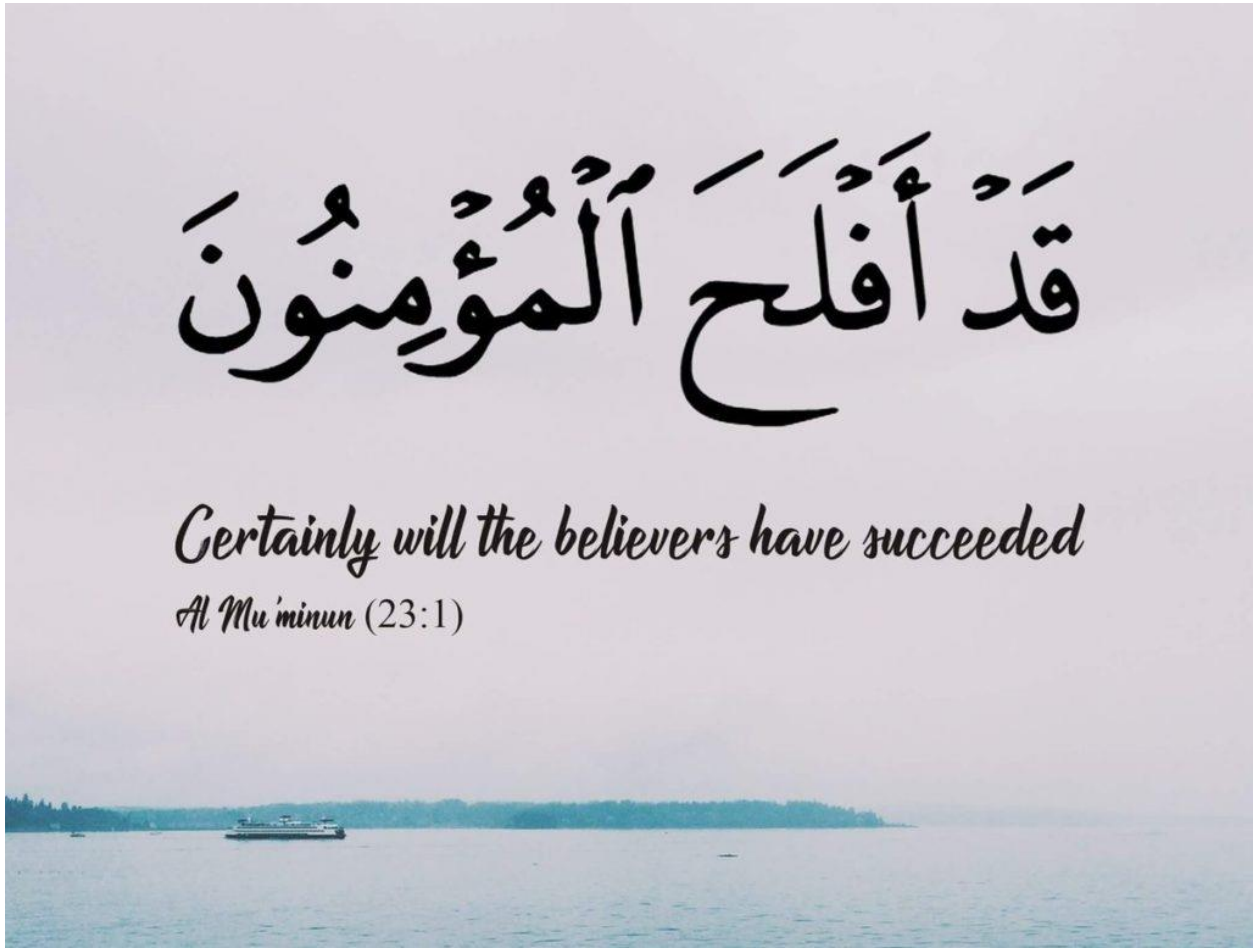
Data Mining

ASSOCIATION RULE MINING
FINDING FREQUENT PATTERN
APRIORI ALGORITHM

Prof. Dr. Hikmat Ullah Khan
Department of Information Technology

UNIVERSITY OF SARGODHA, SARGODHA

Lesson from Holy Quran



MARKET BASKET ANALYSIS

98% of people who purchased items A and B also purchased item C

Observe the table

1.Check the frequent items?

