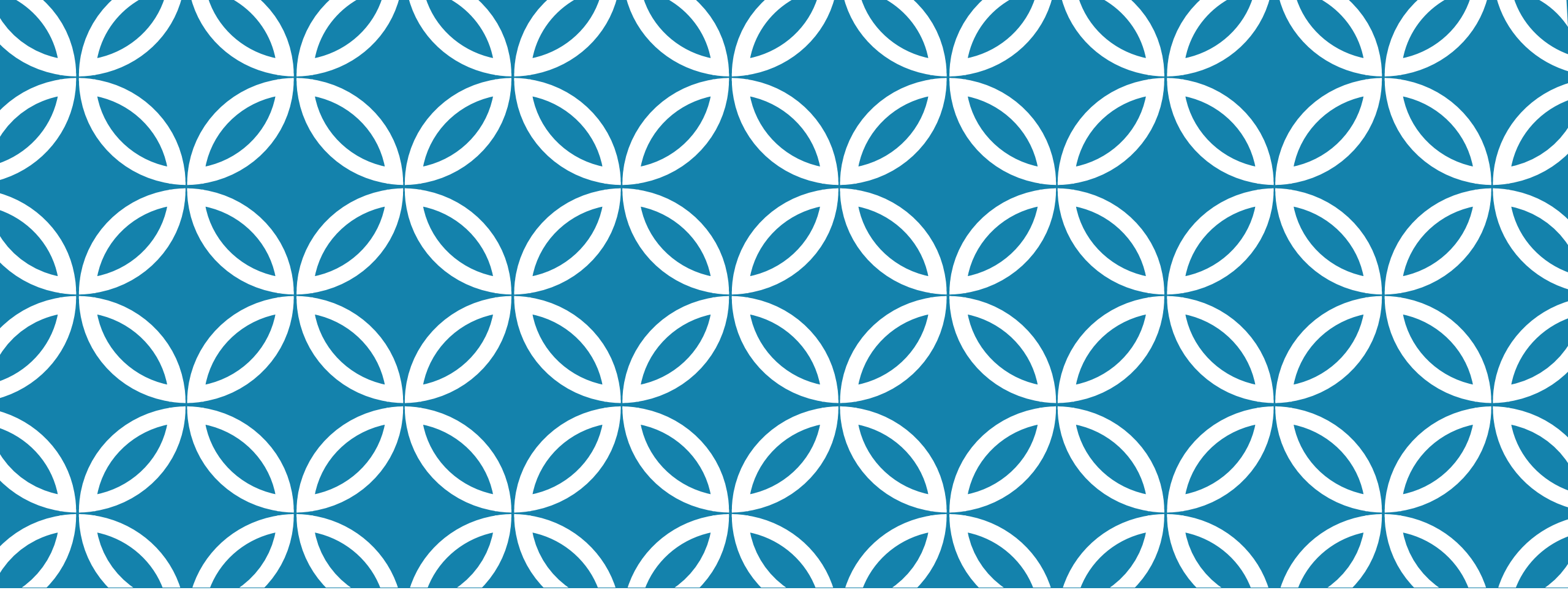


CS5228 LECTURE 4: ASSOCIATION RULE MINING

Bryan Hooi

School of Computing

National University of Singapore

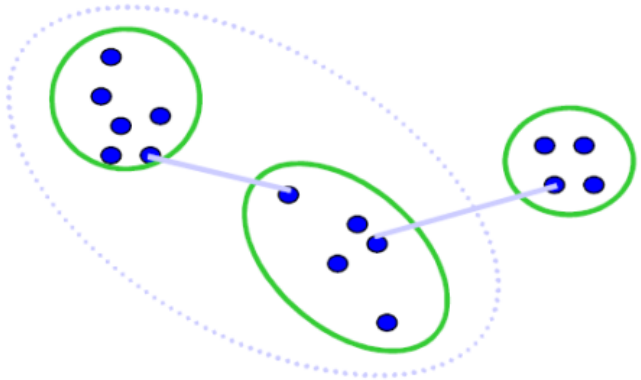
Slide credits: Wang Wei, Ng See Kiong, Wynne Hsu

ANNOUNCEMENTS

- **HW1 and project released**: refer to Canvas > Assignments for details. In particular there is a "Final Project Description" document with info about the project. There is also a survey being done if you want us to randomly match you to a group (1st round: 6-13 Feb; 2nd round: 13-20 Feb)

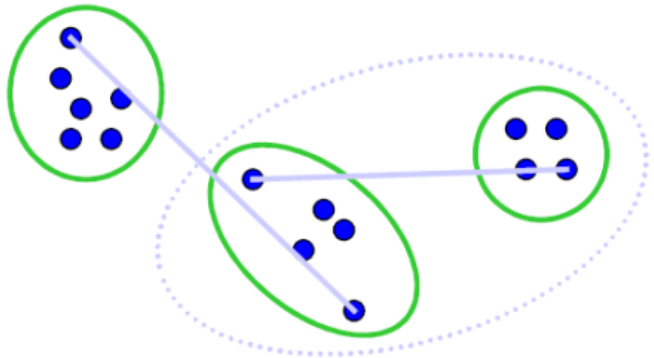
REVIEW: SINGLE, COMPLETE, AVERAGE LINKAGE

Single Linkage



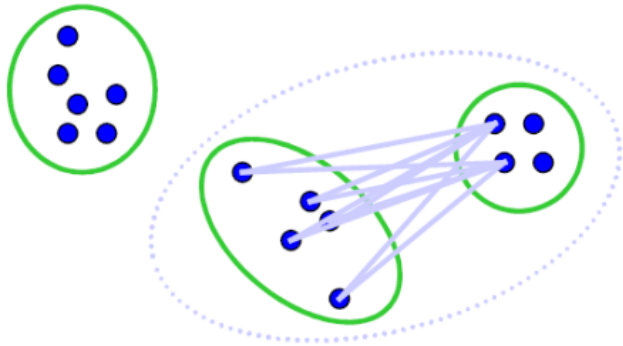
$$\min_{p \in C_i, q \in C_j} d(p,q)$$

Complete Linkage



$$\max_{p \in C_i, q \in C_j} d(p,q)$$

Average Linkage



$$\operatorname{avg}_{p \in C_i, q \in C_j} d(p,q)$$

REVIEW: DBSCAN (CORE, BORDER, OUTLIERS)

Core points are points with at least ***MinPts*** neighbors (including themselves) within their radius of ε.

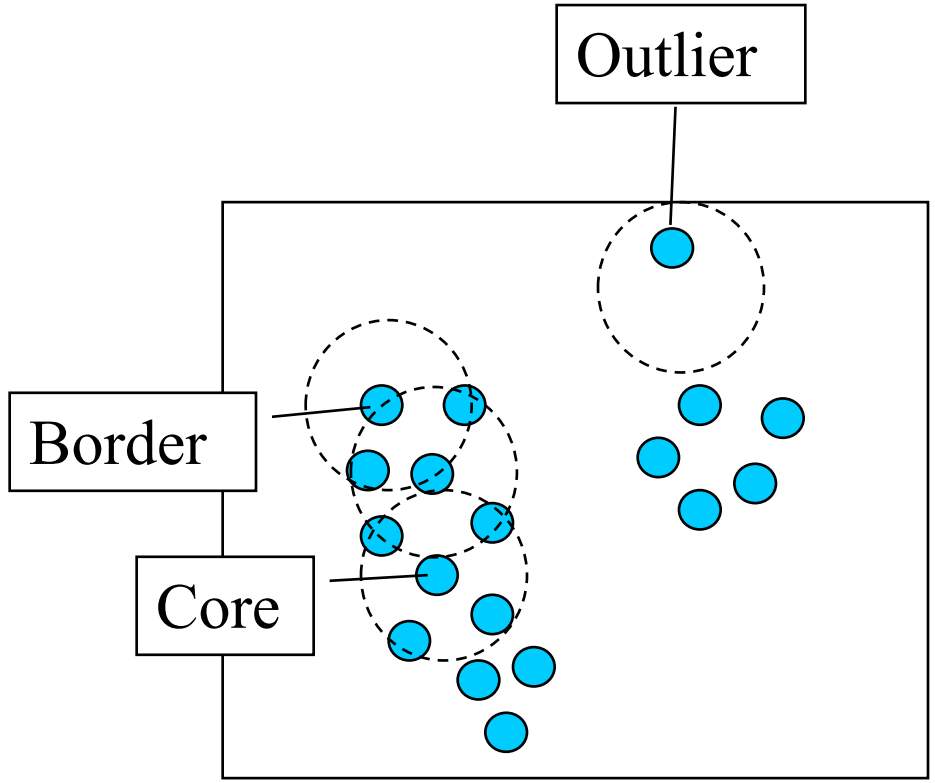
- (these will form the 'interior' of clusters)

Border points are non-core points with at least 1 core point in its neighborhood.

- (these will form the 'border' of clusters)

Outliers are all other points.

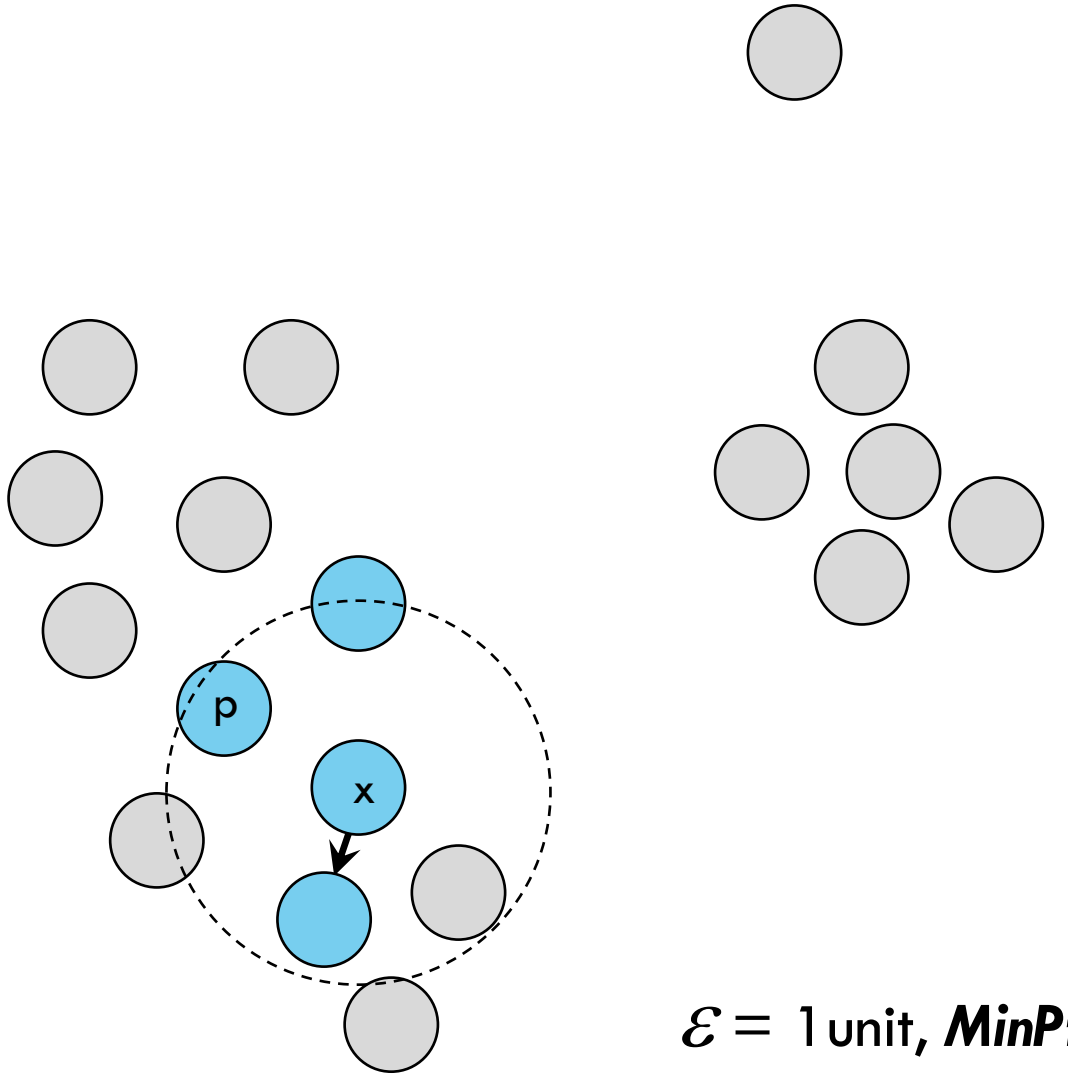
- (these will not belong to any cluster)



ε = 1unit, MinPts = 5

REVIEW: DBSCAN ALGORITHM

1. Arbitrarily select a unexplored core point *p*.
2. Recursively **`explore`** starting at p.
3. When **`exploring`** a node x:
   - Add points in x's neighborhood to the same cluster as x
   - Recursively **`explore`** all unexplored **core** points in x's neighborhood



$\mathcal{E} = 1$ unit, ***MinPts*** $= 5$

REVIEW: CLUSTER EVALUATION

External (Ground Truth)

Cluster Purity

Cluster	Label
Blue → 0	0
Blue → 0	0
Red → 1	1

Rand Index

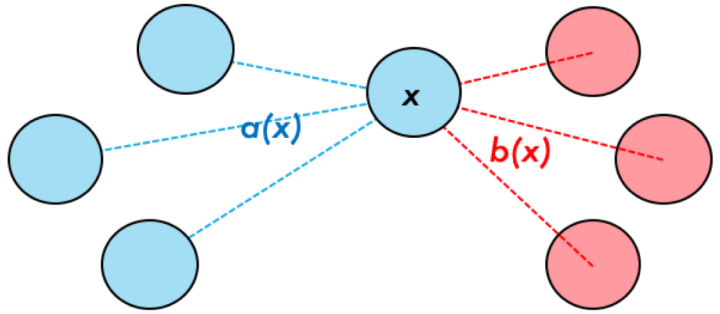
same

Cluster	Label
Blue	0
Blue	1
Red	1

same

Internal

Silhouette Coefficient



$$s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}}$$

Source: https://www.rinapiccolo.com/piccolo-cartoons/

ASSOCIATION RULE MINING

Bryan Hooi

School of Computing

National University of Singapore

ONLINE STORES: 'FREQUENTLY BOUGHT TOGETHER'

