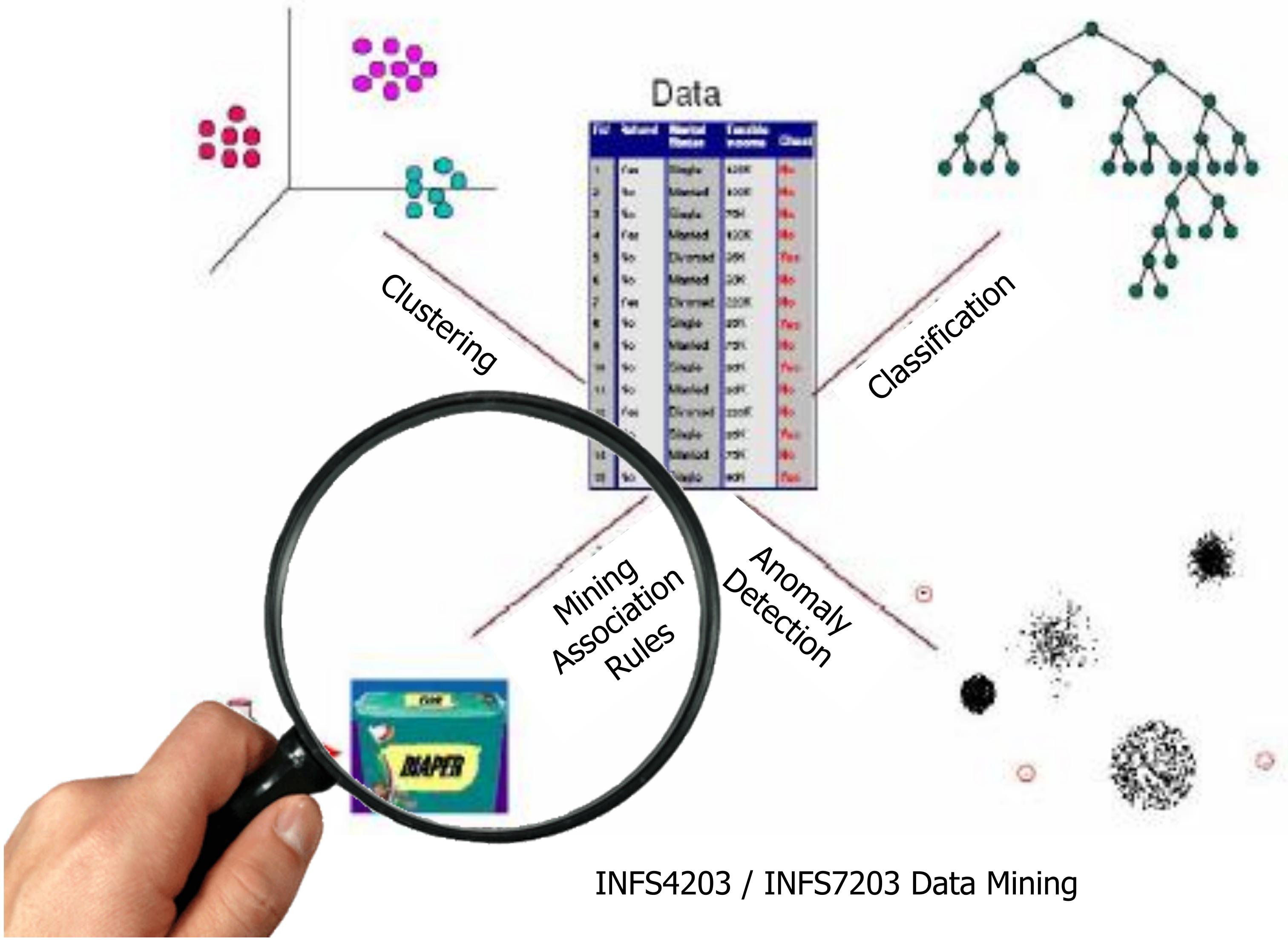
Data Mining Tasks



Data Mining:
Mining Association Rules

What is Association Rule Mining?

