Responsible Data Science

Association rule mining
data profiling continued

Prof. Julia Stoyanovich

Center for Data Science &
Computer Science and Engineering
New York University

data RESPONSIBLY

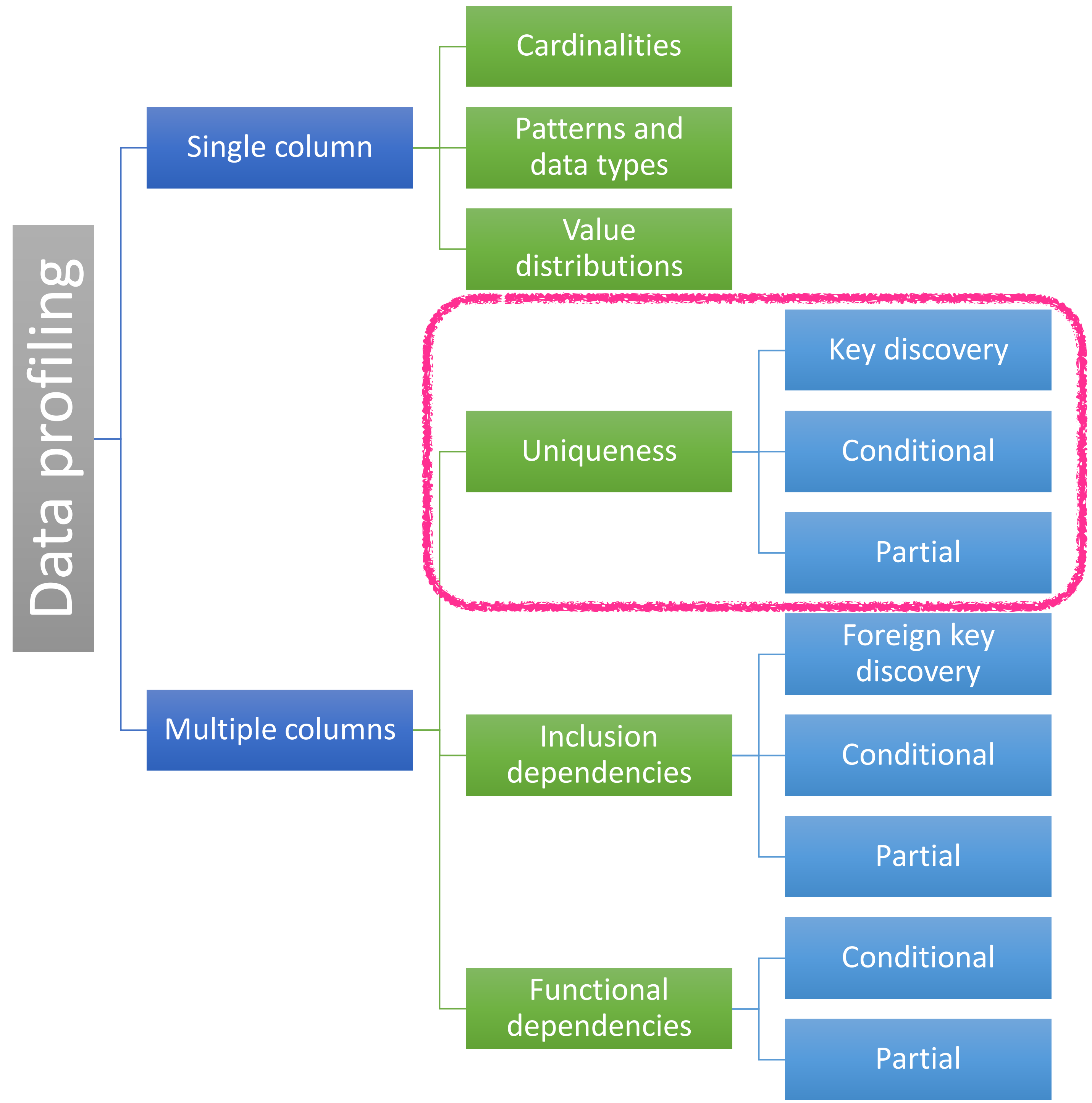
NYU

Classification of data profiling tasks

[Abedjan, Golab, Naumann; *SIGMOD 2017*]

	A	B	C	D	E	F	G	H
1	UID	sex	race	MarriageSta	DateOfBirth	age	juv_fel_cour	decile_score
2	1	0	1	1	4/18/47	69	0	1
3	2	0	2	1	1/22/82	34	0	3
4	3	0	2	1	5/14/91	24	0	4
5	4	0	2	1	1/21/93	23	0	8
6	5	0	1	2	1/22/73	43	0	1
7	6	0	1	3	8/22/71	44	0	1
8	7	0	3	1	7/23/74	41	0	6
9	8	0	1	2	2/25/73	43	0	4
10	9	0	3	1	6/10/94	21	0	3
11	10	0	3	1	6/1/88	27	0	4
12	11	1	3	2	8/22/78	37	0	1
13	12	0	2	1	12/2/74	41	0	4
14	13	1	3	1	6/14/68	47	0	1
15	14	0	2	1	3/25/85	31	0	3
16	15	0	4	4	1/25/79	37	0	1
17	16	0	2	1	6/22/90	25	0	10
18	17	0	3	1	12/24/84	31	0	5
19	18	0	3	1	1/8/85	31	0	3
20	19	0	2	3	6/28/51	64	0	6
21	20	0	2	1	11/29/94	21	0	9
22	21	0	3	1	8/6/88	27	0	2
23	22	1	3	1	3/22/95	21	0	4
24	23	0	4	1	1/23/92	24	0	4
25	24	0	3	3	1/10/73	43	0	1
26	25	0	1	1	8/24/83	32	0	3
27	26	0	2	1	2/8/89	27	0	3
28	27	1	3	1	9/3/79	36	0	3

relational data (here: just one table)