2. Which items are frequent wrt other items?

3. This makes Pattern.

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

What Is Frequent Pattern Analysis?

- Frequent Occurances: a pattern
- **What is a Pattern?**
- (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- **History**
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of two concepts, we study here
  - Finding Frequent Itemsets
  - Finding Association Rule.

What Is Frequent Pattern Analysis?

- **Motivation: Finding inherent regularities in data**
  - What products were often purchased together?—
    - E.g., Milk and Bread
  - What are the subsequent purchases after buying a PC?
    - Peripheral devices
  - What kinds of DNA are sensitive to this new drug?
  - Can you add more examples???

Applications

- Basket data analysis,
  - cross-marketing/sale campaign analysis,
- Document Analysis
- Web Analysis
  - Usage Analysis (Log (click stream) analysis)
  - Content Analysis (C0-Occurance of)
  - Structure Analysis (
- Expert Group Finding
- Social Network Analysis
  - Similar Interest Finding
  - Terrorist Network

Document analysis

- **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

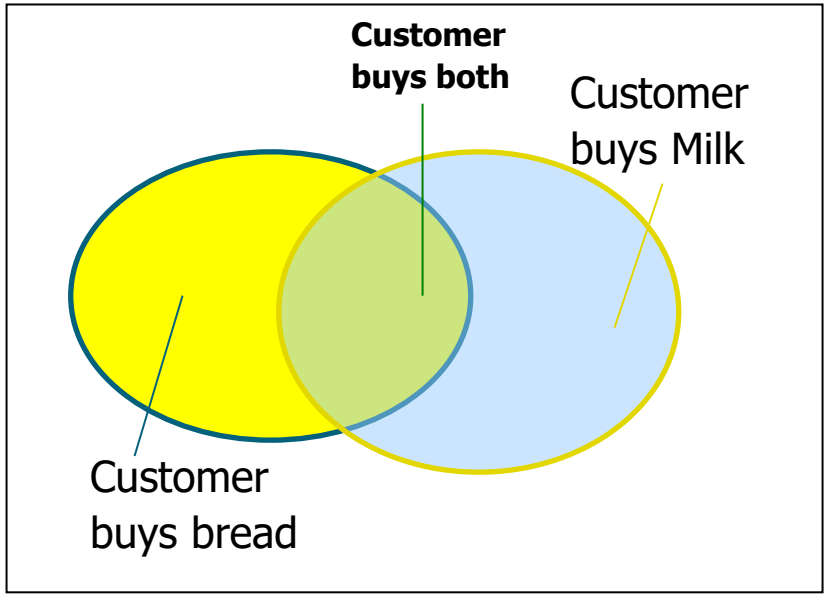
Support & Confidence

- Itemset X = $\{x_1, \ldots, x_k\}$
- Find all the rules *X* → *Y* with minimum support and confidence
  - support, *s*, probability that a transaction contains X ∪ Y
  - confidence, *c*, conditional probability that a transaction having X also contains *Y*

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

Let sup_{min} *= 50%,* $conf_{min}$ *= 50%*
Freq. Pat.: {A:3, B:3, D:4, E:3, AD:3}
Association rules:

A → *D* (60%, 100%)
D → *A* (60%, 75%)



