DATA MINING
&
DATA WAREHOUSING

Module III

- Association Rule Mining
  - ● What is AR
  - ● Methods to discover AR
  - ● Apriori algo
  - ● Partition algo
  - ● Pincer seaarch algo
  - ● FPtree growth algo
  - ● Incremental algo
  - ● Border algo
  - ● Generalized ARs

Incremental Algorithm

Incremental Algorithm

- Earlier FP algorithms , assume that DB does not change .
- But transaction DB is not static .
- When DB is updated the existing ARs may become invalid
- Before deriving the frequent itemsets the DB become updated thereby derived itemsets are no more valid

- Instead of redoing the FP derivation the earlier computations of frequent itemsets must also be used
  - The incremental algorithm aims to achieve this
  - So the earlier computations of frequent itemsets are used in this algorithm

The following notations are used in incremental algorithms

- T_{old} – existing DB
- L_{old} – already computed frequent itemsets
- T_{new} – new set of transactions added to the database
- $T_{whole} = T_{old} \cup T_{new}$
- We have to find L_{whole}

Let L_{new} is set of all frequent itemsets of T_{new} & L_{new} is initially unknown. & Observaions used are

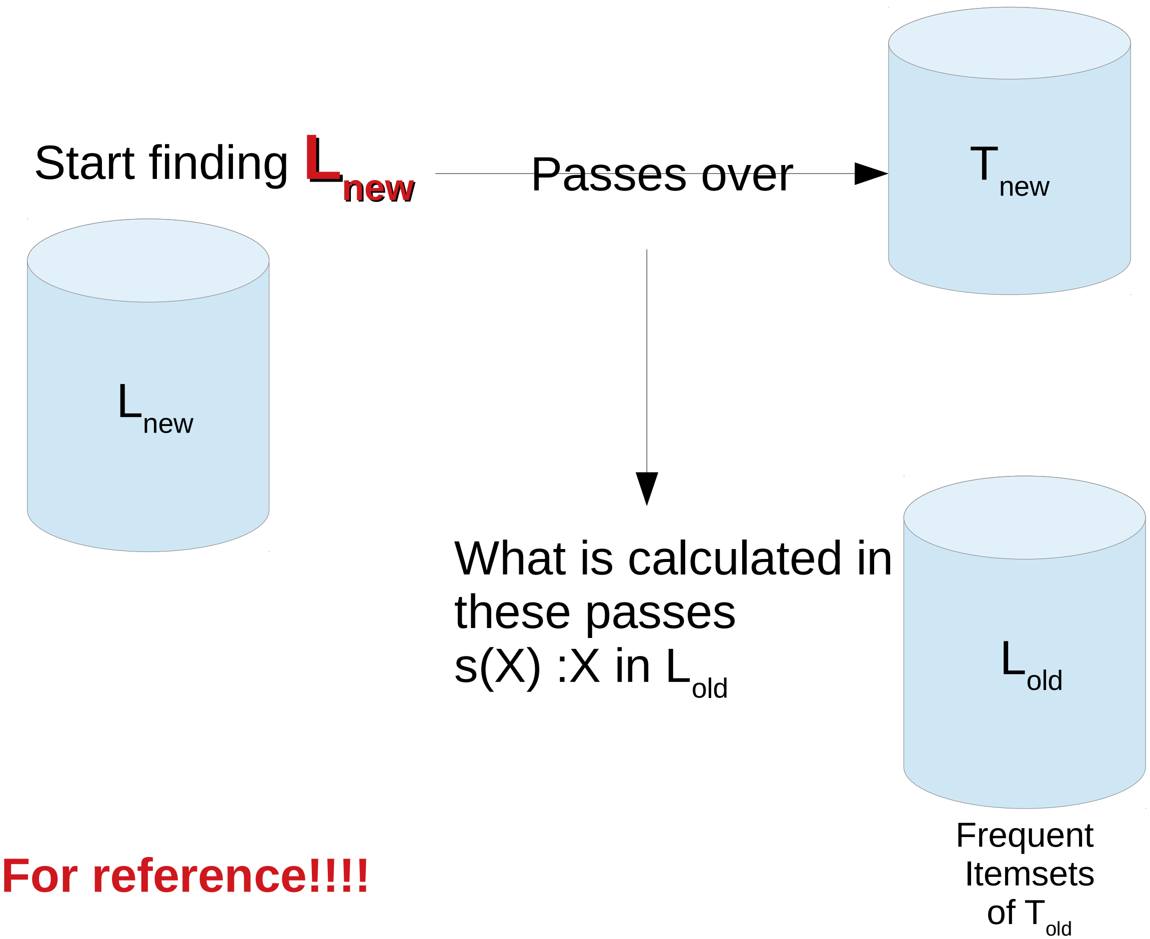
1. An itemset is in L_{whole} if it is an element of both L_{new} and L_{old}
2. Any itemset which is neither in L_{new} or L_{old} cannot be in L_{whole}

we start finding L_{new} and make some passes over T_{new} (we use condition 1 to find L_{whole})

