Advanced IND detection methods

Sebastian Kruse
sebastian.kruse@hpi.de
G-3.1.13 (2.5.13)
Agenda

- High arity techniques to deal with high-arity INDs
- BINDER: divide & conquer based IND detection
- SINDY: scaling out IND detection

**Chart 2**

**Advanced IND detection methods**
Sebastian Kruse, 26th June, 2017
Unary IND discovery has a complexity of $O(n^2)$ ($n$: number of attributes)

- Databases often comprise thousands of columns
  → millions of IND candidates to be checked

Checking an IND candidate requires “aligning” the values of the involved columns

- Databases often comprise millions or billions of tuples
  → huge amounts of data need to be re-organized

Call for efficient, robust, and scalable IND discovery strategies.
BINDER – divide & conquer based IND detection

DeMarchi’s algorithm (Repetition)

- **attributes**
- **dataflow**
- **values**
- **ignored**

**Chart 4**

Needs to fit into main memory!

All intersections are executed, but not all are necessary!

Advanced IND detection methods
Sebastian Kruse,
26th June, 2017
BINDER – divide & conquer based IND detection

SPIDER algorithm (Repetition)

Attributes are accessed individually!

Two Phase Multiway Merge Sort needed for larger datasets!

Attributes are accessed individually!

OS limit on open files / sockets!

Rel. 1

Rel. 2

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017

Chart 5
BINDER – divide & conquer based IND detection

BINDER algorithm – workflow (unary)

- **attributes**
- **values**
- **dataflow**
- **ignored**

**Divide**

**Conquer**

**Validation?**

**Dynamic Memory Handling:** Spill largest buckets to disk if memory is exhausted.

**Lazy Partition Refinement:** Split a partition if it does not fit into main memory.

**Advanced IND detection methods**
Sebastian Kruse, 26th June, 2017
BINDER – divide & conquer based IND detection

BINDER algorithm – validation

1. Iterate attributes
2. Iterate values
3. If value2attr entry exists
   - Intersect candidates with this list
   - Remove value2attr entry
   - If attribute removed from all candidates
     - Remove entry from attr2value

Both indexes fit into main memory due to the partitioning!

see DeMarchi’s algorithm

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017

Chart 7
**BINDER – divide & conquer based IND detection**

**BINDER algorithm – validation example**

<table>
<thead>
<tr>
<th>attr2value</th>
<th>value2attr</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>a b c e f</td>
</tr>
<tr>
<td>C</td>
<td>b c e</td>
</tr>
<tr>
<td>D</td>
<td>a e</td>
</tr>
<tr>
<td></td>
<td>c d</td>
</tr>
</tbody>
</table>

| a b c e f   | A C         |
| b c e      | A B         |
| a e        | A B D       |
| c d        | A B C       |
|            | A           |

Never tested! → f

1. Iterate attributes
2. Iterate values
3. If value2attr entry exists
   - Intersect candidates with this list
   - Remove value2attr entry
   - If attribute removed from all candidates
     - Remove entry from attr2value

**Advanced IND detection methods**
Sebastian Kruse, 26th June, 2017

Chart 8
BINDER – divide & conquer based IND detection

BINDER evaluation

Longest run:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Time (sec)</th>
<th>Time (min)</th>
<th>Time (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMA</td>
<td>0.6</td>
<td>1.2</td>
<td>0.1</td>
</tr>
<tr>
<td>SCOP</td>
<td>5.2</td>
<td>1.1</td>
<td>0.16</td>
</tr>
<tr>
<td>LOD</td>
<td>28</td>
<td>1.8</td>
<td>0.31</td>
</tr>
<tr>
<td>CENSUS</td>
<td>66</td>
<td>4.6</td>
<td>0.77</td>
</tr>
<tr>
<td>BIOSQL</td>
<td>1.2</td>
<td>18</td>
<td>3.00</td>
</tr>
<tr>
<td>WIKIPEDIA</td>
<td>1.1</td>
<td>21</td>
<td>3.51</td>
</tr>
<tr>
<td>WIKIRANK</td>
<td>1.8</td>
<td>18</td>
<td>3.00</td>
</tr>
<tr>
<td>ENSEMBLE</td>
<td>4.6</td>
<td>18</td>
<td>3.00</td>
</tr>
<tr>
<td>CATH</td>
<td>21</td>
<td>18</td>
<td>3.00</td>
</tr>
<tr>
<td>TESMA</td>
<td>5.9</td>
<td>14.2</td>
<td>2.37</td>
</tr>
<tr>
<td>PDB</td>
<td>7.5</td>
<td>14.2</td>
<td>2.37</td>
</tr>
<tr>
<td>TPC_H_70</td>
<td>14.2</td>
<td>14.2</td>
<td>2.37</td>
</tr>
<tr>
<td>PLISTA</td>
<td>14.2</td>
<td>14.2</td>
<td>2.37</td>
</tr>
</tbody>
</table>

128 GB RAM

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017

Chart 9
BINDERS – divide & conquer based IND detection

N-ary IND detection complexity

No n-ary INDs here! Why?

\[ X \subseteq Y : \quad X \cap Y = \emptyset \]

IND Candidates in level k:

\[
\begin{align*}
X & \sim \binom{n}{k} \\
Y & \sim \binom{n-k}{k} \times k!
\end{align*}
\]

nodes

other, non-overlapping nodes

all permutations

Test combination with all other combinations of same size!

