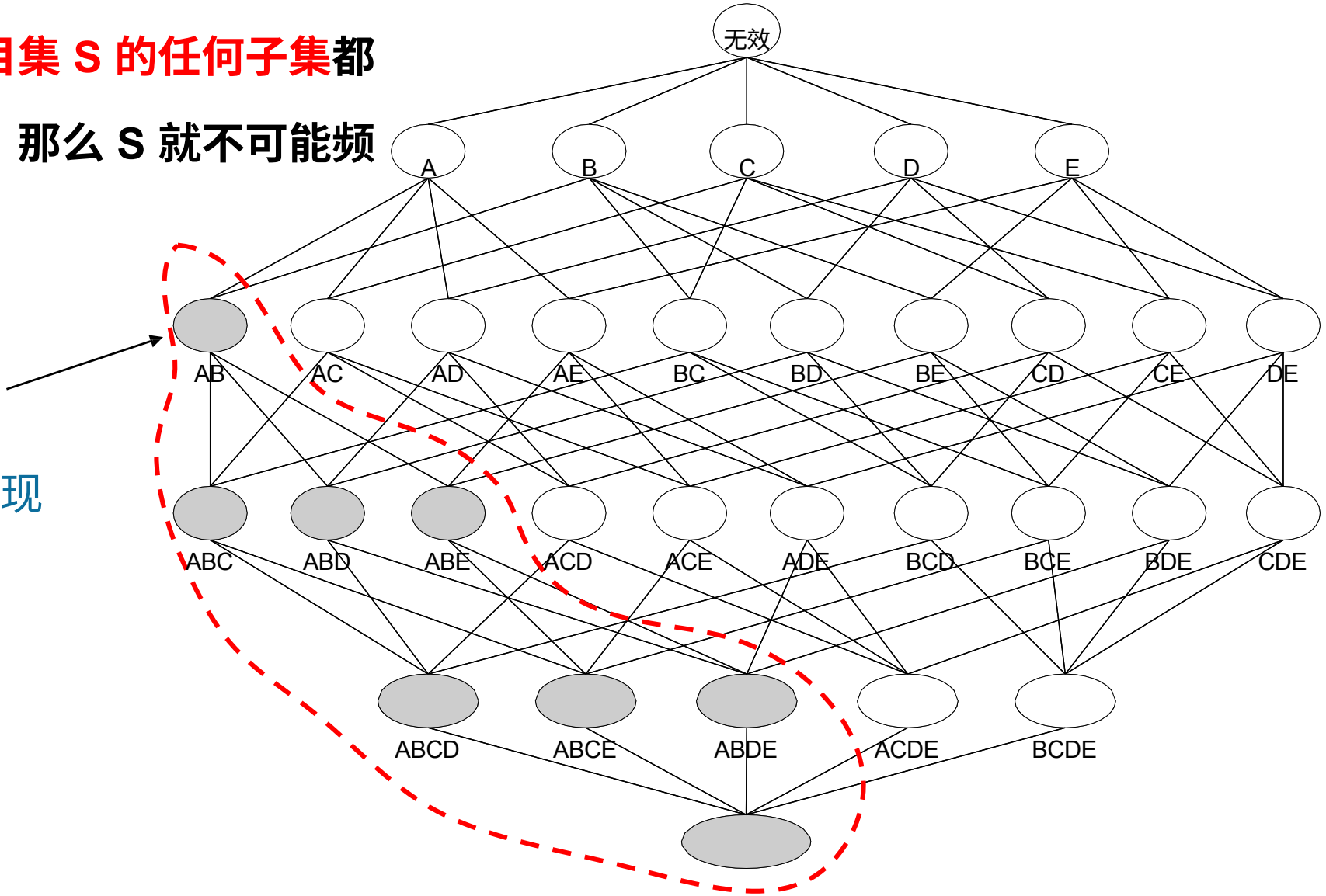
说明 Apriori 原则

如果项目集 S 的任何子集都不频繁，那么 S 就不可能频繁出现

不经常发现



剪枝超重

ABCDE

减少候选人数量

- 由于支持度量的以下特性，Apriori 原则成立：

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

  - 项目集的支持度永远不会超过其子集的支持度
  - 这就是支持的反单调特性

说明 Apriori 原则

TID	项目
1	面包、牛奶
2	啤酒、B读物、尿布、鸡蛋
3	啤酒、可乐、尿布、牛奶
4	啤酒、B读物、尿布、牛奶
5	面包、可乐、尿布、牛奶

项目（1-项目集）

项目	计数
面包	4
可乐	2
牛奶	4
啤酒	3
尿布	4
鸡蛋	1

最低支持率 = 3

如果考虑到每个子集、

$$ {}^{6}C_{1} + {}^{6}C_{2} + {}^{6}C_{3} $$
$$ 6 + 15 + 20 = 41 $$

采用基于支持的修剪、

$$ 6 + 6 + 1 = 13 $$

说明 Apriori 原则

TID	项目
1	面包、牛奶
2	啤酒、B读物、尿布、鸡蛋
3	啤酒、可乐、尿布、牛奶
4	啤酒、B读物、尿布、牛奶
5	面包、可乐、尿布、牛奶

项目（1-项目集）

项目	计数
面包	4
可乐	2
牛奶	4
啤酒	3
尿布	4
鸡蛋	1

最低支持率 = 3

如果考虑到每个子集、

$$ {}_6C_1 + {}_6C_2 + {}_6C_3 $$
$$ 6 + 15 + 20 = 41 $$

采用基于支持的修剪、

$$ 6 + 6 + 1 = 13 $$

说明 Apriori 原则

项目	计数
面包	4
可乐	2
牛奶	4
啤酒	3
尿布	4
鸡蛋	1

项目（1-项目集）

项目集
{面包、牛奶}.
{面包、啤酒 }
{面包、尿布}
{啤酒、牛奶}
{尿布、牛奶}
{啤酒、尿布}

成对（2 个项目集）

（无需生成

涉及可乐或鸡蛋的候选人）

最低支持率 = 3

如果考虑到每个子集、

$$6C1 + 6C2 + 6C3$$
$$6 + 15 + 20 = 41$$

采用基于支持的修剪、

$$6 + 6 + 1 = 13$$

说明 Apriori 原则

项目	计数
面包	4
可乐	2
牛奶	4
啤酒	3
尿布	4
鸡蛋	1

项目（1-项目集）

项目集	计数
{面包、牛奶}.	3
{面包、啤酒}	2
{面包、尿布}	3
{啤酒、牛奶}	2
{尿布、牛奶}	3
{啤酒、尿布}	3

成对（2 个项目集）

（无需生成

涉及可乐或鸡蛋的候选人）

最低支持 = 3

如果考虑到每个子集、

$$ {}_6C_1 + {}_6C_2 + {}_6C_3 $$
$$ 6 + 15 + 20 = 41 $$

采用基于支持的修剪、

$$ 6 + 6 + 1 = 13 $$

说明 Apriori 原则

项目	计数
面包	4
可乐	2
牛奶	4
啤酒	3
尿布	4
鸡蛋	1

项目（1-项目集）

项目集	计数
{面包、牛奶}.	3
{面包、啤酒} 2	2
{面包、尿布}	3
{牛奶、啤酒}	2
{牛奶、尿布}(英文)	3
{啤酒、尿布}	3

成对（2 个项目集）

（无需生成

涉及可乐或鸡蛋的候选人）

三胞胎（3 个项目集）

最低支持率 = 3

如果考虑到每个子集、

$$ {}^{6}C_{1} + {}^{6}C_{2} + {}^{6}C_{3} $$

$$ 6 + 15 + 20 = 41 $$

采用基于支持的修剪、

$$ 6 + 6 + 1 = 13 $$

项目集	计数
{面包、牛奶、尿布}.	2

Apriori 原则

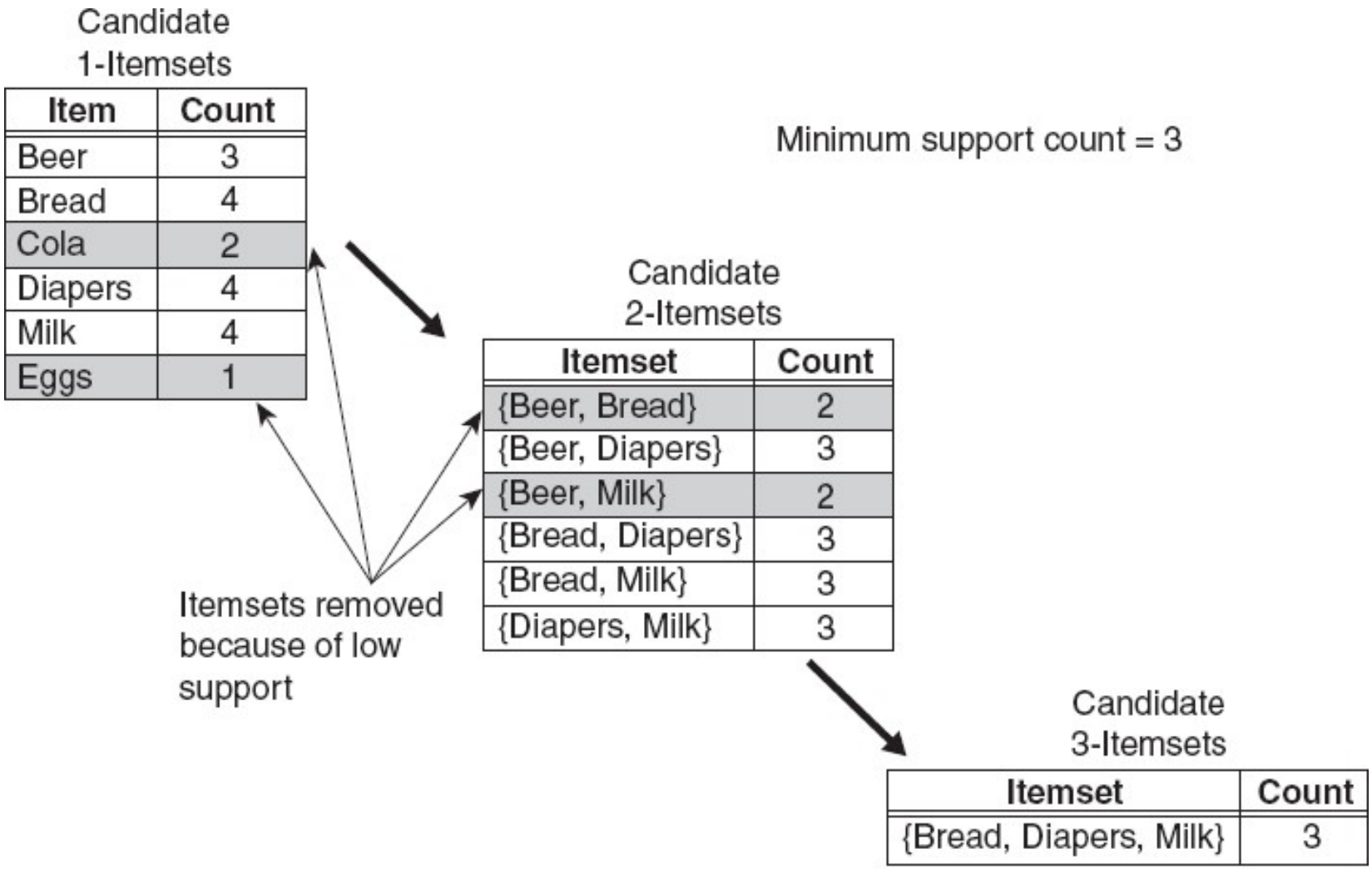


Figure 6.5. Illustration of frequent itemset generation using the *Apriori* algorithm.

Apriori 算法

- F_k：频繁 k 项集
- C_k：候选 k 项集

- 算法
  - 让 k=1
  - 生成 F_1 = {经常出现的 1-itemsets}
  - 重复操作，直到 F_k 空为止
    - **候选人生成**：从 F 生成 C_{k+1k}
    - **候选项剪枝**在 C_{k+1} 中剪切包含长度为 k 的不常见子集的候选项集

Apriori 算法

- **支持率计算**：通过扫描数据库，计算 C_{k+1} 中每位候选人的支持率
- **候选词消除**：消除 C_{k+1} 中不经常出现的候选项，只留下经常出现的候选项 => F_{k+1}

Apriori 算法

- Apriori 概要

  (按级别、候选人**生成和测试)**

  - 最初，扫描数据库一次，以获取频繁的 1 个项目集
  - 重复
    - 从长度为 k 的频繁项集中生成长度为 (k+1) 的候选项集
    - 根据数据库测试候选项，找出频繁出现的 (k+1)- 项目集
    - 设 k := k +1
  - 直到无法生成频繁集或候选集为止