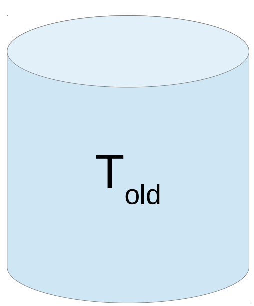
These passes canbe used to find { $s(X)$: X in L_{old} } in T_{new}

i.e., $s(X)_{Tnew}$ for X in L_{old}

Based on condition 1



- But unless we read $T_{old,}$
- We cannot find those Frequent Itemsets, if any,
  - which are frequent T_{new}
  - but infrequent in T_{old}



For reference!!!!

- **Using Incremental Method**
  - **Count s(X) wrt T_{new} for all X in L_{old}**
  - **This may give partial charaterization of L_{whole}**
  - **This is achieved by just one pass over T_{new} only**

- **The above process takes following cases**

a. The itemsets of L_{old} that are frequent in T_{new} (hence, in T_{whole}) can be determined.

b. The itemsets that belong to L_{old} but are not in L_{new} are automatically eliminated.

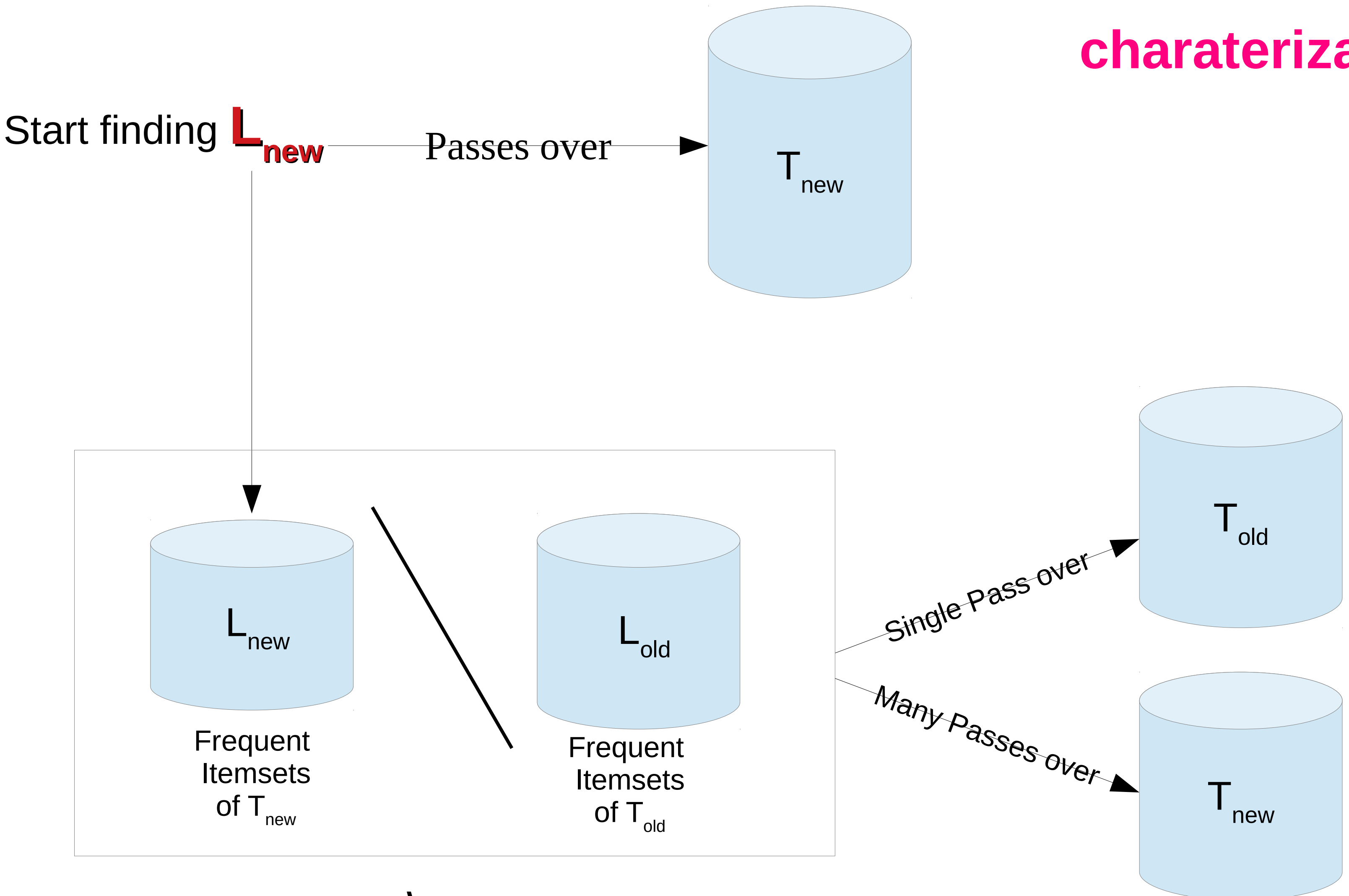
c. The itemsets that are neither in L_{new} nor in L_{old} are not considered.

