频繁模式挖掘

目录
CONTENTS

Chapter 8.1

频繁模式概述

关联规则

- 数据关联规则是数据挖掘算法中使用较早的一种数据分析方法，用于在大数据中挖掘出具有价值的信息，通常在商业中用数据与数据之间的关系来产生更大的价值。

1. 关联规则案例

- 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，如果两个事物或者多个事物之间存在一定的关联关系，那么其中一个事物就能够通过其他事物预测到。典型的例子就是“啤酒与纸尿裤”。

“纸尿裤”和“啤酒”的故事

发现这两件商品存在联系-分析-美国的太太们常叮嘱她们的丈夫下班后为小孩买尿布，而丈夫们在买尿布后又随手带回了两瓶啤酒。这一消费行为导致了这两件商品经常被同时购买.
所以，沃尔玛索性就将纸尿裤和啤酒并排摆在一起销售，结果销量双双增长！

----------“啤酒和草莓酱吐司饼干”的故事

沃尔玛很好地运用了数据仓库、数据挖掘和数据分析的技术。

大数据挖掘的魅力

目前约2.45亿顾客光顾沃尔玛遍布全球的一万九百多家分店，沃尔玛每小时收集来自一百多万客户的2.5TB非结构化数据，来自社交媒体的数据约每周30万条。沃尔玛的大数据生态系统每天处理的新数据高达几个TB，历史数据则是PB级；分析的商品数量达数百万，客户信息过亿。

最关键的是，沃尔玛成功地将数据价值转换为商业价值，其中，线上销售收入增长10%到15%，约10亿美元。

为什么啤酒和纸尿裤的销量会同时增加？

是因为啤酒还是纸尿裤?

纸尿裤 → 啤酒

关联?

如何寻找这种潜在的关联?

购物车分析

- 频繁模式是指频繁出现在数据集中的模式，这些模式包括项集、子序列和子结构等。
- 研究频繁模式的目的是得到关联规则和其他的联系，并在实际中应用这些规则和联系。

面包和牛奶共同出现在购物车中，这代表了什么?

购买了油、牛奶、面包、香蕉、洗衣液、还应该有哪些商品?

买了这么多的鱼子酱，是因为促销吗?

购买了油、牛奶、面包、香蕉、葡萄、洗衣液，还应该有哪些商品?

上图能挖掘出哪些有趣的模式?

例8.1 购物车分析

表8-1 某商店的事务数据

TID	Items
1	牛奶，面包，麦片
2	牛奶，面包，麦片，鸡蛋
3	牛奶，面包，黄油，麦片
4	糖，鸡蛋
5	黄油，麦片
6	糖，鸡蛋

2. 相关概念

表8-1 某商店的事务数据

TID	Items
1	牛奶，面包，麦片
2	牛奶，面包，麦片，鸡蛋
3	牛奶，面包，黄油，麦片
4	糖，鸡蛋
5	黄油，麦片
6	糖，鸡蛋

6个事务

A：项
ABC：项集
AB：2项集
ABC：3项集

一行：事务/记录
全部行：事务集

项集：包含0个或者多个项的集合

- {牛奶、面包}是一个2项集，其中牛奶、面包各是1个项。
- {牛奶、面包、麦片}、{牛奶、面包}、{牛奶、麦片}、{面包、麦片}项集出现了3次；{糖、鸡蛋}、{黄油、麦片}项集出现了2次；其它项集只出现1次。
- 如果出现3次的项集为频繁项集，可以得出："一个购买了面包的用户很可能购买牛奶"这样的关联规则。

– 关联规则

- 设$I=\{i_1,i_2,i_3,\ldots,i_n\}$是事务数据中所有项的集合，$T=\{t_1,t_2,t_3,\ldots,t_n\}$是所有事务的集合，其中每个事务$t_i$都有一个独一无二的标识符*TID*。
- **关联规则**是形如$A \Rightarrow B$的**蕴含式**，其中A称为规则**前件**，B称为规则**后件**，并且A，B满足：A，B是I的真子集，并且A和B的交集为空集。

表8-1 某商店的事务数据

TID	Items
1	牛奶，面包，麦片
2	牛奶，面包，麦片，鸡蛋
3	牛奶，面包，黄油，麦片
4	糖，鸡蛋
5	黄油，麦片
6	糖，鸡蛋

- **支持度**是指事务中同时包含集合A和集合B的百分比：

$$support(A \Rightarrow B)=P(A \cup B)=\frac{\text{AB出现的次数}}{\text{总的次数}}$$

★ $support(B \Rightarrow A)=P(A \cup B)$

- **置信度**是指事务中同时包含集合A与B的事务数与包含集合A的事务数的百分比：

$$confidence(A \Rightarrow B)=P(B|A)=\frac{\text{AB出现的次数}}{\text{A出现的次数}}$$

– 频繁项集和强关联规则

- 满足最小支持度阈值的项集被称为**频繁项集**。
- 同时满足最小支持度和最小置信度阈值要求的所有关联规则被称为**强关联规则**。

例8.6 强关联规则

- 假设最小支持度阈值为30%，最小置信度阈值为70%，而关联规则：

购买面包⇒购买牛奶 { 支持度=3÷6=50%
置信度=3÷3=100%

- 支持度和置信度都满足条件，则该规则为强关联规则。
- 因为满足最小支持度阈值，也可以说{面包、牛奶}是频繁项集。

TID	Items
1	牛奶，面包，麦片
2	牛奶，面包，麦片，鸡蛋
3	牛奶，面包，黄油，麦片
4	糖，鸡蛋
5	黄油，麦片
6	糖，鸡蛋

表 9-3 5 笔交易数据

交易数据编号	数据项
t1	A、B、C、D
t2	A、B
t3	C、D、E
t4	B、C
t5	B、C、E

表 9-4　四个规则的支持度和置信度

关联规则	支持度 s	置信度 c
A=>B		
B=>A		
B=>C		
C=>B		

3. 先验性质

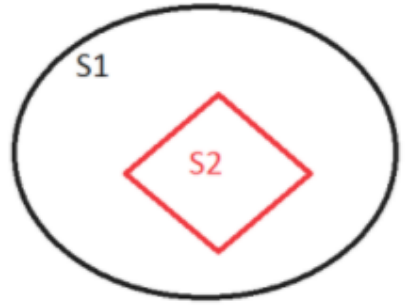
– 关联规则挖掘的任务

- 根据最小支持度阈值，找出数据集中所有的频繁项集；
- 挖掘出频繁项集中满足最小支持度和最小置信度阈值要求的规则，得到强关联规则；
- 对产生的强关联规则进行剪枝，找出有用的关联规则。

– 先验性质

①如果一个项集是频繁的，那么它的所有非空子集也是频繁的。

②如果一个项集是非频繁的，那么它的所有真超项集也是非频繁的。



★如果一个集合S2中的每一个元素都在集合S1中，且集合S1中可能包含S2中没有的元素，则集合S1就是S2的一个超集，反过来，S2是S1的子集。
★S1是S2的超集，若S1中一定有S2中没有的元素，则S1是S2的真超集，反过来S2是S1的真子集。

Chapter 8.2

Apriori算法

关联规则挖掘的步骤

① 找出所有**频繁项集**，即大于或等于最小支持度阈值的项集。

② 由频繁项集产生**强关联规则**，这些强关联规则必须大于或等于最小支持度阈值和最小置信度阈值。

③ 对强关联规则进行**剪枝**，找出**有用的关联规则**。

频繁项集→强关联规则→有用的关联规则

计算复杂度：第①步的开销远大于第②步的开销，因此降低第①步的计算复杂度更重要。

Apriori如何找所有频繁项集的分析

- 是布尔关联规则挖掘频繁项集的原创性算法，算法使用**频繁项集的先验性质**。
- 使用一种称为**逐层搜索的迭代方法**，其中k项集用于搜索(k+1)项集。
- 通过扫描数据库，累计每个1项集的个数，并收集满足最小支持度的1项集，从而得到频繁1项集的集合，该集合记为L_1；
- 使用L_1找出频繁2项集的集合L_2，使用L_2找出L_3；
- 如此下去，直到不能再找到频繁k项集。
- 找出每一个L_K需要**一次数据库的完整扫描**。

Apriori如何找所有频繁项集的实现步骤

- **①连接**
  - 算法初始设置k=1，使用**连接运算**从数据库中找到所有的候选1项集的集合C_1，然后k增加1，直到k等于频繁项集的极大长度或频繁项集为空。
  - 构造**候选k项集的集合**C_k（k不等于1）时，使用前一次迭代中的所有**频繁k-1项集的集合**L_{k-1}的自身元素相互连接来构成，即$C_k = L_{k-1} \bowtie L_{k-1}$。
  - **连接方式：**两个可连接的元素的前k-2个项是相同的，即只有最后一项不同，这样可**简单地确保不产生重复**。

L_3:		L_3:		C_4:
{i1，i2，i3}		{i1，i2，i3}		
{i1，i2，i4}		{i1，i2，i4}		{i1，i2，i3，i4}
{i1，i3，i4}	$\bowtie$	{i1，i3，i4}	=	{i1，i2，i3，i5}
{i1，i3，i5}		{i1，i3，i5}		
{i2，i3，i4}		{i2，i3，i4}		

– **②剪枝**

- 按照**先验性质**对得到的k项候选集的集合C_k进行剪枝，以减小因C_k较大而产生较大的计算量。
- **先验性质**指出如果某一k-1项集的支持度小于最小支持度阈值，那么它的所有真超项集的支持度都小于最小支持度阈值，所以该k-1项集和它的所有真超项集都不是频繁项集，从而可以在C_k中剪掉该k-1项集的真超k项集，在以后的迭代步骤中不予考虑。
- 然后可以根据最小支持度阈值，在剪枝后的候选k项集的集合C_k中剪掉支持度小于最小支持度阈值的k项集，生成频繁k项集的集合L_k 。

L_3:
{i1，i2，i3}
{i1，i2，i4}
{i1，i3，i4}
{i1，i3，i5}
{i2，i3，i4}

C_4：{i1，i2，i3，i4}
{i1，i2，i3，i5}

剪枝 →

C_4：{i1，i2，i3，i4}

- {i1，i2，i3，i4}的3项真子集{i1，i2，i3}、{i1，i2，i4}、{i2，i3，i4}、{i1，i3，i4}都属于L_3，是频繁的。
- {i1，i2，i3，i5}的2项真子集{i1，i2，i5}、{i2，i3，i5}都不属于L_3，不是频繁的。

Apriori如何找所有频繁项集的算法：

输入：项集I，事务数据集D，最小支持度计数阈值Min_sup

输出：D中的所有频繁项集的集合L。

Apriori算法：

（1）求频繁1项集L_1

- 首先通过扫描事务数据集D，找出所有1项集并计算其支持度，作为候选1项集的集合C_1
- 然后从C_1中删除低于最小支持度阈值Min_sup的项集，得到所有频繁1项集的集合L_1

（2）For k=2,3,4，…

（3）连接：将L_{k-1}进行自身连接生成候选k项集的集合C_k，连接方法如下：对于任意$p,q \in L_{k-1}$，若按字典序有$p=\{p_1,p_2,\ldots,p_{k-2},p_{k-1}\}$，$q=\{p_1,p_2,\ldots,p_{k-2},q_{k-1}\}$，且满足$p_{k-1}<q_{k-1}$，则把$p$和$q$连接成$k$项集$\{p_1,p_2,\ldots,p_{k-2},p_{k-1},q_{k-1}\}$作为候选k项集$C_k$中的元素。

Apriori如何找所有频繁项集的算法：

（4）剪枝：删除C_k中的非频繁k项集，即当C_k中一个候选k项集的某个k-1项子集不是L_{k-1}中的元素时，则将它从C_k中删除。（根据先验性质）

（5）计算支持数：通过扫描事务数据集D，计算C_k中每个k项集的支持数。

（6）求L_k：删除C_k中低于最小支持度阈值Min_sup的k项集，得到所有频繁k项集的集合L_k。

（7）若$L_k=\varnothing$，则转第（9）步

（8）END FOR

（9）另$L=L_1 \cup L_2 \cup \ldots \cup L_k$，并输出$L$。

测试数据集

表8-3

TID	Items
1	面包、可乐、麦片
2	牛奶、可乐
3	牛奶、面包、麦片
4	牛奶、可乐
5	面包、鸡蛋、麦片
6	牛奶、面包、可乐
7	牛奶、面包、鸡蛋、麦片
8	牛奶、面包、可乐
9	面包、可乐

例8.7 Apriori算法

假设使用表中的事务数据，该数据库具有9个事务，设最小支持度为2÷9（最小支持度计数为2），试使用Apriori算法挖掘表8-3的事务数据中的频繁项集。

8.2 Apriori算法

找到候选1项集的集合C_1

C_1

项集	支持度计数
牛奶	6
面包	7
可乐	6
鸡蛋	2
麦片	4

生成频繁1项集的集合L_1

L_1

项集	支持度计数
牛奶	6
面包	7
可乐	6
鸡蛋	2
麦片	4

全连接操作：两两组合

连接成候选2项集的集合C_2

无需剪枝

C_2

项集	支持度计数
牛奶，面包	4
牛奶，可乐	4
牛奶，鸡蛋	1
牛奶，麦片	2
面包，可乐	4
面包，鸡蛋	2
面包，麦片	4
可乐，鸡蛋	0
可乐，麦片	1
鸡蛋，麦片	2

生成频繁2项集的集合L_2

L_2

项集	支持度计数
牛奶，面包	4
牛奶，可乐	4
牛奶，麦片	2
面包，可乐	4
面包，鸡蛋	2
面包，麦片	4
鸡蛋，麦片	2

连接成候选3项集的集合C_3

—

项集
牛奶，面包，可乐
牛奶，面包，麦片
牛奶，可乐，麦片
面包，鸡蛋，可乐
面包，鸡蛋，麦片
面包，可乐，麦片

剪枝后的候选3项集的集合C_3

C_3

项集	支持度计数
牛奶，面包，可乐	2
牛奶，面包，麦片	2
面包，鸡蛋，麦片	2

生成频繁3项集的集合L_3

L_3

项集	支持度计数
牛奶，面包，可乐	2
牛奶，面包，麦片	2
面包，鸡蛋，麦片	2

C_4?

