

目录

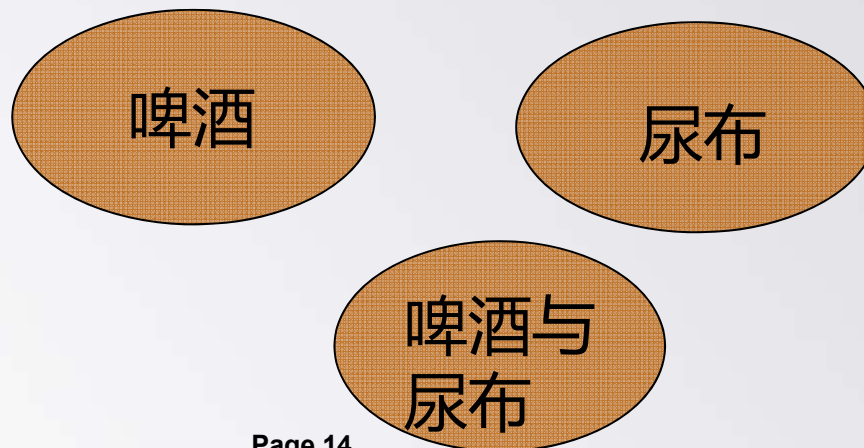
■ 基本概念

■ 频繁项集挖掘方法

- Apriori算法：通过限制候选产生发现频繁项集
- 由频繁项集产生关联规则
- 提高Apriori算法的效率
- 挖掘频繁项集的模式增长方法
- 使用垂直数据格式挖掘频繁项集
- 挖掘闭模式和极大模式

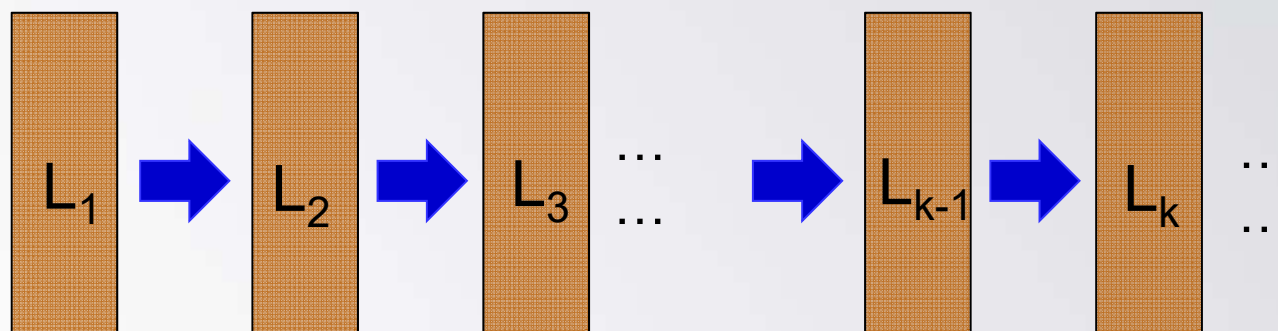
Apriori算法

- Apriori算法一种挖掘关联规则频繁项集的算法
- 利用了Apriori性质：频繁项集的所有非空子集也必须是频繁的
 - $A \cup B$ 不可能比A更频繁的出现
 - 反单调性：一个集合如果不能通过测试，则该集合的所有超集也不能通过相同的测试
 - 通过减少搜索空间，来提高频繁项集产生的效率



Apriori算法

- 通过逐层搜索的迭代方法，将 $k-1$ 项集用于探索 k 项集，来穷尽所有频繁项集
 - 先找到频繁1项集集合 L_1 ，然后用 L_1 找到频繁2项集 L_2 ，接着用 L_2 找 L_3直到找不到频繁项集
 - 找每个 L_k 需要扫描一次数据库



Apriori算法步骤

- 由连接和剪枝两个步骤组成
- 连接：为了找 L_k ，通过 L_{k-1} 与自己连接产生候选k项集的集合，该候选k项集记为 C_k

- L_{k-1} 中的两个元素 L_1 和 L_2 可以执行连接操作的条件是

$$(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \dots \dots (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$$

