Chapter 4: Mining Frequent Patterns, Associations and Correlations

- **4.1 Basic Concepts**
- **4.2 Frequent Itemset Mining Methods**
- **4.3 Which Patterns Are Interesting?**
  - Pattern Evaluation Methods
- **4.4 Summary**

Frequent Pattern Analysis

- **Frequent Pattern:** a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set

- **Goal:** finding inherent regularities in data

  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify Web documents?

- **Applications:**
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Why is Frequent Pattern Mining Important?

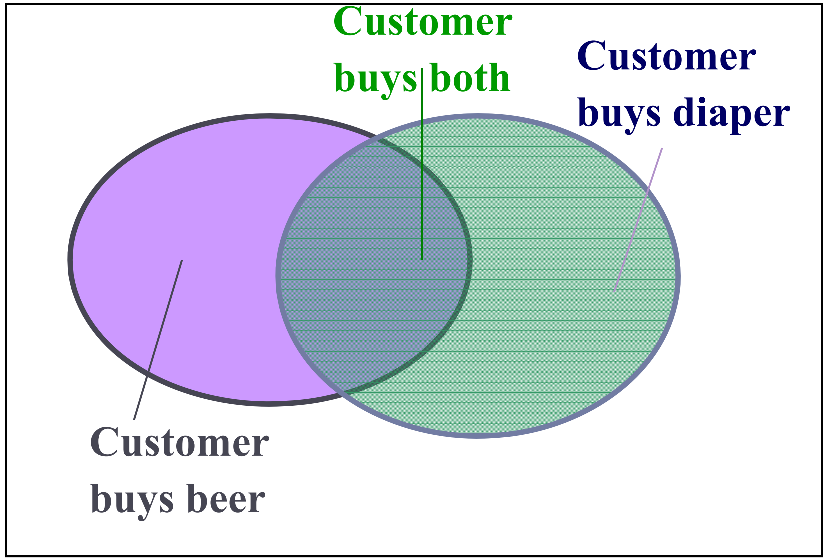
- An important property of datasets

- Foundation for many essential data mining tasks

  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: discriminative, frequent pattern analysis
  - Clustering analysis: frequent pattern-based clustering
  - Data warehousing: iceberg cube and cube-gradient
  - Semantic data compression
  - Broad applications

Frequent Patterns

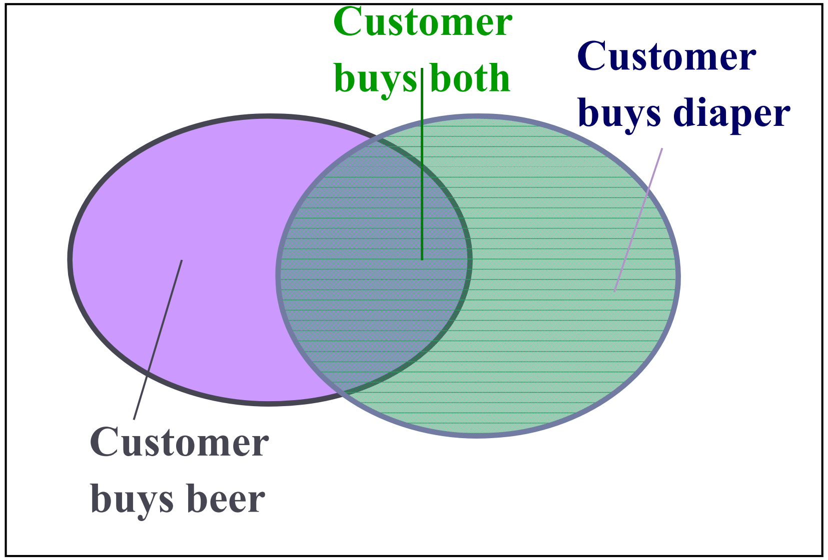
Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- **itemset:** A set of one or more items
- **K-itemset** $X = \{x_1, \ldots, x_k\}$
- **(absolute) support**, or, **support count** of X: Frequency or occurrence of an itemset X
- **(relative) support**, *s*, is the fraction of transactions that contains X (i.e., the **probability** that a transaction contains X)
- An itemset X is **frequent** if X's support is no less than a *minsup* threshold

Association Rules

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- Find all the rules *X* → *Y* with minimum support and confidence threshold
  - **support** , s, probability that a transaction contains X ∪ Y
  - **confidence**, c, conditional probability that a transaction having X also contains Y

Let minsup = 50%, minconf = 50%

Freq. Pat.: Beer:3, Nuts:3, Diaper:4, Eggs:3, {Beer, Diaper}:3

- Association rules: (many more!)
  - Beer → Diaper (60%, 100%)
  - Diaper → Beer (60%, 75%)
- Rules that satisfy both minsup and minconf are called **strong rules**

Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., **$\{a_1, \ldots, a_{100}\}$** contains $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1 =$ **$1.27*10^{30}$ sub-patterns!**

- Solution: *Mine* **closed patterns** *and* **max-patterns** *instead*

- An itemset X is **closed** if X is *frequent* and there exists *no super-pattern* $Y \supset X$, *with the same support* as X

- An itemset X is a **max-pattern** if X is frequent and there exists no frequent super-pattern $Y \supset X$

- Closed pattern is a lossless compression of freq. patterns
  → Reducing the number of patterns and rules

Closed Patterns and Max-Patterns

Example

- DB = {<a_1, …, a_{100}>, < a_1, …, a_{50}>}
- Min_sup=1

- What is the set of **closed itemset**?
  - → <a1, …, a100>: 1
  - → < a1, …, a50>: 2

- What is the set of **max-pattern**?
  - → <a1, …, a100>: 1

- What is the set of **all patterns**?
  - → !!

Computational Complexity

- How many itemsets are potentially to be generated in the worst case?
  - The number of frequent itemsets to be generated is sensitive to the minsup threshold
  - When minsup is low, there exist potentially an exponential number of frequent itemsets
  - The worst case: MN where M: # distinct items, and N: max length of transactions

Chapter 4: Mining Frequent Patterns, Associations and Correlations

- **4.1 Basic Concepts**

- **4.2 Frequent Itemset Mining Methods**

  4.2.1 Apriori: A Candidate Generation-and-Test Approach

  4.2.2 Improving the Efficiency of Apriori

  4.2.3 FPGrowth: A Frequent Pattern-Growth Approach

  4.2.4 ECLAT: Frequent Pattern Mining with Vertical Data Format

- **4.3 Which Patterns Are Interesting?**
  - Pattern Evaluation Methods

- **4.4 Summary**

4.2.1Apriori: Concepts and Principle

- The **downward closure** property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If {beer, diaper, nuts} is frequent, so is {beer, diaper}
  - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- **Apriori pruning principle:** If there is **any** itemset which is infrequent, its superset should not be generated/tested

4.2.1Apriori: Method

- Initially, scan DB once to get frequent 1-itemset
- **Generate** length (k+1) **candidate** itemsets from length k frequent itemsets
- **Test** the candidates against DB
- Terminate when no frequent or candidate set can be generated

Apriori: Example

$Sup_{min} = 2$

Database

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

1st scan

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

Apriori Algorithm

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

L_1 = {frequent items};
for ($k = 1$; $L_k \mathrel{!}= \emptyset$; 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 $\cup_k L_k$;

Candidate Generation

- How to generate candidates?
  - → Step 1: self-joining L_k
  - → Step 2: pruning

- **Example of Candidate Generation**

  - → L_3={abc, abd, acd, ace, bcd}

  - → Self-joining: L_3*L_3: ***abcd*** from ***abc*** and ***abd***, ***acde*** from ***acd*** and ***ace***

  - → Pruning: ***acde*** is removed because ***ade*** is not in L_3

  - → **C4 = {abcd}**

4.2.2 Generating Association Rules

- Once the frequent itemsets have been found, it is straightforward to generate **strong** association rules that satisfy:
  - → **minimum** support
  - → **minimum** confidence

- Relation between support and confidence:

$$\textbf{Confidence}(A \Rightarrow B) = P(B \mid A) = \frac{\text{support\_count}(A \cup B)}{\text{support\_count}(A)}$$

- → **Support_count(A∪B)** is the number of transactions containing the itemsets $A \cup B$
- → **Support_count(A)** is the number of transactions containing the itemset A.

