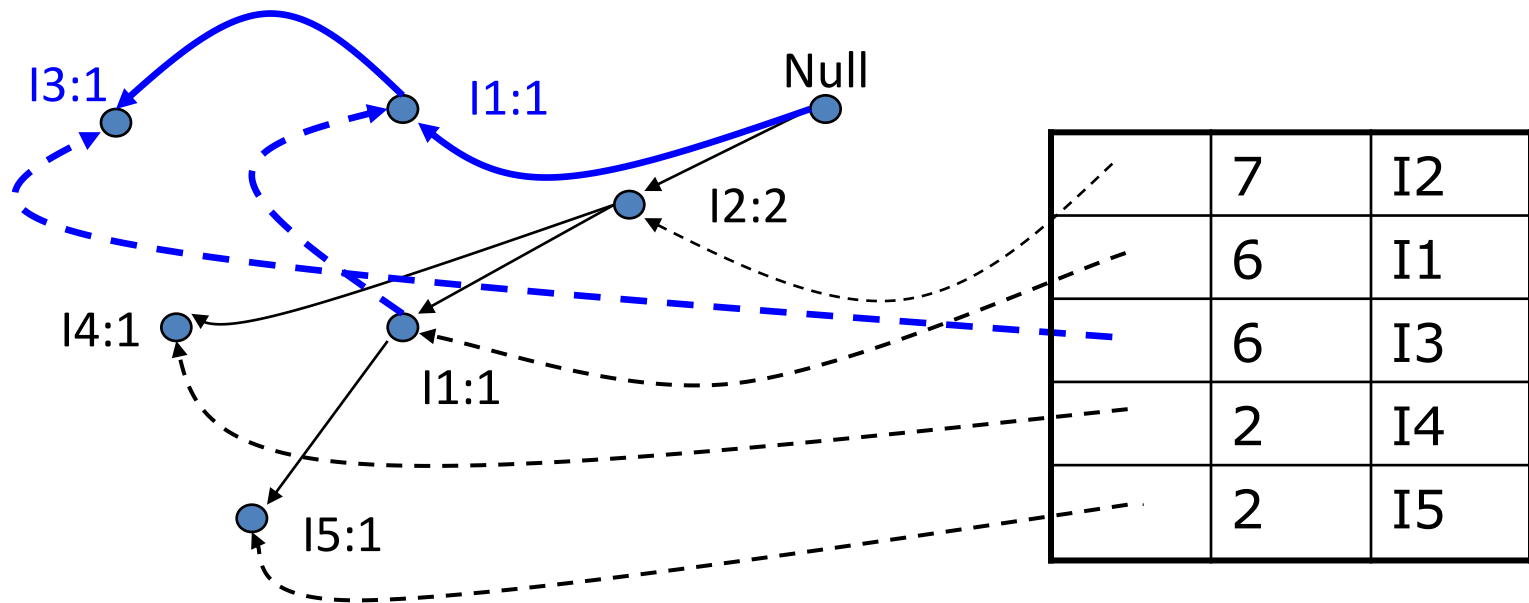
La lecture de T200 va *a priori* générer une branche qui relie la racine à I2 et I2 à I4. Or cette branche partage **un préfixe** (i.e I2) avec une branche qui existe déjà. L'arbre obtenu après lecture de T200 sera



# FP-Trees : exemple

TID	T100	T200	T300	T400	T500	T600	T700	T800	T900
Liste items	I1, I2, I5	I2, I4, I6	I1, I3	I1, I2, I4	I2, I3, I8	I2, I3	I1, I3, I7	I1, I2, I3, I5	I1, I2, I3

En lisant T300, l'ordre selon L est I1, I3. Ceci nous amène à ajouter une branche Null → I1 → I3. Noter qu'elle n'a pas de préfixe commun avec ce qui existe déjà. On obtient

