Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Detecting Inclusion Dependencies

15.6.2017

Felix Naumann

Overview

1. Inclusion Dependencies
2. SQL
3. De Marchi et al.
4. MIND
5. SPIDER
6. MANY

Inclusion Dependencies: Definition

- INDs (typically) involve more than one relation.
- Let D be a relational schema and let I be an instance of D.
- $R[A_1, \ldots, A_n]$ denotes projection of I on attributes $A_1, \ldots A_n$ of relation R: $R[A_1, \ldots, A_n] = \pi_{A1, \ldots, An}(R)$

- IND $R[A_1, \ldots, A_n] \subseteq S[B_1, \ldots, B_n]$, where R, S are (possibly identical) relations of D.
  - Projection on R and S must have same number of attributes.
  - An instance I of D satisfies an IND if $I(R)[A_1, \ldots, A_n] \subseteq I(S)[B_1, \ldots, B_n]$
  - IND is maximal if $R[XA] \subseteq S[YB]$ is invalid for any $A \in R$, $B \in S$
  - Values of R: "dependent values"
  - Values of S: "referenced values"

- Task: Find all maximal, non-trivial INDs
  - Typical assumptions: No repeating attributes, disjoint LHS and RHS

Example

- Each Title in Showings should appear as a Title in Movies
  - Showings[Title] ⊆ Movie[Title]

Movies

Title	*Director*	*Actor*
The Birds	Hitchcock	Hedren
The Birds	Hitchcock	Taylor
Bladerunner	Scott	Hannah
Apocalypse Now	Coppola	Brando

Showings

Theater	*Screen*	*Title*	*Snack*
Rex	1	The Birds	coffee
Rex	1	The Birds	popcorn
Rex	2	Bladerunner	coffee
Rex	2	Bladerunner	popcorn
Le Champo	1	The Birds	tea
Le Champo	1	The Birds	popcorn
Cinoche	1	The Birds	Coke
Cinoche	1	The Birds	wine
Cinoche	2	Bladerunner	Coke
Cinoche	2	Bladerunner	wine
Action Christine	1	The Birds	tea
Action Christine	1	The Birds	popcorn

- Aka. “referential integrity”
  - Referenced attributes need not be a key (or unique)
  - Foreign key: helps prune candidates

Inference rules for INDs

- Reflexivity: $R[X] \subseteq R[X]$
- Projection:
  - $R[A_1, \ldots, A_n] \subseteq S[B_1, \ldots, B_n] \Rightarrow R[A_{i1}, \ldots, A_{im}] \subseteq S[B_{i1}, \ldots, B_{im}]$ for each sequence i1, …,im of Integers in {1,…,n}
- Transitivity:
  - $R[X] \subseteq S[Y]$ and $S[Y] \subseteq T[Z] \Rightarrow R[X] \subseteq T[Z]$
  - Example: “transitive foreign keys” for 1:1 relationships

IND types

- Unary INDs
  - INDs on single attributes: $R[A] \subseteq S[B]$
- n-ary INDs
  - INDs on multiple attributes: $R[X] \subseteq S[Y]$
  - $|X| = |Y|$

- Partial INDs
  - IND $R[A] \subseteq S[B]$ is satisfied for *x*% of all tuples in R
  - IND $R[A] \subseteq S[B]$ is satisfied for all but *x* tuples in R

- Approximate INDs
  - IND $R[A] \subseteq S[B]$ is satisfied with probability *p*.
  - Based on sampling or other heuristics

Examples

- Unary: $R[C] \subseteq S[F]$
- N-ary: $R[B,C] \subseteq S[G,F]$
- Partial: $R[A] \subseteq_{75\%} S[F]$
- Approximate: $R[BA] \subseteq S[GH]$

R	A	B	C
	1	x	1
	2	x	1
	3	y	2
	5	z	4

S	F	G	H
	1	x	1
	2	y	3
	3	z	4
	4	z	4

Further IND types

- Prefix/Suffix INDs
  - IND R[A] $\subseteq$ S[B] is satisfied after removing a fixed (or variable) prefix/suffix from each value of A.
  - Twist: A dependent value can now match multiple referenced values

- Example

A
bbc

B
b
bb

$A \subseteq_s B$
(suffix with variable length)

Further IND types

- Conditional INDs
  - Only useful for partial INDs
  - More another time

Catalog

Unit cost	DBName	ProdID
200 USD	ToyDB	17
50 EUR	ToyDB	18
1000 QAR	FashionDB	19

ToyDB

EntityID	further data
17	abcd…
18	efgh…

FashionDB

EntityID	further data
18	abcd…
19	efgh…

Unary IND detection complexity

Name	Type	Equatorial diameter	Mass	Orbital radius	Orbital period	Rotation period	Confirmed moons	Rings	Atmosphere
Mercury	Terrestrial	0.382	0.06	0.47	0.24	58.64	0	no	minimal
Venus	Terrestrial	0.949	0.82	0.72	0.62	−243.02	0	no	CO_2, N_2
Earth	Terrestrial	1.000	1.00	1.00	1.00	1.00	1	no	N_2, O_2, Ar
Mars	Terrestrial	0.532	0.11	1.52	1.88	1.03	2	no	CO_2, N_2, Ar
Jupiter	Giant	11.209	317.8	5.20	11.86	0.41	67	yes	H_2, He
Saturn	Giant	9.449	95.2	9.54	29.46	0.43	62	yes	H_2, He
Uranus	Giant	4.007	14.6	19.22	84.01	−0.72	27	yes	H_2, He
Neptune	Giant	3.883	17.2	30.06	164.8	0.67	14	yes	H_2, He

Complexity: $O(n^2-n)$
for n attributes

Example:
10 attr ~ 90 checks
1,000 attr ~ 999,000 checks

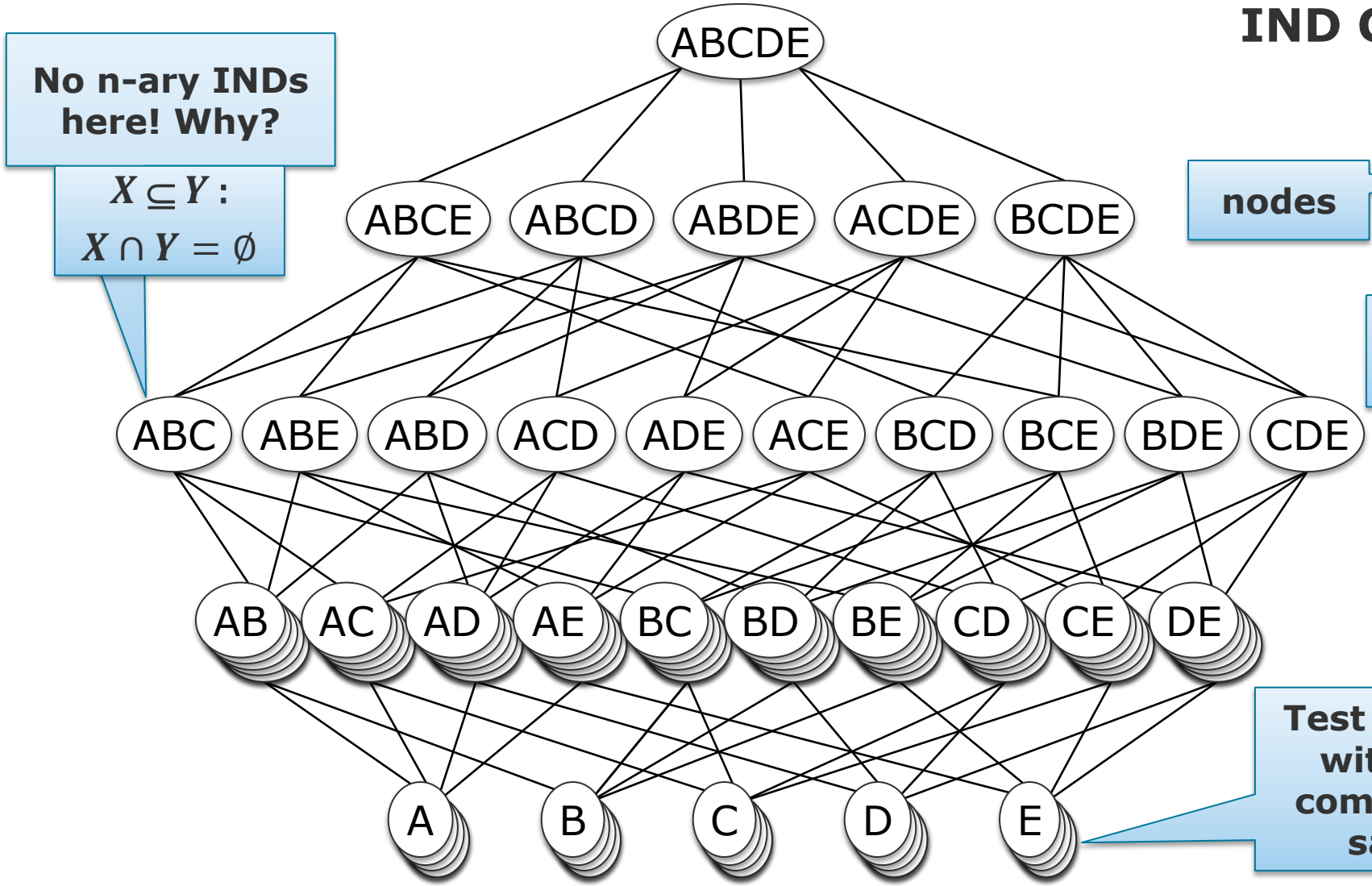
$n_{IND} >> n_{UCC}$

- Name ⊆ Type ?
- Name ⊆ Equatorial_diameter ?
- Name ⊆ Mass ?
- Name ⊆ Orbital_radius ?
- Name ⊆ Orbital_period ?
- Name ⊆ Rotation_period ?
- Name ⊆ Confirmed_moons ?
- Name ⊆ Rings ?
- Name ⊆ Atmosphere ?

- Type ⊆ Name ?
- Type ⊆ Equatorial_diameter ?
- Type ⊆ Mass ?
- Type ⊆ Orbital_radius ?
- Type ⊆ Orbital_period ?
- Type ⊆ Rotation_period ?
- Type ⊆ Confirmed_moons ?
- Type ⊆ Rings ?
- Type ⊆ Atmosphere ?

- Mass ⊆ Name ?
- Mass ⊆ Type ?
- Mass ⊆ Equatorial_diameter ?
- …

N-ary IND detection complexity



No n-ary INDs here! Why?

$X \subseteq Y$:
$X \cap Y = \emptyset$

IND Candidates in level k:

$$\underbrace{\binom{n}{k}}_{X} * \underbrace{\binom{n-k}{k}}_{Y} * k!$$

nodes

all permu-tations

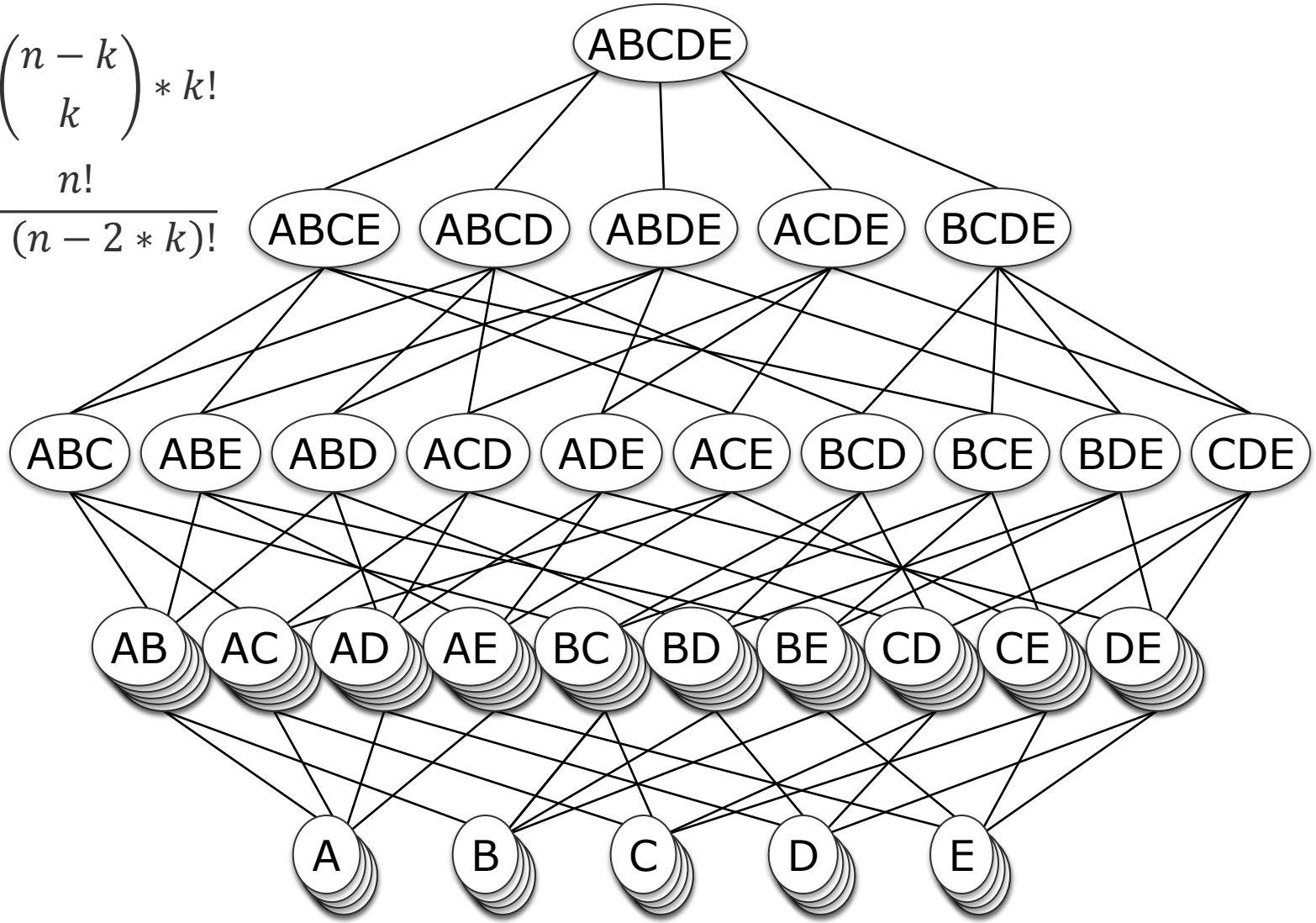
other, non-overlapping nodes

Test combination with all other combinations of same size!

