MaxMiner: Mining Max-patterns

- Review!
  - An itemset X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ɔ X
  - i.e., no such a Y
    - Y is a super-pattern of X
    - The support of Y is greater than minSup
      - The support of Y can be smaller than that of X
- MaxMiner is based on the Apriori algorithm
- R. Bayardo. Efficiently mining long patterns from databases. In *SIGMOD'98*

MaxMiner: Mining Max-patterns

- 1st scan: find frequent items and sort them (ascending order)
  - A, B, C, D, E (E is most frequently occurring)
- 2nd scan: find support for 2-itemsets with max-patterns
  - AB, AC, AD, AE, ABCDE
  - BC, BD, BE, BCDE
  - CD, CE, CDE
  - DE

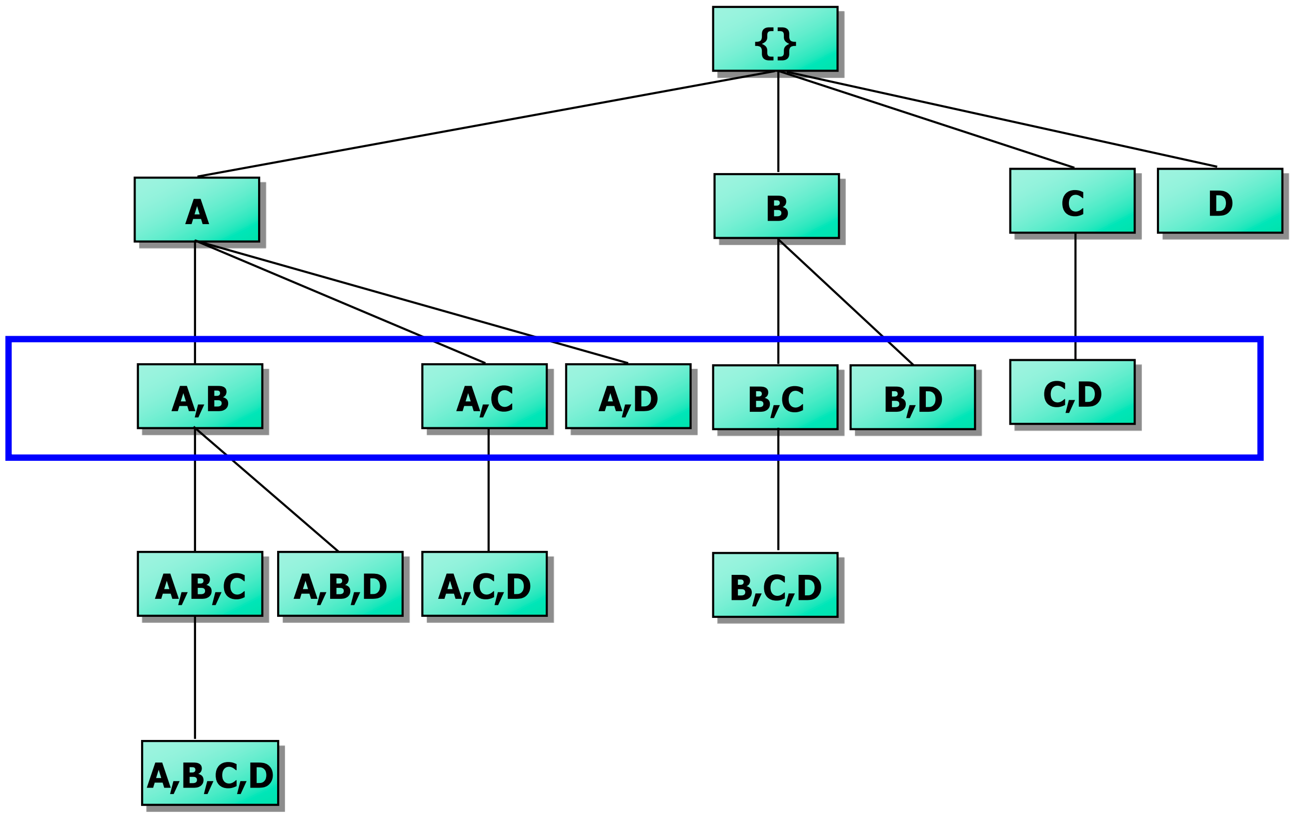
Potential max-patterns

Tid	Items
10	A,B,C,D,E
20	B,C,D,E,
30	A,C,D,F,E

- Reduce a lot of candidates in later stages
  - Since BCDE is a max-pattern, no need to check BCD, BDE, CDE in later scan
  - If AC is infrequent, no need to check ABC in later scans

MaxMiner: Mining Max-patterns

Complete *set-enumeration tree* over four items



MaxMiner: Mining Max-patterns

- 1st scan: find frequent items and sort them (ascending order)
  - A, B, C, D, E (E is most frequently occurring)
- 2nd scan: find support for 2-itemsets with max-patterns
  - AB, AC, AD, AE, ABCDE
  - BC, BD, BE, BCDE
  - CD, CE, CDE
  - DE

Potential max-patterns

Tid	Items
10	A,B,C,D,E
20	B,C,D,E,
30	A,C,D,F,E

- Reduce a lot of candidates in later stages
  - Since BCDE is a max-pattern, no need to check BCD, BDE, CDE in later scan
  - If AC is infrequent, no need to check ABC in later scans

Mining Closed Patterns: CLOSET

- Review
  - An itemset X is closed if X is *frequent* and there exists *no super-pattern* Y ɔ X, *with the same support* as X
  - i.e., no such a Y
    - Y is a super-pattern of X
    - The support of Y is should be the same as that of X

