Apriori Variants to overcome the repeated scans limitation

- DHP (Park ’95): Dynamic Hashing and Pruning
- Candidate large 2-itemsets are huge.
  - DHP: trim them using hashing
- Transaction database is huge that one scan per iteration is costly
  - DHP: prune both number of transactions and number of items in each transaction after each iteration

Hash Table Construction

- Consider two items sets, all itesms are numbered as i1, i2, …in. For any any pair (x, y), has according to
  - Hash function bucket #=

    h({x y}) = ((order of x)*10+(order of y)) % 7
- Example:
  - Items = A, B, C, D, E, Order = 1, 2, 3 4, 5,
  - H({C, E})= (3*10 + 5)% 7 = 0
  - Thus, {C, E} belong to bucket 0.

Example

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E

Generation of C1 & L1(1st iteration)

Itemset	Sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

C1

Itemset	Sup
{A}	2
{B}	3
{C}	3
{E}	3

L1

Hash Table Construction

- Find all 2-itemset of each transaction

TID	2-itemset
100	{A C} {A D} {C D}
200	{B C} {B E} {C E}
300	{A B} {A C} {A E} {B C} {B E} {C E}
400	{B E}

Hash Table Construction

- Hash function

  h({x y}) = ((order of x)*10+(order of y)) % 7

- Hash table

	{C E}	{A E}	{B C}		{B E}	{A B}	{A C}
	{C E}		{B C}		{B E}		{C D}
	{A D}				{B E}		{A C}
	3	1	2	0	3	1	3
Bucket	0	1	2	3	4	5	6

C2 Generation (2nd iteration)

L1*L1	# in the bucket
{A B}	1
{A C}	3
{A E}	1
{B C}	2
{B E}	3
{C E}	3

Resulted C2
{A C}
{B C}
{B E}
{C E}

C2 of Apriori
{A B}
{A C}
{A E}
{B C}
{B E}
{C E}

H_2

Create hash table H_2 using hash function $h(x, y) = ((order\ of\ x) \times 10 + (order\ of\ y))\ mod\ 7$

bucket address	0	1	2	3	4	5	6
bucket count	2	2	4	2	2	4	4
bucket contents	{I1, I4}	{I1, I5}	{I2, I3}	{I2, I4}	{I2, I5}	{I1, I2}	{I1, I3}
	{I3, I5}	{I1, I5}	{I2, I3}	{I2, I4}	{I2, I5}	{I1, I2}	{I1, I3}
			{I2, I3}			{I1, I2}	{I1, I3}
			{I2, I3}			{I1, I2}	{I1, I3}

Apriori Transaction Reduction

- Suppose the minimum support count min_sup=3.
- The algorithm is as follows:
- Step 1: First we have to convert database into the desired database that is with SOT column.
- Step 2: In the first iteration of the algorithm, each item is a member of the set of candidate 1-itemsets, C1. The algorithm simply scans all of the transactions in order to count the number of occurrences of each item.
- Step 3: This algorithm will then generate number of items in each transaction. We called this Size_Of _Transaction (SOT). Step 4: Because of min_sup=3, the set of frequent 1-itemset, L1 can be determined. It consists of the candidate 1-itemset, C1, satisfying minimum support. Step 5: Since the support count of I5,I6,I7 are less than 3, they won't appear in L1. Delete these data from D. In addition, when L1 is generated, now, the value of k is 2, delete those records of transaction having SOT=1 in D. And there won't exist any elements of C2 in the records we find there is only one

- data in the T9. We delete the data and obtain transaction database D1.
- Step 6: To discover the set of frequent 2-itemsets, L2, the algorithm uses the join L1 ∞L1 to generate a candidate set of 2- itemsets, C2. Step 7: The transactions in D1 are scanned and the support count and SOT of each candidate itemset in C2 is accumulated.
- Step 8: The set of frequent 2-itemsets, L2, is then determined, consisting of those candidate 2-itemsets in C2 having minimum support.
- Step 9: After L2 is generated, we can find the transaction record of T1,T2,T4,T6,T10 are only two in D1.Now,the value of k is 2,delete those records of transaction having SOT=2.And there won't exist any elements of C3 in the records. Therefore, these records can be deleted and we obtain transaction database D2.

