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 $\Sigma$ and a dependency $\sigma$, if an instance satisfies $\Sigma$, does it also satisfy $\sigma$?
Many kinds of dependencies

general, 234
generalized dependency constraints, 234
generalized mutual, 234
implication
in view, 221
implication of, 160, 164, 193, 197
implicational (id), 233
implied, 234
inclusion (ind), 161, 192–211, 193, 218, 250
acyclic, 207, 208–210, 211, 250
key-based, 250, 260
typed, 213
unary (uid), 210–211
inference rule, 166, 172, 193, 227, 231
ground, 203
join (jd), 161, 169–173, 170, 218
key, 157, 163–169, 163, 267
logical implication of, 160, 164
finite, 197
unrestricted, 197
multivalued (mvd), 161, 169–173, 170, 186, 218
mutual, 233
named vs. unnamed perspectives, 159
order, 234
partition, 234
projected join, 233
and query optimization, 163
satisfaction, 160
satisfaction by tableau, 175
satisfaction family, 174
and semantic data models, 249–253
and schema design, 253–262
single-head vs. multi-head, 217
sort set, 191, 213, 234
subset, 233
tagged, 164, 221, 241
template, 233, 236
transitive, 234
trivial, 220
tuple-generating (tgd), 217–228
typed, 159
vs. untyped, 192, 217
unirelational, 217
and update anomalies, 162
and views, 221, 222
vs. first-order logic, 159, 234
vs. integrity constraint, 157
vs. tableaux, 218, 234
dependency basis, 172
dependency preserving decomposition, 254
dependent class, 246
dereferencing, 557, 558
derivation, 290

Felix Naumann | Profiling & Cleansing | Summer 2013
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 $I = \pi_{\text{Theater, Screen, Title}}(I) \bowtie \pi_{\text{Theater, Snacks}}(I)$

- **...and inclusion dependencies**

Felix Naumann | Profiling & Cleansing | Summer 2013
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: $R[A_1, \ldots, A_n] = \pi_{A_1, \ldots, A_n}(R)$

- $\text{IND } \sigma = R[A_1, \ldots, A_n] \subseteq S[B_1, \ldots, B_n]$, 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 $\sigma$ 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] ⊆ Movie[Title]

<table>
<thead>
<tr>
<th>Movies</th>
<th>Title</th>
<th>Director</th>
<th>Actor</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Birds</td>
<td>Hitchcock</td>
<td>Hedren</td>
<td></td>
</tr>
<tr>
<td>The Birds</td>
<td>Hitchcock</td>
<td>Taylor</td>
<td></td>
</tr>
<tr>
<td>Bladerunner</td>
<td>Scott</td>
<td>Hannah</td>
<td></td>
</tr>
<tr>
<td>Apocalypse Now</td>
<td>Coppola</td>
<td>Brando</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Showings</th>
<th>Theater</th>
<th>Screen</th>
<th>Title</th>
<th>Snack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rex</td>
<td>1</td>
<td>The Birds</td>
<td>coffee</td>
<td></td>
</tr>
<tr>
<td>Rex</td>
<td>1</td>
<td>The Birds</td>
<td>popcorn</td>
<td></td>
</tr>
<tr>
<td>Rex</td>
<td>2</td>
<td>Bladerunner</td>
<td>coffee</td>
<td></td>
</tr>
<tr>
<td>Rex</td>
<td>2</td>
<td>Bladerunner</td>
<td>popcorn</td>
<td></td>
</tr>
<tr>
<td>Le Champo</td>
<td>1</td>
<td>The Birds</td>
<td>tea</td>
<td></td>
</tr>
<tr>
<td>Le Champo</td>
<td>1</td>
<td>The Birds</td>
<td>popcorn</td>
<td></td>
</tr>
<tr>
<td>Cinoche</td>
<td>1</td>
<td>The Birds</td>
<td>Coke</td>
<td></td>
</tr>
<tr>
<td>Cinoche</td>
<td>1</td>
<td>The Birds</td>
<td>wine</td>
<td></td>
</tr>
<tr>
<td>Cinoche</td>
<td>2</td>
<td>Bladerunner</td>
<td>Coke</td>
<td></td>
</tr>
<tr>
<td>Cinoche</td>
<td>2</td>
<td>Bladerunner</td>
<td>wine</td>
<td></td>
</tr>
<tr>
<td>Action Christine</td>
<td>1</td>
<td>The Birds</td>
<td>tea</td>
<td></td>
</tr>
<tr>
<td>Action Christine</td>
<td>1</td>
<td>The Birds</td>
<td>popcorn</td>
<td></td>
</tr>
</tbody>
</table>