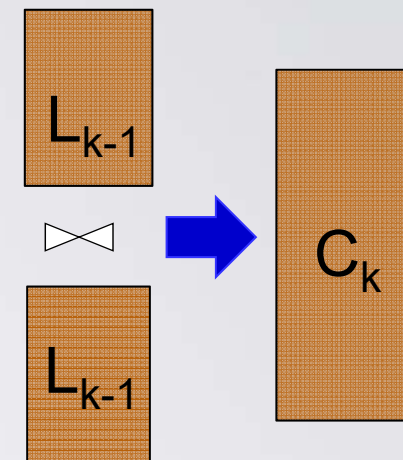
- 例子

$$\{A, C\} \bowtie \{B, C\} = \{A, B, C\}$$

$$\{A, B, C\} \bowtie \{A, B, E\} = \{A, B, C, E\}$$

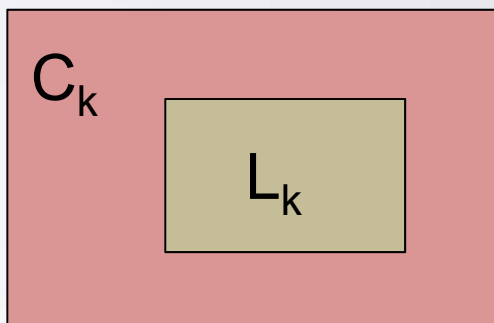
$$\{A, C\} \bowtie \{B, E\} =$$

$$\{A\} \bowtie \{B\} = \{A, B\}$$



Apriori算法步骤

- 候选集 C_k 是频繁 k 项集 L_k 的超集
 - 所有频繁 k 项集都在 C_k 中, 但 C_k 可能还会包含其他元素
 - 通过扫描数据库, 计算 C_k 中每个项集的支持度来得到 L_k
 - 为了减少计算量, 可以使用Apriori性质, 即如果一个 k 项集的 $(k-1)$ 子集不在 L_{k-1} 中, 则其不可能是频繁的, 可以直接从 C_k 删除



Apriori算法——示例

最小支持数为 2

Database

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

C_1

1st scan

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

L_1

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

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_2

2nd scan

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

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

C_3

Itemset
{B, C, E}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

使用Apriori性质由 L_2 产生 C_3

- 连接：
 - $C_3 = L_2 \bowtie L_2 = \{\{A, C\}, \{B, C\}, \{B, E\}, \{C, E\}\} \bowtie \{\{A, C\}, \{B, C\}, \{B, E\}, \{C, E\}\} = \{\{A, B, C\}, \{A, C, E\}, \{B, C, E\}\}$
- 使用Apriori性质剪枝：频繁项集的所有子集必须是频繁的，对候选项 C_3 ，我们可以删除其子集为非频繁的选项：
 - $\{A, B, C\}$ 的2项子集是 $\{A, B\}, \{A, C\}, \{B, C\}$ ，其中 $\{A, B\}$ 不是 L_2 的元素，所以删除这个选项；
 - $\{A, C, E\}$ 的2项子集是 $\{A, C\}, \{A, E\}, \{C, E\}$ ，其中 $\{A, E\}$ 不是 L_2 的元素，所以删除这个选项；
 - $\{B, C, E\}$ 的2项子集是 $\{B, C\}, \{B, E\}, \{C, E\}$ ，它的所有2项子集都是 L_2 的元素，因此保留这个选项。
- 这样，剪枝后得到 $C_3 = \{\{B, C, E\}\}$

L_2

Itemset	sup
$\{A, C\}$	2
$\{B, C\}$	2
$\{B, E\}$	3
$\{C, E\}$	2

由频繁项集产生关联规则

- 同时满足最小支持度和最小置信度的才是强关联规则，从频繁项集产生的规则都满足支持度要求，而其置信度则可由一下公式计算：

$$\text{confidence}(A \Rightarrow B) = P(B \mid A) = \frac{\text{support_count}(A \cup B)}{\text{support_count}(A)}$$

- 每个关联规则可由如下过程产生：
 - 对于每个频繁项集 l ，产生 l 的所有非空子集；
 - 对于每个非空子集 s ，如果 $\frac{\text{support_count}(l)}{\text{support_count}(s)} \geq \text{min_conf}$


则输出规则 $s \Rightarrow (l - s)$

提高Apriori算法的效率

- 低效率的Apriori

- 可能需要产生大量的候选项集.

10^4 个频繁1项集  多达 10^7 个候选2项集

发掘一个频繁模式, $\{a_1, \dots, a_{100}\}$  10^{30} 个候选项集

- 可能需要重复扫描整个数据库，通过模式匹配检验一个很大的候选集合。

提高基于Apriori挖掘效率的算法

- 基于散列的技术
- 事物归约技术
- 划分技术
- 抽样技术
- 动态项集计数技术
- 频繁模式增长

- [1] J. S. Park, M. S. Chen, and P. S. Yu. An Effective Hash-Based Algorithm for Mining Association Rules. In Proc. 1995 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'95, pp.175-186,1995.
- [2] H. Toivonen. Sampling Large Databases for Association Rules. In Proc. 1996 Int. Conf. Very Large Data Bases (VLDB'96), pp.134-145, 1996.

