

数据挖掘

第三章关联模式分析

2023秋

第三章 关联模式分析

- 1 基本概念
- 2 频繁项挖掘算法
- 3 关联分析的评估

1.1 定义：关联分析（association analysis）

- 关联分析用于发现隐藏在大型数据集中的令人感兴趣的联系，所发现的模式通常用**关联规则或频繁项集**的形式表示。
- 关联规则反映一个**事物与其它事物**之间的相互依存性和关联性。如果**两个或者多个事物之间**存在一定的关联关系，那么，其中一个事物发生就能够预测与它相关联的其它事物的发生。

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Rules Discovered:

{Diaper} --> {Beer}

1.1 定义： 频繁项集 (Frequent Itemset)

- **项集 (Itemset)**

- 包含0个或多个项的集合

- 例子： {Milk, Bread, Diaper}

- k-项集

- 如果一个项集包含k个项

- **支持度计数 (Support count) (σ)**

- 包含特定项集的事务个数

- 例如： $\sigma(\{\text{Milk, Bread, Diaper}\}) = 2$

- **支持度 (Support)**

- 包含项集的事务数与总事务数的比值

- 例如： $s(\{\text{Milk, Bread, Diaper}\}) = 2/5$

- **频繁项集 (Frequent Itemset)**

- 满足最小支持度阈值 (minsup) 的所有项集

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

事务Id	项集合
10	A, B, C
20	A, C
30	A, D
40	B, E, F

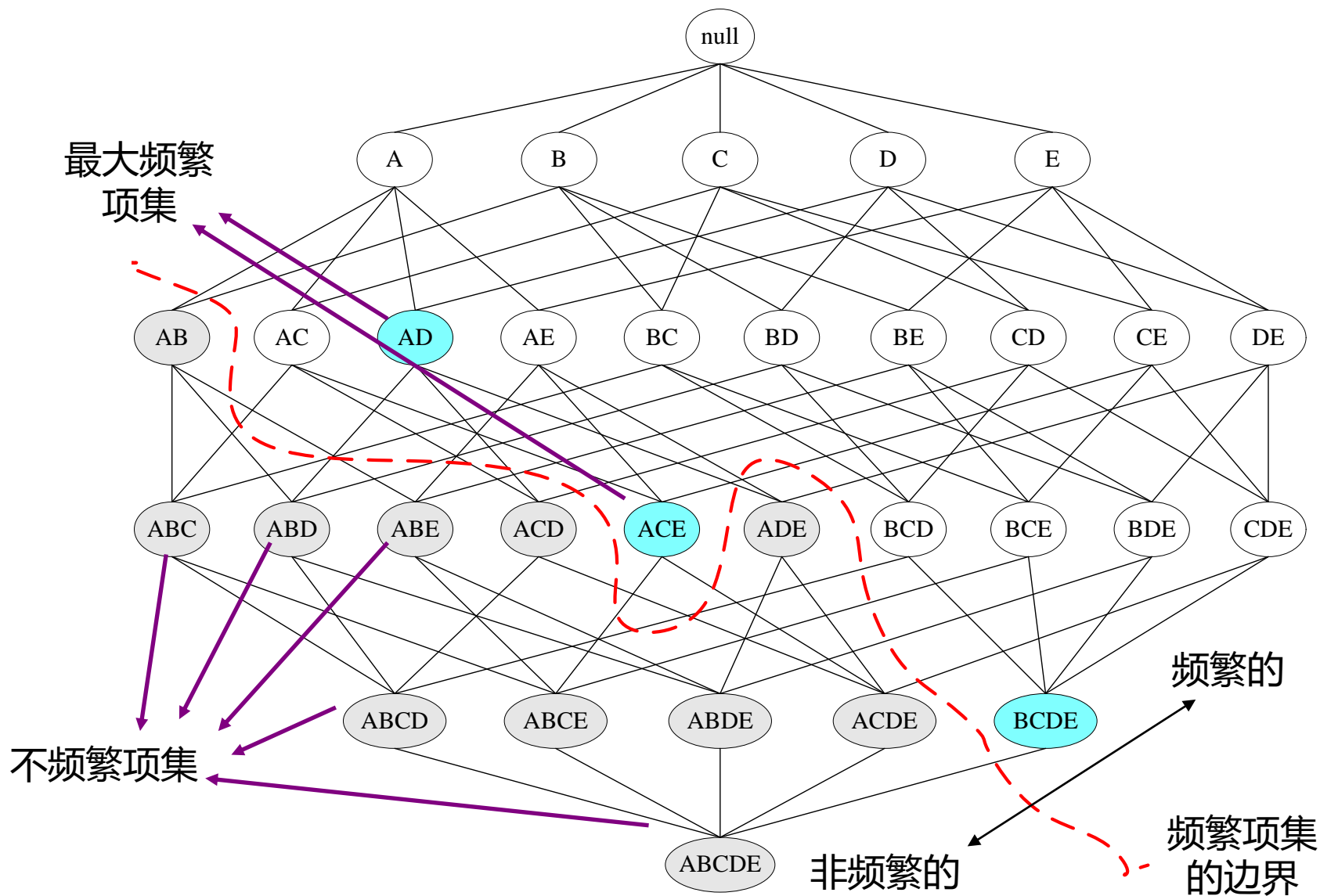
求以下两个规则的支持度和置信度：

$A \Rightarrow C$

$C \Rightarrow A$

1.2 最大频繁项集 (Maximal Frequent Itemset)

最大频繁项集是这样的频繁项集，它的直接超集都不是频繁的



1.3 定义：关联规则 (Association Rule)

- 关联规则

- 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

- 例子:

$\{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\}$

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

- 关联规则的强度

- 支持度 Support (s)

- 确定项集的频繁程度

- 置信度 Confidence (c)

- 确定Y在包含X的事
- 务中出现的频繁程度

Example:

$\{\text{Milk, Diaper}\} \Rightarrow \text{Beer}$

$$s = \frac{\sigma(\text{Milk, Diaper, Beer})}{|T|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk, Diaper, Beer})}{\sigma(\text{Milk, Diaper})} = \frac{2}{3} = 0.67$$

1.4 关联规则挖掘问题

关联规则挖掘问题：

给定事务的集合 T ，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是对应的支持度和置信度阈值

事务Id	项集合
10	A, B, C
20	A, C
30	A, D
40	B, E, F

最小支持度 50%
最小置信度 50%

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

支持度 = $\text{support}(\{A\} \cup \{C\}) =$ [填空1]

置信度 = $\text{support}(\{A\} \cup \{C\}) / \text{support}(\{A\}) =$ [填空2]

作答

关联规则挖掘问题： 给定事务的集合 T, 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是对应的支持度和置信度阈值

事务Id	项集合
10	A, B, C
20	A, C
30	A, D
40	B, E, F

最小支持度 50%
最小置信度 50%

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

规则 $A \Rightarrow C$:

支持度 = $\text{support}(\{A\} \cup \{C\}) = 50\%$

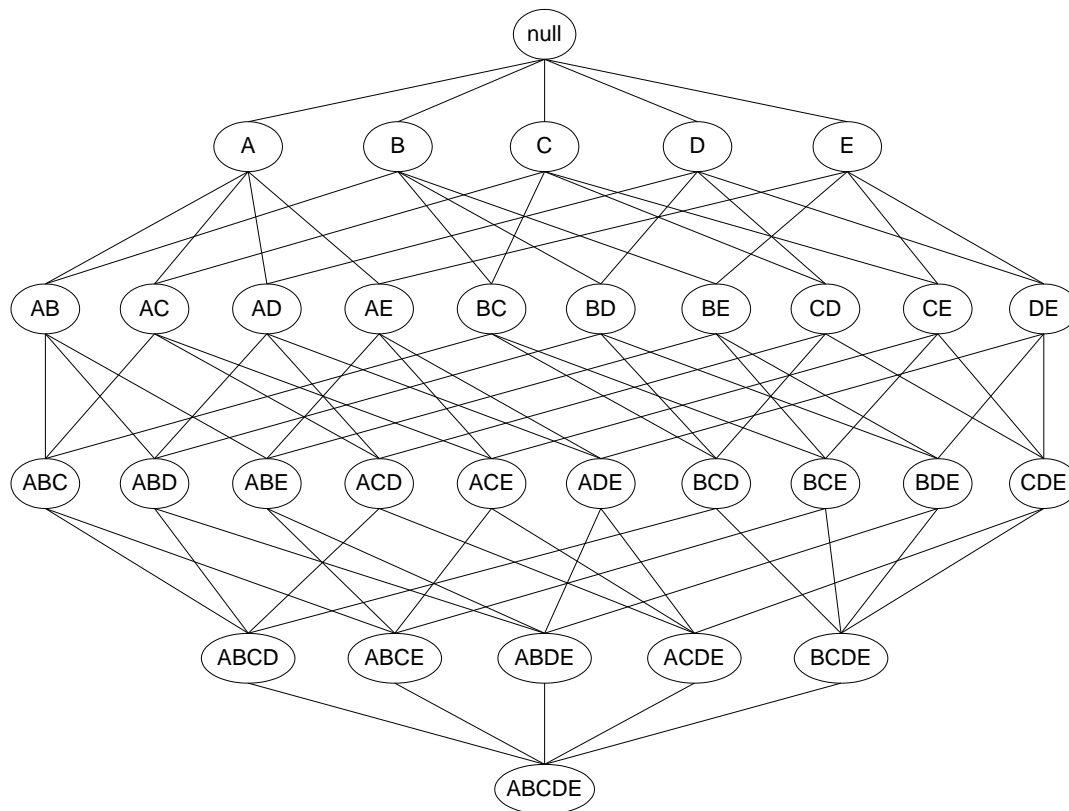
置信度 = $\text{support}(\{A\} \cup \{C\}) / \text{support}(\{A\}) = 66.6\%$

1.4 挖掘关联规则 (Mining Association Rules)

- 大多数关联规则挖掘算法通常采用的一种策略是，将关联规则挖掘任务分解为如下两个主要的子任务：
 - **频繁项集产生 (Frequent Itemset Generation)**
 - 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 (Rule Generation)**
 - 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 (strong rule) 。

1.5 关联规则原始方法

- 挖掘关联规则的一种原始方法是：Brute-force approach：
 - 计算每个可能规则的支持度和置信度
 - 这种方法计算代价过高，因为可以从数据集提取的规则的数量达指数级