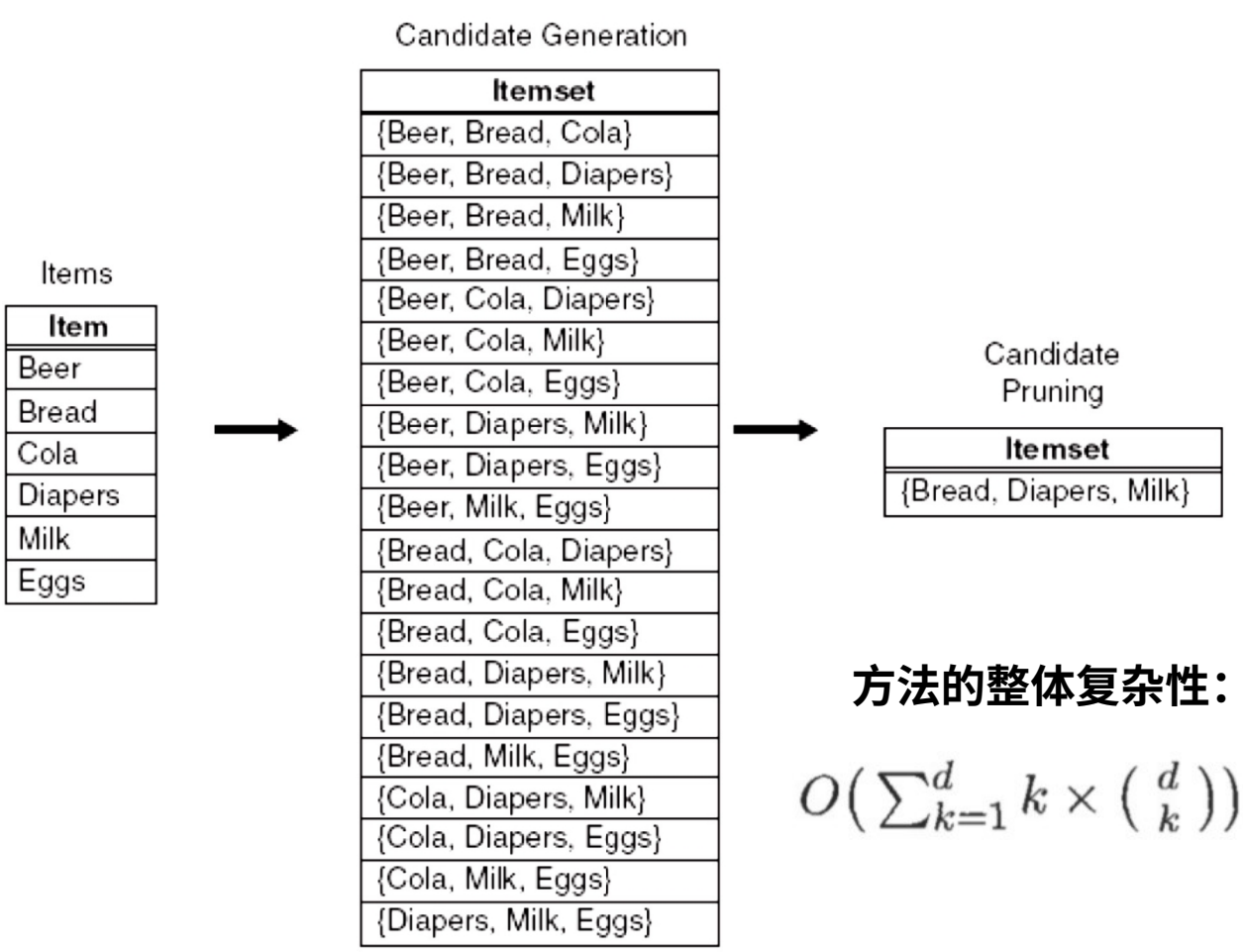
Apriori 算法

- 返回得出的所有频繁项集

候选人产生

- 有效候选人产生程序的**要求**
  - 避免产生过多不必要的候选人
  - 确保候选人集完整
  - 不应多次生成相同的候选项集

候选人的产生：蛮力法



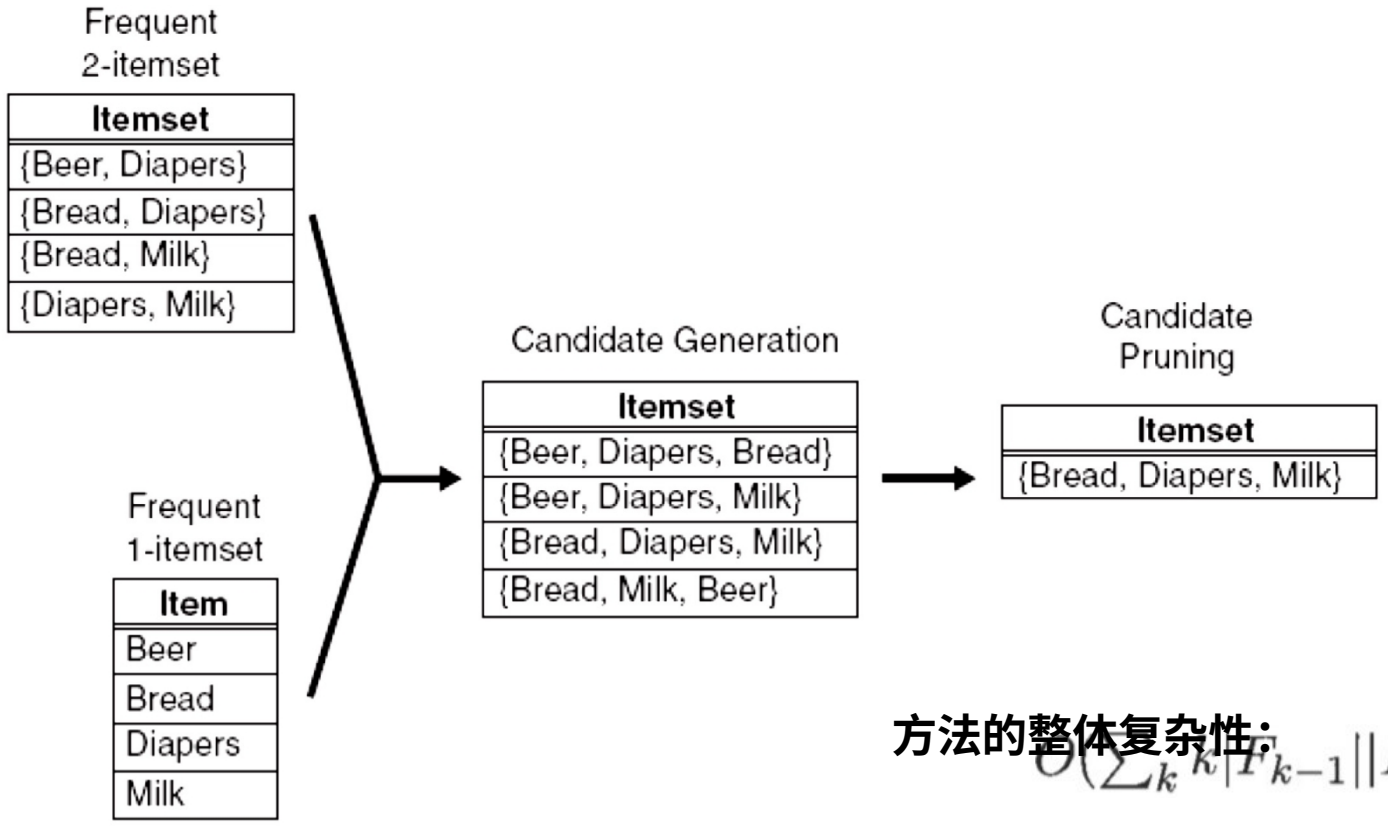
方法的整体复杂性：

$$O\left(\sum_{k=1}^{d} k \times \binom{d}{k}\right) = O\left(d \cdot 2^{d-1}\right)$$

Figure 6.6. A brute-force method for generating candidate 3-itemsets.

候选项生成：合并 Fk-1 和 F1 项目集

用其他频繁项扩展每个频繁 (k-1) 项集



方法的整体复杂性： $O(\sum_k k|F_{k-1}||F_1|)$.

Figure 6.7. Generating and pruning candidate k-itemsets by merging a frequent $(k-1)$-itemset with a frequent item. Note that some of the candidates are unnecessary because their subsets are infrequent.

候选人修剪

- 一个候选 k 项集 $X = \{i_1, i_2, \ldots i\}_k$
  - 确定它的所有适当子集，X - $\{i_j\}$、经常发生。
  - 如果 *k* 个子集中的 *m* 个子集被用来生成候选者，那么我们只需要检查提醒的 *k-m* 个子集即可

候选项生成：合并 Fk-1 和 F1 项目集

- 局限性
  - 可多次产生同一候选人
  - 仍然会产生大量不必要的候选人

候选者生成：Fk-1 x Fk-1 方法

只有当一对频繁 (k-1) 项目集的前 k-2 个项目相同时，才将其合并

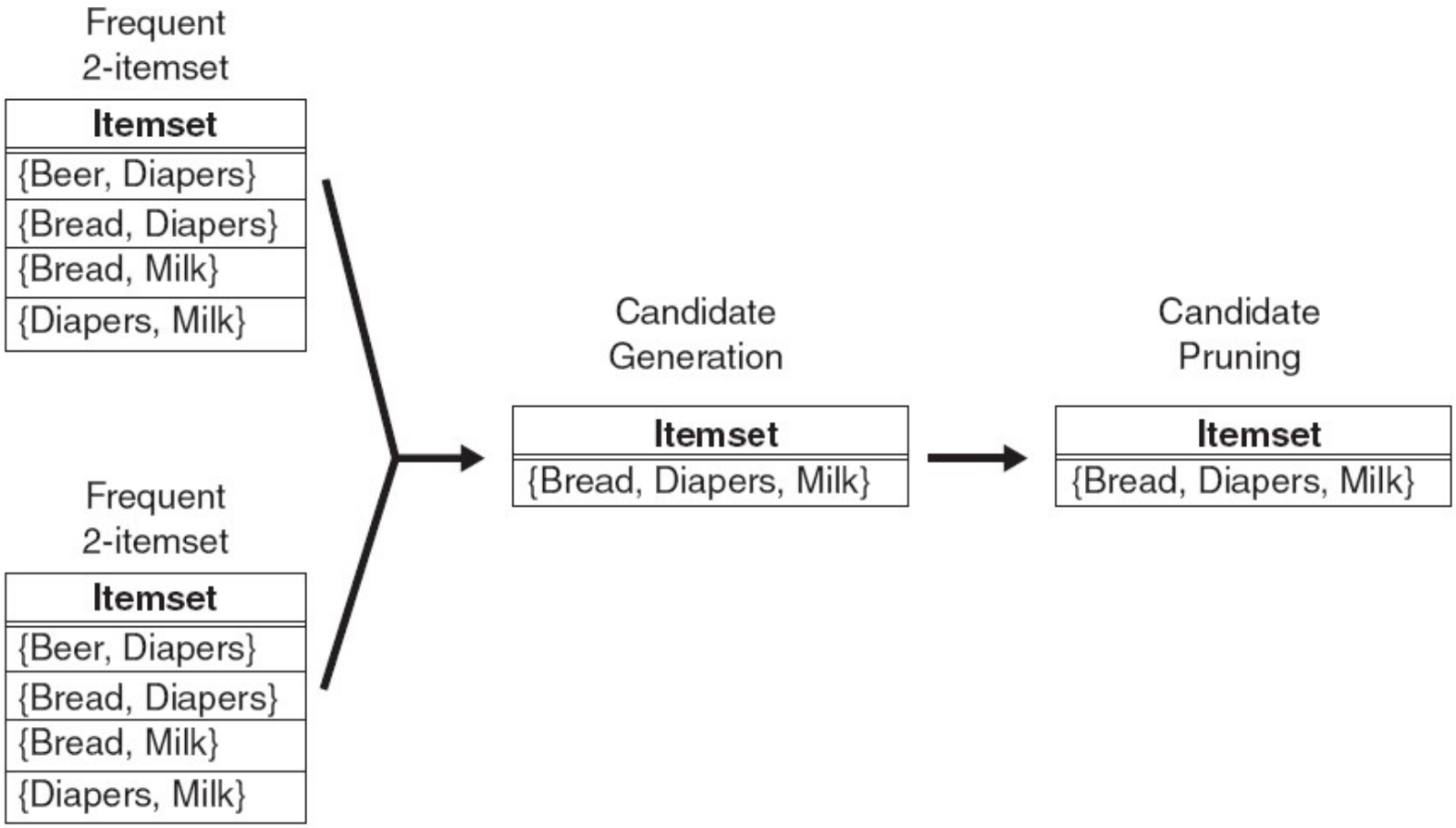


Figure 6.8. Generating and pruning candidate k-itemsets by merging pairs of frequent $(k-1)$-itemsets.

候选世代：$F_{k-1} \times F_{k-1}$ 方法

- 如果两个频繁 (k-1) 项目集的前 (k-2) 个项目相同，则合并这两个项目集

- f_3 = {abc,abd,abe,acd,bcd,bde,cde}
  - 合并（ABC、**ABD**）= ABCD
  - 合并（**ABC**、**ABE**）= ABCE
  - 合并（**ABD**、**ABE**）= ABDE
  - 不要合并（ABD,**ACD**），因为它们只共享长度为 1

候选世代：F_{k-1} x F_{k-1} 方法

而不是 2 的前缀

候选世代：$F_{k-1} \times F_{k-1}$ 方法

- 设 F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE} 为频繁三项目集合

- L_4 = {ABCD,ABCE,ABDE} 是生成的候选 4 项集的集合（来自上一张幻灯片）。

- 候选人修剪
  - 删除 ABCE，因为 ACE 和 BCE 不经常出现

候选世代：$F_{k-1} \times F_{k-1}$ 方法

- 删除 ABDE，因为 ADE 不经常出现

- 候选人剪枝后：L_4 = {ABCD}

替代 $F_{k-1} \times F_{k-1}$ 方法

- 如果第一个项目集的最后 (k-2) 个项目与第二个项目集的前 (k-2) 个项目相同，则合并两个频繁 (k-1) 项目集。

- f_3 = {abc,abd,abe,acd,bcd,bde,cde}
  - 合并（**ABC**、**BCD**）= ABCD
  - 合并（**ABD**、**BDE**）= ABDE
  - 合并（**ACD**，**CDE**） = ACDE
  - 合并（**BCD**、**CDE**） = BCDE

备用 $F_{k-1} \times F_{k-1}$ 方法的候选修剪

- 设 F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE} 为频繁三项目集合

- L_4 = {ABCD,ABDE,ACDE,BCDE} 是生成的候选 4 项集的集合（来自上一张幻灯片）。

- 候选人修剪
  - 删除 ABDE，因为 ADE 不经常出现

- 由于 ACE 和 ADE 不常发生，因此删除 ACDE
- 修剪 BCDE，因为 BCE

- 候选人剪枝后：L_4 = {ABCD}

Apriori 算法

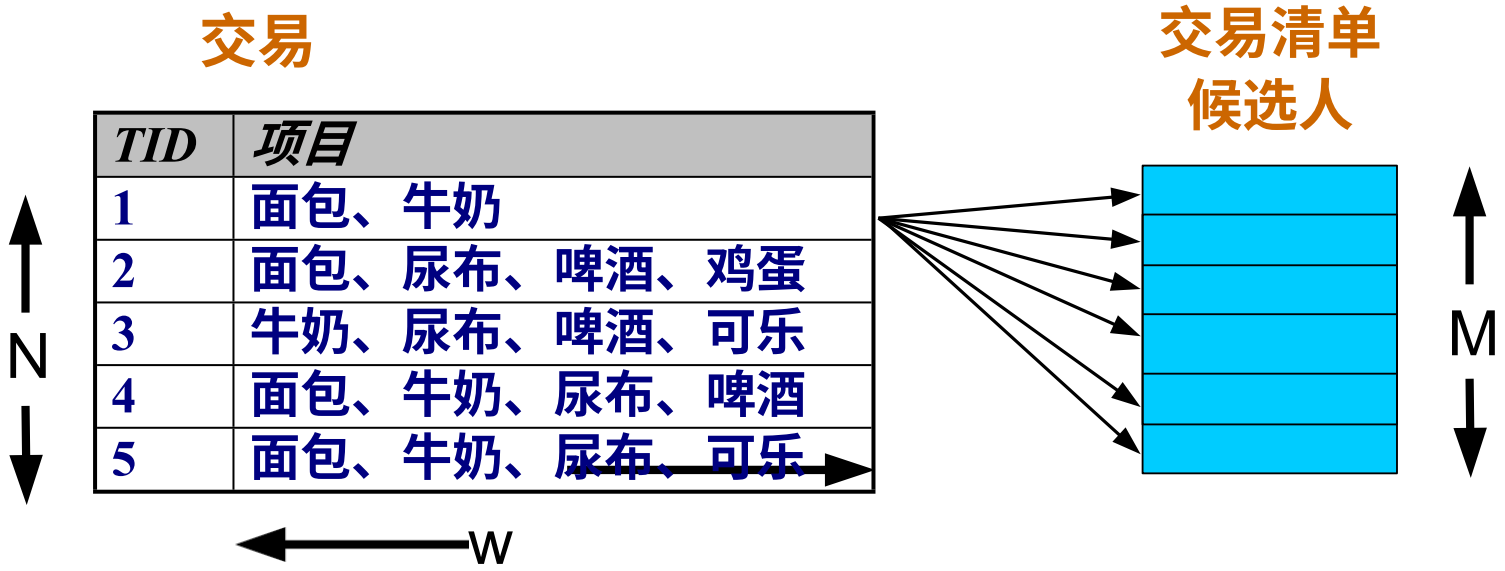
- F_k：频繁 k 项集
- C_k：候选 k 项集

- 算法
  - 让 k=1
  - 生成 F_1 = {经常出现的 1-itemsets}
  - 重复操作，直到 F_k 空为止
    - **候选人生成**：从 F 生成 C_{k+1k}
    - **候选项剪枝**在 C_{k+1} 中剪切包含长度为 k 的不常见子集

的候选项集

- **支持率计算**：通过扫描数据库，计算 C_{k+1} 中每位候选人的支持率
- **候选词消除**：消除 C_{k+1} 中不经常出现的候选项，只留下经常出现的候选项 => F_{k+1}

支持计数


交易
交易清单
候选人
TID
项目
1
面包、牛奶
2
面包、尿布、啤酒、鸡蛋
3
牛奶、尿布、啤酒、可乐
4
面包、牛奶、尿布、啤酒
5
面包、牛奶、尿布、可乐
N
M
w

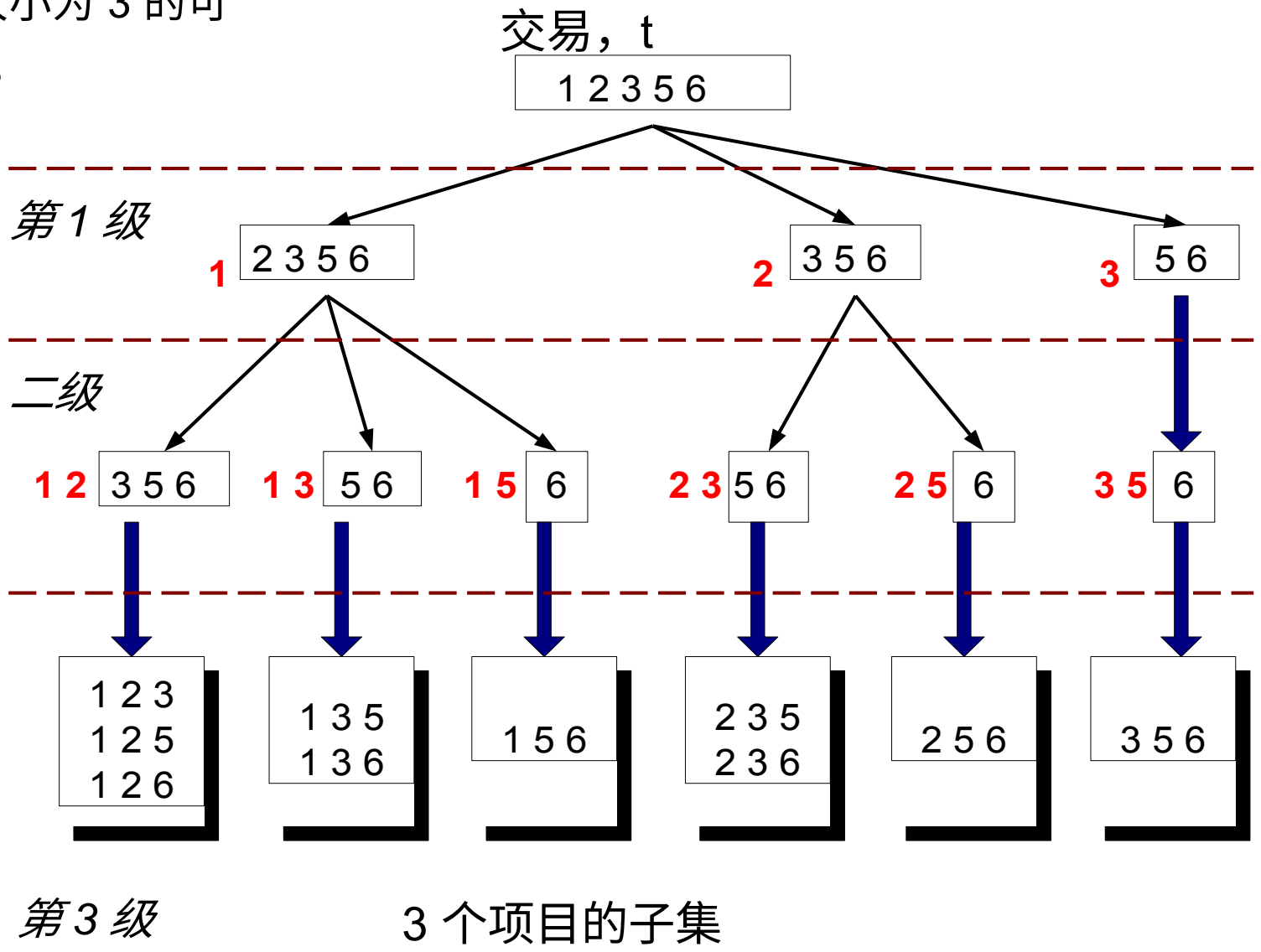
Apriori 算法

Algorithm 6.1 Frequent itemset generation of the *Apriori* algorithm.

