# 大数据导论

# Introduction to Big Data

## 大数据关联规则挖掘

叶允明

计算机科学与技术学院

哈尔滨工业大学（深圳）

# 目录

# 回顾：关联规则

- **关联规则：**
  - 形如 $X \rightarrow Y$ 的蕴涵表达式

**尿布 → 啤酒**

**{牛奶, 面包} → {啤酒，尿布}**

| 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 |

$$age(\text{“25-35”}) \wedge buy(\text{“华为手机”}) \longrightarrow buy(\text{“格力空调”})$$

# 回顾：衡量关联规则“质量”

- **支持度（support）**
  - 给定数据集中$X$和$Y$的共现频度
  - 令 $W = X \cup Y$
  - $s(X \to Y) = \frac{\sigma(W)}{|T|}$
- **置信度（confidence）**
  - 在包含$X$的事务子集中$Y$出现的频繁程度
  - $c(X \to Y) = \frac{\sigma(W)}{\sigma(X)}$

$X \to Y$

| 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} |

$R_1$：I5 → I1

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

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

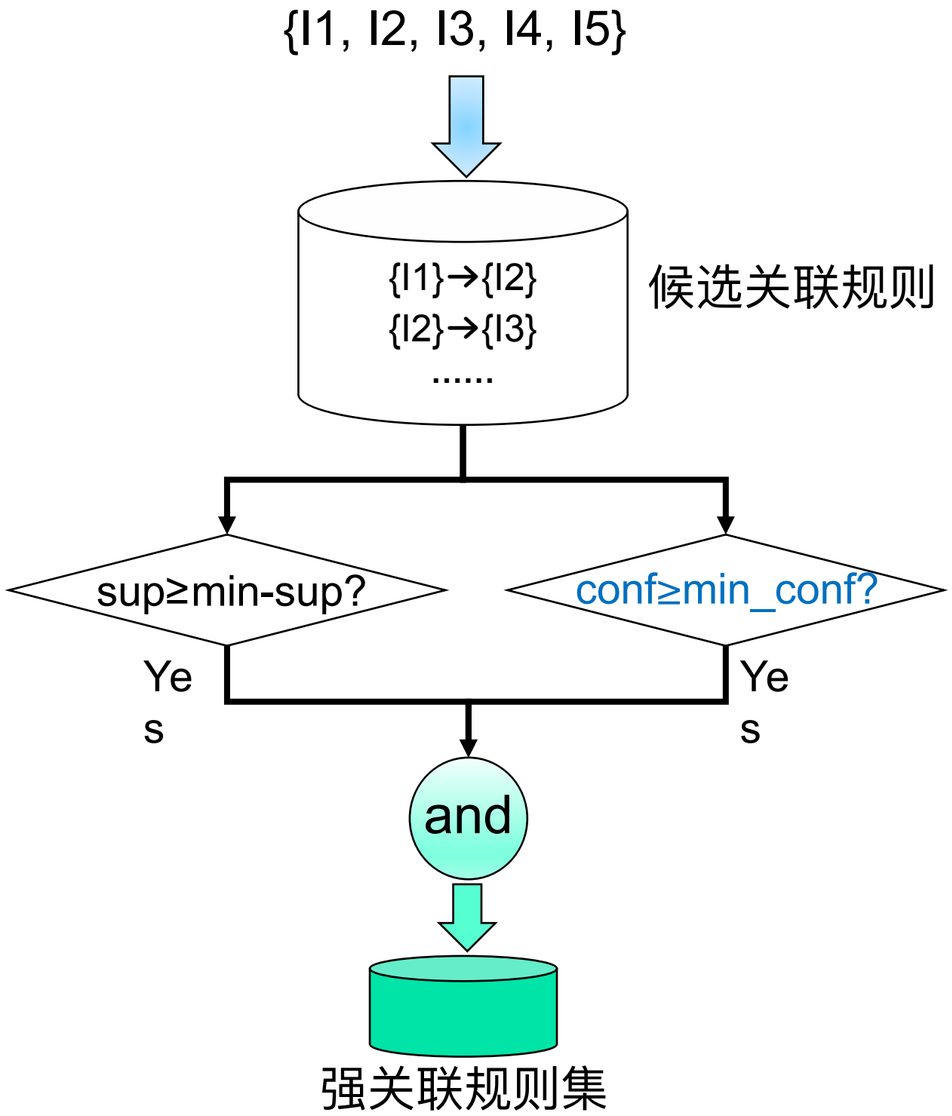
# 回顾：关联规则任务

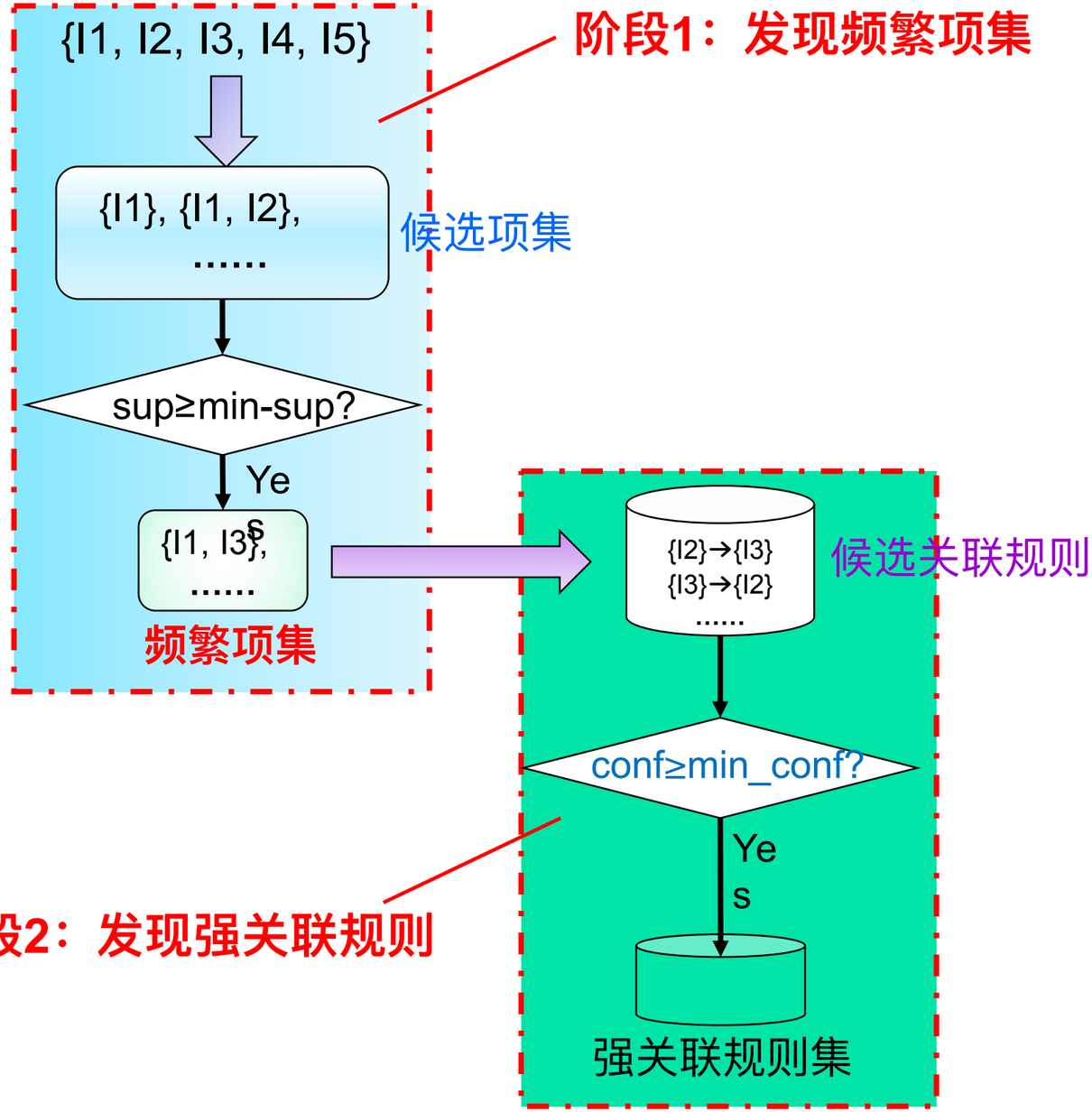
- 定义
  - 最小支持度阈值：*min-sup*，　最小置信度阈值：*min-conf*
  - 关联规则$R$ 是强关联规则，当且仅当：

