The Incremental Association Rule Problem

Presented by…

Susan Imberman Ph.D.

Imberman@mail.csi.cuny.edu

Incremental Association Rules

- What are our options when databases are increasing in size?
  - Rerun Apriori on the "new" database
  - Costly and inefficient. Most of the cost in processing large databases is in scanning the data
- Incremental Association Rule Algorithms
  - Use information gained from the previous increment to reduce database scans

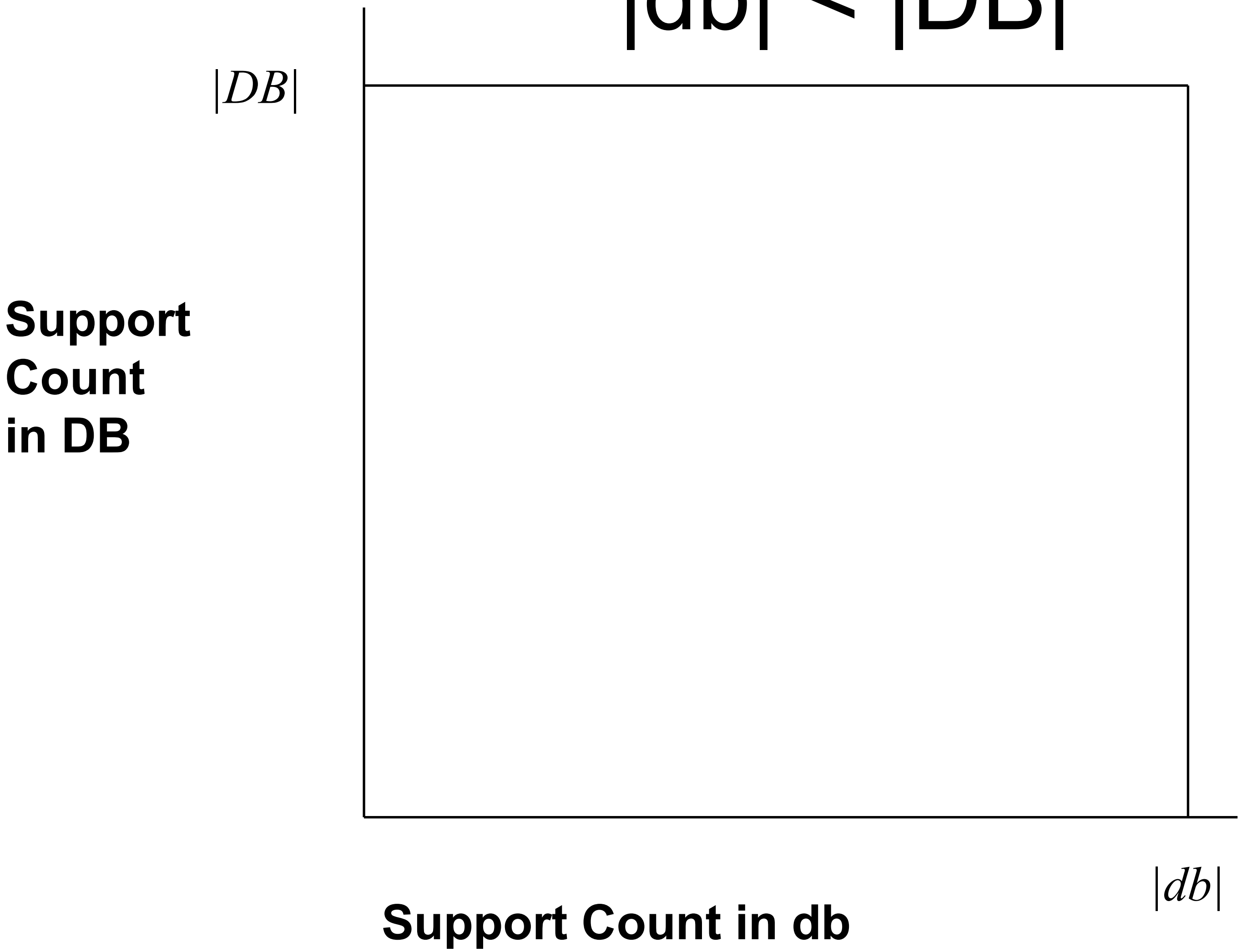
Notation

- *DB* is the set of old transactions (where transactions denote the records in the original database)
- *db* is the set of new coming transactions (the increment)
- *DB*+*db* is the set of old and new coming transactions
- $support_{DB}$ (X) is the support of itemset X in *DB*
- $support_{db}$ (X) is the support of X in *db*,
- $support_{DB+db}$ (X) is the support of X in *DB*+ *db*