Generating Association Rules

- For each frequent itemset **L**, generate all non empty subsets of **L**
- *For* every non empty subset **S** of **L**, output the rule:

$$S \Rightarrow (L-S)$$

If (support_count(L)/support_count(S)) >= min_conf

Example

- Suppose the frequent Itemset **L={I1,I2,I5}**
- Subsets of L are: **{I1,I2}, {I1,I5},{I2,I5},{I1},{I2},{I5}**
- Association rules :

I1 $\wedge$ I2 $\Rightarrow$ I5	confidence = 2/4= 50%
I1 $\wedge$ I5 $\Rightarrow$ I2	confidence=2/2=100%
I2 $\wedge$ I5 $\Rightarrow$ I1	confidence=2/2=100%
I1 $\Rightarrow$ I2 $\wedge$ I5	confidence=2/6=33%
I2 $\Rightarrow$ I1 $\wedge$ I5	confidence=2/7=29%
I5 $\Rightarrow$ I2 $\wedge$ I2	confidence=2/2=100%

If the minimum confidence =70%

Question: what is the difference between association rules and decision tree rules?

Transactional Database

TID	List of item IDS
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

4.2.2 Improving the Efficiency of Apriori

- **Major computational challenges**

  - Huge number of candidates
  - Multiple scans of transaction database
  - Tedious workload of support counting for candidates

- **Improving Apriori: general ideas**

  - Shrink number of candidates
  - Reduce passes of transaction database scans
  - Facilitate support counting of candidates

(A) DHP: Hash-based Technique

Database

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

1st scan → C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

Making a hash table

10: {A,C}, {A, D}, {C,D} 20: {B,C}, {B, E}, {C,E} 30: {A,B}, {A, C}, {A,E}, {B,C}, {B, E}, {C,E} 40: {B, E}

Hash codes	0	1	2	3	4	5	6
Buckets	{C,E} {C,E} {A,D}	{A,E}	{B,C}		{B,E} {B,E} {B,E}	{A,B}	{A,C} {C,D} {A,C}
Buckets counters	3	1	2	0	3	1	3
	1	0	1	0	1	0	1

min-support=2
We have the following binary vector

{A,B}	1
{A, C}	3
{A,E}	3
{B,C}	2
{B, E}	3
{C,E}	3

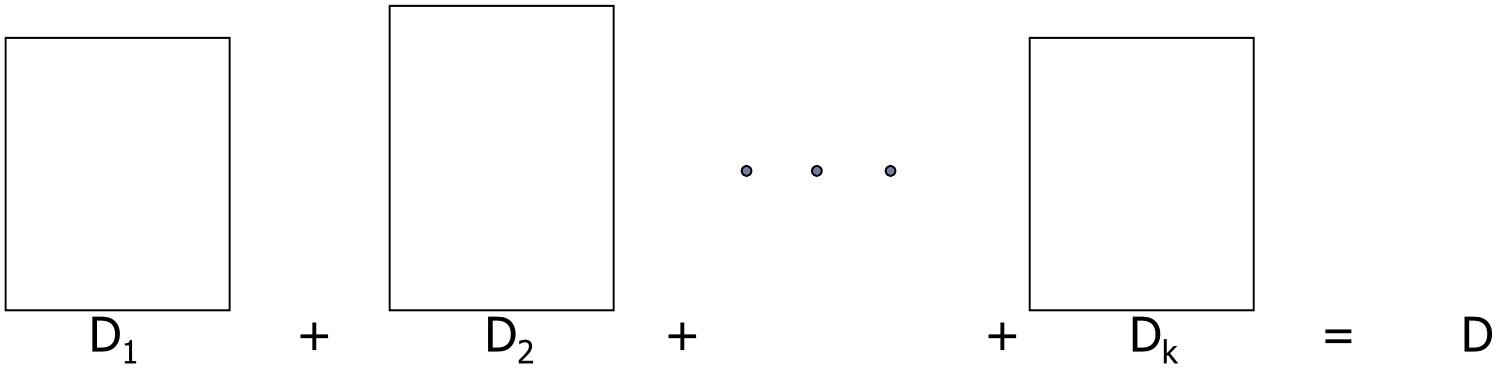
→

{A, C}
{B,C}
{B, E}
{C,E}

J. Park, M. Chen, and P. Yu. An effective hash-based algorithm for mining association rules. SIGMOD'95

(B) Partition: Scan Database Only Twice

- Subdivide the transactions of **D** into **k** non overlapping partitions
- Any itemset that is potentially frequent in **D** must be frequent in at least one of the partitions **Di**
- Each partition can fit into main memory, thus it is read only once
- **Steps:**
  - Scan1: partition database and find local frequent patterns
  - Scan2: consolidate global frequent patterns



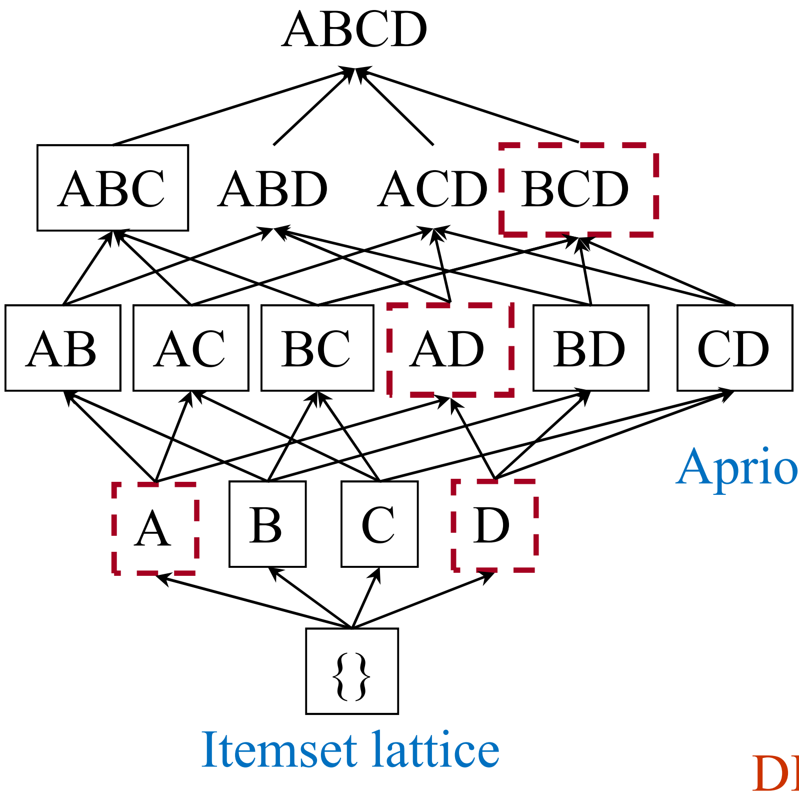
A. Savasere, E. Omiecinski and S. Navathe, VLDB'95