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

- **关联规则挖掘任务：找到所有强关联规则！**

# 回顾：关联规则挖掘的两阶段算法框架



# 频繁项集发现： FP-tree算法

| 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} |

Database
min-sup=3

1st 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 |
|---|
| {I1, I3} |
| {I1, I4} |
| {I1, I5} |
| {I3, I4} |
| {I3, I5} |
| {I4, I5} |

C2

2nd scan

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

C2

## 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！**

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

L2

| Itemset |
|---|
| ~~{I1, I3, I4}~~ |
| {I1, I3, I5} |
| {I3, I4, I5} |

C3

3rd scan

| Itemset | sup |
|---|---|
| ~~{I1, I3, I4}~~ | ~~2~~ |
| {I1, I3, I5} | 3 |
| {I3, I4, I5} | 2 |

C3

| Itemset | sup |
|---|---|
| {I1, I3, I5} | 3 |

L3

# FP-tree：不需要生成候选项集的算法

- 也称为频繁模式生长（Frequent-Pattern Growth），FP-Growth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

**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} |

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

→

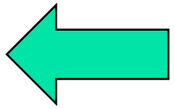
| 项集 | 支持度计数 |
|---|---|
| {I5} | 3 |
| {I1} | 4 |
| {I3} | 4 |
| {I4} | 3 |
| {I6} | 3 |

→

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

# (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

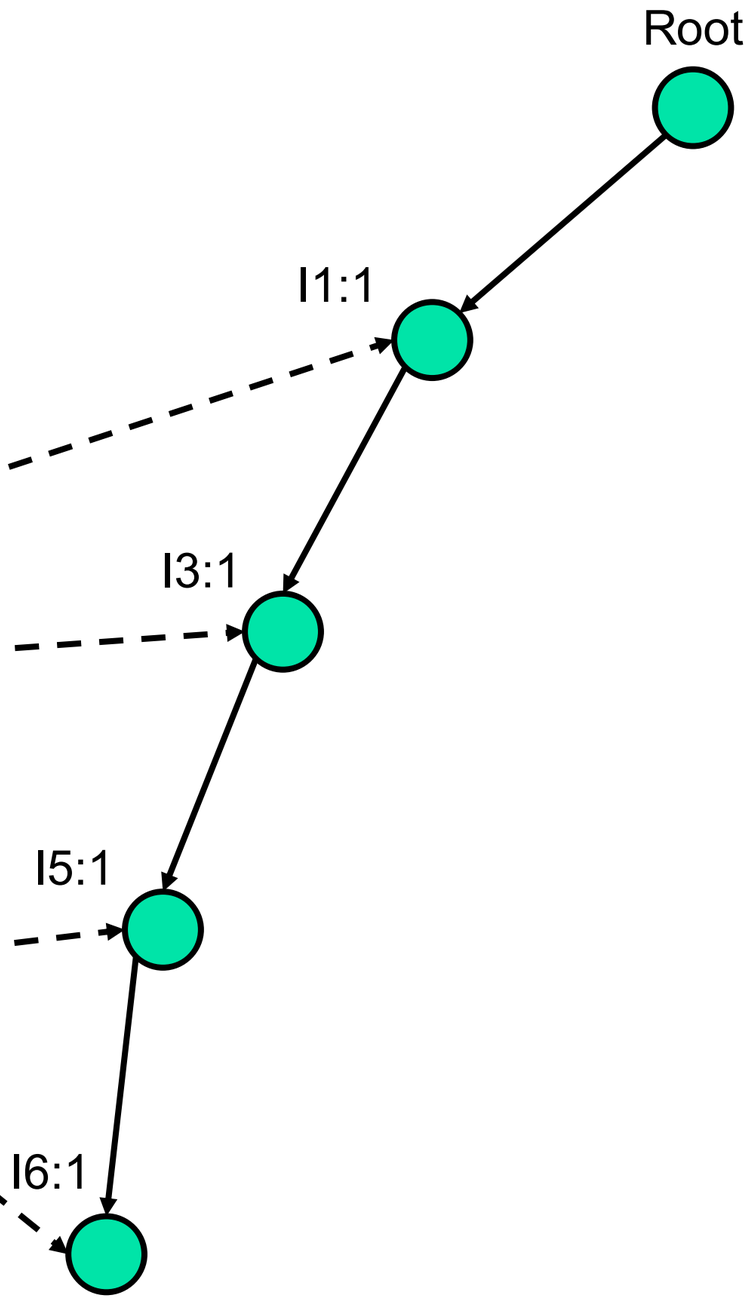
| 项集 | 支持度计数 | 节点链 |
|---|---|---|
| **{I1}** | **4** | |
| **{I3}** | **4** | |
| **{I4}** | **3** | |
| **{I5}** | **3** | |
| **{I6}** | **3** | |