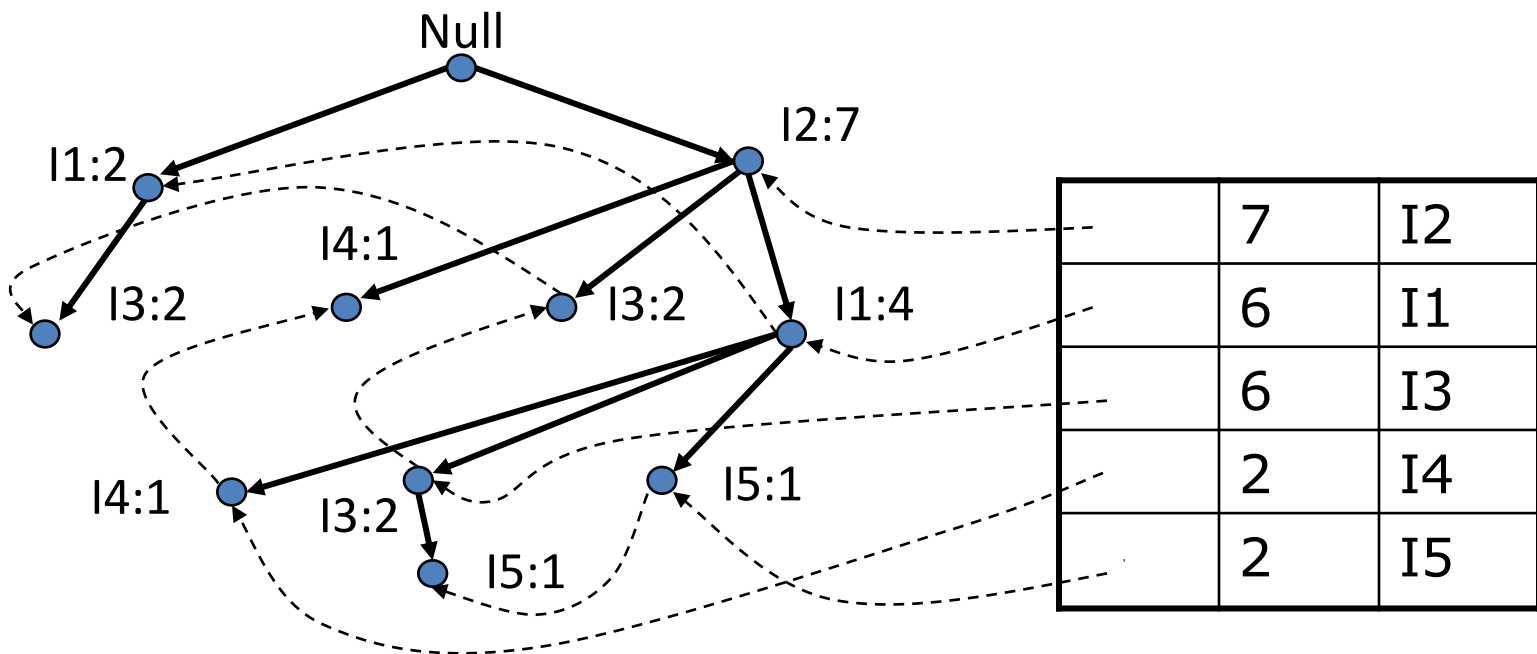




# FP-Trees : exemple

TID	T100	T200	T300	T400	T500	T600	T700	T800	T900
Liste items	I1, I2, I5	I2, I4, I6	I1, I3	I1, I2, I4	I2, I3, I8	I2, I3	I1, I3, I7	I1, I2, I3, I5	I1, I2, I3

Finalemt, le FP-tree obtenu est



- Considérons I5. Il apparaît dans 2 branches.  $I2 \rightarrow I1 \rightarrow I5:1$  et  $I2 \rightarrow I1 \rightarrow I3 \rightarrow I5:1$
- Ainsi, pour le suffixe I5, on a 2 chemins préfixes:  $\langle I2, I1:1 \rangle$  et  $\langle I2, I1, I3:1 \rangle$ . Ils forment sa «table conditionnelle»

TiD	Itemset
1	I2, I1
2	I2, I3, I1

- Le «FP-tree conditionnel» de I5 contient une seule branche  $I2 \rightarrow I1$ . I3 n'en fait pas partie car son support est 1 qui est  $< 2$  (Rappel:  $\text{min\_support}=2$ )
- Ce chemin unique va générer toutes les combinaisons de I5 avec I1 et I2, i.e  $\{I1, I5\}:2$ ,  $\{I2, I5\}:2$ ,  $\{I1, I2, I5\}:2$