- Association rules mining
  - Discovering interesting relations between objects in large databases

- Market basket analysis
  - The problem is to analyse customer buying habits by finding associations between the different items that customers place in their "shopping baskets"

    - Given a set of **transactions**, find **rules** that will **predict** the occurrence of an item based on the occurrences of other items in the transaction

Example: Market Basket Analysis

Anything interesting?

Bread → Milk (100%)
Diapers → Beer (66%)
Diapers → Milk (100%)

Customers who buy diapers also tend to buy beer

↓

Indentify potential cross-selling opportunities among related items

Motivation (market basket analysis)

- If customers are buying milk, how likely is that they also buy bread?

- Such rules help retailers to:
  - Plan the shelf space: by placing milk close to bread they may increase the sales
  - Provide advertisements/recommendation to customers that are likely to buy some products
  - Offer discounts on items that are likely to be bought together to increase the sales

Problem Statement

- **Given:**
  1. A database of transactions
  2. Each transaction is a list of items

     E.g.: items purchased by a customer in a visit

- **Find:**
  - All rules that correlate the presence of one set of items with another set of items

    E.g., 80% of customers who buy {diapers} tend to buy {beer, milk}.

Association Rule Mining

Market-Basket transactions

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Association Rules

{Diaper} → {Beer},
{Milk, Bread} → {Eggs,Coke},
{Beer, Bread} → {Milk},

…

…

Transaction data: documents

- A text document dataset
  - 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

- Unusual words appearing together in a large number of documents, e.g., "Brad" and "Angelina," may indicate an interesting relationship.

Formal Notations

- An **item**: an item in a basket
- 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
- A **transaction**: items purchased in a basket
  - it may have TID (transaction ID)
- A **transactional dataset**: A set of transactions

Formal Notations

- An **association rule** is an implication of the form:
  $\boldsymbol{X} \rightarrow \boldsymbol{Y}$, where:
  - $X, Y \subset I$,
  - I is the set of all items,
  - $X \cap Y = \emptyset$

e.g.: Bread, Butter → Milk
Milk, Diapers → Beer

- Are all rules equally interesting?!

Support and Confidence

- Rule $X \to Y$
- **Rule support**
  - **(absolute) support** or support count:
    - frequency or **count** of an itemset $X \cup Y$
  - **(relative) support**:
    - probability that a transaction contains $X \cup Y$
    - support $(X \to Y) = P(X \cup Y)$
  - **Support S:**
    - percentage of transactions that contain both X and Y

P for probability.

Example: Frequent Itemsets

- Items={milk, coke, pepsi, beer, juice}.
- Support threshold = 3 baskets.

B_1 = {m, c, b}　　　B_2 = {m, p, j}

B_3 = {m, b}　　　B_4 = {c, j}

B_5 = {m, p, b}　　　B_6 = {m, c, b, j}

B_7 = {c, b, j}　　　B_8 = {b, c}

- Frequent itemsets: {m}, {c}, {b}, {j},

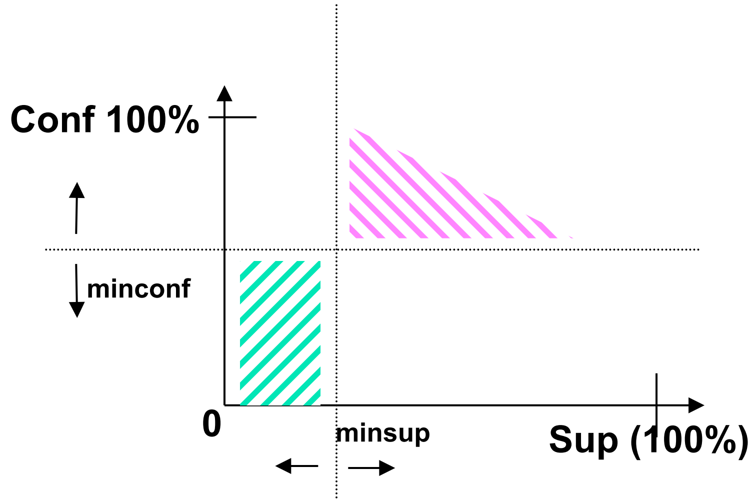
{m,b}, {b,c}, {c,j}.

