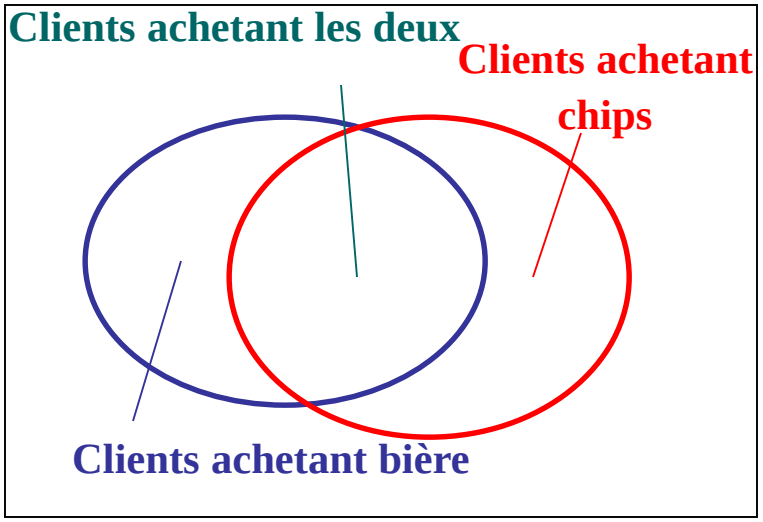
Règles d’association

Recherche des Associations

- Règles d'association :
  - motifs de la forme : Corps → Tête
  - Exemple : achète(x, "cacahuètes") → achète(x, "bière")
- Etant donnés: (1) une base de transactions, (2) chaque transaction est décrite par un identifiant et une liste d'items
  - 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 cacahuètes achètent de la bière*

Mesures: Support et Confiance



Trouver les règles *X & Y ⇒ 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*

Confiance=support(X,Y,Z)/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 ⇒ C (50%, 66.6%)

C ⇒ A (50%, 100%)

Extraction de règles: Exemple

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%

Pour $A \Rightarrow C$:

support = support({*A*, *C*}) = 50%

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

Extraction des associations: 1ère approche

- D: une base de transactions
- I: ensemble de tous les items avec |I|=n
- Algorithme 1: Extraction des ensembles fréquents

```
Fréquents = ∅
Pour chaque J ⊆ I Faire
    count(J)=0
    Pour chaque transaction t∈D Faire
        Si J ⊆ t.items Alors
            count(J)=count(J)+1
    Si count(j) ≥ min_support Alors
        Fréquents=Fréquents += J
Fin
```

Extraction des associations: 1ère approche

- D: une base de transactions
- I: ensemble de tous les items avec |I|=n
- Algorithme 2: Extraction des règles

Règles = ∅

Pour chaque J dans Fréquents

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

Si confiance(r) ≥ min_confiance **Alors**

Règles= Règles+= r

Fin

Extraction des associations: 1ère approche

- Estimation du coût : On ne s'intéresse qu'aux accès disques
  - Algorithme 1: Pour chaque ensemble d'items J, il faut parcourir la base D pour compter le nombre de ses occurrences.
  - |I|=n alors il y a 2^n ensembles J (c'est le nombre de sous ensembles de I)➔2^n parcours de D !!!
  - Algorithme 2: On peut supposer que l'ensemble « Fréquents » réside en mémoire et count(j) est une information déjà calculée par algorithme 1 ➔ pas d'accès
  - Coût global : 2^n parcours de D !!!