1: $k = 1$.
2: $F_k = \{ i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup \}$. {Find all frequent 1-itemsets}
3: **repeat**
4: $k = k + 1$.
5: C_k = apriori-gen(F_{k-1}). {Generate candidate itemsets}
6: **for** each transaction $t \in T$ **do**
7: C_t = subset(C_k, t). {Identify all candidates that belong to t}
8: **for** each candidate itemset $c \in C_t$ **do**
9: $\sigma(c) = \sigma(c) + 1$. {Increment support count}
10: **end for**
11: **end for**
12: $F_k = \{ c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup \}$. {Extract the frequent k-itemsets}
13: **until** $F_k = \emptyset$
14: Result $= \bigcup F_k$.

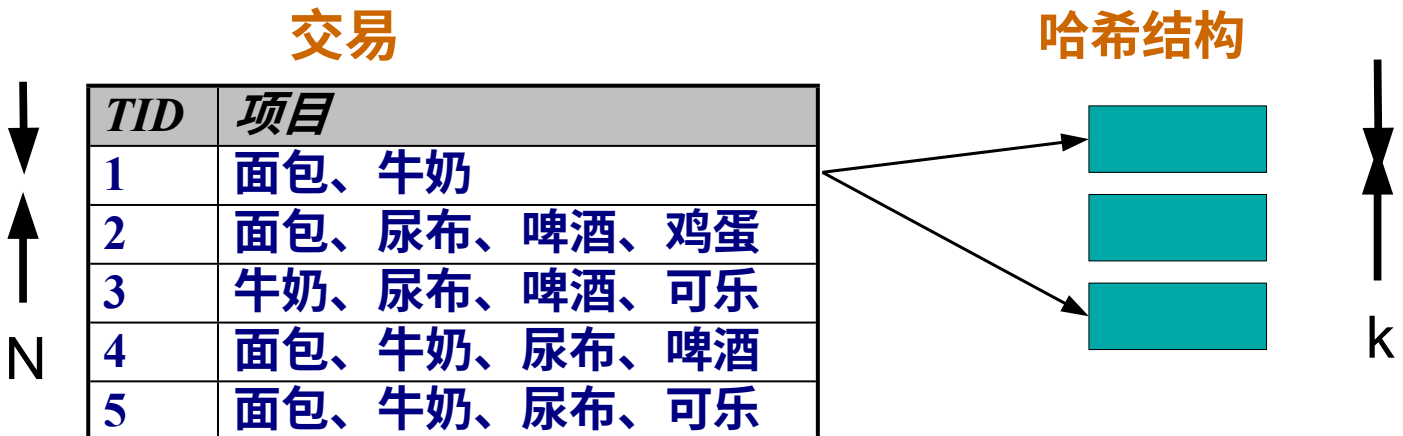
子集操作

给定事务 t，大小为 3 的可能子集有哪些?



减少比较次数

- 候选人计数：
  - 扫描事务数据库，确定每个候选项集的支持度
  - 为减少比较次数，可将候选数据存储在哈希结构中
    - 不将每笔交易与每个候选交易进行匹配，而是将其与散列桶中的候选交易进行匹配



水桶

生成哈希树

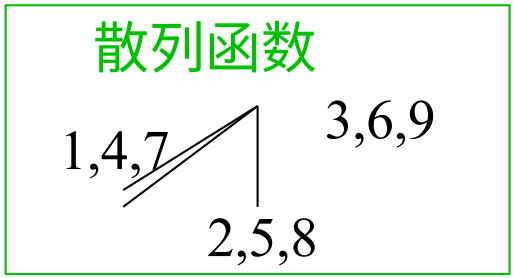
假设有 15 个长度为 3 的候选项集：

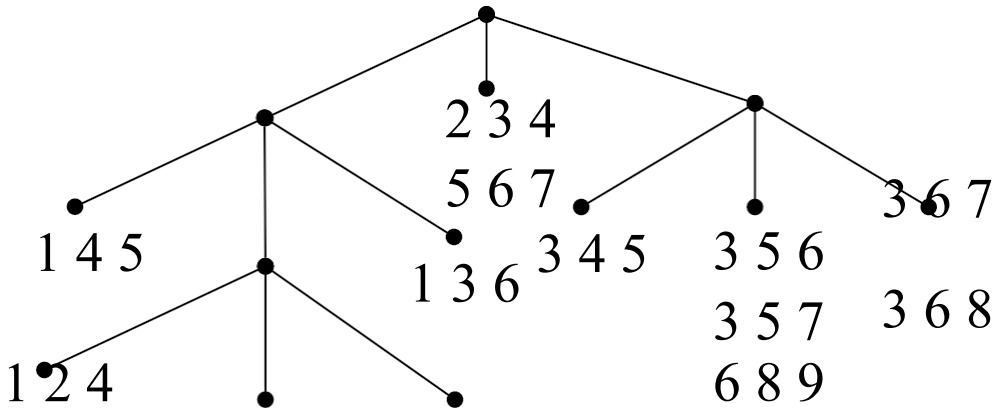
{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

您需要

- **散列函数**

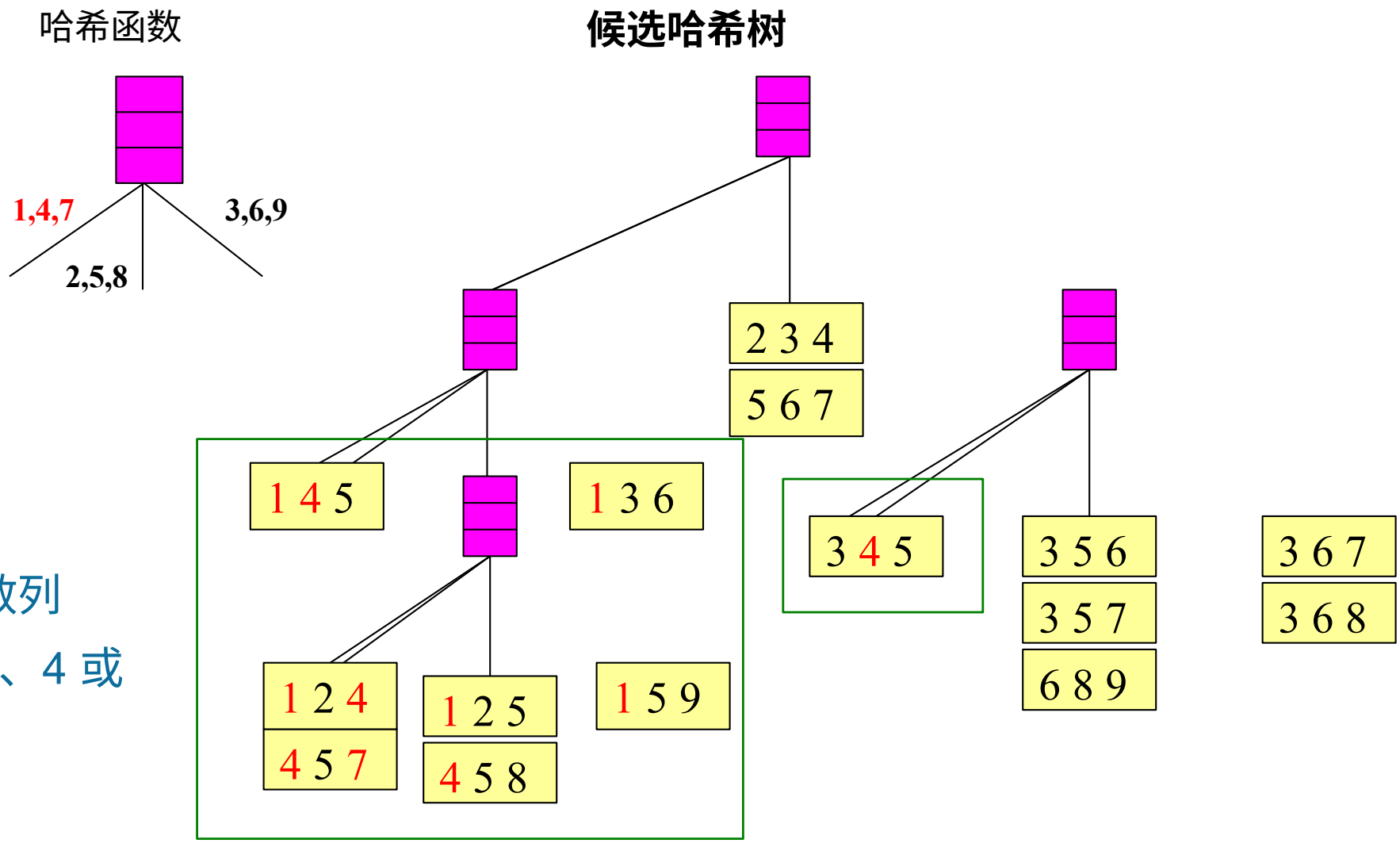
- **最大叶节点大小：叶节点中存储的项目集的最大数量（如果候选项目集的数量超过最大叶节点大小，则拆分节点）**



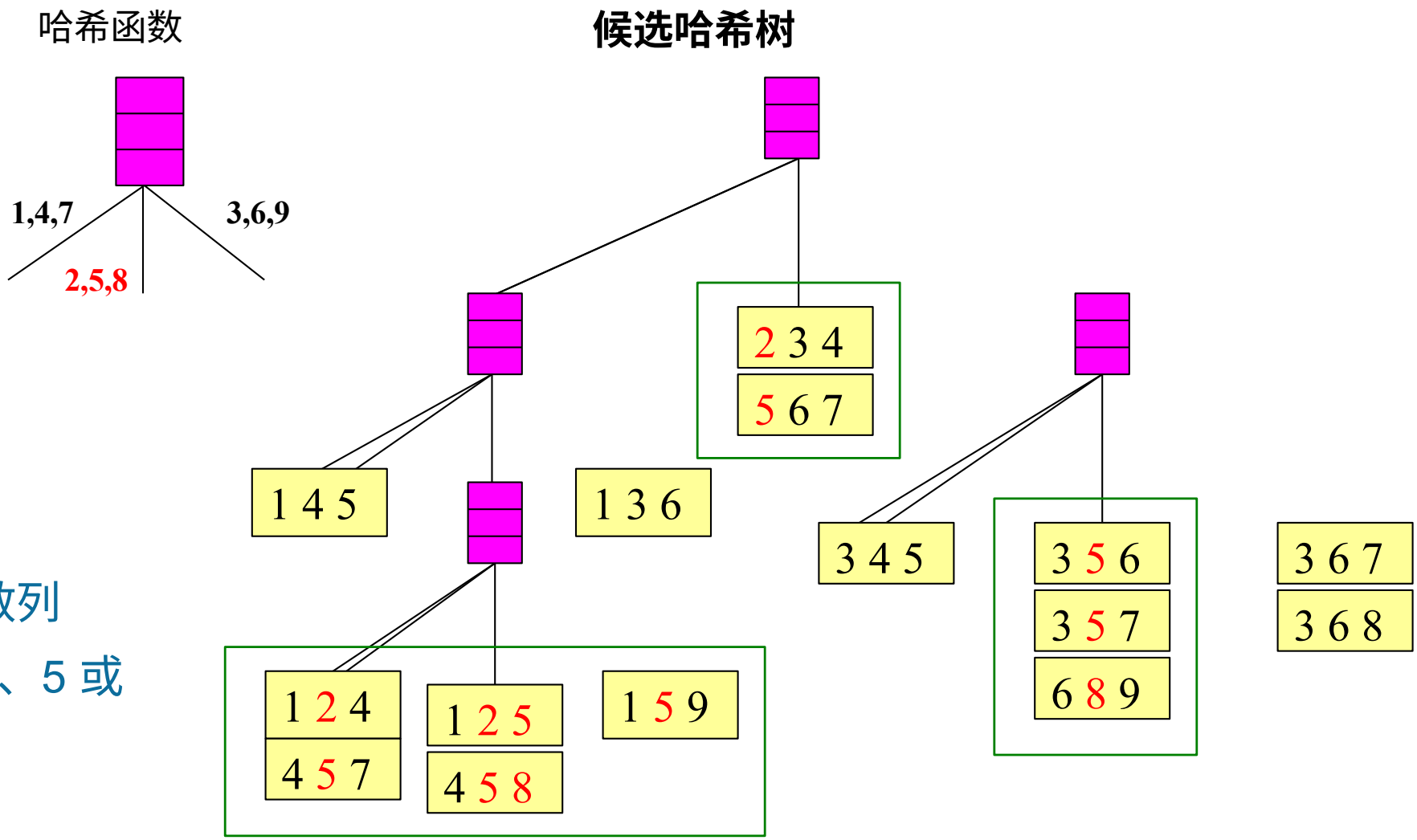


457 125 159
458

关联规则发现哈希树


哈希函数
候选哈希树
1,4,7
2,5,8
3,6,9
2 3 4
5 6 7
1 4 5
1 3 6
1 2 4
4 5 7
1 2 5
4 5 8
1 5 9
3 4 5
3 5 6
3 5 7
6 8 9
3 6 7
3 6 8
散列
1、4 或
7