Support and Confidence

- Rule $X \rightarrow Y$
- **Rule confidence:**
  - **Conditional probability**
  - Confidence $(X \rightarrow Y) = P(Y \mid X) = P(X \cup Y) / P(X)$
  - **Confidence C:**
    - Percentage of transactions that contain Y given they contain X
    - Measures how often items in Y appear in transactions that contain X
  - Example:
    - Support (printer→ink) **=** support (ink→printer)
    - But, confidence (printer→ink) **≠** confidence (ink→printer)

Support and Confidence

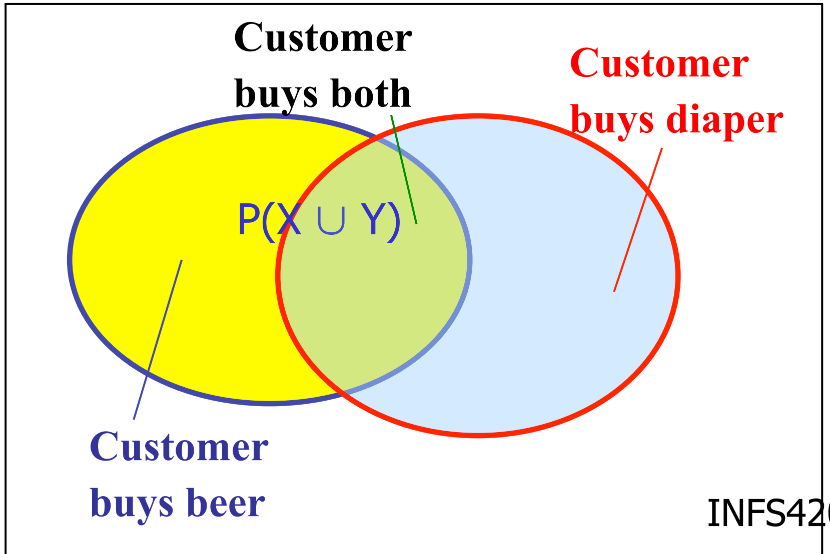
- User-specified **thresholds:**
  - Minimum Support
    - *min_sup*
  - Minimum Confidence
    - *min_conf*



Rules that satisfy both *min_sup* and *min_conf* are called **strong**

Example

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	support count	support
~~Milk~~	~~2~~	~~40%~~
Beer		
Nuts		
Diaper		
Eggs		
Beer, Diaper		
… …		

min_sup=50%

Example

Find all the rules $X \rightarrow Y$ with minimum support and confidence

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

min_sup=50%
min_conf = 50%

Rule: Diaper→Beer
support: 3 (60%)
confidence: P(Beer,Diaper)/**P(Diaper)**
= 60%/80% = **75%**

Rule: Beer→Diaper
support: 3 (60%)
confidence: P(Beer,Diaper)/**P(Beer)**
= 60%/60% = **100%**

Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  1. support ≥ *min_sup* threshold
  2. confidence ≥ *min_conf* threshold

- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *min_sup* or *min_conf* thresholds

  ⇒ Computationally prohibitive!

Mining Association Rules

1. **Apriori Algorithm**

2. Frequent Pattern (FP) Growth Algorithm

Mining Association Rules

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

Observations:

- All the above rules are partitions of the same itemset:
  {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, **decouple** the support and confidence requirements