TID	Items
1	面包、可乐、麦片
2	牛奶、可乐
3	牛奶、面包、麦片
4	牛奶、可乐
5	面包、鸡蛋、麦片
6	牛奶、面包、可乐
7	牛奶、面包、鸡蛋、麦片
8	牛奶、面包、可乐
9	面包、可乐

所有频繁项集的集合$L(=L_1 \cup L_2 \cup L_3)$:

L={{牛奶}：6、{面包}：7、{可乐}：6、{鸡蛋}：2、{麦片}：4、{牛奶，面包}：4、{牛奶，可乐}：4、{牛奶，麦片}：2、{面包，可乐}：4、{面包，鸡蛋}：2、{面包，麦片}：4、{鸡蛋，麦片}：2、{牛奶，面包，可乐}：2、{牛奶，面包，麦片}：2、{面包，鸡蛋，麦片}：2}

Apriori如何由频繁项集产生强关联规则

迭代停止条件：

- C_4 = {牛奶、面包、可乐、麦片}，因为其中子集{面包、可乐、麦片}不在L3中，因此其真超集不是频繁项集，将其剪枝。
- 题目9件事务中只有1件含有最大项集，即4项集，支持度只有九分之一，肯定不是频繁项集，将其剪枝。

- 关联规则的生成过程包括两个步骤：

  ①对于L中的每个频繁项集X，生成X所有的非空真子集Y；

  ②对于X中的每一个非空真子集Y，构造关联规则$Y \Rightarrow (X - Y)$。

- 构造出关联规则后，计算每一个关联规则的置信度，如果大于最小置信度阈值，则该规则为强关联规则。

Apriori如何由频繁项集产生强关联规则举例

TID	Items
1	面包、可乐、麦片
2	牛奶、可乐
3	牛奶、面包、麦片
4	牛奶、可乐
5	面包、鸡蛋、麦片
6	牛奶、面包、可乐
7	牛奶、面包、鸡蛋、麦片
8	牛奶、面包、可乐
9	面包、可乐

对于上例6中L中的频繁3项集{牛奶，面包，麦片}，可以推导出非空子集：

1项集：{{牛奶}，{面包}，{麦片}。

2项集：{牛奶，面包}，{牛奶，麦片}，{面包，麦片}}。

可以构造的关联规则及置信度如下：

{牛奶} ⇒ {面包，麦片}，置信度=2/6=33%

{面包} ⇒{牛奶，麦片}，置信度=2/7=29%

{麦片} ⇒ {牛奶，面包}，置信度=2/4=50%

{牛奶，面包} ⇒ {麦片}，置信度=2/4=50%

{牛奶，麦片} ⇒ {面包}，置信度=2/2=100%

{面包，麦片} ⇒ {牛奶}，置信度=2/4=50%

令最小置信度为70%，则得到的强关联规则有：

{牛奶，麦片} ⇒ {面包}，置信度=2/2=100%

Apriori算法举例

【例 】 下表是交易数据集，假设最小支持度为50%，最小置信度为50%，通过Apriori算法求关联规则。

交易数据编号	商品项
t1	牛奶、面包、鸡蛋、啤酒
t2	面包、鸡蛋、啤酒
t3	牛奶、鸡蛋、黄油
t4	面包、啤酒

Apriori 算法的计算过程

交易数据编号	商品项
t1	牛奶、面包、鸡蛋、啤酒
t2	面包、鸡蛋、啤酒
t3	牛奶、鸡蛋、黄油
t4	面包、啤酒

1-项集

L1	项集	支持度
	{牛奶}	50%
	{面包}	75%
	{鸡蛋}	75%
	{黄油}	25%
	{啤酒}	75%

<50%

K1	项集	支持度
	{牛奶}	50%
	{面包}	75%
	{鸡蛋}	75%
	{啤酒}	75%

L2	项集	支持度
	{牛奶，面包}	25%
	{牛奶，鸡蛋}	50%
	{牛奶，啤酒}	25%
	{面包，鸡蛋}	50%
	{面包，啤酒}	75%
	{鸡蛋，啤酒}	50%

<50%

<50%

2-项集

K2	项集	支持度
	{牛奶，鸡蛋}	50%
	{面包，鸡蛋}	50%
	{面包，啤酒}	75%
	{鸡蛋，啤酒}	50%

L3	项集	支持度
	{牛奶，面包，鸡蛋}	25%
	{牛奶，鸡蛋，啤酒}	25%
	{面包，鸡蛋，啤酒}	50%

<50%

3-项集

K3	{面包，鸡蛋，啤酒}	50%

最后只有一个频繁项集 K3：{面包、鸡蛋、啤酒}，算法到此结束。

对于频繁项集{面包、鸡蛋、啤酒}，它的非空真子集有：

{面包}、{鸡蛋}、{啤酒}

{面包、鸡蛋}、{面包、啤酒}、{鸡蛋、啤酒}

据此生成关联规则并计算其置信度，结果见下表。

规则	置信度
{面包}—{鸡蛋，啤酒}	0.50/0.75=66.70%
{鸡蛋}—{面包，啤酒}	0.50/0.75=66.70%
{啤酒}—{面包，鸡蛋}	0.50/0.75=66.70%
{鸡蛋，啤酒}—{面包}	0.50/0.50=1
{面包，啤酒}—{鸡蛋}	0.50/0.75=66.70%
{面包，鸡蛋}—{啤酒}	0.50/0.50=1

结果可信？

置信度计算结果

规则	置信度
{面包}—{鸡蛋，啤酒}	0.50/0.75=66.70%
{鸡蛋}—{面包，啤酒}	0.50/0.75=66.70%
{啤酒}—{面包，鸡蛋}	0.50/0.75=66.70%
{鸡蛋，啤酒}—{面包}	0.50/0.50=1
{面包，啤酒}—{鸡蛋}	0.50/0.75=66.70%
{面包，鸡蛋}—{啤酒}	0.50/0.50=1

>50%

- 从上表可以看到，置信度值都超过了阈值 50%，所以，{面包、鸡蛋、啤酒}就是最终得到的关联规则。
- Python的第三方库已经实现了Apriori算法，可以使用pip install apyori命令安装库apyori。

Apriori如何由频繁项集产生强关联规则复杂度

- 可以通过**枚举**频繁项集生成所有的关联规则，并通过计算关联规则的置信度来判断该规则是否为强关联规则。

- 但当一个频繁项集包含的**项很多**时，就会生成大量的候选关联规则，一个频繁X项集能够生成$2^{|X|}-2$个（即除去空集及自身之外的子集）候选关联规则。

- 上例中X=3，$2^{|X|}-2=6$。

Apriori如何由频繁项集产生强关联规则的降复杂度方法

- 可以逐层生成关联规则，并利用如下关联规则的性质进行剪枝，以减少关联规则生成的计算工作量。

- 关联规则性质：设X为频繁项集，Ø≠Y⊂X且Ø≠Y′ ⊂Y。若Y ⇒ $(X - Y)$不是强关联规则，则Y′ ⇒ $(X - Y')$也不是强关联规则。

- 首先产生后件（被support的集合）只包含1个项的关联规则，然后两两合并关联规则的后件，生成后件包含2个项的候选关联规则，从这些候选关联规则中再找出强关联规则，以此类推。

TID	Items
1	面包、可乐、麦片
2	牛奶、可乐
3	牛奶、面包、麦片
4	牛奶、可乐
5	面包、鸡蛋、麦片
6	牛奶、面包、可乐
7	牛奶、面包、鸡蛋、麦片
8	牛奶、面包、可乐
9	面包、可乐

对于上例6中L中的频繁3项集{牛奶，面包，麦片}，可以推导出非空子集：

{{牛奶}，{面包}，{麦片}，{牛奶，面包}，{牛奶，麦片}，{面包，麦片}}。

可以构造的后件只包含一个项的关联规则及置信度如下：

{牛奶，面包} ⇒{麦片}，置信度=2/4=50%

{牛奶，麦片} ⇒ {面包}，置信度=2/2=100%

{面包，麦片} ⇒ {牛奶}，置信度=2/4=50%

令最小置信度为70%，则得到的强关联规则有：

{牛奶，麦片} ⇒ {面包}，置信度=2/2=100%

生成后件包含两个项的关联规则及置信度如下：

{麦片} ⇒ {牛奶，面包}，置信度=2/4=50%

{牛奶} ⇒ {麦片，面包}，置信度=2/4=50%

★根据前面的结论，{牛奶，面包} ⇒{麦片}、{面包，麦片} ⇒ {牛奶}生成后件包含两个项的关联规则及置信度不用计算了。

关联规则挖掘中重要的基础理论：

1.频繁项集的性质：

①如果X是频繁项集，则它的任何非空子集X′ 也是频繁项集。即**频繁项集的子集也是频繁项集**。

②如果X是非频繁项集，则它的所有真超集都是非频繁项集。即**非频繁项集的超集也是非频繁项集**。

★用于找出频繁项时的剪枝！

关联规则挖掘中重要的基础理论：

2. 关联规则的性质：

①设X为频繁项集，Ø≠Y⊂X且Ø≠Y' ⊂Y。若$Y' \Rightarrow (X - Y')$为强关联规则，

则Y $\Rightarrow (X - Y)$也是强关联规则。

②设X为频繁项集，Ø≠Y⊂X且Ø≠Y' ⊂Y。若Y $\Rightarrow (X - Y)$不是强关联规则，

则$Y' \Rightarrow (X - Y')$也不是强关联规则。

★用于产生强关联规则的剪枝！

Chapter 8.3

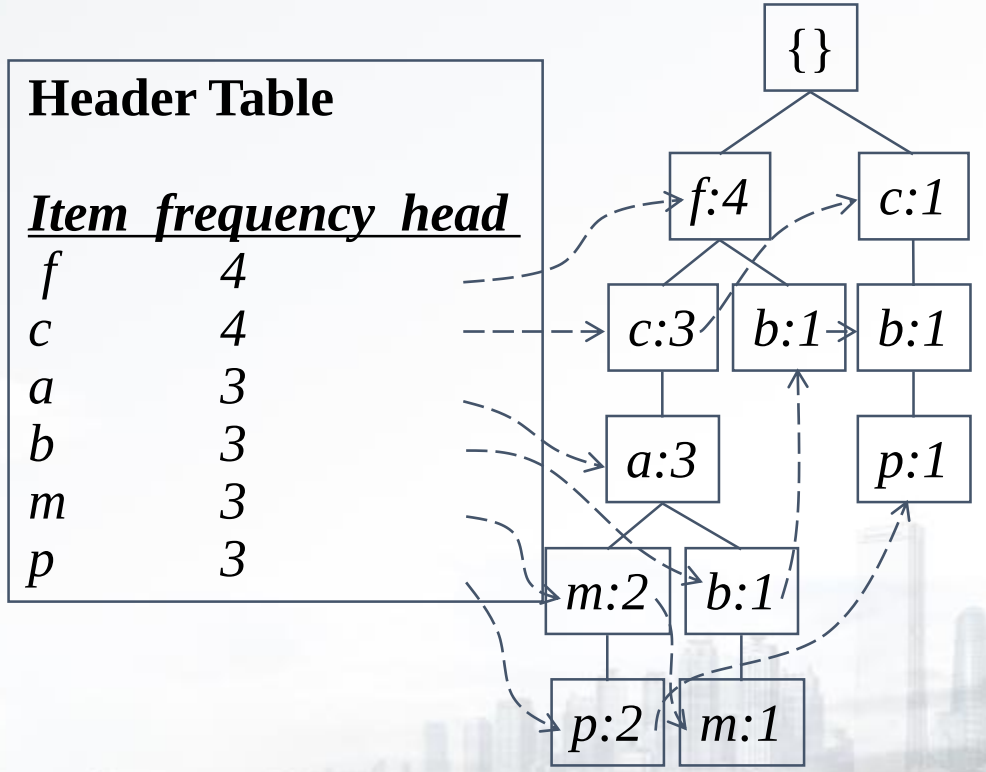
FP-growth算法

Apriori算法的优缺点

- **优点**
  - 算法原理简单，易于理解。
- **缺点：**
  - 需要多次扫描数据集
    - 如果频繁项集最多包含10个项，需要扫描事务数据集10次，这需要很大的I/O负载
  - 产生大量频繁项集
    - 如数据集有100项，可能产生的候选项个数为1.27×10^{30}

频繁模式增长(frequent-pattern growth FP-Growth)

- 将提供频繁项集的数据库压缩到FP-树，但仍保持项集关联信息;
- 压缩后的数据库分成一组条件数据库，每个数据库关联一个频繁项，并分别挖掘每个条件数据库;