Discovering uniques

Given a relation schema ***R*** *(A, B, C, D)* and a relation instance ***r***, a **unique column combination** (or a **"unique"** for short) is a set of attributes ***X*** whose **projection** contains no duplicates in ***r***

$Episodes(season, num, title, viewers)$

season	num	title	viewers
1	1	Winter is Coming	2.2 M
1	2	The Kingsroad	2.2 M
2	1	The North Remembers	3.9 M

Projection is a relational algebra operation that takes as input relation ***R*** and returns a new relation ***R'*** with a subset of the columns of ***R***.

$\pi_{season}(Episodes)$

season
1
1
2

non-unique

$\pi_{season,num}(Episodes)$

season	num
1	1
1	2
2	1

unique

$\pi_{title}(Episodes)$

title
Winter is Coming
The Kingsroad
The North Remembers

unique

Discovering uniques

Given a relation schema ***R*** *(A, B, C, D)* and a relation instance ***r***, a **unique column combination** (or a **"unique"** for short) is a set of attributes ***X*** whose **projection** contains no duplicates in ***r***

Episodes(season, num, title, viewers)

season	num	title	viewers
1	1	Winter is Coming	2.2 M
1	2	The Kingsroad	2.2 M
2	1	The North Remembers	3.9 M

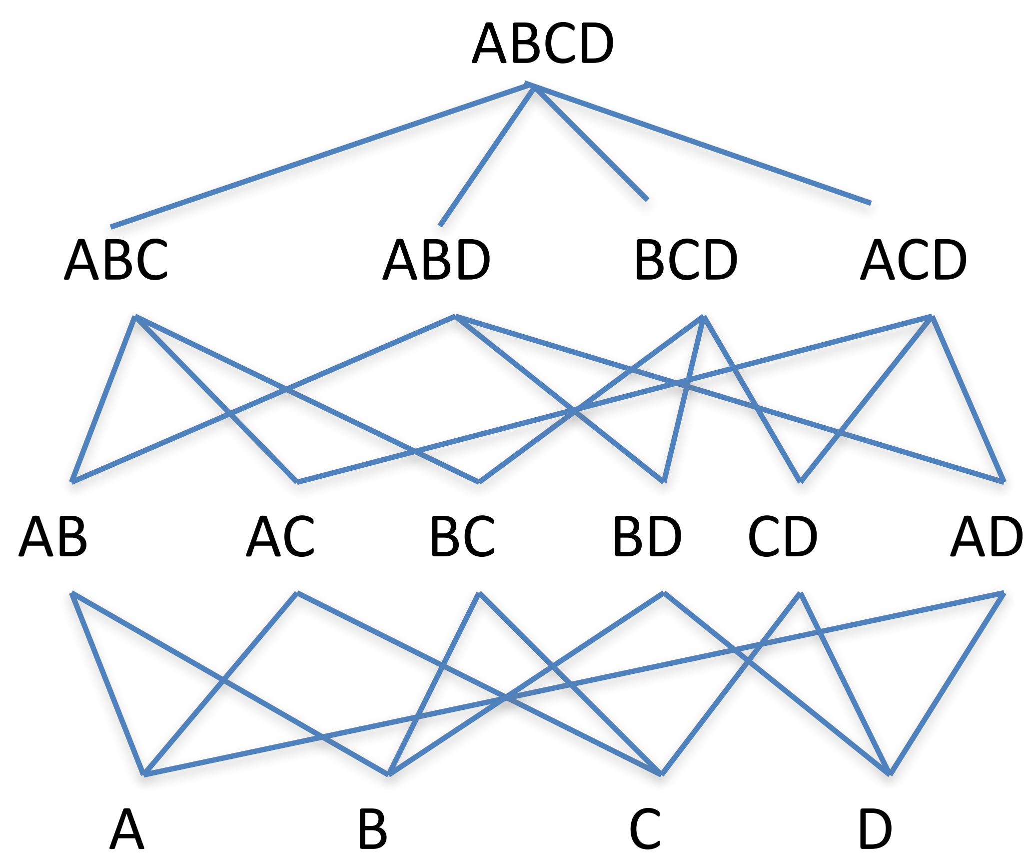
Projection is a relational algebra operation that takes as input relation ***R*** and returns a new relation ***R'*** with a subset of the columns of ***R***.

- Recall that more than one set of attributes **X** may be unique
- It may be the case that **X** and **Y** are both unique, and that they are not disjoint. When is this interesting?