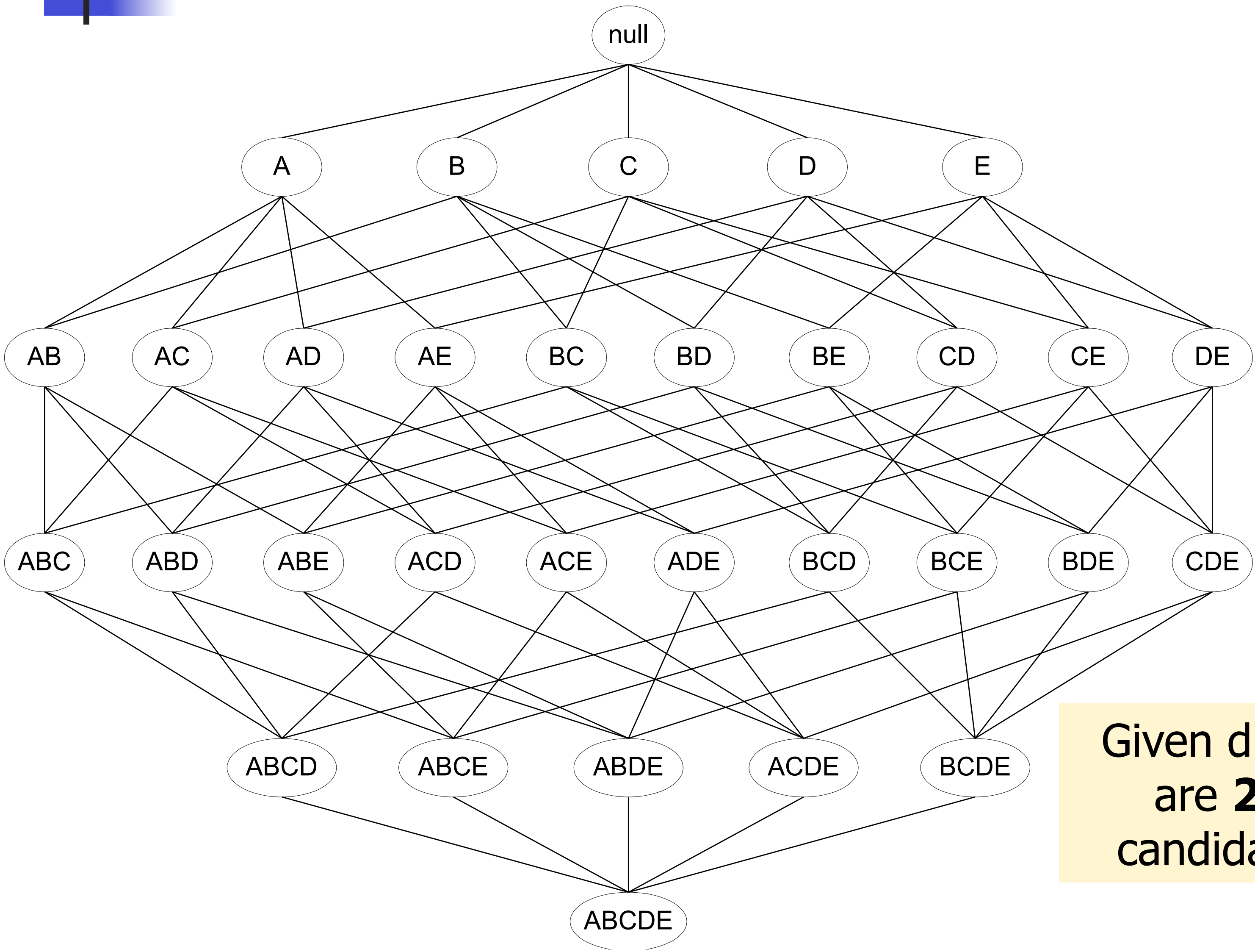
Mining Association Rules

- Two-step approach:

  1. Frequent Itemset Generation
     - Generate all itemsets whose support ≥ minsup

  2. Rule Generation
     - Generate high confidence rules from each frequent itemset
       - each rule is a partitioning of a frequent itemset