FP-growth算法实现步骤

①第一次扫描事务数据集D，确定每个1项集的支持度计数，将频繁1项集按照支持度计数降序排序，得到排序后的频繁1项集集合L_1。

②第二次扫描事务数据集D，读出每个事务并**构建根结点为null的FP-tree**。

i. 创建FP-tree的根结点，用null标记；

ii. 根据L_1中的元素，**删除**事务数据集D中每个事务的**非频繁项**，并按照L_1中元素的顺序**重新排列每个事务中元素的顺序**；并对每个事务**创建一个分支**。

iii. 当为一个事务考虑增加分支时，沿**共同前缀**上的每个结点的**计数加1**，为跟随前缀后的项**创建结点并连接**；

iv. 创建一个**项头表**，以方便遍历，每个项通过一个结点链指向它在树中的位置，多次出现该项，都需要链接。

挖掘频繁项集

③从1项集的频繁项集中支持度最低的项开始，从项头表的结点链头指针，沿循每个频繁项的链接来遍历FP-tree，找出**该频繁项的所有前缀路径**，构造该**频繁项的条件模式基**，并**计算这些条件模式基中每一项的支持度**；

④通过条件模式基构造**条件FP-tree**，**删除**其中前缀路径中支持度低于最小支持度阈值的部分,再对剩下的元素与单个元素两两组合**构建新的条件FP树**，满足最小支持度阈值的部分则是频繁项集；

⑤递归地挖掘每个条件FP-tree，直到找到FP-tree为空或者FP-tree只有一条路径，该路径上的所有项的组合都是频繁项集。

测试数据集

TID	Items
1	面包、可乐、麦片
2	牛奶、可乐
3	牛奶、面包、麦片
4	牛奶、可乐
5	面包、鸡蛋、麦片
6	牛奶、面包、可乐
7	牛奶、面包、鸡蛋、麦片
8	牛奶、面包、可乐
9	面包、可乐

例8.8 FP-growth算法

该数据集具有9个事务，设最小支持度为2，频繁项集的极大长度为3。试使用FP-growth算法挖掘左表的事务数据中的频繁项集。

读出每个事务并构建根结点为null的FP-tree：

表 6-4 更新后的某商店的详细事务数据

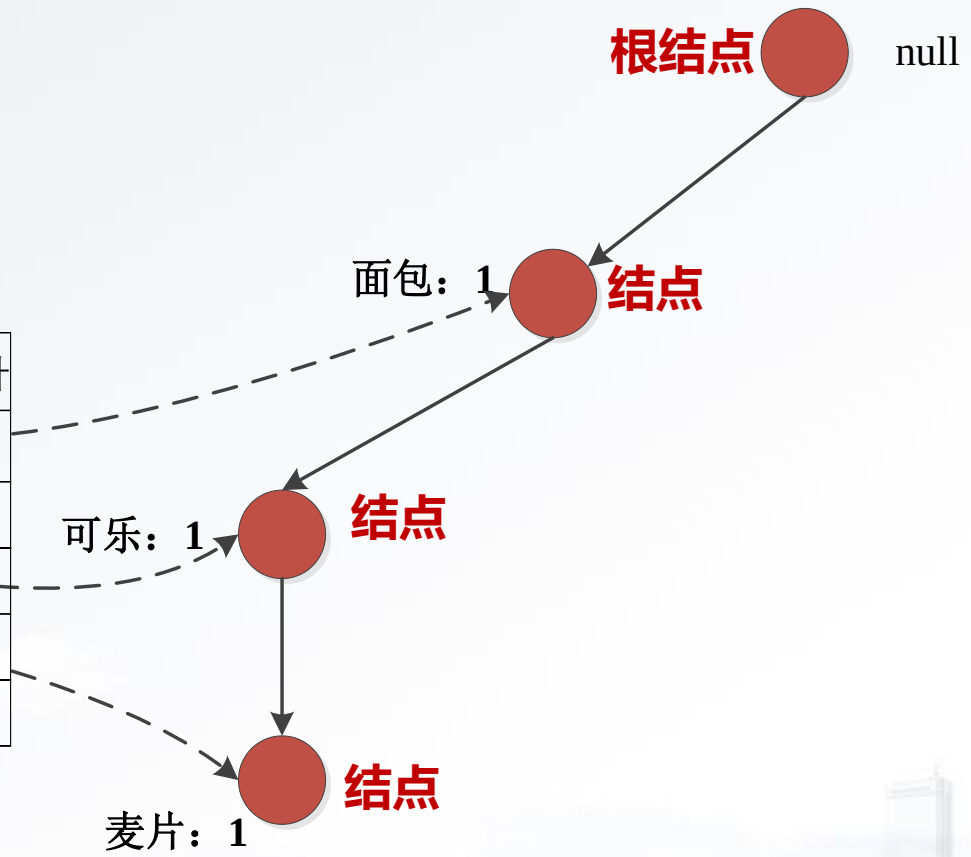
TID	Items	Sorted Items
√ 1	面包，可乐，麦片	面包，可乐，麦片
2	牛奶，可乐	牛奶，可乐
3	牛奶，面包，麦片	面包，牛奶，麦片
4	牛奶，可乐	牛奶，可乐
5	面包，鸡蛋，麦片	面包，麦片，鸡蛋
6	牛奶，面包，可乐	面包，牛奶，可乐
7	牛奶，面包，鸡蛋，麦片	面包，牛奶，麦片，鸡蛋
8	牛奶，面包，可乐	面包，牛奶，可乐
9	面包，可乐	面包，可乐

重新排列每个事务中元素的顺序

项头表

支持度计数降序排序

项集	支持度计数	结点链头指针
面包	7	
牛奶	6	
可乐	6	
麦片	4	
鸡蛋	2	



1项集支持度计数的大小决定了在树中的顺序

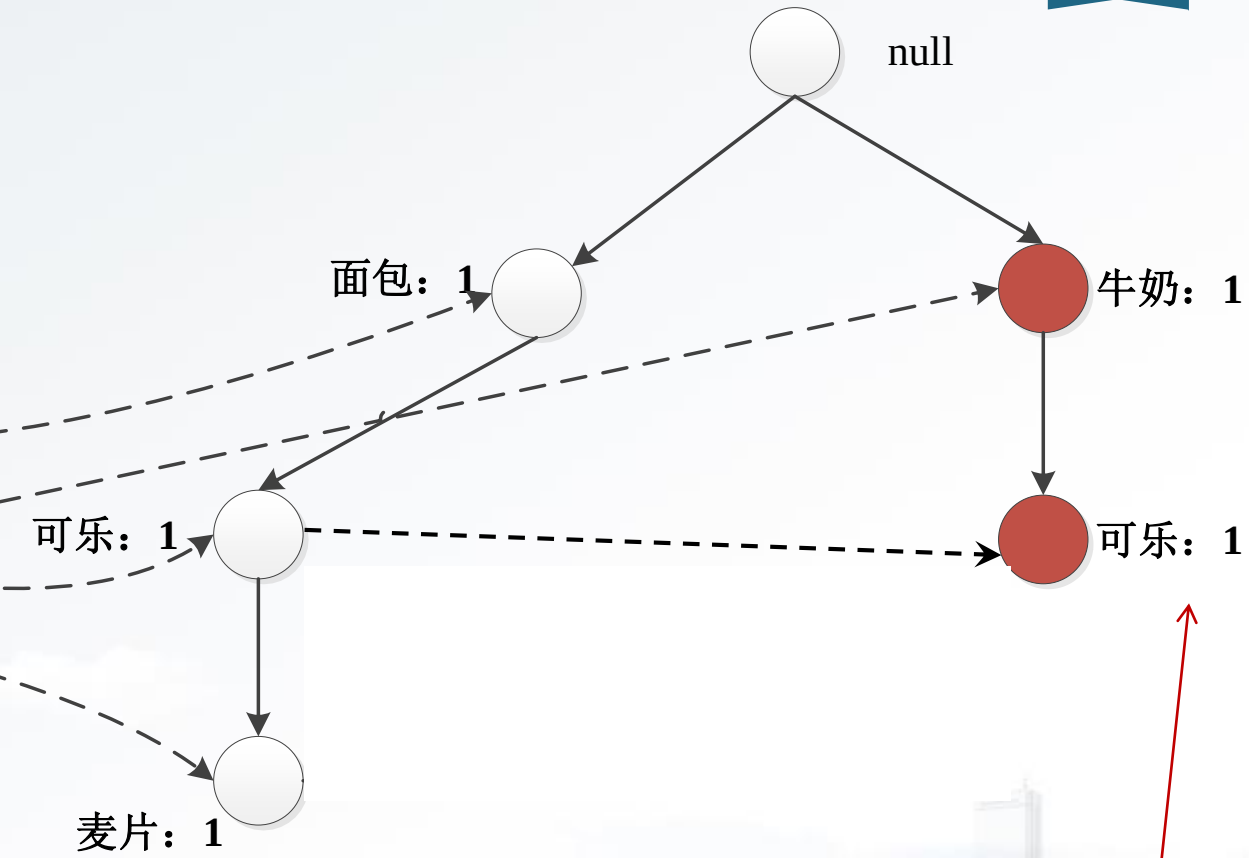
图8-11(1) 处理TID=1后的FP-tree

读出每个事务并构建根结点为null的FP-tree：

表 6-4 更新后的某商店的详细事务数据

TID	Items	Sorted Items
1	面包，可乐，麦片	面包，可乐，麦片
√ 2	牛奶，可乐	牛奶，可乐
3	牛奶，面包，麦片	面包，牛奶，麦片
4	牛奶，可乐	牛奶，可乐
5	面包，鸡蛋，麦片	面包，麦片，鸡蛋
6	牛奶，面包，可乐	面包，牛奶，可乐
7	牛奶，面包，鸡蛋，麦片	面包，牛奶，麦片，鸡蛋
8	牛奶，面包，可乐	面包，牛奶，可乐
9	面包，可乐	面包，可乐

项集	支持度计数	结点链头指针
面包	7	
牛奶	6	
可乐	6	
麦片	4	
鸡蛋	2	



没有共同前缀，因此产生新的分支。

结点链头指针指向该项在树中的第一个位置，再指向第二个位置，如图中可乐

图8-11(2) 处理TID=2后的FP-tree

读出每个事务并构建根结点为null的FP-tree：

表 6-4 更新后的某商店的详细事务数据

TID	Items	Sorted Items
1	面包，可乐，麦片	面包，可乐，麦片
2	牛奶，可乐	牛奶，可乐
√ 3	牛奶，面包，麦片	面包，牛奶，麦片
4	牛奶，可乐	牛奶，可乐
5	面包，鸡蛋，麦片	面包，麦片，鸡蛋
6	牛奶，面包，可乐	面包，牛奶，可乐
7	牛奶，面包，鸡蛋，麦片	面包，牛奶，麦片，鸡蛋
8	牛奶，面包，可乐	面包，牛奶，可乐
9	面包，可乐	面包，可乐