Frequent Itemsets

Frequently Bought Together

Price for all three: $74.20

Add all three to Cart | Add all three to Wish List

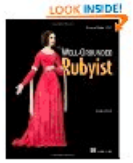
Show availability and shipping details

- **This item:** Beginning Ruby: From Novice to Professional (Expert's Voice in Open Source) by Peter Cooper Paperback $27.78
- Learn to Program, Second Edition (The Facets of Ruby Series) by Chris Pine Paperback $16.94
- Ruby on Rails Tutorial: Learn Web Development with Rails (2nd Edition) (Addison-Wesley Professional Ruby … by Michael Hartl Paperback $29.48

Association Rules

Customers Who Bought This Item Also Bought

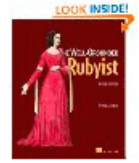
Learn to Program, Second Edition (The Facets of…
Chris Pine
★★★★½ 42
Paperback
$16.94 Prime



The Well-Grounded Rubyist
› David A. Black
★★★★★ 39
Paperback
$32.49 Prime

Ruby on Rails Tutorial: Learn Web Development…
› Michael Hartl
★★★★☆ 70
Paperback
$29.48 Prime

The Ruby Programming Language
› David Flanagan
★★★★½ 74
Paperback
$26.35 Prime



The Well-Grounded Rubyist
› David A. Black
★★★★½ 19
#1 Best Seller in Ruby Programming Computer
Paperback
$29.67 Prime

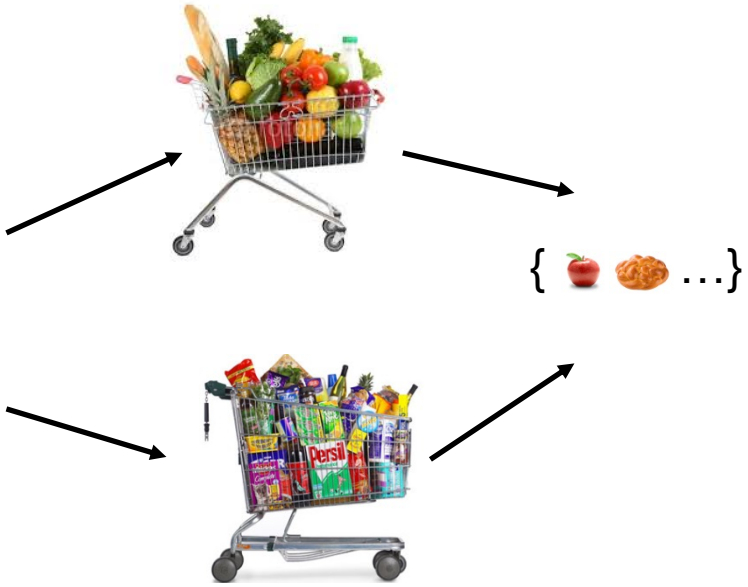
DATA MINING FOLKLORE

In 1992, a manager of a retail consulting group analyzed 1.2 million “market baskets” from Osco Drug stores.

The analysis discovered that “**between 5:00 and 7:00 p.m. consumers frequently bought beer and diapers together**".

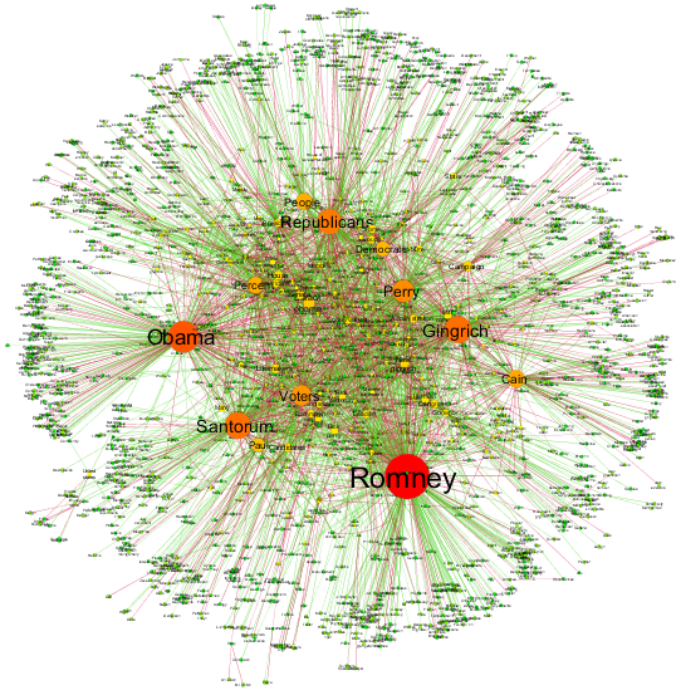
What do my customers buy?

What items are bought together?



OTHER APPLICATIONS

Bioinformatics: find frequently co-occurring proteins / genes



Text mining: each document can be treated as a 'basket' of words

https://www.patheos.com/blogs/altmuslim/2018/05/9-health-benefits-and-tips-for-beneficial-ramadan-and-otherwise-fasting/diabetes-wordcloud/

OVERVIEW

1. **Basic Concepts**
   - What are frequent itemsets and association rules?

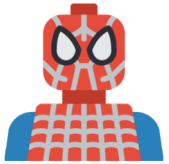
2. **Problem**
   - Brute Force Approach

3. **Apriori Algorithm**
   - Frequent Itemset Mining
   - Association Rule Mining

OVERVIEW

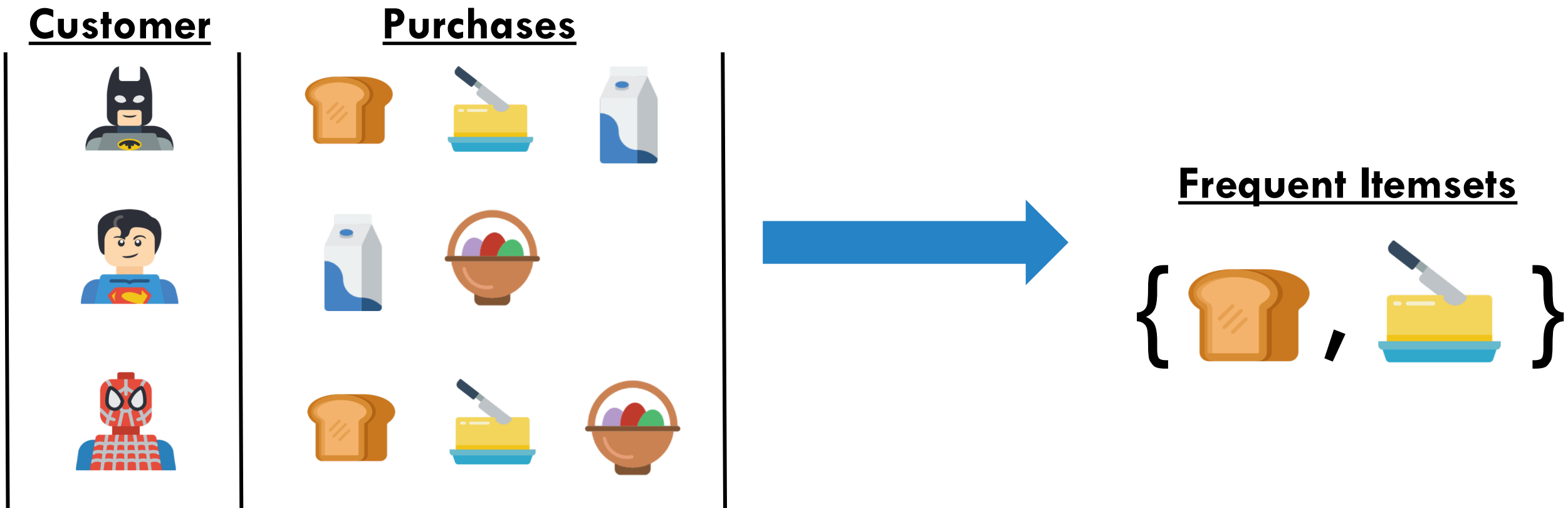
1. **Basic Concepts**
   - What are frequent itemsets and association rules?
2. **Problem**
   - Brute Force Approach
3. **Apriori Algorithm**
   - Frequent Itemset Mining
   - Association Rule Mining

THE MARKET-BASKET MODEL

Customer	Purchases
	
	
	

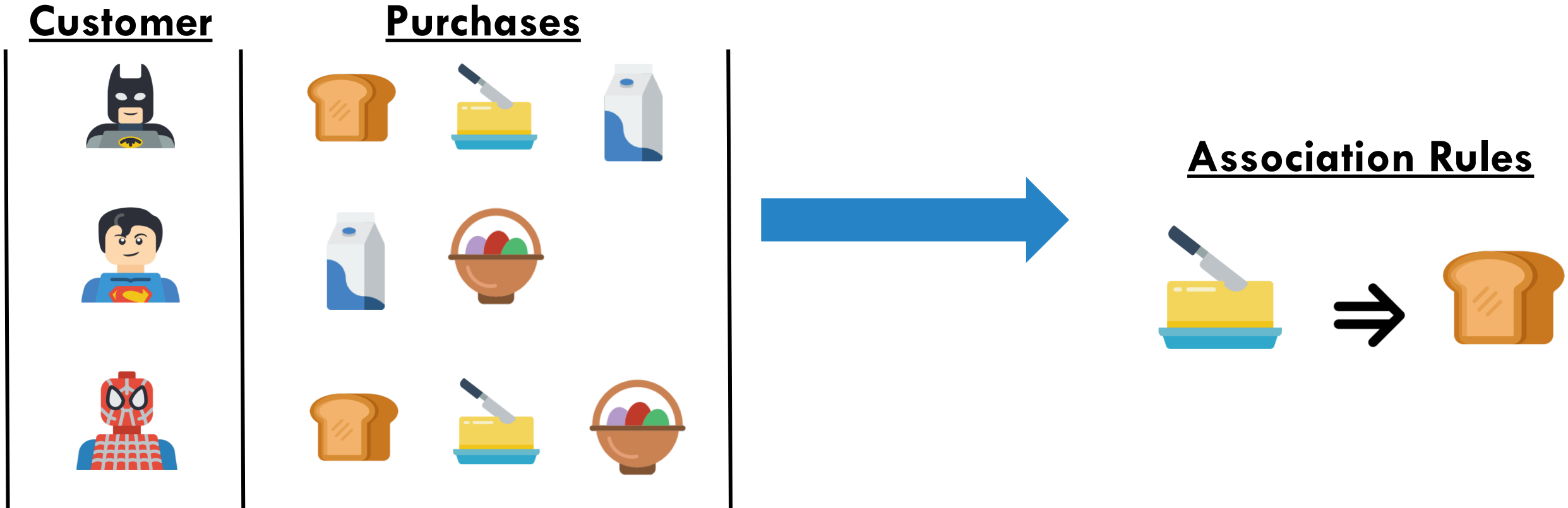
TWO GOALS OF MARKET BASKET ANALYSIS

1. **Frequent Itemsets:** what combinations of items are popular?



TWO GOALS OF MARKET BASKET ANALYSIS

1. **Frequent Itemsets:** what combinations of items are popular?
2. **Association Rules:** do any sets of items typically imply the presence of any other items?



MARKET-BASKET MODEL: ITEMS & TRANSACTIONS

- Consist of a set of **items**, e.g. products in a supermarket, words in a document, etc.

- **Transactions / baskets** are sets of items purchased by each customer

Transaction ID

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

DEFINITIONS: ITEMSET, K-ITEMSET

- **Itemset**: a subset of items
  - E.g. {Milk, Bread, Eggs}
- **K-itemset**: a subset containing k items
  - E.g. {Milk, Bread, Eggs} is a 3-itemset

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

DEFINITIONS: SUPPORT COUNT, SUPPORT

- **Support Count**: the no. of transactions containing an itemset
  - E.g. SC({Milk, Bread, Diaper}) = 2
- **Support**: the fraction of transactions containing an itemset
  - E.g. Supp({Milk, Bread, Diaper}) = 2/5

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

DEFINITION: FREQUENT ITEMSETS

- **Frequent Itemset**: an itemset whose support is greater than or equal to a minimum (*minsup*) threshold
  - E.g. if *minsup* = 2/5, then {Bread, Milk, Diaper} is a frequent itemset.

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

DEFINITION: ASSOCIATION RULE

An **association rule** is an expression of the form X → Y, where X and Y are itemsets, meaning that "buying X tends to imply buying Y"

- E.g. {Milk, Diaper} → {Bread}
- Note: implications means co-occurrence, not causality!

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

DEFINITION: SUPPORT AND CONFIDENCE

Support of an association rule X ⟶ Y is the fraction of transactions containing **all** its items:

- E.g. Supp({Milk, Diaper} ⟶ {Bread}) = 2/5
- E.g. Supp({Milk, Bread} ⟶ {Diaper}) = 2/5
- This is the same as the support of the itemset XUY.