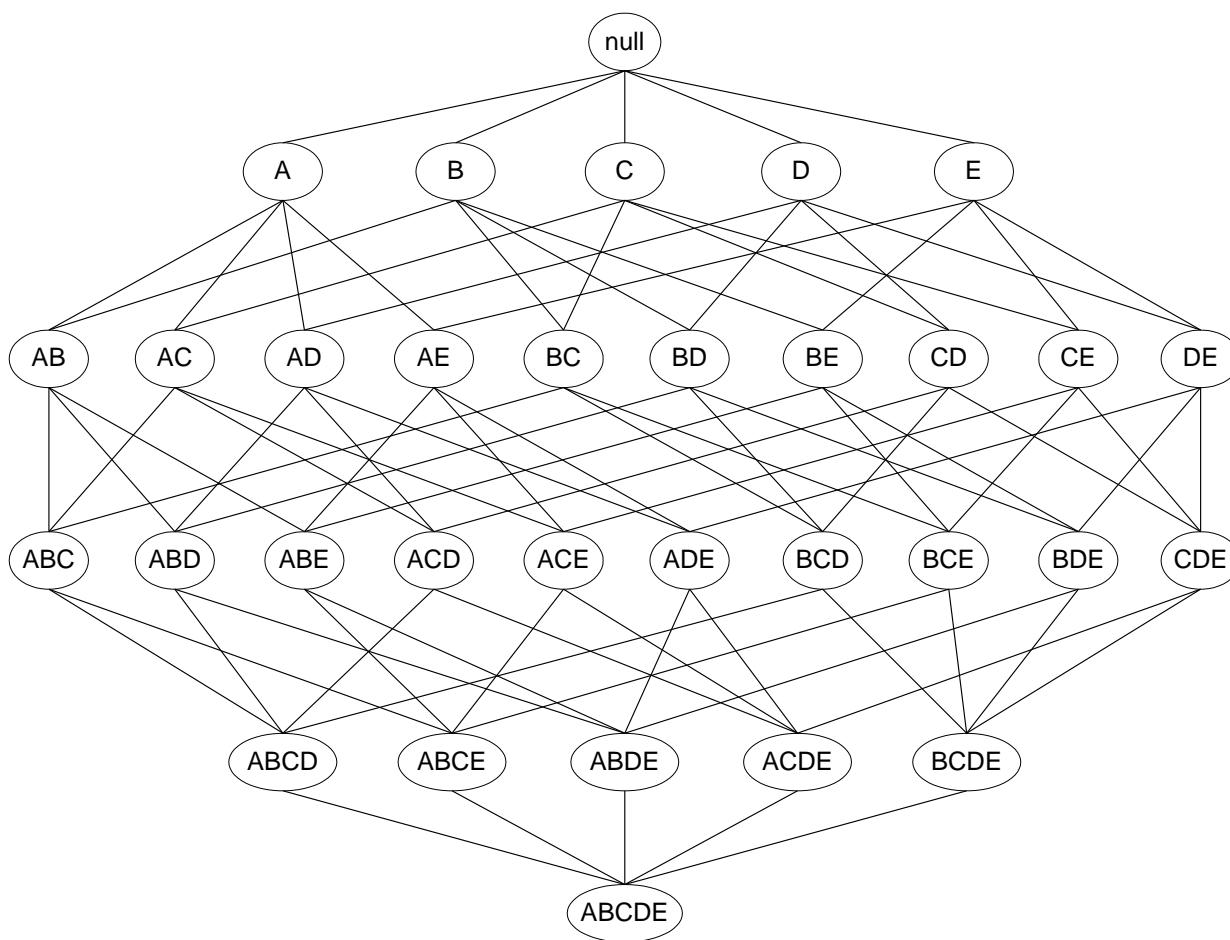


第三章 关联模式分析

- 3.1 基本概念
- 2.2 频繁项挖掘算法
- 2.3 关联分析的评估

2 频繁项集产生 (Frequent Itemset Generation)

- Brute-force 方法:
 - 计算开销大



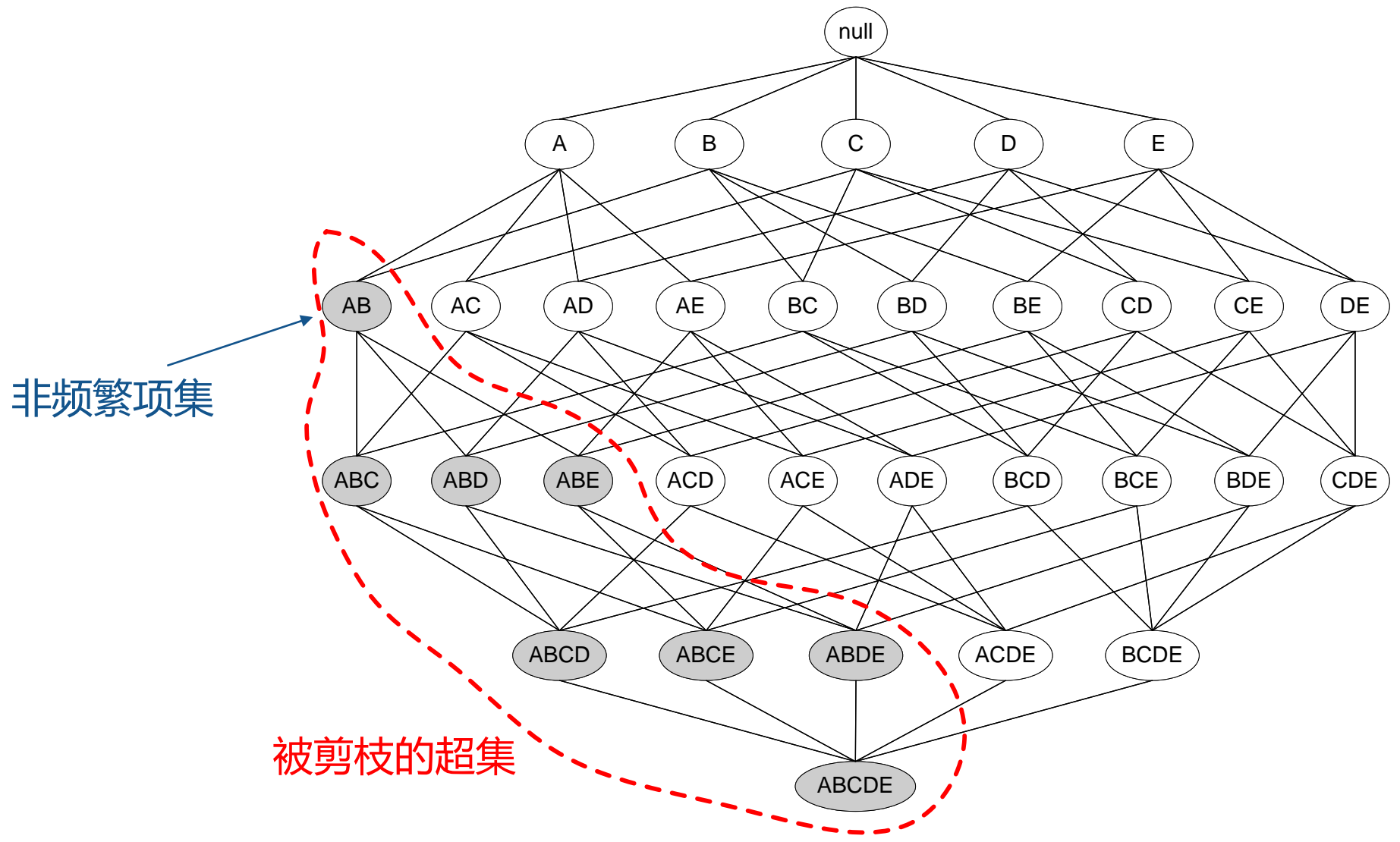
降低产生频繁项集计算复杂度的方法

- 减少候选项集的数量
 - 先验原理：(Apriori)
- 减少比较的次数
 - 替代将每个候选项集与每个事务相匹配，可以使用更高级的数据结构，或存储候选项集或压缩数据集，来减少比较次数
(FPGrowth)