| 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** | |

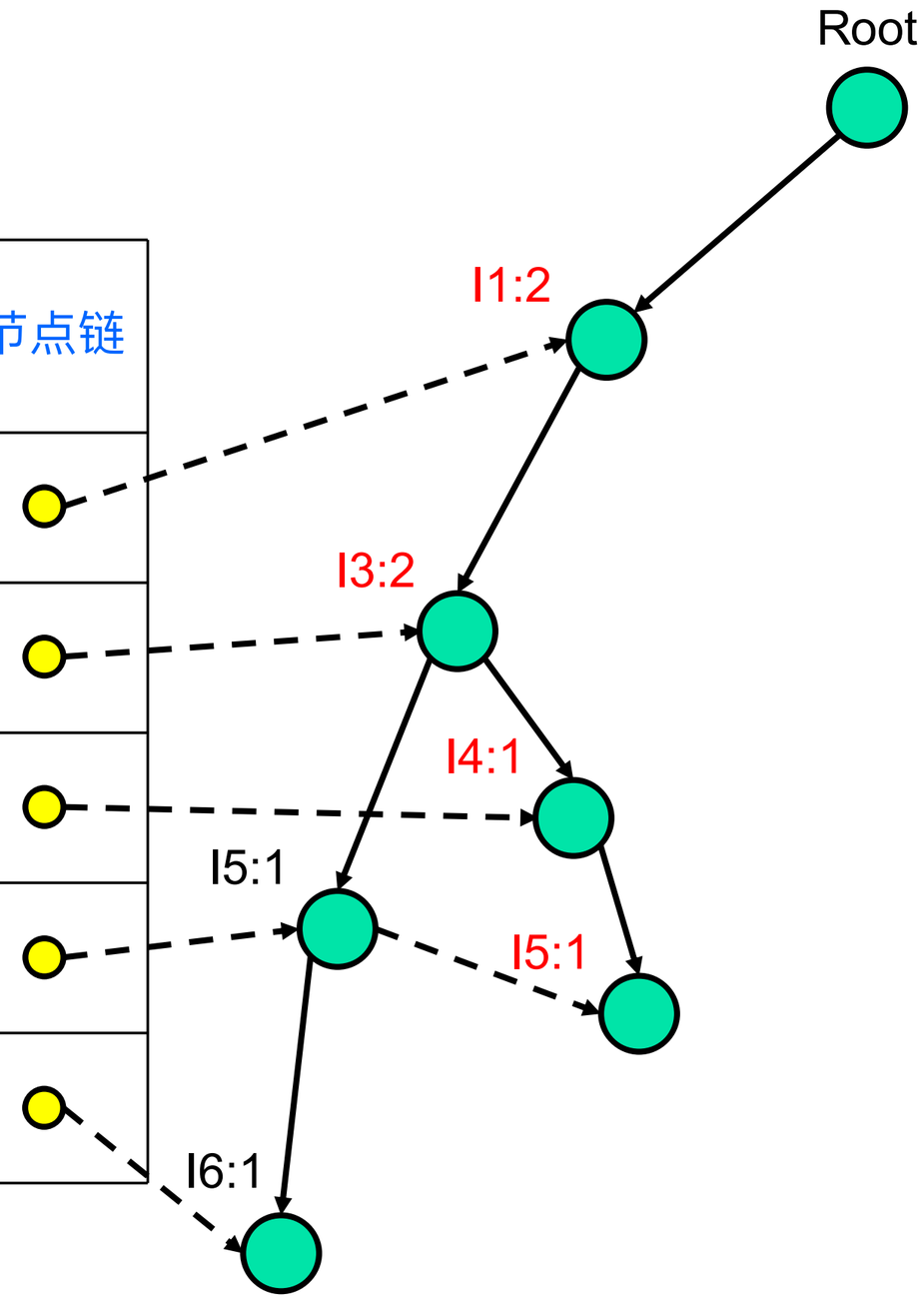

| TID（事务ID） | 商品ID列表 |
| --- | --- |
| **T001** | **{I1, I3, I5, I6}** |
| T002 | {I1, I3, I4, I5} |
| T003 | {I1, I4} |
| T004 | {I1, I3, I5, I6} |
| T005 | {I3, I4, I6} |

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



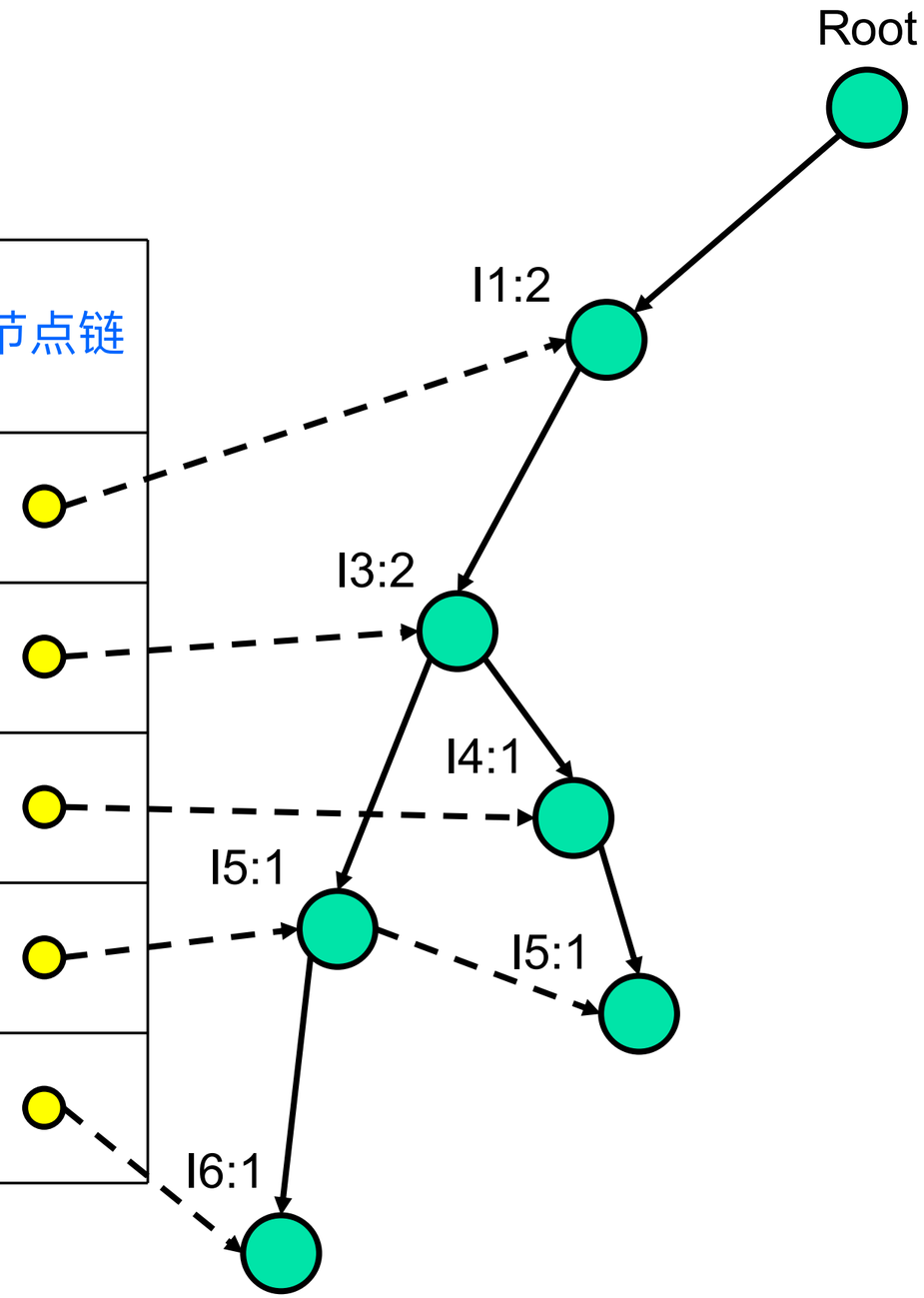

| TID（事务ID） | 商品ID列表 |
| --- | --- |
| T001 | {I1, I3, I5, I6} |
| **T002** | **{I1, I3, I4, I5}** |
| T003 | {I1, I4} |
| T004 | {I1, I3, I5, I6} |
| T005 | {I3, I4, I6} |

