On Multi-Column Foreign Key Discovery

Meihui Zhang  Marios Hadjieleftheriou  Beng Chin Ooi
Cecilia M. Procopiuc  Divesh Srivastava

VLDB: September 2010

Daisy - AAU

Presented by: Benjamin Krogh

November 11, 2010
Introduction

- Important constraint on data
- Designers frequently fail to specify foreign keys
- Most previous work focuses on inclusion dependencies
- Inclusion dependencies yields many false positives
- Multi-column foreign keys have not been considered yet
On Multi-Column Foreign Key Discovery

Meihui Zhang
National University of Singapore
mhzhang@comp.nus.edu.sg

Marios Hadjieleftheriou
Beng Chin Ooi
Cecilia M. Procopiuc
Divesh Srivastava

VLDB: September 2010 (AAU)

Introduction - Example

Figure 1: A small subset of the TPC-E schema with one multi-column and several single-column foreign keys.
Overview

1. Introduction
2. Traits of Foreign Key
3. Overall Design
4. Inclusion Dependencies
5. Randomness Test
6. Experiments
7. Conclusion
Traits of a Foreign Key

1. Should have significant cardinality
2. Should have good coverage of the primary key
3. Should not be the primary key of many other foreign keys
4. Its values should not be a subset of many primary keys
5. The average length should be similar to that of the primary key
6. The column names of foreign/primary keys should be similar
Randomness

The values of a foreign key will appear to be a random sample of the primary key.

![Diagram showing randomness](image)
Motivation for Randomness

Intuition

The logic that generates the primary key, is disconnected from the logic generating the fk.

<table>
<thead>
<tr>
<th>Cars</th>
<th>Owns</th>
<th>Persons</th>
</tr>
</thead>
<tbody>
<tr>
<td>♦ CID int</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ Model varchar</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ Color Color</td>
<td></td>
<td></td>
</tr>
<tr>
<td>♦ CID int</td>
<td></td>
<td></td>
</tr>
<tr>
<td>♦ PID int</td>
<td></td>
<td></td>
</tr>
<tr>
<td>♦ PID int</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Name varchar</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Overall Design

1. Find candidate foreign/primary key pairs based on inclusion dependencies
2. Rank the pairs such that pairs with similar distribution scores best
3. Return top X-%
Inclusion Dependencies

Approach

- Dirty data due to unenforced constraints.
- Include pairs fulfilling: $\sigma(F, P) = \frac{|F \cap P|}{|F|} \geq \theta$
- Use bottom-k sketches to estimate inclusion dependencies

<table>
<thead>
<tr>
<th>CID</th>
<th>SMB</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>INTC</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>AAPL</td>
<td></td>
</tr>
<tr>
<td>217</td>
<td>GOOG</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h(10</td>
</tr>
<tr>
<td>$h(2</td>
</tr>
<tr>
<td>$h(217</td>
</tr>
</tbody>
</table>

Bottom-1 sketch

- Algorithm B.1: Include pairs fulfilling $\sigma(F, P) = \frac{|F \cap P|}{|F|} \geq \theta$
Randomness Test - Single-Column

Wilcoxon test

A standard statistical test for randomness

1. Sort values in \( F \cup P \)
2. Assign ranks
3. Compute rank-sum of duplicate values

![Wilcoxon Test Diagram]

Figure 4: The Wilcoxon test: 1. Sort values in \( F \cup P \)
Figure 3: A column containing numeric values might falsely appear to be a random sample of the distribution. Intuitively, if the rank-sum is too small, then most values in \( F \) are contained in a prefix of \( P \); see Figure 4.

Finally, compute the sum of ranks of all values in \( F \) and rank them. Since \( F \subset P \), we estimate \( \rho_{F,P} \) by di-

\[ \rho_{F,P} = \frac{\sum_{x \in F} \text{rank}(x)}{\text{count}(F)} \]

For example, consider the multi-column key \( (\text{CA}, \text{FL}) \) and \( (\text{CA}, \text{TX}) \). An independent Wilcoxon test in either dimension is not a uniform sample due to the projection of \( (\text{CA}, \text{FL}) \) and \( (\text{CA}, \text{TX}) \). The effort is the amount of work needed to convert the set of values of the foreign key into the set of values of the primary key. If we regard each distribution as piles of dirt spread across a flat surface and then vertically sort the first set of piles into the second, the effort is the amount of work needed to vertically sort the set of values of the foreign key into the set of values of the primary key. If we regard each distribution as piles of dirt spread across a flat surface, we can use a combination sort (same as the Unix “sort -n” command). For columns containing both numeric, alphanumeric, and string values, we are sorted lexicographically. The implicit assumption is that it is the same value in both dimensions. An independent Wilcoxon test in either dimension is not a uniform sample due to the projection of \( (\text{CA}, \text{FL}) \) and \( (\text{CA}, \text{TX}) \). An independent Wilcoxon test in either dimension is not a uniform sample due to the projection of \( (\text{CA}, \text{FL}) \) and \( (\text{CA}, \text{TX}) \).
Earth Mover’s Distance

Assume two piles of dirt $A$ and $B$, then $EMD(A, B)$ is the amount of work to convert $A$ to $B$.