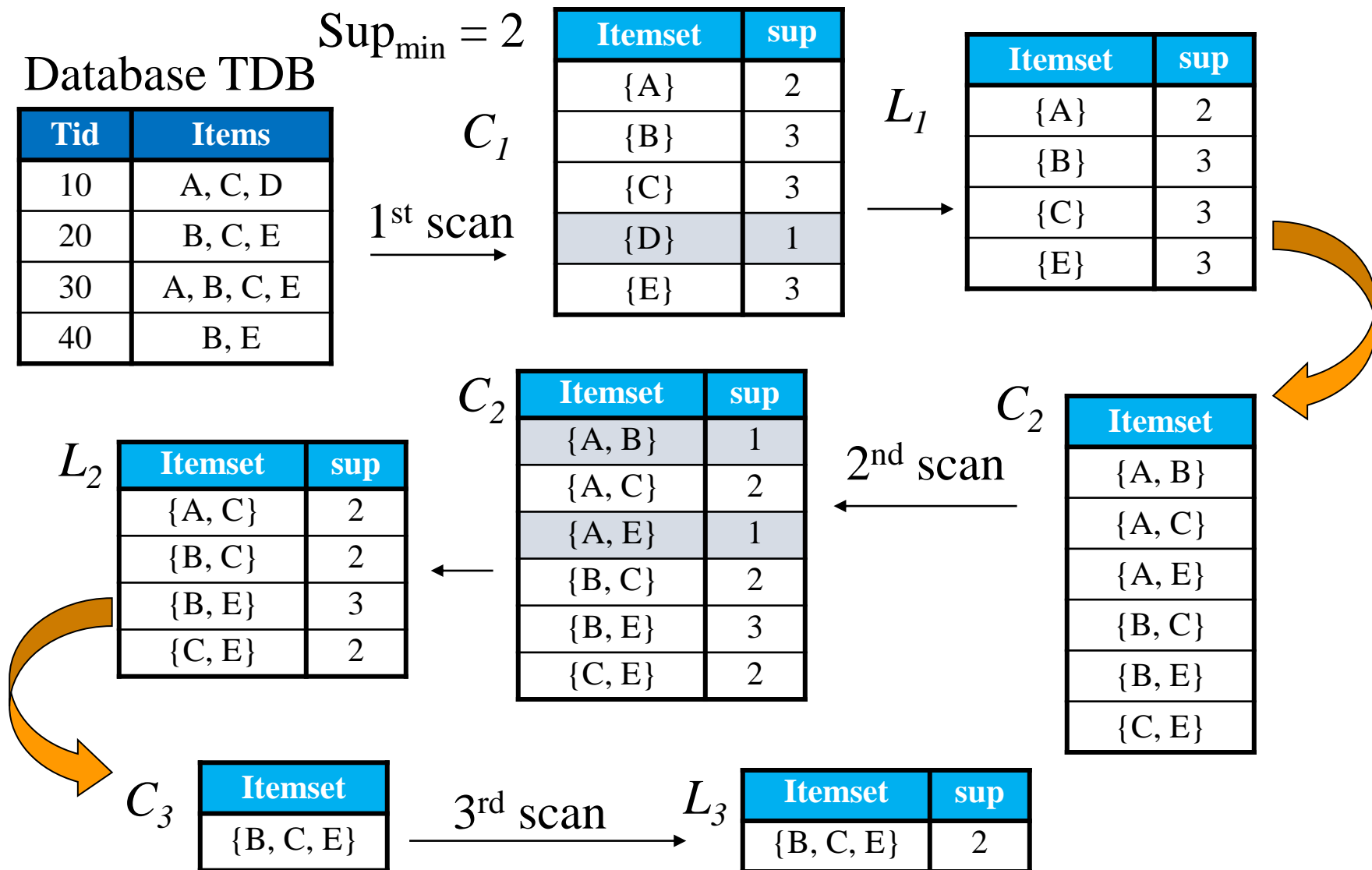
2.1 Apriori

- **先验原理:**
 - 如果一个项集是频繁的, 则它的所有子集一定也是频繁的
 - 相反, 如果一个项集是非频繁的, 则它的所有超集也一定是非频繁的

例子



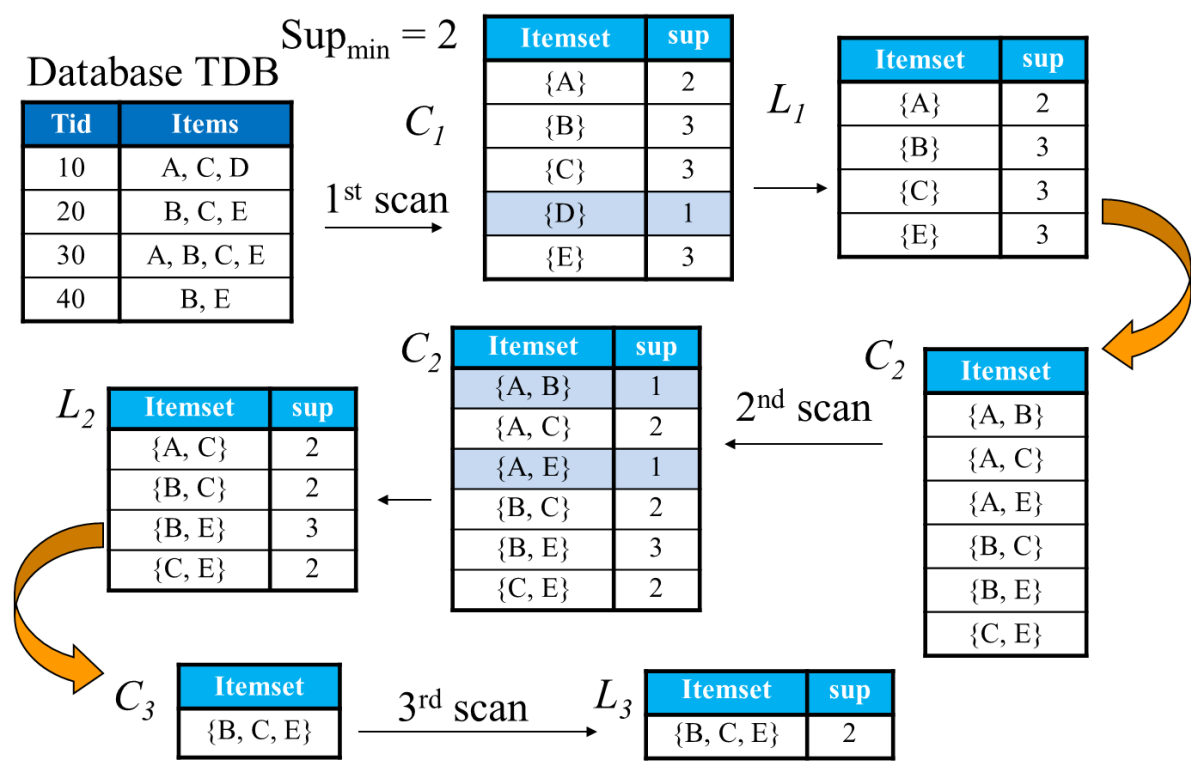
2.1 Apriori算法过程：最小支持度计数=2



2.5.2 Apriori算法注意问题 —— 项的字典序

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

尽管集合具有无序性，但为了快速连接操作，**通常对所有商品做一个默认的排序**（类似于建立一个字典索引）



2.5.2 Apriori算法注意问题 —— 项的连接

对于任何2个需要连接的项集

A,B,C

A,B,E

2.5.2 Apriori算法注意问题 —— 项的连接

对于任何2个需要连接的项集

A,B,~~C~~ **去掉第1个项集的尾项**

A,B,E

2.5.2 Apriori算法注意问题 —— 项的连接

对于任何2个需要连接的项集

A,B,~~C~~ 去掉第1个项集的尾项

A,B,~~E~~ 去掉第2个项集的尾项

2.5.2 Apriori算法注意问题 —— 项的连接

对于任何2个需要连接的项集

A,B, C	去掉第1个项集的尾项	若剩下的项 一样 →	A,B
A,B, E	去掉第2个项集的尾项		A,B

2.5.2 Apriori算法注意问题 —— 项的连接

对于任何2个需要连接的项集



2.5.2 Apriori算法注意问题 —— 项的连接

对于任何2个需要连接的项集

A,B,~~C~~ 去掉第1个项集的尾项

A,B,~~E~~ 去掉第2个项集的尾项

若剩下的项
一样

A,B 则可连接

A,B

A,B,C,E

对于任何2个需要连接的项集

A,B,~~D~~ 去掉第1个项集的尾项

B,C,~~E~~ 去掉第2个项集的尾项

若剩下的项
不一样

A,B

B,C

则不可连接

2.5.2 Apriori算法注意问题 —— 项的连接

对于任何2个需要连接的项集

A,B,~~C~~ 去掉第1个项集的尾项

A,B,~~E~~ 去掉第2个项集的尾项

若剩下的项
一样

A,B 则可连接

A,B

A,B,C,E

对于任何2个需要连接的项集

A,B,~~D~~ 去掉第1个项集的尾项

B,C,~~E~~ 去掉第2个项集的尾项

若剩下的项
不一样

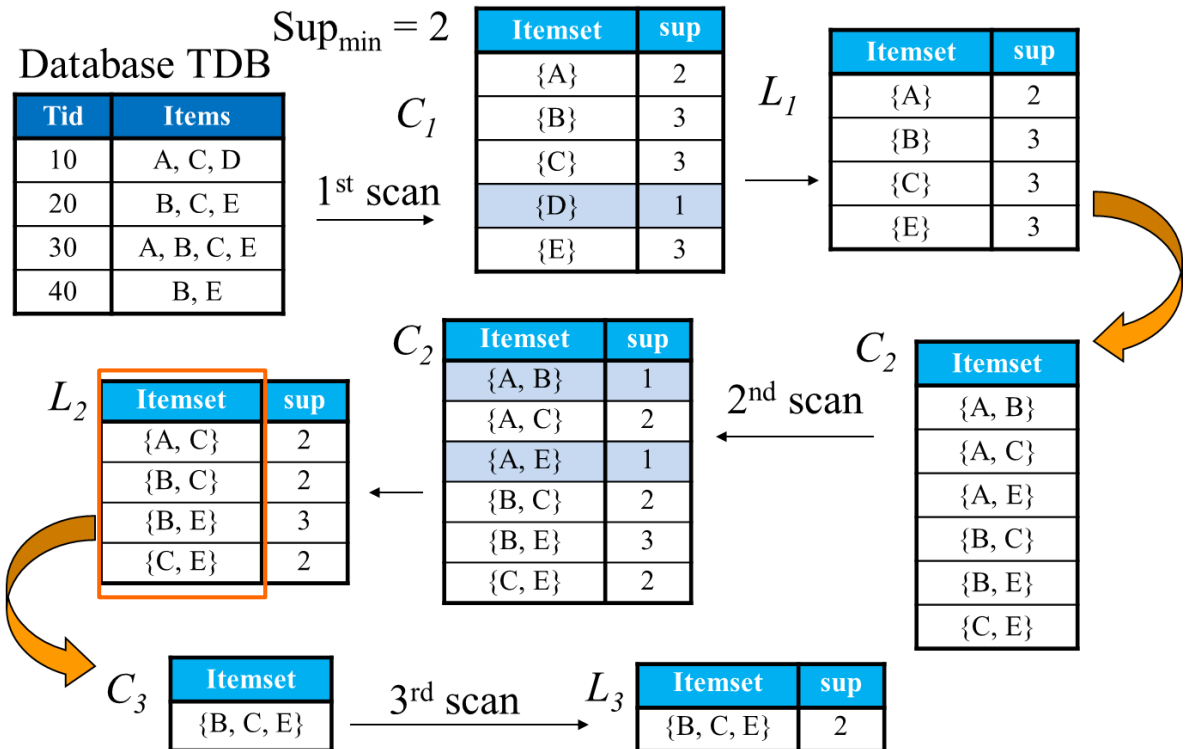
A,B

B,C

则不可连接

频繁2项集生成的候选

3项集是什么?



