目录

4 频繁模式挖掘

在复杂工业过程中，除了由传感器检测获得的数值属性类型的过程数据外，还广泛存在报警日志、操作记录、系统状态变化等大量标称属性类型的文本数据。这些文本数据通常隐含了丰富的经验知识，需要对其进行深层次挖掘，提取并发现其中蕴藏的频繁模式和规则，从而揭示报警事件、操作行为、系统状态变化等的关联关系和生成模式，为工业报警系统的优化设计、生产运行过程的操作决策等提供有效决策支持。

图 4.1 给出了一个工业报警事件数据的例子，在工业生产运行过程中，当系统出现异常或故障时，会产生报警信号并以事件形式记录在历史日志库中，包括报警发生时间、报警标签和设备单元等信息。通过对该数据库进行数据挖掘，可以发现图右侧所示的报警模式，如{Tag04.PVHI,Tag44.PVHI,Tag02.PVHI}，表示这三个报警事件在生产运营过程中经常一起发生。分析这些报警频繁模式有助于揭示报警事件间的关系，为报警响应策略和维护检修计划的制定提供决策支持，对于提升报警系统效率和降低生产运营风险具有重要作用。由于工业数据集往往较为庞大，通过人工方式进行频繁模式挖掘的难度较大且不可靠，因此迫切需要数据挖掘技术，实现对频发模式的高效准确发掘。

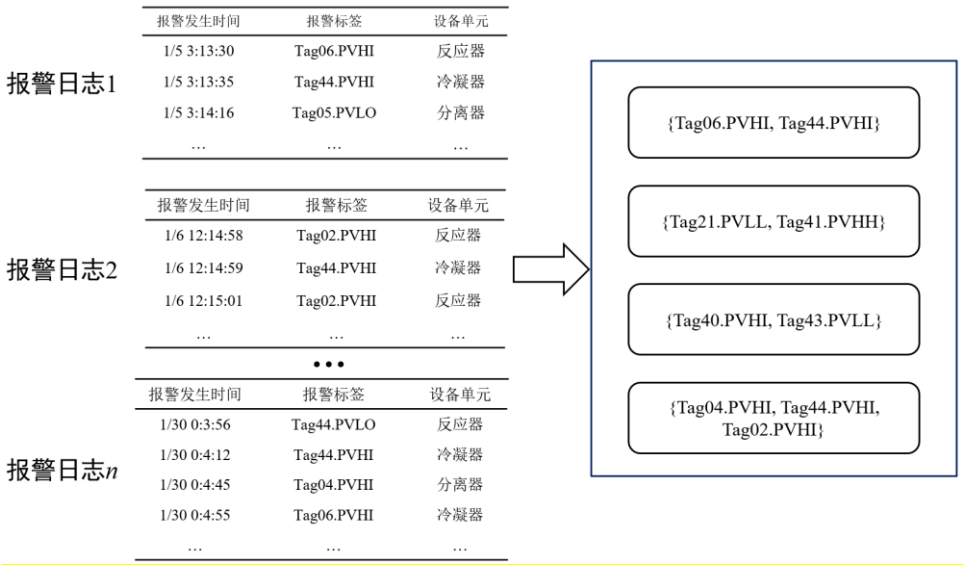

图 4.1 报警事件数据库中的频繁项集

本章首先介绍频繁模式挖掘相关的基本概念；其次，展开讲解频繁项集挖掘技术，包括三种经典的频繁项集挖掘算法，以及频繁项集模式压缩方法；接着，进一步引入关联规则挖掘任务，该任务主要用来揭示频繁项集之间的关联

性，并利用常用的模式评估指标揭示模式或规则在支持度-置信度框架之外真实有效的内在联系；最后，具体介绍序列模式挖掘技术，包括两种经典的序列模式挖掘算法，能够挖掘出频繁项集之间的另一重要信息，即时间先后关系。

4.1 频繁模式挖掘的基本概念

本节对频繁模式的基本概念进行介绍，包括项、项集、事务、事务数据库、模式支持度、频繁项集等，以帮助更好地理解频繁项集挖掘、关联规则挖掘、序列模式挖掘等后续内容。

4.1.1 项与项集、事务、事务数据库

频繁模式指频繁出现在数据集的模式，包括项集、子序列或子结构等。要想理解什么是频繁模式，必须首先认识模式的构成和来源。这里我们可以借助表 4.1 中的例子来了解这些概念。

表 4.1 事务数据库

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

项（item）：组成频繁模式的最小单元，同时也是事务数据库的最小单元，通常用于表示数据库的具有唯一区分标签的事务。例如，项在报警日志中代表生产过程中的一个报警事件。

项集（itemset）：由多个项组成的非空集合。例如，$\{A, B\}$为一个项集。

事务（transaction）：在事务数据库中存在对应唯一标识符（TransactionIDentifier，TID）的项集，一个事务通常代表数据库中单次行为的含义。例如，表 4.1 中的第三行$T_3 = \{B, C\}$为一个事务。

事务数据库：全体事务构成的集合，也是频繁模式挖掘的对象，记作$\mathbb{D} = \{T_1, T_2, \dots, T_m\}$，其中，$m$为数据库$\mathbb{D}$中全体事务的总数，即$|\mathbb{D}| = m$。

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，$T_2 = \{B, D\}$，$T_4 = \{A, B, D\}$，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，记为$X \subset Y$。

不难发现，每一个事务T_i（$i = 1,2, \dots, m$）都对应项的集合$I = \{I_1, I_2, \dots, I_n\}$上的一个子集，事务数据库$\mathbb{D} = \{T_1, T_2, \dots, T_m\}$则是由一系列具有唯一标识符 TID 的事务组成。

4.1.2 模式支持度与频繁模式

判断某种模式是否为频繁模式需要一个标准的评估指标，当模式出现的次数超过设定阈值时视其为频繁模式。模式的支持度由此而来，它直观地反映了模式出现频次水平的高低。根据计算方法的不同，支持度又分为绝对支持度和相对支持度。

绝对支持度：模式出现在事务中的次数。对于一个模式，单个事务只有存在或者不存在两种结果，即单次事务最多出现一次该模式。例如，观察表 4.1 可以发现项A在事务T_1、T_4、T_5中出现，故项A的绝对支持度为 3，即

$$\mathrm{Sup}(A)_{\text{绝对}} = |T_1, T_{4,} T_5| = 3$$

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，在表 4.1 中项A出现了 3 次，而事务总数为 5 个，故相对支持度为 0.6，即

$$\mathrm{Sup}(A)_{\text{相对}} = 3/5 = 0.6$$

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。根据数据结构不同，频繁模式的形式可以是多样的，包括频繁项集、频繁序列或关联规则等。例如，假设$\mathrm{minsup} = 2$，在表 4.1 中项集$\{A, B\}$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。候选集一般是在频繁模式挖掘过程中存在的中间模式的集合，其中满足支持度条件的模式保留，否则进行舍弃。

通常，频繁模式通过发现满足最小支持度阈值minsup的候选集来确定，但实际工业中的数据规模往往较为庞大，模式搜索空间复杂度呈指数规模。对于包含 k 项的数据库，存在有$2^k - 1$个候选项集。以上表 4.1 中的数据库为例，数据库一共包含 5 项，可以产生$\mathrm{C}_5^1+\mathrm{C}_5^2+\mathrm{C}_5^3+\mathrm{C}_5^4+\mathrm{C}_5^5=2^5-1=31$ 个候选项集。例如，C_5^1=5 个 1-候选项集为$\{A\}$、$\{B\}$、$\{C\}$、$\{D\}$、$\{E\}$，C_5^2=10 个 2-候选项集为$\{A, B\}$、$\{A, C\}$、$\{A, D\}$、$\{A, E\}$、$\{B, C\}$、$\{B, D\}$、$\{B, E\}$、$\{C, D\}$、$\{C, E\}$、$\{D, E\}$，以此类推。通过人工穷举的形式来进行模式挖掘往往不太现实，特别是针对大型的数据库。因此，需要选用合适的挖掘算法，从数据库中自动获得有意义的频繁模式，接下来将对频繁模式挖掘、关联规则挖掘和序列模式挖掘三种模式挖掘技术进行详细介绍。

4.2 频繁项集挖掘

遍历检测候选集中模式的支持度可以搜索出全部的频繁模式，然而这种方法需要先根据数据库生成频繁模式候选集，在面向规模庞大的工业数据库时显得尤其低效。众多学者对此开展了大量研究并相继开发了多种频繁模式挖掘算法，实现了对频繁模式的高效挖掘。频繁项集模式是一类最普遍也是最简单的频繁模式范式，掌握频繁项集挖掘算法是理解其它频繁模式挖掘的基础。常见的频繁项集挖掘算法包括 Apriori 算法、频繁模式增长（FrequentPattern Growth，简称 FP-Growth）算法以及垂直数据结构算法。其中，Apriori 算法和 FP-Growth 算法的挖掘对象都是 TID-项集数据库（称为水平数据格式），但二者在搜索策略上有所不同，前者是使用广度优先，后者则使用深度优先；垂直数据结构算法则是对项集-TID 数据库（称为垂直数据格式）进行挖掘。

实际上，挖掘出的频繁项集仍然存在冗余度较高而导致整体模式价值密度较低的问题，不仅会影响对模式结果的分析，而且造成大量计算资源的浪费。针对该问题，一种朴素的解决思想是在挖掘频繁模式的过程不选择遍历输出所有频繁模式，而是仅仅选择部分具有代表性的模式，这些模式在数量上相比全部的频繁模式大大降低，但是信息损失很少，因此可以代表全部的频繁模式。除了上述思想，还可以使用模式压缩的方法降低模式冗余度：利用聚类找出数据集中相似的频繁项集，并从各簇中选择输出一个具有代表性的频繁项集。本节将对以上的内容进行依次介绍，并结合具体例子进行分析。

4.2.1Apriori 算法

Apriori 算法使用逐层搜索的方法，即通过k-项集产生$(k+1)$-项集的候选集，有效抑制了候选集的指数增长。该算法首先从事务数据库$\mathbb{D}$中检查各项支持度，找出频繁项的集合L_1；接着从$k=1$开始，利用L_k逐层连接生成候选项集C_{k+1}，通过检查其中模式支持度得到长度为$k+1$的频繁模式的集合L_{k+1}；以此类推，直到无法再生成频繁项集。为了降低计算复杂度，可以利用先验原理来减少候选项集的数目，避免不必要的搜索带来的计算开销。

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。

基于先验原理，Apriori 算法通过连接步和剪枝步，在每次迭代过程中从L_{k-1}找出L_k，该算法的主要步骤如下：

（1）连接步

连接步的目的是对 Apriori 算法每次搜索得到的L_{k-1}，通过将其中的频繁项集相互连接，从而生成长度为k的频繁模式候选集C_k。

执行连接时首先假定项集中的元素按字典排序，即包含$k-1$个元素的项集I_i，存在顺序$I_i[1]<I_i[2]<\cdots<I_i[k-1]$，其中$I_i[j]$为$I_i$的第$j$项。对于频繁项集的集合$L_{k-1}$，若其中的两个频繁项集$I_1$与$I_2$的前$(k-2)$个项均相同而最后一项不同，则认为$I_1$和$I_2$可连接，需要将最后一项不同的项进行连接生成长度为$k$的项集。例如连接$I_1$和$I_2$产生的结果项集是$\{I_1[1],I_1[2],\ldots,I_1[k-1],I_2[k-1]\}$。

（2）剪枝步

剪枝步的目的是对长度为k的频繁模式候选集C_k进行支持度统计，从而确定频繁项集构成的集合L_k。在此步骤中可运用先验性质对C_k进行压缩，即当C_k中一个项集I_i的$(k-1)$-项子集不在L_{k-1}中时，可以将I_i从C_k删除，因为任何非频繁的$(k-1)$-项集都不可能是频繁k-项集的子集。

Apriori 算法伪代码如下。

输入：事务数据库$\mathbb{D}$，最小支持度阈值minsup
输出：所有的频繁项集L_k

1. 扫描数据库得到L_1，赋值$k=2, L=L_1$
2. Repeat
3. 连接步骤：从L_{k-1}中生成候选集C_k
4. 剪枝步骤：压缩C_k，删除支持度小于minsup的项集，生成L_k
5. 更新变量：$k=k+1$，$L=L\cup L_k$
6. until 不能生成频繁集或候选集为止
7. 返回L

