

# Vorlesung Wissensentdeckung Apriori

Katharina Morik, Claus Weihs

LS 8 Informatik  
 Computergestützte Statistik  
 Technische Universität Dortmund

12.4.2010

## Lernen von Assoziationsregeln

Gegeben:

- $R$  eine Menge von Objekten, die binäre Werte haben
- $t$  eine Transaktion,  $t \subseteq R$
- $r$  eine Menge von Transaktionen
- $s_{min} \in [0, 1]$  die minimale Unterstützung,
- $conf_{min} \in [0, 1]$  die minimale Konfidenz

Finde alle Regeln  $c$  der Form  $X \rightarrow Y$ , wobei  $X \subseteq R, Y \subseteq R, X \cap Y = \{\}$

$$s(r, c) = \frac{|\{t \in r \mid X \cup Y \in t\}|}{|r|} \geq s_{min} \quad (1)$$

$$conf(r, c) = \frac{|\{t \in r \mid X \cup Y \in t\}|}{|\{t \in r \mid X \in r\}|} \geq conf_{min} \quad (2)$$

## Gliederung

1 Apriori

2 FP-Tree

## Binäre Datenbanken

Sei  $R$  eine Menge von Objekten, die binäre Werte haben, und  $r$  eine Menge von Transaktionen, dann ist  $t \subseteq R$  eine Transaktion.

$$R = \{A, B, C\}$$

$$t = \{B, C\} \in R$$

A	B	C	ID
0	1	1	1
1	1	0	2
0	1	1	3
1	0	0	4

## Warenkorbanalyse

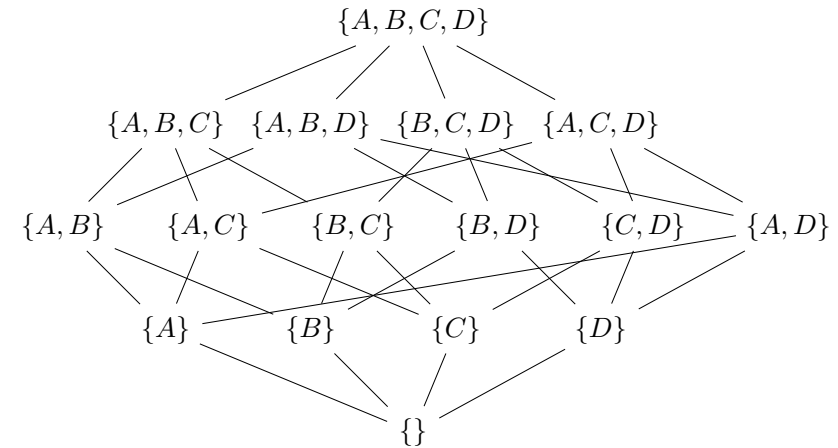
Aftershave	Bier	Chips	EinkaufsID
0	1	1	1
1	1	0	2
0	1	1	3
1	0	0	4

- {Aftershave} → {Bier}  $s = \frac{1}{4}, conf = \frac{1}{2}$
- {Aftershave} → {Chips}  $s = 0$
- {Bier} → {Chips}  $s = \frac{1}{2}, conf = \frac{2}{3}$  (zusammen anbieten?)
- {Chips} → {Aftershave}  $s = 0$
- {Aftershave} → {Bier, Chips}  $s = 0$

## Ordnungsrelation

- Hier ist die Ordnungsrelation die Teilmengenbeziehung.
- Eine Menge  $S_1$  ist größer als eine Menge  $S_2$ , wenn  $S_1 \supseteq S_2$ .
- Eine kleinere Menge ist allgemeiner.

## Verband



## Assoziationsregeln

*LH*: Assoziationsregeln sind keine logischen Regeln!

- In der Konklusion können mehrere Attribute stehen
- Attribute sind immer nur binär.
- Mehrere Assoziationsregeln zusammen ergeben kein Programm.

*LE*: Binärvektoren (Transaktionen)

- Attribute sind eindeutig geordnet.