Discovering uniques

R (A, B, C, D)

attribute lattice of ***R***



$$\binom{4}{4} = 1$$

$$\binom{4}{3} = 4$$

$$\binom{4}{2} = 6$$

$$\binom{4}{1} = 4$$

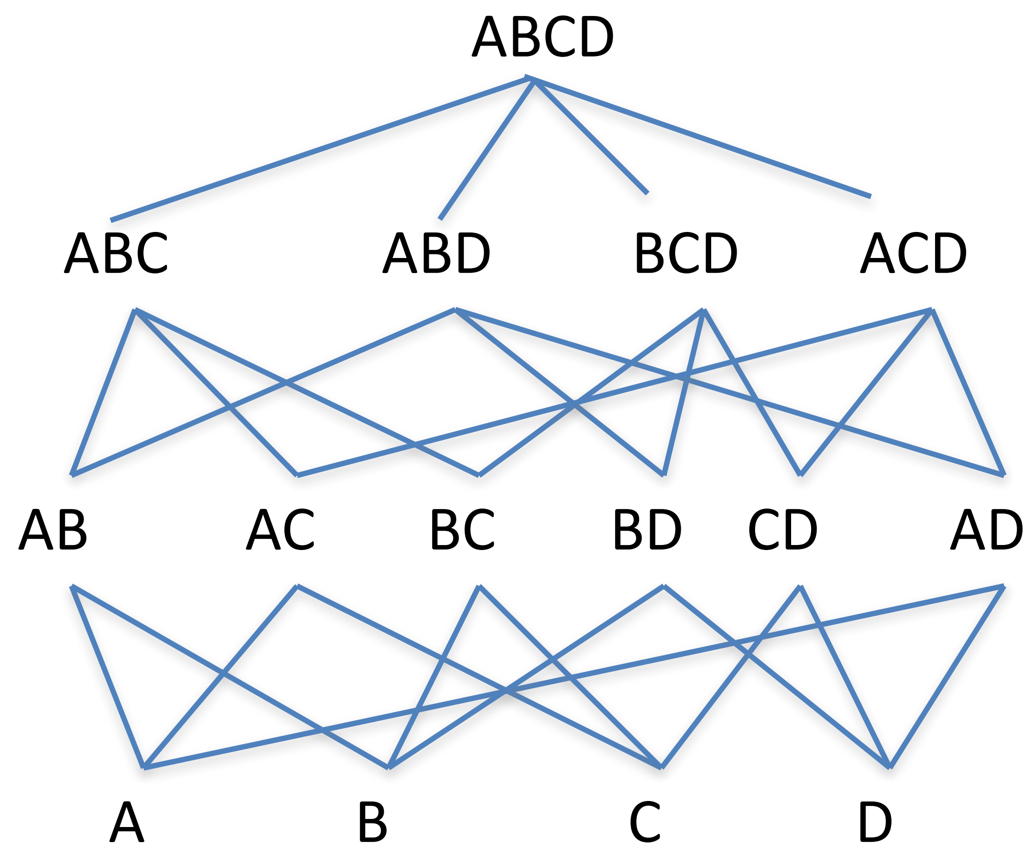
What's the size of the attribute lattice of ***R***?

Look at all attribute combinations?

Discovering uniques

R (A, B, C, D)

attribute lattice of R



- If **X** is unique, then what can we say about its **superset Y**?
- If **X** is non-unique, then what can we say about its **subset Z**?