算法具体流程：首先从数据库删除支持度少于minsup的项，得到频繁1-项集的集合L_1；接着将L_1与自身进行连接产生长度为 2 的频繁项集的候选集C_2，通过对其进行剪枝操作，删除C_2中支持度少于minsup的项集得到L_2；迭代并重复该过程，直到无法再生产新的频繁项集。在每轮迭代过程中，新的频繁模式候选项集都由前一次迭代挖掘的频繁项集的集合连接自身产生。

为了更加深刻地理解该过程，现通过一个例子具体阐述 Apriori 算法的挖掘过程，包括如何执行连接和剪枝步骤以及如何搜索频繁项集。

例 4.1 表 4.2 是一个事务数据库$\mathbb{D}$，共包含 4 个事务和 5 个项，设最小支持度阈值为 2，尝试用 Apriori 算法挖掘$\mathbb{D}$中的频繁项集，挖掘过程如图 4.2 所示。

表 4.2 事务数据库$\mathbb{D}$

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

1）对数据库中各项支持度进行统计并构成候选集C_1，其中D的支持度为 1，小于minsup，因此将其删除得到L_1；

2）将L_1与自身连接产生频繁2-项集的候选集C_2，通过对C_2中项集的支持度进行统计发现项集$\{A,B\}$、$\{A,E\}$非频繁，因此剪枝时将它们从C_2中删除得到L_2；

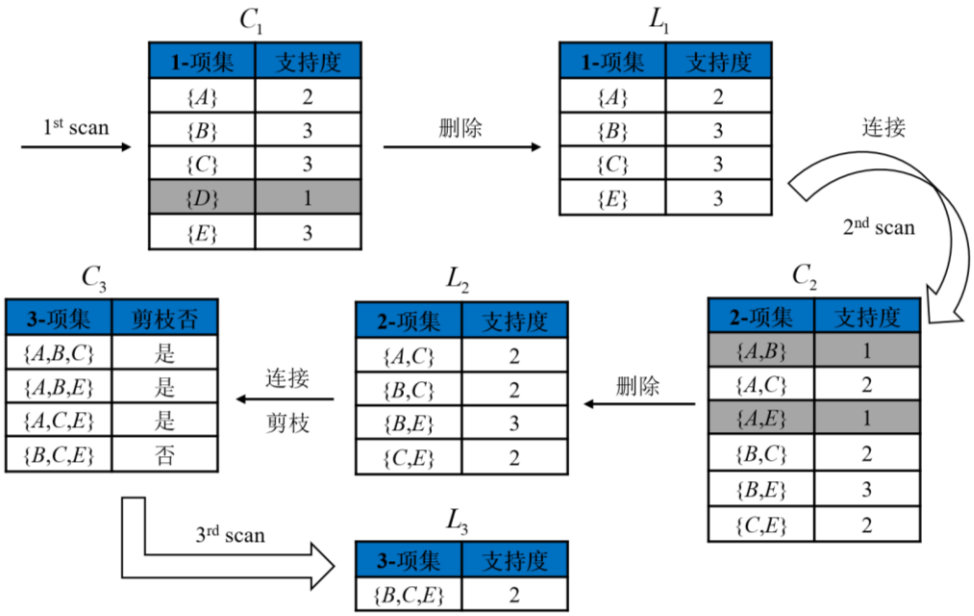

图 4.2 Apriori 算法频繁项集挖掘过程

3）将L_2与自身连接产生候选集合C_3，由于$\{A,B,C\}$的子集$\{A,B\}$、$\{A,B,E\}$的子集$\{A,B\}$和$\{A,E\}$、$\{A,C,E\}$的子集$\{A,E\}$都不是频繁项集，因此可以直接将$\{A,B,C\}$、$\{A,B,E\}$、$\{A,C,E\}$其从C_3中剪枝，接着通过对剩下的项集进行支持度统计，生成$L_3=\{B,C,E\}$；

4）L_3仅含有一个频繁项集，因此算法结束，输出所有的频繁项集。

不难发现，Apriori 算法的优点在于原理简单且易实现，适合于数据量较小的稀疏数据集，其缺点在于需要多次遍历数据集，且产生候选集时计算代价较大。这就导致了该算法面临大数据集时，存在算法复杂度高、效率低、耗时等缺点。为了避免产生候选项集时消耗巨大的计算开销，可以考虑使用频繁模式增长（FP-Growth）算法。

4.2.2FP-Growth 算法

频繁模式增长（FP-Growth）算法采用分治策略，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。与 Apriori 算法相比，FP-Growth 算法无需重复扫描数据库，而是在 FP 树上递归地搜索频繁项并为每个频繁项添加后缀使频繁模式不断增长，从而得到以某个频繁项为前缀的频繁模式。FP-Growth 算法在执行过程中仅搜索与当前模式所关联的部分数据库。随

着模式的增长，搜索空间的大小得到了显著压缩，因此该算法对于长频繁模式的挖掘具有较高的效率。

FP-Growth 算法伪代码如下。

输入：事务数据库$\mathbb{D}$，最小支持度阈值minsup
输出：所有的频繁模式

1. 扫描数据库得到频繁项（按支持度降序排序）的集合L
2. 创造 FP-Tree 根结点，标记为“null”
3. 根据$\mathbb{D}$中每个事务对 FP-Tree 添加分支
4. 对各结点建立项头表，从支持度最低的项进行回溯
5. Repeat
6. 寻找条件模式基：FP-Tree 尾缀的前缀路径
7. 构造条件 FP 树：删除路径中小于minsup的结点
8. until 条件 FP 树没有元素
9. 返回所有频繁模式

算法具体流程：首先扫描数据库得到所有频繁项并按支持度降序排列；之后以空节点作为根节点建立 FP 树：对数据库中每个事务创建分支，其中沿共同前缀的每个结点支持度计数加一，为前缀之后的项创建结点和链接；最后以项头表的结构自底依次向上地遍历 FP 树，根据频繁项寻找对应的条件模式基和条件 FP 树（为了便于理解，条件模式基和条件 FP 树这两个概念将结合下面的例子进行介绍），递归挖掘频繁项集，直到 FP 树中没有元素为止。

例 4.2 使用 FP-Growth 算法对例 4.1 中的事务数据库进行频繁项集挖掘，其中频繁项集的支持度阈值minsup设为 2。

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：$L = \{\{B:3\},\{C:3\},\{E:3\},\{A:2\}\}$，更新后如表 4.3 所示（非频繁项已删除）；

表 4.3 去除非频繁项后的事务数据库$\mathbb{D}$

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

2）构建 FP 树。首先创建根节点（用“null”标记），接着再次扫描数据库，各事务中的项都按L中的次序处理并对每个事务创建分支。具体步骤如下：

首先，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$< C:1 >$、$< A:1 >$，此时C作为根，A作为子节点连到根；

然后，添加事务$T_2=\{B,C,E\}$，由于B和C为不同项，需要新创建一个分支，包含三个结点：$<B:1>$、$<C:1>$、$<E:1>$，以B作为子节点连接至根节点，C连接至B，E紧接着连接至C；

接着，添加第三个事务$T_3=\{B,C,E,A\}$，该分支包含四个结点：$<B:1>$、$<C:1>$、$<E:1>$、$<A:1>$，并与T_2已存在的路径共享前缀B，因此只需在原分支基础上将结点B、C、E的计数加 1，并创建一个新结点$<A:1>$连接至 E；

最后，添加第四个事务$T_4=\{B,E\}$，该分支包含2个结点：$<B:1>$、$<E:1>$，并与T_2共享前缀B，因此需要将子结点B的计数增加1，并创建一个新结点$<E:1>$连接到$<B:3>$。所构造的 FP 树如图 4.3 所示。

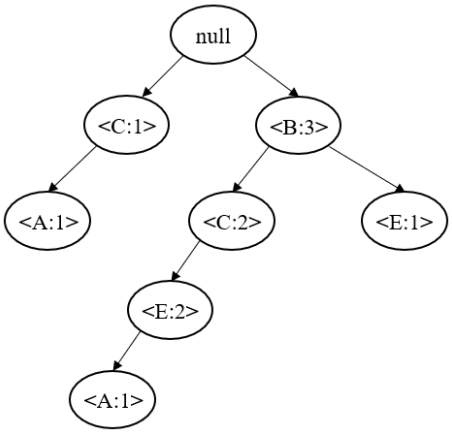

图 4.3 FP 树的构造

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，用虚线（结点链）连接 FP 树与表头，使每个项指向它在树中的位置，如图 4.4 所示。

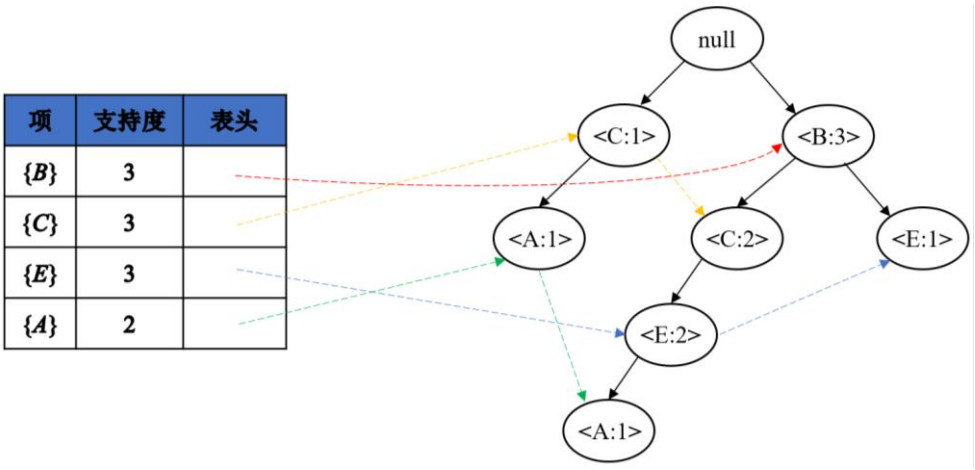

图 4.4 FP 树的项头表

4）对图 4.4 中的项头表按支持度从低到高构造各项的条件模式基（由 FP 树中与某个项作为后缀的所有前缀路径组成），依次构造条件 FP 树（FP 树中与某个项对应的频繁项集相关的子树），最后通过条件 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，该项在 FP 树中有$\{C:1\}$、$\{B,C,E:2\}$这两条前缀路径，条件模式基为$\{\{C:1\},\{B,C,E:2\}\}$；再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$<B,C,E:2>$；最后将这些频繁项 FP 树中的元素与项A组合为更大的频繁项（需要注意条件 FP 树中的该项是否频繁），直到条件 FP 树没有元素为止，得到对应的频繁模式为$\{B,C,E:2\}$。

表 4.4 FP 树挖掘频繁模式

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

接下来对更复杂一点的事务数据库使用 FP-Growth 算法进行频繁项集挖掘，以更深刻地理解本算法的原理。

例 4.3 表 4.5 给出了一个事务数据库$\mathbb{D}$，即$|\mathbb{D}|=9$，共有5个项，分别为A,B,C,D,E，请使用 FP-Growth 算法挖掘$\mathbb{D}$中的频繁项集（设$\text{minsup}=2$）。

表 4.5 事务数据库$\mathbb{D}$

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	$\{A,B,C\}$

1）扫描数据库得到频繁项的集合L，并按支持度计数递减序排序：

$$L=\{\{B:7\},\{A:6\},\{C:6\},\{D:2\},\{E:2\}\}$$

按支持度重排事务数据库，结果如下表 4.6 所示。

表 4.6 事务数据库$\mathbb{D}$

TID	事务
1	$\{B,A,E\}$

2	$\{B, D\}$
3	$\{B, C\}$
4	$\{B, A, D\}$
5	$\{A, C\}$
6	$\{B, C\}$
7	$\{A, C\}$
8	$\{B, A, C, E\}$
9	$\{B, A, C\}$

2）构造 FP 树和建立项头表

创建树的根结点（用“null”标记），再次扫描数据库$\mathbb{D}$并依次添加事务。以$T_1 = \{B, A, E\}$构造树的第一个分支（包含三个结点：$< B{:}1 >$、$< A{:}1 >$、$< E{:}1 >$），其中B为子节点，A连接到B，E连接到A；接着添加事务$T_2 = \{B, D\}$，该分支（包含两个结点：$< B{:}1 >$、$< D{:}1 >$）与T_1已存在的路径共享前缀B，因此将结点B的计数增加1，并创建一个新结点$< D{:}1 >$连接到$< B{:}2 >$；以此类推，可以得到如图 4.5 所示的 FP 树，并建立项头表。

