

智能制造过程大数据技术

Big Data Technology in Intelligent Manufacturing Process

第四讲：频繁模式挖掘

Lecture 4: Frequent Pattern Mining

丁敏 dingmin@cug.edu.cn



中国地质大学(武汉) 自动化学院

School of Automation, China University of Geosciences

- 频繁模式挖掘的基本概念
- 频繁项集挖掘
- 关联规则挖掘
- 序列模式挖掘
- 本章小结



- 频繁模式挖掘的基本概念
- 频繁项集挖掘
- 关联规则挖掘
- 序列模式挖掘
- 本章小结



1 频繁模式挖掘的基本概念

4

➤ 背景与意义

报警日志1

报警发生时间	报警标签	设备单元
1/5 3:13:30	Tag06.PVHI	反应器
1/5 3:13:35	Tag44.PVHI	冷凝器
1/5 3:14:16	Tag05.PVLO	分离器
...

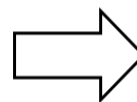
报警日志2

报警发生时间	报警标签	设备单元
1/6 12:14:58	Tag02.PVHI	反应器
1/6 12:14:59	Tag44.PVHI	冷凝器
1/6 12:15:01	Tag02.PVHI	反应器
...

...

报警日志 n

报警发生时间	报警标签	设备单元
1/30 0:3:56	Tag44.PVLO	反应器
1/30 0:4:12	Tag44.PVHI	冷凝器
1/30 0:4:45	Tag04.PVHI	分离器
1/30 0:4:55	Tag06.PVHI	冷凝器
...



{Tag06.PVHI, Tag44.PVHI}

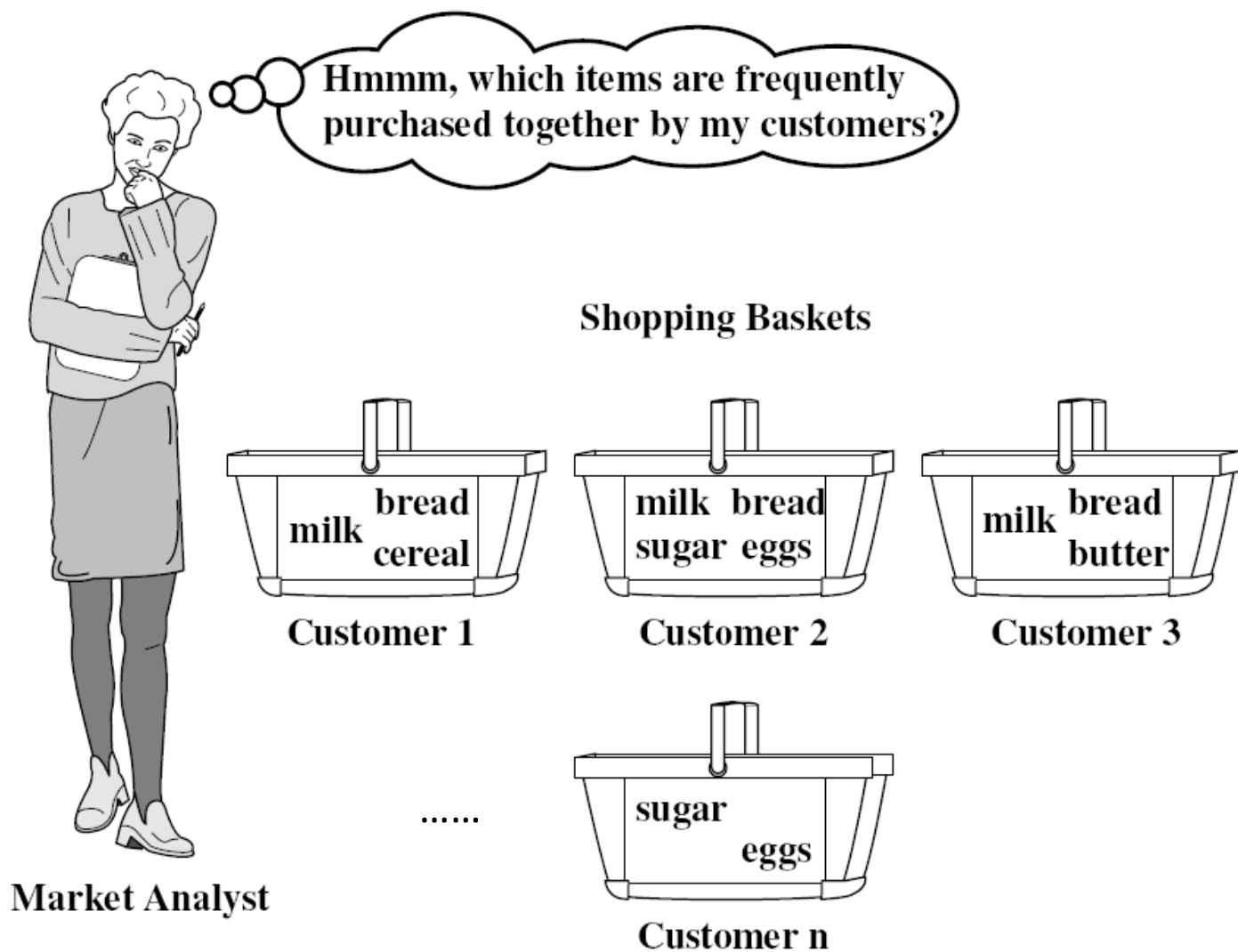
{Tag21.PVLL, Tag41.PVHH}

{Tag40.PVHI, Tag43.PVLL}

{Tag04.PVHI, Tag44.PVHI,
Tag02.PVHI}

报警事件数据库中的频繁项集

1 频繁模式挖掘的基本概念

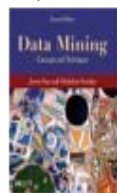


1 频繁模式挖掘的基本概念

6

Better Together

Buy this book with [Data Mining: Practical Machine Learning Tools and Techniques, Second Edition \(Morgan Kaufmann Series in Data Management Systems\)](#) by Ian H. Witten today!



+



Buy Together Today: \$97.72



Customers who bought this item also bought

[Data Mining: Practical Machine Learning Tools and Techniques, Second Edition \(Morgan Kaufmann Series in Data Management Systems\)](#) by Ian H. Witten

[Introduction to Data Mining, \(First Edition\)](#) by Pang-Ning Tan

[Data Preparation for Data Mining \(The Morgan Kaufmann Series in Data Management Systems\)](#) by Dorian Pyle

[Principles of Data Mining \(Adaptive Computation and Machine Learning\)](#) by David J. Hand

[Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management](#) by Michael J. A. Berry

► [Explore similar items](#) : [Books](#) (50)

1 频繁模式挖掘的基本概念

7

➤ **关联规则**反映一个事物与其他事物之间的**相互依存性和关联性**，如果两个或多个事物之间存在一定的关联关系，那么其中一个事物的发生能够预测与它相关的其他事物的发生