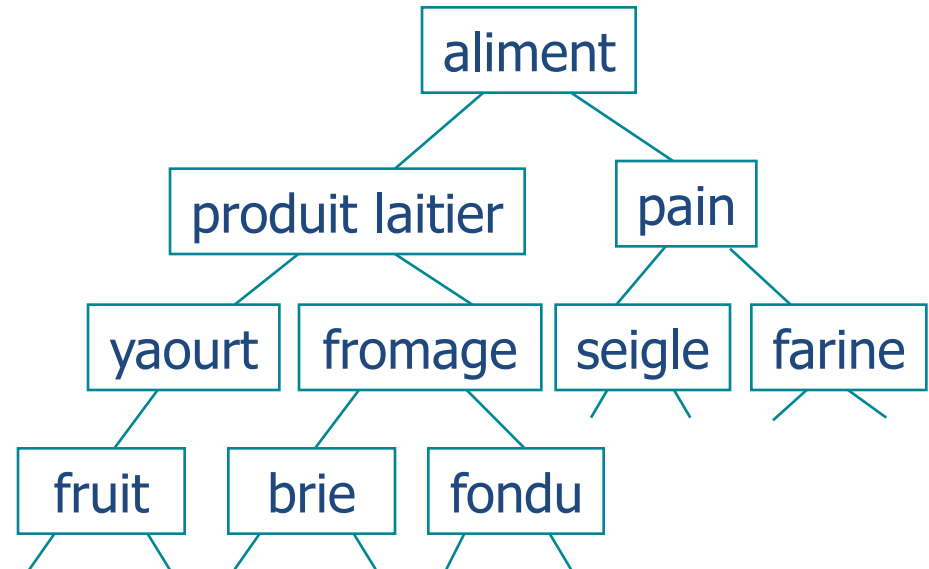
- Considérons I4. Sa table conditionnelle est formée de  $\langle I2, I1:1 \rangle$  et  $\langle I2:1 \rangle$
- Le FP-Tree conditionnel ne contient donc qu'un seul nœud I2
- Nous obtenons donc un itemset fréquent qui est  $\{I2, I4\}:2$

Item	Base conditionnelle	FP-tree conditionnel	Itemsets générés
I5	$\langle I2, I1 \rangle : 1, \langle I2, I1, I3 \rangle : 1$	$I2:2 \rightarrow I1:2$	$\{I2, I5\} : 2$ $\{I1, I5\} : 2$ $\{I2, I1, I5\} : 2$
I4	$\langle I2, I1 \rangle : 1, \langle I2 \rangle : 1$	$I2:2$	$\{I2, I4\} : 2$
I3	$\langle I2, I1 \rangle : 2, \langle I2 \rangle : 2, \langle I1 \rangle : 2$	$I2:4 \rightarrow I1:2$ $I1:2$	$\{I2, I3\} : 4$ $\{I1, I3\} : 4$ $\{I2, I1, I3\} : 2$
I1	$\langle I2 \rangle : 4$	$I2:4$	$\{I2, I1\} : 4$

Ce n'est pas la peine de regarder I2 car ça va donner les combinaisons avec les autres items qui ont déjà été considérés

## Règles d'association multi-niveaux

- Les items forment des hiérarchies.
- Les items au niveau inférieur ont des supports inférieurs
- Les bases de transactions peuvent prendre en compte les niveaux
- Item {111} représente le « yaourt fruit »



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 }

- On dispose d'une hiérarchie de concepts
- Les niveaux sont numérotés de 0 à  $n$  en commençant par le niveau le plus général
- Les concepts (ou items) de chaque niveau  $i$  sont numérotés de 1 à  $m_i$  avec  $m_i$  qui représente le nombre de concepts dans le niveau  $i$
- Les données dont on dispose représentent les transactions avec les items du plus bas niveau
- Avec une chaîne de  $n$  digits, on peut identifier tous les items du plus bas niveau.

- Approche Top-Down:
  - ♦ On utilise la propriété de non monotonie: Si un concept (ou ensemble de concepts) est non fréquent, alors tous ses fils sont non fréquents
  - ♦ Ex: Si produit\_laitier est non fréquent alors yaourt ne l'est certainement pas.
  - ♦ Ex: Si {produit\_laitier, pain} est non fréquent alors {yaourt, pain\_au\_seigle} ne l'est certainement pas
- Problème : difficulté de régler le support minimal