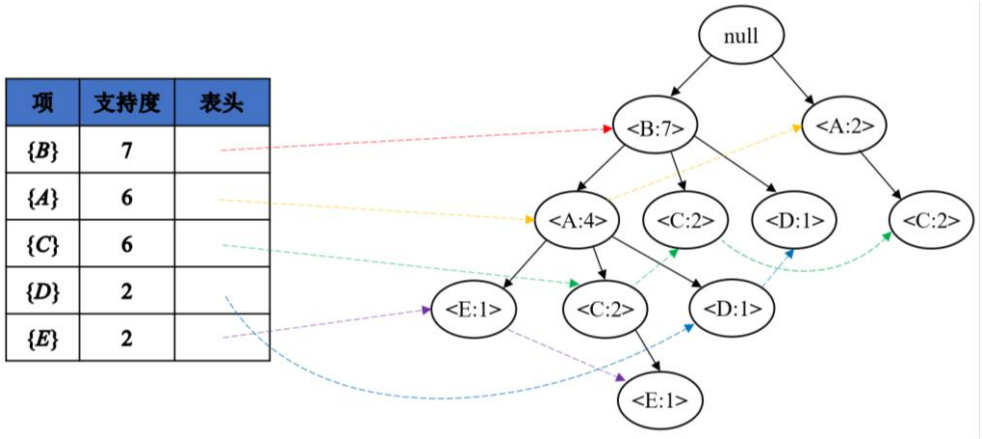

图 4.5 构造 FP 树（右）以及建立项头表（左）

3）遍历项头表，构造条件模式基和条件 FP 树，生成频繁项集

从E项开始，在 FP 树中寻找前缀路径组合并对路径中的各项支持度重新计数，分别为$B \to A \to E$（即$< B, A{:}1 >$）和$B \to A \to C \to E$（即$< B, A, C{:}1 >$），如图 4.6 所示，对路径中各项支持度重新计数以确定条件模式基为$\{\{B, A{:}1\}, \{B, A, C{:}1\}\}$；构建条件 FP 树时，仅保留条件模式基中共同路径支持度计数不低于minsup的部分（删除结点C），因此E对应的条件 FP 树为$< B{:}2, A{:}2 >$；通过将E项与条件 FP 树中的项（B和A）进行组合生成频繁项集，则有$\{B, E{:}2\}$、$\{A, E{:}2\}$、$\{B, A, E{:}2\}$。

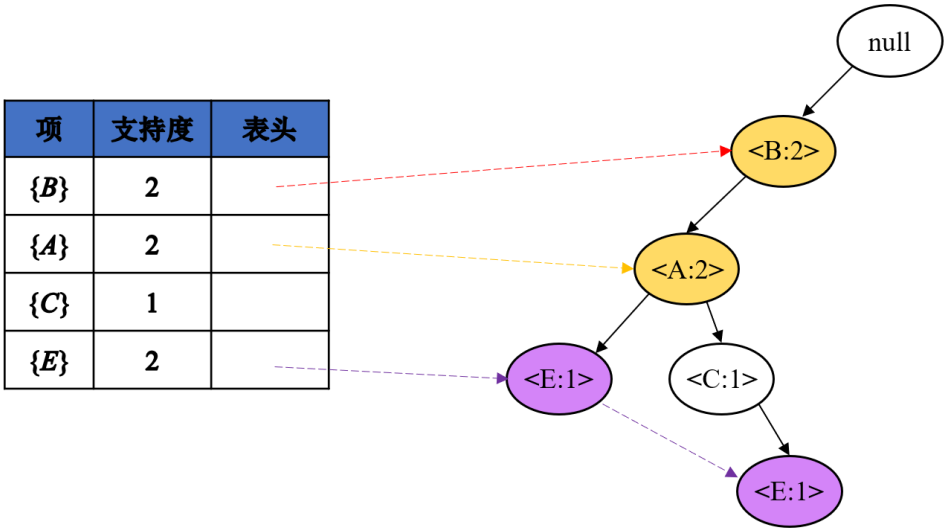

图 4.6 前缀路径组合形成条件模式基

以此类推，D项的两个前缀路径形成条件模式基为$\{\{B,A:1\},\{B:1\}\}$，由于这两条路径属于以B为子结点的第一分支，并且只共享一个单结点B，其条件 FP 树为$<B:2>$，通过与D项进行组合只能找到一个频繁模式：$\{B,D:2\}$。

C项的三个前缀路径形成条件模式基为$\{\{B,A:2\},\{B:2\},\{A:2\}\}$，其中，$B\rightarrow A\rightarrow C$和$B\rightarrow C$这两条前缀路径属于以$B$为子结点的第一分支，构成的条件 FP 树为$<B:4,A:2>$，$A\rightarrow C$这条前缀路径属于以$A$为子结点的第二分支，构成的条件 FP 树为$<A:2>$；将这两个条件 FP 树与$C$项进行组合可以得到三个频繁模式：$\{B,C:4\}$，$\{A,C:4\}$，$\{B,A,C:2\}$；这里需要说明的是，由于在条件 FP 树中，单独看结点B和A，它们俩的支持度都为 4，故对应频繁模式的支持度也为 4，而同时看结点B和A，其支持度则为 2，故对应频繁模式的支持度也为 2。

A项只有一条前缀路径形成条件模式基为$\{\{B:4\}\}$，这条前缀路径属于以B为子结点的第一分支，其条件 FP 树为$<B:4>$，因此产生一个频繁模式：$\{B,A:4\}$。频繁模式挖掘结果如表 4.7 所示。

表 4.7 FP 树挖掘频繁模式

项	条件模式基	条件 FP 树	频繁模式
$\{E\}$	$\{\{B,A:1\},\{B,A,C:1\}\}$	$<B:2,A:2>$	$\{B,E:2\},\{A,E:2\},\{B,A,E:2\}$
$\{D\}$	$\{\{B,A:1\},\{B:1\}\}$	$<B:2>$	$\{B,D:2\}$
$\{C\}$	$\{\{B,A:2\},\{B:2\},\{A:2\}\}$	$<B:4,A:2>$ $<A:2>$	$\{B,C:4\},\{A,C:4\},\{B,A,C:2\}$
$\{A\}$	$\{\{B:4\}\}$	$<B:4>$	$\{B,A:4\}$

FP-Growth 算法将挖掘长频繁模式问题转换成在较小的条件数据库中递归搜索一些较短模式，再通过连接后缀使频繁模式不断增长。不难发现，FP-

growth 算法的优点在于不产生候选集，并且只需要两次遍历数据库，大大提高了搜素效率。

4.2.3 垂直数据结构算法

从数据库格式的角度分析，Apriori 算法和 FP-growth 算法处理的数据库都属于 TID-项集格式，这种数据格式称为水平数据格式（如表 4.2）。除了水平数据格式外，数据库也可以用项-TID 集格式表示，其中 TID 集是包含项的事务的标识符的集合，这种格式称为垂直数据格式（如表 4.8）。数据的水平结构和垂直结构在信息表达方面完全等价，仅存在结构的区别，但是在一些情况下，将水平数据结构转化为垂直结构后进行数据挖掘和分析存在显著优势，因此有必要学习针对垂直数据的挖掘算法的基本原理与流程。下面结合例子，对适用于垂直数据结构的频繁项集挖掘方法，即等价类变换（EquivalenceCLAss Transformation, Eclat）算法进行介绍。

算法具体流程：首先通过扫描数据库将水平格式的数据转换为垂直格式，转换后的各项都有对应的 TID 集；从$k=1$开始，对频繁k-项集执行连接步骤以构造$(k+1)$-项集，对应的 TID 集通过取频繁k-项集的 TID 集的交集得到（可利用频繁项集的先验性质避免支持度检测时多次扫描数据库）；对$(k+1)$-项集对应的 TID 集进行支持度检测确定频繁$(k+1)$-项集；重复上述过程直到无法再生成频繁项集，并输出所有频繁项集。

例 4.4 使用垂直数据结构算法对例 4.1 中的事务数据库进行频繁项集挖掘（设 $\text{minsup}=2$）。

1）将水平数据格式转换为垂直数据结构，结果如下表 4.8 所示；

表 4.8 垂直数据结构的事务数据库

项集	TID 集
$\{A\}$	$\{T_1, T_3\}$
$\{B\}$	$\{T_2, T_3, T_4\}$
$\{C\}$	$\{T_1, T_2, T_3\}$
$\{D\}$	$\{T_1\}$
$\{E\}$	$\{T_2, T_3, T_4\}$

2）删除数据库中的非频繁项D后对剩余项集互相连接，生成的 2-项集对应的 TID 集等于这两个项集对应 TID 集的交集运算，通过对 TID 集进行支持度检测确定频繁 2-项集的垂直数据结构表，如下表 4.9 所示；

表 4.9 频繁 2-项集的垂直数据结构表

项集	TID 集
$\{A, C\}$	$\{T_1, T_3\}$
$\{B, C\}$	$\{T_2, T_3\}$
$\{B, E\}$	$\{T_2, T_3, T_4\}$

$\{C,E\}$	$\{T_2,T_3\}$

3）令$k=2$，重复 2）过程得到频繁 3-项集的垂直数据结构表。

表 4.10 频繁 3-项集的垂直数据结构表

项集	TID 集
$\{B,C,E\}$	$\{T_2,T_3\}$

为了更好的理解垂直数据结构算法，接下来会对 FP-Grown 算法里的例 4.3 中的事务数据库进行挖掘。

例 4.5 使用垂直数据结构算法对例 4.3 中的事务数据库$\mathbb{D}$进行频繁项集挖掘（设 minsup $=2$）。

1）将水平数据格式转换成垂直数据结构；

表 4.11 垂直数据结构的事务数据库

项集	TID 集
$\{A\}$	$\{T_1,T_4,T_5,T_8,T_9\}$
$\{B\}$	$\{T_1,T_2,T_3,T_4,T_6,T_8,T_9\}$
$\{C\}$	$\{T_3,T_5,T_6,T_7,T_8,T_9\}$
$\{D\}$	$\{T_2,T_5\}$
$\{E\}$	$\{T_1,T_8\}$

2）数据库中的各项都是频繁的，对任意两项进行连接（共进行 10 次运算），接着根据支持度检测删除非频繁项集$\{A,D\}$和$\{C,E\}$，结果如下表 4.12 所示；

表 4.12 频繁 2-项集的垂直数据结构表

项集	TID 集
$\{A,B\}$	$\{T_1,T_4,T_8,T_9\}$
$\{A,C\}$	$\{T_5,T_7,T_8,T_9\}$
$\{A,E\}$	$\{T_1,T_8\}$
$\{B,C\}$	$\{T_3,T_6,T_8,T_9\}$
$\{B,D\}$	$\{T_2,T_4\}$
$\{B,E\}$	$\{T_1,T_8\}$

3）根据先验性质可知，由于$\{C,D\}$非频繁，因此无需对$\{B,C\}$和$\{B,D\}$执行连接，同理$\{A,C\}$和$\{A,E\}$、$\{B,C\}$和$\{B,E\}$、$\{B,D\}$和$\{B,E\}$也无需连接。其余项集通过执行连接步生成候选 3-项集：$\{A,B,C\}$和$\{A,B,E\}$，通过支持度检测可得出它们都是频繁 3-项集，结果如表 4.13 所示。