Frequent Itemset Generation



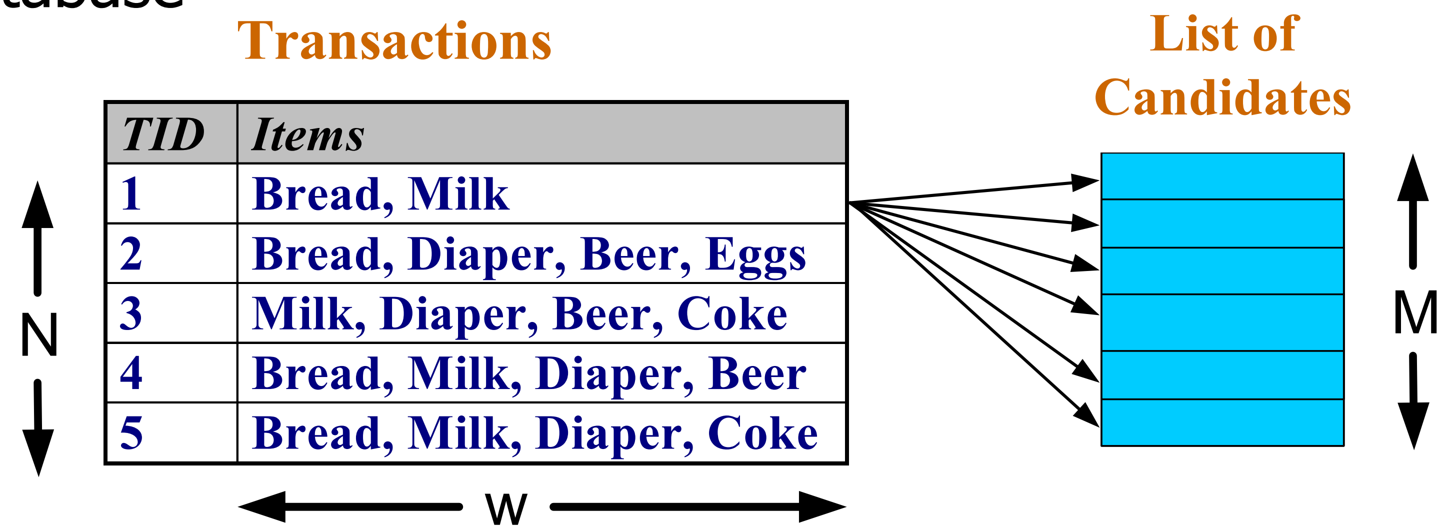
Given d items, there are **2^d** possible candidate itemsets

Frequent Itemset Generation

- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database

Transactions

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke



  - Match each transaction <u>against every candidate</u>
  - Complexity ~ O(NMw) => Expensive since $M = 2^d$!!!

Scale of the Problem

- WalMart:
  - Sells 100,000 items and can store billions of baskets.
- The Web
  - Has billions of words and many billions of pages.