2.5.2 Apriori算法注意问题 —— 项的连接

对于任何2个需要连接的项集

A,B,~~C~~ 去掉第1个项集的尾项

A,B,~~E~~ 去掉第2个项集的尾项

若剩下的项
一样

A,B 则可连接

A,B,C,E

对于任何2个需要连接的项集

A,B,~~D~~ 去掉第1个项集的尾项

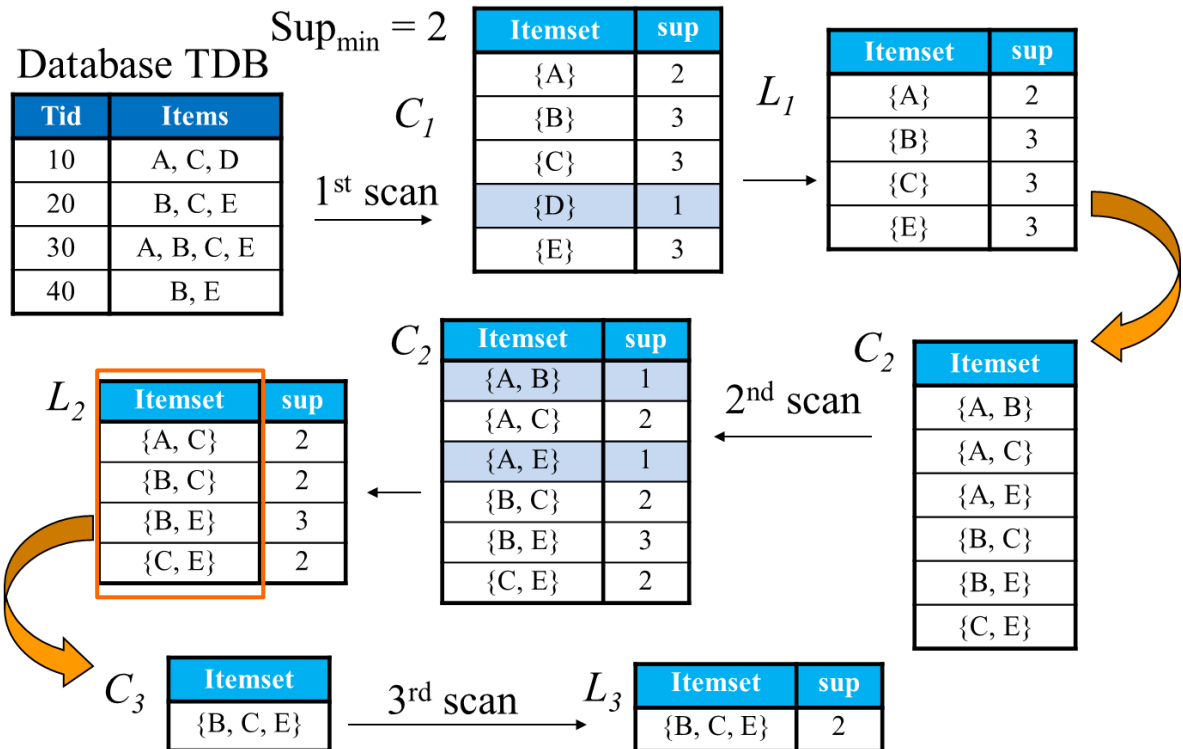
B,C,~~E~~ 去掉第2个项集的尾项

A,B

B,C

则不可连接

~~A,B,C~~
~~A,C,E~~
~~A,B,E~~
B,C,E



2.5.2 Apriori算法注意问题 —— 项的连接

对于任何2个需要连接的项集

A,B,~~C~~ 去掉第1个项集的尾项

A,B,~~E~~ 去掉第2个项集的尾项

若剩下的项
一样

A,B 则可连接

A,B,C,E

对于任何2个需要连接的项集

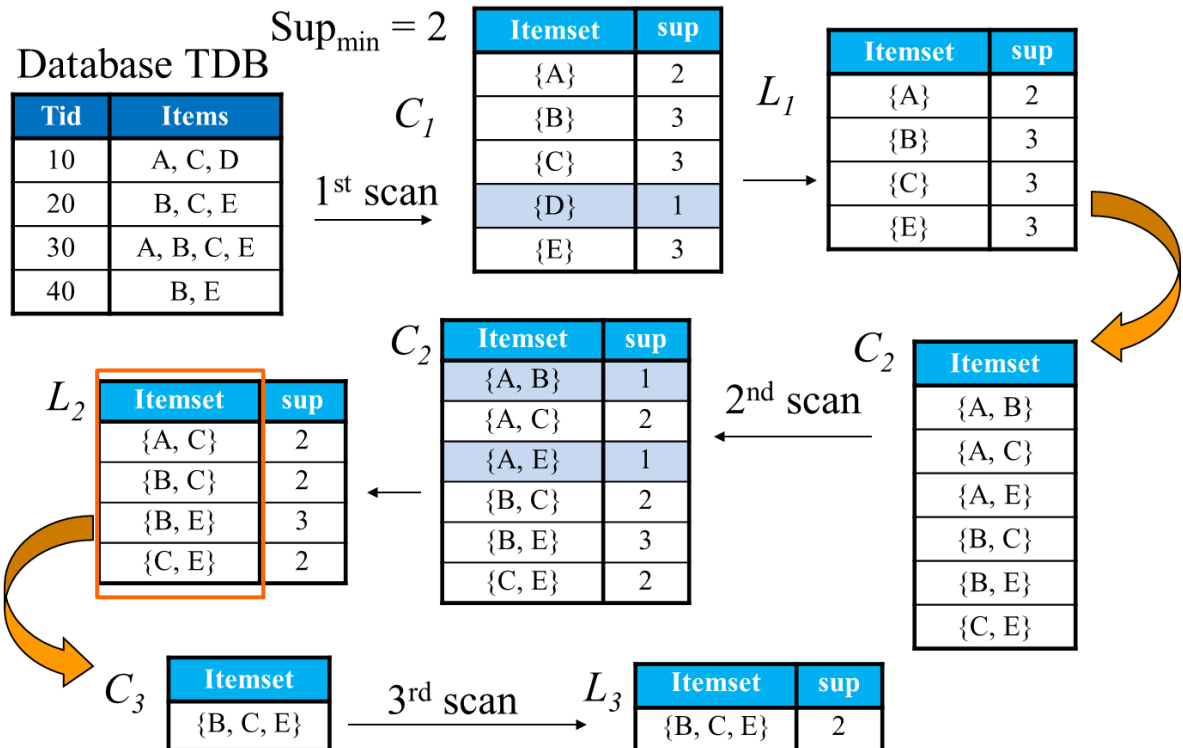
A,B,~~D~~ 去掉第1个项集的尾项

B,C,~~E~~ 去掉第2个项集的尾项

A,B

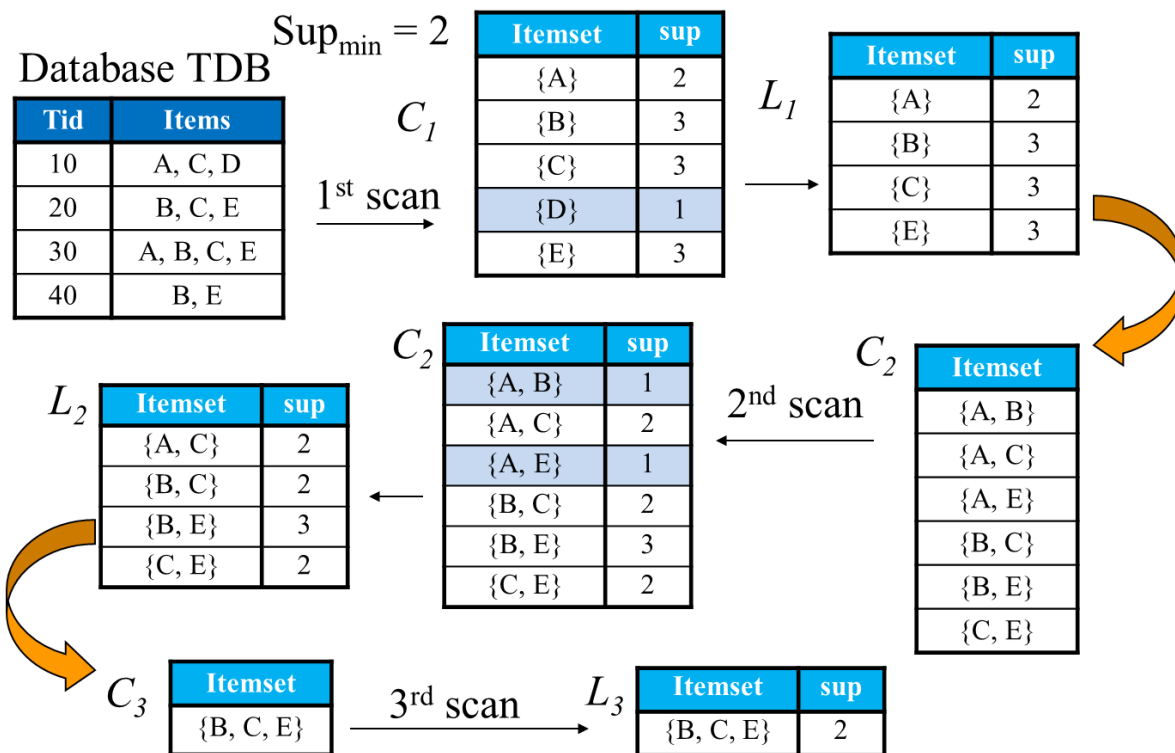
B,C

则不可连接



2.5.3 Apriori算法特点

- 多次扫描数据库
- 候选项规模庞大
- 计算支持度开销大



2.1 提高Apriori算法性能的方法

Hash-based
itemset counting

散列项集
计数

Transaction
reduction

事务压缩

Partitioning

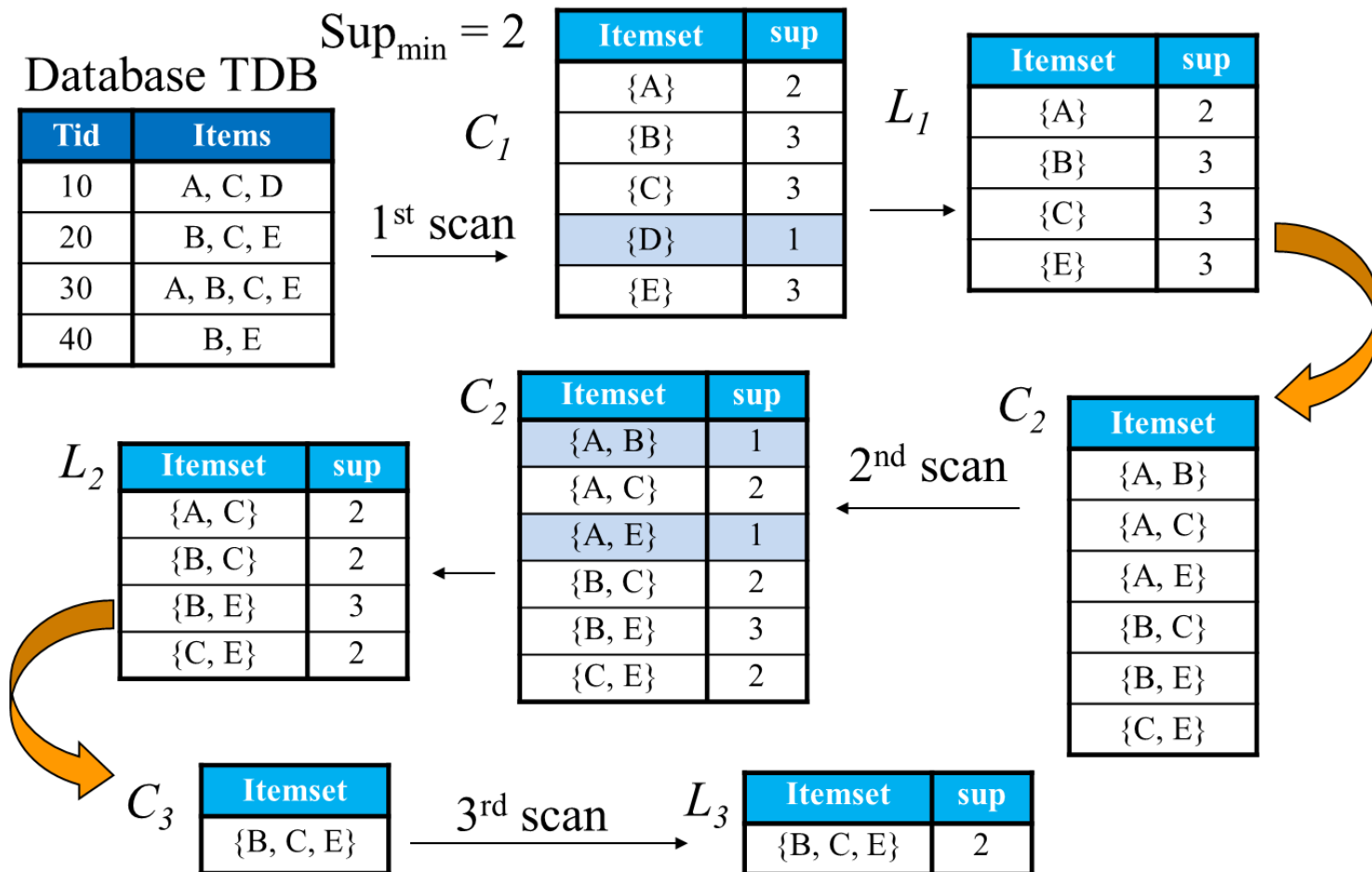
划分

Sampling

采样

2.1 Apriori算法缺点

Apriori算法需要反复的生成候选项，如果项的数目比较大，候选项的数目将达到**组合爆炸式的增长**



2.2 FPGrowth

- 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（FP-Tree）
 - 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，并把**每个事务映射到FP树中的一条路径来构造。**

2.2 构造FP树： 第一遍扫描

原始事务

TID	Items
1	{B,F,A}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{A}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}
11	{G}



删除支持度小于2的项

Item	Count
A	8
B	7
C	6
D	5
E	3
F	1
G	1



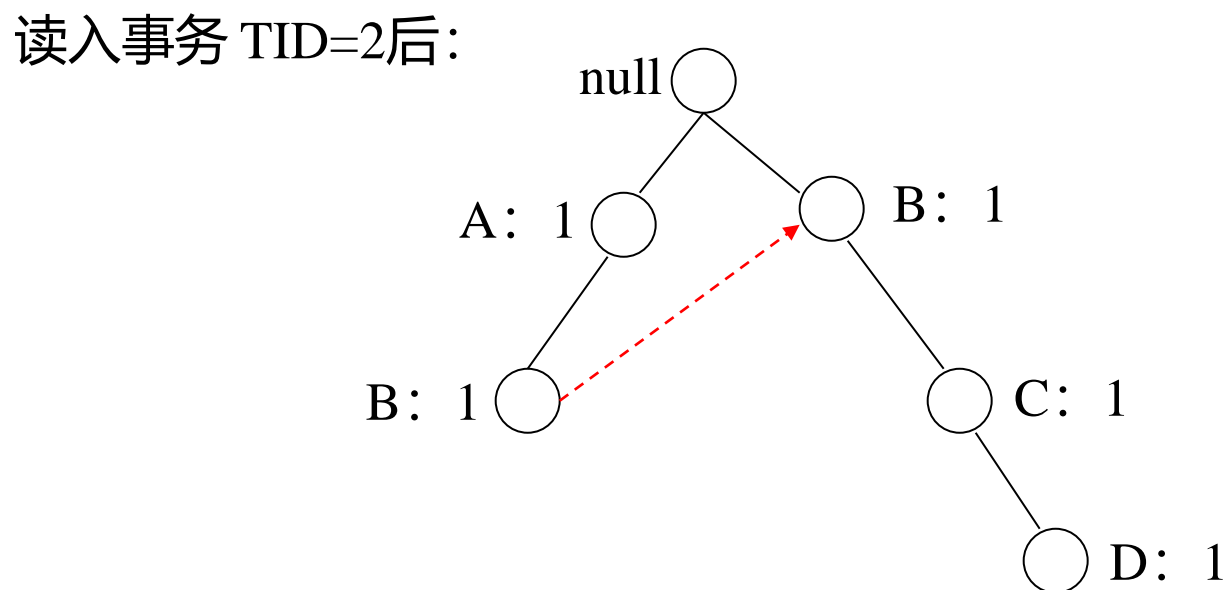
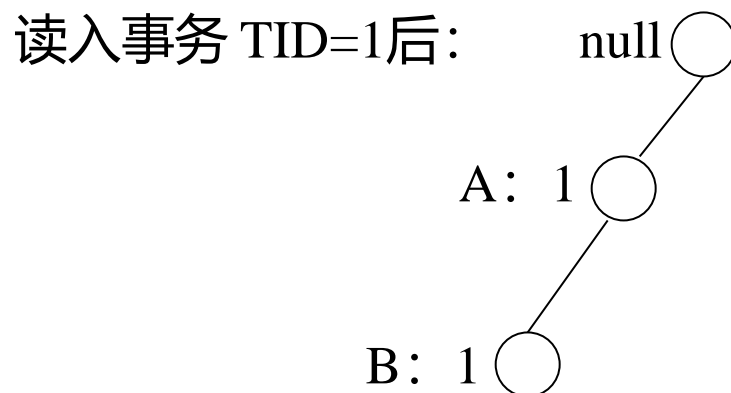
假设事先指定最小支持度计数为2