表 4.13 频繁 3-项集的垂直数据结构表

频繁项集	TID 集
$\{A,B,C\}$	$\{T_8,T_9\}$
$\{A,B,E\}$	$\{T_1,T_8\}$

垂直结构数据算法的主要优点在于无需扫描数据库来确定候选项集的支持度，这是因为项集的 TID 集已经包含了支持度的信息，只需要集合运算就可以判断支持度，因此数据垂直结构非常适合用于项多但事务数量较少的数据库。但如果数据的 TID 集本身规模较大，那么则需要大量内存空间，同时集合运算也带来了大量时间开销，这种情况下采用数据水平结构会明显更具有优势。因此，根据数据库的特点选择合适的频繁项集挖掘算法是十分必要的。

4.2.4 模式压缩

在实际大型工业过程的数据集的频繁模式挖掘任务中，面临一个共同问题，就是产生频繁项集的数量往往会很多而难以做出有效决策。在数据库规模一定的情况下，频繁模式的数量仅由支持度阈值决定：当阈值过低时，长项集包含了指数级别数量的短频繁子模式，导致模式数量爆炸，这种情况下不利于频繁模式的分析和存储。但借助提高支持度阈值以降低模式数量时，又将会导致另一个问题：当阈值过高时，挖掘出的模式过于常识性，造成有用信息的损失。

为了减少挖掘产生的巨大频繁模式集，同时维持高质量模式，可以采用一些策略对模式进行压缩，找出具有代表性的项集。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。接下来，将介绍两种具有代表性的项集：极大项集和闭合项集。

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$\mathrm{Sup}(Y) = \mathrm{Sup}(X)$的真超集$Y$，则$X$是一个闭合项集。

根据上述两个定义可知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。

例 4.6 表 4.14 给出了一个包含 8 个频繁项集模式的事务数据库$\mathbb{D}$，需要找出其中的闭合频繁项集和极大频繁项集。

表 4.14 事务数据库$\mathbb{D}$

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

事务数据库$\mathbb{D}$中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，根据上述定义得出P_3和P_5是极大项集；由于项集P_1的支持度$\mathrm{Sup}(P_1)$=35，与其全部的真超集P_2、P_3、P_4的支持度均不相等（$\mathrm{Sup}(P_2)$=$\mathrm{Sup}(P_3)$=$\mathrm{Sup}(P_4)$ =30），因此P_1是一个闭合项集，同理P_6也是闭合项集。根据以上分析可以整理出基于表 4.5.1 的所有极大频繁项集和闭合频繁项集，如下所示。

表 4.15 极大项集和闭合项集

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	$\{A,C,D,G\}$

通过上述例子，不难发现这两种具有代表性项集的特点：极大项集可以大大降低模式数量，但压缩后的模式会损失部分支持度信息，因此不适用于支持度差距显著的情形；闭合频繁项集则是一种无损压缩，不会丢失模式的支持度信息，但是闭合频繁项集往往压缩不彻底，剩余模式仍有较高冗余，不适用于模式支持度差距较小的情形。

例 4.7 表 4.16 给出了一个大型数据集的频繁项集的一个子集，$|\mathbb{D}| = 9$，共有6个项，分别为A,B,C,D,E,F，需要对该表中的频繁模式进行压缩。

表 4.16 频繁项集的子集

ID	频繁项集	支持度
P_1	$\{B,C,D,E\}$	205227
P_2	$\{B,C,D,E,F\}$	205211
P_3	$\{A,B,C,D,E,F\}$	101758
P_4	$\{A,C,D,E,F\}$	161563
P_5	$\{C,D,E\}$	161576

根据上述定义，表中是已经使用闭频繁项集压缩后的结果，可以看出闭合频繁项集显然会导致剩余模式的高冗余，压缩效果十分不理想；考虑到项集P_2、P_3和P_4的支持度与表中唯一的极大频繁项集P_3存在显著差异，如果使用P_3代表整体的频繁模式，将导致损失大量其它模式的支持度信息。

通过观察，注意到两对模式（P_1,P_2）和（P_4,P_5）中的项元素和支持度大小都比较相似，因此可以直观地将P_2、P_3和P_4当作该数据集的一个最优压缩结果。综合上述两种代表性项集的特点，这种压缩策略可以看作是在频繁模式支持度相差较小的情形下，使用极大项集对闭合项集的结果进一步压缩。

在使用闭合项集压缩的基础上，为了严谨地描述这种直观的压缩策略，需要设计一个与支持度的差值有关的指标ξ，根据ξ是否大于设定阈值从而选择是

否使用极大项集。给定两个频繁项集P_1和P_2，其中$P_1 \subseteq P_2$，那么两者的支持度满足$\mathrm{Sup}(P_1) \geqslant \mathrm{Sup}(P_2)$，$P_1$和$P_2$的支持度差异指标$\xi$定义为：

$$\xi(P_1, P_2) = \frac{\mathrm{Sup}(P_1) - \mathrm{Sup}(P_2)}{\mathrm{Sup}(P_2)}$$

当$\xi(P_1, P_2)$过大时，即模式P_1和P_2支持度差异过大，应该同时保留P_1和P_2以避免损失P_1的支持度信息，这种情形下不应该再使用极大项集进一步压缩模式；反之，当$\xi(P_1, P_2)$过小时，模式P_1和P_2支持度差异比较小，只需要保留模式P_2代替P_1和P_2，即应该使用极大项集进一步压缩频繁模式（$\xi(P_1, P_2)$=0 是一种极端情况，此时P_1不再是闭合项集，因此在压缩时直接选择用P_2代替P_1和P_2即可）。为了量化模式P_1和P_2支持度差异是否过大，可以设定一个比对阈值$\xi_{\min}$。

ID	频繁项集	支持度
P_1	$\{B, C, D, E\}$	205227
P_2	$\{B, C, D, E, F\}$	205211
P_3	$\{A, B, C, D, E, F\}$	101758
P_4	$\{A, C, D, E, F\}$	161563
P_5	$\{A, C, D, E\}$	161576
P_6	$\{A, C, D, F\}$	161563

$\xi_{\min} = 0.1$

$P_1 \subseteq P_2\ \xi(P_1, P_2) < \xi_{\min}\ P_1 \rightarrow P_2$

$P_5 \subseteq P_4\ \xi(P_4, P_5) < \xi_{\min}\ P_5 \rightarrow P_4$

ID	频繁项集
P_2	$\{B, C, D, E, F\}$
P_3	$\{A, B, C, D, E, F\}$
P_4	$\{A, C, D, E, F\}$

图 4.7 频繁模式压缩

除了以上方法，还可以通过模式聚类的方法在闭合项集的基础上压缩频繁模式。模式聚类是将频繁模式按照模式间的距离将其分为若干个簇中，使同一个簇内的频繁模式之间相互相似，而不同簇中的模式之间不相似，从而在每个簇中选取并输出一个频繁模式集作为代表模式。为了合理恰当地量化模式之间的相似度，需要定义模式距离。

模式距离：假设P_1和P_2是两个闭合频繁项集，令$\mathrm{T}(P_1)$和$\mathrm{T}(P_2)$是模式P_1和P_2的事务集合，模式距离的计算公式如下：

$$\mathrm{D}(P_1, P_2) = 1 - \frac{|\mathrm{T}(P_1) \cap \mathrm{T}(P_2)|}{|\mathrm{T}(P_1) \cup \mathrm{T}(P_2)|} \tag{4.2}$$

其中，$\mathrm{T}(P_1) \cap \mathrm{T}(P_2)$表示模式$P_1$和$P_2$的公共事务集个数，$\mathrm{T}(P_1) \cup \mathrm{T}(P_2)$表示模式$P_1$和$P_2$的事务并集的个数。这种模式距离定义有效包含了两个模式共同以及各自的支持度信息，如果距离度量比较小则说明了模式之间的相似度较高，即两个模式就很可能在相同的事务中同时出现，因此将这两个模式分为同一个簇中。

例 4.8 下表 4.17 给出一个频繁项集数据集，并且每个频繁项集都有其对应 TID 集，给定距离阈值δ=0.45，试对频繁项集进行聚类。

表 4.16 频繁项集的子集

ID	频繁项集	TID 集
P_1	$\{B,C,D,E\}$	$\{T_1,T_2,T_3,T_4\}$
P_2	$\{A,B,C,D,E,F\}$	$\{T_2,T_3,T_4\}$
P_3	$\{C,D,E\}$	$\{T_1,T_2,T_3,T_4,T_5,T_6,T_7,T_8,T_9\}$
P_4	$\{A,C,D,E\}$	$\{T_2,T_3,T_4,T_5,T_6,T_8\}$
P_5	$\{D,E,F,G\}$	$\{T_3,T_4,T_5,T_6,T_8\}$

首先计算两两频繁项集之间的模式距离，例如计算P_1和P_2的距离时，得出两者的公共 TID 集为$\{T_2,T_3,T_4\}$，$|\mathrm{T}(P_1)\cap\mathrm{T}(P_2)|$=3，$P_1$或者$P_2$的事务并集为$\{T_1,T_2,T_3,T_4\}$，即$|\mathrm{T}(P_1)\cup\mathrm{T}(P_2)|$=4，因此$\mathrm{D}(P_1,P_2)$=1-3/4=0.25；依次类推计算其他的模式距离并将结果以三角矩阵的形式列出，之后对比各模式与其他模式的距离并与δ比较，从而确定簇划分的情况：P_1和P_2为一簇，P_3、P_4和P_5为另一簇；由于该例子中的频繁模式是项集模式，因此划分为簇后可以通过合并的方法对频繁项集进行压缩，即$P_1^{'}=P_1\cup P_2=\{A,B,C,D,E,F\}$，$P_2^{'}=P_3\cup P_4\cup P_5=\{A,C,D,E,F,G\}$，具体过程如下图所示。于是，经过模式聚类后，原模式可以用$P_1^{'}$、$P_2^{'}$两个模式代替。

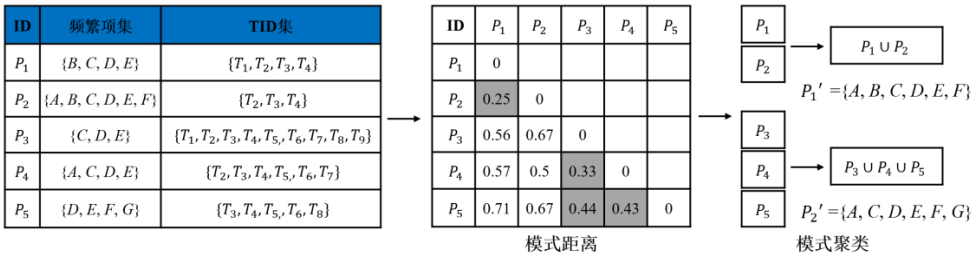

图 4.8 模式聚类过程以及结果

4.3 关联规则挖掘

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，关联规则挖掘有助于发现数据集中数据之间的关联或相关联系。通过对工厂在生产运行的过程中产生的海量历史数据进行关联规则挖掘和分析，可以发现工业数据（包括报警事件、操作记录、系统状态转换等）之间的联系，从而帮助决策者分析和解决问题。例如，建立操作模式与报警变量之间的联系，可以根据操作模式的切换状态，对报警进行动态抑制，以消除由正常操作导致的虚假报警。关联规则挖掘与频繁项集挖掘的不同之处在于，后者是在数据库中搜索出频繁出现的项集模式，前者则是在频繁项集挖掘的基础上发现这些频繁项集之间的关联性。

4.3.1 关联规则的产生

在掌握频繁项集挖掘的基础上，可以借助以下概念理解关联规则挖掘的内涵和具体流程，再结合具体例子加深理解。

关联规则：形如$X \to Y$的表达式，X和Y是两个互不相交的频繁项集，其中X是关联规则的条件（又称为先导），Y则是在条件X下导致的后验结果（又称为后继）。关联规则$X \to Y$描述了在项集X代表的事件发生的情况下，项集Y代表的事件可能发生的关联性，当这种关联性达到一定水平时，可以认为关联规则$X \to Y$是一个强关联规则。

置信度(confidence)：判断关联规则是否为强关联规则的指标，关联规则的置信度越高则表明该关联规则越值得信任，一般通过条件概率计算关联规则$X \to Y$的置信度：

$$\mathrm{Conf}(X \to Y) = \mathrm{P}(X|Y) = \frac{\mathrm{Sup}(X \cup Y)}{\mathrm{Sup}(X)} \tag{4.3}$$