Extraction des associations: A priori

Principe : Si un ensemble est non fréquent, alors tous ses sur-ensembles ne sont pas fréquents

- Si {A} n'est pas fréquent alors {AB} ne peut pas l'être
- si {*AB*} est fréquent alors {*A*} et{*B*} le sont

- Itérativement, trouver les itemsets fréquents dont la cardinalité varie de 1 a k (k-itemset)
- Utiliser les itemsets fréquents pour générer les règles d'association

L'algorithme Apriori

- Etape de jointure: C_k est généré en joignant L_{k-1}avec lui même
- Etape 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 = {items fréquents};

for (k = 1; L_k !=∅; k++) **do begin**

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

for each transaction t dans la base do

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

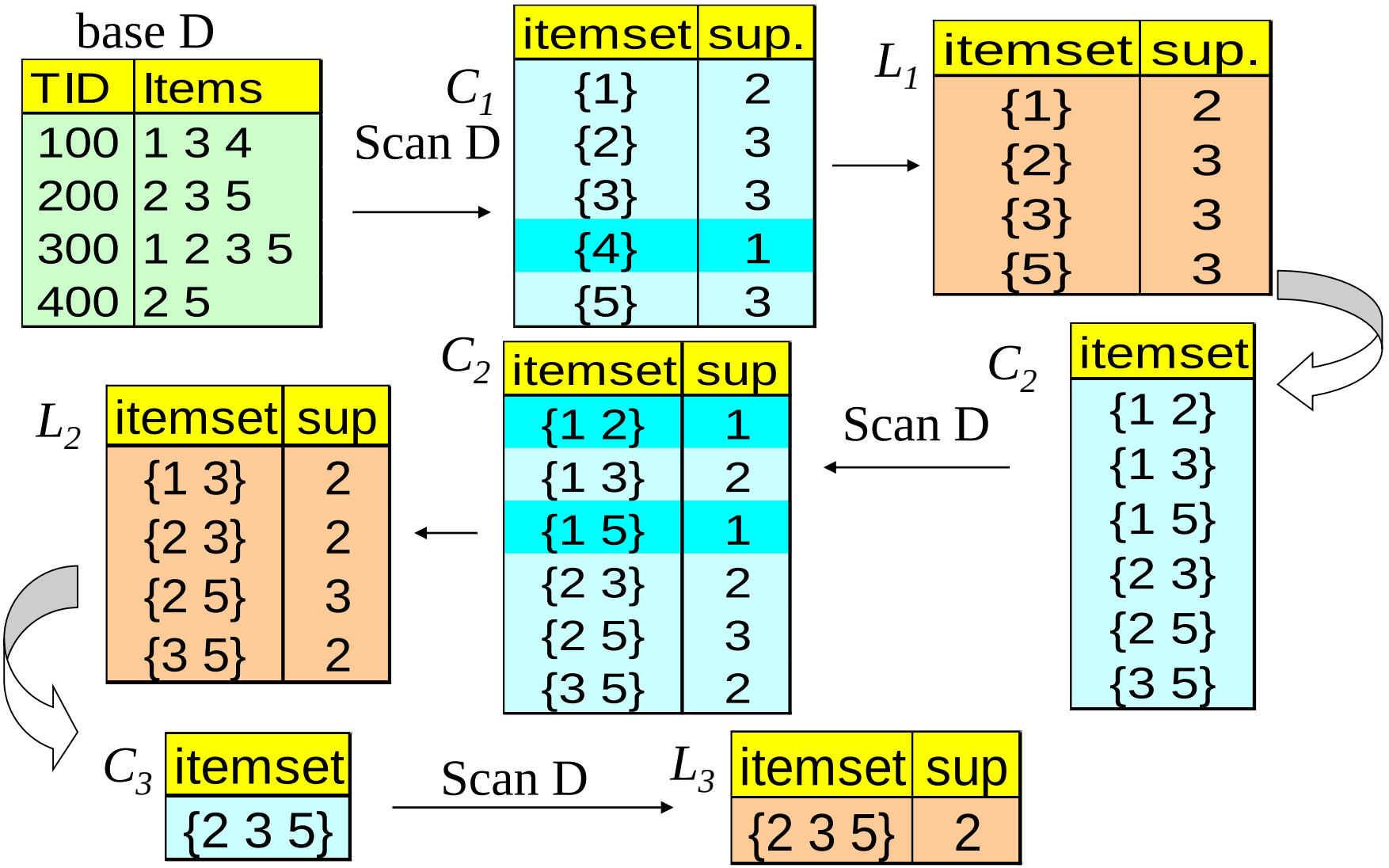
end for

End for

return $\cup_k L_k$

Apriori — Exemple

Avec min_support=2



Génération des Candidats

- Supposons que les items de L_{k-1} sont triés
- Etape 1: self-join de L_{k-1}

  Insert into $\boldsymbol{C_k}$

  select $\boldsymbol{p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}}$

  from $\boldsymbol{L_{k-1}\ p, L_{k-1}\ q}$

  where $\boldsymbol{p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}}$
- Etape 2: pruning (elagage)

  Pour chaque ***itemset c dans C_k Faire***

  Pour chaque ***(k-1)-sous-ensemble s de c*** Faire

  Si *(s n'est pas dans L_{k-1})* Alors supprimer c de C_k

Exemple de Génération de Candidats

- L_3={*abc, abd, acd, ace, bcd*}
- Self-join: L_3*L_3
  - *abcd* à partir *de abc* et *abd*