- Step 10: To discover the set of frequent 3-itemsets, L3, the algorithm uses the join L2∞ L2 to generate a candidate set of 3- itemsets C3, where C3= L2 ∞L2= I1， I2， I3},{ I2， I3， I4}}. There are a number of elements in C3. According to the property of Apriori algorithm, C3 needs to prune. Because {I1 ， I2} not belongs to L2, we remove it from C3. Because the 2-subsets {I2， I3}, I2， I4} and {I3， I4} all belong to L2, they should remain in C3.
- Step 11: The transactions in D2 are scanned and the support count of each candidate itemset in C3 is accumulated. Use C3 to generate L3.
- Step 12: L3 has only one 3-itemsets so that C4 =□. The algorithm will stop and give out all the frequent itemsets.
- Step 13: Algorithm will be generated for Ck until Ck+1 becomes empty

Tid	Items
T1	I1,I3,I7
T2	I2,I3,I7
T3	I1,I2,I3
T4	I2,I3
T5	I2,I3,I4,I5
T6	I2,I3
T7	I1,I2,I3,I4,I6
T8	I2,I3,I4,I6
T9	I1
T10	I1,I3

Tid	Items	SOT
T1	I1,I3,I7	3
T2	I2,I3,I7	3
T3	I1,I2,I3	3
T4	I2,I3	2
T5	I2,I3,I4,I5	4
T6	I2,I3	2
T7	I1,I2,I3,I4,I6	5
T8	I2,I3,I4,I6	4
T9	I1	1
T10	I1,I3	2

C1

Itemset	Sup_count
I1	5
I2	7
I3	9
I4	3
I5	1
I6	2
I7	2

L1

Itemset	Sup_count
I1	5
I2	7
I3	9
I4	3

D1

I6	2
I7	2

D1

Tid	Items	SOT
T1	I1,I3	2
T2	I2,I3	2
T3	I1,I2,I3	3
T4	I2,I3	2
T5	I2,I3,I4	3
T6	I2,I3	2
T7	I1,I2,I3,I4	4
T8	I2,I3,I4,~~I6~~	3
T10	I1,I3	2

C2

Itemset	Sup_count
I1,I2	2
I1,I3	4
I1,I4	1
I2,I3	7
I2,I4	3
I3,I4	3

L2

Itemset	Sup_count
I1,I3	4
I2,I3	7
I2,I4	3
I3,I4	3

D2

Tid	Items	SOT
T3	I1,I2,I3	3
T5	I2,I3,I4	3
T7	I1,I2,I3,I4	4
T8	I2,I3,I4,~~I6~~	3

Itemset	Sup_count
I2,I3,I4	3

C3

L3

Itemset	Sup_count
I2,I3,I4	3

Vertical Transaction Approach to FIM

itemset	*TID_set*
I1	{T100, T400, T500, T700, T800, T900}
I2	{T100, T200, T300, T400, T600, T800, T900}
I3	{T300, T500, T600, T700, T800, T900}
I4	{T200, T400}
I5	{T100, T800}

2-Itemsets in Vertical Data Format

itemset	*TID_set*
{I1, I2}	{T100, T400, T800, T900}
{I1, I3}	{T500, T700, T800, T900}
{I1, I4}	{T400}
{I1, I5}	{T100, T800}
{I2, I3}	{T300, T600, T800, T900}
{I2, I4}	{T200, T400}
{I2, I5}	{T100, T800}
{I3, I5}	{T800}

