- Data Mining
  : KDD (Knowledge Discovery from Data)
  : Data Intelligence
  : Data Management

- Association Metrics
  relative support
  Transaction에서 몇%?
  ① support(s)
  Transaction
  $s = \sigma(\text{itemset}) / |T|$
  ② confidence(c) ← itemset x의 지지율(비율)
  $c = \sigma(X \text{ and } Y) / \sigma(X)$
  support count
  ⇓

finding frequent itemset이 중요하다
  support ≥ minsup (threshold)

Association Rule
Apriori, FP-growth
① Frequent itemset generation
② Rule generation
  (X→Y 같은거!)
  * same frequent itemset에서 발생한 rule (binary partition)
    은 support는 같고
    confidence 다르다

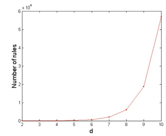
Association Rule Mining – ① Frequent Itemset Mining

a) Brute-force approach

: 각 Itemset의 candidate itemset의 support 다 계산하는 것 (오랑)

Given d unique items:

- Total number of itemsets = 2^d
- Total number of possible association rules:



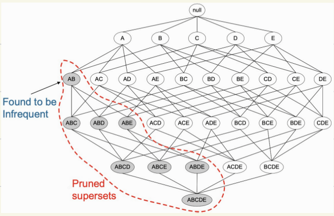
b) Apriori approach

- anti-monotone property of support

  $\forall X, Y : (X \subseteq Y) \Rightarrow S(X) \geq S(Y)$

  ⊜ subset의 support가 < minsup이면

  superset의 support도 < minsup



- Algorithm

k-1 size의 frequent itemset에서

k size의 candidate itemset 생성

```
1)  L_1 = {large 1-itemsets};
2)  for ( k = 2; L_{k-1} ≠ ∅; k++ ) do begin
3)     C_k = apriori-gen(L_{k-1});  // New candidates
4)     forall transactions t ∈ D do begin
5)        C_t = subset(C_k, t);  // Candidates contained in t
6)        forall candidates c ∈ C_t do
7)           c.count++;
8)     end
9)     L_k = {c ∈ C_k | c.count ≥ minsup}
10) end
11) Answer = ∪_k L_k;
```

Apriori-gen function

- Self-join

insert into C_k
select $p.\text{item}_1, p.\text{item}_2, \ldots, p.\text{item}_{k-1}, q.\text{item}_{k-1}$
from $L_{k-1}\ p, L_{k-1}\ q$
where $p.\text{item}_1 = q.\text{item}_1, \ldots, p.\text{item}_{k-2} = q.\text{item}_{k-2}, p.\text{item}_{k-1} < q.\text{item}_{k-1}$;

- Pruning

forall itemsets $c \in C_k$ **do**
  forall $(k-1)$-subsets s of c **do**
    if $(s \notin L_{k-1})$ **then**
      delete c from C_k;

- generate C_{k+1} from L_k (self-join)
  - (k-1) Item들을 공유한 두 row에 대해서
  - 각 row는 increasing order
  - later가 더 크면 join

Item 1	Item 2	Item 3
1	2	3
1	2	5
1	4	5

동일 (rows 1–2), 다음

1	2	3	5

join

- only frequent candidate itemsets! (pruning)
  - anti-monotone 이용

    subset이 infrequent하면 superset 다 prune

↳ Apriori needs multiple DB scan ⊕ generate candidate / test support

c) FP Growth

Frequent-Pattern mining
without candidate generation

· Heuristic Property

P(frequent Itemset)

S (set of transaction with P)

X (item)

if X = frequent itemset in S

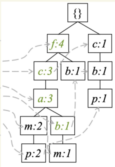
then {x} ∪ P must be frequent itemset

· FP-tree에서 Frequent pattern set?

① Conditional pattern bases

Pattern-base | α

: item α의 prefix sets



Pattern-base | {m}

⇒ <f,c,a> : support = 2

⇒ <f,c,a,b> : support = 1

② Conditional FP-trees

FP-tree | α

: Pattern-base | α 에서 minsup 만족하는 애들만 @

③ Find Frequent Pattern

: 각 조합의 { minsup 넘는애들!)