□ 频繁项集产生

其目标是发现**满足最小支持度阈值的所有项集**，这些项集称作频繁项集

□ 规则的产生

其目标是从上一步发现的频繁项集中提取所有**高置信度的规则**，这些规则称作**强规则**

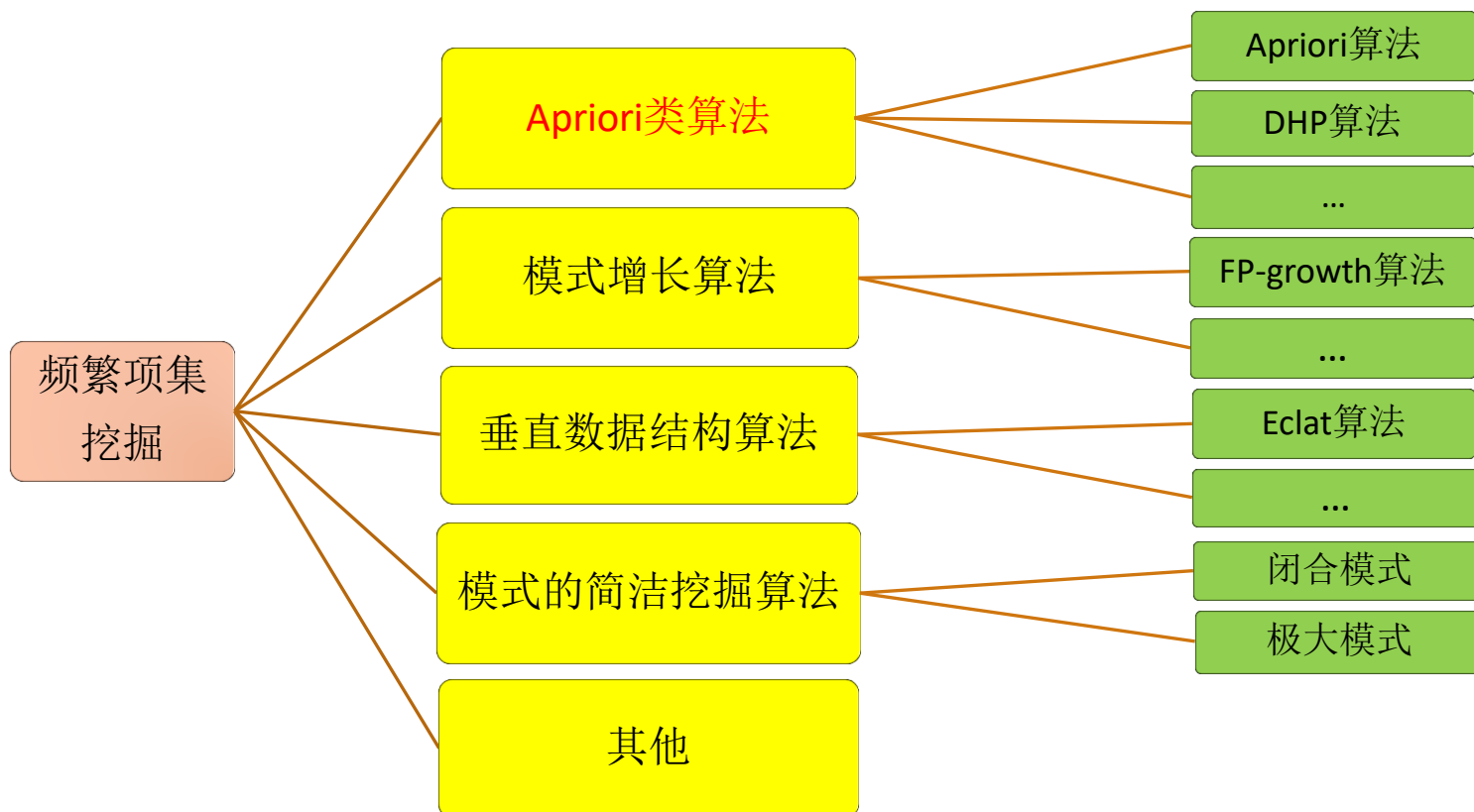
- 频繁模式挖掘的基本概念
- 频繁项集挖掘
- 关联规则挖掘
- 序列模式挖掘
- 本章小结



2 频繁项集挖掘

9

➤ 频繁项集挖掘



1 频繁模式挖掘的基本概念

10

➤ 项与项集、事务、事务数据库

- ❑ **项(item)**: 频繁模式的最小单元, 例如A和B
- ❑ **项集(itemset)**: 多个项组成的非空集合, 例如{A,B}
- ❑ **事务(transaction)**: 项组成的非空集合, 例如
 $T_3=\{B,C\}$, 每个事务对应唯一的标识符(TID)
- ❑ **事务数据库(database)**: 全体事务构成的集合, 也是频繁模式挖掘的对象, 记作

TID	事务
1	{A, B, E}
2	{B, D}
3	{B, C}
4	{A, B, D}
5	{A, C}

$$D=\{T_1, T_2, T_3, T_4, T_5\}$$

- ❑ **超集与子集**: 若项集X、Y满足 $X\subseteq Y$, 则Y是X的一个超集, X是Y的一个子集, 例如, T_4 是 T_2 的一个超集, T_2 是 T_4 的一个子集
- ❑ **真超集与真子集**: 若项集X中每个项都包含在Y中, Y中至少有一个项不包含在X中, 则Y是X的真超项集, X是Y的一个真子集

➤ 模式支持度与频繁模式

□ 支持度(support)：量化模式出现频次水平高低

- 支持度计数（绝对支持度）：模式出现在事务中的次数，例如 $\text{Sup}(A)_{\text{绝对}} = \sigma = |T_1, T_4, T_5| = 3$
- 支持度（相对支持度）：模式出现在事务中的次数与事务总数之比，例如 $\text{Sup}(A)_{\text{相对}} = 3/5 = 0.6$