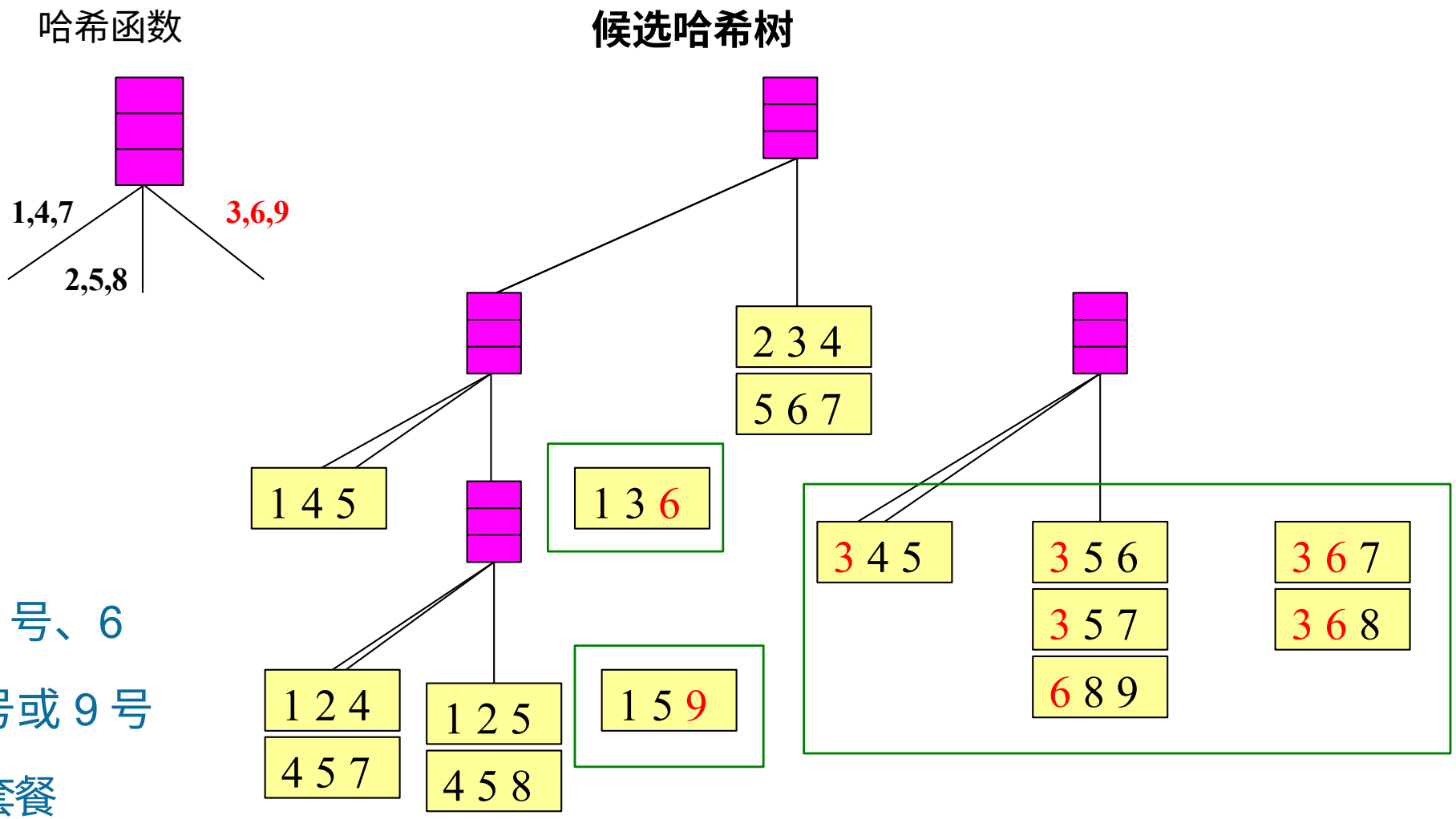
关联规则发现哈希树



散列

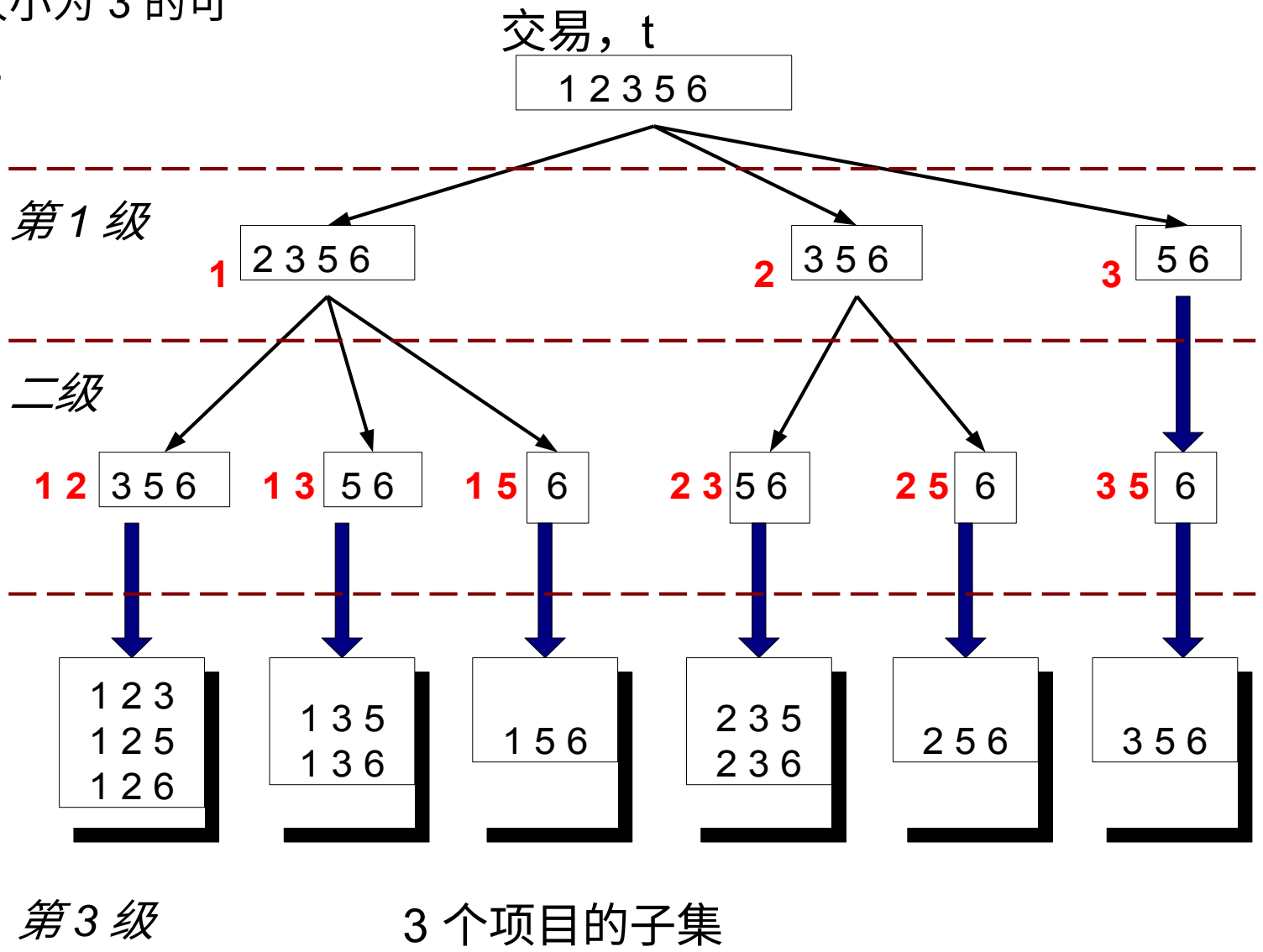
2、5 或 8

关联规则发现哈希树

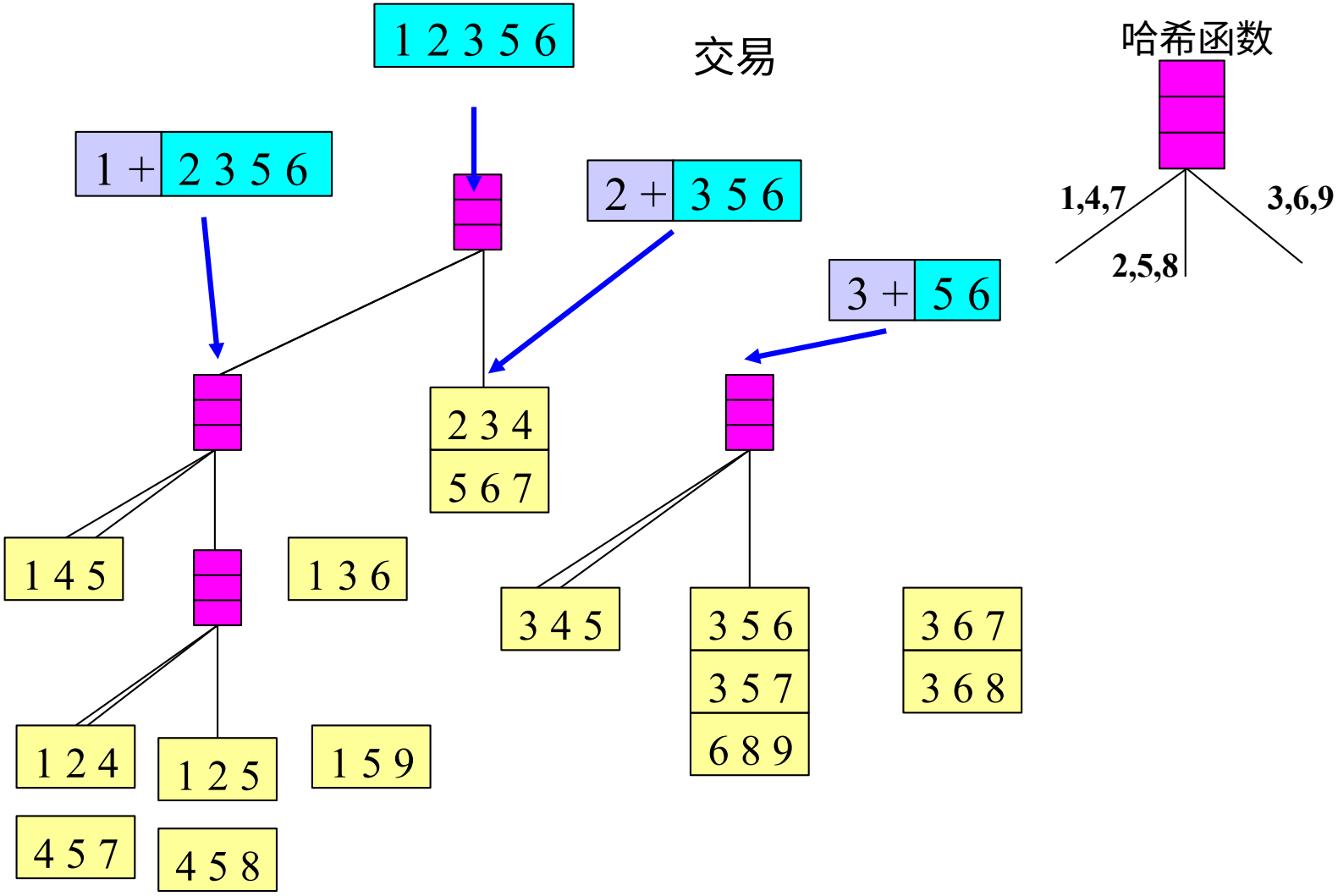

哈希函数
候选哈希树
1,4,7
3,6,9
2,5,8
2 3 4
5 6 7
1 4 5
1 3 6
1 2 4
4 5 7
1 2 5
4 5 8
1 5 9
3 4 5
3 5 6
3 5 7
6 8 9
3 6 7
3 6 8
3 号、6
号或 9 号
套餐

子集操作

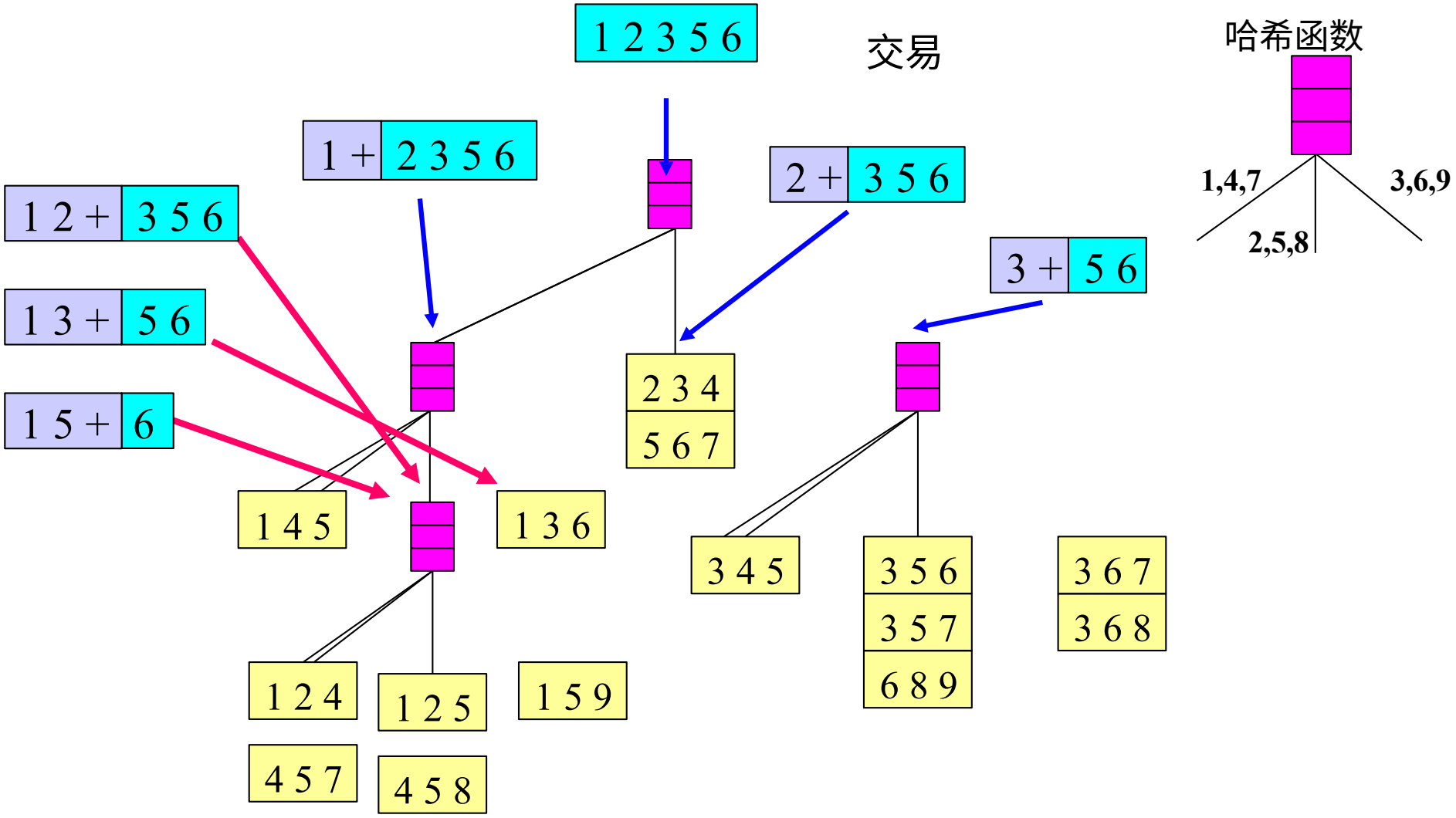
给定事务 t，大小为 3 的可能子集有哪些?



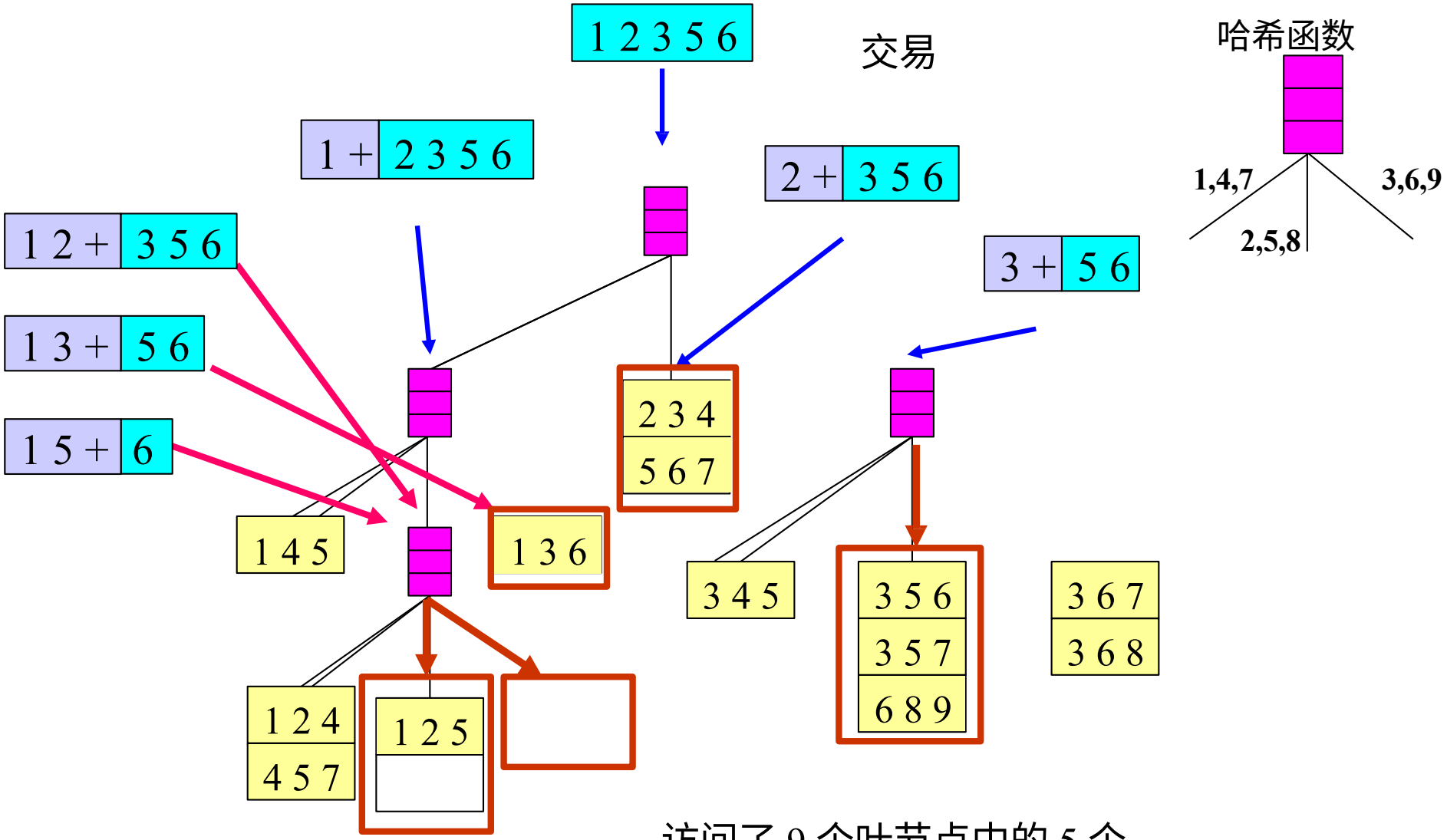
使用哈希树进行子集操作



使用哈希树进行子集操作



使用哈希树进行子集操作



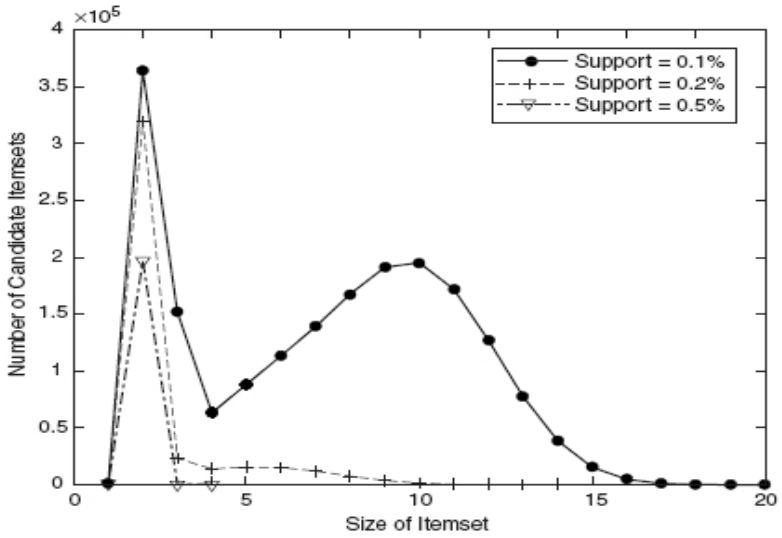
访问了 9 个叶节点中的 5 个

使用哈希树进行子集操作

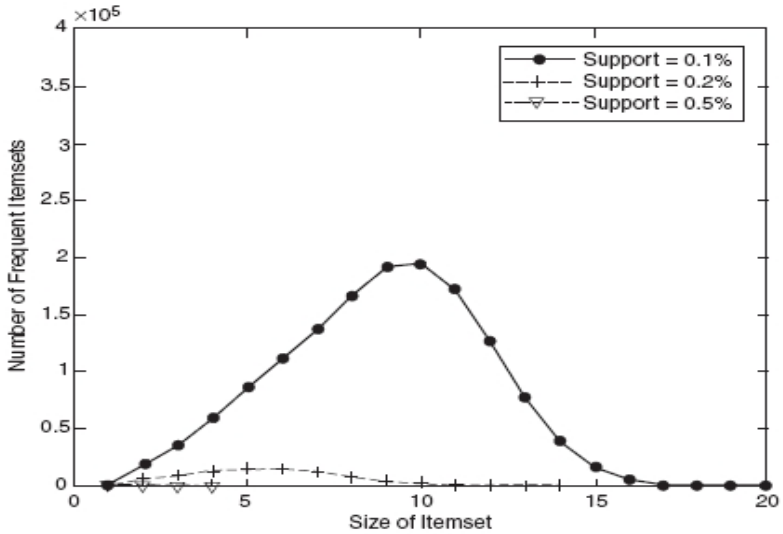
与 15 位候选人中的 9 位进行匹配交易

影响复杂性的因素

- 选择最低支助阈值
  - 降低支持阈值会产生更多的频繁项集
  - 这可能会增加候选项的数量和频繁项集的最大长度



(a) Number of candidate itemsets.



