Association Analysis

Meganingrum Arista Jiwanggi
meganingrum@cs.ui.ac.id

Data Mining for Big Data
Short Course

Pusilkom UI
16 – 20 Juli 2018

Market Basket Analysis

Retail stores frequently use this for: **assist in marketing, advertising, floor placement, and inventory control**

WordNet --> kamus + thesaurus

Sample Data

- **Tuple** berkaitan dengan id transaksi

A tuple is the list of items purchased at one time

- **Itemset** tidak berhubungan dg transaksi

➢ An itemset is a set of items.

e.g. : {Bread, Jelly, Peanut Butter}

➢ k-itemset

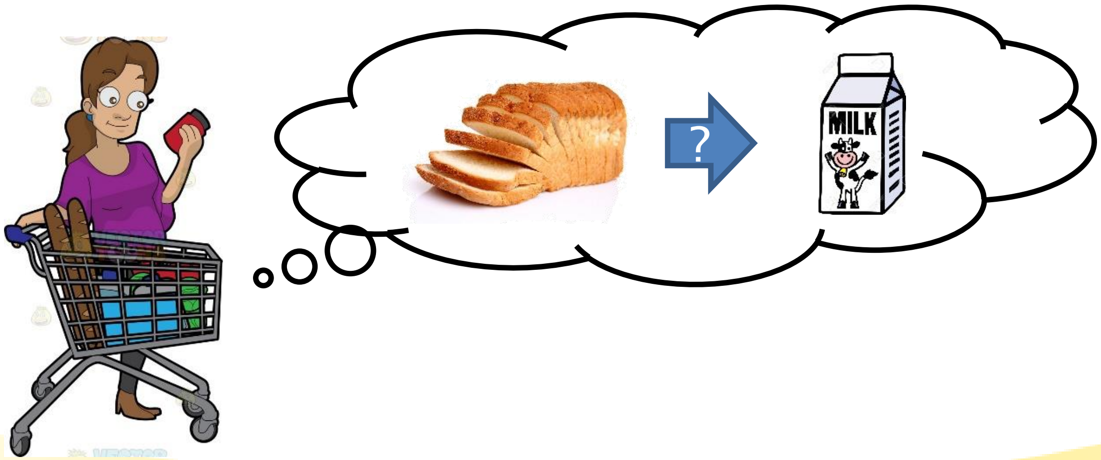
An itemset that contains k items

TID	Items
t1	Bread, Jelly, Peanut Butter
t2	Bread, Peanut Butter
t3	Bread, Milk, Peanut Butter
t4	Beer, Bread
t5	Beer, Milk

Association Rule

Association Rule

Given a set of items $I = \{I_1, I_2, \dots, I_m\}$ and a database of transactions $\mathrm{D} = \{t_1, t_2, \dots, t_m\}$ where $t_i = \{t_{i1}, t_{i2}, \dots, t_{ik}\}$ and $I_{ij} \epsilon I$, an ASSOCIATION RULE is an implication of the form $X \Rightarrow Y$ where $X, Y \subset I$ are sets of items called the itemsets and $X \cap Y = \emptyset$



Support and Confidence

Definition: Support count and Support

The **support count (σ)** for an association rule $X \Rightarrow Y$ is the frequency of occurrence of an itemset

bread --> Peanut butter
3

The **support (s)** for an association rule $X \Rightarrow Y$ is the precentage of transactions in the database that contain $X \cup Y$

bread --> Peanut butter
3/5 = 60%

Definition: Confidence / strength (α)

The **confidence / strength (α)** for an association rule $X \Rightarrow Y$ is the ration of the number of transactions that contain $X \cup Y$ to the number of transactions that contain X

bread --> Peanut butter
3/4 = 75%

pusilkom ui
quality results

EXAMPLE

TID	Items
t1	Bread, Jelly, Peanut Butter
t2	Bread, Peanut Butter
t3	Bread, Milk, Peanut Butter
t4	Beer, Bread
t5	Beer, Milk

Determine the support and confidence of the rules:

1. $Bread \Rightarrow PeanutButter$
2. $\{Bread, Milk\} \Rightarrow PeanutButter$

confidence 100%

Association Rule Problem

Association Rule Problem

Given a set of items $I = \{I_1, I_2, \dots, I_m\}$ and a database of transactions $\mathrm{D} = \{t_1, t_2, \dots, t_m\}$ where $t_i = \{t_{i1}, t_{i2}, \dots, t_{ik}\}$ and $I_{ij} \epsilon I$, the **association rule problem** is to identify all association rules $X \Rightarrow Y$ with a minimum support and confidence.