Mining Closed Patterns: CLOSET

- Use the FP-tree for finding frequent patterns
- Flist: list of all frequent items in a support descending order
  - Flist: c-e-f-a-d
- Divide search space
  - Patterns having d
  - Patterns having a but no d
  - Patterns having f but no d and a
  - Patterns having e but no d, a, and f
  - Patterns having c but no d, a, f, and e

Min_sup=2

TID	Items
10	a, c, d, e, f
20	a, b, e
30	c, e, f
40	a, c, d, f
50	c, e, f

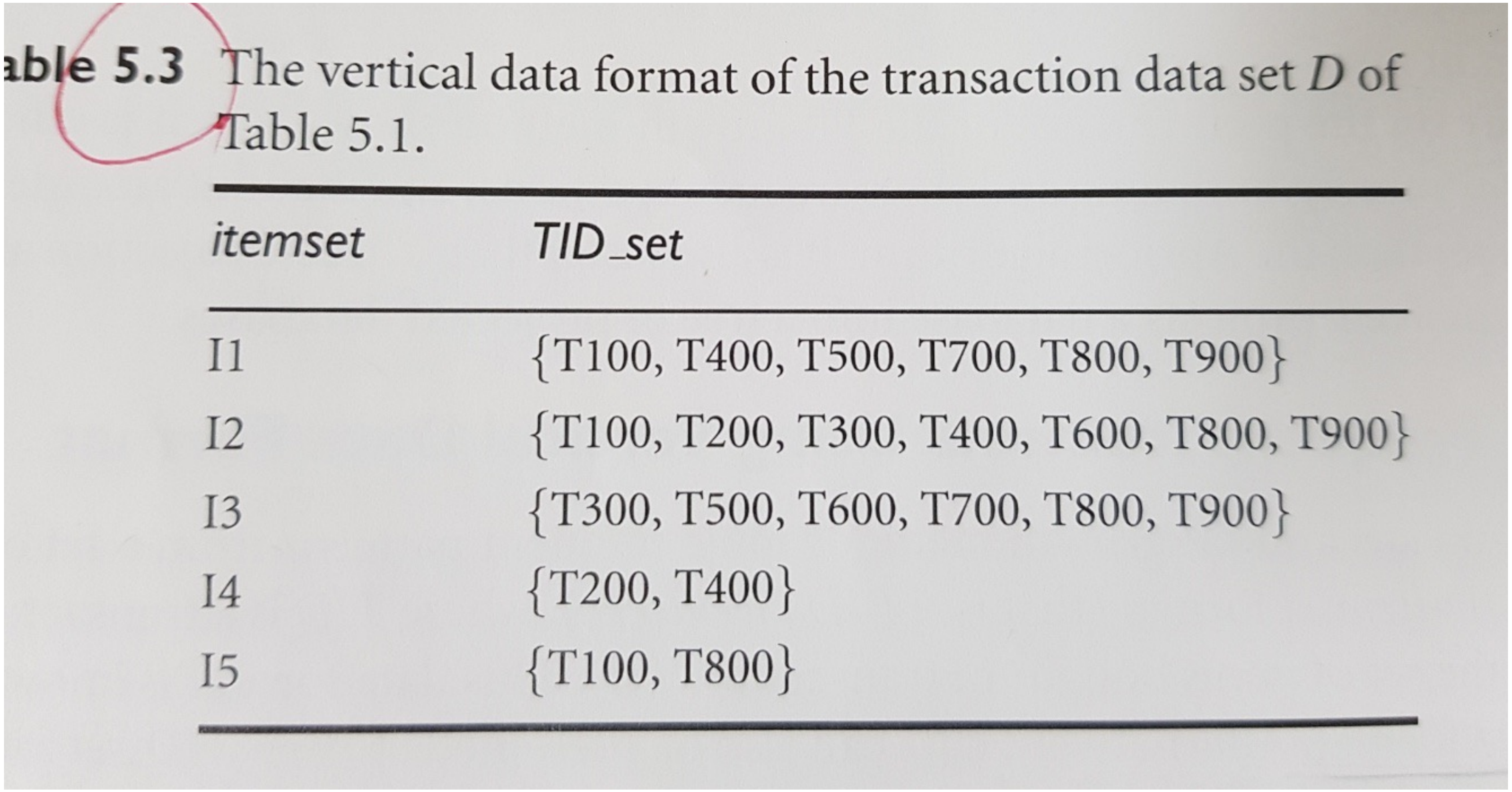
Mining Closed Patterns: CLOSET

- Naïve approach: Quite costly!
  - To mine a complete set of all frequent itemsets
  - To remove every frequent itemset whose support is the same as that of its superset
- Find only the closed itemsets recursively in an efficient way during the mining process using the FP-tree
  - Key idea: every transaction having d also has cfa => cfad is a frequent closed pattern
  - You can consider details by referring to FP-Growth
- J. Pei, J. Han & R. Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets", DMKD'00

CHARM: Mining by Exploring Vertical Data Format

- Vertical format: t(AB) = {T_{11}, T_{25}, …}
  - tid-list: list of trans.-ids containing an itemset
- Algorithm
  - Transform a horizontally formatted data to a vertically format by scanning the dataset once
    - Easy: # of items is much smaller than that of transactions
  - Starting with k=1, construct candidate (k+1)-itemsets from frequent k-itemsets
    - Using the TID-sets intersection and Apriori property
  - Repeat this process with k incremented by 1 until no frequent itemsets can be found

CHARM: Mining by Exploring Vertical Data Format



able 5.3 The vertical data format of the transaction data set *D* of Table 5.1.