N-ary IND detection complexity

$$\binom{n}{k} * \binom{n-k}{k} * k!$$

$$= \frac{n!}{k! * (n - 2 * k)!}$$



$$\binom{5}{5} * \binom{5-5}{5} * 5! \sim 0$$

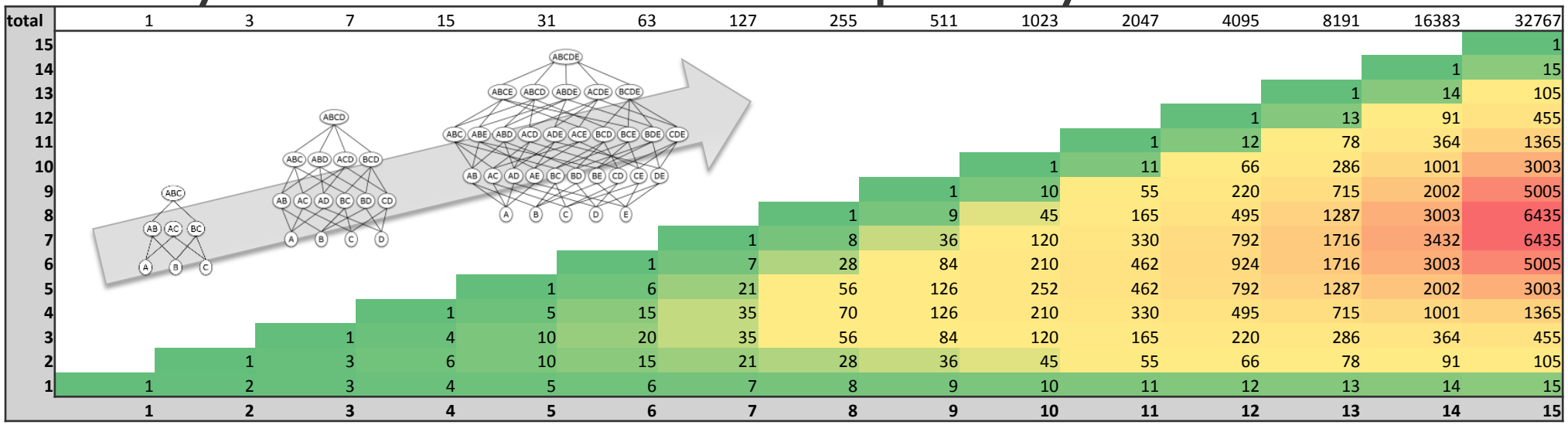
$$\binom{5}{4} * \binom{5-4}{4} * 4! \sim 0$$

$$\binom{5}{3} * \binom{5-3}{3} * 3! \sim 0$$

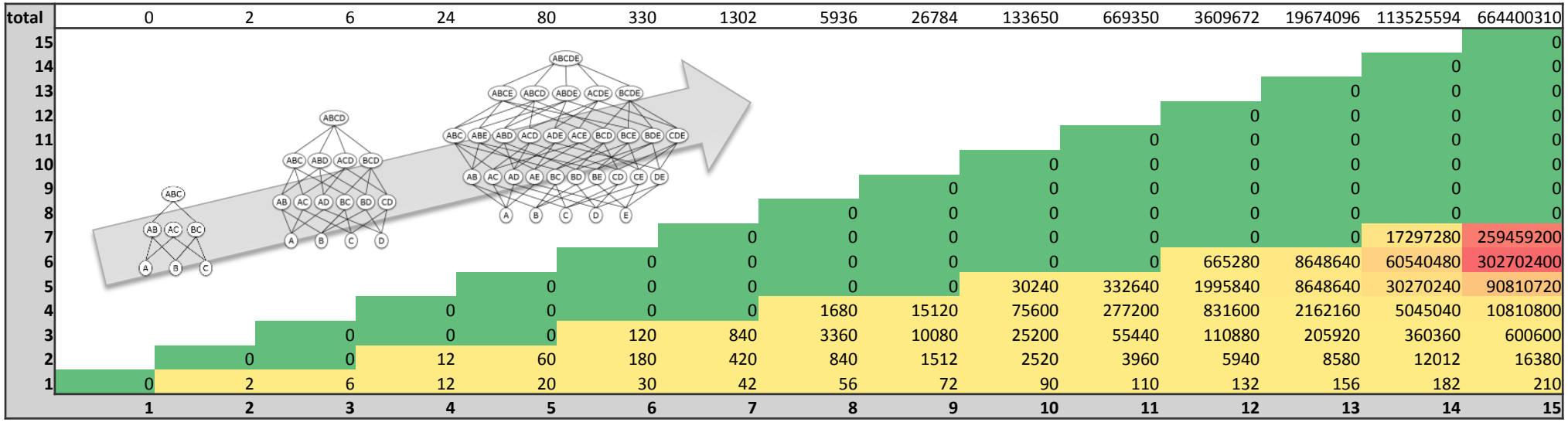
$$\binom{5}{2} * \binom{5-2}{2} * 2! = 60$$

$$\binom{5}{1} * \binom{5-1}{1} * 1! = 20 = n^2 - n$$

N-ary IND detection complexity



Unique Column Combinations



Inclusion Dependencies

Felix Naumann
Data Profiling
Summer 2017

Motivation for IND discovery

- General insight into data

- Detect unknown foreign keys
- Example
  - PDB: Protein Data Bank
  - OpenMMS provides relational schema
    - Parses protein and nucleic acid macromolecular structure data from the standard mmCIF format.
  - 175 tables with primary key constraints
  - 2705 attributes
  - But: Not a single foreign key constraint!

```
_pdbx_poly_seq_scheme.pdb_strand_id
_pdbx_poly_seq_scheme.pdb_ins_code
_pdbx_poly_seq_scheme.hetero
A 1 1  DC 1  1  1  DC C A . n
A 1 2  DC 2  2  2  DC C A . n
A 1 3  DG 3  3  3  DG G A . n
A 1 4  DT 4  4  4  DT T A . n
A 1 5  DA 5  5  5  DA A A . n
A 1 6  DC 6  6  6  DC C A . n
A 1 7  DG 7  7  7  DG G A . n
A 1 8  DT 8  8  8  DT T A . n
A 1 9  DA 9  9  9  DA A A . n
A 1 10 DC 10 10 10 DC C A . n
A 1 11 DG 11 11 11 DG G A . n
A 1 12 DG 12 12 12 DG G A . n
#
loop_
_refine_B_iso.class
_refine_B_iso.details
_refine_B_iso.treatment
_refine_B_iso.pdbx_refine_id
'ALL ATOMS'  TR isotropic 'X-RAY DIFFRACTION'
'ALL WATERS' TR isotropic 'X-RAY DIFFRACTION'
#
loop_
_refine_occupancy.class
_refine_occupancy.treatment
_refine_occupancy.pdbx_refine_id
'ALL ATOMS'  fix 'X-RAY DIFFRACTION'
'ALL WATERS' fix 'X-RAY DIFFRACTION'
#
loop_
_pdbx_version.entry_id
_pdbx_version.revision_date
_pdbx_version.major_version
_pdbx_version.minor_version
_pdbx_version.revision_type
_pdbx_version.details
116D 2008-05-22 3 2    'Version format compliand
116D 2011-07-13 4 0000 'Version format compliand
#
_software.name             NUCLSQ
```

Motivation for IND discovery

- Ensembl – genome database
  - Shipped as MySQL dump files
  - More than 200 tables
  - Not a single foreign key constraint!

- Web tables: No schema, no constraints, but many connections

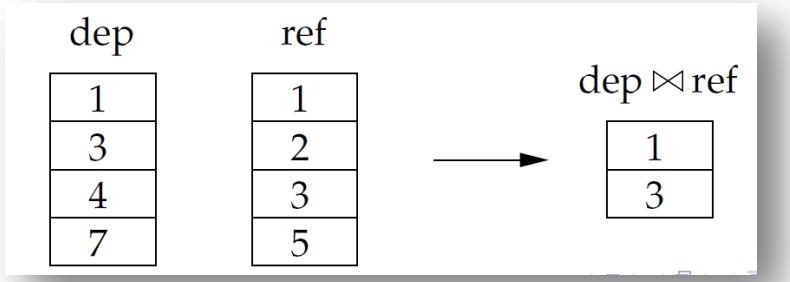
- Why are FKs missing?
  - Lack of support for checking foreign key constraints in the host system
    - Example: Oracle did not support FKs up to v6
  - Fear that checking such constraints would impede database performance
  - Lack of database knowledge within the development team
  - Dirty data prevents setting the constraint

Overview

1. Inclusion Dependencies
2. **SQL**
3. De Marchi et al.
4. MIND
5. SPIDER
6. MANY

The JOIN

- `SELECT COUNT(depColumn) AS numDeps FROM depTable`
- `SELECT COUNT(*) AS matchedDeps FROM depTable JOIN refTable ON depTable.depColumn = refTable.refColumn`
  - Assumption: ref is a unique column



- (numDeps = matchedDeps) ⇔ depColumn ⊆ refColumn

- Missed opportunity
  - DBMS could stop early: As soon as we observe a dependent value without a join partner

The EXCEPT

```
SELECT count(*) AS unmatchedDeps FROM
  ((SELECT to char (depColumn)
    FROM   depTable
    WHERE  depColumn IS NOT NULL
   EXCEPT
    SELECT to char (refColumn)
    FROM   refTable
   )
   FETCH FIRST 1 ROWS ONLY
  )
```

- unmatchedDeps = 0 ⇔ depColumn ⊆ refColumn

The “Antijoin”

- ```
  SELECT COUNT(*) AS unmatched
  FROM R
  WHERE A IS NOT NULL
  AND A NOT IN
     (SELECT B FROM S)
  FETCH FIRST 1 ROWS ONLY
  ```

- depColumn $\subseteq$ refColumn
  $\Leftrightarrow$ unmatched = 0

- ```
  SELECT COUNT(*) AS unmatched
  FROM R
  WHERE A IS NOT NULL
  AND NOT EXISTS
    (SELECT * FROM S WHERE
  R.A=S.B)
  FETCH FIRST 1 ROWS ONLY
  ```

- depColumn $\subseteq$ refColumn
  $\Leftrightarrow$ unmatched = 0

Measurements (2006)

- High efficiency of joins in DBMS
- Inability of DBMS optimizer to move STOP operator into inner queries

- Overall problems
  - Still too slow
  - One SQL statement per attribute pair
  - Each attribute joined *n* times (many sorts/hashes)

	CATH	SCOP	UniProt	TPC-H	PDB
DB size	20 MB	17,5 MB	900 MB	1.3 GB	2.8 GB
# attributes	25	22	68	61	1,215
# IND candidates	68	43	910	477	139,807
# INDs	0	11	36	33	4,972
join	6 s	7 s	9 m 04 s	25 m 02 s	16 h 14 m
except	15 m 27 s	16 m 05 s	27 m 35 s	1 h 09 m	–
not in	5 s	52 m 11 s	6 h 33 m	7 m 45 s	–
not exists	5 s	6 s	3 m 57 s	7 m 51 s	10 h 20 m

Table 4.1: Runtime performance of the SQL approaches. IND candidates are restricted to cover unique referenced attributes. We used only a fraction of PDB.