- Aka. “referential integrity”
Inference rules for INDs

- **Reflexivity:** \( R[X] \subseteq R[X] \)

- **Projection:**
  - \( R[A_1, ..., A_n] \subseteq S[B_1, ..., B_n] \)
  - \( \Rightarrow R[A_{i1}, ..., A_{im}] \subseteq S[B_{i1}, ..., B_{im}] \) for each sequence \( i_1, ..., i_m \) of Integers in \( \{1, ..., n\} \)

- **Transitivity:**
  - \( R[X] \subseteq S[Y] \) and \( S[Y] \subseteq T[Z] \)
  - \( \Rightarrow R[X] \subseteq T[Z] \)
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**: \( R[C] \subseteq S[F] \)

- **N-ary**: \( R[B,C] \subseteq S[G,F] \)

- **Partial**: \( R[A] \subseteq \text{75\% } S[F] \)

- **Approximate**: \( R[BA] \subseteq S[HG] \)
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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>bbc</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>bb</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Catalog</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit cost</td>
<td>DBName</td>
<td>ProdID</td>
<td></td>
</tr>
<tr>
<td>200 USD</td>
<td>ToyDB</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>50 EUR</td>
<td>ToyDB</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>1000 QAR</td>
<td>FashionDB</td>
<td>18</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ToyDB</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EntityID</td>
<td>further data</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>abcд...</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>efgh...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FashionDB</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EntityID</td>
<td>further data</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>abcд...</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>efgh...</td>
<td></td>
</tr>
</tbody>
</table>
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!
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) ⇔ depColumn ⊆ refColumn

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

- \[ \text{SELECT} \ \text{count}(*) \ \text{AS} \ \text{unmatchedDeps} \ \text{FROM} \]
  \[ (\text{(SELECT} \ \text{to char} \ (\text{depColumn}) \]
  \[ \ \text{FROM} \ \text{depTable} \]
  \[ \ \text{WHERE} \ \text{depColumn} \ \text{IS NOT NULL} \]
  \[ \ \text{EXCEPT} \]
  \[ \ \text{(SELECT} \ \text{to char} \ (\text{refColumn}) \]
  \[ \ \text{FROM} \ \text{refTable} \]
  \[ \) \]
  \[ \ \text{FETCH FIRST} \ 1 \ \text{ROWS} \ \text{ONLY} \]
  \[ \) \]

- \[ \text{unmatchedDeps} = 0 \iff \text{depColumn} \subseteq \text{refColumn} \]
The “Antijoin”

- \[ \text{SELECT COUNT}(*) \text{ AS } \text{unmatched} \]
  \[ \text{FROM R} \]
  \[ \text{WHERE } A \text{ IS NOT NULL} \]
  \[ \text{AND } A \text{ NOT IN} \]
  \[ (\text{SELECT B FROM S}) \]
  \[ \text{FETCH FIRST 1 ROWS ONLY} \]

- \[ \text{depColumn} \subseteq \text{refColumn} \]
  \[ \iff \text{unmatched} = 0 \]

- \[ \text{SELECT COUNT}(*) \text{ AS } \text{unmatched} \]
  \[ \text{FROM R} \]
  \[ \text{WHERE } A \text{ IS NOT NULL} \]
  \[ \text{AND NOT EXISTS} \]
  \[ (\text{SELECT * FROM S WHERE R.A=S.B}) \]
  \[ \text{FETCH FIRST 1 ROWS ONLY} \]

- \[ \text{depColumn} \subseteq \text{refColumn} \]
  \[ \iff \text{unmatched} = 0 \]
Measurements (2006)

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

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
Key idea: For a given domain associate each value with every attribute having this value.