The <u>input</u> of the problem is the values of (s, α)

Definition : A large (frequent) itemset

An itemset whose number of occurences is above a threshold, s.

pusilkom ui
quality results

Association Rule Problem (cont'd)

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

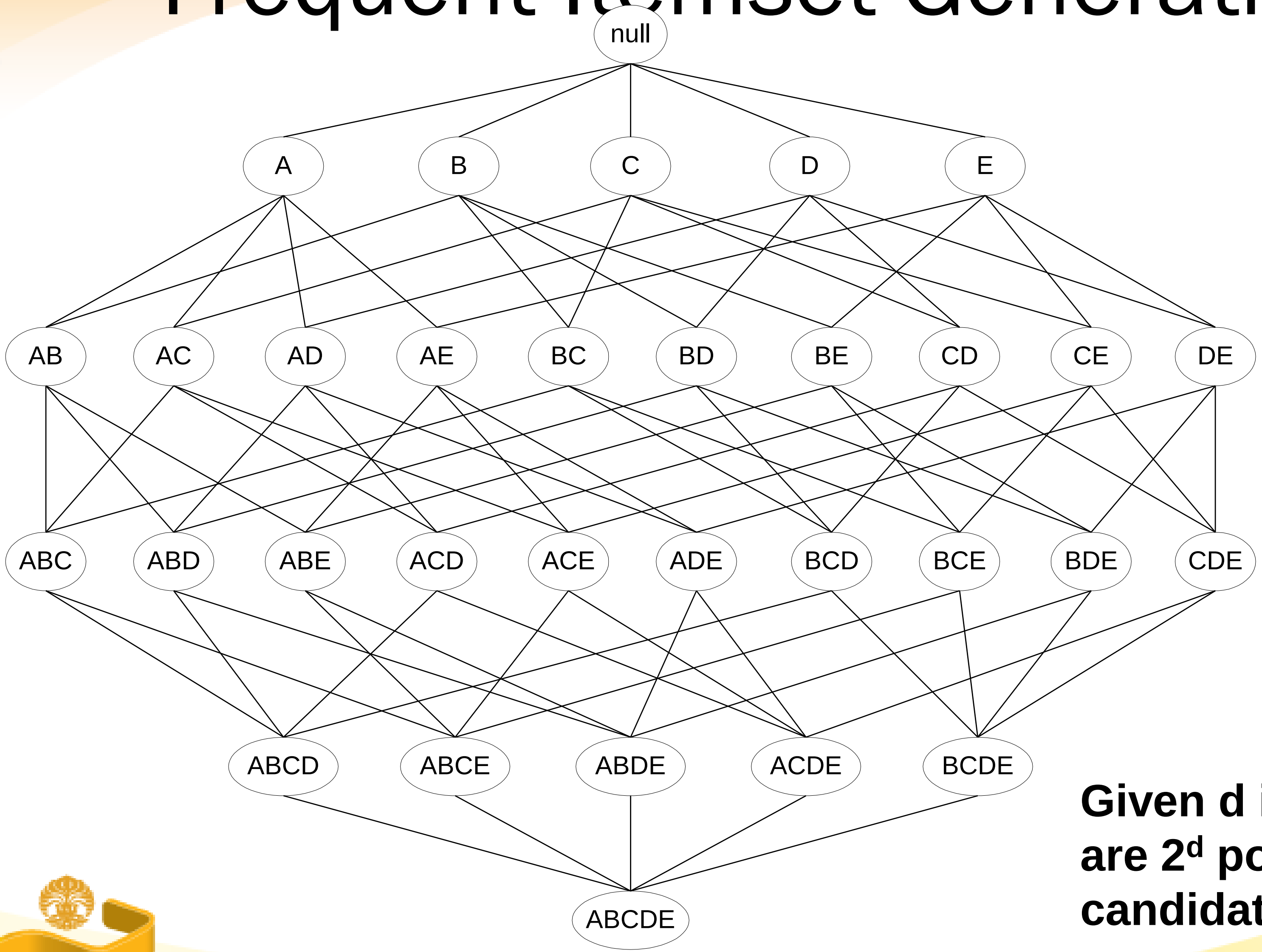
- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minimum support* and *minimum confidence* thresholds
  - Given a set of items of size m, there are 2^m-1 candidate subsets (all possible subsets except an empty set) ⇒ **Computationally prohibitive!**

Association Rule Problem (cont'd)