An example

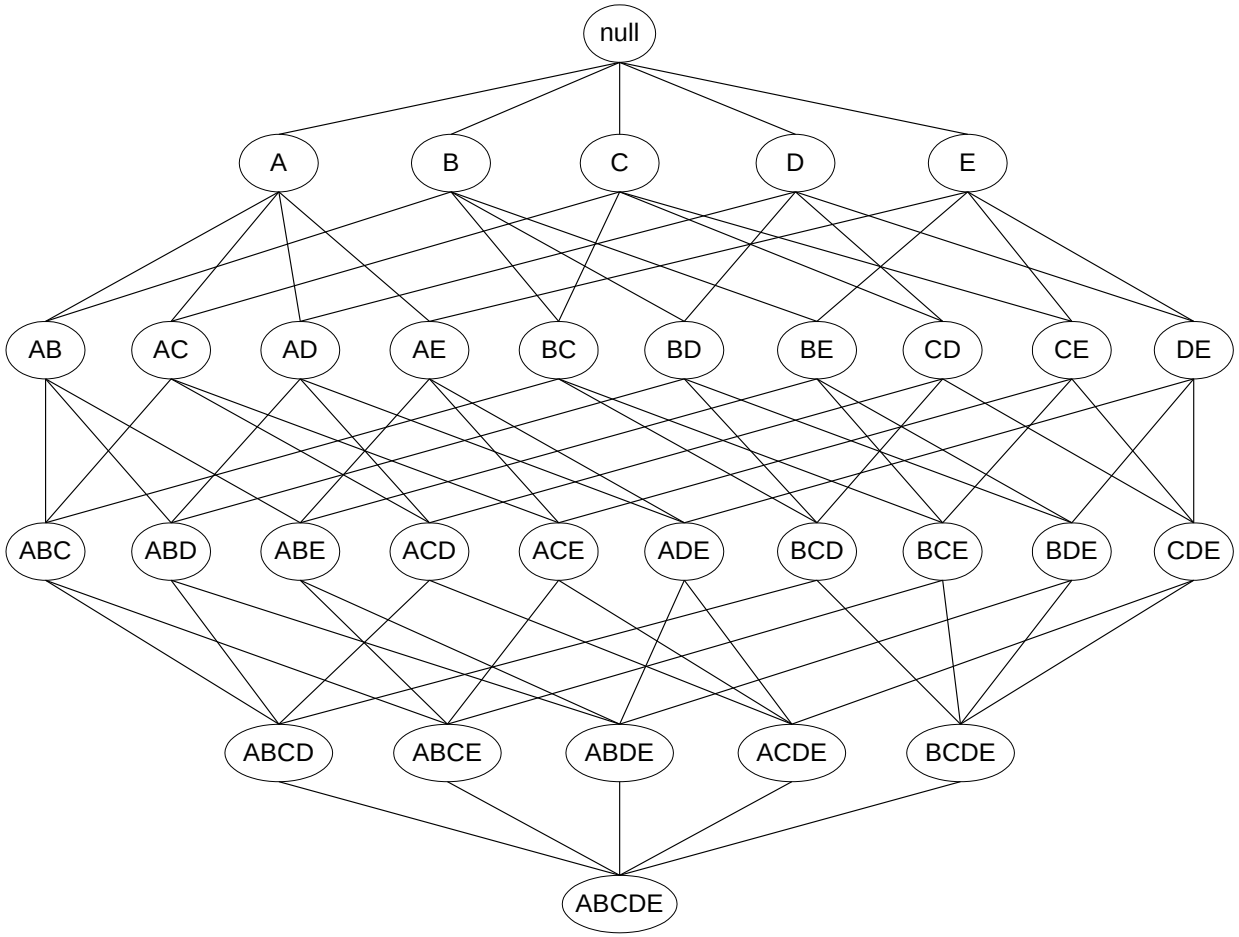
- Take Transaction data
- An example frequent *itemset*: {Chicken, Clothes, Milk}

Find the support of this frequent itemset

Assume any rule and find its Confidence

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

How many itemsets are there?