  - *acde* à partir *acd* et *ace*
- Pruning:
  - *acde* est supprimé car *ade* n'est pas dans L_3
- C_4={*abcd*}

Exemple: Règles d'association

- Supposons que les données soient dans une BD relationnelle avec la table Transaction(Tid, Item). On a 10^8 tuples concernant 10^7 transactions et l'on a 10^5 items différents. En moyenne chaque transaction concerne 10 items.

  La requête suivante sélectionne les paires d'items fréquents

```
SELECT t1.item, t2.item
FROM   transaction t1, transaction t2
WHERE  t1.Tid= t2.Tid AND t1.item < t2.item
GROUP BY t1.item, t2.item
HAVING COUNT(*) >= seuil*taille de la base
```

- Pour chaque transaction on a $C_2^{10}=45$ paires à regarder ainsi la jointure a $45*10^7$ tuples

Exemple: Règles d'association

- Remarque: si {item_1} n'est pas fréquent alors certainement la paire {item_1, item_i} ne l'est pas. Considérons la requête

```
SELECT *
FROM transaction
GROUP BY item
HAVING COUNT(*) >= seuil*taille de la base
```

si seuil = 0,01 alors au plus 1000 item seront dans le résultat.

Raison: il y a 10^8 occurrences d'items. Pour qu'un item soit fréquent il faut qu'il apparaisse $0,01 * 10^7 = 10^5$ fois.

- Pour chercher les paires fréquentes, utiliser le résultat de la requête précédente plutôt que la table initiale

Problèmes d'Apriori

- 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
- Là où l'algo pèche : génération des candidats
  - Beaucoup:
    - 10^4 1-itemsets fréquents générant 10^7 2-itemsets candidats
    - 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

