

SQL Data Profiling of Foreign Keys

Mozhgan Memari, Sebastian Link^(✉), and Gillian Dobbie

Department of Computer Science, University of Auckland,
Auckland, New Zealand

{m.memari,s.link,g.dobbie}@auckland.ac.nz

Abstract. Referential integrity is one of the three inherent integrity rules and can be enforced in databases using foreign keys. However, in many real world applications referential integrity is not enforced since foreign keys remain disabled to ease data acquisition. Important applications such as anomaly detection, data integration, data modeling, indexing, reverse engineering, schema design, and query optimization all benefit from the discovery of foreign keys. Therefore, the profiling of foreign keys from dirty data is an important yet challenging task. We raise the challenge further by diverting from previous research in which null markers have been ignored. We propose algorithms for profiling unary and multi-column foreign keys in the real world, that is, under the different semantics for null markers of the SQL standard. While state of the art algorithms perform well in the absence of null markers, it is shown that they perform poorly in their presence. Extensive experiments demonstrate that our algorithms perform as well in the real world as state of the art algorithms perform in the idealized special case where null markers are ignored.

Keywords: Data profiling · Foreign key · Null · Semantics in data · SQL

1 Introduction

Motivation. Domain, entity, and referential integrity form the three inherent integrity constraints in relational database systems [5]. Entity integrity is enforced by primary keys to ensure that entities of an application domain are uniquely represented within a database [12]. Referential integrity is enforced by foreign keys to ensure that the intrinsic links between entities of an application domain are correctly represented within a database [12]. In the TPC-C database, the ORDER table contains the foreign key o_c_id, o_d_id, o_w_id, which references the primary key c_id, c_d_id, c_w_id of the CUSTOMER table. The primary key uniquely identifies a customer c_id within a district c_d_id of a warehouse c_w_id. The foreign key references the unique customer in a district of a warehouse who placed the order. Primary and foreign keys are fundamental building blocks of database system, and are indispensable for most database applications [5, 12].

In practice, designers fail to specify primary and foreign keys for various reasons. For example, they do not comprehend well enough the intrinsic links between data in the application domain, it is infeasible to enforce integrity due

to inconsistencies arising from data integration or database evolution over time, or because integrity enforcement will inhibit data acquisition [17]. Databases at enterprise level frequently contain hundreds of tables and thousands of columns, and are often insufficiently documented. In such situations it difficult to identify foreign keys. As a consequence, the discovery of primary and foreign keys is an important yet challenging core activity of data profiling [13]. This is due to the large number of candidates and the fact that the given data source may not even satisfy the meaningful primary and foreign keys; since these are not always enforced. Previous work on profiling primary and foreign keys has exclusively focused on purely relational databases, in which null markers are treated naively as any other domain value [17]. In the real world, null markers occur frequently and the SQL standard recommends a designated treatment for them.

Goal. This background motivates our objective to establish algorithms that efficiently profile unary and multi-column foreign keys from dirty and incomplete data. Efficiency refers to the following factors: (i) there should be a different algorithm for each of the semantics for foreign keys as proposed by the SQL standard, i.e. simple, full, and partial; (ii) the precision and recall of the algorithms should be similar to those of the state of the art algorithm for the idealized special case in which null markers are treated naively, and (iii) the profiling time of our algorithms should be similar to that of the state of the art algorithm.

Key Idea. Let X denote the sequence of distinct foreign key columns on table R_1 and Y the equal-length sequence of distinct primary key columns on R_2 . Then we write $R_1[X] \subseteq R_2[Y]$ to denote the foreign key on R_1 . For example, $\text{ORDER}[\text{o.c.id}, \text{o.d.id}, \text{o.w.id}] \subseteq \text{CUSTOMER}[\text{c.id}, \text{c.d.id}, \text{c.w.id}]$ denotes the foreign key on ORDER from the TPC-C benchmark. For the projection $r[Z]$ of a table instance r over table R onto the columns in sequence Z over R , the *inclusion coefficient* between $r_1[X]$ and $r_2[Y]$ has been defined as $\sigma(r_1[X], r_2[Y]) = |r_1[X] \cap r_2[Y]| / |r_1[X]|$ where $|S|$ denotes the number of elements in S . Intuitively, the inclusion coefficient $\sigma(r_1[X], r_2[Y])$ measures the proportion of values in $r_1[X]$ that are also in $r_2[Y]$. The higher $\sigma(r_1[X], r_2[Y])$ the more likely it is that $R_1[X] \subseteq R_2[Y]$ forms a true foreign key. However, the inclusion coefficient has only been defined for relations in which no null markers occur [14, 17]. The SQL standard recommends three different options for the semantics of foreign keys, which can be declared in the MATCH clause as simple, full, and partial [12]. The semantics differ in how they treat tuples with occurrences of NULL in foreign key columns. Simple semantics regards such tuples as non-offensive, while full semantics regards them as offensive. Finally, partial semantics regards tuples $t \in r_1[A_1, \dots, A_n]$ only as offensive if there is no tuple $t' \in r_2[B_1, \dots, B_n]$ that *subsumes* t , i.e., where $t[A_i] = t'[B_i]$ on all attributes A_i where $t[A_i] \neq \text{NULL}$. We say that a tuple t is X -total on a sequence X of distinct attributes, if $t[A] \neq \text{NULL}$ on all $A \in X$. Furthermore, we call the tuple t over table R total if it is R -total, and a relation r over R total if all tuples in r are total. We thus obtain the notions of simple σ_s , full σ_f , and partial inclusion coefficients σ_p by defining

