InfoLab

IC612: Data Warehousing and Data Mining

Lecture 5:
Association Pattern Mining: Advanced Concepts

Min-Soo Kim

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

Introduction

- **Association pattern mining algorithms often discover a large number of patterns**
  - difficult to use this large output for application-specific tasks
  - may be uninteresting or redundant for a specific application
- **Summarization**
  - a smaller set of discovered itemsets is much easier to understand and assimilate
- **Querying**
  - users may wish to query them for smaller summaries
- **Constraint incorporation**
  - one may wish to incorporate application-specific constraints into the itemset generation process

Pattern Summarization

- **Vast majority of the generated patterns are often redundant**
  - because of the downward closure property
  - all subsets of a frequent itemset are also frequent
- **Maximal patterns**
- **Closed patterns**
- **Approximate frequent patterns**

Maximal patterns

Definition 5.2.1 (Maximal Frequent Itemset) *A frequent itemset is maximal at a given minimum support level minsup if it is frequent and no superset of it is frequent.*

- **Trivial method to find maximal itemsets**
  - step 1: use any frequent itemset mining algorithm to find all itemsets
  - step 2: find the maximal ones by post-processing phase
    - examining itemsets in decreasing order of length, and removing proper subsets
    - continued until all itemsets have either been examined or removed

- **Better method**
  - design algorithms that can directly prune parts of the search space of patterns during frequent itemset discovery
  - $P \cup F(P)$ is a subset of a frequent pattern, P is removed

Closed patterns

Definition 5.2.2 (Closed Itemsets) *An itemset X is closed, if none of its supersets have exactly the same support count as X.*

- **X: closed itemset**
- **S(X): equi-support subsets of X**

Observation 5.2.1 *Let X be a closed itemset, and $\mathcal{S}(X)$ be its equi-support subsets. For any itemset $Y \in \mathcal{S}(X)$, the set of transactions $\mathcal{T}(Y)$ containing Y is exactly the same. Furthermore, there is no itemset Z outside $\mathcal{S}(X)$ such that the set of transactions in $\mathcal{T}(Z)$ is the same as $\mathcal{T}(X)$.*

Definition 5.2.3 (Closed Frequent Itemsets) *An itemset X is a closed frequent itemset at minimum support minsup, if it is both closed and frequent.*

Approximate frequent patterns

- **Description in terms of transactions**
  - “almost” closures, where the closure property is not exactly satisfied but is approximately specified
  - transaction membership

- **Description in terms of itemsets themselves**
  - the frequent itemsets are clustered
  - representatives can be drawn from each cluster to provide a concise summary
  - structure of the itemset

Approximation in terms of transactions

- **Allow a "play" δ, within a range of supports**

- **All frequent itemsets F can be segmented into a disjoint set of k "almost equi-support" groups F_1 . . .F_k**
  - for any pair of itemsets X, Y within any group F_i, the value of |sup(X)−sup(Y)| is at most δ
  - when δ is chosen to be 0, this is exactly the set of closed itemsets

Approximation in terms of itemsets

- **Distance function Dist(X, Y) between itemsets X and Y**
  - any distance function for set-valued data, such as the Jaccard coefficient, may be used

Definition 5.2.4 (δ-Approximate Sets) *The set of representatives $\mathcal{J} = \{J_1 \ldots J_k\}$ is δ-approximate, if for each frequent pattern $X \in \mathcal{F}$, and each $J_i \in \mathcal{J}$, the following is true:*

$$Dist(X, J_i) \leq \delta \quad (5.1)$$

- **Goal : to determine the smallest value of k for a particular level of compression δ**
- **Greedy algorithm**
  1. start with J = {}
  2. add the first element from F to J that covers the maximum number of itemsets in F (the covered itemsets are removed from F)
  3. repeat iteratively until the set F is empty