其中X和Y是频繁项集，$\mathrm{Sup}(X \cup Y)$是包含X和Y的事务数，而$\mathrm{Sup}(X)$是包含项集X的事务数，$\mathrm{Conf}(X \to Y)$是同时包含 X 和 Y 的事务数量占包含 X 的事务数量的百分比，该式在已知频繁项集时可直接用于发现强关联规则。

强关联规则挖掘过程：首先扫描数据库得到项集集合L；然后通过迭代以下两个步骤产生关联规则：第一步，对L中的每个项集X，产生X的所有非空真子集；第二步，对X的每个非空真子集Y，判断关联规则$Y \to (X - Y)$是否满足其置信度不小于最小置信度阈值 minconf：

$$\mathrm{Conf}(Y \to X - Y) = \frac{\mathrm{Sup}(X)}{\mathrm{Sup}(Y)} \geq \mathrm{minconf} \tag{4.4}$$

如果满足上述式子，则输出强关联规则$Y \to (X - Y)$。

当频繁项集包含很多项时会导致产生大量候选关联规则，可利用先验性质提前剪枝以减少计算量。关联规则先验性质如下：设X为频繁项集且$\varnothing \subset Y' \subset Y \subset X$，若$Y \to (X - Y)$不是关联规则，则$Y' \to (X - Y')$也不是强关联规则。

例 4.8 基于例 4.3 的事务数据库$\mathbb{D}$，上一节已利用 FP-Growth 算法和垂直数据结构算法挖掘出了频繁项集。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对应的置信度（设置最小置信度阈值为minconf=70%）。

如图 4.9 所示，首先产生后验结果仅包含一项的关联规则；其次，根据先验性提前将低于置信度阈值的弱关联规则剔除；然后再合并后验结果生成两个项的关联规则，以此类推。

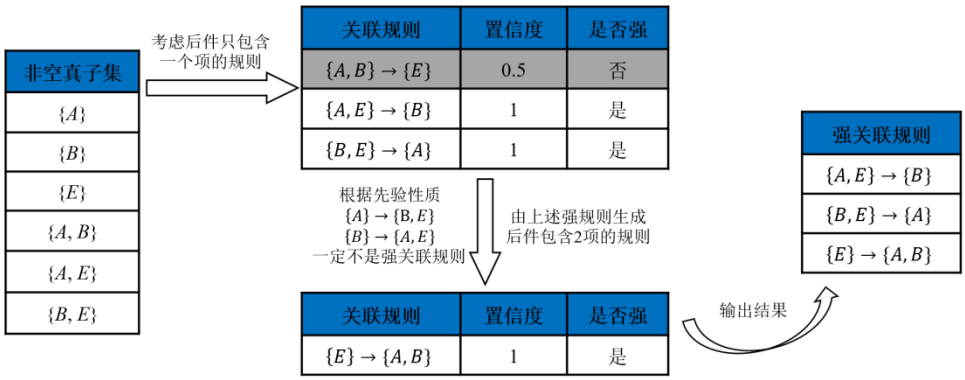

图 4.9 产生关联规则

所有的关联规则挖掘结果如表 4.18 所示，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。

表 4.18 关联规则

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

4.3.2 关联规则的评估

目前大部分关联规则都使用支持度和置信度的评估判断强关联规则，然而仅使用这两种指标仍然会导致一些错误的规则。下面以工厂运行过程中的报警事件之间的关联规则为例，分析支持度-置信度评估框架的不足。

例 4.12 针对工业过程的两个监控过程变量，分别为产品分离器温度（T11）和产品分离器压力（P13）。其中，用TH表示产品分离器的温度过高（即监测数值超过上限阈值），用PL表示产品分离器的压力过小（即监测数值小于下限阈值）。假设 100 个工业报警事务中，60 个报警事务包含TH，75 个报警事务包含 PL，而 40 个报警事务同时包含TH和PL。使用最小支持度 30%，最小置信度 60%，找出其关联规则。

通过计算分析，可以发现这样的强关联规则：TH → PL。因为，它的支持度为 40%，置信度为 66%，分别满足最小支持度阈值和最小置信度阈值。然而，实际上该关联规则是误导，因为产品分离器的压力变小的概率是 75%，比 66%还高。事实上，TH和PL是负相关的，因为产品分离器的温度上升实际上降低了产品分离器的压力变小的可能性。

本质上来说，关联规则的置信度有一定的误导性，因为它无法客观度量报警A和B之间的实际关联性。因此除了置信度之外，还使用其它评估指标对模式进行度量，这里主要介绍以下几类指标：

（1）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$\mathrm{P}(A \cup B) = \mathrm{P}(A)\mathrm{P}(B)$，则项集$A$与$B$独立，否则项集$A$、$B$具有一定相关性，在此基础上给出提升度的公式

$$\mathrm{lift}(A,B) = \frac{\mathrm{P}(A \cup B)}{\mathrm{P}(A)\mathrm{P}(B)} \tag{4.5}$$

根据计算结果，有以下情况：①$\mathrm{lift}(A,B) > 1$，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；②$\mathrm{lift}(A,B) < 1$，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不出现；③$\mathrm{lift}(A,B) = 1$，则A和B独立。可以利用提升度来筛选类似$\mathrm{TH} \rightarrow \mathrm{PL}$的错误关联规则。

例 4.13 基于例 4.12，设¬TH表示不包含产品分离器温度过高的事务，¬PL表示不包含产品分离器压力过小的事务，将这些报警事件汇总在一个相依表 4.28 中。

表 4.28 报警事件相依表

	TH	¬TH	∑row
PL	4000	3500	7500
¬PL	2000	500	2500
∑col	6000	4000	10000

由该表可以看出，产品分离器温度过高的概率$\mathrm{P}(\mathrm{TH}) = 0.6$，产品分离器压力过小的概率$\mathrm{P}(\mathrm{PL}) = 0.75$，而两者同时发生的概率$\mathrm{P}(\mathrm{TH} \cup \mathrm{PL}) = 0.4$。提升度为：$\mathrm{lift}(\mathrm{TH},\mathrm{PL}) = \mathrm{P}(\mathrm{TH} \cup \mathrm{PL})/(\mathrm{P}(\mathrm{TH})\mathrm{P}(\mathrm{PL})) = 0.4/(0.75 \times 0.6) = 0.89 < 1$，因此TH和PL存在负相关，而这种负相关是不能被支持度-置信度框架识别。

（2）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。它是另一种相关性度量，计算公式如下：

$$\chi^2 = \sum \frac{(\text{观望值} - \text{期望值})^2}{\text{期望值}} \tag{4.6}$$

根据计算结果，有以下情况：①$\chi^2 = 0$，则表明两个分布一致；②χ^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，即表明接近假设；③χ^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，即表明远离假设。

例 4.14 使用χ^2对表 4.28 的相依表进行相关性分析。

表 4.29 报警事件相依表

	TH	¬TH	∑row
PL	4000(4500)	3500(3000)	7500
¬PL	2000(1500)	500(1000)	2500
∑col	6000	4000	10000

相依表每个位置对应观测值，为了方便计算，将期望值显示在括号内。此时，很容易计算其卡方距离度量为：

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

由于卡方距离的值大于 1，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，这与提升度分析得到的结果一致。从上面的两个例子可发现提升度或者卡方距离可以揭示更多的模式内在联系。

（3）全置信度、最大置信度、Kulczynski 和余弦

给定两个项集A和B，它们的全置信度定义为：

$$\text{allconf}(A,B) = \frac{\text{Sup}(A \cup B)}{\max\{\text{Sup}(A), \text{Sup}(B)\}} \tag{4.7}$$

其中，$\max\{\text{Sup}(A), \text{Sup}(B)\}$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“$A \to B$”和“$B \to A$”的最小置信度。

最大置信度定义为：

$$\text{maxconf}(A,B) = \max\{\text{P}(A|B), \text{P}(B|A)\} \tag{4.8}$$

Kulczynski 对两个置信度求平均值，定义为：

$$\text{Kulc}(A,B) = \frac{1}{2}(\text{P}(A|B) + \text{P}(B|A)) \tag{4.9}$$

余弦度量对A和B的概率的乘积取平方根，因此该指标不受事务总个数的影响，定义为：

$$\cos(A,B) = \sqrt{\text{P}(A|B) \times \text{P}(B|A)} \tag{4.10}$$

这 4 种度量有两个共同性质：①仅受条件概率$P(A|B)$的影响，而不受事务总个数的影响；②取值范围为[0,1]，值越大相关性越强。

以上 6 种模式评估度量中，lift和χ^2会受到零事务的影响。零事务是不包含任何项集的事务，当数据集中存在大量零事务时，选择不受到零事务影响的度量对于挖掘有意义的关联规则至关重要。

对于这 4 种零不变度量，哪种度量能更好地发现有意义的关联模式，需要引进不平衡比来评估规则蕴含式中两个项集A和B的不平衡程度，定义为：

$$\mathrm{IR}(A,B)=\frac{|\mathrm{Sup}(A)-\mathrm{Sup}(B)|}{\mathrm{Sup}(A)+\mathrm{Sup}(B)-\mathrm{Sup}(A\cup B)} \tag{4.11}$$

其中，分子是项集A和B的支持度之差的绝对值，而分母是包含项集A或B的事务数。如果A和B的两个方向的蕴含相同，则$\mathrm{IR}(A,B)$为0；否则，两者之差越大，即IR值越大，说明两个项集之间越不平衡。

接下来，将在一个数据集上考察这几种度量的性能，并通过分析度量值评估所发现的关联模式之间的联系。

例 4.15 基于例 4.14 把产品分离器温度和产品分离器压力的历史特征记录汇总在2×2相依表 4.30 中，其中，TP代表包含TH和PL的事务，(¬T)P代表不包含TH但包含PL的事务，T(¬P)代表包含TH但不包含PL的事务，¬(TP)代表既不包含 TH 又不包含 PL 的事务。表 4.31 中显示了 6 组报警事务数据集的联列表，要求计算以上的模式评估指标并对结果进行分析，确定哪种度量为最优评估指标。

表 4.30 报警事件相依表

	TH	¬TH	∑row
PL	TP	(¬T)P	P
¬PL	T(¬P)	¬(TP)	¬P
∑col	T	¬T	∑

表 4.31 工业数据集的联列表

数据集	TP	(¬T)P	T(¬P)	¬(TP)
D_1	10000	1000	1000	100000
D_2	10000	1000	1000	100
D_3	100	1000	1000	100000
D_4	1000	1000	1000	100000
D_5	1000	100	10000	100000
D_6	1000	10	100000	100000

通过计算能够得到 6 个评估度量的值，如下表 4.32 所示。

表 4.32 基于表 4.31 数据集的模式评估度量

数据集	**lift**	χ^2	**全置信度**	**最大置信**	**Kluc**	**余弦**
D_1	9.26	90557	0.91	0.91	0.91	0.91
D_2	1	0	0.91	0.91	0.91	0.91
D_3	8.44	670	0.09	0.09	0.09	0.09
D_4	25.75	24740	0.5	0.5	0.5	0.5
D_5	9.18	8173	0.09	0.91	0.5	0.29
D_6	1.97	965	0.01	0.99	0.5	0.1

先考察前 4 个数据集$D_1 \sim D_4$，从表 4.31 以看出，TH和PL在数据集D_1和D_2中是正关联的，因为$\mathrm{TP} = 10000$显著大于$(\neg \mathrm{T})\mathrm{P} = 1000$和$\mathrm{T}(\neg \mathrm{P}) = 1000$。直观地，对于包含TH的事务数（$\mathrm{T} = 10000 + 1000 = 11000$）而言，产品分离器压力也极有可能过小（$\mathrm{TP}/\mathrm{T} = 10/11 = 91\%$），反之亦然。在$D_3$中是负关联的，而在$D_4$中是中性的。