Aufgabe:

- Aus häufigen Mengen Assoziationsregeln herstellen

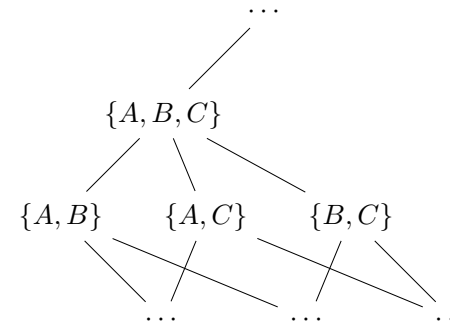
# Apriori Algorithmus (Agrawal, Mannila, Srikant, Toivonen, Verkamo 1996)

LH des Zwischenschritts: Häufige Mengen  $L_k = X \cup Y$  mit  $k$  Objekten (large itemsets, frequent sets)

- Wenn eine Menge häufig ist, so auch all ihre Teilmengen. (Anti-Monotonie)
- Wenn eine Menge selten ist, so auch all ihre Obermengen. (Monotonie)
- Wenn  $X$  in  $L_{k+1}$  dann alle  $S_i \subseteq X$  in  $L_k$  (Anti-Monotonie)
- Alle Mengen  $L_k$ , die  $k - 1$  Objekte gemeinsam haben, werden vereinigt zu  $L_{k+1}$ .

Dies ist der Kern des Algorithmus, die Kandidatengenerierung.

# Beispiel



- Wenn  $\{A, B, C\}$  häufig ist, dann sind auch  $\{A, B\}$ ,  $\{A, C\}$ ,  $\{B, C\}$  häufig.
- Das bedeutet, daß  $\{A, B\}$ ,  $\{A, C\}$ ,  $\{B, C\}$  ( $k = 2$ ) häufig sein müssen, damit  $\{A, B, C\}$  ( $k + 1 = 3$ ) häufig sein kann.
- Also ergeben die häufigen Mengen aus  $L_k$  die Kandidaten  $C_{k+1}$

# Beispiel

Gesucht werden Kandidaten mit  $k + 1 = 5$   
 $L_4 = \{\{ABCD\}, \{ABCE\}, \{ABDE\}, \{ACDE\}, \{BCDE\}\}$

- $k - 1$  Stellen gemeinsam vereinigen zu:

$$l = \{ABCDE\}$$

- Sind alle  $k$  langen Teilmengen von  $l$  in  $L_4$ ?  
 $\{ABCD\}\{ABCE\}\{ABDE\}\{ACDE\}\{BCDE\}$  - ja!
- Dann wird  $l$  Kandidat  $C_5$ .

$L_4 = ABCD, ABCE$   
 $l = ABCDE$

- Sind alle Teilmengen von  $l$  in  $L_4$ ?  
 $\{ABCD\}\{ABCE\}\{ABDE\}\{ACDE\}\{BCDE\}$  - nein!
- Dann wird  $l$  nicht zum Kandidaten.

# Kandidatengenerierung

- Erzeuge-Kandidaten( $L_k$ )

- $C_{k+1} := \{\}$
- For all  $l_1, l_2$  in  $L_k$ , sodass

$$l_1 = \{i_1, \dots, i_{k-1}, i_k\} \text{ und}$$

$$l_2 = \{i_1, \dots, i_{k-1}, i'_k\} \text{ mit } i'_k < i_k$$

- $l := \{i_1, \dots, i_{k-1}, i_k, i'_k\}$
- if alle  $k$ -elementigen Teilmengen von  $l$  in  $L_k$  sind, then

$$C_{k+1} := C_{k+1} \cup \{l\}$$

- return  $C_{k+1}$

- Prune( $C_{k+1}, r$ ) vergleicht Häufigkeit von Kandidaten mit  $s_{min}$ .