□ 频繁模式：绝对支持度不小于最小支持度阈值（"minsup"）的模式，其中包含 k 项的频繁模式称为频繁 k -模式，其构成的集合记作 L_k 。

- 例如，假设minsup=2，项集{A,B}的绝对支持度为2，那么频繁模式{A,B}就是一个频繁2-项集

□ 候选集：用于获取频繁模式的模式集合，包含 k 项的候选集记作 C_k

TID	事务
1	{A, B, E}
2	{B, D}
3	{B, C}
4	{A, B, D}
5	{A, C}

B的相对支持度为

☒ A 0.8

☐ B 0.5

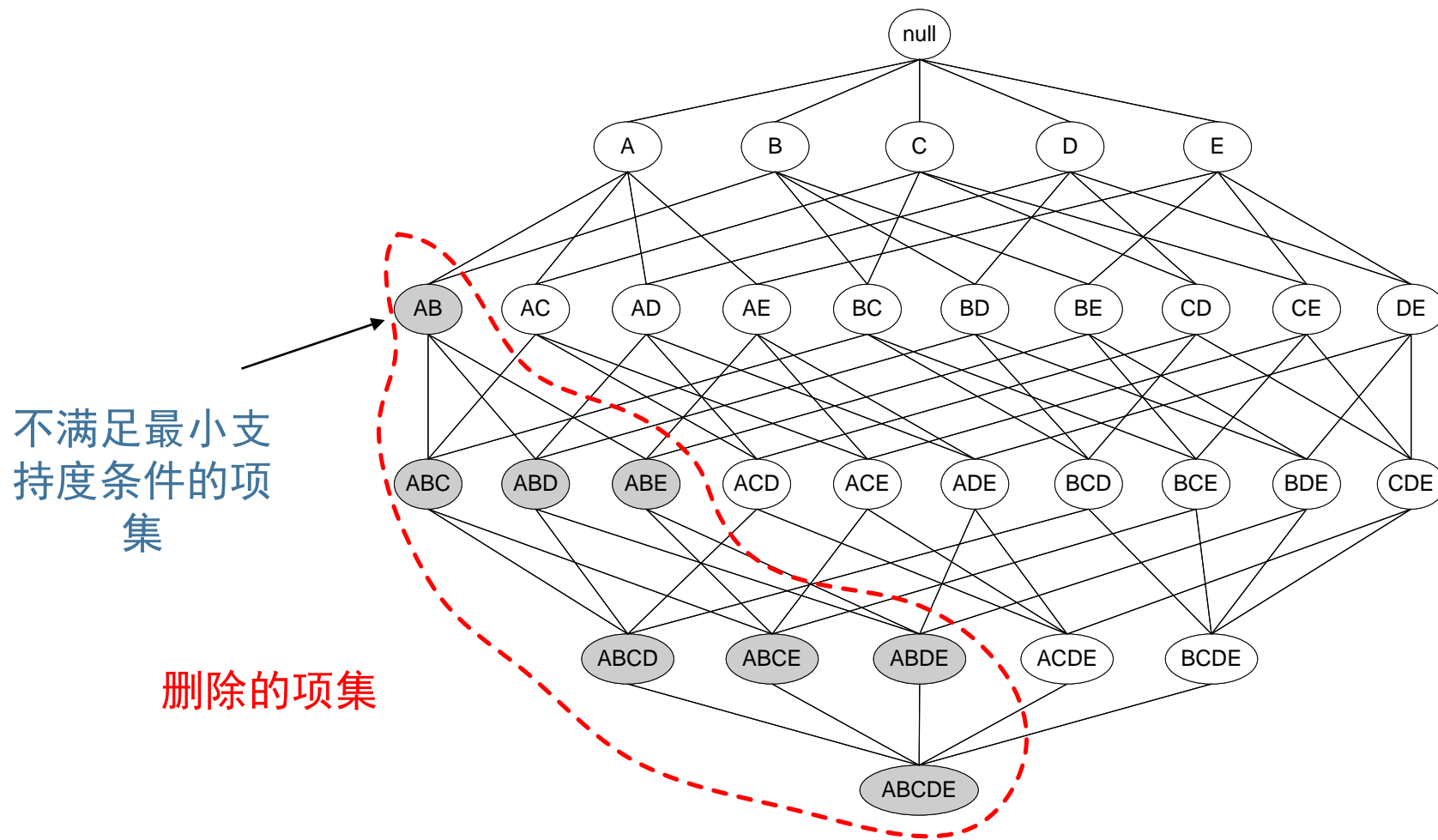
☐ C 4

☐ D 3

TID	事务
1	{A, B, E}
2	{B, D}
3	{B, C}
4	{A, B, D}
5	{A, C}

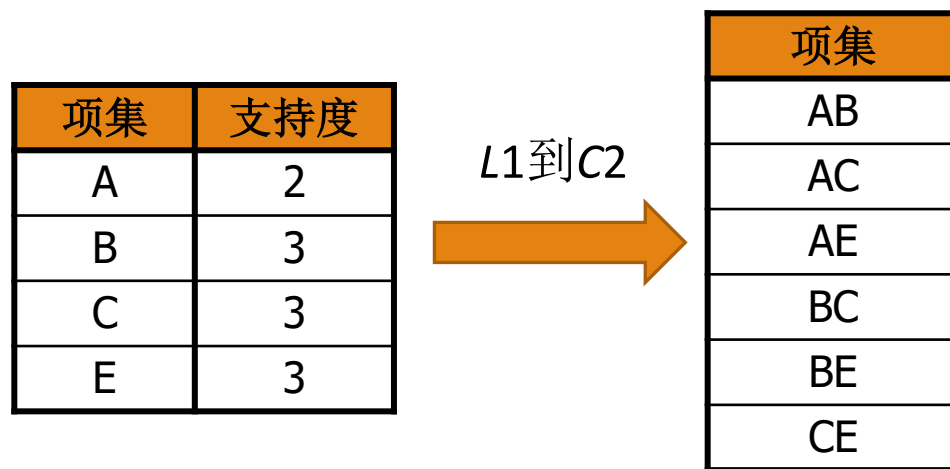
提交

- **先验原理**：如果一个模式是**频繁的**，则它的所有**子模式**一定也是**频繁的**；相反，如果一个模式不是频繁的，则它的所有**超模式**也一定**不是频繁的**



➤ Apriori算法基础

- ❑ 基本思想：从 D 中计算各项支持度，找出 L_1 ；从 $k=1$ 开始，利用 L_k 逐层连接生成候选项集 C_{k+1} ，扫描得到长度为 $k+1$ 的频繁项集 L_{k+1} ；以此类推，直到无法再生成频繁项集
- ❑ 基于先验原理，Apriori算法通过连接步和剪枝步，在每次迭代过程，由 L_k 得到 L_{k+1}
 - 连接步(由 L_k 得到 C_{k+1})：对 L_k 的项集按照字母顺序排序，项集连接生成 C_{k+1}



- 剪枝步(压缩候选集)：生成 C_{k+1} 时，根据先验定理去除 C_{k+1} 中不在 L_k 的 k 项子集，从而确定频繁项集构成的集合 L_k

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

minsup = 2

➤ Apriori算法流程

输入：数据库 D ; 支持度阈值 min_sup

输出：频繁项集的集合 L

□ 扫描数据库得到 L_1

□ 赋值 $k=1$ $L = L_1$

□ 迭代循环

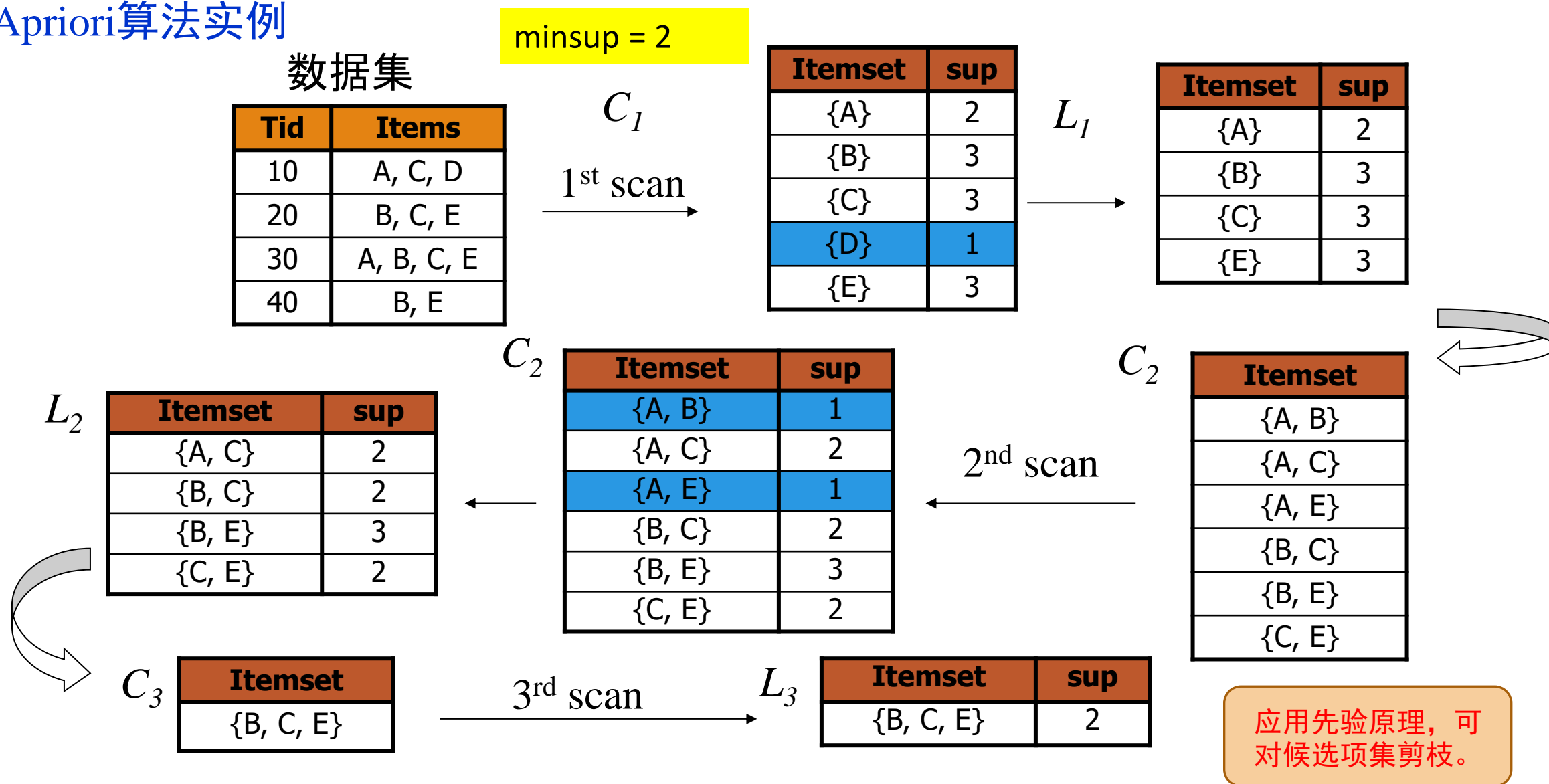
- 连接步骤：从 L_k 中生成候选集 C_{k+1}
- 剪枝步骤：根据先验性质压缩 C_{k+1} ，再扫描 D 去掉支持度小于 min_sup 项集，生成 L_{k+1}
- 更新变量： $k=k+1$ $L=L \cup L_k$
- 直到不能生成频繁集或候选集为止，循环结束

□ 返回 L

2 频繁项集挖掘

16

➤ Apriori算法实例



2 频繁项集挖掘

17

➤ Apriori算法实例

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



	频繁项集
L_1	{A, B, C, E}
L_2	{{A, C}, {B, C}, {B, E}, {C, E}}
L_3	{B, C, E}
$L=L_1 \cup L_2 \cup L_3$	{A, B, C, E, {A, C}, {B, C}, {B, E}, {C, E}, {B, C, E}}

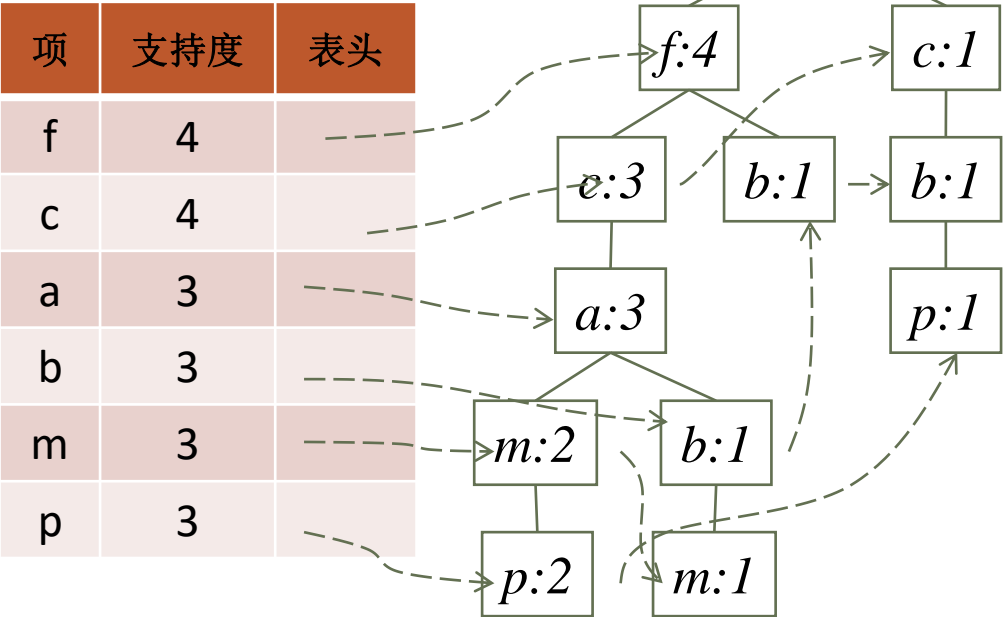
➤ FP-growth算法基础

- 背景：韩家炜等人，2000年
- 基本思想：
 - ✓ 只扫描数据库**两遍**，构造频繁模式树
 - ✓ **自底向上**递归产生频繁项集
 - ✓ **FP树**是一种输入数据的压缩表示，它通过逐个读入事务，把每个事务映射到FP树中的一条路径来构造