Items按照出现次数降序排列

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

2.2 构造FP树：第二遍扫描

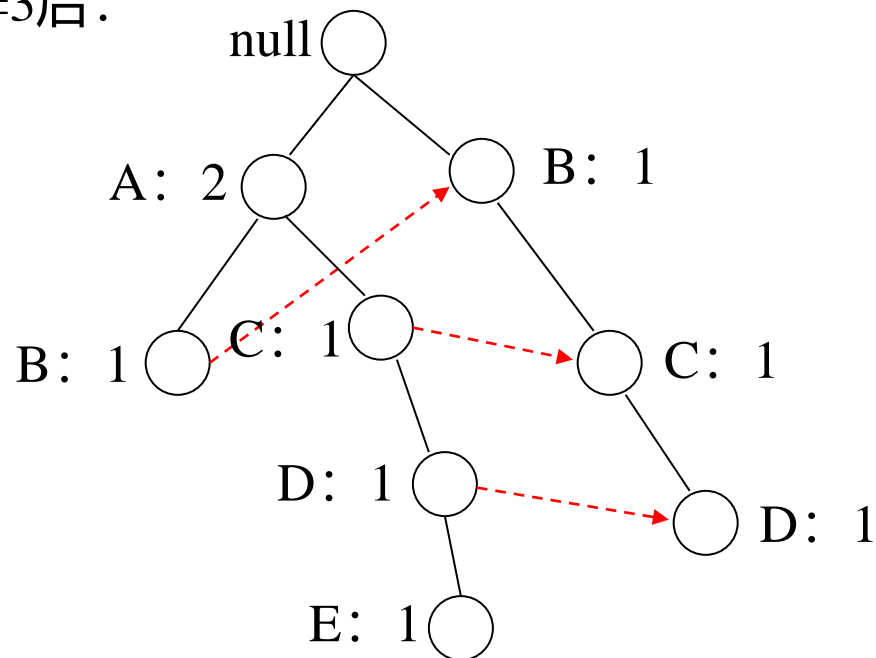
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{A}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}



2.2 构造FP树：第二遍扫描

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

读入事务 TID=3后：

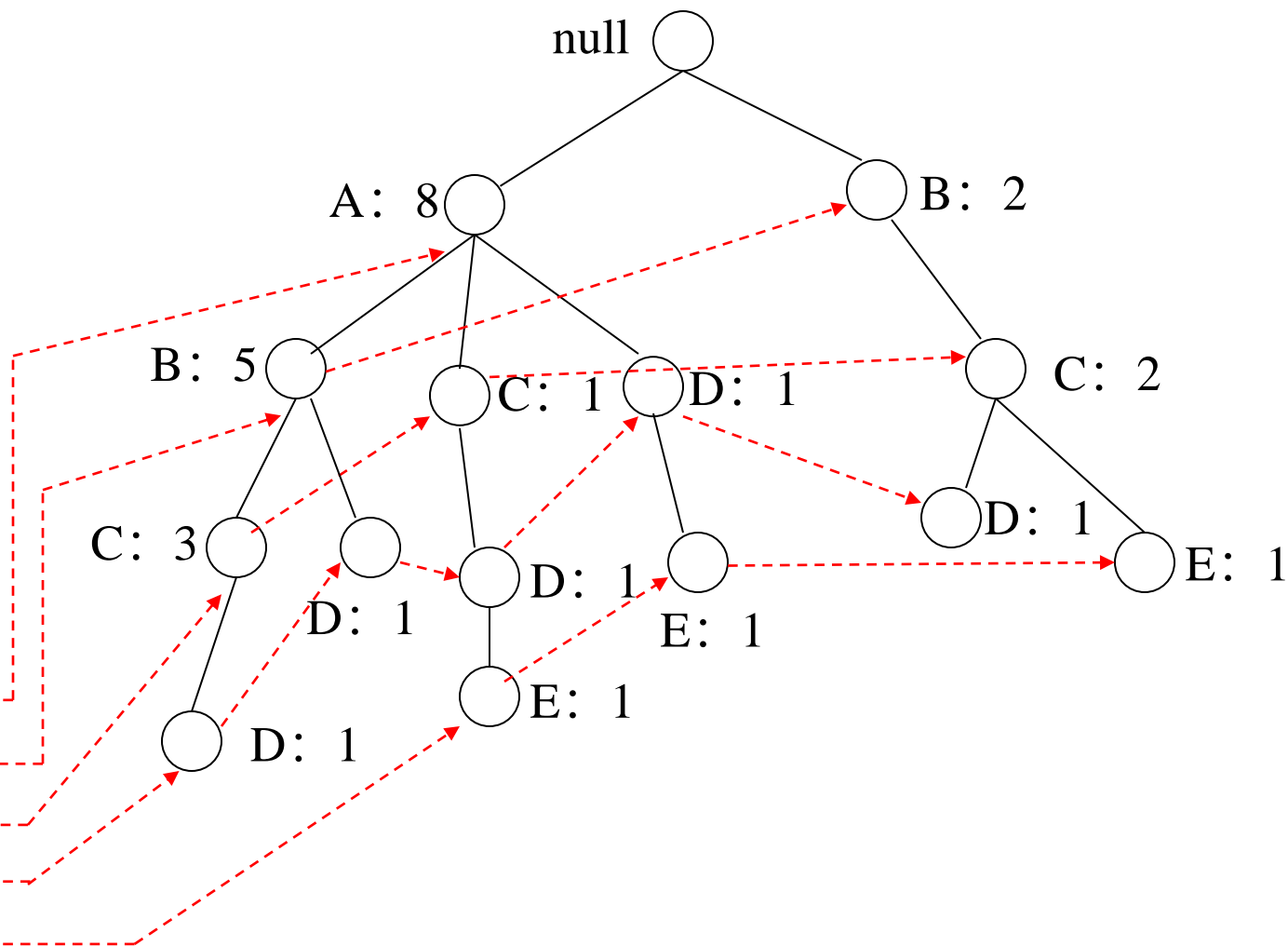


2.2 构造FP树：第二遍扫描

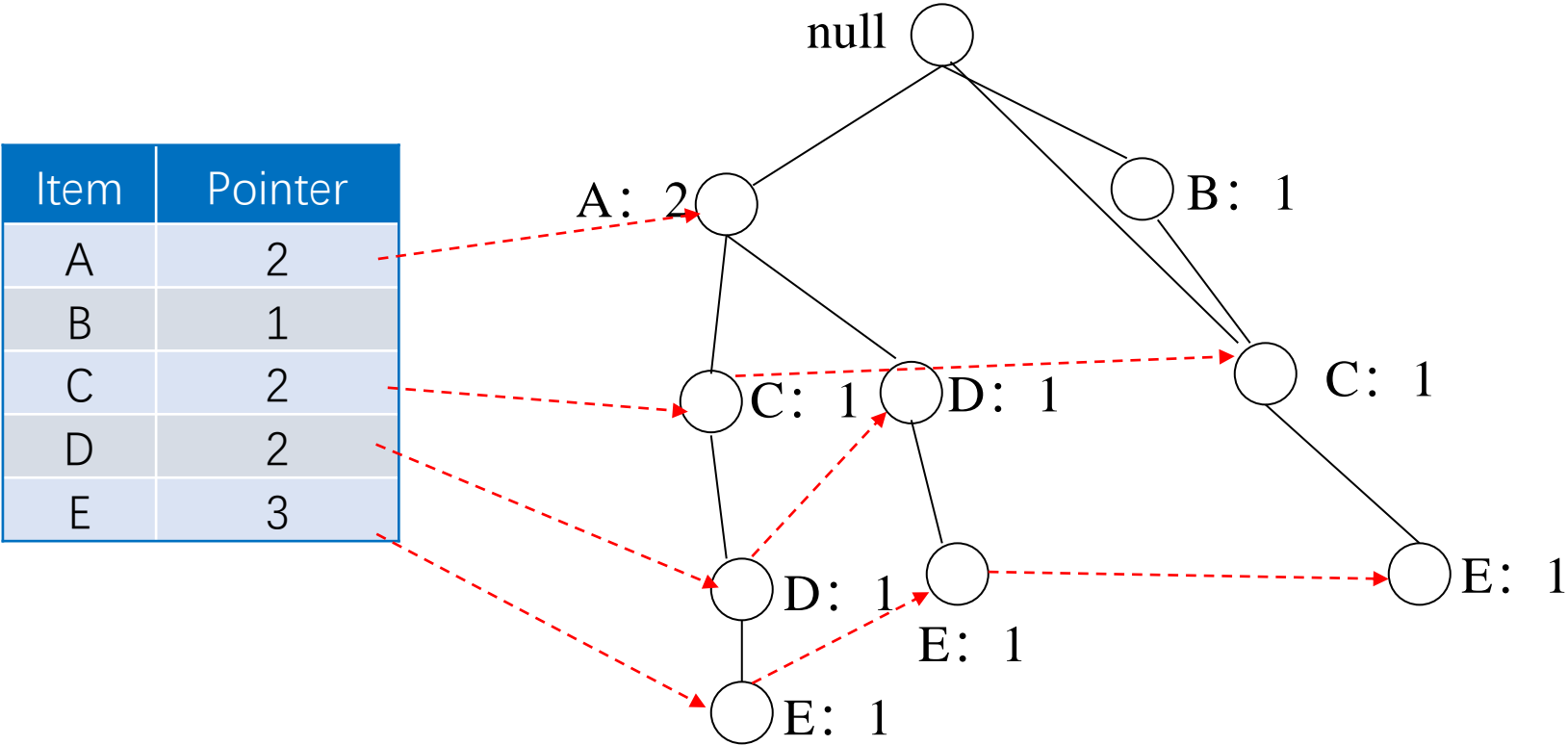
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{A}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

Header table

Item	Pointer
A
B
C
D
E



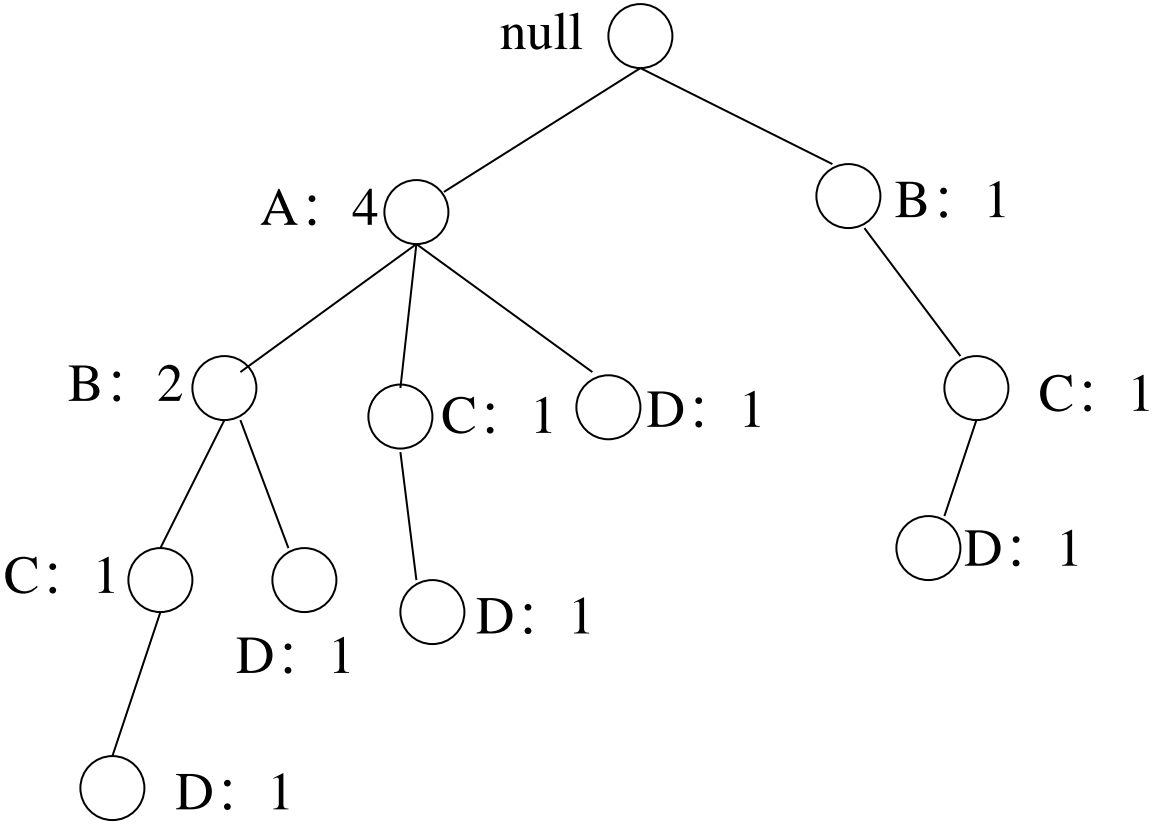
构造E节点的条件FP树



Item	条件模式基	条件FP树	频繁项集
E	{{ACD:1},{AD:1},{BC:1}}	<A:1,C:1,D:1>,<A:1,D:1>,<C:1>	{AE:2},{CE:2},{DE:2},{ACE:1},{ADE:2},{CDE:1},{ACDE:1}