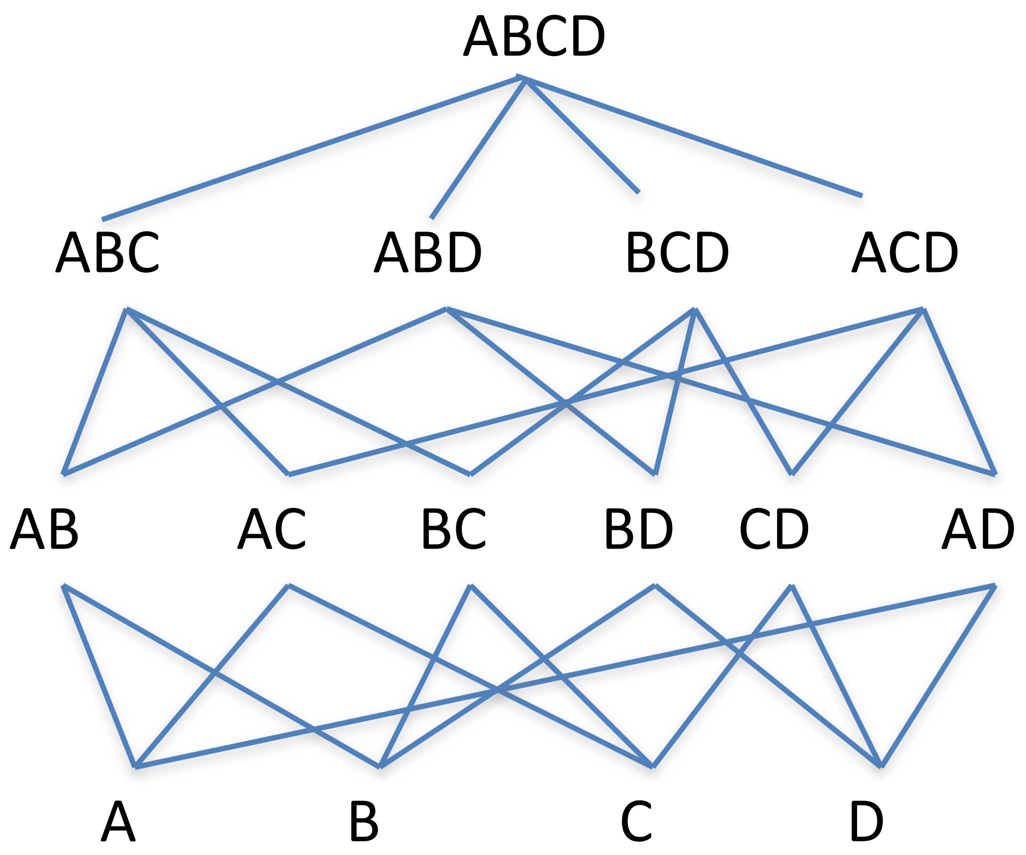
Discovering uniques

Given a relation schema ***R*** *(A, B, C, D)* and a relation instance ***r***, a **unique column combination** (or a **"unique"** for short) is a set of attributes ***X*** whose **projection** contains no duplicates in ***r***

Given a relation schema ***R*** *(A, B, C, D)* and a relation instance ***r***, a set of attributes ***Y*** is **non-unique** if its projection contains duplicates in ***r***

X is **minimal unique** if every subset ***Y*** of ***X*** is non-unique

Y is maximal non-unique if every superset ***X*** of ***Y*** is unique



From uniques to candidate keys

Given a relation schema ***R*** *(A, B, C, D)* and a relation instance ***r***, a **unique column combination** is a set of attributes ***X*** whose **projection** contains no duplicates in ***r***

$$Episodes(season, num, title, viewers)$$

season	num	title	viewers
1	1	Winter is Coming	2.2 M
1	2	The Kingsroad	2.2 M
2	1	The North Remembers	3.9 M

A set of attributes is a **candidate key** for a relation if:
(1) no two distinct tuples can have the same values for all key attributes (candidate key **uniquely identifies** a tuple), *and*
(2) this is not true for any subset of the key attributes (candidate key **is minimal**)

A minimal unique of a relation instance is a (possible) candidate key of the relation schema. To find all possible candidate keys, find all minimal uniques in a relation instance.

association rule mining

The early days of data mining

- Problem formulation due to Agrawal, Imielinski, Swami, SIGMOD 1993
- Solution: the **Apriori** algorithm by Agrawal & Srikant, VLDB 1994
- Initially for **market-basket data** analysis, has many other applications, we'll see one today
- We wish to answer two related questions:
  - **Frequent itemsets:** Which items are often purchased together, e.g., milk and cookies are often bought together
  - **Association rules:** Which items will likely be purchased, based on other purchased items, e.g., if diapers are bought in a transaction, beer is also likely bought in the same transaction

Market-basket data

- $I = \{i_1, i_2, \ldots, i_m\}$ is the set of available items, e.g., a product catalog of a store
- $X \subseteq I$ is an **itemset**, e.g., {milk, bread, cereal}
- **Transaction** t is a set of items purchased together, $t \subseteq I$, has a transaction id (TID)

  t_1: {bread, cheese, milk}

  t_2: {apple, eggs, salt, yogurt}

  t_3: {biscuit, cheese, eggs, milk}
- Database T is a set of transactions $\{t_1, t_2, \ldots, t_n\}$
- A transaction t **supports** an itemset X if $X \subseteq t$
- Itemsets supported by at least ***minSupp*** transactions are called **frequent itemsets**

minSupp, which can be a number or a percentage, is specified by the user

Itemsets

TID	Items
1	A
2	A C
3	A B D
4	A C
5	A B C
6	A B C

minSupp = 2 transactions

How many possible itemsets are there (excluding the empty itemset)?

$2^4 - 1 = 15$

	itemset	support
★	A	6
★	B	3
★	C	4
	D	1
★	A B	3
★	A C	4
	A D	1
★	B C	2
	B D	1
	C D	0
★	A B C	2
	A B D	1
	B C D	0
	A C D	0
	A B C D	0

Association rules

An **association rule** is an implication $X \rightarrow Y$, where $X, Y \subset I$, and $X \cap Y = \varnothing$

- example: {milk, bread} → {cereal}
- "A customer who purchased X is also likely to have purchased Y in the same transaction"
- we are interested in rules with a **single item** in Y
- can we represent {milk, bread} → {cereal, cheese}?