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



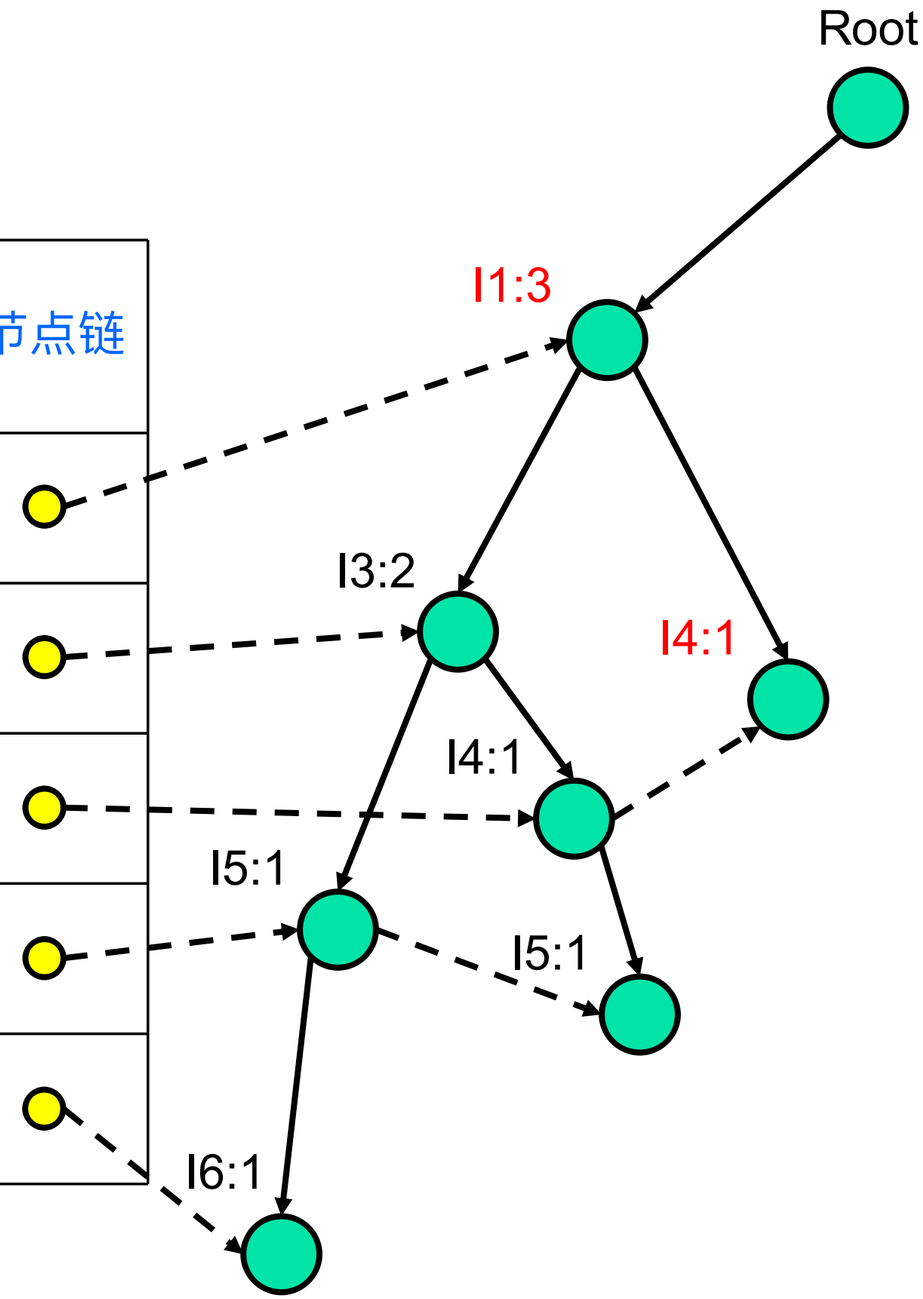

| TID（事务ID） | 商品ID列表 |
| --- | --- |
| T001 | {I1, I3, I5, I6} |
| **T002** | **{I1, I3, I4, I5}** |
| T003 | {I1, I4} |
| T004 | {I1, I3, I5, I6} |
| T005 | {I3, I4, I6} |

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



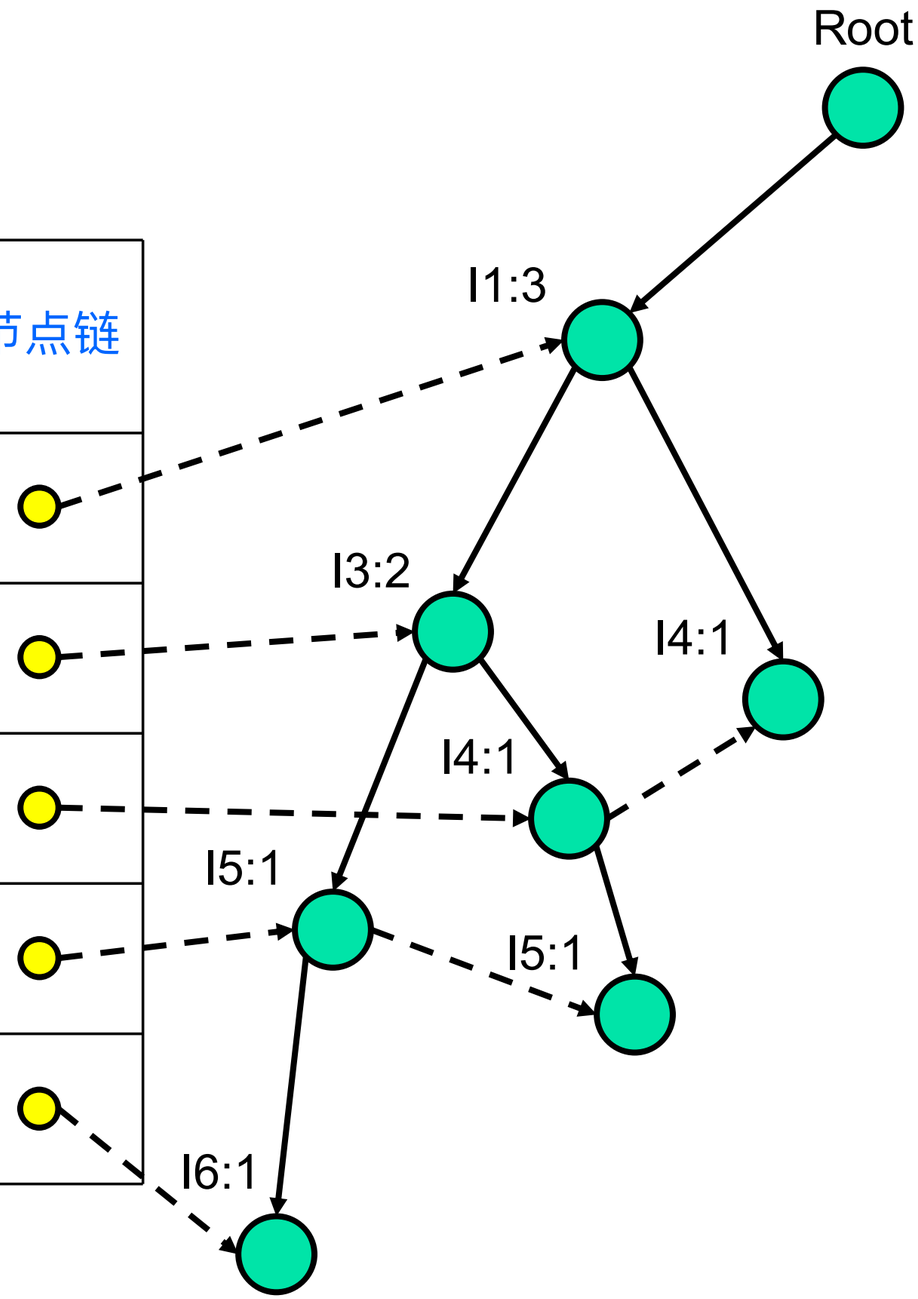

| TID（事务ID） | 商品ID列表 |
| --- | --- |
| T001 | {I1, I3, I5, I6} |
| T002 | {I1, I3, I4, I5} |
| **T003** | **{I1, I4}** |
| T004 | {I1, I3, I5, I6} |
| T005 | {I3, I4, I6} |