Discussion on data profiling experiments

- What can we assume? What is the scenario?
  - Index every column
  - Statistics for each table and column
    - Example: min and max values for pruning
  - Where is the data originally
    - In a database
    - In files
  - Do I count importing the data?
    - Could then do statistics on the fly

Overview

1. Inclusion Dependencies
2. SQL
3. **De Marchi et al.**
4. MIND
5. SPIDER
6. MANY

Pruning for Unary INDs

- Datatype
  - Incompatible datatypes cannot be included in one-another
- Statistics
  - card(R[A]) > card(S[B]) ⇒ R.A ⊈ S.B
    - card(R[X]) > card(S[Y]) ⇒ R.X ⊈ S.Y
  - max(R[A]) > max(S[B]) ⇒ R.A ⊈ S.B
  - min(R[A]) < min(S[B]) ⇒ R.A ⊈ S.B
- Bloom filters

	UniProt	TPC-H	PDB	
	900 MB	1.3 GB	2.8 GB	32 GB
# attributes	68	61	1,215	1,297
# IND candidates	1,393	877	219,106	245,562
# satisfied INDs	36	33	4,972	5,431
# attributes in INDs	31	20	448	478
distinct				
# IND candidates	910	477	139,807	157,818
# attributes in IND candidates	68	58	1,208	1,297
distinct & max				
# IND candidates	541	295	72,016	83,321
# attributes in IND candidates	59	54	997	1,080
distinct & min				
# IND candidates	345	275	61,920	68,664
# attributes in IND candidates	54	57	990	1,042
distinct & max & min				
# IND candidates	174	137	22,655	25,821
# attributes in IND candidates	49	52	670	709

Data preprocessing

- Key idea: For a given domain, associate each value with every attribute having this value.
  - Create binary relation $B \subseteq$ Values x Attributes with $(v,A) \in B$ iff $v \in \pi_A(R)$
  - Analogy: Inverted index

Efficient Algorithms for Mining Inclusion Dependencies, Fabien De Marchi, Stéphane Lopes, and Jean-Marc Petit, In: EDBT 2002

Example

- Three domains: int, real, and string
- Example for domain “int”
  - Values = {1,2,3,4,6,7,9}
  - Attributes = {A,C,E,G,K}
  - Examples for relation B: (1,A), (1,E), (1,K)
- Can build relation with single full scan of each base relation

r

A	B	C	D
1	X	3	11.0
1	X	3	12.0
2	Y	4	11.0
1	X	3	13.0

s

E	F	G	H
1	X	3	11.0
2	Y	4	12.0
4	Z	6	14.0
7	W	9	14.0

t

I	J	K	L
11.0	11.0	1	X
12.0	12.0	2	Y
11.0	14.0	4	Z
11.0	9.0	7	W
13.0	13.0	9	R

Example „Extraction contexts“

r

A	B	C	D
1	X	3	11.0
1	X	3	12.0
2	Y	4	11.0
1	X	3	13.0

s

E	F	G	H
1	X	3	11.0
2	Y	4	12.0
4	Z	6	14.0
7	W	9	14.0

t

I	J	K	L
11.0	11.0	1	X
12.0	12.0	2	Y
11.0	14.0	4	Z
11.0	9.0	7	W
13.0	13.0	9	R

int

$\mathbb{V}$	$\mathbb{U}$
1	A E K
2	A E K
3	C G
4	C E G K
6	G
7	E K
9	G K

real

$\mathbb{V}$	$\mathbb{U}$
9.0	J
11.0	D H I J
12.0	D H I J
13.0	D I J
14.0	H J

string

$\mathbb{V}$	$\mathbb{U}$
R	L
X	B F L
Y	B F L
Z	F L
W	F L

IND discovery

- Insight: If all values of attribute *A* can be found in values of *B (i.e., $A \subseteq B$)*, then by construction *B* will be present in all lines of the binary relation containing *A*.

$$A \subseteq B \iff B \in \bigcap_{v \in \mathbb{V} \mid (v,A) \in \mathbb{B}} \{C \in \mathbb{U} \mid (v, C) \in \mathbb{B}\}$$

int

$\mathbb{V}$	$\mathbb{U}$
1	A E K
2	A E K
3	C G
4	C E G K
6	G
7	E K
9	G K

real

$\mathbb{V}$	$\mathbb{U}$
9.0	J
11.0	D H I J
12.0	D H I J
13.0	D I J
14.0	H J

string

$\mathbb{V}$	$\mathbb{U}$
R	L
X	B F L
Y	B F L
Z	F L
W	F L

IND discovery algorithm

Input: the triplet $\mathbb{V}, \mathbb{U}, \mathbb{B}$, associated with **d** and t.
Output: $\mathcal{I}_1$ the set of unary INDs verified by **d** between attributes of type t.

1: **for all** $A \in \mathbb{U}$ **do** $rhs(A) = \mathbb{U}$;
2: **for all** $v \in \mathbb{V}$ **do**
3: **for all** A s.t. $(v, A) \in \mathbb{B}$ **do**
4: $rhs(A) = rhs(A) \cap \{B \mid (v, B) \in \mathbb{B}\}$;
5: **for all** $A \in \mathbb{U}$ **do**
6: **for all** $B \in rhs(A)$ **do**
7: $\mathcal{I}_1 = \mathcal{I}_1 \cup \{A \subseteq B\}$;
8: return $\mathcal{I}_1$.

All attributes are ref candidates

Remove candidates

Generate output

int

$\mathbb{V}$	$\mathbb{U}$
1	A E K
2	A E K
3	C G
4	C E G K
6	G
7	E K
9	G K

real

$\mathbb{V}$	$\mathbb{U}$
9.0	J
11.0	D H I J
12.0	D H I J
13.0	D I J
14.0	H J

string

$\mathbb{V}$	$\mathbb{U}$
R	L
X	B F L
Y	B F L
Z	F L
W	F L

IND discovery algorithm: Example

- Step 0: rhs(A) = … = rhs(K) = {A,C,E,G,K}
- Step 1 (v=1): rhs(A)={A,E,K}, rhs(E)={A,E,K}, rhs(K)={A,E,K}, rhs(C) = rhs(G) = {A,C,E,G,K}
- Step 2 (v=2): unchanged
- Step 3 (v=3): rhs(C)={C,G}, rhs(G)={C,G}
- …
- Step 9: rhs(A)={A,E,K}, rhs(C)={C,G}, rhs(E)={E,K}, rhs(G)={G}, rhs(K)={K}
- A ⊆ E, A ⊆ K, C ⊆ G, and E ⊆ K

int

𝕍	𝕌
1	A E K
2	A E K
3	C G
4	C E G K
6	G
7	E K
9	G K

real

𝕍	𝕌
9.0	J
11.0	D H I J
12.0	D H I J
13.0	D I J
14.0	H J

string

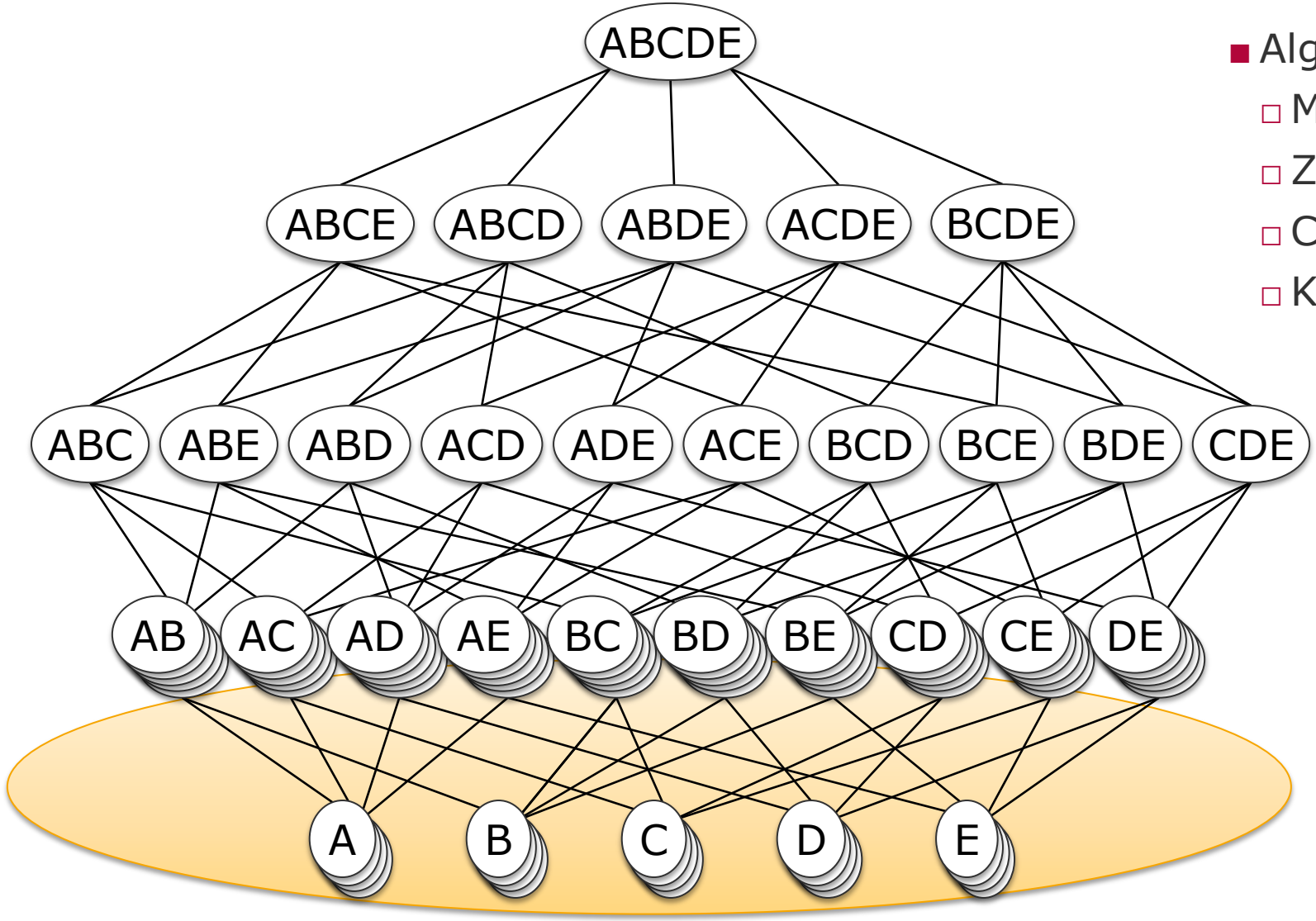
𝕍	𝕌
R	L
X	B F L
Y	B F L
Z	F L
W	F L

Question: Why distinguish domains?

Overview

1. Inclusion Dependencies
2. SQL
3. De Marchi et al.
4. **MIND**
5. SPIDER
6. MANY

N-ary IND detection

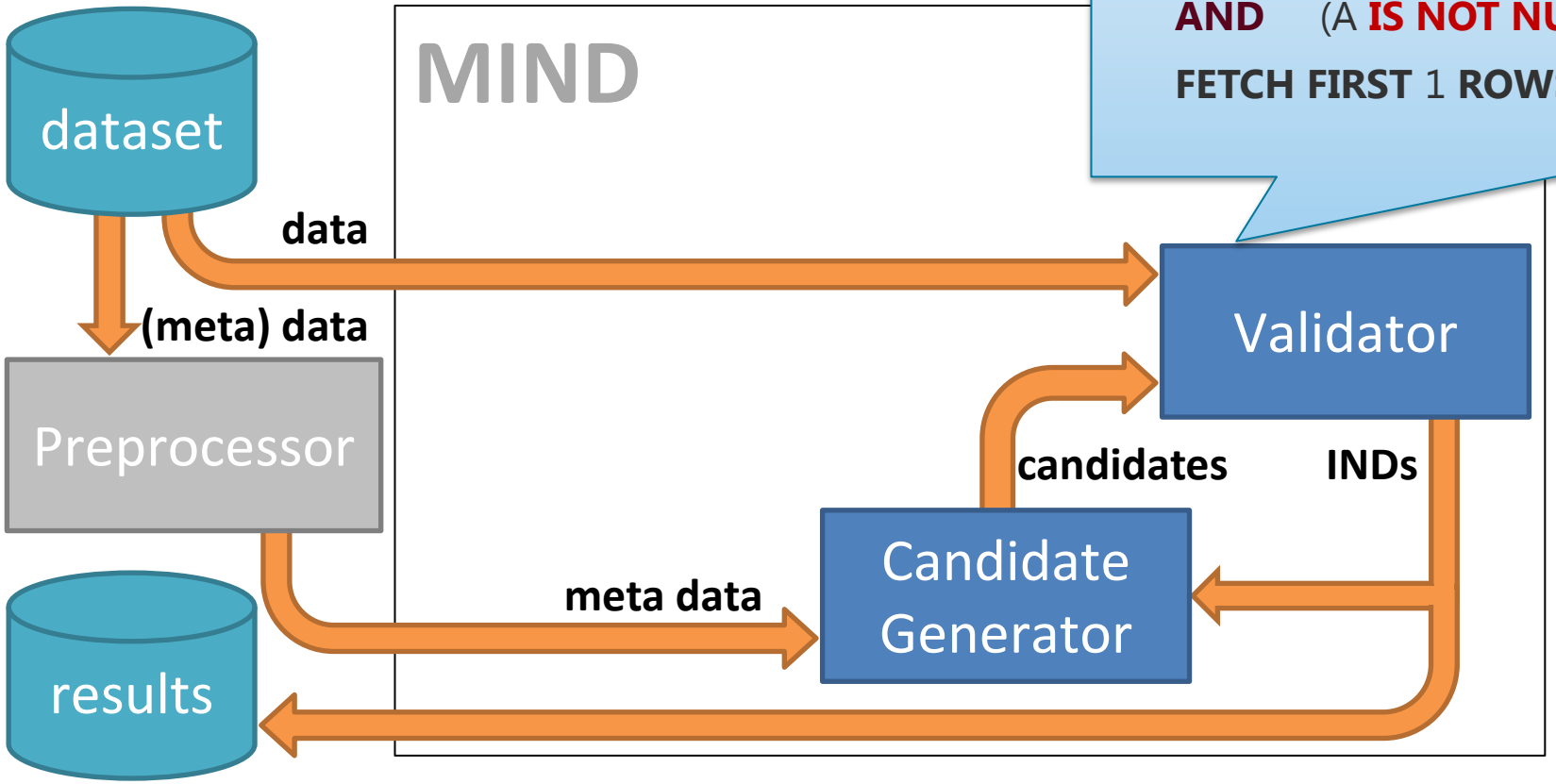


- Algorithms for n-ary detection:
  - MIND (apriori bottom-up)
  - ZigZag (bottom-up and top-down)
  - Clim (closed item set mining)
  - Koeller (aggressive bottom up)

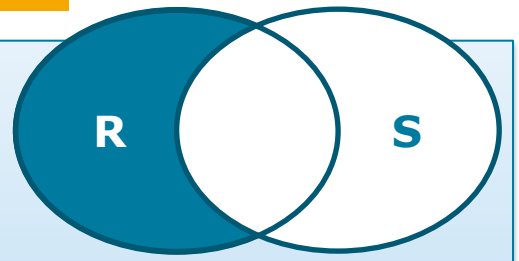
Pruning for INDs

- $R[A_1, \ldots, A_n] \subseteq S[B_1, \ldots, B_n] \Rightarrow R[A_{i1}, \ldots, A_{im}] \subseteq S[B_{i1}, \ldots, B_{im}]$
- Pruning down
  - Example: $R[AB] \subseteq S[DE] \Rightarrow R[A] \subseteq S[D]$ and $R[B] \subseteq S[E]$
- Pruning up
  - Example: $R[AB] \nsubseteq S[DE] \Rightarrow R[ABC] \nsubseteq S[DEF]$
  - Apriori: Use only satisfied INDs to generate next level candidates
- Pruning laterally
  - $R[AB] \subseteq S[DE] \Rightarrow R[BA] \subseteq S[ED]$
  - Define permutation strategy
    - Lexicographic ordering of attribute labels for LHS