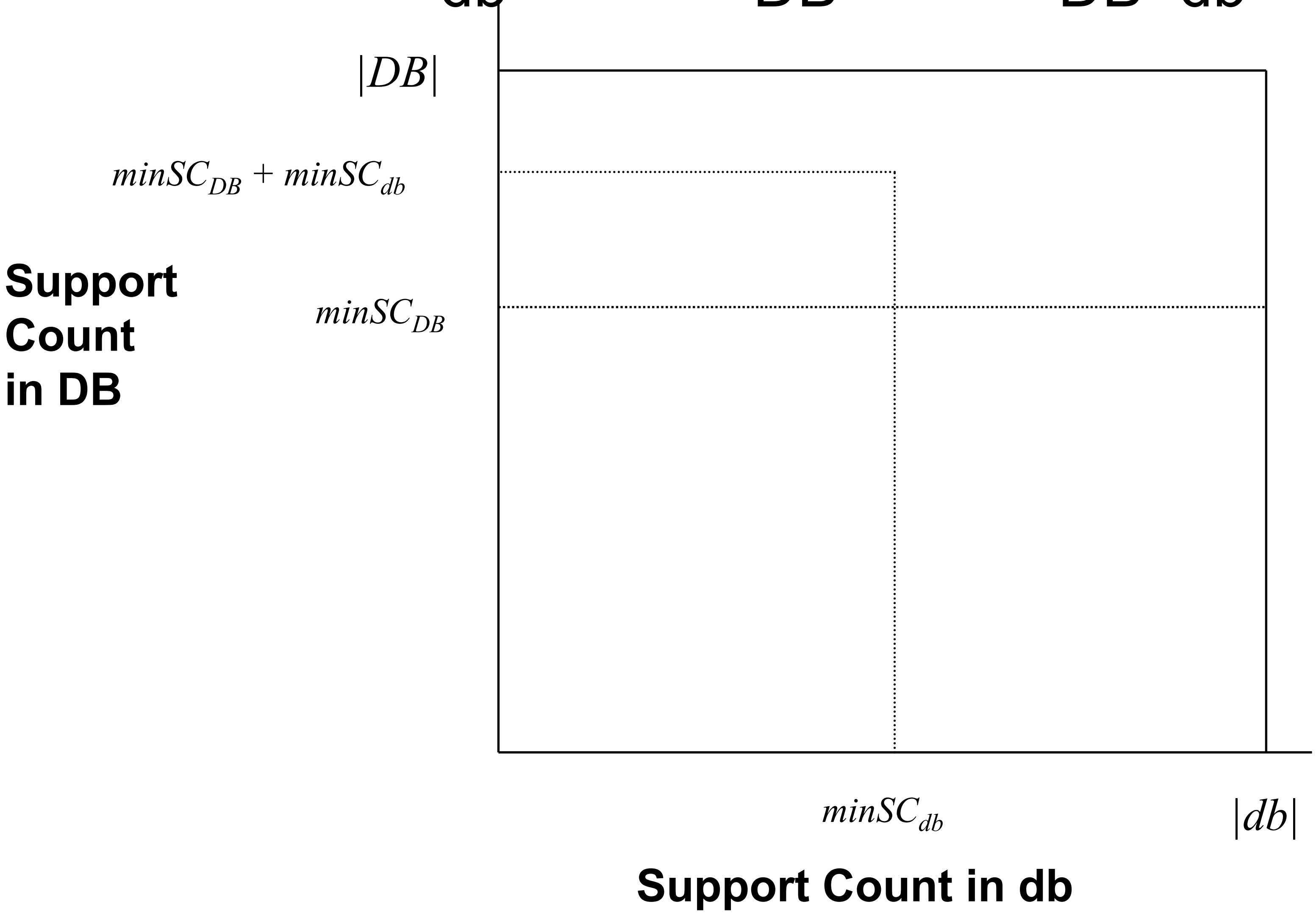
Notation – cont.