- Si  $\text{niveau}_i < \text{niveau}_j$   
alors  $\text{min-support}(i) > \text{min-support}(j)$ 
  - ♦ min-support du niveau 1 est 50% et  
min-support de niveau 2 est de 10%
- Dans ce cas, il y a différentes approches
  - ♦ Indépendance entre niveaux
  - ♦ Filtre sur un seul item
  - ♦ Filtre sur k items

- Dans ce cas, on n'utilise aucune connaissance sur les parents
  - ♦ Ce n'est pas parce qu'un parent n'est pas fréquent que le fils ne le sera pas
  - ♦ Ce n'est pas parce qu'un fils est fréquent que le parent le sera
- Le calcul peut être Top-Down ou Bottom-Up

- Un k-itemset est vérifié au niveau i
  - si les k-parents au niveau i-1 sont fréquents
    - ♦ {pain-au-seigle, yaourt} sera vérifié seulement si {pain, produit-laitier} est fréquent
- Une approche Top-Down
- Risque de perte de règles intéressantes aux niveaux inférieurs

- Un item au niveau  $i$  sera examiné si son parent est considéré comme fréquent
  - ♦ Si Pain est fréquent  
alors on peut voir si Pain-au-seigle l'est.
  - ♦ Si produit-laitier est fréquent  
alors on peut voir si yaourt l'est.
- Approche Top-Down.

- Certaines règles peuvent être redondantes à cause des relations de “parenté” entre items
- Exemple
  - ♦  $\text{Produit\_laitier} \Rightarrow \text{pain\_farine}$  [ $s = 8\%$ ,  $c = 70\%$ ]
  - ♦  $\text{fromage} \Rightarrow \text{pain\_farine}$  [ $s = 2\%$ ,  $c = 72\%$ ]
- On dit que la première règle est un ancêtre de la seconde
- Une règle est redondante si son support est très proche du **support prévu**, en se basant sur sa règle ancêtre
  - ♦ Dans l'exemple, s'il y a 4 fils du nœud  $\text{Produit\_laitier}$  dont Fromage, alors le support 2% est « prévisible »

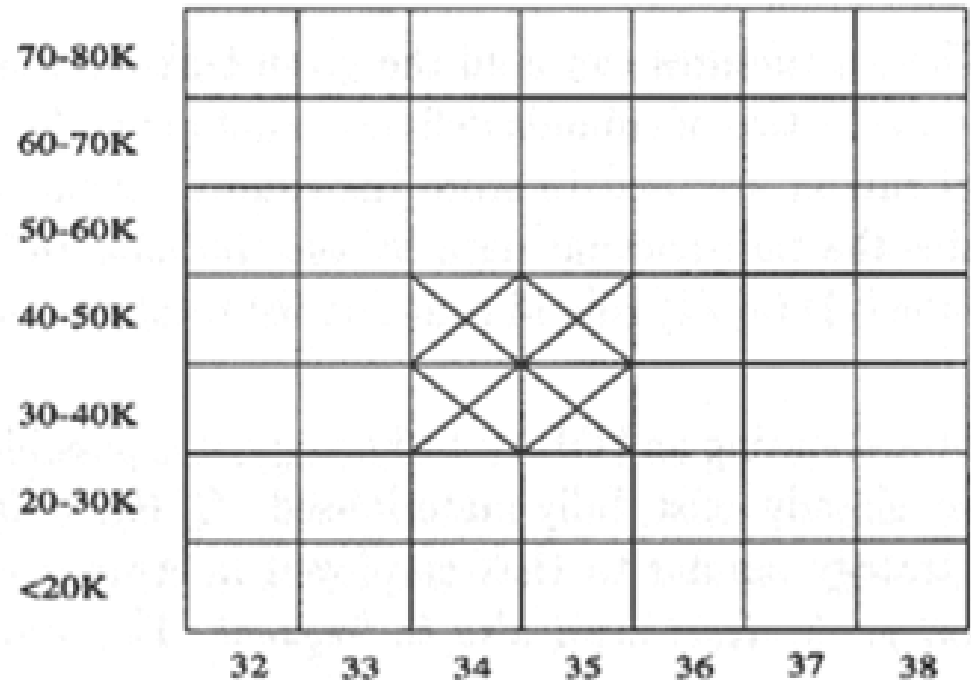
- Règles uni-dimensionnelles:
  - $\text{achète}(X, \text{"lait"}) \Rightarrow \text{achète}(X, \text{"pain"})$
- Règles multi-dimensionnelles: 2 dimensions ou prédicats
  - ♦ Règles inter-dimensions (pas de prédicats répétés)
    - $\text{age}(X, \text{"19-25"}) \wedge \text{occupation}(X, \text{"étudiant"}) \Rightarrow \text{achète}(X, \text{"Cola"})$
  - ♦ Règles hybrides (prédicats répétés)
    - $\text{age}(X, \text{"19-25"}) \wedge \text{achète}(X, \text{"popcorn"}) \Rightarrow \text{achète}(X, \text{"Cola"})$
- Attributs de catégorie
  - ♦ Un nombre fini de valeurs, pas d'ordre entre les valeurs
- Attributs quantitatifs
  - ♦ numériques, il existe un ordre (implicite) entre les valeurs