MIND – workflow



Validate A,B ⊆ C,D:




```
SELECT R.A, R.B, S.C, S.D
FROM   R LEFT OUTER JOIN S ON A = C AND B = D
WHERE (C IS NULL AND D IS NULL)
AND    (A IS NOT NULL OR B IS NOT NULL)
FETCH FIRST 1 ROWS ONLY;
```

```
SELECT depTable.AAGE, depTable.ACLSWKR,
       depTable.ADTIND, depTable.ADTOCC,
       depTable.AGI, depTable.AHGA
FROM CENSUS6 depTable LEFT OUTER JOIN
     CENSUS refTable ON
     depTable.AAGE = refTable.AAGE AND
     depTable.ACLSWKR = refTable.ACLSWKR  AND
     depTable.ADTIND = refTable.ADTIND AND
     depTable.ADTOCC = refTable.ADTOCC AND
     depTable.AGI = refTable.AGI AND
     depTable.AHGA = refTable.AHGA
WHERE refTable.AAGE IS NULL AND
      refTable.ACLSWKR IS NULL AND
      refTable.ADTIND IS NULL AND
      refTable.ADTOCC IS NULL AND
      refTable.AGI IS NULL AND
      refTable.AHGA IS NULL
AND (depTable.AAGE IS NOT NULL OR
     depTable.ACLSWKR IS NOT NULL OR
     depTable.ADTIND IS NOT NULL OR
     depTable.ADTOCC IS NOT NULL OR
     depTable.AGI IS NOT NULL OR
     depTable.AHGA IS NOT NULL)
FETCH FIRST 1 ROWS ONLY;
```

Algorithm 2 MIND

Input: **d** a database, and $\mathcal{I}_1$ the set of unary INDs satisfied by **d**.
Output: Inclusion dependencies satisfied by **d**

1: $\mathcal{C}_2 := GenNext(\mathcal{I}_1)$;
2: $i := 2$;
3: **while** $\mathcal{C}_i \neq \emptyset$ **do**
4: **forall** $I \in \mathcal{C}_i$ **do**
5: **if d** $\models I$ **then**
6: $\mathcal{I}_i := \mathcal{I}_i \cup \{I\}$;
7: $\mathcal{C}_{i+1} := GenNext(\mathcal{I}_i)$;
8: $i := i + 1$;
9: **end while**
10: return $\cup_{j<i} \mathcal{I}_j$

Start with unary INDs from previous algorithm

Use SQL queries

Fabien De Marchi, Stéphane Lopes, Jean-Marc Petit:
Unary and n-ary inclusion dependency discovery in relational databases. J. Intell. Inf. Syst. 32(1): 53-73 (2009)

The MIND Algorithm – Apriori-based generation

Algorithm 3 *GenNext* : Generation of candidate INDs of size $i+1$

Input: $\mathcal{I}_i$, inclusion dependencies of size i.
Output: $\mathcal{C}_{i+1}$, sequence of candidate inclusion dependencies of size i+1

```
1:  insert into C_{i+1}
2:  select p.lhs.rel"["p.lhs[1], p.lhs[2], ..., p.lhs[i], q.lhs[i]"]"     " ⊆ "
    p.rhs.rel"["p.rhs[1], p.rhs[2], ..., p.rhs[i], q.rhs[i]"]"
3:  from I_i p, I_i q
4:  where p.lhs.rel = q.lhs.rel and p.rhs.rel = q.rhs.rel
5:        and p.lhs[1] = q.lhs[1] and p.rhs[1] = q.rhs[1]
6:        and ...
7:        and p.lhs[i − 1] = q.lhs[i − 1] and p.rhs[i − 1] = q.rhs[i − 1]
8:        and p.lhs[i] < q.lhs[i]
9:        and p.rhs[i] <> q.rhs[i]
10: for all I ∈ C_{i+1} do
11:    for all J ≺ I and J of size i do
12:       if J ∉ I_i then
13:          C_{i+1} = C_{i+1} \ {I}
14:       end if
15:    end for
16: end for
```

All LHS (and RHS) attributes from same relation

Same prefix of size $i-1$

Avoid superfluous permutations

Disallow repeated attributes

Generation

Pruning

Remove candidate if not all (ordered) subsets are in lower level

Overview

1. Inclusion Dependencies
2. SQL
3. De Marchi et al.
4. MIND
5. **SPIDER**
6. MANY

Making use of order

- Idea: Order each column only once
  - As index in DBMS
  - Or as sorted columns in individual files
    - `SELECT DISTINCT A FROM R ORDER BY A`
    - Store result in file

A
1
3
4
7

B
1
2
3
5

- Simulate merging procedure (merge join, merge sort)
  - Move cursor along both columns
  - Stop after first dependent value that is not in referenced attribute

Testing a single IND candidate

- 2 ordered lists of distinct values: depValues and refValues
- **while** (depValues has next)
  - currentDep = depValues.next();
  - **while** (true)
    - currentRef = refValues.next();
    - **if** (currentDep = currentRef) **then** break;
    - **else if** (currentDep < currentRef) **then return** false;
    - **else if** not(refValues has next) **then return** false;
- **return** true;

A	B	C
1	1	1
3	2	2
4	3	3
	5	4

Brute force approach

- Sequentially check each column pair
- At least: Re-use order for each attribute

- Problem: Run through data multiple times
  - A ⊆ C
  - A ⊆ D
  - B ⊆ C
  - B ⊆ D

A
1
3
4

B
2
3
5

C
1
3
7
8

D
1
2
3
5

SPIDER
Single Pass Inclusion DEpendency Recognition

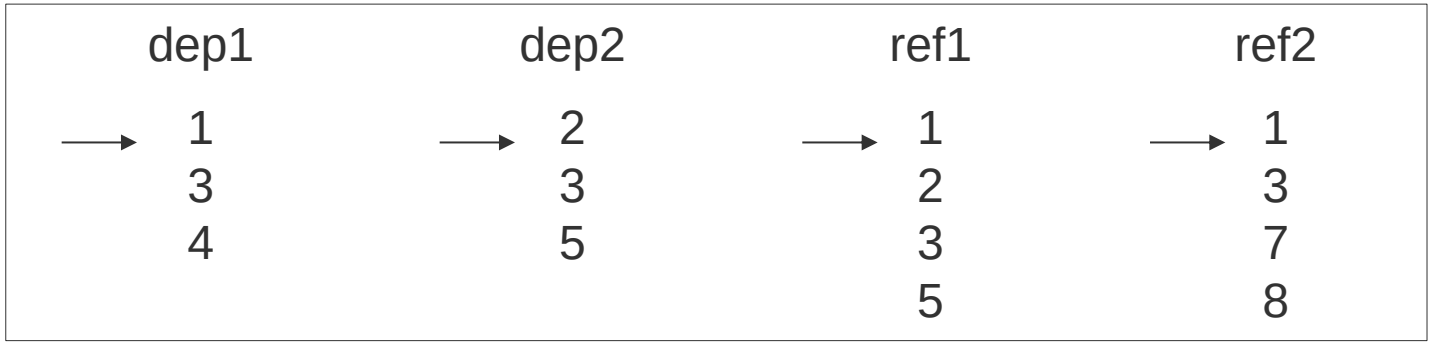
- Main ideas
  - Test all IND-candidate pairs simultaneously.
    - In both directions
  - Read attribute values only once.
  - Stop test of an IND-candidate after first counter-example.
  - Reduce number of value comparisons by specialized data structure.
  - No need to build inverted index.

- Two steps:
  - Sort and distinct all attribute‘s values and write them to disk
    - For each attribute: `SELECT DISTINCT A FROM R ORDER BY A`
  - Test all IND candidate pairs simultaneously

- Jana Bauckmann and Ulf Leser and Felix Naumann. *Efficiently Computing Inclusion Dependencies for Schema Discovery*. In Proceedings of the International Conference on Data Engineering Workshops (ICDE workshops), 2006.
- Jana Bauckmann, Ulf Leser, Felix Naumann, Véronique Tietz: Efficiently Detecting Inclusion Dependencies. In: ICDE , 2007.

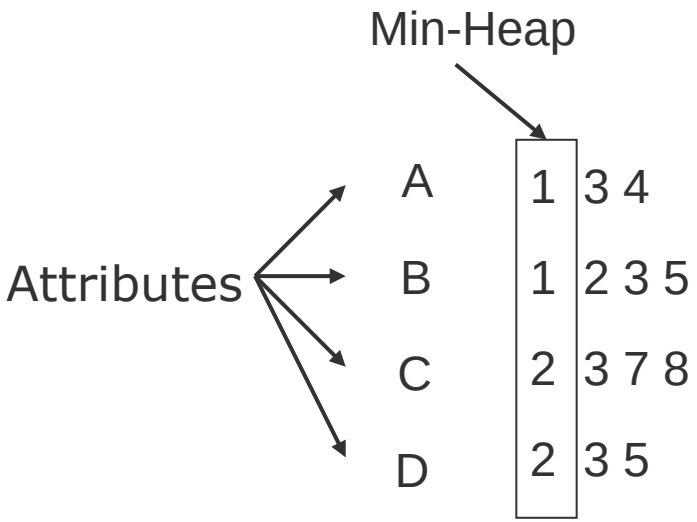
SPIDER

- Parallel generation and test of all IND candidates
  - Reads each value at most once
- Challenge: Synchronize reading of values of all attributes
  - Each dependent attribute value influences when a referenced attribute value can be read.
  - Each referenced attribute value influences when a dependent attribute value can be read.
- Move cursor *r* on a referenced file R when all cursors to dependent files point to values that are greater than the current value pointed to by *r*.
- Move a cursor *d* on a dependent file D one step further, when *d*'s value is smaller than all values currently pointed to in referenced files.



SPIDER: Idea

- All values within each attribute are sorted.
- Attributes themselves are sorted by current minimum value (in a min-heap).
  - Order changes all the time
- IND candidates represented as a list for each dependent attribute, containing all referenced attributes.



SPIDER by example

- In each step: Intersect „attributes to process“ with each refs list of previous step
  - Intersection only with refs-list of “attributes to process”
- Attributes with empty refs-list can be removed from consideration

attributes A, B, C

A	B	C
s		s
t	t	t
x		
y	y	y
		z

	attributes to process	dep A refs	dep B refs	dep C refs
Init		B,C	A,C	A,B
Step 1	A,C	C	A,C	A
Step 2	A,B,C	C	A,C	A
Step 3	A	∅	A,C	A
Step 4	A,B,C	∅	A,C	A
Step 5	C	∅	A,C	∅

SPIDER results (data in DBMS)

	UniProt	TPC-H	PDB	
DB size	900 MB	1.3 GB	2.8 GB	32GB
# attributes	68	61	1215	1297
# IND cand.	910	477	139,807	157,818
# INDs	36	33	4,972	5,431
join	9m04s	25m02s	16h14m	> 7 days
Marchi et al.	9h 58m	-	-	-
Brute force	2m11s	6m30s	3h29m	19h51m
SPIDER	**1m51s**	**6m25s**	**23m36s**	**6h07m**

Analysis and Extension

- Complexity: O(*nt* log *t*) comparisons for *n* attributes and *t* tuples
  - Sorting all columns: O(*nt* log *t*)
  - Insertion into minHeap (of size *n*): O(log *n*) for each value
    - O(*nt* log *n*) for all values
  - Popping from heap again O(*nt* log *n*)
  - Intersections in constant time (bit vectors), so O(*nt*) for all
  - Assuming *t* >> *n*: O(*nt* log *t*)
  - I/O complexity is also dominated by sorting

- Extension for partial INDs
  - During intersection:
    - Count how many times intersection removed and attributes.
    - Remove only after *k* unsuccessful intersections

Overview

1. Inclusion Dependencies
2. SQL
3. De Marchi et al.
4. MIND
5. SPIDER
6. **MANY**

Detecting Inclusion Dependencies on Very Many Tables
F. Tschirschnitz, T. Papenbrock, F. Naumann (TODS 2017)

Planets

Planeten Tabelle

Drucken

Geschrieben von Denise. Veröffentlicht in Planeten

Die Planeten des Sonnensystems im Vergleich

	Abstand zur Sonne in Mio km	Durchmesser in km	Masse in kg	Umlaufzeit um die Sonne in Tagen	Geschwindigkeit
Merkur	58	4.879	$3{,}30 \times 10^{23}$	88	172.332 km/h
Venus	108	12.103	$4{,}87 \times 10^{24}$	225	126.072 km/h
Erde	150	12.734	$5{,}97 \times 10^{24}$	365	107.208 km/h
Mars	228	6.772	$6{,}42 \times 10^{23}$	687	86.868 km/h
Jupiter	778	138.346	$1{,}90 \times 10^{27}$	4329 (11 Jahre, 314 Tage)	47.052 km/h
Saturn	1.433	114.632	$5{,}69 \times 10^{26}$	10751 (29 Jahre, 166 Tage)	34.884 km/h
Uranus	2.872	50.532	$8{,}68 \times 10^{25}$	30664 (84 Jahre, 4 Tage)	24.516 km/h
Neptun	4.495	49.105	$1{,}02 \times 10^{26}$	60148 (164 Jahre, 288 Tage)	19.548 km/h
Diagramm					