Union of itemsets X and Y

$$\text{Supp}(X \longrightarrow Y) = \text{Supp}(XUY) = \text{SC}(XUY) \,/\, N$$

Total no. of items

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

DEFINITION: SUPPORT AND CONFIDENCE

Confidence of an association rule $X \rightarrow Y$ is the probability of Y being in a basket given that X is:

- E.g. Conf({Milk, Diaper} → {Bread}) = 2/3

$$Conf(X \rightarrow Y) = P(Y \mid X)$$

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

DEFINITION: SUPPORT AND CONFIDENCE

Confidence of an association rule X ⟶ Y is the probability of Y being in a basket given that X is:

- E.g. Conf({Milk, Diaper} ⟶ {Bread}) = 2/3

$$\text{Conf}(X \longrightarrow Y) = P(Y \mid X) = SC(X \cup Y) \,/\, SC(X)$$

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

DEFINITION: SUPPORT AND CONFIDENCE

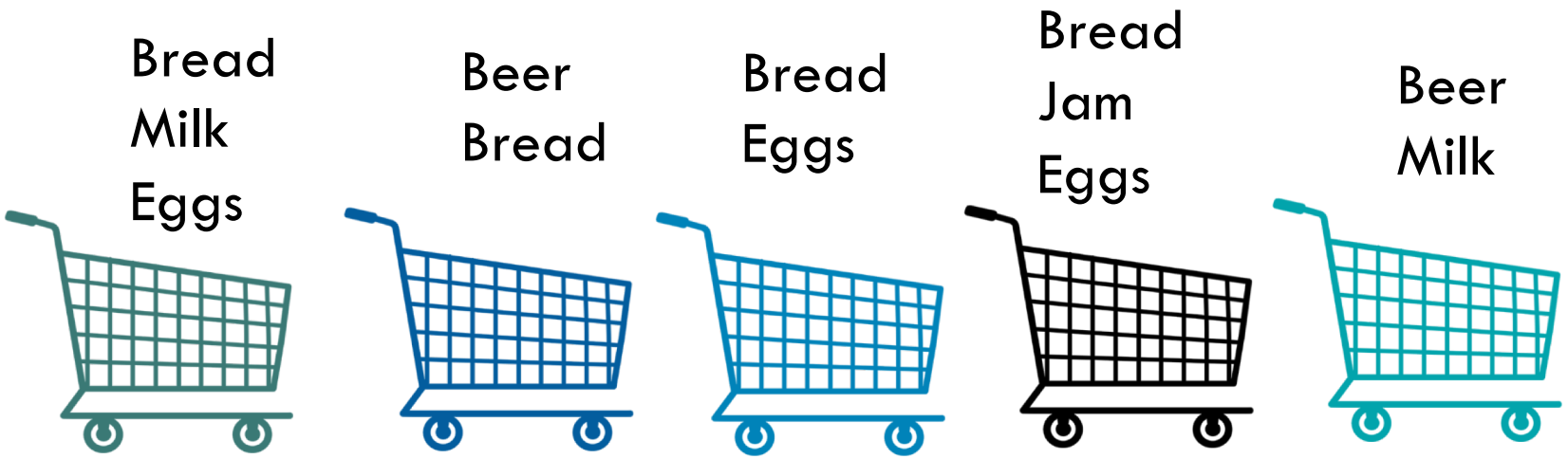
Confidence of an association rule X ⟶ Y is the probability of Y being in a basket given that X is:

- E.g. Conf({Milk, Diaper} ⟶ {Bread}) = 2/3

$$\begin{aligned} Conf(X \rightarrow Y) &= P(Y \mid X) \\ &= SC(X \cup Y) \,/\, SC(X) \\ &= Supp(X \cup Y) \,/\, Supp(X) \end{aligned}$$

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

PRACTICE EXAMPLE

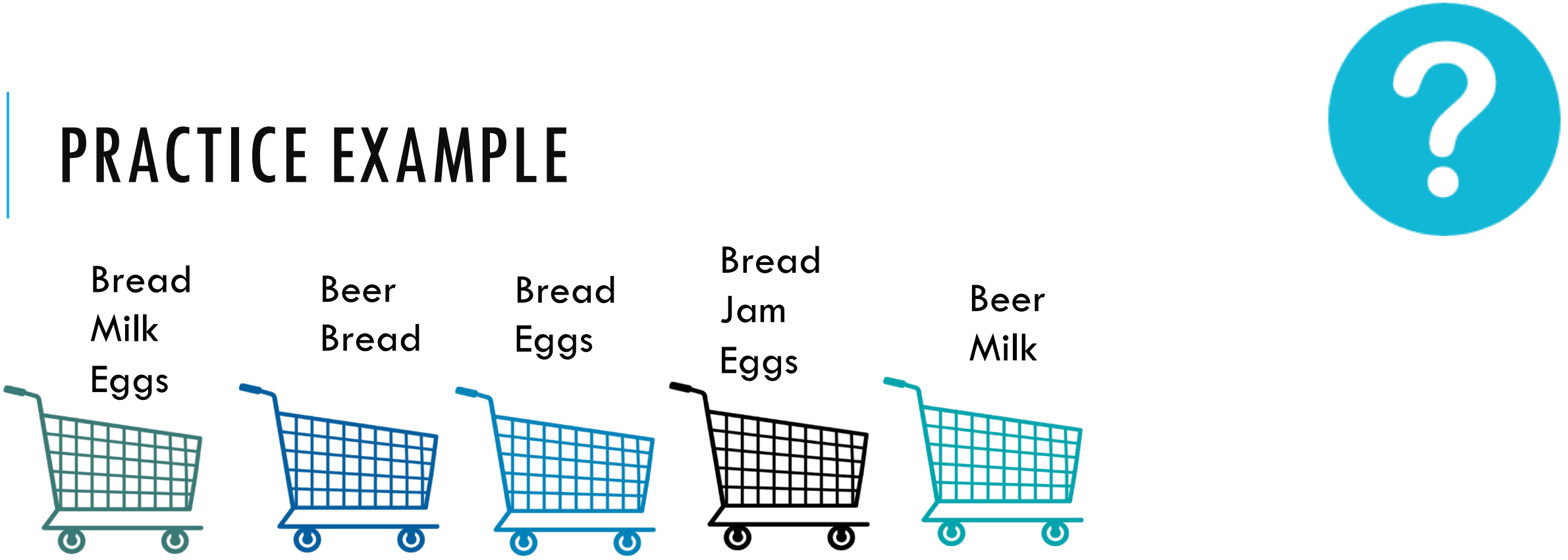


Rule	Supp	Conf
Bread → Eggs	?	?
Eggs → Bread	?	?

PRACTICE EXAMPLE

Rule	Supp	Conf
Bread → Eggs	0.6	0.75
Eggs → Bread	?	?

PRACTICE EXAMPLE



Rule	Supp	Conf
Bread → Eggs	0.6	0.75
Eggs → Bread	0.6	1

OVERVIEW

1. **Basic Concepts**
   - What are frequent itemsets and association rules?

2. **Problem**
   - Brute Force Approach

3. **Apriori Algorithm**
   - Frequent Itemset Mining
   - Association Rule Mining

FREQUENT ITEMSET MINING

Given: a minimum support threshold *minsup*

Find: all itemsets with support ≥ *minsup*

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

minsup = 0.6

Find all frequent itemsets for this threshold.

FREQUENT ITEMSET MINING

Given: a minimum support threshold *minsup*

Find: all itemsets with support ≥ *minsup*

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

minsup = 0.6

Frequent itemsets:

{Diaper, Beer}, {Milk, Diaper}, {Bread, Diaper}, {Milk, Bread}, {Diaper}, {Beer}, {Milk}, {Bread}

ASSOCIATION RULE MINING

Given: a minimum support threshold *minsup*, and a minimum confidence threshold *minconf*

Find: all association rules with support ≥ *minsup*, and confidence ≥ *minconf*

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

minsup = 0.6 minconf = 1

Find all association rules for these thresholds.

ASSOCIATION RULE MINING

Given: a minimum support threshold *minsup*, and a minimum confidence threshold *minconf*

Find: all association rules with support ≥ *minsup*, and confidence ≥ *minconf*

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

minsup = 0.6 minconf = 1

Frequent itemsets:

{Diaper, Beer}, {Milk, Diaper}, {Bread, Diaper}, {Milk, Bread}, {Diaper}, {Beer}, {Milk}, {Bread}

Diaper → Beer, Beer → Diaper,
Milk → Diaper, Diaper → Milk,
Bread → Diaper, Diaper → Bread,
Milk → Bread, Bread → Milk

ASSOCIATION RULE MINING

Given: a minimum support threshold *minsup*, and a minimum confidence threshold *minconf*

Find: all association rules with support ≥ *minsup*, and confidence ≥ *minconf*

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

minsup = 0.6 minconf = 1

Frequent itemsets:

{Diaper, Beer}, {Milk, Diaper}, {Bread, Diaper}, {Milk, Bread}, {Diaper}, {Beer}, {Milk}, {Bread}

Diaper → Beer, Beer → Diaper,
Milk → Diaper, Diaper → Milk,
Bread → Diaper, Diaper → Bread,
Milk → Bread, Bread → Milk

ASSOCIATION RULE MINING: BRUTE FORCE APPROACH

Brute Force Approach:

1. List all rules X $\rightarrow$ Y (e.g. each item can be put into X, Y, or neither)
2. Compute their support and confidence, and filter these rules based on thresholds *minsup*, and *minconf*

- **Problem:** given d unique items, there are close to 3^d rules
- This is computationally prohibitive!

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

minsup = 0.6 minconf = 1

OVERVIEW

1. **Basic Concepts**
   - What are frequent itemsets and association rules?

2. **Problem**
   - Brute Force Approach

3. **Apriori Algorithm**
   - Frequent Itemset Mining
   - Association Rule Mining

Key Challenge

How to automatically and efficiently discover these frequent patterns and association rules?

DECOUPLING CONFIDENCE AND SUPPORT

Given: *minsup*, *minconf*

Find: association rules with support ≥ *minsup*, and confidence ≥ *minconf*

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

Recall: all association rules over the same itemset have the same support!

- E.g. Supp({Milk, Diaper} → {Beer}) = 2/5
  Supp({Milk, Beer} → {Diaper}) = 2/5

So, we can **first** find all itemsets with support ≥ *minsup,* **then** find all rules over them with confidence ≥ *minconf*.

APRIORI ALGORITHM: TWO PART APPROACH

Apriori Algorithm:

Part 1: Frequent Itemset Generation

- Generate all itemsets whose support ≥ *minsup*

Part 2: Rule Generation

- Generate rules from each frequent itemset, where each rule is **a binary partition** of a frequent itemset with confidence ≥ *minconf*

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

Thresholds *minsup*, *minconf*

Frequent Itemsets:
{Bread, Milk}, {Bread, Diaper}, …

Association Rules:
{Bread} → {Milk}, …

APRIORI ALGORITHM: TWO PART APPROACH

Apriori Algorithm:

Part 1: Frequent Itemset Generation

- Generate all itemsets whose support ≥ *minsup*

Part 2: Rule Generation

- Generate rules from each frequent itemset, where each rule is **a binary partition** of a frequent itemset with confidence ≥ *minconf*

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

Thresholds *minsup*, *minconf*

Frequent Itemsets:
{Bread, Milk}, {Bread, Diaper}, …

Association Rules:
{Bread} → {Milk}, …

APRIORI PRINCIPLE (OR ANTI-MONOTONICITY)

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

Supp({Bread, Milk}) = 3/5

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

Supp({Bread, Milk, Diaper}) = 2/5

$\Rightarrow$ Supp({Bread, Milk}) $\geq$ Supp({Bread, Milk, Diaper})

Key Fact: support of a subset is always greater than or equal to that of a superset!

APRIORI PRINCIPLE (OR ANTI-MONOTONICITY)

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

Supp({Bread, Milk}) = 3/5

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

Supp({Bread, Milk, Diaper}) = 2/5

⇒ If Supp({Bread, Milk, Diaper}) ≥ *minsup*:

then Supp({Bread, Milk}) ≥ *minsup*

Key Fact: subset of a frequent itemset must be frequent!

APRIORI PRINCIPLE (OR ANTI-MONOTONICITY)

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

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

Apriori / Anti-monotonicity Principle:

Given itemsets X and Y where $X \subseteq Y$:

- $Supp(X) \geq Supp(Y)$.

APRIORI PRINCIPLE (OR ANTI-MONOTONICITY)

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

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

Apriori / Anti-monotonicity Principle:

Given itemsets X and Y where $X \subseteq Y$:

- Supp(X) $\geq$ Supp(Y).
- If Y is frequent, then X is also frequent.

APRIORI PRINCIPLE (OR ANTI-MONOTONICITY)

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

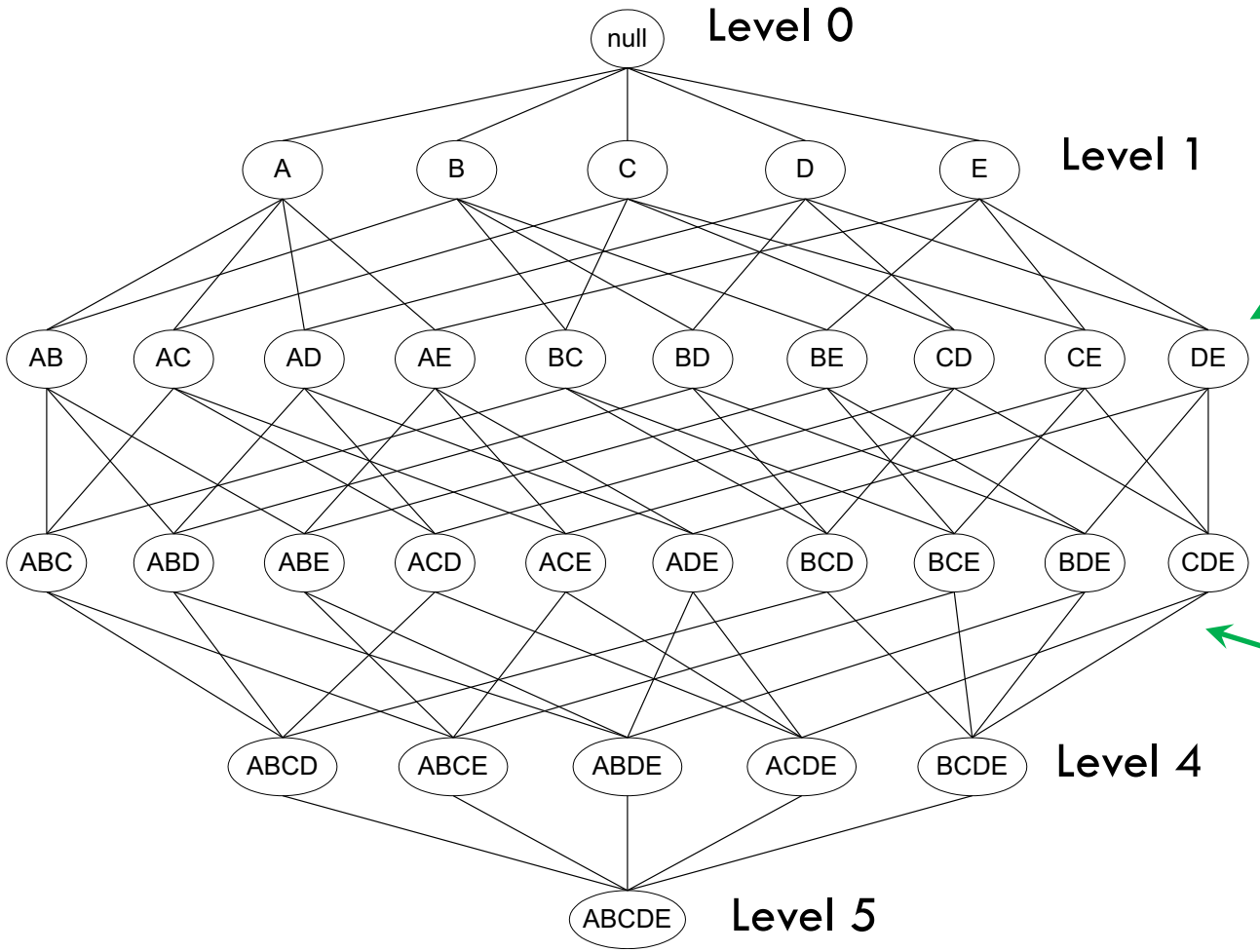
TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Apriori / Anti-monotonicity Principle:

Given itemsets X and Y where $X \subseteq Y$:

- $Supp(X) \geq Supp(Y)$.
- If Y is frequent, then X is also frequent.
- If X is not frequent, then Y is also not frequent.