- Create binary relation \( B \subseteq \text{Values} \times \text{Attributes} \) 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

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“

<table>
<thead>
<tr>
<th>r</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>3</td>
<td>11.0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>3</td>
<td>12.0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Y</td>
<td>4</td>
<td>11.0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>3</td>
<td>13.0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>s</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>3</td>
<td>11.0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Y</td>
<td>4</td>
<td>12.0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Z</td>
<td>6</td>
<td>14.0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>W</td>
<td>9</td>
<td>14.0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>t</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.0</td>
<td>11.0</td>
<td>1</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>12.0</td>
<td>12.0</td>
<td>2</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>11.0</td>
<td>14.0</td>
<td>4</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>11.0</td>
<td>9.0</td>
<td>7</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>13.0</td>
<td>13.0</td>
<td>9</td>
<td>R</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>V</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A E K</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>A E K</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>C G</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>C E G K</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>E K</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>G K</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>real</th>
<th>V</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.0</td>
<td>J</td>
<td></td>
</tr>
<tr>
<td>11.0</td>
<td>D H I J</td>
<td></td>
</tr>
<tr>
<td>12.0</td>
<td>D H I J</td>
<td></td>
</tr>
<tr>
<td>13.0</td>
<td>D I J</td>
<td></td>
</tr>
<tr>
<td>14.0</td>
<td>H J</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>V</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>B F L</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>B F L</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>F L</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>F L</td>
<td></td>
</tr>
</tbody>
</table>
Insight: If all values of attribute $A$ can be found in values of $B$ ($A \subseteq B$), then by construction $B$ will be present in all lines of the binary relation containing $A$.

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

<table>
<thead>
<tr>
<th>int</th>
<th>real</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V$</td>
<td>$V$</td>
<td>$V$</td>
</tr>
<tr>
<td>1</td>
<td>A E K</td>
<td>R</td>
</tr>
<tr>
<td>2</td>
<td>A E K</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>C G</td>
<td>B F L</td>
</tr>
<tr>
<td>4</td>
<td>C E G K</td>
<td>Y</td>
</tr>
<tr>
<td>6</td>
<td>G</td>
<td>Z</td>
</tr>
<tr>
<td>7</td>
<td>E K</td>
<td>F L</td>
</tr>
<tr>
<td>9</td>
<td>G K</td>
<td>W</td>
</tr>
</tbody>
</table>
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 \( \text{rhs}(A) = \mathbb{U} \);
2: for all \( v \in \mathbb{V} \) do
3: for all \( A \) s.t. \( (v, A) \in \mathbb{B} \) do
4: \( \text{rhs}(A) = \text{rhs}(A) \cap \{B \mid (v, B) \in \mathbb{B}\} \);
5: for all \( A \in \mathbb{U} \) do
6: for all \( B \in \text{rhs}(A) \) 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
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

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 dependent value that is not in referenced attribute
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
  - \( A \subseteq C \)
  - \( A \subseteq D \)
  - \( B \subseteq C \)
  - \( B \subseteq D \)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td></td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
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
\[
\begin{array}{ccc}
1 \\
3 \\
4 \\
\end{array}
\]

B
\[
\begin{array}{ccc}
1 \\
2 \\
3 \\
\end{array}
\]

C
\[
\begin{array}{ccc}
1 \\
2 \\
3 \\
5 \\
\end{array}
\]
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:

- 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.