构造D节点的条件FP树

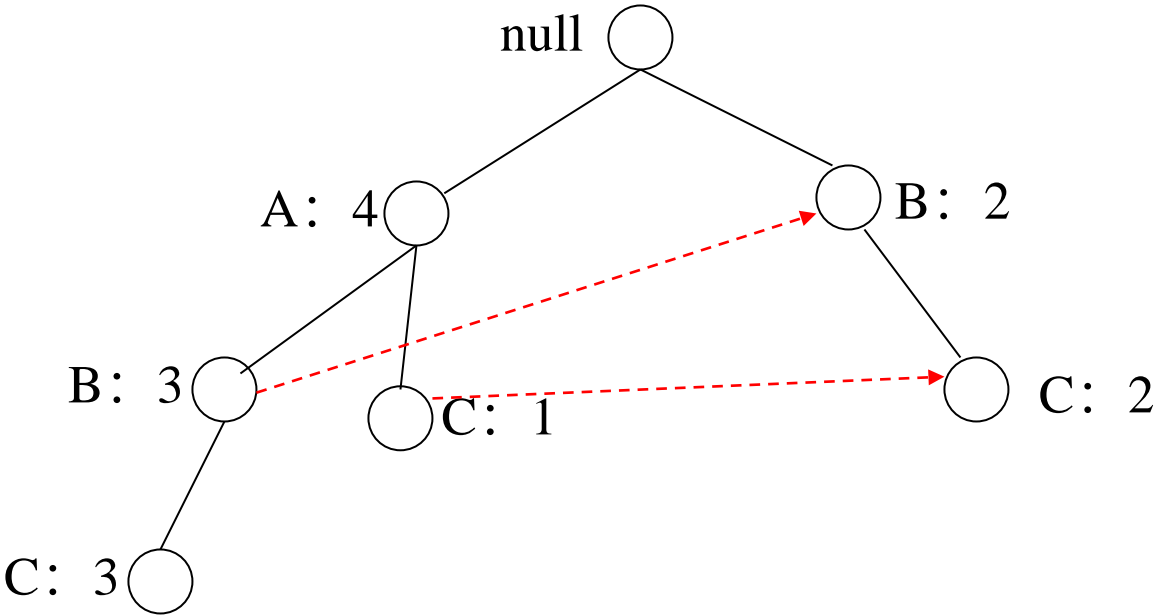
Item	Pointer
A	4
B	3
C	3
D	5



Item	条件模式基	条件FP树	频繁项集
E	{{ACD:1},{AD:1},{BD:1}}	<A:1,C:1,D:1>,<A:1,D:1>,<C:1>	{AE:2},{CE:2},{DE:2},{ACE:1},{ADE:2},{CDE:1},{ACDE:1}
D			

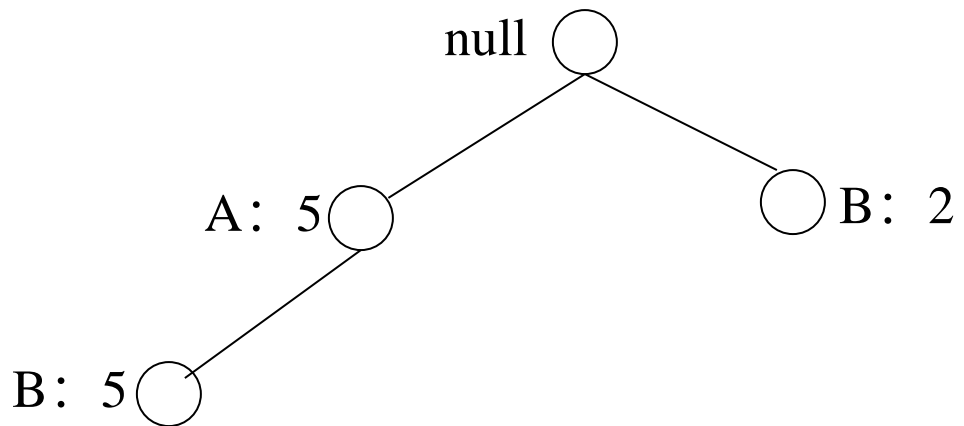
构造C节点的条件FP树

Item	Pointer
A	4
B	5
C	6



Item	条件模式基	条件FP树	频繁项集
E	{{ACD:1},{AD:1},{BD:1}}	<A:1,C:1,D:1>,<A:1,D:1>,<C:1>	{AE:2},{CE:2},{DE:2},{ACE:1},{ADE:2},{CDE:1},{ACDE:1}
D	{{ABC:1},{AB:1},{AC:1},{A:1},{BC:1}}	<A:1,B:1,C:1>,<A:1,B:1>,<A:1,C:1>,<A:1>,<B:1,C:1>	{AD:4},{BD:3},{CD:3},{ABD:2},{ACD:2},{BCD:2}
C			

Item	Pointer
A	5
B	7



Item	条件模式基	条件FP树	频繁项集
E	{{ACD:1},{AD:1},{BD:1}}	<A:1,C:1,D:1>,<A:1,D:1>,<C:1>	{AE:2},{CE:2},{DE:2},{ADE:2}
D	{{ABC:1},{AB:1},{AC:1},{A:1},{BC:1}}	<A:1,B:1,C:1>,<A:1,B:1>,<A:1,C:1>,<A:1>,<B:1,C:1>	{AD:4},{BD:3},{CD:3},{ABD:2},{ACD:2},{BCD:2}
C	{{AB:3},{A:1},{BC:2}}	<A:3,B:3>,<A:1>,<B:2>	{AC:4},{BC:5},{ABC:3}
B			

2.2 构造FP树：用FP-tree挖掘频繁集

- 基本思想 (分治)
 - 用FP-tree递归增长频繁集
- 方法
 - 对每个项，生成它的**条件模式基**，然后生成它的**条件 FP-tree**
 - 对每个新生成的条件FP-tree，重复这个步骤
 - 直到结果FP-tree为**空**，或只含**唯一的一个路径** (此路径的每个子路径对应的项集都是频繁集)

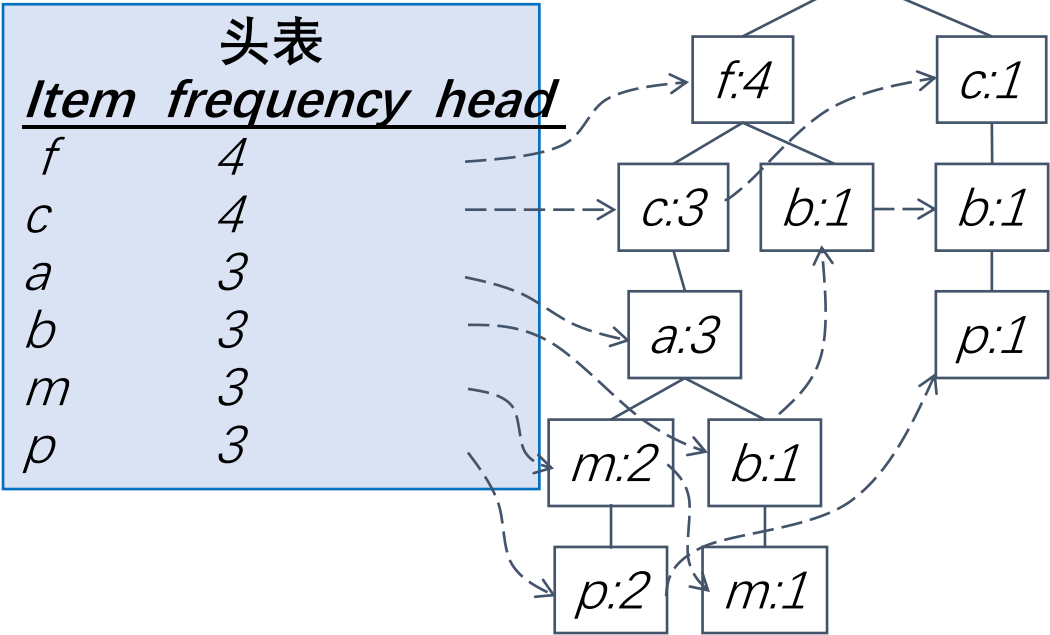
练习：构造FP树

<i>TID</i>	<i>Items bought</i>	<i>(ordered) frequent items</i>
100	{ <i>f, a, c, d, g, i, m, p</i> }	{ <i>f, c, a, m, p</i> }
200	{ <i>a, b, c, f, l, m, o</i> }	{ <i>f, c, a, b, m</i> }
300	{ <i>b, f, h, j, o</i> }	{ <i>f, b</i> }
400	{ <i>b, c, k, s, p</i> }	{ <i>c, b, p</i> }
500	{ <i>a, f, c, e, l, p, m, n</i> }	{ <i>f, c, a, m, p</i> }

最小支持度 = 0.5

步骤：

- 扫描数据库一次，得到频繁1-项集
- 把项按支持度递减排序
- 再一次扫描数据库，建立FP-tree



练习：构造FP树

<i>TID</i>	<i>Items bought</i>	<i>(ordered) frequent items</i>
100	{ <i>f, a, c, d, g, i, m, p</i> }	{ <i>f, c, a, m, p</i> }
200	{ <i>a, b, c, f, l, m, o</i> }	{ <i>f, c, a, b, m</i> }
300	{ <i>b, f, h, j, o</i> }	{ <i>f, b</i> }
400	{ <i>b, c, k, s, p</i> }	{ <i>c, b, p</i> }
500	{ <i>a, f, c, e, l, p, m, n</i> }	{ <i>f, c, a, m, p</i> }

最小支持度 = 0.5



步骤：

- 扫描数据库一次，得到频繁1-项集
- 把项按支持度递减排序
- 再一次扫描数据库，建立FP-tree

头表	
<i>Item frequency head</i>	
<i>f</i>	4
<i>c</i>	4
<i>a</i>	3
<i>b</i>	3
<i>m</i>	3
<i>p</i>	3

练习：构造FP树

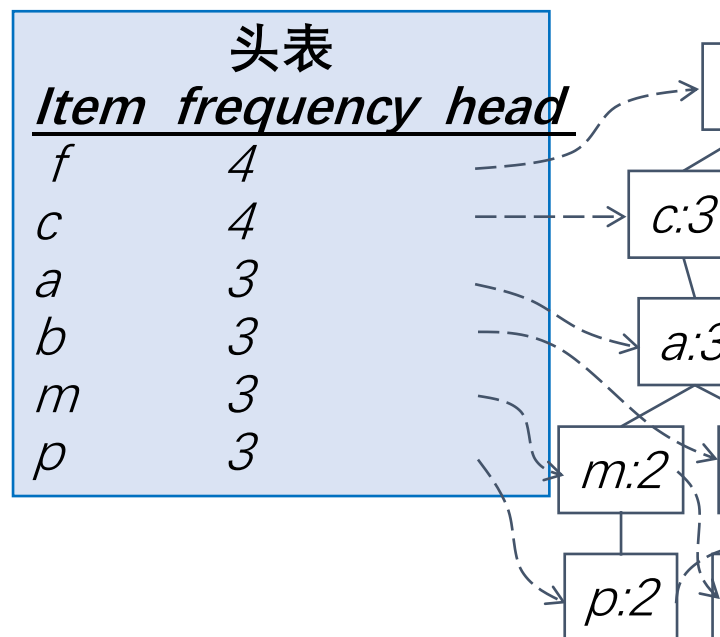
<i>TID</i>	<i>Items bought</i>	<i>(ordered) frequent items</i>
100	$\{f, a, c, d, g, i, m, p\}$	$\{f, c, a, m, p\}$
200	$\{a, b, c, f, l, m, o\}$	$\{f, c, a, b, m\}$
300	$\{b, f, h, j, o\}$	$\{f, b\}$
400	$\{b, c, k, s, p\}$	$\{c, b, p\}$
500	$\{a, f, c, e, l, p, m, n\}$	$\{f, c, a, m, p\}$

