



# 大数据导论

## Introduction to Big Data



### 第6-7讲 关联规则挖掘

叶允明

计算机科学与技术学院

哈尔滨工业大学（深圳）

# 目录

- (1) 关联规则是什么?
- (2) 关联规则挖掘的两阶段算法框架
- (3) 频繁项集生成方法
- (4) 关联规则生成及相关性评价
- (5) 挖掘多维量化关联规则

# 主要参考资料

- Jiawei Han, Micheline Kamber, Jian Pei著；范明，孟小峰等译. 数据挖掘：概念与技术. 机械工业出版社, ISBN: 9787111391401, 2012.
- 第6章 挖掘频繁模式、关联和相关性：基本概念和方法

关联规则是什么？

# 什么是关联规则 ( Association Rule ) ？

牛奶 → 面包

尿布 → 啤酒

{牛奶, 面包} → {啤酒, 尿布}

**$X \rightarrow Y$**

TID (事务ID)	商品列表
T001	牛奶、啤酒、尿布
T002	鸡蛋、牛奶、面包、啤酒、尿布
T003	鸡蛋、牛奶、面包
T004	面包、啤酒
T005	牛奶、面包、啤酒、尿布

age("25-35")  $\wedge$  buy("华为手机")  $\rightarrow$  buy("格力空调")

# 关联规则有什么用？

- 零售行业：优化货架
- 电商行业：商品推荐
- 库存优化
- .....

# 关联规则的形式化定义

- 基本概念

- 事务(transaction)

- 项 (item)

- **项集(item set)**

- ✓ k-项集

- **关联规则:**

- ✓ 形如  $X \rightarrow Y$  的蕴涵表达式

- $X$  和  $Y$  为非空集合

- $X$  和  $Y$  是**不相交**的两个项集

TID (事务ID)	商品列表
T001	牛奶、啤酒、尿布
T002	鸡蛋、牛奶、面包、啤酒、尿布
T003	鸡蛋、牛奶、面包
T004	面包、啤酒
T005	牛奶、面包、啤酒、尿布

TID (事务ID)	商品列表
T001	尿布、牛奶、啤酒
T002	尿布、鸡蛋、牛奶、面包、啤酒
T003	鸡蛋、牛奶、面包
T004	面包、啤酒
T005	尿布、牛奶、面包、啤酒



TID (事务ID)	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

**$X \rightarrow Y$  很多!**



# 如何衡量关联规则的“质量”？

## 项集的支持度计数：

✓ 在数据库中包含项集  $W$  的事务个数

$$\sigma(W) = |\{t_i | W \subseteq t_i, t_i \in T\}|$$

## 支持度 (support)

✓ 给定数据集中  $X$  和  $Y$  的共现频度

✓ 令  $W = X \cup Y$

$$\sigma(X \rightarrow Y) = \frac{\sigma(W)}{|T|}$$

## 置信度 (confidence)

✓ 在包含  $X$  的事务子集中  $Y$  出现的频繁程度

$$c(X \rightarrow Y) = \frac{\sigma(W)}{\sigma(X)}$$

$X \rightarrow Y$

$R_1: I5 \rightarrow I1$

TID (事务ID)	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

$$s(R_1) = \frac{\sigma(\{I5, I1\})}{|T|} = \frac{3}{5} = 0.6$$

$$c(R_1) = \frac{\sigma(\{I5, I1\})}{\sigma(\{I5\})} = \frac{3}{4} = 0.75$$

# 强关联规则 (strong association rule)

- 定义

- 最小支持度阈值: *min-sup*, 最小置信度阈值: *min-conf*

- 关联规则  $R$  是强关联规则, 当且仅当:

$$(1). s(R) \geq \textit{min-sup} \qquad (2). c(R) \geq \textit{min-conf}$$

- 关联规则挖掘任务: 找到所有**强**关联规则!

# 关联规则挖掘的两阶段算法框架

# 关联规则挖掘任务

- 输入：

- 事务数据库、
- 阈值参数：  $min-sup$  ,  $min-conf$

- 输出：

- 所有满足  $min-sup$  、  $min-conf$  的强关联规则
- 包括每条强关联规则的支持度和置信度取值

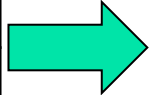
TID (事务ID)	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

# 最简单的关联规则挖掘算法——枚举法

- 假设:  $min-sup = 0.6$ ,  $min-conf = 0.8$
- 枚举法:
  - Step 1: 列出全部可能的关联规则
  - Step 2: 计算每条关联规则的支持度和置信度
  - Step 3: 筛选出满足min-sup和min-conf条件的规则

TID (事务ID)	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}



关联规则	支持度	置信度
{I1}→{I2}	0.2	0.33
{I2}→{I1}	0.2	0.5
{I1}→{I3}	0.6	1.0
{I3}→{I1}	0.6	0.75
.....	...	...
{I1, I2}→{I3}	0.2	1.0
{I3}→{I1, I2}	0.2	0.25
.....	...	...
{I1, I2, I3}→{I4}	0.2	1.0
{I4}→{I1, I2, I3}	0.2	0.25
.....	...	...
{I1, I2, I3, I4}→{I5}	0.2	1.0
{I5}→{I1, I2, I3, I4}	0.2	0.25
.....	...	...

min-sup=0.6

min-conf=0.8



关联规则	支持度	置信度
{I1}→{I3}	0.6	1.0
{I1}→{I5}	0.6	1.0
{I5}→{I3}	0.6	1.0
{I1}→{I3, I5}	0.6	1.0
{I3, I5}→{I1}	0.6	1.0
{I1, I5}→{I3}	0.6	1.0
{I1, I3}→{I5}	0.6	1.0

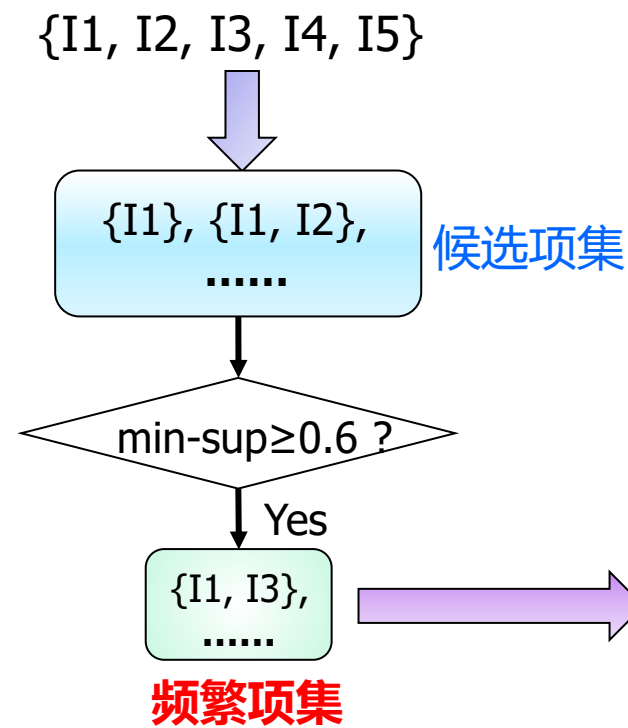
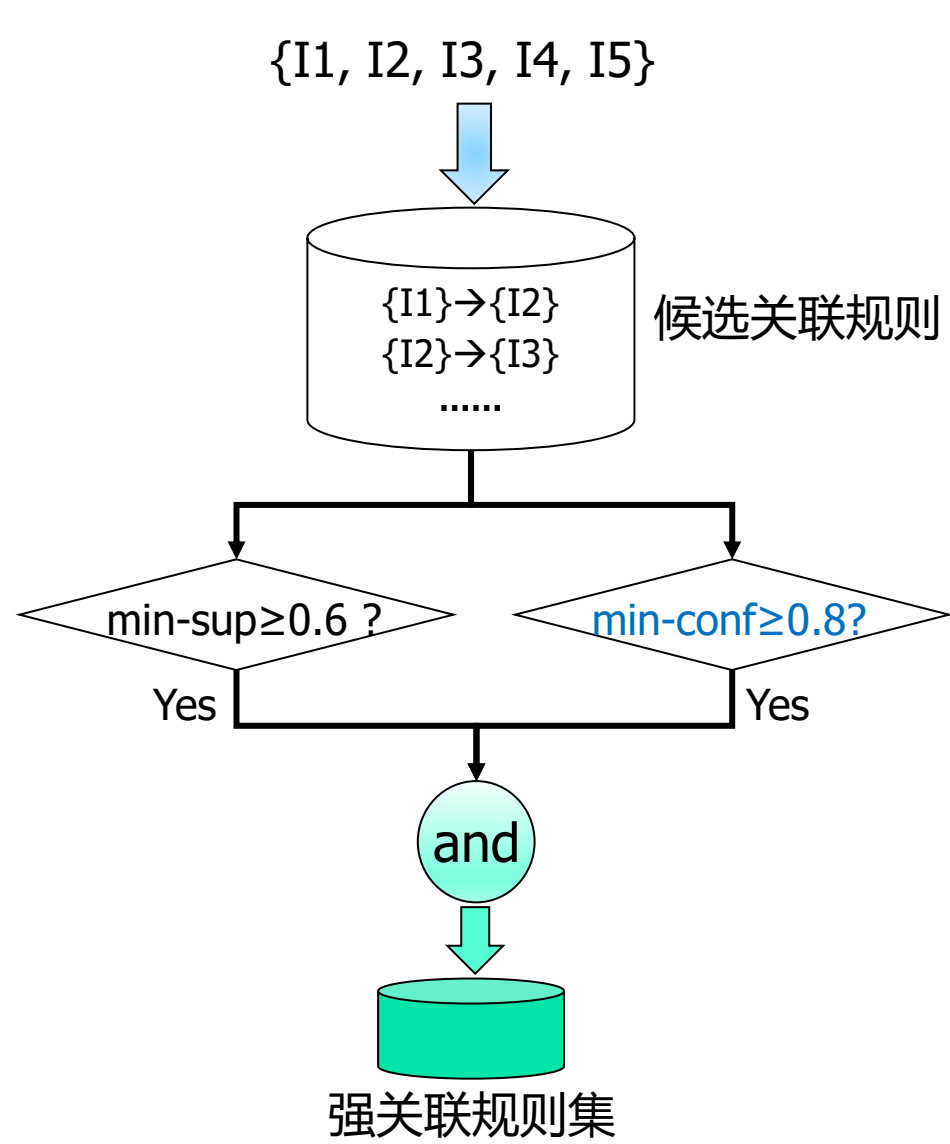
# 枚举法的特点分析

- 优点：实现简单，容易理解
- 缺点：待计算的关联规则数量随数据集增大呈指数增长
  - 包含n个项的数据集可提取的关联规则总数M为： $M = 3^n - 2^{n+1} + 1$

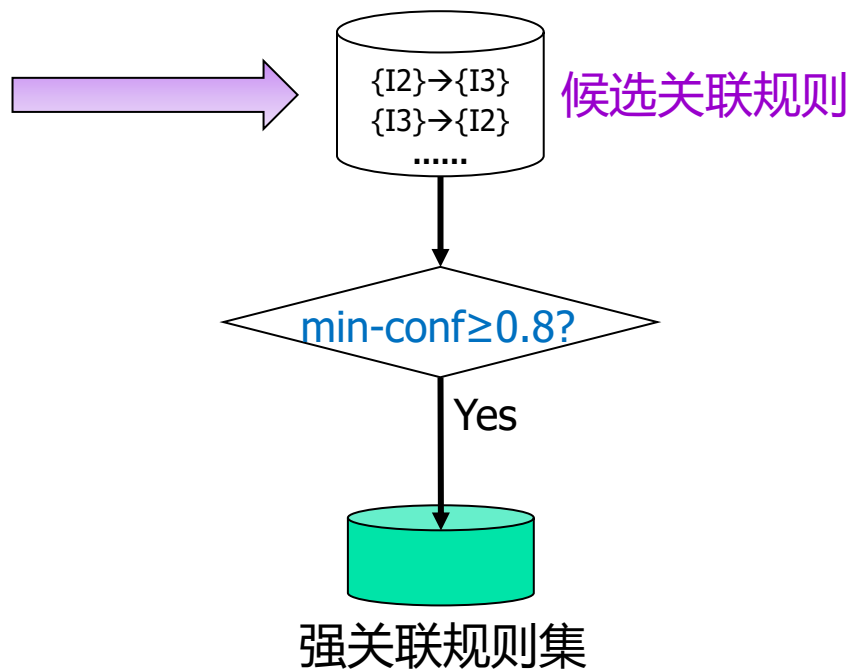
TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

n=5, 可产生180条关联规则!

