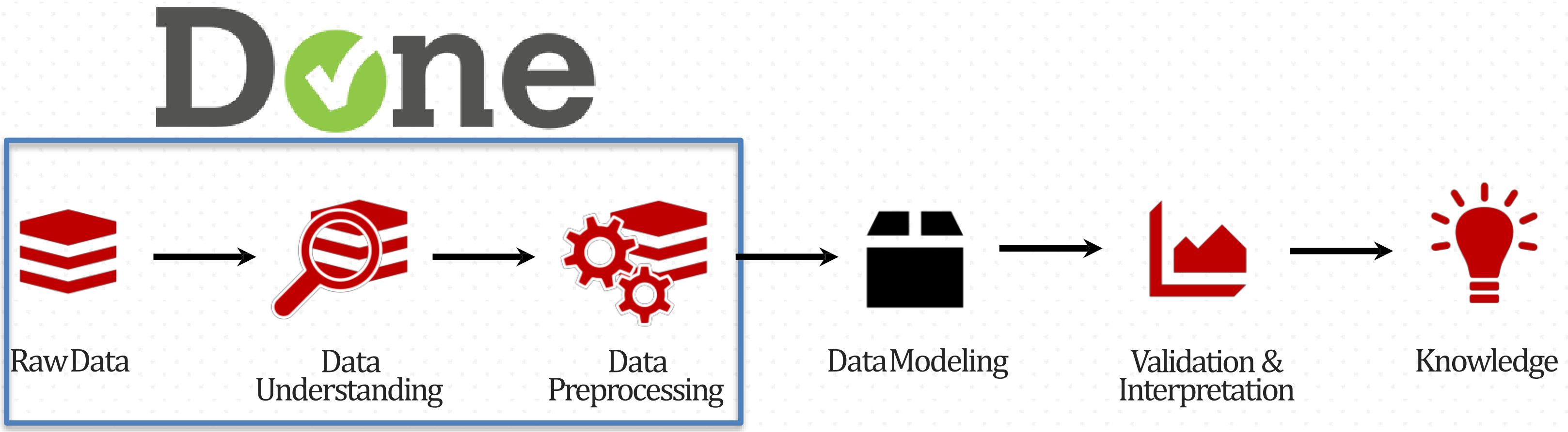
Chapter 6

Meng Jiang

CSE 40647/60647 Data Science Fall 2017

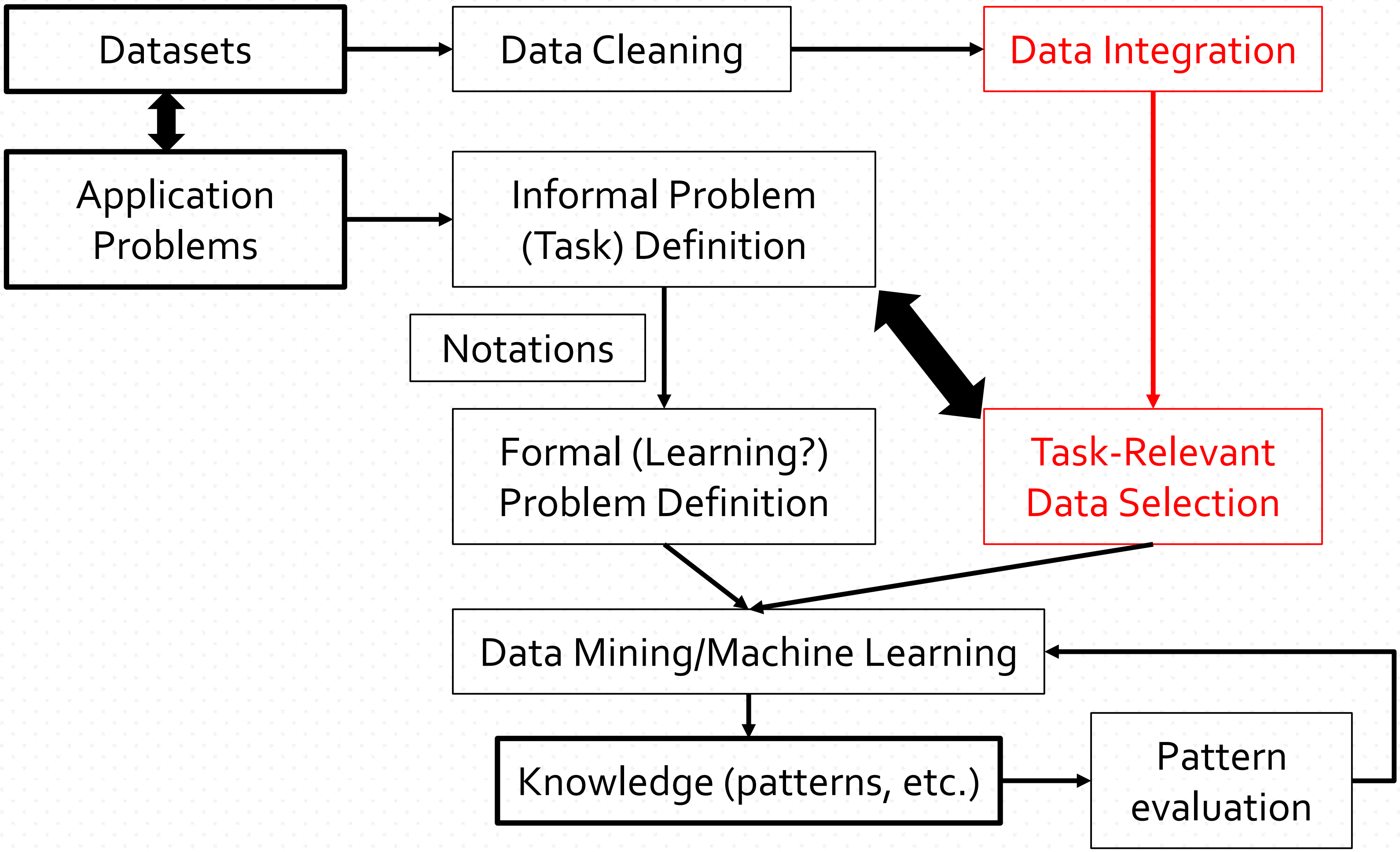
Introduction to Data Mining

Chapter 1-3. Data Processing



Previously on Data Science …

- Chapter 1. Introduction.



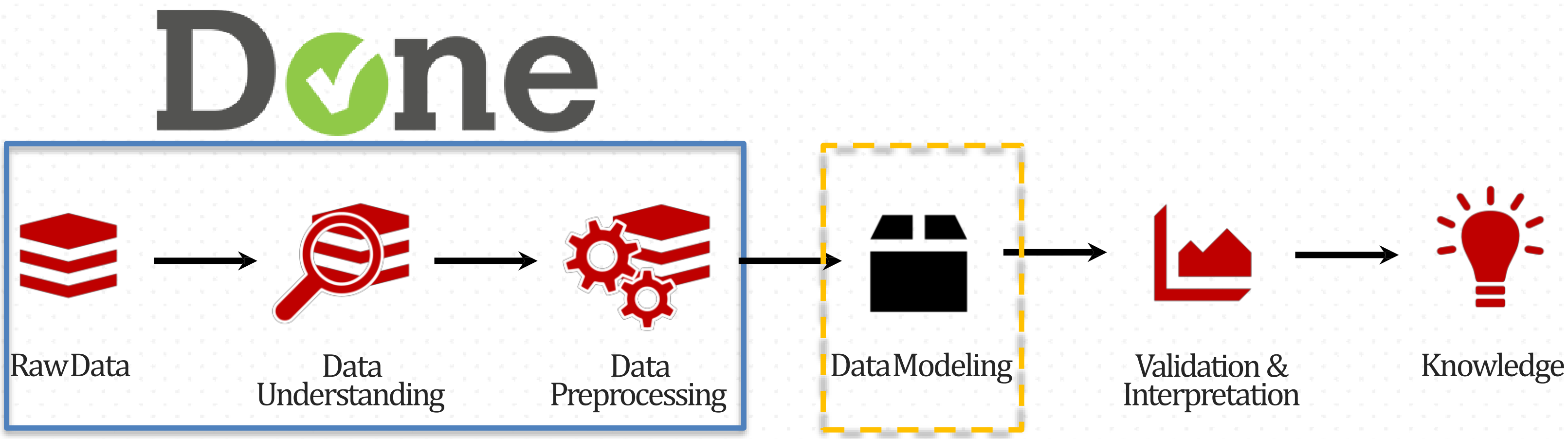
Previously on Data Science …

- Chapter 2. Get to Know Your Data.
  - Data Objects and Attribute Types
  - Basic Statistical Descriptions
    - Central tendency (mean, median, mode, etc.)
    - Outlierness (variance, standard deviation, z-score, etc.)
  - Data Visualization
    - Box plot, Histogram, Bar chart, Q plot, Q-Q plot, Scatter plot, etc.
  - Measuring Data Similarity and Dissimilarity
    - Minkowski distances
    - Jaccard/cosine similarity
    - KL divergence

Previously on Data Science …

- Chapter 3. Data Processing.
  - Data cleaning: Missing data, Noisy data
  - Data integration: Redundant data
    - Correlation analysis: Chi-square test, Covariance
  - Data reduction
    - Regression analysis: Linear, non-Linear
    - Histogram, Clustering, Sampling
    - Normalization: Min-max, Z-score, Decimal scaling
  - Dimensionality reduction
    - Feature selection
    - Feature extraction: PCA (eigenvectors), etc.

