HPI Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Detecting Inclusion Dependencies

25.4.2013

Felix Naumann

Overview

- Dependencies
- Inclusion Dependencies
- SQL
- De Marchi et al.
- SPIDER
- Foreign Key Detection

Constraints in Databases

- Relational model defines very high level semantics
  - The „relation“
- But no intended „meaning“ of the stored tuples
- No implicit metadata

- Constraints are a form to add such metadata
  - „Integrity constraints“
  - Must be satisfied by all instances of a database schema
- In general: Any expression from first-order logic
- Restricted class of constraints: Dependencies
  - More feasible to reason about and validate
- Important topic in database theory
  - Main question there: logical implication
  - Given a set of dependencies Σ and a dependency σ, if an instance satisfies Σ, does it also satisfy σ?

Many kinds of dependencies

From Abiteboul, Hull, Vianu: Foundations of Databases, 1995

Some important dependencies

- Functional dependencies
  - Values of some attributes functionally determine those of other attributes.
  - Movies: Title -> Director
  - Showings: Theater, Screen -> Title

- Key dependency: Special case of FDs
  - Left side of FD implies all (other) attributes

Some important dependencies

- Join dependency
  - Multivalued dependencies (MVDs) are a special case
  - Showings(Theater, Screen, Title, Snack)
  - Instance I = $\pi_{\text{Theater, Screen,Titel}}(I) \bowtie \pi_{\text{Theater, Snacks}}(I)$

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

- ...and inclusion dependencies

Overview

- Dependencies
- Inclusion Dependencies
- SQL
- De Marchi et al.
- SPIDER
- Foreign Key Detection

Inclusion Dependencies: Definition

- INDs 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 $\sigma = R[A_1, \ldots, A_n] \subseteq S[B_1, \ldots, B_n]$, 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 σ if $I(R)[A_1, \ldots, A_n] \subseteq I(S)[B_1, \ldots, B_n]$
- Values of R: “dependent values”
- Values of S: “referenced values”

Example

- Each Title in Showings should appear as a Title in Movies
  - Showings[Title] $\subseteq$ 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”

Inference rules for INDs

- Reflexivity: $R[X] \subseteq R[X]$

- Projection:
  - $R[A_1, \ldots, A_n] \subseteq S[B_1, \ldots, B_n]$
  - => $R[A_{i1}, \ldots, A_{im}] \subseteq S[B_{i1}, \ldots, B_{im}]$ for each sequence i1, ...,im of Integers in {1,...,n}

- Transitivity:
  - $R[X] \subseteq S[Y]$ and $S[Y] \subseteq T[Z]$
  - => $R[X] \subseteq T[Z]$

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

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

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 match multiple referenced values

- Example

A
bbc

B
b
bb

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

IND types

- Conditional INDs
  - Only useful for partial INDs
  - More next week

Catalog

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

ToyDB

EntityID	further data
17	abcd...
18	efgh...

FashionDB

EntityID	further data
18	abcd...
19	efgh...

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 complianc
116D 2011-07-13 4 0000 'Version format complianc
#
_software.name                NUCLSO
```

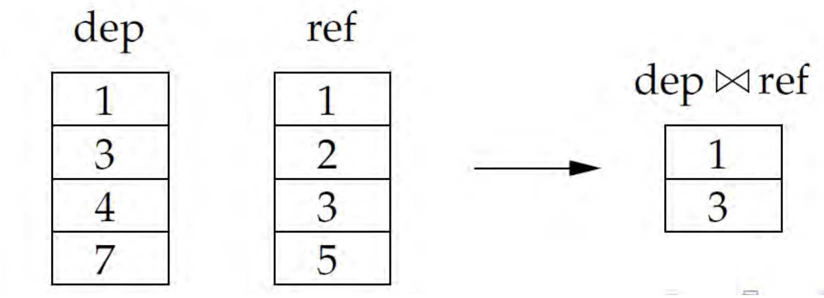
Motivation for IND discovery

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

Overview

- Dependencies
- Inclusion Dependencies
- SQL
- De Marchi et al.
- SPIDER
- Foreign Key Detection

The JOIN