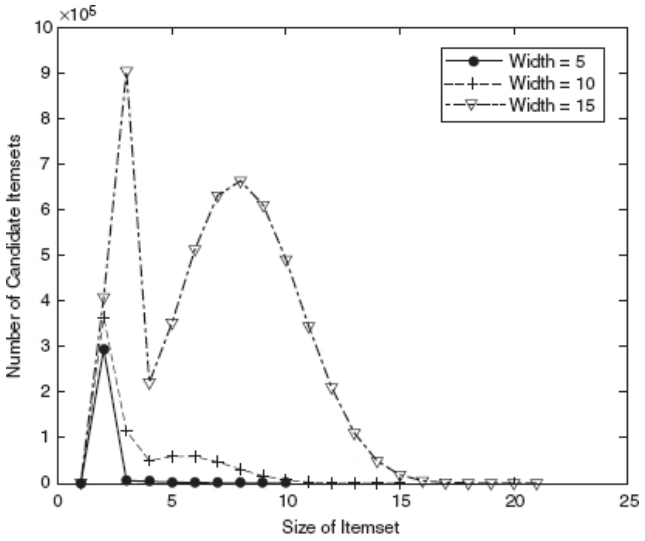
(b) Number of frequent itemsets.

影响复杂性的因素

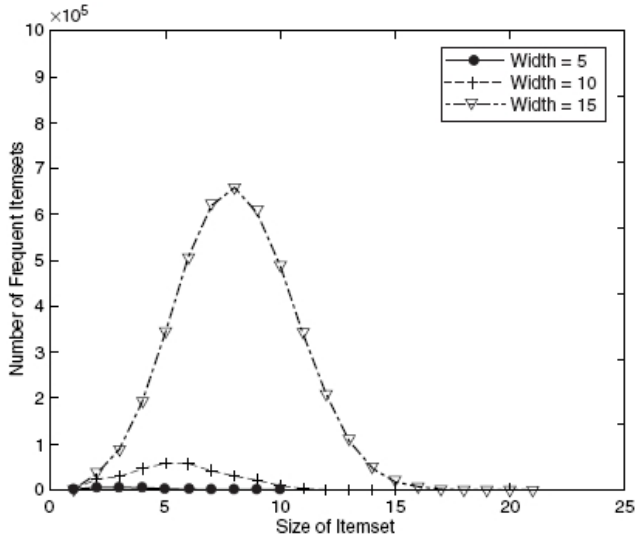
- 数据集的维度（项目数
  - 需要更多空间来存储每个项目的支持计数
  - 如果频繁项的数量也增加，计算和 I/O 成本也会增加
- 数据库大小
  - 由于 Apriori 会进行多次传递，因此算法的运行时间可能会随着事务数量的增加而增加

影响复杂性的因素

- 平均交易宽度
  - 交易宽度随数据集密度的增加而增加
  - 这可能会增加频繁项集的最大长度和哈希树的遍历次数（事务中的子集数随其宽度而增加）



(a) Number of candidate itemsets.



规则生成

- 给定频繁项集 L，找出所有非空子集 f ⊂ L，使得 f → L - f 满足最小置信度要求
  - 如果 {A,B,C,D} 是一个频繁项集，则候选规则为

ABC →D、	ABD →C、	ACD →B、	BCD →A、
A →BCD、	B →ACD、	C →ABD、	D →ABC
AB →CD、	AC → BD、	公元前 → 公元前、	BC →AD、
BD →AC、	CD →AB、		

- 如果 |L| = k，则有 $2^k - 2$ 条候选关联规则（忽略 L → ∅ 和 ∅ → L）。

规则生成

- 一般来说，置信度不具有反单调特性

  c(ABC →D) 可以大于或小于 c(AB →D)

- 但是，由同一项目集生成的规则的置信度具有反单调特性
  - 例如，假设 {A,B,C,D} 是一个频繁 4 项集：c(ABC → D) ≥ c(AB → CD) ≥ c(A → BCD)

规则生成

- 置信度与规则 RHS 上的项目数是反单调的

规则生成

- 定理

如果一条规则 $X \rightarrow Y - X$ 不满足置信度阈值、

则任何规则 $X' \rightarrow Y - X'$，其中 $X'' \subset X$，一定也不满足置信度阈值。

示例

B_1 = {m, c, b}　B_2 = {m, p, j}

B_3 = {m, c, b, n}　B_4 = {c, j}

B_5 = {m, p, b}　B_6 = {m, c, b, j}

B_7 = {c, b, j}　B_8 = {b, c}

- 支持阈值 $s = 3$，置信度 $c = 0.75$
- 1) 频项集：

  - {b,m}{b，c} {c，m} {c,j{c,j}{m,c,b}。
- 2) 生成规则：

  - b→m： c=4/6　b→c： c=5/6　b,c→m： c=3/5

- **m→b**: c=4/5

...

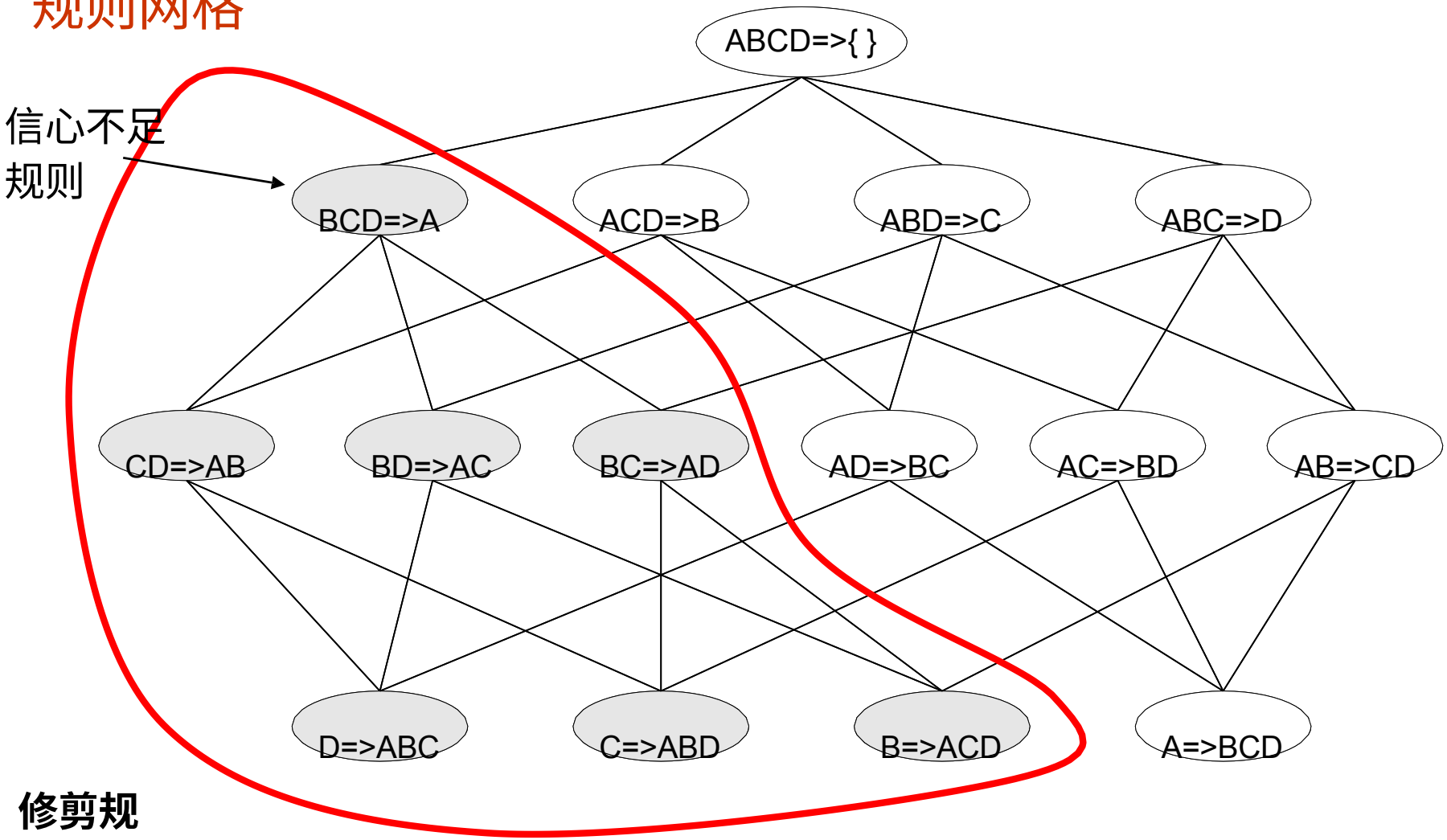
b，m→c： c=3/4

-

~~**b→c,m**: c=3/6~~

Apriori 算法的规则生成

规则网格



修剪规则

常项集的紧凑表示法

TID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

- 频繁项集数

$$= 3\times \sum_{k=1}^{10} \left\lfloor \begin{matrix} 10 \\ k \end{matrix} \right\rfloor$$