Moving On…


Done
Raw Data
Data Understanding
Data Preprocessing
Data Modeling
Validation & Interpretation
Knowledge

Concrete Learning Goals

- **Can process raw data: data cleaning, data integration, data reduction, dimension reduction**
- Can describe data warehouse, OLAP, data cube concepts and technology that work on multi-dimensional datasets
- **Can use Apriori and FP-Growth for frequent pattern mining**
- Can describe diverse patterns, sequential patterns, graph patterns
- **Can use Decision Tree, Naïve Bayes, Ensembles for classification**
- Can describe SVMs and Neural Networks for classification
- **Can use K-Partitioning Methods (K-Means, etc.) for clustering**
- Can describe Kernel-based Clustering and Density-based Clustering
- **Can use appropriate measures to evaluate results of different functionalities**

Tabular Data

Semester	Cliff Bar	Apples	Hummus Cup
Fall	50	200	75
Spring	120	55	200
Summer	85	645	12

Tabular Data

Semester	Cliff Bar	Apples	Hummus Cup
Fall	50	200	75
Spring	120	55	200
Summer	85	645	12

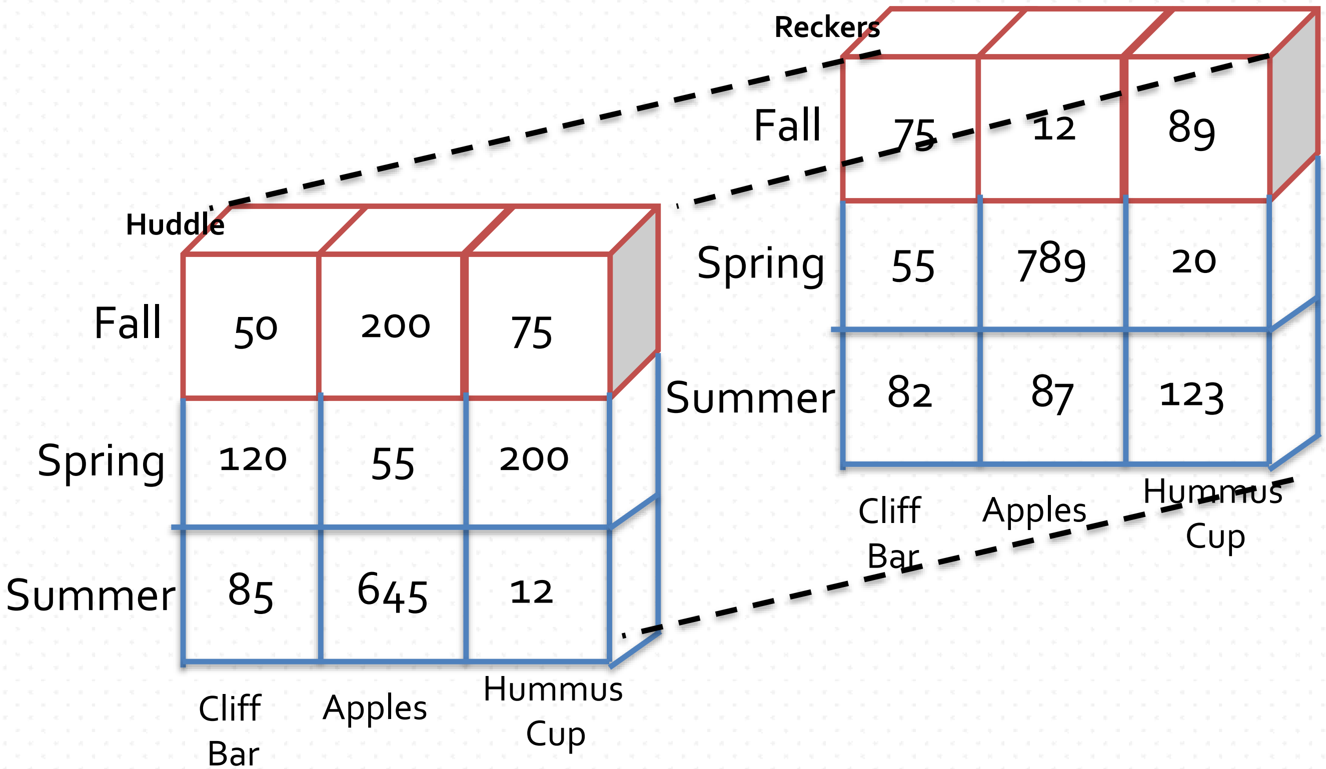
Time				
	Fall	50	200	75
	Spring	120	55	200
	Summer	85	645	12
		Cliff Bar	**Apples**	**Hummus Cup**
		Item (type)		

What if We Add More Locations?

	Huddle			Reckers		
Semester	Cliff Bar	Apples	Hummus Cup	Cliff Bar	Apples	Hummus Cup
Fall	50	200	75	75	12	89
Spring	120	55	200	55	789	20
Summer	85	645	12	82	87	123

What if We Add More Locations?

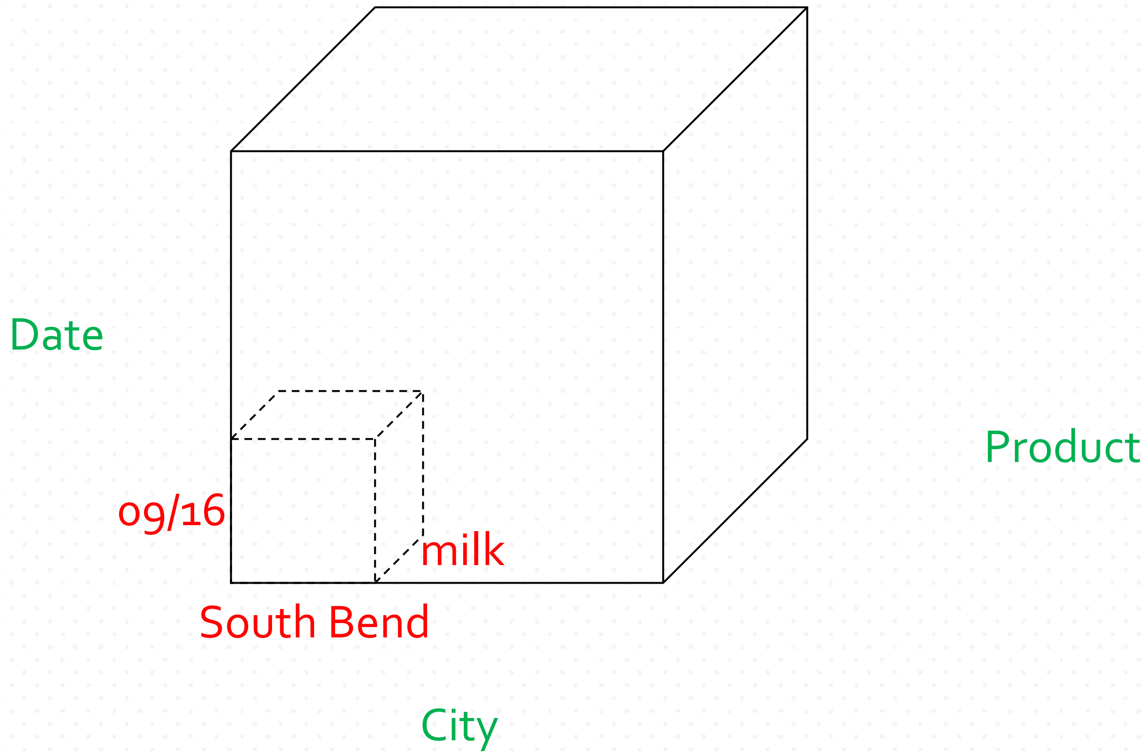
	Huddle			Reckers		
Semester	Cliff Bar	Apples	Hummus Cup	Cliff Bar	Apples	Hummus Cup
Fall	50	200	75	75	12	89
Spring	120	55	200	55	789	20
Summer	85	645	12	82	87	123



Preview – Data Cube: Cube Computation

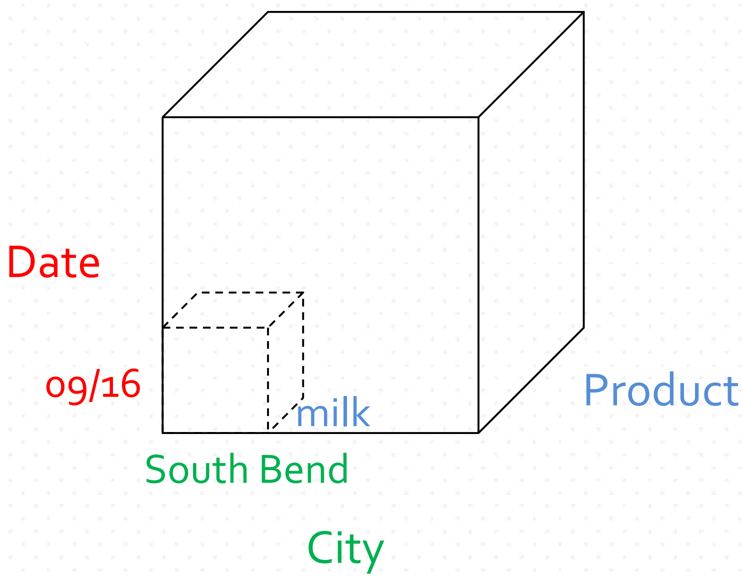
What Makes Up A Data Cube?

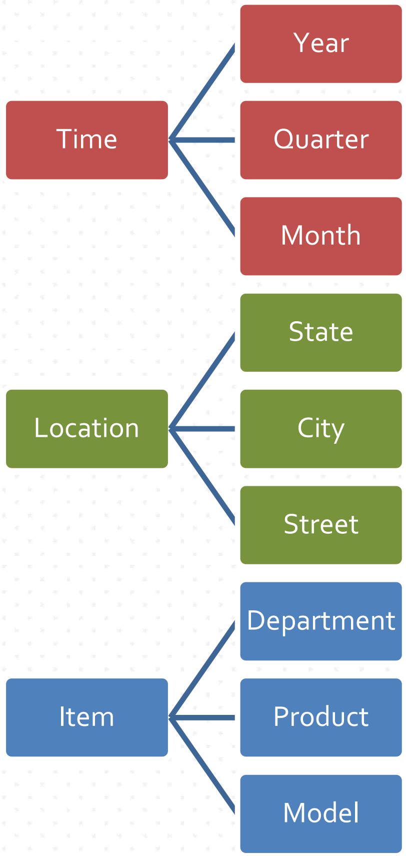
Dimensions



Dimension Tables

Dimension Level and Concept Hierarchy

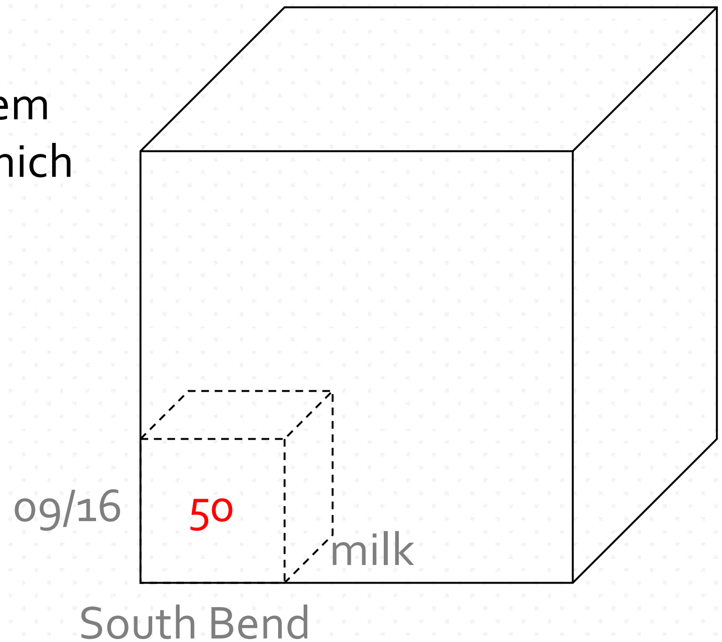




What Makes Up A Data Cube?