➤ FP-growth算法基础

- ❑ 基本思想：采用分治策略，以自底向上的方式进行搜索，并以频繁模式树（FP树）的形式产生频繁项集
- ❑ 项头表和FP树：用于存储D的数据机构，按支持度降序把频繁项放入表头并依次指向FP树中对应项的位置
- ❑ 条件模式基和条件模式树：对于一个频繁项，FP树中该频繁项的前缀路径集合称为条件模式基，根据条件模式基生成数据结构为条件模式树

minsup = 3



TID	Items in the Transaction	Ordered, frequent itemlist
100	{f, a, c, d, g, i, m, p}	f, c, a, m, p
200	{a, b, c, f, l, m, o}	f, c, a, b, m
300	{b, f, h, j, o, w}	f, b
400	{b, c, k, s, p}	c, b, p
500	{a, f, c, e, l, p, m, n}	f, c, a, m, p

项集m的条件模式基为
{ f, c, a: 2 }
{ f, c, a, b:1 }

条件模式树：
{ f, c, a: 3 }

产生的频繁项集：
{f,c,a.m}

➤ FP-growth算法流程

输入：数据库 D ; 支持度阈值 min_sup

输出：频繁项集的集合 L

- ❑ 建立项表头：扫描数据库得到 $L_1 = \{I_a \mid \text{Sup}(I_a) \geq \text{min_sup}, a = 1, 2, \dots, |L_1|\}$ ，然后将 L_1 放入项头表，并按照支持度降序排列
- ❑ 建立FP树：以“null”为根节点，对数据库中每个事务创建分枝，沿共同前缀的每个结点计数增加1，为前缀之后的项创建节点和链接
- ❑ 赋值： $k=1 \quad L=L_1$
- ❑ 迭代循环
 - 从项头表的底部项依次向上找到 I_k 的条件模式基以及对应的条件模式树，挖掘得到频繁项集 L_K
 - 更新变量： $k=k+1 \quad L=L \cup L_K$
 - 直到不能生成频繁集或候选集为止，循环结束
- ❑ 返回 L

FP-growth算法只需要扫描数据集两次，且不产生候选集，有效减少计算过程

➤ FP-growth算法实例

□ 步骤1：扫描数据库中对各项计数，按支持度递减降序对频繁项排序

设：minsup=2

$L = \{\{B:3\}, \{C:3\}, \{E:3\}, \{A:2\}\}$

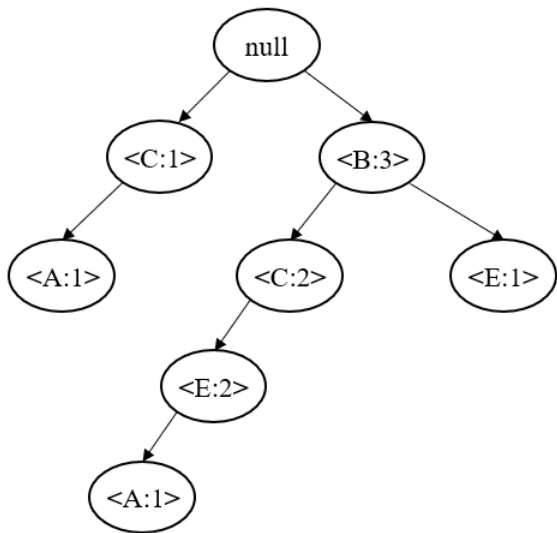
TID	事务
1	{A, C, D}
2	{B, C, E}
3	{A, B, C, E}
4	{B, E}

去除非频繁项
→

TID	事务
1	{C, A}
2	{B, C, E}
3	{B, C, E, A}
4	{B, E}

➤ FP-growth算法实例

TID	事务
1	$\{C, A\}$
2	$\{B, C, E\}$
3	$\{B, C, E, A\}$
4	$\{B, E\}$

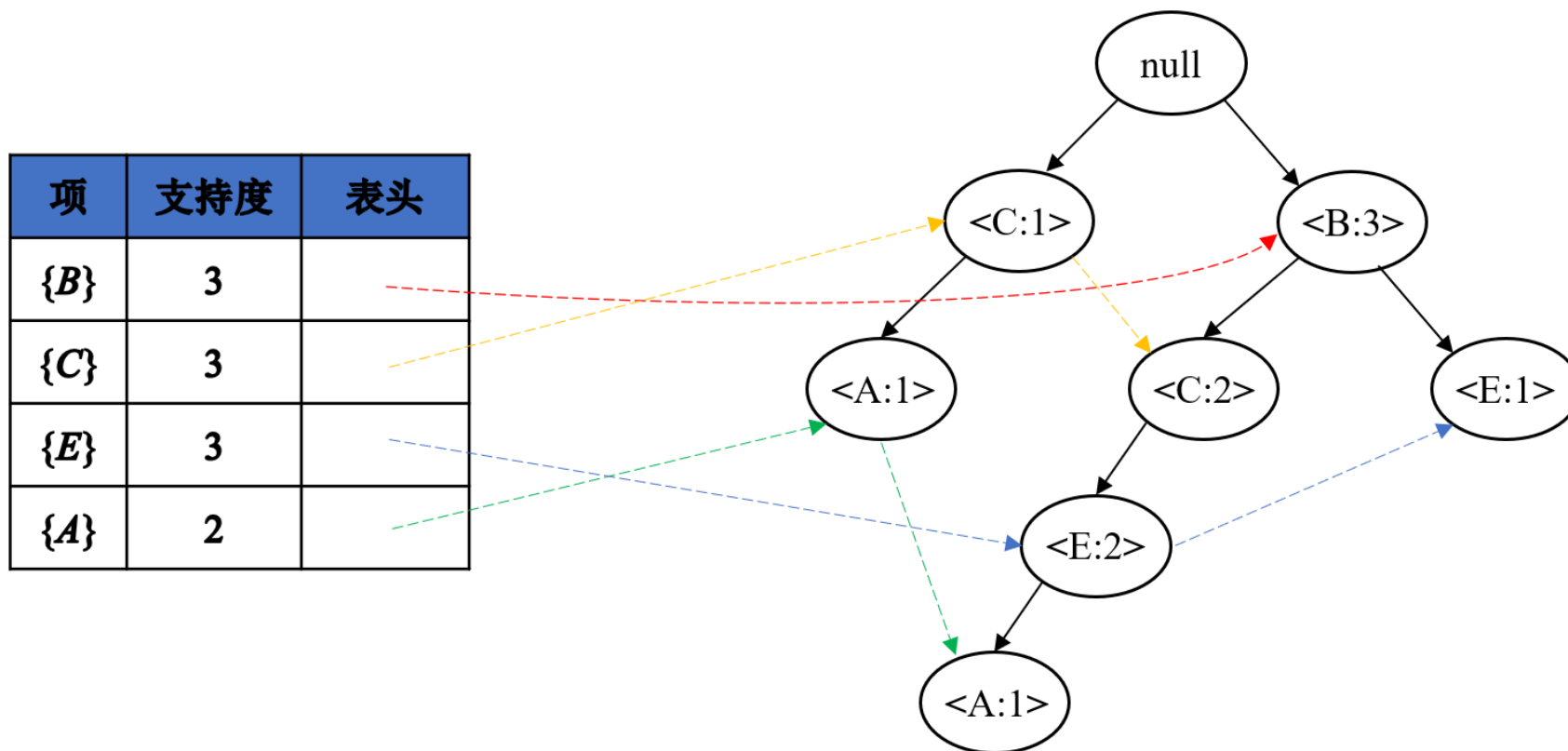


□ 步骤2：更新后数据库的事务 $T_1 = \{C, A\}$ 包含两个项，故构造树的第一个分支包含两个结点，即

- 更新后数据库的事务 $T_1 = \{C, A\}$ 包含两个项，故构造树的第一个分支包含两个结点，即 $\langle C:1 \rangle$ 、 $\langle A:1 \rangle$
- 添加事务 $T_2 = \{B, C, E\}$ ，包含三个结点： $\langle B:1 \rangle$ 、 $\langle C:1 \rangle$ 、 $\langle E:1 \rangle$
- 事务 $T_3 = \{B, C, E, A\}$ ，该分支包含四个结点： $\langle B:1 \rangle$ 、 $\langle C:1 \rangle$ 、 $\langle E:1 \rangle$ 、 $\langle A:1 \rangle$ ，
- 事务 $T_4 = \{B, E\}$ ，该分支包含2个结点： $\langle B:1 \rangle$ 、 $\langle E:1 \rangle$