Exploration sans génération de candidats

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

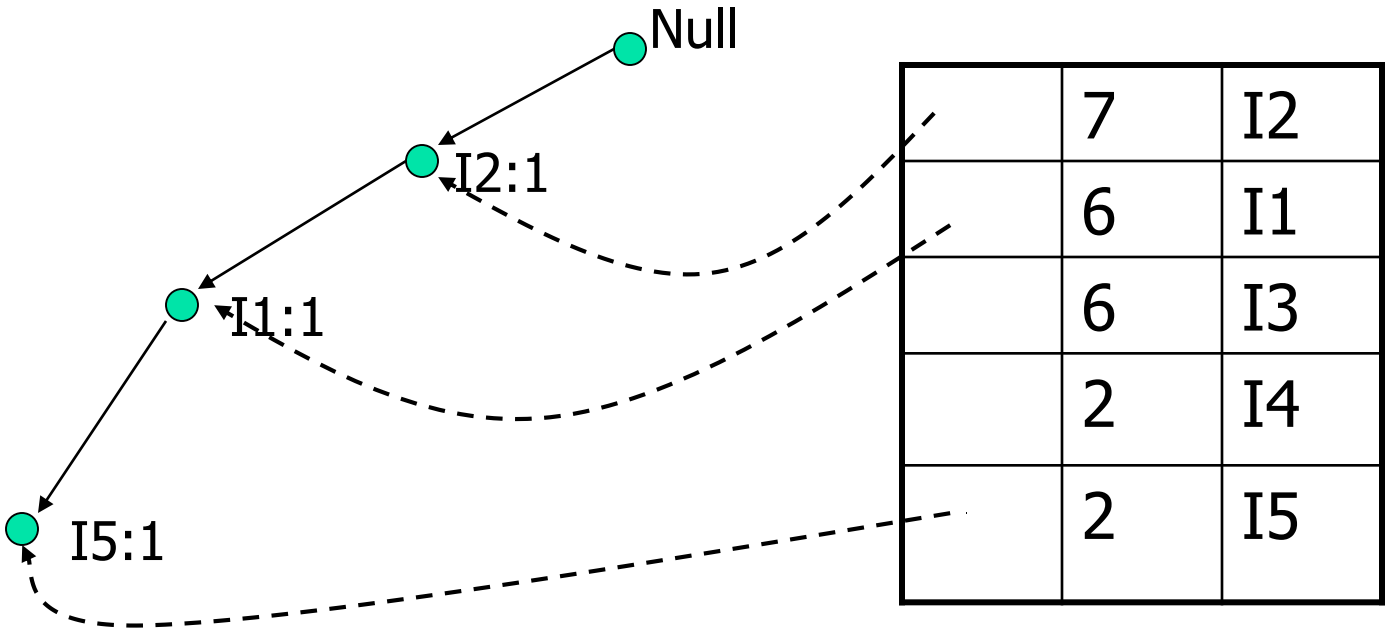
FP-Trees: Exemple

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

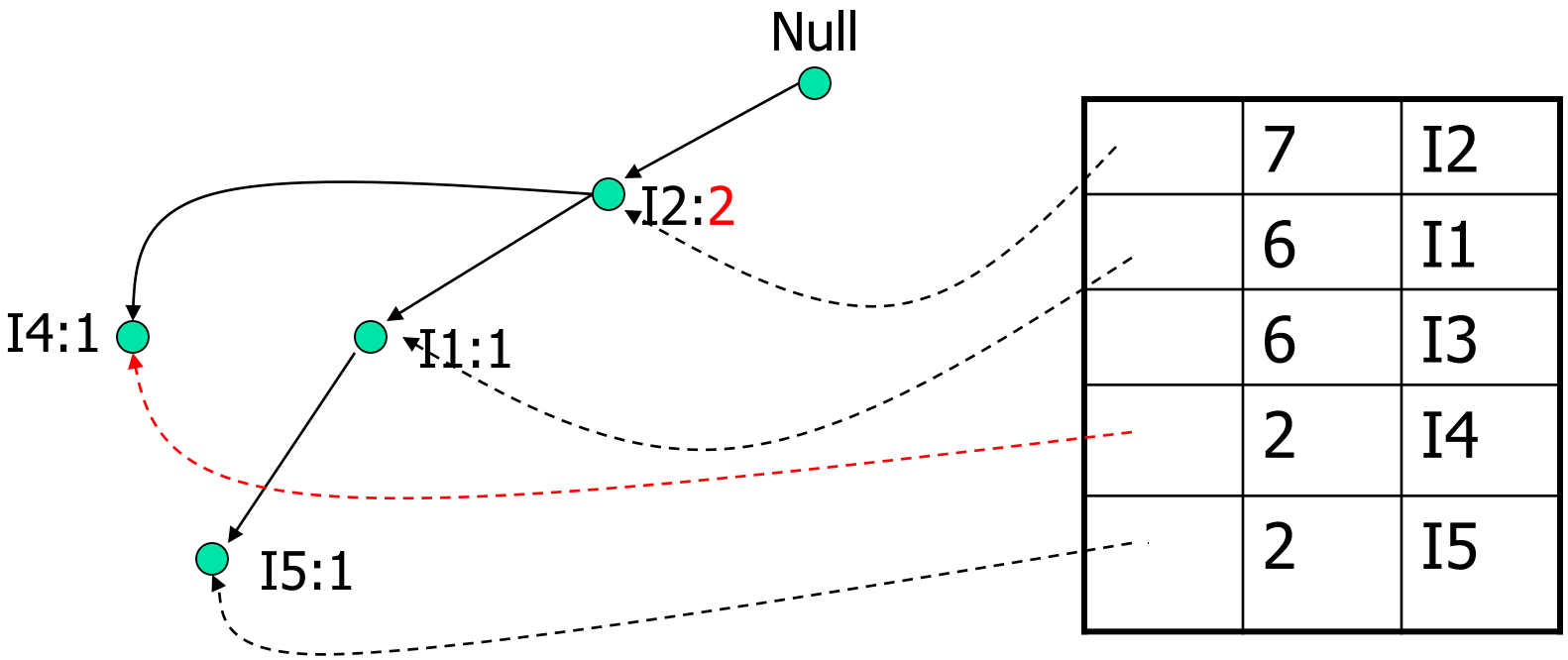
On parcourt une 2ème fois la base. On lit les transactions selon l'ordre des items dans L, i.e pour T100 on a I2,I1,I5. La lecture de T100 donne