Facts

Facts are numerical measures. Think of them as the quantities by which we want to analyze relationships between dimensions

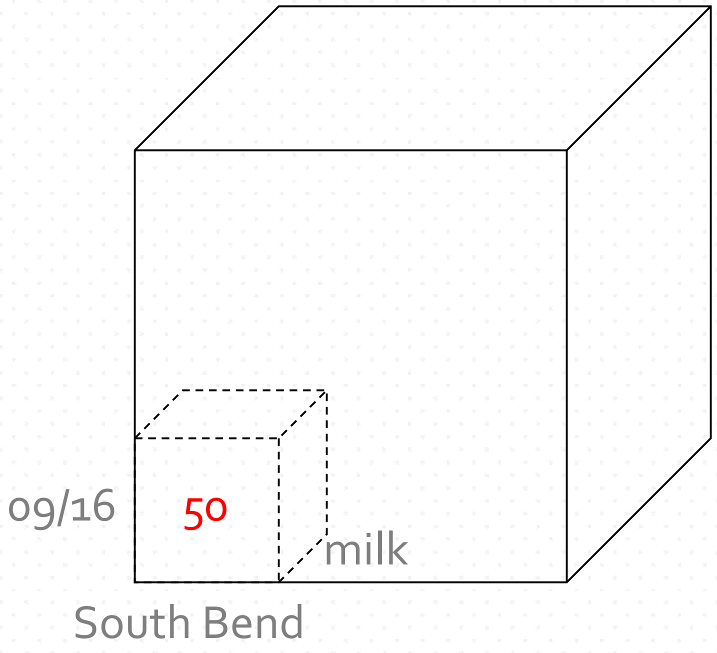


Base Cells and Aggregate Cells

Suppose a cuboid has 3 dimensions (time, location, item) at specific dimension levels (date, city, product).

- Base cells
  - (09/16, South Bend, milk)

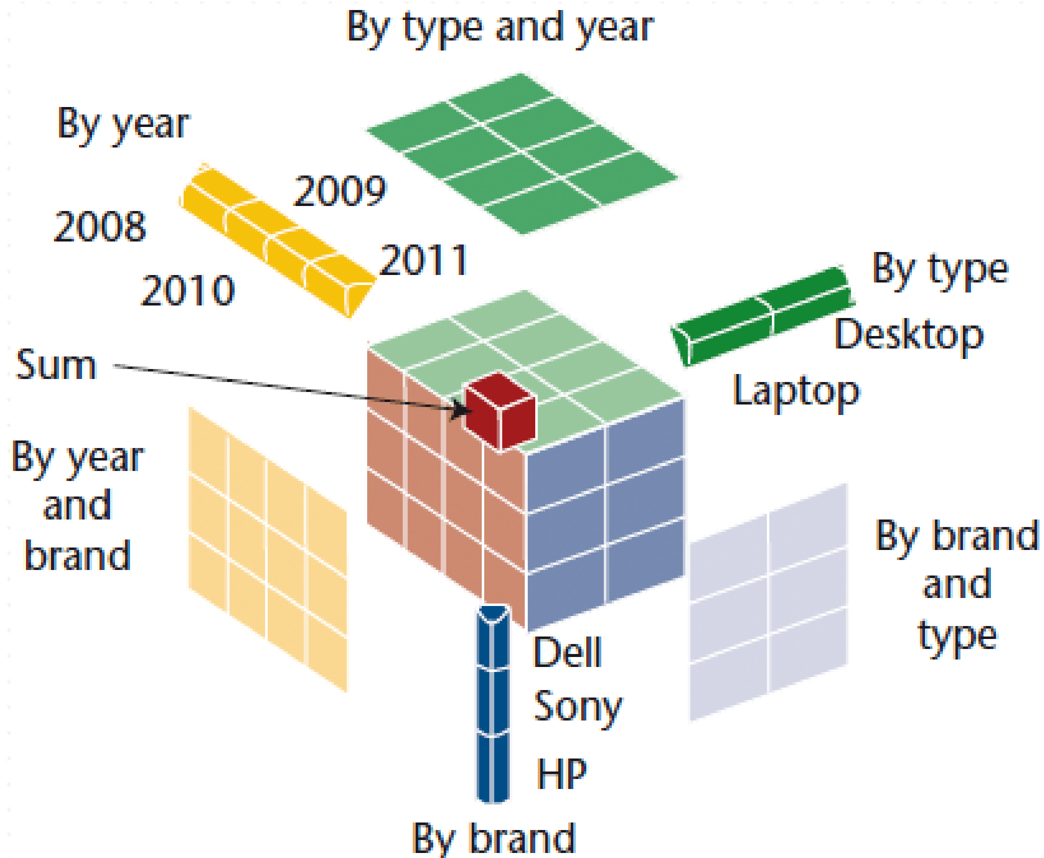
parent vs child cells
ancestor vs descendant cells
sibling cell:
(09/16, Mishawaka, milk)



Base Cells and Aggregate Cells

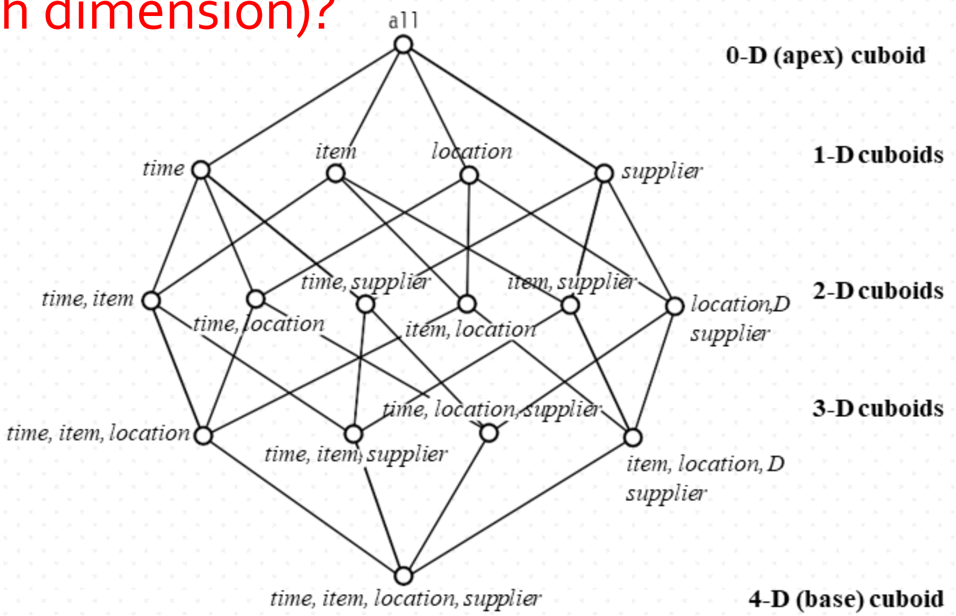
Suppose a cuboid has 3 dimensions (time, location, item) at specific dimension levels (date, city, product).

- Aggregate cells
  - (*, South Bend, milk)
  - (09/16, *, milk)
  - (09/16, South Bend, *)
  - (*, *, milk)
  - (*, South Bend, *)
  - (09/16, *, *)
  - (*, *, *),
    - called the Apex cell



(N-Dimensional) Data Cube

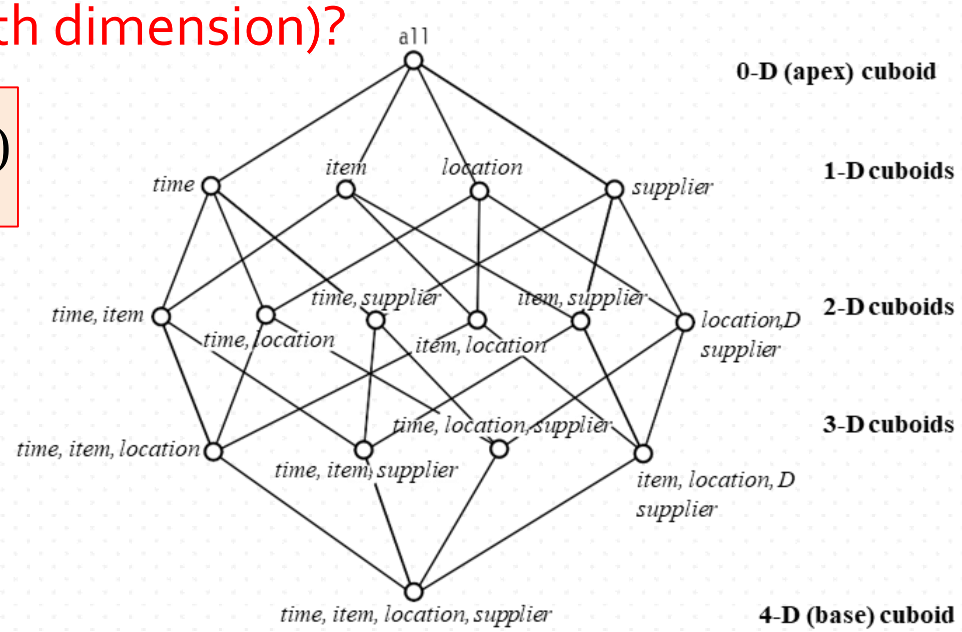
- Data cube can be viewed as a lattice of cuboids
  - The bottom-most cuboid is the base cuboid
  - The top-most cuboid (apex) contains only one cell
  - How many cuboids in an n-dimensional cube with L_i levels (at the i-th dimension)?