- SC_{DB} X support count of X in DB
- SC_{db} X support count of X in db
- s_{db} minimum support for itemset X in db
- s_{DB} minimum support for itemset X in DB
- s minimum support for itemset X in DB+db
- T - transaction in DB or db
- $minsupp_A$ is the minimum support threshold for database/increment A

Plotting Support Count in DB + db
|db| < |DB|



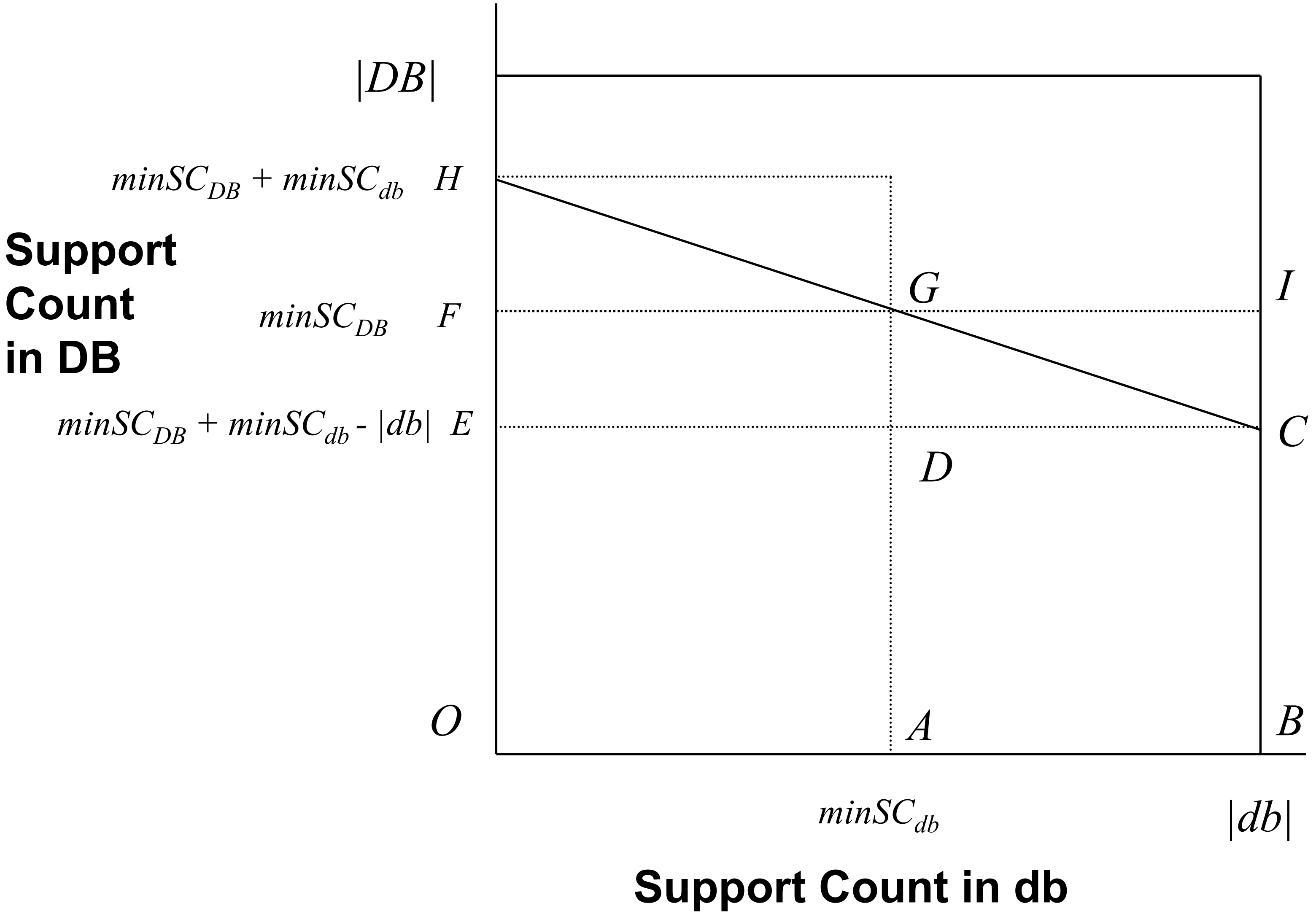
For each point (SC_{db}, SC_{DB})
$SC_{db} + SC_{DB} = SC_{DB+db}$