FP-Trees: exemple

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

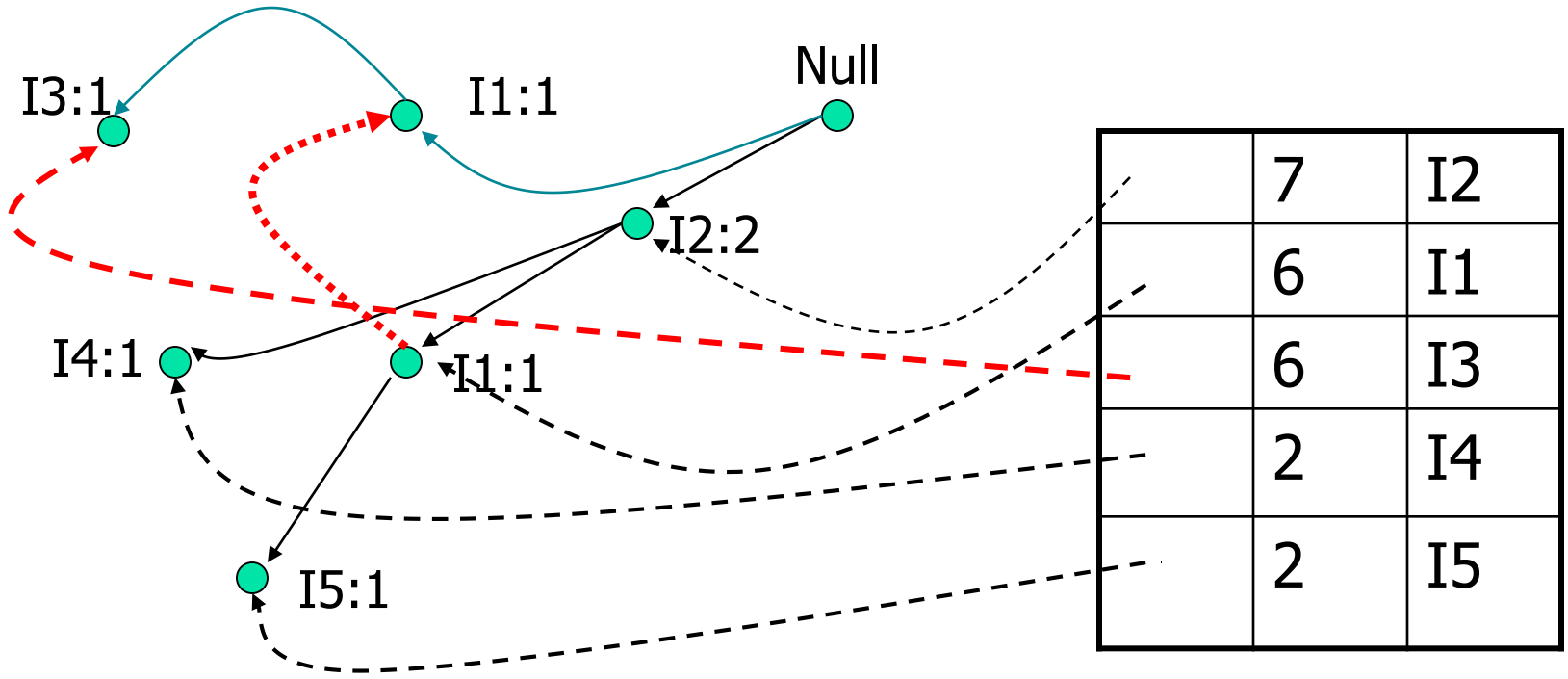
La lecture de T200 va a priori générer une branche qui relie la racine à I2 et I2 à I4. Or cette branche partage un préfixe (i.e I2) avec une branche qui existe déjà. L'arbre obtenu après lecture de T200 sera