## Häufige Mengen

### ● Häufige-Mengen( $R, r, s_{min}$ )

- $C_1 := \cup_{i \in R} i, k = 1$
- $L_1 := \text{Prune}(C_1)$
- while  $L_k \neq \{\}$ 
  - $C_{k+1} := \text{Erzeuge-Kandidaten}(L_k)$
  - $L_{k+1} := \text{Prune}(C_{k+1}, r)$
  - $k := k + 1$
  - return  $\cup_{j=2}^k L_j$

## Regelgenerierung

Aus den häufigen Mengen werden Regeln geformt. Wenn die Konklusion länger wird, kann die Konfidenz sinken. Die Ordnung der Attribute wird ausgenutzt:

$$\begin{array}{lll}
 l_1 = \{i_1, \dots, i_{k-1}, i_k\} & c_1 = \{i_1, \dots, i_{k-1}\} \rightarrow \{i_k\} & conf_1 \\
 l_1 = \{i_1, \dots, i_{k-1}, i_k\} & c_2 = \{i_1, \dots\} \rightarrow \{i_{k-1}, i_k\} & conf_2 \\
 \dots & \dots & \dots \\
 l_1 = \{i_1, \dots, i_{k-1}, i_k\} & c_k = \{i_1\} \rightarrow \{\dots, i_{k-1}, i_k\} & conf_k
 \end{array}$$

$$conf_1 \geq conf_2 \geq \dots \geq conf_k$$

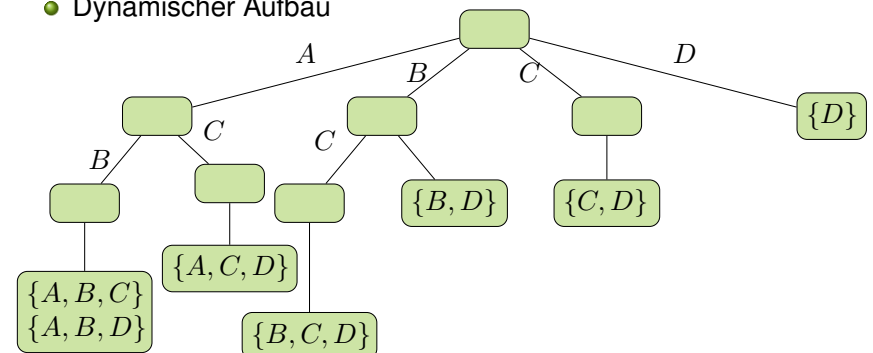
## APRIORI

### ● Apriori( $R, r, s_{min}, conf_{min}$ )

- $L := \text{Häufige-Mengen}(R, r, s_{min})$
- $c := \text{Regeln}(L, conf_{min})$
- return  $c$

## Implementierung

- Hash-Tree für den Präfixbaum, der sich aus der Ordnung der Elemente in den Mengen ergibt.
- An jedem Knoten werden Schlüssel und Häufigkeit gespeichert.
- Dynamischer Aufbau



## Was wissen Sie jetzt?

- Assoziationsregeln sind keine logischen Regeln.
- Anti-Monotonie der Häufigkeit: Wenn eine Menge häufig ist, so auch all ihre Teilmengen.
- Man erzeugt häufige Mengen, indem man häufige Teilmengen zu einer Menge hinzufügt und diese Mengen dann auf Häufigkeit testet. Bottom-up Suche im Verband der Mengen.
- Monotonie der Seltenheit: Wenn eine Teilmenge selten ist, so auch jede Menge, die sie enthält.
- Man beschneidet die Suche, indem Mengen mit einer seltenen Teilmenge nicht weiter betrachtet werden.

## Prinzipien für Regelbewertungen

- 1  $RI(A \rightarrow B) = 0$ , wenn  $|A \rightarrow B| = \frac{|A||B|}{|r|}$   
 $A$  und  $B$  sind unabhängig.
- 2  $RI(A \rightarrow B)$  steigt monoton mit  $|A \rightarrow B|$ .
- 3  $RI(A \rightarrow B)$  fällt monoton mit  $|A|$  oder  $|B|$ .

Also:

- $RI > 0$ , wenn  $|A \rightarrow B| > \frac{|A||B|}{|r|}$ , d.h. wenn  $A$  positiv mit  $B$  korreliert ist.
- $RI < 0$ , wenn  $|A \rightarrow B| < \frac{|A||B|}{|r|}$ , d.h. wenn  $A$  negativ mit  $B$  korreliert ist.