1. Assign a probability mass to each point, such that the sum of probabilities is 1.
2. For each point $a$ in $A$, find the distance to the nearest point in $B \setminus A$ and multiply by $a$'s mass.

---

Randomness Test - Multi-Column

**Phase 1:**

1. Assign a probability mass to each point, such that the sum of probabilities is 1.
2. For each point $a$ in $A$, find the distance to the nearest point in $B \setminus A$ and multiply by $a$'s mass.

**Diagram:**

- A grid representing points $P$ and $F$ with probabilities.
- An arrow from $P$ to $F$ indicating the distance and mass.
- A similar representation for $F'$.
Problem
Exact algorithm for EMD has cubic complexity, i.e. unfeasible for large tables.

Solution
Use quantiles to summarize values. The motivation for $n$-quantiles is to divide ordered data into $n$ essentially equal-sized data subsets.

An example 4-quantile, dividing a collection into 4 equal sized parts.
Problem

Exact algorithm for EMD has cubic complexity, i.e. unfeasible for large tables.

Solution

Use quantiles to summarize values.
Experiments - 1

![Graph showing EMD, recall, precision, and F-measure for TPC-H dataset.

- EMD: Line graph with markers.
- Recall: Dashed line graph with markers.
- Precision: Dotted line graph with markers.
- F-measure: Dashed-dotted line graph with markers.

Y-axis: Top-X% values ranging from 0 to 1.
X-axis: Top-X% values ranging from 5 to 30.

TPC-H dataset is used.

The graph illustrates the performance of various metrics with respect to the size of the original column. This is generally too large for practical purposes (e.g., the Wikipedia database has size $O(10^9)$, so the size of 1%-sketches is $O(10^7)$).
Experiments - 2

The graph shows the performance of EMD, recall, precision, and F-measure as a function of Top-X%. The X-axis represents the percentage of top-ranked results, while the Y-axis shows the values of EMD, recall, precision, and F-measure. The graph includes data for Wikipedia, with different markers and lines indicating the performance across various datasets.

For practical purposes (e.g., the Wikipedia database has size $O(10^9)$, so the size of 1%-sketches is $O(10^7)$). Note that, even though the corresponding primary key contains sufficient identifiers to uniquely identify a row, it is still possible to have false positive matches (in our case, we used the extension TPC-E documentation). The algorithm also fails to discover implied constraints; the pairs $(\sigma f \in \text{fk/pk pairs from four out of five different databases})$ is easy to generate instances of progressively larger sizes. We used counter-example for the randomness rule.

5.4 Column Names

For TPC-H can be largely attributed to the use of rule 7 (match-average value over all classifiers being 0.95). The best classifier (J48) is reported in [17]. As reported in that paper, the best classifier (J48) is best across all datasets.

5.5 Comparison With Alternatives

We then delete all pairs from these sets whose column names are identical, and compute the precision/recall on the resulting answer set. For TPC-E, we also report results using the extended set of valid constraints. For WP there is no single pair with identical column names, hence we exclude it from this experiment.

We considered two alternative estimators for the inclusion estimator $I$. The estimator proposed in [3], which is unbiased. However, it is defined over sketches whose sizes are a user-defined fraction of the total dataset size.

Table 4: Results after eliminating non-matching column names.

The resulting names are identical only if the pair is a valid constraint. For WP we can delete these prefixes and compare the remaining strings.

We then test the scalability of our method on TPC-H, for which it performs well. The time are shown in Figure 9. For readability, we use a logarithmic scale on both axes. As expected, each phase takes linear time. The time are shown in Figure 9.

Figure 7: Utility measures on TPC-H, Wikipedia and IMDB.

Figure 9: Scalability results.
Conclusion

- Linear approach to finding foreign key
- Ranking based on *Randomness* property
- Distance measure quantifying randomness
- Fast approximate algorithms for evaluating randomness over a large set of columns
- Comprehensive experimental validation using both synthetic and real datasets.
• A better approach to fk discovery, requiring no domain knowledge
• Novel idea of *Randomness*
• Evaluation on real world schemas and datasets
• They propose a solution to a very real and very important problem
• The first to consider multi-column fks.
1. "Highly unlikely that a database design incurs a bias" on p. 2 is an unsupported claim.

2. Algorithm in appendix is very dense and unexplained (A more little handholding please).

3. It is unclear why Wilcoxon test is relevant, nearly half a page is spent.

4. Approximate quantiles on p. 6 are not defined.

5. Example on fig. 1 is not too helpful.

6. Paper is fragmented into many topics, difficult to read.

7. From the description of EMD it is unclear how the probability mass affects the result.

8. Not explained why a dataset is dirty.
Relevance to Project

- Foreign keys are an integral part of good database design
- Given foreign keys many additional checks can be implemented
- The majority of open source systems we have examined does not have fks