Reducing Number of Candidates

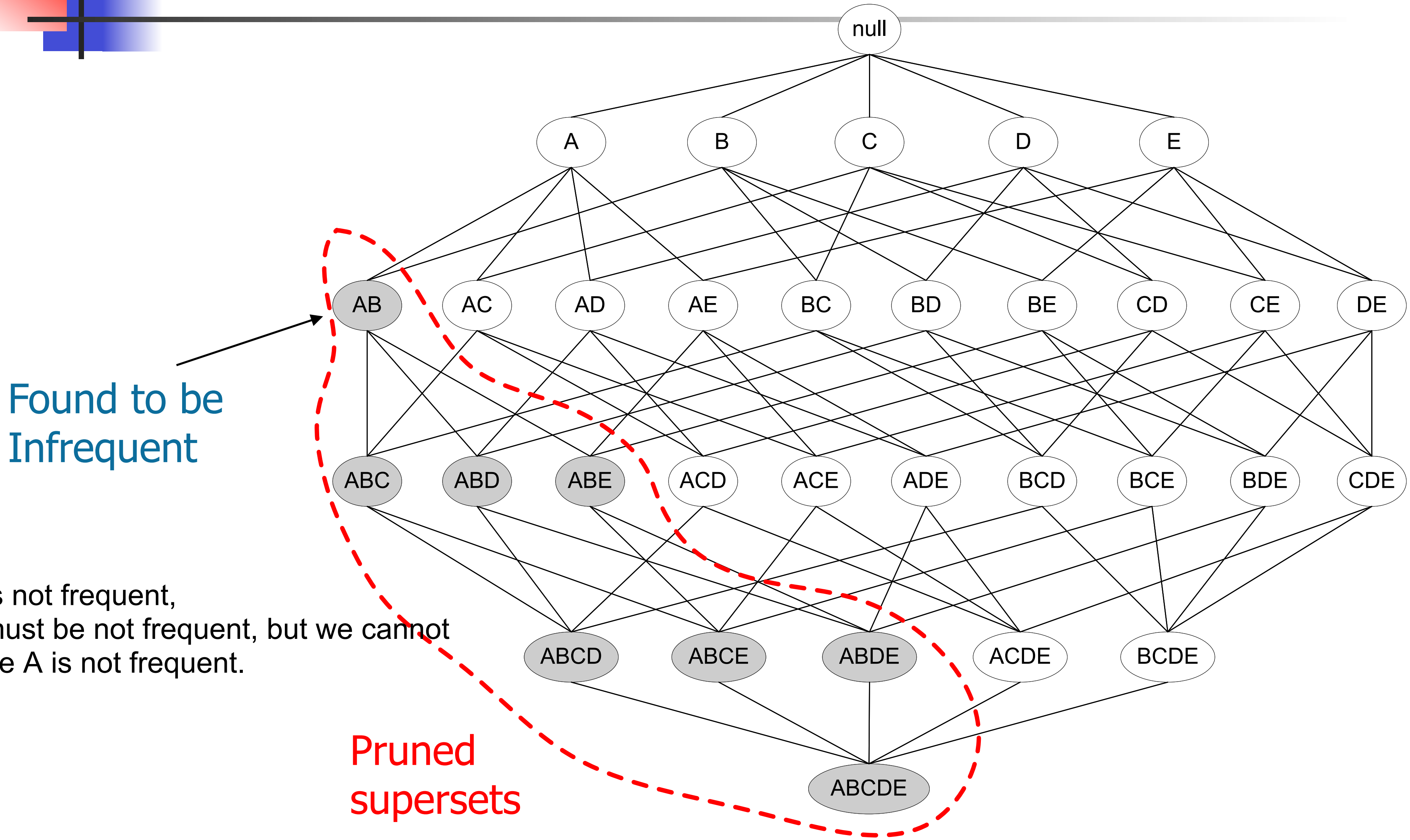
- Apriori principle:
  - If an itemset is frequent, then all of its **subsets** must also be frequent

- Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

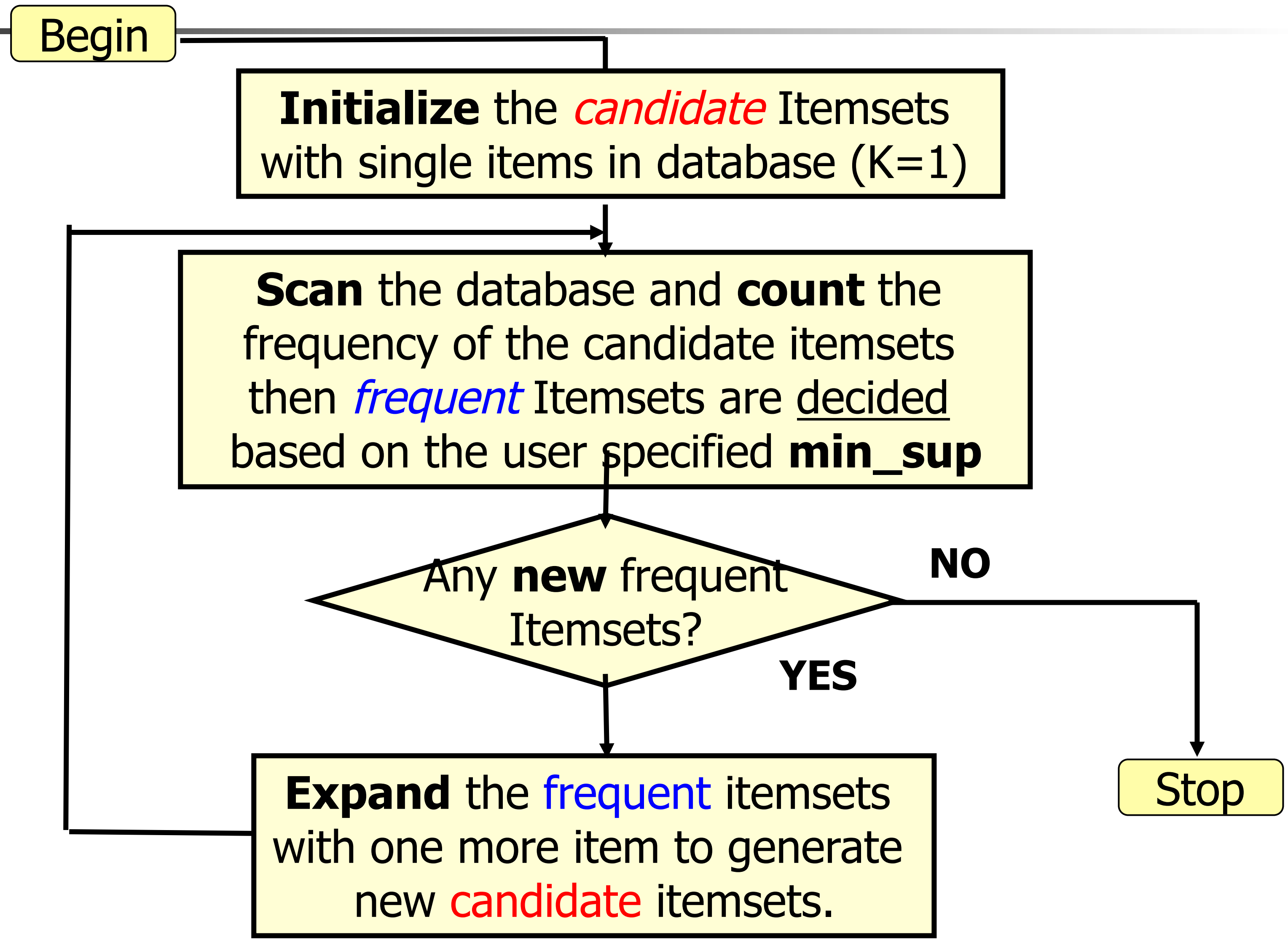
  - Support of an itemset never exceeds the support of its subsets
  - This is known as the anti-monotone property of support