全置信度、最大置信度、Kulczynski 和余弦这 4 个度量在数据集D_1和D_2上的度量值为 0.91，这表明TH和PL是强正关联的。由于 lift 和χ^2这两个度量对零事务$\neg(\mathrm{TP})$敏感，在D_1和D_2上的度量值显著不同。然而在许多实际情况下，$\neg(\mathrm{TP})$通常都很大且不稳定。例如，在工业报警历史数据库中，事务总数可能按天波动，并且显著超过包含任意特定项集的事务数。因此，好的度量不应该受不包含目标的事务影响，否则就如D_1和D_2所示，会产生不稳定的结果。

类似地，在D_3中，全置信度、最大置信度、Kulczynski 和余弦这 4 个度量都表明TH和PL是强负关联的，因为TP与PL之比等于TP与TH之比，即$100/1100 = 9.1\%$。然而 lift 和χ^2都错误地与此相悖：对于D_2，它们的值都在对应的D_1和D_3的值之间。对于数据集D_4，lift 和χ^2都显示了TH和PL之间强正关联，而其他度量都指示中性关联，因为TP与$(\neg \mathrm{T})$P之比等于TP与$\mathrm{T}(\neg \mathrm{P})$之比为1，即如果产品分离器温度过高发生高阈值报警，则其压力过小发生低阈值报警的概率为50%。

为了避免 lift 和χ^2识别上述事务数据集中的模式关联关系的能力差的问题，必须考虑零事务。lift 和χ^2很难识别有意义的模式关联关系，因为它们都受$\neg$TP的影响很大。典型地，零事务的个数可能大大超过单个过程变量发生变化的个数，因为许多事务都既不包含TH也不包含PL。而另外 4 个度量的定义消除了$\neg(\mathrm{TP})$的影响（即它们不受零事务个数的影响），能更好发现有意义的模式关联关系。

对于数据集D_5和D_6，这两个报警具有不平衡的条件概率，即TP与PL的比大于 0.9，这意味PL出现的同时TH极大可能出现。TP与TH的比小于 0.1，表明TH出现的同时PL很可能不出现。对于这两种情况，全置信度和余弦度量都视两个数据集为负关联，而最大置信度视其为强正关联，只有 Kluc 度量视其为中性。

之所以会出现如此不同的结果，是由于数据“平衡地”倾斜，因此很难通过这 4 个度量的结果来说明两个数据集具有正的还是负的关联性。从一个角度分析，D_5中只有$\mathrm{TP}/(\mathrm{TP} + (\mathrm{T}(\neg \mathrm{P})) = 1000/(1000 + 10000) = 9.09\%$的与TH相关的事务包含PL；而在$D_6$中这个比例为$1000/(1000 + 100000) = 0.99\%$，两者都指示TH与PL之间的负关联。而$D_5$中 99.9%和$D_6$中 9%包含PL的事务也包含TH，这表明TH与PL正相关。

对于这种“平衡的”倾斜，Kluc 把它看做是中性可能会更客观，同时用不平衡比（IR）指出它的倾斜型。对于D_4，有IR(TH, PL) = 0，则说明该数据集是一种很好的平衡情况；对于D_5和D_6，IR(TH, PL)分别为 0.89 和 0.99，则说明这两个数据集都是相当不平衡的情况；因此，将 Kluc 和IR这两个度量一起使用，可以为数据集$D_4 \sim D_6$提供清晰的描绘。

总之，仅使用支持度和置信度度量来挖掘关联可能产生错误的关联规则，而利用附加的模式评估度量不仅能显著地减少所产生规则的数量，还可以发现更有意义的关联规则。考虑零不变性是重要的，在 4 个零不变度量（即全置信度、最大置信度、Kulczynski 和余弦）中，推荐 Kluc 与IR配合使用。

4.4 序列模式挖掘

序列模式挖掘是指从序列数据库中发现频繁子序列模式，最早由阿格拉沃尔（Agrawal）等人提出，可以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。生活中存在很多序列模式挖掘的应用场景，最典型的便是购物篮例子，超市通过对销售数据库进行序列模式挖掘，发现了啤酒与纸尿裤之间的序列关系，并以此进行组合促销获得更大利润。除此之外，序列模式挖掘还被应用在包括网络访问模式分析、科学实验的分析、疾病治疗的早期诊断、自然灾害的预测、DNA 序列的破译等方面。

在工业过程中，为了监测设备、生产线以及整个系统的运行状态，通常会在各个关键点配有传感器，以采集数据并进行处理。这些数据往往都包含时间戳属性，利用该信息可以将一段时间内发生的事件拼成序列。事件之间可能存在某种序列关系，通常表现为基于时间或空间的先后次序，工厂可根据该关系制定操作策略，为操作员提供决策支持。之前介绍的频繁项集挖掘以及关联模式挖掘都是针对同时出现的项集，并未考虑到其中的序列信息。但是序列信息对于识别动态系统的重要特征以及预测特定事件的未来发生非常有价值。因此，本节将展开介绍序列模式的基本概念以及挖掘算法。

4.4.1 序列模式挖掘的基本概念

序列（sequence）：由不同的元素按照一定顺序排列组成，其中，每一个元素可以是简单项或者项集，记作

$$s = < I_1, I_2, \dots, I_n >$$

序列数据库：由多条序列组成的数据库，记作$\mathbb{S} = < s_1, s_2, \dots, s_n >$。与事务数据库不同，序列数据库中的每一条记录不再用 TID 表示，而是使用 SID（SequenceIDentifier）作为数据库主键。

子序列性质：对于两个序列$s_1 =< q_1, q_2, \dots, q_m >$与$s_2 =< p_1, p_2, \dots, p_n >$，如果存在整数$1 \le i_1 < i_2 < \cdots < i_m \le n$，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。序列模式中的元素之间是有序的，而且单个项可能会出现多次。

在频繁项集挖掘中，事务数据库$\mathbb{D}$的每一条记录称为事务（由不同项组成），不同的事务可能由是由同一对象产生。以工业生产过程中的报警事件数据为例，假设过程设备在上午发生了报警A、B、C，在下午又发生了报警A、C、D，在晚上还发生了报警A、D、E，于是，该设备产生了 3 条报警事件记录。如果对应到事务数据库$\mathbb{D}$中，就是前 3 条事务，即$T_1 = \{A, B, C\}$，$T_2 = \{A, C, D\}$，$T_3 = \{A, D, E\}$。而在序列数据库$\mathbb{S}$中，每一条记录称为序列（由不同项或项集按时间先后顺序排列），不同的序列一定由不同的对象产生。借用上述例子，该过程设备在不同时间里发生了 3 次报警事件，对应到序列数据库$\mathbb{S}$中，需要将其记录在一个序列之中，即$S_1 =< \{A, B, D\}, \{A, C, D\}, \{A, D, E\} >$，其中，同一次的所有报警事件用括号括起来，并按事件发生时间排序。如果序列数据库中存在 4 条序列，则说明它们分别来自 4 个不同对象。

4.4.2AprioriAll 算法

序列模式挖掘的任务就是从给定的序列库找出全部频繁序列，最基础的算法为 AprioriAll 算法，其本质是对 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。算法流程具体分为五个阶段，分别为排序阶段、大集合阶段、转换阶段、序列阶段以及选最大项阶段。下面结合例子，对 AprioriAll 算法的计算流程进行介绍。

算法具体流程：首先按事务发生时间将原事务数据库$\mathbb{D}$转换成由事务序列组成的序列数据库$\mathbb{S}$；通过扫描数据库$\mathbb{S}$得到频繁项集 L 并通过子集检测将$\mathbb{S}$转换为一个新的数据库$\mathbb{S}'$，其事务由频繁项集组成；再根据 Apriori 算法的思想挖掘出$\mathbb{S}'$中的序列模式集；最后在频繁序列模式集合中找出最大频繁序列模式。

例 4.10 表 4.19 给出一个基于时间戳的报警事务数据库$\mathbb{D}$，其中$|\mathbb{D}| = 5$且共有8个项，分别为A, B, C, D, E, F, G, H，设最小支持度阈值为 minsup=2，使用 AprioriAll 算法对$\mathbb{D}$进行序列模式挖掘。

表 4.19 事务数据库$\mathbb{D}$

TID	报警发生时间	报警事件
1	2022:5:05:37 2022:5:05:42	C H
2	2022:5:06:25	A, B

	2022:5:07:05 2022:5:07:15	C D,F,G
3	2022:5:04:52	C,E,G
4	2022:5:08:12 2022:5:08:24 2022:5:08:45	C D,G H
5	2022:5:09:06	H

1）排序阶段：T_1表示在 2022 年凌晨 5 时 5 分 37 秒发生了报警A，紧接着在 5 时 5 分 42 秒发生了报警H。将其转换成序列时，由于A先发生，而H后发生，则写为$S_1 =< \{A\},\{H\} >$，以此类推，更新$\mathbb{D}$可得到下表的序列数据库$\mathbb{S}$。

表 4.20 序列数据库$\mathbb{S}$

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

2）大集合阶段：对序列数据库求出各项的支持度，找出所有频繁项集组成的集合L，具体挖掘过程如下图 4.10 所示。实际上，也同步得到所有频繁 1-序列组成的集合，即频繁 1-序列$L = \{\{C\},\{D\},\{G\},\{D,G\},\{H\}\}$。

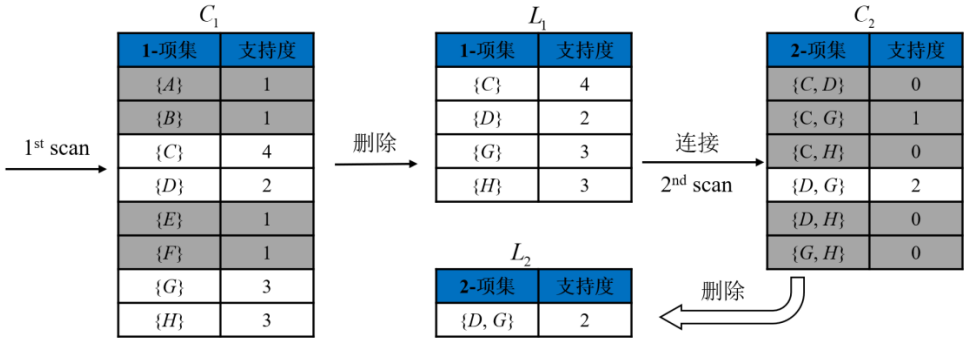

图 4.10 频繁项集挖掘

3）转换阶段：转换策略分为三种情况：①将序列中的事务替换为原事务产生的所有频繁项集，例如 4.20 表T_4里的$\{D,G\}$被$\{\{D\},\{G\},\{D,G\}\}$替代；②事务不包含任何频繁项集则删除，例如在表 4.20 中，T_2里$\{A,B\}$中的A和B都不是频繁项集，$\{D,F,G\}$中的 F 不是频繁项集，则直接删除；③如果一个序列不包含任何频繁项集，则将该序列删除。

按照以上转换策略，对表 4.20 进行转换可以得到如表 4.21 所示的序列。

表 4.21 序列转换表

SID	原始序列	新序列

1	$<\{C\},\{H\}>$	$<\{C\},\{H\}>$
2	$<\{A,B\},\{C\},\{D,F,G\}>$	$<\{C\},\{D\},\{G\},\{D,G\}>$
3	$<\{C,E,G\}>$	$<\{C,G\}>$
4	$<\{C\},\{D,G\},\{H\}>$	$<\{C\},\{D\},\{G\},\{D,G\},\{H\}>$
5	$<\{H\}>$	$<\{H\}>$

4）序列阶段：设 minsup 为 2，对序列数据库进行频繁模式挖掘。首先扫描序列数据库得到频繁1-序列的集合L_1，长度为 1 的频繁序列有5个：$<\{C\}>,<\{D\}>,<\{G\}>,<\{D,G\}>,<\{H\}>$；

接着将L_1和L_1进行连接得到C_2，这个过程实际上不需要剪枝，因为C_2中每个2-序列的所有子序列一定属于L_1；再次扫描数据库，删除小于minsup的序列得到频繁2-序列的集合L_2；长度为2的频繁序列有7个：$<\{C\},\{D\}>,<\{C\},\{G\}>,<\{C\},\{D,G\}>,<\{C\},\{H\}>,<\{D\},\{G\}>,<\{D\},\{D,G\}>,<\{G\},\{D,G\}>$；