FP-Trees: exemple

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

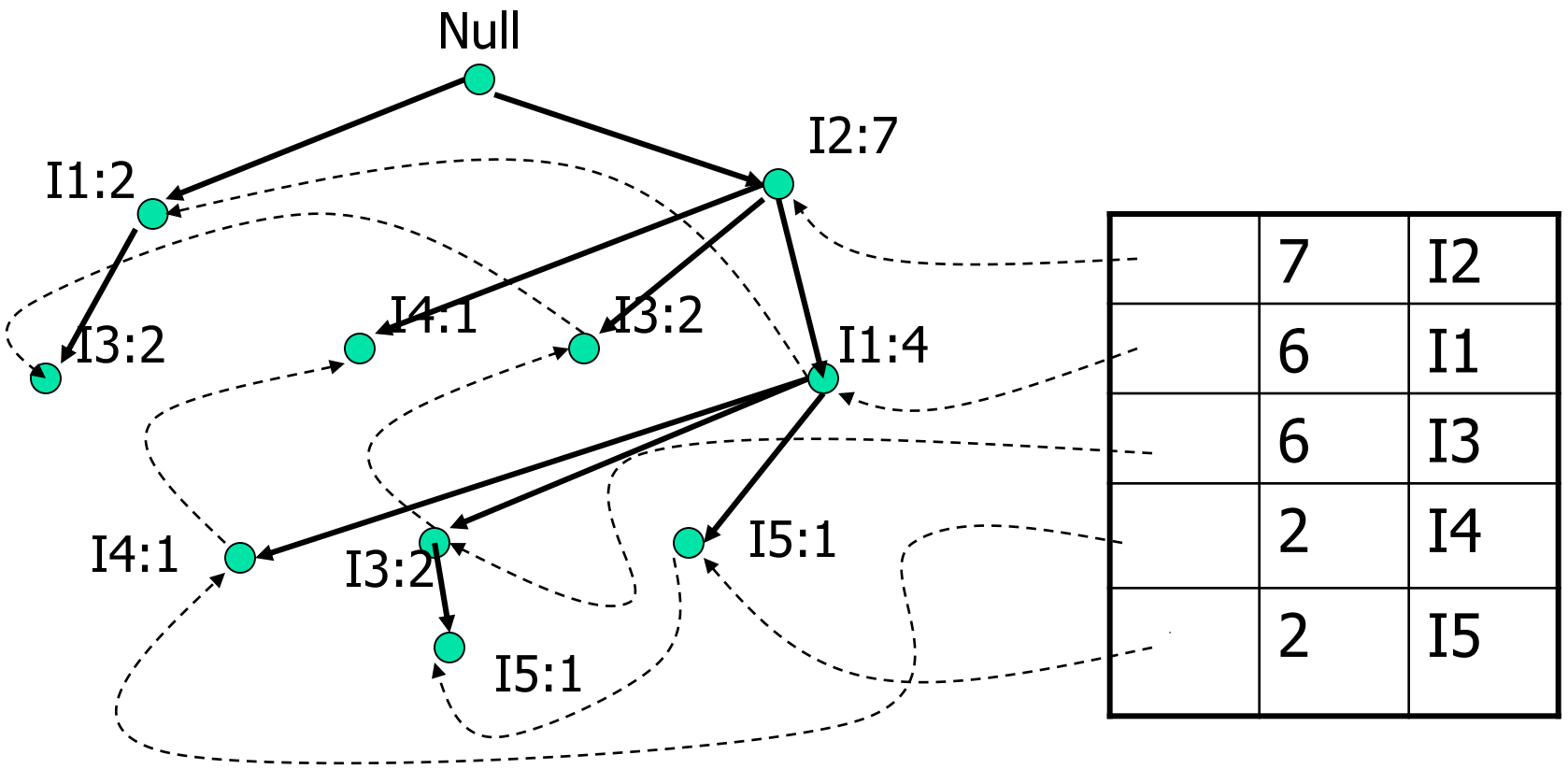
En lisant T300, l'ordre selon L est I1,I3. Ceci nous amène à ajouter une branche Null→I1→ I3. Noter qu'elle n'a pas de préfixe commun avec ce qui existe déjà. On obtient