(C) Sampling for Frequent Patterns

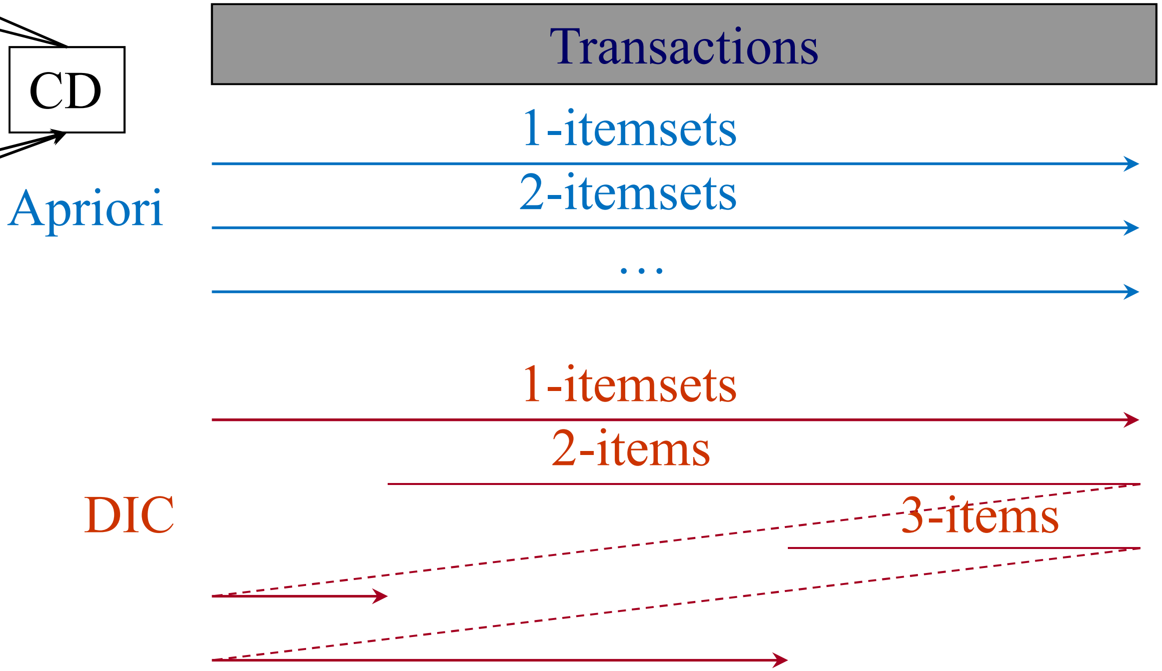
- Select a sample of the original database
- Mine frequent patterns within the sample using Apriori
- Use a lower support threshold than the minimum support to find local frequent itemsets
- Scan the database once to verify the frequent itemsets found in the sample
- Only broader frequent patterns are checked
  - Example: check abcd instead of ab, ac,…, etc.
- Scan the database again to find missed frequent patterns

H. Toivonen. Sampling large databases for association rules. In VLDB'96

(D) Dynamic: Reduce Number of Scans



- Once both A and D are determined frequent, the counting of AD begins
- Once all length-2 subsets of BCD are determined frequent, the counting of BCD begins



S. Brin R. Motwani, J. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In SIGMOD'97

4.2.3 FP-growth: Frequent Pattern-Growth

- Adopts a divide and conquer strategy
- Compress the database representing frequent items into a **frequent –pattern tree** or **FP-tree**
  - Retains the itemset association information
- Divid the compressed database into a set of conditional databases, each associated with one frequent item
- Mine each such databases separately

Example: FP-growth

- The first scan of data is the same as Apriori
- Derive the set of frequent 1-itemsets
- Let min-sup=2
- Generate a set of ordered items

Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2

Transactional Database

TID	List of item IDS
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

Construct the FP-Tree

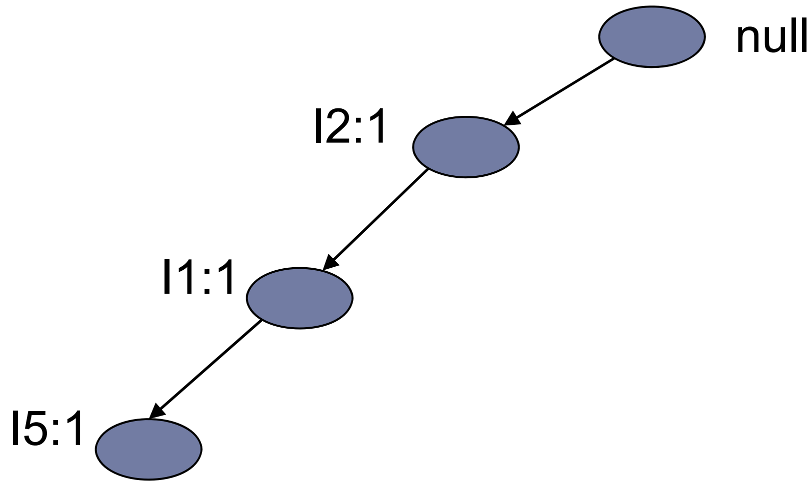
Transactional Database

TID	Items	TID	Items	TID	Items
T100	I1,I2,I5	T400	I1,I2,I4	T700	I1,I3
T200	I2,I4	T500	I1,I3	T800	I1,I2,I3,I5
T300	I2,I3	T600	I2,I3	T900	I1,I2,I3

- Create a branch for each transaction
- Items in each transaction are processed in order

1- Order the items T100: {I2,I1,I5}
2- Construct the first branch: <I2:1>, <I1:1>,<I5:1>

Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2



Construct the FP-Tree

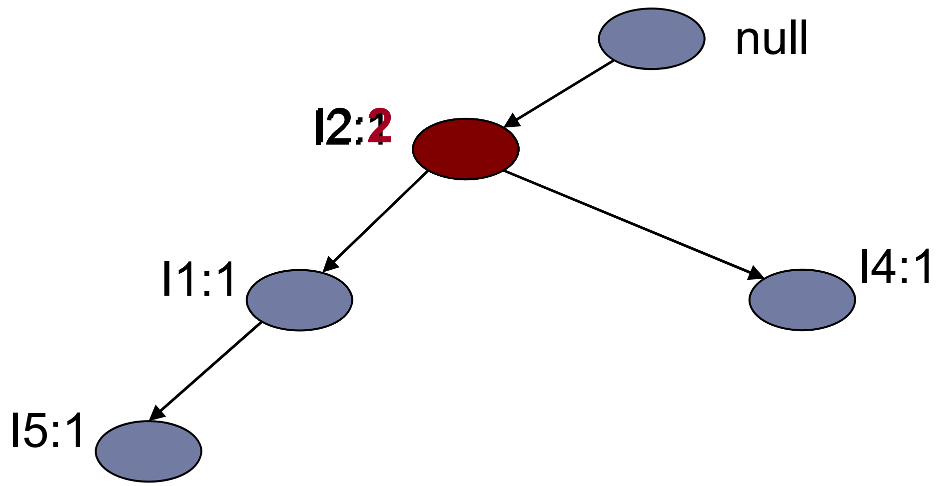
Transactional Database

TID	Items	TID	Items	TID	Items
T100	I1,I2,I5	T400	I1,I2,I4	T700	I1,I3
T200	I2,I4	T500	I1,I3	T800	I1,I2,I3,I5
T300	I2,I3	T600	I2,I3	T900	I1,I2,I3