然后重复以上步骤，可以得到频繁3-序列的集合L_3和频繁4-序列的集合L_4，即长度为 3 的频繁序列有4个：$<\{C\},\{D\},\{G\}>,<\{C\},\{D\},\{D,G\}>,<\{C\},\{G\},\{D,G\}>,<\{D\},\{G\},\{D,G\}>$，而长度为 4 的频繁序列只有1个：$<\{C\},\{D\},\{G\},\{D,G\}>$；

由于L_4中只有一个序列，由它产生的C_5为空，L_5也为空，至此算法结束。序列模式挖掘过程如下图 4.11 所示。

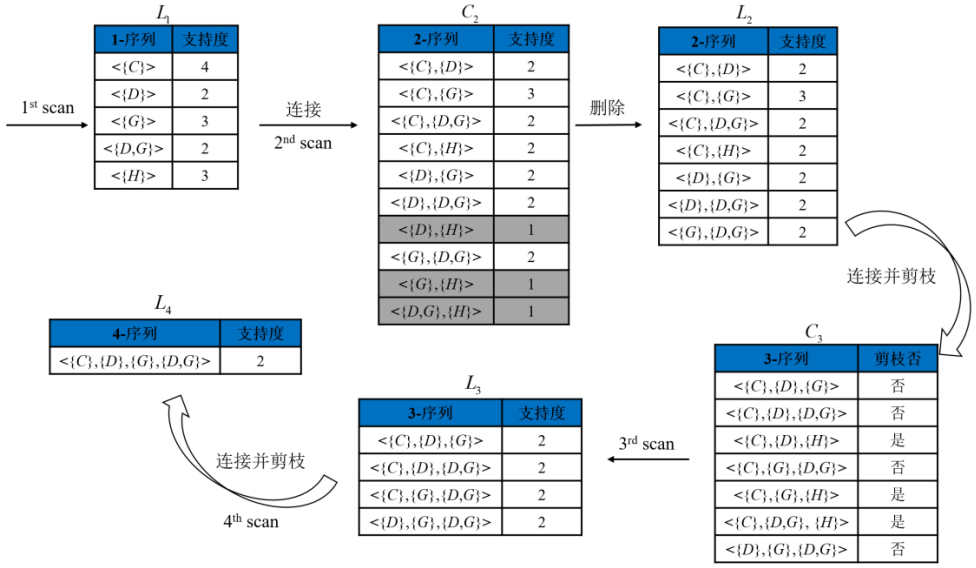

图 4.11 序列模式挖掘过程

5）最大序列化：所有频繁序列集合$L=\{<\{C\}>,<\{D\}>,<\{G\}>,<\{D,G\}>,<\{H\}>,<\{C\},\{D\}>,<\{C\},\{G\}>,<\{C\},\{D,G\}>,<\{C\},\{H\}>,<\{D\},\{G\}>,<\{D\},\{D,G\}>,<\{G\},\{D,G\}>,<\{C\},\{D\},\{G\}>,<\{C\},\{D\},\{D,G\}>,<\{C\},\{G\},\{D,G\}>,<\{D\},\{G\},\{D,G\}>,<\{C\},\{D\},\{G\},\{D,G\}>\}$。

由于最长序列的长度为4，则所有频繁4-序列都是最大序列。这里只有一个频繁4-序列，此时$<\{C\},\{D\},\{G\},\{D,G\}>$是唯一最大序列。删除子序列得到最大序列的过程如下：

对于4-序列$<\{C\},\{D\},\{G\},\{D,G\}>$从$L$中删除它的3-子序列$<\{C\},\{D\},\{G\}>$、$<\{C\},\{D\},\{D,G\}>$、$<\{C\},\{G\},\{D,G\}>$、$<\{D\},\{G\},\{D,G\}>$，2-子序列$<\{C\},\{D\}>$、$<\{C\},\{G\}>$、$<\{C\},\{D,G\}>$、$<\{D\},\{G\}>$、$<\{D\},\{D,G\}>$、$<\{G\},\{D,G\}>$和1-子序列$<\{C\}>$、$<\{D\}>$、$<\{G\}>$、$<\{D,G\}>$，剩下序列集合为：$\{<\{H\}>,<\{C\},\{H\}>,<\{C\},\{D\},\{G\},\{D,G\}>\}$，此时没有频繁3-序列，频繁2-序列$<\{C\},\{H\}>$为最大序列，从$L$中删除它的1-子序列$<\{H\}>$，至此$L$中已没有可以再删除的子序列，挖掘结束。可得到如下表 4.23 所示的序列模式。

表 4.23 序列模式

序列模式	支持度
$<\{C\},\{D\},\{G\},\{D,G\}>$	2
$<\{C\},\{H\}>$	2

AprioriAll 算法属于 Apriori 类算法，需要多次遍历数据库生成候选序列后，再利用 Apriori 性质进行剪枝得到频繁序列。每次遍历都在之前频繁序列的基础上生成更长的候选序列，然后扫描每个候选序列验证其是否为频繁序列，因此计算和存储开销都很大。为了更加集中地挖掘序列模式，接下来将介绍另一种算法，即 PrefixSpan 算法。

4.4.3PrefixSpan 算法

PrefixSpan 算法的全称是 Prefix-Projected Pattern Growth，即前缀投影的模式挖掘。其基本思想为通过前缀投影序列挖掘模式，从长度为 1 的前缀开始挖掘序列模式，搜索对应的投影数据库得到前缀对应的频繁序列，然后递归地挖掘长度为 2 的前缀所对应的频繁序列，以此类推直到无法挖掘到更长的序列模式。

给定一个序列$\alpha=<e_1,e_2,\ldots,e_n>$，其中$e_i$为对应数据库的一个项或者项集（项集元素按照字母顺序排列）。为了更加清楚 PrefixSpan 算法的执行过程，需要对以下几个概念进行介绍：

前缀：序列的部分子序列。对于α，若序列$\beta=<e_1'e_2'\ldots e_m'>(m\le n)$满足：①对于任意$1\le i\le m-1$，总有$e_i'=e_i$；②$e_m'\subseteq e_m$；③$(e_m-e_m')$中所有的项按字母顺序排列在$e_m'$之后，则$\beta$是$\alpha$的一个前缀。

例如，序列$\alpha=<\{a\},\{a,b,c\},\{a,c\},d,\{c,f\}>$，$\beta=<\{a\},\{a,b,c\},\{a\}>$，则$\beta$是$\alpha$的前缀。除$\beta$以外，$<\{a\}>$、$<\{a\},\{a\}>$和$<\{a\},\{a,b\}>$也是$\alpha$的前缀。

后缀：在序列中，前缀之后的子序列即为该前缀对应的后缀。对于前缀$\beta =< e_1e_2\ldots e_{m-1}e_m^{'} > (m \le n)$，序列$\gamma =< e_m^{''}e_{m+1}\ldots e_n >$称为$\alpha$对于$\beta$的后缀，表示为$\gamma = \alpha|\beta$，其中，$e_m^{''} = e_m - e_m^{'}$。若$\beta$=$\alpha$，则$\alpha$对于$\beta$的后缀为空集。

例如，对于序列$\alpha =< \{a\},\{a,b,c\},\{a,c\},d,\{c,f\} >$，以$< \{a\} >$为前缀对应的后缀为$< \{a,b,c\},\{a,c\},d,\{c,f\} >$；以$< \{a\},\{a\} >$为前缀对应的后缀为$< \{_,b,c\},\{a,c\},d,\{c,f\} >$。“_”下标符代表前缀最后的项是项集的一部分。

投影数据库：在 PrefixSpan 算法中，相同前缀对应的所有后缀的结合称为前缀对应的投影数据库。令α是序列数据库$\mathbb{S}$的一个序列模式，$\alpha -$投影数据库为$\mathbb{S}|_\alpha$，表示关于前缀α对应的后缀序列集合。

投影数据库中的支持度计数：令α是序列数据库$\mathbb{S}$的一个序列模式，如果β是以α为前缀的一个序列，那么β的支持度计算表示为$\mathrm{Sup}_{\mathbb{S}|\alpha}(\beta)$，是$\mathbb{S}|_\alpha$中序列$\gamma$的数量，$\beta \subseteq \alpha \cdot \beta$。

PrefixSpan 算法的伪代码如下。

输入：序列数据库$\mathbb{S}$，最小支持度阈值 minsup
输出：全部序列模式

调用过程PrefixSpan（<>，0，$\mathbb{S}$）
PrefixSpan $(\alpha, 1, \mathbb{S}|_\alpha)$
1. 扫描一次$\mathbb{S}|_\alpha$，找到每一个频繁项集β
   (a) β能够被组装到α的最后一个元素，形成一个序列模式；
   (b) $< \beta >$能够被追加到α，形成一个序列模式
2. 对于每一个频繁项β，把它加到α中形成序列模式α'，并且输出α'
3. 对于每一个α'，构α'-投影数据库$\mathbb{S}|_{\alpha'}$，并且调用 PrefixSpan(α',1+1,$\mathbb{S}|_{\alpha'}$)

参数α是序列模式，l 是α的长度。如果$\alpha \neq <>$，$\mathbb{S}|_\alpha$是α-投影数据库，否则，参数$\mathbb{S}|_\alpha$即为序列数据库$\mathbb{S}$。

例 4.11 表 4.24 给出一个序列数据库$\mathbb{S}$，其中$|\mathbb{S}| = 4$且共有7个项，分别为A, B, C, D, E, F, G，设最小支持度阈值为 minsup=2，使用 PrefixSpan 算法对$\mathbb{S}$进行序列模式挖掘。

表 4.24 序列数据库$\mathbb{S}$

SID	序列
1	$< \{A\},\{A,B,C\},\{A,C\},\{D\},\{C,F\} >$
2	$< \{A,D\},\{C\},\{B,C\},\{A,E\} >$
3	$< \{E,F\},\{A,B\},\{D,F\},\{C\},\{B\} >$
4	$< \{E\},\{G\},\{A,F\},\{C\},\{B\},\{C\} >$

1）扫描数据库获得长度为 1 的序列模式（频繁项）。需要注意的是在一个项在单条序列中最多出现一次，例如A项在

$S_1 =< \{A\},\{A,B,C\},\{A,C\},\{D\},\{C,F\} >$序列中出现了多次，但仍计数一次。结果如下表 4.26 所示；

表 4.26 长度为 1 的序列模式

序列模式	$< \{A\} >$	$< \{B\} >$	$< \{C\} >$	$< \{D\} >$	$< \{E\} >$	$< \{F\} >$
支持度	4	4	4	3	3	3

2）划分搜索空间。根据这六个前缀，序列模式集合被划分为以下六个子集：前缀为$< \{A\} >$的子集，前缀为$< \{B\} >$的子集，…，前缀为$< \{F\} >$的子集。

3）通过构造对应投影数据的集合，递归挖掘序列模式的子集。过程如下：

①找到前缀为$< \{A\} >$的序列模式

构成A的投影数据库，包括四个后缀序列：$< \{A,B,C\},\{A,C\},\{D\},\{C,F\} >$，$< \{\_,D\},\{C\},\{B,C\},\{A,E\} >$，$< \{\_,B\},\{D,F\},\{C\},\{B\} >$，$< \{\_,F\},\{C,B,C\} >$。在投影数据库中，对$A$的后缀进行计数得到频繁项是：$\{A{:}2$，$B{:}4$，$\_B{:}2$，$C{:}4$，$D{:}2$，$F{:}2\}$。此时可以得到所有长度为 2 的以$< \{A\} >$的为前缀的序列模式：$\{< \{A\},\{A\} >{:}2, < \{A\},\{B\} >{:}4, < \{A,B\}{:}2 >, < \{A\},\{C\} >{:}4, < \{A\},\{D\} >{:}2, < \{A\},\{F\} >{:}2\}$。

所有前缀为$< \{A\} >$的序列模式划分为六个子集：前缀为$< \{A\},\{A\} >$的子集，前缀为$< \{A\},\{B\} >$的子集，…,前缀为$< \{A\},\{F\} >$的子集，分别构造其投影数据库进行递归挖掘，步骤如下：