挖掘频繁项集的模式增长方法

- **频繁增长模式适应了分治策略：**
 - 将代表频繁项集的数据库压缩到一颗频繁模式树（FP-tree），该树仍保留项集的关联信息。
 - 把这种压缩后的数据库分解成一组条件数据库，每个数据库关联一个频繁项或“模式段”并且分别挖掘每个条件数据库。

示例

TID	List of item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

第一步

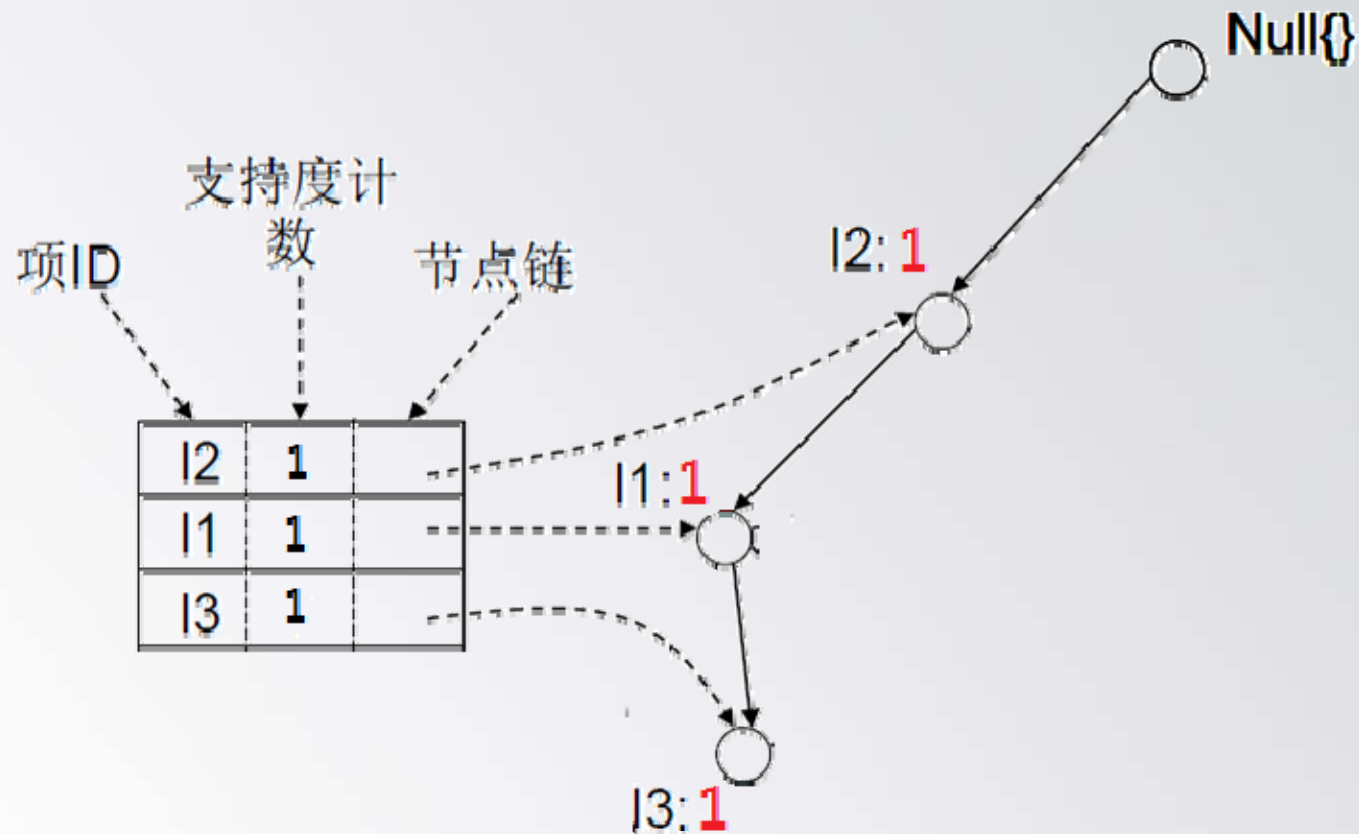
- 规定最小支持度计数，例子中最小支持度计数是2.
- 数据库的第一次扫描和Apriori算法一样，它导出频繁项的集合并得到它们的支持度计数。
- 频繁项的集合按支持度计数的递减排序。结果集或标记为L