- Si tous les attributs sont catégoriels, on peut se ramener au cas classique
  - ♦ Achète(T100,Bob,cola), Achète(T100,Bob,popcorn), Occupation(Bob,étudiant) est remplacé par la transaction  $T=\{\text{cola,popcorn,étudiant}\}$
  - ♦ Si  $I$  est fréquent et si la règle  $\text{cola, étudiant} \rightarrow \text{popcorn}$  a une confiance suffisante alors la règle multidimensionnelle
  - ♦  $\text{Achète}(X,\text{cola}), \text{occupation}(X,\text{étudiant}) \rightarrow \text{Achète}(X, \text{popcorn})$  est générée

- Chercher les ensembles à k-prédicats fréquents:
  - ♦ Exemple: {âge, occupation, achète} est un ensemble à 3 prédicats.
  - ♦ Le seul attribut quantitatif est âge. Les techniques peuvent être distinguées sur le mode de traitement de l'attribut âge.
- 1. Discrétisation statique
  - ♦ Remplacer les valeurs d'âge par des intervalles 0..20, 21..40, ...Chaque intervalle devient donc une catégorie.
- 2. Règles quantitatives
  - ♦ Les attributs quantitatifs sont dynamiquement discrétisés en se basant sur la distribution des données.
- 3. Règles basées sur une notion de Distance
  - ♦ C'est un processus de discrétisation dynamique basé sur la distance entre les données



- Les attributs numériques sont dynamiquement discrétisés de sorte à maximiser la confiance ou la compacité des règles sont maximisées
- Considérons le cas des règles:  $A_{\text{quan1}} \wedge A_{\text{quan2}} \wedge A_{\text{cat1}}$
- Regrouper les règles “adjacentes” pour former des règles générales en utilisant une grille 2-D
- Exemple:  
 $\text{age}(X, "34..35") \wedge$   
 $\text{revenu}(X, "31K..50K")$   
 $\Rightarrow \text{achète}(X, "TV \text{ grand écran} ")$



- C'est une sorte de regroupement.
- Ex: la distance entre éléments d'un groupe inférieure à 4

Prix	equi-largeur (10)	equi-profondeur (2)	Distance
7	[0,10]	[7,20]	[7,7]
20	[11,20]	[22,50]	[20,22]
22	[21,30]	[51,53]	[50,53]
50	[31,40]		
51	[41,50]		
53	[51,60]		

- La discrétisation est ici basée sur la proximité des éléments d'un intervalle en tenant compte des données réelles

- Mesures objectives:
  - ♦ *support* et
  - ♦ *confiance*
- Mesures subjectives
  - ♦ Une règle est intéressante
  - ♦ Si elle est *inattendue* et/ou
  - ♦ *actionnable* (l'utilisateur peut faire quelque chose avec)

## Critiques des notions de support et de confiance

- Parmi 5000 étudiants
  - ♦ 3000 jouent au basket
  - ♦ 3750 prennent des céréales
  - ♦ 2000 jouent au basket et prennent des céréales
- *Jouer au basket* → *prendre des céréales* [40%, 66.7%] n'est pas informative car il y a 75% d'étudiants qui prennent des céréales ce qui est plus que 66.7%.
- *jouer au basket* → *pas de céréales* [20%, 33.3%] est plus pertinente même avec un support et une confiance inférieurs

	basket	non basket	$\Sigma$
céréales	2000	1750	3750
non céréale	1000	250	1250
sum(col.)	3000	2000	5000

- Exemple 2:
  - X et Y: positivement corrélés,
  - X et Z, négativement corrélés
  - Les support et confiance de  $X \rightarrow Z$  dominant