改进思路：从单步筛选变为“**两阶段**”筛选

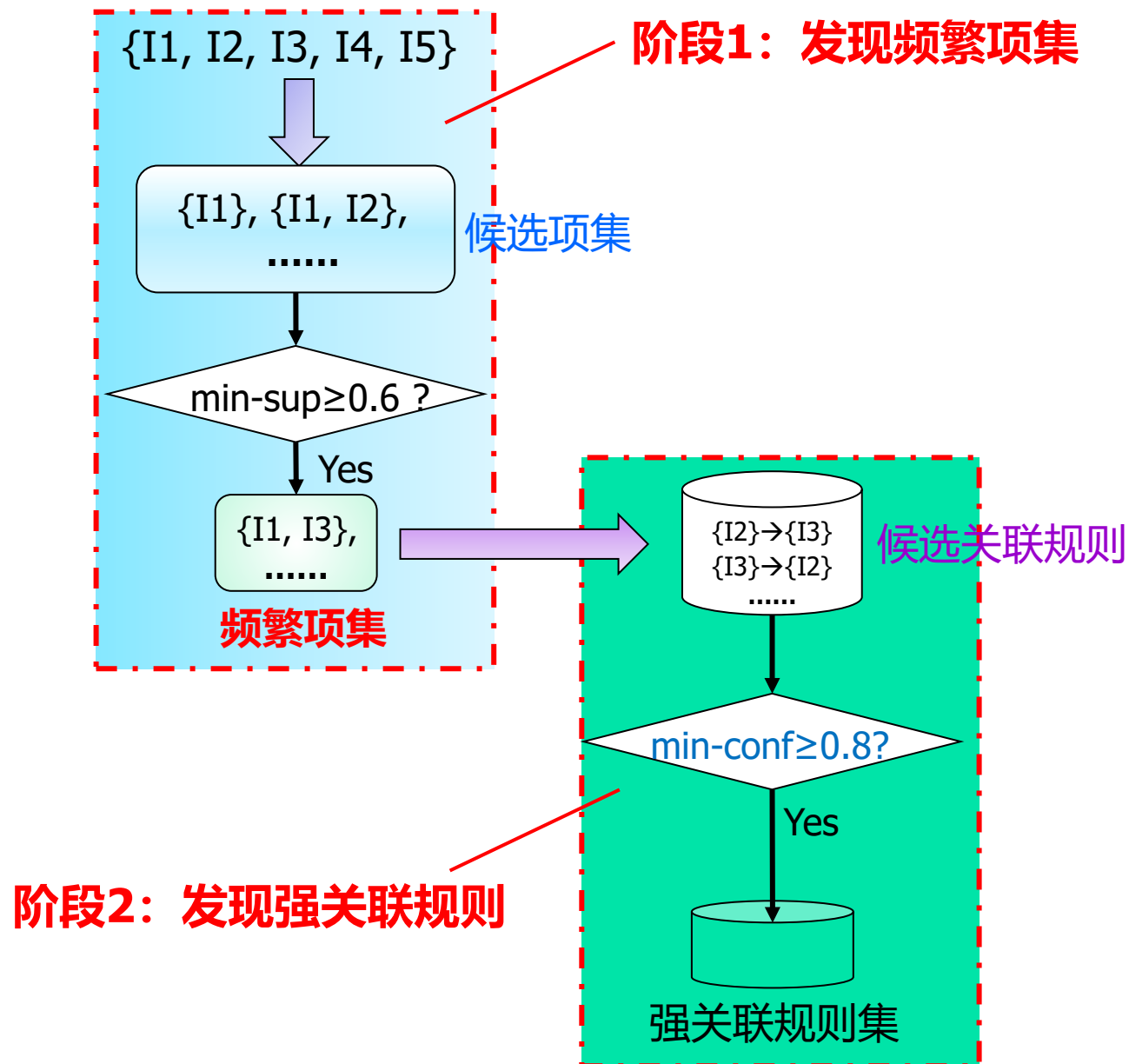
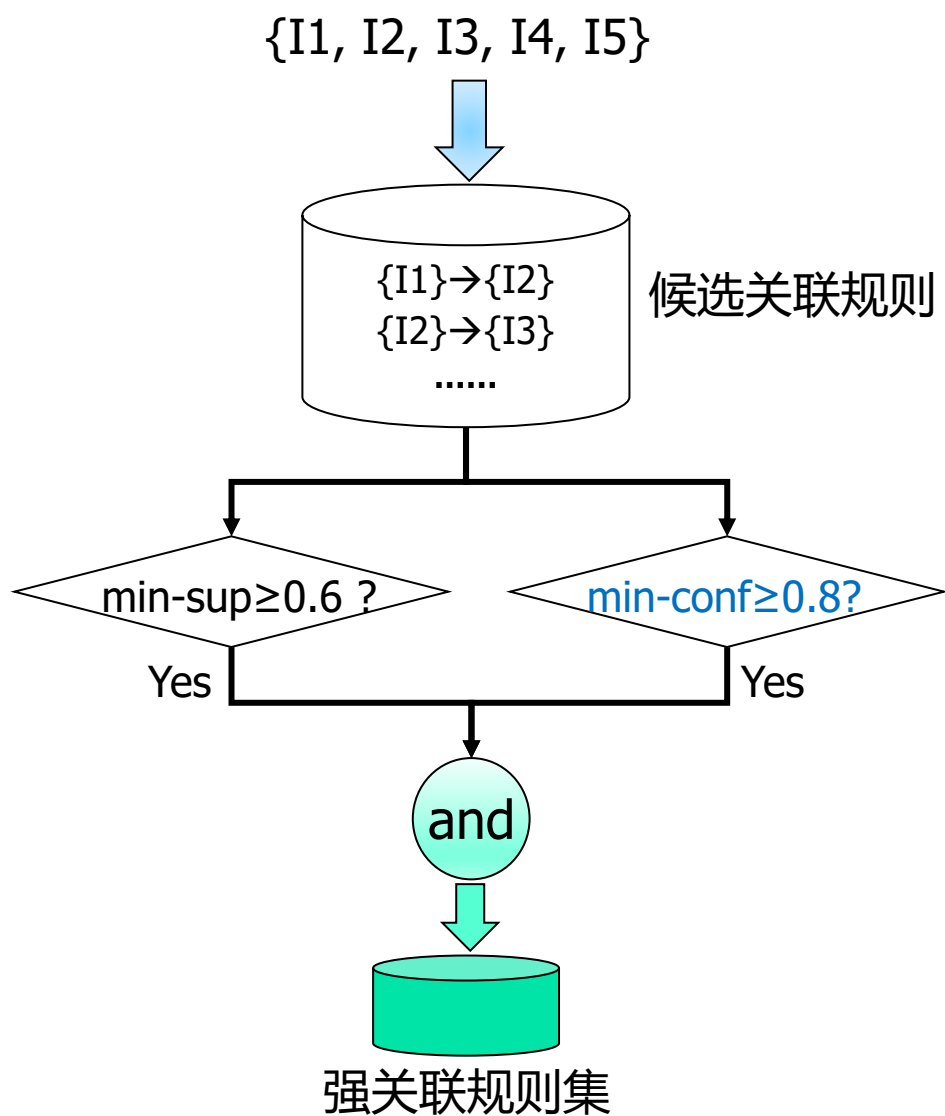


TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

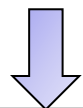




# 关联规则挖掘的两阶段算法框架



{I1, I2, I3, I4, I5}



{I1}, {I1, I2},  
.....

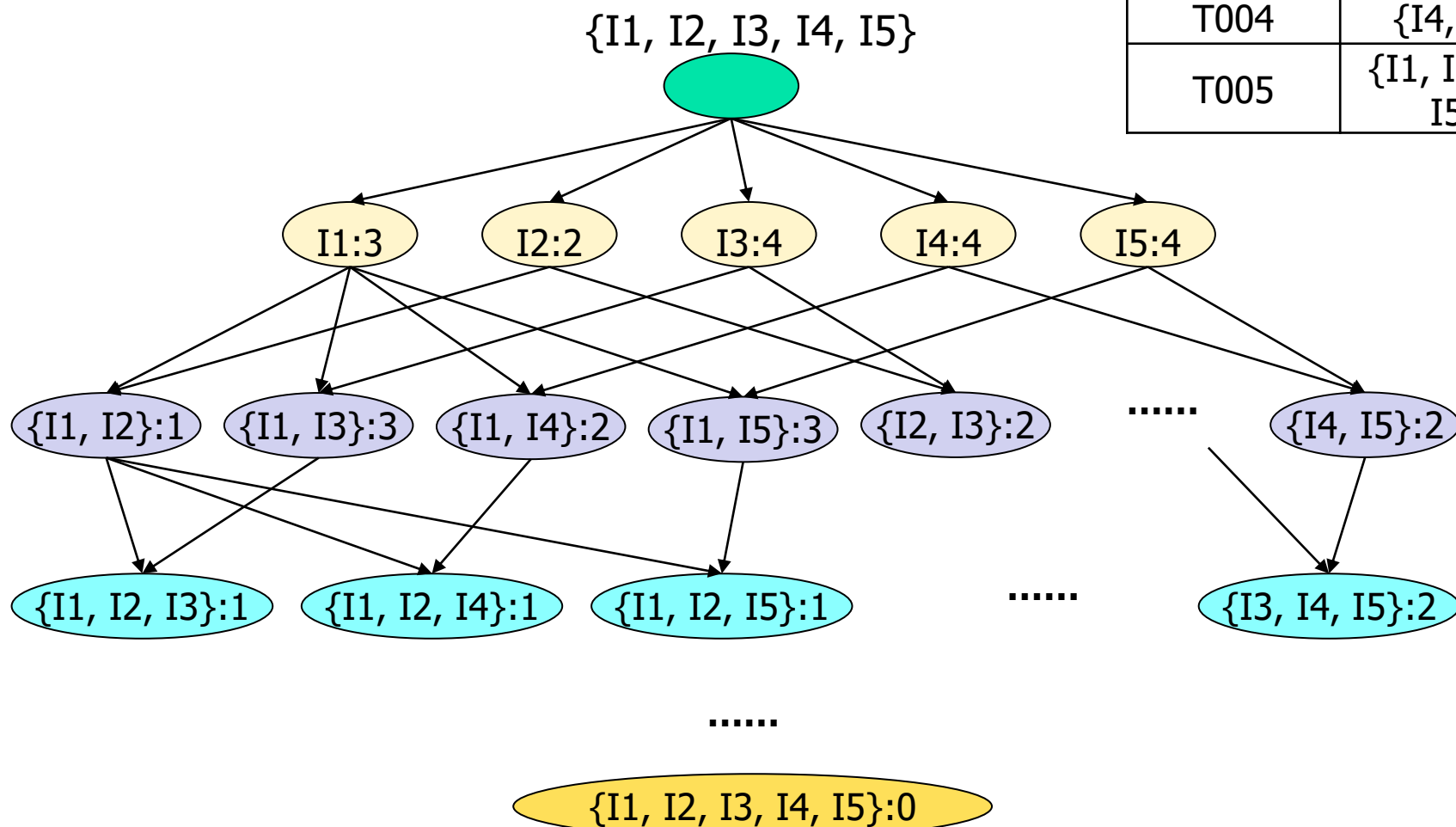
候选项集

## 阶段1：发现频繁项集

$\text{min-sup} = 0.6$

TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

频繁项集	支持度计数
{I1}	3
{I3}	4
{I4}	4
{I5}	4
{I1, I3}	3
{I1, I5}	3
{I3, I4}	3
{I3, I5}	3
{I4, I5}	3
{I1, I3, I5}	3

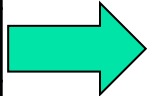


# 阶段2：发现强关联规则

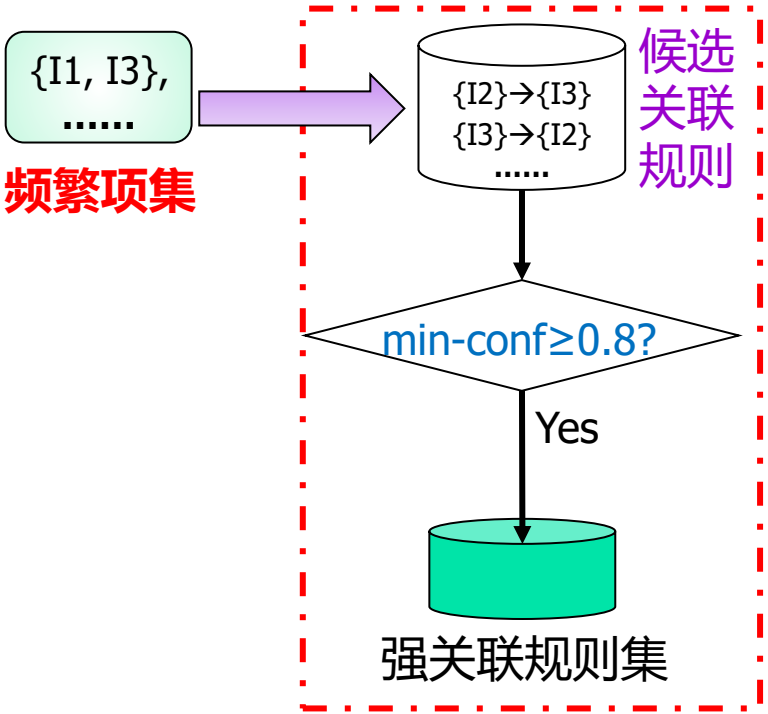
$min-conf = 0.8$

TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

频繁项集	支持度 计数
{I1}	3
{I3}	4
{I4}	4
{I5}	4
{I1, I3}	3
{I1, I5}	3
{I3, I4}	3
{I3, I5}	3
{I4, I5}	3
{I1, I3, I5}	3

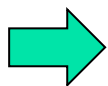


频繁项集	支持度 计数
{I1, I3}	3
{I1, I5}	3
{I3, I4}	3
{I3, I5}	3
{I4, I5}	3
{I1, I3, I5}	3

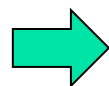


# 两阶段算法与单步枚举法的计算量比较

TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}



频繁项集	支持度计数
{I1}	3
{I3}	4
{I4}	4
{I5}	4
{I1, I3}	3
{I1, I5}	3
{I3, I4}	3
{I3, I5}	3
{I4, I5}	3
{I1, I3, I5}	3



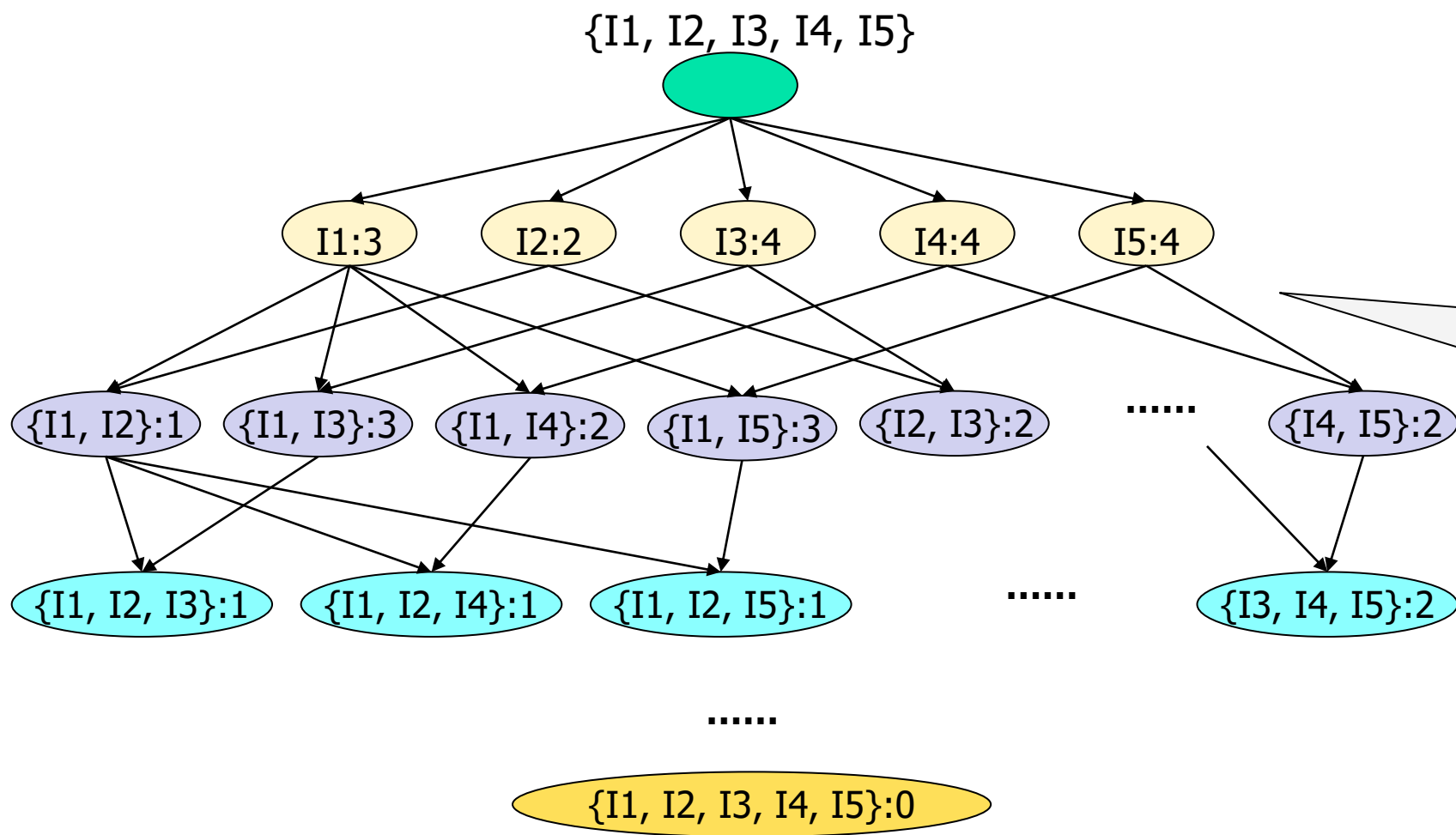
关联规则	置信度
{I1}→{I3}	1.0
{I1}→{I5}	1.0
{I5}→{I3}	1.0
{I1}→{I3, I5}	1.0
{I3, I5}→{I1}	1.0
{I1, I5}→{I3}	1.0
{I1, I3}→{I5}	1.0