- Create a branch for each transaction
- Items in each transaction are processed in order

1- Order the items T200: {I2,I4}
2- Construct the second branch: <I2:1>, <I4:1>

Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2



Construct the FP-Tree

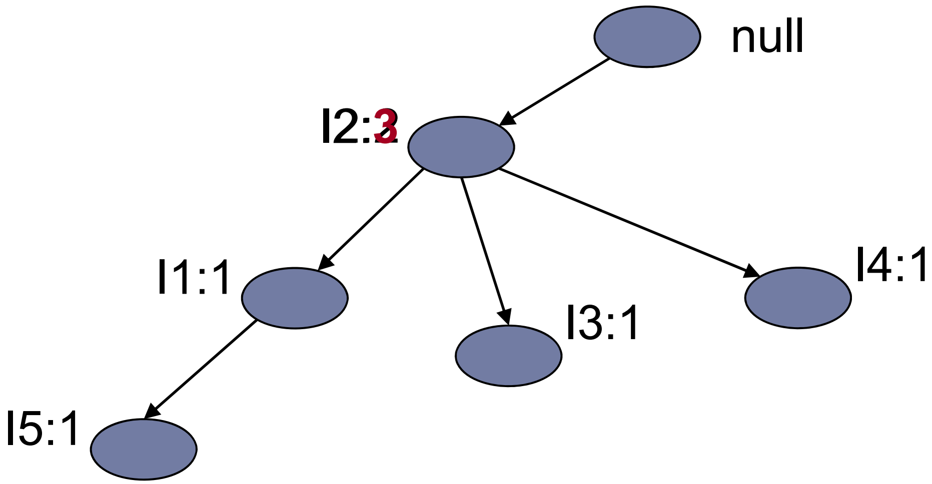
Transactional Database

TID	Items	TID	Items	TID	Items
T100	I1,I2,I5	T400	I1,I2,I4	T700	I1,I3
T200	I2,I4	T500	I1,I3	T800	I1,I2,I3,I5
T300	I2,I3	T600	I2,I3	T900	I1,I2,I3

- Create a branch for each transaction
- Items in each transaction are processed in order

1- Order the items T300: {I2,I3}
2- Construct the third branch: <I2:2>, <I3:1>

Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2



Construct the FP-Tree

Transactional Database

TID	Items	TID	Items	TID	Items
T100	I1,I2,I5	T400	I1,I2,I4	T700	I1,I3
T200	I2,I4	T500	I1,I3	T800	I1,I2,I3,I5
T300	I2,I3	T600	I2,I3	T900	I1,I2,I3

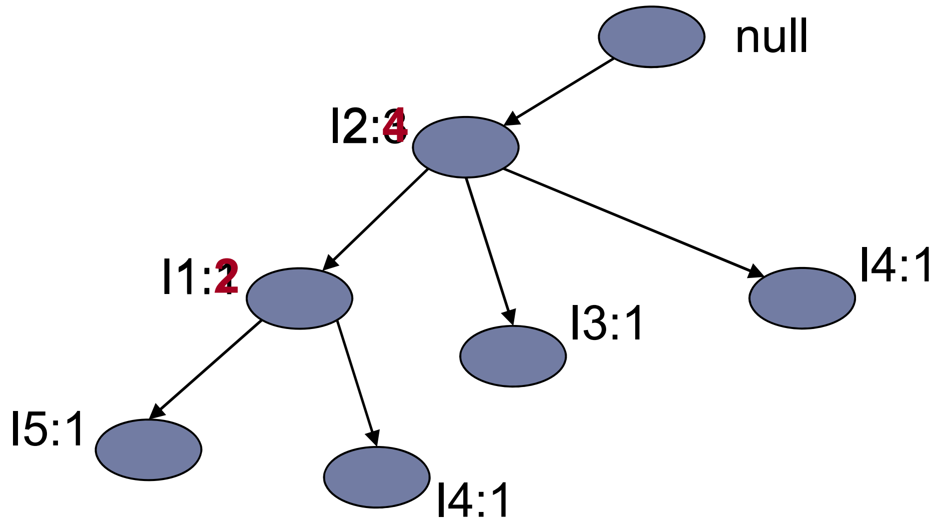
- Create a branch for each transaction

- Items in each transaction are processed in order

1- Order the items T400: {I2,I1,I4}

2- Construct the fourth branch: <I2:3>, <I1:1>,<I4:1>

Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2



Construct the FP-Tree

Transactional Database

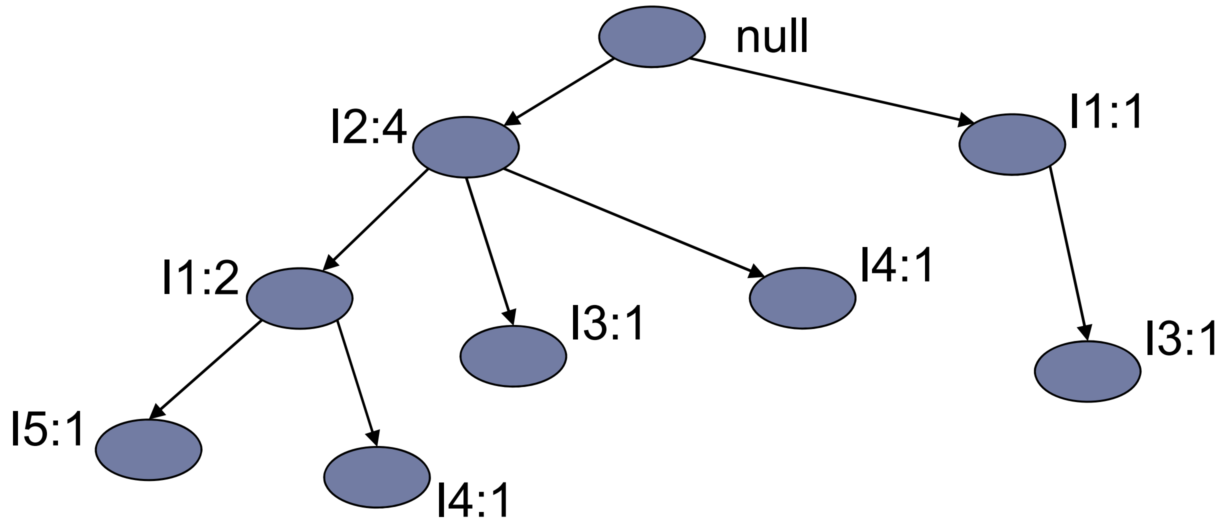
TID	**Items**	**TID**	**Items**	**TID**	**Items**
T100	I1,I2,I5	T400	I1,I2,I4	T700	I1,I3
T200	I2,I4	T500	I1,I3	T800	I1,I2,I3,I5
T300	I2,I3	T600	I2,I3	T900	I1,I2,I3

- Create a branch for each transaction
- Items in each transaction are processed in order