- Two-step approach:
  1. Frequent Itemset Generation
     - Generate all itemsets whose support ≥ minsup   expensive
  2. Rule Generation
     - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is still computationally expensive

pusilkom ui
quality results

Frequent Itemset Generation



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

pusilkom ui
quality results

Frequent Itemset Generation

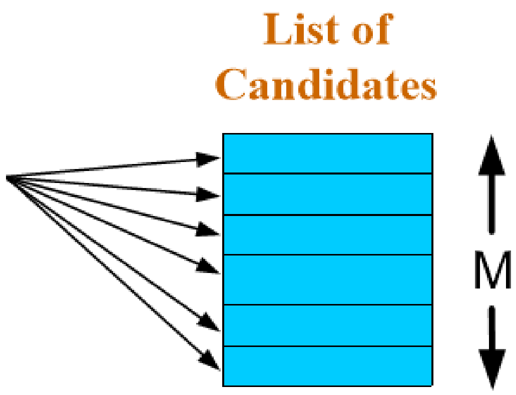
- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database

Transactions

TID	Items
t1	Bread, Jelly, Peanut Butter
t2	Bread, Peanut Butter
t3	Bread, Milk, Peanut Butter
t4	Beer, Bread
t5	Beer, Milk

N

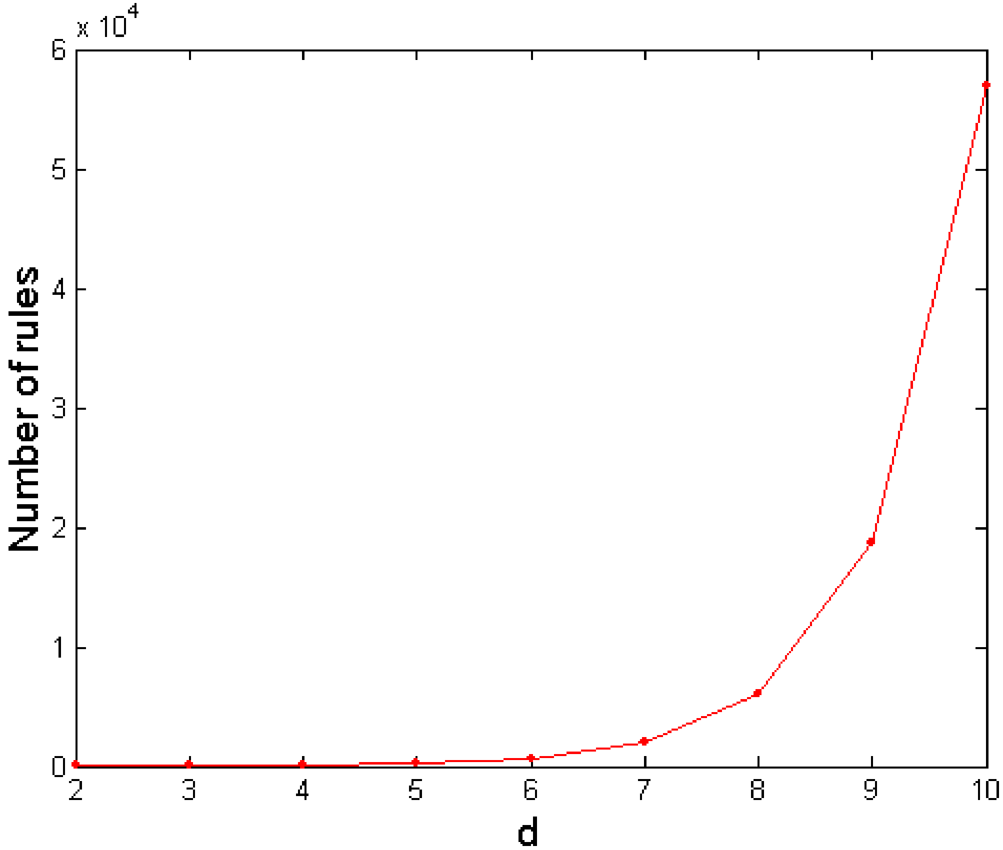
w



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

Computational Complexity

- Given d unique items:
  - Total number of itemsets = 2^d
  - Total number of possible association rules:



$$R = \sum_{k=1}^{d-1}\left[\binom{d}{k}\times\sum_{j=1}^{d-k}\binom{d-k}{j}\right]$$