(N-Dimensional) Data Cube

- Data cube can be viewed as a lattice of cuboids
  - The bottom-most cuboid is the base cuboid
  - The top-most cuboid (apex) contains only one cell
  - How many cuboids in an n-dimensional cube with L_i levels (at the i-th dimension)?

$$T = \prod_{i=1}^{n} (L_i + 1)$$

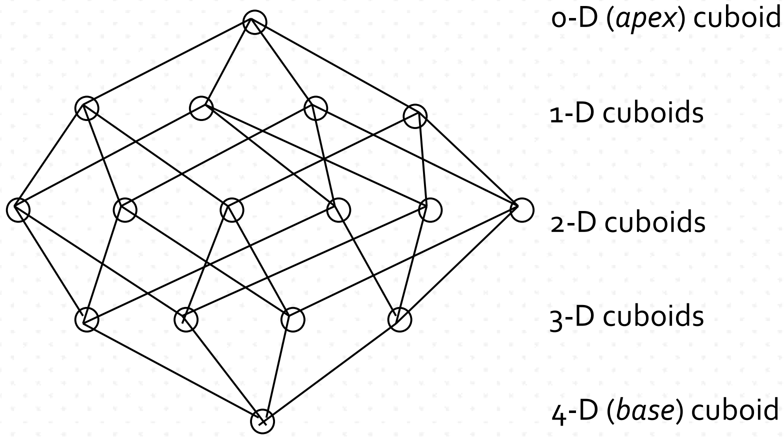


Preview: Data Cube Measures: Three Categories

- Distributive: if the result derived by applying the function to *n* aggregate values is the same as that derived by applying the function on all the data without partitioning
  - E.g., count(), sum(), min(), max()
- Algebraic: if it can be computed by an algebraic function with *M* arguments (where *M* is a bounded integer), each of which is obtained by applying a distributive aggregate function
  - avg(x) = sum(x) / count(x)
- Holistic: if there is no constant bound on the storage size needed to describe a sub-aggregate.
  - E.g., median(), mode(), rank()
- Q: How about standard_deviation(), Q1(), Q3()?

Efficient Data Cube Computation

- Materialization of data cube
  - **Full materialization**: Materialize every (cuboid)
  - **No materialization**: Materialize none (cuboid)
  - **Partial materialization**: Materialize some cuboids
    - Which cuboids to materialize?
      - Selection based on size, sharing, access frequency, etc.



Remember, Total Cuboids

$$\boxed{T = \prod_{i=1}^{n} (L_i + 1)}$$

Example

- Example: A cube with 100 dimensions
  - Suppose it contains only 2 base cells and the count of each cell is 1:
    - $\{(a_1, a_2, a_3, \ldots., a_{100}) : 1, (a_1, a_2, b_3, \ldots, b_{100}) : 1\}$
  - How many **aggregate cells** if “having count >= 1” (**non-empty**)?

Suppose it contains only 2 base cells: $\{(a_1, a_2, a_3, \ldots, a_{100}), (a_1, a_2, b_3, \ldots, b_{100})\}$

How many non-empty aggregate cells?

The total # of non-base cells should be 2 * (2^{100} – 1) – 4.

- (a1, a2, a3 . . . , a100) will generate 2^{100} - 1 non-base cells
- (a1, a2, b3, . . . , b100) will generate 2^{100} - 1 non-base cells

Among these, 4 cells are overlapped and thus minus 4 so we get:
2*2^{100} - 2 – 4

These 4 cells are:

- (a1, a2, *, …, *): 2
- (a1, *, *, …, *): 2
- (*, a2, *, …, *): 2
- (*, *, *, …, *): 2

Cube Materialization: Full Cube vs. Iceberg Cube

- Full cube vs. iceberg cube

  compute cube sales iceberg as

  select date, product, city, department, count(*)

  from salesInfo

  cube by date, product, city

  having count(*) >= min support
- Compute *only* the **cells** whose **measure** satisfies the iceberg condition
- Only a small portion of cells may be "above the water" in a sparse cube
- Ex.: Show only those cells whose **count** is no less than 100

Why Iceberg Cube?

- Advantages of computing iceberg cubes
  - No need to save nor show those cells whose value is below the threshold (iceberg condition)
  - Efficient methods may even avoid computing the un-needed, intermediate cells
  - Avoid explosive growth

Is Iceberg Cube Good Enough?

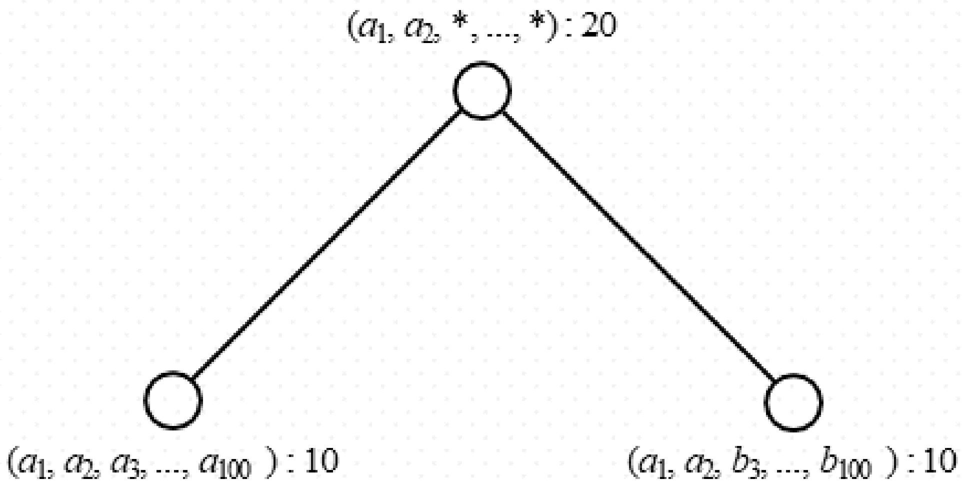
- Let cube P have only 2 base cells:
  - $\{(a_1, a_2, a_3 \ldots, a_{100})$:10
  - $(a_1, a_2, b_3, \ldots, b_{100})$:10}

How many cells will the iceberg cube contain if "having count(*) ≥ 10"?

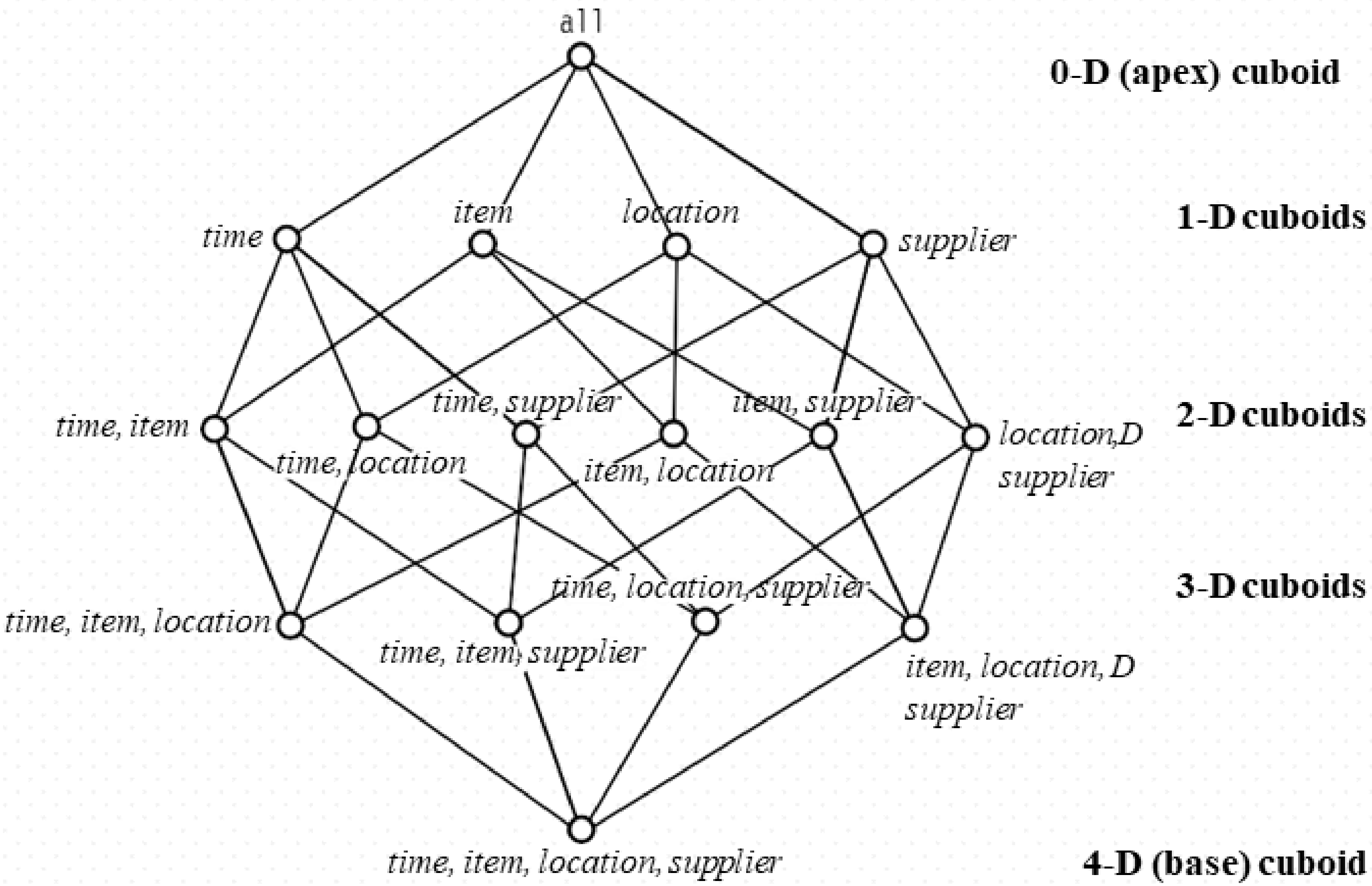
Answer: $2^{101} - 4$ (base+aggregate; still too big!)