Table 1. TPC-C foreign key and its different inclusion coefficients

ORDER				CUSTOMER				
...	o_c_id	o_d_id	o_w_id	...	c_id	c_d_id	c_w_id	...
	1	2	1		1	2	1	
	NULL	2	2		1	2	2	
	3	3	3		3	3	3	$\sigma_f = 4/10 = 0.4$
	NULL	5	3		2	5	1	
	2	3	2		2	5	2	$\sigma_s = 9/10 = 0.9$
	3	6	3		3	6	3	
	3	NULL	3		4	9	2	$\sigma_p = 8/10 = 0.8$
	4	2	2		4	2	2	
	4	NULL	2		7	9	3	
	2	5	NULL		6	5	1	

- $r_1[X] \cap_s r_2[Y] := \{t \in r_1[X] \mid \text{if } t \text{ is } X\text{-total, then } t \in r_2[Y]\}$,
- $r_1[X] \cap_f r_2[Y] := \{t \in r_1[X] \mid t \text{ is } X\text{-total and } t \in r_2[Y]\}$, and
- $r_1[X] \cap_p r_2[Y] := \{t \in r_1[X] \mid \exists t' \in r_2[Y] \text{ such that } t \text{ is subsumed by } t'\}$.

Indeed, when $r_1[X]$ and $r_2[Y]$ are total relations, simple σ_s , full σ_f , and partial inclusion coefficients σ_p all coincide with the standard inclusion coefficient.

Example 1. For illustration consider the foreign key FK $ORDER[o_c_id, o_d_id, o_w_id] \subseteq CUSTOMER[c_id, c_d_id, c_w_id]$ of the TPC-C schema. Table 1 shows sample data that violates the foreign key under all three semantics, since the fifth tuple - which is total - is not present in $CUSTOMER$. An intriguing question is if FK can still be discovered as a good foreign key candidate. For total relations this question has been answered affirmatively [17], but our data features $NULL$.

Indeed, \cap_f consists of only four tuples since full semantics regards incomplete tuples as offensive, \cap_s consists of nine tuples since simple semantics regards incomplete tuples as non-offensive, and \cap_p consists of eight tuples since the fourth and fifth $ORDER$ tuples are not subsumed by any $CUSTOMER$ tuple. \square

Example 1 illustrates that the three null marker semantics, proposed by the SQL standard, require different techniques to profile foreign keys.

Contributions. Firstly, we demonstrate that current methods for foreign key profiling over complete data are not suitable for incomplete data. These methods already exhibit poor precision and recall when few null markers occur. Secondly, based on the recommended semantics of foreign keys by the SQL standard, we propose three notions of inclusion coefficients. Each notion leads to a different algorithm for profiling SQL foreign keys in dirty and incomplete data. We focus on foreign keys that reference a primary key, the most common case in practice. Thirdly, experiments with benchmark data demonstrate that our algorithms for simple and partial semantics perform as well for dirty and incomplete data as the current state of the art algorithm performs in the idealized special case where incomplete data is absent. Our results for precision and recall are robust under different rates of incompleteness. The penalty for sustaining strong profiling results for dirty and incomplete data is only marginal when comparing our profiling time to that of the state of the art for complete data.