项集	支持度计数	结点链头指针
面包	7	
牛奶	6	
可乐	6	
麦片	4	
鸡蛋	2	

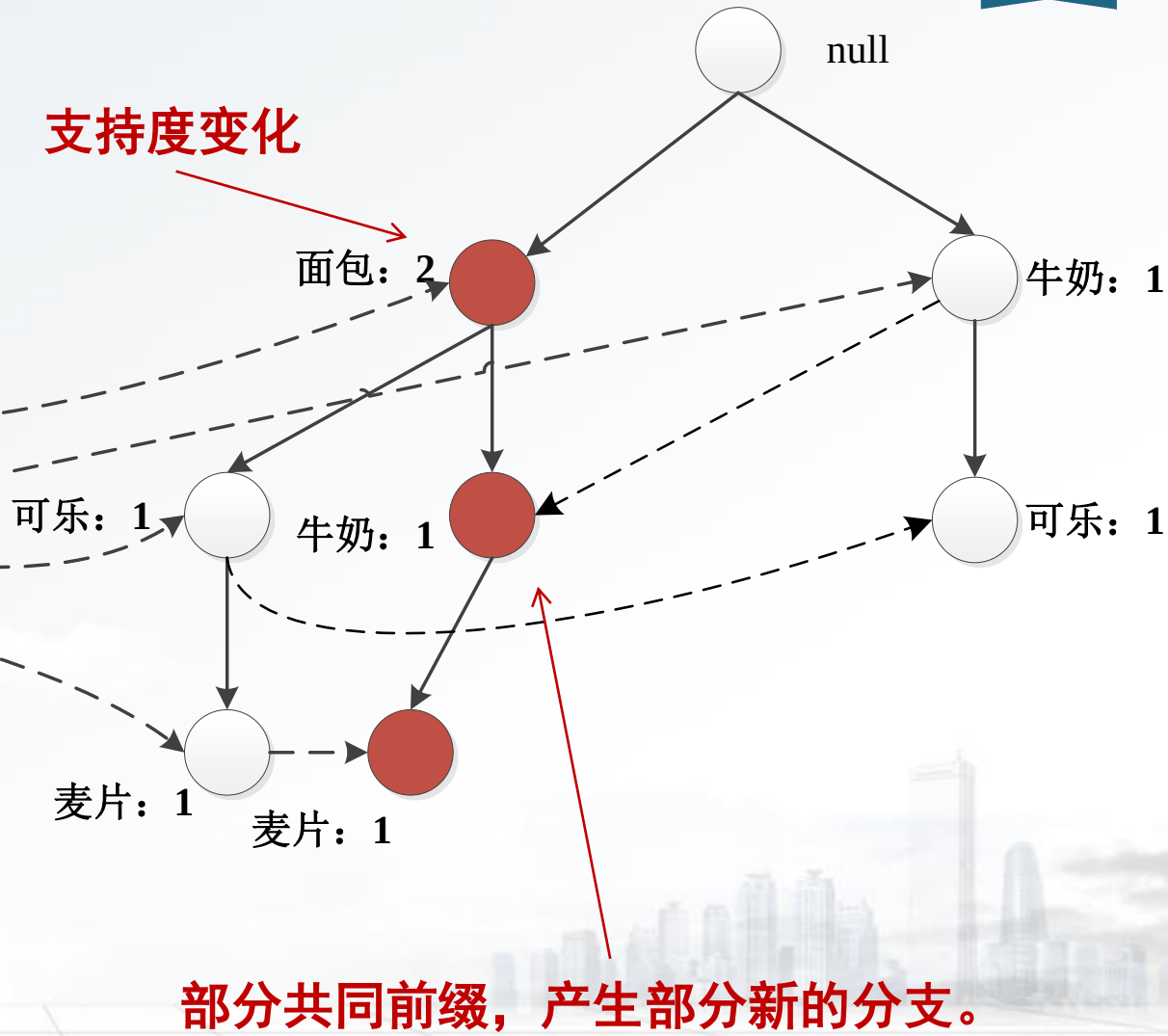


图8-11(3) 处理TID=3后的FP-tree

读出每个事务并构建根结点为null的FP-tree：

表 6-4 更新后的某商店的详细事务数据

TID	Items	Sorted Items
1	面包，可乐，麦片	面包，可乐，麦片
2	牛奶，可乐	牛奶，可乐
3	牛奶，面包，麦片	面包，牛奶，麦片
√ 4	牛奶，可乐	牛奶，可乐
5	面包，鸡蛋，麦片	面包，麦片，鸡蛋
6	牛奶，面包，可乐	面包，牛奶，可乐
7	牛奶，面包，鸡蛋，麦片	面包，牛奶，麦片，鸡蛋
8	牛奶，面包，可乐	面包，牛奶，可乐
9	面包，可乐	面包，可乐

项集	支持度计数	结点链头指针
面包	7	
牛奶	6	
可乐	6	
麦片	4	
鸡蛋	2	

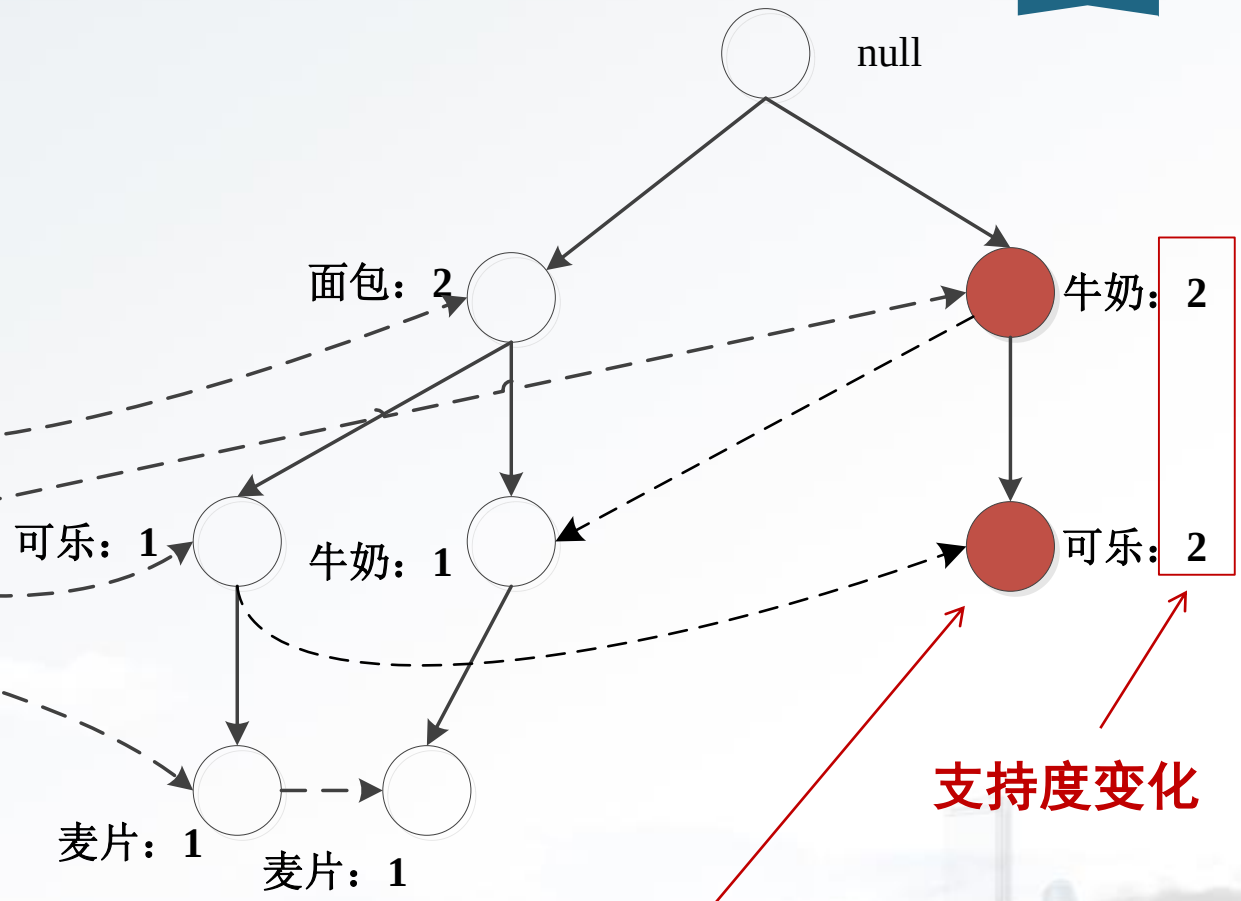


图8-11(4) 处理TID=4后的FP-tree

读出每个事务并构建根结点为null的FP-tree：

表 6-4 更新后的某商店的详细事务数据

TID	Items	Sorted Items
1	面包，可乐，麦片	面包，可乐，麦片
2	牛奶，可乐	牛奶，可乐
3	牛奶，面包，麦片	面包，牛奶，麦片
4	牛奶，可乐	牛奶，可乐
√ 5	面包，鸡蛋，麦片	面包，麦片，鸡蛋
6	牛奶，面包，可乐	面包，牛奶，可乐
7	牛奶，面包，鸡蛋，麦片	面包，牛奶，麦片，鸡蛋
8	牛奶，面包，可乐	面包，牛奶，可乐
9	面包，可乐	面包，可乐

项集	支持度计数	结点链头指针
面包	7	
牛奶	6	
可乐	6	
麦片	4	
鸡蛋	2	

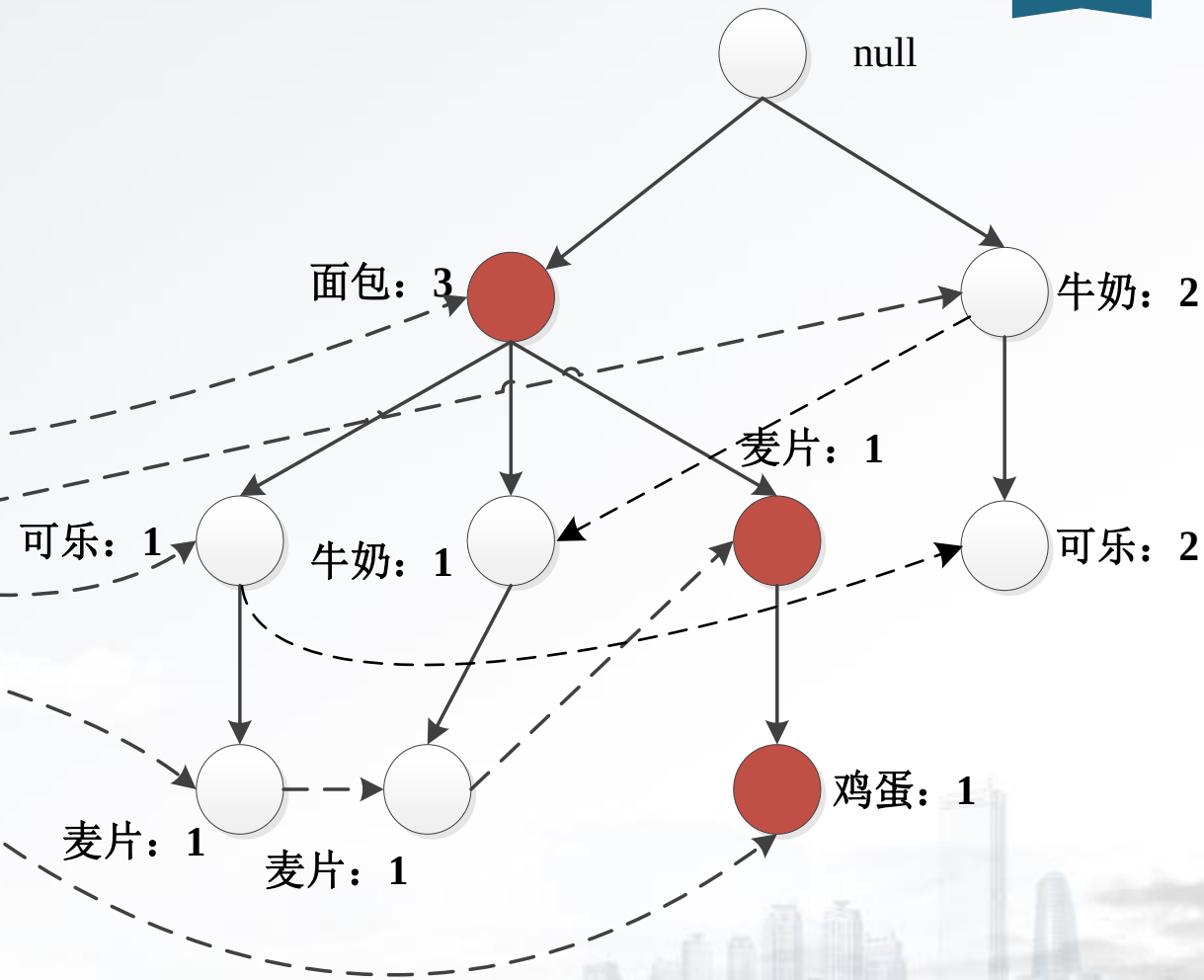


图8-11(5) 处理TID=5后的FP-tree

读出每个事务并构建根结点为null的FP-tree：

表 6-4 更新后的某商店的详细事务数据

TID	Items	Sorted Items
1	面包，可乐，麦片	面包，可乐，麦片
2	牛奶，可乐	牛奶，可乐
3	牛奶，面包，麦片	面包，牛奶，麦片
4	牛奶，可乐	牛奶，可乐
5	面包，鸡蛋，麦片	面包，麦片，鸡蛋
√ 6	牛奶，面包，可乐	面包，牛奶，可乐
7	牛奶，面包，鸡蛋，麦片	面包，牛奶，麦片，鸡蛋
8	牛奶，面包，可乐	面包，牛奶，可乐
9	面包，可乐	面包，可乐