Closed Cube & Cube Shell

- **Close cube:**
  - A cell c is ***closed*** if there exists no cell *d*, such that *d* is a descendant of *c*, and *d* has the same measure value as *c*
  - A ***closed cube*** is a cube consisting of only closed cells

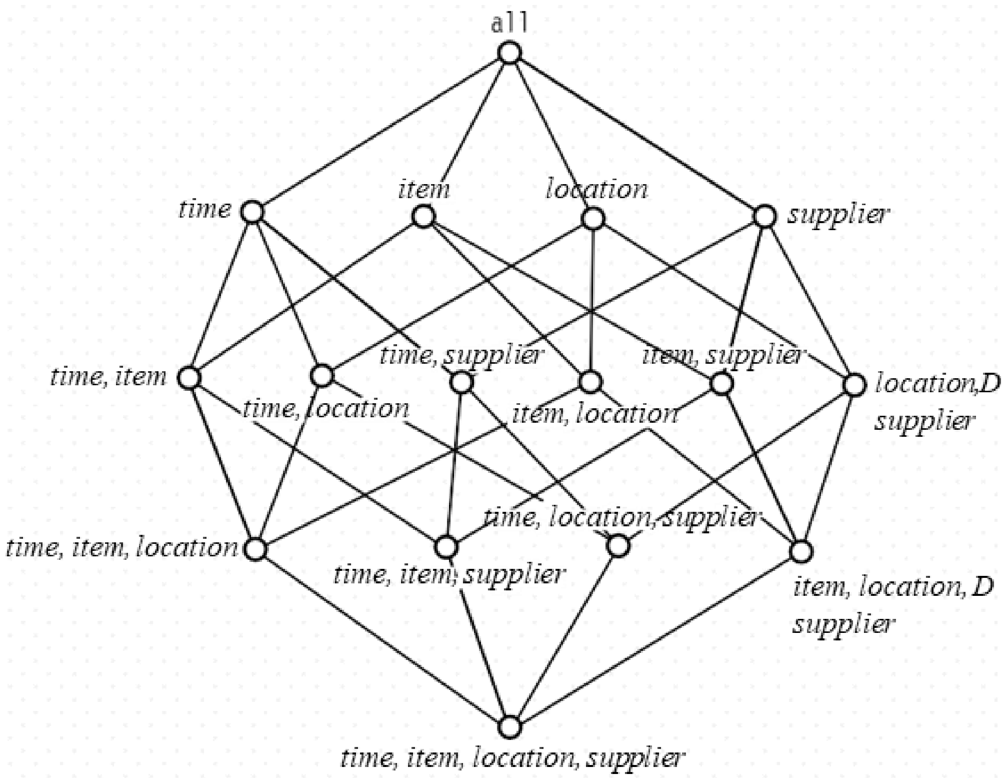


Smarter Manifestation of Cuboid



Thinking Back to Iceberg Cubes

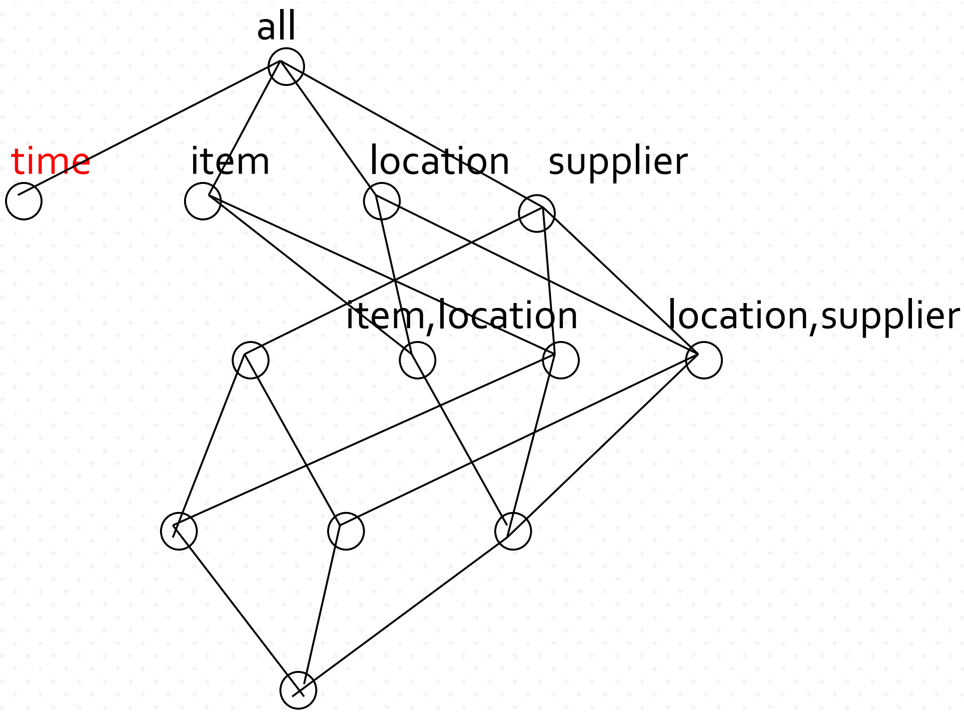
How Does This Help?



- Full cube vs. iceberg cube
  compute cube sales iceberg as
  select date, product, city, department, count(*)
  from salesInfo
  cube by date, product, city
  having count(*) >= min support
- Compute *only* the **cells** whose **measure** satisfies the iceberg condition
- Only a small portion of cells may be "above the water" in a sparse cube
- Ex.: Show only those cells whose **count** is no less than 100

Smarter Manifestation of Cuboid

Divides dimensions into partitions and facilitates **iceberg pruning**
If a partition does not satisfy *min_sup*, its **descendants** can be pruned



Principle

If an cuboid is frequent, then all of its subsets must also be frequent

- This principle holds true because of the following property of support measure:

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

  - The support of an item never exceeds that of its subsets
  - This is known as the **anti-monotone** property of support


TIME FOR A BREAK

Pattern Discovery

Patterns: A set of items, subsequences, or substructures that occur frequently together (or strongly correlated) in a data set

Pattern Discovery: Why Is It Important?

- Finding inherent regularities in a data set

- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Mining sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: Discriminative pattern-based analysis
  - Cluster analysis: Pattern-based subspace clustering

- Broad applications
  - Market basket analysis, cross-marketing, catalog design, sale campaign analysis, Web log analysis, biological sequence analysis

Applications

- Retail: *what sells with what*
- Marketing : *population segments, recommendations, etc.*
- Finance: *investment portfolios, "basket of stocks"*
- Biology: *genetics, microarrays, gene expressions*
- What code segments likely contain copy-and-paste bugs?

Pattern Mining Definitions - Basics

$I = \{i_1, i_2, \dots, i_n\}$: a set of literals, called **items**

Transaction (itemset) T: a set of items such that $T \subseteq I$

items

TID	Items
T1	Bread, milk
T2	Bread, diaper, beer, eggs
T3	Milk, diaper, beer, coke
T4	Bread, milk, diaper, beer
T5	Bread, milk, diaper, coke

Transaction

Frequent Patterns (Itemsets)

- *absolute support* of X:
  - The number of occurrences of an itemset X
    - Denoted as ($\boldsymbol{\sigma}$)
- *relative support of X:*
  - The fraction of transactions that contains X
    - i.e., the probability that a transaction contains X
    - Denoted as (s)
- An itemset X is *frequent* if the support of X is no less than a *minsup* threshold

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

Frequent Patterns (Itemsets)

- *relative support of X:*
  - The fraction of transactions that contains X
- Let *minsup = 50%*
  - Freq. 1-itemsets:
    - Beer: 3 (60%)
    - Nuts: 3 (60%)
    - Diaper: 4 (80%)
    - Eggs: 3 (60%)

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

Frequent Patterns (Itemsets)

- *relative support of X:*
  - The fraction of transactions that contains X
- Let *minsup = 50%*
  - Freq. 1-itemsets:
    - Beer: 3 (60%)
    - Nuts: 3 (60%)
    - Diaper: 4 (80%)
    - Eggs: 3 (60%)
  - Freq. 2-itemsets
    - {Beer, Diaper}: 3 (60%)

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

Frequent Patterns (Itemsets)

- *relative support of X:*
  - The fraction of transactions that contains X
- Let *minsup = 50%*
  - Freq. 1-itemsets:
    - Beer: 3 (60%)
    - Nuts: 3 (60%)
    - Diaper: 4 (80%)
    - Eggs: 3 (60%)
  - Freq. 2-itemsets
    - {Beer, Diaper}: 3 (60%)
  - **Freq k-itemset**
    - ?

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

Challenge: There Are Too Many Frequent Patterns!

- A long pattern contains a combinatorial number of sub-patterns
- How many frequent itemsets does the following TDB_1 contain?
  - TDB_1: T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$
  - Assuming (absolute) *minsup* = 1

1-itemsets: $\{a_1\}$: 2, $\{a_2\}$: 2, …, $\{a_{50}\}$: 2, $\{a_{51}\}$: 1, …, $\{a_{100}\}$: 1,

2-itemsets: $\{a_1, a_2\}$: 2, …, $\{a_1, a_{50}\}$: 2, $\{a_1, a_{51}\}$: 1 …, …, $\{a_{99}, a_{100}\}$: 1, …

99-itemsets: $\{a_1, a_2, \ldots, a_{99}\}$: 1, …, $\{a_2, a_3, \ldots, a_{100}\}$: 1

100-itemset: $\{a_1, a_2, \ldots, a_{100}\}$: 1

- In total: $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1$ sub-patterns!

A too large a set to compute or store!

Several ways to reduce the computational complexity:

Today

- **Reduce the number of candidate itemsets**
  - **Apriori Algorithm**

Next

- **Reduce the number of comparisons FP Growth**

Expressing Patterns in Compressed Form: Closed Patterns

- Solution 1: **Closed patterns**: A pattern (itemset) X is closed if X is *frequent*, and there exists *no super-pattern* Y ⊃ X, ***with the same support*** **as X**
  - Let Transaction DB TDB_1: T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$
  - Suppose *minsup* = 1. How many closed patterns does TDB_1 contain?
    - Two: P_1: "$\{a_1, \ldots, a_{50}\}$: 2"; P_2: "$\{a_1, \ldots, a_{100}\}$: 1"
- Closed pattern is a lossless compression of frequent patterns
  - Reduces the # of patterns but does not lose the support information!
  - You will still be able to say: "$\{a_2, \ldots, a_{40}\}$: 2", "$\{a_5, a_{51}\}$: 1"