Organization. In Sect. 2 we recall the three semantics the SQL standard proposes for foreign keys. Our profiling algorithms are proposed in Sect. 3. The algorithms are evaluated in Sect. 4. Their combination with schema-driven techniques is evaluated in Sect. 5. Related work is discussed in Sect. 6 before we conclude and comment on future work in Sect. 7.

2 Referential Integrity and the SQL Standard

Inclusion dependencies, and foreign keys in particular, are referential integrity constraints that enforce the intrinsic links between data items of an application domain within a database system [8, 11]. An *inclusion dependency* (IND) over a database schema \mathbf{R} is an expression of the form $R_1[X] \subseteq R_2[Y]$, where R_1, R_2 are relation schemata of \mathbf{R} , and X and Y are equal-length sequences of distinct columns from R_1 and R_2 , respectively. Having $X = f_1, \dots, f_n$ and $Y = k_1, \dots, k_n$, the inclusion dependency holds in the given database over \mathbf{R} if for every tuple t in the relation over R_1 there is some tuple t' in the relation over R_2 such that $t[f_i] = t'[k_i]$ for $1 \leq i \leq n$. An inclusion dependency $R_1[X] \subseteq R_2[Y]$ is a *foreign key* if Y forms a key of R_2 [11].

Match Options to Handle Nulls. An exception to the semantics of foreign keys are occurrences of the null marker NULL. If some foreign key column features an occurrence of NULL, then the semantics of a foreign key is defined by the MATCH clause of the SQL standard [12]. Available options are full, simple and partial, each proposing different ways to handle occurrences of NULL.

Under *full* semantics, the foreign key $R_1[f_1, \dots, f_n] \subseteq R_2[k_1, \dots, k_n]$ is satisfied if every tuple $t \in r_1$ over R_1 is $\{f_1, \dots, f_n\}$ -total and there is some tuple $t' \in r_2$ over R_2 such that $t[f_i] = t'[k_i]$ for all $i = 1, \dots, n$. Hence, full semantics does not permit any occurrences of NULL on any of the foreign key columns.

Under *simple* semantics, the foreign key $R_1[f_1, \dots, f_n] \subseteq R_2[k_1, \dots, k_n]$ is satisfied if for every tuple $t \in r_1$ over R_1 , either $t[f_i] = \text{NULL}$ for some $i \in \{1, \dots, n\}$, or there is some tuple $t' \in r_2$ over R_2 such that $t[f_i] = t'[k_i]$ for all $i = 1, \dots, n$. Hence, simple referential integrity is never violated by tuples that are partially defined on the foreign key columns.

Under *partial* semantics, the foreign key $R_1[f_1, \dots, f_n] \subseteq R_2[k_1, \dots, k_n]$ is satisfied if for every tuple $t \in r_1$ over R_1 there is some tuple $t' \in r_2$ over R_2 such that $t[f_1, \dots, f_n]$ is subsumed by $t'[k_1, \dots, k_n]$. Hence, partial referential integrity may also be violated by tuples that are partially defined on the foreign key columns. It can be understood as a compromise that balances the pessimistic full semantics and the optimistic simple semantics.

In Table 1, only tuples 1, 3, 6, and 8 of the ORDER table satisfy full semantics, while all tuples except for tuple 5 satisfy simple semantics, and all tuples except for tuples 4 and 5 satisfy partial semantics.

3 Profiling Foreign Keys from Dirty and Incomplete Data

We propose two-step algorithms for profiling full, simple, and partial foreign keys from dirty and incomplete data. In phase one, we apply full, simple, and partial

inclusion coefficients to identify good candidates for foreign keys. In phase two, a randomness test validates whether the distribution of the foreign key values is similar to that of the key values.

Inclusion Coefficients for Incomplete Data. The first challenge is to find reasonable candidates from data that may violate meaningful foreign keys. Similar to [2, 6, 17], a foreign key is a candidate when its inclusion coefficient meets a user-defined threshold θ . We introduce three different inclusion coefficients, resulting from the SQL standard semantics for null markers.

Example 2. When $\theta = 0.8$, 20% inconsistencies in the given data are considered acceptable. The simple and partial coefficients of the foreign key FK from Example 1 meet θ , but the full inclusion coefficient does not. \square

Exact computations of inclusion coefficients are infeasible for big data sizes, so we approximate them by bottom- k sketches [17]. A bottom- k sketch consists of those values of a given data set that have the k smallest ranks assigned to them by a hash function. For a bottom- k sketch of a multi-column key, the hash function is applied to the concatenation of the values on each column [17].