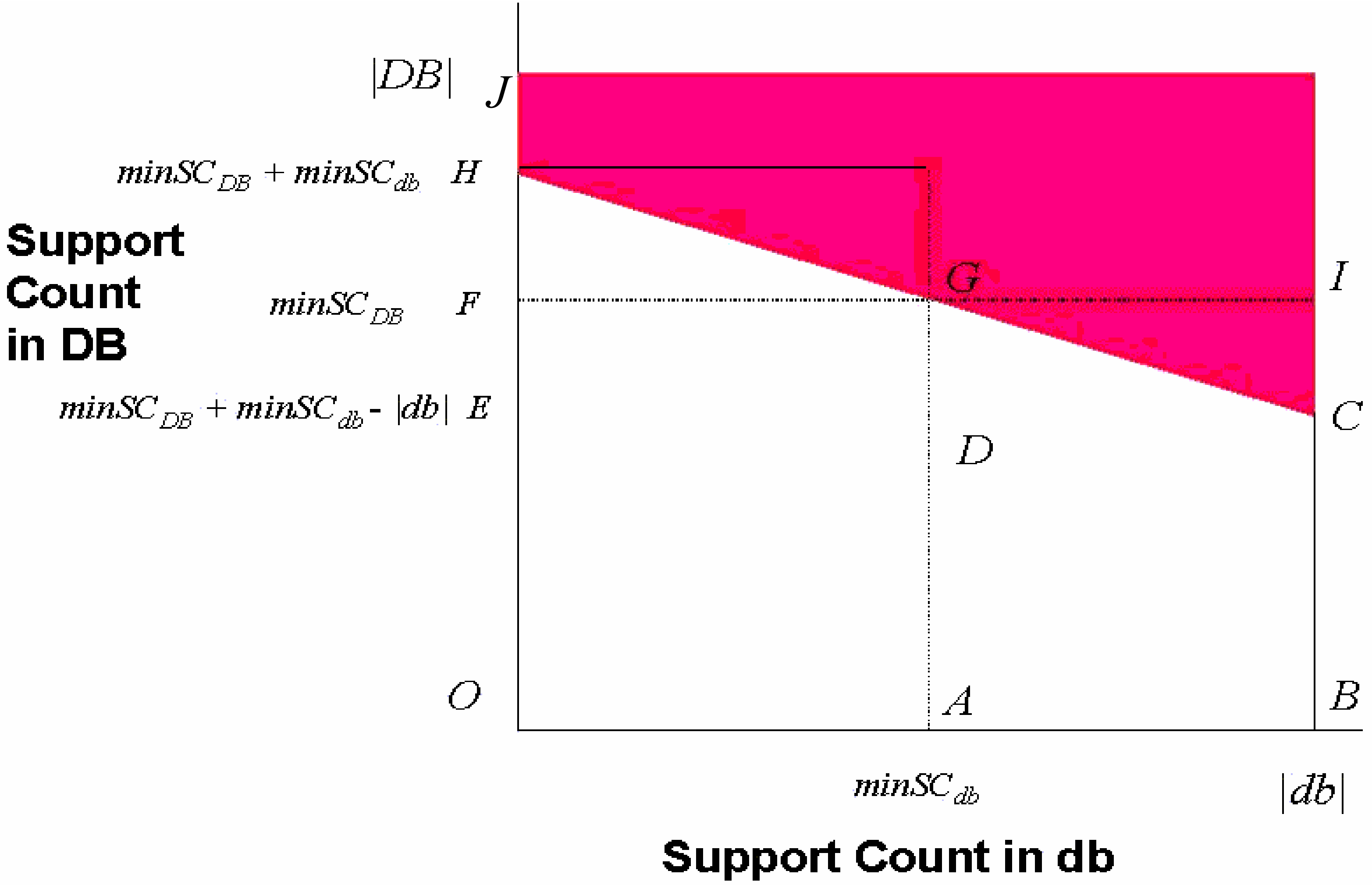
$minSC_{db}$ = *minium # transactions to be large in db*

$minSC_{DB}$ = *minium # transactions to be large in DB*

For all points G on line HC $SC_{db} + SC_{DB} = minSC_{DB+db}$



Line HC partitions the space of itemsets.
All itemsets above and including HC are large.
All below HC are small. Incremental itemset problem, find all itemsets in the red area