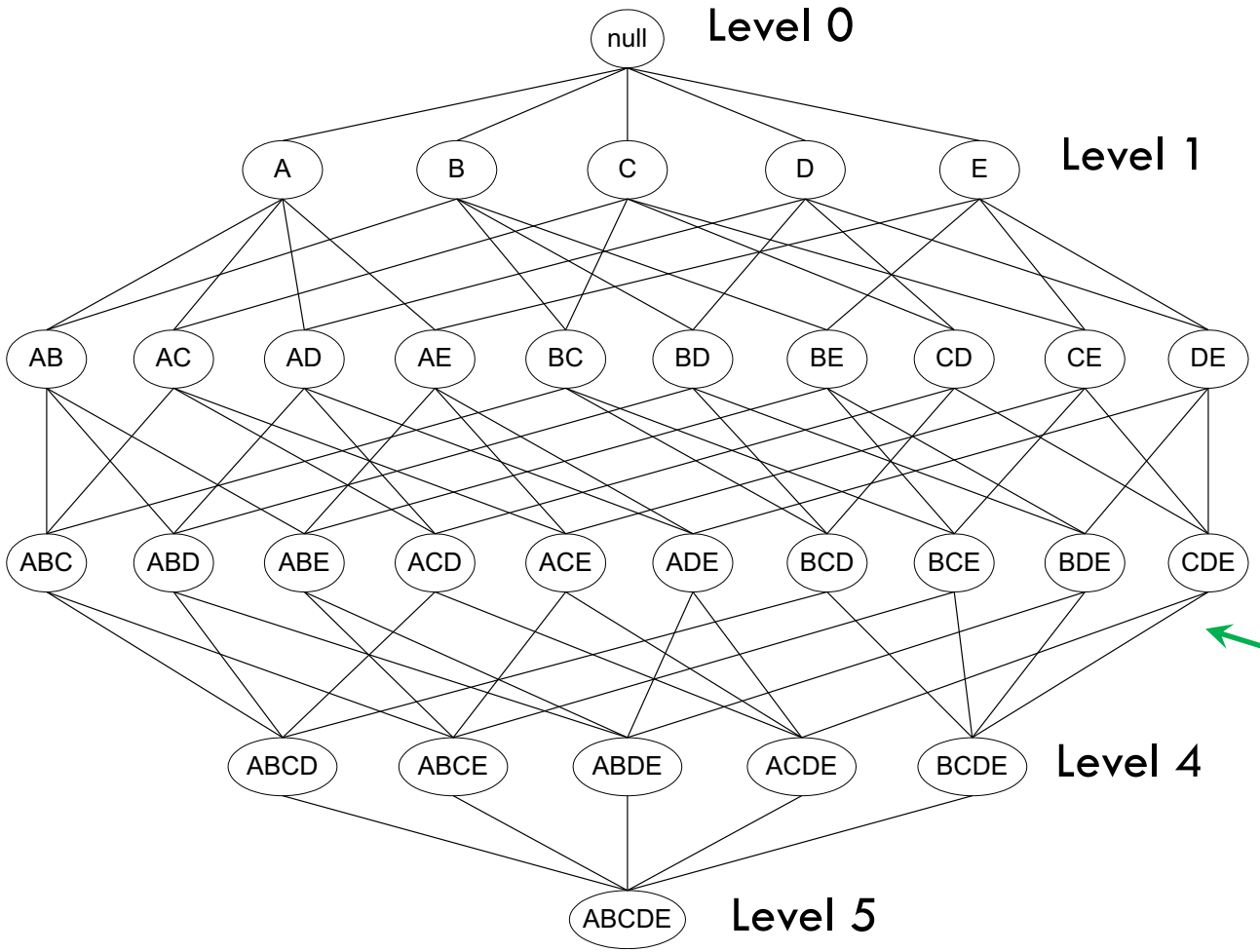
THE ITEMSET LATTICE



Nodes represent itemsets

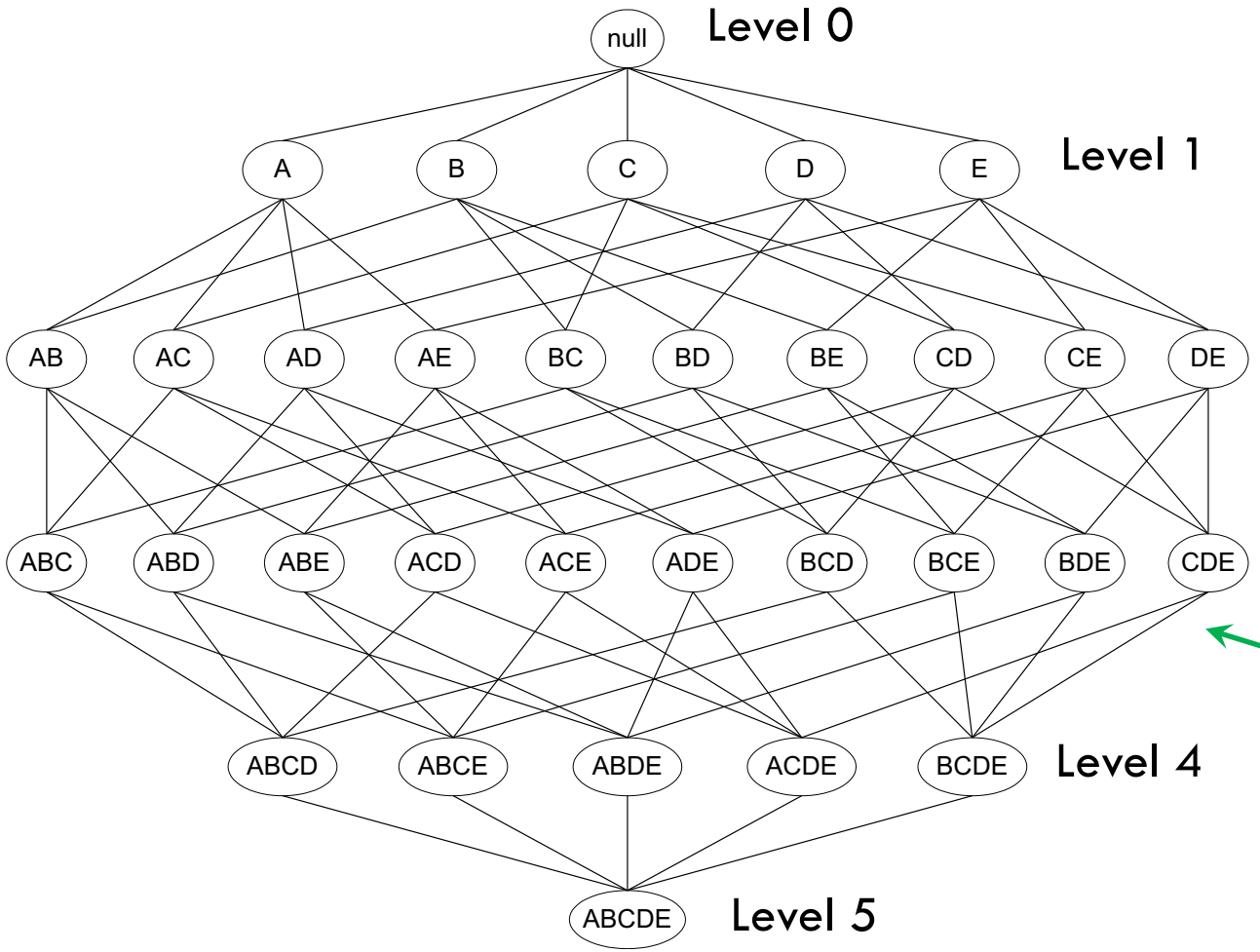
Edges represent containment relationships: e.g. itemset {B,C,D,E} contains {C,D,E}

ANTI-MONOTONICITY ALONG THE ITEMSET LATTICE



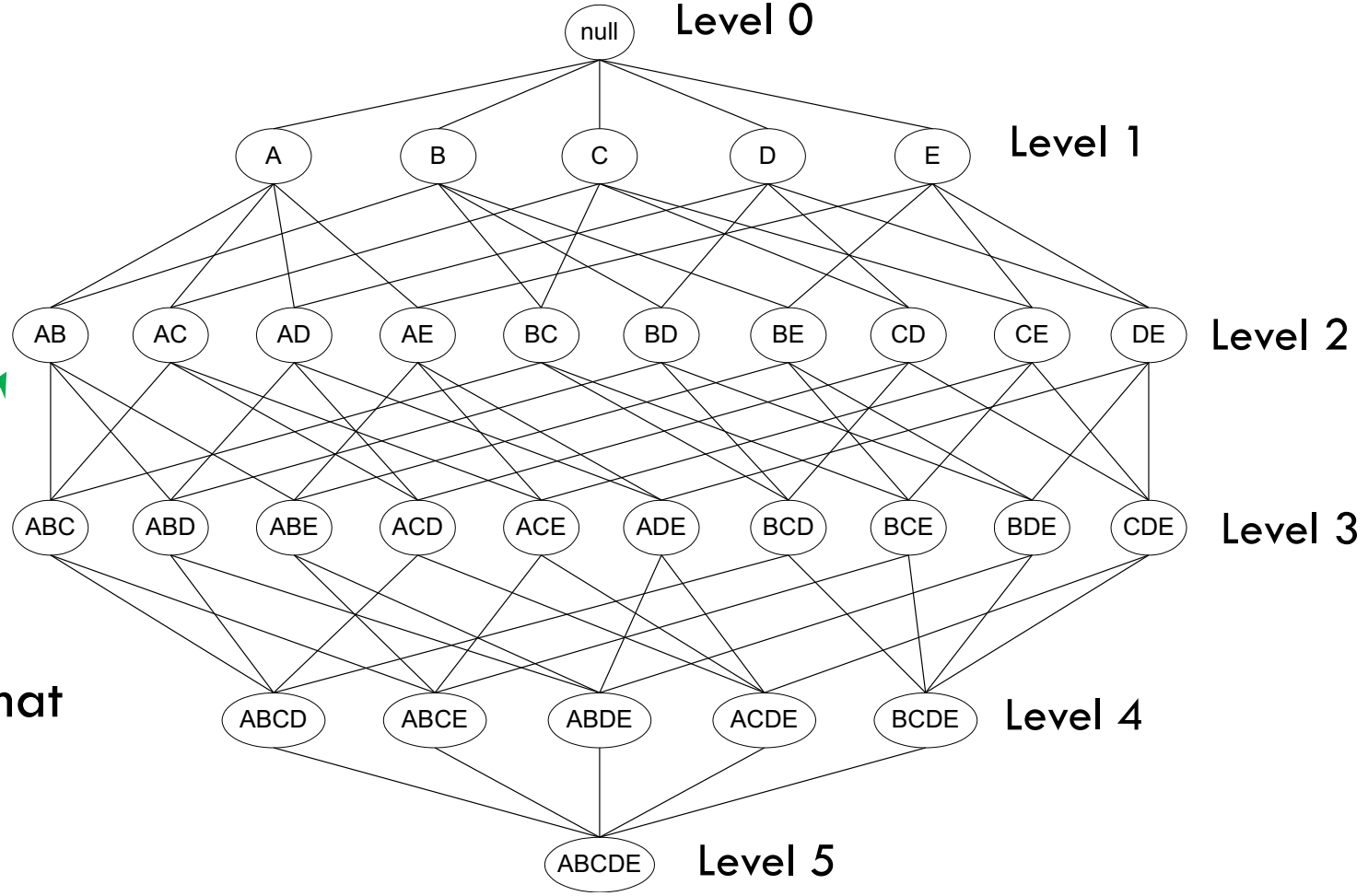
Anti-monotonicity: "if an itemset is frequent, then any subset is frequent"