1- Order the items T400: {I1,I3}

2- Construct the fifth branch: <I1:1>, <I3:1>

Item ID	**Support count**
I2	7
I1	6
I3	6
I4	2
I5	2

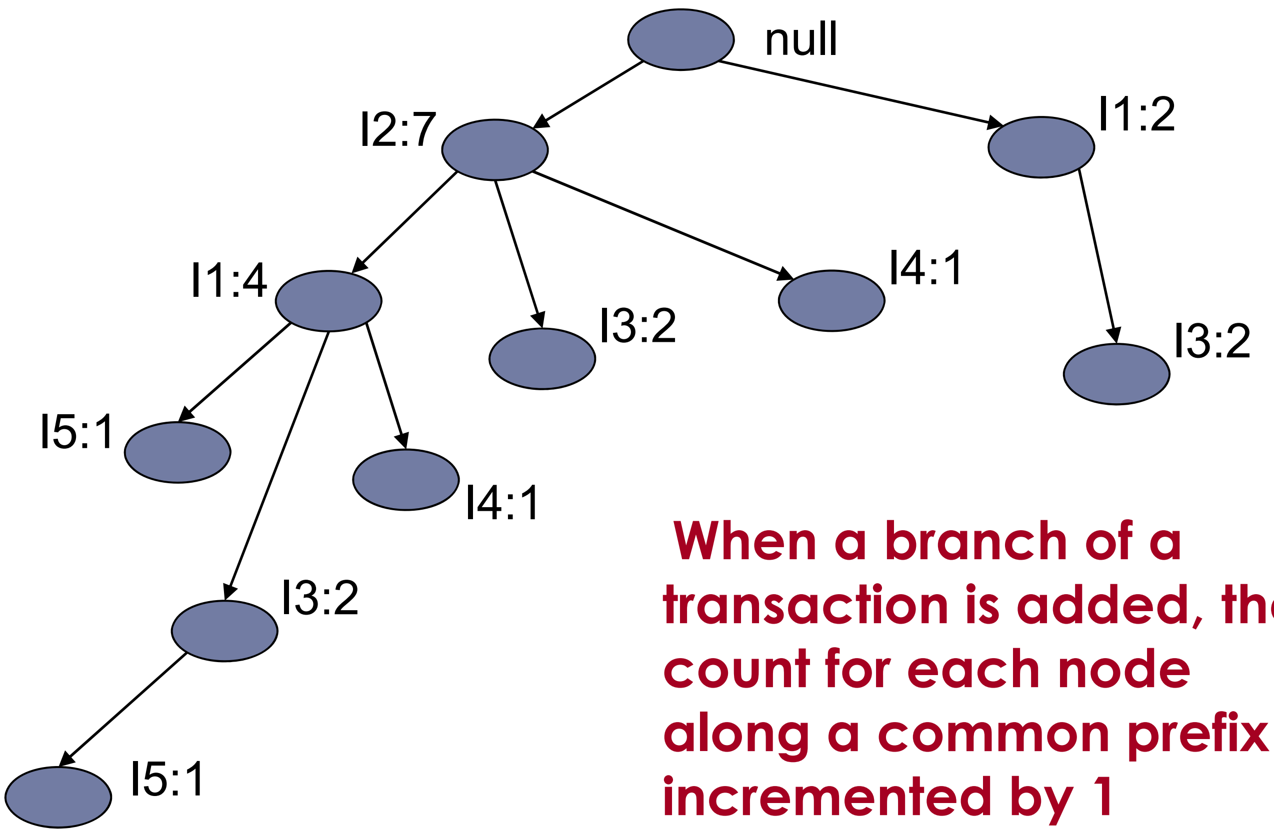


Construct the FP-Tree

Transactional Database

TID	Items	TID	Items	TID	Items
T100	I1,I2,I5	T400	I1,I2,I4	T700	I1,I3
T200	I2,I4	T500	I1,I3	T800	I1,I2,I3,I5
T300	I2,I3	T600	I2,I3	T900	I1,I2,I3

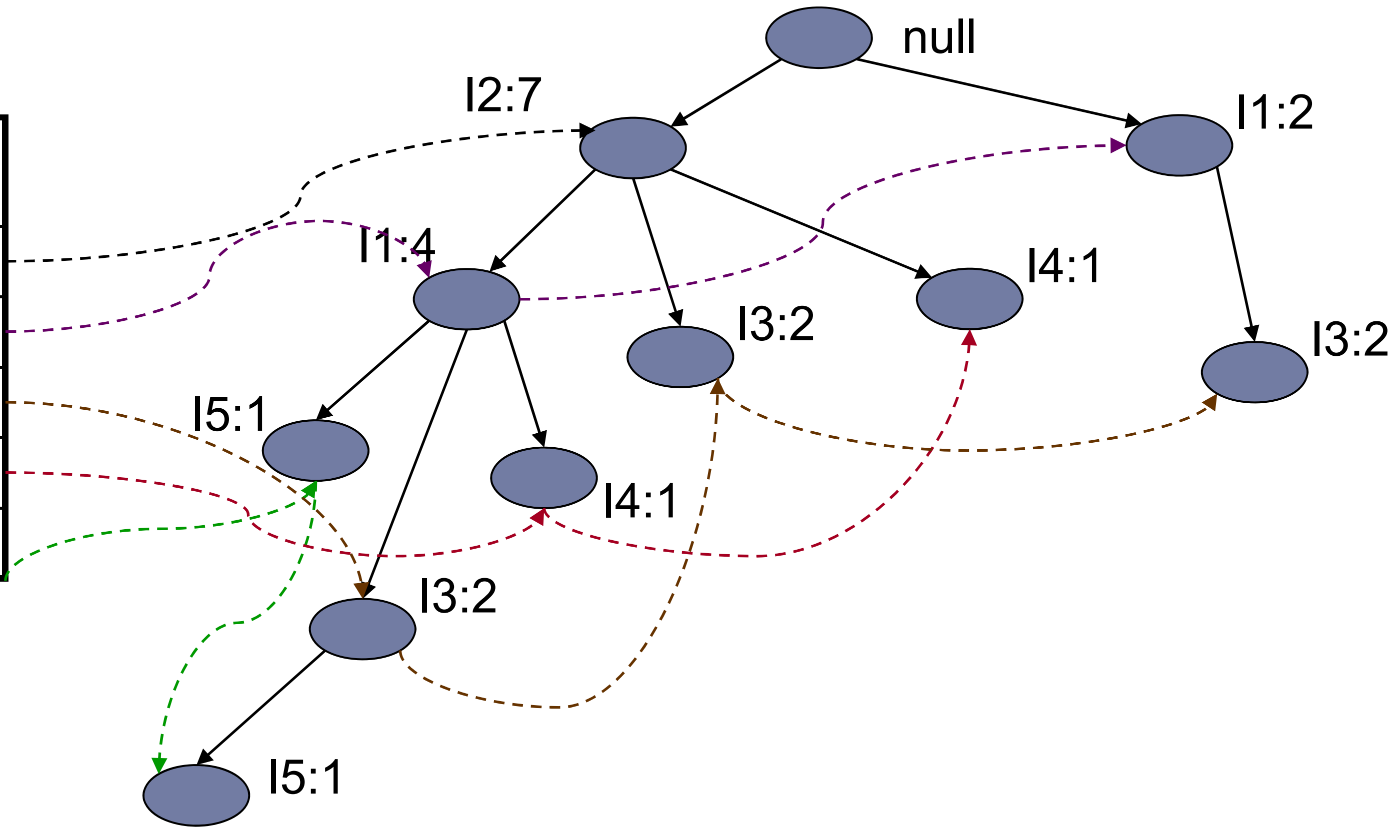
Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2



When a branch of a transaction is added, the count for each node along a common prefix is incremented by 1