The possibilities

	Large in DB	Small in DB
Large in db	Large in DB + db	??
Small in db	??	Small in DB + db

FUP – David W. Cheung, Jiawei Han, Vincent T. Ng, C.Y. Wong

- Based on Apriori and DHP
- Iteration 1 – Find large 1 itemsets
  - For all itemsets in L_{1DB} check support in db and update
  - Remove "losers" . Place winners in L_{1DB+db}.
  - while scanning db, find all 1-itemsets in db that are not in L_{1DB}. These are candidate 1 – itemsets and are placed in C_1
  - Prune from C_1 all itemsets that have support below threshold *s* in db. These have no chance of being large.
  - Find count of each remaining itemset X in C_1 from DB. If $support_{DB+db}(X) > s$ then place in L_{1DB+db}.
  - Reduce database

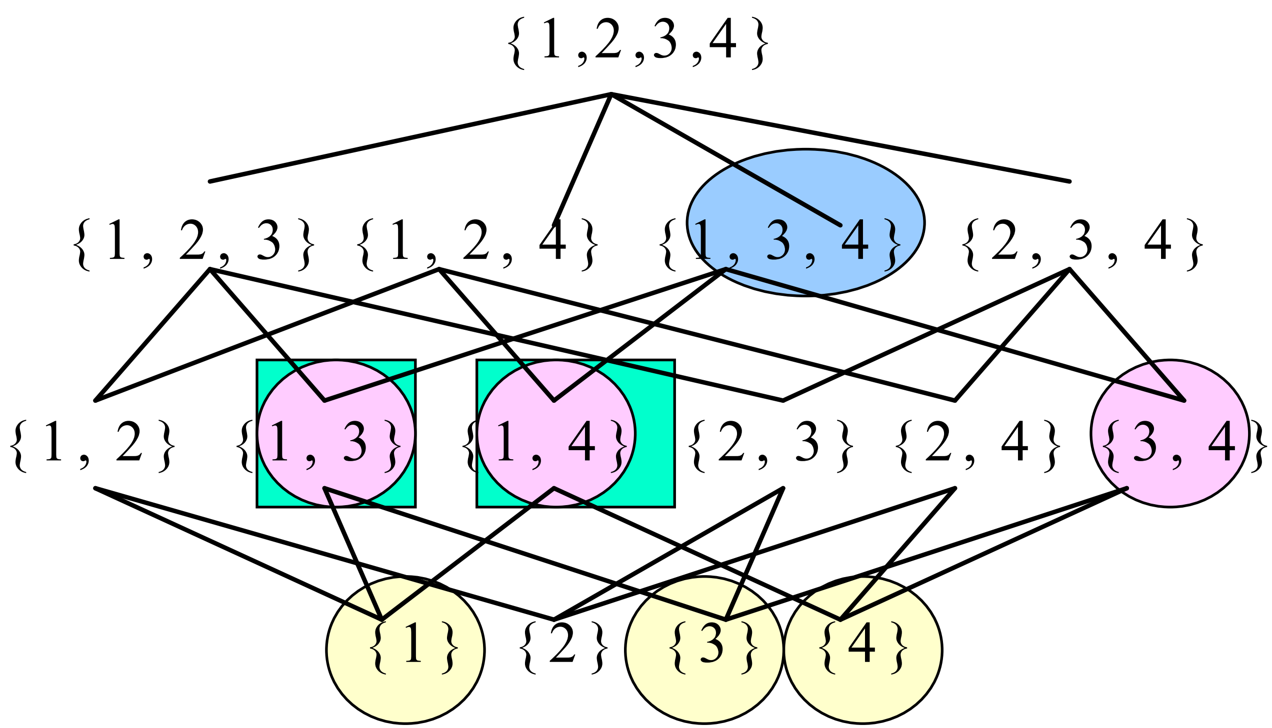
K - iteration

- **Remove k - superset of any small 1-itemset**
- **For all items in $L_{k\ DB}$ check support in db.**
- **Remove losers. Winners are in $L_{k\ DB+db}$**
- **C_k = apriorigen ($L_{k-1\ DB+db}$) - $L_{k\ DB}$**
- **Scan db for support of all X in C_k**
- **While scanning reduce db**
- **Remove X from C_k if $support_{db}(X) < s_{db}$ and place X in P**
- **For remaining X in C_k scan DB and update support.**
- **If support $_{DB+db}$ (X) in $C_k > s_{DB+db}$ then $X \in L_{k\ DB+db}$**
- **Any items in DB that are not in C_k or $L_{k\ DB+db}$ can be pruned from DB when scanning for support $_{DB+db}$ (X)**