Die gerundeten (und genauen) Verhältnisse zwischen den Umlaufzeiten der Planeten

Merkur	☿	2:5	(2:5,11)	♀	Venus
Venus	♀	8:13	(8:13,004)	♁	Erde
Erde	♁	1:2	(1:1,88)	♂	Mars
Mars	♂	1:6	(1:6,31)	♃	Jupiter
Jupiter	♃	2:5	(2:4,97)	♄	Saturn
Saturn	♄	1:3	(1:2,85)	♅	Uranus
Uranus	♅	1:2	(1:1,96)	♆	Neptun

Beispiele: Einige Fluchtgeschwindigkeiten

Himmelskörper	v_2 am Äquator in km/s
Merkur	4,3
Venus	10,2
Erde	11,2
Mond	2,3
Mars	5,0
Jupiter	59,6
Saturn	35,5
Uranus	21,3
Neptun	23,3
Pluto	1,1
Sonne	617,3
Sonne im Erdabstand	42,0

Moons of Planets

Zentralkörper	Nr.	Name	*a* Halbachse /km	*e* Exz.	*i* Neig. /°	*T* Umlauf /d	*d* /km	*m* Masse /kg
Erde	I	(Erd-)Mond (Luna)	384.400	0,0055	5,145	27,3217	3476	$7{,}35 \times 10^{22}$
Mars	I	Phobos	9.378	0,0151	1,08	0,3189	22	$1{,}07 \times 10^{16}$
Mars	II	Deimos	23.459	0,0003	0,93	1,262	12	$1{,}8 \times 10^{15}$
Jupiter	I	Io	421.800	0,004	0,036	1,77	3643	$8{,}9 \times 10^{22}$
Jupiter	II	Europa	671.100	0,009	0,467	3,55	3122	$4{,}8 \times 10^{22}$
Jupiter	III	Ganymed	1.070.400	0,001	0,172	7,16	5262	$1{,}5 \times 10^{23}$
Jupiter	IV	Kallisto	1.882.700	0,007	0,307	16,69	4821	$1{,}1 \times 10^{23}$
Jupiter	V	Amalthea	18					18
Jupiter	VI	Himalia	11.					8
Jupiter	VII	Elara	11.					7
Jupiter	VIII	Pasiphae	23.					7
Jupiter	IX	Sinope	23.					6
Jupiter	X	Lysithea	11.					6
Jupiter	XI	Carme	23.					7
Jupiter	XII	Ananke	21.276.000	0,244	148,9	610,5	28	$3{,}0 \times 10^{16}$
Jupiter	XIII	Leda	11.165.000	0,164	27,46	240,9	18	$1{,}1 \times 10^{16}$
Jupiter	XIV	Thebe	221.900	0,018	1,070	0,68	98	$7{,}6 \times 10^{17}$
Jupiter	XV	Adrastea	128.900	0,002	0,027	0,30	16	$1{,}9 \times 10^{16}$
Jupiter	XVI	Metis	128.100	0,001	0,021	0,30	44	$9{,}6 \times 10^{16}$
Jupiter	XVII	Callirrhoe	24.102.000	0,283	147,1	758,8	9	$8{,}7 \times 10^{14}$
Jupiter	XVIII	Themisto	7.507.000	0,242	43,08	130,0	9	$6{,}9 \times 10^{14}$
Jupiter	XIX	Megaclite	23.806.000	0,421	152,8	752,8	6	$2{,}1 \times 10^{14}$

Project goal: Combine (=join) sets of related tables to gather (all) information about a specific entity type.*

*using unary INDs

A Typical Web Table

10 rows

5 columns

Year	Date	""	Name	Team
2006	1	Ōizumi	I am a colspan	
2005	2	F		Canada
2010, (I am destroying 1NF)	3	M	Democratic	Germany
2008	4			France
2003	I am a real cool row- and colspan, you won't find me in a DBMS		Re-Elected	-
2000				" "
2011	6			United Kingdom
2002	7			Japan
	8			-
2009				Australia

no defined foreign key constraints

heterogeneous "null" representation

"uncommon" in relational world

A first approach

*.html (WikiTables project)

```
<table>
  <tr>
    <th>Planet</th>
    <th>Durchmesser</th>
  </tr>
  <tr>
    <td>Merkur</td>
    <td>4879 km</td>
  </tr>
  <tr>
    <td>Venus</td>
    <td>12103 km</td>
  </tr>
...
```

Filtering & Extraction

*.csv

```
Planet,Durchmesser
Merkur,4.879
Venus,12.103
Erde,12.734
Mars,6.772
Jupiter,138.346
Saturn,114.632
Uranus,50.532
Neptun,49.105
```

(N = **1,398,105**)

SPIDER

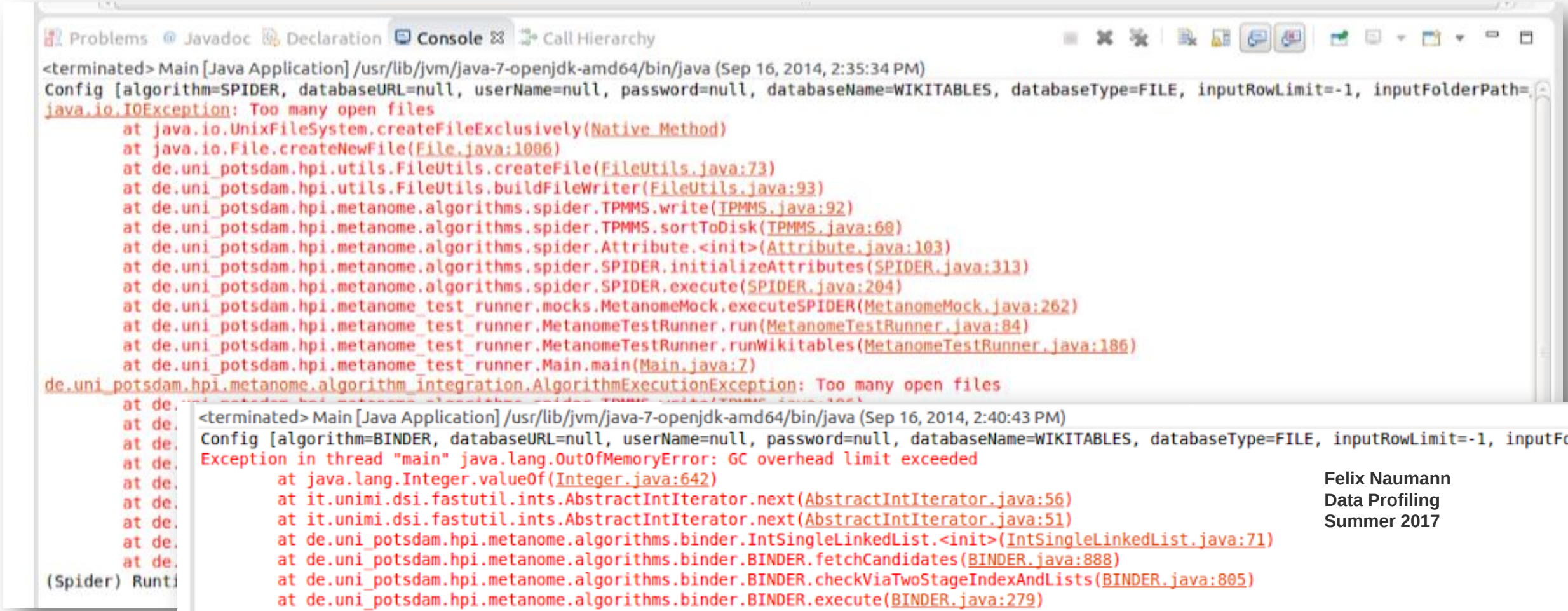
BINDER
(see next slide deck)

INDs

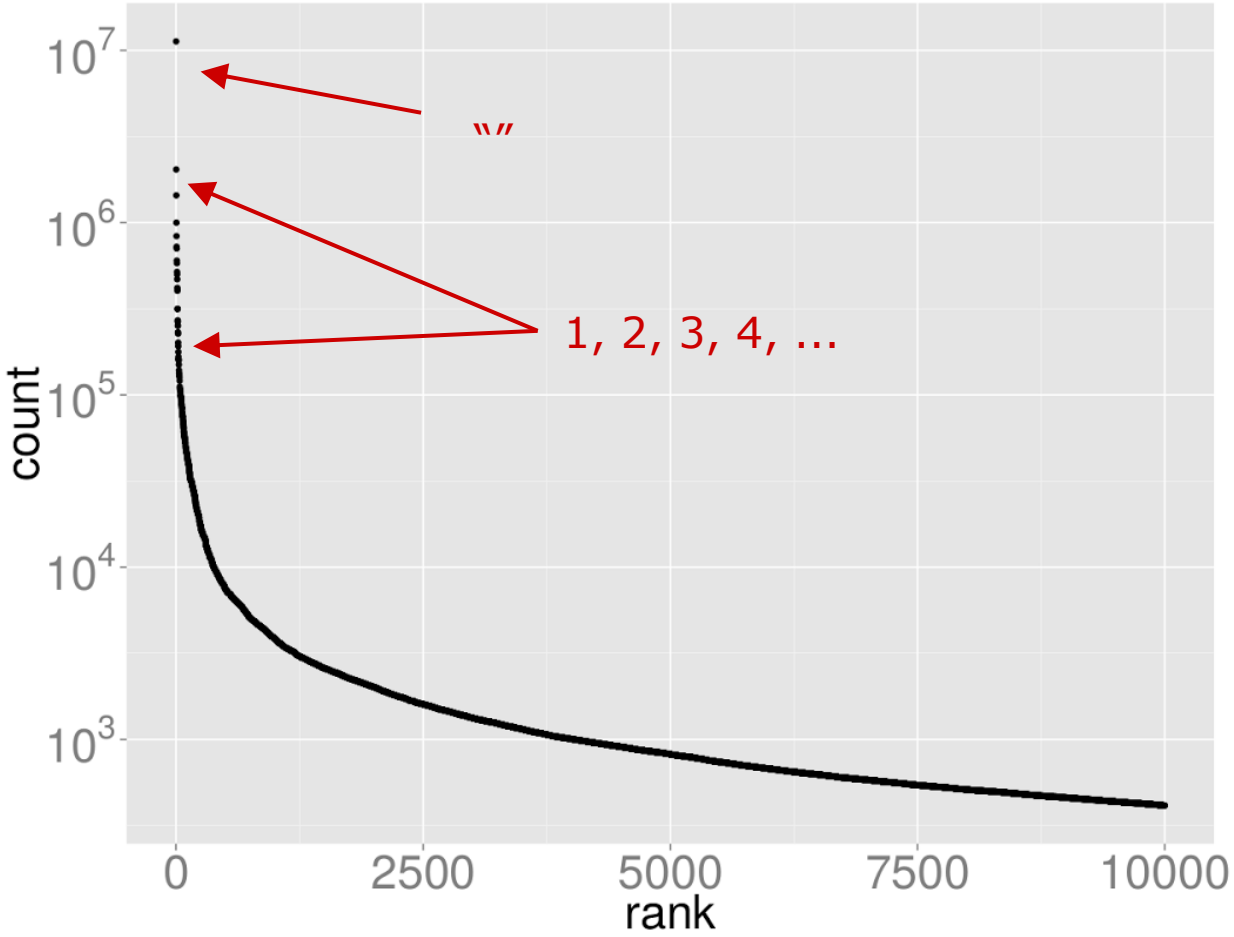
```
[table2.column1]⊆
[table13.column3],
[table2.column7]⊆
[table13.column9],
[table1.column2]⊆
[table13.column8],
[table42.column1]⊆
[table13.column1],
[table6.column1]⊆
...
```

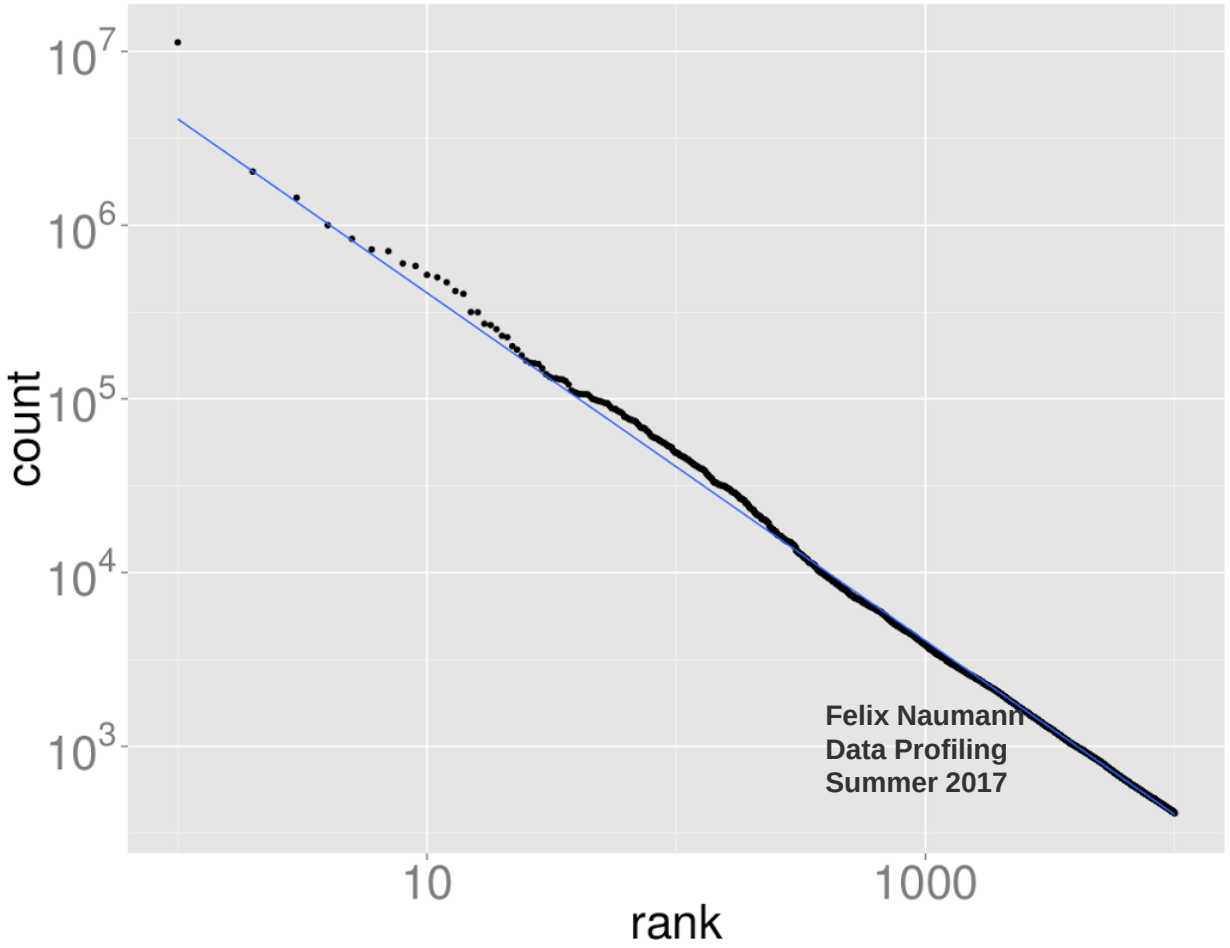
worst case: ≈(1,398,105*5 columns)² INDs