- ```
  SELECT COUNT(depColumn) AS numDeps
  FROM depTable
  ```
- ```
  SELECT COUNT(*) AS matchedDeps
  FROM depTable JOIN refTable
  ON depTable.depColumn = refTable.refColumn
  ```

- (numDeps = matchedDeps) $\Leftrightarrow$ depColumn $\subseteq$ 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 $\Leftrightarrow$ depColumn $\subseteq$ 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 ⊆ refColumn
  ⇔ 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 ⊆ refColumn
  ⇔ unmatched = 0

Measurements (2006)

DB size	CATH 20 MB	SCOP 17,5 MB	UniProt 900 MB	TPC-H 1.3 GB	PDB 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.

- 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)

Discussion on data profiling experiments

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

Overview

- Dependencies
- Inclusion Dependencies
- SQL
- De Marchi et al.
- SPIDER
- Foreign Key Detection

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

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

Example

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

- 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)
- Build relation with single full scan of each base relation

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* *(A⊆B)*, then by construction *B* will be present in all lines of the binary relation containing *A*.

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

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 $\mathbf{d}$ and t.

Output: $\mathcal{I}_1$ the set of unary INDs verified by $\mathbf{d}$ between attributes of type t.

```
for all A ∈ 𝕌 do rhs(A) = 𝕌;
for all v ∈ 𝕍 do
    for all A s.t. (v, A) ∈ 𝔹 do
        rhs(A) = rhs(A) ∩ {B | (v, B) ∈ 𝔹};
for all A ∈ 𝕌 do
    for all B ∈ rhs(A) do
        I₁ = I₁ ∪ {A ⊆ B};
return I₁.
```

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

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

- 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

Question: Why distinguish domains?

Overview

- Dependencies
- Inclusion Dependencies
- SQL
- De Marchi et al.
- SPIDER
- Foreign Key Detection

Making use of order

- Idea: Order each column only once