Rule $X \rightarrow Y$ holds with **support** *supp* in T if *supp* of transactions contain $X \cup Y$

Rule $X \rightarrow Y$ holds with confidence *conf* in T if *conf* % of transactions that contain X also contain Y

- $conf \approx \Pr(Y \mid X)$
- $conf\ (X \rightarrow Y) = supp\ (X \cup Y)\ /\ supp\ (X)$

Association rules

minSupp = 2 transactions
minConf = 0.75

	supp = 3
$A \rightarrow B$	conf = 3 / 6 = 0.5
$B \rightarrow A$	conf = 3 / 3 = 1.0 ★
	supp = 2
$B \rightarrow C$	conf = 2 / 3 = 0.67
$C \rightarrow B$	conf = 2 / 4 = 0.5
	supp = 4
$A \rightarrow C$	conf = 4 / 6 = 0.67
$C \rightarrow A$	conf = 4 / 4 = 1.0 ★
	supp = 2
$AB \rightarrow C$	conf = 2 / 3 = 0.67
$AC \rightarrow B$	conf = 2 / 4 = 0.5
$BC \rightarrow A$	conf = 2 / 2 = 1.0 ★

	itemset	support
★	A	6
★	B	3
★	C	4
	D	1
★	A B	3
★	A C	4
	A D	1
★	B C	2
	B D	1
	C D	0
★	A B C	2
	A B D	1
	B C D	0
	A C D	0
	A B C D	0

conf **(*X* → *Y*) = *supp* (X U Y) / *supp* (X)**

Association rule mining

- Goal: find all association rules that satisfy the user-specified minimum support and minimum confidence

- Algorithm outline
  - Step 1: find all frequent itemsets
  - Step 2: find association rules

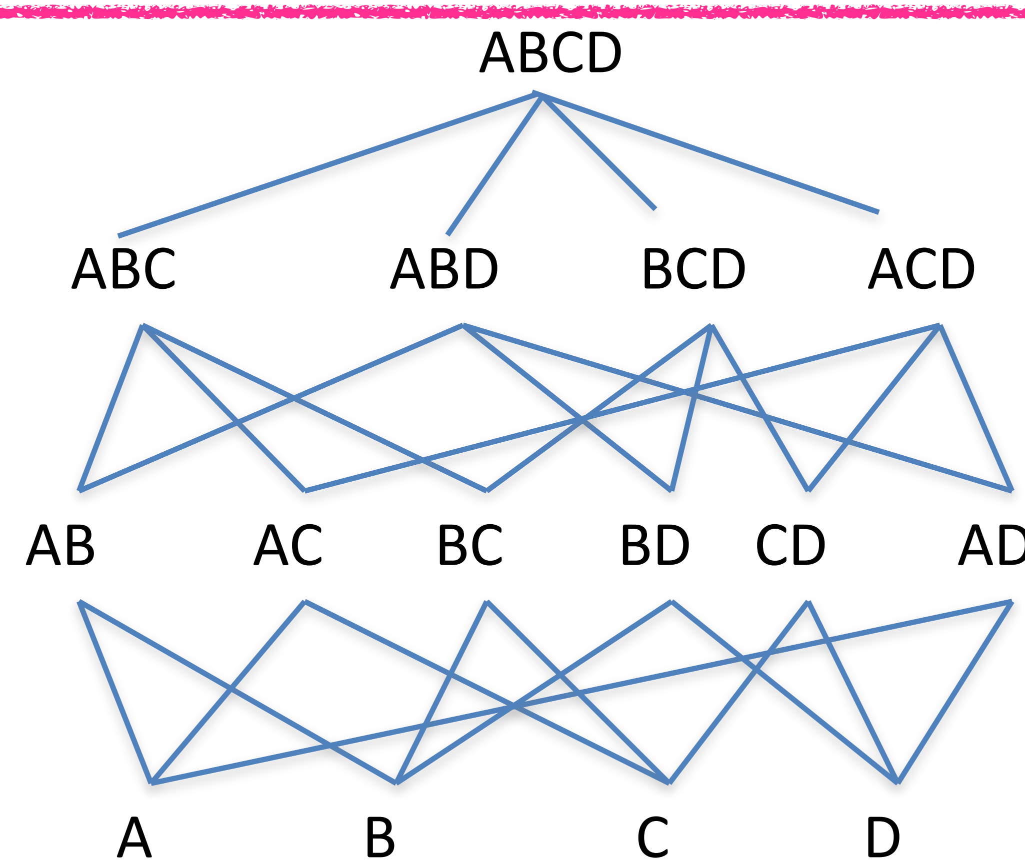
- Take 1: naïve algorithm for frequent itemset mining
  - Enumerate all subsets of ***I***, check their support in ***T***
  - **What is the complexity?**

Key idea: downward closure

	itemset	support
★	A	6
★	B	3
★	C	4
	D	1
★	A B	3
★	A C	4
	A D	1
★	B C	2
	B D	1
	C D	0
★	A B C	2
	A B D	1
	B C D	0
	A C D	0
	A B C D	0

All subsets of a frequent itemset ***X*** are themselves frequent

So, if some subset of X is infrequent, then X cannot be frequent, we know this **apriori**



The converse is not true! If all subsets of ***X*** are frequent, ***X*** is not guaranteed to be frequent

The *Apriori* algorithm

Algorithm Apriori(*T*, *minSupp*)

F_1 = {frequent 1-itemsets};

for ($k = 2$; $F_{k-1} \neq \emptyset$; k++) **do**

$C_k \leftarrow$ **candidate-gen**(F_{k-1});

for each transaction $t \in T$ **do**

for each candidate $c \in C_k$ **do**

if c is contained in t **then**

c.count++;

end

end

$F_k \leftarrow \{c \in C_k \mid c.count \geq minSupp\}$

end

return $F \leftarrow \bigcup_k F_k$;