The Result – too many open files (SPIDER) or out of memory (BINDER)



Value distribution

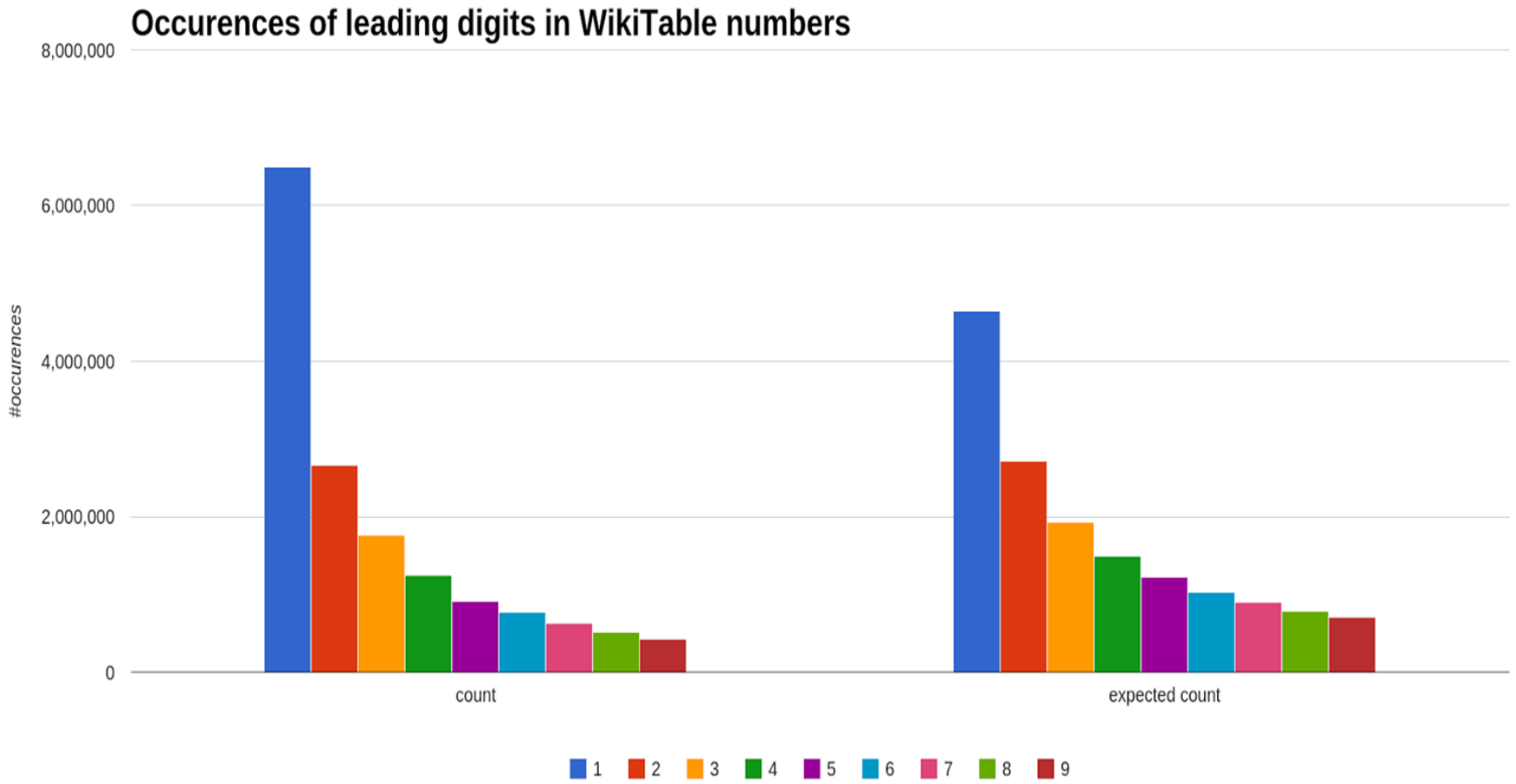




Null values

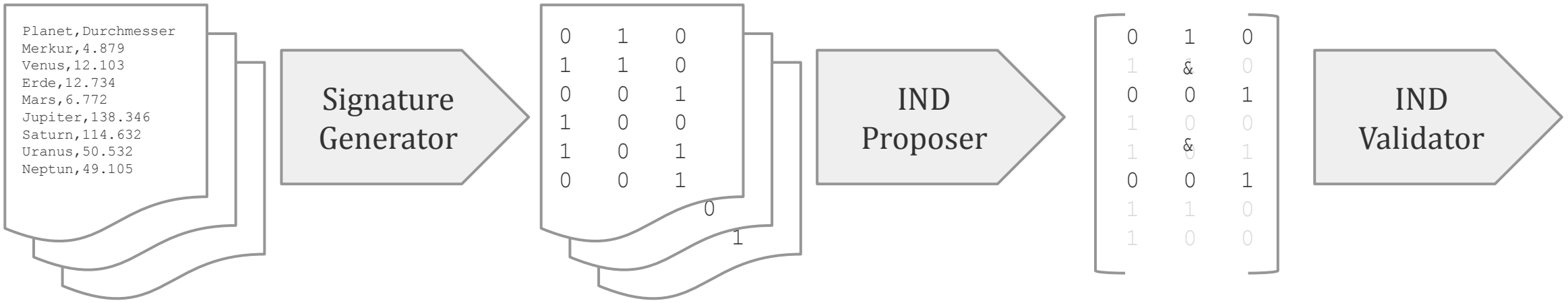
`null` repr.	*tf* rank	occurences	`null` repr.	*tf* rank	occurences
"__"	6th	724,917	"n/a"	182nd	26,596
"-"	11th	501,114	"•"	198st	23,574
"_"	19th	252,008	"- - -"	578th	6,619
"N/A"	33th	131,339	"."	604th	6,367
"?"	49th	97,236	"??"	645th	6,053
"Unknown"	146th	32,484	"(n/a)"	915th	4,228
"- -"	176th	27,898			

Newcomb-Benford's law



Felix Naumann
Data Profiling
Summer 2017

The MANY algorithm



Algorithm 1 MANY

Require: Database schema $D = \{R_1, R_2, \ldots, R_s\}$, instance of database schema $d = \mathcal{I}(D) = \{r_1, r_2, \ldots, r_s\}$, Bloom filter size $m \in \mathbb{N}$, hash-function count $k \in \{1, \ldots, m\}$, *strategy* $\in \{dep2refs, ref2deps\}$

Ensure: All valid INDs *uinds*

```
1:  colIndicesMap ← BUILDCOLINDICESMAP(D)
2:  function MANY(D, d)
3:      uinds ← ∅
4:      sigMatrix ← GENERATESIGNATURES(D, d)
5:      candidates ← GENERATECANDIDATES(sigMatrix)
6:      for all candidate ∈ candidates do
7:          if ISVALID(candidate, d) then
8:              uinds ← uinds ∪ {candidate}
9:      return uinds
10: return MANY(D, d)
```

Reminder: Bloom filters to check presence of value

	"earth"	"mars"	"earth"?	"venus"?
k = 2	$hash_1$("earth") = 2 $hash_2$("earth") = 4	$hash_1$("mars") = 4 $hash_2$("mars") = 7	$hash_1$("earth") = 2 $hash_2$("earth") = 4	$hash_1$("venus") = 0 $hash_2$("venus") = 4
m = 8	0 0 0 0 0 0 0 0	0 0 **1** 0 **1** 0 0 0	0 0 **1** 0 **1** 0 0 **1**	**0** 0 **1** 0 **1** 0 0 **1**

Signature generator

table1

Planet	Symbol	Earth-like
Earth	♁	Y
Mars	♂	Y
Venus	♀	Y

table2

Orb	Atmosphere
Mercury	Y
Venus	Y
Earth	Y
Mars	Y

Bloom filter
m = 8, k = 1

$$\begin{pmatrix}1\\1\\0\\1\\0\\0\\0\\0\end{pmatrix}\quad\begin{pmatrix}0\\0\\1\\0\\0\\0\\1\\1\end{pmatrix}\quad\begin{pmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{pmatrix}\qquad\begin{pmatrix}1\\1\\0\\1\\1\\0\\0\\0\end{pmatrix}\quad\begin{pmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{pmatrix}$$

table1.Planet ⊆ table2.Orb
=>

$$\begin{pmatrix}1\\1\\0\\1\\0\\0\\0\\0\end{pmatrix}\subseteq\begin{pmatrix}1\\1\\0\\1\\1\\0\\0\\0\end{pmatrix}$$

table1.Symbol ⊈ table2.Orb
<=

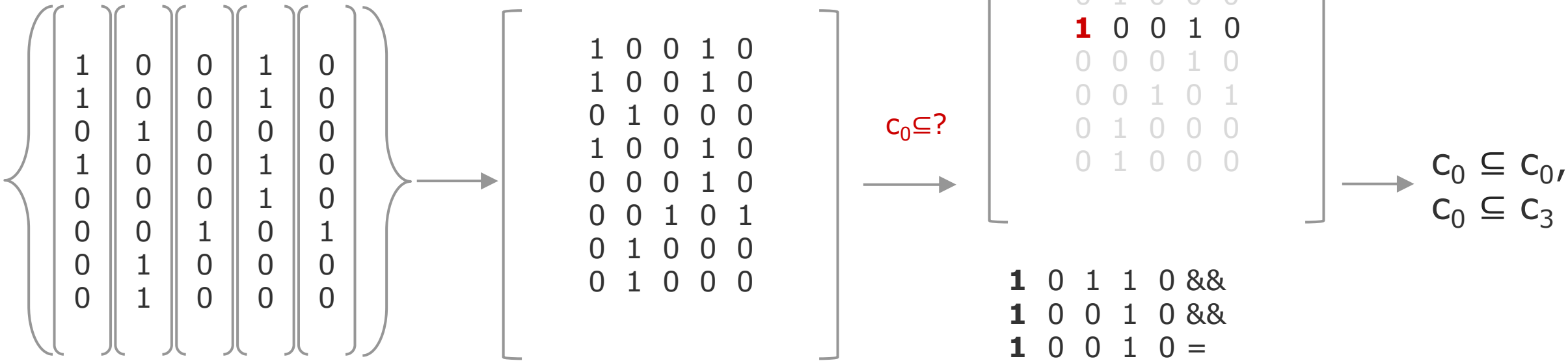
$$\begin{pmatrix}0\\0\\1\\0\\0\\0\\1\\1\end{pmatrix}\nsubseteq\begin{pmatrix}1\\1\\0\\1\\1\\0\\0\\0\end{pmatrix}$$

IND Proposer

table1

	Planet	Symbol	Earth-like	Orb	Atmosphere
index	0	1	2	3	4

table2



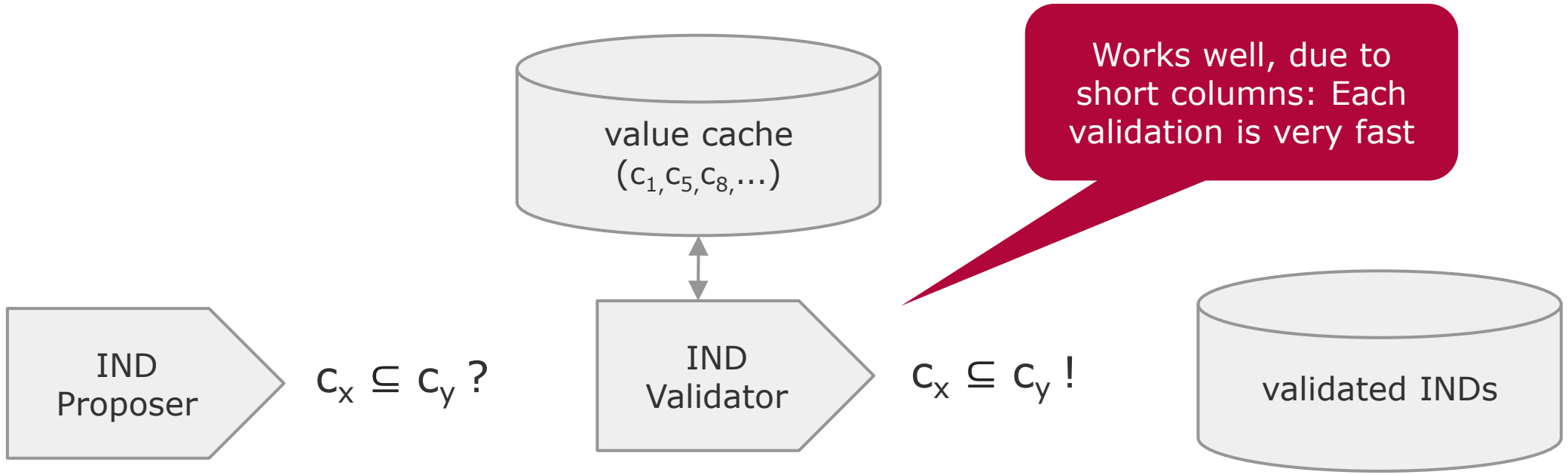
IND Validator

Bloom filter are probabilistic DS:

bf({"Foo", "ABC", "123"}) = (1, 0, 1, 1)
bf({"Bar", "XYZ"}) = (1, 0, 0, 1)