Construct the FP-Tree

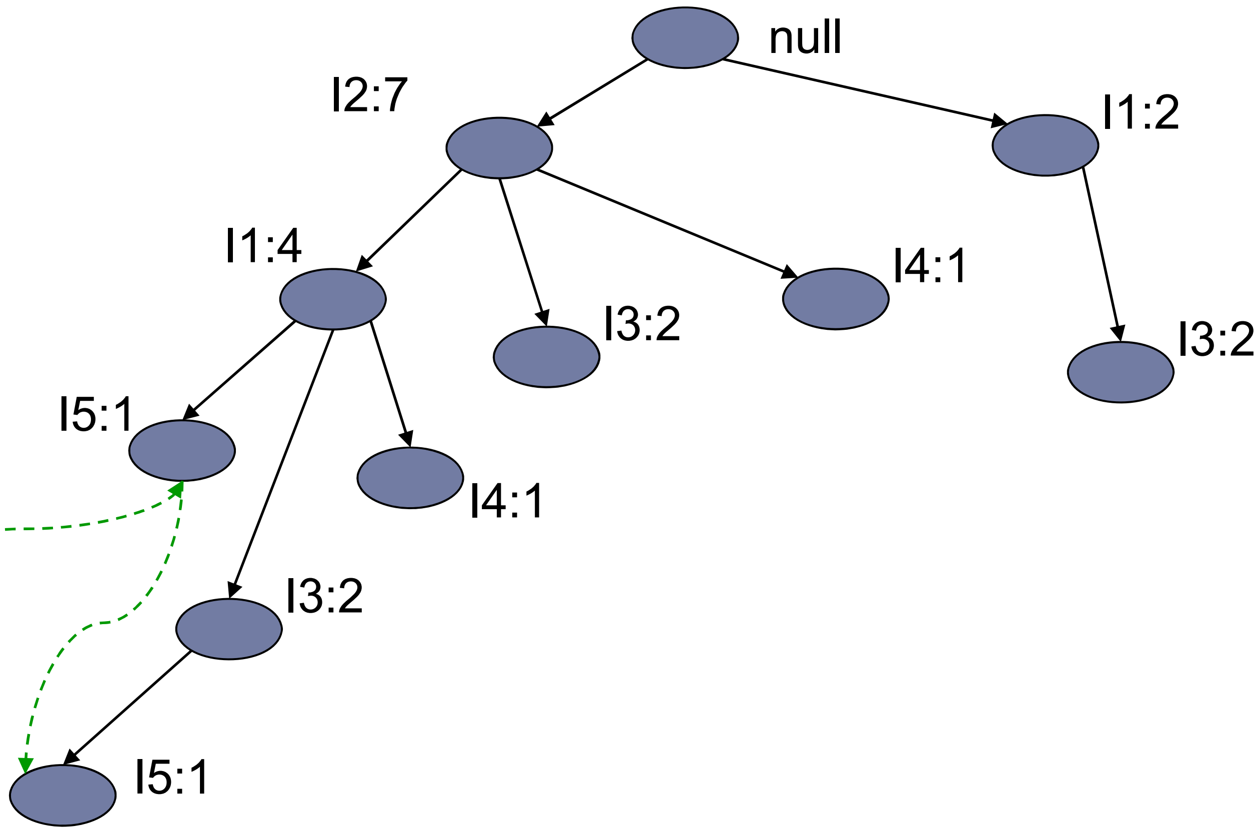
Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2



The problem of mining frequent patterns in databases is transformed to that of mining the FP-tree

Construct the FP-Tree

Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2



-Occurrences of I5: <I2,I1,I5> and <I2,I1,I3,I5>

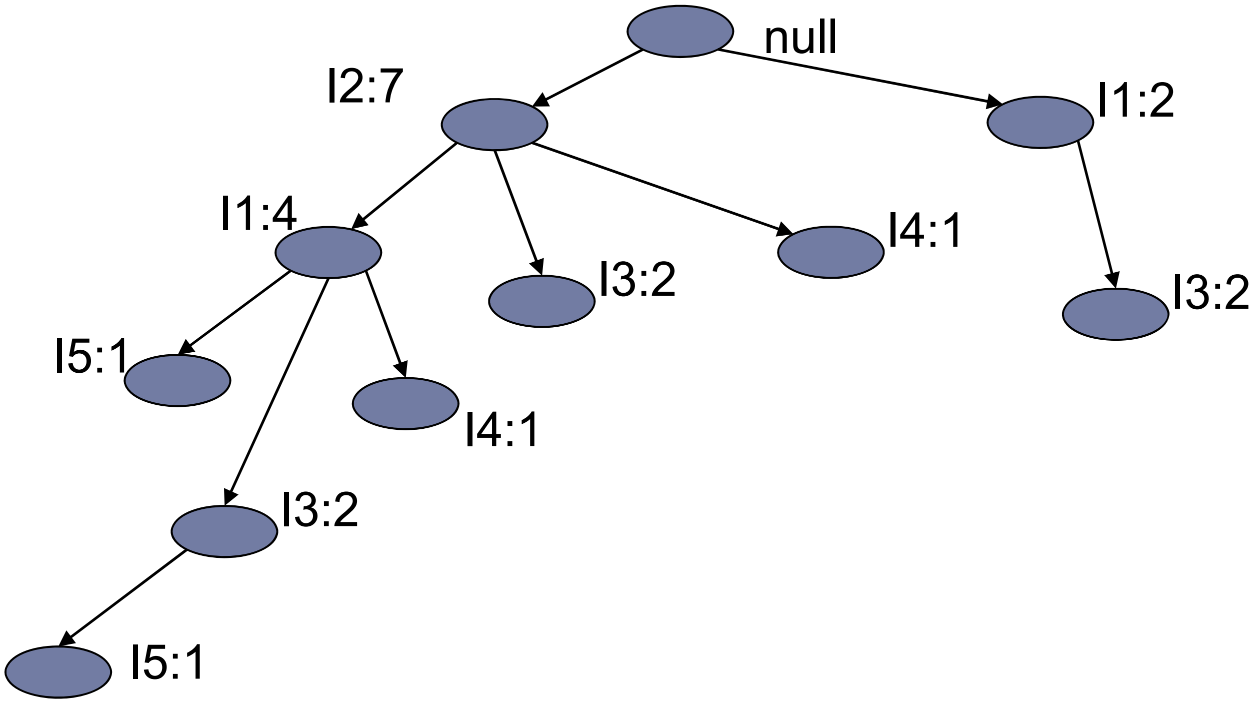
-Two prefix Paths <I2, I1: 1> and <I2,I1,I3: 1>

-Conditional FP tree contains only <I2: 2, I1: 2>, I3 is not considered because its support count of 1 is less than the minimum support count.