第一步的结果

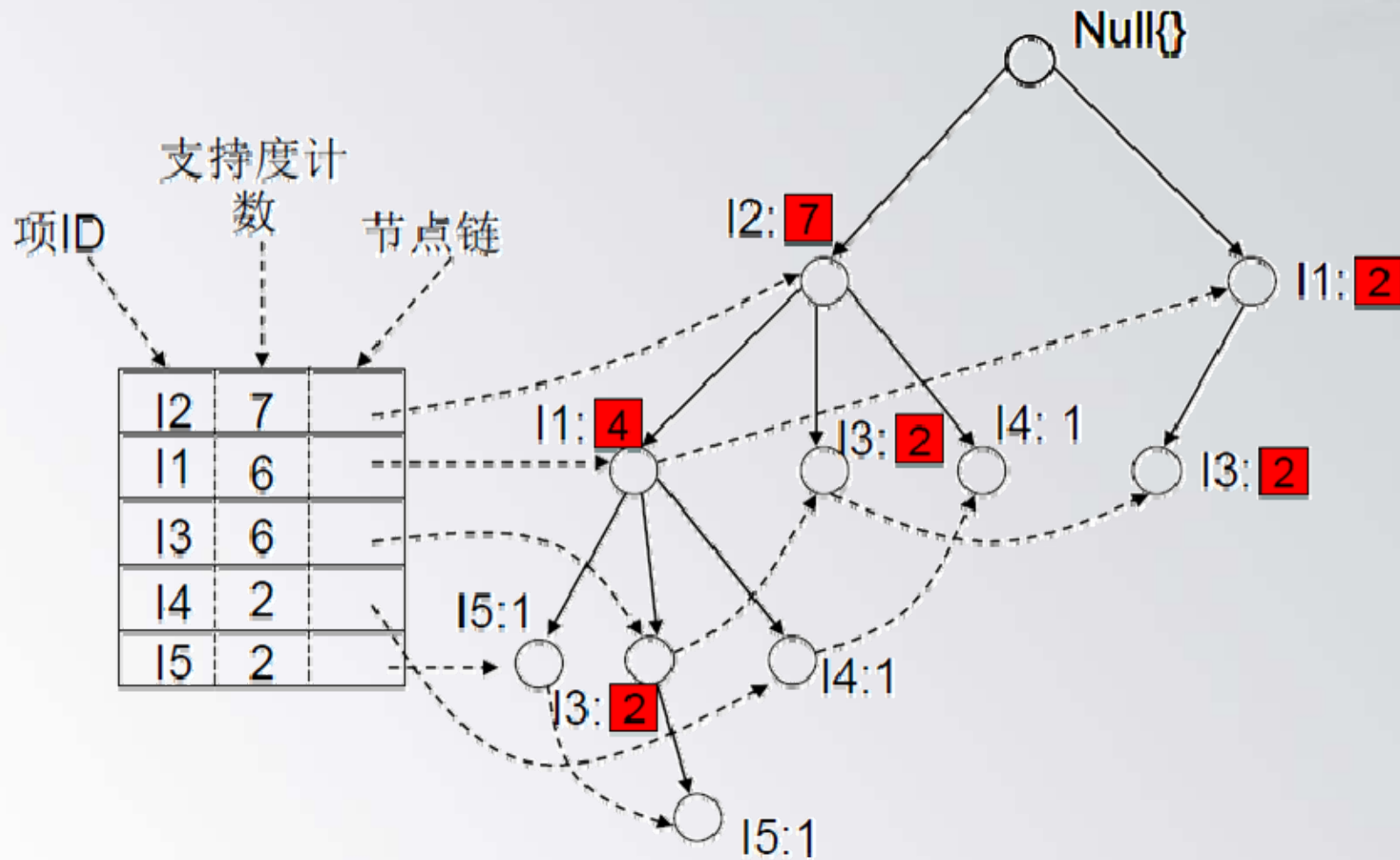
项	支持度计数 (frequencies)
I2	7
I1	6
I3	6
I4	2
I5	2