<table>
<thead>
<tr>
<th>dep1</th>
<th>dep2</th>
<th>ref1</th>
<th>ref2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>
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.

```
Min-Heap
  1  3  4
  1  2  3  5
  2  3  7  8
  2  3  5
  dep1  ...  ref1, ref2
  ref1
  ref2
  dep2  ...  ref1, ref2
```
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

<table>
<thead>
<tr>
<th></th>
<th>attributes to process</th>
<th>dep A refs</th>
<th>dep B refs</th>
<th>dep C refs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Init</strong></td>
<td></td>
<td>B,C</td>
<td>A,C</td>
<td>A,B</td>
</tr>
<tr>
<td><strong>Step 1</strong></td>
<td>A,C</td>
<td>C</td>
<td>A,C</td>
<td>A</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>A,B,C</td>
<td>C</td>
<td>A,C</td>
<td>A</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>A</td>
<td>Ø</td>
<td>A,C</td>
<td>A</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>A,B,C</td>
<td>Ø</td>
<td>A,C</td>
<td>A</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>C</td>
<td>Ø</td>
<td>A,C</td>
<td>Ø</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>attributes</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>z</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Felix Naumann | Profiling & Cleansing | Summer 2013
### SPIDER results

<table>
<thead>
<tr>
<th></th>
<th>UniProt</th>
<th>TPC-H</th>
<th>PDB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DB size</strong></td>
<td>900 MB</td>
<td>1.3 GB</td>
<td>2.8 GB</td>
</tr>
<tr>
<td><strong># attributes</strong></td>
<td>68</td>
<td>61</td>
<td>1215</td>
</tr>
<tr>
<td><strong># IND cand.</strong></td>
<td>910</td>
<td>477</td>
<td>139,807</td>
</tr>
<tr>
<td><strong># INDs</strong></td>
<td>36</td>
<td>33</td>
<td>4,972</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>join</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPIDER</td>
<td>1m51s</td>
<td>6m25s</td>
<td>23m36s</td>
</tr>
<tr>
<td>Bell &amp; Brockhausen</td>
<td>4m39s</td>
<td>-</td>
<td>1h32m</td>
</tr>
<tr>
<td>Marchi et al.</td>
<td>9h 58m</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Brute force</td>
<td>2m11s</td>
<td>6m30s</td>
<td>3h29m</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

According to the table, SPIDER shows significantly better performance in terms of time compared to other methods. For example, for the UniProt database, SPIDER's join time is 1m51s, whereas the Brute force method takes 2m11s. Similarly, for the PDB, SPIDER's join time is 6h07m, whereas the Brute force method takes 19h51m.
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 $A$.

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.

■ 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.
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 \textit{name}(A) and \textit{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 \textit{name}(A) ends with a substring that indicates a foreign key.
  - Currently only “id”, “key”, and “nr” (German for “number”)

Examples:
- \texttt{FILMTEXTE.FILMTEXTTYPNR} \rightarrow \texttt{FILMTEXTTYPEN.FILMTEXTTYPNR}
- \texttt{CUSTOMER.C_NATIONKEY} \rightarrow \texttt{NATION.N_NATIONKEY}
- \texttt{FILMTEXTE.FILMTEXTTYPNR} \rightarrow \texttt{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

<table>
<thead>
<tr>
<th>Test database</th>
<th>Naive Bayes</th>
<th>SVM</th>
<th>J48</th>
<th>DecisionTab</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>UniProt</td>
<td>0.86</td>
<td>0.92</td>
<td>0.84</td>
<td>0.8</td>
<td>0.855</td>
</tr>
<tr>
<td>Filmdienst</td>
<td>0.80</td>
<td>0.86</td>
<td>0.86</td>
<td>0.93</td>
<td>0.817</td>
</tr>
<tr>
<td>Movielens</td>
<td>0.71</td>
<td>0.71</td>
<td>1.0</td>
<td>0.8</td>
<td>0.805</td>
</tr>
<tr>
<td>SCOP</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>TPC-H</td>
<td>0.86</td>
<td>0.90</td>
<td>0.95</td>
<td>0.95</td>
<td>0.915</td>
</tr>
<tr>
<td>Average</td>
<td>0.846</td>
<td>0.78</td>
<td>0.930</td>
<td>0.896</td>
<td></td>
</tr>
</tbody>
</table>

- Results for MSD, trained on all others

<table>
<thead>
<tr>
<th>Test database</th>
<th>Naive Bayes</th>
<th>J48</th>
<th>DecisionTab</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSD</td>
<td>0.84</td>
<td>0.78</td>
<td>0.79</td>
</tr>
</tbody>
</table>
Summary

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