ANTI-MONOTONICITY ALONG THE ITEMSET LATTICE



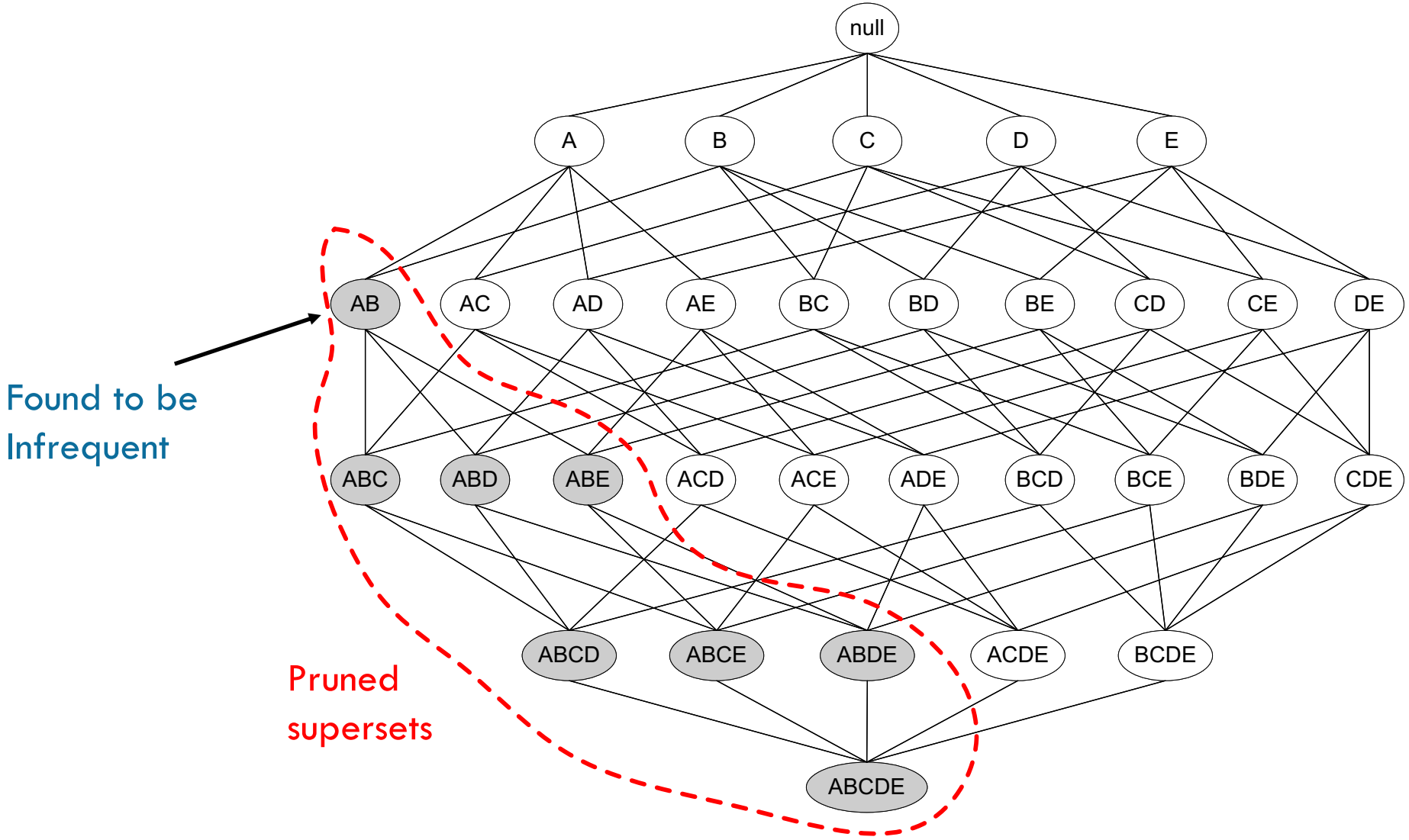
Anti-monotonicity: along each edge, if the lower node is a frequent itemset, the upper node is also frequent.

ANTI-MONOTONICITY ALONG THE ITEMSET LATTICE



If AB is not a frequent itemset, what can we conclude about ABC?

ILLUSTRATING APRIORI PRINCIPLE



ILLUSTRATING APRIORI PRINCIPLE

Q: Assume that among 1-itemsets, only {A}, {B}, {E} are frequent. Among 2-itemsets, which of the following could be frequent (according to the Anti-Monotonicity Principle?)

- (a) {A, E}
- (b) {A, C}
- (c) {C, D}

ILLUSTRATING APRIORI PRINCIPLE

Q: Assume that among 1-itemsets, only {A}, {B}, {E} are frequent. Among 2-itemsets, which of the following could be frequent (according to the Anti-Monotonicity Principle?

- **(a) {A, E}**
- (b) {A, C}
- (c) {C, D}

APRIORI ALGORITHM: FREQUENT ITEMSETS

- **Goal**: find all itemsets with support ≥ *minsup*
- Apriori is an **exact** algorithm: it finds all such itemsets

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

APRIORI ALGORITHM: FREQUENT ITEMSETS

- **Goal**: find all itemsets with support ≥ *minsup*
- Apriori is an **exact** algorithm: it finds all such itemsets

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

Apriori Algorithm (Overview):

- For k = 1, 2, …
  - **Generate** candidate frequent k-itemsets
  - **Filter** candidates to get all frequent k-itemsets

APRIORI ALGORITHM: ANTI-MONOTONICITY

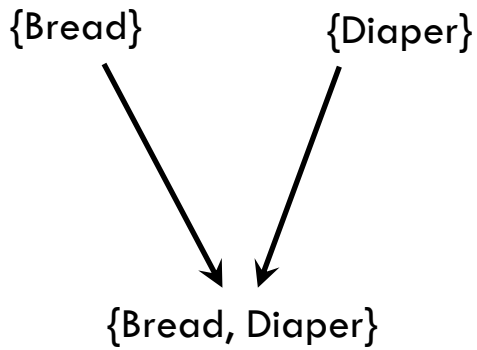
Apriori Algorithm (Overview):

- For k = 1, 2, …
  - **Generate** candidate frequent k-itemsets
  - **Filter** candidates to get all frequent k-itemsets

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

Generation step uses the Anti-monotonicity property:

- E.g. Imagine we have found all frequent 1-itemsets.
- Then for {Bread, Diaper} to be frequent, both {Bread} and {Diaper} must be frequent!
- This can greatly reduce the number of itemsets we need to consider



Database D

TID	Items
10	a, c, d
20	b, c, e
30	a, b, c, e
40	b, e

***minsup* =0.5**

Database D

TID	Items
10	a, c, d
20	b, c, e
30	a, b, c, e
40	b, e

***minsup* =0.5**

Scan D

1-candidates

Itemset	SC
a	
b	
c	
d	
e	

Database D

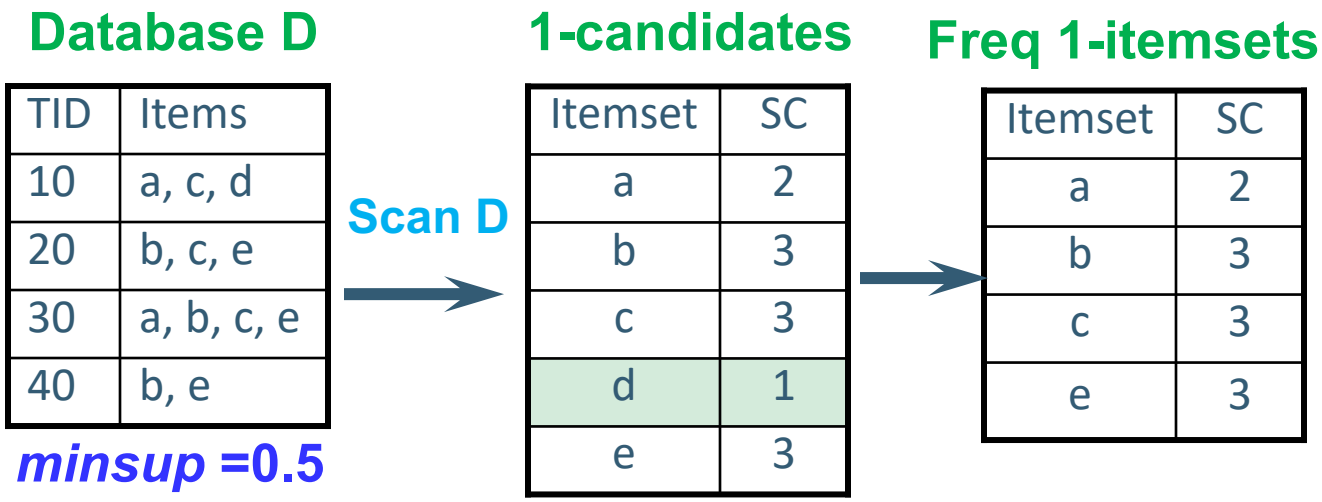
TID	Items
10	a, c, d
20	b, c, e
30	a, b, c, e
40	b, e