-Frequent patterns {I2,I5:2}, {I1,I5:2},{I2,I1,I5:2}

Construct the FP-Tree

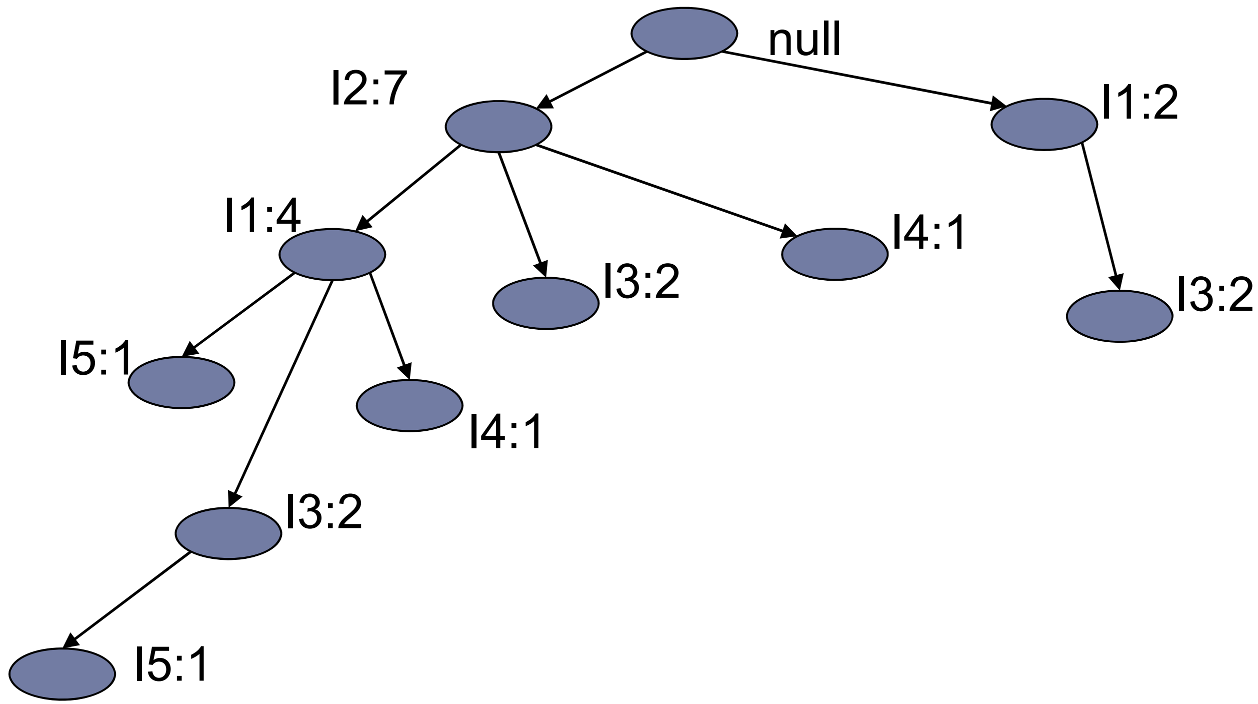
Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2



TID	Conditional Pattern Base	Conditional FP-tree
I5	{{I2,I1:1},{I2,I1,I3:1}}	<I2:2,I1:2>
I4	{{I2,I1:1},{I2,1}}	<I2:2>
I3	{{I2,I1:2},{I2:2}, {I1:2}}	<I2:4,I1:2>,<I1:2>
I1	{I2,4}	<I2:4>

Construct the FP-Tree

Item ID	Support count
I2	7
I1	6
I3	6
I4	2
I5	2



TID	Conditional FP-tree	Frequent Patterns Generated
I5	<I2:2,I1:2>	{I2,I5:2}, {I1,I5:2},{I2,I1,I5:2}
I4	<I2:2>	{I2,I4:2}
I3	<I2:4,I1:2>,<I1:2>	{I2,I3:4},{I1,I3:4},{I2,I1,I3:2}
I1	<I2:4>	{I2,I1:4}

FP-growth properties

- FP-growth transforms the problem of finding long frequent patterns to searching for shorter once recursively and concatenating the suffix
- It uses the least frequent suffix offering a good selectivity
- It reduces the search cost
- If the tree does not fit into main memory, partition the database
- Efficient and scalable for mining both long and short frequent patterns

4.2.4 ECLAT: FP Mining with Vertical Data Format

- Both **Apriori** and **FP-growth** use **horizontal data format**

TID	List of item IDS
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

- Alternatively data can also be represented in **vertical format**

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 Algorithm by Example

- Transform the horizontally formatted data to the vertical format by scanning the database once

TID	List of item IDS
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

→

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}

- The support count of an itemset is simply the length of the TID_set of the itemset

ECLAT Algorithm by Example

Frequent 1-itemsets in vertical format

min_sup=2

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}

- The frequent k-itemsets can be used to construct the candidate (k+1)-itemsets based on the Apriori property

Frequent 2-itemsets in vertical format

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}

ECLAT Algorithm by Example

min_sup=2

Frequent 3-itemsets in vertical format

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

- This process repeats, with k incremented by 1 each time, until no frequent items or no candidate itemsets can be found
- **Properties of mining with vertical data format**
  - Take the advantage of the Apriori property in the generation of candidate (k+1)-itemset from k-itemsets
  - No need to scan the database to find the support of (k+1) itemsets, for k>=1
  - The TID_set of each k-itemset carries the complete information required for counting such support
  - The TID-sets can be quite long, hence expensive to manipulate
  - Use ***diffset*** technique to optimize the support count computation

Chapter 4: Mining Frequent Patterns, Associations and Correlations

- **4.1 Basic Concepts**

- **4.2 Frequent Itemset Mining Methods**

  4.2.1 Apriori: A Candidate Generation-and-Test Approach

  4.2.2 Improving the Efficiency of Apriori

  4.2.3 FPGrowth: A Frequent Pattern-Growth Approach

  4.2.4 ECLAT: Frequent Pattern Mining with Vertical Data Format

- **4.3 Which Patterns Are Interesting?**
  - Pattern Evaluation Methods

- **4.4 Summary**

Strong Rules Are Not Necessarily Interesting

- Whether a rule is interesting or not can be assesses either subjectively or objectively
- Objective interestingness measures can be used as one step toward the goal of finding interesting rules for the user

- **Example of a misleading "strong" association rule**
  - Analyze transactions of AllElectronics data about computer games and videos
  - Of the **10,000** transactions analyzed
    - **6,000** of the transactions include **computer games**
    - **7,500** of the transactions include **videos**
    - **4,000** of the transactions include **both**
  - Suppose that min_sup=30% and min_confidence=60%
  - The following association rule is discovered:

Buys(X, "computer games") ⇒ buys(X, "videos")[support =40%, confidence=66%]

Strong Rules Are Not Necessarily Interesting

Buys(X, "computer games") ⇒ buys(X, "videos")[support 40%, confidence=66%]

- This rule is strong but it is misleading
- The probability of purshasing videos is **75%** which is even larger than **66%**
- In fact computer games and videos are negatively associated because the purchase of one of these items actually decreases the likelihood of purchasing the other
- The confidence of a rule **A ⇒ B** can be deceiving

→ It is only an estimate of the conditional probability of itemset **B** given itemset **A**.

→ It does not measure the real strength of the correlation implication between **A** and **B**

- Need to use **Correlation Analysis**

From Association to Correlation Analysis

- Use **Lift**, a simple correlation 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 the occurences of **A** and **B** is given by

$$\mathbf{Lift(A,B)=P(A \cup B)/P(A)P(B)}$$

→ If > 1, then A and B are positively correlated (the occurrence of one implies the occurrence of the other)
→ If <1, then A and B are negatively correlated
→ If =1, then A and B are independent

- **Example: P({game, video})=0.4/(0.60 × 0.75)=0.89**

Chapter 4: Mining Frequent Patterns, Associations and Correlations

- **4.1 Basic Concepts**
- **4.2 Frequent Itemset Mining Methods**

  4.2.1 Apriori: A Candidate Generation-and-Test Approach

  4.2.2 Improving the Efficiency of Apriori

  4.2.3 FPGrowth: A Frequent Pattern-Growth Approach

  4.2.4 ECLAT: Frequent Pattern Mining with Vertical Data Format
- **4.3 Which Patterns Are Interesting?**
  - Pattern Evaluation Methods
- **4.4 Summary**

4.4 Summary

- **Basic Concepts:** association rules, support-confident framework, closed and max patterns

- **Scalable frequent pattern mining methods**

  → Apriori (Candidate generation & test)
  → Projection-based (FPgrowth)
  → Vertical format approach (ECLAT)

- **Interesting Patterns**
  → Correlation analysis

Applications and Tools in Data Mining

1. Financial Data Analysis

- Banks and Institutions offer a wise variety of banking services
  - Checking and savings accounts for business or individual customers
  - Credit business, mortgage, and automobile loans
  - Investment services (mutual funds)
  - Insurance services and stock investment services
- Financial data is relatively complete, reliable, and of high quality
- What to do with this data?