Advanced IND detection methods

Sebastian Kruse,
26th June, 2017

Chart 10
BINDER – divide & conquer based IND detection

MIND (Recap)

**Preprocessor**

**Dataset**

**Data**

**MIND**

**Preprocessor** → **MIND** → **Validator** → **Candidate Generator** → **Results**

**Validator**

**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;
```

**Candidate Generator**

**Preprocessor**

**Validator**

**Candidate Generator** → **Validator** → **Preprocessor**

**Preprocessor**

**Dataset**

**Data**

**MIND**

**Preprocessor** → **MIND** → **Validator** → **Candidate Generator** → **Results**

**Validator**

**Validate** A, B ⊆ C, D:

```
SELECT fromTable.AAGE, fromTable.ACLSWKR, fromTable.ADTIND, fromTable.ADTOCC, fromTable.AGI, fromTable.AHGA
FROM CENSUS6 fromTable LEFT OUTER JOIN CENSUS exceptTable ON fromTable.AAGE = exceptTable.AAGE AND fromTable.ACLSWKR = exceptTable.ACLSWKR AND fromTable.ADTIND = exceptTable.ADTIND AND fromTable.ADTOCC = exceptTable.ADTOCC AND fromTable.AGI = exceptTable.AGI AND fromTable.AHGA = exceptTable.AHGA
WHERE exceptTable.AAGE IS NULL AND exceptTable.ACLSWKR IS NULL AND exceptTable.ADTIND IS NULL AND exceptTable.ADTOCC IS NULL AND exceptTable.AGI IS NULL AND exceptTable.AHGA IS NULL
AND (fromTable.AAGE IS NOT NULL OR fromTable.ACLSWKR IS NOT NULL OR fromTable.ADTIND IS NOT NULL OR fromTable.ADTOCC IS NOT NULL OR fromTable.AGI IS NOT NULL OR fromTable.AHGA IS NOT NULL)
FETCH FIRST 1 ROWS ONLY;
```
BINDER – divide & conquer based IND detection

N-ary BINDER – workflow

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017

Chart 12
Apriori algorithm:
- Bottom-up lattice traversal strategy
- Authors: R. Agrawal and R. Srikant
- Publication: Fast algorithms for mining association rules in large databases
- Input: all frequent item sets of size n
- Output: all candidate frequent item sets of size n+1

Adaption for n-ary IND detection:
- Let $R_i$ be the $i$-th relation in the relational schemata $R$. For each valid IND $R_j[X] \subseteq R_k[Y]$ with $|X| = |Y| = n$ generate all IND candidates $R_j[XA] \subseteq R_k[YB]$ so that:
  1. $R_j[X] \subseteq R_k[Y]$ and $R_j[A] \subseteq R_k[B]$ (both are valid INDs)
  2. $\forall X_i \in X: X_i < A$ (INDs are permutable; do not generate them twice)
  3. $A \not\subseteq XY$, $B \not\subseteq XY$ and $R_j[A] \neq \emptyset$ (do not generate degenerate candidates)
Assume that we need to check \( AB \subseteq FE \) and \( AB \subseteq FG \).

BINDER – divide & conquer based IND detection

BINDER algorithm – workflow (n-ary)

**Divide**

**Conquer**

---

**Advanced IND detection methods**

Sebastian Kruse, 26th June, 2017

Chart 14
BINDER – divide & conquer based IND detection

N-ary BINDER evaluation

Termination (7 days)

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017

Chart 15
BINDER – divide & conquer based IND detection

N-ary BINDER evaluation

More distinct values per attribute combination.

Potentially very many attribute combinations (cf. complexity).

Value combinations take up more space than single values.

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017

Chart 16
Agenda

**SINDY**
scaling out
IND detection

**BINDER**
divide & conquer
based IND detection

**High arity**
techniques to deal
with high-arity INDs

---

**Advanced IND detection methods**
Sebastian Kruse,
26th June, 2017
Some problems are intrinsically hard

- “defeat it with iron”: use more/better hardware for the computation

Scalability $\neq$ efficiency

- Efficient = fast / spare resources
- Scalable = improvement by leveraging more resources

“Do the same work in less time.” $\neq$ “Do more work in the same time.”
Scaling dimensions

- **Scale up**: faster CPUs/disk, more main memory, …
  - lowest impact on code
  - expensive, limited, shift of bottlenecks

- **Scale in**: more cores, (RAID)
  - thread-level parallelization, cache coherency
  - limited, shift of bottlenecks

- **Scale out**: computer clusters
  - Actors, message passing, data partition
  - Less limited, most complicated

- Is **problem suited** to certain scaling direction?
Multiple independent nodes

- can communicate and exchange data

- oftentimes data distributed among nodes

- no shared state

- network new potential bottleneck
  - network topology relevant

- fault tolerance important

- load balancing important
Distributed Application Frameworks

- **Script languages**
  - JAQL
  - Meteor
  - Pig
  - Hive

- **Operator-centric frameworks**
  - Flink/Stratosphere
  - Spark
  - Storm

- **Low-level frameworks/resource management**
  - Hadoop
  - YARN
  - Mesos
  - Akka
  - Zookeeper

- **Storage**
  - HDFS
  - HBase
  - S3
  - Cassandra

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017

NB: overview not complete!
SINDY – scaling out IND detection

Writing an Apache Flink program

- Four ingredients
  - Datasets
  - First-order functions
    - high-level data operations
  - Second-order functions
    - one passed to each first-order functions
    - refine semantics of first-order functions
    - transform data
  - Directed acyclic graph
    - starts with data sources, ends in data sinks
    - describes workflow
SINDY – scaling out IND detection
Classic example: word count

- implemented as DAG with two first-order functions (and two second-order functions)
- specifies
  - operations on a logical level
- does not specify
  - how to parallelize
  - data serialization and shipping
  - handling when available main memory is exceeded
  - fault tolerance
  - ...

---

Chart 23

input file

Map

Reduce

output

split line into words
for each word, output (word, 1)

group by word

for each input pair (word, n1) and (word, n2), output (word, n1+n2)
SINDY – scaling out IND detection

Classic example: word count

“big data is big”

(input file)

Map

Reduce

output

split line into words
for each word, output (word, 1)

group by word

for each input pair (word, n1) and (word, n2), output (word, n1+n2)
SINDY – scaling out IND detection

Classic example: word count