项集	支持度计数	结点链头指针
面包	7	
牛奶	6	
可乐	6	
麦片	4	
鸡蛋	2	

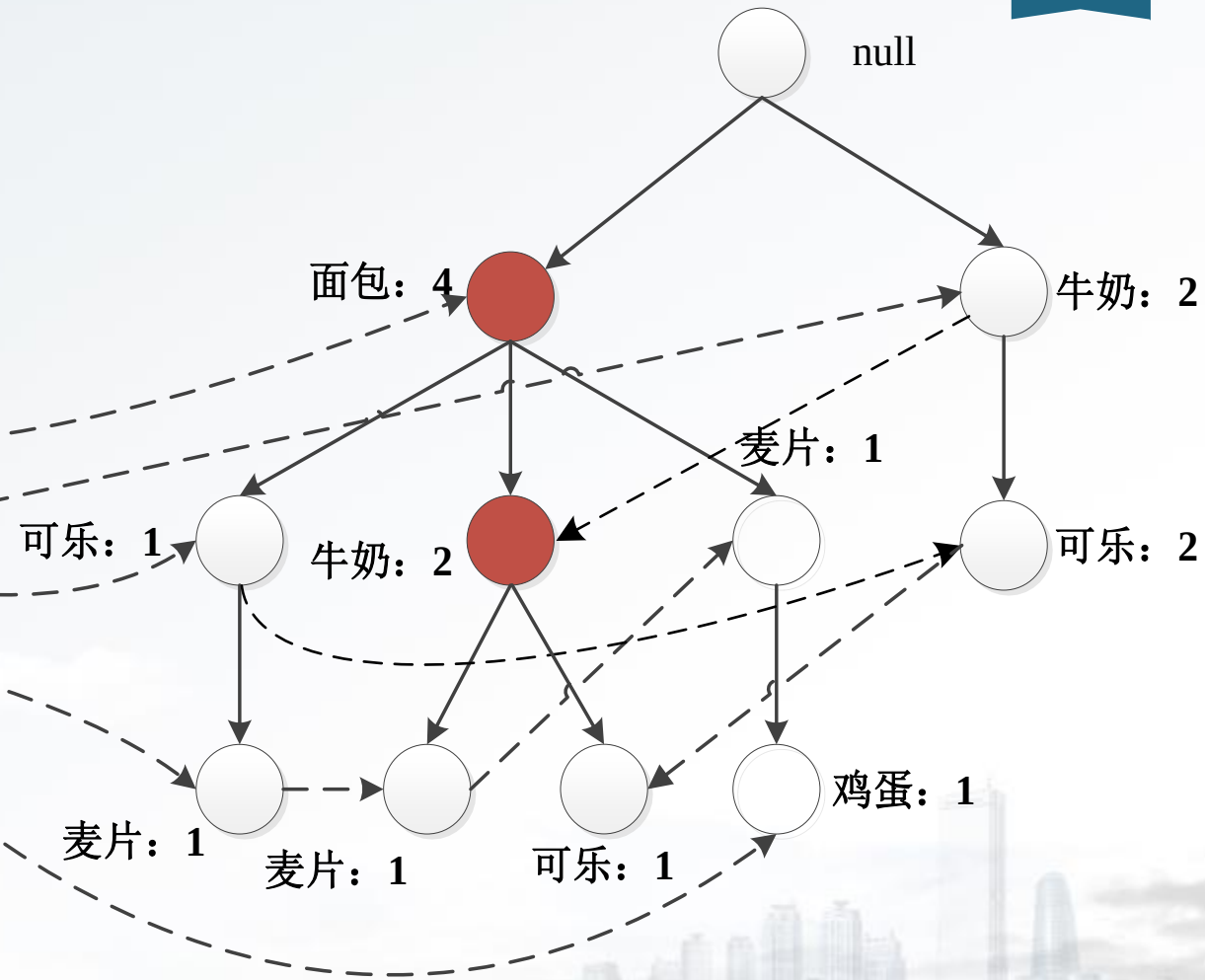


图8-11(6) 处理TID=6后的FP-tree

读出每个事务并构建根结点为null的FP-tree：

表 6-4 更新后的某商店的详细事务数据

TID	Items	Sorted Items
1	面包，可乐，麦片	面包，可乐，麦片
2	牛奶，可乐	牛奶，可乐
3	牛奶，面包，麦片	面包，牛奶，麦片
4	牛奶，可乐	牛奶，可乐
5	面包，鸡蛋，麦片	面包，麦片，鸡蛋
6	牛奶，面包，可乐	面包，牛奶，可乐
√ 7	牛奶，面包，鸡蛋，麦片	面包，牛奶，麦片，鸡蛋
8	牛奶，面包，可乐	面包，牛奶，可乐
9	面包，可乐	面包，可乐

项集	支持度计数	结点链头指针
面包	7	
牛奶	6	
可乐	6	
麦片	4	
鸡蛋	2	

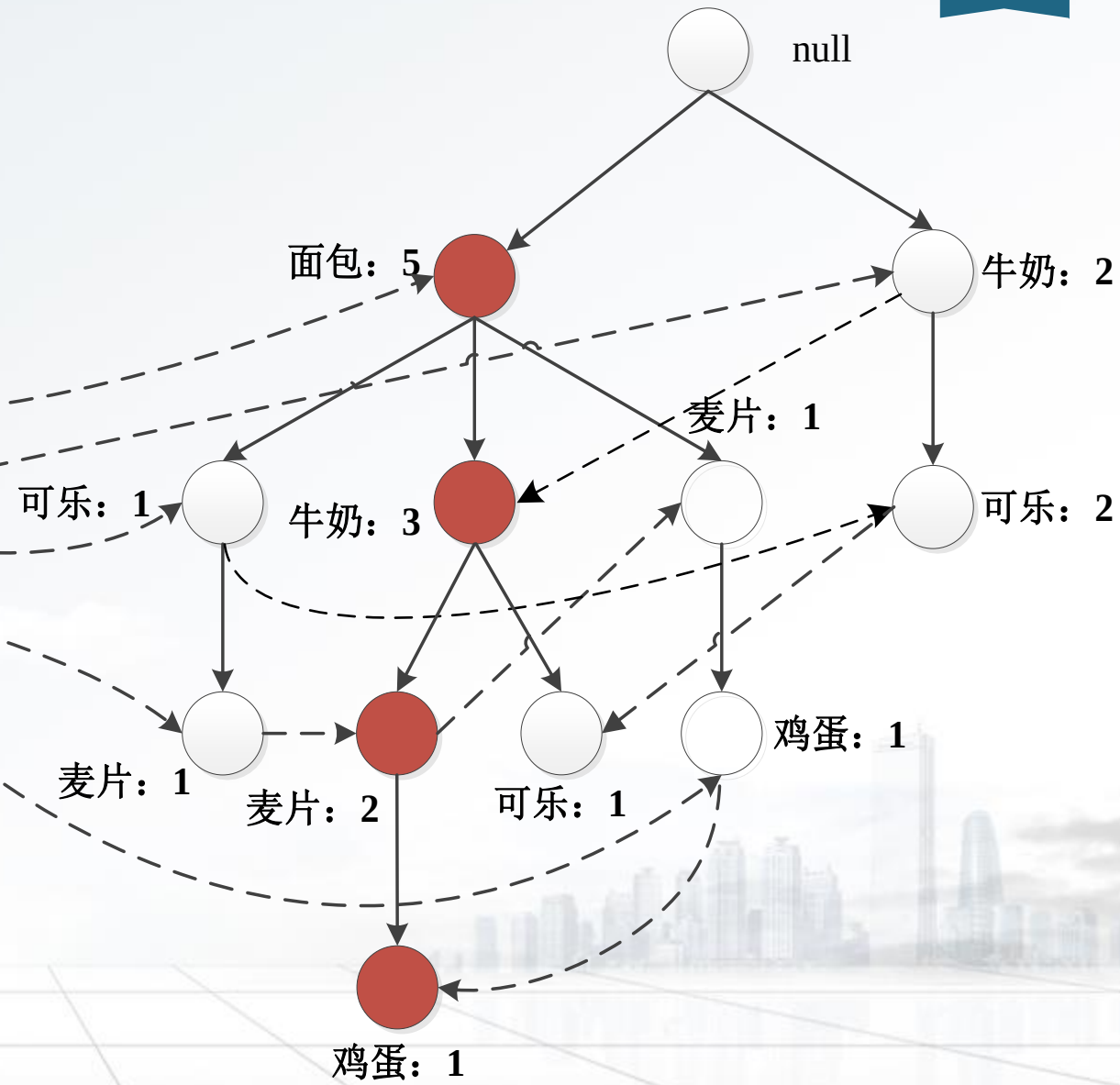


图8-11(7) 处理TID=7后的FP-tree

读出每个事务并构建根结点为null的FP-tree：

表 6-4 更新后的某商店的详细事务数据

TID	Items	Sorted Items
1	面包，可乐，麦片	面包，可乐，麦片
2	牛奶，可乐	牛奶，可乐
3	牛奶，面包，麦片	面包，牛奶，麦片
4	牛奶，可乐	牛奶，可乐
5	面包，鸡蛋，麦片	面包，麦片，鸡蛋
6	牛奶，面包，可乐	面包，牛奶，可乐
7	牛奶，面包，鸡蛋，麦片	面包，牛奶，麦片，鸡蛋
√ 8	牛奶，面包，可乐	面包，牛奶，可乐
9	面包，可乐	面包，可乐

项集	支持度计数	结点链头指针
面包	7	
牛奶	6	
可乐	6	
麦片	4	
鸡蛋	2	

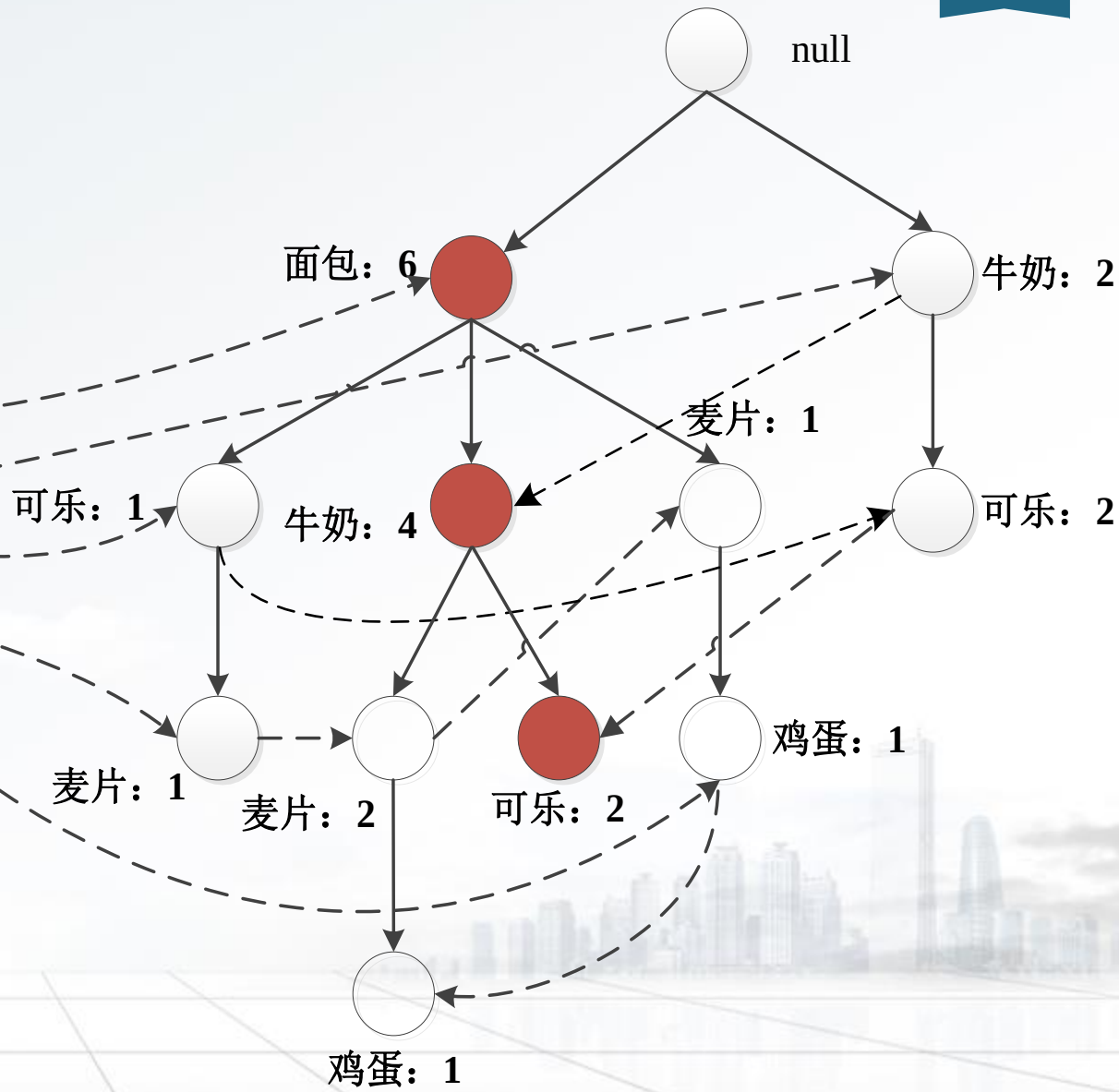


图8-11(8) 处理TID=8后的FP-tree

读出每个事务并构建根结点为null的FP-tree：

表 6-4 更新后的某商店的详细事务数据

TID	Items	Sorted Items
1	面包，可乐，麦片	面包，可乐，麦片
2	牛奶，可乐	牛奶，可乐
3	牛奶，面包，麦片	面包，牛奶，麦片
4	牛奶，可乐	牛奶，可乐
5	面包，鸡蛋，麦片	面包，麦片，鸡蛋
6	牛奶，面包，可乐	面包，牛奶，可乐
7	牛奶，面包，鸡蛋，麦片	面包，牛奶，麦片，鸡蛋
8	牛奶，面包，可乐	面包，牛奶，可乐
√ 9	面包，可乐	面包，可乐