FUP and database reduction

- From DHP - An itemset *I* at level k+1 has k+1 subsets at level k
- Any one item $i \in I$ will appear in only k of those subsets.
- Therefore if we look at each item *i* in a transaction T from db, and count the subsets in T relative to C_k and $L_{k\ DB+db}$ in which *i* appears, if *i* is in fewer than k sets, it can't be in any k+1 itemsets.

The Lattice (Again!)



1. **Notice set {1,3,4} at level 3 has 3 subsets**
2. **Each item 1, 3, and 4 only occurs in 2 subsets of {1,3,4} at level 2**

An FUP Example

DB	
TID	Items
1	A,C,D,E,F
2	B,D,F
3	A,D,E
4	A,B,D,E,F
5	A,B,C,F
6	B.F
7	A,D,E,F
8	A,B,D,F
9	A,D,F

db	
TID	Items
1	A,F
2	B,C,F
3	A,C
4	B,F
5	A,B,C
6	A,C,D

Let s = 30%

Thus s_{DB} = .30 * |DB| = 3

s_{db} = .30 * |db| = 2

s = .30 * |DB+db| = 5

L_1		L_2		L_3		L_4	
Itemset	Support	Itemset	Support	Itemset	Support	Itemset	Support
{A}	7	{AB}	3	{ABF}	3	{ADEF}	3
{B}	5	{AD}	6	{ADE}	4		
{D}	7	{AE}	4	{ADF}	5		
{E}	4	{AF}	6	{AEF}	3		
{F}	8	{BD}	3	{BDF}	3		
		{BF}	5	{DEF}	3		
		{DE}	4				
		{DF}	6				
		{EF}	3				

Update support from db

L_1	
Itemset	Support $_{DB+db}$
{A}	11
{B}	8
{D}	7
{E}	4
{F}	11

⇨

$L_{1\ DB+db}$	
Itemset	Support
{A}	11
{B}	8
{D}	7
{F}	11
{C}	6

C_1	
Itemset	Support$_{db}$
{C}	4

⇦

C_1	
Itemset	Support$_{DB+db}$
{C}	6

Remove 2 - supersets of any small 1-itemset

L_2	
Itemset	Support
{AB}	3
{AD}	6
~~{AE}~~	4
{AF}	6
{BD}	3
{BF}	5
~~{DE}~~	4
{DF}	6
~~{EF}~~	3

For all items in $L_{2\ DB}$ check support in db and update for DB+db. Place "Winners" in $L_{2\ DB+db}$.

L_2		$L_{2\ DB+db}$	
Itemset	Support $_{DB+db}$	Itemset	Support
{AB}	4	{AD}	7
{AD}	7	{AF}	7
{AF}	7	{BF}	7
{BD}	3	{DF}	6
{BF}	7		
{DF}	6		

C_2 = apriorigen ($L_{1\ DB+db}$) - $L_{2\ DB}$

Scan db for support of all X in C_2

If support $_{DB+db}$ (X) in C_2 > s_{DB+db} then $X \in L_{2\ DB+db}$

C_2		
Itemset	$Support_{db}$	$Support_{DB+db}$
{AC}	3	5
{BC}	2	2
{CD}	1	1
{CF}	1	3

$L_{2\ DB+db}$	
Itemset	Support
{AD}	7
{AF}	7
{BF}	7
{DF}	6
{AC}	5

Remove 3 - supersets of any small 1-itemset

L_3	
Itemset	Support
{ABF}	3
~~{ADE}~~	4
{ADF}	5
~~{AEF}~~	3
{BDF}	3
~~{DEF}~~	3

For all items in $L_{3\ DB}$ check support in db and update for DB+db. Place "Winners" in $L_{3\ DB+db}$.