```
(input file)

Map

Reduce

output

split line into words
for each word, output (word, 1)

group by word

for each input pair (word, n1) and (word, n2), output (word, n1+n2)
```

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017

Chart 25
public class CountWords() {
    public static void main(String[] args) {
        ExecutionEnvironment env =
            ExectionEnvironment.getExectutionEnvironment();
        env.readTextFile(args[0])
            .flatMap((String line, Collector<WordCount> out) -> {
                Arrays.stream(line.split("\W+"))
                    .forEach(t -> out.collect(new WordCount(t, 1)))
            })
            .groupBy("word")
            .sum("count")
            .print();
        env.execute();
    }
}
### Intrinsinc limitations of IND algorithms

- Observations: all IND algorithms follow a common pattern

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Phase 1: Data Reorganization</th>
<th>Phase 2: Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>De Marchi</td>
<td>Create Inverted Index</td>
<td>Intersect Attribute Groups</td>
</tr>
<tr>
<td>SPIDER</td>
<td>Sort Columns</td>
<td>Simultaneous Iteration</td>
</tr>
<tr>
<td>BINDER</td>
<td>Partition Columns</td>
<td>In-Memory Partition Comparison</td>
</tr>
</tbody>
</table>

- e.g., IND $A \subseteq B$
  - to prove, need to read A completely
  - to disprove, need to read B completely
- Data reorganization is the most expensive phase
  - I/O-heavy workload, but other phase brings considerable I/O as well
SINDY – scaling out IND detection

General IND approach

original database

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>g</td>
<td>d</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
<td>e</td>
<td>b</td>
</tr>
<tr>
<td>a</td>
<td>c</td>
<td>g</td>
<td>b</td>
</tr>
<tr>
<td>j</td>
<td>j</td>
<td>g</td>
<td>g</td>
</tr>
<tr>
<td>j</td>
<td>i</td>
<td>c</td>
<td>f</td>
</tr>
<tr>
<td>e</td>
<td>f</td>
<td>g</td>
<td>f</td>
</tr>
<tr>
<td>h</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>i</td>
<td>a</td>
<td>i</td>
<td>a</td>
</tr>
</tbody>
</table>

full outer join (w/o duplicates)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>-</td>
<td>a</td>
<td>-</td>
<td>-</td>
<td>a</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>-</td>
<td>b</td>
<td>-</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td>-</td>
<td>-</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>d</td>
<td>e</td>
<td>-</td>
<td>-</td>
<td>d</td>
</tr>
<tr>
<td>e</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>-</td>
<td>-</td>
<td>g</td>
<td>f</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>g</td>
<td>g</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>h</td>
<td>-</td>
<td>-</td>
<td>h</td>
<td>-</td>
<td>h</td>
<td>-</td>
</tr>
<tr>
<td>i</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>i</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>j</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>j</td>
<td>j</td>
</tr>
</tbody>
</table>

attribute groups

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>F</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>G</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>E</td>
<td>G</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>F</td>
</tr>
<tr>
<td>A</td>
<td>F</td>
<td>G</td>
</tr>
</tbody>
</table>

Advanced IND detection methods

Sebastian Kruse, 26th June, 2017

Chart 28
SINDY – scaling out IND detection

Distributed IND detection: general idea

1. Calculate full outer join
2. Intersect attribute groups
SINDY – scaling out IND detection

SINDY: Calculate outer join

Dataset

Cells

Local pre-join (cache-based)

Global join

Attribute Groups

(a, \{A\}) \rightarrow (a, \{A\})
(b, \{B\}) \rightarrow (b, \{B\})
(h, \{F\}) \rightarrow (h, \{F\})
(c, \{A\}) \rightarrow (c, \{A\})
(b, \{B\}) \rightarrow (b, \{B\})
(i, \{F\}) \rightarrow (i, \{F\})
(a, \{A\}) \rightarrow (a, \{A\})
(c, \{B\}) \rightarrow (c, \{B\})

\ldots \ldots \ldots

(a, \{A,F\}) \rightarrow \{A,F\}
(b, \{B,F\}) \rightarrow \{B,F\}
(c, \{A,B,F\}) \rightarrow \{A,B,F\}