最小支持度 = 0.5

$$\{$$

步骤：

- 扫描数据库一次，得到频繁1-项集
- 把项按支持度递减排序
- 再一次扫描数据库，建立FP-tree



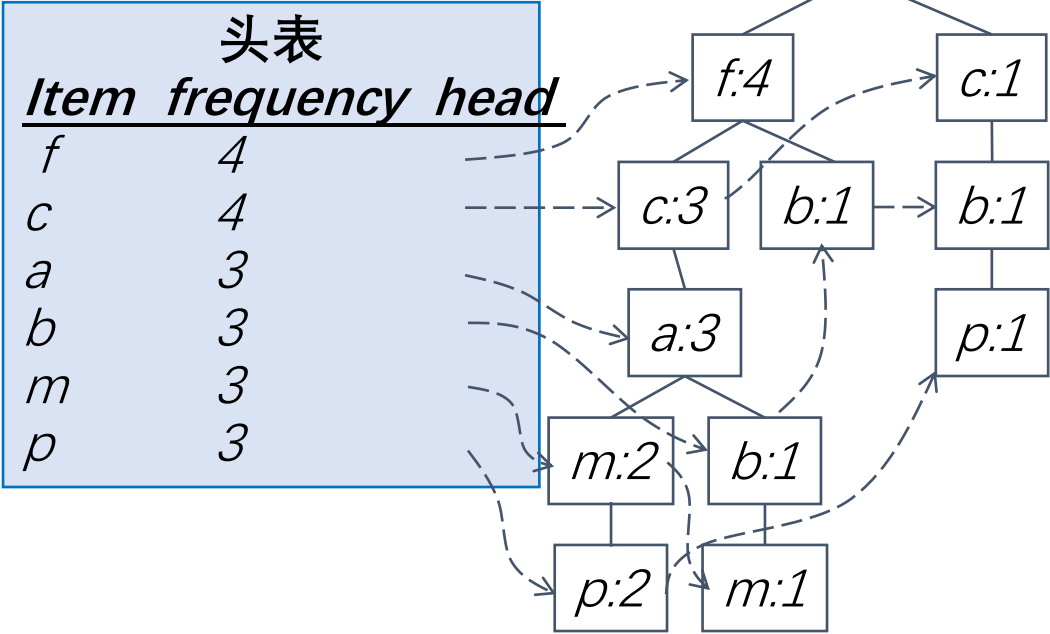
练习：构造FP树

<i>TID</i>	<i>Items bought</i>	<i>(ordered) frequent items</i>
100	{ <i>f, a, c, d, g, i, m, p</i> }	{ <i>f, c, a, m, p</i> }
200	{ <i>a, b, c, f, l, m, o</i> }	{ <i>f, c, a, b, m</i> }
300	{ <i>b, f, h, j, o</i> }	{ <i>f, b</i> }
400	{ <i>b, c, k, s, p</i> }	{ <i>c, b, p</i> }
500	{ <i>a, f, c, e, l, p, m, n</i> }	{ <i>f, c, a, m, p</i> }

最小支持度 = 0.5

步骤：

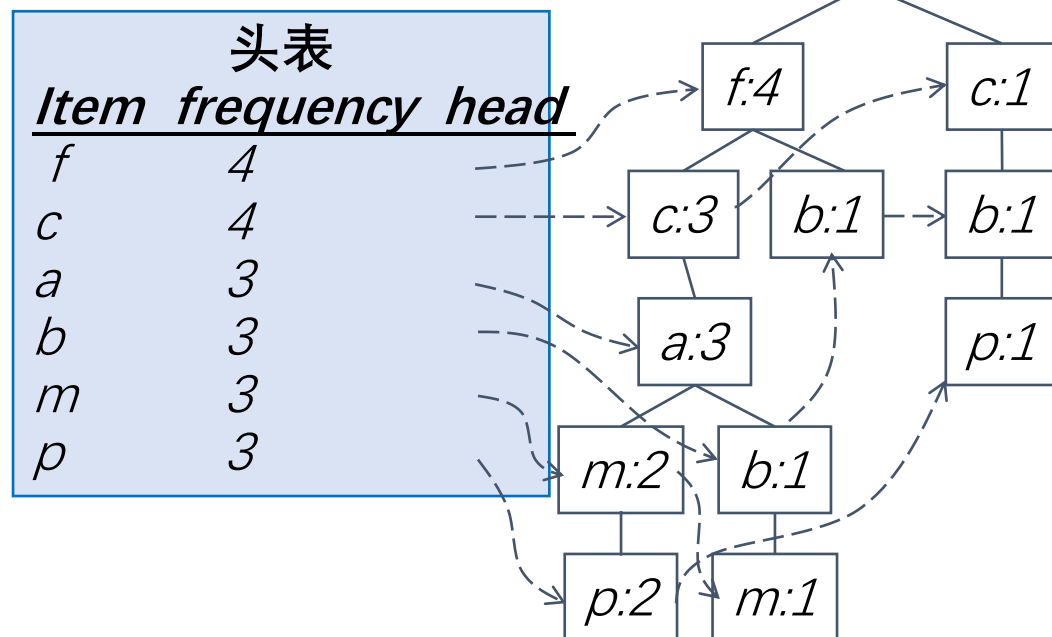
- 扫描数据库一次，得到频繁1-项集
- 把项按支持度递减排序
- 再一次扫描数据库，建立FP-tree



练习：生成条件模式

- 从FP-tree的头表开始
- 按照每个频繁项的连接遍历 FP-tree
- 列出能够到达此项的所有前缀路径，得到条件模式基

条件模式库	
<i>item</i>	<i>cond. pattern base</i>
<i>c</i>	<i>f</i> : 3
<i>a</i>	<i>fc</i> : 3
<i>b</i>	<i>fca</i> : 1, <i>f</i> : 1, <i>c</i> : 1
<i>m</i>	<i>fca</i> : 2, <i>fcab</i> : 1



P的条件模式基为：

A

f: 2,c: 2,a: 2,m: 2,p : 2

B

f: 2,c: 1,a: 1,m: 1,p : 1

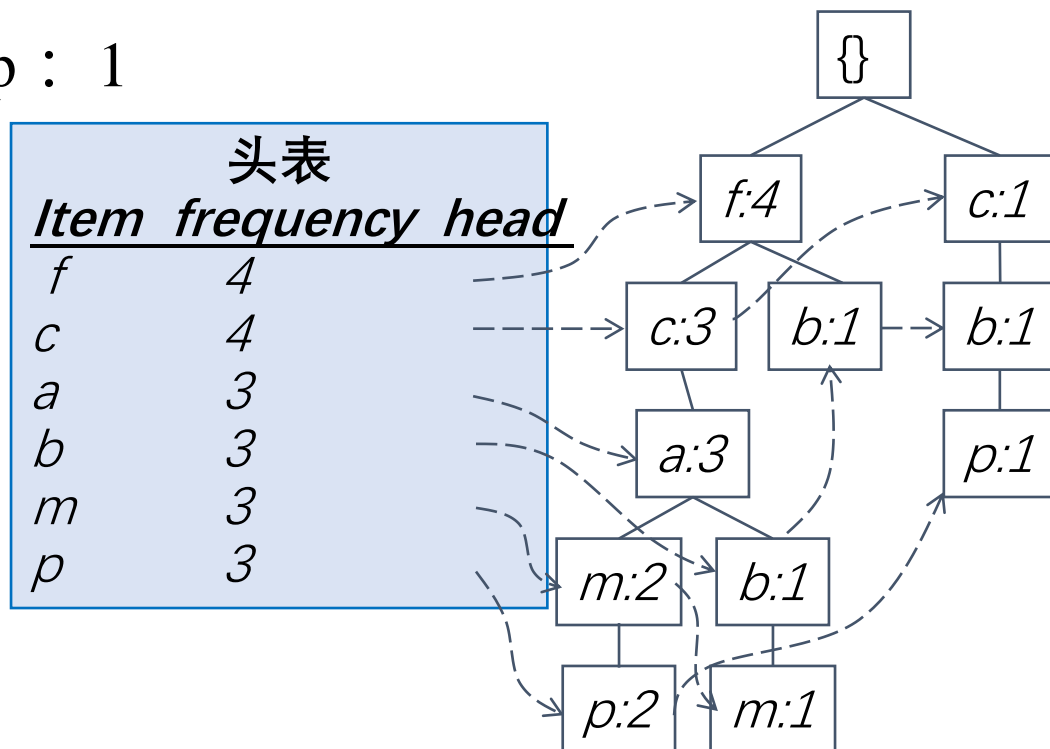
C

f: 1,c: 1,a: 1,m: 1,p : 1

D

c: 1,b: 1,p : 1

提交

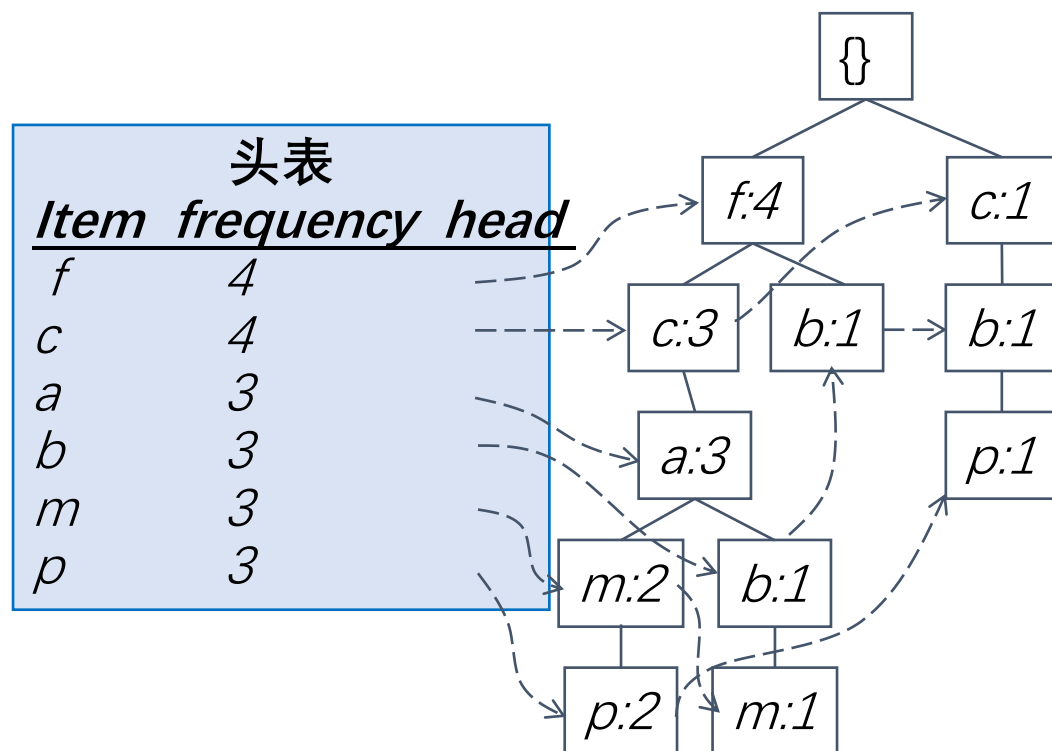


2.2 构造FP树：用FP-tree挖掘频繁集

- 基本思想 (分治)
 - 用FP-tree递归增长频繁集
- 方法
 - 对每个项，生成它的**条件模式基**，然后生成它的**条件 FP-tree**
 - 对每个新生成的条件FP-tree，重复这个步骤
 - 直到结果FP-tree为**空**，或只含**唯一的一个路径** (此路径的每个子路径对应的项集都是频繁集)