Item	Conditional Pattern Base	Conditional FP-tree	Frequent Patterns Generated
I4	{I2,I1,I3:1},{I2,I3:1}	{I2:2, I3:2}	{I2,I4:2},{I3,I4:2},{I2,I3,I4:2}
I3	{I2,I1:3},{I2:1}	{I2:4, I1:3}	{I2,I3:4}, {I1:I3:3}, {I2,I1,I3:3}
I1	{I2:4}	{I2:4}	{I2,I1:4}

* FP-tree

: frequent pattern data 정보담고있음

- How to make FP-tree

① Make F-list

· DB scan해서 1-size item을 frequent순으로 나열
(desc order)

F-list = f-c-a-b-m-p

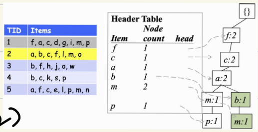
여기서 count된거

DB scan(1)

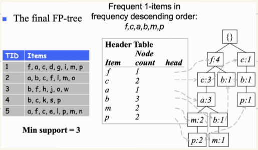
② order items in itemset (F-list 순서로)

TID	Items	Ordered items
1	f, a, c, d, g, i, m, p	f, c, a, m, p
2	a, b, c, f, l, m, o	f, c, a, b, m
3	b, f, h, j, o, w	f, b
4	b, c, k, s, p	c, b, p
5	a, f, c, e, l, p, m, n	f, c, a, m, p

③ row-wise 하면서 F-list 순서대로 Tree 만들기



DB scan(2)



Implication in Property.

- Process of mining frequent patterns can be viewed as first mining frequent 1-itemsets and then progressively growing each such itemset by mining its conditional pattern base, which can in turn be done similarly
- We successfully transform a frequent k-itemset mining problem into a sequence of k frequent 1-itemset mining problems via a set of conditional pattern bases

Why Is FP-Growth the Winner?

- Divide-and-conquer
  - Decomposing both the mining task and database according to the frequent patterns obtained so far
  - Leading to focused search of smaller databases
- Other factors
  - **No candidate generation, no candidate test**
  - Compressed database: FP-tree structure
  - No repeated scan of the entire database
  - Cheap operations: counting local frequent items and building sub FP-trees, but no pattern search and matching

번외

* reduce # of Comparison

- Support Count = Transaction Scan

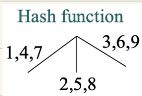
줄이기

Store candidates in HashTable

- key : Candidate Itemset
- Value : Support Count

① generate hash tree

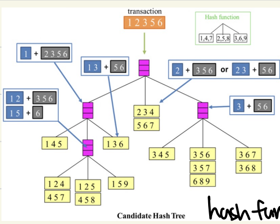
- hash function

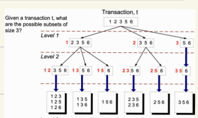


: hash-tree에서 해당 key의 location

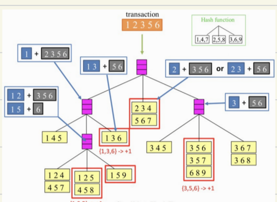
- Max leaf size

② update hash table





hash function 기반으로 leaf 찾기



일치하는 candidate만 update

* reduce complexity

- minsup 선택
  : 높이면 frequent itemset 개수 줄일 수 있음
- Dimensionality (# of items)
  : 얘 늘어나면 computation / I/O 늘어날 수 있음
- size of DB
  : Apriori는 multiple pass라서, run time of algorithm may increase with # of transactions

Association Rule Mining ② - Rule Generation

· Given frequent itemset 에서
Candidate rule 만들기
(공집합, 전체집합 빼고! 2^d-2 개)

If {A,B,C,D} is a frequent itemset, candidate rules:

ABC→D,	ABD→C,	ACD→B,	BCD→A,
A→BCD,	B→ACD,	C→ABD,	D→ABC
AB→CD,	AC→BD,	AD→BC,	BC→AD,
BD→AC,	CD→AB,		

· Who to remove?
Confidence가 minconfidence보다 작은애들을 ⓟ
* Confidence는 anti-monotone 성질이니,

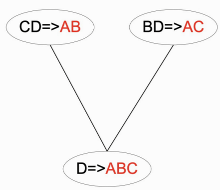
e.g., L = {A,B,C,D}: $c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$

$$\frac{\sigma(ABCD)}{\sigma(ABC)} \geq \frac{\sigma(ABCD)}{\sigma(AB)} \geq \frac{\sigma(ABCD)}{\sigma(A)}$$

Confidence is anti-monotone w.r.t. the number of items on the RHS of the rule

이경우는 된다 ⓟ
그래서 superset이 infrequent = 나머지다 infrequent
subset이 frequent = 나머지다 frequent

- A candidate rule is generated by merging two rules that share the same prefix
  in the rule consequent
- Example:
  - join(CD→AB, BD→AC) would produce the candidate rule D→ABC
  - **D→ABC is pruned if its subset BD→AC does not have high confidence**



superset이 infrequent한거 ⓟ

anti-monotone