项集	支持度计数	结点链头指针
面包	7	
牛奶	6	
可乐	6	
麦片	4	
鸡蛋	2	

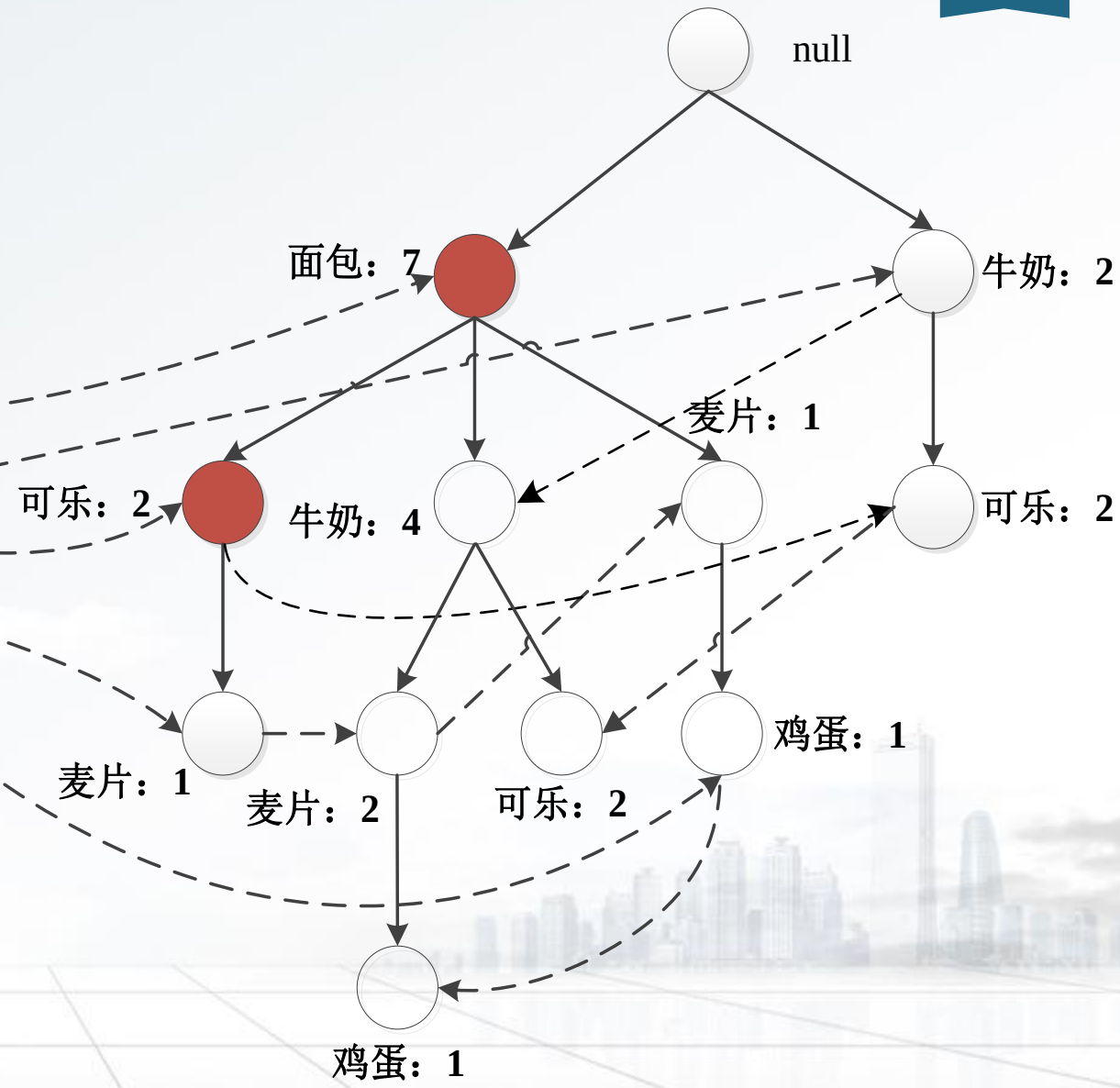


图8-11(9) 处理TID=9后的FP-tree

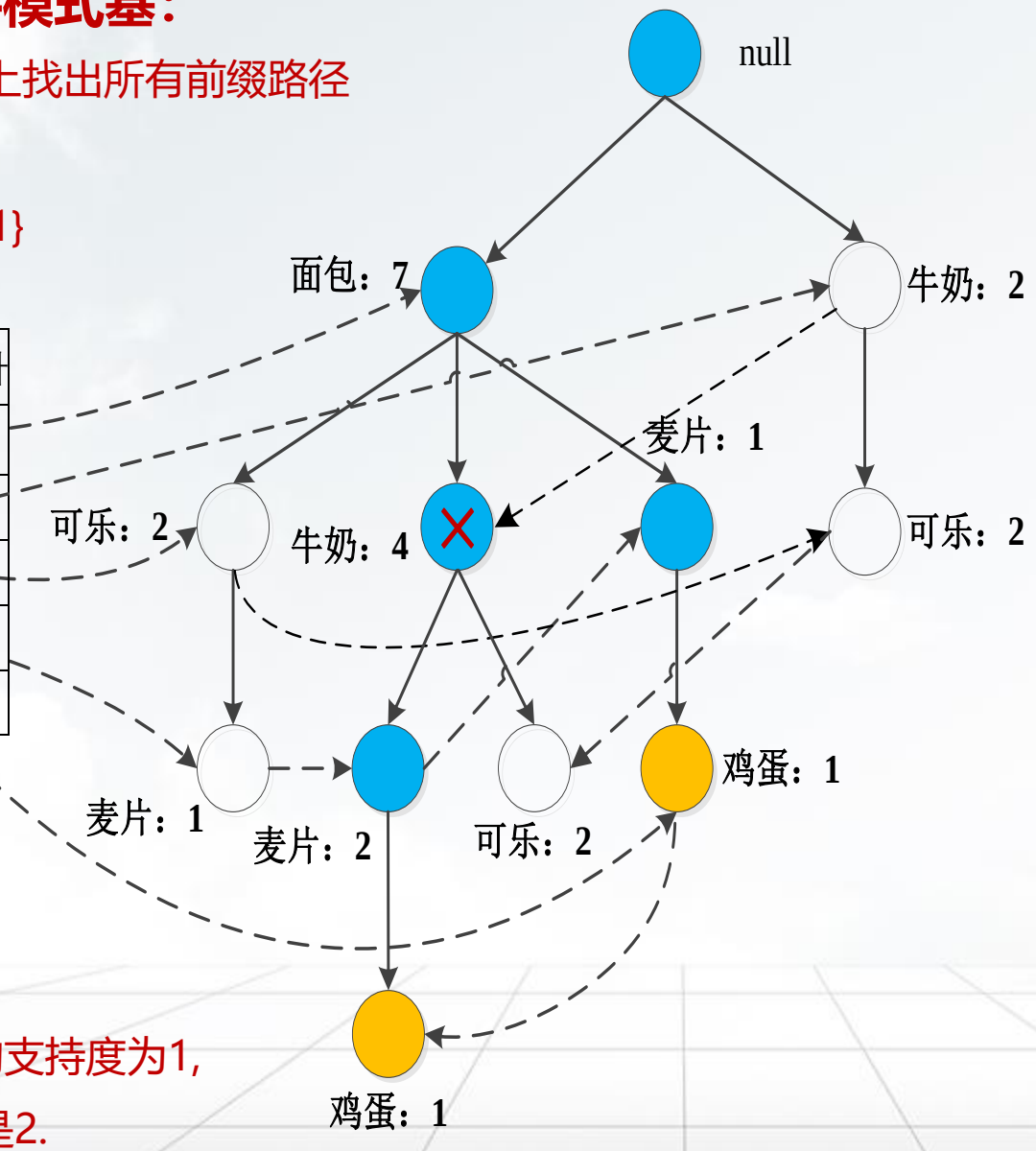
构造该频繁项的条件模式基：

对于项集{鸡蛋}，开始向上找出所有前缀路径，找出其条件模式基：

{面包，牛奶，麦片：1}

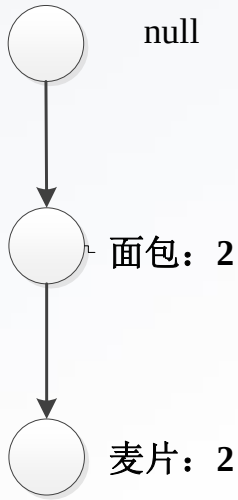
{面包，麦片：1}

项集	支持度计数	结点链头指针
面包	7	
牛奶	6	
可乐	6	
麦片	4	
√鸡蛋	2	



两条路径{牛奶}对{鸡蛋}的支持度为1，小于阈值2！面包和麦片是2.