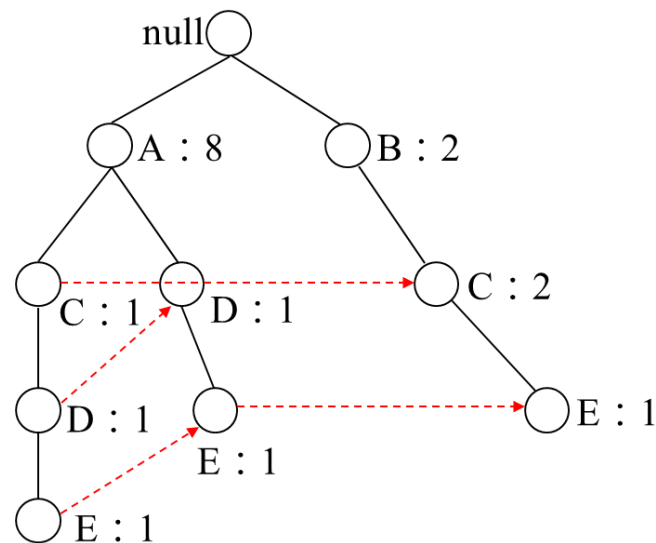
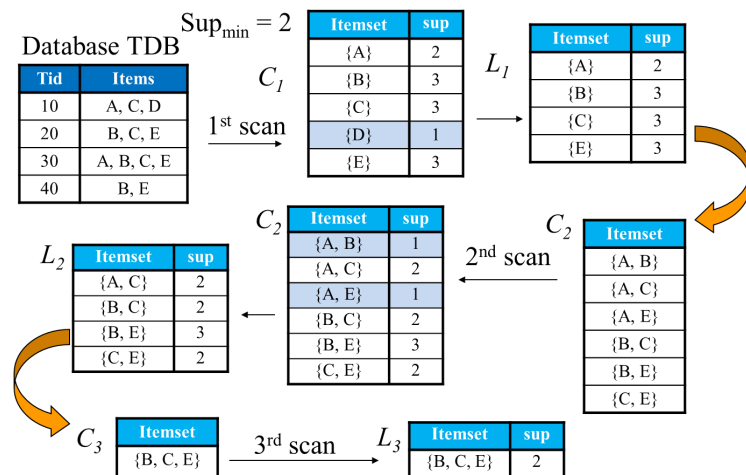
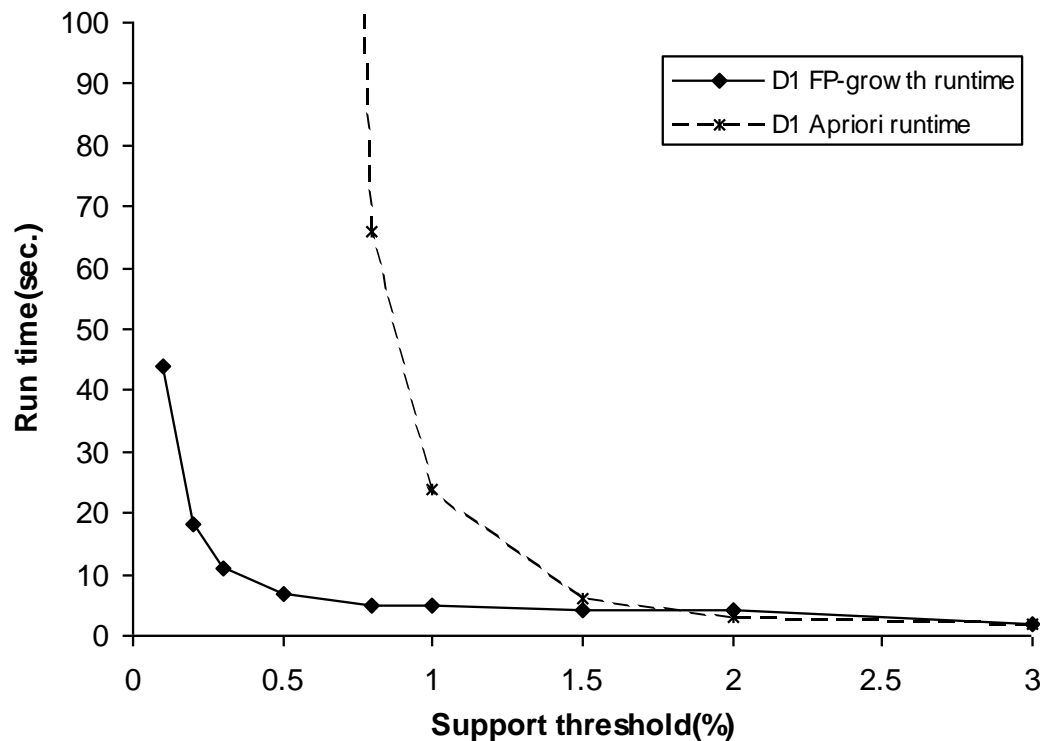
2.2 FP-tree 结构的优点

- 完备：
 - 不会打破交易中的任何模式
 - 包含了频繁模式挖掘所需的全部信息
- 紧密
 - 支持度降序排列：支持度高的项在FP-tree中共享的机会也高
 - 决不会比原数据库大



2.2 FP-tree 结构的优点 —— 性能对比

Data set T25I20D10K



2.3 挖掘关联规则 (Mining Association Rules)

- 大多数关联规则挖掘算法通常采用的一种策略是，将关联规则挖掘任务分解为如下两个主要的子任务：
 - **频繁项集产生 (Frequent Itemset Generation)**
 - 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 (Rule Generation)**
 - 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 (strong rule) 。

2.3 产生关联规则

- 任务描述：给定频繁项集 Y , 查找 Y 的所有非空真子集 $X \subset Y$, 使得 $X \rightarrow Y$ 的置信度超过最小置信度阈值 minconf
 - 例子：If $\{A,B,C\}$ is a frequent itemset, 候选规则如下：
 $AB \rightarrow C, \quad AC \rightarrow B, \quad BC \rightarrow A$
 $A \rightarrow BC, \quad B \rightarrow AC, \quad C \rightarrow AB$
- 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 $\emptyset \rightarrow Y$)

2.3 产生关联规则

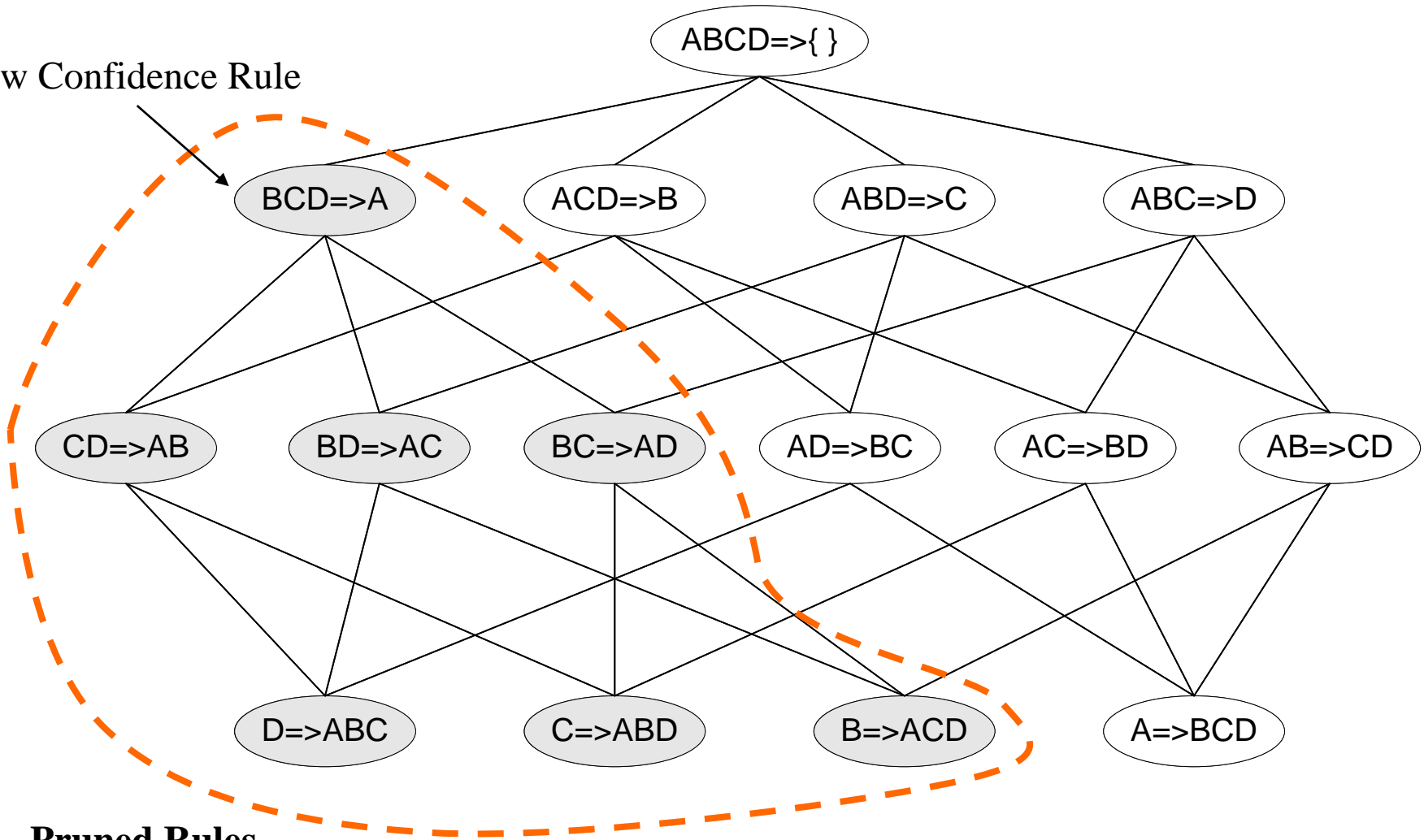
- How to efficiently generate rules from frequent itemsets?
 - 通常置信度不满足反单调性 (anti-monotone property) , 例如:
 - $c(ABC \rightarrow D)$ 可能大于也可能小于 $c(AB \rightarrow D)$
 - 但是, 针对同一个频繁项集的关联规则, 如果规则的后件满足子集关系, 那么这些规则的置信度间满足反单调性
 - e.g., $Y = \{A, B, C, D\}$:

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

2.3 Rule Generation for Apriori Algorithm

Lattice of rules

Low Confidence Rule



Pruned Rules

第三章 关联模式分析

- 3.1 基本概念
- 2.2 频繁项挖掘算法
- 2.3 关联分析的评估

3 关联分析的评估 (Pattern Evaluation)

- play basketball \Rightarrow eat cereal [支持度= , 置信度=]

	Basket ball	Not basketball	Sum (row)
Cereal 麦片	2000	1750	3750
Not cereal	1000	250	1250
Sum (col.)	3000	2000	5000

3 关联分析的评估 (Pattern Evaluation)

- play basketball \Rightarrow eat cereal [40%, 66.7%] is misleading
 - The overall % of students eating cereal is 75% > 66.7%.
- play basketball \Rightarrow not eat cereal [20%, 33.3%] is more accurate, although with lower support and confidence
- Measure of dependent/correlated events: **lift 提升度**

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

$$lift(B, C) = \frac{2000/5000}{3000/5000 * 3750/5000} = 0.89$$

$$lift(B, \neg C) = \frac{1000/5000}{3000/5000 * 1250/5000} = 1.33$$

	Basket ball	Not basketball	Sum (row)
Cereal	2000	1750	3750
Not cereal	1000	250	1250
Sum (col.)	3000	2000	5000