- 阶段一：频繁项集发现
  - 计算31个项集的支持度
  - 筛选得到12个频繁项集
- 阶段二：关联规则生成
  - 计算12条候选关联规则的置信度
  - 筛选得到7条强关联规则
- 单步枚举法需要计算180个关联规则的支持度和置信度
- 计算量对比：360 → 43**

# 频繁项集发现：Apriori算法

# 用枚举法发现频繁项集的问题

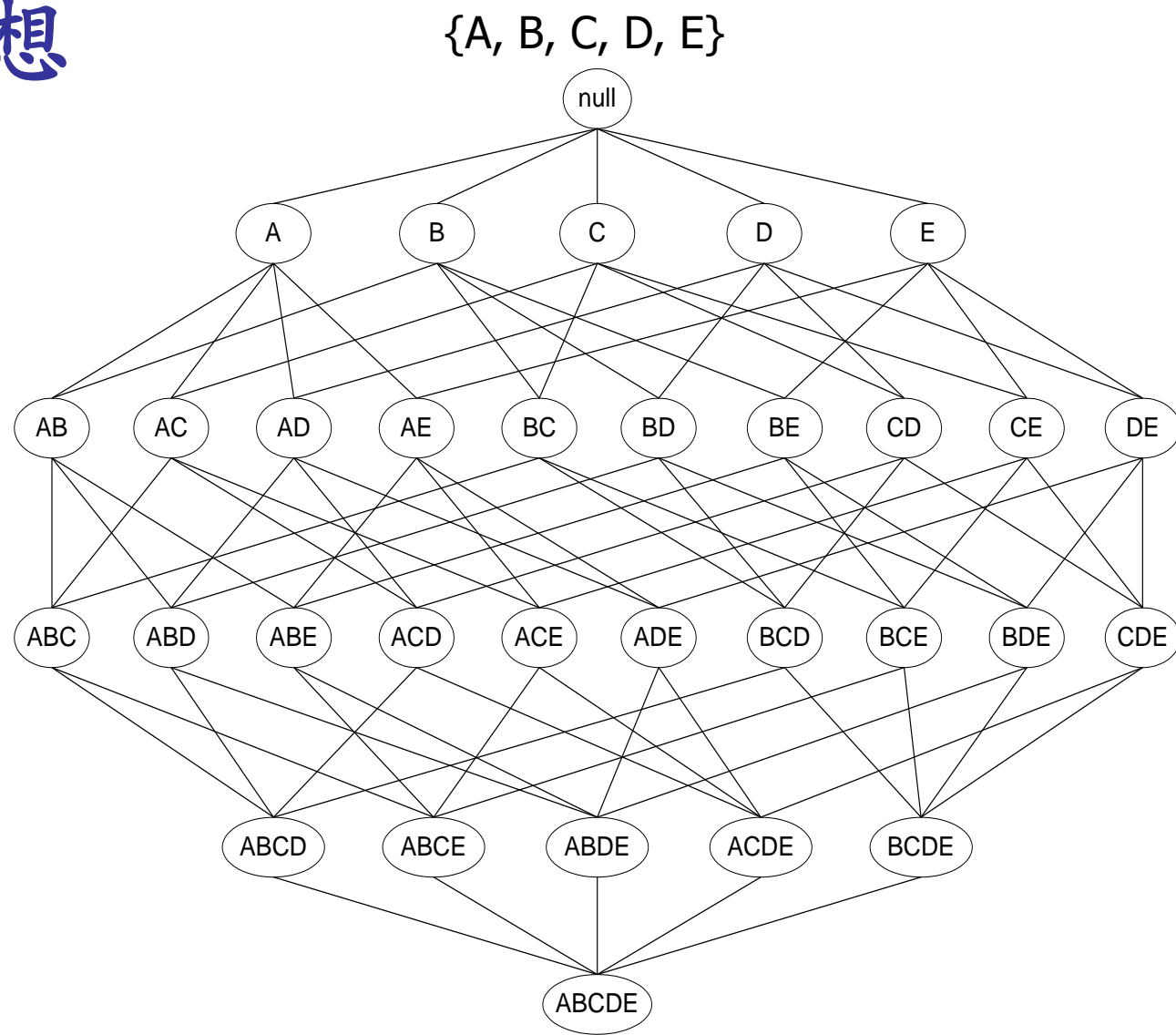
TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}



k个项  
有 $2^k-1$ 个可能的候选项集  
计算开销大!

# Apriori算法的基本思想

- 减少候选项集的数量!
- 剪枝!



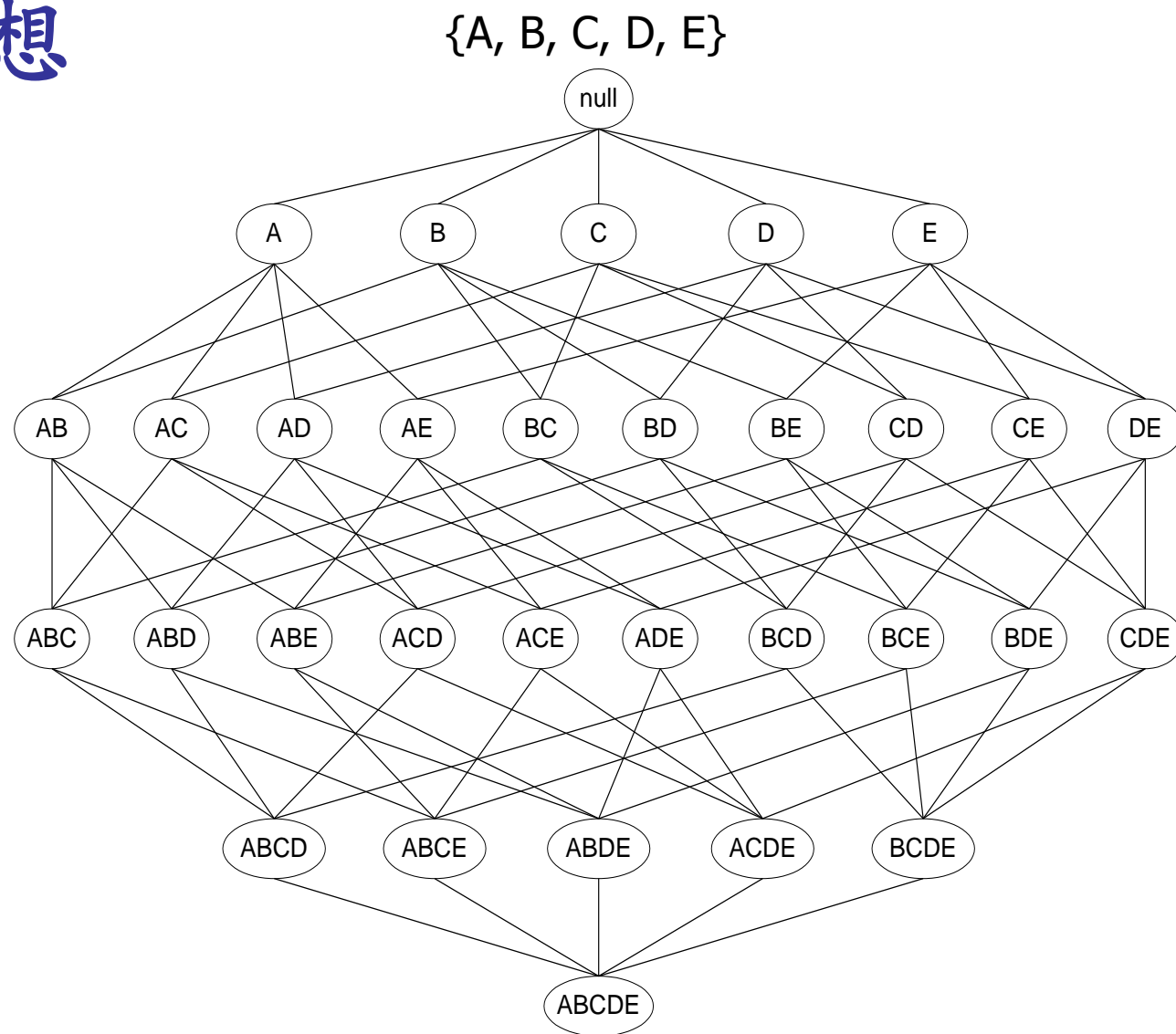
# Aprior算法的基本思想

- 先验原理

- 频繁项集的子集一定也是频繁的！

- 反单调性

- 一个项集的支持度不超过它的子集的支持度
- 如果对于项集Y的每个真子集X (即 $X \subset Y$ ) , 有 $f(Y) \leq f(X)$  , 那么称度量f具有反单调性。

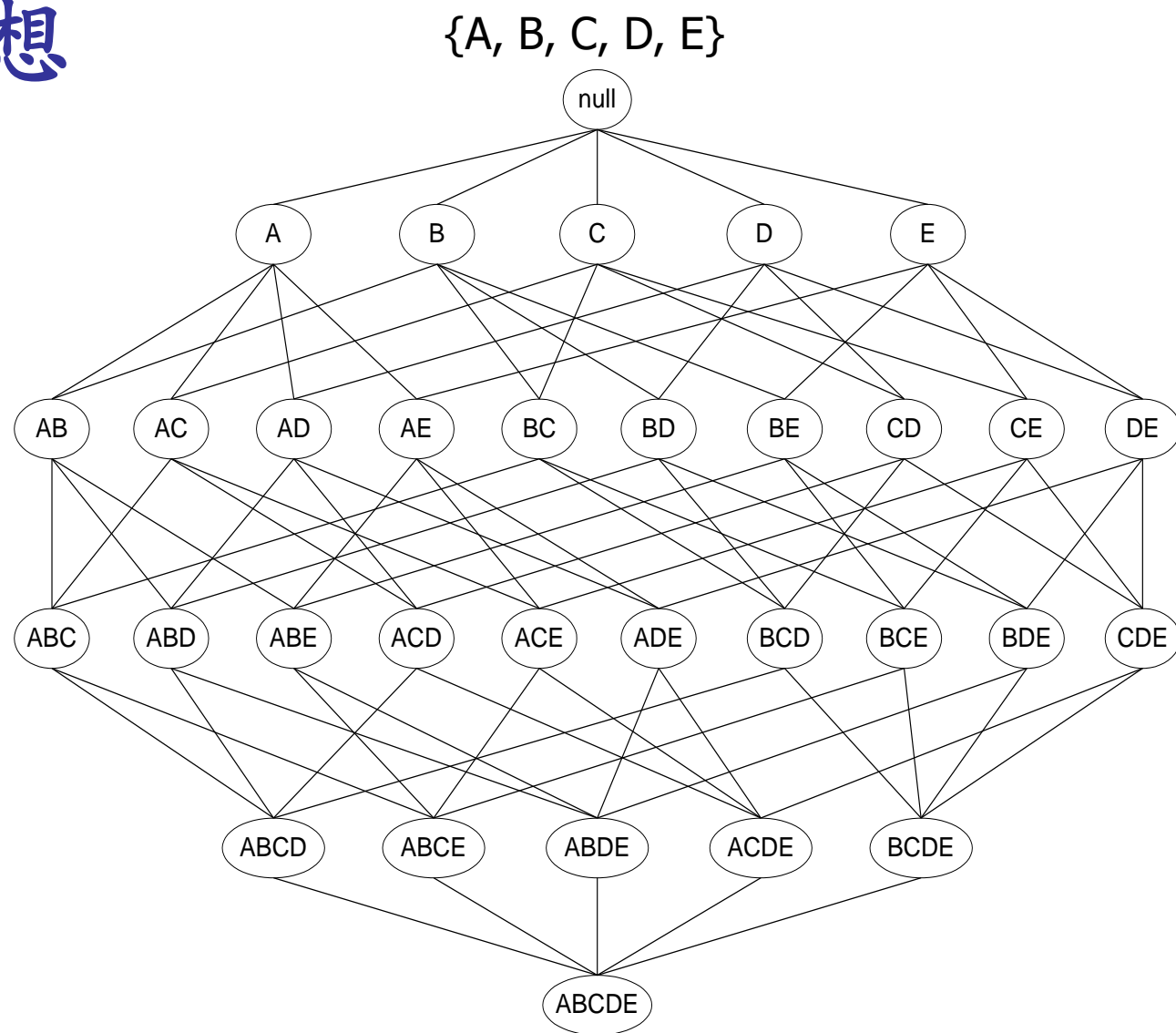




# Apriori算法的基本思想

- 基本思想:

- 逐层搜索迭代
- 用上一轮迭代得到的k项集来探索下一轮迭代的 (k+1) 项集



# Aprior算法的主要步骤

(1). 扫描数据库，得到所有频繁1项集

$min-sup = 0.6$

(2). 用k-频繁项集生成 (k+1)-候选项集

(3). 扫描数据库计算每个 (k+1)-候选项集的支持

度，如果超过  $min-sup$  则为 (k+1)-频繁项集

(4). 重复 (2)、(3) 直到无法生成新的候选项集

TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

TID	商品ID列表
T001	{I1, I2, I3, I4, I5}
T002	{I1, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I1, I3, I5}
T005	{I4, I5}

1<sup>st</sup> scan

Itemset	sup
{I1}	
{I2}	
{I3}	
{I4}	
{I5}	

C1



Itemset	sup

L1



Itemset

C2

2<sup>nd</sup> scan

Itemset	sup

C2



Itemset	sup

L2



Itemset

C3

3<sup>rd</sup> scan

Itemset	sup

C3



Itemset	sup

L3

*min-sup* = 0.6

min-sup计数=3

TID	商品ID列表
T001	{I1, I2, I3, I4, I5}
T002	{I1, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I1, I3, I5}
T005	{I4, I5}

1<sup>st</sup> scan

Itemset	sup
{I1}	3
{I2}	2
{I3}	4
{I4}	4
{I5}	4

C1



Itemset	sup
{I1}	3
{I3}	4
{I4}	4
{I5}	4

L1



Itemset	sup
{I1, I3}	3
{I1, I4}	3
{I1, I5}	3
{I3, I4}	3
{I3, I5}	3
{I4, I5}	3

C2

2<sup>nd</sup> scan

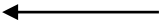
Itemset	sup
{I1, I3}	3
{I1, I4}	2
{I1, I5}	3
{I3, I4}	3
{I3, I5}	3
{I4, I5}	3