- 有些项目集是多余的，因为它们与超项目集的支持完全相同

- 需要紧凑的表示方法

压缩输出

- **为了减少规则的数量，我们可以对它们进行后处理，只输出规则：**
  - **最大频繁项集**

    不经常立即叠加

    - 提供更多修剪

  或

  - **封闭项目集：**

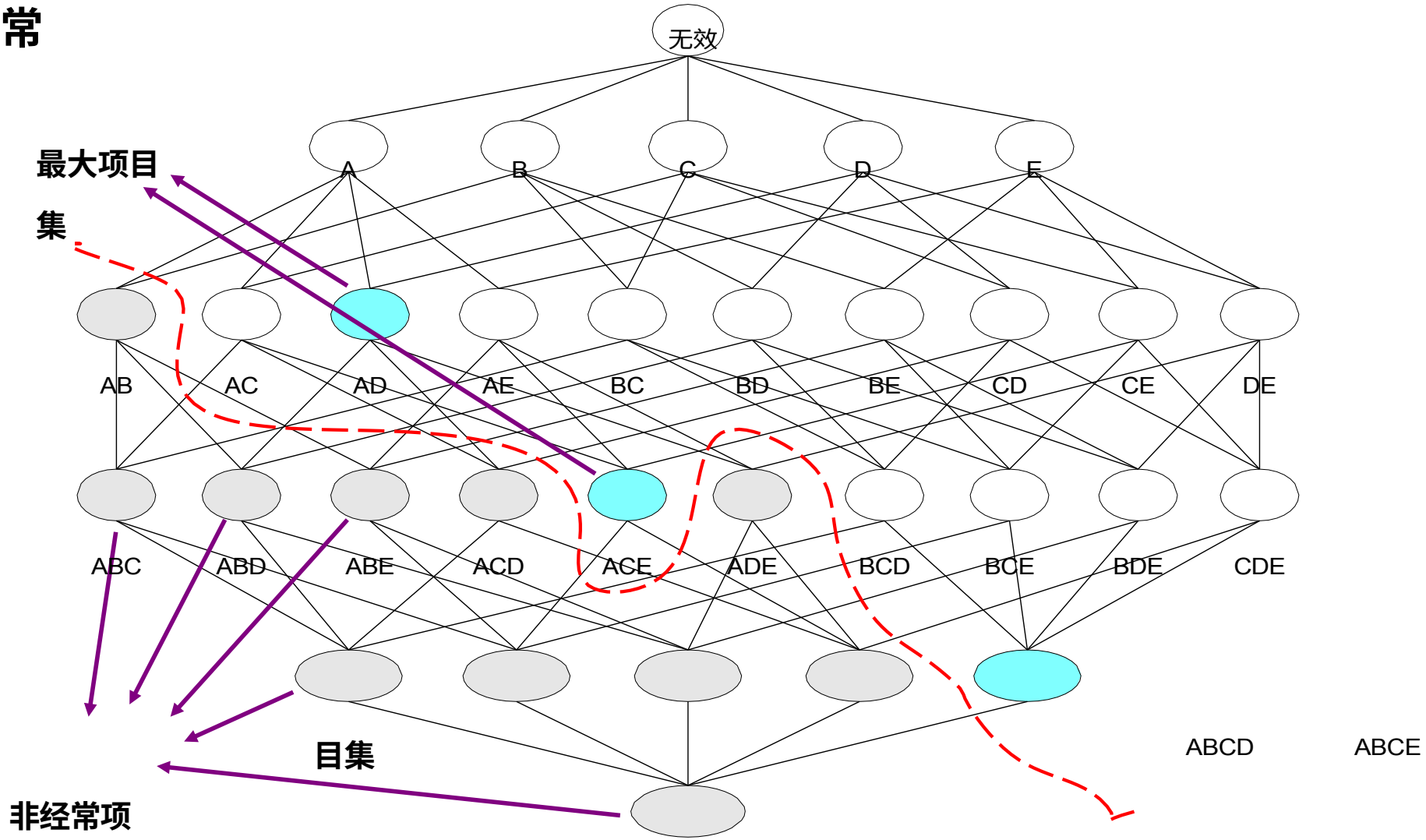
    没有直接超集具有相同的计数（> 0）

- 不仅能存储频繁信息，还能存储精确计数

最大常项集

如果一个项集没有一个直接超集，那么它就是最大频繁项集

经常

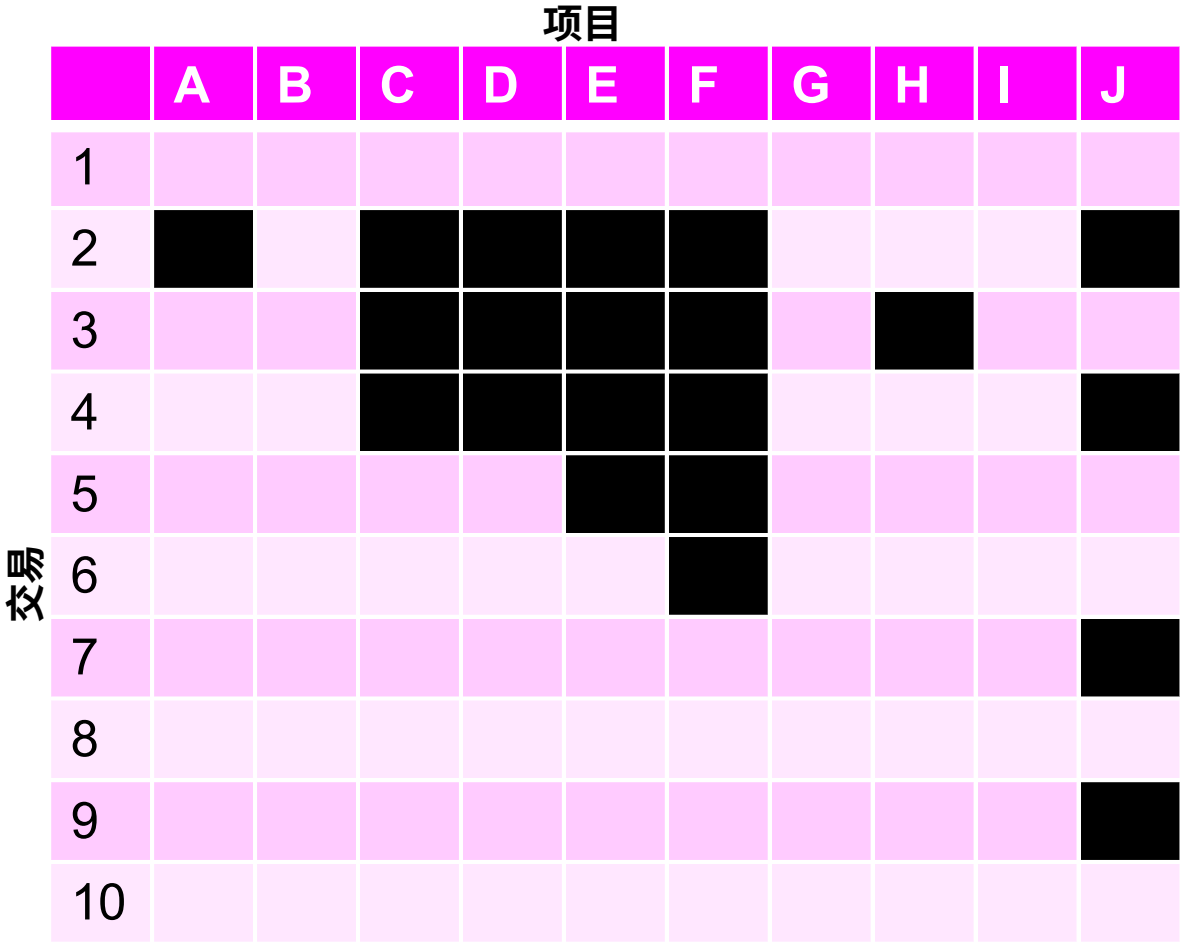


A
B
C
D

边境

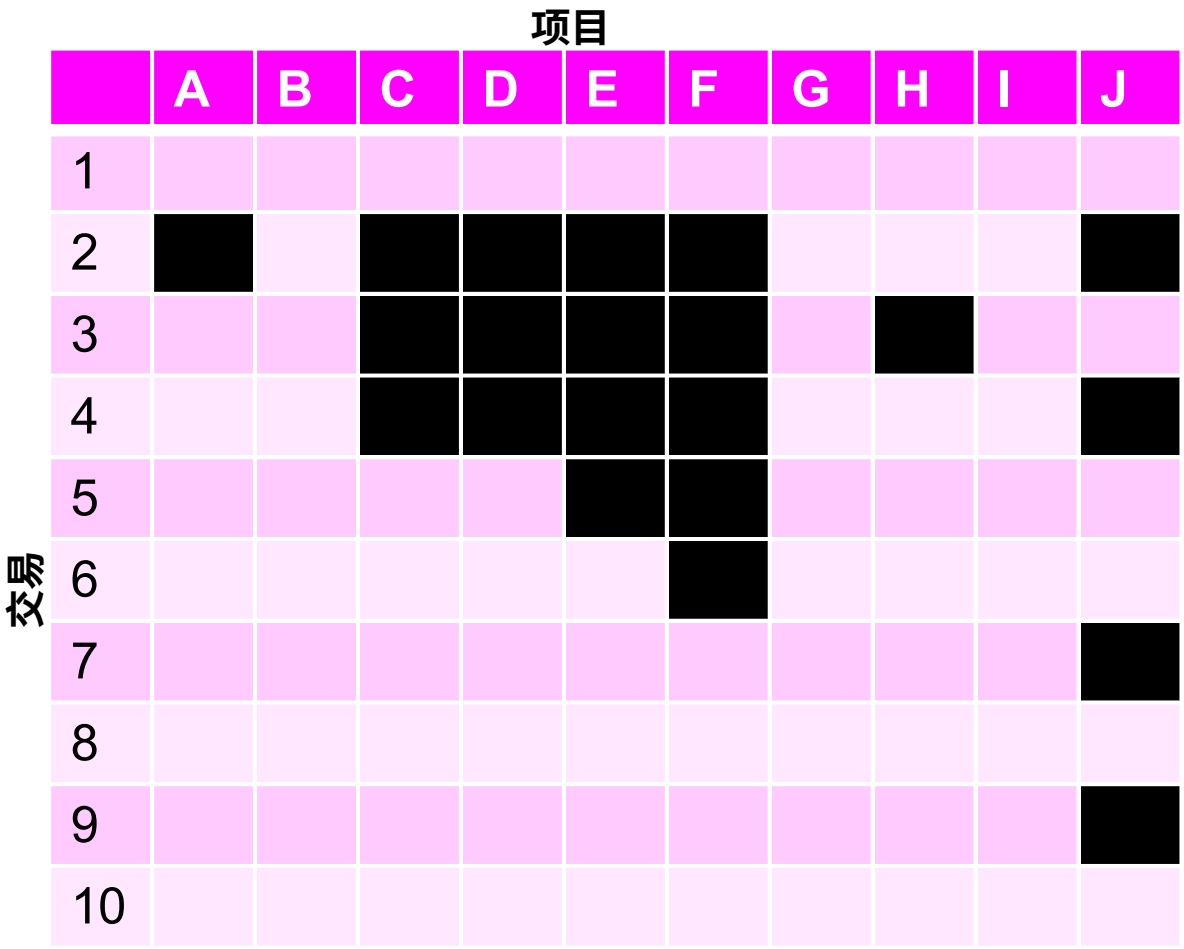
E

举例说明



支持阈值（按计数）：5 频项集：{F}

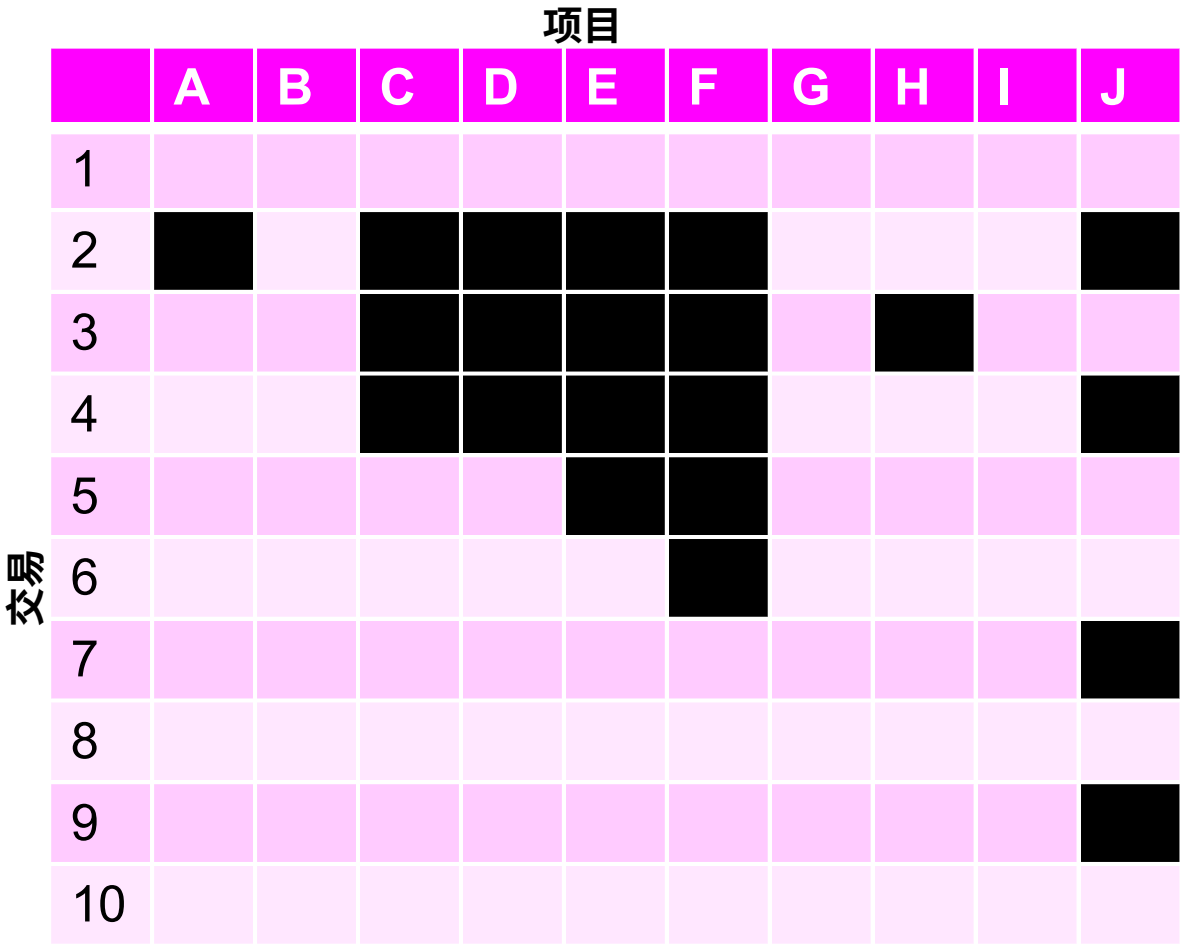
举例说明



支持阈值（按计数）：5 频项集：{F}

支持阈值（按计数）：4 频繁项集：？

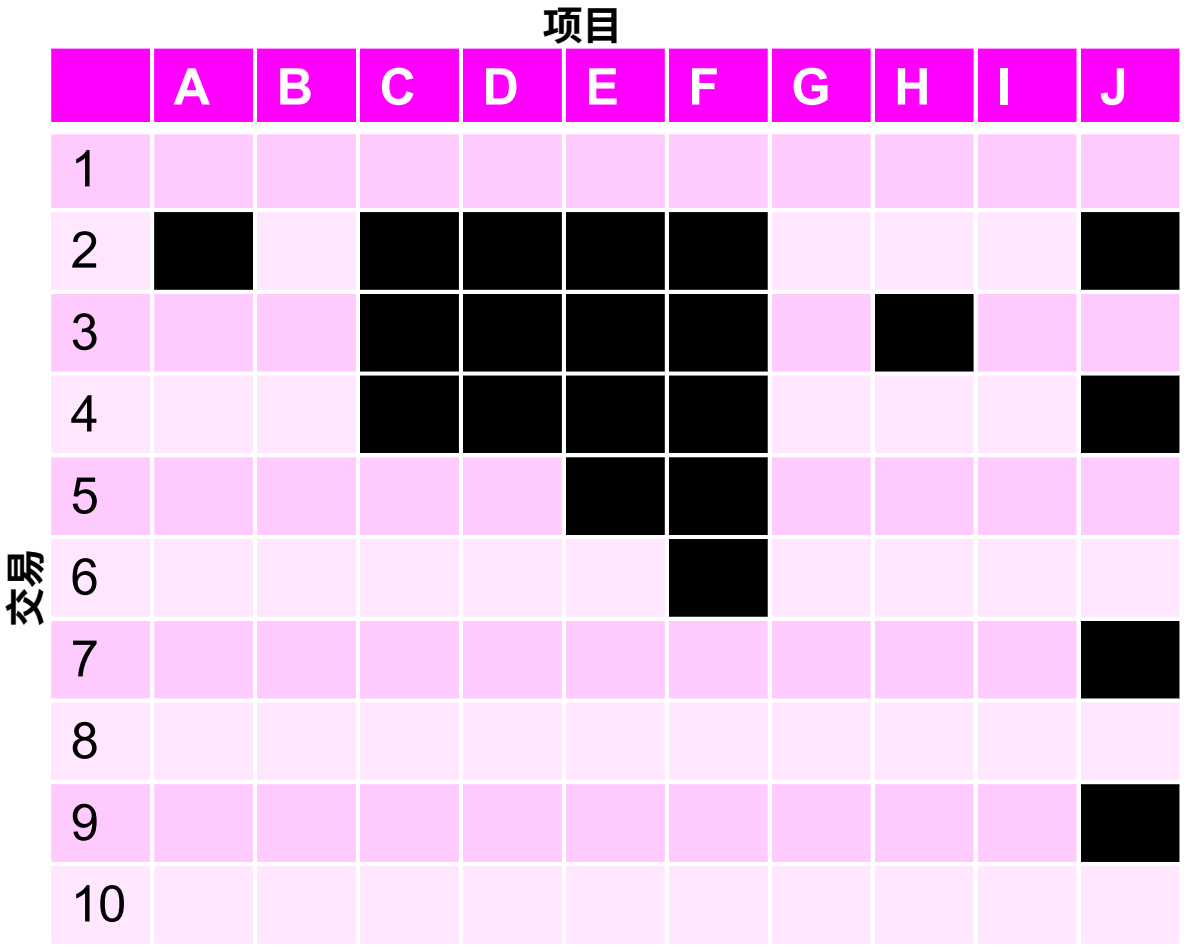
举例说明



支持阈值（按计数）：5 频项集：{F}

支持阈值（按计数）：4 频项目集：{e}, {f}, {e,f}, {j}

举例说明

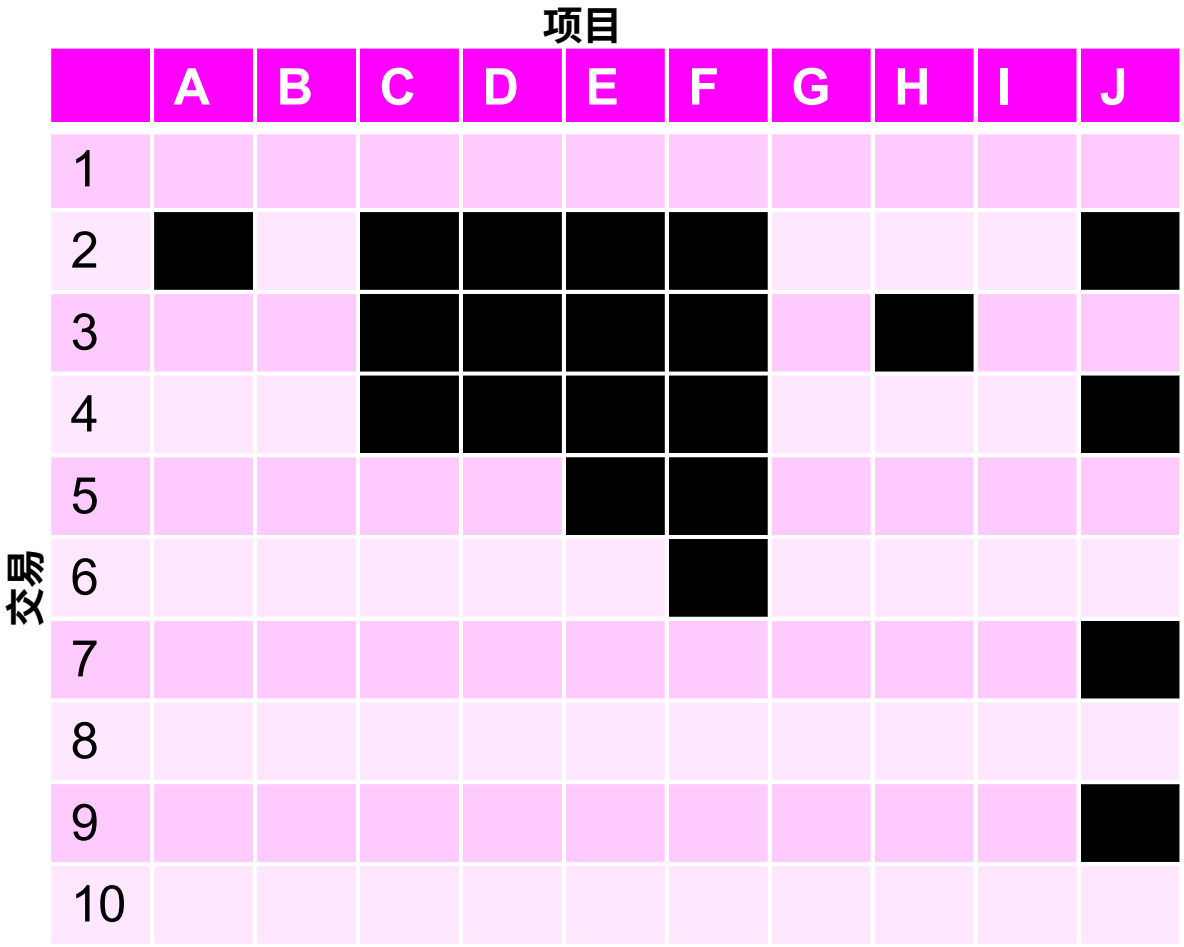


支持阈值（按计数）：5 频项集：{F}

支持阈值（按计数）：4 频项目集：{e}, {f}, {e,f}, {j}

支持阈值（按计数）：3 频繁项集：？

举例说明



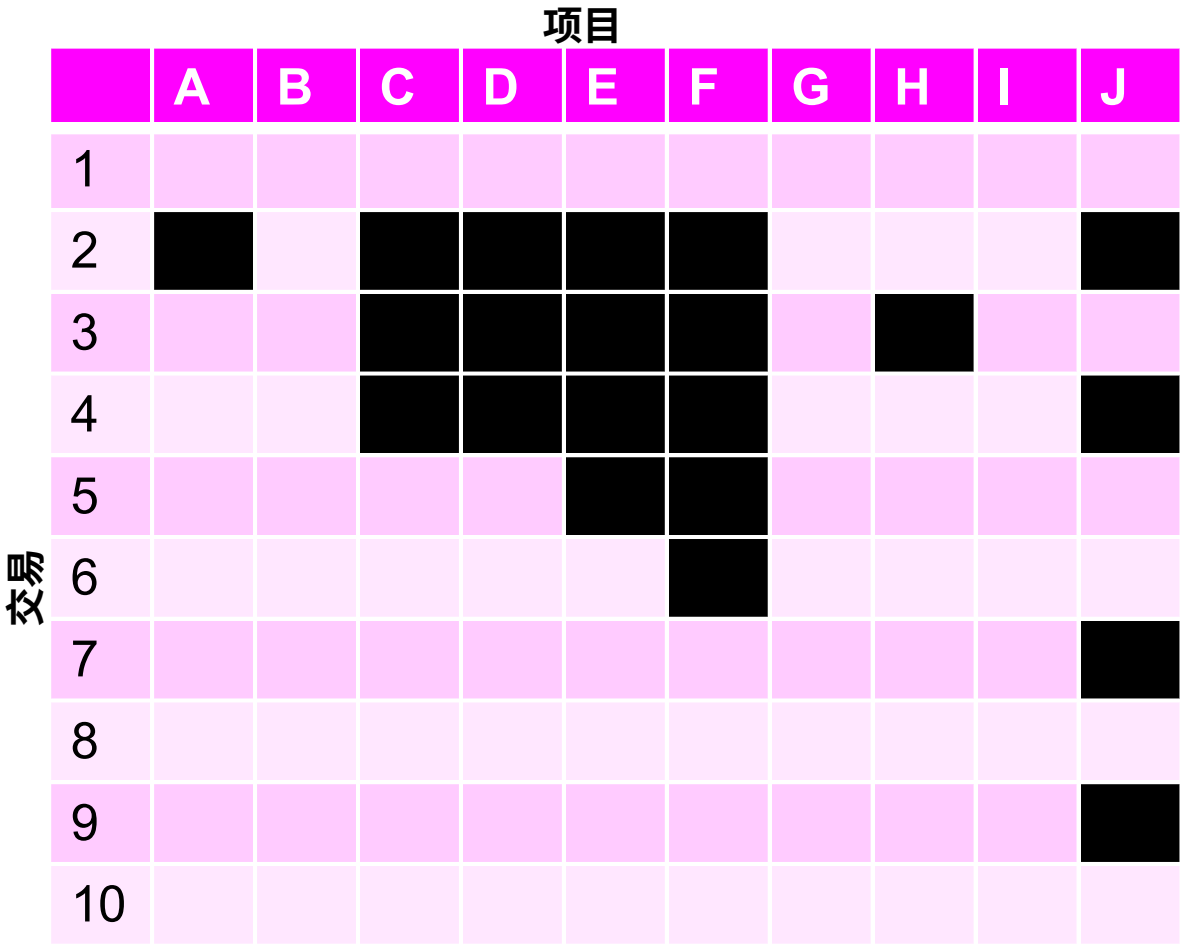
支持阈值（按计数）：5 频项集：{F}

支持阈值（按计数）：4 频项目集：{e}, {f}, {e,f}, {j}

支持阈值（按计数）：3 频繁项集：

{C,D,E,F}的所有子集+ {J}

举例说明



支持阈值（按计数）：5 频项集：{F}

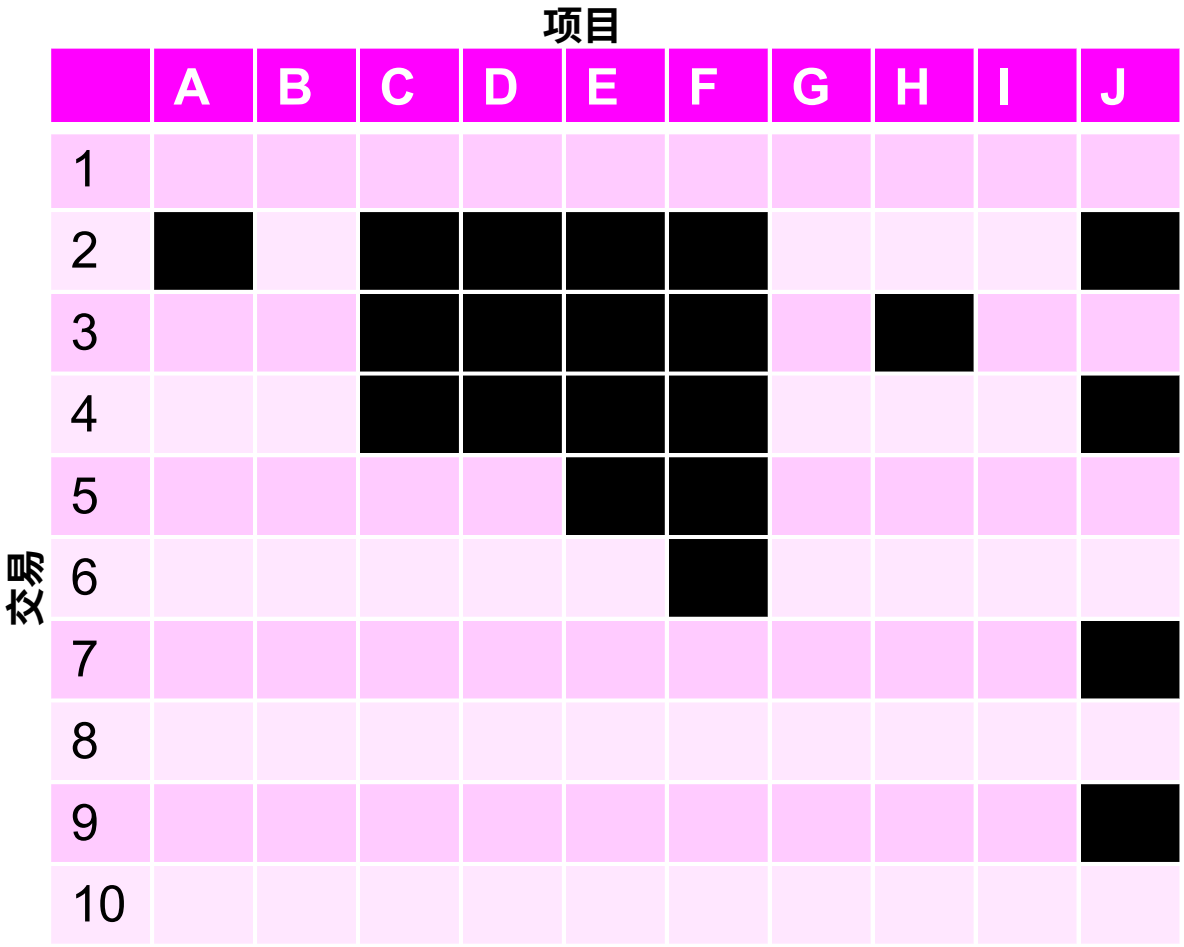
最大项目集：？

支持阈值（按计数）：4 频繁项集：{e}、{f}、{e,f}、{j}

最大项集：？

支持阈值（按计数）：3 频繁项集：{C,D,E,F}的所有子集+ {J}最大项集：？

举例说明

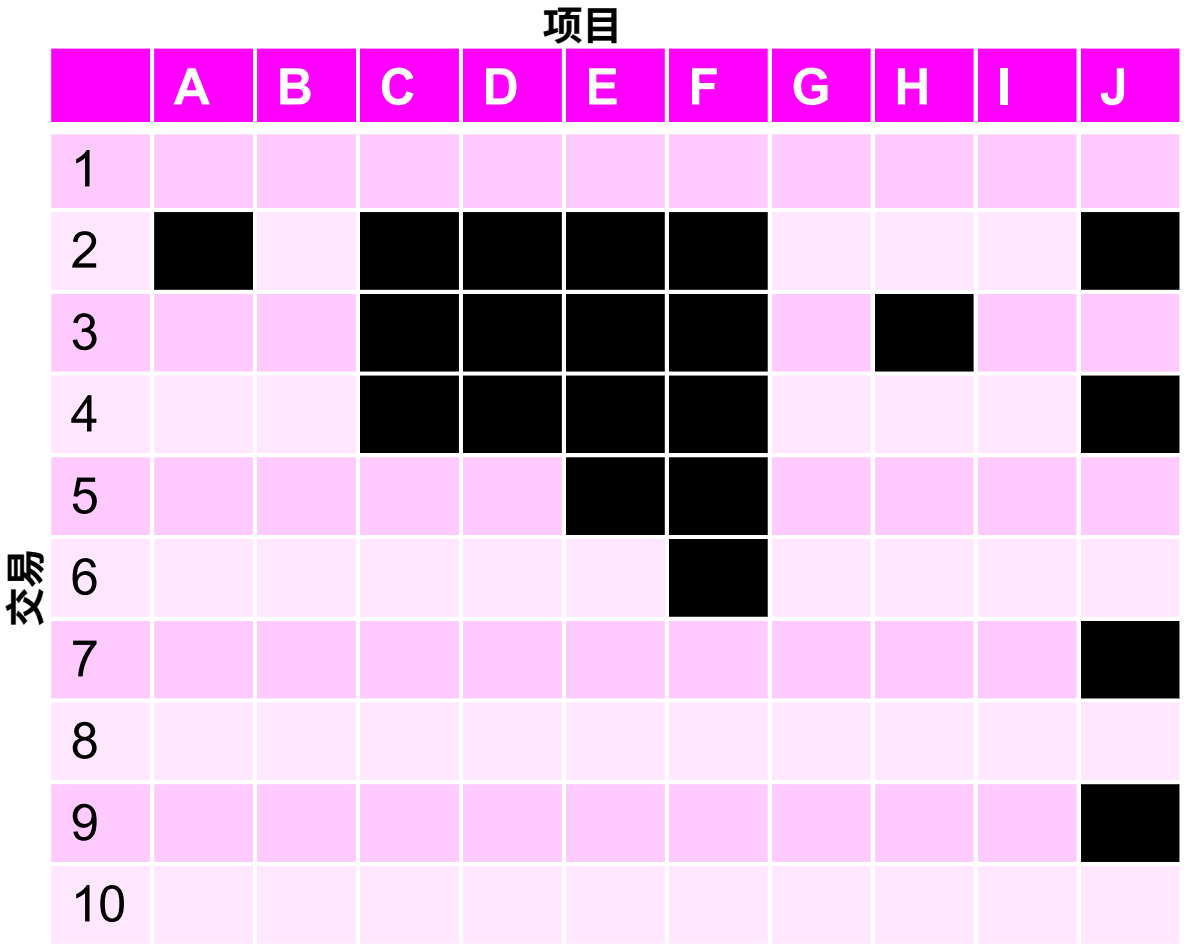