第二步：频繁模式树

T900 {I2, I1, I3}



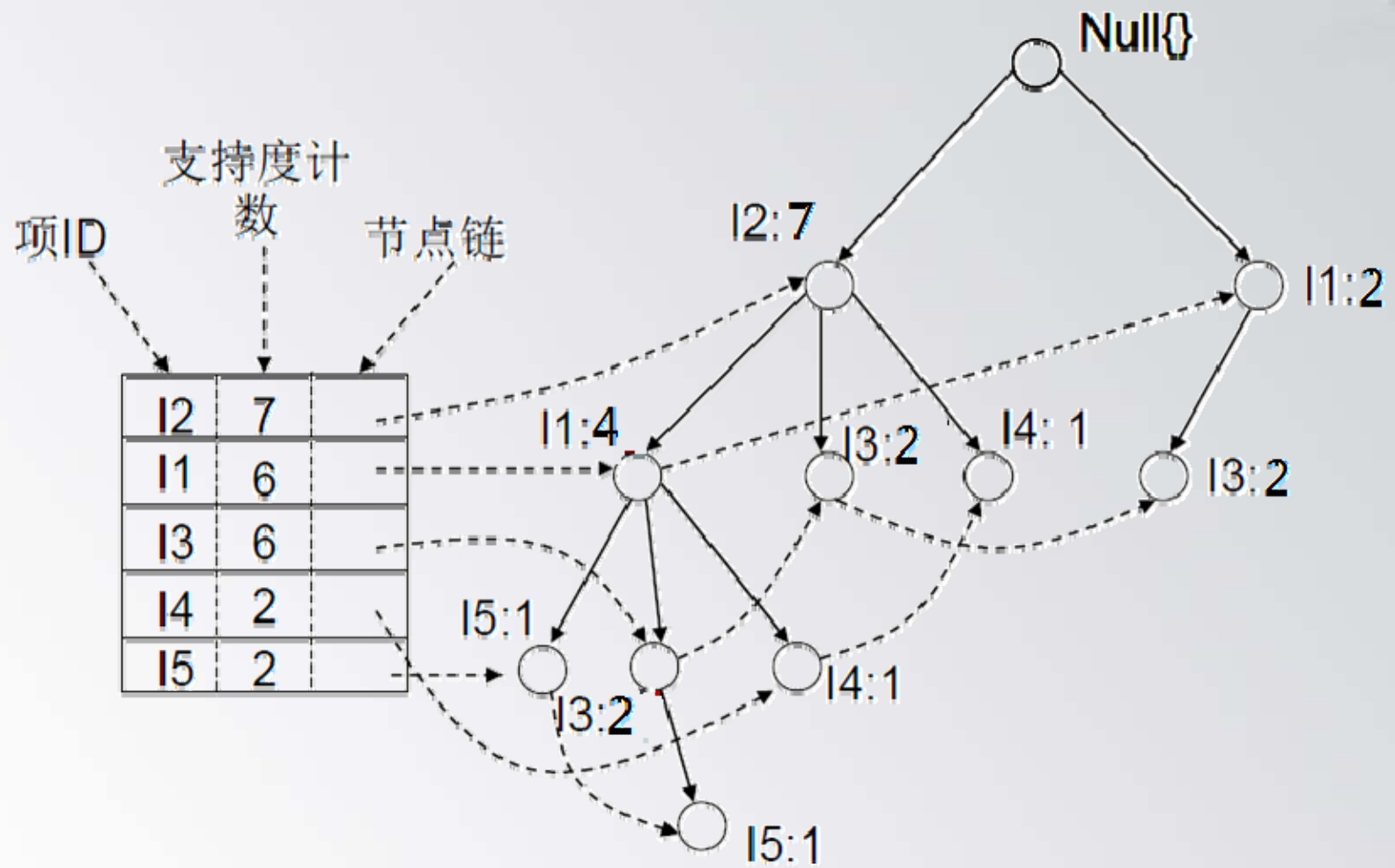
第二步: 频繁模式树



第三步：频繁模式树挖掘

- 由长度为1的频繁模式（初始后缀模式）开始，构造它的条件模式基。
- 构造它的（条件）FP树，并递归地在该树上进行挖掘。
- 模式增长通过后缀模式与条件FP树产生的频繁模式连接实现。