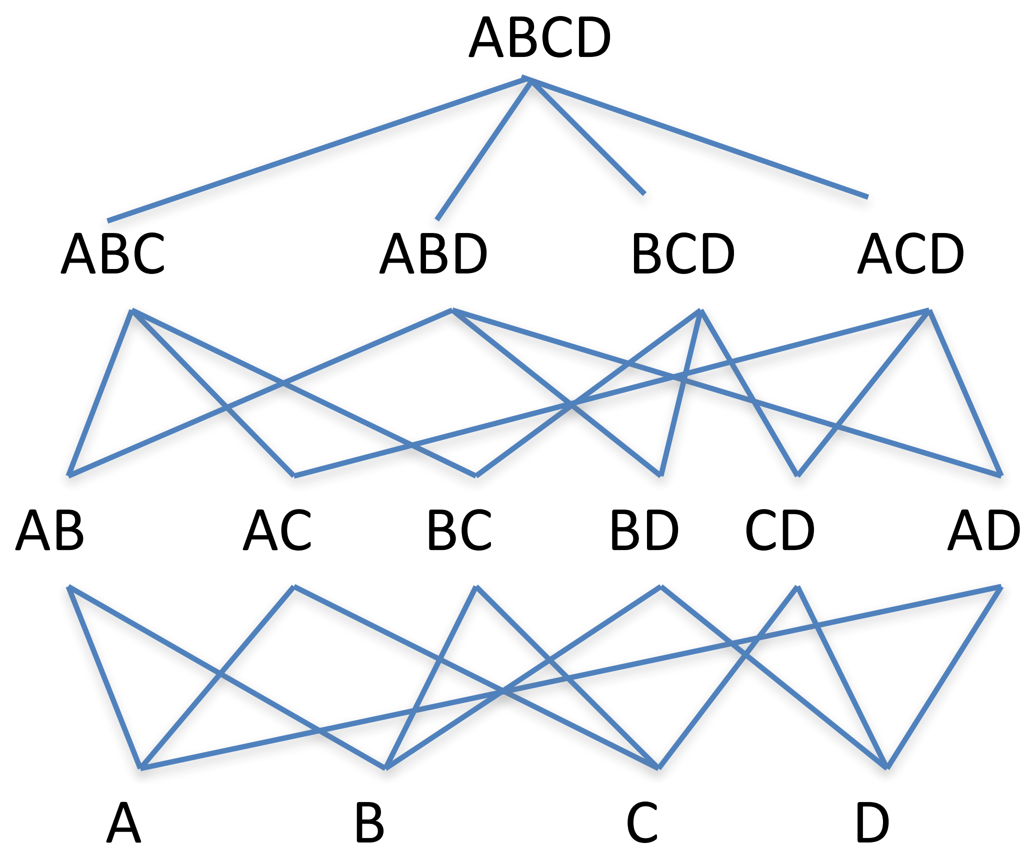
	itemset	support
★	A	6
★	B	3
★	C	4
	D	1
★	A B	3
★	A C	4
	A D	1
★	B C	2
	B D	1
	C D	0
★	A B C	2
	A B D	1
	B C D	0
	A C D	0
	A B C D	0

Candidate generation

The **candidate-gen** function takes F_{k-1} and returns a superset (called the candidates) of the set of all frequent k-itemsets. It has two steps:

Join: generate all possible candidate itemsets C_k of length k

Prune: optionally remove those candidates in C_k that have infrequent subsets



Candidate generation: join

```
Insert into C_k (
 select   p.item_1, p.item_2, …, p.item_{k-1}, q.item_{k-1}
 from     F_{k-1} p, F_{k-1} q
 where    p.item_1 = q.item_1
   and      p.item_2 = q.item_2
  and     …
  and     p.item_{k-1} < q.item_{k-1})
```

	itemset	support
★	A	6
★	B	3
★	C	4
	D	1
★	A B	3
★	A C	4
	A D	1
★	B C	2
	B D	1
	C D	0
★	A B C	2
	A B D	1
	B C D	0
	A C D	0
	A B C D	0

F_1 as p

A
B
C

F_1 as q

A
B
C

C_2

A	**B**
A	**C**
B	**C**

Candidate generation: join

```
Insert into C_k (
 select   p.item_1, p.item_2, …, p.item_{k-1}, q.item_{k-1}
 from     F_{k-1} p, F_{k-1} q
 where    p.item_1 = q.item_1
   and      p.item_2 = q.item_2
  and     …
  and     p.item_{k-1} < q.item_{k-1})
```

itemset	support
★ A	6
★ B	3
★ C	4
D	1
★ A B	3
★ A C	4
A D	1
★ B C	2
B D	1
C D	0
★ A B C	2
A B D	1
B C D	0
A C D	0
A B C D	0

F_2 as p

A	**B**
A	C
B	C

F_2 as q

A	B
A	**C**
B	C

C_3

A	**B**	**C**

Candidate generation

Assume a lexicographic ordering of the items

Join

```
Insert into C_k (
  select  p.item_1, p.item_2, …, p.item_{k-1}, q.item_{k-1}
  from    F_{k-1} p, F_{k-1} q
  where   p.item_1 = q.item_1
  and     p.item_2 = q.item_2
  and     …
  and     p.item_{k-1} < q.item_{k-1} )
```

why not $p.item_{k-1} \neq q.item_{k-1}$?