➤ FP-growth算法实例

- 步骤3：按支持度递减排序创建项头表，通过观察可知B在树中的第二分支，C和A在树中的第一分支和第二分支，E在树中的第二分支和第三分支



➤ FP-growth算法实例

□ 步骤4：按支持度从低到高构造各项的条件模式基，依次构造条件FP树，通过条件FP树与项组合得到频繁项集

- 获取条件模式基的方法是对项头表中每一个项，都沿着FP树向上回溯并得到该项对应的所有前缀路径。以A为例，条件模式基为 $\{\{C:1\},\{B,C,E:1\}\}$
- 根据条件模式基创建条件FP树，方法是对照FP树仅保留不低于minsup的公共路径，此时应将 $\{C:1\}$ 删除，得到条件FP树为 $\langle B,C,E:2 \rangle$
- 将这些频繁项FP树中的元素与项A组合为更大的频繁项，直到条件FP树没有元素为止，得到对应的频繁模式为 $\{B,C,E:2\}$

项	条件模式基	条件FP树	产生的频繁模式
{A}	$\{\{C:1\},\{B,C,E:1\}\}$	$\langle C:2 \rangle$	$\{C,A:2\}$
{E}	$\{\{B,C:2\},\{B:1\}\}$	$\langle B,C:2 \rangle$	$\{B,C,E:2\}$
{C}	$\{\{B:2\}\}$	$\langle B:2 \rangle$	$\{B,C:2\}$

➤ Apriori和FP-growth算法比较

□ 减少候选项集的数量

- 先验原理：Apriori

□ 减少比较的次数

- 使用更高的数据结构，或存储候选项集或压缩数据集，减少比较次数（FP-Growth）

- 支持度阈值高时，Apriori产生的候选项集少，2种算法时间相当；反之，FP-Growth快

- 不产生候选集，遍历数据库2次

下面说法正确的是

- ☒ A Apriori算法本质上是一种 “宽度优先搜索策略”
- ☐ B Apriori算法本质上是一种 “深度优先搜索策略”
- ☐ C FP-Growth算法本质上是一种 “宽度优先搜索策略”
- ☒ D FP-Growth算法本质上是一种 “深度优先搜索策略”

提交

➤ 垂直结构结构算法

- ❑ Apriori算法和FP-growth算法的对象都是TID-项集格式的数据库，称为**水平数据格式**
- ❑ 数据库也可以用项-TID集格式表示，称为**垂直数据格式**
- ❑ 当水平结构的项目集数量较大时，可以转为垂直结构加快搜索速度

数据水平结构	Tid	项集
	10	a, c, d, e
	20	a, b, e
	30	b, c, e

数据垂直结构

Item	列表
a	10, 20
b	20, 30
c	10, 30
d	10
e	10, 20, 30

➤ 使用垂直数据结构挖掘频繁项集

- ❑ 首先将数据集转换为垂直数据结构
- ❑ 从 $k=1$ ，对频繁 k -项集执行连接步骤，构造 $(k+1)$ -项集
- ❑ 对 $(k+1)$ -项集对应的TID集进行支持度检测确定 $(k+1)$ -项集；重复该过程，每次 $k+1$
- ❑ 直到无法再生成频繁项集，返回所有频繁项集

2 频繁项集挖掘

28

➤ 使用垂直数据结构挖掘频繁项集

设: minsup=2

TID	事务
1	{A, C, D}
2	{B, C, E}
3	{A, B, C, E}
4	{B, E}



项集	TID集
{A}	{T ₁ , T ₃ }
{B}	{T ₂ , T ₃ , T ₄ }
{C}	{T ₁ , T ₂ , T ₃ }
{D}	{T ₁ }
{E}	{T ₂ , T ₃ , T ₄ }



项集	TID集
{B, C, E}	{T ₂ , T ₃ }



项集	TID集
{A, C}	{T ₁ , T ₃ }
{B, C}	{T ₂ , T ₃ }
{B, E}	{T ₂ , T ₃ , T ₄ }
{C, E}	{T ₂ , T ₃ }

➤ 模式压缩

□ **模式压缩**是用一部分模式作为整体的表示，同时保证信息量尽可能少的丢失，提升模式质量。

- 极大项集

频繁项集 X 在数据集 \mathbb{D} 中不存在真超集 Y ，则 X 是一个极大项集

- 闭合项集

频繁项集 X 在数据集 \mathbb{D} 中不存在满足 $\text{Sup}(Y) = \text{Sup}(X)$ 的真超集 Y ，则 X 是一个闭合项集

极大项集是闭合项集的充分不必要条件，因此一个极大项集必定是一个闭合项集。

➤ 模式压缩

ID	频繁项集	支持度
P_1	$\{B, C, D, E\}$	35
P_2	$\{B, C, D, E, F\}$	30
P_3	$\{A, B, C, D, E, F\}$	30
P_4	$\{A, C, D, E, F\}$	30
P_5	$\{A, C, D, E, G\}$	38
P_6	$\{A, C, D, G\}$	52
P_7	$\{A, C, G\}$	52
P_8	$\{A, D, G\}$	52

- P_1 、 P_2 、 P_4 存在真超集 P_3 ， P_6 、 P_7 、 P_8 的存在真超集 P_5 ，而项集 P_3 和 P_5 不存在真超集—— P_3 和 P_5 是极大项集
- P_1 的支持度 $\text{Sup}(P_1)=35$ ，与其全部的真超集 P_2 、 P_3 、 P_4 的支持度均不相等（ $\text{Sup}(P_2)=\text{Sup}(P_3)=\text{Sup}(P_4)=30$ ）



ID	极大频繁项集	ID	闭合频繁项集
P_3	$\{A, B, C, D, E, F\}$	P_1	$\{B, C, D, E\}$
P_5	$\{A, C, D, E, G\}$	P_3	$\{A, B, C, D, E, F\}$
		P_5	$\{A, C, D, E, G\}$
		P_6	$\{A, C, D, G\}$

- 频繁模式挖掘的基本概念
- 频繁项集挖掘
- **关联规则挖掘**
- 序列模式挖掘
- 本章小结



➤ 关联规则挖掘问题

- 给定事务的集合，关联规则发现是指找出支持度大于等于minsup，并且置信度大于等于minconf的所有规则，minsup和minconf是对应的支持度阈值和置信度阈值

TID	事务
1	{A, B, C}
2	{A, C}
3	{A, D}
4	{B, E, F}

频繁模式	支持度
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

最小支持度minsup=50%

最小置信度minconf=50%

- ✓ 关联规则是形如 $X \rightarrow Y$ 的蕴含表达式，其中X和Y是不相交的项集。例如：规则 $A \rightarrow C$ ：

$$\text{sup}(\{A\} \cup \{C\}) = \frac{\sigma(A, C)}{|T|} = 50\%$$

$$\text{conf}(A \rightarrow C) = \text{sup}(\{A\} \cup \{C\}) / \text{sup}(\{A\}) = 66.6\%$$

➤ 关联规则的产生

- ❑ **关联规则(association rule)**: X 和 Y 是频繁项集且满足置信度阈值, 可以表示为 $X \rightarrow Y$ 的范式
- ❑ **置信度(confidence)**: 用于量化关联规则的可靠性, 可以用条件概率描述, 计算公式为

$$\text{Conf}(X \rightarrow Y) = P(Y | X) = \frac{\text{Sup}(X \cup Y)}{\text{Sup}(X)}$$

minsup = 3

minconf = 0.8

$$\text{Sup}(\{\text{Beer}, \text{Diaper}\}) = 3$$

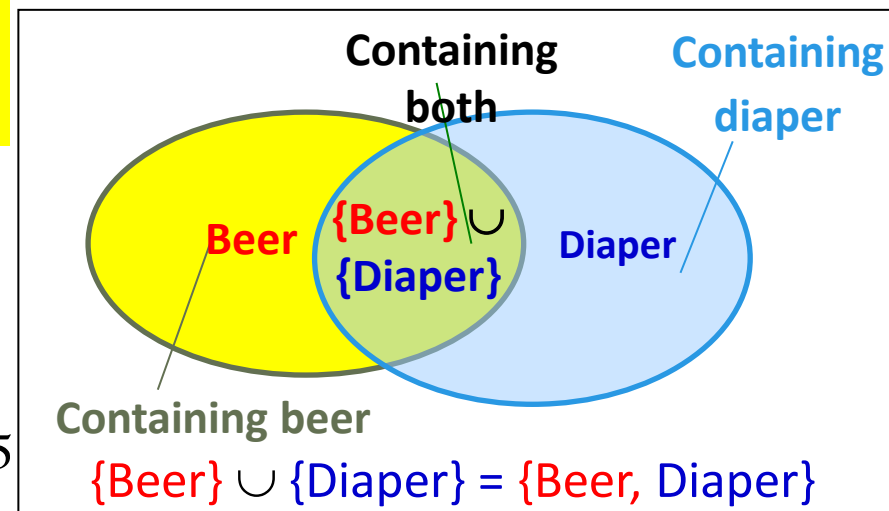
Beer \rightarrow Diaper
是强关联规则

$$\text{Conf}(\text{Beer} \rightarrow \text{Diaper}) = \frac{\text{Sup}(\{\text{Diaper}, \text{Beer}\})}{\text{Sup}(\text{Beer})} = 1$$