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



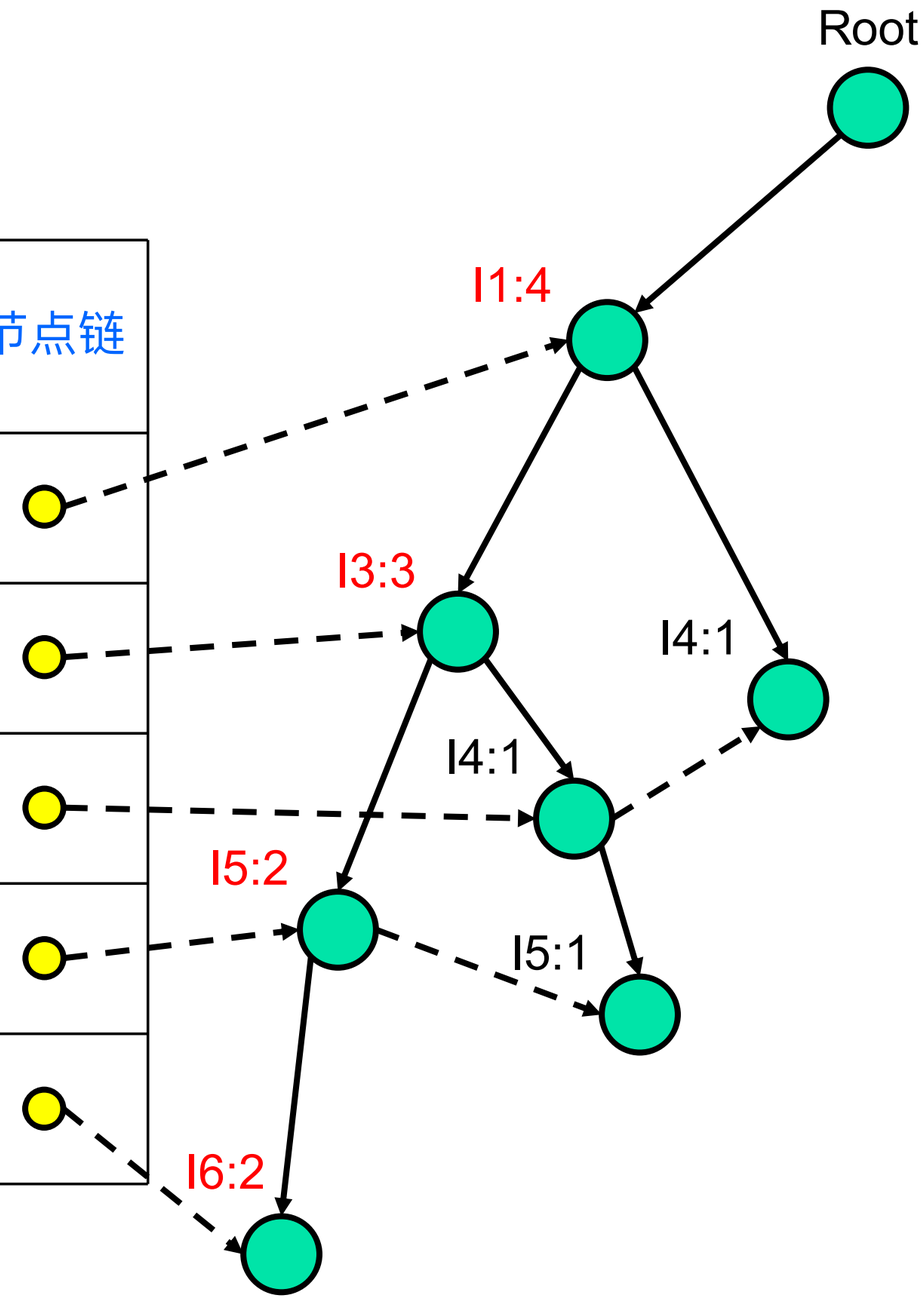

| TID（事务ID） | 商品ID列表 |
|---|---|
| T001 | {I1, I3, I5, I6} |
| T002 | {I1, I3, I4, I5} |
| **T003** | **{I1, I4}** |
| T004 | {I1, I3, I5, I6} |
| T005 | {I3, I4, I6} |

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



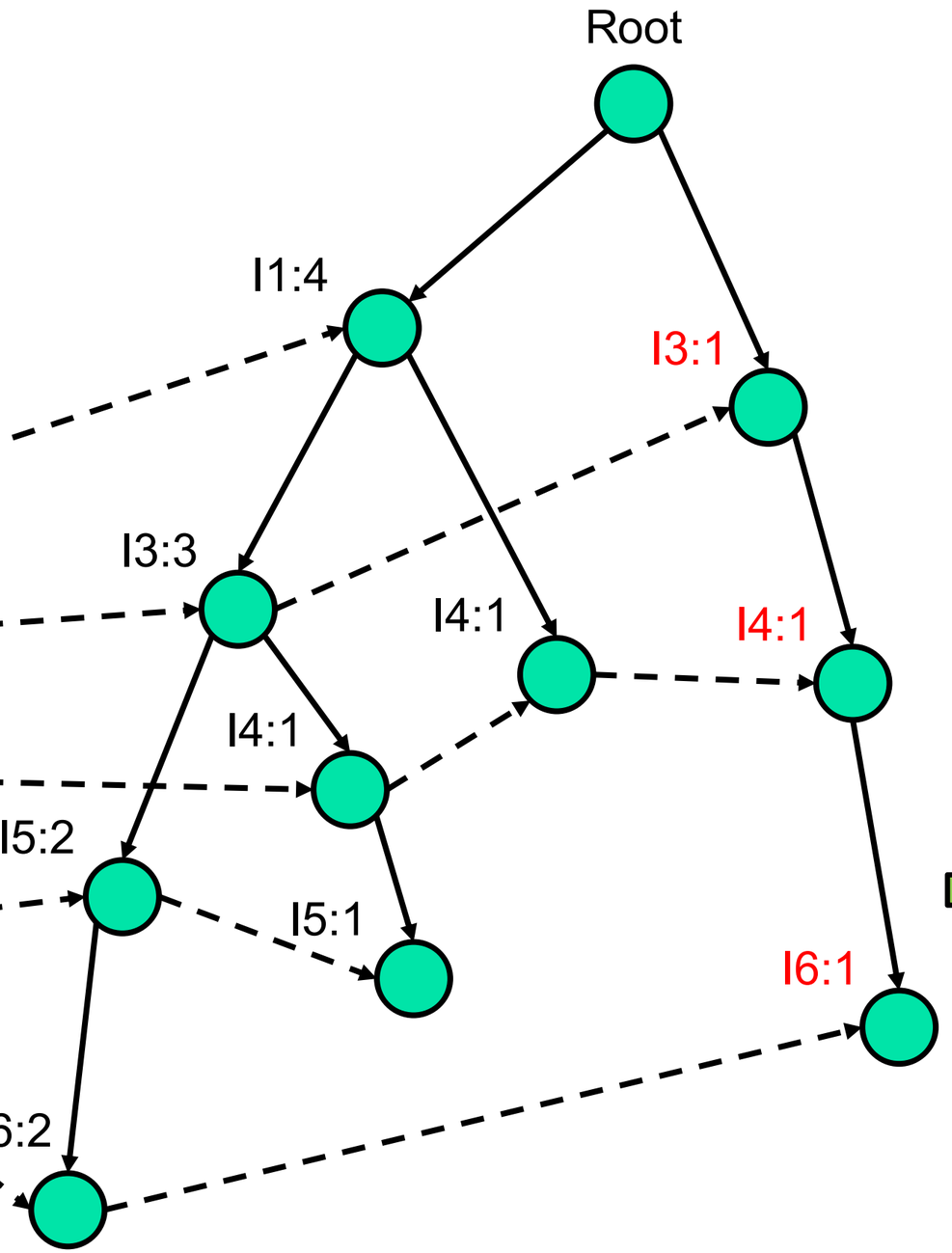

| TID（事务ID） | 商品ID列表 |
|---|---|
| T001 | {I1, I3, I5, I6} |
| T002 | {I1, I3, I4, I5} |
| T003 | {I1, I4} |
| **T004** | **{I1, I3, I5, I6}** |
| T005 | {I3, I4, I6} |

| 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 |
| --- | --- |
| 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** | |