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

- Nous avons besoin d'une mesure de corrélation

$$corr_{A,B} = \frac{P(AB)}{P(A)P(B)}$$

- est aussi appelé le **lift** de  $A \Rightarrow B$

Règle	Support	Confiance
$X \rightarrow Y$	25%	50%
$X \rightarrow Z$	37,50%	75%

- Intérêt (corrélation)  $\frac{P(A \wedge B)}{P(A)P(B)}$ 
  - ♦ Prendre en compte  $P(A)$  et  $P(B)$
  - ♦  $P(A \& B) = P(B) * P(A)$ , si A et B sont des événements indépendants
  - ♦ A et B négativement corrélés, si  $\text{corr}(A, B) < 1$ .

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	Intérêt
X,Y	25%	2
X,Z	37,50%	0,9
Y,Z	12,50%	0,57

- Exploration interactive où l'utilisateur pose des conditions en plus des minima de support et confiance
- Quels types de conditions ?
  - ♦ Type de connaissance recherchée: classification, association, etc.
  - ♦ Contraintes sur les données:
    - Trouver les paires de produits vendus à Toulouse en Décembre 98
  - ♦ Contraintes sur l'intérêt
    - support, confiance, corrélation
  - ♦ Contraintes sur les dimensions:
    - En rapport à région, prix, marque, catégorie client
  - ♦ Contraintes sur les règles
    - Nombres de prédicats dans le corps

- Exemple, Base: (1) trans (TID, Itemset ),  
(2) itemInfo (Item, Type, Prix)
- Une requête d'association contrainte (RAC) est une expression de la forme  $\{(S_1 \rightarrow S_2) / C\}$ ,
  - ♦ où  $C$  est un ensemble de contraintes sur  $S_1$  et  $S_2$  incluant la contrainte de fréquence
- Une classification de contraintes (à une variable) :
  - ♦ Contraintes de classe :  
*ex.  $S_2 \subset I_1$*
  - ♦ Contrainte de domaine :  
*ex.  $S_2.Prix < 100$*
  - ♦ Contraintes d'agrégation :  
*ex.  $avg(S_2.Prix) < 100$*



- Soit une  $RAC = \{ (S1 \rightarrow S2) / C \}$ , l'algorithme doit être:
  - ♦ **Correct** : Il ne trouve que les itemsets fréquents qui satisfont C
  - ♦ **Complet** : Il trouve tous les itemsets fréquents qui satisfont C
- Solution naïve :
  - ♦ Appliquer A priori pour trouver les itemsets fréquents puis éliminer les itemsets ne satisfaisant pas C
- Autre approche :
  - ♦ Analyser les propriétés des contraintes pour les intégrer dans A priori lors de la phase de l'exploration des itemsets fréquents.

- Une contrainte  $C_a$  est **anti-monotone** ssi pour chaque itemset  $S$ , si  $S$  ne satisfait pas  $C_a$ , alors aucun de ses sur-ensembles ne satisfait  $C_a$
- $C_m$  est **monotone** ssi pour chaque  $S$ , si  $S$  satisfait  $C_m$ , alors chacun de ses sur-ensembles satisfait  $C_m$

- Anti-monotonie : *Si  $S$  viole la contrainte alors chaque sur-ensemble de  $S$  viole aussi la contrainte*
- Exemples :
  - ♦  $sum(S.Prix) \leq v$  est anti-monotone
  - ♦  $sum(S.Prix) \geq v$  n'est pas anti-monotone
  - ♦  $sum(S.Prix) = v$  est partiellement anti-monotone
- Application :
  - ♦ Pousser la condition " $sum(S.prix) \leq 1000$ " lors des itérations du calcul des ensembles fréquents

- $C: \text{sum}(S.\text{Prix}) = v$  est **partiellement** anti-monotone dans le sens suivant :
  - ♦  $C$  est équivalente à
$$C': \text{sum}(S.\text{Prix}) \leq v \text{ et } C'': \text{sum}(S.\text{Prix}) \geq v$$
  - ♦ Si  $S$  ne satisfait pas  $C$  alors une des deux conditions de la conjonction est fausse
  - ♦ Si c'est  $C''$  qui est fausse, alors on peut faire des conclusions sur les  $S' \supset S$