Diaper \rightarrow Beer
不是强关联规则

$$\text{Conf}(\text{Diaper} \rightarrow \text{Beer}) = \frac{\text{Sup}(\{\text{Diaper}, \text{Beer}\})}{\text{Sup}(\text{Diaper})} = 0.75$$

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



与频繁项集相比, 关联规则反映了一个事务与其它事务之间的相关性和相互依存性

➤ 产生关联规则


- ❑ 任务描述：给定频繁项集 X ，查找 X 的所有非空子集 $Y \subset X$ ，使得 $Y \rightarrow (X - Y)$ 的置信度超过最小置信度阈值 minconf
- ❑ 例：If $\{A, B, C\}$ is a frequent itemset, 候选规则如下：
 - ✓ $AB \rightarrow C, AC \rightarrow B, BC \rightarrow A, A \rightarrow BC, B \rightarrow AC, C \rightarrow AB$
- ❑ 如果 $|Y| = k$ ，那么会有 $2^k - 2$ 个候选关联规则（不包括 $Y \rightarrow \emptyset$ and $\emptyset \rightarrow Y$ ）


➤ 产生关联规则的先验原理


- 针对同一个频繁项集的关联规则，如果规则的后件满足子集关系，那么这些规则的置信度间满足反单调性

✓ eg. $Y = \{A, B, C, D\}$

✓ $c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$


$$c = \frac{\sigma(A, B, C, D)}{\sigma(A, B, C)}$$

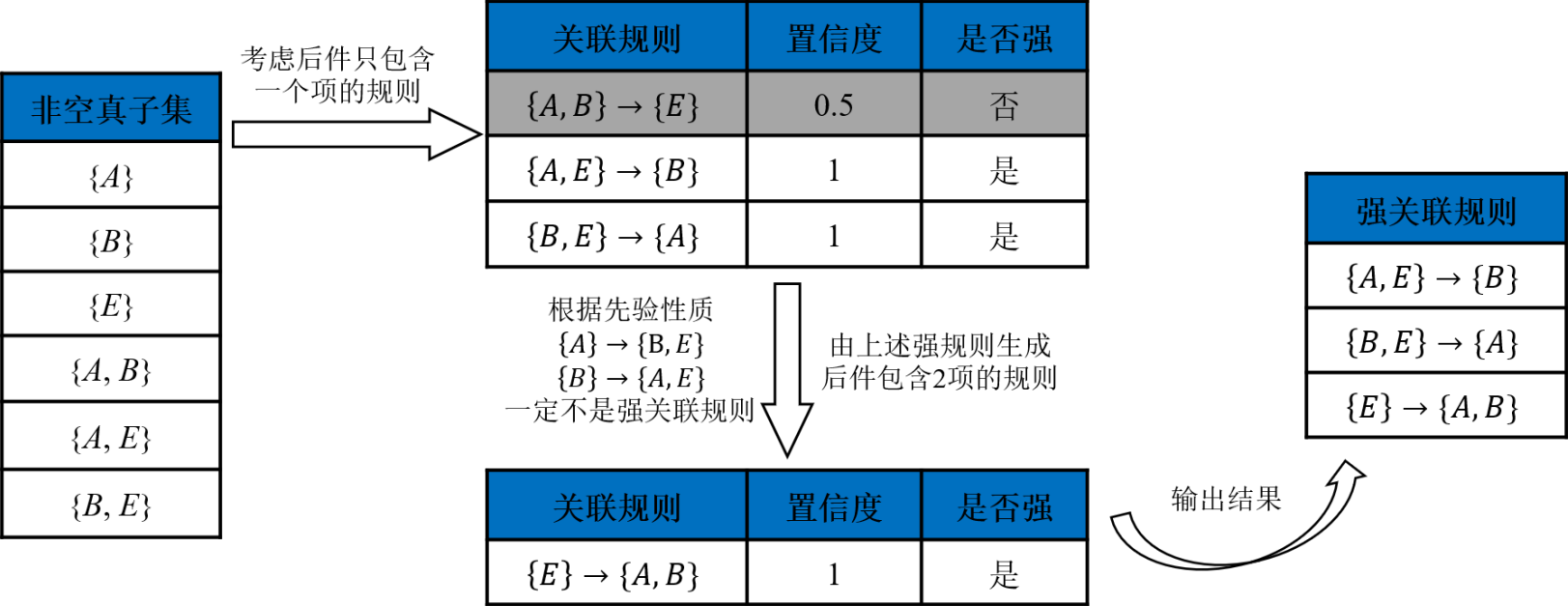

$$c = \frac{\sigma(A, B, C, D)}{\sigma(A, B)}$$


$$c = \frac{\sigma(A, B, C, D)}{\sigma(A)}$$

➤ 关联规则挖掘实例

TID	事务
1	{A, B, E}
2	{B, D}
3	{B, C}
4	{A, B, D}
5	{A, C}
6	{B, C}
7	{A, C}
8	{A, B, C, E}
9	{A, B, C}

- 频繁项集 $X = \{A, B, E\}$ ， X 的非空真子集是 $\{A\}$ 、 $\{B\}$ 、 $\{E\}$ 、 $\{A, B\}$ 、 $\{A, E\}$ 和 $\{B, E\}$ ，需要对频繁项集 X 和它的非空真子集进行强关联规则挖掘并列出的置信度（设置最小置信度阈值为 $\text{minconf}=70\%$ ）



只有 $\{A, E\} \rightarrow \{B\}$ ， $\{B, E\} \rightarrow \{A\}$ 以及 $\{E\} \rightarrow \{A, B\}$ 可以作为强关联规则

➤ 关联规则挖掘实例

关联规则	置信度	是否强关联规则
$\{A, B\} \rightarrow \{E\}$	0.5	否
$\{A, E\} \rightarrow \{B\}$	1	是
$\{B, E\} \rightarrow \{A\}$	1	是
$\{A\} \rightarrow \{B, E\}$	0.33	否
$\{B\} \rightarrow \{A, E\}$	0.29	否
$\{E\} \rightarrow \{A, B\}$	1	是

➤ 强关联规则挖掘过程

- ❑ 扫描数据库得到项集构成的集合 L
- ❑ 通过迭代以下两步，循环产生关联规则：
 - 第一步：对 L 中的每个项集 X ，产生 X 的所有非空真子集 Y
 - 第二步：对 X 的每个非空真子集 Y ，构造关联规则 $Y \rightarrow (X - Y)$ ，如果满足

$$\text{Conf}(Y \rightarrow X - Y) = \frac{\text{Sup}(X)}{\text{Sup}(Y)} \geq \text{min conf}$$

则输出强关联规则 $Y \rightarrow X - Y$

- ❑ 直到不能生成频繁集或候选集为止，循环结束

关联规则挖掘分为两步，可以看作是在频繁项集挖掘基础上进行的

➤ 关联规则的评估

- **提升度(lift)**: 提升度是一种简单的相关性度量, 首先用条件概率考察事件之间的独立性: 如果 $P(A \cup B) = P(A)P(B)$, 则项集A与B独立, 否则项集A、B具有一定相关性

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

- $\text{lift}(A, B) > 1$, 则A和B正相关, 且值越大正相关性越高, 即一个项集出现都可能导致另一个项集出现
- $\text{lift}(A, B) < 1$, 则A和B负相关, 且值越小负相关性越高, 即一个项集出现都可能导致另一个项集出现
- $\text{lift}(A, B) = 1$, 则A和B独立

	TH	\neg TH	Σ row
PL	4000	3500	7500
\neg PL	2000	500	2500
Σ col	6000	4000	10000

\neg TH表示不包含产品分离器温度过高的事务, \neg PL表示不包含产品分离器压力过小的事务

产品分离器温度过高的概率 $P(\text{TH}) = 0.6$

产品分离器压力过小的概率 $P(\text{PL}) = 0.75$

两者同时发生的概率 $P(\text{TH} \cup \text{PL}) = 0.4$

$$\text{lift}(\text{TH}, \text{PL}) = P(\text{TH} \cup \text{PL}) / (P(\text{TH})P(\text{PL})) = 0.4 / (0.75 \times 0.6) = 0.89 < 1$$