Wir wissen, dass immer  $|A \rightarrow B| \leq |A| \leq |B|$  gilt, also

- $RI_{min}$ , wenn  $|A \rightarrow B| = |A|$  oder  $|A| = |B|$
- $RI_{max}$ , wenn  $|A \rightarrow B| = |A| = |B|$

Piatetsky-Shapiro 1991

## Probleme von Apriori

- Im schlimmsten Fall ist Apriori exponentiell in  $R$ , weil womöglich alle Teilmengen gebildet würden. In der Praxis sind die Transaktionen aber spärlich besetzt. Die Beschneidung durch  $s_{min}$  und  $conf_{min}$  reicht bei der Warenkorbanalyse meist aus.
- Apriori liefert unglaublich viele Regeln.
- Die Regeln sind höchst redundant.
- Die Regeln sind irreführend, weil die Kriterien die a priori Wahrscheinlichkeit nicht berücksichtigen. Wenn sowieso alle Cornflakes essen, dann essen auch hinreichend viele FuSSballer Cornflakes.

## Konfidenz

- Die Konfidenz erfüllt die Prinzipien nicht! (Nur das 2.) Auch unabhängige Mengen  $A$  und  $B$  werden als hoch-konfident bewertet.
- Die USA-Census-Daten liefern die Regel  
 aktiv-militär  $\rightarrow$  kein-Dienst-in-Vietnam  
 mit 90% Konfidenz. Tatsächlich ist  
 $s(\text{kein-Dienst-in-Vietnam}) = 95\%$  Es wird also wahrscheinlicher, wenn aktiv-militär gegeben ist!
- Gegeben eine Umfrage unter 2000 Schülern, von denen 60% Basketball spielen, 75% Cornflakes essen. Die Regel

Basketball  $\rightarrow$  Cornflakes

hat Konfidenz 66% Tatsächlich senkt aber Basketball die Cornflakes Häufigkeit!

## Signifikanztest

- Ein einfaches Maß, das die Prinzipien erfüllt, ist:

$$|A \rightarrow B| - \frac{|A||B|}{|r|}$$

- Die Signifikanz der Korrelation zwischen  $A$  und  $B$  ist:

$$\frac{|A \rightarrow B| - \frac{|A||B|}{|r|}}{\sqrt{|A||B| \left(1 - \frac{|A|}{r}\right) \left(1 - \frac{|B|}{r}\right)}}$$

## Was wissen Sie jetzt?


- Sie haben drei Prinzipien für die Regelbewertung kennengelernt:
  - Unabhängige Mengen sollen mit 0 bewertet werden.
  - Der Wert soll höher werden, wenn die Regel mehr Belege hat.
  - Der Wert soll niedriger werden, wenn die Mengen weniger Belege haben.
- Sie haben drei Maße kennen gelernt, die den Prinzipien genügen:
  - Einfaches Maß
  - statistisches Maß und
  - Sicherheitsmaß

## Sicherheitsmaß

Shortliffe, Buchanan 1990 führten ein Sicherheitsmaß  $CF$  ein (für Regeln in Wissensbasen)

- Wenn  $conf(A \rightarrow B) > s(B)$   
 $CF(A \rightarrow B) = conf(A \rightarrow B) - \frac{s(B)}{1-s(B)}$
- Wenn  $conf(A \rightarrow B) < s(B)$   
 $CF(A \rightarrow B) = conf(A \rightarrow B)$
- Sonst  
 $CF(A \rightarrow B) = 0$

Das Sicherheitsmaß befolgt die Prinzipien für Regelbewertung. Wendet man Signifikanztest oder Sicherheitsmaß an, erhält man weniger (irrelevante, irreführende) Assoziationsregeln.