FP-Trees: exemple

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

Finalement, le FP_tree obtenu est



Phase de l'exploration

- Considérons I5. Il apparaît dans 2 branches. I2→I1→I5:1 et I2→I1→I3→I5:1

- Ainsi, pour le suffixe I5, on a 2 chemins préfixes: <I2,I1:1> et <I2,I1,I3:1>. Ils forment sa «table conditionnelle»

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

Phase d'exploration (suite)

- Le «FP-tree conditionnel» de I5 contient une seule branche I2→I1. I3 n'en fait pas partie car son support est 1 qui est <2 (Rappel: 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

Phase de l'exploration

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

Phase de l'exploration

Item	Base conditionnelle	FP-tree conditionnel	Itemsets générés
I5	<I2,I1>:1,<I2,I1,I3>:1	I2:2→I1:2	{I2, I5}:2 {I1, I5}:2 {I2,I1,I5}:2
I4	<I2,I1>:1,<I2>:1	I2:2	{I2, I4}:2
I3	<I2,I1>:2 ,<I2>:2, <I1>:2	I2:4→I1:2 I1:2	{I2, I3}:4 {I1,I3}:4 {I2, I1, I3}:2
I1	<I2>: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

Critiques des notions de Support et de confiance

- Parmi 5000 étudiants
  - 3000 jouent au basket
  - 3750 prennent des céréales
  - 2000 jouent du 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	Σ
céréales	2000	1750	3750
non céréale	1000	250	1250
sum(col.)	3000	2000	5000

Critiques des notions de Support et de confiance

- Exemple 2:
  - X et Y: positivement corrélés,
  - X et Z, négativement corrélés
  - Les support et confiance de X→Z dominent

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

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

Règle	Support	Confiance
X-->Y	25%	50%
X-->Z	37,50%	75%

- est aussi appelé le lift de A => B

Autres mesures

- 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 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

Résumé

- Les règles d'association sont générées en 2 étapes:
  1. Les itemsets fréquents sont retournés
  2. Les règles en sont induites

Résumé (suite)

- Apriori travaille par niveaux (levelwise) correspondants aux tailles des itemsets
  - Générer les candidats (réduction du nombre)
  - Tester les candidats
  - Optimisations
- FP_growth: génère après 2 passes sur la base un arbre résumant les données
  - Pas de génération de candidats
  - Pas de tests de fréquence sur la base

Résumé (suite)

- Tenir compte des corrélations pour ne pas prendre des décisions hâtives