Expressing Patterns in Compressed Form: Max-Patterns

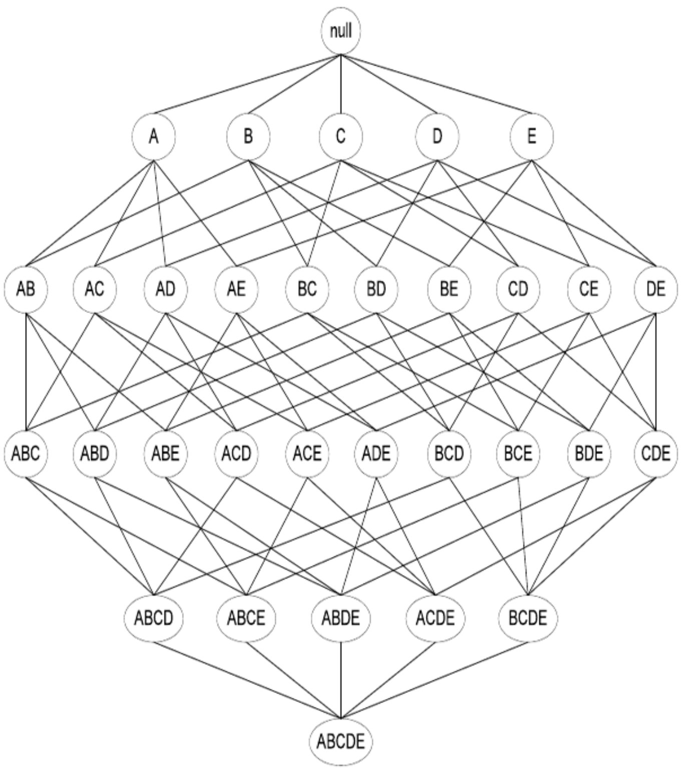
- Solution 2: **Max-patterns**: A pattern X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ⊃ X, ~~with the same support as X~~
- Difference from close-patterns?
  - Do not care the real support of the sub-patterns of a max-pattern
  - Let Transaction DB TDB_1: T_1: {a_1, …, a_{50}}; T_2: {a_1, …, a_{100}}
  - Suppose *minsup* = 1. How many max-patterns does TDB_1 contain?
    - One: P: "{a_1, …, a_{100}}: 1"
- Max-pattern is a lossy compression!
  - We only know {a_1, …, a_{40}} is frequent
  - But we do not know the real support of {a_1, …, a_{40}}, …, any more!
- Thus in many applications, mining closed-patterns is more desirable than mining max-patterns

The Downward Closure Property of Frequent Patterns: Apriori

- Observation: From $TDB_{1:}$ T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$
  - We get a frequent itemset: $\{a_1, \ldots, a_{50}\}$
  - Also, its subsets are all frequent: $\{a_1\}, \{a_2\}, \ldots, \{a_{50}\}, \{a_1, a_2\}, \ldots, \{a_1, \ldots, a_{49}\}, \ldots$
  - There must be some hidden relationships among frequent patterns!
- The downward closure (also called "Apriori") property of frequent patterns
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - Every transaction containing {beer, diaper, nuts} also contains {beer, diaper}
  - **Apriori: Any subset of a frequent itemset must be frequent**
- Efficient mining methodology
  - If any subset of an itemset S is infrequent, then there is no chance for S to be frequent—why do we even have to consider S!?

A sharp knife for pruning!

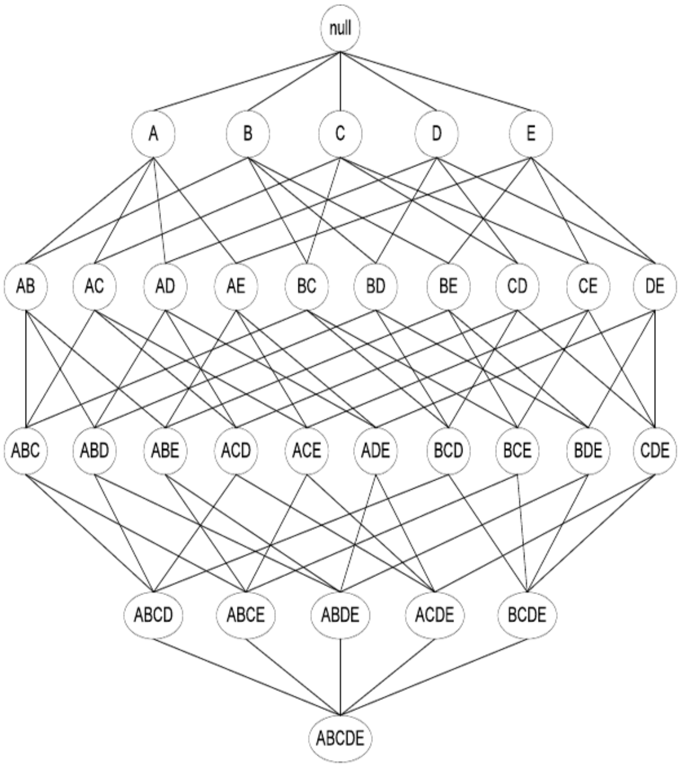
Let's Represent this Differently

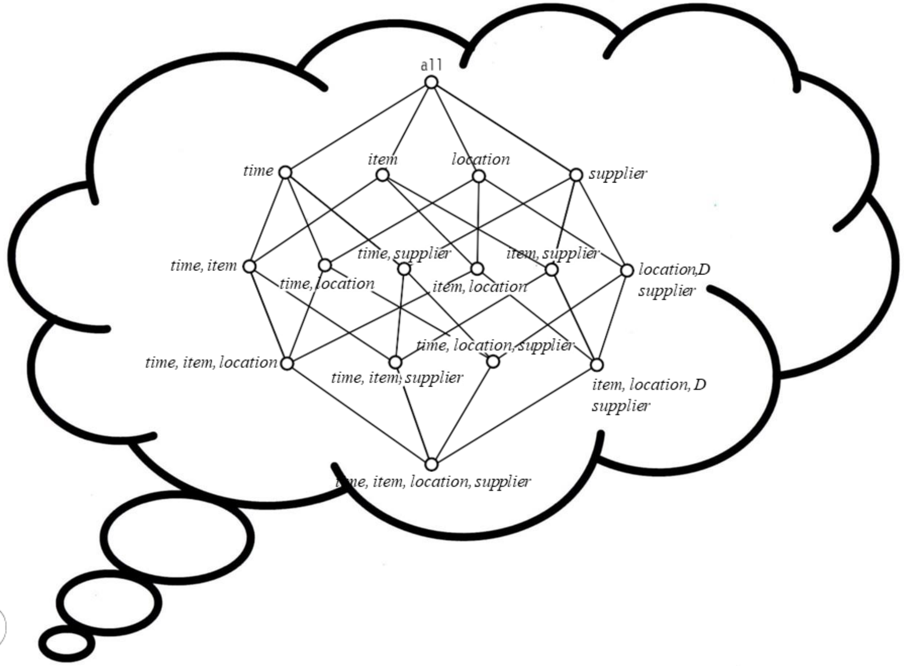


Itemset lattice

- For a set with *n* items:
  - 2^n-1 possible itemsets
- Each of these is called a *candidate* frequent itemset