Illustrating Apriori Principle



if AB is not frequent,
ABC must be not frequent, but we cannot
assume A is not frequent.

The Apriori Algorithm



The Apriori Algorithm—An Example

Database TDB

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

$Sup_{min} = 2$

1st scan

candidates

C_1

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

frequent items

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

The Apriori Algorithm (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 $!=\emptyset$; 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 $\cup_k L_k$;

Implementation of Apriori

- 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
  - C_4 = {*abcd*}

开头两个一样的 组合在一起

abc and abd

acd and ace

all aubset should be frequent. but in acde, there is not ade

Candidate Generation: SQL Implementation

- SQL Implementation of candidate generation
  - Suppose the items in L_{k-1} are listed in an order
  - Step 1: self-joining L_{k-1}

    insert into $\boldsymbol{C_k}$

    select $\boldsymbol{p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}}$

    from $\boldsymbol{L_{k-1}\ p, L_{k-1}\ q}$

    where $\boldsymbol{p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2},}$

    $\boldsymbol{p.item_{k-1} \mathrel{!=} q.item_{k-1}}$
  - Step 2: pruning

    forall ***itemsets c in*** $\boldsymbol{C_k}$ do

    forall ***(k-1)-subsets s of c*** do

    if *(s is not in* L_{k-1}*)* **then delete** *c* **from** *C*

Rule Generation

- Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ **satisfies** the minimum confidence requirement

  - If {A,B,C,D} is a frequent itemset, candidate rules:

    ABC →D, ABD →C, ACD →B, BCD →A,
    A →BCD, B →ACD, C →ABD, D →ABC
    AB →CD, AC → BD, AD → BC, BC→AD,
    BD →AC, CD →AB,

Factors Affecting Complexity

- Choice of minimum support threshold
  - lower support threshold results in more frequent itemsets
- Dimensionality (number of items) in the data set
  - 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
  - This may increase max length of frequent itemsets

Midterm Guidelines

- Wednesday Sept 19th @8:00am
- Location:
  - 01-E109
  - 76-228
- Duration: 90 minutes
- Up to and including material covered last week!
- Problem solving, short fill-in questions
- Calculator & Student ID
- Please check last years final exam questions!