  - Index in DBMS
  - Sorted columns as individual files

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

A	B
1	1
3	2
4	3
7	5

Brute force approach

- Sequentially check each column pair
- Re-use order for each attribute
  - For each attribute: **`SELECT DISTINCT A FROM R ORDER BY A`**
  - Store result in file
- Problem: Run through data multiple times
  - $A \subseteq C$
  - $A \subseteq D$
  - $B \subseteq C$
  - $B \subseteq D$

A	B	C	D
1	2	1	1
3	3	3	2
4	5	7	3
		8	5

Testing a single IND candidate

- 2 ordered lists of distinct values: depValues and refValues
- **while** (depValues has next)
  - currentDep = depValues.next();
  - **if** (refValues is empty) **then return** false;
  - **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

SPIDER

Single Pass Inclusion DEpendency Recognition

- Main ideas
  - Test all IND-candidate pairs in parallel.
  - 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 in parallel
- Sources:
  - 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.

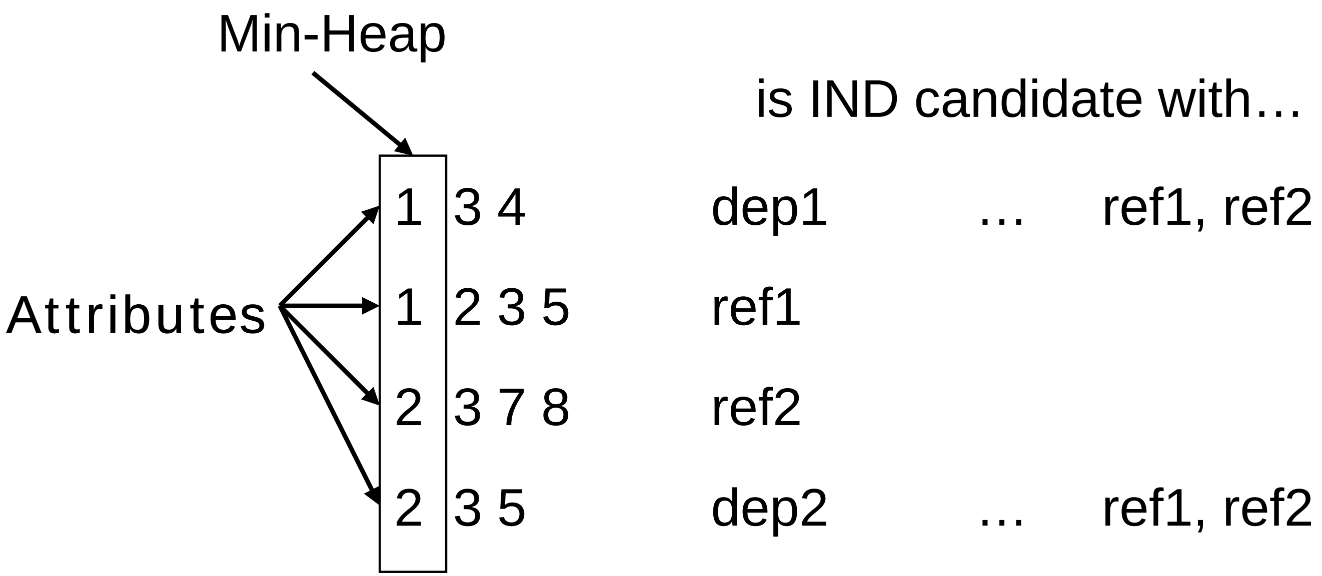
SPIDER

- 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.

dep1	dep2	ref1	ref2
→ 1	→ 2	→ 1	→ 1
3	3	2	3
4	5	3	7
		5	8

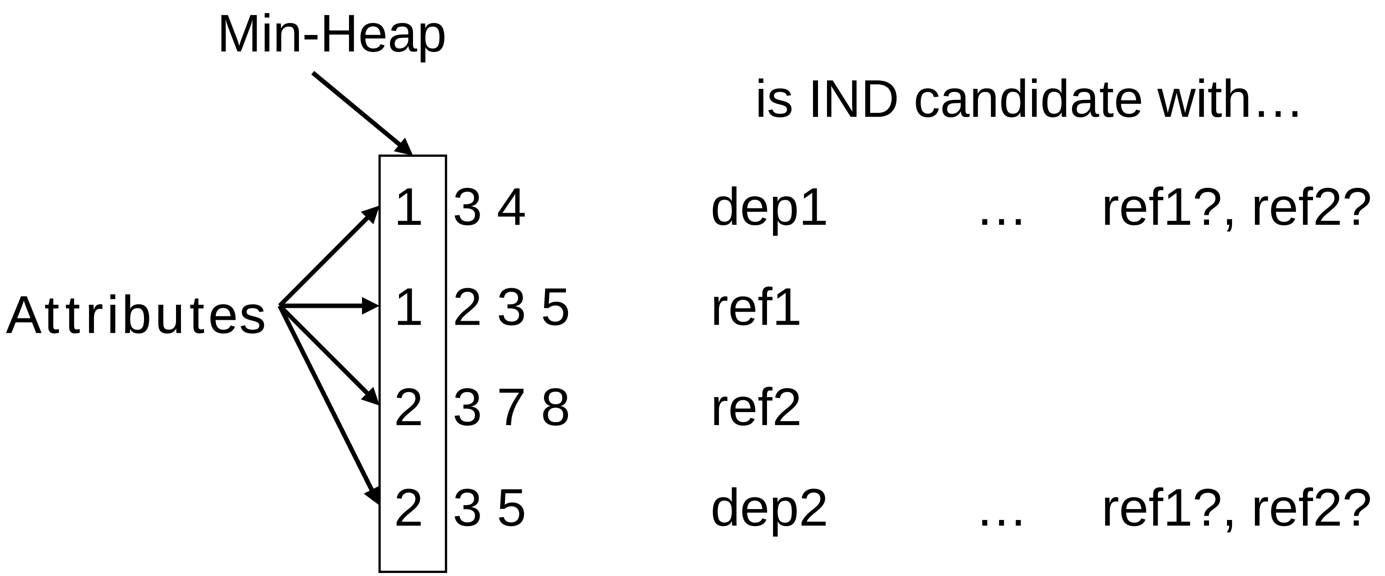
SPIDER: Idea

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



SPIDER: Idea

- In reference list: Distinguish for referenced attribute whether current dependent value has
  - been seen in referenced attribute, or
  - not (yet) been seen in referenced attribute.
- Simultaneous processing of all attributes with same current value, checking all (still valid) IND candidates



SPIDER by example

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	∅

- In each step: Intersect „attributes to process“ with each refs list of previous step

SPIDER results

	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
Bell & Brockhausen	4m39s	-	1h32m	-
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

- Dependencies
- Inclusion Dependencies
- SQL
- De Marchi et al.
- SPIDER
- Foreign Key Detection

Problem: Automatic Determination of Foreign Keys

- Given
  - Relational schema
  - Database instance of that schema