Algorithm 3.1. DISCOVERY($\mathbf{C}, \mathbf{T}, \mathbf{P}_u, \mathbf{P}_m, \theta, O$)

```

main
for all  $C \in \mathbf{C} : \hat{C} \leftarrow \text{BottomK}[C]$ 
for all  $P = \{C_1, \dots, C_n\} \in (\mathbf{P}_u \cup \mathbf{P}_m)$ 
  do  $\left\{ \begin{array}{l} \text{for } p \leftarrow 1 \text{ to } n \\ \quad \text{do } \left\{ \begin{array}{l} \text{for all } \hat{C}_f \in \mathbf{C} \\ \quad \sigma := \begin{cases} \text{output (INCLUSION-S}(\hat{C}_f, \hat{C}_p)), & \text{if } O = \mathbf{s} \text{ or } O = \mathbf{p} \\ \text{output (INCLUSION-F}(\hat{C}_f, \hat{C}_p)), & \text{if } O = \mathbf{f} \end{cases} \\ \quad \text{if } \sigma \geq \theta \\ \quad \quad \text{then } \begin{cases} \mathbf{F}_u \leftarrow (C_p, C_f), & \text{if } n = 1 \\ S[P, C_p] \leftarrow C_f, & \text{if } n > 1 \end{cases} \end{array} \right.$ 
for all  $P = \{C_1, \dots, C_n\} \in \mathbf{P}_m$ 
  do  $\left\{ \begin{array}{l} \text{for all } \mathbf{T} \in \text{schema} \\ \quad \text{do } F_m \leftarrow (\{C'_1, \dots, C'_n\} | C'_i \in S[P, C_i] \cap T), P) \\ \quad \text{for all } F = (\{C'_1, \dots, C'_n\}, P) \in F_m \\ \quad \quad \hat{F} \leftarrow \text{BottomK}[F] \\ \quad \quad \hat{P} \leftarrow \text{BottomK}[P] \\ \quad \quad \sigma := \begin{cases} \text{output (INCLUSIONS}(\hat{F}, \hat{P})), & \text{if } O = \mathbf{s} \\ \text{output (INCLUSIONF}(\hat{F}, \hat{P})), & \text{if } O = \mathbf{f} \\ \text{output (INCLUSIONP}(\hat{F}, \hat{P})), & \text{if } O = \mathbf{p} \end{cases} \\ \quad \quad \text{if } \sigma(\hat{F}, \hat{P}) \geq \theta \\ \quad \quad \quad \text{then } \mathbf{F}_m \leftarrow (F, P) \end{array} \right.$ 
return  $(\mathbf{F}_u \cup \mathbf{F}_m)$ 

```

Algorithm 3.1 shows the pseudo-code of our proposed algorithm. Like in [17] we assume that a set of unary (\mathbf{P}_u) and multi-column (\mathbf{P}_m) primary keys has been obtained already. \mathbf{C} and \mathbf{T} , respectively, refer to the set of columns and tables in the given schema. The option O for the MATCH clause of the SQL standard are denoted by \mathbf{f} for full, \mathbf{s} for simple, and \mathbf{p} for partial. \hat{C} refers to the bottom- k sketch of column C . For $O \in \{\mathbf{f}, \mathbf{s}, \mathbf{p}\}$, INCLUSION- $O(\hat{F}, \hat{P})$ computes an approximation of the full, simple, or partial inclusion coefficient by counting the number of foreign key tuples in the current bottom- k sketch \hat{F} that meet

the requirements of full, simple, or partial semantics, respectively, and dividing this number by the total number of foreign key tuples in \hat{F} . If f_j denotes the j th foreign key tuple in \hat{F} , then f_j meets the requirements of (i) full semantics, when $f_j[C'] \neq \text{NULL}$ for all columns C' and f_j occurs in \hat{P} ; (ii) simple semantics, when either $f_j[C'] = \text{NULL}$ for some column C' or f_j occurs in \hat{P} ; and (iii) partial semantics, when there is some $p \in \hat{P}$ that subsumes f_j .

Algorithm 3.1 starts by profiling unary inclusion dependencies as in [6, 17]. The novel idea is to profile foreign keys under different SQL semantics. For unary candidates, simple and partial semantics coincide. Thus, INCLUSION-S computes unary inclusion coefficients under simple and partial semantics.