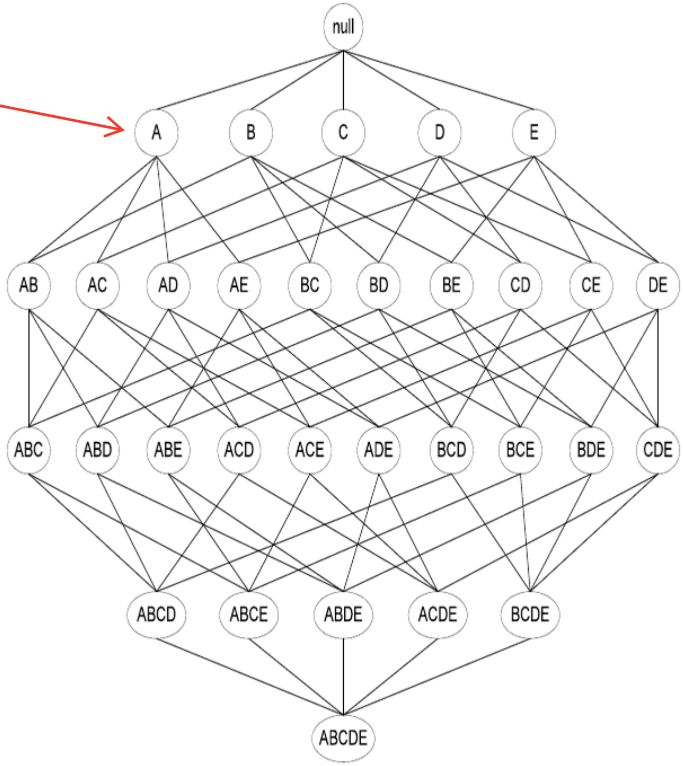
Let's Represent this Differently





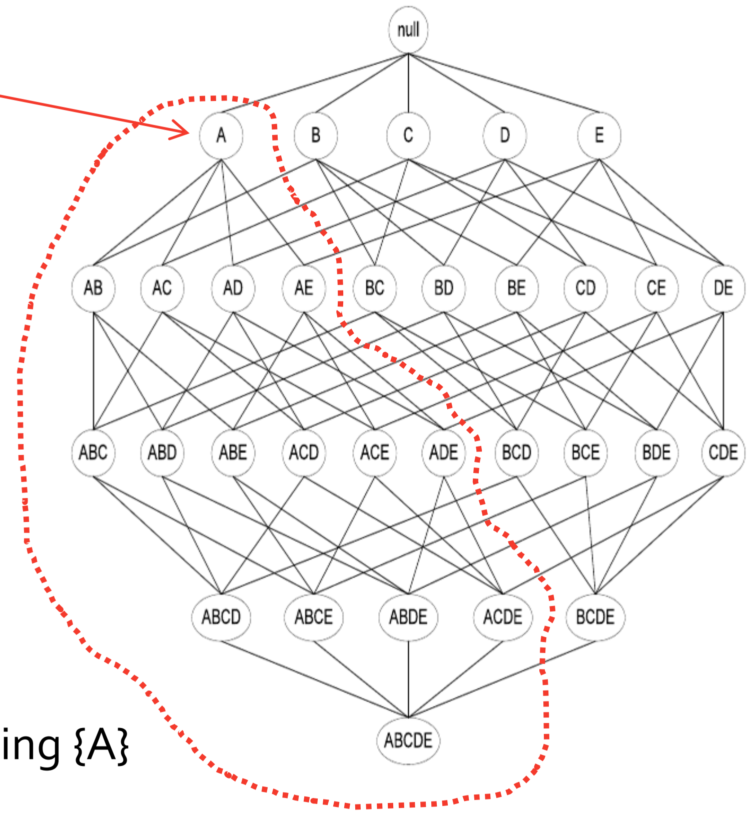
Apriori Algorithm

If {A} is infrequent



Apriori Algorithm

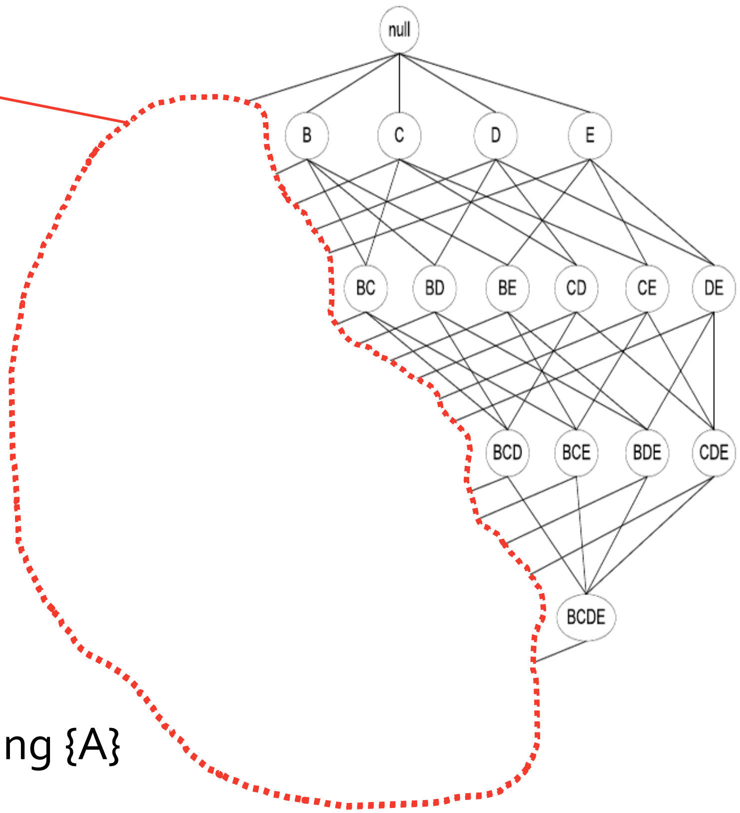
If {A} is infrequent



Prune all itemsets containing {A}

Apriori Algorithm

If {A} is infrequent



Prune all itemsets containing {A}

Apriori: A Candidate Generation & Test Approach

- Outline of Apriori (level-wise, candidate generation and test)
  - Initially, scan DB once to get frequent 1-itemset
  - Repeat
    - Generate length-(k+1) candidate itemsets from length-k frequent itemsets
    - Test the candidates against DB to find **frequent** (k+1)-itemsets
    - Set k := k +1
  - Until no frequent or candidate set can be generated
  - Return all the frequent itemsets derived

The Apriori Algorithm (Pseudo-Code)

C_k: Candidate itemset of size k
F_k : Frequent itemset of size k

K := 1;
F_k := {frequent items}; // frequent 1-itemset

While (F_k != ∅) **do {** // when F_k is non-empty
  C_{k+1} := candidates generated from F_k; // candidate generation
  Derive F_{k+1} by counting candidates in C_{k+1} with respect to *TDB* at minsup;
  k := k + 1
}

return $\cup_k F_k$ // return F_k generated at each level

Apriori Algorithm Illustrated

D

TID	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	BE

$minsup_{count} = 2$

Apriori Algorithm Illustrated

D

TID	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	BE

$minsup_{count} = 2$

C_1

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

Apriori Algorithm Illustrated

D

TID	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	BE

$$minsup_{count} = 2$$

C_1

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

F_1

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

C_2

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

Apriori Algorithm Illustrated

D

TID	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	BE

$$minsup_{count} = 2$$

C_1

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

F_1

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

C_2

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

C_2

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

Apriori Algorithm Illustrated

D

TID	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	BE

$$minsup_{count} = 2$$

C_1

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

F_1

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

C_2

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

C_2

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

F_2

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

Apriori Algorithm Illustrated Cont.

D

TID	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	BE

$minsup_{count} = 2$

F_1

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

F_2

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

C_3

Itemset	Support
{A,B,C}	1
{A,C,E}	1
{B,C,E}	2
{A,B,E}	1

F_3

Itemset	Support
{B,C,E}	2

C_4

Itemset	Support
{A,B,C,E}	1

$F_4 = \emptyset$

Done !

Recall the Apriori principle: *All subsets of a frequent subset must also be frequent*

Apriori Algorithm Illustrated Result

D

TID	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	BE

$minsup_{count} = 2$

F_1

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

F_2

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

F_3

Itemset	Support
{B,C,E}	2

Can this be improved?

Can this Be Improved

D

TID	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	BE

$$minsup_{count} = 2$$

F_1

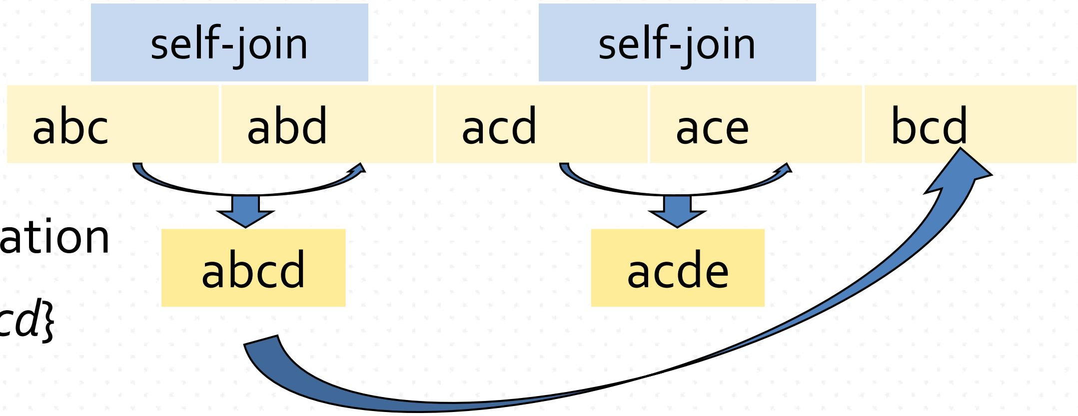
Itemset	Support
{A}	2
{B}	3
{C}	3
{E}	3

F_2

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

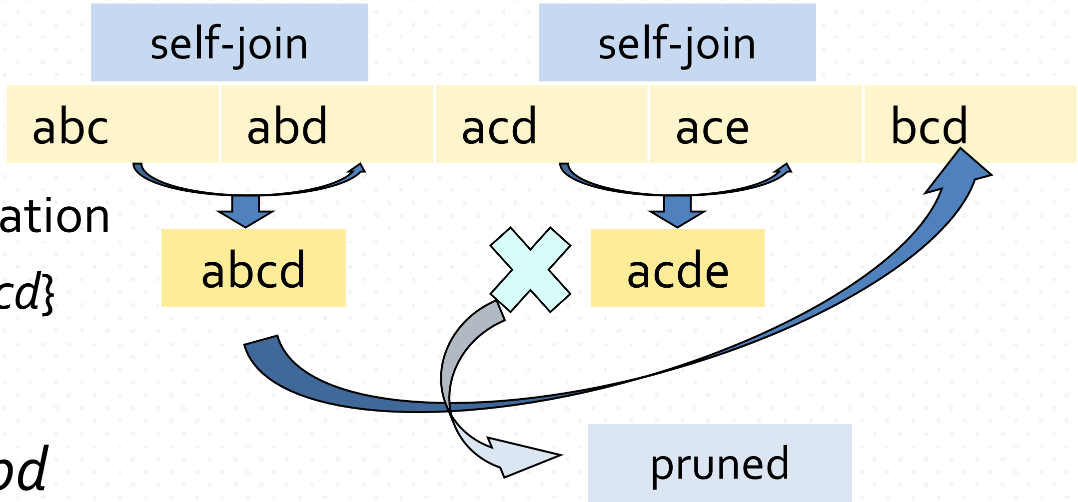
Apriori: Implementation Tricks

- Step 1: self-joining F_k
- Step 2: pruning
  - Example of candidate-generation
    - $F_3 = \{abc, abd, acd, ace, bcd\}$
  - Self-joining: $F_3 * F_3$
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*



Apriori: Implementation Tricks

- Step 1: self-joining F_k
- Step 2: pruning
  - Example of candidate-generation
    - $F_3 = \{abc, abd, acd, ace, bcd\}$
  - Self-joining: $F_3 * F_3$
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in F_3
  - $C_4 = \{abcd\}$



In Class

D

TID	Items
1	A, B,C
2	B,D
3	B,C
4	A,B,D
5	A,C
6	B,C
7	A,C
8	A,B,C,E
9	A,B,E

$$minsup_{count} = 2$$

In Class (Lets Try The Trick)

D

TID	Items
1	A, B,C
2	B,D
3	B,C
4	A,B,D
5	A,C
6	B,C
7	A,C
8	A,B,C,E
9	A,B,E

F_1

Itemset	Support
{A}	6
{B}	7
{C}	6
{D}	2
{E}	2

$minsup_{count} = 2$

In Class

D

TID	Items
1	A, B,C
2	B,D
3	B,C
4	A,B,D
5	A,C
6	B,C
7	A,C
8	A,B,C,E
9	A,B,E

F_1

Itemset	Support
{A}	6
{B}	7
{C}	6
{D}	2
{E}	2

F_2

Itemset	Support
{A,B}	4
{A,C}	4
{A,E}	2
{B,C}	4
{B,D}	2
{B,E}	2

$$minsup_{count} = 2$$

In Class

D

TID	Items
1	A, B,C
2	B,D
3	B,C
4	A,B,D
5	A,C
6	B,C
7	A,C
8	A,B,C,E
9	A,B,E

F_1

Itemset	Support
{A}	6
{B}	7
{C}	6
{D}	2
{E}	2

F_2

Itemset	Support
{A,B}	4
{A,C}	4
{A,E}	2
{B,C}	4
{B,D}	2
{B,E}	2

F_3

Itemset	Support
{A,B,C}	2
{A,B,E}	2

$minsup_{count} = 2$

From Frequent Itemsets to Association Rules

- Association rules: $X \rightarrow Y$
  - *If I buy X then I will buy Y*

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

Support Is Not Enough for a Rule

- Note that the support of a rule $X \Rightarrow Y$ depends only on the support of $X \cup Y$
  - All rules below have the same support:
    - $\{Beer, Diapers\} \Rightarrow \{Milk\}$
      $\{Diapers, Milk\} \Rightarrow \{Beer\}$
      $\{Milk\} \Rightarrow \{Beer, Diapers\}$
      $\{Beer\} \Rightarrow \{Milk, Diapers\}$

Support and **Confidence**

- Example:
  - 5% of transactions contain both these items
  - 30% of the transactions containing beer also contain diapers

- Beer⇒Diapers(0.05, 0.30)
  - 5% – Support of the rule
  - 30% – Confidence of the rule

Association rules

- Why use *support* and *confidence*?
  - Rules with low support may occur simply by chance
  - Confidence measures the reliability of the inference made by a rule.
    - For $X \Rightarrow Y$, the higher the confidence, the more likely it is for Y to be present in transactions containing X

The Association Rule Mining Problem

Given a set of transactions T, find all the rules having <u>support $\geq minsup$</u> and <u>confidence $\geq minconf$</u>

where $minsup$ and $minconf$ are the corresponding support and confidence thresholds.