TH和PL存在负相关

➤ 关联规则的评估

- ❑ **卡方距离(χ^2)**: 卡方距离度量是观察频数与期望频数之间距离的一种度量指标, 也是假设成立与否的度量指标

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

- $\chi^2 = 0$, 则表明两个分布一致
- χ^2 越小, 则表明观察频数与期望频数越接近, 两者之间的差异越小
- χ^2 越大, 说明观察频数与期望频数差别越大, 两者之间的差异越大

	TH	¬TH	Σrow
PL	4000(4500)	3500(3000)	7500
¬PL	2000(1500)	500(1000)	2500
Σcol	6000	4000	10000

卡方距离的值大于卡方临界值, 并且 (TH,PL) 的观测值等于 4000, 小于期望值 4500, 因此 **TH与PL是负相关的**

$$\chi^2 = \frac{(4000 - 4500)^2}{4500} + \frac{(3500 - 3000)^2}{3000} + \frac{(2000 - 1500)^2}{1500} + \frac{(500 - 1000)^2}{1000} = 55.56$$

➤ 全置信度、最大置信度、Kulczynski和余弦

□ 全置信度

$$\text{allconf}(A, B) = \frac{\text{Sup}(A \cup B)}{\max\{\text{Sup}(A), \text{Sup}(B)\}}$$

- $\max\{\text{Sup}(A), \text{Sup}(B)\}$ 是 A 和 B 的最大支持度。因此， $\text{allconf}(A, B)$ 又称为两个与 A 和 B 相关的关联规则 “ $A \rightarrow B$ ” 和 “ $B \rightarrow A$ ” 的最小置信度

□ 最大置信度

$$\text{maxconf}(A, B) = \max\{P(A|B), P(B|A)\}$$

□ Kulczynski: 对两个置信度求平均值

$$\text{Kulc}(A, B) = \frac{1}{2} (P(A|B) + P(B|A))$$

- 频繁模式挖掘的基本概念
- 频繁项集挖掘
- 关联规则挖掘
- **序列模式挖掘**
- 本章小结



➤ 序列模式挖掘：针对带有时间属性的数据库进行频繁序列挖掘以发现某种规律

□ 序列(sequence)：由不同的元素按照一定顺序排列组成

$$s = \langle I_1, I_2, \dots, I_n \rangle$$

□ 序列数据库：由多条序列组成的数据库

$$S = \langle s_1, s_2, \dots, s_n \rangle$$

□ 子序列性质：对于两个序列 $s_1 = \langle q_1, q_2, \dots, q_m \rangle$ 与 $s_2 = \langle p_1, p_2, \dots, p_n \rangle$ ，如果存在整数 $1 \leq i_1 < i_2 < \dots < i_m \leq n$ ，满足以下约束条件 $q_1 \subseteq p_{i_1}, q_2 \subseteq p_{i_2}, \dots, q_m \subseteq p_{i_m}$ ，则序列 s_2 包含序列 s_1 ，称 s_1 为 s_2 的子序列

□ 频繁序列模式：在序列数据库中，若一条序列在库中出现的次数超过了设定的最小支持度阈值时，就称这个序列是一个频繁序列模式

序列号	序列
1	<abc>
2	<abcd>
3	<dabc>
4	<da>

minsup = 3

则有频繁序列

$$S_1 = \langle a \ b \ c \rangle$$

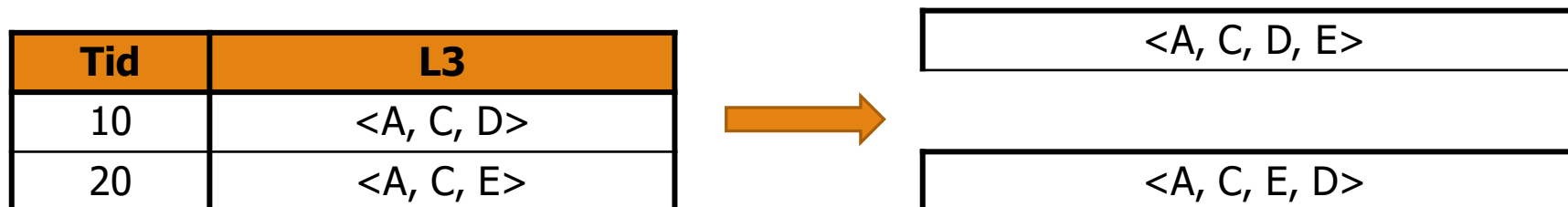
$$S_2 = \langle a \rangle \ S_3 = \langle b \rangle \ S_4 = \langle c \rangle$$

$$S_5 = \langle a \ b \rangle \ S_6 = \langle b \ c \rangle$$

$$S_7 = \langle a \ c \rangle$$

➤ AprioriAll算法基础

- ❑ 基本思想：基于Apriori算法的扩展，只是在产生候选序列和频繁序列方面考虑序列元素有序的特点，将项集的处理改为序列的处理
- ❑ 迭代过程：由 L_k 得到 L_{k+1}
 - 连接步(由 L_k 得到 C_{k+1})：对 L_k 中任意的两个序列 s_1 和 s_2 ，如果 s_1 和 s_2 的前 $k-1$ 项相同，则合并序列 s_1 和 s_2 ，并得到2个候选序列



- 剪枝步(压缩候选集)：一个候选 k 序列，如果它的任意一个 $k-1$ 子序列是非频繁的，则删除它

➤ AprioriAll算法流程

输入：序列数据库 S ; 支持度阈值 min_sup

输出：频繁序列的集合 L

- ❑ 找出所有的频繁项集 L_1
- ❑ 赋值 $k=1$ $L = L_1$
- ❑ 迭代循环
 - 连接步骤：从频繁序列 L_k 中生成候选集 C_{k+1}
 - 剪枝步骤：扫描 S 并计数 C_{k+1} 中的每一个序列 c ,去掉支持度小于 min_sup 项集，生成 L_{k+1}
 - 更新变量： $k=k+1$ $L=L \cup L_1$
 - 直到不能生成频繁序列为止，循环结束
- ❑ 返回 L

4 序列模式挖掘

➤ AprioriAll算法实例

SID	Items
10	<{AE},B,C,D>
20	<A,C,D,{CE}>
30	<A,B,C,D>
40	<A,C,E>
50	<D,E>

L_1

Sequences	sup
<A>	4
<C>	4
<D>	4
<E>	4

C_2

Sequences	sup
<A,C>	4
<A,D>	3
<A,E>	2
<C,D>	3
<C,E>	2
<D,E>	2

L_2

Sequences	sup
<A, C>	4
<A, D>	3
<C, D>	3

C_3

合并的2-sequences	3-sequences	是否剪枝	sup
<A, C> <A, D>	<A,C,D>	否	2
	<A,D,C>	是 (<D,C>不属于L2)	0

L_3

\emptyset

minsup = 3

➤ AprioriAll算法实例

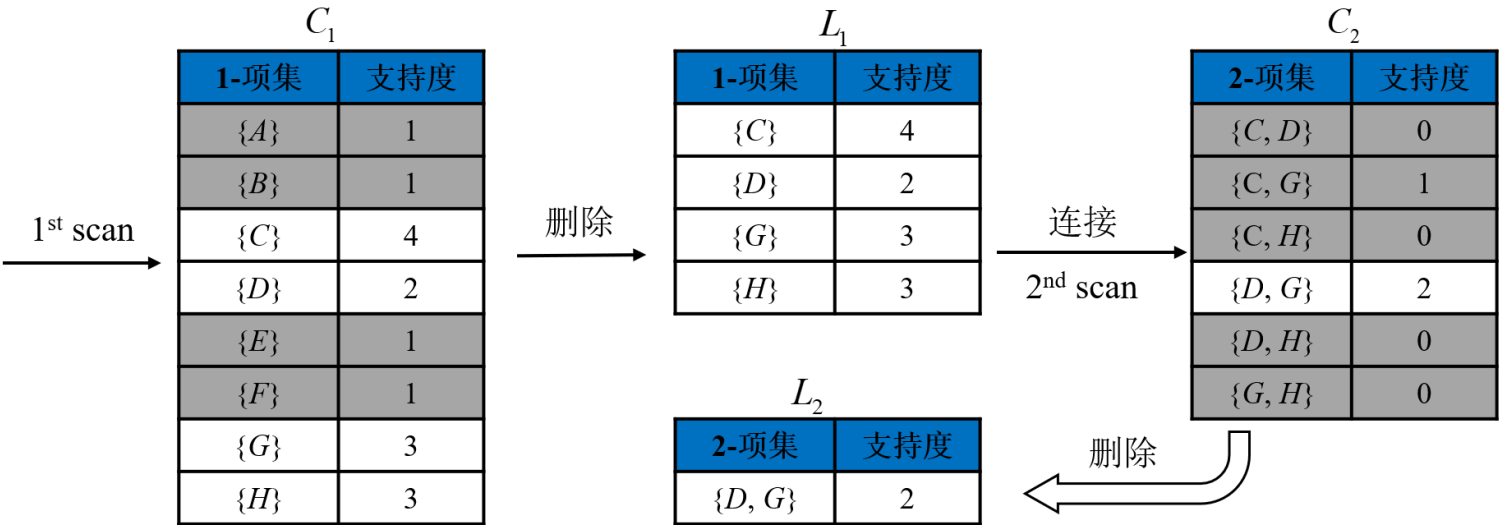
TID	报警发生时间	报警事件
1	2022:5:05:37	A
	2022:5:05:42	H
2	2022:5:06:25	A, B
	2022:5:07:05	C
	2022:5:07:15	D, F, G
3	2022:5:04:52	C, E, G
4	2022:5:08:12	C
	2022:5:08:24	D, G
	2022:5:08:45	H
5	2022:5:09:06	H