Pattern querying

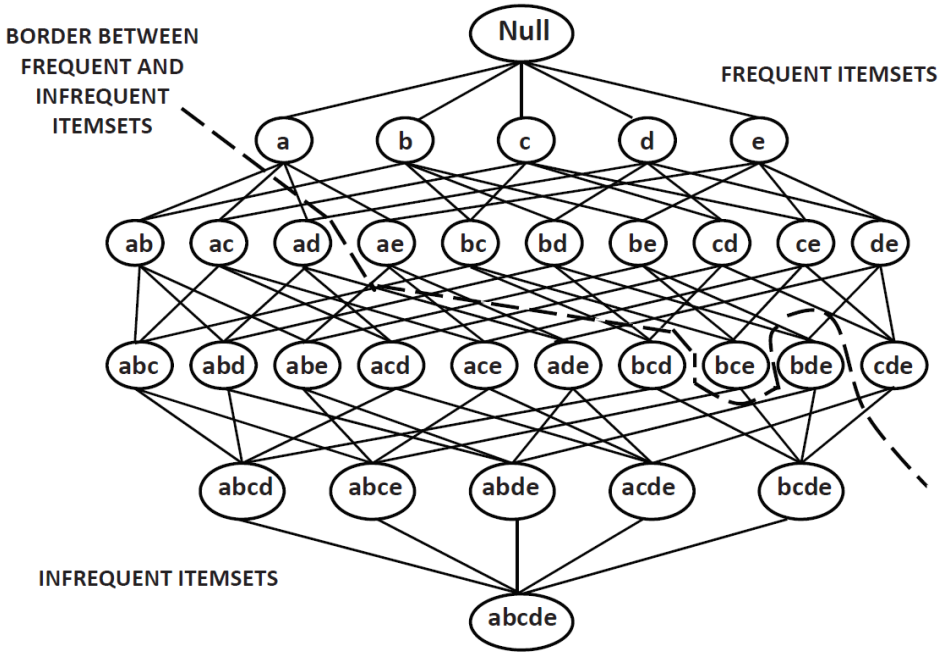
- **Query responses provide the relevant sets of patterns in an application**
  - the relevant set is usually much smaller than the full set of patterns
- **Examples**
  - Report all the frequent patterns containing *X* that have a minimum support of *minsup*
  - Report all the association rules containing X that have a minimum support of minsup and a minimum confidence of *minconf*
- **Two classes of methods**
  - Preprocess-once query-many paradigm
  - Constraint-based pattern mining

Preprocess-once Query-many Paradigm

- **Determine all the frequent patterns at a very low value of the support**
- **Resulting itemsets are arranged in the form of a data structure for querying**
  - simplest data structure is the itemset lattice (graph data structure for querying)
- **Very effective for the case of simpler queries**

Leveraging the Itemset Lattice

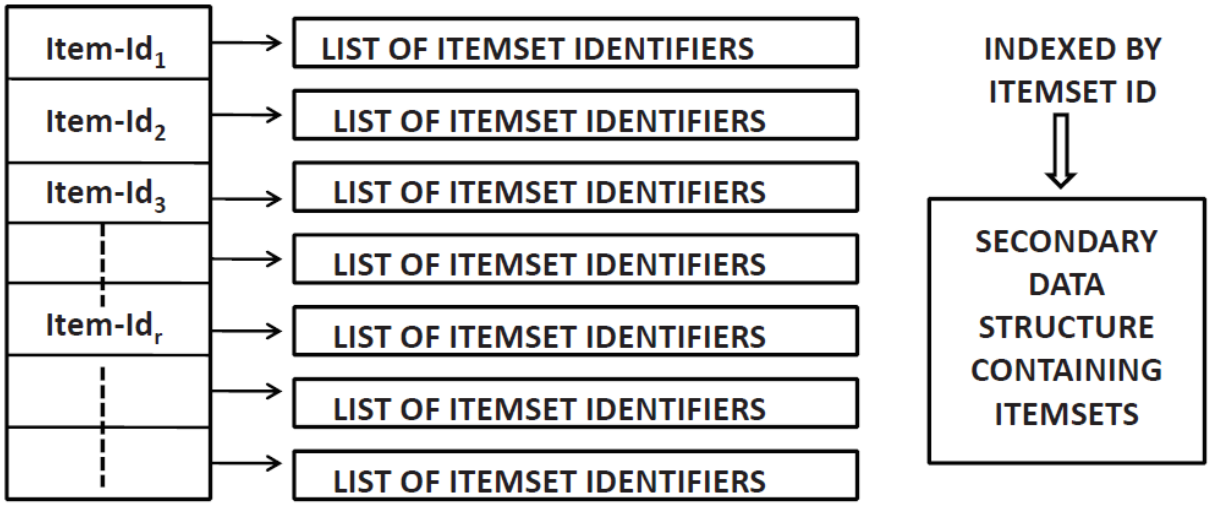
- **Query 1: determine all itemsets containing a set X**
- **Query 2: determine maximal itemsets directly**
  - during the traversal by identifying nodes that do not have edges to their immediate supersets
- **…**



Leveraging Data Structures for Querying

- **In some cases, it is desirable to use disk-resident representations for querying**