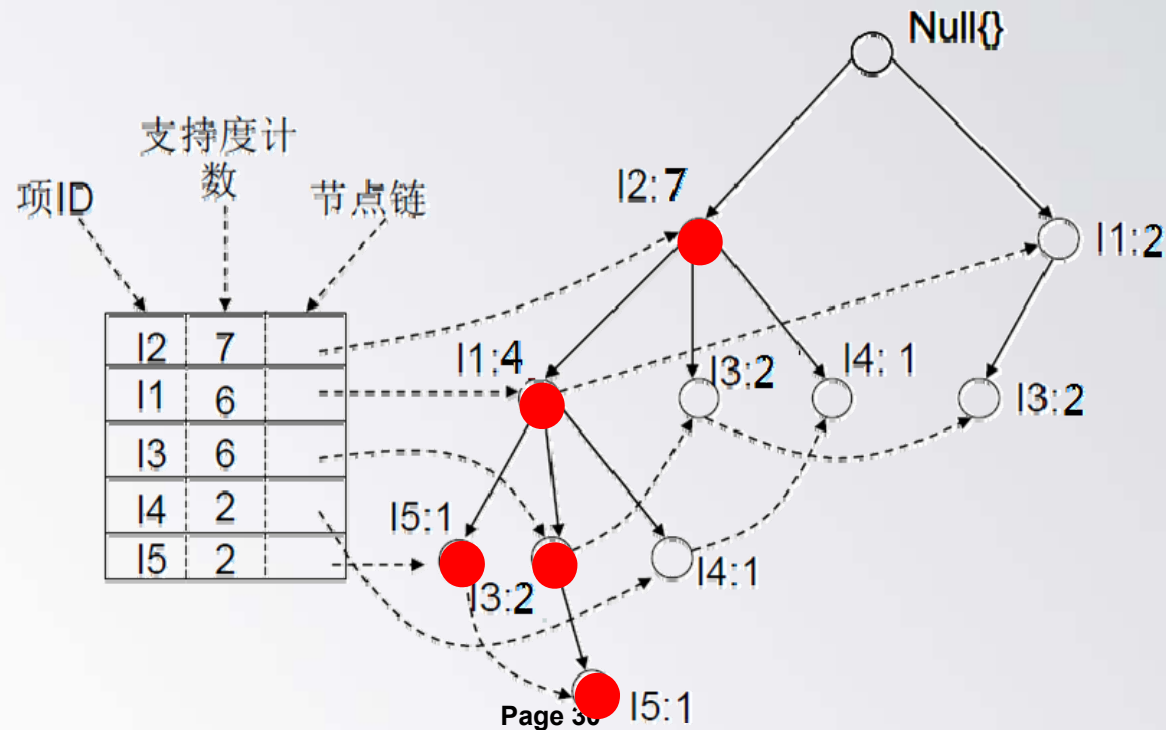
频繁模式树



对项I5挖掘

产生的频繁模式

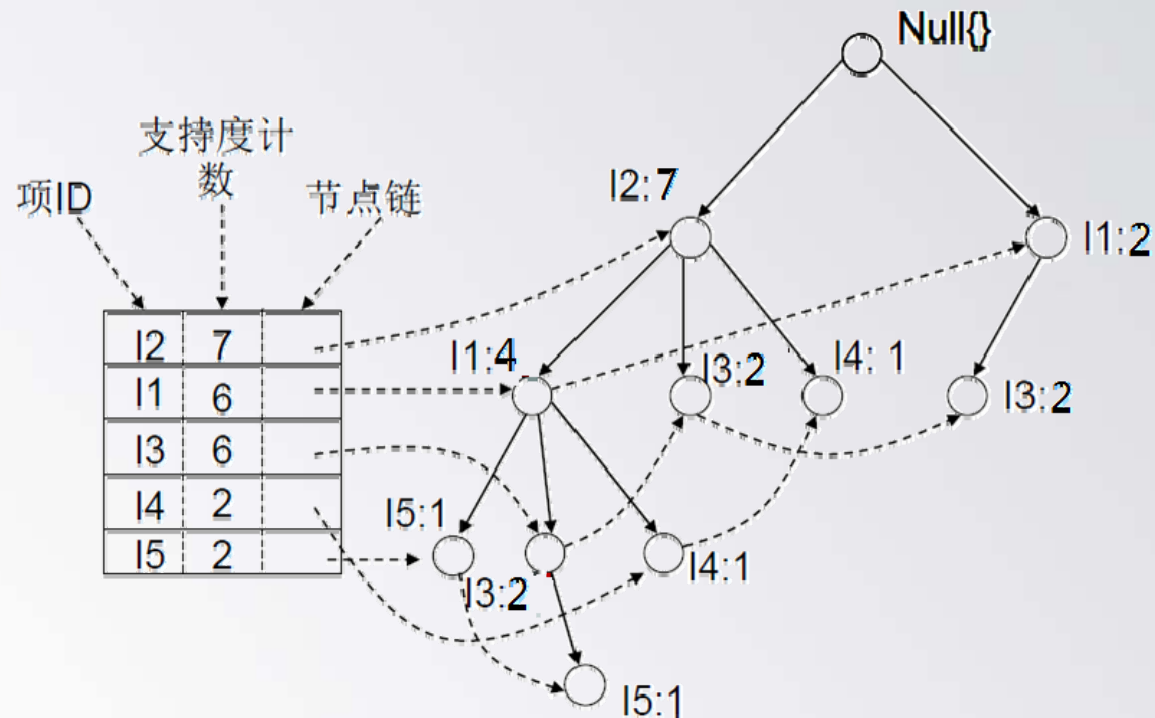
项	条件模式基	条件FP树	产生的频繁模式
I5	{ {I2,I1:1},{I2,I1,I3:1} }	$\langle I2:2, I1:2 \rangle$	{I2,I5:2},{I1,I5:2},{I2,I1,I5:2}