C2



Itemset	sup
{I1, I3}	3
{I1, I5}	3
{I3, I4}	3
{I3, I5}	3
{I4, I5}	3

L2



Itemset	sup
{I1, I3, I4}	2
{I1, I3, I5}	3
{I3, I4, I5}	2

C3

3<sup>rd</sup> scan

Itemset	sup
{I1, I3, I4}	2
{I1, I3, I5}	3
{I3, I4, I5}	2

C3



Itemset	sup
{I1, I3, I5}	3

L3

Database  
min-sup=3

# Aprior算法的特点分析

- 优点

- 原理简单，易于实现
- 适合于稀疏数据集中的频繁模式挖掘

- 缺点

- 候选项集数量可能很大
  - ✓ 假设 $L_1$ 有 $10^4$ 项，则 $C_2$ 将包含 $10^7$ 项
  - ✓ 假设要挖掘 $L_{100}$ ，需要产生 $2^{100} \approx 10^{30}$ 个候选项
- 重复扫描数据库
  - ✓ 每轮迭代对候选项集进行支持度计数时，都需要扫描一遍数据库，从而产生不可忽视的I/O开销

# 如何进一步提高频繁项集生成的效率?

- 减少候选项集的数量(**M**)
- 减少需要扫描的事务(记录)数量(**N**)
- 减少候选项集与事务的比较次数 (**NM**)

<b>TID</b>	<b>商品ID列表</b>
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

# 改进Aprior算法的两个思路

- 压缩迭代时扫描的事务数
  - 如果一个事务不包含任何一个频繁k-项集，那么这个事务也一定不包含任何一个频繁 (k+1) -项集
- 基于散列(hashing)优化支持度计数
  - 如散列2-项集时，可以设散列函数：

$$h(x,y) = (f(x) * 10 + f(y)) \bmod 7$$

TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

基于散列优化

自定义映射：

$$h(\{x\ y\}) = \{\{\text{order of } x\} * 10 + \{\text{order of } y\}\} \bmod 7$$

TID	Items
T001	A, C, D
T002	B, C, E
T003	A, B, C, E
T004	B, E

Itemset	order
{A}	1
{B}	2
{C}	3
{D}	4
{E}	5

T001	{A, C}, {A, D}, {C, D}
T002	{B, C}, {B, E}, {C, E}
T003	{A, B}, {A, C}, {A, E}, {C, E}
T004	{B, E}

$$h(\{A, C\}) = (1 * 10 + 3) \bmod 7 = 6$$

min-sup=3

C1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L1

Itemset	sup
{B}	3
{C}	3
{E}	3

桶内容	{C, E}	{C, E}	<del>{A, E}</del>	<del>{B, C}</del>	{B, E}	{B, E}	{B, E}	<del>{A, B}</del>	{A, C}	{A, C}	{C, D}	{A, C}
桶计数	3	1	2	0	3	1	3	1	3	3	3	3
$h(\{x, y\})$	0	1	2	3	4	5	6	5	6	6	6	6

提取原理  
桶计数  
> min-sup  
的项

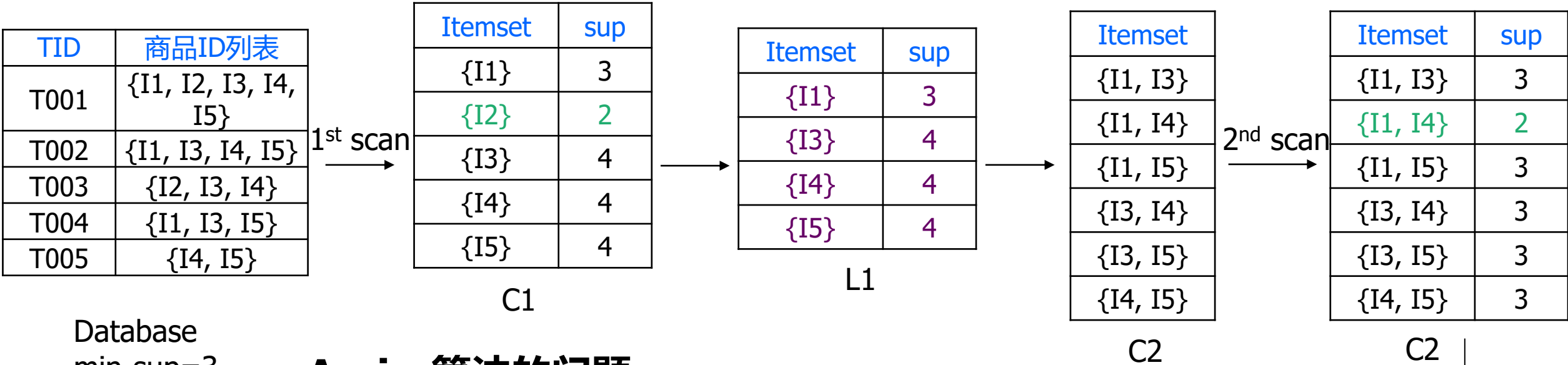
Itemset	sup
<del>{C, E}</del>	/
<del>{A, D}</del>	/
{B, E}	3
<del>{A, C}</del>	/
<del>{C, D}</del>	/

根据支持度筛选

Itemset	sup
{B, E}	3

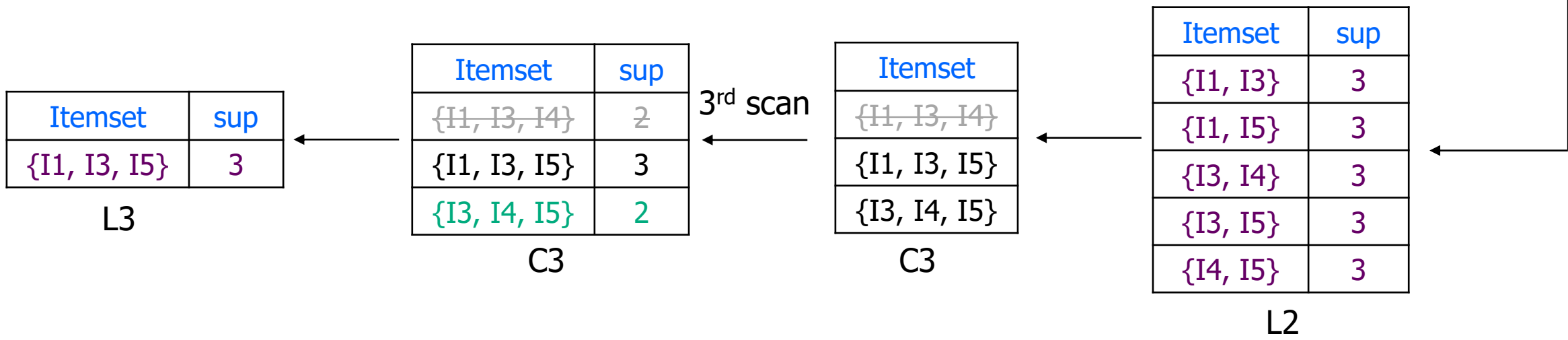


## 频繁项集发现：FP-tree算法



Aprior算法的问题:

- (1) 生成大量的候选项集!      (2) 需多次扫描整个数据库!



# FP-tree: 不需要生成候选项集的算法

- 裴健等人于2000年提出另一种算法——频繁模式生长 (Frequent-  
Pattern Growth) , 也称为FP-tree算法

- 基本思想

- 假设“abc”是频繁项集
- 找到数据集中包含“abc”的记录:

DB|abc —— 条件数据库

- “d”是DB|abc中的频繁项

→ “abcd”也是频繁项集

# FP-tree算法的基本框架

- 使用一种紧凑的数据结构——**FP树** (FP-tree) 来组织数据，并从中发现**频繁项集**
- 主要步骤：
  - 构建FP-tree
  - 从FP-tree中发现（生成）频繁项集
- 特点：
  - 只需要扫描两次原始数据库
  - 在数据库较大、记录较长时，可比Apriori算法快多个数量级

# FP-tree的构建

- (1) 扫描数据库找到频繁1-项集
- (2) 将频繁1-项集按降序排序
- (3) 再次扫描数据库，构建FP-tree

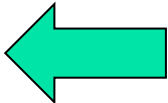
项集	支持度 计数		项集	支持度 计数		项集	支持度 计数
{I5}	3		{I5}	3		{I1}	4
{I1}	4		{I1}	4		{I3}	4
{I2}	1	→	{I3}	4		{I4}	3
{I3}	4		{I4}	3		{I5}	3
{I6}	3		{I6}	3		{I6}	3
{I4}	3						
{I7}	2						

最小支持度计数设为3

TID (事务ID)	商品ID列表
T001	{I5, I1, I2, I3, I6}
T002	{I4, I5, I1, I3}
T003	{I1, I4, I7}
T004	{I5, I1, I3, I6, I7}
T005	{I4, I3, I6}

(3) 再次扫描数据库，构建FP-tree

项集	支持度计数	节点链
{I1}	4	
{I3}	4	
{I4}	3	
{I5}	3	
{I6}	3	



项集	支持度计数
{I1}	4
{I3}	4
{I4}	3
{I5}	3
{I6}	3

原事务数据库

TID	商品ID列表
T001	{I5, I1, I2, I3, I6}
T002	{I4, I5, I1, I3}
T003	{I1, I4, I7}
T004	{I5, I1, I3, I6, I7}
T005	{I4, I3, I6}



剔除非频繁项、并排序!

TID	商品ID列表
T001	
T002	
T003	
T004	
T005	

(3) 再次扫描数据库，构建FP-tree

项集	支持 度计 数	节点 链
{I1}	4	
{I3}	4	
{I4}	3	
{I5}	3	
{I6}	3	

原事务数据库

TID	商品ID列表
T001	{I5, I1, I2, I3, I6}
T002	{I4, I5, I1, I3}
T003	{I1, I4, I7}
T004	{I5, I1, I3, I6, I7}
T005	{I4, I3, I6}



剔除非频繁项、并排序!