3-Itemsets in Vertical Data Format

itemset	*TID_set*
{I1, I2, I3}	{T800, T900}
{I1, I2, I5}	{T100, T800}

Partitioning Variant to Apriori

Sample Database

A1	A2	A3	A4	A5	A6	A7	A8	A9
1	0	0	0	1	1	0	1	0
0	1	0	1	0	0	0	1	0
0	0	0	1	1	0	1	0	0
0	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0
0	1	1	1	0	0	0	0	0
0	1	0	0	0	1	1	0	1
0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	0
0	0	1	0	1	0	1	0	0
0	0	1	0	1	0	1	0	0
0	0	0	0	1	1	0	1	0
0	1	0	1	0	1	1	0	0
1	0	1	0	1	0	1	0	0
0	1	1	0	0	0	0	0	1

```
P = partition_database(T); n = Number of partitions
// Phase I
    for i = 1 to n do begin
        read_in_partition(T_i in P)
        L^i = generate all frequent itemsets of T_i using a priori method in main memory.
    end

// Merge Phase
    for (k = 2; L_k^i ≠ ∅, i = 1, 2, ..., n; k++) do begin
        C_k^G = ∪_{i=1}^{n} L_i^k
    end

// Phase II
    for i = 1 to n do begin
        read_in_partition(T_i in P)
        for all candidates c ∈ C^G compute s(c)_{T_i}
    end
    L^G = {c ∈ C^G | s(c)_{T_i} ≥ σ}
    Answer = L^G
```

3 Partitions with 5 transactions in each and non overlapping from 1 to 5, 6 to 10 and 11 to 15. Let local support be equal to global support of 20%.

The local frequent sets of the T_1 partition are the itemsets X, such that $s(X)_{T_1} \geq \sigma_1$.

L^1 := { {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {1, 5}, {1, 6}, {1, 8}, {2, 3}, {2, 4}, {2, 8}, {4, 5}, {4, 7}, {4, 8}, {5, 6}, {5, 8}, {5, 7}, {6, 7}, {6, 8}, {1,6, 8}, {1,5,6}, {1,5,8}, {2,4,8}, {4,5,7}, {5,6,8}, {5,6,7}, {1,5,6,8} }

Similarly,

L^2 := { {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {2,3}, {2,4}, {2,6}, {2,7}, {2,9}, {3,4}, {3, 5}, {3, 7}, {5, 7}, {6,7}, {6,9}, {7,9}, {2,3,4}, {2,6,7}, {2,6,9}, {2,7,9}, {3, 5, 7}, {2,6,7,9} }

L^3 := { {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {1,3}, {1,5}, {1,7}, {2,3}, {2,4}, {2,6}, {2,7}, {2,9}, {3,5}, {3,7},{3,9},{4,6}, {4,7}, {5,6}, {5,7}, {5,8}, {6,7}, {6,8}, {1,3,5}, {1,3,7}, {1,5,7}, {2,3,9}, {2,4,6}, {2,4,7}, {3,5,7}, {4,6,7}, {5,6,8}, {1, 3, 5, 7}, {2, 4, 6, 7} }

In Phase II, we have the candidate set as

$C := L^1 \cup L^2 \cup L^3$

C := { {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {1,3}, {1, 5}, {1, 6}, {1,7}, {1, 8}, {2,3}, {2, 4}, {2,6}, {2,7}, {2, 8}, {2,9},{3,4}, {3,5}, {3,7}, {3,9}, {4, 5}, {4,6}, {4,7}, {4, 8}, {5, 6},{5,7}, {5, 8}, {5, 7}, {6, 7}, {6, 8}, {6,9}, {7,9},{1,3,5}, {1,3,7}, {1,5,6}, {1,5,7}, {1,5,8}, {1,6,8}, {2,3,4}, {2,3,9}, {2,4,6}, {2,4,7}, {2,4,8}, {2,6,7}, {2,6,9}, {2,7,9}, {3,5,7}, {4,5,7}, {4,6,7}, {5,6,8}, {5,6,7}, {1,5,6,8}, {2,6,7,9}, {1, 3, 5, 7}, {2, 4, 6, 7} }

Read the database once to compute the global support of the sets in C and get the final set of frequent sets.