-  Jiawei Han and Micheline Kamber  
 Data Mining: Concepts and Techniques  
 Slides for Textbook - Chapter 6  
 Intelligent Database Systems Research Lab.  
 School of Computing Science.  
 Simon Fraser University, Canada.  
<http://www.cs.sfu.ca>

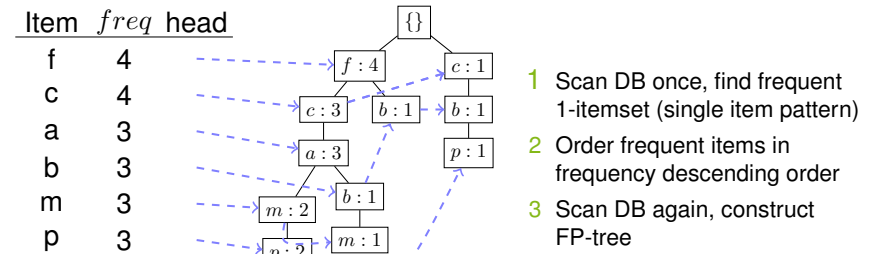
## Mining Frequent Patterns Without Candidate Generation

- Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure
  - highly condensed, but complete for frequent pattern mining
  - avoid costly database scans
- Develop an efficient, FP-tree-based frequent pattern mining method
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones
  - Avoid candidate generation: sub-database test only!

## Construct FP-tree from a Transaction DB

TID	Items bought	(ordered) frequent items
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}

support<sub>min</sub> = 0.5



## FP-Tree

- Ein *FP* Tree ist nach Häufigkeiten (von oben nach unten) geordnet.
- Ein *FP* Tree fasst Transaktionen als Wörter auf und stellt gemeinsame Präfixe verschiedener Wörter dar.
- Für jede Transaktion lege einen Pfad im *FP* Tree an:
  - Pfade mit gemeinsamem Präfix - Häufigkeit +1, Suffix darunter hängen.
  - Kein gemeinsamer Präfix vorhanden - neuen Zweig anlegen.
- Header Tabelle verweist auf das Vorkommen der items im Baum. Auch die Tabelle ist nach Häufigkeit geordnet.

## Benefits of the *FP*-tree Structure

- Completeness:
  - never breaks a long pattern of any transaction
  - preserves complete information for frequent pattern mining
- Compactness:
  - reduce irrelevant information - infrequent items are gone
  - frequency descending ordering: more frequent items are more likely to be shared
  - never be larger than the original database (if not count node-links and counts)
  - Example: For Connect-4 DB, compression ratio could be over 100

## Mining Frequent Patterns Using *FP*-tree

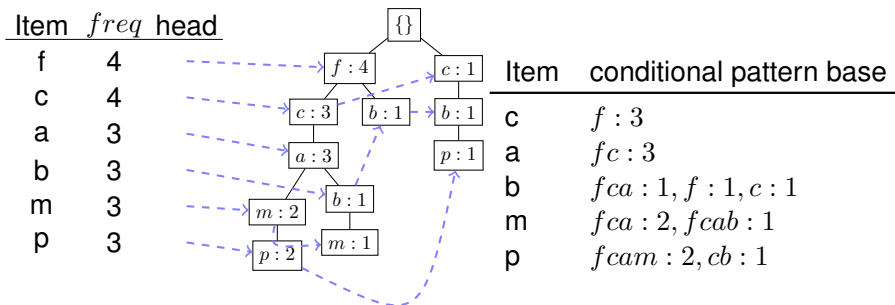
- General idea (divide-and-conquer)
  - Recursively grow frequent pattern path using the *FP*-tree
- Method
  - For each item, construct its *conditional pattern-base*, and then its *conditional FP-tree*
  - Repeat the process on each newly created conditional *FP-tree*
  - Until the resulting *FP-tree* is *empty*, or it contains *only one path* (single path will generate all the combinations of its sub-paths, each of which is a frequent pattern)

## Major Steps to Mine *FP*-tree

- 1 Construct conditional pattern base for each node in the *FP*-tree
- 2 Construct conditional *FP-tree* from each conditional pattern-base
- 3 Recursively mine conditional *FP-trees* and grow frequent patterns obtained so far
  - If the conditional *FP-tree* contains a single path, simply enumerate all the patterns

## Step 1: From *FP*-tree to Conditional Pattern Base

- Starting at the frequent header table in the *FP*-tree
- Traverse the *FP*-tree by following the link of each frequent item
- Accumulate all of transformed prefix paths of that item to form a conditional pattern base



## Vom *FP* Tree zur Cond. Pattern Base

- Die Header Tabelle von unten (selten) nach oben durchgehen. Die Verweise führen zu den Pfaden, in denen das item vorkommt.
  - Das item wird als Suffix betrachtet und alle Präfixe davon als Bedingungen für dies Suffix.
  - Die Häufigkeiten der Präfixe werden von unten nach oben propagiert.