TID	商品ID列表
T001	{I1, I3, I5, I6}
T002	{I1, I3, I4, I5}
T003	{I1, I4}
T004	{I1, I3, I5, I6}
T005	{I3, I4, I6}

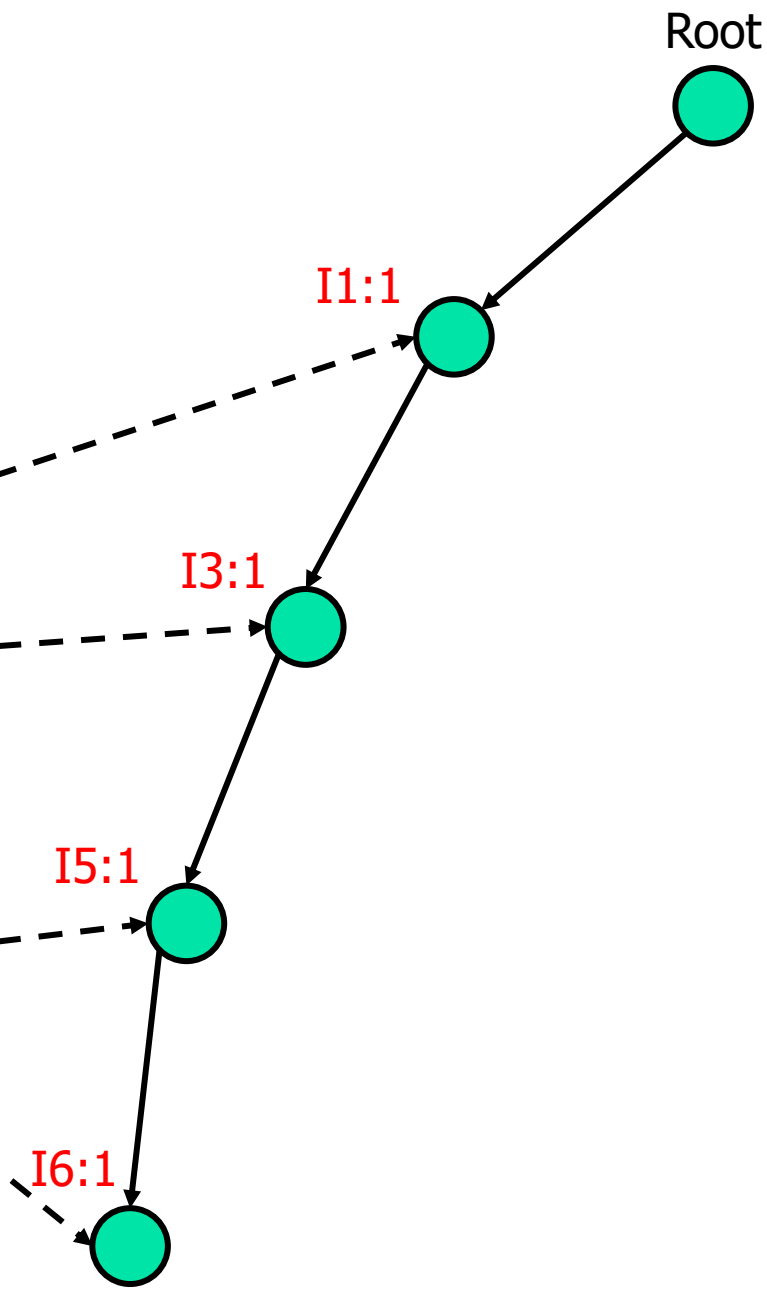
(3) 再次扫描数据库，构建FP-tree

项集	支持 度计 数	节点 链
{I1}	4	
{I3}	4	
{I4}	3	
{I5}	3	
{I6}	3	






TID (事务ID)	商品ID列表
T001	{I1, I3, I5, I6}
T002	{I1, I3, I4, I5}
T003	{I1, I4}
T004	{I1, I3, I5, I6}
T005	{I3, I4, I6}

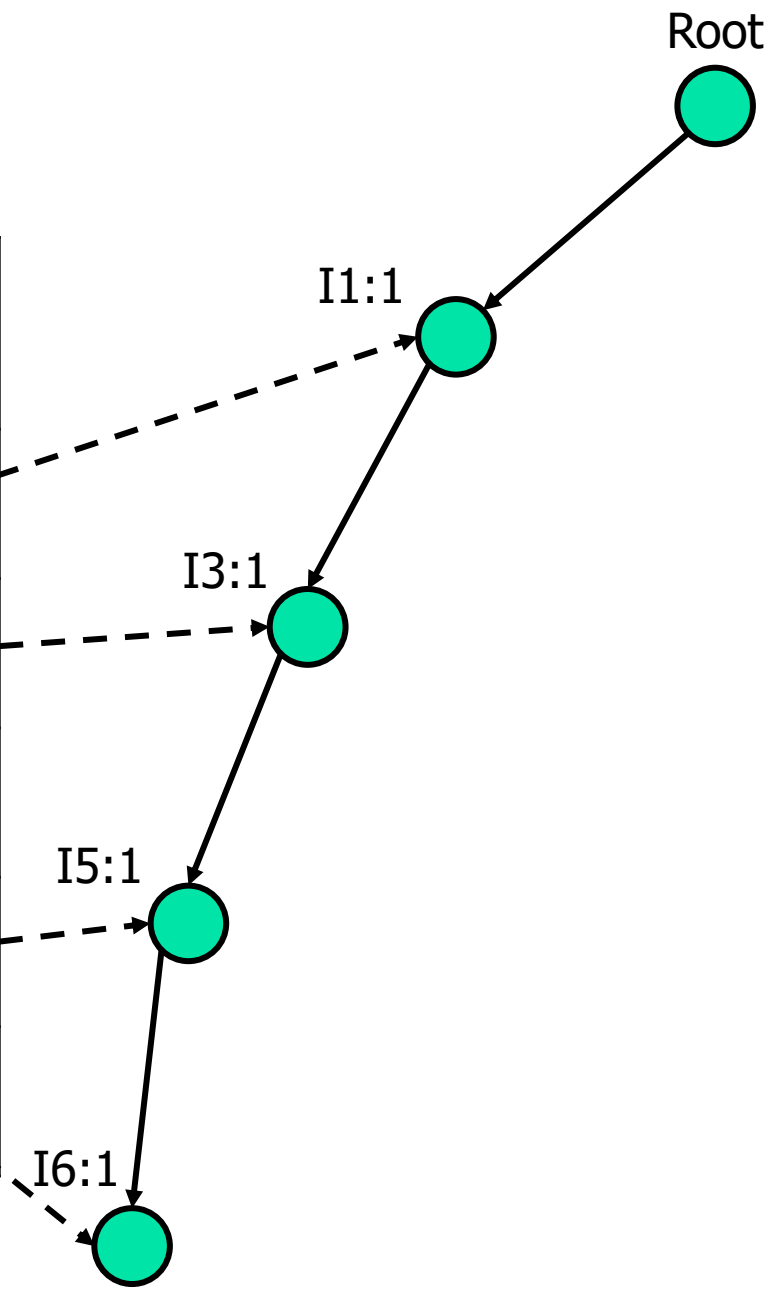


项集	支持度 计数	节点链
{I1}	4	
{I3}	4	
{I4}	3	
{I5}	3	
{I6}	3	








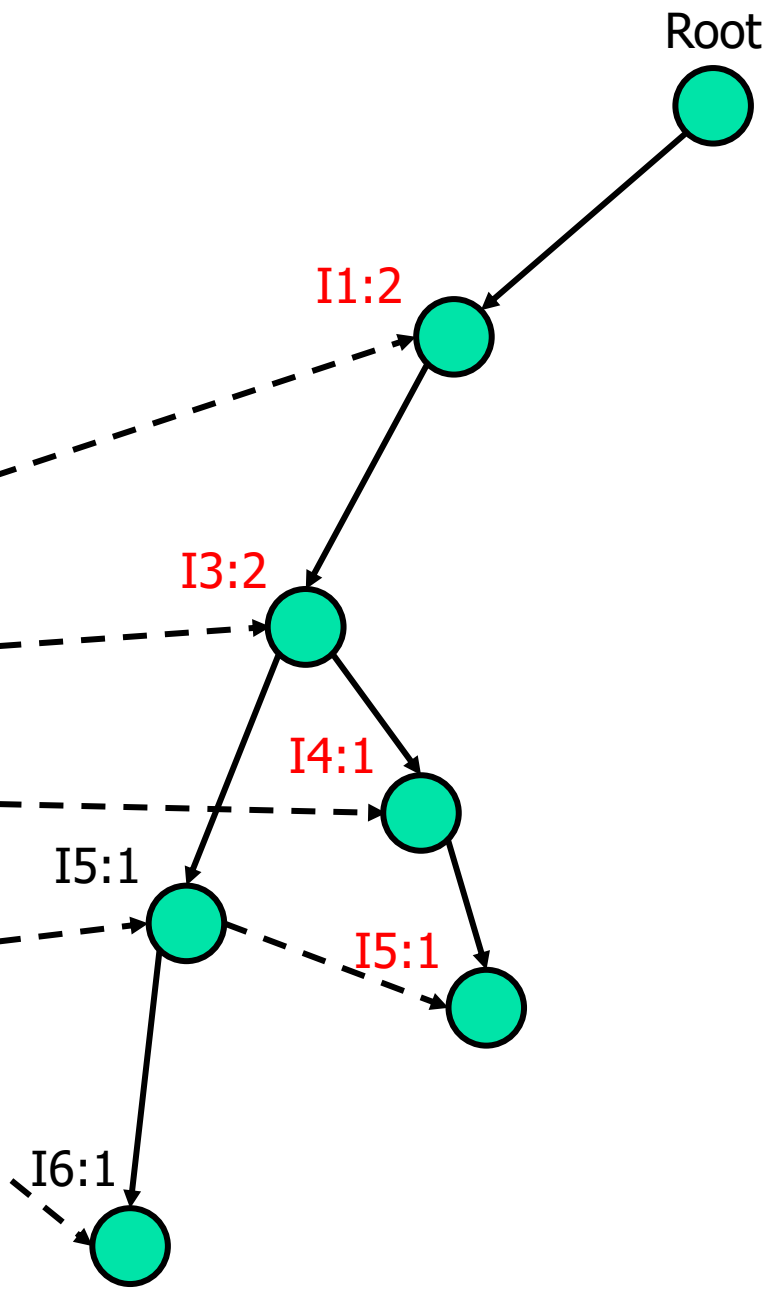
TID (事务ID)	商品ID列表
<b>T001</b>	<b>{I1, I3, I5, I6}</b>
T002	{I1, I3, I4, I5}
T003	{I1, I4}
T004	{I1, I3, I5, I6}
T005	{I3, I4, I6}

项集	支持度 计数	节点链
{I1}	4	
{I3}	4	
{I4}	3	
{I5}	3	
{I6}	3	








TID (事务ID)	商品ID列表
T001	{I1, I3, I5, I6}
<b>T002</b>	<b>{I1, I3, I4, I5}</b>
T003	{I1, I4}
T004	{I1, I3, I5, I6}
T005	{I3, I4, I6}

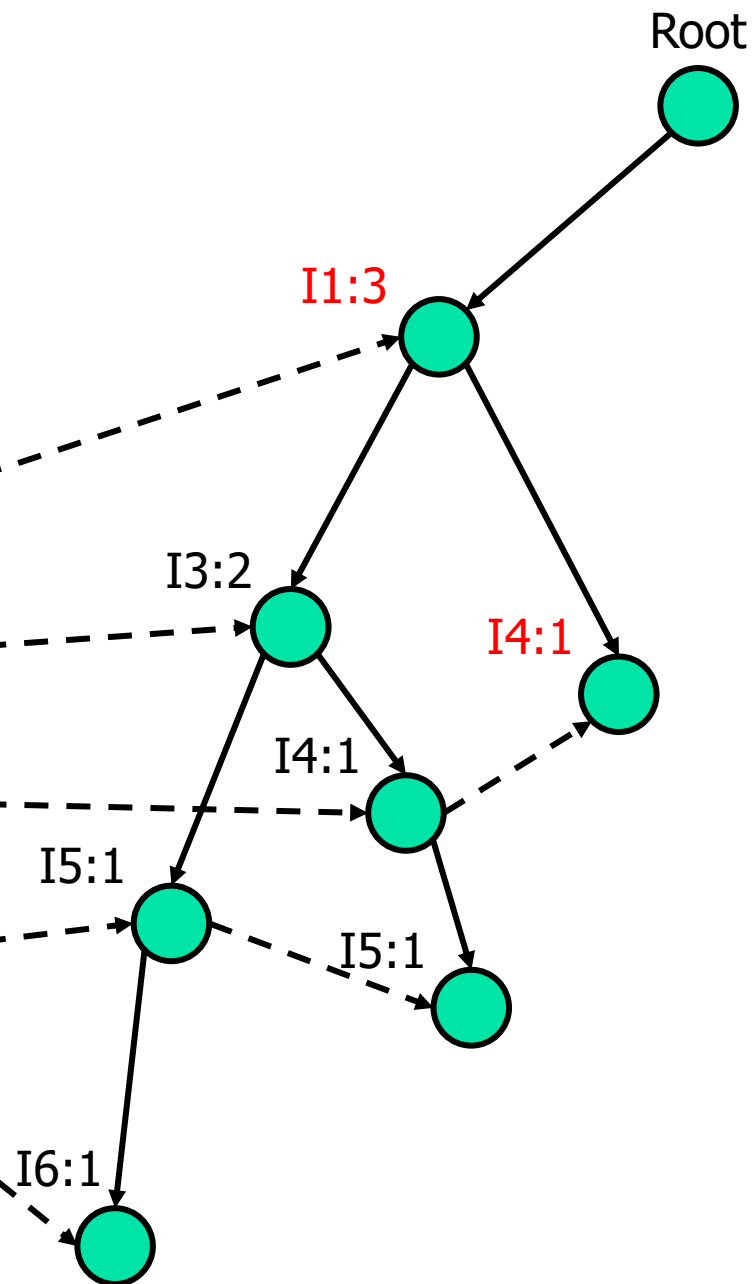
项集	支持度 计数	节点链
{I1}	4	
{I3}	4	
{I4}	3	
{I5}	3	
{I6}	3	








TID (事务ID)	商品ID列表
T001	{I1, I3, I5, I6}
<b>T002</b>	<b>{I1, I3, I4, I5}</b>
T003	{I1, I4}
T004	{I1, I3, I5, I6}
T005	{I3, I4, I6}

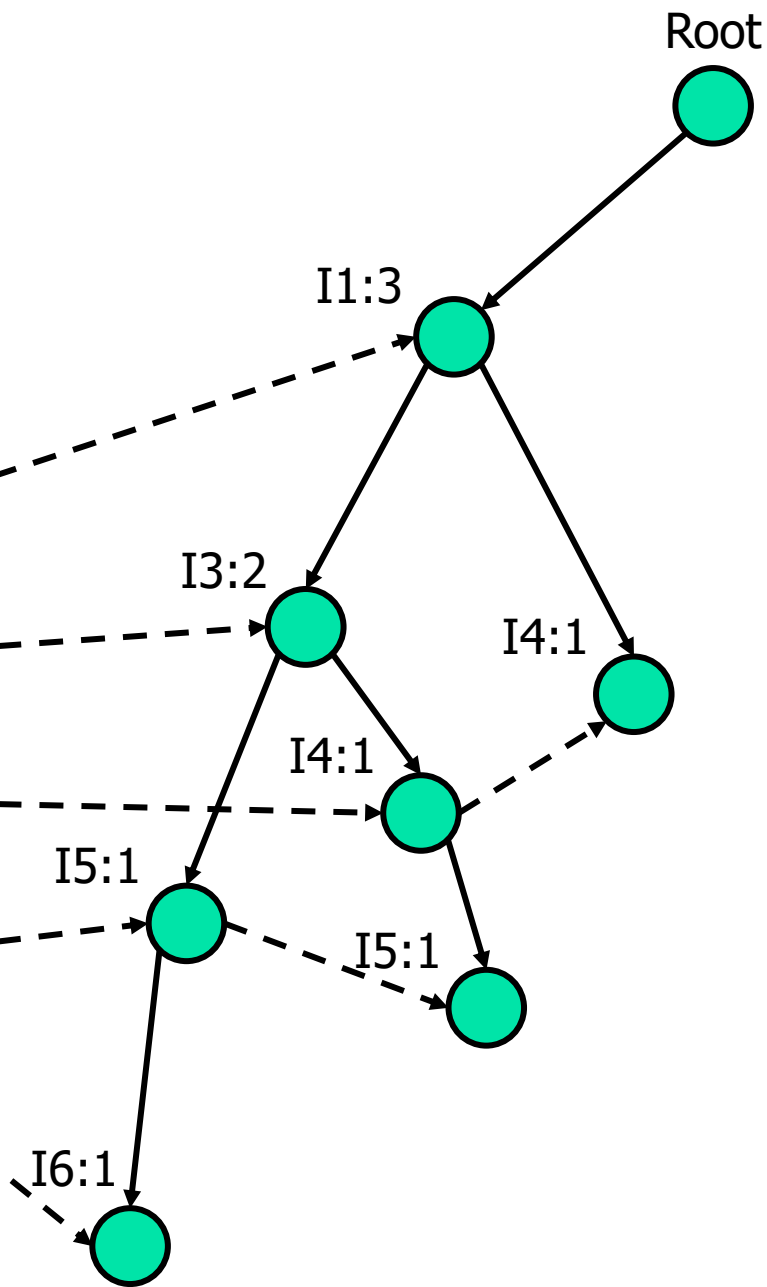


项集	支持度 计数	节点链
{I1}	4	
{I3}	4	
{I4}	3	
{I5}	3	
{I6}	3	








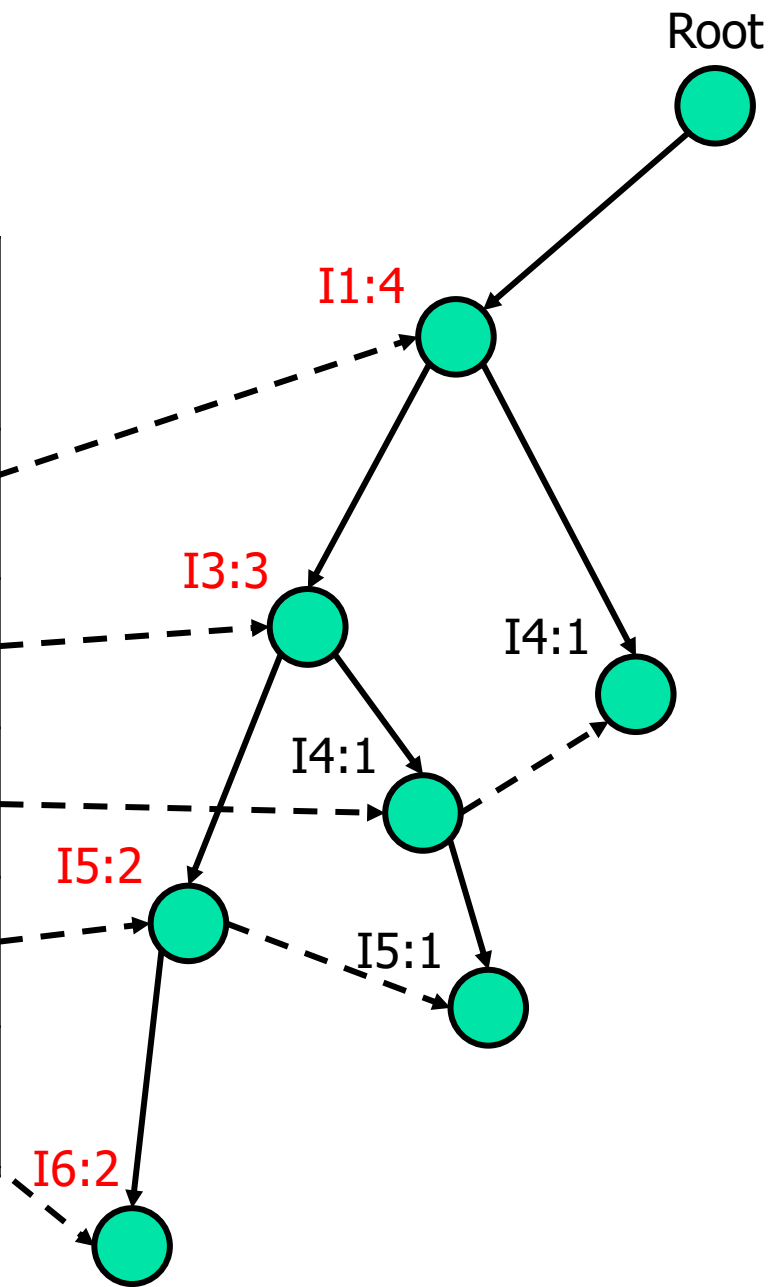
TID (事务ID)	商品ID列表
T001	{I1, I3, I5, I6}
T002	{I1, I3, I4, I5}
<b>T003</b>	<b>{I1, I4}</b>
T004	{I1, I3, I5, I6}
T005	{I3, I4, I6}