Problem:

{"Bar", "XYZ"} ⊄ {"Foo", "ABC", "123"}
recall = 1
precision ≤ 1



Insight: Check superset relationship, too

table1

Planet	Symbol	Earth-like
Earth	♁	Y
Mars	♂	Y
Venus	♀	Y

table2

Orb	Atmosphere
Mercury	Y
Venus	Y
Earth	Y
Mars	Y

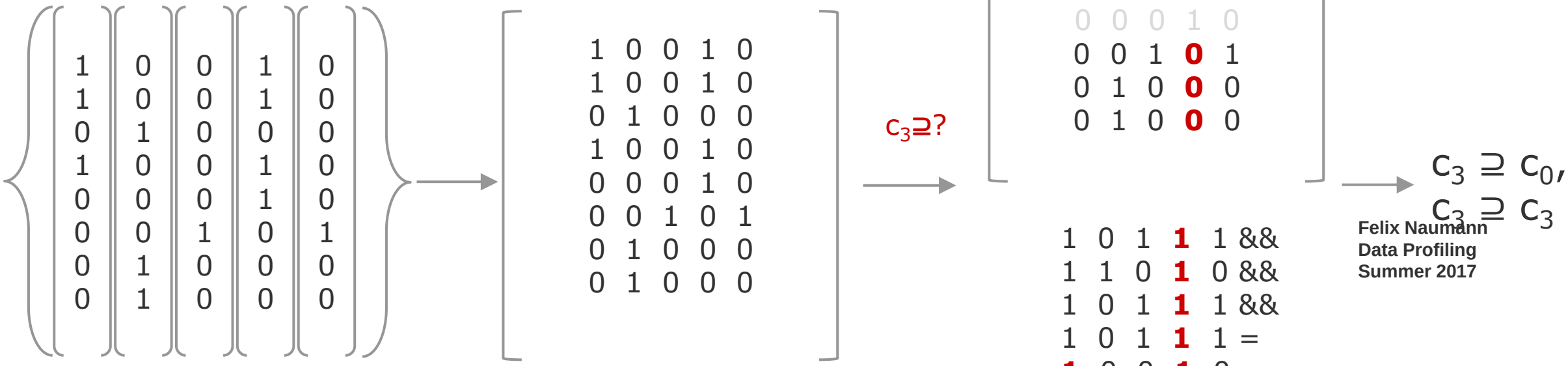
$$\begin{pmatrix}1\\1\\0\\1\\0\\0\\0\\0\end{pmatrix}\quad\begin{pmatrix}0\\0\\1\\0\\0\\0\\1\\1\end{pmatrix}\quad\begin{pmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{pmatrix}\quad\begin{pmatrix}1\\1\\0\\1\\1\\0\\0\\0\end{pmatrix}\quad\begin{pmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{pmatrix}$$

Checking superset relationships to generate candidates

table1

	Planet	Symbol	Earth-like	Orb	Atmosphere
index	0	1	2	3	4

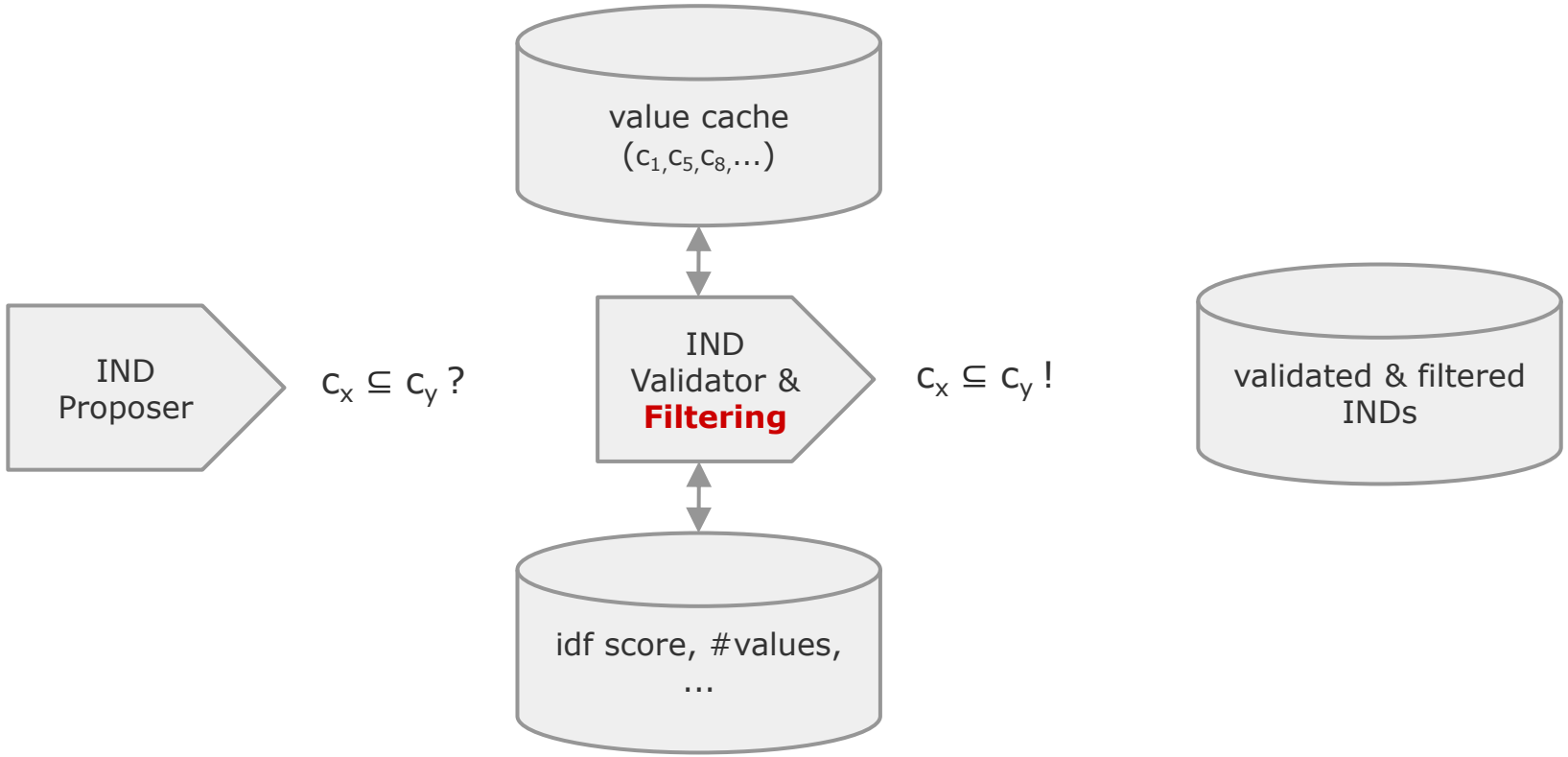
table2



Incremental detection: For each new column, easily generate all candidates

Problem: Too many INDs

#tables	1,000	2,500	5,000	10,000	25,000	50,000
#INDs	122,755	774,375	2,586,716	10,214,320	63,917,448	259,429,353



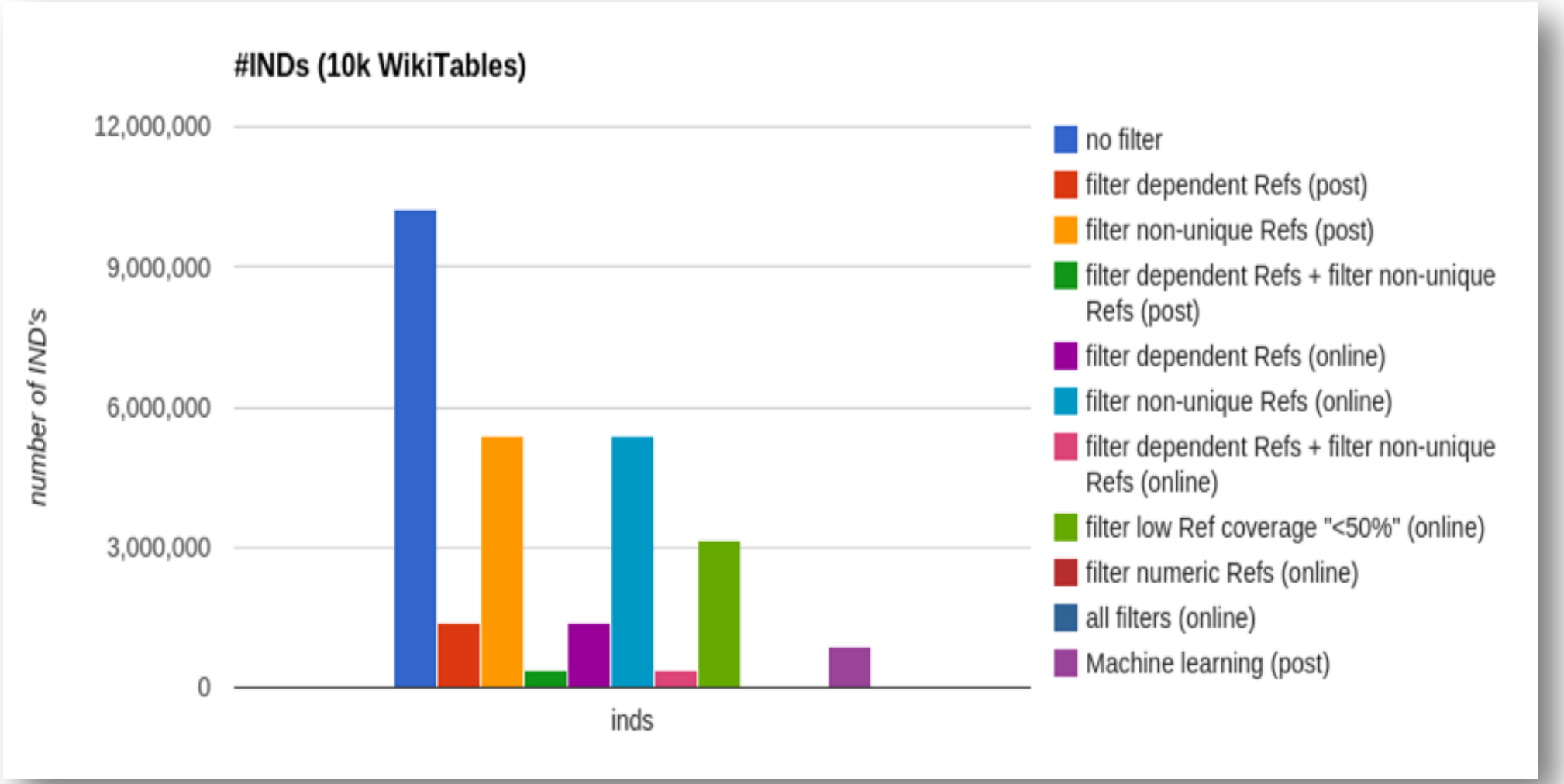
Filters

- Filter INDs with non-unique referenced column
- Filter all transitive INDs {A⊆B, B⊆C, A⊆C}
  - Goal: Reference largest possible column
  - Lose some information
- Filter “uninteresting” (e.g., numeric) columns
- Filter empty columns
- Filter INDs with low coverage on the referenced values