- **Inverted index / signature table**
  - drawback: not allow an ordered exploration of the set of frequent patterns



Pushing Constraints into Pattern Mining

- **Advantage**
  - constraints can be used to prune out many of the intermediate itemsets during the execution of the frequent pattern mining algorithms
  - this allows the use of much lower minimum support levels

Constraint-Based Frequent Pattern Mining

- **Pattern space pruning constraints**
  - Anti-monotonic: If constraint c is violated, its further mining can be terminated
  - Monotonic: If c is satisfied, no need to check c again
  - Succinct: c must be satisfied, so one can start with the data sets satisfying c
  - Convertible: c is not monotonic nor anti-monotonic, but it can be converted into it if items in the transaction can be properly ordered
- **Data space pruning constraint**
  - Data succinct: Data space can be pruned at the initial pattern mining process
  - Data anti-monotonic: If a transaction t does not satisfy c, t can be pruned from its further mining

Pattern Space Pruning with Anti-Monotonicity Constraints

- A constraint C is *anti-monotone* if the super pattern satisfies C, all of its sub-patterns do so too
- In other words, *anti-monotonicity:* If an itemset S **violates** the constraint, so does any of its superset
- Ex. 1. $sum(S.price) \leq v$ is anti-monotone
- Ex. 2. range(S.profit) $\leq$ 15 is anti-monotone
  - Itemset *ab* violates C
  - So does every superset of *ab*
- Ex. 3. $sum(S.Price) \geq v$ is not anti-monotone
- Ex. 4. *support count* is anti-monotone: core property used in Apriori

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Pattern Space Pruning with Monotonicity Constraints

- A constraint C is *monotone* if the pattern satisfies C, we do not need to check C in subsequent mining
- Alternatively, monotonicity: *If an itemset S* ***satisfies*** *the constraint, so does any of its superset*
- Ex. 1. $sum(S.Price) \geq v$ is monotone
- Ex. 2. $min(S.Price) \leq v$ is monotone
- Ex. 3. C: range(S.profit) $\geq$ 15
  - Itemset *ab* satisfies C
  - So does every superset of *ab*

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Data Space Pruning with Data Anti-monotonicity

- A constraint c is *data anti-monotone* if for a pattern p cannot satisfy a transaction t under c, p's superset cannot satisfy t under c either
- The key for data anti-monotone is *recursive data reduction*
- Ex. 1. $sum(S.Price) \geq v$ is data anti-monotone
- Ex. 2. $min(S.Price) \leq v$ is data anti-monotone
- Ex. 3. C: $range(S.profit) \geq 25$ is data anti-monotone
  - Itemset {b, c}'s projected DB:
    - T10': {d, f, h}, T20': {d, f, g, h}, T30': {d, f, g}
  - since C cannot satisfy T10', T10' can be pruned

TDB (min_sup=2)