1. Financial Data Analysis

- **Design of data warehouses for multidimensional data analysis and data mining**

  - Construct **data warehouses** (data come from different sources)

  - **Multidimensional Analysis**: e.g., view the revenue changes by month. By region, by sector, etc. along with some statistical information such as the mean, the average, the maximum and the minimum values, etc.

  - Characterization and class comparison

  - Outlier analysis

1. Financial Data Analysis

- **Loan Payment Prediction and costumer credit policy analysis**

  → Attribute selection and attribute relevance ranking may help indentifying important factors and eliminate irrelevant ones

  → Example of factors related to the risk of loan payment

    - Term of the loan
    - Debt ratio
    - Payment to income ratio
    - Customer level income
    - Education level
    - Residence region

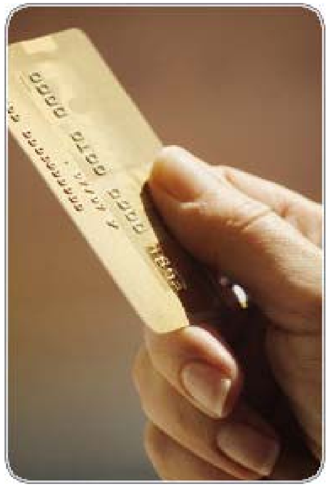
  → The bank can adjust its decisions according to the subset of factors selected (use classification)

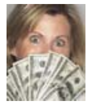
2. Retail Industry

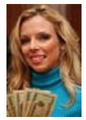
- Collect huge amount of data on sales, customer shopping history, goods transportation, consumption and service, etc.

- Many stores have web sites where you can buy online. Some of them exist only online (e.g., Amazon)

- Data mining helps to
  - Identify costumer buying behaviors
  - Discover customers shopping patterns and trends
  - Improve the quality of costumer service
  - Achieve better costumer satisfaction
  - Design more effective good transportation
  - Reduce the cost of business







2. Retail Industry

- Design **data warehouses**

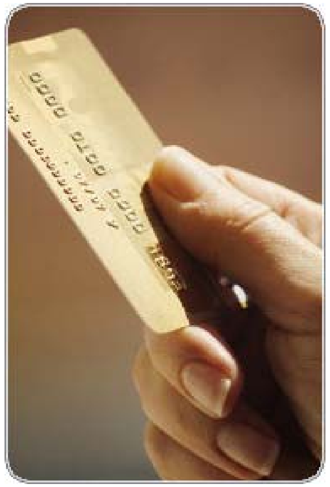
- **Multidimensional** analysis

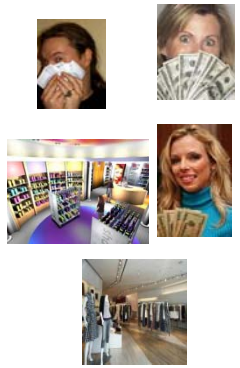
- Analysis of the **effectiveness of sales campaigns**

  - → Advertisements, coupons, discounts, bonuses, etc
  - → Comparing transactions that contain sales items during and after the campaign

- **Costumer retention**
  - → Analyze the change in costumers behaviors

- Product **Recommendation**
  - → Mining association rules
  - → Display associative information to promote sales





3. Telecommunication Industry

- Many different ways of communicating
  - Fax, cellular phone, Internet messenger, images, e-mail, computer and Web data transmission, etc.
- Great demand of data mining to help
  - Understanding the business involved
  - Indentifying telecommunication patterns
  - Catching fraudulent activities
  - Making better use of resources
  - Improve the quality of service

3. Telecommunication Industry

- **Multidimensional** analysis (several attributes)
  - Several features: Calling time, Duration, Location of caller, Location of callee, Type of call, etc.
  - Compare data traffic, system workload, resource usage, user group behavior, and profit

- **Fraudulent Pattern Analysis**
  - Identify potential fraudulent users
  - Detect attempts to gain fraudulent entry to costumer accounts
  - Discover unusual patterns (outlier analysis)

4. Many Other Applications

- **Biological** Data Analysis
  - → E.g., identification and analysis of human genomes and other species

- **Web** Mining
  - → E.g., explore linkage between web pages to compute authority scores (Page Rank Algorithm)

- **Intrusion detection**
  - → Detect any action that threaten file integrity, confidentiality, or availability of a network resource

How to Choose a Data Mining System (Tool)?

- Do data mining systems share the same well defined operations and a standard query language?

- **No**

- Many commercial data mining systems have a little in common

  - Different functionalities
  - Different methodology
  - Different data sets

- You need to carefully choose the data mining system that is appropriate for your task

How to Choose a Data Mining System (Tool)?

- **Data Types and Sources**
  - Available systems handle formatted record-based, relational-like data with numerical, and nominal attributes
  - That data could be on the form of ASCII text, relational databases, or data warehouse data
  - It is important to check which kind of data the system you are choosing can handle
  - It is important that the data mining system supports ODBC connections (Open Database Connectivity)

- **Operating System**
  - A data mining system may run only on one operating system
  - The most popular operating systems that host data mining tools are UNIX/LINUX and Microsoft Windows

How to Choose a Data Mining System (Tool)?

- **Data Mining functions and Methodologies**
  - Some systems provide only one data mining function(e.g., classification). Other system support many functions
  - For a given data mining function (e.g., classification), some systems support only one method. Other systems may support many methods (k-nearest neighbor, naive Bayesian, etc.)
  - Data mining system should provide default settings for non experts

How to Choose a Data Mining System (Tool)?

- **Coupling data mining with databases(data warehouse) systems**
  - → **No Coupling**
    - A DM system will not use any function of a DB/DW system
    - Fetch data from particular resource (file)
    - Process data and then store results in a file
  - → **Loose coupling**
    - A DM system use some facilities of a DB/DW system
    - Fetch data from data repositories managed by a DB/DW
    - Store results in a file or in the DB/DW
  - → **Semi-tight coupling**
    - Efficient implementation of few essential data mining primitives (sorting, indexing, histogram analysis) is provided by the DB/DW
  - → **Tight coupling**
    - A DM system is smoothly integrated into the DB/DW
    - Data mining queries are optimized
- **Tight coupling is highly desirable because it facilitates implementations and provide high system performance**

How to Choose a Data Mining System (Tool)?

- **Scalability**
  - Query execution time should increase linearly with the number of dimensions
- **Visualization**
  - “A picture is worth a thousand words”
  - The quality and the flexibility of visualization tools may strongly influence usability, interpretability and attractiveness of the system
- **Data Mining Query Language and Graphical user Interface**
  - High quality user interface
  - It is not common to have a query language in a DM system

Examples of Commercial Data Mining Tools

Database system and graphics vendors

- Intelligent Miner (IBM)
- Microsoft SQL Server 2005
- MineSet (Purple Insight)
- Oracle Data Mining (ODM)

Examples of Commercial Data Mining Tools

Vendors of statistical analysis or data mining software

- Clementine (SPSS)
- Enterprise Miner (SAS Institute)
- Insightful Miner (Insightful Inc.)

Examples of Commercial Data Mining Tools

Machine learning community

- CART (Salford Systems)
- See5 and C5.0 (RuleQuest)
- Weka developed at the university Waikato (open source)

End of The Data Mining Course

Questions? Suggestions?