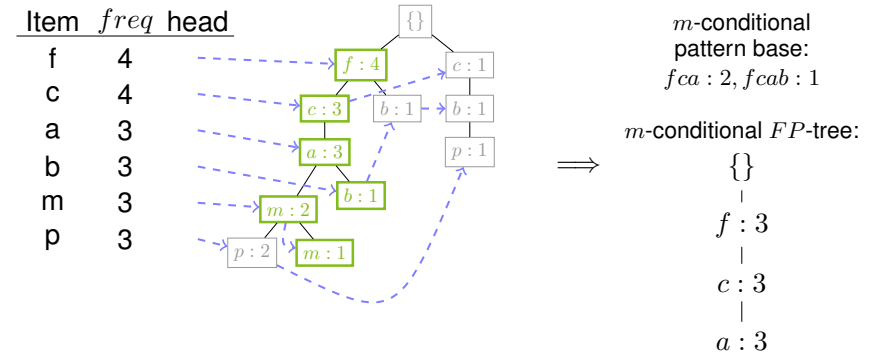


Properties of FP-tree for Conditional Pattern Base Construction

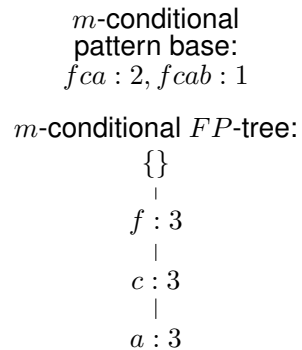
- Node-link property
  - For any frequent item  $a_i$ , all the possible frequent patterns that contain  $a_i$  can be obtained by following  $a_i$ 's node-links, starting from  $a_i$ 's head in the FP-tree header
- Prefix path property
  - To calculate the frequent patterns for a node  $a_i$  in a path  $P$ , only the prefix sub-path of  $a_i$  in  $P$  need to be accumulated, and its frequency count should carry the same count as node  $a_i$ .

Step 2: Construct Conditional FP-tree

- For each pattern-base
  - Accumulate the count for each item in the base
  - Construct the FP-tree for the frequent items of the pattern base



$m$ -conditional FP-tree



⇒

- All frequent patterns concerning  $m$
- $m$
  - $fm, cm, am$
  - $fcm, fam, cam$
  - $fcam$

Mining Frequent Patterns by Creating Conditional Pattern-Bases

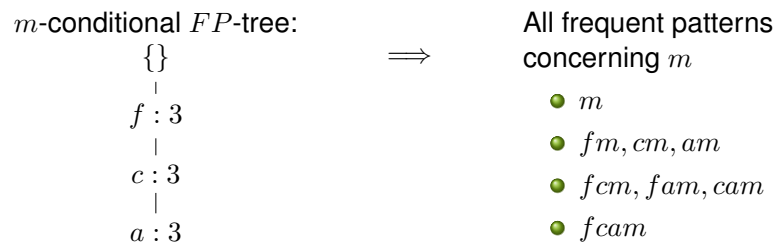
Item	Conditional pattern-base	conditional FP-tree
p	$\{(fcam : 2), (cb : 1)\}$	$\{(c : 3)\}p$
m	$\{(fca : 2), (fcab : 1)\}$	$\{(f : 3, c : 3, a : 3)\}m$
b	$\{(fca : 1), (f : 1), (c : 1)\}$	Empty
a	$\{(fc : 3)\}$	$\{(f : 3, c : 3)\}a$
c	$\{(f : 3)\}$	$\{(f : 3)\}c$
f	Empty	Empty