支持阈值（按计数）：5 频项集：{F}
最大项集：{F}

支持阈值（按计数）：4 频项目集：{e}, {f}, {e,f}, {j}最大项集：？

支持阈值（按计数）：3 频繁项集：{C,D,E,F}的所有子集+ {J}最大项集：？

举例说明



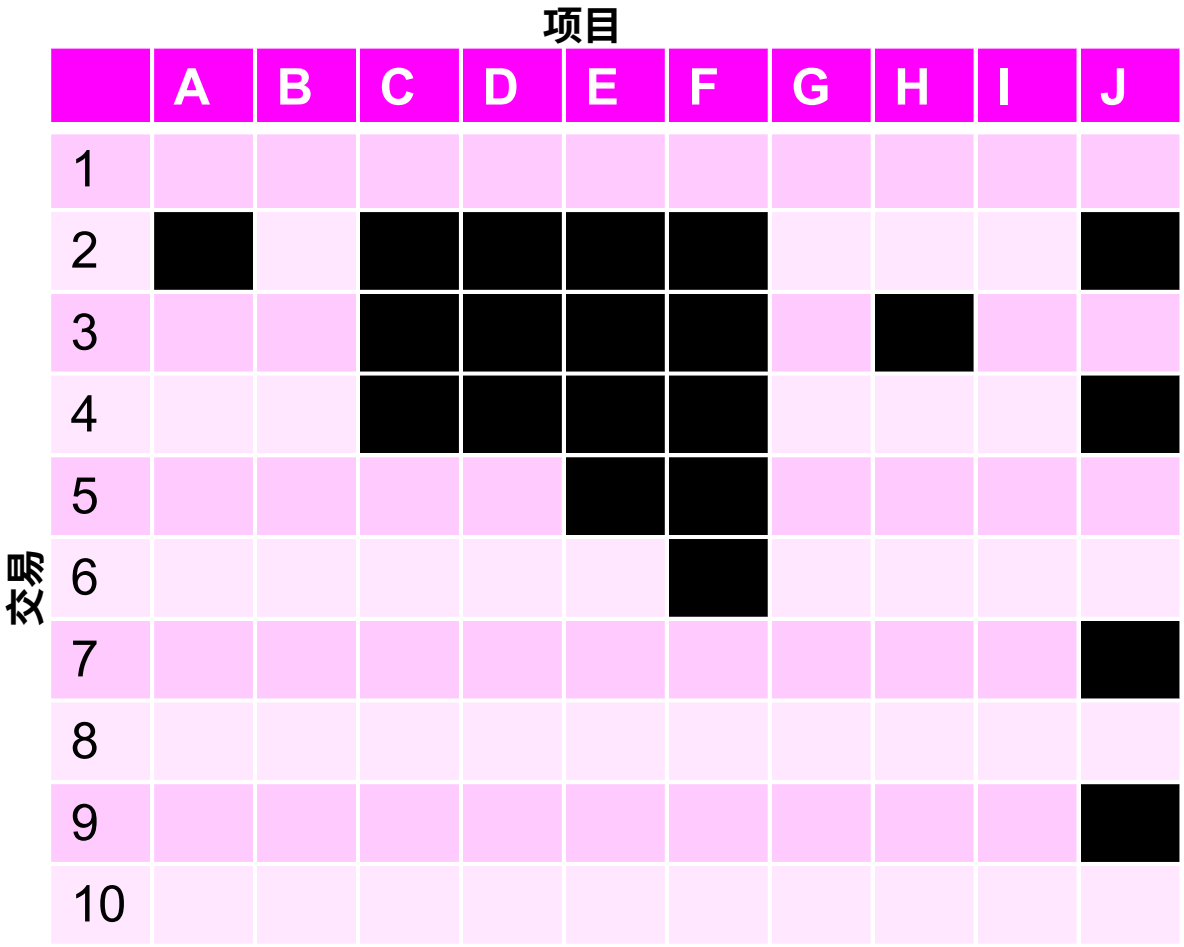
支持阈值（按计数）：5 频项集：{F}

最大项集：{F}

支持阈值（按计数）：4 频项目集：{e}, {f}, {e,f}, {j}最大项集：{E,F}、{J}

支持阈值（按计数）：3 频繁项集：{C,D,E,F}的所有子集+ {J}最大项集：？

举例说明



支持阈值（按计数）：5 频项集：{F}

最大项集：{F}

支持阈值（按计数）：4 频项目集：{e}, {f}, {e,f}, {j}最大项集：{E,F}、{J}

支持阈值（按计数）：3 频繁项集：{C,D,E,F}的所有子集+ {J}最大项集：

{C、D、E、F}、{J}。

最大常项集

- 最大频繁项集为频繁项集提供了一种紧凑的表示方法
  - 形成最小的项目集，从中可以导出所有频繁项目集
- 不包含其子集的支持信息
  - **Max-pattern 是一种有损压缩!**

非公开项目组

- 如果一个项集的直接超集都没有与该项集相同的支持度，那么该项集就是封闭的
- 如果至少有一个直接超集的支持数与 X 相同，则 X 不是封闭的。

TID	项目
1	{A、B}
2	{B、C、D}
3	{A、B、C、D}
4	{A、B、D}
5	{A、B、C、D}

项目集	支持
{A}	4
{B}	5
{C}	3
{D}	4
{A、B}	4
{A、C}	2
{A、D}	3
{B、C}	3
{B,D}	4
{C、D}	3

项目集	支持
{A、B、C}	2
{A、B、D}	3
{A、C、D}	2
{B、C、D}	2
{A、B、C、D}	2

这些数据中有哪些封闭项集?

TID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

非公开项目组

- 删除多余的关联规则

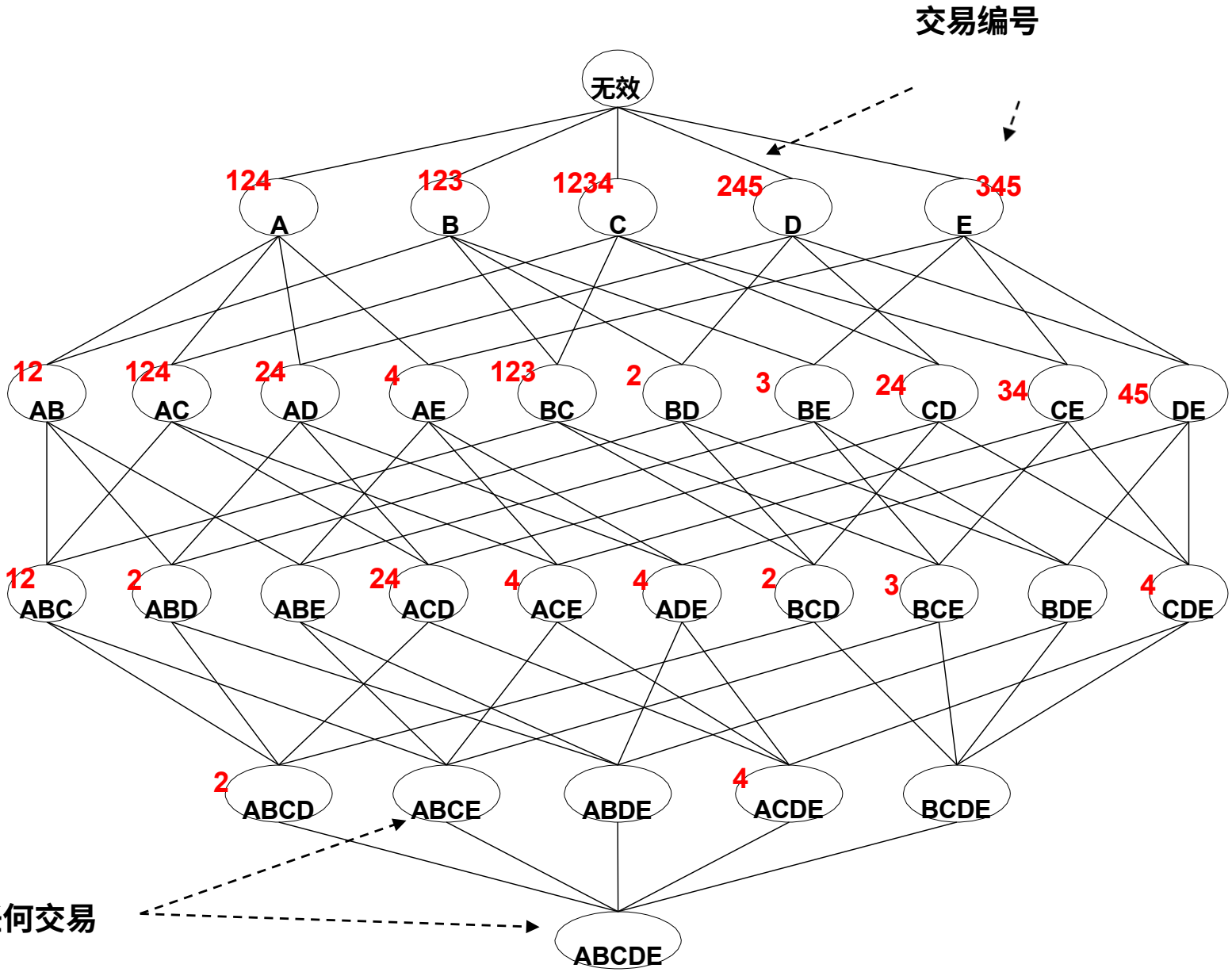
如果存在另一条规则 $X'' \rightarrow Y''$，其中 $X \subset X'$ 和 $Y \subset Y'$，使得两条规则的支持度和置信度相同，则关联规则 $X \rightarrow Y$ 是多余的。