$$= 3^d - 2^{d+1} + 1$$

If d=6, R = 602 rules

Frequent Itemset Generation Strategies

- Reduce the number of candidates (M)
  - Complete search: $M=2^d$
  - Use pruning techniques to reduce M
- Reduce the number of transactions (N)
  - Reduce size of N as the size of itemset increases
  - Used by Direct Hashing and Pruning Algorithm and vertical-based mining algorithms
- Reduce the number of comparisons (NM)
  - Use efficient data structures to store the candidates or transactions
  - No need to match every candidate against every transaction

pusilkom ui
quality results

APRIORI ALGORITHM

Reducing Number of Candidates

mengurangi jumlah kandidat

- 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

pusilkom ui
quality results

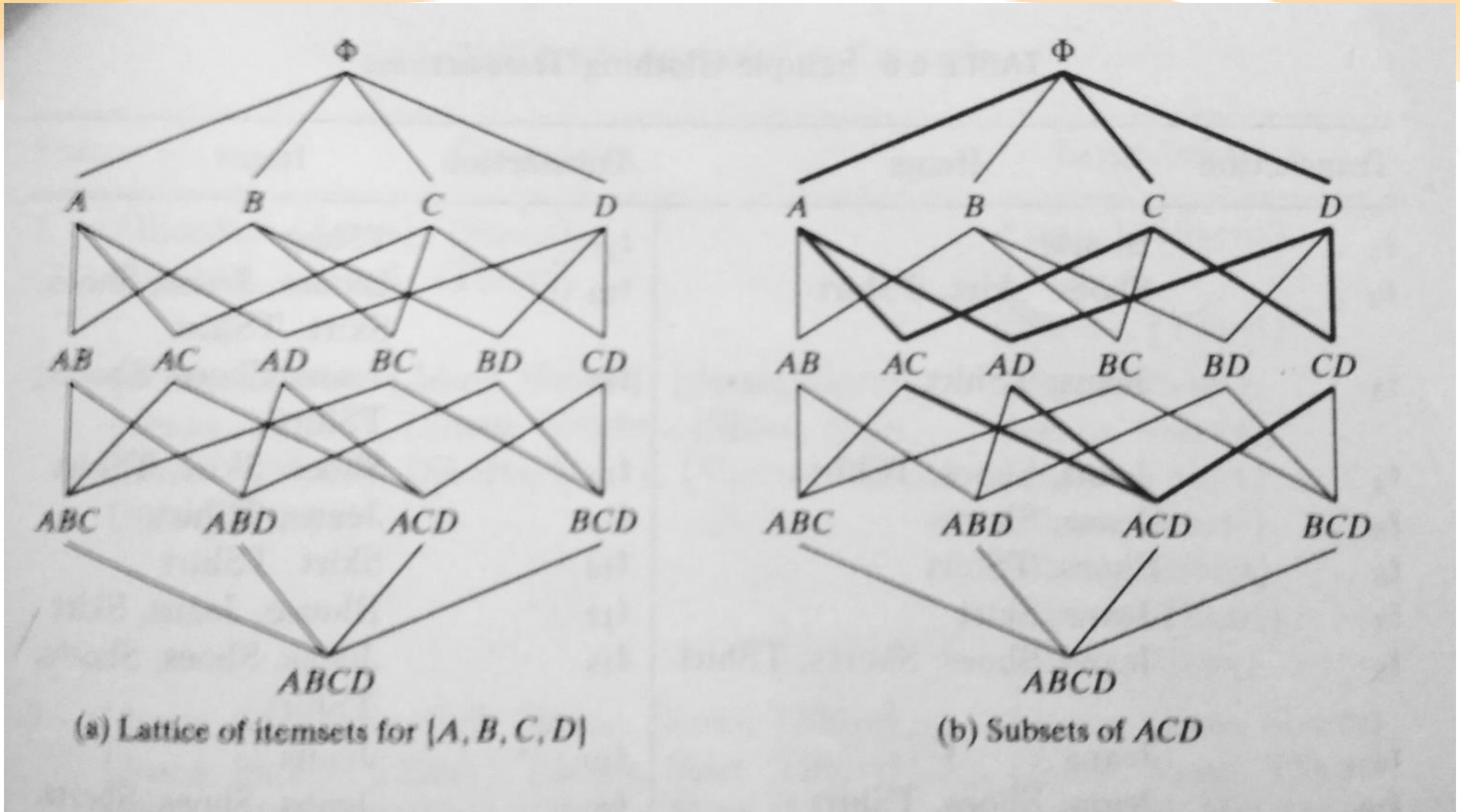


FIGURE 6.1: Downward closure.