Prune

for each c in C_k **do**
  for each (k-1) subset s of c **do**
    if (s not in F_{k-1}) **then**
      delete c from C_k

Generating association rules

```
Rules = ∅
for each frequent k-itemset X do
          for each 1-itemset A ⊂ X do
            compute conf (X / A → A) = supp(X) / sup (X / A)
            if conf (X / A → A) ≥ minConf then
              Rules ← "X / A → A"
          end
     end
end
return Rules
```

Performance of *Apriori*

- The possible number of frequent itemsets is exponential, O($\boldsymbol{2^m}$), where $\boldsymbol{m}$ is the number of items
- Apriori exploits sparseness and locality of data
  - Still, it may produce a large number of rules: thousands, tens of thousands, ….
  - So, thresholds should be set carefully. What are some good heuristics?

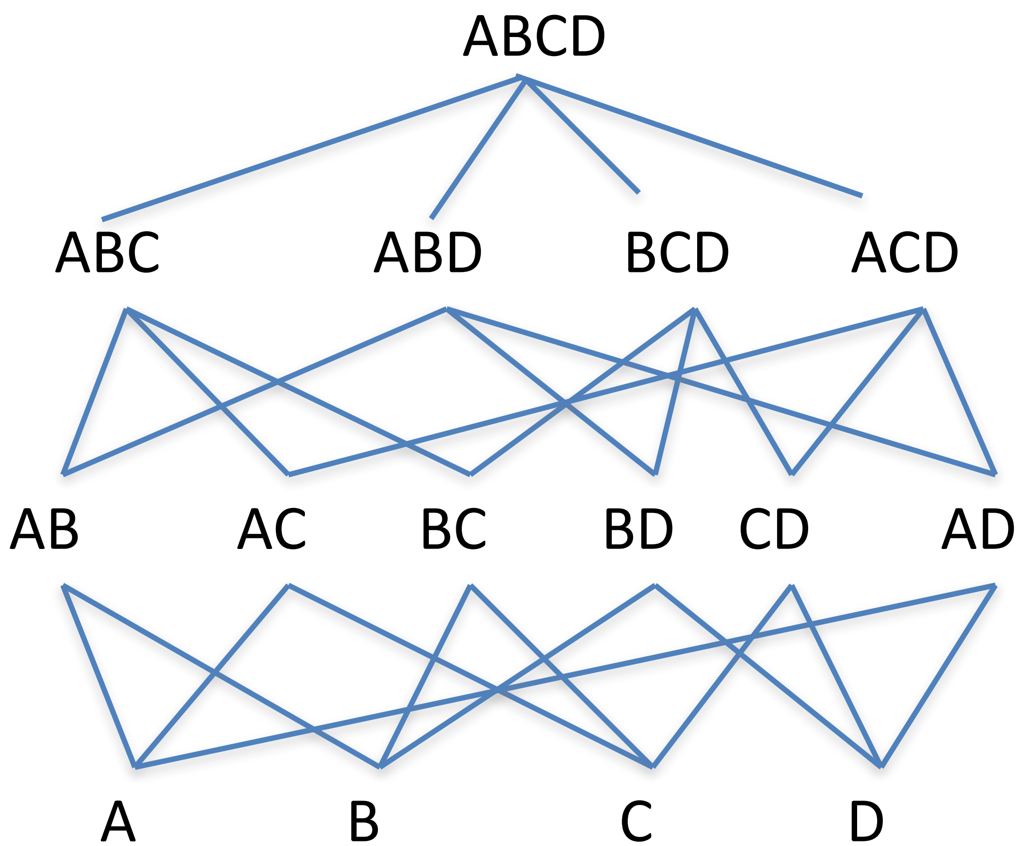
Back to data profiling: Discovering uniques

Given a relation schema ***R** (A, B, C, D)* and a relation instance ***r***, a **unique column combination** (or a **"unique"** for short) is a set of attributes ***X*** whose **projection** contains no duplicates in ***r***

Given a relation schema ***R** (A, B, C, D)* and a relation instance ***r***, a set of attributes ***Y*** is **non-unique** if its projection contains duplicates in ***r***