  - Complete set of (observed) inclusion dependencies
    - Attributes A and B with $R[A] \subseteq S[B]$ (in short $A \subseteq B$)
- Find
  - All foreign key constraints: attributes A and B with $A \rightarrow B$
- Difficulty
  - Foreign keys are not intrinsic to data, but defined by humans
  - Discover semantics
- An aside: INDs, FKs, and humans: Cannot be „discovered“

Characterizing foreign keys

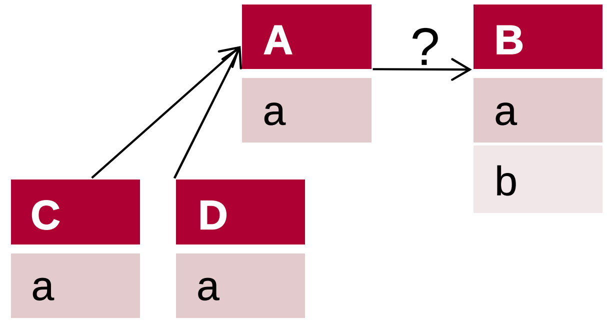
- Find set of characteristic features
  - Easily verifiable
  - Carefully developed
  - Not necessarily independent

- Notation-reminder
  - FK candidate: $A \rightarrow B$
    - Given IND $A \subseteq B$
  - Let *s*(A) denote set of distinct values in attribute A.
  - Let *name*(A) denote the label of attribute A.

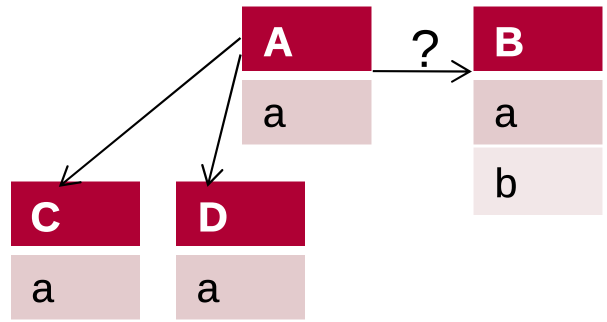
- Source: Alexandra Rostin, Oliver Albrecht, Jana Bauckmann, Felix Naumann, UlfLeser: A Machine Learning Approach to Foreign Key Discovery. In: WebDB 2009

Features

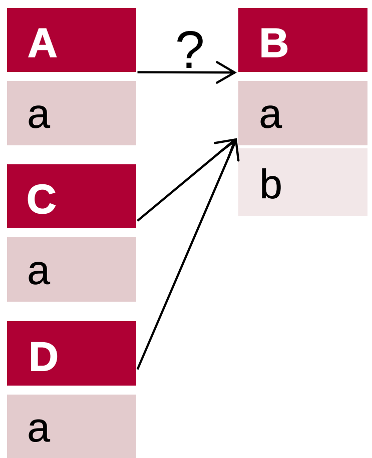
- DependentAndReferenced (F3)
  - Counts how often the dependent attribute A appears as referenced attribute in the set of all INDs.
  - Usually, a foreign key is not also a primary key that is referenced as foreign key by other tables.



- MultiDependent (F4)
  - Counts how often A appears as dependent attribute in the set of all INDs.
  - If *s*(A) is contained in the set of values of many other attributes, the likelihood for each of these INDs being a FK is decreased.



- MultiReferenced (F5)
  - Counts how often B appears as referenced attribute in the set of all INDs.
  - Often, primary keys are referenced by more than one foreign key.



Features

- DistinctDependentValues (F1)
  - The cardinality of *s*(A).
  - Usually, attributes that are foreign keys contain at least some different values.

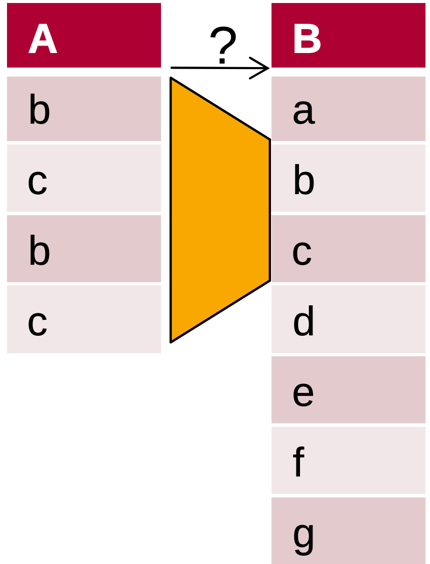
A	? →	B
a		a
a		b
a		c
a		d
a		e

- ValueLengthDiff (F7)*
  - Difference between the average value length (as string) in *s*(A) and *s*(B).
  - Usually, average length of the values is similar whenever foreign keys reference a non-biased sample of the primary keys.

A	? →	B
abab		abab
abab		b
abab		c
c		d
d		e

Features

- Coverage (F2)*