{b,c}是封闭的，{b}->{d,e}是多余的，因为它与{b,c}->{d,e}具有相同的支持度和置信度。

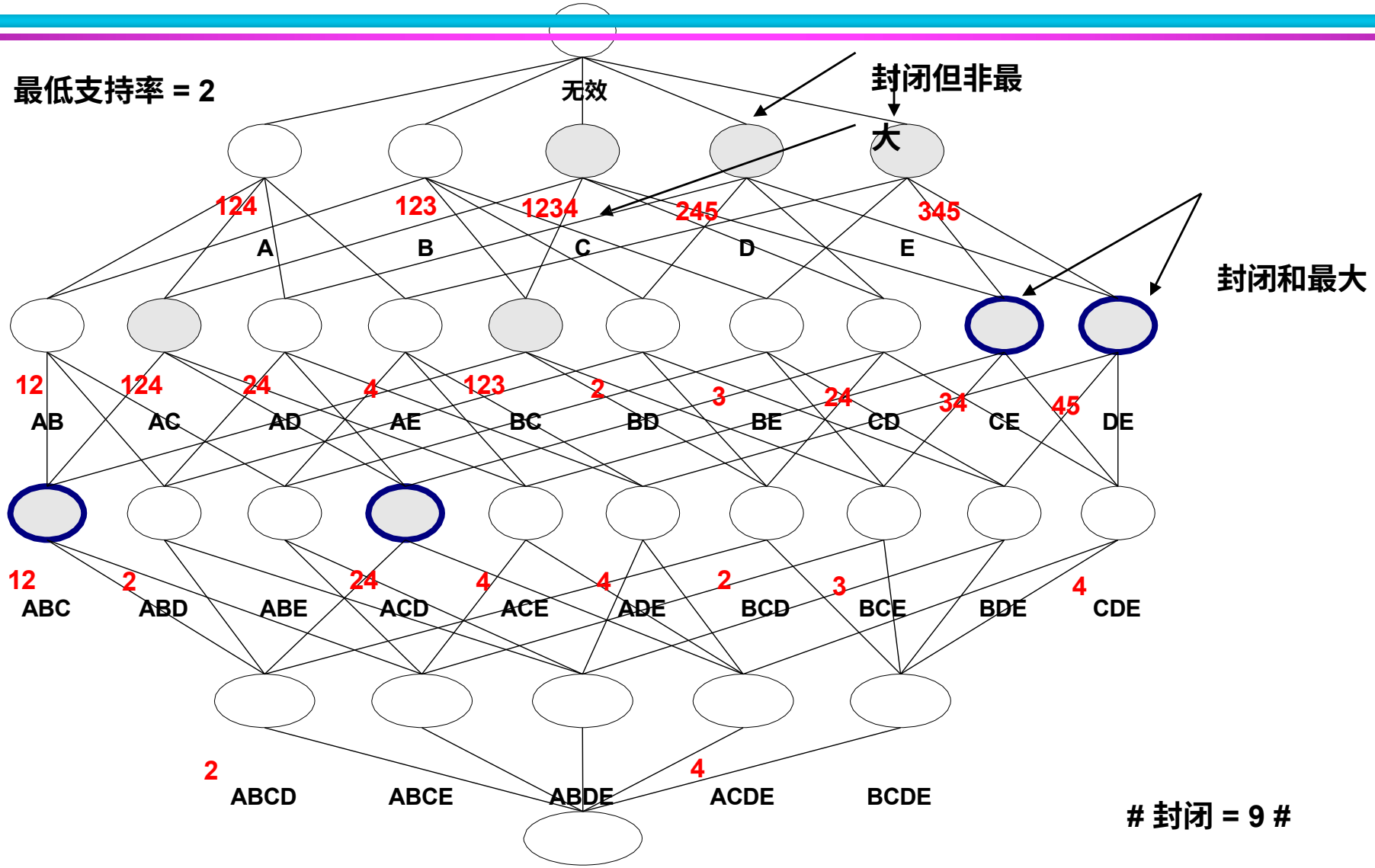
- **封闭模式是对频繁模式的无损压缩**

最大项集与封闭项集

TID	项目
1	美国广播公司
2	ABCD
3	BCE
4	ACDE
5	可选择丢弃



最大常项集与封闭常项集



封闭 = 9

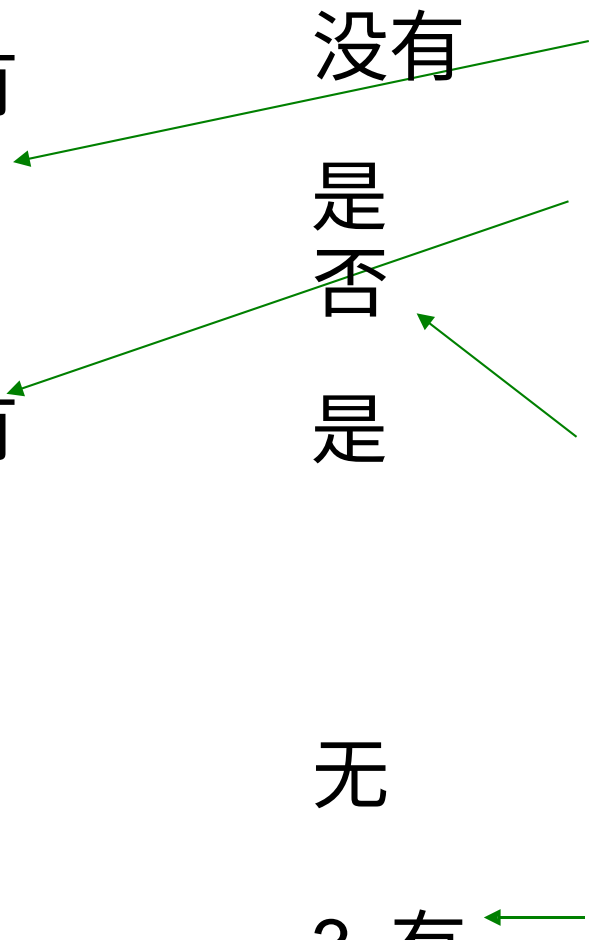
最大 = 4

ABCDE

示例：最大/闭合

支持		最大值(s=3)	关闭	
A	4	没有	没有	**ABC** 2
B	5	无	是	
C	3	没有	否	
AB	4	是	是	
AC	2	无	无	

不列颠哥伦比亚省 3 有



经常发生，但超集 BC 也经常发生。

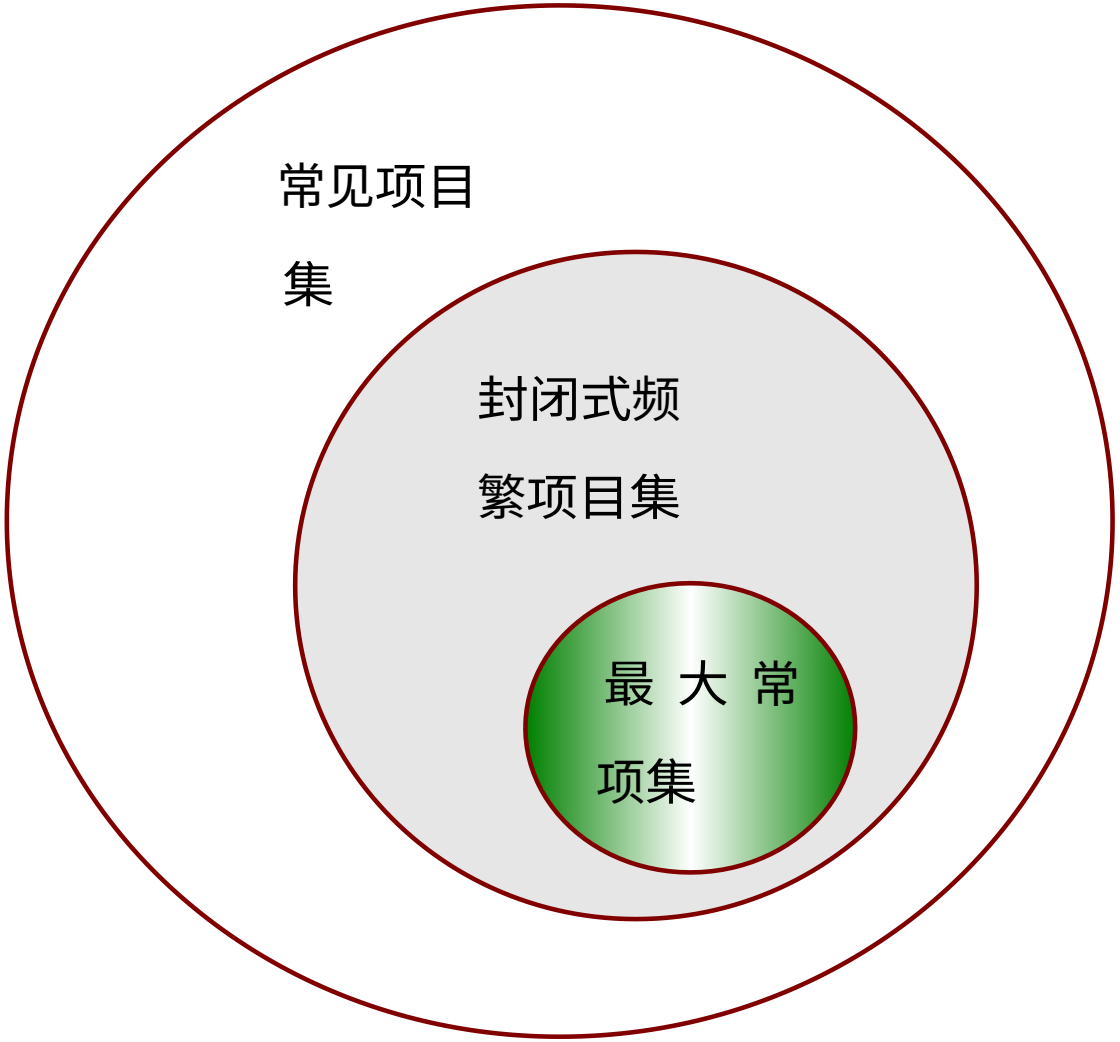
经常

它只是超集，ABC，而不是频率。

Superset BC 的计数相同。

其唯一的超级套餐 ABC 的数量较少。

最大项集与封闭项集



最大项集与封闭项集

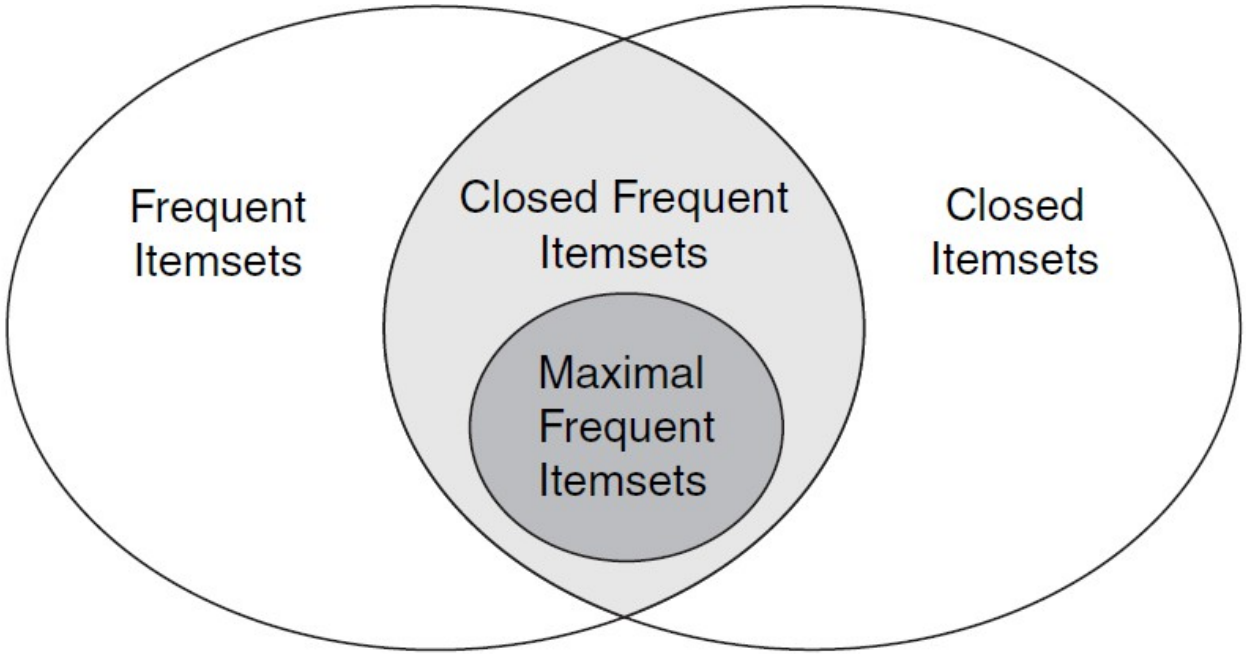
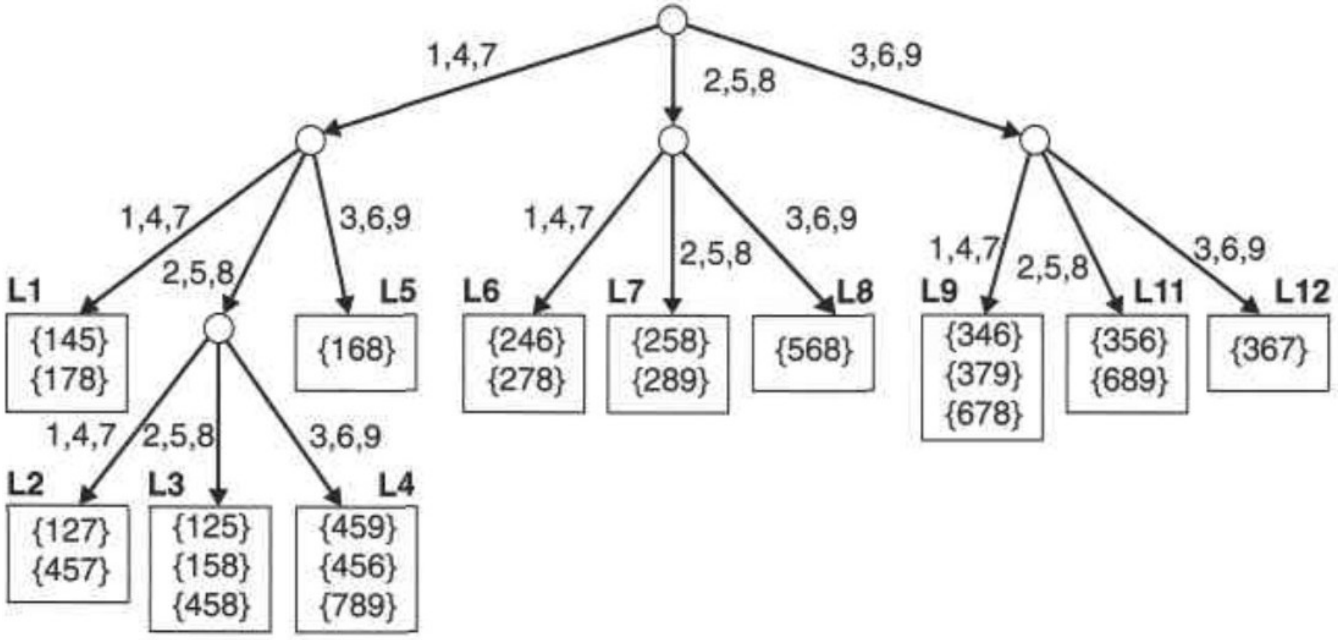


Figure 5.18. Relationships among frequent, closed, closed frequent, and maximal frequent itemsets.

小测

验

- Apriori 算法使用哈希树数据结构来有效计算候选项集的支持度。下面是候选 3 项集的散列树。



(a) 给定一个包含项目 {1,3,4,5,8}的事务，在查找该事务的候选项目时，将访问哈希树的哪个叶节点?

(b) 使用 (a) 部分中访问过的叶节点，确定事务 {1,3,4,5,8} 中包含的候选项集

小测

验

小测

验

找出 (a) 所有最大频繁项集；

(b) 所有封闭的频繁项集；

(c) 频繁但既不是最大项集也不是封闭项集。(s=0.3)

Transaction ID	Items Bought
1	$\{a, b, d, e\}$
2	$\{b, c, d\}$
3	$\{a, b, d, e\}$
4	$\{a, c, d, e\}$
5	$\{b, c, d, e\}$
6	$\{b, d, e\}$
7	$\{c, d\}$
8	$\{a, b, c\}$
9	$\{a, d, e\}$
10	$\{b, d\}$