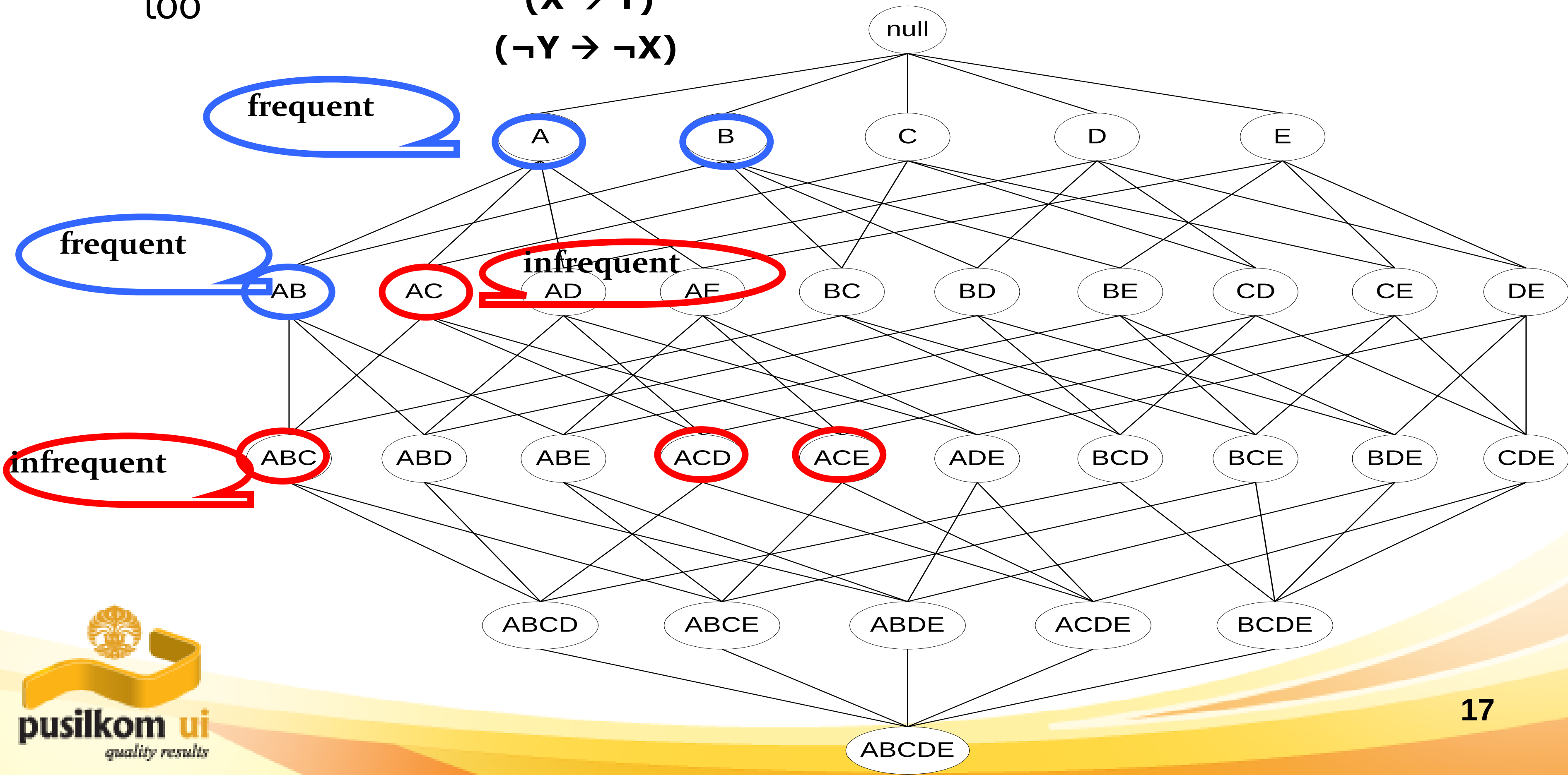
Any subset of a large itemset must be large