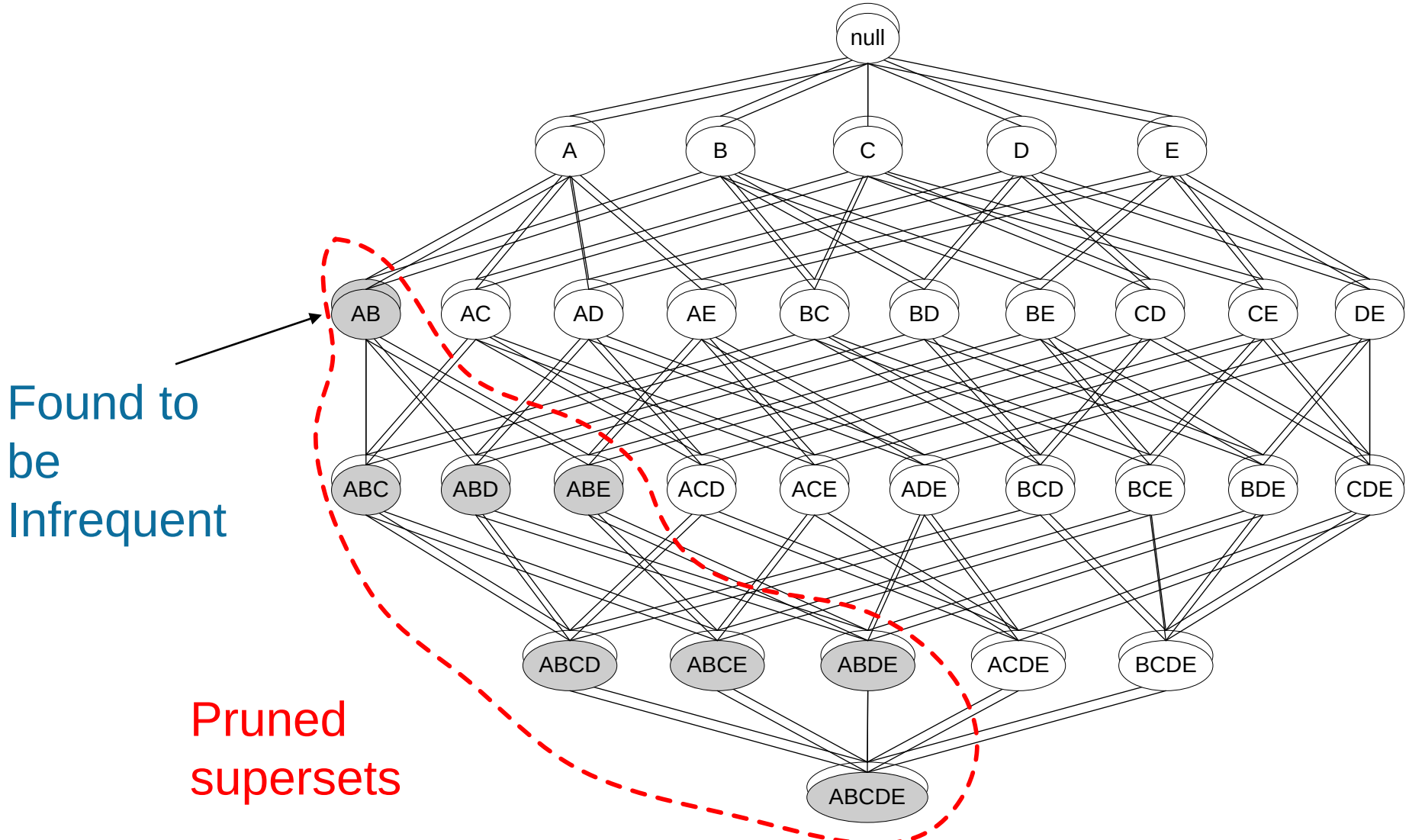
Given d items, there are 2^d possible itemsets

Apriori Principle

- How to avoid to many computations?
- Solution:
  - Apriori Pruning principle

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested! Method:

Illustrating the Apriori principle



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

1^{st} scan →

$Sup_{min} = 2$

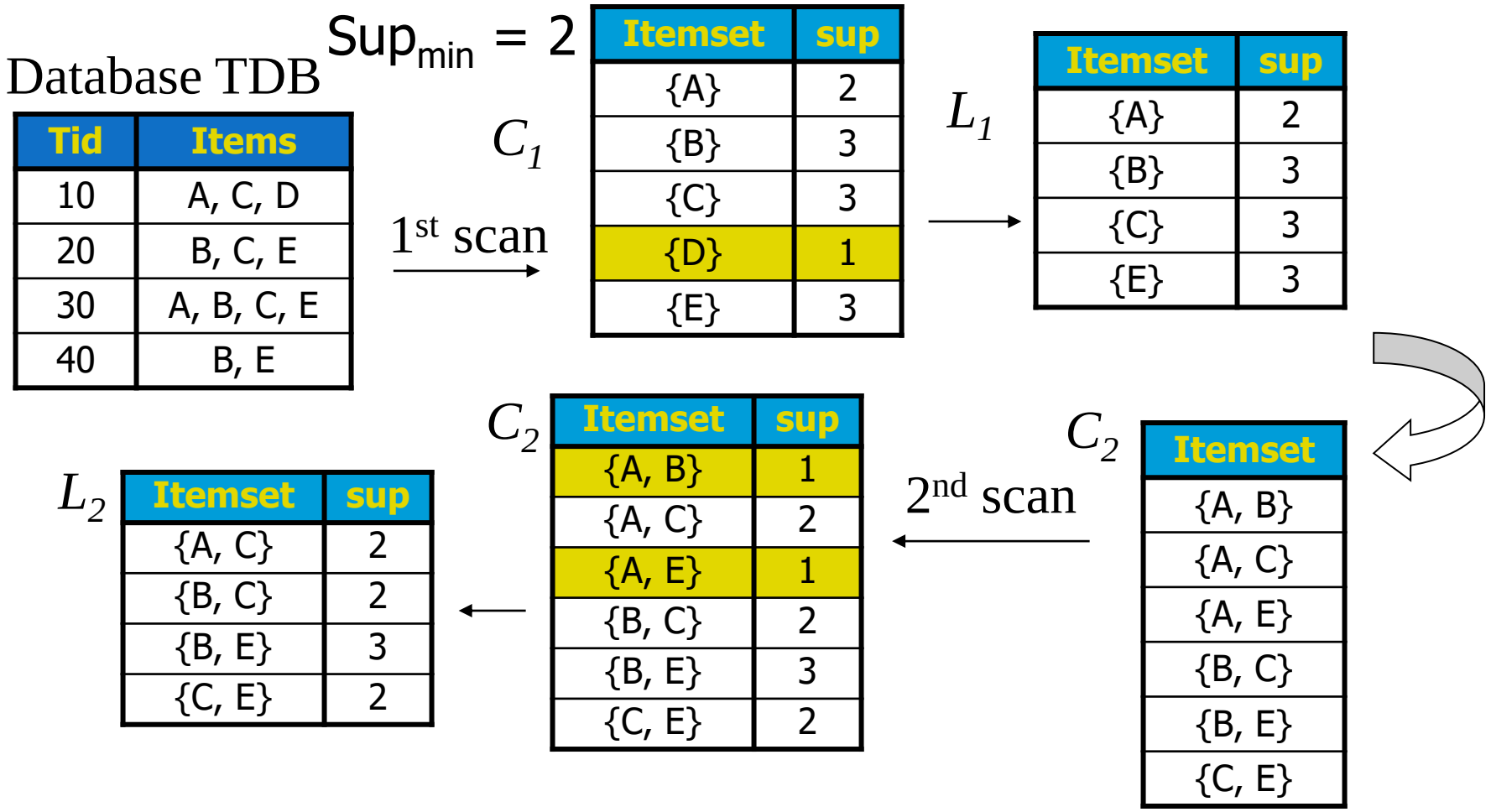
C_1

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

L_1 →

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

The Apriori Algorithm—An Example



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

$Sup_{min} = 2$

C_1

1st scan

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

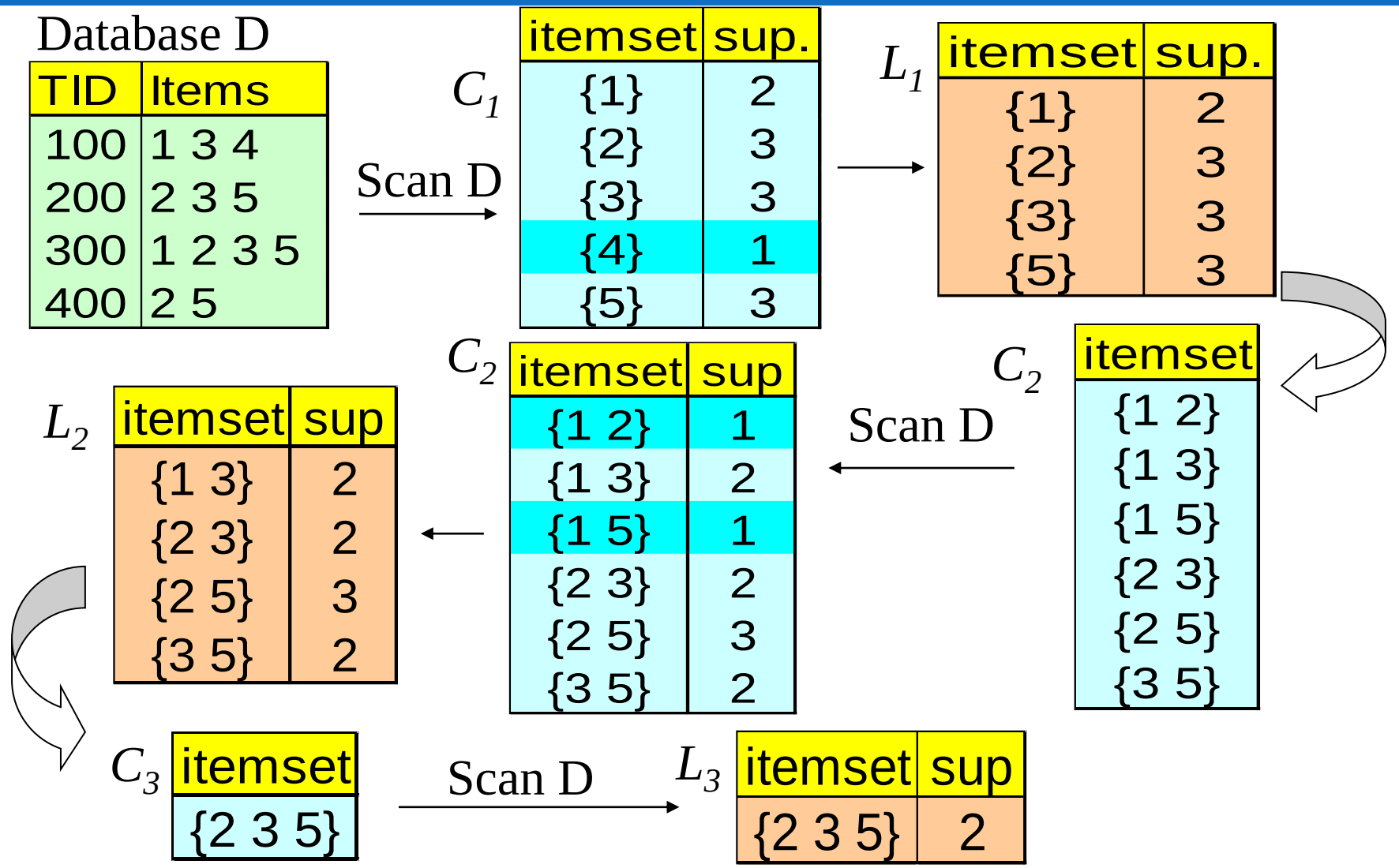
Itemset	sup
{B, C, E}	2