minsup = 2

□ 步骤1：排序阶段

SID	报警序列
1	< {A}, {H} >
2	< {A, B}, {C}, {D, F, G} >
3	< {C, E, G} >
4	< {C}, {D, G}, {H} >
5	< {H} >

□ 步骤2：大集合阶段



➤ AprioriAll算法实例

□ 步骤3：转换阶段

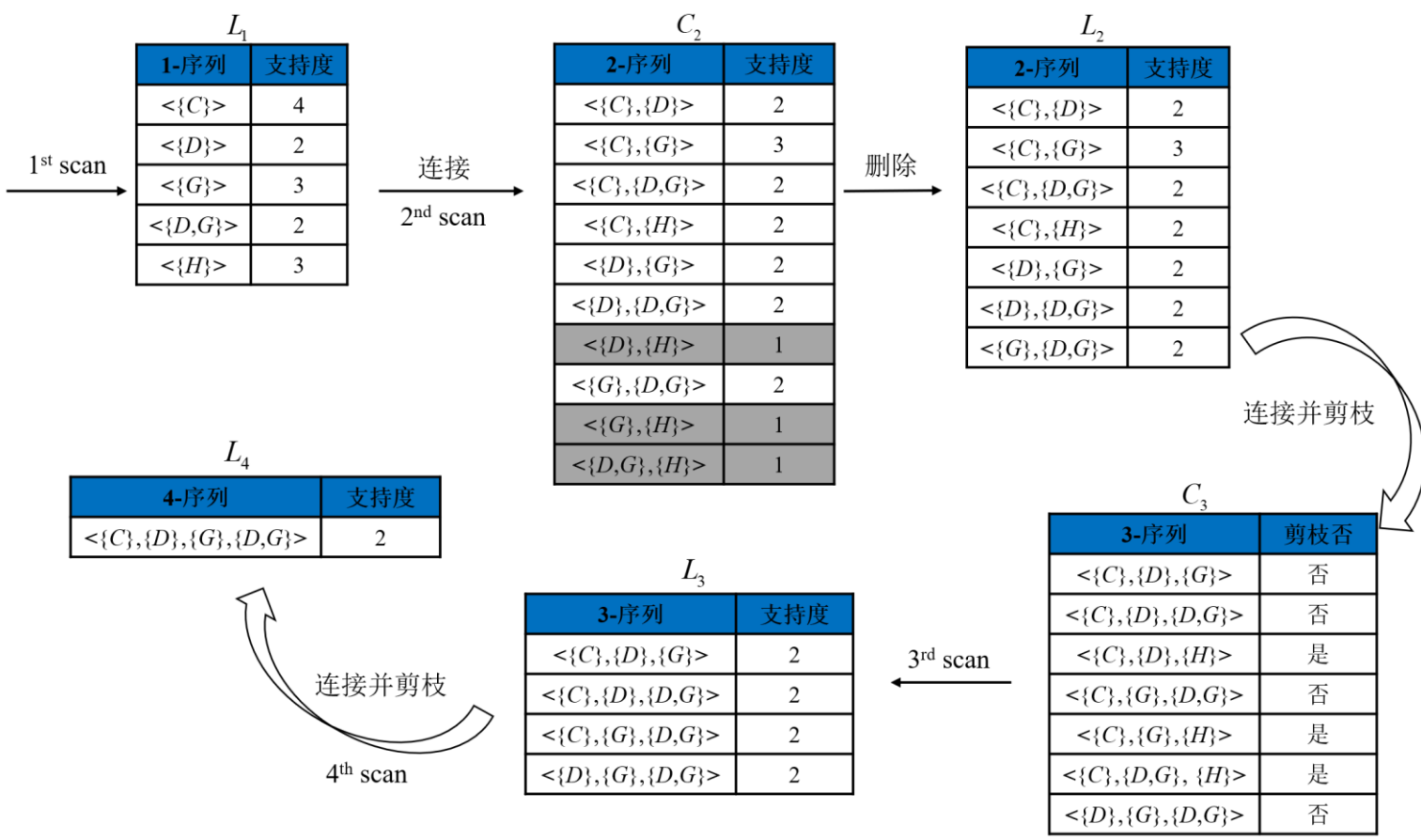
- 将序列中的事务替换为原事务产生的所有频繁项集
- 事务不包含任何频繁项集则删除
- 如果一个序列不包含任何频繁项集，则将该序列删除

SID	原始序列	新序列
1	$\langle \{C\}, \{H\} \rangle$	$\langle \{C\}, \{H\} \rangle$
2	$\langle \{A, B\}, \{C\}, \{D, F, G\} \rangle$	$\{C\}, \{D\}, \{G\}, \{D, G\} \rangle$
3	$\langle \{C, E, G\} \rangle$	$\langle \{C, G\} \rangle$
4	$\langle \{C\}, \{D, G\}, \{H\} \rangle$	$\langle \{C\}, \{D\}, \{G\}, \{D, G\}, \{H\} \rangle$
5	$\langle \{H\} \rangle$	$\langle \{H\} \rangle$

➤ AprioriAll算法实例

❑ 步骤4：序列阶段

- 对序列数据库进行频繁模式挖掘



❑ 步骤5：最大序列化

- 列出所有频繁序列集合后删除子序列得最大序列

序列模式	支持度
$\langle \{C\}, \{D\}, \{G\}, \{D,G\} \rangle$	2
$\langle \{C\}, \{H\} \rangle$	2

➤ 频繁模式挖掘的基本概念

- 频繁模式挖掘的意义
- 基本概念：项、事务、数据库、模式支持度

➤ 频繁项集挖掘

- 频繁项集的基本概念以及性质
- 挖掘频繁项集的算法：Apriori算法、FP-Growth算法、垂直数据结构算法

➤ 关联规则挖掘

- 关联规则的概念、产生和评估
- 模式评估指标：提升度、卡方距离、全置信度、最大置信度、Kulczynski、余弦、不平衡比

➤ 序列模式挖掘

- 序列模式概念
- 序列模式算法：AprioriAll算法、PrefixSpan算法

- ❑ Aggarwal and C. C. *Data mining: the textbook*. Heidelberg: Springer, 2015.
- ❑ J. Han, J. Pei, and M. Kamber. *Data mining: concepts and techniques*. Amsterdam: Elsevier, 2011.
- ❑ W. Hu, T. Chen and S. L. Shah. Detection of frequent alarm patterns in industrial alarm floods using itemset mining methods. *IEEE Transactions on Industrial Electronics*, 65(9):7290-7300, 2018.
- ❑ W. Hu, T. Chen and S. L. Shah. Discovering association rules of mode-dependent alarms from alarm and event logs. *IEEE Transactions on Control Systems Technology*, 99:1-13, 2017.
- ❑ B. Zhou, W. Hu and T. Chen. Pattern extraction from industrial alarm flood sequences by a modified CloFAST algorithm. *IEEE Trans. Industrial Informatics*, 18(1): 288-296, 2022.
- ❑ S. Naulaerts, P. Meysman, W. Bittremieux, Vu, T. N. Vu, B. Goethals, and K. Laukens. A primer to frequent itemset mining for bioinformatics. *Briefings in bioinformatics*, 16(2): 216–231, 2015.
- ❑ E. Glatz, S. Mavromatidis, B. Ager and X. Dimitropoulos. Visualizing big network traffic data using frequent pattern mining and hypergraphs. *Computing*, 96(1):27–38, 2014.
- ❑ Y. Duan, X. Fu, B. Luo, Z. Wang, J. Shi, X. Du. Detective: Automatically identify and analyze malware processes in forensic scenarios via DLLs. In: *Proc. 2015 IEEE 30 International Conference on Communications*, London, United Kingdom, 2015.