Apriori Principle

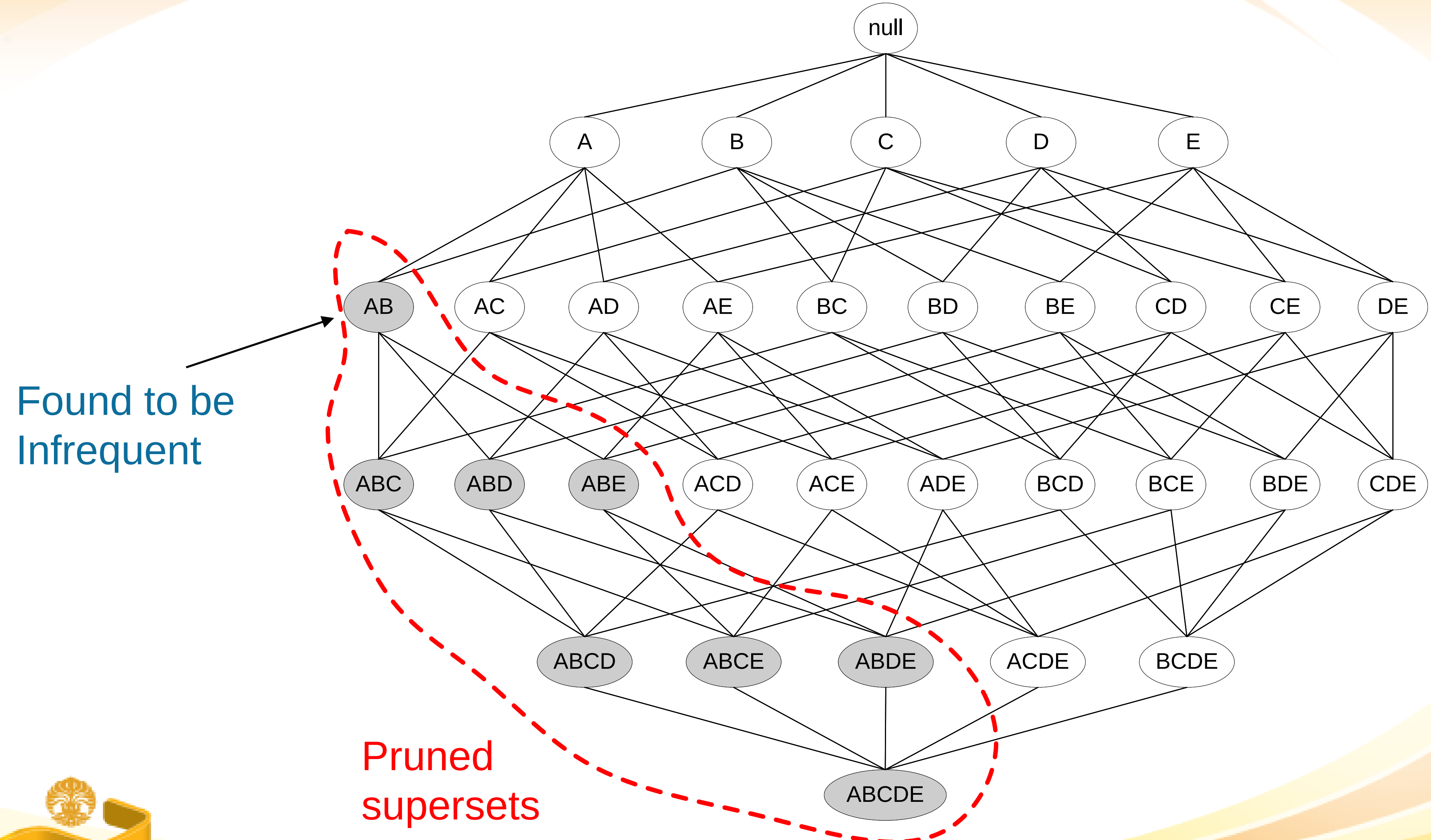
- If an itemset is frequent, then all of its subsets must also be frequent
- If an itemset is infrequent, then all of its supersets must be infrequent too

$(X \rightarrow Y)$

$(\neg Y \rightarrow \neg X)$



Illustrating Apriori Principle



pusilkom ui
quality results

Illustrating Apriori Principle

- Sample Data 2

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**

Illustrating Apriori Principle (cont'd)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Items (1-itemsets)

Itemset	Count
{Bread,Milk}	**3**
{Bread,Beer}	**2**
{Bread,Diaper}	**3**
{Milk,Beer}	**2**
{Milk,Diaper}	**3**
{Beer,Diaper}	**3**

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	**3**

If every subset is considered,

$${}^6C_1 + {}^6C_2 + {}^6C_3 = 41$$

With support-based pruning,

$$6 + 6 + 1 = 13$$

pusilkom ui

quality results

Apriori Algorithm

- Method:
  - Let k=1
  - Generate frequent itemsets of length 1
  - Repeat until no new frequent itemsets are identified
    - Generate length (k+1) candidate itemsets from length k frequent itemsets
    - Prune candidate itemsets containing subsets of length k that are infrequent
    - Count the support of each candidate by scanning the DB
    - Eliminate candidates that are infrequent, leaving only those that are frequent

Example
A database has five transactions. Let the min sup = 50% and min conf = 80%.