(e, \{A\}) \rightarrow \{A\}
(g, \{B\}) \rightarrow \{B\}
(i, \{A,B,F\}) \rightarrow \{A,B,F\}

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017

Chart 30
SINDY – scaling out IND detection

Determine INDs from join result

<table>
<thead>
<tr>
<th>Attribute Groups</th>
<th>IND candidates</th>
<th>Pre-consolidation (cache-based)</th>
<th>Consolidation</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A,F}</td>
<td>A ⊆ {F}</td>
<td></td>
<td>A ⊆ {}</td>
<td>A is not included in any other attribute</td>
</tr>
<tr>
<td></td>
<td>F ⊆ {A}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{A,B,F}</td>
<td>A ⊆ {B,F}</td>
<td></td>
<td>B ⊆ {}</td>
<td>B is not included in any other attribute</td>
</tr>
<tr>
<td></td>
<td>B ⊆ {A,F}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>F ⊆ {A,B}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{A}</td>
<td>A ⊆ {}</td>
<td></td>
<td>A ⊆ {}</td>
<td>A is not included in any other attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{B}</td>
<td>A ⊆ {}</td>
<td></td>
<td>F ⊆ {A}</td>
<td>F is only included in A</td>
</tr>
<tr>
<td></td>
<td>B ⊆ {}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>F ⊆ {A}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{A,B,F}</td>
<td>A ⊆ {B,F}</td>
<td></td>
<td>B ⊆ {}</td>
<td>B is not included in any other attribute</td>
</tr>
<tr>
<td></td>
<td>B ⊆ {A,F}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>F ⊆ {A,B}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{A,F}</td>
<td>A ⊆ {}</td>
<td></td>
<td>A ⊆ {}</td>
<td>A is not included in any other attribute</td>
</tr>
<tr>
<td></td>
<td>F ⊆ {A}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{A,F}</td>
<td></td>
<td></td>
<td>F ⊆ {A}</td>
<td>F is only included in A</td>
</tr>
<tr>
<td>{A}</td>
<td>A ⊆ {}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{A,F}</td>
<td></td>
<td></td>
<td>F ⊆ {A}</td>
<td>F is only included in A</td>
</tr>
</tbody>
</table>

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017
SINDY – scaling out IND detection

Implementation on Flink (unary INDs)

input files → Map → Combine → Reduce

Map: split input tuple into a set of cells (value, {attribute})
Combine: group by value
Reduce: union attributes

Map: split attributes into IND candidates
Combine: group by dependent attribute
Reduce: intersect referenced attributes

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017
Chart 32
SINDY – scaling out IND detection
Implementation on Flink (n-ary INDs)

input files → Map → Combine → Reduce → output

- **Map**: split input tuple into a set of cells (value combination, {attribute combination})
- **Combine**: group by value
- **Reduce**: union attributes
- **output**: Filter pseudo INDs!

- **Map**: split attributes into IND candidates
- **Combine**: group by dependent attribute
- **Reduce**: intersect referenced attributes

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017
Chart 33
SINDY – scaling out IND detection

Scale-out behavior

runtime [s] vs. degree of parallelism

- MB-core
- TPC-H
- LOD
- BIOSQLSP
- WIKIPEDIA
- CATH
- CENSUS
- SCOP
- COMA

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017
Chart 34
SINDY – scaling out IND detection

Performance comparison with SPIDER

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017

Chart 35
Problem of IND detection can be scaled out with the Map/Reduce paradigm

- Comes with a certain loss of control

- SINDY does not employ pruning (except for apriori proceeding)
  - A general problem for distributed algorithms
  - Not a big issue for IND detection

- Only suitable for large datasets

- Arising questions
  - To what extent is attribute scaling possible? → MANY
  - What if some INDs are n-ary for some larger n?
Agenda

**BINDER**
divide & conquer
based IND detection

**SINDY**
scaling out
IND detection

**High arity**
techniques to deal with high-arity INDs

**Advanced IND detection methods**
Sebastian Kruse, 26th June, 2017

<table>
<thead>
<tr>
<th>Name</th>
<th>Mass</th>
<th>Orbital radius</th>
<th>Rotation period</th>
<th>Atmosphere</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mercury</td>