对项I4挖掘

产生的频繁模式

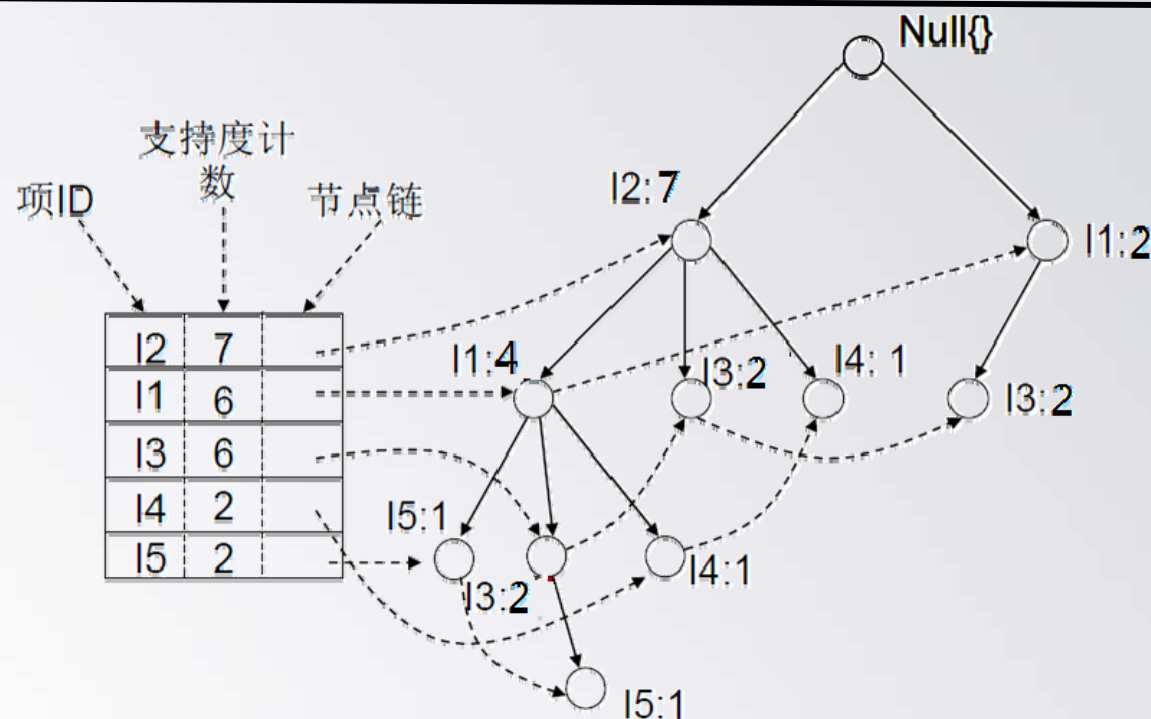
项	条件模式基	条件FP树	产生的频繁模式
I5	{ {I2,I1:1},{I2,I1,I3:1} }	<I2:2,I1:2>	{I2,I5:2},{I1,I5:2},{I2,I1,I5:2}
I4	{ {I2,I1:1},{I2:1} }	<I2:2>	{I2,I4:2}