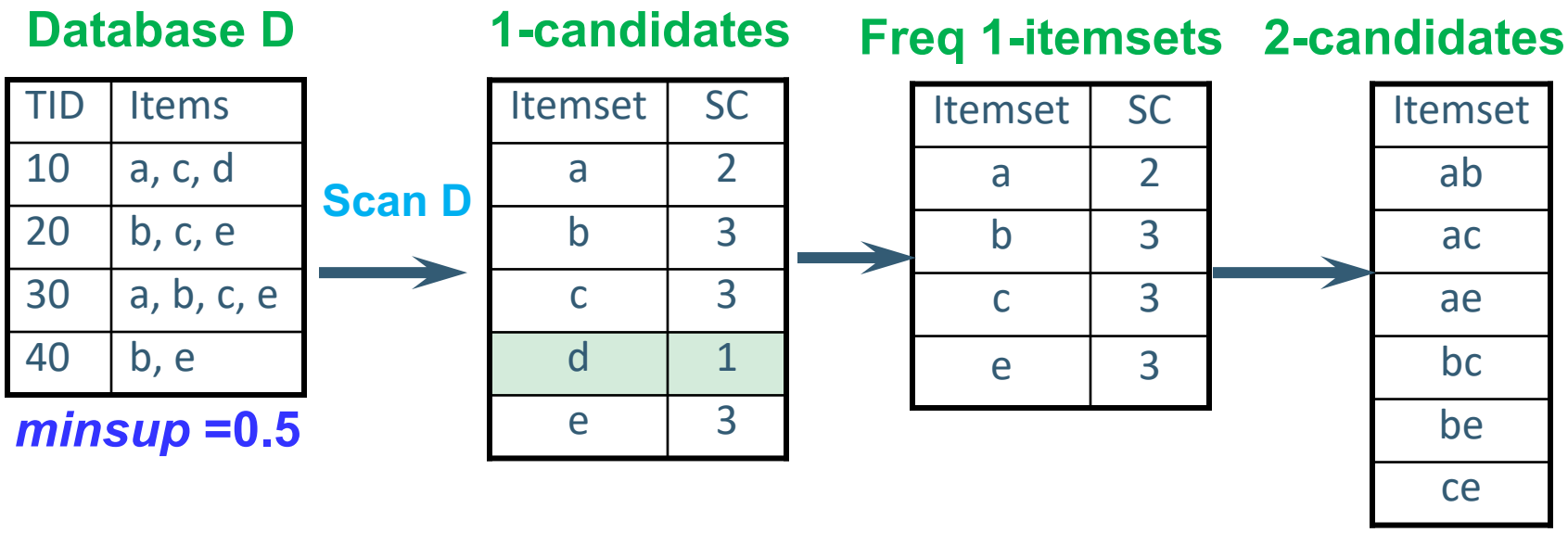
***minsup* =0.5**

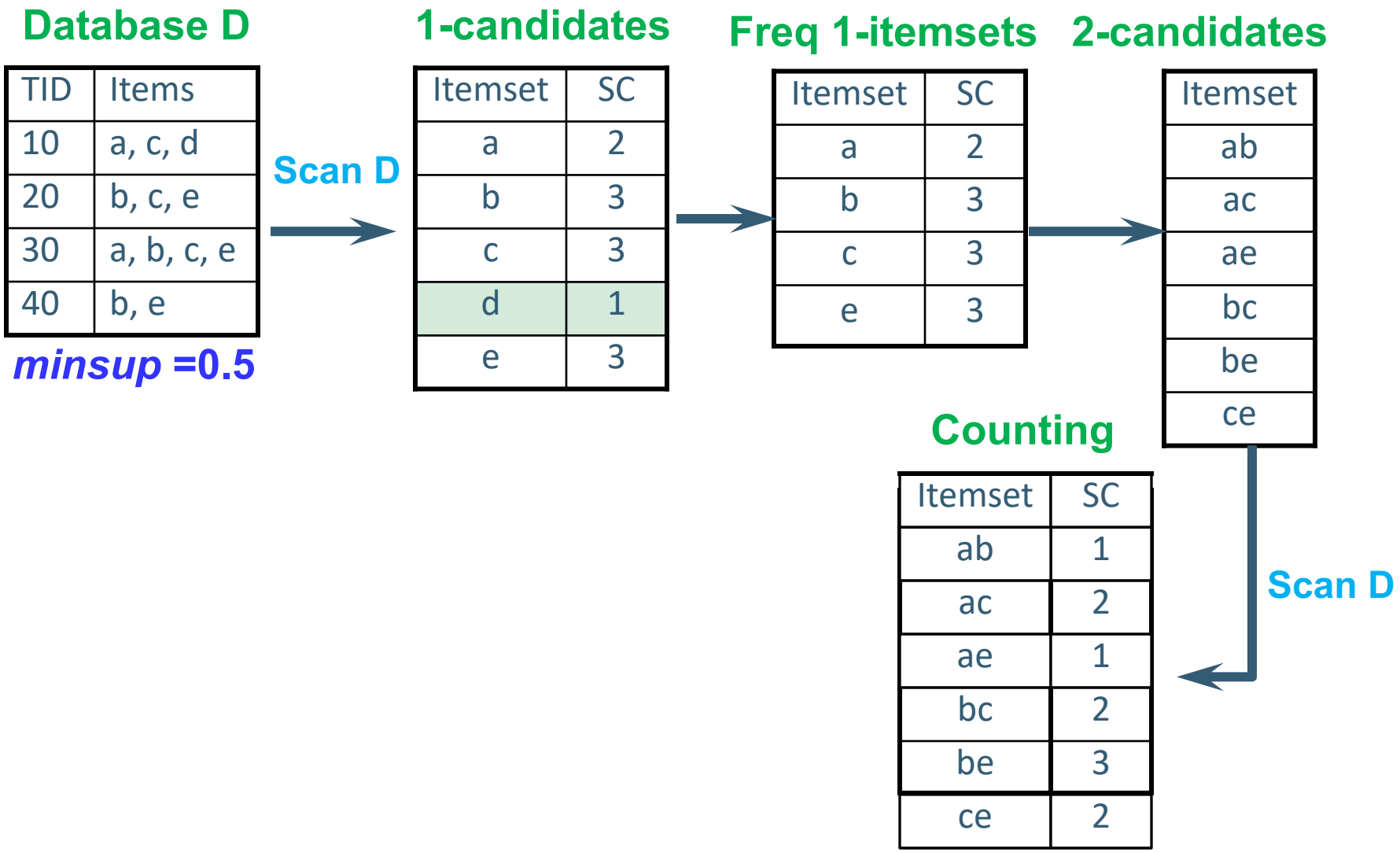
Scan D

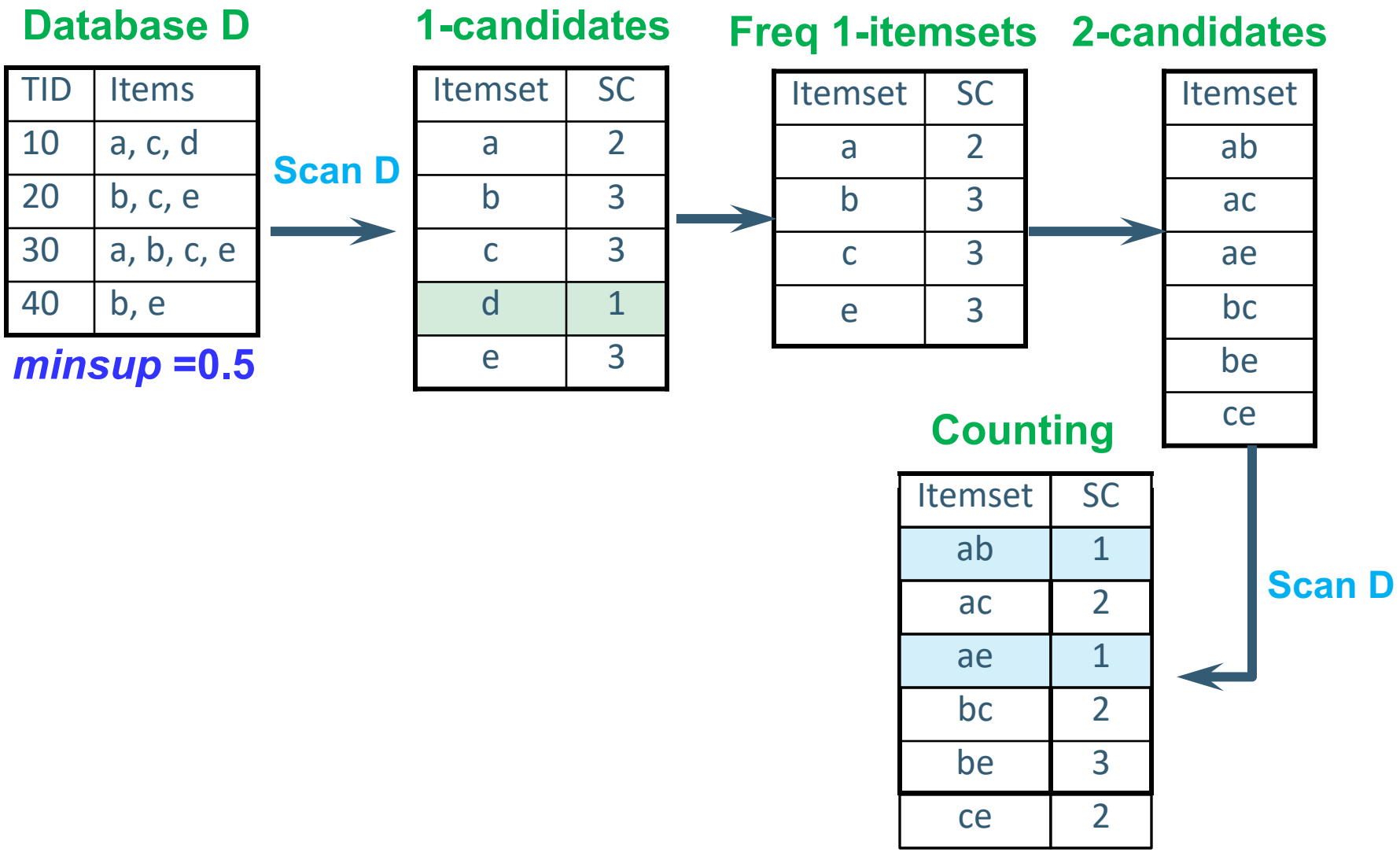
1-candidates

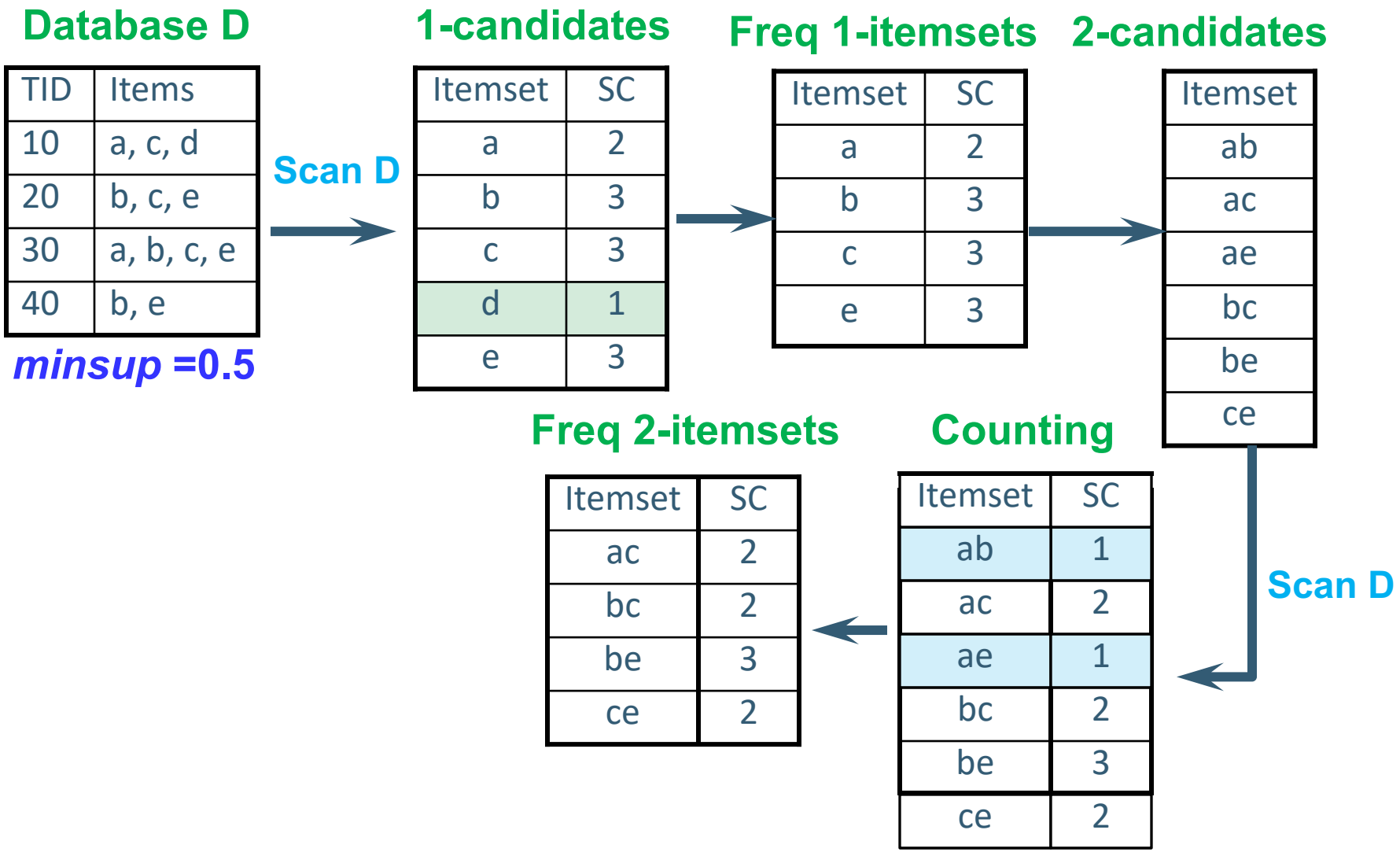
Itemset	SC
a	2
b	3
c	3
d	1
e	3

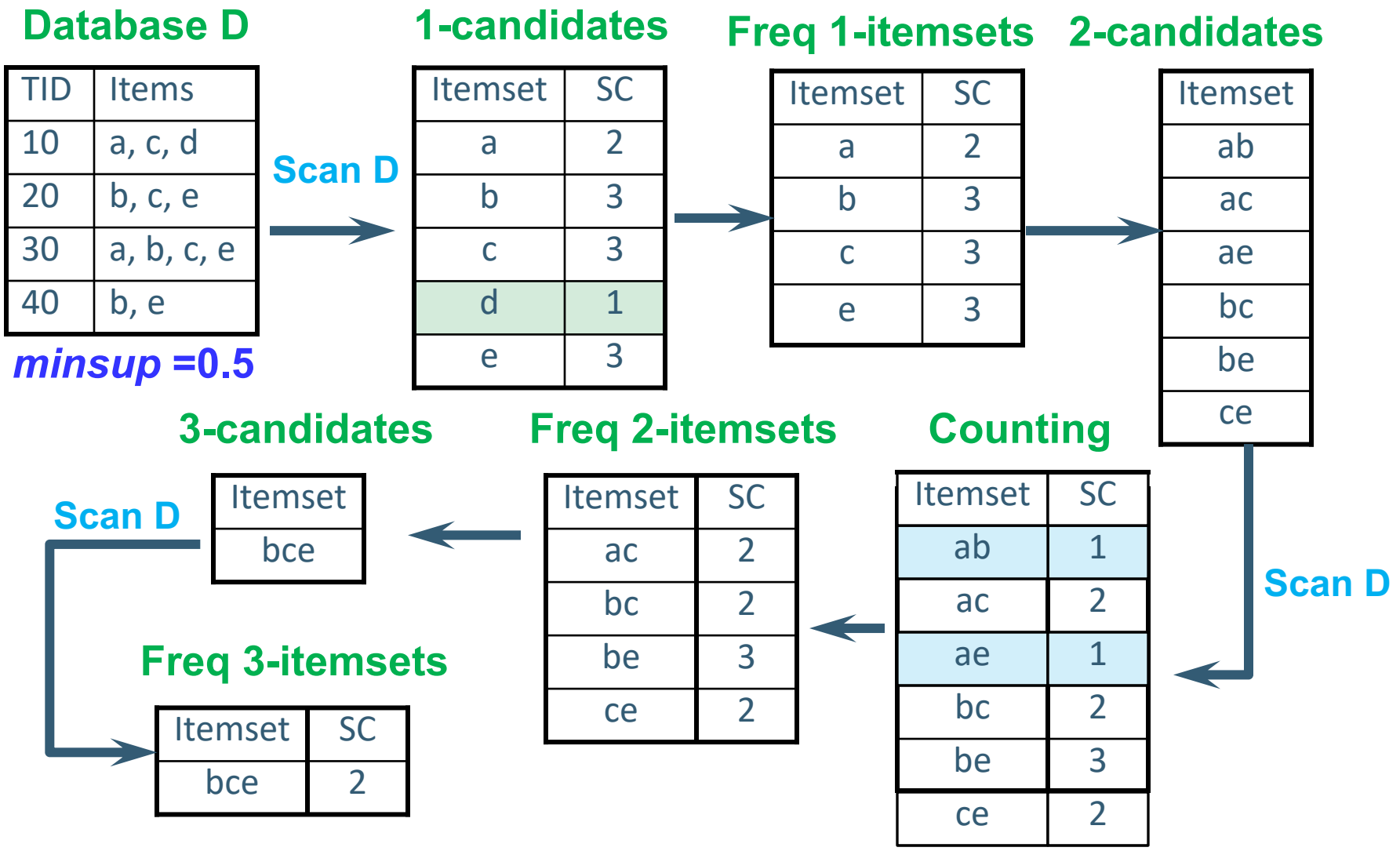

Database D
TID
Items
10
a, c, d
20
b, c, e
30
a, b, c, e
40
b, e
minsup =0.5
Scan D
1-candidates
Itemset
SC
a
2
b
3
c
3
d
1
e
3
Freq 1-itemsets
Itemset
SC
a
2
b
3
c
3
e
3