Mining Association Rules

Two-step approach:

We know how to do this…

- Frequent Itemset Generation
  - Generate all item sets whose support ≥ $minsup$

Mining Association Rules

Two-step approach:

- Frequent Itemset Generation
  - Generate all item sets whose support ≥ $minsup$
- Rule Generation
  - Generate high confidence (strong) rules from each frequent itemset

Association Rules

- Association rules: $X \rightarrow Y$ (s, c)
  - Support, *s*: The probability that a transaction contains $X \cup Y$
  - Confidence, *c: The* conditional probability that a transaction containing X also contains *Y*
  - $c = sup(X \cup Y) / sup(X)$

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

Association rule mining: Find all of the rules, $X \rightarrow Y$, with minimum support and confidence

- Frequent itemsets: Let *minsup = 50%*
  - Freq. 1-itemsets:
  - Freq. 2-itemsets:

Association Rules

- Association rules: $X \rightarrow Y$ (s, c)
  - Support, *s*: The probability that a transaction contains $X \cup Y$
  - Confidence, *c: The* conditional probability that a transaction containing X also contains *Y*
  - c = sup($X \cup Y$) / sup(X)

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

Association rule mining: Find all of the rules, $X \rightarrow Y$, with minimum support and confidence

- Frequent itemsets: Let *minsup = 50%*

  Freq. 1-itemsets:

  Beer: 3 (60%)

  Nuts: 3 (60%)

  Diaper: 4 (80%)

  Eggs: 3 (60%)

  Freq. 2-itemsets

  {Beer, Diaper}: 3 (60%)

Association Rules

- Association rules: $X \rightarrow Y$ (s, c)
  - Support, *s*: The probability that a transaction contains $X \cup Y$
  - Confidence, *c: The* conditional probability that a transaction containing X also contains *Y*
  - $c = sup(X \cup Y) / sup(X)$

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

Association rule mining: Find all of the rules, $X \rightarrow Y$, with minimum support and confidence

- Frequent itemsets: *minsup = 50%*

  Freq. 1-itemsets:

  Beer: 3 (60%)

  Nuts: 3 (60%)

  Diaper: 4 (80%)

  Eggs: 3 (60%)

  Freq. 2-itemsets

  {Beer, Diaper}: 3 (60%)

- Association rules: Let *minconf = 50%*

  Beer → *Diaper* (60%, 100%)

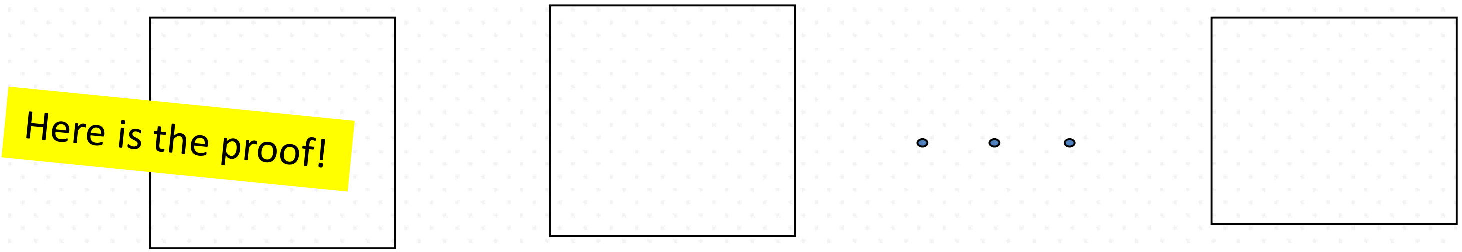
  Diaper → *Beer* (60%, 75%)

Association rules – Caution!

- Association rules results should be interpreted with caution
  - They do not imply causality, which requires extra knowledge of your data
  - Instead, they simply imply a strong co-occurrence relationship between items

Partitioning for Parallelization

- Theorem: Any itemset that is potentially frequent in TDB must be frequent in at least one of the partitions of TDB



$$TDB_1 + TDB_2 + \cdots + TDB_k = TDB$$

$$sup_1(X) < \sigma|TDB_1| \quad sup_2(X) < \sigma|TDB_2| \quad \cdots \quad sup_k(X) < \sigma|TDB_k| \quad sup(X) < \sigma|TDB|$$

- Method: (A. Savasere, E. Omiecinski and S. Navathe, VLDB'95)
  - Scan 1: Partition database and find local frequent patterns
  - Scan 2: Consolidate global frequent patterns

Apriori References

- Agrawal & Srikant @VLDB'94
- Mannila, et al. @ KDD' 94)
- Scalable mining Methods: Three major approaches
  - **Level-wise, join-based approach: Apriori** (Agrawal & Srikant@VLDB'94)
  - **Vertical data format approach: Eclat** (Zaki, Parthasarathy, Ogihara, Li@KDD'97)
  - **Frequent pattern projection and growth: FPgrowth** (Han, Pei, Yin @SIGMOD'00)

Apriori: Improvements and Alternatives

- Reduce passes of transaction database scans
  - Partitioning (e.g., Savasere, et al., 1995)
  - Dynamic itemset counting (Brin, et al., 1997)
- Shrink the number of candidates
  - Hashing (e.g., DHP: Park, et al., 1995)
  - Pruning by support lower bounding (e.g., Bayardo 1998)
  - Sampling (e.g., Toivonen, 1996)
- Exploring special data structures
  - Tree projection (Agarwal, et al., 2001)
  - H-miner (Pei, et al., 2001)
  - Hypecube decomposition (e.g., LCM: Uno, et al., 2004)

Discussion

- How do you define frequent patterns in scientific knowledge discovery and technology exploration?
  - {"social spam detection", "matrix factorization"}
  - {"social spam detection", "Twitter"}
  - …
- Do you believe in the association: Diapers → Beer?

References – Data Cube

- S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. VLDB’96
- K. Beyer and R. Ramakrishnan. Bottom-Up Computation of Sparse and Iceberg CUBEs.. SIGMOD’99
- J. Han, J. Pei, G. Dong, K. Wang. Efficient Computation of Iceberg Cubes With Complex Measures. SIGMOD’01
- L. V. S. Lakshmanan, J. Pei, and J. Han, Quotient Cube: How to Summarize the Semantics of a Data Cube, VLDB'02
- X. Li, J. Han, and H. Gonzalez, High-Dimensional OLAP: A Minimal Cubing Approach, VLDB'04
- X. Li, J. Han, Z. Yin, J.-G. Lee, Y. Sun, “Sampling Cube: A Framework for Statistical OLAP over Sampling Data”, SIGMOD’08
- K. Ross and D. Srivastava. Fast computation of sparse datacubes. VLDB’97
- D. Xin, J. Han, X. Li, B. W. Wah, Star-Cubing: Computing Iceberg Cubes by Top-Down and Bottom-Up Integration, VLDB'03
- Y. Zhao, P. M. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. SIGMOD’97
- D. Burdick, P. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan. OLAP over uncertain and imprecise data. VLDB’05

References Data Cube (cont.)

- R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. ICDE’97
- B.-C. Chen, L. Chen, Y. Lin, and R. Ramakrishnan. Prediction cubes. VLDB’05
- B.-C. Chen, R. Ramakrishnan, J.W. Shavlik, and P. Tamma. Bellwether analysis: Predicting global aggregates from local regions. VLDB’06
- Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang, Multi-Dimensional Regression Analysis of Time-Series Data Streams, VLDB'02
- R. Fagin, R. V. Guha, R. Kumar, J. Novak, D. Sivakumar, and A. Tomkins. Multi-structural databases. PODS’05
- J. Han. Towards on-line analytical mining in large databases. SIGMOD Record, 27:97–107, 1998
- T. Imielinski, L. Khachiyan, and A. Abdulghani. Cubegrades: Generalizing association rules. Data Mining & Knowledge Discovery, 6:219–258, 2002.
- R. Ramakrishnan and B.-C. Chen. Exploratory mining in cube space. Data Mining and Knowledge Discovery, 15:29–54, 2007.
- K. A. Ross, D. Srivastava, and D. Chatziantoniou. Complex aggregation at multiple granularities. EDBT'98
- S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. EDBT'98
- G. Sathe and S. Sarawagi. Intelligent Rollups in Multidimensional OLAP Data. *VLDB'01*

References – Aprioi

- R. Agrawal, T. Imielinski, and A. Swami, “Mining association rules between sets of items in large databases”, in Proc. of SIGMOD'93
- R. J. Bayardo, “Efficiently mining long patterns from databases”, in Proc. of SIGMOD'98
- N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, “Discovering frequent closed itemsets for association rules”, in Proc. of ICDT'99
- J. Han, H. Cheng, D. Xin, and X. Yan, “Frequent Pattern Mining: Current Status and Future Directions”, Data Mining and Knowledge Discovery, 15(1): 55-86, 2007
- R. Agrawal and R. Srikant, “Fast algorithms for mining association rules”, VLDB'94
- A. Savasere, E. Omiecinski, and S. Navathe, “An efficient algorithm for mining association rules in large databases”, VLDB'95
- J. S. Park, M. S. Chen, and P. S. Yu, “An effective hash-based algorithm for mining association rules”, SIGMOD'95
- S. Sarawagi, S. Thomas, and R. Agrawal, “Integrating association rule mining with relational database systems: Alternatives and implications”, SIGMOD'98
- M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, “Parallel algorithm for discovery of association rules”, Data Mining and Knowledge Discovery, 1997
- J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation”, SIGMOD’00

References Aprioi (cont.)

- M. J. Zaki and Hsiao, "CHARM: An Efficient Algorithm for Closed Itemset Mining", SDM'02
- J. Wang, J. Han, and J. Pei, "CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets", KDD'03
- C. C. Aggarwal, M.A., Bhuiyan, M. A. Hasan, "Frequent Pattern Mining Algorithms: A Survey", in Aggarwal and Han (eds.): Frequent Pattern Mining, Springer, 2014
- C. C. Aggarwal and P. S. Yu. A New Framework for Itemset Generation. PODS'98
- S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. SIGMOD'97
- M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. CIKM'94
- E. Omiecinski. Alternative Interest Measures for Mining Associations. TKDE'03
- P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the Right Interestingness Measure for Association Patterns. KDD'02
- T. Wu, Y. Chen and J. Han, Re-Examination of Interestingness Measures in Pattern Mining: A Unified Framework, Data Mining and Knowledge Discovery, 21(3):371-397, 2010