项集	支持度 计数	节点链
{I1}	4	
{I3}	4	
{I4}	3	
{I5}	3	
{I6}	3	








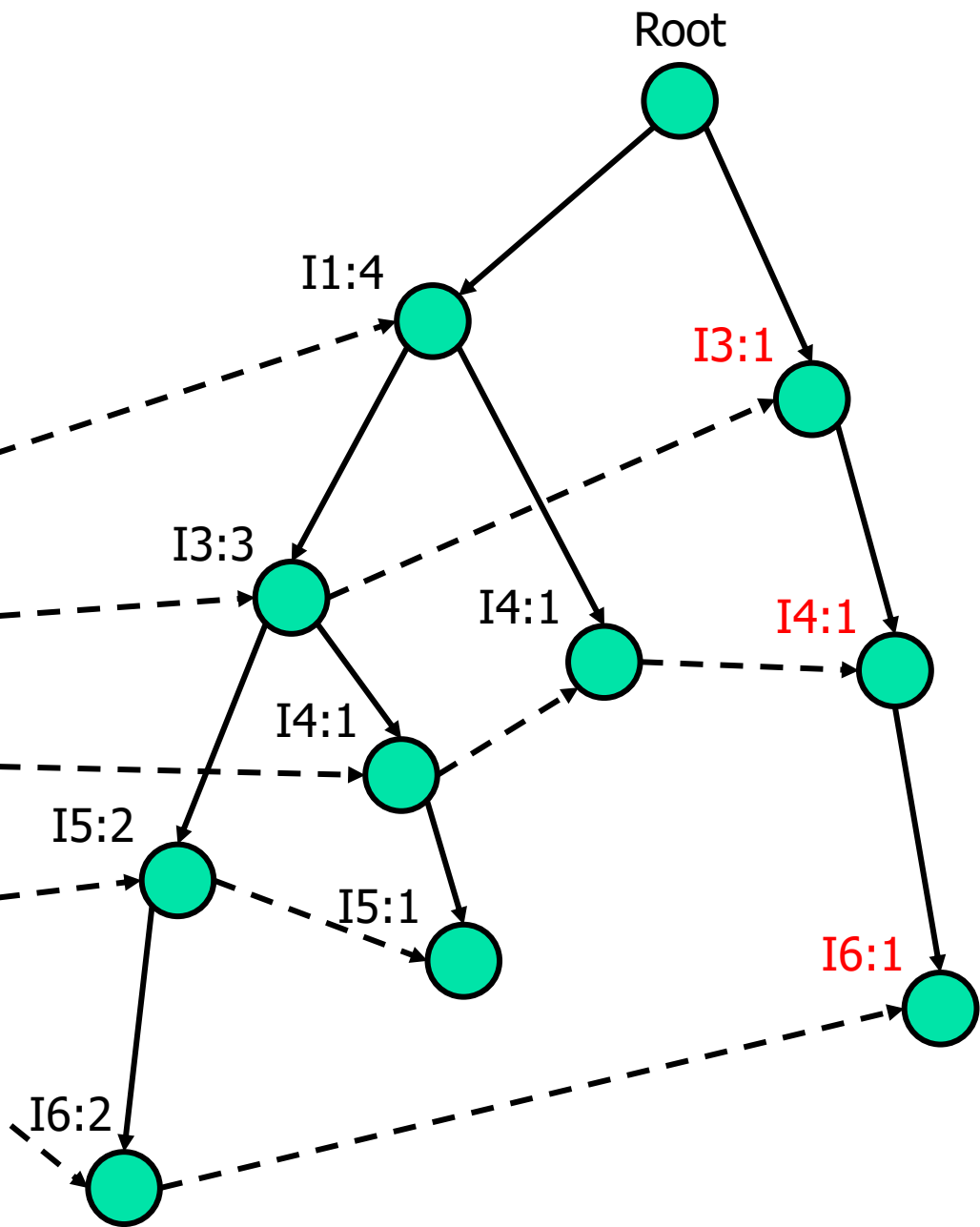
TID (事务ID)	商品ID列表
T001	{I1, I3, I5, I6}
T002	{I1, I3, I4, I5}
T003	{I1, I4}
<b>T004</b>	<b>{I1, I3, I5, I6}</b>
T005	{I3, I4, I6}

项集	支持度 计数	节点链
{I1}	4	
{I3}	4	
{I4}	3	
{I5}	3	
{I6}	3	



TID (事务ID)	商品ID列表
T001	{I1, I3, I5, I6}
T002	{I1, I3, I4, I5}
T003	{I1, I4}
<b>T004</b>	<b>{I1, I3, I5, I6}</b>
T005	{I3, I4, I6}

项集	支持度 计数	节点链
{I1}	4	
{I3}	4	
{I4}	3	
{I5}	3	
{I6}	3	



TID (事务ID)	商品ID列表
T001	{I1, I3, I5, I6}
T002	{I1, I3, I4, I5}
T003	{I1, I4}
T004	{I1, I3, I5, I6}
<b>T005</b>	<b>{I3, I4, I6}</b>

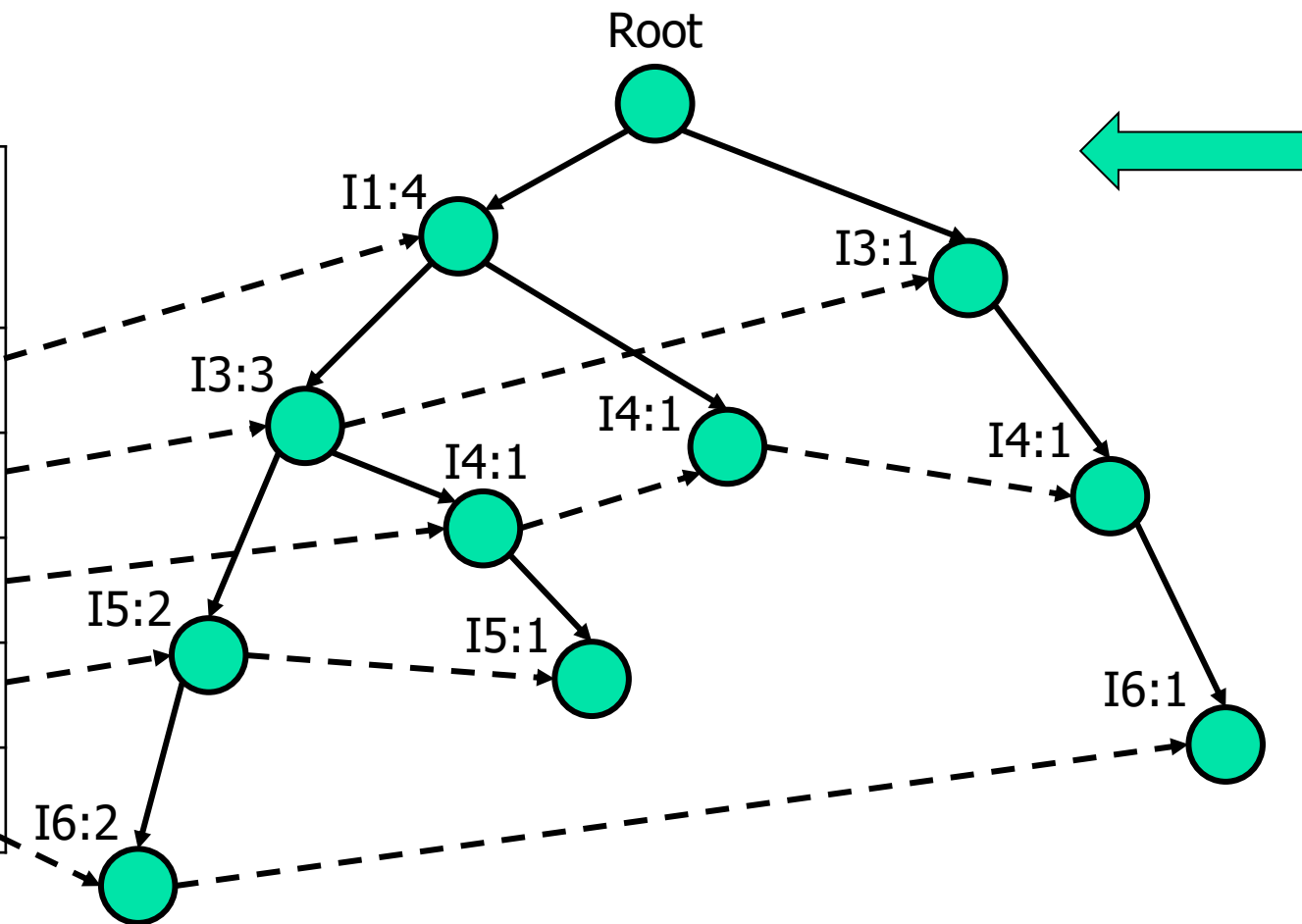


# FP-tree的特性

- 完整性 (Completeness) : 对于频繁模式发现来说
- 紧凑性 (Compactness)
  - 能有效压缩事务数据库
  - 压缩比可超100倍

# 如何从FP-tree中发现频繁项集？

项集	支持度计数	节点链
{I1}	4	
{I3}	4	
{I4}	3	
{I5}	3	
{I6}	3	



TID	商品ID列表
T001	{I5, I1, I2, I3, I6}
T002	{I4, I5, I1, I3}
T003	{I1, I4, I7}
T004	{I5, I1, I3, I6, I7}
T005	{I4, I3, I6}

➤ 结点链的作用：找到条件数据库！

# 从FP-tree中生成频繁项集的算法

- **分治法：**将挖掘全体频繁项集的问题分为**多个子问题**

--挖掘以I6结尾的频繁项集

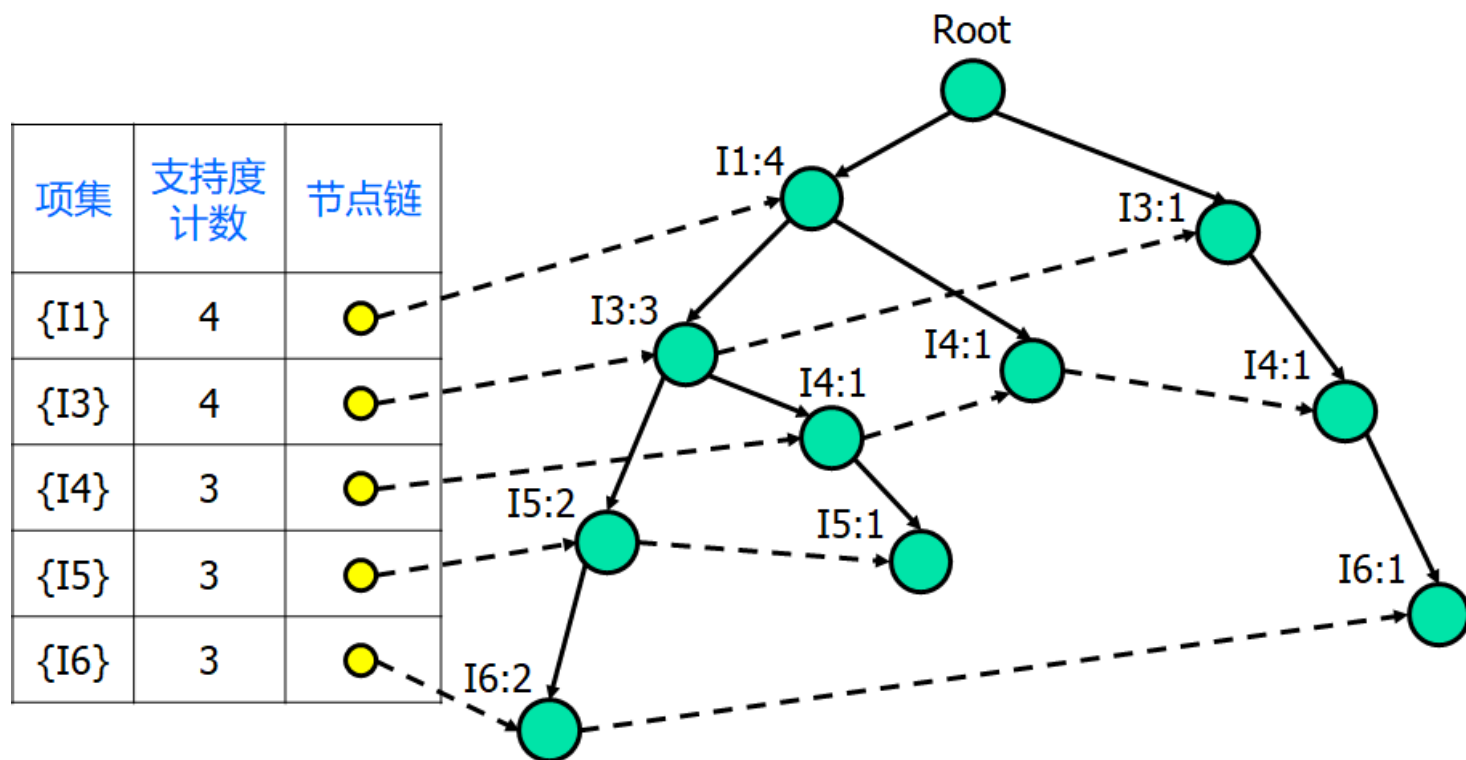
--挖掘以I5结尾的频繁项集

--挖掘以I4结尾的频繁项集

--挖掘以I3结尾的频繁项集

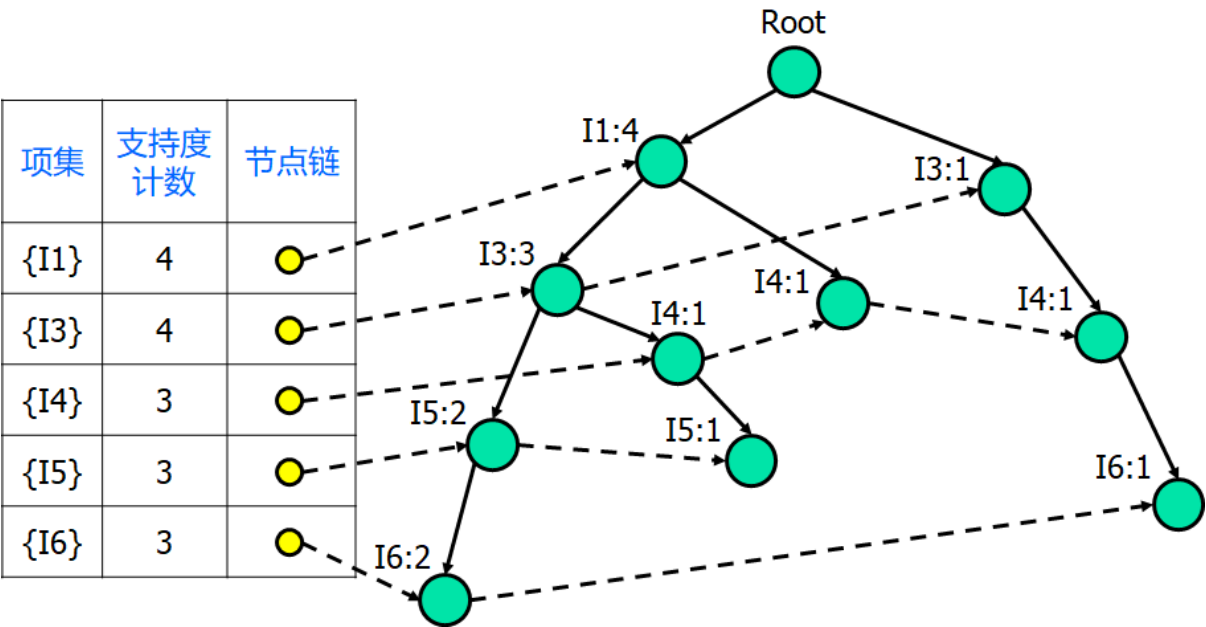
--挖掘以I1结尾的频繁项集

- **递归法：**求解每个子问题



# Procedure FP-growth(*Tree*, $\alpha$ )

- if *Tree* 只包含单个分支 *P* then
  - for each  $\theta$  (节点组合) in  $C_\theta$  (*P* 中节点的全部组合)
    - ✓ 生成新的频繁项集:  $\beta = \theta \cup \alpha$
- else for each entry  $e_i$  in head table (头部表)
  - 生成新的频繁项集:  $\beta = e_i \cup \alpha$
  - 构建  $\beta$  的条件数据库:  $D_\beta$
  - 基于  $D_\beta$  构建  $\beta$  的条件FP-tree: *Tree* $_\beta$
  - If *Tree* $_\beta \neq \emptyset$  then
    - ✓ 递归调用 **FP\_growth**(*Tree* $_\beta$ ,  $\beta$ )
  - 递归的停止条件:
    - if *Tree* 只包含单个分支 *P*
    - if *Tree* $_\beta = \emptyset$