Database

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E

Solution
Step 1: Find all Frequent Itemsets

Example

A database has five transactions. Let the min sup = 50% and min conf = 80%.

Database

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E

Solution

Step 1: Find all Frequent Itemsets

Database

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E

L_1

1-Itemsets	Support
{A}	2
{B}	3
{C}	3
{E}	3

Join

C_2

Item	Support
{A B}	1
{A C}	2
{A E}	1
{B C}	2
{B E}	3
{C E}	2

Prune

L_2

2-Itemsets	Support
{A C}	2
{B C}	2
{B E}	3
{C E}	2

Join

C_3

itemset
{B C E}

Prune

L_3

3-Itemsets	Support
{B C E}	2

Frequent Itemset:

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

Step 2: Generate strong association rules from the frequent itemsets

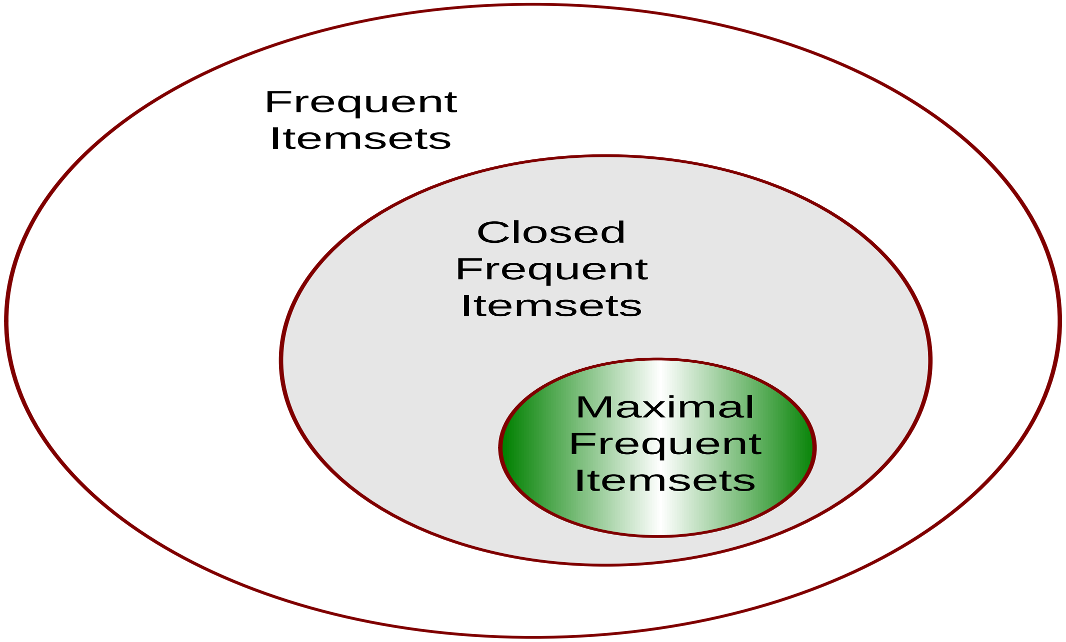
Example
A database has five transactions. Let the min sup = 50% and min conf = 80%.