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}

CHARM: Mining by Exploring Vertical Data Format

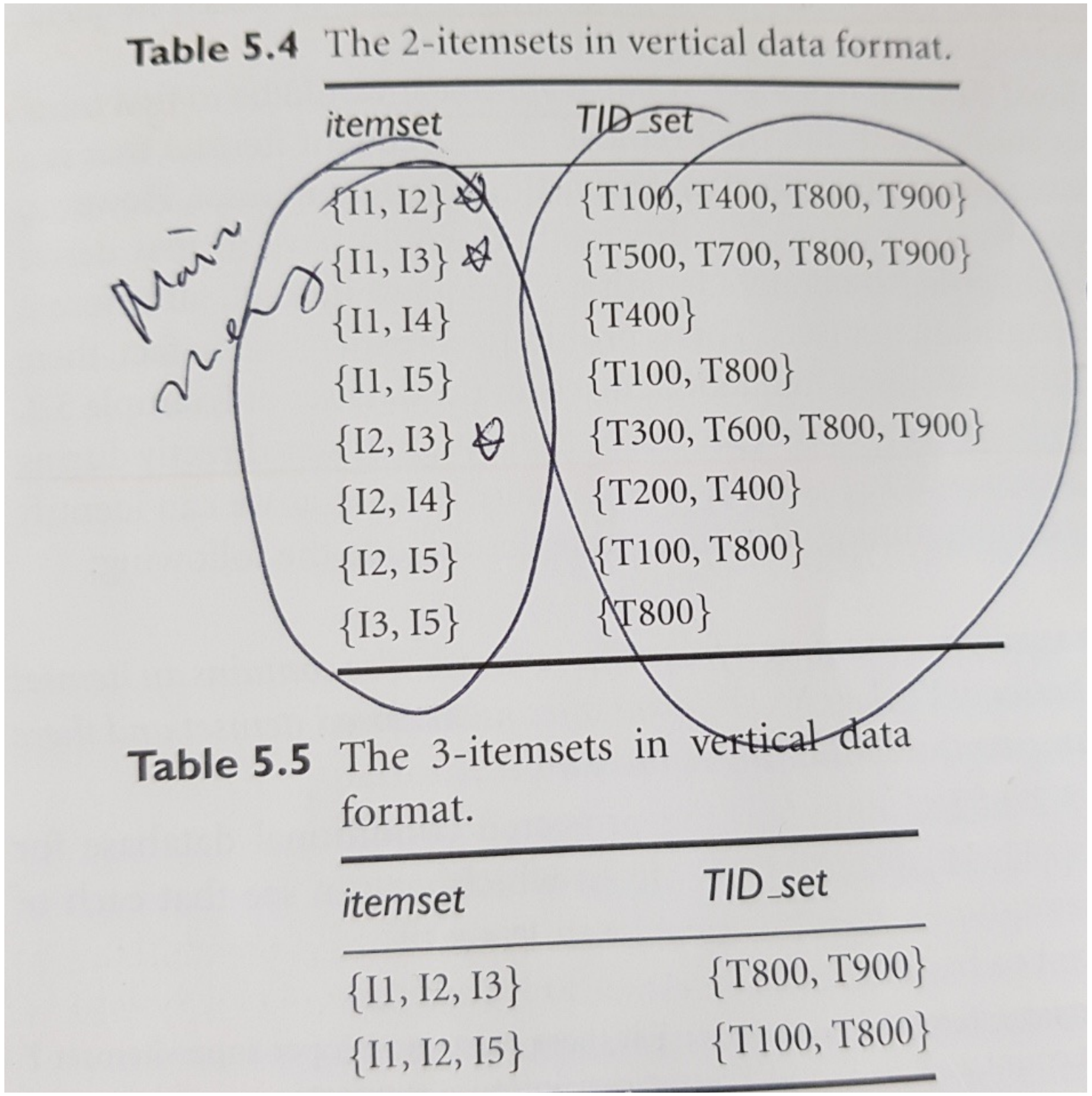


Table 5.4 The 2-itemsets in vertical data format.

Main memory

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}

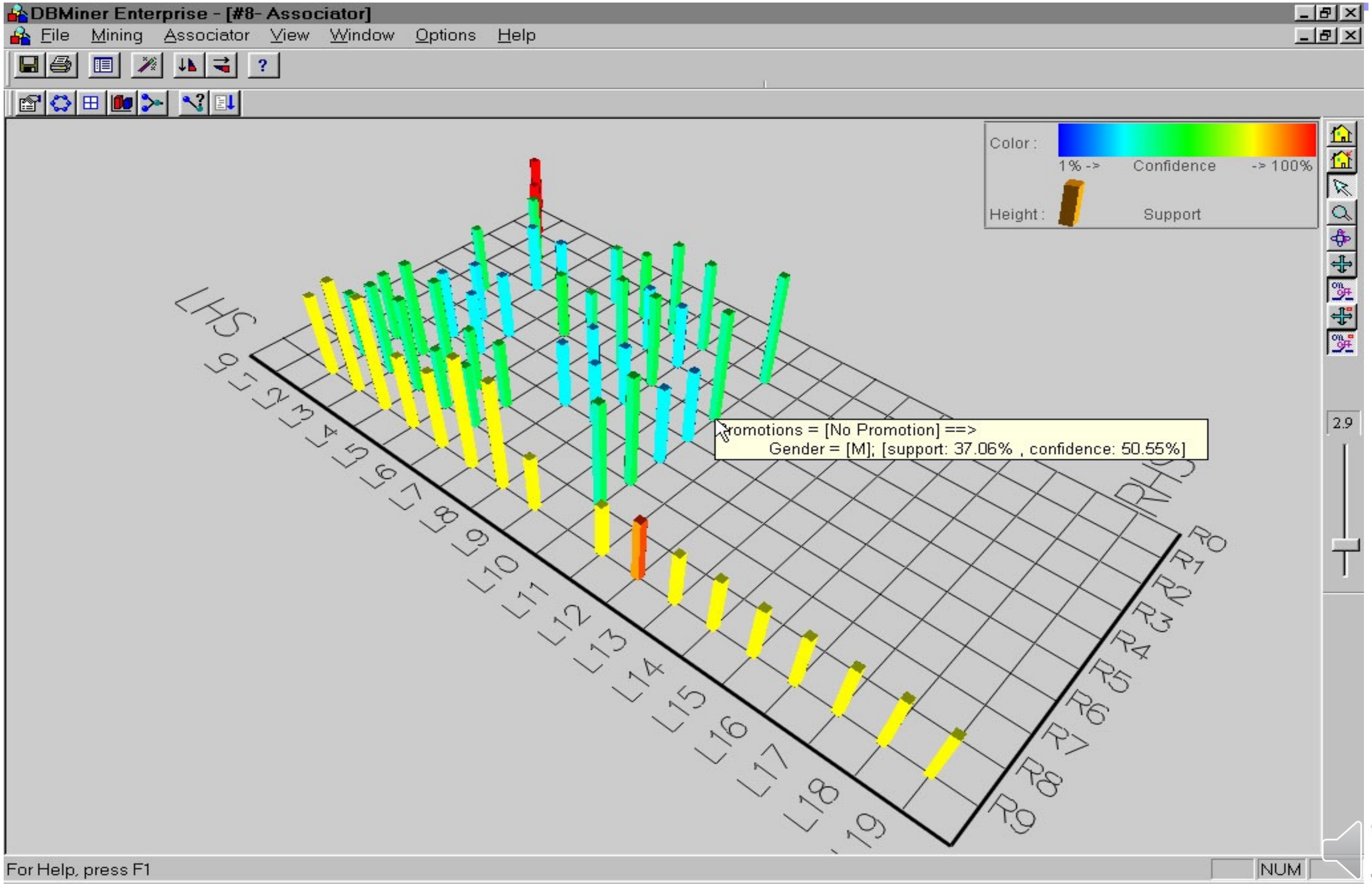
Table 5.5 The 3-itemsets in vertical data format.

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

CHARM: Mining by Exploring Vertical Data Format

- No need to scan a database to find the support of (k+1) itemsets
  - TID set of each k-itemset carries sufficient information including a support value
  - But, it is quite long and requires large space for intersection

Visualization of Association Rules: Plane Graph



DBMiner Enterprise - [#8- Associator]
File Mining Associator View Window Options Help
Color :
1% ->
Confidence
-> 100%
Height :
Support
LHS
RHS
L0 L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12 L13 L14 L15 L16 L17 L18 L19
R0 R1 R2 R3 R4 R5 R6 R7 R8 R9
2.9
For Help, press F1
NUM

Promotions = [No Promotion] ==>
Gender = [M]; [support: 37.06% , confidence: 50.55%]

Chapter 5: Mining Frequent Patterns, Association and Correlations

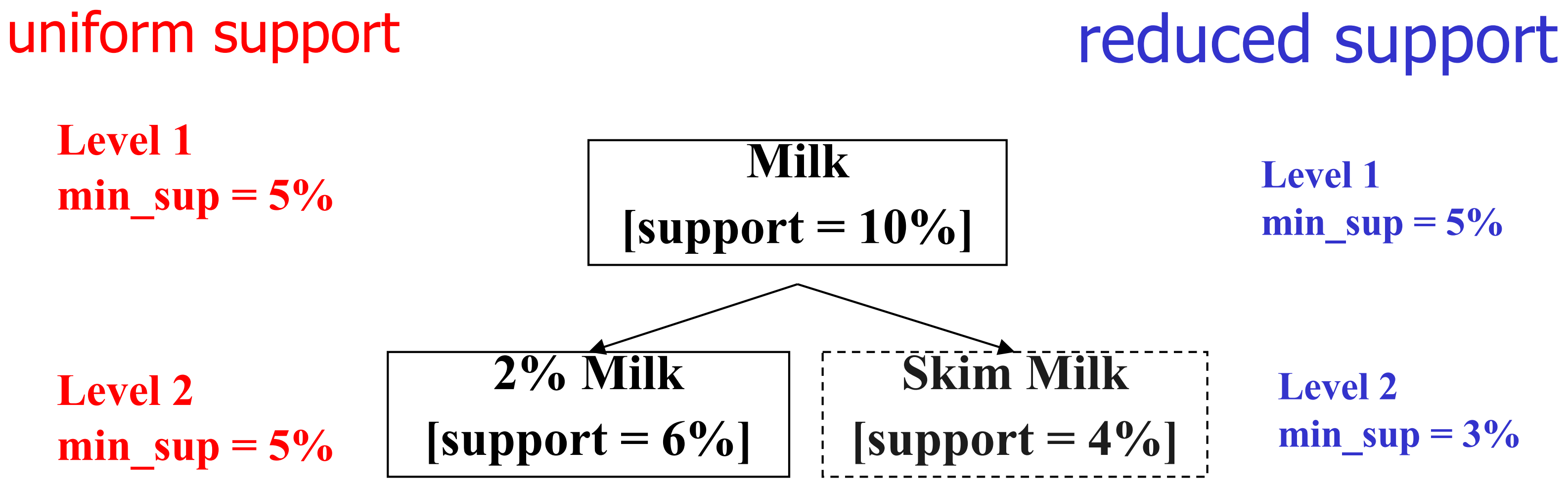
- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Mining Various Kinds of Association Rules

- Mining multilevel association
- Miming multidimensional association
- Mining quantitative association
- Mining interesting correlation patterns

Mining Multiple-Level Association Rules

- Items often form hierarchies
- Flexible support settings
  - Items at the lower level are expected to have lower support
- Exploration of *shared* multi-level mining (Agrawal & Srikant@VLDB'95, Han & Fu@VLDB'95)



Multi-level Association: Redundancy Filtering

- Some rules may be redundant due to "ancestor" relationships between items.
- Example
  - milk $\Rightarrow$ wheat bread [support = 8%, confidence = 70%]
  - 2% milk $\Rightarrow$ wheat bread [support = 2%, confidence = 72%]
- We say the first rule is an ancestor of the second rule.
- A (descendent) rule is redundant if
  - Its support is close to the *"expected" value*, based on the rule's ancestor
  - Its confidence is close to that of the rule's ancestor

Mining Multi-Dimensional Association

- Single-dimensional rules: (having a dimension or a predicate)

  buys(X, “milk”) $\Rightarrow$ buys(X, “bread”): milk $\Rightarrow$ bread

- Multi-dimensional rules: $\geq$ 2 dimensions or predicates
  - Inter-dimension assoc. rules (*no repeated predicates*)

    age(X,”19-25”) $\wedge$ occupation(X, “student”) $\Rightarrow$ buys(X, “coke”)
  - hybrid-dimension assoc. rules (*repeated predicates*)

    age(X,”19-25”) $\wedge$ buys(X, “popcorn”) $\Rightarrow$ buys(X, “coke”)

Attribute Types

- age(X,"19-25") ∧ occupation(X, "student") ⇒ buys(X, "coke")

  attributes

- Categorical Attributes
  - Finite number of possible values, no ordering among values
- Quantitative Attributes
  - Numeric, implicit ordering among values
  - Discretization and clustering approaches required (**why?**)

Mining Quantitative Associations

- Techniques can be categorized by how numerical attributes, such as age or salary are treated

1. Static discretization based on predefined concept hierarchies (data cube methods)
2. Dynamic discretization based on data distribution (quantitative rules, e.g., Agrawal & Srikant@SIGMOD96)
3. Clustering: Distance-based association (e.g., Yang & Miller@SIGMOD97)

Static Discretization of Quantitative Attributes

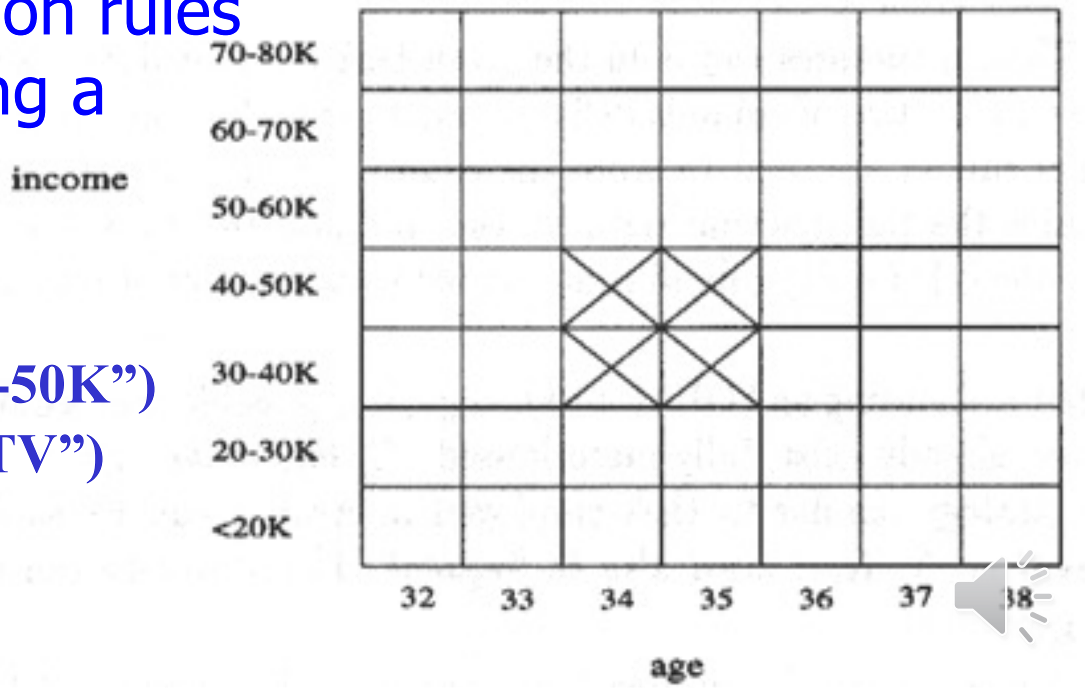
- Discretized prior to mining using a concept hierarchy
  - age(X,"19-25") $\wedge$ occupation(X, "student") $\Rightarrow$ buys(X, "coke")
  - Numeric values are replaced by ranges (as a categorical value)
- In a relational database, finding all frequent k-predicate sets will require *k* or *k*+1 table scans.

Quantitative Association Rules

- Proposed by Lent, Swami and Widom ICDE'97
- Numeric attributes are *dynamically* discretized
- 2-D quantitative association rules: $A_{quan1} \wedge A_{quan2} \Rightarrow A_{cat}$
  - The confidence is higher than threshold
  - The support is higher than threshold
- Cluster *adjacent* association rules to form general rules using a 2-D grid
- Example

age(X,"34-35") ∧ income(X,"30-50K")
⇒ buys(X,"high resolution TV")

Note: simplified!



Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Interestingness Measure: Correlations (Lift)

- *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%] is misleading
  - The overall % of students eating cereal is 75% > 66.7%.
- *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%] is more meaningful, although it has lower support and confidence
- Measure of dependent/correlated events: lift

$$lift = \frac{P(A \cup B)}{P(A)P(B)}$$

Contingency table

	Basketball	Not basketball	Sum (row)
Cereal	2000	1750	3750
Not cereal	1000	250	1250
Sum(col.)	3000	2000	5000

$$lift(B,C) = \frac{2000/5000}{3000/5000*3750/5000} = 0.89$$

$$lift(B,\neg C) = \frac{1000/5000}{3000/5000*1250/5000} = 1.33$$

Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Constraint-based (Query-Directed) Mining

- Finding all the patterns in a database autonomously? — unrealistic!
  - The patterns could be too many but not focused!
- Data mining should be an interactive process
  - User directs what to be mined
  - She/he uses a data mining query language (or a graphical user interface)
- Constraint-based mining
  - User flexibility: provides constraints on what to be mined
  - System optimization: explores such constraints for efficient mining—constraint-based mining

Constraints in Data Mining

- Knowledge type constraint:
  - classification, association, clustering etc.
- Data constraint — using SQL-like queries
  - find product pairs sold together in stores in Chicago in Dec.'21
- Dimension/level constraint
  - in relevance to region, price, brand, customer category
- Interestingness constraint
  - strong rules: min_support $\geq$ 3%, min_confidence $\geq$ 60%

Constrained Mining vs. Other Operations

- Constrained mining vs. constraint-based search
  - Both are aimed at reducing search space
    - Constrained mining: finding all patterns satisfying constraints
    - Constraint-based search: finding some (or one) answer in constraint-based search in AI

Constrained Mining vs. Other Operations

- Constrained mining vs. query processing in DBMS
  - Both are aimed at finding all answers
    - Query processing: finding tuples in a database
    - Constrained mining: discovering patterns hidden in a database
  - Constrained mining shares a similar philosophy as pushing selections deeply in query processing

Anti-Monotonicity in Constraint Pushing

- Anti-monotonicity on a constraint
  - *When an intemset S* ***violates*** *the constraint, so does any of its superset*
  - *sum(S.Price)* $\leq$ *v* is anti-monotone
  - *sum(S.Price)* $\geq$ *v* is not anti-monotone
- Example. C: range(S.profit) $\leq$ 15 is anti-monotone
  - Itemset *ab* violates 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

Monotonicity for Constraint Pushing

- Monotonicity on a constraint
  - *When an intemset S **satisfies** the constraint, so does any of its superset*
  - *sum(S.Price)* $\geq$ *v* is monotone
  - *min(S.Price)* $\leq$ *v* is monotone
- Example. C: range(S.profit) $\geq$ 15
  - Itemset *ab* satisfies C
  - So does every superset of *ab*

TDB (min_sup=2)

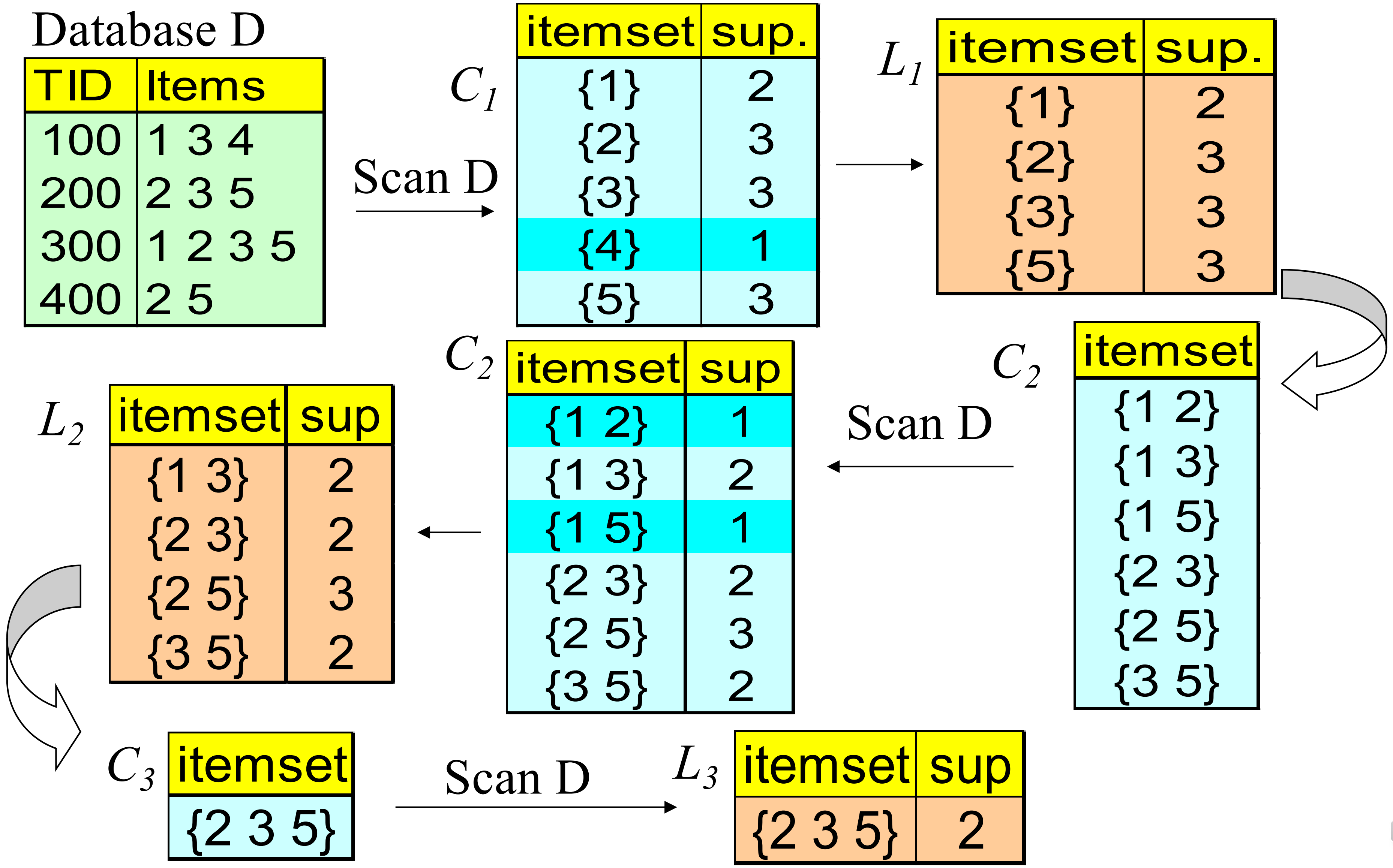
TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

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

Succinctness

- Succinctness on a constraint
  - When a set of items (say A_1) satisfies a constraint *C*, any set *S* satisfying *C* is 'simply computed' based on A_1 (In this case, *S* contains a subset belonging to A_1)
    - $min(S.Price) \leq v$ is succinct
    - $sum(S.Price) \geq v$ is not succinct
- Good thing
  - Without looking at the transaction database, whether an itemset *S* satisfies constraint C can be determined based on the selection of items
- Optimization: If *C* is succinct, *C* is pre-counting pushable

The Apriori Algorithm — Example



Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

L_3

itemset	sup
{2 3 5}	2

Naïve Algorithm: Apriori + Constraint

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
~~{5}~~	~~3~~

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

C_3

itemset
{2 3 5}

Scan D

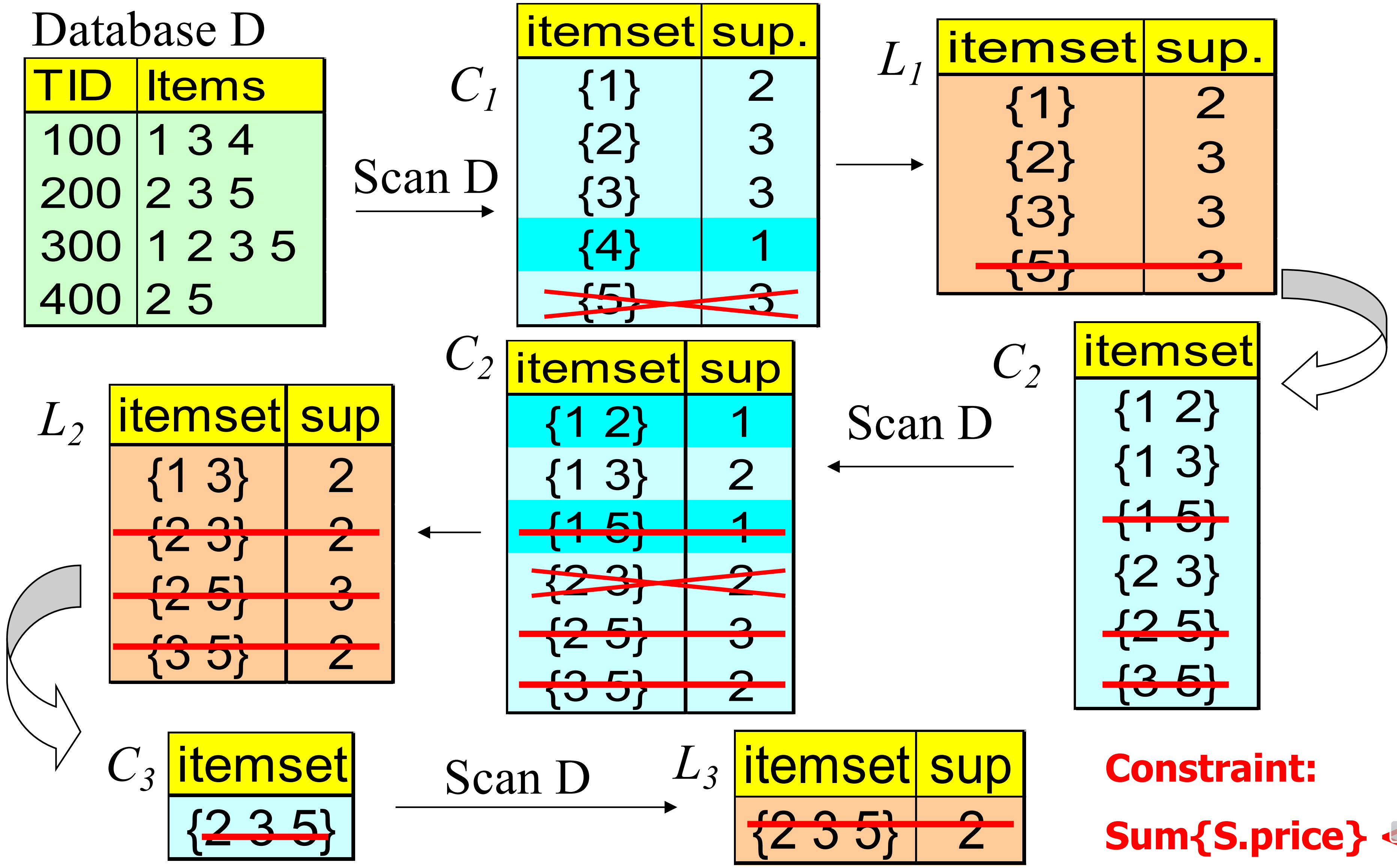
L_3

itemset	sup
~~{2 3 5}~~	~~2~~

Constraint:

Sum{S.price} < 5

The Constrained Apriori Algorithm: Push an Anti-monotone Constraint Deep



Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
~~{5}~~	~~3~~

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
~~{5}~~	~~3~~

C_2

itemset
{1 2}
{1 3}
~~{1 5}~~
{2 3}
~~{2 5}~~
~~{3 5}~~

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
~~{1 5}~~	~~1~~
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

C_3

itemset
~~{2 3 5}~~

Scan D

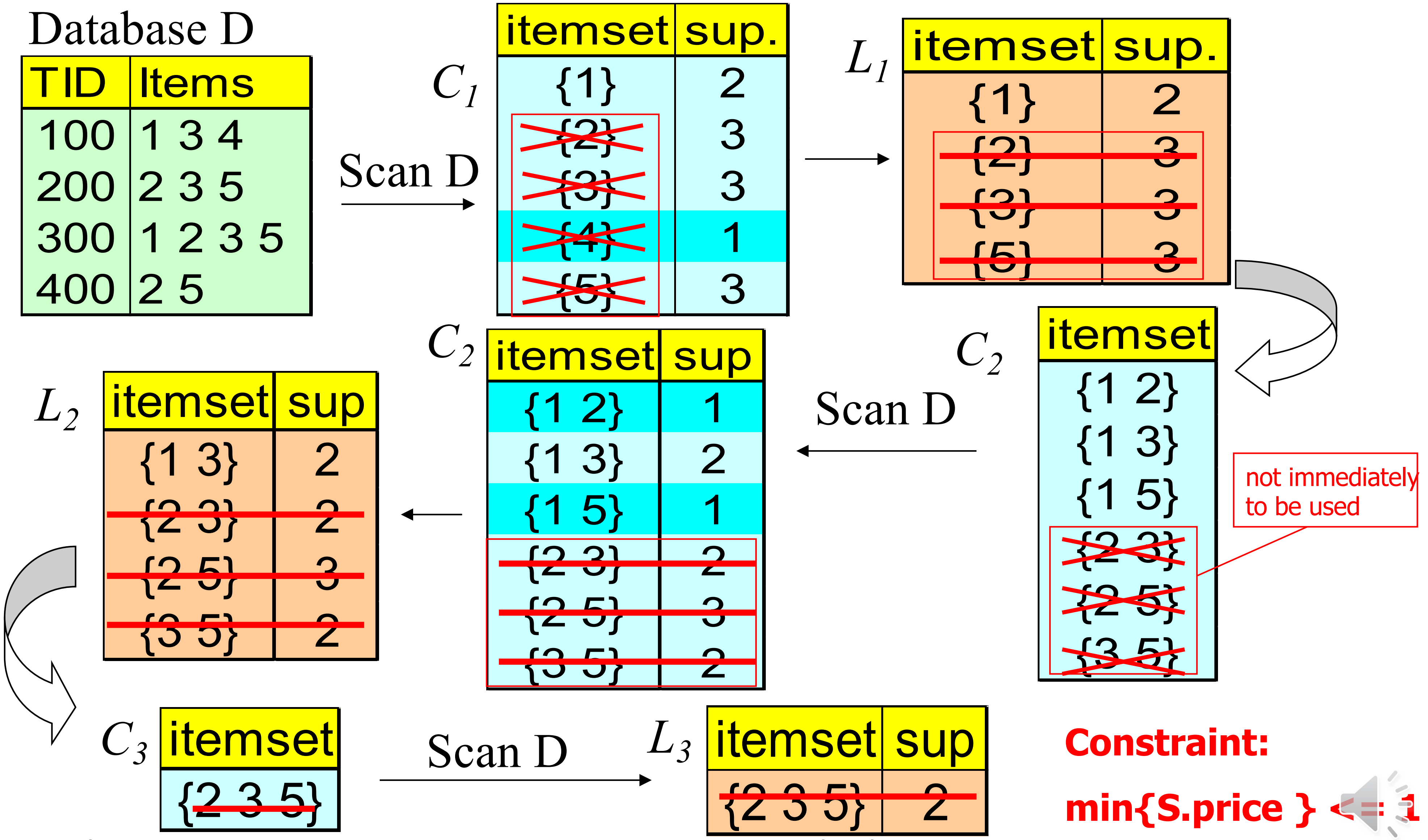
L_3

itemset	sup
~~{2 3 5}~~	~~2~~

Constraint:

Sum{S.price} < 5

The Constrained Apriori Algorithm: Push a Succinct Constraint Deep



Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

C_1

itemset	sup.
{1}	2
~~{2}~~	3
~~{3}~~	3
~~{4}~~	1
~~{5}~~	3

L_1

itemset	sup.
{1}	2
~~{2}~~	~~3~~
~~{3}~~	~~3~~
~~{5}~~	~~3~~

C_2

itemset
{1 2}
{1 3}
{1 5}
~~{2 3}~~
~~{2 5}~~
~~{3 5}~~

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

C_3

itemset
~~{2 3 5}~~

L_3

itemset	sup
~~{2 3 5}~~	~~2~~

Constraint:

min{S.price } <= 1

Converting “Tough” Constraints

- 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*
    - 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

Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Frequent-Pattern Mining: Summary

- Frequent pattern mining—an important task in data mining
- Scalable frequent pattern mining methods
  - Apriori (Candidate generation & test)
  - Projection-based (FPgrowth, CLOSET+, ...)
  - Vertical format approach (CHARM, ...)
- Mining a variety of rules and interesting patterns
- Constraint-based mining
- Extensions and applications