- Le problème avec ces contraintes est que l'on ne peut pas les intégrer lors de l'évaluation des itemsets fréquents
- Exemples
  - ♦  $\text{Sum}(S.\text{prix}) \geq 100$  est monotone
  - ♦  $\text{Sum}(S.\text{prix}) \leq v$  n'est pas monotone
- Par contre, si  $S' \supset S$  et  $\text{Sum}(S.\text{prix}) \geq 100$  alors ce n'est pas la peine de tester  $\text{Sum}(S'.\text{prix})$

- Une contrainte est succincte si on peut générer statiquement tous les itemsets qui la satisfont
- Exemple :
  - ♦  $A \in S$  avec  $I = \{A, B, C, D\}$
  - ♦ L'ensemble des itemsets  $S$  qui satisfont la condition sont tous les sous-ensembles de  $I$  qui contiennent  $A$

$S \theta v, \theta \in \{=, \leq, \geq\}$	<b>oui</b>
$v \in S$	<b>oui</b>
$S \supseteq V$	<b>oui</b>
$S \subseteq V$	<b>oui</b>
$S = V$	<b>oui</b>
$\min(S) \leq v$	<b>oui</b>
$\min(S) \geq v$	<b>oui</b>
$\min(S) = v$	<b>oui</b>
$\max(S) \leq v$	<b>oui</b>
$\max(S) \geq v$	<b>oui</b>
$\max(S) = v$	<b>oui</b>
$\text{count}(S) \leq v$	<b>faiblement</b>
$\text{count}(S) \geq v$	<b>faiblement</b>
$\text{count}(S) = v$	<b>faiblement</b>
$\text{sum}(S) \leq v$	<b>non</b>
$\text{sum}(S) \geq v$	<b>non</b>
$\text{sum}(S) = v$	<b>non</b>
$\text{avg}(S) \theta v, \theta \in \{=, \leq, \geq\}$	<b>non</b>
(contrainte de fréquence)	<b>non</b>

- Supposons que tous les items dans les motifs soient triés selon l'ordre  $O$
- Une contrainte  $C$  est **convertible anti-monotone** ssi un motif  $S$  satisfait  $C$  implique que chaque suffixe de  $S$  (respectivement à  $O$ ) satisfait aussi  $C$
- Une contrainte  $C$  est **convertible monotone** ssi un motif  $S$  satisfait  $C$  implique que chaque motif dont  $S$  est un suffixe (respectivement à  $O$ ) satisfait aussi  $C$



- Soit  $S$  l'ensemble de valeurs (par ordre décroissant)  $\{9, 8, 6, 4, 3, 1\}$
- $\text{Avg}(S) \geq v$  est monotone convertible respectivement à  $S$ 
  - ♦ Si  $S$  est un suffixe de  $S_1$ ,  
alors  $\text{avg}(S_1) \geq \text{avg}(S)$ 
    - $\{8, 4, 3\}$  est un suffixe de  $\{9, 8, 4, 3\}$
    - $\text{avg}(\{9, 8, 4, 3\})=6 \geq \text{avg}(\{8, 4, 3\})=5$
  - ♦ Si  $S$  satisfait  $\text{avg}(S) \geq v$ , alors  $S_1$  aussi
    - $\{8, 4, 3\}$  satisfait  $\text{avg}(S) \geq 4$ , ainsi que  $\{9, 8, 4, 3\}$

- Les règles d'association sont générées en 2 étapes:
  1. Les itemsets fréquents sont retournés
  2. Les règles en sont induites
- On distingue les associations selon plusieurs critères
  - ♦ Booléennes vs. Quantitatives
  - ♦ Uni. vs. multidimensionnelles
  - ♦ Mono. vs. multi-niveaux

- Apriori travaille par niveaux (levelwise) correspondant aux tailles des itemsets
  - ♦ Générer les candidats (réduction du nombre)
  - ♦ Tester les candidats
  - ♦ Optimisations (hachage, sampling, réduction de la base ...)
- FP-trees : génère, en 2 passes sur la base, un arbre résumant les données
  - ♦ Pas de génération de candidats
  - ♦ Pas de tests de fréquence sur la base

- Les règles multi-niveaux peuvent être générées selon différentes approches
  - ♦ Mêmes vs. différents supports selon les niveaux
  - ♦ Différents types de discrétisation
- Tenir compte des corrélations pour ne pas prendre des décisions hâtives
- Introduction des contraintes pour l'extraction des règles.
  - ♦ Optimisation selon le type des contraintes.