通过条件模式基构造条件FP-tree：



<面包：2，麦片：2>

图8-12 {鸡蛋}的条件FP-tree

根据{鸡蛋}的条件FP-tree产生的频繁项集的所有组合为：

{面包，鸡蛋：2}

{麦片，鸡蛋：2}

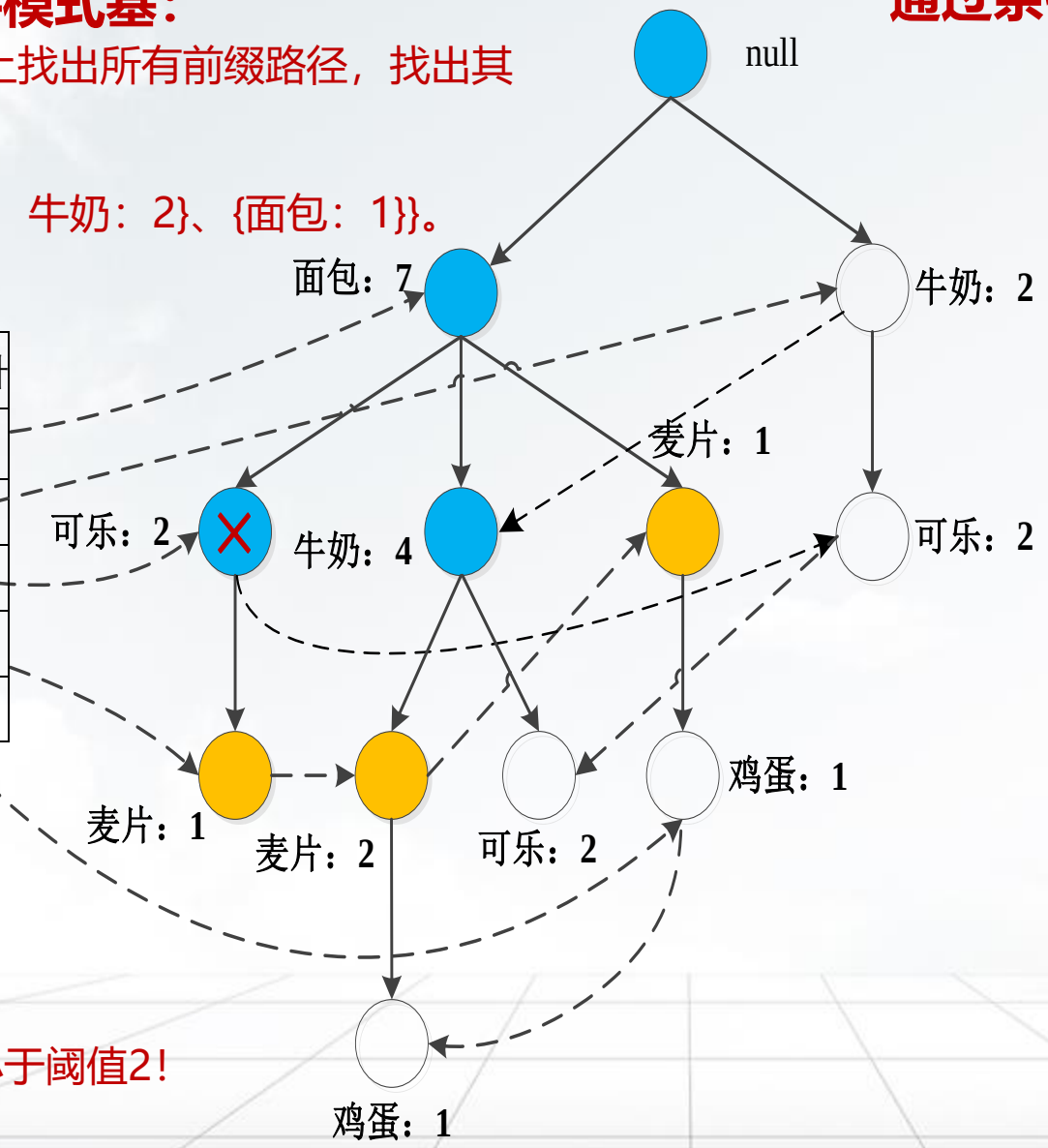
{面包，麦片，鸡蛋：2}

构造该频繁项的条件模式基：

对于项集{麦片}，开始向上找出所有前缀路径，找出其条件模式基：

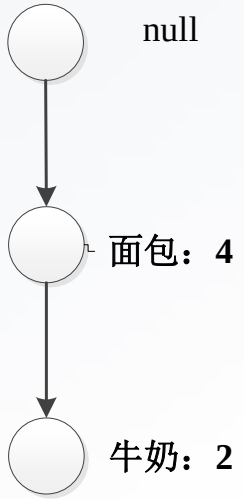
{{面包，可乐：1}、{面包，牛奶：2}、{面包：1}}。

项集	支持度计数	结点链头指针
面包	7	
牛奶	6	
可乐	6	
√麦片	4	
鸡蛋	2	



{可乐}对{麦片}的支持度小于阈值2！

通过条件模式基构造条件FP-tree：



<面包：4，麦片：2>

图8-13 {麦片}的条件FP-tree

根据{麦片}的条件FP-tree找出其中的频繁模式：

{面包，麦片：4}

{牛奶，麦片：2}

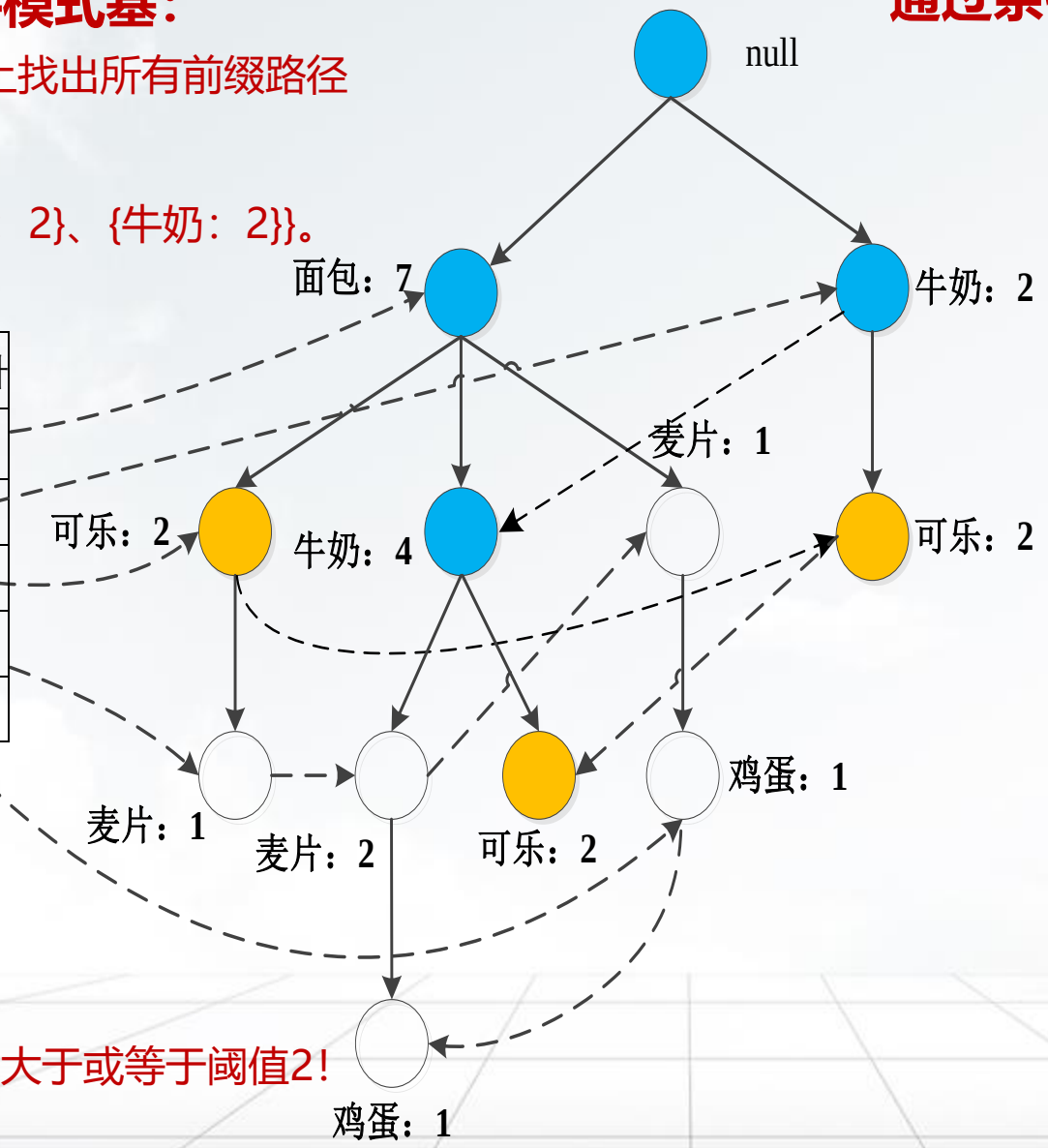
{面包，牛奶，麦片：2}

构造该频繁项的条件模式基：

对于项集{可乐}，开始向上找出所有前缀路径，找出其条件模式基：

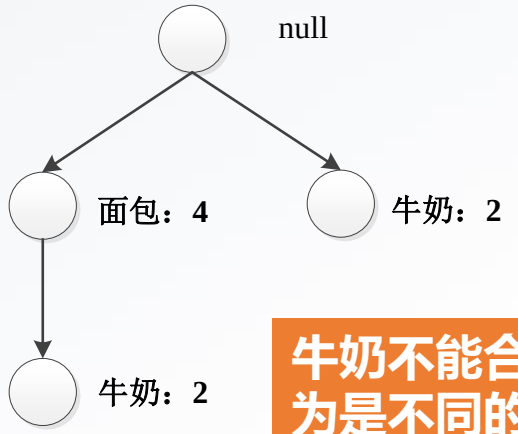
{{面包：2}、{面包，牛奶：2}、{牛奶：2}}。

项集	支持度计数	结点链头指针
面包	7	
牛奶	6	
√可乐	6	
麦片	4	
鸡蛋	2	



每个项对{可乐}的支持度都大于或等于阈值2!

通过条件模式基构造条件FP-tree：



牛奶不能合并,因为是不同的路径

<面包：4，牛奶：2>&<牛奶：2>

图8-14 {可乐}的条件FP-tree

根据{可乐}的条件FP-tree找出其中的频繁模式：

{面包，可乐：4}

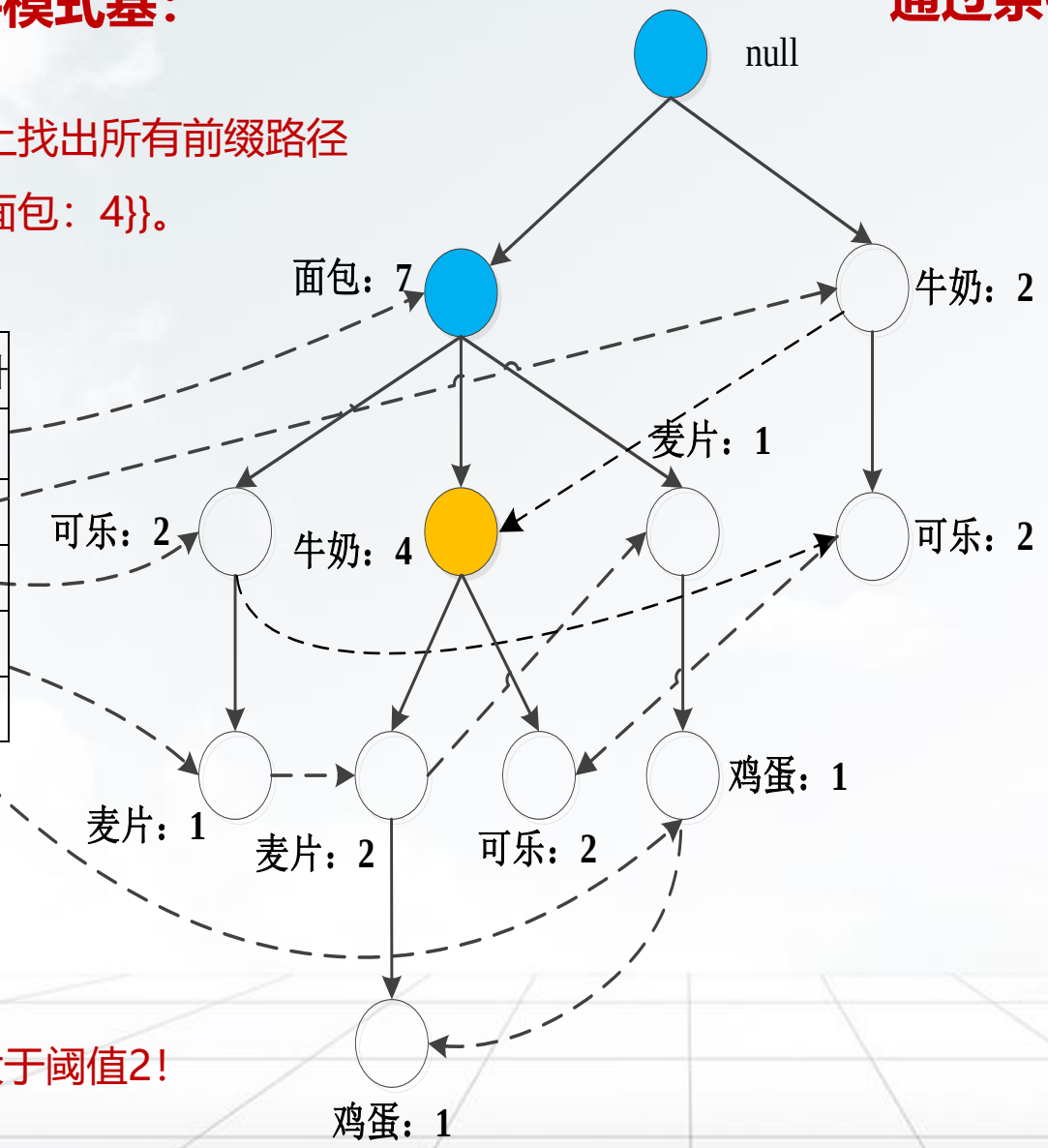
{牛奶，可乐：4}

{面包，牛奶，可乐：2}

构造该频繁项的条件模式基：

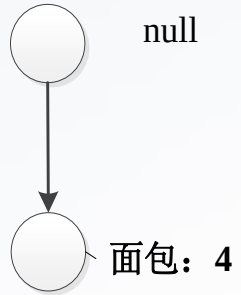
对于项集{牛奶}，开始向上找出所有前缀路径，找出其条件模式基：{{面包：4}}。

项集	支持度计数	结点链头指针
面包	7	
√牛奶	6	
可乐	6	
麦片	4	
鸡蛋	2	



{面包}对{牛奶}的支持度大于阈值2！

通过条件模式基构造条件FP-tree：



<面包：4>

图8-15 {牛奶}的条件FP-tree

根据{牛奶}的条件FP-tree找出其中的频繁模式：

{面包，牛奶：4}

表8-5 FP-tree 挖掘过程

项集	条件模式基	条件FP-tree	频繁模式
鸡蛋	{{面包，牛奶，麦片：1}，{面包，麦片：1}}	<面包：2，麦片：2>	{面包，鸡蛋：2}，{麦片，鸡蛋：2}，{面包，麦片，鸡蛋：2}
麦片	{{面包，可乐：1}，{面包，牛奶：2}，{面包：1}}	<面包：4，牛奶：2>	{面包，麦片：4}，{牛奶，麦片：2}，{面包，牛奶，麦片：2}
可乐	{{面包：2}，{面包，牛奶：2}，{牛奶：2}}	<面包：4，牛奶：2>，<牛奶：2>	{面包，可乐：4}，{牛奶，可乐：4}，{面包，牛奶，可乐：2}
牛奶	{{面包：4}}	<面包：4>	{面包，牛奶：4}

– **性能研究表明**

- FP-树比Apriori算法快一个数量级

– **原因**

- 减少没有候选集的产生，没有候选测试；
- 使用简洁的数据结构；
- 除去了重复的数据库扫描。

例子：该数据集具有6个事务，设最小支持度为2，频繁项集的极大长度为3。试使用FP-growth算法挖掘下表的事务数据中的频繁项集。

事务ID	事务中的元素项
001	r, z, h, j, p
002	z, y, x, w, v, u, t, s
003	z
004	r, x, n, o, s
005	y, r, x, z, q, t, p
006	y, z, x, e, q, s, t, m

https://codeleading.com/article/54594412323/

Chapter 8.4

压缩频繁项集

8.4 压缩频繁项集

- 在实际应用中，当最小支持度**阈值较低**或者**数据规模较大**时，使用频繁模式挖掘事务数据可能产生**过多的频繁项集**；
- 而**闭频繁模式**、**极大模式等模式**可以显著减少频繁模式挖掘所产生的频繁项集数量。

1. 挖掘闭模式

- 如果 $X \in Y$，且Y中至少有一项不在X中，那么Y是X的**真超项集**。
- 如果在数据集中不存在频繁项集X的真超项集Y，使得X、Y的支持度相等，那么称项集X是这个数据集的**闭频繁项集**。

2. 剪枝的策略

– **项合并**

- 如果包含频繁项集*X*的每个事务都包含项集*Y*，但不包含*Y*的任何真超集，则*X*∪*Y*形成一个**闭频繁项集**，并且不必搜索包含*X*但不包含*Y*的任何项集。