<td>0.06</td>
<td>0.47</td>
<td>58.64</td>
<td>minimal</td>
</tr>
<tr>
<td>Venus</td>
<td>0.82</td>
<td>0.72</td>
<td>−243.02</td>
<td>CO₂, N₂</td>
</tr>
<tr>
<td>Earth</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>N₂, O₂, Ar</td>
</tr>
<tr>
<td>Mars</td>
<td>0.11</td>
<td>1.52</td>
<td>1.03</td>
<td>CO₂, N₂, Ar</td>
</tr>
<tr>
<td>Jupiter</td>
<td>317.8</td>
<td>5.20</td>
<td>0.41</td>
<td>H₂, He</td>
</tr>
<tr>
<td>Saturn</td>
<td>95.2</td>
<td>9.54</td>
<td>0.43</td>
<td>H₂, He</td>
</tr>
<tr>
<td>Uranus</td>
<td>14.6</td>
<td>19.22</td>
<td>−0.72</td>
<td>H₂, He</td>
</tr>
<tr>
<td>Neptune</td>
<td>17.2</td>
<td>30.06</td>
<td>0.67</td>
<td>H₂, He</td>
</tr>
</tbody>
</table>
High-arity INDs do not go well together with apriori-based algorithms

- Consider an IND of arity $n$
- Then there are $2^n - 2$ sub-INDs to be verified
- No pruning possible
- Recall the hardness of $n$-ary IND discovery

Different approaches necessary

- Cf. TANE and HyFD
Most (maximal) INDs are of low arity, but we do find high-arity INDs when...

- There are table duplicates – ORDER_ITEMS vs. ORDER_ITEMS_TEMP
- The dataset contains materialized views – `SELECT * FROM STUDENTS WHERE SINCE_YEAR = 2015;`
- Datasets contain many similar columns – `PLACES (ID in [0 .. 60], VERSION = "1.0", DESCRIPTION = "", …)` – `TRACKS (ID in [0 .. 10000], VERSION = "1.0", DESCRIPTION = "", …)`
- Because they reflect actual foreign keys – SAP S4 schema contains keys with >10 columns
- Functional dependencies

**Motivation**

Sebastian Kruse, 26th June, 2017

Advanced IND detection methods

Chart 39
High-arity INDs - ZigZag

A different notation for n-ary INDs

\[ R[A, B, C] \subseteq S[A', B', C'] \]
High-arity INDs - ZigZag

A different notation for n-ary INDs

\[
\text{ens} \left( R[A, B, C] \subseteq S[A', B', C'] \right) = \\
\{ R[A] \subseteq S[A'], R[B] \subseteq S[B'], R[C] \subseteq S[C'] \}
\]
High-arity INDs - ZigZag

A different notation for n-ary INDs

\[ \{ R[A] \subseteq S[A'], R[B] \subseteq S[B'], R[C] \subseteq S[C'] \} \]

\[ \{ R[A] \subseteq S[A'], R[B] \subseteq S[B'] \} \quad \{ R[B] \subseteq S[B'], R[C] \subseteq S[C'] \} \]

\[ \{ R[A] \subseteq S[A'], R[C] \subseteq S[C'] \} \]

\[ \{ R[A] \subseteq S[A'] \} \quad \{ R[B] \subseteq S[B'] \} \quad \{ R[C] \subseteq S[C'] \} \]

Has \(2^n-2\) “sub-candidates” that all need to be tested.
High-arity INDs - ZigZag

Optimistic and pessimistic strategies

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017

Chart 43
High-arity INDs - ZigZag

IND borders

5-ary IND candidates
4-ary IND candidates
Ternary IND candidates
Binary IND candidates
Unary IND candidates

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017

Chart 44
High-arity INDs - ZigZag

IND borders

5-ary IND candidates

4-ary IND candidates

ternary IND candidates

binary IND candidates

unary IND candidates

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017

Chart 45
High-arity INDs - ZigZag

IND borders

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017

Chart 46
High-arity INDs - ZigZag

IND borders

Optimistic skip

5-ary IND candidates

4-ary IND candidates

ternary IND candidates

binary IND candidates

unary IND candidates

IND • Maximal IND • IND candidate

Non-IND • Minimal non-IND

Advanced IND detection methods
Sebastian Kruse,
26th June, 2017

Chart 47
Given two tables, and a set of known INDs and/or non-INDs, how can we determine the optimistic IND border?

- All IND candidates that are (i) not known non-INDs and (ii) are maximal w.r.t. property (i).

**FIND**

- Determine hypergraph cliques based on INDs

**ZigZag**

- Determine hitting sets based on non-INDs
Example:
- Unary INDs = \{A, B, C, D, E\}
- INDs = \{AB, AC, AE, BC, BD, BE, CE\}
- non-INDs = \{AD, CD, DE\}

Goal:
- find *all maximal* sets \(\subseteq\) ABCDE that are *no supersets* of AD, CD, or DE

General strategy
- Determine *minimal sets* that *intersect* with *all* non-INDs (ACE, D)
- Remove these minimal sets from ABCDE (BD, ABCE)
Input: non-INDs N

N=[AD, CD, DE]
High-arity INDs - ZigZag

**Calculate the optimistic IND border (adv.)**

- **Input:** non-INDs N, unary INDs U
  
  \[ N = \{ \text{AD, CD, DE} \}, \ U = \{ \text{A, B, C, D, E} \} \]
Current state

\[ N = [CD, DE], \quad S = \]

\[ (2) \text{ remove head } H \text{ of } N \]

\[ H = CD, \quad N = [DE] \]

\[ (3) \text{ invert } H \text{ wrt } \]

\[ U = AB \]

\[ (4) \text{ remove subsets of } I \text{ from } S \]

\[ S = \quad , \quad L = \{A\} \]

\[ (5) \text{ for each removed solution in } L, \quad L = \{AC, AD\} \]

\[ (6) \text{ combine it with all unary INDs in } H \]

\[ \text{update } S \text{ with all elements in } S \text{ except a subset is already in } S \]

High-arity INDs - ZigZag

Calculate the optimistic IND border (adv.)
Current state

N=[DE], S=

(2) remove head H of N
H=DE, N=[]

(3) invert H wrt. U
I=ABC

(4) remove subsets of I from S
S=

(5) for each removed solution in L,
L={ACD, ACE}

(6) combine it with all unary INDs in H

(7) update S with all elements in S except a subset is already in S

(8) invert: border = {BD, ABCE}

High-arity INDs - ZigZag
Calculate the optimistic IND border (adv.)

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017
Chart 53
For the attentive student:

(6) update $S$ with all elements in $L$ except a subset is already in $S$

- What if an existing solution is a superset of a new solution?

This is not possible (inductive proof):

- Assume, we introduced a new element $N$ that is a subset of some existing element $E$ in $S$
- Then we would have obtained $N$ from some $N' = N \setminus I$ for some unary IND in $U$.
- Hence, $S$ must have been already in an inconsistent state.
- Initially, $S = \{\varnothing\}$, which is a consistent state.
- What about two elements being added in a single iteration, though?
  → Figure out yourselves.
High-arity INDs - ZigZag

Non-INDs in the optimistic IND border

Optimistic skip: No IND, though

- 5-ary IND candidates
- 4-ary IND candidates
- ternary IND candidates
- binary IND candidates
- unary IND candidates

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017

Chart 55
High-arity INDs - ZigZag Non-INDs in the optimistic IND border

Optimistic skip: No IND, though

20-ary IND candidates

for promising IND candidates

for unpromising IND candidates

unary IND candidates

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017
Chart 56
Promising IND candidate $R[X] \subseteq S[Y]$ if $g'_3(R[X] \subseteq S[Y]) \leq \varepsilon$

$g'_3(R[X] \subseteq S[Y])$: proportion of distinct values in $R[X]$ to be removed, such that $R[X] \subseteq S[Y]$ is a valid IND

Alternative: proportion of tuples to be removed from $R$

Example: $g'_3(R[\text{Planet}] \subseteq S[\text{Planet}]) = 1 / 10 = 0.1$

The value “Ceres” has to be removed out of 10 distinct values

<table>
<thead>
<tr>
<th>Planet</th>
<th>Mean distance</th>
<th>Relative mean distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mercury</td>
<td>57.91</td>
<td>1</td>
</tr>
<tr>
<td>Venus</td>
<td>108.21</td>
<td>1.86859</td>
</tr>
<tr>
<td>Earth</td>
<td>149.6</td>
<td>1.3825</td>
</tr>
<tr>
<td>Mars</td>
<td>227.92</td>
<td>1.52353</td>
</tr>
<tr>
<td>Ceres</td>
<td>413.79</td>
<td>1.81552</td>
</tr>
<tr>
<td>Jupiter</td>
<td>778.57</td>
<td>1.88154</td>
</tr>
<tr>
<td>Saturn</td>
<td>1,433.53</td>
<td>1.84123</td>
</tr>
<tr>
<td>Uranus</td>
<td>2,872.46</td>
<td>2.00372</td>
</tr>
<tr>
<td>Neptune</td>
<td>4,495.06</td>
<td>1.56488</td>
</tr>
<tr>
<td>Pluto</td>
<td>5,869.66</td>
<td>1.3058</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Planet</th>
<th>Calculated (in AU)</th>
<th>Observed (in AU)</th>
<th>Perfect octaves</th>
<th>Actual distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mercury</td>
<td>0.4</td>
<td>0.387</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Venus</td>
<td>0.7</td>
<td>0.723</td>
<td>1</td>
<td>1.1</td>
</tr>
<tr>
<td>Earth</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Mars</td>
<td>1.6</td>
<td>1.524</td>
<td>4</td>
<td>3.7</td>
</tr>
<tr>
<td>Ceres</td>
<td>4.1</td>
<td>4.1</td>
<td>8</td>
<td>7.8</td>
</tr>
<tr>
<td>Jupiter</td>
<td>5.2</td>
<td>5.203</td>
<td>16</td>
<td>15.7</td>
</tr>
<tr>
<td>Saturn</td>
<td>10</td>
<td>9.539</td>
<td>32</td>
<td>29.9</td>
</tr>
<tr>
<td>Uranus</td>
<td>19.6</td>
<td>19.191</td>
<td>64</td>
<td>61.4</td>
</tr>
<tr>
<td>Neptune</td>
<td>38.8</td>
<td>30.061</td>
<td>96</td>
<td>-96.8</td>
</tr>
<tr>
<td>Pluto</td>
<td>77.2</td>
<td>39.529</td>
<td>128</td>
<td>127.7</td>
</tr>
</tbody>
</table>
1. **Input:** tables $R$ and $S$, pessimistic levels $k$, promising error $\varepsilon$
2. Calculate lower $k$ levels with a pessimistic approach (e.g., BINDER)
3. Calculate optimistic IND border from non-INDs
4. **For each** IND candidate $I$ in the optimistic IND border
   1. Calculate error $g'_3(I)$ of $I$
   2. **If** $g'_3(I) = 0$ **then** output IND $I$
   3. **Else if** $g'_3(I) \leq \varepsilon$ **then** traverse lattice top-down breadth-first from $I$
   4. **Else** add all $k+1$-ary parent IND candidates of $I$ to the pessimistic IND candidates
5. Check all pessimistic IND candidates
6. **If** there are open IND candidates, set $k = k+1$ and start over with **step 3**
A strategy for handling more than two tables is missing

Several optimizations are possible, e.g., not all kinds of unary INDs can be combined to valid n-ary IND candidates (attribute repetition)

Empirical evidence on the actual advantages of optimistic IND discovery is missing

→ Thorough evaluation all IND algorithms is called for!

The original article on ZigZag proposes to do use SQL-based error checks for n-ary INDs (cf. MIND)

→ the traversal strategy, however, is orthogonal to IND error checks

→ more efficient techniques, such as those of BINDER and SINDY, could be used instead
Consider following dependencies: \([ZIP', City'] \subseteq [ZIP, City]\) and \(ZIP \rightarrow City\)

**Rule 1)** Then \(ZIP' \rightarrow City'\) is also a valid FD.

- Because FDs cannot be violated by removing tuples.
- Additionally, consider \([Name', ZIP'] \subseteq [Name, ZIP]\)

**Rule 2)** Then \([Name', ZIP', City'] \subseteq [Name, ZIP, City]\) is an IND.

- Because if \(t[ZIP] = t[ZIP']\), then \(t[City] = t[City']\).

<table>
<thead>
<tr>
<th>Name</th>
<th>Zip</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim</td>
<td>10627</td>
<td>Berlin</td>
</tr>
<tr>
<td>Tom</td>
<td>10627</td>
<td>Berlin</td>
</tr>
<tr>
<td>Tom</td>
<td>14482</td>
<td>Potsdam</td>
</tr>
<tr>
<td>Sandy</td>
<td>10324</td>
<td>Berlin</td>
</tr>
<tr>
<td>Inge</td>
<td>14469</td>
<td>Potsdam</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name'</th>
<th>Zip'</th>
<th>City'</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim</td>
<td>10627</td>
<td>Berlin</td>
</tr>
<tr>
<td>Tom</td>
<td>10627</td>
<td>Berlin</td>
</tr>
<tr>
<td>Inge</td>
<td>14469</td>
<td>Potsdam</td>
</tr>
</tbody>
</table>
Augmentation rules

- Idea: split INDs into “core” INDs and “augmentation rules”
  - IND: \([\text{Name}', \text{ZIP}'] \subseteq [\text{Name}, \text{ZIP}]\)
  - AR: \([\text{ZIP}'] \subseteq [\text{ZIP}] \rightarrow [\text{City}'] \subseteq [\text{City}]\)

- Separates INDs into core and supplemental INDs
  - Useful for foreign key discovery and understanding

- Potentially reduces the size of the result set
  - Speed up discovery and make results more manageable

<table>
<thead>
<tr>
<th>Name</th>
<th>Zip</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim</td>
<td>10627</td>
<td>Berlin</td>
</tr>
<tr>
<td>Tom</td>
<td>10627</td>
<td>Berlin</td>
</tr>
<tr>
<td>Tom</td>
<td>14482</td>
<td>Potsdam</td>
</tr>
<tr>
<td>Sandy</td>
<td>10324</td>
<td>Berlin</td>
</tr>
<tr>
<td>Inge</td>
<td>14469</td>
<td>Potsdam</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name'</th>
<th>Zip'</th>
<th>City'</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim</td>
<td>10627</td>
<td>Berlin</td>
</tr>
<tr>
<td>Tom</td>
<td>10627</td>
<td>Berlin</td>
</tr>
<tr>
<td>Inge</td>
<td>14469</td>
<td>Potsdam</td>
</tr>
</tbody>
</table>

Advanced IND detection methods
Sebastian Kruse, 26th June, 2017
Chart 61
Assume, we know $[ZIP'] \subseteq [ZIP]$ and $[ZIP', City'] \subseteq [ZIP, City]$