The Apriori Algorithm — Exercise

Database D

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

The Apriori Algorithm — Exercise Solved



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 !=∅; 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

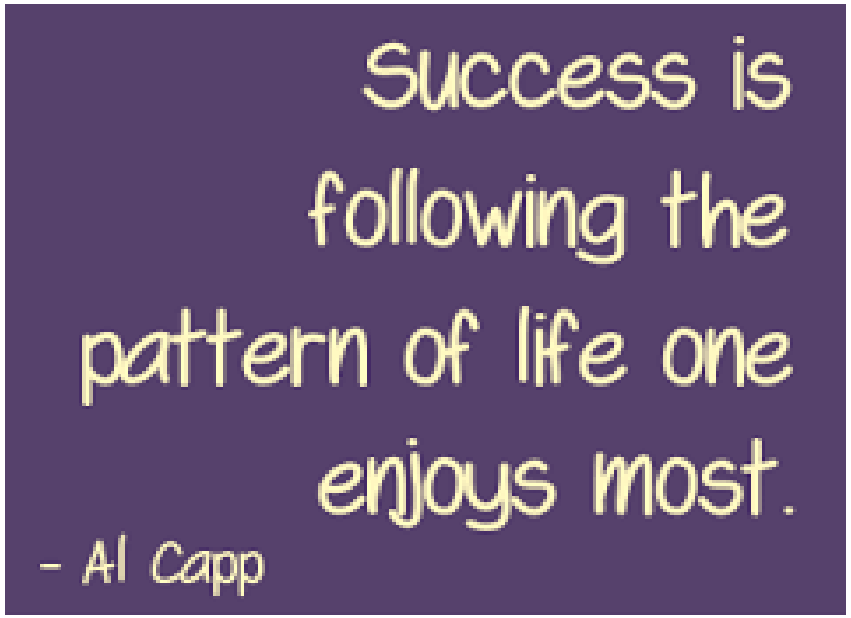
  L_{k+1} = candidates in C_{k+1} with min_support

  end

return $\cup_k L_k$;

Implementation of Apriori: Optional but recommended

- Options
- Languages
  - Python/R
- API/Libraries
- Python/R


Success is following the pattern of life one enjoys most.
- Al Capp