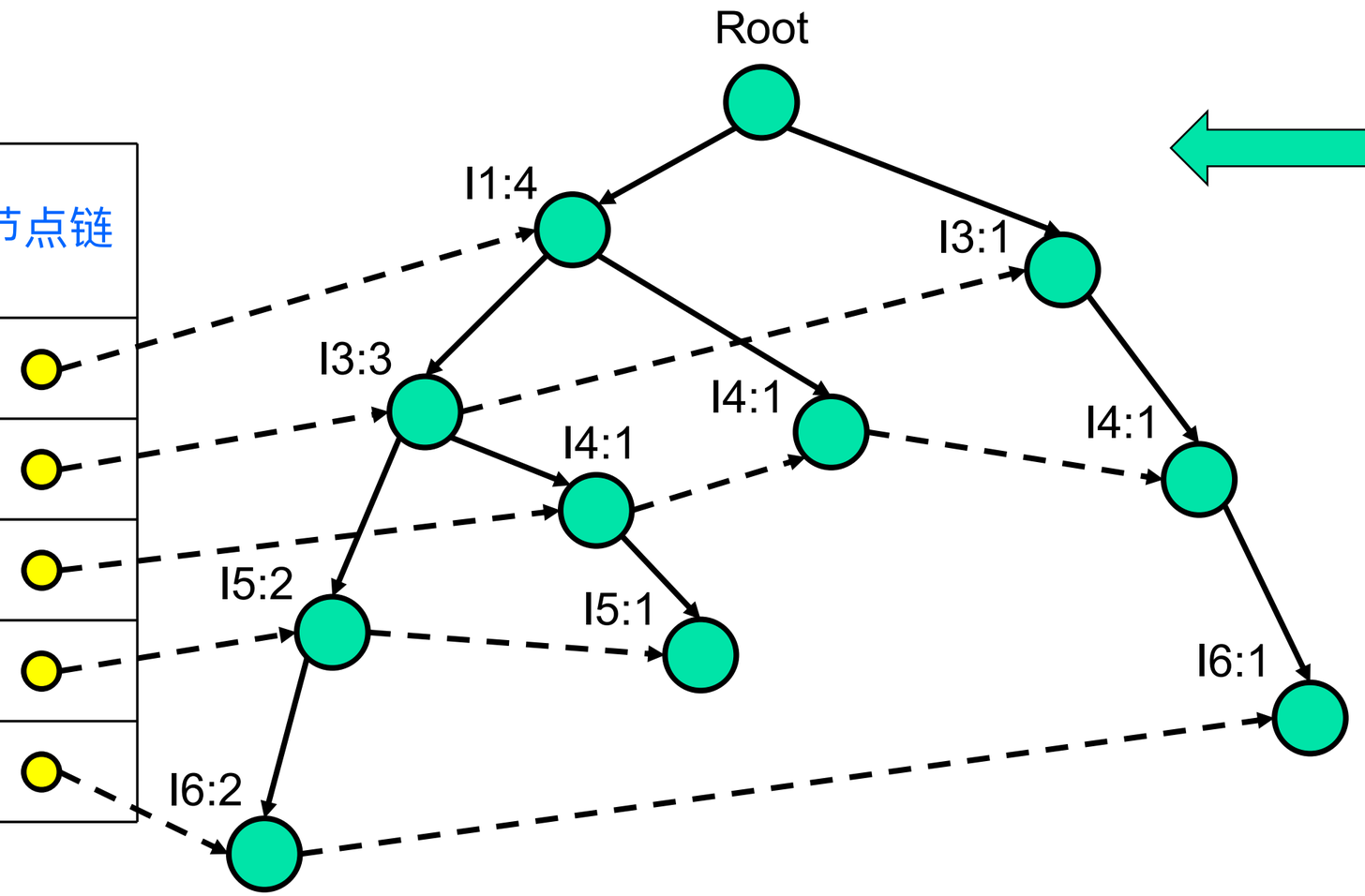

| TID（事务ID） | 商品ID列表 |
| --- | --- |
| T001 | {I1, I3, I5, I6} |
| T002 | {I1, I3, I4, I5} |
| T003 | {I1, I4} |
| T004 | {I1, I3, I5, I6} |
| **T005** | **{I3, I4, I6}** |