Cond. Pattern Base - Cond. FP Tree

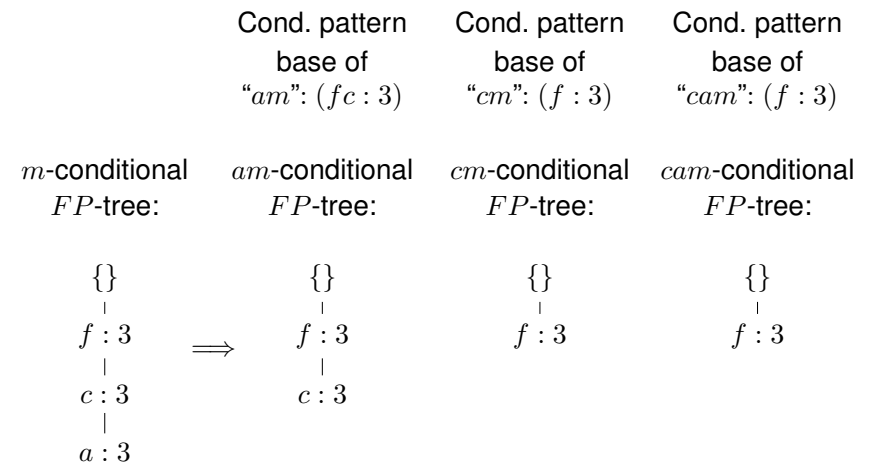
- Präfixpfade eines Suffixes bilden die bedingte Basis.
- Diejenigen Präfixpfade, die häufiger als  $support_{min}$  sind, bilden den bedingten FP Tree.
- Falls mehrere dieser Präfixpfade zu einem Suffix gleich sind (vom Anfang bis zu einer bestimmten Stelle), wird daraus ein Pfad bis zu dieser Stelle und die ursprünglichen Häufigkeiten werden addiert.
- Ansonsten gibt es mehrere Pfade im bedingten Baum.

Single FP-tree Path Generation

- Suppose an FP-tree  $T$  has a single path  $P$
- The complete set of frequent pattern of  $T$  can be generated by enumeration of all the combinations of the sub-paths of  $P$



Step 3: Recursively mine the conditional FP-tree



Cond. FP Tree - frequent sets

- Alle Teilmuster im bedingten FP Baum, der nur ein Zweig ist, und des Suffixes bilden die Menge häufiger Muster.
- Die gesuchte Menge der häufigen Mengen ist die Gesamtheit aller häufiger Muster aus allen bedingten FP Bäumen.

## Principles of Frequent Pattern Growth

- Pattern growth property
  - Let  $\alpha$  be a frequent itemset in  $DB$ ,  $B$  be  $\alpha$ 's conditional pattern base, and  $\beta$  be an itemset in  $B$ . Then  $\alpha \cup \beta$  is a frequent itemset in  $DB$  iff  $\beta$  is frequent in  $B$ .
- "abcdef" is a frequent pattern, if and only if
  - "abcde" is a frequent pattern, and
  - "f" is frequent in the set of transactions containing "abcde"

## $insert\_tree([p|P], T)$

- Wenn  $T$  ein Kind  $N$  hat mit  $N.item\_name = p.item\_name$  dann erhöhe Häufigkeit von  $N + 1$ .
- Sonst bilde neuen Knoten  $N$  mit Häufigkeit = 1 direkt unter  $T$  und füge Knotenverweise zu den Knoten mit dem selben  $item.name$  ein.
- Solange  $P$  nicht  $\{\}$  ist,  $insert\_tree(P, N)$ .

## Algorithmus $FP\_growth$

Input:

- $D$  eine Transaktionsdatenbank
  - $support_{min}$  ein Schwellwert der Häufigkeit
- 1 Scan von  $D$ , Erstellen der Menge  $F$  häufiger items und ihrer Häufigkeiten, Ordnen von  $F$  in absteigender Häufigkeit.
  - 2 Wurzel des  $FP$  Trees ist Null. Für jede Transaktion  $Trans$  in  $D$ :  
 nach Häufigkeit gemäß  $F$  geordnete items in  $Trans$  werden zur Liste  $[p|P]$ , wobei  $p$  das erste item und  $P$  die restlichen sind.  $insert\_tree([p|P], T)$
  - 3  $FP\_growth(FP\_tree, null)$