- To get **complete characterisation** is
  - first compute L_{new} by making one pass over T_{new}
  - Next compute support of itemsets in $L_{new} \setminus L_{old}$ by making one pass over T_{old}
    - i.e., it requires one pass over T_{old} and many passes over T_{new}

Based on case 1

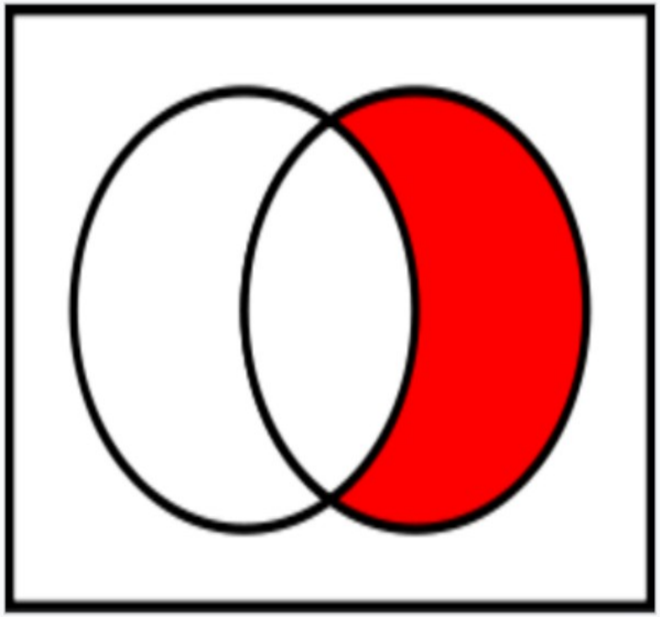
For reference!!!!

- To get **complete charaterization**



$s(X) : X \text{ in } L_{new} \setminus L_{old}$

For reference!!!!



The **relative complement** of A (left circle) in B (right circle): $B \cap A^{\complement} = B \setminus A$

The relative complement of A in B is denoted $B \setminus A$ according to the ISO 31-11 standard. It is sometimes written $B - A$, but this notation is ambiguous, as in some contexts it can be interpreted as the set of all elements $b - a$, where b is taken from B and a from A.

Formally:

$$B \setminus A = \{x \in B \mid x \notin A\}.$$

Examples [edit]

- $\{1, 2, 3\} \setminus \{2, 3, 4\} = \{1\}$.
- $\{2, 3, 4\} \setminus \{1, 2, 3\} = \{4\}$.
- If $\mathbb{R}$ is the set of real numbers and $\mathbb{Q}$ is the set of rational numbers, then $\mathbb{R} \setminus \mathbb{Q}$ is the set of irrational numbers.