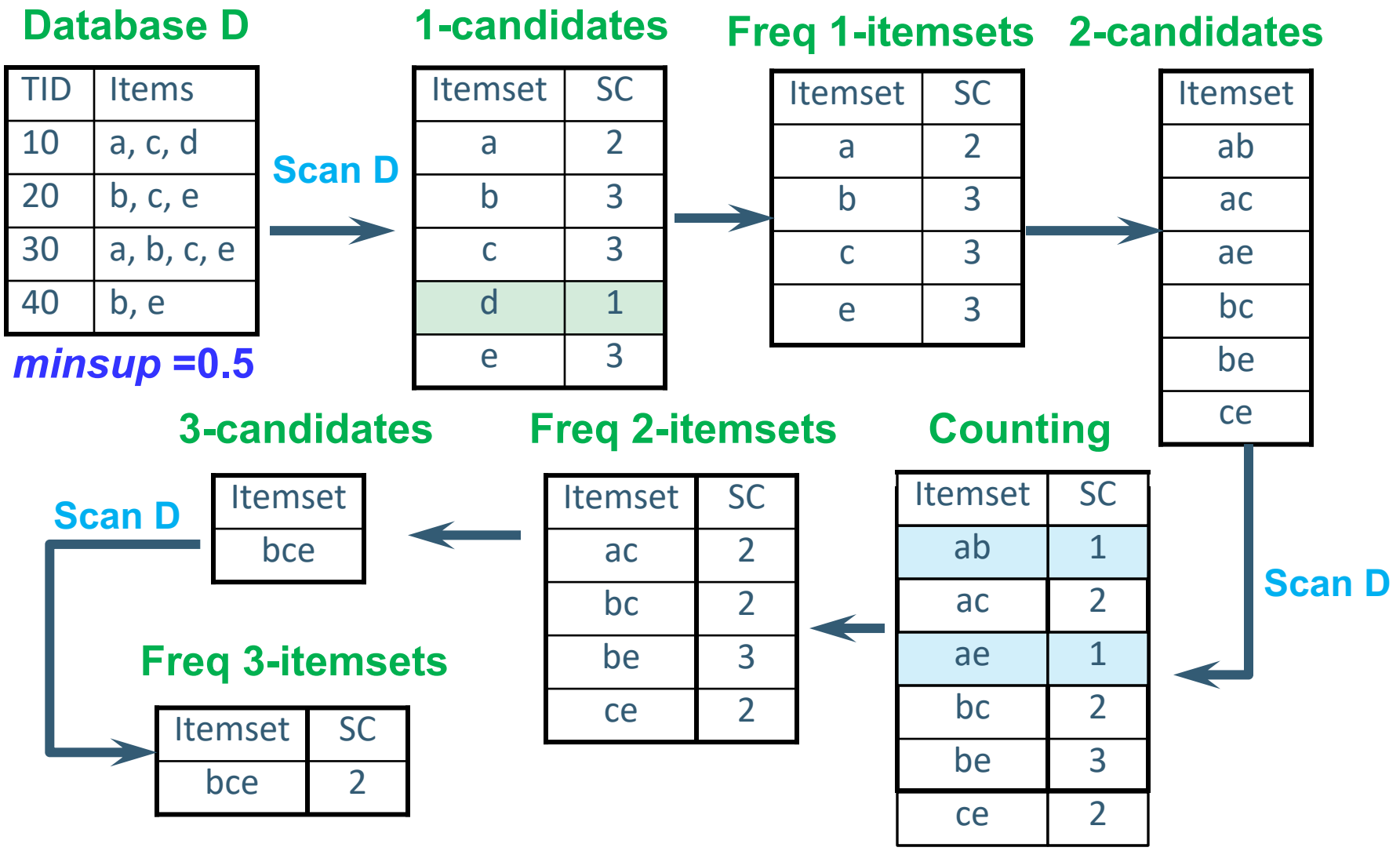

Database D
TID
Items
10
a, c, d
20
b, c, e
30
a, b, c, e
40
b, e
minsup =0.5
Scan D
1-candidates
Itemset
SC
a
2
b
3
c
3
d
1
e
3
Freq 1-itemsets
Itemset
SC
a
2
b
3
c
3
e
3
2-candidates
Itemset
ab
ac
ae
bc
be
ce


Database D
TID
Items
10
a, c, d
20
b, c, e
30
a, b, c, e
40
b, e
minsup =0.5
Scan D
1-candidates
Itemset
SC
a
2
b
3
c
3
d
1
e
3
Freq 1-itemsets
Itemset
SC
a
2
b
3
c
3
e
3
2-candidates
Itemset
ab
ac
ae
bc
be
ce
Counting
Itemset
SC
ab
1
ac
2
ae
1
bc
2
be
3
ce
2
Scan D


Database D
TID
Items
10
a, c, d
20
b, c, e
30
a, b, c, e
40
b, e
minsup =0.5
Scan D
1-candidates
Itemset
SC
a
2
b
3
c
3
d
1
e
3
Freq 1-itemsets
Itemset
SC
a
2
b
3
c
3
e
3
2-candidates
Itemset
ab
ac
ae
bc
be
ce
Counting
Itemset
SC
ab
1
ac
2
ae
1
bc
2
be
3
ce
2
Scan D


Database D
TID
Items
10
a, c, d
20
b, c, e
30
a, b, c, e
40
b, e
minsup =0.5
Scan D
1-candidates
Itemset
SC
a
2
b
3
c
3
d
1
e
3
Freq 1-itemsets
Itemset
SC
a
2
b
3
c
3
e
3
2-candidates
Itemset
ab
ac
ae
bc
be
ce
Scan D
Counting
Itemset
SC
ab
1
ac
2
ae
1
bc
2
be
3
ce
2
Freq 2-itemsets
Itemset
SC
ac
2
bc
2
be
3
ce
2


Database D
TID
Items
10
a, c, d
20
b, c, e
30
a, b, c, e
40
b, e
minsup =0.5
Scan D
1-candidates
Itemset
SC
a
2
b
3
c
3
d
1
e
3
Freq 1-itemsets
Itemset
SC
a
2
b
3
c
3
e
3
2-candidates
Itemset
ab
ac
ae
bc
be
ce
Scan D
Counting
Itemset
SC
ab
1
ac
2
ae
1
bc
2
be
3
ce
2
Freq 2-itemsets
Itemset
SC
ac
2
bc
2
be
3
ce
2
3-candidates
Itemset
bce
Scan D
Freq 3-itemsets
Itemset
SC
bce
2



There is no "abc" or "ace". Why?

APRIORI ALGORITHM & FILTERING STEPS

Q: As described, around how many times does this algorithm need to scan over our dataset?

- (a) Number of frequent itemsets
- (b) Maximum size of a frequent itemset
- (c) Number of items

APRIORI ALGORITHM & FILTERING STEPS

Q: As described, around how many times does this algorithm need to scan over our dataset?

- (a) Number of frequent itemsets
- **(b) Maximum size of a frequent itemset**
- (c) Number of items

APRIORI ALGORITHM

Apriori Algorithm:

- For k = 1, 2, …
  - **Generate** candidate frequent k-itemsets
  - **Filter** candidates to get all frequent k-itemsets

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

GENERATION STEP (FOR K=1)

Apriori Algorithm:

- For k = 1, 2, …
  - **Generate** candidate frequent k-itemsets
  - **Filter** candidates to get all frequent k-itemsets

- If k=1, the candidates are just **all 1-itemsets**
- (In the subsequent Filter step, we will count them to find which are frequent)

1-candidates

Itemset	SC
a	2
b	3
c	3
d	1
e	3

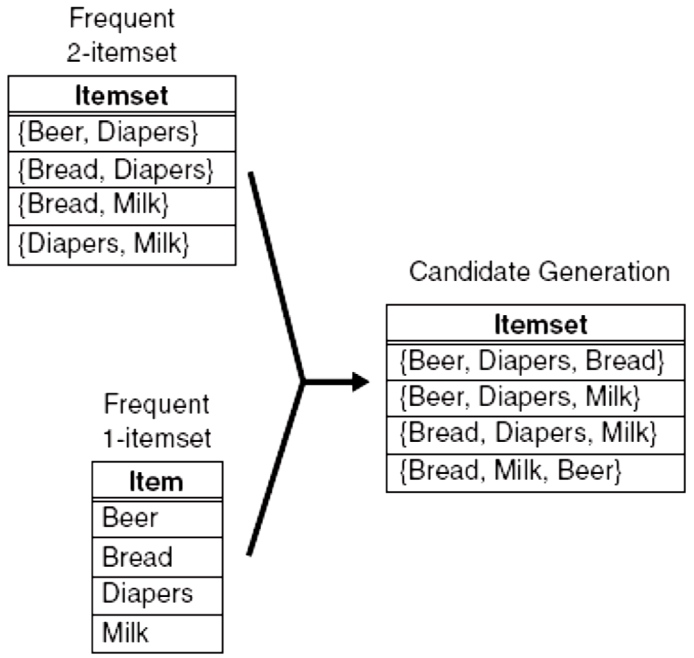
- How to handle higher values of k?

GENERATION STEP (F_{K-1} X F_1 METHOD)

Apriori Algorithm:

- For k = 1, 2, …
  - **Generate** candidate frequent k-itemsets
  - **Filter** candidates to get all frequent k-itemsets

- **Merge** frequent (k-1) and 1-itemsets, then
- **Prune** resulting k-itemsets if they have a (k-1) subset which is not frequent

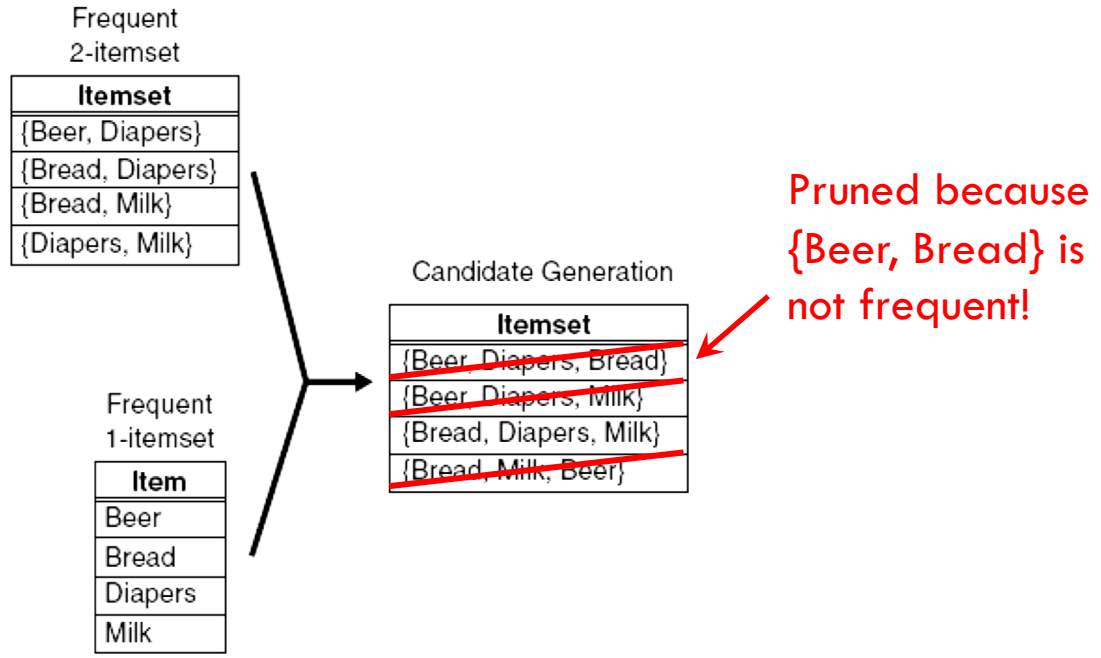


GENERATION STEP (F_{K-1} X F_1 METHOD)

Apriori Algorithm:

- For k = 1, 2, …
  - **Generate** candidate frequent k-itemsets
  - **Filter** candidates to get all frequent k-itemsets

- **Merge** frequent (k-1) and 1-itemsets, then
- **Prune** resulting k-itemsets if they have a (k-1) subset which is not frequent

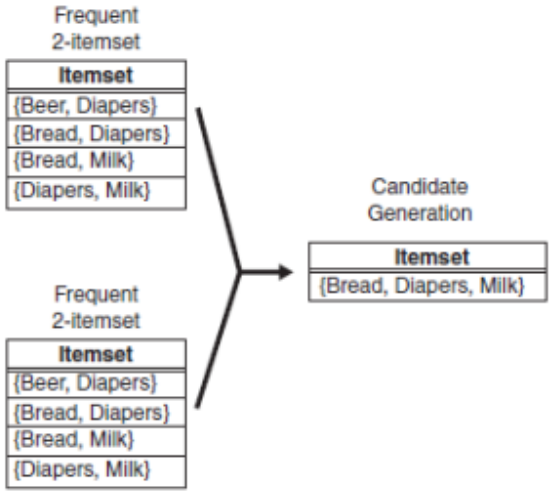


GENERATION STEP (F_{K-1} X F_{K-1} METHOD)

Apriori Algorithm:

- For k = 1, 2, …
  - **Generate** candidate frequent k-itemsets
  - **Filter** candidates to get all frequent k-itemsets

- Alternate approach: **merge** two frequent (k-1) itemsets if their first (k-2) items are identical
  - E.g. Merge(**AB**C, **AB**D) = **AB**CD
    Merge(**AB**C, **AB**E) = **AB**CE
- Then perform pruning as before



FILTER STEP

Apriori Algorithm:

- For k = 1, 2, …
  - **Generate** candidate frequent k-itemsets
  - **Filter** candidates to get all frequent k-itemsets

- Query the database to compute support of each candidate
- Filter away candidates with support < *minsup*

Itemset	SC
ab	1
ac	2
ae	1
bc	2
be	3
ce	2

SPEEDUP USING SUPPORT COUNTING

During each **Filter** step, we need to query the database to compute the support of each itemset

For **each** transaction we receive, we need to compare it to **each** itemset to see if the itemset is contained in it – this can be slow!