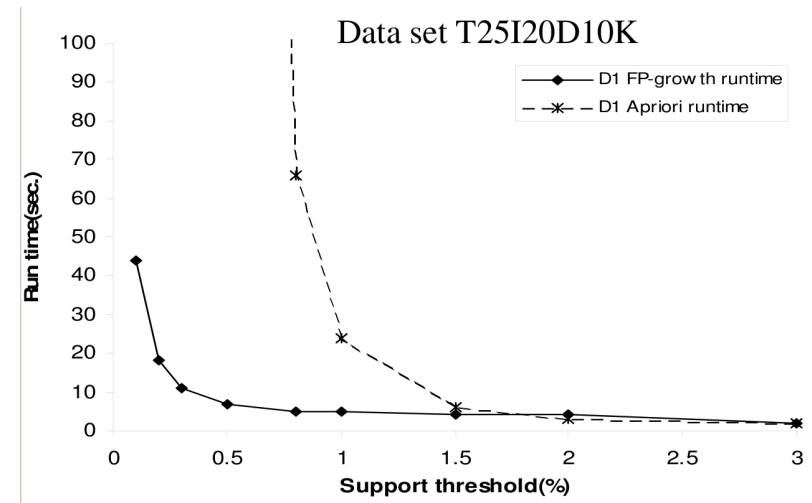
## $fp\_growth(Tree, \alpha)$

- Wenn Tree ein einziger Pfad  $P$  ist,
  - dann generiere für jede Kombination  $\beta$  von Knoten in  $P$  Muster  $\beta \cup \alpha$  mit  $support = support_{min}$  eines items in  $\beta$ .
- Sonst für jedes  $a_i$  in header von Tree
  - generiere Muster  $\beta = a_i \cup \alpha$  mit  $s = a_i.s$
  - konstruiere  $\beta$  cond. base und daraus  $\beta$  cond.  $FP$  tree  $Tree_\beta$
  - Wenn  $Tree_\beta$  nicht  $\{\}$ , dann  $fp\_growth(Tree_\beta, \beta)$

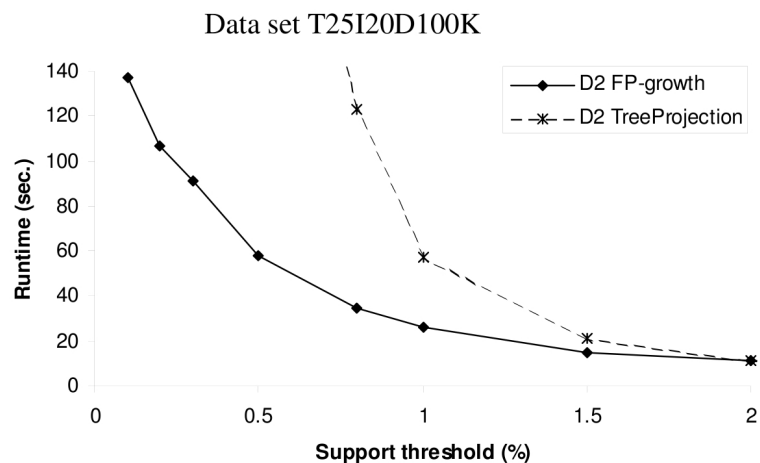
## Why Is Frequent Pattern Growth Fast?

- Our performance study shows
  - *FP*-growth is an order of magnitude faster than Apriori, and is also faster than tree-projection
- Reasoning
  - No candidate generation, no candidate test
  - Use compact data structure
  - Eliminate repeated database scan
  - Basic operation is counting and *FP*-tree building

## *FP*-growth vs. Apriori: Scalability With the Support Threshold



## *FP*-growth vs. Apriori: Scalability With the Support Threshold



## Was wissen wir jetzt?

- *FP*-growth als Alternative zu Apriori
  - Schneller, weil keine Kandidaten generiert werden
  - Kompaktes Speichern
  - Basisoperation ist einfach Zählen.
- Der *FP*-Baum gibt Präfixbäume für ein Suffix an.
- Die Ordnungsrelation ist die Häufigkeit der items.
  - Der Baum wird vom häufigsten zum seltensten gebaut.
  - Die bedingte Basis wird vom seltensten Suffix zum häufigsten erstellt.