L_3		$L_{3\ DB+db}$	
Itemset	Support $_{DB+db}$	Itemset	Support
{ABF}	3	{ADF}	5
{ADF}	5		
{BDF}	3		

C_3 = apriorigen ($L_{2\ DB+db}$) - $L_{3\ DB}$
Scan db for support of all X in C_3
If support $_{DB+db}$ (X) in C_3 > s_{DB+db} then $X \in L_{3\ DB+db}$

C_3		
Itemset	$Support_{db}$	Support $_{DB+db}$
{ACF}	0	2
{ACD}	1	2

$L_{3\ DB+db}$	
Itemset	Support
{ADF}	5

Remove 4 - supersets of any small 1-itemset

L_4	
Itemset	Support
~~{ADEF}~~	3

Why are deletions harder than insertions?

support = itemset count in database/ number records in database

	Large in DB	Small in DB
Large in db+	Large in DB + db (need not rescan DB)	??
Small in db+	??	Small in DB + db (need not rescan DB)

	Large in DB	Small in DB
Large in db-	??	??
Small in db-	??	??

FUP$_2$ - David Cheung, S. D. Lee, Benjamin Kao

Generalization of FUP

Notation:

- Δ^- deletions where db- $\subseteq$ DB
- Δ^+ insertions
- DB+db = (DB - Δ^-) U Δ^+

database	support count of itemset X	Large k-itemsets
Δ^+	δ_X^+	—
D^-	—	—
Δ^-	δ_X^-	—
$D = \Delta^- \cup D^-$	σ_X	L_k
$D' = D^- \cup \Delta^+$	σ'_X	L'_k

Table 1: Definitions of several symbols

Deletions Only $db^+ = \emptyset$

- L1 is found by checking support of all items I
- An efficiency – Partition Ck into P and Q where P= Ck $\cap$Lk . Therefore to find support in DB+db, you need only to update with the support of the itemset in the increment.
- $Q_{,k}$ contains all itemsets that were previously small. If an itemset from Q_k is large in the increment, then it can't be large in the updated database.

1. Obtain a candidate set C_k of itemsets. Halt if $C_k = \emptyset$.
2. Partition C_k into P_k and Q_k, where $P_k = C_k \cap L_k$ and $Q_k = C_k - P_k$.
3. Scan Δ^- to find out δ_X^- for each $X \in P_k \cup Q_k$.
4. For each $X \in P_k$, Calculate σ'_X.
5. Delete from Q_k those candidates X where $\delta_X^- \geq |\Delta^-| \times s\%$. (Application of Lemma 2.)
6. Scan D^- to find out σ'_X of the remaining candidates $X \in Q_k$.
7. Add to L'_k those candidates X from $P_k \cup Q_k$ for which $\sigma'_X \geq |D'| \times s\%$.
8. Halt if $|L'_k| < k + 1$.

Optimize – we need not find, in the increment, the support of some small itemsets

- Supersets of a small itemsets are small
- Don't scan the increment for those itemsets from Q_k that contain a small 1-itemset.

FUP_2 Insertions and Deletions

- Use $L_{k-1\ DB+db}$ to generate candidates C_k
- Partition C_k into P_k and Q_k
- We know $support_{DB}$ of the itemsets in P_k, therefore only scan db^+ and db^- to update support
- Add itemsets with sufficient updated support to $L_{k\ DB+db}$
- Prune from Q_k those itemsets that are small in $db^+ + db^-$. Since these were previously small, they can't be in $L_{k\ DB+db}$

More Optimizations

- For $X \subseteq Y$, support $_X \leq$ support $_y$
- The k-1 subset of a k-itemset Y will have the smallest support of all subsets of Y of size < k.
- Therefore, the minimum support of all k-1 itemsets gives us an upper bound (b^+_x) on the support of Y in db^+ and an upper bound (b^-_x) on the support of Y in db^-.
- Prune from Q_k those itemsets where
  $b^+_x \leq (|db^{+|} - |db^-|) * s\%$
- Prune from P_k those itemsets, X where
  $support_{DB} + (b^+_x) < (|DB+db|) * s\%$
- Prune from Q_k those itemsets where
  $b^+_x - support_{db-\,x} \leq (|db^+| - |db^-|) * s\%$
- Prune from P_k those itemsets, X where
  $support_{DB} + b^+_x - support_{db-\,x} < (|DB+db|) * s\%$
- if $|db^-| < |db^+|$ place into R_k from $Q_{,k}$ those itemsets where
  $b^+_x - b^-_x \leq (|db^+| - |db^-|) * s\%$
- You can prune from R_k those itemsets where
  $support_{db+} \leq (|db^+| - |db^-|) * s\%$