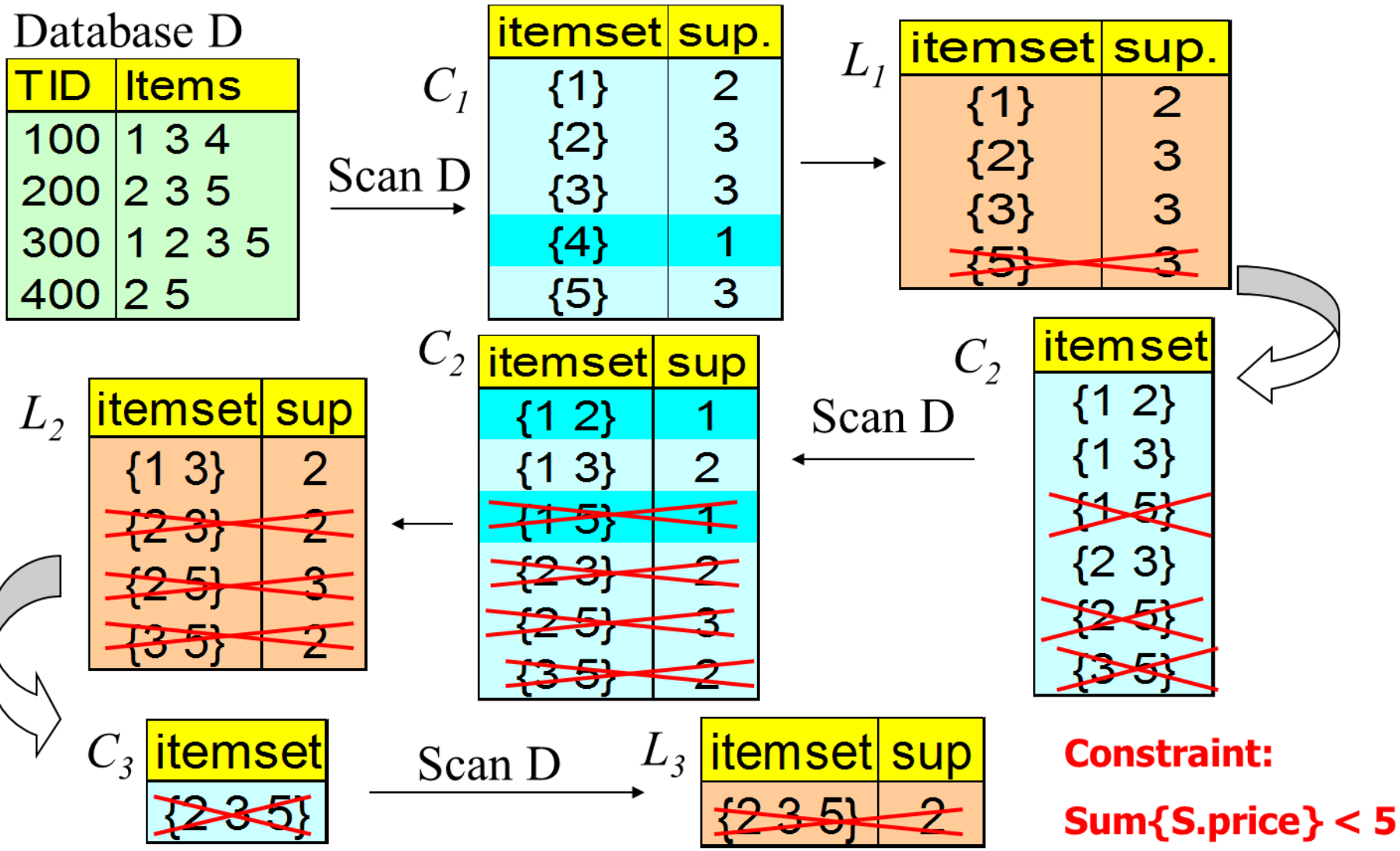
TID	Transaction
10	a, b, c, d, f, h
20	b, c, d, f, g, h
30	b, c, d, f, g
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	-15
e	-30
f	-10
g	20
h	-5