$< \{A\},\{A\} >$的投影数据库由两个前缀为$< \{A\},\{A\} >$的非空（后缀）子序列组成：$< \{_,B,C\},\{A,C\},\{D\},\{C,F\} >$，$< \{_,E\} >$。由于不可能从这个投影数据库中生成任何频繁的子序列，所以对$< \{A\},\{A\} >$的投影数据库的处理终止。

$< \{A\},\{B\} >$的投影数据库由三个后缀序列组成：$< \{_,C\},\{A,C\},\{D\},\{C,F\} >$，$< \{_,C\},\{A\} >$，$< \{C\} >$。递归$< \{A\},\{B\} >$的投影数据库生成四个序列模式：$\{< \{_,C\} >, < \{_,C\},\{A\} >, < \{A\} >, < \{C\} >\}$。

$< \{A,B\} >$的投影数据库仅包含$< \{_C\},\{A,C\},\{D\},\{C,F\} >$和$< \{D,F\},\{C\},\{B\} >$这两个序列，这导致发现以$< \{A\},\{B\} >$为前缀的序列模式如下：$\{< \{C\} >, < \{D\} >, < \{F\} >, < \{D\},\{C\} >\}$。

以此类推，递归挖掘$< \{A\},\{C\} >$，$< \{A\},\{D\} >$，$< \{A\},\{F\} >$的投影数据库，得到对应的序列模式。

②继续找到以$< \{B\} >$，$< \{C\} >$，$< \{D\} >$以及$< \{F\} >$为前缀的序列模式。该过程与上一步一致，最终全部的序列模式是上述步骤结果的并集。

为了方便表示，将对序列的表达形式进行简化处理：①如果事件是单独发生，则无需用大括号将其包括，例如，$< \{A\} > \Rightarrow < A >$，$< \{A\},\{A\} > \Rightarrow < AA >$；

②如果事件是同时发生，则用小括号将其包括，例如，$<\{A,B,C\}>\Rightarrow<(ABC)>$，$<\{\_,C\}>\Rightarrow<\_C>$。序列模式的挖掘结果如表 4.27 所示。

表 4.27 序列模式

前缀	投影数据库（后缀）	序列模式
$<A>$	$<(ABC)(AC)\,D\,(CF)>$， $<(\_D)\,C\,(BC)\,(AE)>$， $<(\_B)\,(DF)\,CB>$，$<(\_F)\,CBC>$	$<A>$，$<AA>$，$<AB>$，$<A\,(BC)>$， $<A\,(BC)\,A>$，$<ABA>$，$<ABC>$， $<(AB)>$，$<(AB)\,C>$，$<(ABD)>$， $<(AB)\,F>$，$<((AB)\,DC>$，$<AC>$， $<ACA>$，$<ACB>$，$<ACC>$， $<AD>$，$<ADC>$，$<AF>$
$<B>$	$<(\_C)\,(AC)D\,(CF)>$， $<(\_C)\,(AE)>$， $<(DF)\,CB>$，$<C>$	$<B>$，$<BA>$，$<BC>$， $<(BC)>$，$<(BC)A>$，$<BD>$， $<BDC>$，$<BF>$
$<C>$	$<(AC)\,D\,(CF)>$，$<$ $(BC)\,(AE)>$，$<B>$，$<C>$	$<C>$， $<CA>$，$<CB>$，$<CC>$
$<D>$	$<(CF)>$，$<C\,(DF)\,(AE)>$， $<(\_F)\,CB>$	$<D>$， $<DB>$，$<DC>$，$<DCB>$
$<E>$	$<(\_F)\,(AB)\,(DF)\,CB>$， $<(AF)\,CBC>$	$<E>$，$<EA>$，$<EAB>$，$<EAC>$， $<EACB>$，$<EB>$，$<EBC>$， $<EC>$，$<ECB>$，$<EF>$，$<$ $EFB>$，$<EFC>$，$<EFCB>$
$<F>$	$<(AB)\,(DF)\,CB>$，$<CBC>$	$<F>$，$<FB>$，$<FBC>$，$<FC>$， $<FCB>$

通过以上的例子可以发现，PrefixSpan 算法不生成大量的候选序列以及不需要反复扫描数据库，和 AprioriAll 算法相比更快更有效，特别是在支持度比较低的情况下优势更加明显。

AprioriAll 算法在稀疏数据集的应用中比较合适，其优势在于使用较为简单，但不适合稠密数据集的应用。而 PrefixSpan 算法效率高，在稠密数据集的应用上优势明显，但实现难度较大。

4.5 本章小结

频繁模式挖掘是通过挖掘出数据集中的频繁模式来分析数据之间的关联性和相关性等。本章首先介绍频繁模式的基本概念；针对项集模式挖掘，详细阐述了三种频繁项集挖掘算法以及模式压缩和模式聚类；为了挖掘模式之间的关系，介绍了关联规则挖掘和模式评估指标；针对序列数据库，介绍了两种经典的序列模式挖掘算法。具体内容如下：

基本概念：包括频繁模式的概念，包括项、事务、数据库以及模式支持度。

频繁项集挖掘：包括频繁项集的基本概念，以及 Apriori 算法、FP-Growth 算法以及垂直数据结构算法的原理、算法流程和实例分析，并对挖掘出来的频繁项集进行模式压缩和模式聚类。

关联规则挖掘：包括关联规则的概念、产生和评估，通过条件概率形式表达的计算公式得到关联规则的产生，引入模式评估指标（提升度、卡方距离、全置信度、最大置信度、Kulczynski、余弦、不平衡比），通过对工业实例中的数据集分析比较不同度量之间的评估效果，并选择最为合适的评估指标。

序列模式挖掘：包括序列模式的概念，以及两种序列模式算法（AprioriAll 算法、PrefixSpan 算法），通过实例分析比较了两种算法的不同适用场合。

本章习题

1. 请对比 Apriori 算法和 FP-算法的异同，并给出两种算法的各自的优缺点。
2. 下表 4.1 给出一个事务数据库，假定支持度阈为 40%，请找出频繁闭项集。

题表 4.1 事务数据库$\mathbb{D}$

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

3. 下面的频繁 3-项集的集合：{*A, B, C*}, {*A, B, D*}, {*A, B, E*}, {*A, C, D*}, {*A, C, E*}, {*B, C, D*}, {*B, C, E*}, {*C, D, E*}。假定数据集中只有 5 个项。请使用合并策略，由候选产生过程得到的所有候选 4-项集，并列出由 Apriori 算法的候选产生过程得到的所有候选 4-项集以及剪枝操作后的所有候选 4-项集。
4. 对于下面每一个问题，请在工业领域举出一个满足下面条件的关联规则的例子。此外，指出这些规则是否为强关联规则。
   1）具有高支持度和高置信度的规则；
   2）具有相当高的支持度却有较低置信度的规则；
   3）具有低支持度和低置信度的规则；
   4）具有低支持度和高置信度的规则。
5. 对题表 4.2 的事务数据库$\mathbb{D}$，给出置信度为 70%，支持度为 50%。使用 Apriori 算法找出频繁项集并找出强关联规则。

题表 4.2 事务数据库$\mathbb{D}$

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

5	$\{B,E,D\}$

6. 假设有一个数据集，包含 100 个事务和 20 个项。其中 A 和 B 为其中的两项，假设 A 的支持度为 25%，B 的支持度为 90%，且$\{A,B\}$的支持度为 20%。令最小支持度阈值和最小置信度阈值分别为 10%和 60%。计算关联规则 $\{A\} \rightarrow \{B\}$的置信度。根据置信度，这条规则为强关联规则吗？
7. 设最小支持度阈值为 33.3%，最小置信度阈值为 50%。按照 Apriori 算法的步骤，给出每次扫描题表 4.3 中的事务数据库后得到的所有频繁项集。在频繁项集的基础上，产生所有的强关联规则。

题表 4.3 事务数据库$\mathbb{D}$

TID	**事务**
1	$\{A,B,C,D,E\}$
2	$\{A,B,D,E\}$
3	$\{B,C,D\}$
4	$\{C,D,E\}$
5	$\{A,C,E\}$
6	$\{A,B,D\}$

8. 如题表 4.4 所示的事务数据库$\mathbb{D}$有 5 个报警事务，设支持度=60%，给出置信度为 80%，最小支持度为 60%。分别使用 Apriori 算法、FP-Growth 算法以及垂直数据结构算法找出所有的频繁项集并找出满足$\{X,Y\} \rightarrow Z$形式的强关联规则。

表 4.4 事务数据库$\mathbb{D}$

TID	**事务**
1	$\{D,F,G,H,I,K\}$
2	$\{C,D,F,H,I,K\}$
3	$\{A,D,F,G\}$
4	$\{B,F,G,J,K\}$
5	$\{B,D,E,F,I\}$

9. 阅读 PrefixSpan 算法的参考文献，思考当序列数据集很大，不同的项数又较多，且每个序列都需要建立一个投影数据库时，如何减少投影数据库的数量和大小。
10. 针对题表 4.5 中的工业实际的 8 个数据集，请对其计算提升度、卡方距离度、全置信度、最大置信度、Kulczynski、余弦模式和不平衡比等模式评估度量指标并对结果进行分析。

题表 4.5 工业报警事件数据库列联表

数据集	TP	¬TP	T(¬P)	¬(TP)
E_1	8123	83	424	1370
E_2	8330	2	622	1046
E_3	3954	3080	5	2961
E_4	2886	1363	1320	4431
E_5	1500	2000	500	6000
E_6	4000	2000	1000	3000
E_7	9481	298	127	94
E_8	4000	2000	2000	2000

参考文献

[1] 陈封能.数据挖掘导论. 北京:机械工业出版社, 2019.7.

[2] 石胜飞. 大数据分析与挖掘. 北京: 人民邮电出版社, 2018.8.

[3] 田春华. 工业大数据分析实战. 北京：电子工业出版社，2021.2.

[4] 韩家炜.数据挖掘：技术与概念.北京：机械工业出版社. 2012.7.

[5] R. S. Michalski, G. Tecuci. Machine learning: A multistrategy approach[M]. *Morgan Kaufmann*, 1994.

[6] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, andA. I. Verkamo. Fast discovery of association rules[J]. *Advances in Knowledge Discovery and Data Mining*, 1996, 12(1): 307-328.

[7] J, Han, J, Pei, and Y. Yin. Mining frequent patterns without candidate generation[J]. *ACM Sigmod Record*, 2000, 29(2): 1-12.

[8] M. J. Zaki. Scalable algorithms for association mining[J]. *IEEE Transactions on Knowledge and Data Engineering*, 2000, 12(3): 372-390.

[9] G. Piatetsky-Shapiro. Notes of AAAI'91 workshop knowledge discovery in databases (KDD'91)[J]. *Anaheim, CA*, July, 1991.

[10]M. S. Chen, J. Han, andP. S. Yu. Data mining: an overview from a database perspective[J]. *IEEE Transactions on Knowledge and data Engineering*, 1996, 8(6): 866-883.

[11]K. M. Ahmed, M. Nagwa. El-Makky, Yousry Taha: A note on "Beyond Market Baskets: Generalizing Association Rules to Correlations"[J]. *The Proceedings of SIGKDD Explorations*, 2000, 1(2): 46-48.

[12]E. R. Omiecinski. Alternative interest measures for mining associations in databases[J]. *IEEE Transactions on Knowledge and Data Engineering*, 2003, 15(1): 57-69.

[13]T. Wu, Y. Chen, and J. Han. Re-examination of interestingness measures in pattern mining: a unified framework[J]. *Data Mining and Knowledge Discovery*, 2010, 21(3): 371-397.

[14]D. Apiletti, E. Baralis, T. Cerquitelli, P. Garza, F. Pulvirenti, and L. Venturini. Frequent itemsets mining for big data: A comparative analysis[J].*Big Data Research*,2017,9: 67-83.

[15]W. Hu, T. Chen and S. L. Shah. Detection of frequent alarm patterns in industrial alarm floods using itemset mining methods[J]. *IEEE Transactions on Industrial Electronics*, 2018,65(9):7290-7300, 2018.

[16]B. Zhou, W. Hu and T. Chen. Pattern extraction from industrial alarm flood sequences by a modified CloFAST algorithm[J]. *IEEE Trans. Industrial Informatics*, 2022, 18(1): 288-296.