- **The disadvantage here of complete charaterization process is**
  - **FIs may not be genetated and pass is wasted**

- If we **know in advance whether DB pass is required or not** then it will help reducing the unnecesary passes
  - i.e., search for FI of T_{whole} which is not in L_{old}
    - if no such FIs then no DB pass is required
- For which a new concept called '**promoted border set**' is used
- If there is a 'promoted border set' we need to scan T_{old}

Promoted Border Set

- An itemset that was border set before update and becomes a frequent set after update is called **'promoted border set '**
  - If it exists we have to read T_{old} again
  - $L_{whole} \cap (L_{new} \setminus L_{old}) \; != \emptyset$ iff there exixts promoted border itemsets

- **Simple modification to the apriori algo gives border sets ....**

Modified *A Priori* Algorithm to Generate Border Sets and Frequent Sets

Initialize: $k := 1$, C_1 = all the 1-itemsets;
read the database to count the support of C_1 to determine L_1.
$L_1 :=$ {frequent 1-itemsets};
$k := 2$; *// k represents the pass number//*
while $(L_{k-1} \neq \varnothing)$ ***do***
begin
    $C_k :=$ ***gen_candidate_item sets*** *with the given* L_{k-1}
    prune C_k
    for all transactions $t \in T$ ***do***
    increment the count of all candidates in C_k that are contained in t;
    $L_k := \{c \in C_k \mid s(c)_T \geq \sigma\}$;
    $B_k := \{c \in C_k \mid s(c)_T < \sigma\}$;
    $k := k + 1$;
end
    $L := \cup_k L_k$;
    $B := \cup_k B_k$;

Border Algorithm

- This algorithm is based on a new incremental method for generating the frequent sets, which are the basis for the association rules.
- The border algo maintains support count for all frequent sets & border sets
  - if the border set is promoted then DB is passed/scanned, if not no scanning of DB

- **Notations used are**
  - **F – frequent itemsets,**
  - **B- promoted border stes ,**

The Borders algorithm works by constantly maintaining the count information for all frequent sets and all border sets in the current relation.

1. L_{old} and B_{old} and their support count values are known
2. count Support value for all items in $L_{old} \cup B_{old}$ in T_{new}
    - this requires 1 pass over the T_{new} , the algo collects 2 categories of info F & B

F : contains itemsets of L_{old} which becomes frequent in T_{whole}

B : promoted border sets

- **if no promoted border set**
  - **- F contains all frequent set of T_{whole}**
- **if there is promoted border set**
  - **– generate candidate sets which are supersets of promoted border sets**
    - **- make one pass over the T_{new} , one pass over the T_{whole}**

Border Alg

- **Read T_{new} and calculate s(X) for all X in L_{old} U B_{old}**
- **Find**
  - **F = {X| X ∈ L_{old} and s(X)T_{whole}>=min_sup}**
  - **B = {X| X ∈ B_{old} and s(X)T_{whole}>=min_sup}**
- **Candidate generation**
  - **Generate all the candidate sets C_k from B_{k-1} and F_{k-1}**
  - **Prune all the candidate sets C_k**
  - **C=UC_k**
- **Read T_{whole} and calculate support count value of all the candidates in C_k**
- **new-Frequent-Items= {X| X ∈ C and s(X)T_{whole}>=min_sup}**
- **L_{whole}=F U new-Frequent-Items**
- **B_{whole}={B_{old}\B} U {X ∈ C and s(X)T_{whole} < min_sup and all its subsets are in L_{whole}}**

Border Algorithm

read T_{new} and increment the support count of X for all $X \in L_{old} \cup B_{old}$

$F := \{X \mid X \in L_{old}$ and $s(X)_{T_{whole}} \geq \sigma\}$

$B := \{X \mid X \in B_{old}$ and $s(X)_{T_{whole}} \geq \sigma\}$

Let m be the size of the largest element in B.

Candidate-generation

for all itemsets $l_1 \in B_{k-1} \cup C_{k-1}$ ***do begin***

for all itemsets $l_2 \in B_{k-1} \cup F_{k-1} \cup C_{k-1}$ ***do begin***

    if $l_1[1] = l_2[1] \wedge l_1[2] = l_2[2] \wedge \ldots \wedge l_1[k-1] < l_2[k-1]$ ***then***

    $c = l_1[1], l_1[2] \ldots l_1[k-1], l_2[k-1]$

    $C_k = C_k \cup \{c\}$

end do

end do

Prune C_k for all k:

all subsets of $k-1$ size should be present in $B_{k-1} \cup F_{k-1} \cup C_{k-1}$

$k := k+1$

Candidate $C := \cup C_k$

read T_{whole} and count the support values of each itemset in C.

new_frequent_sets $:= \{X \mid X \in C$ and $s(X)_{T_{whole}} \geq \sigma\}$

$L_{whole} := F \cup$ new_frequent_sets

$B_{whole} := (B_{old} \setminus B) \cup \{X \in C$ and $s(X)_{T_{whole}} < \sigma$ and all its subsets are in $L_{whole}\}$