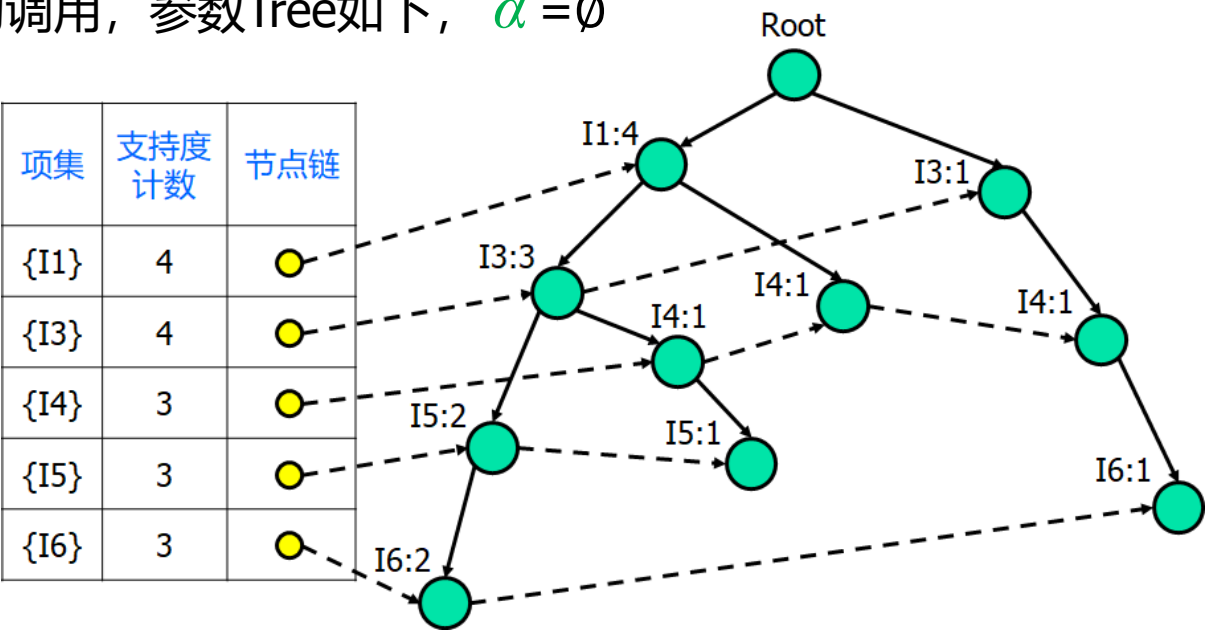
# Procedure FP-growth(Tree, $\alpha$ )

最初调用，参数Tree如下，  $\alpha = \emptyset$

- if  $Tree$  只包含单个分支  $P$  then
  - for each  $\theta$  (节点组合) in  $C_\theta$  ( $P$  中节点的全部组合)
    - 生成新的频繁项集:  $\beta = \theta \cup \alpha$
- else for each entry  $e_i$  in head table (头部表)
  - 生成新的频繁项集:  $\beta = e_i \cup \alpha$
  - 构建  $\beta$  的条件数据库:  $D_\beta$
  - 基于  $D_\beta$  构建  $\beta$  的条件FP-tree:  $Tree_\beta$
  - If  $Tree_\beta \neq \emptyset$  then
    - 递归调用  $FP\_growth(Tree_\beta, \beta)$

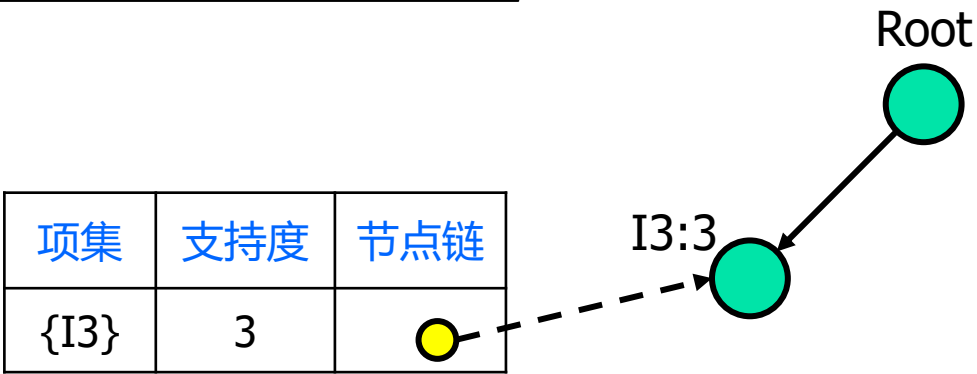
$e_i = \{I6\}$        $\beta = e_i \cup \alpha = \{I6\}$

第1个子问题: 挖掘以I6结尾的频繁项集



$D_\beta$ :  $\{I1, I3, I5\}:2, \{I3, I4\}:1$

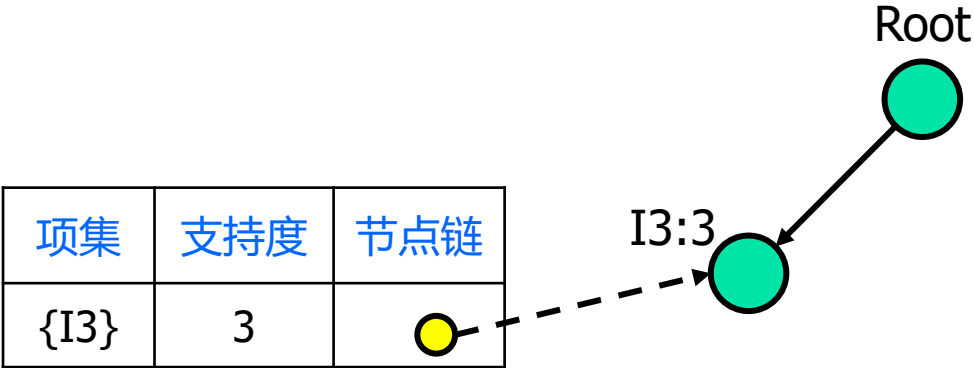
$Tree_\beta$ :



# Procedure FP-growth(Tree, $\alpha$ )

递归调用，参数Tree如下，  $\alpha = \{I6\}$

- if  $Tree$  只包含单个分支  $P$  then
  - for each  $\theta$  (节点组合) in  $C_\theta$  ( $P$  中节点的全部组合)
    - ✓ 生成新的频繁项集:  $\beta = \theta \cup \alpha$
- else for each entry  $e_i$  in head table (头部表)
  - 生成新的频繁项集:  $\beta = e_i \cup \alpha$
  - 构建  $\beta$  的条件数据库:  $D_\beta$
  - 基于  $D_\beta$  构建  $\beta$  的条件FP-tree:  $Tree_\beta$
  - If  $Tree_\beta \neq \emptyset$  then
    - ✓ 递归调用  $FP\_growth(Tree_\beta, \beta)$



生成新的频繁项集:

$$\theta \cup \alpha = \{I3\} \cup \{I6\} = \{I3, I6\}$$

➢ 解决第1个子问题: 挖掘以I6结尾的频繁项集

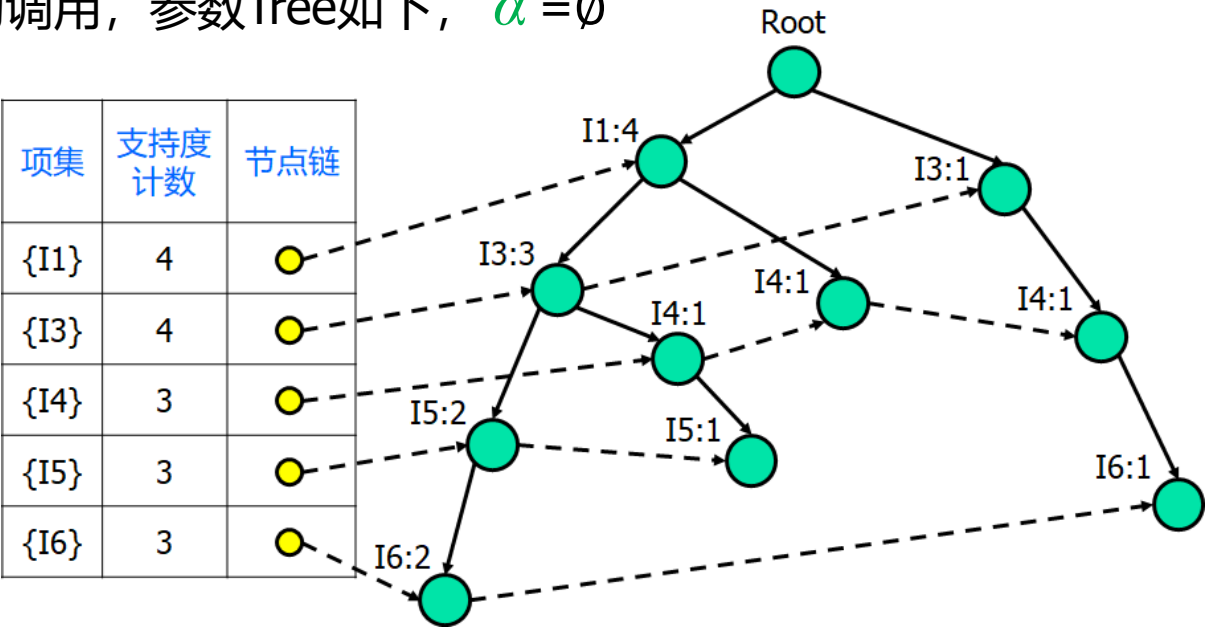
# Procedure FP-growth(Tree, $\alpha$ )

最初调用，参数Tree如下，  $\alpha = \emptyset$

- if  $Tree$  只包含单个分支  $P$  then
  - for each  $\theta$  (节点组合) in  $C_\theta$  ( $P$  中节点的全部组合)
    - 生成新的频繁项集:  $\beta = \theta \cup \alpha$
- else for each entry  $e_i$  in head table (头部表)
  - 生成新的频繁项集:  $\beta = e_i \cup \alpha$
  - 构建  $\beta$  的条件数据库:  $D_\beta$
  - 基于  $D_\beta$  构建  $\beta$  的条件FP-tree:  $Tree_\beta$
  - If  $Tree_\beta \neq \emptyset$  then
    - 递归调用  $FP\_growth(Tree_\beta, \beta)$

$e_i = \{I5\}$        $\beta = e_i \cup \alpha = \{I5\}$

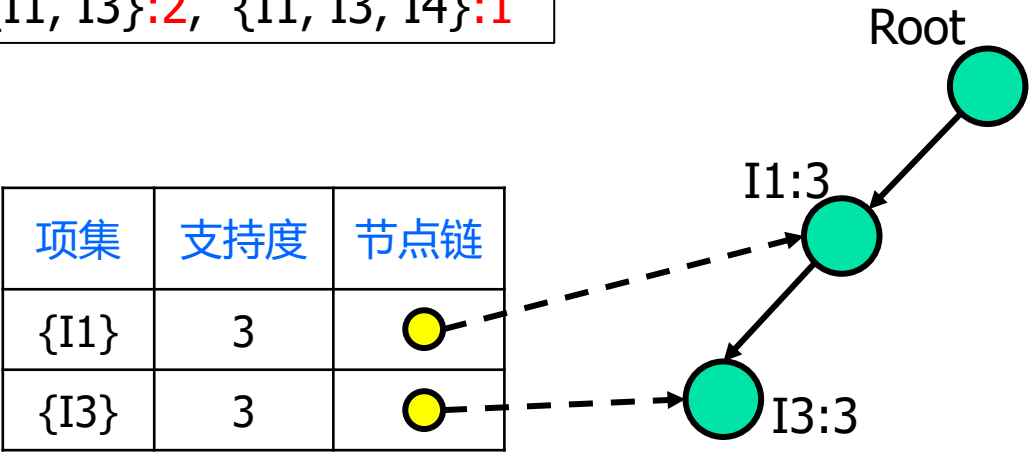
第2个子问题: 挖掘以I5结尾的频繁项集



$D_\beta$ :

{I1, I3}:2, {I1, I3, I4}:1
----------------------------

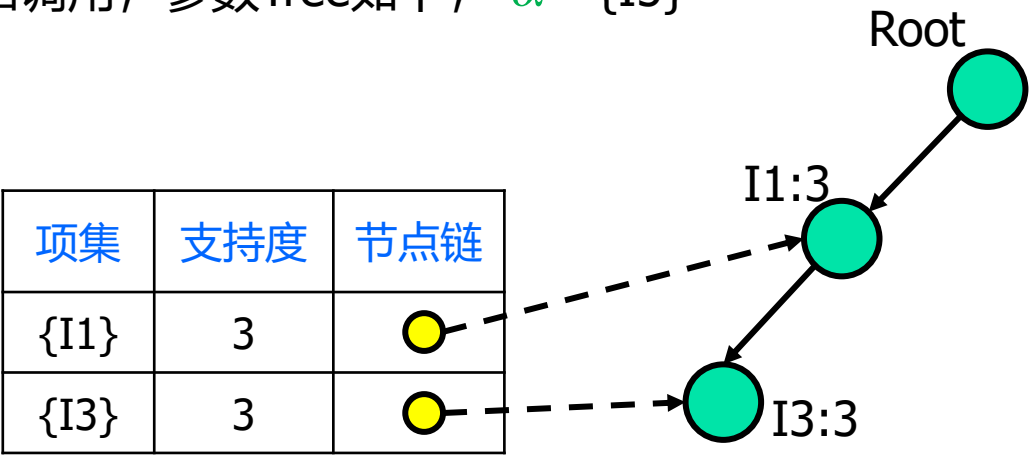
$Tree_\beta$ :