# 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

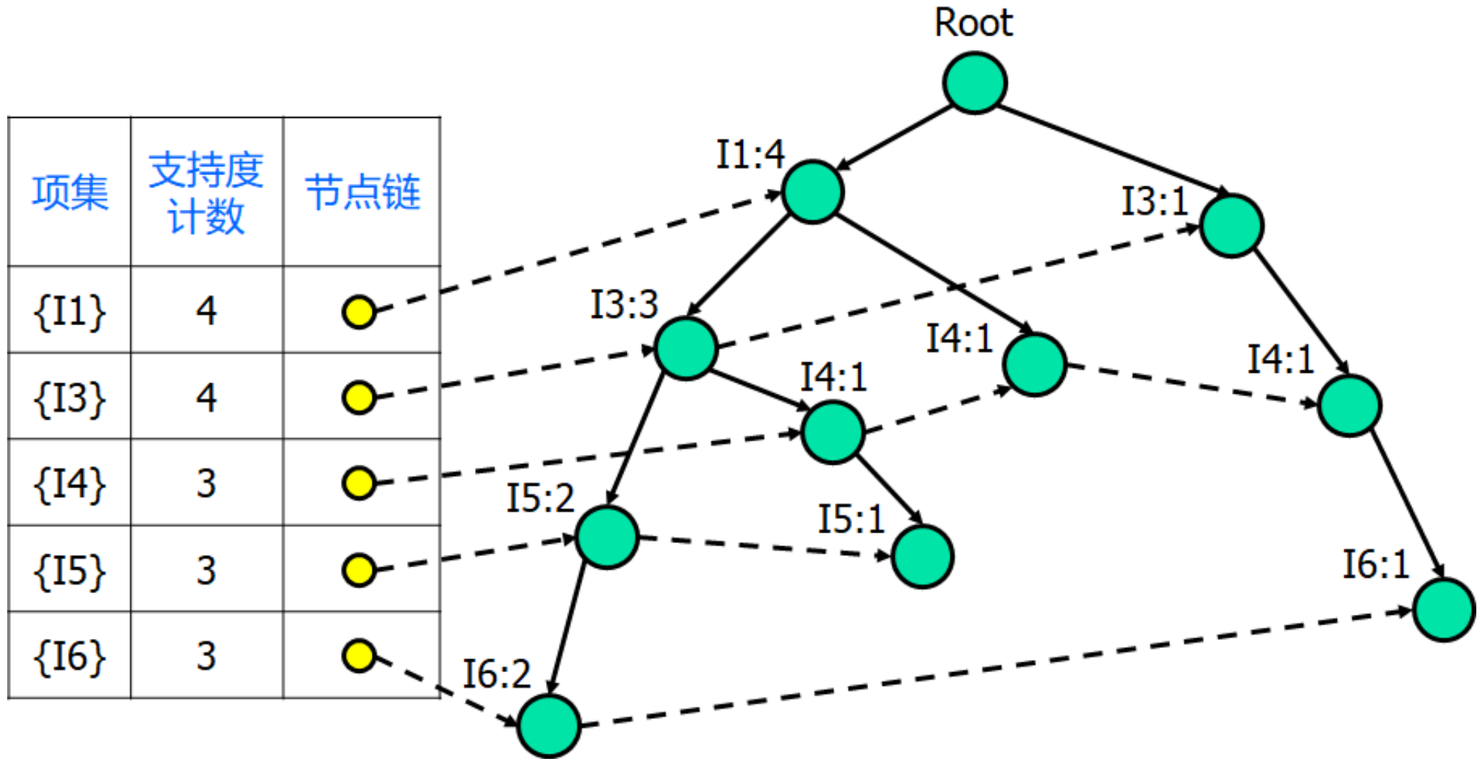
  --挖掘以I6结尾的频繁项集　　--挖掘以I5结尾的频繁项集

  --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**

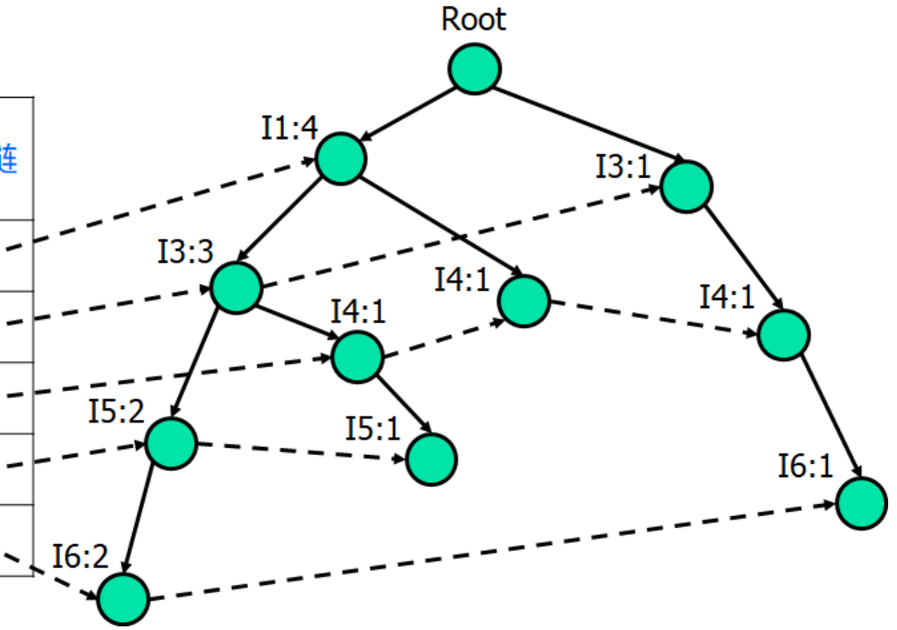
| 项集 | 支持度计数 | 节点链 |
|---|---|---|
| {I1} | 4 | |
| {I3} | 4 | |
| {I4} | 3 | |
| {I5} | 3 | |
| {I6} | 3 | |



# 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**($Tree_\beta, \beta$)

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




- **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*, $\alpha$)

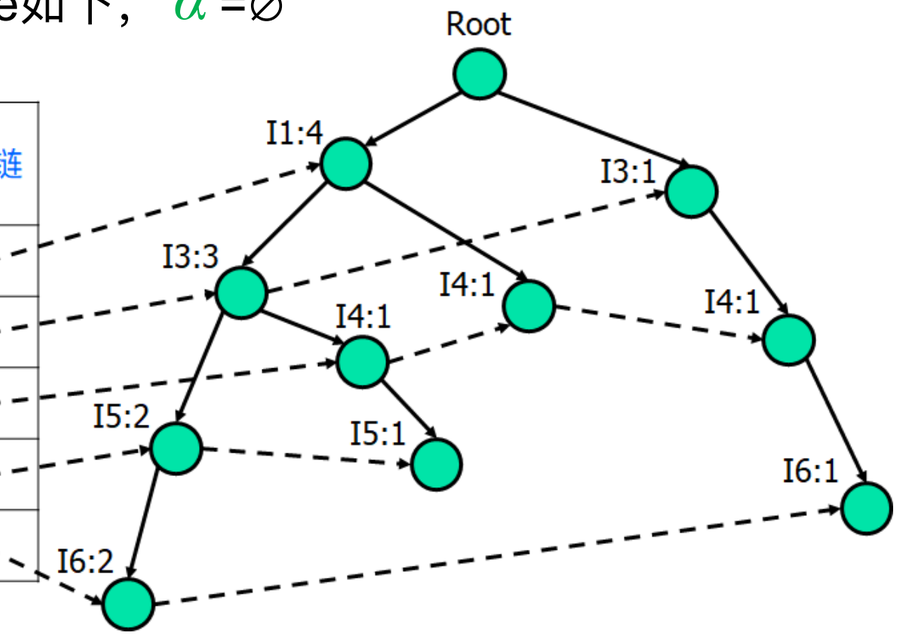
- **if** *Tree* 只包含单个分支 $P$ **then**
  - **for each** $\theta$ (节点组合) **in** $C_\theta$ ($P$中节点的全部组合)
    - ✓ 生成新的频繁项集: $\beta = \theta \cup \alpha$
- **else for each entry** $e_i$ **in** head table（头部表）
  - 生成新的频繁项集: $\beta = e_i \cup \alpha$
  - 构建 $\beta$ 的条件数据库： $D_\beta$
  - 基于$D_\beta$构建 $\beta$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

最初调用，参数Tree如下， $\alpha$ =∅

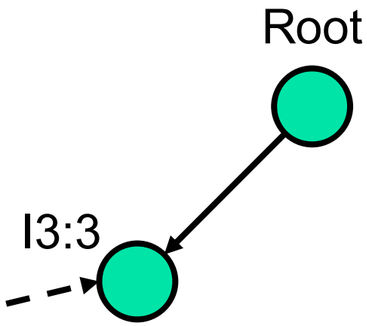
| 项集 | 支持度计数 | 节点链 |
|---|---|---|
| {I1} | 4 | |
| {I3} | 4 | |
| {I4} | 3 | |
| {I5} | 3 | |
| {I6} | 3 | |




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

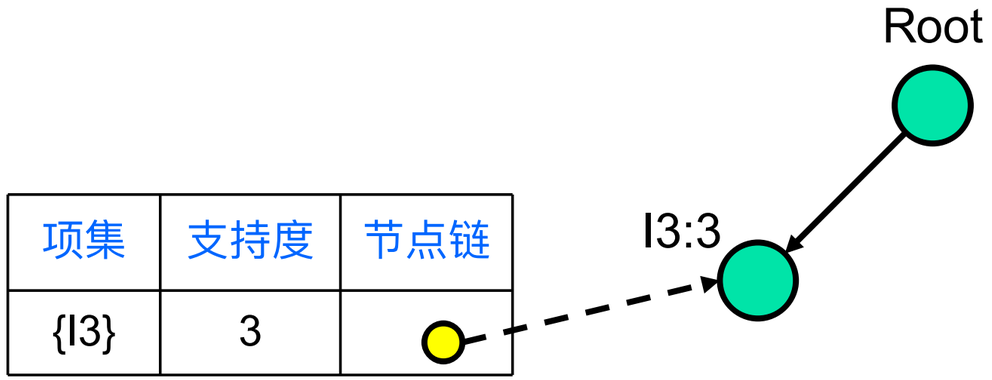
$Tree_\beta$:

| 项集 | 支持度 | 节点链 |
|---|---|---|
| {I3} | 3 | |



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

递归调用，参数Tree如下， $\alpha$ ={I6}




| 项集 | 支持度 | 节点链 |
|---|---|---|
| {I3} | 3 | |

- **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} ∪ {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*, $\alpha$)