Database

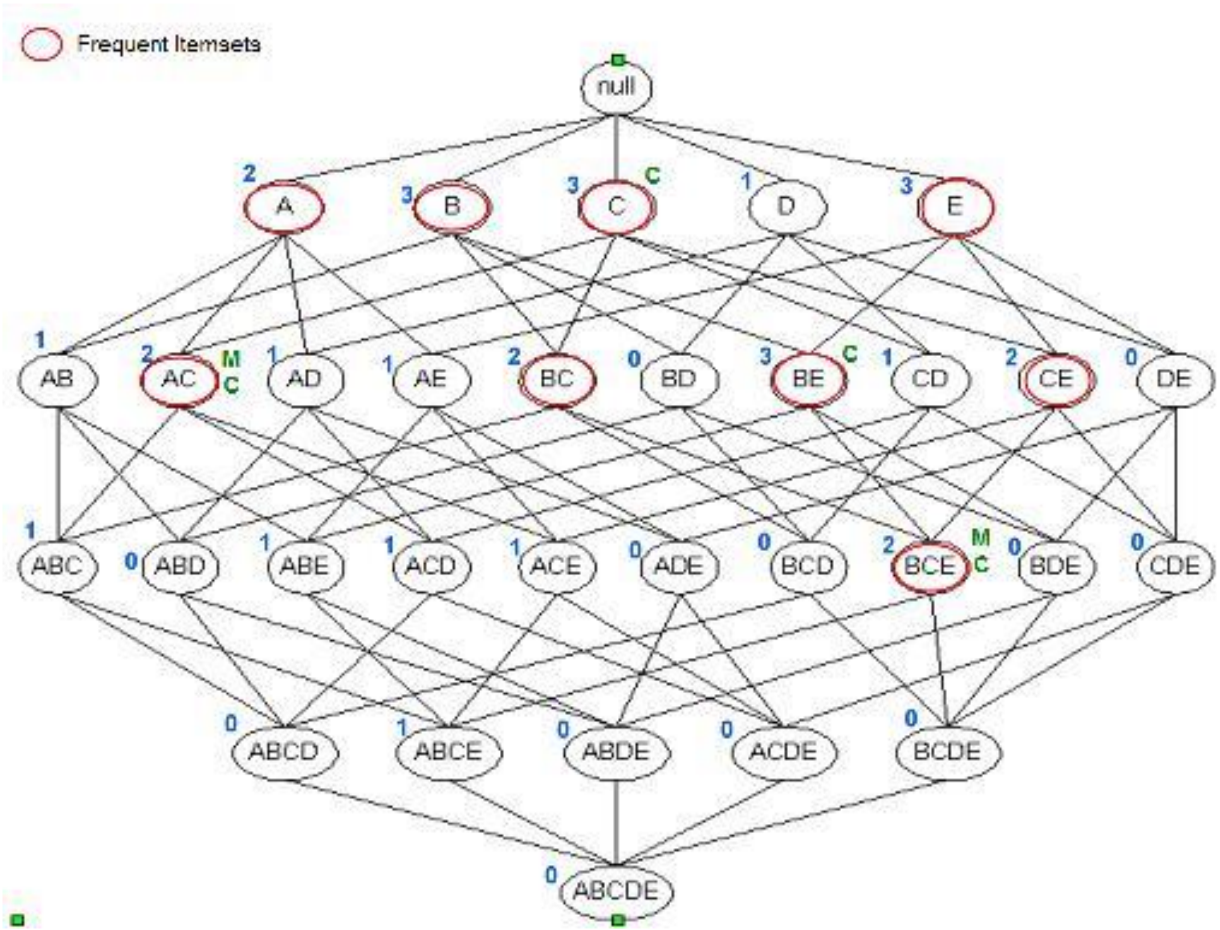
TID	Items
100	A C D
200	B C E
300	A B C E
400	B E

Rules	Support (X Y)	Support(X)	Confidence
{A} -> {C}	2	2	100
{B} -> {C}	2	3	66.6666667
{B} -> {E}	3	3	100
{C} -> {E}	2	3	66.6666667
{B} -> {C E}	2	3	66.6666667
{C} -> {B E}	2	3	66.6666667
{E} -> {B C}	2	3	66.6666667
{C} -> {A}	2	3	66.6666667
{C} -> {B}	2	3	66.6666667
{E} -> {B}	3	3	100
{E} -> {C}	2	3	66.6666667
{C E} -> {B}	2	2	100
{B E} -> {C}	2	3	66.6666667
{B C} -> {E}	2	2	100

Closed Itemset: support of all parents are not equal to the support of the itemset.
Maximal Itemset: all parents of that itemset must be infrequent.



Itemset {c} is closed as support of parents (supersets) {A C}:2, {B C}:2, {C D}:1, {C E}:2 not equal support of {c}:3. And the same for {A C}, {B E} & {B C E}.
Itemset {A C} is maximal as all parents (supersets) {A B C}, {A C D}, {A C E} are infrequent. And the same for {B C E}.



Algorithms to find frequent pattern

- **Apriori:** uses a generate-and-test approach – generates candidate itemsets and tests if they are frequent
  - Generation of candidate itemsets is expensive (in both space and time)
  - Support counting is expensive
    - Subset checking (computationally expensive)
    - Multiple Database scans (I/O)
- **FP-Growth:** allows frequent itemset discovery without candidate generation. Two step:
  - 1.Build a compact data structure called the FP-tree
    - 2 passes over the database
  - 2.extracts frequent itemsets directly from the FP-tree
    - Traverse through FP-tree

pusilkom ui
quality results

Summary

- Association Analysis
- Finding frequent pattern
- Apriori Algorithm

ANY QUESTION?

pusilkom ui
quality results