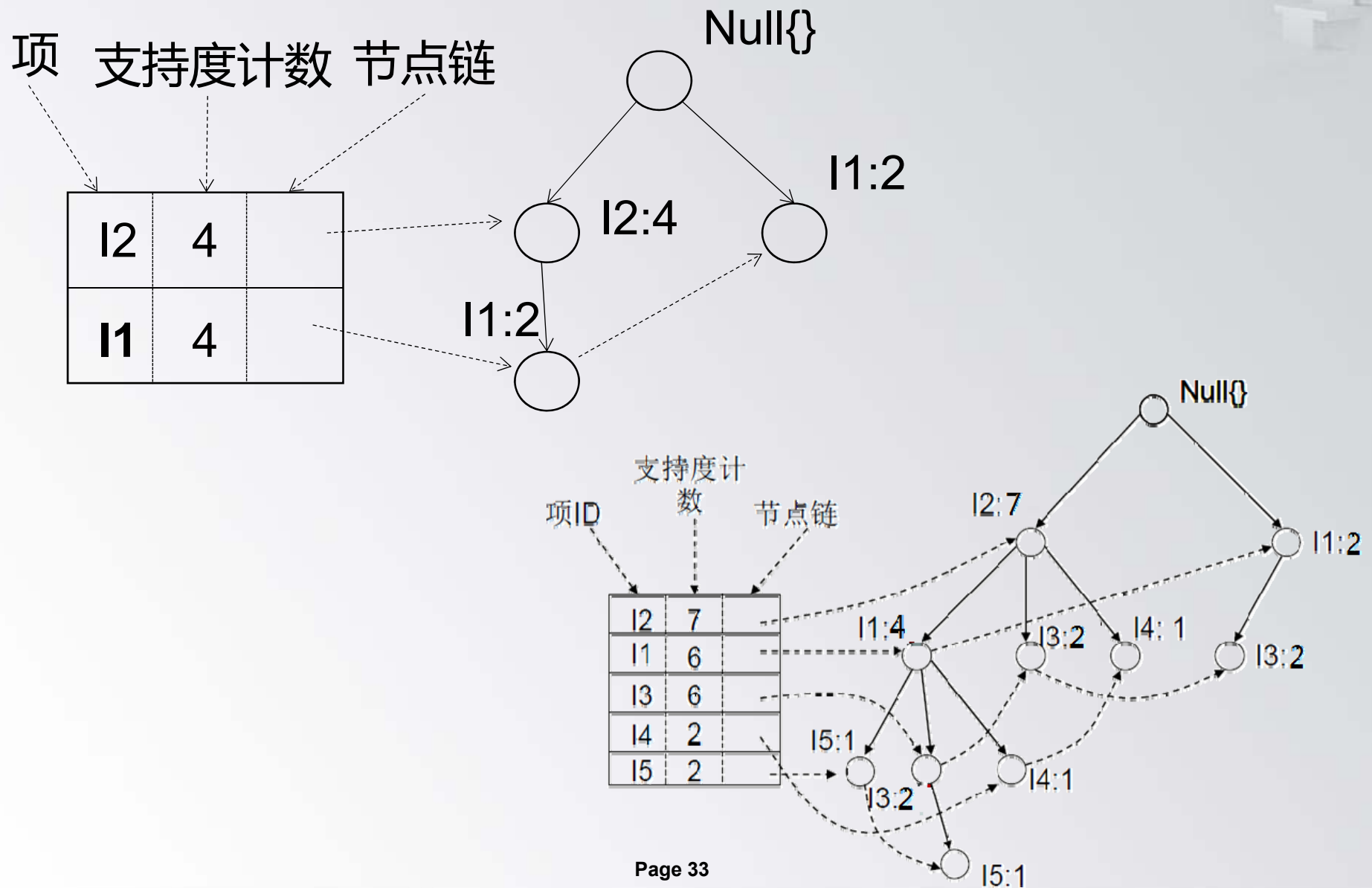
对项I4挖掘

产生的频繁模式

项	条件模式基	条件FP树	产生的频繁模式
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}



与条件节点 I_3 相关联的条件FP树



挖掘结果



项	条件模式基	条件FP树	产生的频繁模式
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}

频繁模式增长的优势

- 将发现的长频繁模式问题转换成较小的条件数据库中递归地搜索一些较短的模式，然后连接后缀。
- 对于挖掘长的频繁模式和短的频繁模式它都是有效的和可伸缩的，并且大约比 Apriori 算法快一个数量级。

频繁模式增长的局限

- 当数据库很大时, 构造基于主存的FP树有时是不现实的。
- 将数据库划分成投影数据库的集合, 然后在每个投影数据库上构造FP树并在每个投影数据库中挖掘。

使用垂直数据格式挖掘频繁项集

最小支持度计数为2

TID	项集
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

L1

项集	TID集
A	{10, 30}
B	{20, 30, 40, }
C	{10, 20, 30}
D	{10}
E	{20, 30, 40}

L2

项集	TID集
{A, C}	{10, 30}
{B, C}	{20, 30}
{B, E}	{20, 30, 40}
{C, E}	{20, 30}

L3

项集	TID集
{B, C, E}	{20, 30}

目录

- 基本概念
- 频繁项集挖掘方法
- 哪些模式是有趣的：模式评估方法
 - 强规则不一定是有趣的
 - 从关联分析到相关分析
 - 模式评估度量比较

强规则不一定是有趣的

- 示例：假设我们对涉及购买计算机游戏和录像的allelectronices的事务感兴趣，在10000个事务中，6000个顾客事务包含计算机游戏，7500个事务包含录像，4000个事务同时包含，规定minsup=30%，mincov=60%，并且服从规则：

$$buys(X, "computer_games") \Rightarrow buys(X, "videos")$$

可知这是强关联规则，其sup=40%，cov=66%，分别满足minsup和mincov，然而购买录像的概率为75%，比66%还高。

- 结论：规则A => B的置信度有一定的欺骗性，这并不能度量A 和B之间的实际强度。

由关联分析到相关分析

- 需要一种度量事件间的相关性或依赖性的指标
 - A与B的相关性,
$$\text{corr}_{A,B} = \frac{P(A \cup B)}{P(A)P(B)} = P(B | A) / P(B)$$
- 当项集A的出现独立于项集B的出现时, $P(A \cup B) = P(A)P(B)$, 即 $\text{corr}(A,B) = 1$, 表明A与B无关, $\text{corr}(A,B) > 1$ 表明A与B正相关, $\text{corr}(A,B) < 1$ 表明A与B负相关
 - 将相关性指标用于前面的例子, 可以得出录像带和游戏将的相关性为
 - $P(\{\text{game}, \text{video}\}) / (P(\{\text{game}\}) \times P(\{\text{video}\})) = 0.4 / (0.75 \times 0.6) = 0.89$
 - 结论: 录像带和游戏之间存在负相关

模式评估度量比较

- 评估度量的模式:

- 度
- λ^2
- 全置信度
- 最大置信度
- Kulczynski
- 余弦

练习

- 练习关联规则的编程实现和工具使用



Thank You!

Q&A