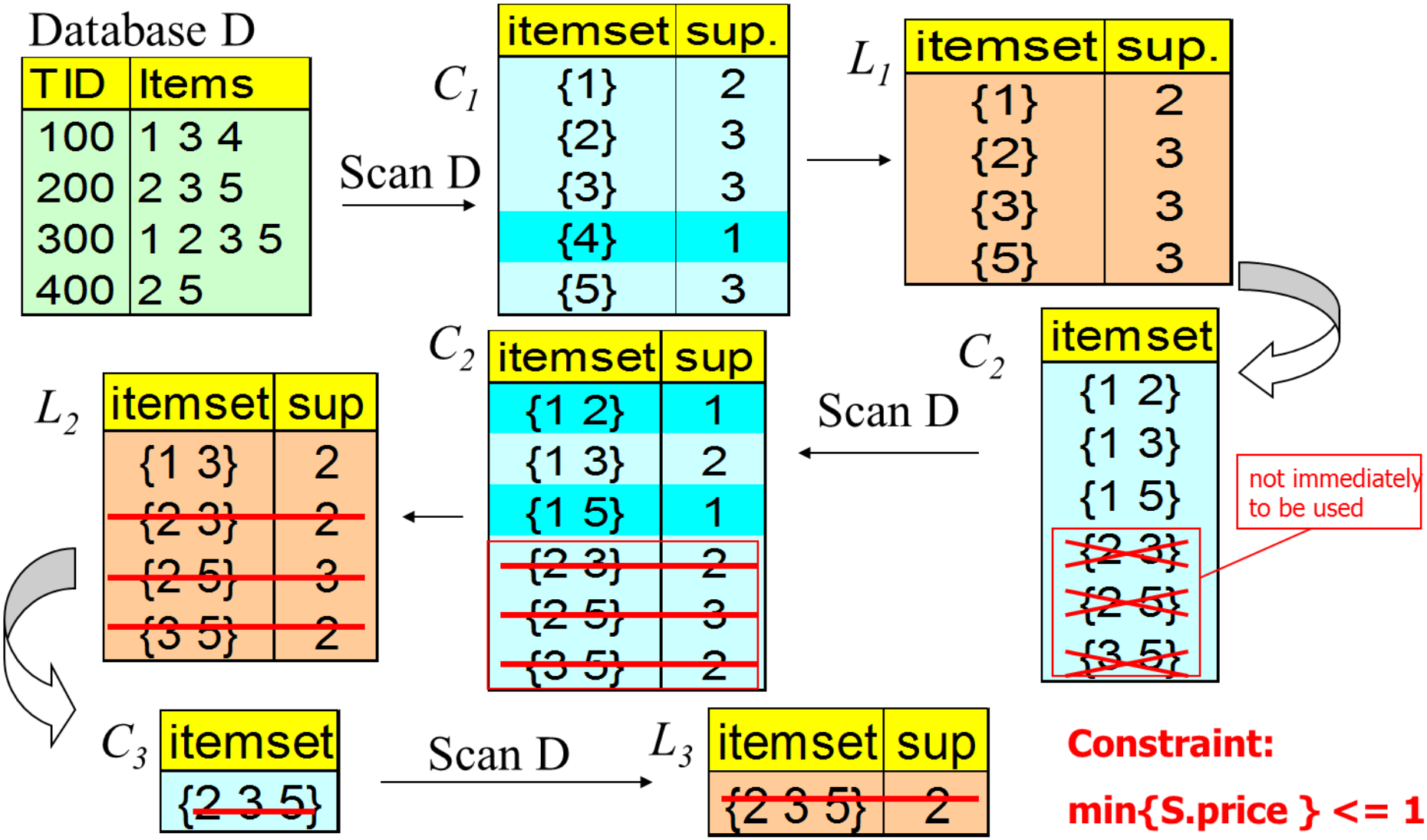
Pattern Space Pruning with Succinctness

- Succinctness:
  - Given A_1, the set of items satisfying a succinctness constraint C, then any set S satisfying C is based on A_1 , i.e., S contains a subset belonging to A_1
  - Idea: Without looking at the transaction database, whether an itemset S satisfies constraint C can be determined based on the selection of items
  - $min(S.Price) \leq v$ is succinct
  - $sum(S.Price) \geq v$ is not succinct
- Optimization: If C is succinct, C is pre-counting pushable

Apriori + Constraint



Constrained Apriori : Push a Succinct Constraint Deep



Convertible Constraints: Ordering Data in Transactions

- Convert tough constraints into anti-monotone or monotone by properly ordering items
- Examine C: avg(*S*.profit) ≥ 25
  - Order items in value-descending order
    - <*a, f, g, d, b, h, c, e*>
  - If an itemset *afb* violates C
    - So does *afbh, afb**
    - It becomes anti-monotone!

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Strongly Convertible Constraints

- avg(X) ≥ 25 is convertible anti-monotone w.r.t. item value descending order R: <*a, f, g, d, b, h, c, e*>
  - If an itemset *af* violates a constraint C, so does every itemset with *af* as prefix, such as *afd*
- avg(X) ≥ 25 is convertible monotone w.r.t. item value ascending order R^{-1}: <*e, c, h, b, d, g, f, a*>
  - If an itemset *d* satisfies a constraint *C*, so does itemsets *df* and *dfa*, which having *d* as a prefix
- Thus, avg(X) ≥ 25 is strongly convertible

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Can Apriori Handle Convertible Constraints?

- A convertible, not monotone nor anti-monotone nor succinct constraint cannot be pushed deep into the an Apriori mining algorithm
  - Within the level wise framework, no direct pruning based on the constraint can be made
  - Itemset df violates constraint C: avg(X) >= 25
  - Since adf satisfies C, Apriori needs df to assemble adf, df cannot be pruned
- But it can be pushed into frequent-pattern growth framework!

Item	Value
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Pattern Space Pruning w. Convertible Constraints

- C: avg(X) >= 25, min_sup=2
- List items in every transaction in value descending order R: <a, f, g, d, b, h, c, e>
  - C is convertible anti-monotone w.r.t. R
- Scan TDB once
  - remove infrequent items
    - Item h is dropped
  - Itemsets a and f are good, …
- Projection-based mining
  - Imposing an appropriate order on item projection
  - Many tough constraints can be converted into (anti)-monotone

Item	Value
a	40
f	30
g	20
d	10
b	0
h	-10
c	-20
e	-30

TDB (min_sup=2)

TID	Transaction
10	a, f, d, b, c
20	f, g, d, b, c
30	a, f, d, c, e
40	f, g, h, c, e

InfoLab

Thank you!

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology