Mining Association Rules in Large Databases

Mining Association Rules in Large Databases

- Association rule mining
- Mining single-dimensional Boolean association rules from transactional databases
- Mining multilevel association rules from transactional databases
- Mining multidimensional association rules from transactional databases and data warehouse
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Association rule mining

- Proposed by Agrawal et al in 1993.
- It is an important data mining model studied extensively by the database and data mining community.
- Assume all data are categorical.
- No good algorithm for numeric data.
- Initially used for Market Basket Analysis to find how items purchased by customers are related.

Bread $\rightarrow$ Milk [sup = 5%, conf = 100%]

The model: data

- $I = \{i_1, i_2, ..., i_m\}$: a set of *items*.
- Transaction t :
  - t a set of items, and $t \subseteq I$.
- Transaction Database T: a set of transactions $T = \{t_1, t_2, ..., t_n\}$.

Transaction data: supermarket data

- Market basket transactions:

  t1: {bread, cheese, milk}
  t2: {apple, eggs, salt, yogurt}
  … …
  tn: {biscuit, eggs, milk}
- Concepts:
  - An *item*: an item/article in a basket
  - *I*: the set of all items sold in the store
  - A *transaction*: items purchased in a basket; it may have TID (transaction ID)
  - A *transactional dataset*: A set of transactions

Transaction data: a set of documents

- **A text document data set. Each document is treated as a "bag" of keywords**

doc1:	Student, Teach, School
doc2:	Student, School
doc3:	Teach, School, City, Game
doc4:	Baseball, Basketball
doc5:	Basketball, Player, Spectator
doc6:	Baseball, Coach, Game, Team
doc7:	Basketball, Team, City, Game

What Is Association Mining?

- Association rule mining:
  - Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.
- Applications:
  - Basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, classification, etc.
- Examples.
  - Rule form: "Body $\rightarrow$ Head [support, confidence]".
  - buys(x, "diapers") $\rightarrow$ buys(x, "beers") [0.5%, 60%]
  - major(x, "CS") ^ takes(x, "DB") $\rightarrow$ grade(x, "A") [1%, 75%]

Association Rule: Basic Concepts

- Given: (1) database of transactions, (2) each transaction is a list of items (purchased by a customer in a visit)
- Find: all rules that correlate the presence of one set of items with that of another set of items
  - E.g., *98% of people who purchase tires and auto accessories also get automotive services done*
- Applications
  - $* \Rightarrow$ *Maintenance Agreement* (What the store should do to boost Maintenance Agreement sales)
  - *Home Electronics* $\Rightarrow *$ (What other products should the store stocks up?)
  - Attached mailing in direct marketing

Association Rules

Based on the types of values, the association rules can be classified into two categories:

- Boolean Association Rules and
- Quantitative Association Rules

Example:

Boolean Association Rule

Keyboard $\Rightarrow$ Mouse [support = 6%, confidence = 70%]

Quantitative Association Rule

(***Age*** = ***26***...30) $\Rightarrow$ (Cars =1, 2) [Support 3%, confidence = 36%]

Association Rule Mining: A Road Map

- Boolean vs. quantitative associations
- (Based on the types of values handled)
  - buys(x, "SQLServer") ^ buys(x, "DMBook") $\rightarrow$ buys(x, "DBMiner") [0.2%, 60%]
  - age(x, "30..39") ^ income(x, "42..48K") $\rightarrow$ buys(x, "PC") [1%, 75%]
- Single dimension vs. multiple dimensional associations
- Single level vs. multiple-level analysis
  - What brands of beers are associated with what brands of diapers?

The model: rules

- A transaction t contains X, a set of items (itemset) in I, if $X \subseteq t$.
- An association rule is an implication of the form:

$$X \to Y, \text{ where } X, Y \subset I, \text{ and } X \cap Y = \emptyset$$

- An itemset is a set of items.
  - E.g., X = {milk, bread, cereal} is an itemset.
- A *k*-itemset is an itemset with *k* items.
  - E.g., {milk, bread, cereal} is a 3-itemset

Minimum Support Threshold

The support of an association pattern is the percentage of task-relevant data transactions for which the pattern is true.

IF A ⇒ B

$$support\,(A \Rightarrow B) = \frac{\#\_tuples\_containing\_both\_A\_and\_B}{total\_\#\_of\_tuples}$$

Minimum Confidence Threshold

Confidence is defined as the measure of certainty or trustworthiness associated with each discovered pattern.

IF A ⇒ B

$$confidence(A \Rightarrow B) = \frac{\#\_tuples\_containing\_both\_A\_and\_B}{\#\_tuples\_containing\_A}$$

Rule strength measures

- Support: The rule holds with support *sup* in *T* (the transaction data set) if sup% of transactions contain $X \cup Y$.
  - $sup = \Pr(X \cup Y)$.
- Confidence: The rule holds in *T* with confidence *conf* if *conf*% of transactions that contain *X* also contain *Y*.
  - $conf = \Pr(Y \mid X)$
- An association rule is a pattern that states when *X* occurs, *Y* occurs with certain probability.

- Support count: The support count of an itemset X, denoted by $X.count$, in a data set T is the number of transactions in T that contain X. Assume T has n transactions.
- Then,

$$support = \frac{(X \cup Y).count}{n}$$

$$confidence = \frac{(X \cup Y).count}{X.count}$$

Goal and key features

- **Goal:** Find all rules that satisfy the user-specified *minimum support* (minsup) and *minimum confidence* (minconf).
- **Key Features**
  - Completeness: find all rules.
  - No target item(s) on the right-hand-side
  - Mining with data on hard disk (not in memory)

An example

- Transaction data
- Assume:

  minsup = 30%

  minconf = 80%
- An example *frequent itemset*:

  {Chicken, Clothes, Milk} [sup = 3/7]
- Association rules from the itemset:

  Clothes → Milk, Chicken [sup = 3/7, conf = 3/3]

  … …

  Clothes, Chicken → Milk, [sup = 3/7, conf = 3/3]

t1:	Beef, Chicken, Milk
t2:	Beef, Cheese
t3:	Cheese, Boots
t4:	Beef, Chicken, Cheese
t5:	Beef, Chicken, Clothes, Cheese, Milk
t6:	Chicken, Clothes, Milk
t7:	Chicken, Milk, Clothes

Transaction data representation

- A simplistic view of shopping baskets,
- Some important information not considered. E.g,
  - the quantity of each item purchased and
  - the price paid.

Itemset

- A set of items is referred to as itemset.
- An itemset containing *k items is called* ***k-itemset*.**
- An itemset can also be seen as a conjunction of items (or a predicate)

Frequent Itemsets

- Suppose *min_sup is the minimum support* threshold.
- An itemset satisfies minimum support if the occurrence frequency of the itemset is greater than or equal to *min_sup*.
- If an itemset satisfies minimum support, then it is a frequent itemset.

Strong Rules

Rules that satisfy both a minimum support threshold and a minimum confidence threshold are called strong.

Association Rule Mining

- Find all frequent itemsets
- Generate strong association rules from the frequent itemsets

Many mining algorithms

- There are a large number of them!!
- They use different strategies and data structures.
- Their resulting sets of rules are all the same.
  - Given a transaction data set *T*, and a minimum support and a minimum confident, the set of association rules existing in *T* is uniquely determined.
- Any algorithm should find the same set of rules although their computational efficiencies and memory requirements may be different.
- We study only one: the Apriori Algorithm

The Apriori Algorithm: Basics

The Apriori Algorithm is an influential algorithm for mining frequent itemsets for boolean association rules.

Key Concepts :

- Frequent Itemsets: The sets of item which has minimum support (denoted by L_i for i^{th}-Itemset).
- Apriori Property: Any subset of frequent itemset must be frequent.
- Join Operation: To find L_k, a set of candidate k-itemsets is generated by joining L_{k-1} with itself.

Apriori Property

Reducing the search space to avoid finding of each L_k requires one full scan of the database

If an itemset *I does not satisfy the minimum* support threshold, *min_sup, the I is not frequent*, that is, P (*I*) < *min_sup*.

If an item A is added to the itemset *I, then the* resulting itemset (i.e.,I∪A) cannot occur more frequently than *I. Therefore, I ∪ A is not frequent* either, that is, P (I ∪A) < *min_sup*.

The Apriori Algorithm in a Nutshell

Find the *frequent itemsets:* the sets of items that have minimum support

–A subset of a frequent itemset must also be a frequent itemset

i.e., if *{AB} is a frequent itemset, both {A} and {B} should be a frequent itemset*

–Iteratively find frequent itemsets with cardinality from 1 to *k (k-itemset)*

– Use the frequent itemsets to generate association rules.

The Apriori Algorithm: Example

TID	List of Items
T100	I1, I2, I5
T100	I2, I4
T100	I2, I3
T100	I1, I2, I4
T100	I1, I3
T100	I2, I3
T100	I1, I3
T100	I1, I2 ,I3, I5
T100	I1, I2, I3

Consider a database, D , consisting of 9 transactions.

Suppose min. support count required is 2 (i.e. min_sup = 2/9 = 22 %)

Let minimum confidence required is 70%.

We have to first find out the frequent itemset using Apriori algorithm.

Then, Association rules will be generated using min. support & min. confidence.

Step 1: Generating 1-itemset Frequent Pattern

Scan D for count of each candidate

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

C_1

Compare candidate support count with minimum support count

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

L_1

The set of frequent 1-itemsets, L_1, consists of the candidate 1-itemsets satisfying minimum support.

In the first iteration of the algorithm, each item is a member of the set of candidate.

Step 2: Generating 2-itemset Frequent Pattern

Generate C_2 candidates from L_1

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

C_2

Scan D for count of each candidate

Itemset	Sup. Count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

C_2

Compare candidate support count with minimum support count

Itemset	Sup Count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

L_2

Step 3: Generating 3-itemset Frequent Pattern

Scan D for count of each candidate

Itemset
{I1, I2, I3}
{I1, I2, I5}

C_3

Scan D for count of each candidate

Itemset	Sup. Count
{I1, I2, I3}	2
{I1, I2, I5}	2

C_3

Compare candidate support count with min support count

Itemset	Sup Count
{I1, I2, I3}	2
{I1, I2, I5}	2

L_3

The generation of the set of candidate 3-itemsets, C_3, involves use of the Apriori Property.

In order to find C_3, we compute L_2 *Join* L_2.

C_3= L_2 *Join* L_2 = {{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}}.

Now, Join step is complete and Prune step will be used to reduce the size of C_3. Prune step helps to avoid heavy computation due to large C_k.

Step 3: Generating 3-itemset Frequent Pattern

Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that four latter candidates cannot possibly be frequent. How ?

For example , lets take {I1, I2, I3}.The 2-item subsets of it are {I1, I2}, {I1, I3} & {I2, I3}. Since all 2-item subsets of {I1, I2, I3} are members of L2, We will keep {I1, I2, I3} in C3.

- Lets take another example of {I2, I3, I5}which shows how the pruning is performed. The 2-item subsets are {I2, I3}, {I2, I5} & {I3,I5}.
- BUT, {I3, I5} is not a member of L2 and hence it is not frequent violating Apriori Property. Thus We will have to remove {I2, I3, I5} from C3.
- Therefore, C3= {{I1, I2, I3}, {I1, I2, I5}} after checking for all members of result of Join operation for Pruning.
- Now, the transactions in D are scanned in order to determine L3, consisting of those candidates 3-itemsets in C3having minimum support.

Step 4: Generating 4-itemset Frequent Pattern

The algorithm uses L3 Join L3 to generate a candidate set of 4-itemsets, C_4. Although the join results in {{I1, I2, I3, I5}}, this itemset is pruned since its subset {{I2, I3, I5}} is not frequent.

- Thus, $C_4 = \phi$, and algorithm terminates, having found all of the frequent items. This completes our Apriori Algorithm.
- What's Next ?

These frequent itemsets will be used to generate strong association rules (where strong association rules satisfy both minimum support & minimum confidence).

Step 5: Generating Association Rules from Frequent Itemsets

Procedure:

- For each frequent itemset " *l* ", generate all nonempty subsets of *l*.
- For every nonempty subset s *of* l, *output the rule*

"s ->(l-s)" if

support_count(*l*) / support_count(s) >= min_conf
where min_conf is minimum confidence threshold.

Back To Example

We had L = {{I1}, {I2}, {I3}, {I4}, {I5}, {I1,I2}, {I1,I3}, {I1,I5}, {I2,I3}, {I2,I4}, {I2,I5}, {I1,I2,I3}, {I1,I2,I5}}.

- Lets take ***l*** = {I1,I2,I5}
- Its all nonempty subsets are {I1,I2}, {I1,I5}, {I2,I5}, {I1}, {I2}, {I5}

Step 5:Generating Association Rules from Frequent Itemsets

Let minimum confidence threshold is , say 70%.

The resulting association rules are shown below, each listed with its confidence.

– R1: I1 ^ I2 ->I5

Confidence = sc{I1,I2,I5}/sc{I1,I2} = 2/4 = 50%

R1 is Rejected.

– R2: I1 ^ I5 ->I2

Confidence = sc{I1,I2,I5}/sc{I1,I5} = 2/2 = 100%

R2 is Selected.

– R3: I2 ^ I5 ->I1

Confidence = sc{I1,I2,I5}/sc{I2,I5} = 2/2 = 100%

R3 is Selected.

Step 5: Generating Association Rules from Frequent Itemsets

–R4: I1 -> I2 ^ I5

Confidence = sc{I1,I2,I5}/sc{I1} = 2/6 = 33%

R4 is Rejected.

–R5: I2 -> I1 ^ I5

Confidence = sc{I1,I2,I5}/{I2} = 2/7 = 29%

R5 is Rejected.

–R6: I5-> I1 ^ I2

Confidence = sc{I1,I2,I5}/ {I5} = 2/2 = 100%

R6 is Selected.

In this way, We have found three strong association rules.

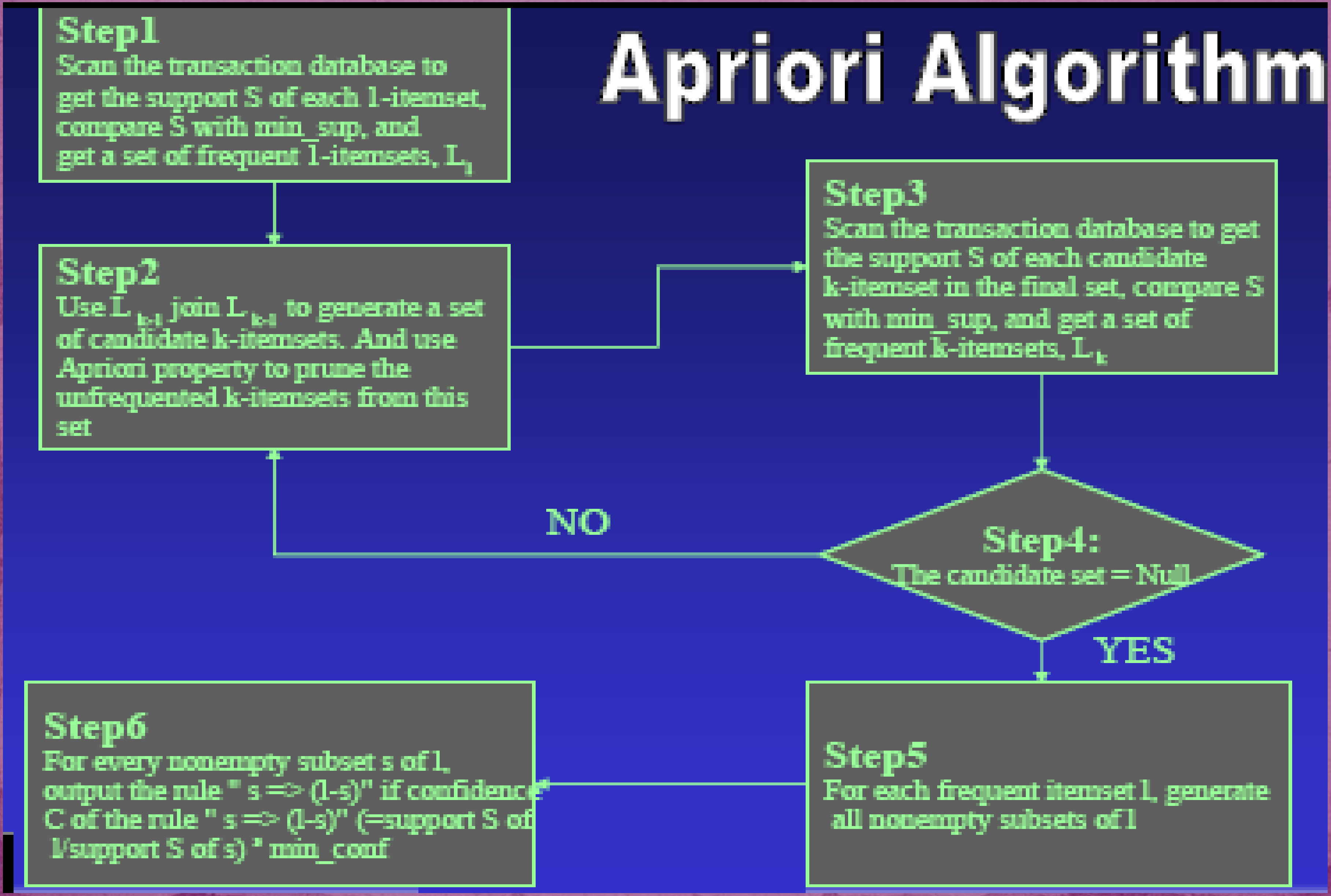
Example – Finding frequent itemsets

Dataset T

minsup=0.5

TID	Items
T100	1, 3, 4
T200	2, 3, 5
T300	1, 2, 3, 5
T400	2, 5

Apriori Algorithm



The Apriori Algorithm

- Join Step: C_k is generated by joining L_{k-1} with itself
- Prune Step: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset
- Pseudo-code:

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

```
L_1 = {frequent items};
for (k = 1; L_k !=∅; k++) do begin
    C_k+1 = candidates generated from L_k;
   for each transaction t in database do
          increment the count of all candidates in C_k+1
        that are contained in t
   L_k+1  = candidates in C_k+1 with min_support
   end
return ∪_k L_k;
```

On Apriori Algorithm

Seems to be very expensive

- Level-wise search
- K = the size of the largest itemset
- It makes at most K passes over data
- In practice, K is bounded (10).
- The algorithm is very fast. Under some conditions, all rules can be found in linear time.
- Scale up to large data sets

Apriori Advantages/Disadvantages

- Advantages
  - Uses large itemset property
  - Easily parallelized
  - Easy to implement
- Disadvantages
  - Assumes transaction database is memory resident.
  - Requires many database scans.

More on association rule mining

- Clearly the space of all association rules is exponential, $O(2^m)$, where m is the number of items in *I*.
- The mining exploits sparseness of data, and high minimum support and high minimum confidence values.
- Still, it always produces a huge number of rules, thousands, tens of thousands, millions, …

Example 2: **Problem data**

An example with a transactional data D contents a list of 5 transactions in a supermarket.

TID	List of items (item_IDs)
1	Beer(I1), Diaper(I2), Baby Powder(I3), Bread(I4), Umbrella(I5)
2	Diaper(I2), Baby Powder(I3)
3	Beer(I1), Diaper(I2), Milk(I6)
4	Diaper(I2), Beer(I1), Detergent(I7)
5	Beer(I1), Milk(I6), Coca Cola (I8)

Solution Procedure

Step 1 *min_sup = 40% (2/5)*

C1 → **L1**

Item_ID	Item	Support
I1	Beer	4/5
I2	Diaper	4/5
I3	Baby powder	2/5
~~I4~~	~~Bread~~	~~1/5~~
~~I5~~	~~Umbrella~~	~~1/5~~
I6	Milk	2/5
~~I7~~	~~Detergent~~	~~1/5~~
~~I8~~	~~Coca-cola~~	~~1/5~~

Item_ID	Item	Support
I1	Beer	4/5
I2	Diaper	4/5
I3	Baby	2/5
I6	Milk	2/5

$$support(A \Rightarrow B) = \frac{\#\_tuples\_containing\_both\_A\_and\_B}{total\_\#\_of\_tuples}$$

Solution Procedure

Step 2

C_2

Item ID	Item	Support
{I1, I2}	Beer, Diaper	3/5
~~{I1, I3}~~	~~Beer, Baby powder~~	~~1/5~~
{I1, I6}	Beer, Milk	2/5
{I2, I3}	Diaper, Baby powder	2/5
~~{I2, I6}~~	~~Diaper, Milk~~	~~1/5~~
~~{I3, I6}~~	~~Baby powder, Milk~~	~~0~~

Step 3

L_2

Item_ID	Item	Support
{I1, I2}	Beer, Diaper	3/5
{I1, I6}	Beer, Milk	2/5
{I2, I3}	Diaper, Baby powder	2/5

Solution Procedure

Step 4: L_2 is not Null, so repeat Step2

Item_ID	Item
{I1, I2, I3}	Beer, Diaper, Baby powder
{I1, I2, I6}	Beer, Diaper, Milk
(I1, I3, I6}	Beer, Baby powder, Milk
(I2, I3, I6}	Diaper, Baby powder, Milk

C3 =Null

Solution Procedure

Step 5

min_sup=40% min_conf=70%

Item_ID	Item	Support(A B)	Support A	Confidence
I1 I2	Beer Diaper	60%	80%	75%
~~I1 I6~~	~~Beer Milk~~	~~40%~~	~~80%~~	~~50%~~
~~I2 I3~~	~~Diaper Baby powder~~	~~40%~~	~~80%~~	~~50%~~
I2 I1	Diaper Beer	60%	80%	75%
I6 I1	Milk Beer	40%	40%	100%
I3 I2	Baby powder Diaper	40%	40%	100%

$$support(A \Rightarrow B) = \frac{\#\_tuples\_containing\_both\_A\_and\_B}{total\_\#\_of\_tuples}$$

$$confidence(A \Rightarrow B) = \frac{\#\_tuples\_containing\_both\_A\_and\_B}{\#\_tuples\_containing\_A}$$

Solution Procedure

TID	List of items (item_IDs)
1	Beer(I1), Diaper(I2), Baby Powder(I3), Bread(I4), Umbrella(I5)
2	Diaper(I2), Baby Powder(I3)
3	Beer(I1), Diaper(I2), Milk(I6)
4	Diaper(I2), Beer(I1), Detergent(I7)
5	Beer(I1), Milk(I6), Coca Cola (I8)

Item_ID	Item	Support(A B)	Support A	Confidence
I1 I2	Beer Diaper	60%	80%	75%
~~I1 I6~~	~~Beer Milk~~	~~40%~~	~~80%~~	~~50%~~
~~I2 I3~~	~~Diaper Baby powder~~	~~40%~~	~~80%~~	~~50%~~
I2 I1	Diaper Beer	60%	80%	75%
I6 I1	Milk Beer	40%	40%	100%
I3 I2	Baby powder Diaper	40%	40%	100%

Solution Procedure

Step 6

min_sup = 40% min_conf = 70%

Strong rules		Support	Confidence
I1=> I2	Beer=> Diaper	60%	75%
I2=> I1	Diaper=> Beer	60%	75%
I6 => I1	Milk=> Beer	40%	100%
I3 => I2	Baby powder=> Diaper	40%	100%

Results

- Some rules are believable, like Baby powder ⇒Diaper.
- Some rules need additional analysis, like Milk ⇒Beer.
- Some rules are unbelievable, like Diaper ⇒ Beer.

Note this example could contain unreal results because its small data.

Mining Association Rules in Large Databases

- Association rule mining
- Mining single-dimensional Boolean association rules from transactional databases
- Mining multilevel association rules from transactional databases
- Mining multidimensional association rules from transactional databases and data warehouse
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Methods to Improve Apriori's Efficiency

- Hash-based itemset counting: A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
- Transaction reduction: A transaction that does not contain any frequent k-itemset is useless in subsequent scans
- Partitioning: Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
- Sampling: mining on a subset of given data, lower support threshold + a method to determine the completeness
- Dynamic itemset counting: add new candidate itemsets only when all of their subsets are estimated to be frequent

Is Apriori Fast Enough? — Performance Bottlenecks

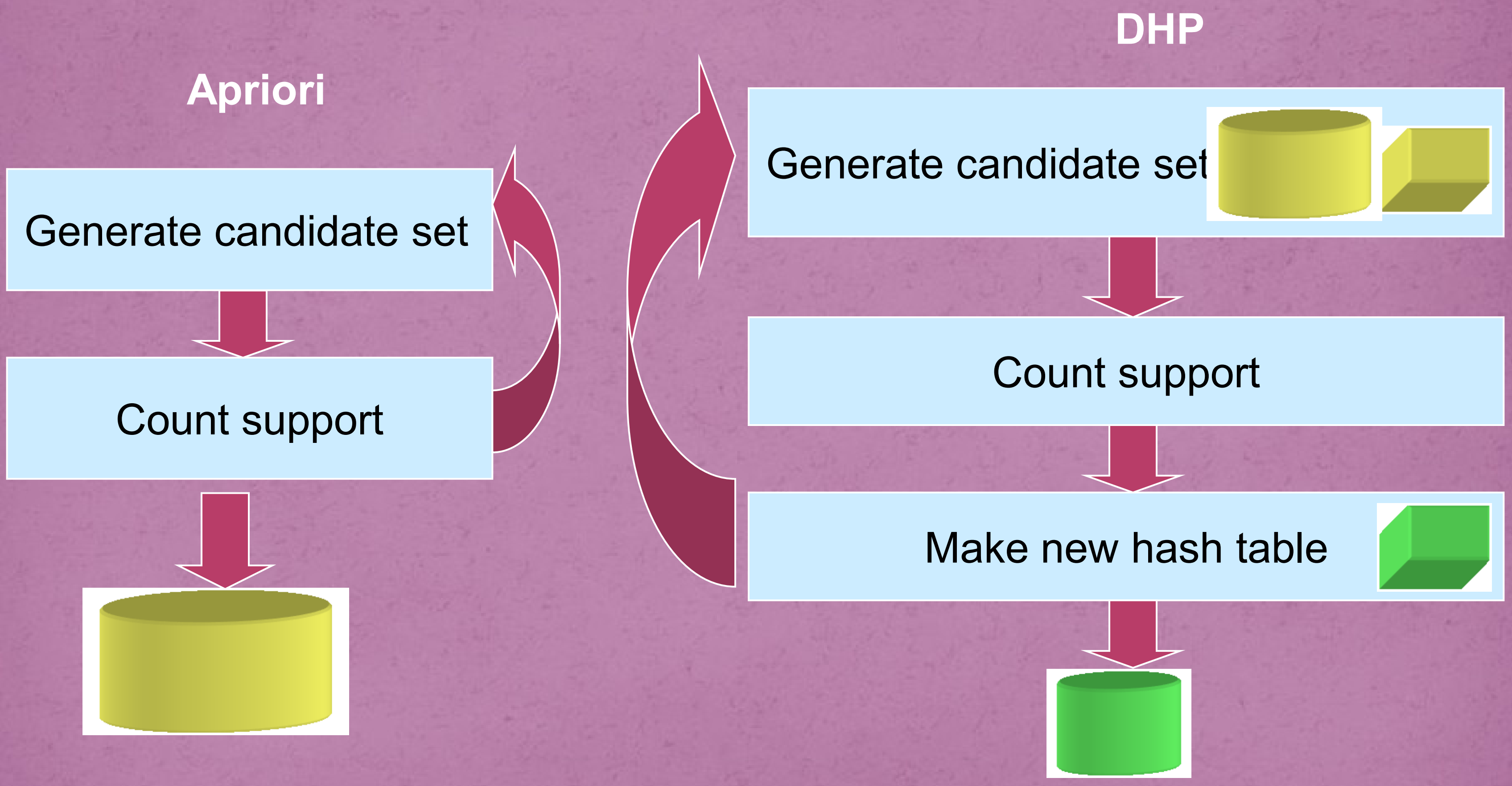
- The core of the Apriori algorithm:
  - Use frequent (k – 1)-itemsets to generate candidate frequent k-itemsets
  - Use database scan and pattern matching to collect counts for the candidate itemsets
- The bottleneck of *Apriori*: candidate generation
  - Huge candidate sets:
  - Multiple scans of database:

Speeding up Association rules : Hash Based Technique

- Reduce the number of candidates:

  A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent.
- While scanning to generate 1-itemsets , we can generate 2-itemsets for each transaction and hash(map) them into different buckets of a hash table structure and increase the corresponding bucket counts. A 2-itemset whose corresponding bucket count is below the support threshold can not be frequent.

How does it look like?



Hash Table Construction

- Consider two item sets, all items are numbered as i_1, i_2, ...i_n. For any pair (x, y), has according to
  - Hash function bucket #=

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

- 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.

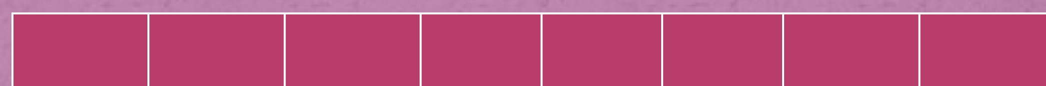
Create hash table H_2 using hash function $h(x, y) = ((order\ of\ x) \times 10 + (order\ of\ y))\ mod\ 7$ →

H_2

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

How to trim candidate itemsets

- In k-iteration, hash all "appearing" k+1 itemsets in a hashtable, count all the occurrences of an itemset in the correspondent bucket.



- In k+1 iteration, examine each of the candidate itemset to see if its correspondent bucket value is above the support (necessary condition)

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 (2)

- 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}

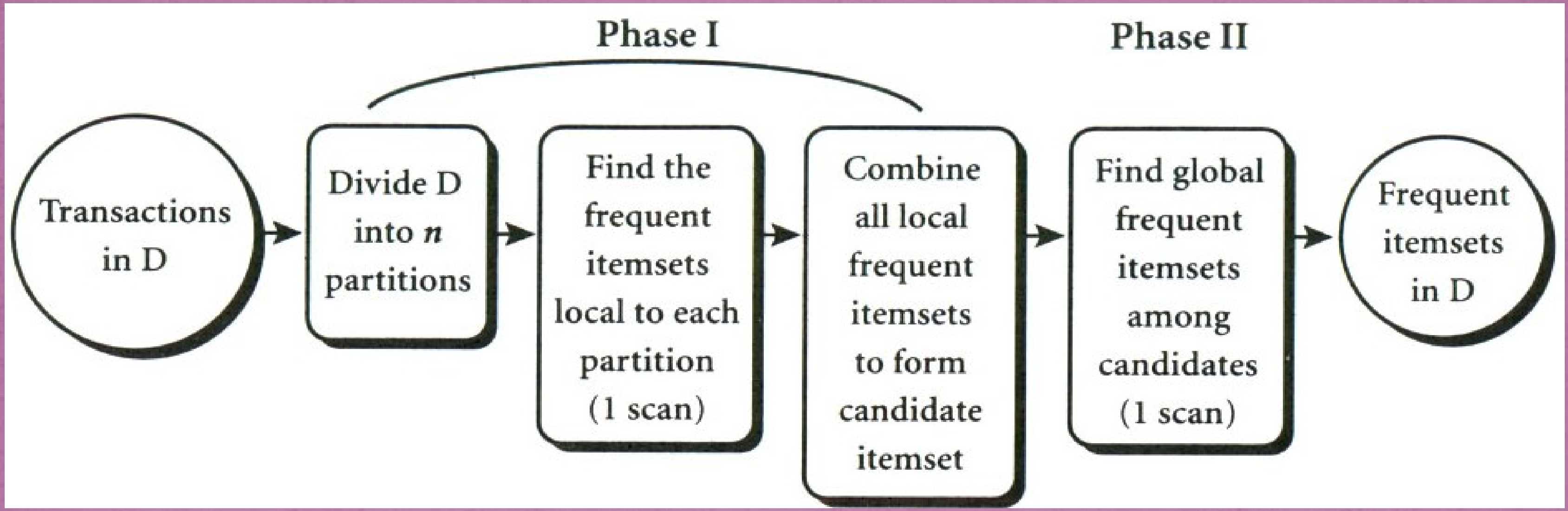
Mining Frequent Patterns Without Candidate Generation

- Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure
  - highly condensed, but complete for frequent pattern mining
  - avoid costly database scans
- Develop an efficient, FP-tree-based frequent pattern mining method
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones
  - Avoid candidate generation: sub-database test only!

Transaction Reduction

- A transaction that does not contain any frequent k-item sets cannot contain any frequent (k+1) item sets.
- Such transactions can be marked or remoned from further considerations.

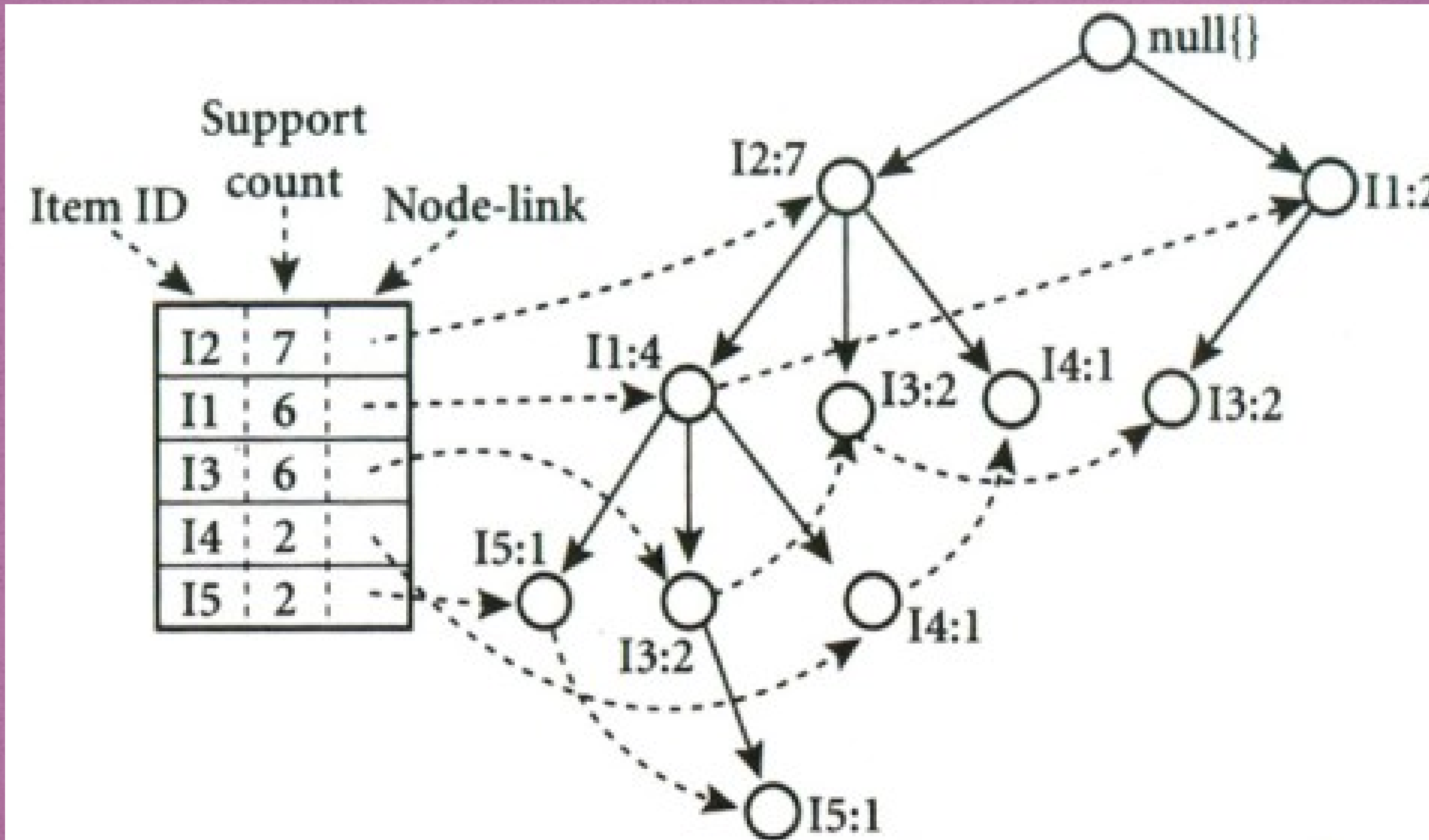
Partitioning



FP-growth Algorithm

- Use a compressed representation of the database using an FP-tree
- Once an FP-tree has been constructed, it uses a recursive divide-and-conquer approach to mine the frequent itemsets

FP-growth Algorithm(Cont…)



FP-growth Algorithm(Cont...)

Item	Conditional Pattern Base	Conditional FP-tree	Frequent Patterns Generated
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}

Mining Frequent Itemsets using Vertical Data Format

- ECLAT(Equivalence CLASS Transformation Algorithm) : Developed by Zaki 2000.

- For each item, store a list of transaction ids (tids); vertical data layout

Instead of {TID: itemset} it stores {item: TID_set}

ECLAT(Equivalence CLASS Transformation Algorithm)

Horizontal Data Layout

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

Vertical Data Layout

A	B	C	D	E
1	1	2	2	1
4	2	3	4	3
5	5	4	5	6
6	7	8	9	
7	8	9		
8	10			
9				

TID-list

ECLAT(Equivalence CLASS Transformation Algorithm)

- Determine support of any k-itemset by intersecting tid-lists of two of its (k-1) subsets.

A
1
4
5
6
7
8
9

$\wedge$

B
1
2
5
7
8
10

$\rightarrow$

AB
1
5
7
8

- 3 traversal approaches:
  - top-down, bottom-up and hybrid
- Advantage: very fast support counting
- Disadvantage: intermediate tid-lists may become too large for memory

ECLAT Example

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}

ECLAT Example

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}

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

Complexity of Association Mining

- Choice of minimum support threshold
  - lowering support threshold results in more frequent itemsets
  - this may increase number of candidates and max length of frequent itemsets
- Dimensionality (number of items) of the data set
  - more space is needed to store support count of each item
  - if number of frequent items also increases, both computation and I/O costs may also increase
- Size of database
  - since Apriori makes multiple passes, run time of algorithm may increase with number of transactions
- Average transaction width
  - transaction width increases with denser data sets
  - This may increase max length of frequent itemsets and traversals of hash tree (number of subsets in a transaction increases with its width)

Mining Association Rules in Large Databases

- Association rule mining
- Mining single-dimensional Boolean association rules from transactional databases
- Mining multilevel association rules from transactional databases
- Mining multidimensional association rules from transactional databases and data warehouse
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

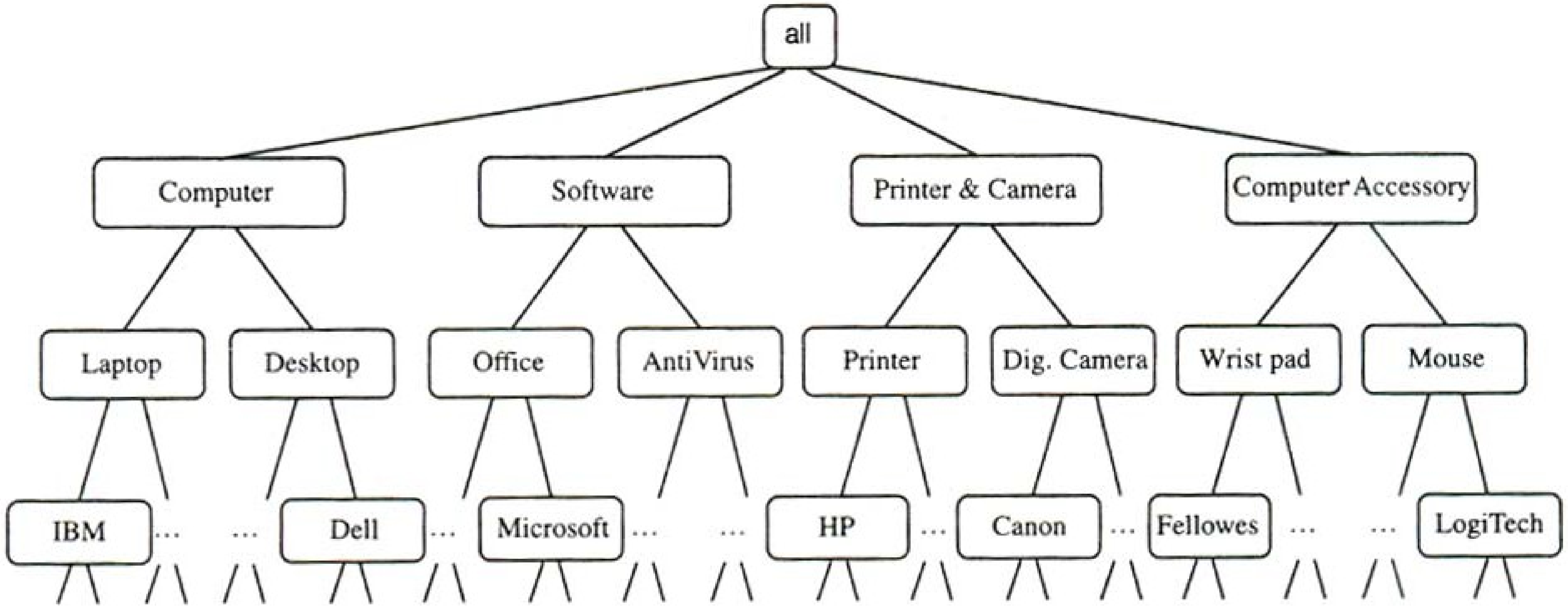
Multiple-Level Association Rules

- Items often form hierarchy.
- Items at the lower level are expected to have lower support.
- Rules regarding itemsets at appropriate levels could be quite useful.
- ARs generated form mining data at multiple levels of abstraction are called multiple-level or multilevel AR
- Multilevel ARs can be mined under a support-confidence framework.

Task-relevant data, *D*.

TID	*Items Purchased*
T100	IBM-ThinkPad-T40/2373, HP-Photosmart-7660
T200	Microsoft-Office-Professional-2003, Microsoft-Plus!-Digital-Media
T300	Logitech-MX700-Cordless-Mouse, Fellowes-Wrist-Rest
T400	Dell-Dimension-XPS, Canon-PowerShot-S400
T500	IBM-ThinkPad-R40/P4M, Symantec-Norton-Antivirus-2003
...	...

Multiple-Level Association Rules : Example



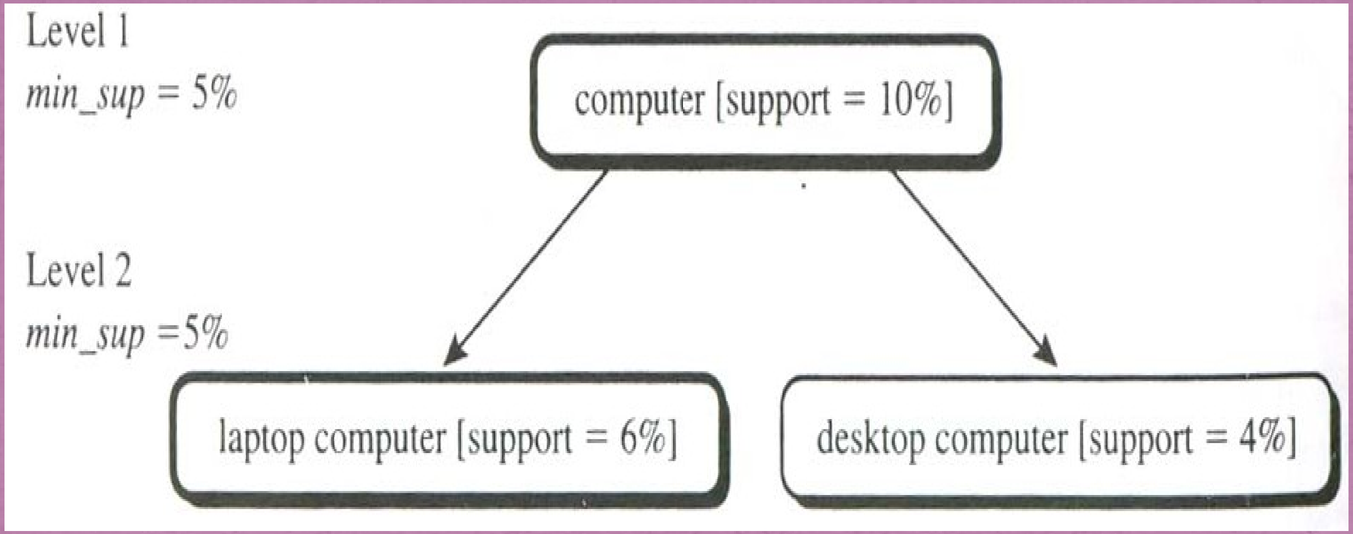
A concept hierarchy for *AllElectronics* computer items.

Multiple-Level Association Rules (cont..)

- Top-down strategy is employed, counts are accumulated for the calculation of frequent itemsets at each concept level, starting from level1 and working downward in the hierarchy for more specific concept levels until no frequent itemsets may be used.
- The variations are :
- Uniform minimum support for all levels
- Using reduced minimum support at lower levels

Uniform Support

- The same min.support is used when mining at each level of abstraction.
- Example:



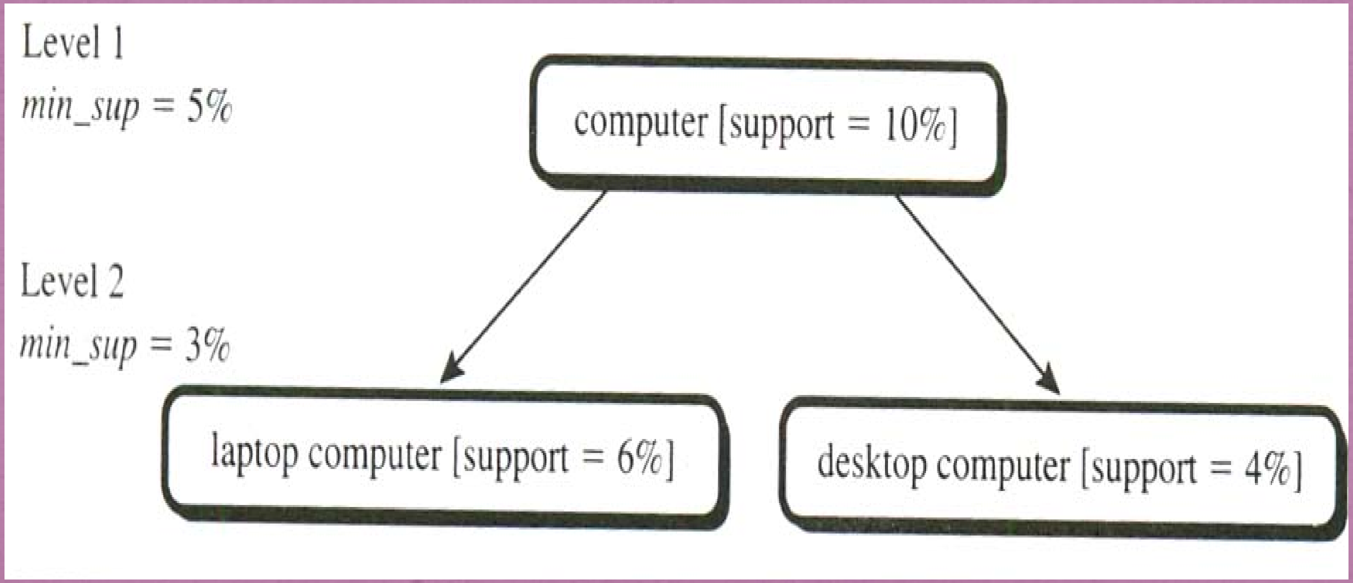
Uniform Support (Cont..)

- Advantages:
  - The search procedure is simplified
  - Users are only required to specify min. support threshold.
  - Apriori like optimization technique can be applied.
  - No need to examine itemsets containing any item whose ancestors do not have minimum support.

- Disadvantages:
  - Unlikely that items at lower levels of abstraction will occur as frequently as those at higher levels of abstraction.
  - If min. Sup. is too high : It could miss some meaningful associations occurring at low abstraction levels.
  - If min. Sup. is too low : It may generate uninteresting associations occurring at high abstraction levels.

Reduced Support

- Each level of abstraction has its own minimum support threshold.
- The deeper the level of abstraction , the smaller the corresponding threshold.



Mining multidimensional AR from Relational database and Data Warehouse

- Single dimensional or Inter-dimensional AR : It contains a single predicate (i.e. buys) with multiple occurrences.
- Ex: buys(X, "digital camera") => buys(X, "HP Printer")
- Such rules are generally used for transactional data.
- Such data can be stored in relational database or data warehouse (which is multidimensional by definition)
- Each database attribute or warehouse dimension can be referred as predicate.
- So we, mine AR containing multiple predicates:
- Ex: age(X, "20….24") ^ occupation(X, "student")=> buys(X, "laptop")
- Multidimensional AR: ARs that involve two or more dimensions or predicates ({ age, occupation, buys})

Mining Multidimensional AR from Relational Database and Data Warehouse (Cont..)

- Each of which occurs only once in the rule , it has no repeated predicates : Inter-dimensional AR
- Hybrid dimensional AR: Multidimensional AR with repeated predicates.
- Ex: age(X, "20….29") ^ buys(X, "laptop")=> buys (X, "HP Printer")
- Database attributes can be:
  - Categorical (finite no. of values with no ordering among the values, occupation, brand, color), are also called nominal attributes.
  - Quantitative (numeric and have implicit ordering among values ,(ex: age , income, price)

Mining Multidimensional AR from Relational Database and Data Warehouse (Cont..)

- Techniques for mining multidimensional ARs for quantitative attributes are :

(a) Discretized using predefined concept hierarchy

- Ex: income attribute may be discretized as:

“0……20k”, “21…..30k”, “31…..40k” and so on.

- The discretization occurs before mining hence known as static discretization.
- Referred as mining multidimensional AR using static discretization of quantitative attributes.

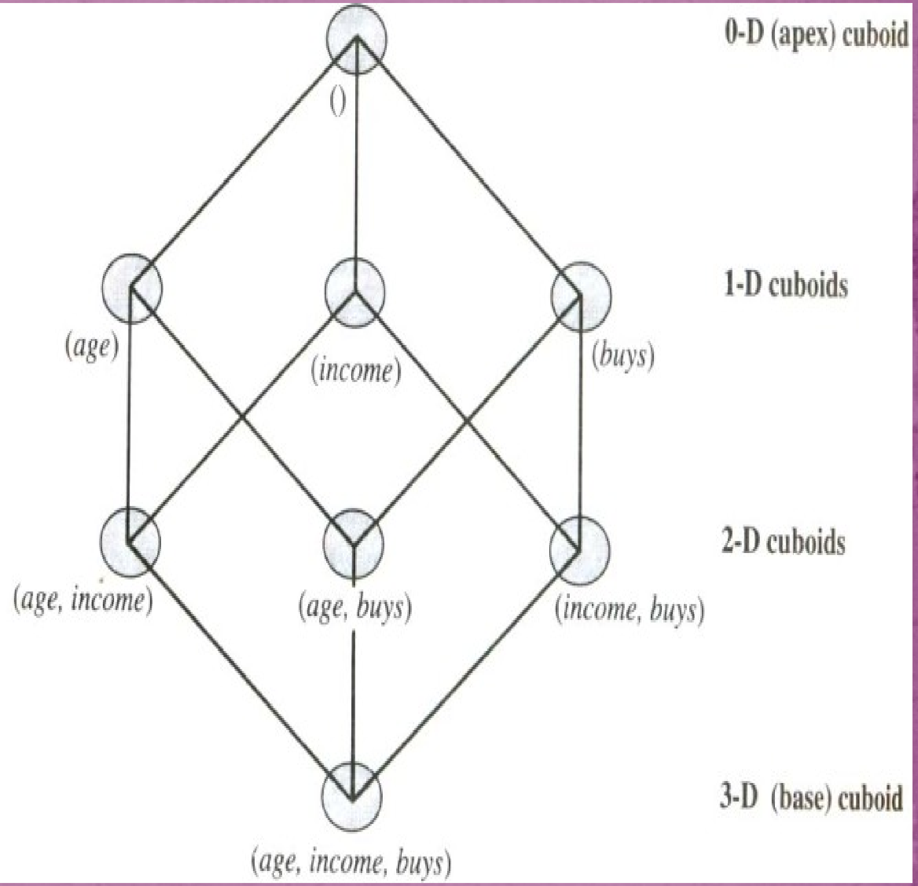
Mining Multidimensional AR from Relational Database and Data Warehouse (Cont..)

(b) Discretized using bins

- Bins may be further combined during mining process
- The process is dynamic
- Strategy treats the numeric attribute values as quantities rather that ranges or categories
- Referred to as Dynamic quantitative ARs.

Mining Multidimensional AR from Static Discretization of quantitaive attributes

- Discretized prior to mining using concept hierarchy.
- Numeric values are replaced by ranges.
- In relational database, finding all frequent k-predicate sets will require *k* or *k*+1 table scans.
- Data cube is well suited for mining.
- The cells of an n-dimensional cuboid correspond to the predicate sets.
- Mining from data cubes can be much faster.



Chapter 6: Mining Association Rules in Large Databases

- Association rule mining
- Mining single-dimensional Boolean association rules from transactional databases
- Mining multilevel association rules from transactional databases
- Mining multidimensional association rules from transactional databases and data warehouse
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Interestingness Measurements

Two popular measurements:

- *support;* and
- *Confidence*

Ex: If A=> B is a rule , then
Support=P(A U B)
Confidence = P(A U B)/ P (B/A)

Criticism to Support and Confidence

- Example 1: (Aggarwal & Yu, PODS98)
  - Among 5000 students
    - 3000 play basketball
    - 3750 eat cereal
    - 2000 both play basket ball and eat cereal
  - *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%] is misleading because the overall percentage of students eating cereal is 75% which is higher than 66.7%.
  - *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%] is far more accurate, although with lower support and confidence

	basketball	not basketball	sum(row)
cereal	2000	1750	3750
not cereal	1000	250	1250
sum(col.)	3000	2000	5000

Criticism to Support and Confidence (Cont.)

Example 2: Of 10,000 transactions analyzed , the data show 6,000 transactions included Computer Games , while 7,500 included Videos, and 4,000 included both Computer Games and Videos. (Min. Sup= 30 % and Min. Conf= 60%). Rule is :

buys(X, “Computer Games”) => buys (X, “Videos”) [40 %, 66%]

Misleading:

- Because probability of purchasing Videos is 75% which is larger than 66%.
- Computer Games and Videos are negatively associated because , the purchase of one item actually decreases the likelihood of purchasing the other.
- It does not measure the real strength of correlation and implication between A and B.

This may lead to unwise business decision.

From Association to Correlation Analysis

To tackle the weakness , a correlation measure can be used :

Correlation Rules

A => B [Support, Confidence, Correlation]

Various correlation measures are there :

-Lift measure
-Chi-square correlation analysis
-All-confidence
-Cosine measure

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

Lift Measure

- The occurrence of itemset A is independent of the occurrence of itemset B , if

$$P(A \cup B) = P(A). P(B)$$

Otherwise, itemsets A and B are dependent and correlated as events.

The lift between A and B can be measured as : $lift_{A,B} = \frac{P(A \cup B)}{P(A)P(B)}$

If value < 1 : The occurrence of A and B are –vely correlated

If value > 1 : The occurrence of A and B are +vely correlated

If value = 1 : The occurrence of A and B are independent and no correlation exists between them.

Lift Measure (Cont…)

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

Is equivalent to :

P(B/A)/ P(B)

or

confidence (A=> B)/ Support(B)

In other words, it asses the degree to which the occurrence of one lifts the occurrence of other.

Example:

If A=Computer Games & B= Videos then,

Given the market conditions , the sale of games is said to increase or lift the likelihood of the sale of videos by a factor of the value returned by the equation.

Lift Measure (Example)

	game	$\overline{game}$	Σ_{row}
video	4,000	3,500	7,500
$\overline{video}$	2,000	500	2,500
Σ_{col}	6,000	4,000	10,000

From the table :

P({game})=0.60

P({video})=0.75

P({game, video})=0.40

Lift (game, video)=P({ game, video})/P({game}). P({video})

= 0.40/(0.60 * 0.75) = 0.89

0.89 < 1 , -ve correlation between occurrence of {game} and {video}

The numerator => likelihood of a customer purchasing both while,

denominator => what the likelihood have been if the two purchases are completely independent.

Such negative correlations can not be found using support and confidence framework.

Correlation using Chi-square

	game	$\overline{game}$	Σ_{row}
video	4,000 (4,500)	3,500 (3,000)	7,500
$\overline{video}$	2,000 (1,500)	500 (1,000)	2,500
Σ_{col}	6,000	4,000	10,000

Because Chi-square value > 1 and the observed value of the slot (game, video)=4,000, which is less than the expected value 4,500, buying game and buying video are negatively correlated.

Constraint-Based Mining

- Interactive, exploratory mining giga-bytes of data?
  - Could it be real? — Making good use of constraints!
- What kinds of constraints can be used in mining?
  - Knowledge type constraint: classification, association, etc.
  - Data constraint: Specify the task relevant data, SQL-like queries
  - Dimension/level constraints: Specify the desired dimension of data.
    - in relevance to region, price, brand, customer category.
  - Rule constraints: Specify the form of rules to be mined
    - small sales (price $< \$10$) triggers big sales (sum $> \$200$).
  - Interestingness constraints: Specify thresholds or statistical measures
    - strong rules (min_support $\geq 3\%$, min_confidence $\geq 60\%$).

References

- R. Agarwal, C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent itemsets. In Journal of Parallel and Distributed Computing (Special Issue on High Performance Data Mining), 2000.
- R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. SIGMOD'93, 207-216, Washington, D.C.
- R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB'94 487-499, Santiago, Chile.
- R. Agrawal and R. Srikant. Mining sequential patterns. ICDE'95, 3-14, Taipei, Taiwan.
- R. J. Bayardo. Efficiently mining long patterns from databases. SIGMOD'98, 85-93, Seattle, Washington.
- S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. SIGMOD'97, 265-276, Tucson, Arizona.
- S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket analysis. SIGMOD'97, 255-264, Tucson, Arizona, May 1997.
- K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. SIGMOD'99, 359-370, Philadelphia, PA, June 1999.
- D.W. Cheung, J. Han, V. Ng, and C.Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. ICDE'96, 106-114, New Orleans, LA.
- M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D. Ullman. Computing iceberg queries efficiently. VLDB'98, 299-310, New York, NY, Aug. 1998.

References (2)

- G. Grahne, L. Lakshmanan, and X. Wang. Efficient mining of constrained correlated sets. ICDE'00, 512-521, San Diego, CA, Feb. 2000.
- Y. Fu and J. Han. Meta-rule-guided mining of association rules in relational databases. KDOOD'95, 39-46, Singapore, Dec. 1995.
- T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. SIGMOD'96, 13-23, Montreal, Canada.
- E.-H. Han, G. Karypis, and V. Kumar. Scalable parallel data mining for association rules. SIGMOD'97, 277-288, Tucson, Arizona.
- J. Han, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. ICDE'99, Sydney, Australia.
- J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. VLDB'95, 420-431, Zurich, Switzerland.
- J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD'00, 1-12, Dallas, TX, May 2000.
- T. Imielinski and H. Mannila. A database perspective on knowledge discovery. Communications of ACM, 39:58-64, 1996.
- M. Kamber, J. Han, and J. Y. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. KDD'97, 207-210, Newport Beach, California.
- M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A.I. Verkamo. Finding interesting rules from large sets of discovered association rules. CIKM'94, 401-408, Gaithersburg, Maryland.

References (3)

- F. Korn, A. Labrinidis, Y. Kotidis, and C. Faloutsos. Ratio rules: A new paradigm for fast, quantifiable data mining. VLDB'98, 582-593, New York, NY.
- B. Lent, A. Swami, and J. Widom. Clustering association rules. ICDE'97, 220-231, Birmingham, England.
- H. Lu, J. Han, and L. Feng. Stock movement and n-dimensional inter-transaction association rules. SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'98), 12:1-12:7, Seattle, Washington.
- H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. KDD'94, 181-192, Seattle, WA, July 1994.
- H. Mannila, H Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery, 1:259-289, 1997.
- R. Meo, G. Psaila, and S. Ceri. A new SQL-like operator for mining association rules. VLDB'96, 122-133, Bombay, India.
- R.J. Miller and Y. Yang. Association rules over interval data. SIGMOD'97, 452-461, Tucson, Arizona.
- R. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. SIGMOD'98, 13-24, Seattle, Washington.
- N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. ICDT'99, 398-416, Jerusalem, Israel, Jan. 1999.

References (4)

- J.S. Park, M.S. Chen, and P.S. Yu. An effective hash-based algorithm for mining association rules. SIGMOD'95, 175-186, San Jose, CA, May 1995.
- J. Pei, J. Han, and R. Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. DMKD'00, Dallas, TX, 11-20, May 2000.
- J. Pei and J. Han. Can We Push More Constraints into Frequent Pattern Mining? KDD'00. Boston, MA. Aug. 2000.
- G. Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In G. Piatetsky-Shapiro and W. J. Frawley, editors, Knowledge Discovery in Databases, 229-238. AAAI/MIT Press, 1991.
- B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. ICDE'98, 412-421, Orlando, FL.
- J.S. Park, M.S. Chen, and P.S. Yu. An effective hash-based algorithm for mining association rules. SIGMOD'95, 175-186, San Jose, CA.
- S. Ramaswamy, S. Mahajan, and A. Silberschatz. On the discovery of interesting patterns in association rules. VLDB'98, 368-379, New York, NY..
- S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. SIGMOD'98, 343-354, Seattle, WA.
- A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. VLDB'95, 432-443, Zurich, Switzerland.
- A. Savasere, E. Omiecinski, and S. Navathe. Mining for strong negative associations in a large database of customer transactions. ICDE'98, 494-502, Orlando, FL, Feb. 1998.

References (5)

- C. Silverstein, S. Brin, R. Motwani, and J. Ullman. Scalable techniques for mining causal structures. VLDB'98, 594-605, New York, NY.
- R. Srikant and R. Agrawal. Mining generalized association rules. VLDB'95, 407-419, Zurich, Switzerland, Sept. 1995.
- R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. SIGMOD'96, 1-12, Montreal, Canada.
- R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. KDD'97, 67-73, Newport Beach, California.
- H. Toivonen. Sampling large databases for association rules. VLDB'96, 134-145, Bombay, India, Sept. 1996.
- D. Tsur, J. D. Ullman, S. Abitboul, C. Clifton, R. Motwani, and S. Nestorov. Query flocks: A generalization of association-rule mining. SIGMOD'98, 1-12, Seattle, Washington.
- K. Yoda, T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Computing optimized rectilinear regions for association rules. KDD'97, 96-103, Newport Beach, CA, Aug. 1997.
- M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. Parallel algorithm for discovery of association rules. Data Mining and Knowledge Discovery, 1:343-374, 1997.
- M. Zaki. Generating Non-Redundant Association Rules. KDD'00. Boston, MA. Aug. 2000.
- O. R. Zaiane, J. Han, and H. Zhu. Mining Recurrent Items in Multimedia with Progressive Resolution Refinement. ICDE'00, 461-470, San Diego, CA, Feb. 2000.