  - The ratio of values in *s*(B) that are covered by *s*(A) compared to all values in *s*(B).
  - Usually, foreign keys cover a considerable number of primary key values.
    - 60% of FK-attribute values cover all ref-values
    - Each covers at least 10%
- OutOfRange (F8)*
  - Percentage of values in *s*(B) that are not within [min(*s*(A)), max(s(A))].
  - Usually, the dependent values should be evenly distributed over the referenced values.
  - Mostly, less than 5% of values outside of range
- TableSizeRatio (F10)
  - Ratio of number of tuples in A and number of tuples in B.
  - Usually in life sciences databases, table sizes do not differ wildly



Features

- ColumnName (F6)*
  - Similarity between *name*(A) and *name*(B), also considering the name of the table of which B is an attribute.
  - Currently: Exact matches or complete containment
- TypicalNameSuffix (F9)
  - Checks whether *name*(A) ends with a substring that indicates a foreign key.
  - Currently only „id“, „key“, and „nr“ (German for “number”)

SG_BIOENTRY.TAX_OID
→ SG_TAXON.OID

COURSE.STUDENT
→ STUDENT.ID

SG_SEQFEATURE.ENT_OID
→ SG_COMMENT.ENT_OID

CUSTOMER.C_NATIONKEY
→ NATION.N_NATIONKEY

FILMTEXTE.FILMTEXTTYPNR
→ FILMTEXTTYPEN.FILMTEXTTYPNR

Learning to classify based on features

- Four (supervised) machine learning methods
  - Naive Bayes
  - Support Vector Machine
  - J48 decision tree
  - Decision tables
- Implementation as provided by WEKA
  - http://www.cs.waikato.ac.nz/ml/weka/
- Cross validation at database level
  - Not at IND level
- Validation with unknown data source
  - MSD

F-Measure results

- Cross-validation
  - Training on all but test database
  - MSD held back completely

Test database	Naive Bayes	SVM	J48	DecisionTab	Avg
UniProt	0.86	0.92	0.84	0.8	0.855
Filmdienst	0.80	0.86	0.86	0.93	0.817
Movielens	0.71	0.71	1.0	0.8	0.805
SCOP	1.0	1.0	1.0	1.0	1.0
TPC-H	0.86	0.90	0.95	0.95	0.915
Average	**0.846**	**0.78**	**0.930**	**0.896**	

- Results for MSD, trained on all others

Test database	Naive Bayes	J48	DecisionTab
MSD	0.84	0.78	0.79

Summary

- Dependencies
- Inclusion Dependencies
- SQL
- De Marchi et al.
- SPIDER
- Foreign Key Detection