- Problem: we need to know if $ZIP \rightarrow City$ is an FD to tell if $[ZIP'] \subseteq [ZIP] \rightarrow [City'] \subseteq [City]$ is an AR.

- Solution 1: Discover FDs beforehand (e.g., with HyFD).

- Solution 2: Check relevant FD candidates on-the-fly.

- $ZIP \rightarrow City \leftrightarrow |\pi(ZIP)| = |\pi(ZIP, City)|$ (cf. TANE)

- We have to group our data anyways, so we can “piggyback” the counting of distinct values at little extra cost.

<table>
<thead>
<tr>
<th>Name</th>
<th>Zip</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim</td>
<td>10627</td>
<td>Berlin</td>
</tr>
<tr>
<td>Tom</td>
<td>10627</td>
<td>Berlin</td>
</tr>
<tr>
<td>Tom</td>
<td>14482</td>
<td>Potsdam</td>
</tr>
<tr>
<td>Sandy</td>
<td>10324</td>
<td>Berlin</td>
</tr>
<tr>
<td>Inge</td>
<td>14469</td>
<td>Potsdam</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name’</th>
<th>Zip’</th>
<th>City’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim</td>
<td>10627</td>
<td>Berlin</td>
</tr>
<tr>
<td>Tom</td>
<td>10627</td>
<td>Berlin</td>
</tr>
<tr>
<td>Inge</td>
<td>14469</td>
<td>Potsdam</td>
</tr>
</tbody>
</table>
High-arity INDs - Andy

Augmentation rule discovery

- Special case: columns/INDs with only a single value
  - \([\text{Status}'] \subseteq [\text{Status}]\) is valid
  - \(X \rightarrow \text{Status}\) is a valid AR for any column \(X\) in Students
- \(\emptyset \rightarrow [\text{Status}'] \subseteq [\text{Status}]\) is a valid AR
- This is a very frequent case
  - Empty or constant columns can be found in many databases
  - They are highly susceptible to form \(n\)-ary INDs

<table>
<thead>
<tr>
<th>Name</th>
<th>Zip</th>
<th>City</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim</td>
<td>10627</td>
<td>Berlin</td>
<td>Student</td>
</tr>
<tr>
<td>Tom</td>
<td>10627</td>
<td>Berlin</td>
<td>Student</td>
</tr>
<tr>
<td>Tom</td>
<td>14482</td>
<td>Potsdam</td>
<td>Student</td>
</tr>
<tr>
<td>Sandy</td>
<td>10324</td>
<td>Berlin</td>
<td>Student</td>
</tr>
<tr>
<td>Inge</td>
<td>14469</td>
<td>Potsdam</td>
<td>Student</td>
</tr>
</tbody>
</table>

Students

<table>
<thead>
<tr>
<th>Name’</th>
<th>Zip’</th>
<th>City’</th>
<th>Status’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim</td>
<td>10627</td>
<td>Berlin</td>
<td>Student</td>
</tr>
<tr>
<td>Tom</td>
<td>10627</td>
<td>Berlin</td>
<td>Student</td>
</tr>
<tr>
<td>Inge</td>
<td>14469</td>
<td>Potsdam</td>
<td>Student</td>
</tr>
</tbody>
</table>

HPI Students

Advanced IND detection methods
Sebastian Kruse,
26th June, 2017

Chart 63
High-arity INDs - Andy

Mixed IND and AR discovery

Initial state: unary INDs with distinct values of LHS/dep.

\[ \{\} \rightarrow S \text{ is an AR.} \]
Apply \{\} \to S.

We can prune all n-ary INDs that comprise S, because they can be inferred.
Test remaining binary IND candidates.

Z → C is another AR.
Apply $Z \rightarrow C$.

All IND candidates containing $Z$ and $C$ can be pruned.
High-arity INDs - Andy

Mixed IND and AR discovery

Final result:

- **INDs:** \([\text{Name}', \text{ZIP}'] \subseteq [\text{Name}, \text{ZIP}], [\text{Name}', \text{City}'] \subseteq [\text{Name}, \text{City}]\)
- **ARs:** \([\text{ZIP}'] \subseteq [\text{ZIP}] \rightarrow [\text{City}'] \subseteq [\text{City}], \{\} \rightarrow [\text{Status}'] \subseteq [\text{Status}]\)

Checked only 3 out of 11 (valid) IND candidates.

<table>
<thead>
<tr>
<th>Name</th>
<th>Zip</th>
<th>City</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim</td>
<td>10627</td>
<td>Berlin</td>
<td>Student</td>
</tr>
<tr>
<td>Tom</td>
<td>10627</td>
<td>Berlin</td>
<td>Student</td>
</tr>
<tr>
<td>Tom</td>
<td>14482</td>
<td>Potsdam</td>
<td>Student</td>
</tr>
<tr>
<td>Sandy</td>
<td>10324</td>
<td>Berlin</td>
<td>Student</td>
</tr>
<tr>
<td>Inge</td>
<td>14469</td>
<td>Potsdam</td>
<td>Student</td>
</tr>
</tbody>
</table>

**Students**

**HPI Students**

---

Advanced IND detection methods
Sebastian Kruse,
26th June, 2017

Chart 68
Andy uses SINDY-style candidate checking based on Flink. Both run on a single machine but ANDY uses 2 cores/4 threads.
Advanced IND detection methods

Sebastian Kruse
Thorsten Papenbrock