# Procedure FP-growth(Tree, $\alpha$ )

递归调用，参数Tree如下，  $\alpha = \{I5\}$

- if  $Tree$  只包含单个分支  $P$  then
    - for each  $\theta$  (节点组合) in  $C_\theta$  ( $P$ 中节点的全部组合)
      - ✓ 生成新的频繁项集:  $\beta = \theta \cup \alpha$
  - else for each entry  $e_i$  in head table (头部表)
    - 生成新的频繁项集:  $\beta = e_i \cup \alpha$
    - 构建  $\beta$  的条件数据库:  $D_\beta$
    - 基于  $D_\beta$  构建  $\beta$  的条件FP-tree:  $Tree_\beta$
    - If  $Tree_\beta \neq \emptyset$  then
      - ✓ 递归调用  $FP\_growth(Tree_\beta, \beta)$
- 解决第2个子问题: 挖掘以I5结尾的频繁项集



生成新的频繁项集:

$$\theta \cup \alpha = \{I3\} \cup \{I5\} = \{I3, I5\}$$

$$\theta \cup \alpha = \{I1\} \cup \{I5\} = \{I1, I5\}$$

$$\theta \cup \alpha = \{I1, I3\} \cup \{I5\} = \{I1, I3, I5\}$$



- 接着依次求解以I4、I3、I1结尾的频繁项集集合，得到最终结果

后缀项	条件数据库	挖掘到的频繁项集
{I6}	{I1, I3, I5}:2, {I3, I4}:1	{I3, I6}
{I5}	{I1, I3}:2, {I1, I3, I4}:1	{I1, I5}, {I3, I5}, {I1, I3, I5}
{I4}	{I1}:1, {I3}:1, {I1, I3}:1	\
{I3}	{I1}:3	{I1, I3}
{I1}	\	\

L1	{I1}, {I3}, {I4}, {I5}, {I6}
L2	{I1, I3}, {I1, I5}, {I3, I5}, {I3, I6}
L3	{I1, I3, I5}

# FP-tree算法的特点分析

- 优点

- 使用一个高度压缩的数据结构存储了事务数据库的信息，整个过程只需扫描两次数据集，相关研究表明，在挖掘某些事务数据集时，FP-tree算法比Apriori算法快多个数量级。

- 缺点

- 由于FP-tree算法在执行过程中需要递归生成条件数据库和条件FP-tree，所以内存开销较大，且当生成的FP-tree十分茂盛时，如满前缀树，算法产生的子问题数量会剧增，导致性能显著下降。

# 规则的相关性评价指标

# 相关性评价指标的分类

- 客观兴趣度度量

- 支持度-置信度

- 提升度

- ...

- 主观兴趣度度量

- 因用户而异，来自领域专家经验

# 支持度-置信度框架的缺陷

- 例：假设  $min-sup = 10\%$ ,  $min-conf = 70\%$
- 考虑规则：{爱喝牛奶}  $\rightarrow$  {爱喝咖啡}
- 其支持度15%，置信度75%，为强关联规则

	爱喝咖啡	不爱喝咖啡	总计
爱喝牛奶	150	50	200
不爱喝牛奶	650	150	800
总计	800	200	1000

- 若一个人爱喝牛奶，则他爱喝咖啡的可能性反而从80%下降到75%：负相关！
- “置信度”作为评价指标可能具有误导性！

$$c(X \rightarrow Y) = \frac{\sigma(XY)}{\sigma(X)}$$

- 改进支持度-置信度框架

- 引入提升度度量(lift)

- ✓ 对规则  $X \rightarrow Y$ ,  $P(X)$ 表示X发生的概率

$$lift(X, Y) = \frac{P(X \cup Y)}{P(X) \times P(Y)}$$

- ✓ 当规则中有n项时

$$lift(X_1, X_2, \dots, X_n) = \frac{P(X_1 \cup X_2 \cup \dots \cup X_n)}{P(X_1) \times P(X_2) \times \dots \times P(X_n)}$$

- 提升度度量的评判准则：

$$\text{lift}(X, Y) \begin{cases} = 1, & X \text{ 和 } Y \text{ 是独立的} \\ > 1, & X \text{ 和 } Y \text{ 呈正相关} \\ < 1, & X \text{ 和 } Y \text{ 呈负相关} \end{cases}$$

	爱喝咖啡	不爱喝咖啡	总计
爱喝牛奶	150	50	200
不爱喝牛奶	650	150	800
总计	800	200	1000

- 重新考察规则：{爱喝牛奶} → {爱喝咖啡}

$$\text{lift}(X, Y) = \frac{P(X \cup Y)}{P(X) \times P(Y)} = \frac{0.15}{0.2 \times 0.8} = 0.9375$$

提升度小于1，拒绝该规则！

- 为什么可以做到？

$$c(X \rightarrow Y) = \frac{\sigma(\textcolor{red}{XY})}{\sigma(X)}$$

- 其它相关性评价指标:

- $\chi^2$  度量
- 全置信度 (all-confidence)
- 最大置信度(max-confidence)
- 余弦度量(cosine)
- Kulczynski (Kulc) 度量
- ...



•  $\chi^2$  度量

➢ 对规则  $X \rightarrow Y$ , 假设两者统计独立

$$\chi^2 = \sum \frac{(\text{观测值} - \text{期望值})^2}{\text{期望值}}$$

• 评价准则:

- 计算自由度, 并设定显著性水平  $\alpha$
- 查询  $\chi^2$  分布的临界值表
- 将计算的  $\chi^2$  值与临界值比较, 判断是否拒绝  
关联规则

	爱喝咖啡	不爱喝咖啡	总计
爱喝牛奶	150 (160)	50 (40)	200
不爱喝牛奶	650 (640)	150 (160)	800
总计	800	200	1000

自由度  $n=1$ , 设定  $\alpha=0.05$ , 此时卡方临界值是 3.84

3.90625 > 3.84, 拒绝原假设!

观测值 150 小于期望值 160, 说明两者是负相关的

$$\chi^2 = \frac{(150 - 160)^2}{160} + \frac{(50 - 40)^2}{40} + \frac{(650 - 640)^2}{640} + \frac{(150 - 160)^2}{160} = 3.90625$$

- 全置信度:  $all\_conf(W) = \frac{\sup(W)}{\max\_item\_sup(W)} \quad W = XY$

- 最大置信度:  $max\_conf(X, Y) = \max\{P(X|Y), P(Y|X)\}$

- 余弦度量:  $consine(X, Y) = \frac{P(X \cup Y)}{\sqrt{P(X) \times P(Y)}} = \frac{\sup(X \cup Y)}{\sqrt{\sup(X) \times \sup(Y)}}$

- Kulczynski度量:  $Kulc(X, Y) = \frac{1}{2}(P(X|Y) + P(Y|X))$

➤ 四种度量取值范围为[0, 1], 评判标准:

$$\begin{cases} = 0.5, & X \text{和} Y \text{中性关联} \\ > 0.5, & X \text{和} Y \text{呈正相关} \\ < 0.5, & X \text{和} Y \text{呈负相关} \end{cases}$$

## 零记录（ null-transactions ）问题

Set	mc	$\overline{m}C$	$m\overline{C}$	$\overline{m}\overline{C}$	$\chi^2$	lift	all_conf.	max_conf.	Kulc.	cosine
$D_1$	10,000	1000	1000	100,000	90557	9.26	0.91	0.91	0.91	0.91
$D_2$	10,000	1000	1000	100	0	1	0.91	0.91	0.91	0.91
$D_3$	100	1000	1000	100,000	670	8.44	0.09	0.09	0.09	0.09
$D_4$	1000	1000	1000	100,000	24740	25.75	0.5	0.5	0.5	0.5
$D_5$	1000	100	10,000	100,000	8173	9.18	0.09	0.91	0.5	0.29
$D_6$	1000	10	100,000	100,000	965	1.97	0.01	0.99	0.5	0.10

$$lift(X, Y) = \frac{P(X \cup Y)}{P(X) \times P(Y)}$$

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

# 具有零记录不变性（ Null-invariant ）的相关性度量指标

- 全置信度:  $all\_conf(W) = \frac{\sup(W)}{\max\_item\_sup(W)} \quad W = XY$
- 最大置信度:  $max\_conf(X, Y) = \max\{P(X|Y), P(Y|X)\}$
- 余弦度量  $consine(X, Y) = \frac{P(X \cup Y)}{\sqrt{P(X) \times P(Y)}} = \frac{\sup(X \cup Y)}{\sqrt{\sup(X) \times \sup(Y)}}$
- Kulczynski度量:  $Kulc(X, Y) = \frac{1}{2} (P(X|Y) + P(Y|X))$

Set	mc	$\overline{m}C$	$m\overline{C}$	$\overline{\overline{m}C}$	$\chi^2$	lift	all_conf.	max_conf.	Kulc.	cosine
$D_1$	10,000	1000	1000	100,000	90557	9.26	0.91	0.91	0.91	0.91
$D_2$	10,000	1000	1000	100	0	1	0.91	0.91	0.91	0.91
$D_3$	100	1000	1000	100,000	670	8.44	0.09	0.09	0.09	0.09
$D_4$	1000	1000	1000	100,000	24740	25.75	0.5	0.5	0.5	0.5
$D_5$	1000	100	10,000	100,000	8173	9.18	0.09	0.91	0.5	0.29
$D_6$	1000	10	100,000	100,000	965	1.97	0.01	0.99	0.5	0.10

$$all\_conf(W) = \frac{\sup(W)}{\max\_item\_sup(W)}$$

$$max\_conf(X, Y) = \max\{P(X|Y), P(Y|X)\}$$

$$consine(X, Y) = \frac{\sup(X \cup Y)}{\sqrt{\sup(X) \times \sup(Y)}}$$

$$Kulc(X, Y) = \frac{1}{2}(P(X|Y) + P(Y|X))$$

# 相关性度量中的不平衡数据问题

Set	mc	$\bar{m}C$	$m\bar{C}$	$\overline{mC}$	$\chi^2$	lift	all_conf.	max_conf.	Kulc.	cosine
$D_1$	10,000	1000	1000	100,000	90557	9.26	0.91	0.91	0.91	0.91
$D_2$	10,000	1000	1000	100	0	1	0.91	0.91	0.91	0.91
$D_3$	100	1000	1000	100,000	670	8.44	0.09	0.09	0.09	0.09
$D_4$	1000	1000	1000	100,000	24740	25.75	0.5	0.5	0.5	0.5
$D_5$	1000	100	10,000	100,000	8173	9.18	0.09	0.91	0.5	0.29
$D_6$	1000	10	100,000	100,000	965	1.97	0.01	0.99	0.5	0.10

➤ 解决方法：引入不平衡比 (Imbalance Ratio)

$$IR(A, B) = \frac{|\sup(A) - \sup(B)|}{\sup(A) + \sup(B) - \sup(A \cup B)}$$

# 各种相关性度量指标的特点总结

- 提升度 (**lift**) 和  $\chi^2$  通常不适合大型事务数据库，因为它们不具有**零记录不变性**
- *all\_confidence*, *max\_confidence*, *Kulc* 和 *cosine* 具有**零记录不变性**，适合大型事务数据库
- 对于不平衡数据，推荐将 *Kulc* 和 *IR* 配合使用

# 挖掘多维量化关联规则



# 什么是多维量化关联规则？

TID	商品列表
T001	牛奶、啤酒、尿布
T002	鸡蛋、牛奶、面包、啤酒、尿布
T003	鸡蛋、牛奶、面包
T004	面包、啤酒
T005	牛奶、面包、啤酒、尿布

TID	顾客ID	年龄	月收入	职业	购买商品列表
T001	C1	20	8000	会计	牛奶, 啤酒, 面包
T002	C1	20	8000	会计	牛奶, 面包
T003	C2	33	14000	程序员	啤酒, 尿布, 面包
T004	C2	33	14000	程序员	牛奶, 啤酒, 尿布
T005	C3	23	5000	快递员	牛奶, 尿布
T006	C5	28	12000	程序员	啤酒, 面包

**{牛奶, 面包} → {啤酒}**

**Age(X, "20-25") ∧ Salary(X, "5K-10K")**

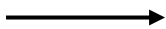
**→ buys(X, "牛奶") ∧ buys(X, "啤酒")**

某商店部分顾客购物记录数据

TID	CID	年龄	月收入/元	职业	商品列表
T001	C1	A1	S1	J1	I1, I2, I4
T002	C1	A1	S1	J1	I1, I4
T003	C2	A3	S2	J2	I2, I3, I4
T004	C2	A3	S2	J2	I1, I2, I3
T005	C3	A1	S1	J3	I1, I3
T006	C5	A2	S2	J2	I2, I4

转换后的数据表

TID	年龄	月收入/元	职业	商品列表
T001	A1	S1	J1	I1, I2, I4
T002	A1	S1	J1	I1, I4
T003	A3	S2	J2	I2, I3, I4
T004	A3	S2	J2	I1, I2, I3
T005	A1	S1	J3	I1, I3
T006	A2	S2	J2	I2, I4



列合并后的数据集

TID	项集
T001	A1, S1, J1, I1, I2, I4
T002	A1, S1, J1, I1, I4
T003	A3, S2, J2, I2, I3, I4
T004	A3, S2, J2, I1, I2, I3
T005	A1, S1, J3, I1, I3
T006	A2, S2, J2, I2, I4

TID	项集
T001	A1, S1, J1, I1, I2, I4
T002	A1, S1, J1, I1, I4
T003	A3, S2, J2, I2, I3, I4
T004	A3, S2, J2, I1, I2, I3
T005	A1, S1, J3, I1, I3
T006	A2, S2, J2, I2, I4

min-sup=3

频繁3-项集	支持度计数
A1, S1, I1	3
S2, J2, I2	3

候选1-项集	支持度计数
A1	3
A2	1
A3	2
S1	3
S2	3
J1	2
J2	3
J3	1
I1	4
I2	4
I3	3
I4	4

频繁2-项集	支持度计数
A1, S1	3
S2, J2	3
A1, I1	3
S1, I1	3
S2, I2	3
J2, I2	3
I2, I4	3

频繁1-项集	支持度计数
A1	3
S1	3
S2	3
J2	3
I1	4
I2	4
I3	3
I4	4

- 假设最小置信度阈值为0.7

得到部分满足条件的强关联规则如下：

强关联规则	置信度
$\{A1\} \rightarrow \{I1\}$	1.0
$\{S1\} \rightarrow \{I1\}$	1.0
$\{A1, S1\} \rightarrow \{I1\}$	1.0
$\{S2\} \rightarrow \{I2\}$	1.0
$\{J2\} \rightarrow \{I2\}$	1.0
$\{S2, J2\} \rightarrow \{I2\}$	1.0

将单维关联规则转化为多维关联规则

$\{A1, S1\} \rightarrow \{I1\}$



$\text{Age}(X, "20-25") \wedge \text{Salary}(X, "5K-10K") \rightarrow \text{buys}(X, "I1")$