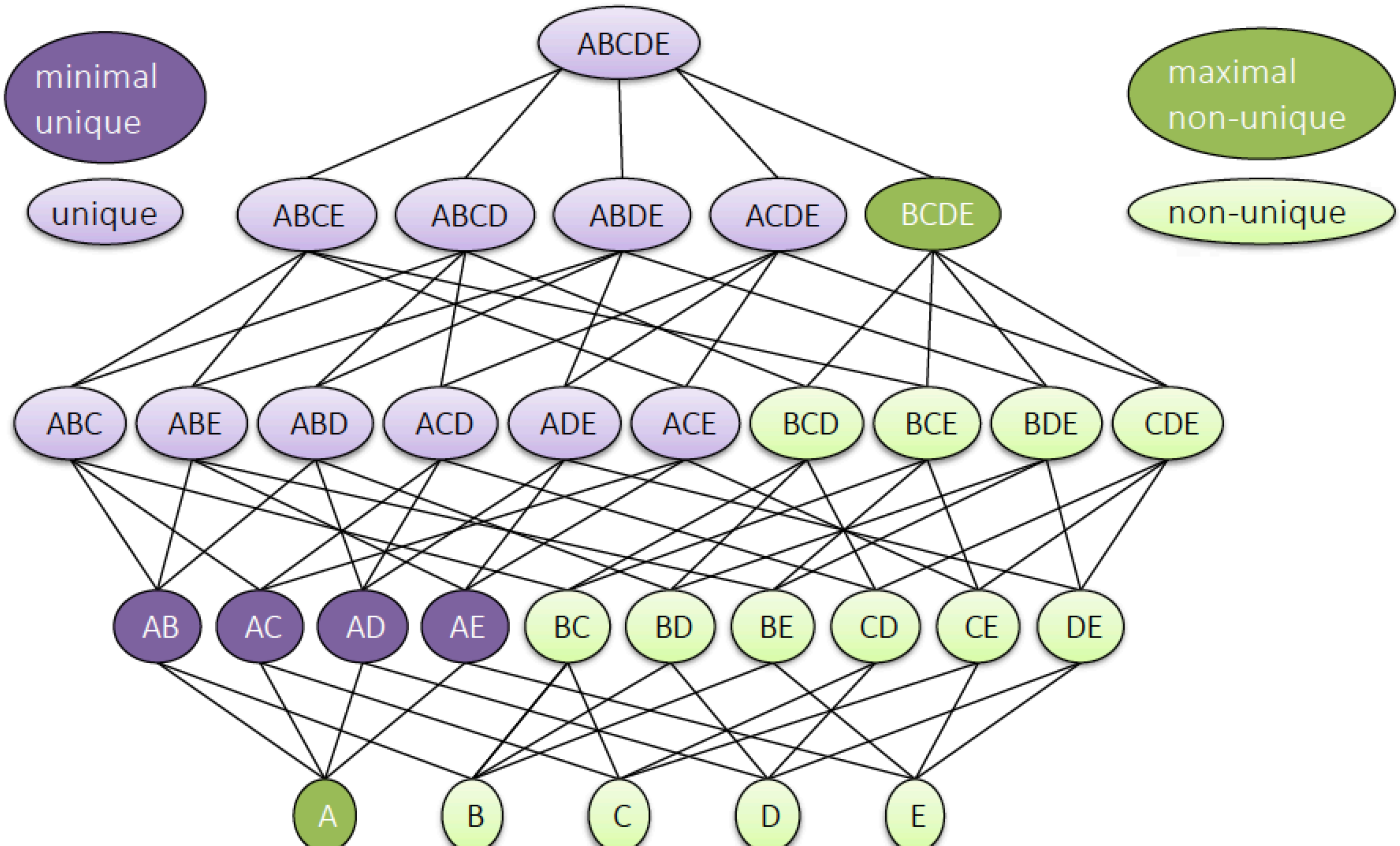
X is **minimal unique** if every subset ***Y*** of ***X*** is non-unique

Y is maximal non-unique if every superset ***X*** of ***Y*** is unique



[Abedjan, Golab, Naumann; *SIGMOD 2017*]

Output



Data Profiling | SIGMOD 2017 | Chicago

From uniques to candidate keys

Given a relation schema ***R*** *(A, B, C, D)* and a relation instance ***r***, a **unique column combination** is a set of attributes ***X*** whose **projection** contains no duplicates in ***r***

$$Episodes(season, num, title, viewers)$$

season	num	title	viewers
1	1	Winter is Coming	2.2 M
1	2	The Kingsroad	2.2 M
2	1	The North Remembers	3.9 M

A set of attributes is a **candidate key** for a relation if:

(1) no two distinct tuples can have the value values for all key attributes (candidate key **uniquely identifies** a tuple), *and*

(2) this is not true for any subset of the key attributes (candidate key **is minimal**)

A minimal unique of a relation instance is a (possible) candidate key of the relation schema. To find such possible candidate keys, find all minimal uniques in a given relation instance.

Apriori-style uniques discovery

[Abedjan, Golab, Naumann; *SIGMOD 2017*]

A minimal unique of a relation instance is a **(possible) candidate key** of the relation schema.

Algorithm Uniques **// sketch, similar to HCA**

U_1 = {1-uniques} N_1 = {1-non-uniques}

for ($k = 2$; $N_{k-1} \neq \emptyset$; k++) **do**

$C_k \leftarrow$ **candidate-gen**(N_{k-1})

$U_k \leftarrow$ **prune-then-check** (C_k)

// prune candidates with unique sub-sets, and with **value distributions that cannot be unique**

// check each candidate in pruned set for uniqueness

$N_k \leftarrow C_k \setminus U_k$

end

return $U \leftarrow \bigcup_k U_k$;

breadth-first bottom-up strategy for attribute lattice traversal