Database D

TID	Items
10	a, c, d
20	b, c, e
30	a, b, c, e
40	b, e

Itemset	SC
ab	1
ac	2
ae	1
bc	2
be	3
ce	2

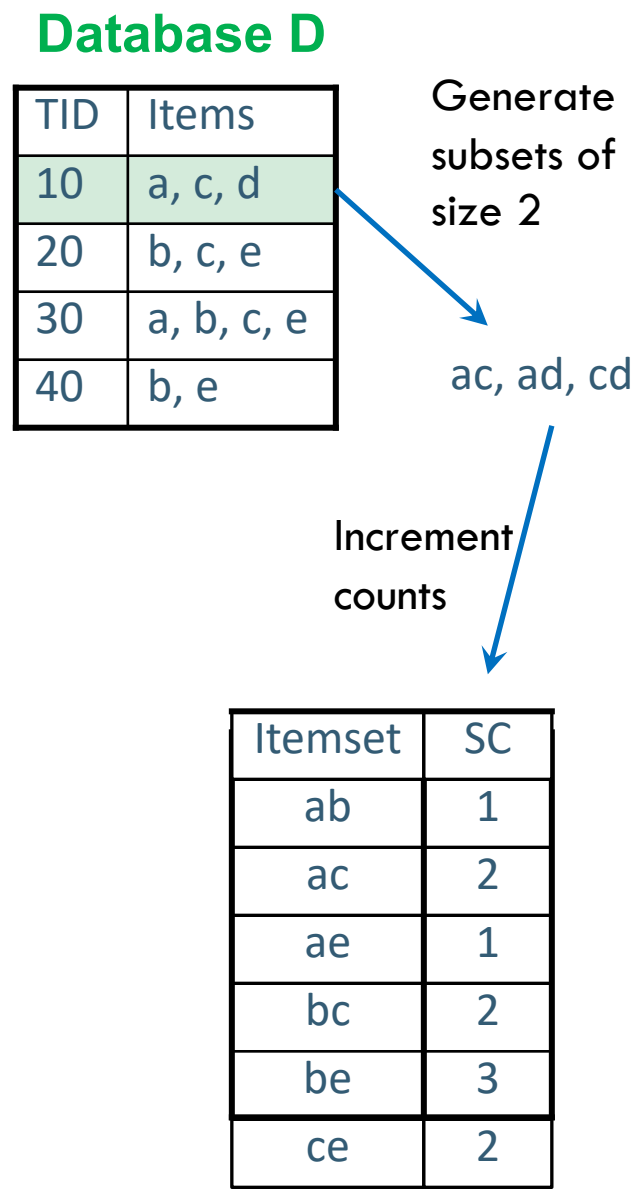
SPEEDUP USING SUPPORT COUNTING

During each **Filter** step, we need to query the database to compute the support of each itemset

For **each** transaction we receive, we need to compare it to **each** itemset to see if the itemset is contained in it – this can be slow!

Support Counting:

- For each transaction, generate all its subsets of size k
- E.g. {a, c, d} → {a, c}, {a, d}, {c, d}
- Then increment counters for each of these 2-element subsets
- The counts for each itemset can be kept in a hash table
- (For this to work, the generated subsets should be sorted, e.g. lexicographically)



APRIORI ALGORITHM: TWO PART APPROACH

Apriori Algorithm:

Part 1: Frequent Itemset Generation

- Generate all itemsets whose support ≥ *minsup*

Part 2: Rule Generation

- Generate rules from each frequent itemset, where each rule is **a binary partition** of a frequent itemset with confidence ≥ *minconf*

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

Thresholds *minsup*, *minconf*

Frequent Itemsets:
{Bread, Milk}, {Bread, Diaper}, …

Association Rules:
{Bread} → {Milk}, …

GENERATING RULES: AN EXAMPLE

Suppose {A,B,C} is frequent, with sup=0.5

- Proper nonempty subsets: {A,B}, {A,C}, {B,C}, {A}, {B}, {C}, with sup=0.5, 0.5, 0.75, 0.75, 0.75, 0.75 respectively
- These generate the following association rules:
  - A,B $\rightarrow$ C, confidence = 1
  - A,C $\rightarrow$ B, confidence = 1
  - B,C $\rightarrow$ A, confidence = 0.67
  - A $\rightarrow$ B,C, confidence = 0.67
  - B $\rightarrow$ A,C, confidence = 0.67
  - C $\rightarrow$ A,B, confidence = 0.67
  - All rules have support = 0.5

$$Conf(X \rightarrow Y) = Supp(X \cup Y) / Supp(X)$$

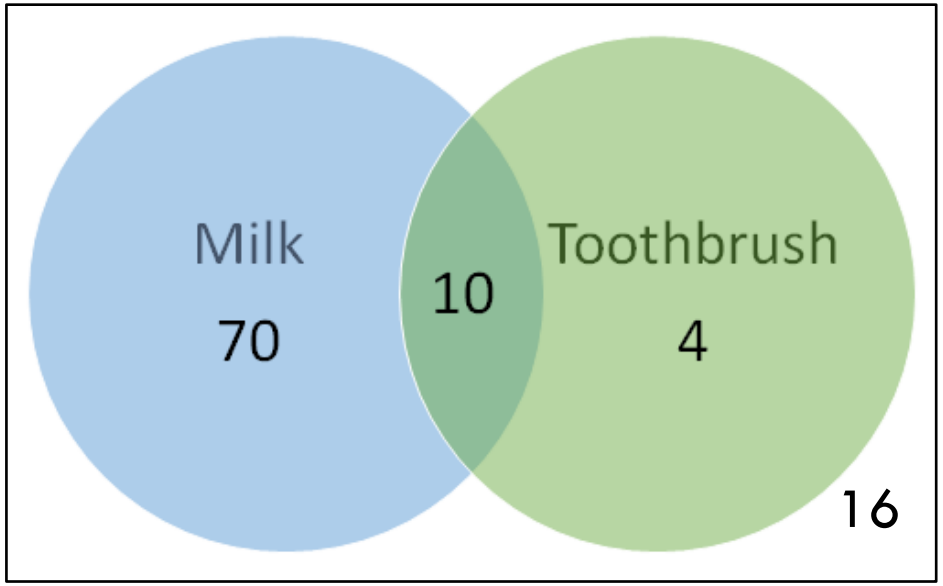
GENERATING RULES: SUMMARY

In order to compute Conf($X \rightarrow Y$), we need to have Sup($X \cup Y$) and Sup(X)

All the required information for confidence computation has already been recorded in itemset generation. No need to scan the transaction data any more.

Thus, rule generation is typically faster than itemset generation.

OTHER METRICS: CONFIDENCE IS NOT EVERYTHING



Confidence for {Toothbrush} → {Milk} = 10/(10+4) = 0.71

But the probability of having milk on the cart without any knowledge about toothbrush or any other items = 80/100 = 0.8

So when buying Toothbrush, Milk is actually less likely!

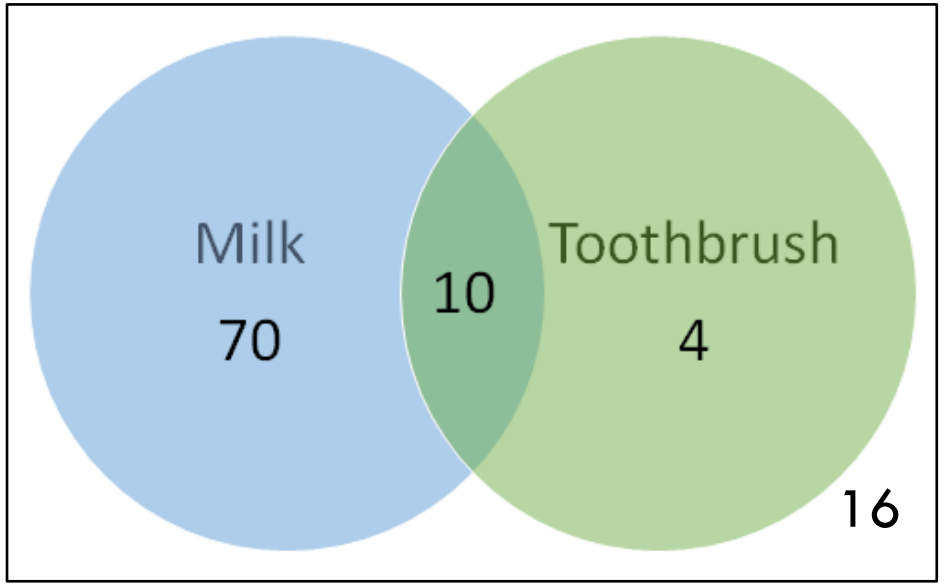
ADJUSTING FOR RANDOMNESS: LIFT / INTEREST

This aims to adjust for the "baseline" probability of observing Y

Lift: The $lift(X \rightarrow Y)$ of the rule $X \rightarrow Y$ is defined as

$$lift(X \rightarrow Y) = \frac{P(Y \mid X)}{P(Y)} = \frac{Supp(X \cup Y)}{Supp(X) * Supp(Y)}$$

If lift is greater than 1, X and Y are positively correlated.
E.g. lift({Toothbrush} → {Milk})
= (10/14) / (80/100) = 0.89



ADJUSTING FOR RANDOMNESS: LIFT / INTEREST

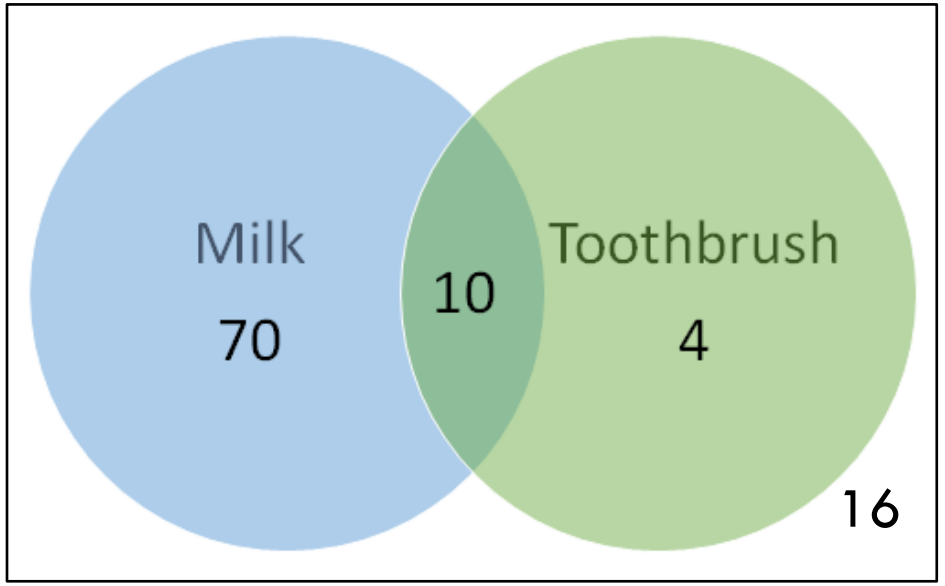
Both of these aim to adjust for the "baseline" probability of observing Y

Lift: The $lift(X \rightarrow Y)$ of the rule $X \rightarrow Y$ is defined as

$$lift(X \rightarrow Y) = \frac{P(Y \mid X)}{P(Y)} = \frac{Supp(X \cup Y)}{Supp(X) * Supp(Y)}$$

Q: What is the lift of the rule {Milk} → {Toothbrush})

(a) 1.12
(b) 1
(c) 0.89



ADJUSTING FOR RANDOMNESS: LIFT / INTEREST

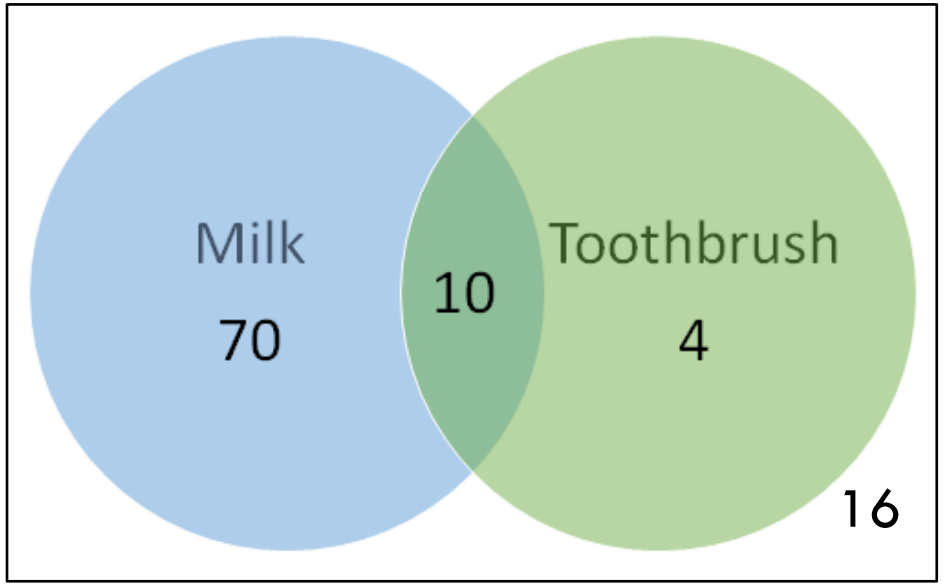
Both of these aim to adjust for the "baseline" probability of observing Y

Lift: The $lift(X \rightarrow Y)$ of the rule $X \rightarrow Y$ is defined as

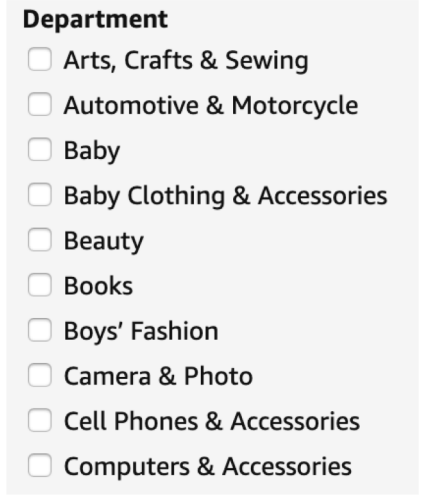
$$lift(X \rightarrow Y) = \frac{P(Y \mid X)}{P(Y)} = \frac{Supp(X \cup Y)}{Supp(X) * Supp(Y)}$$

Q: What is the lift of the rule {Milk} → {Toothbrush})

(a) 1.12
(b) 1
(c) 0.89



DISCUSSION / CONCLUSION



A (over?)-simplified view of shopping baskets

- Important and useful information not considered:
  - Quantity purchased
  - Price paid
  - Time / sequence of purchases
  - Categories / sectors of products
  - User features
- User features can be incorporated using **collaborative filtering**, which we will discuss later in the class