– **子项集剪枝**

- 如果频繁项集*X*是一个已经发现的闭频繁项集*Y*的真子集，并且两者的支持度计数相等，则*X*和*Y*的所有后代都不可能是闭频繁项集，因此**可以剪枝**。

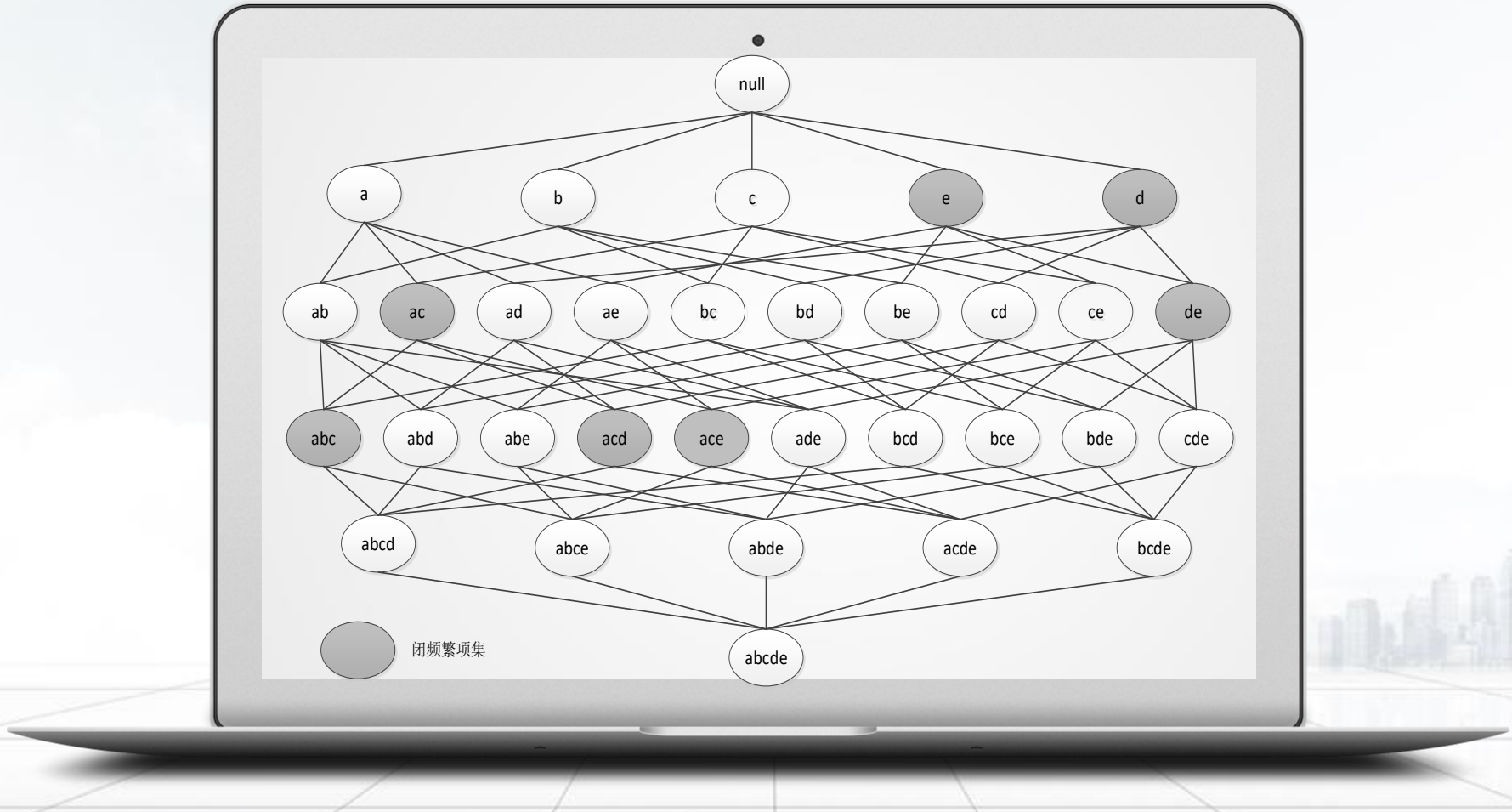
测试数据集

表8-6 项集事务

TID	Items
1	abc
2	abcd
3	ace
4	acde
5	de

测试数据集

最小支持度为2，可以得到{{d}、{e}、{ac}、{de}、{abc}、{acd}、{ace}}这些频繁项集的支持度都大于等于2，并且他们都不存在等于他们支持度的真超集，所以他们是**闭频繁项集**。

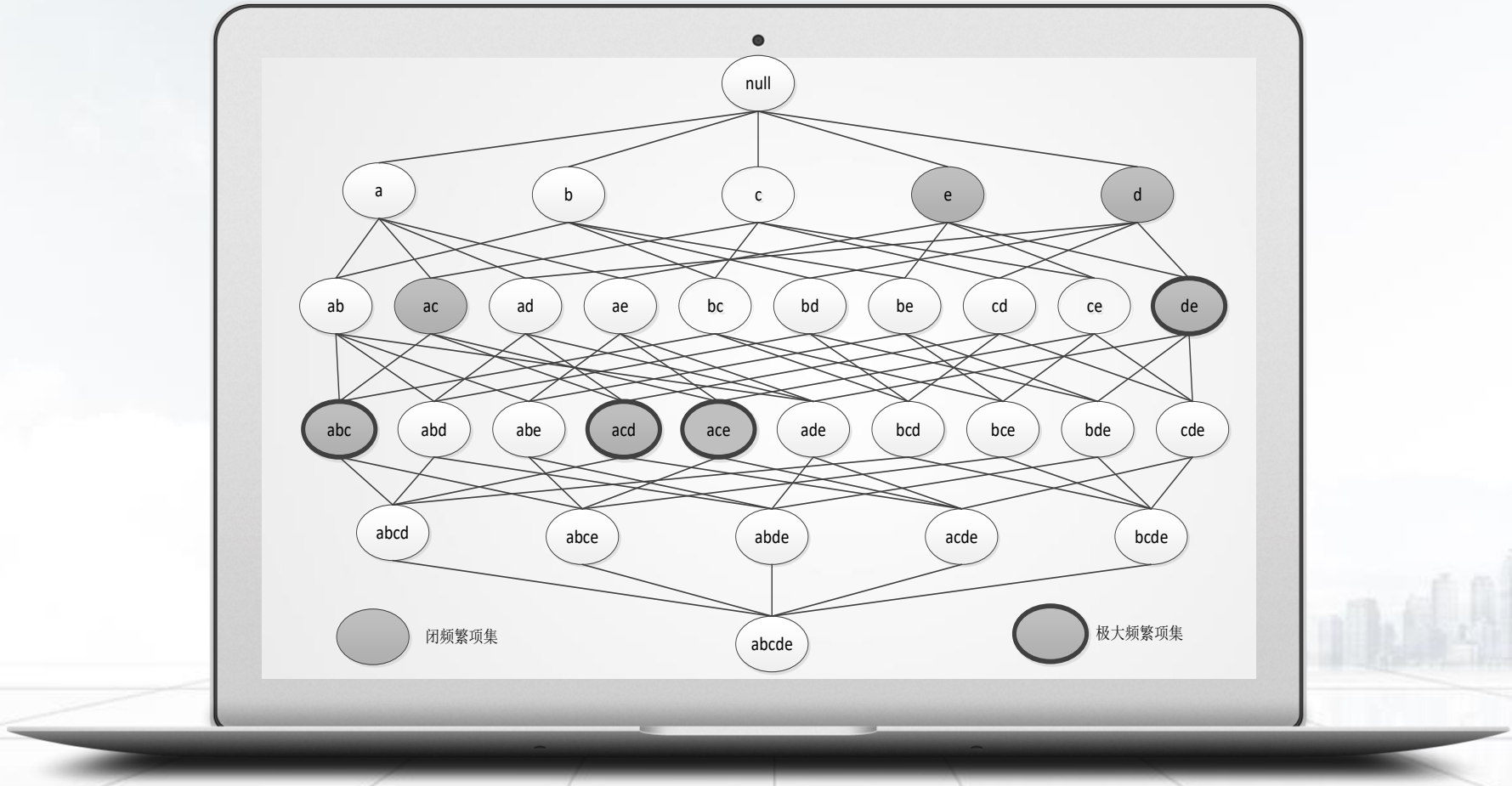


3. 极大频繁项集：

- 如果在数据集中不存在频繁项集X的真超项集Y，使得X属于Y，并且Y也是频繁项集，那么称项集X是这个数据集的**极大频繁项集**。
- 可以推导出极大频繁项集是闭频繁项集，而闭频繁项集不一定是极大频繁项集。
- 极大频繁项集形成了所有频繁项集的最小的项集的集合，即有效地提供了**频繁项集的紧凑表示**。

假设最小支持度为2，从图中可知闭频繁项集{{d}、{e}、{ac}、{de}、{abc}、{acd}、{ace}}

根据定义{{de}、{abc}、{acd}、{ace}}则是**极大频繁项集**。



Chapter 8.5

关联模式评估

1. 支持度-置信度框架

- 强关联规则（支持度-置信度框架）不一定是用户感兴趣的。
- 最小支持度阈值为0.3，最小置信度阈值为0.6。

表8-5 1000个人的手机偏爱

	买苹果手机	不买苹果手机	行和
买小米手机	400	350	750
不买小米手机	200	50	250
列和	600	400	1000

- 根据计算，买苹果手机和买小米手机是强关联规则，但是买苹果手机和买小米手机是互斥的。
- 没有考虑到买了苹果手机后，再买小米手机这个条件过程。

2. 相关性分析

①提升度

- 令*A*和*B*表示不同的项集，$P(*)$表示项集*在总体数据集中的出现概率。根据统计学定义，如果项集*A*和项集*B*的$P(A \cup B) = P(A)P(B)$，那么项集*A*和项集*B*是相互独立的，否则两者是相互依赖的。
- 项集*A*和项集*B*的提升度定义如式(8-3)所示。

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

如果A和B的提升度的值等于1，说明A和B相互独立；若是A和B的提升度的值大于1，说明A和B正相关；如果A和B的提升度的值小于1，说明A和B负相关。

Lift({苹果手机}，{小米手机})=0.4/(0.75×0.6)=0.89

2. 相关性分析

②杠杆度

杠杆度和提升度的含义相近，其定义如式（8-4）所示。

$$leverage(A,B) = P(A \cup B) - P(A)P(B) \qquad (8\text{-}4)$$

如果*A*和*B*的杠杆度的值等于0，则说明*A*和*B*相互独立；

如果*A*和*B*的杠杆度的值大于0，说明*A*和*B*正相关，并且杠杆度越大，说明*A*和*B*的关系越密切；

如果*A*和*B*的杠杆度的值小于0，说明*A*和*B*负相关。

{苹果手机}、{小米手机} 的杠杆度为

$$leverage(\{苹果手机\},\{小米手机\}) = 0.4 - 0.6 \times 0.75 = -0.05$$

苹果手机和小米手机是负相关的。

2. 相关性分析

③皮尔森相关系数

皮尔森相关系数能够反映两个变量的相似程度，皮尔森相关系数值越大表明两个变量的相关性越强。对于二元变量，皮尔森相关系数定义如式(8-4)所示。

⇒

$$\rho(A,B)=\frac{P(A\cup B)P(\overline{A}\cup\overline{B})-P(\overline{A}\cup B)P(A\cup\overline{B})}{\sqrt{P(A)P(\overline{A})P(B)P(\overline{B})}}$$

{苹果手机}和{小米手机}的皮尔森相关系数为

$$\rho(\{苹果手机\},\{小米手机\})=\frac{(0.4*0.05-0.35*0.2)}{\sqrt{0.6*0.4*0.75*0.25}}=-0.2357$$

说明两者一定程度负相关。

2. 相关性分析

④IS度量

IS度量通常用于处理非对称二元变量，IS度量定义如式(8-5)所示。

$$IS(A,B)=\frac{P(A\cup B)}{\sqrt{P(A)P(B)}}$$

IS度量的数值越大则说明A和B之间的关联越强。

{苹果手机}和{小米手机}的IS度量为

$$IS(\{苹果手机\},\{小米手机\})=\frac{0.4}{\sqrt{0.6*0.75}}=0.5963$$

说明A和B关联一般。

2. 相关性分析

⑤确信度

确信度能够度量一个规则的强度，同时衡量A和B之间的独立性。确信度定义如式(8-6)所示。

$$Conviction(A,B)=\frac{P(A)P(\bar{B})}{\sqrt{P(A\cup\bar{B})}}$$

确信度越大，A和B关系越紧密。

{苹果手机}和{小米手机}的确信度为

$$Conviction(\{苹果手机\},\{小米手机\})=\frac{0.6*0.25}{\sqrt{0.2}}=0.33541$$

，说明苹果手机和小米手机的关系不紧密。

3. 模式评估度量

不包含任何考察项集的事务被称作零事务。提升度、皮尔森相关系数和卡方系数等度量在很大程度上受零事务的影响，因此它们识别关联模式关联关系的能力较差。

①全置信度

全置信度反映了规则$A \Rightarrow B$和规则$B \Rightarrow A$的最小置信度。全置信度定义如式(8-7)所示。

- $$all\_conf(A,B) = \frac{P(A \cup B)}{\max\{P(A),P(B)\}} = \min\{P(A|B),P(B|A)\} \quad (8\text{-}7)$$
- 对于项集*A*和*B*，全置信度越大，说明规则$A \Rightarrow B$和规则$B \Rightarrow A$的最小置信度越大，那么*A*和*B*关系越紧密，反之*A*和*B*关系越疏远。

  {苹果手机}和{小米手机}的全置信度为

$$all\_conf(\{\text{苹果手机}\},\{\text{小米手机}\}) = \frac{0.4}{\max\{0.6,0.75\}} = 0.5333$$

  说明苹果手机和小米手机的关系一般。

3. 模式评估度量

②极大置信度

极大置信度则反映了规则$A \Rightarrow B$和规则$B \Rightarrow A$的最大置信度。极大置信度定义如式(8-8)所示。

$$\max\_conf(A,B) = \max\{P(A|B), P(B|A)\} \quad (8\text{-}8)$$

对于项集*A*和*B*，极大置信度越大，*A*和*B*关系越紧密。

{苹果手机}和{小米手机}的极大置信度为

$$\max\_conf(\{\text{苹果手机}\},\{\text{小米手机}\}) = \max\left(\frac{0.4}{0.6}, \frac{0.4}{0.75}\right) = 0.667$$

说明两者可能关系一般。

3. 模式评估度量

③Kulczynski度量

Kulczynski度量表示在项集*A*存在的情况下项集*B*也存在的条件概率和在项集*B*存在的情况下项集*A*也存在的条件概率之和的平均值。Kulczynski度量定义如式(8-9)所示。

$$Kulc(A,B)=\frac{1}{2}\left(P(A|B)+P(B|A)\right) \qquad (8\text{-}9)$$

对于项集*A*和*B*，Kulczynski度量越大，说明平均可信程度越大，那么*A*和*B*关系越紧密。

{苹果手机}和{小米手机}的**.(库尔钦斯基)**Kulczynski度量为

$$Kulc(\{\text{苹果手机}\},\{\text{小米手机}\})=\frac{1}{2}\left(\frac{0.4}{0.6}+\frac{0.4}{0.75}\right)=0.6$$

说明两者关系一般。