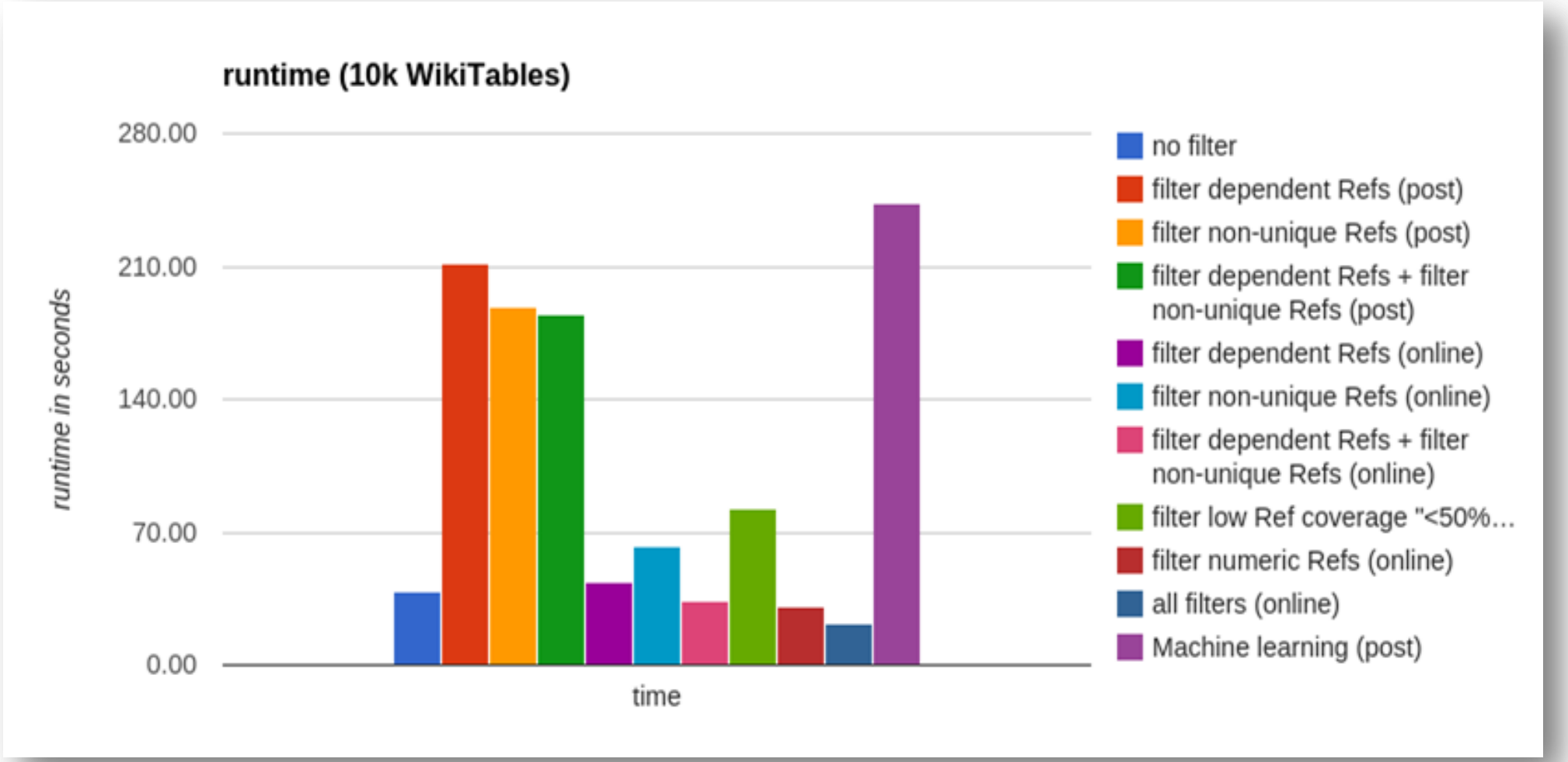
“Online” filter implementation: Use both directions of candidate generation

- Non-unique columns
  - Remove as referenced candidate
- Numeric and null-columns
  - Remove as referenced and as dependent candidate
- Columns referencing other columns
  - Remove as referenced candidate
- Low coverage filter
  - Expensive -> Postprocessing to reduce result-size

Filter Evaluation



Filter Evaluation



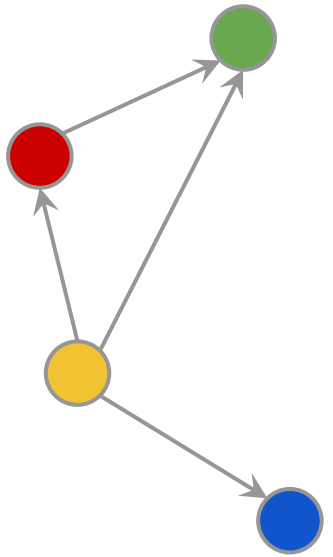
Visualisation

```
[1011066.Name] =] [1011057.Name]
[129284.Reference] =] [1223862.null] [586920.Ref.] [1030730.RCDB page] [108435.No.] [1248790.Source]
  [983315.References] [207338.Home railway (external link)] [975850.Ref] [1375996.Source] [1129539.References]
  [1168707.References] [744488.Ref] [1169311.Ref] [1068498.Ref] [163214.Reference] [604676.References] [1002900.Ref]
  [749972.Reference] [951640.References] [939700.Page] [900853.Ref] [788203.Ref] [788409.References] [978758.Ref]
  [652885.Link] [652377.Ref] [1320358.Reference] [1287392.Ref] [1012269.Report] [1180077.References] [1274408.Ref]
  [856227.NFL Recap] [1286480.Ref] [1354142.null] [525501.References] [630016.Notes] [762537.Refs] [902406.Report]
  [1005369.Link] [1255682.Source] [1157534.Source] [1065320.Ref] [956840.Ref] [775466.References] [988811.Ref]
  [1005838.Link] [1005593.Link] [576411.References] [1134428.Ref] [1170953.Reference(s)] [699144.Note]
  [268733.References] [931606.Notes] [1284557.Ref.] [1357973.Source] [1238931.Report] [867400.Reference]
  [794774.Ref] [716064.Refs] [377521.References] [995370.Ref] [1282132.References] [1358158.Ref.] [1120007.Ref]
  [1342522.Ref] [1319381.null] [889114.Ref] [1004839.Link] [697527.Website] [980509.Ref(s)] [1078901.Ref]
[1390416.Rank] =] [1169921.Rank] [1183098.Rank] [1011765.Rank] [1225076.Rank] [454782.Rank] [1186535.Rank]
  [1209635.Rank] [1161665.Rank] [708465.Rank] [708648.Rank]
[637307.Date] =] [1311505.Date] [1337020.Date]
[1083420.Event] =] [976659.Event] [976901.Event] [975917.Event] [1060037.Event] [1068182.Event] [1067251.Event]
  [1067097.Event] [1000067.Event] [972968.Event] [1058267.Event] [988323.Event] [1003312.Event] [1063506.Event]
  [1027145.Event] [1078507.Event] [1062268.Event]
[302006.Role:] =] [391330.Role:] [703281.Role:] [387497.Role:] [735612.Role:] [151885.Role:] [150598.Role:]
[1083410.Event] =] [983546.Event] [975773.Event] [1071989.Event] [1068219.Event] [1002900.Event] [1074984.Event]
  [967160.Event] [1052352.Event] [1066949.Event] [1082562.Event] [1151162.Event] [1042660.Event] [1056643.Event]
  [950860.Event] [958921.Event] [1063309.Event] [973967.Event] [1027145.Event] [1062263.Event]
[73362.State] =] [1185141.State]
[1083402.Event] =] [1083339.Event] [1068498.Event] [1060027.Event] [1002823.Event] [1046135.Event] [1249836.Event]
  [1000145.Event] [994576.Event] [990543.Event]
[854590.Venue] =] [883202.Venue] [890993.Venue] [1104659.Venue]
[648260.TEAM] =] [1286540.Club] [1308745.Club]
[627822.Division Record] =] [466958.Sets W - L]
[1236345.Match] =] [1231569.Match]
...
```

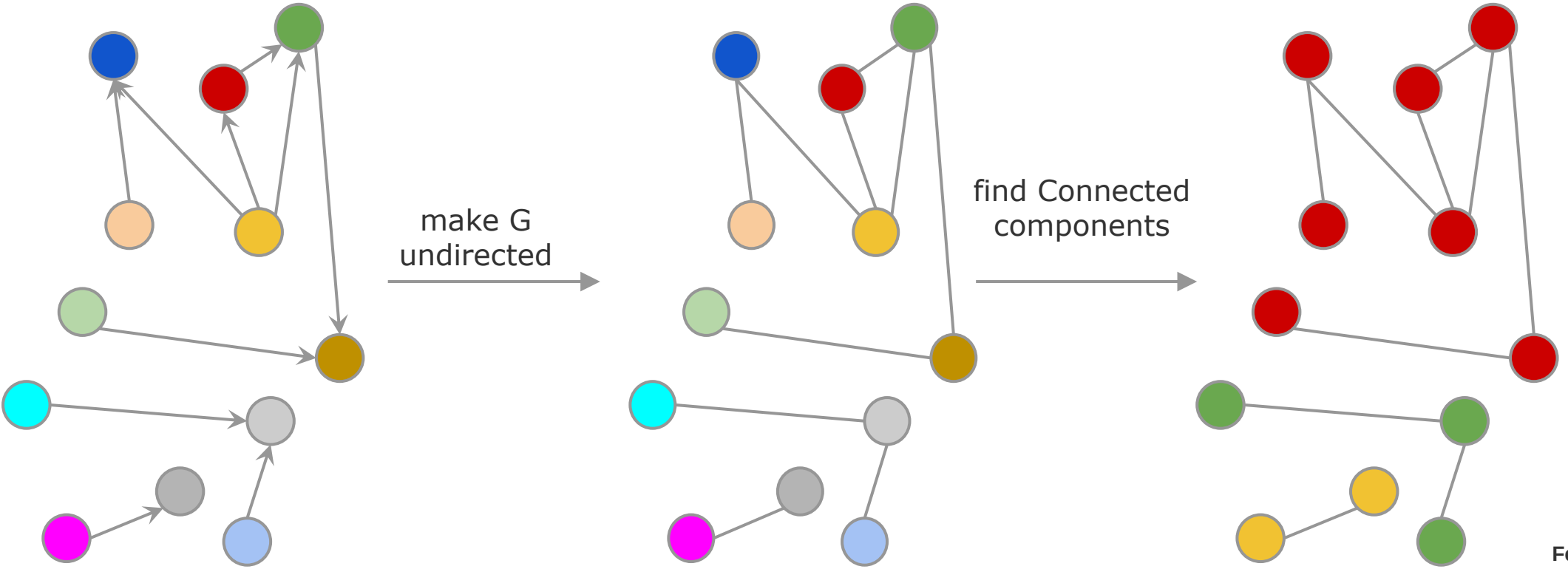
Visualisation

$INDs = \{$
$R_1.A \subseteq R_2.B,$
$R_3.A \subseteq R_1.D,$
$R_3.C \subseteq R_2.A,$
$R_3.B \subseteq R_4.A$
$\}$

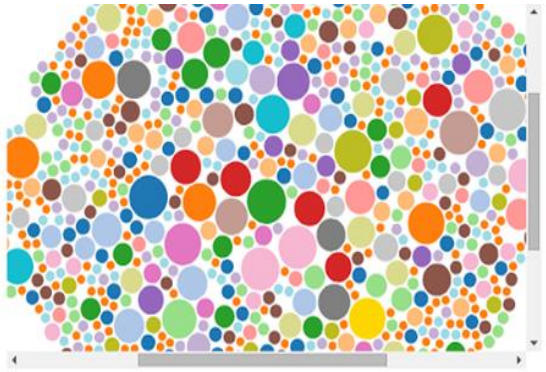
$G = ($
$V = \{$
R_1, R_2, R_3, R_4
$\},$
$E = \{(R_1, R_2), (R_3, R_1), (R_3, R_2), (R_3, R_4)$
$\}$
$)$

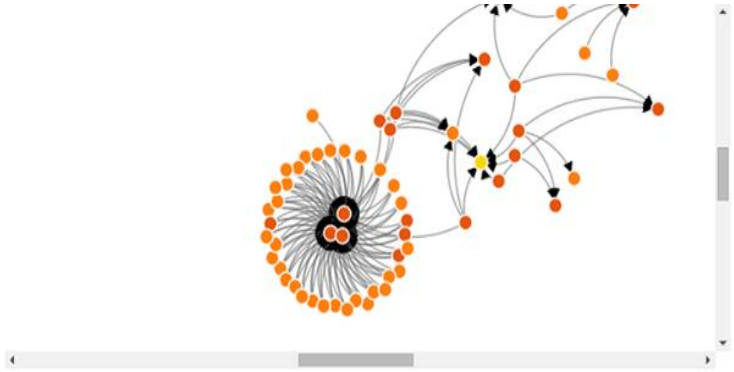


Visualisation



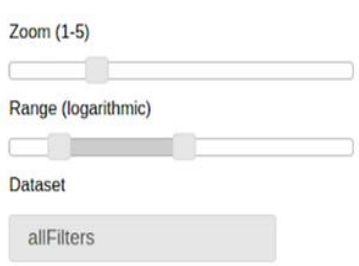
Application





96242-1	Association'.csv
43666-3	43666-3.'BBC_Radio_Stoke'.'Programming'.csv
53064-1	53064-1.'Rotation_period'.'Rotation period of selected objects'.csv
562884-4	562884-4.'Planets_in_astrology'.'Ruling planets of the astrological signs and houses'.csv
175797-1	175797-1.'Sun_sign_astrology'.'Sun signs'.csv
177750-2	177750-2.'BBC_Radio_Manchester'.'Programming'.csv
89462-4	89462-4.'Astrology_and_the_classical_elements'.'Triplicities by season'.csv
213213-1	213213-1.'Dalton_Park'.'Opening times'.csv
	470402-

Celestial Objects	Rotation period	Rotation period
Sun	25.379995 days (equatorial) 35 days (high latitude)	25 d 9 h 7 m 11.6 s 35 d
Mercury	58.6462 days	58 d 15 h 30 m 30 s
Venus	?243.0187 days	?243 d 0 h 26 m
Earth	0.99726968 days	0 d 23 h 56 m 4.100 s
Moon	27.321661 days (synchronous toward Earth)	27 d 7 h 43 m 11.5 s
Mars	1.02595675 days	1 d 0 h 37 m 22.663 s
Ceres	0.37809 days	0 d 9 h 4 m 27.0 s
Jupiter	0.4135344 days (deep interior) 0.41007 days (equatorial) 0.41369942 days (high latitude)	0 d 9 h 55 m 29.37 s 0 d 9 h 50 m 30 s 0 d 9 h 55 m 43.63 s
Saturn	0.44403 days (deep interior) 0.426 days (equatorial) 0.443 days (high	0 d 10 h 39 m 24 s 0 d 10 h 14 m 0 d 10 h 38 m



Felix Naumann
Data Profiling
Summer 2017

Summary

1. Inclusion Dependencies
2. SQL
3. De Marchi et al.
4. MIND
5. SPIDER
6. MANY