1. Obtain a candidate set C_k of itemsets. Halt if $C_k = \emptyset$.
2. Calculate b_X^+ for each $X \in C_k$.
3. Partition C_k into P_k and Q_k.
4. For each $X \in P_k$, remove it if $\sigma_X + b_X^+ < |D'| \times s\%$.
5. For each $X \in Q_k$, remove it if $b_X^+ \leq (|\Delta^+| - |\Delta^-|) \times s\%$.
6. If $|\Delta^-| \leq |\Delta^+|$, let $R_k = \emptyset$. Otherwise, calculate b_X^- for each $X \in Q_k$ and if $b_X^+ - b_X^- \geq (|\Delta^+| - |\Delta^-|) \times s\%$, move it to R_k and assign b_X^- to δ_X^-.
7. Scan Δ^- to find out δ_X^- for each $X \in P_k \cup Q_k$.
8. Delete from P_k those candidates X where $\sigma_X + b_X^+ - \delta_X^- < |D'| \times s\%$.
9. Delete from Q_k those candidates with $b_X^+ - \delta_X^- \leq (|\Delta^+| - |\Delta^-|) \times s\%$.
10. Scan Δ^+ to find δ_X^+ for each $X \in P_k \cup Q_k \cup R_k$.
11. For each candidate $X \in P_k$, calculate σ'_X.
12. For each candidate $X \in Q_k$, delete X if $\delta_X^+ - \delta_X^- \leq (|\Delta^+| - |\Delta^-|) \times s\%$.
13. For each candidate $X \in R_k$, delete X if $\delta_X^+ \leq (|\Delta^+| - |\Delta^-|) \times s\%$.
14. Scan D^- and get the count of each $X \in Q_k \cup R_k$. Then, add this count to δ_X^+ to get σ'_X.
15. Add to L'_k those candidates X from $P_k \cup Q_k \cup R_k$ where $\sigma'_X \geq |D'| \times s\%$.
16. Halt if $|L'_k| < k + 1$.

UWEP An Incremental Association Rule Algorithm

- Scan db and find counts for all 1-itemsets
- Pruning – for all itemsets in DB that are found small we prune the supersets of that itemset from DB.
- Check the large itemsets in DB whose items are absent in db $support_{DB}(X) = support_{DB+db}(X)$
- Check large itemsets in db to see if they are large in DB, these are large by definition
- For all large itemsets that are large in DB and have not been checked, check to see if they are large in DB + db

QUESTIONS?????????

K - iteration

- **Remove k - superset of any small 1-itemset**
- **For all items in $L_{k\ DB}$ check support in db.**
- **Remove losers. Winners are in $L_{k\ DB+db}$**
- **C_k = apriorigen ($L_{k-1\ DB+db}$) - $L_{k\ DB}$**
- **Scan db for support of all X in C_k**
- **While scanning reduce db**
- **Remove X from C_k if $support_{db}(X) < s_{db}$ and place X in P**
- **For remaining X in C_k scan DB and update support.**
- **If $support_{DB+db}(X)$ in $C_k > s_{DB+db}$ then $X \in L_{k\ DB+db}$**
- **Any items in DB that are not in C_k or $L_{k\ DB+db}$ can be pruned from DB when scanning for $support_{DB+db}(X)$**

K - iteration

- **Remove k - superset of any small 1-itemset**
- **For all items in $L_{k\ DB}$ check support in db.**
- **Remove losers. Winners are in $L_{k\ DB+db}$**
- **C_k = apriorigen ($L_{k-1\ DB+db}$) - $L_{k\ DB}$**
- **Scan db for support of all X in C_k**
- **While scanning reduce db**
- **Remove X from C_k if $support_{db}(X) < s_{db}$ and place X in P**
- **For remaining X in C_k scan DB and update support.**
- **If $support_{DB+db}(X)$ in $C_k > s_{DB+db}$ then $X \in L_{k\ DB+db}$**
- **Any items in DB that are not in C_k or $L_{k\ DB+db}$ can be pruned from DB when scanning for $support_{DB+db}(X)$**