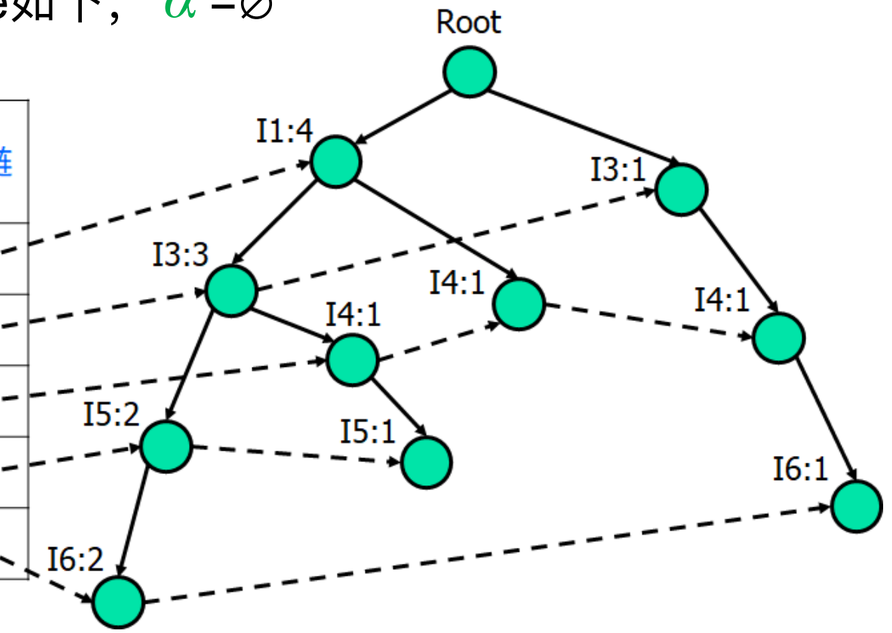
- **if** *Tree* 只包含单个分支 $P$ **then**
  - **for each** $\theta$ (节点组合) **in** $C_\theta$ ($P$中节点的全部组合)
    - ✓ 生成新的频繁项集: $\beta = \theta \cup \alpha$
- **else for each entry** $e_i$ **in** head table（头部表）
  - 生成新的频繁项集: $\beta = e_i \cup \alpha$
  - 构建 $\beta$ 的条件数据库： $D_\beta$
  - 基于$D_\beta$构建 $\beta$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

最初调用，参数Tree如下， $\alpha$ =∅

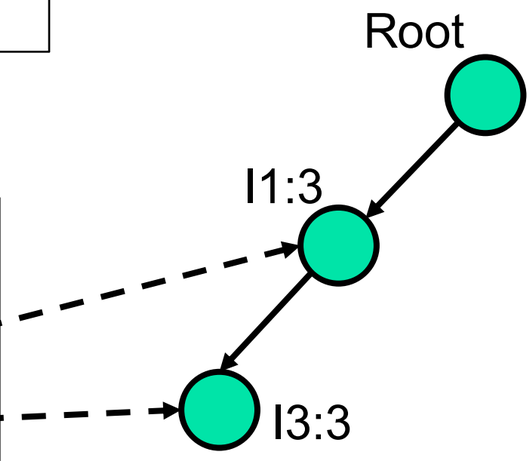
| 项集 | 支持度计数 | 节点链 |
|---|---|---|
| {I1} | 4 | |
| {I3} | 4 | |
| {I4} | 3 | |
| {I5} | 3 | |
| {I6} | 3 | |




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

$Tree_\beta$:

| 项集 | 支持度 | 节点链 |
|---|---|---|
| {I1} | 3 | |
| {I3} | 3 | |

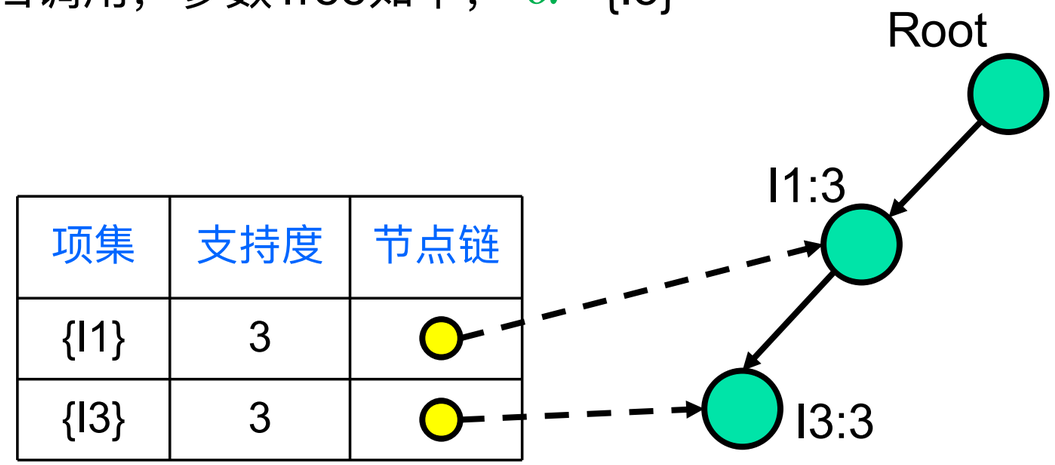


# Procedure FP-growth(*Tree*, $\alpha$)

- **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**($Tree_\beta, \beta$)

递归调用，参数Tree如下， $\alpha$ ={I5}

| 项集 | 支持度 | 节点链 |
|---|---|---|
| {I1} | 3 | |
| {I3} | 3 | |




**生成新的频繁项集：**

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

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

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

- **解决第2个子问题**：挖掘以I5结尾的频繁项集

- 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,<br>{I3, I4}:1 | {I3, I6} |
| {I5} | {I1, I3}:2,<br>{I1, I3, I4}:1 | {I1, I5}, {I3, I5}, {I1, I3, I5} |
| {I4} | {I1}:1, {I3}:1,<br>{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} → {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
  - 引入相关性度量，如提升度度量(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

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

- 提升度度量的评判准则：

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

| | 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}

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

提升度小于1，拒绝该规则!

- 其它相关性度量指标：
  - $\chi^2$ 度量
  - 全置信度（all-confidence）
  - 最大置信度(max-confidence)
  - 余弦度量(cosine)
  - Kulczynski（Kulc）度量
  - …

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

- 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：$\mathrm{Kulc}(X, Y) = \frac{1}{2}\left(P(X|Y) + P(Y|X)\right)$

  - 四种度量取值范围为[0, 1]，评判标准：$\begin{cases} = 0.5，X和Y中性关联 \\ > 0.5，X和Y呈正相关 \\ < 0.5，X和Y呈负相关 \end{cases}$

# 零记录（null-transactions）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 |

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

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

# 具有零记录不变性（ Null-invariant ）的相关性度量指标

- 全置信度: $all\_conf(W) = \frac{\sup(W)}{\max\_item\_sup(W)}$ $W = XY$
- 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: $\text{Kulc}(X,Y) = \frac{1}{2}\left(P(X|Y) + P(Y|X)\right)$

| Set | mc | $\overline{m}c$ | $m\overline{c}$ | $\overline{mc}$ | $\chi^2$ | 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 |

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

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

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

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

# 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 |

{牛奶, 面包} → {啤酒}

| 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 |

**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

| 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1-项集 | 支持度计数 |
|---|---|
| A1 | 3 |
| S1 | 3 |
| S2 | 3 |
| J2 | 3 |
| 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3-项集 | 支持度计数 |
|---|---|
| A1, S1, I1 | 3 |
| S2, J2, I2 | 3 |

- 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}→{I1} | 1.0 |
| {S1}→{I1} | 1.0 |
| {A1, S1}→{I1} | 1.0 |
| {S2}→{I2} | 1.0 |
| {J2}→{I2} | 1.0 |
| {S2, J2}→{I2} | 1.0 |

将单维关联规则转化为多维关联规则

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

$$\downarrow$$

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

- 挖掘多维量化关联规则的主要步骤总结：
  - 量化属性离散化
  - 合并属性列，转化为单维关联规则挖掘任务
  - 根据给定的*min-sup*，应用Apriori等算法生成频繁项集
  - 根据给定的*min-conf*，基于频繁项集生成（单维）强关联规则
  - 将挖掘出的单维关联规则还原为多维关联规则
- 根据应用需求，筛选出需要的关联规则
  - 例如，前件只有{年龄、月收入、职业}三种属性，后件只包含商品项的集合
  - 如：Age(X, “20-25”) ∧ Salary(X, “5K-10K”) → buys(X, “I1”) ∧ buys(X, “I6”)