All candidates for unary inclusion dependencies are stored in memory for profiling multi-column foreign keys. If a unary inclusion dependency references some $P \in \mathbf{P}_u$, it is stored as a unary foreign key in \mathbf{F}_u . If there is some multi-column primary key, Algorithm 3.1 calculates the inclusion coefficient for a set of unary inclusion dependencies which occur in one table and pair-wisely reference the columns of this primary key. Finally, all pairs (F, P) which pass the inclusion test are stored in \mathbf{F}_m . Algorithm 3.1 returns a set $\mathbf{F}_u \cup \mathbf{F}_m$ of unary and multi-column candidate foreign keys.

Example 3. Consider the TPC-C data set from Example 1 and a threshold of $\theta = 0.9$. Algorithm 3.1 calculates inclusion coefficients for unary candidates first:

	$ \text{ORDER}[o_c_id, o_d_id, o_w_id] $	σ_p	σ_s	σ_f
$O[o_c_id] \subseteq C[c_id]$	5	1	1	0.9
$O[o_d_id] \subseteq C[c_d_id]$	5	1	1	0.9
$O[o_w_id] \subseteq C[c_w_id]$	4	1	1	0.75
FK	10	0.8	0.9	

In particular, $\sigma_f(o_w_id, c_w_id) = 0.75 < \theta$, which means this unary inclusion dependency does not meet the requirements of full semantics. Therefore, our $FK \text{ ORDER}[o_c_id, o_d_id, o_w_id] \subseteq \text{CUSTOMER}[c_id, c_d_id, c_w_id]$ is not further considered under full semantics. Final results show that this foreign key is profiled under simple but not under partial semantics, since $\sigma_p = 0.8 < \theta$. \square

Testing Randomness on Dirty and Incomplete Data. The randomness test from [14, 17] checks if the distinct values of the referencing column set X have the same distribution as the distinct values in the referenced column set Y . This test is good for eliminating false positives [14, 17]. For complete data the Earth Mover’s Distance (EMD) represents the least amount of work required to move the set of values in the candidate foreign key to the set of values in the referenced primary key. The smaller the EMD the closer the distributions for the candidate foreign and primary key values. We extend the randomness test of [14, 17] from complete to incomplete data. Our method measures the likelihood of the candidate pair (F, P) dependent on either full, simple, or partial semantics.

Algorithm 3.2 returns a set of candidates ranked in increasing order of their EMDs. Firstly, Algorithm 3.2 applies the extended randomness test to further

prune the candidates found by Algorithm 3.1. As in [17], we apply quantile histograms to approximate EMDs. The histogram is calculated by a user-defined quantile constant for every column of a primary key. For multi-column keys, a quantile grid is constructed from the quantile histogram of each column. The grid for a candidate foreign key is constructed by populating the associated primary key grid with the values in its sketch. The approximate EMD is computed from the distance between the quantile grids of primary and foreign key. We empower this method to deal with real world data by imputing null markers with actual domain values that are consistent with simple and partial semantics, respectively. The procedure SIMPLE replaces the incomplete bottom- k sketch \hat{F} of the given foreign key F by the completed sketch \hat{F}_{total} in which NULL occurrences in \hat{F} have been imputed with randomly chosen domain values from the referenced primary key sketch \hat{P} . This is consistent with the simple semantics. The procedure PARTIAL imputes NULL occurrences in *subsumed* foreign key tuples t with randomly chosen domain values from tuples of the primary key sketch that subsume t . This is consistent with the partial semantics. Based on the completed sketches, Algorithm 3.2 proceeds by approximating the EMD of candidate foreign keys based on quantiles of \hat{F}_{total} .

Algorithm 3.2. RANDOMNESS TEST($\mathbf{C}, \mathbf{T}, \mathbf{P}_u, \mathbf{P}_m, \theta, O$)

```

procedure PARTIAL( $\hat{F}, \hat{P}$ )
   $F_{total} \leftarrow \emptyset$ 
  for all  $j \leftarrow 1$  to  $|\hat{F}|$ 
    do  $\left\{ \begin{array}{l} \text{if } \{C'_i \in C' \mid \hat{f}_j[C'_i] = \text{NULL}\} \neq \emptyset \text{ and } \exists \hat{p} \in \hat{P} (\hat{p} \text{ subsumes } \hat{f}_j) \\ \text{then } \hat{f}_j[C'] \leftarrow \hat{p}[C] \\ F_{total} \leftarrow F_{total} \cup \{\hat{f}_j[C']\} \end{array} \right.$ 
  return ( $F_{total}$ )

procedure SIMPLE( $\hat{F}, \hat{P}$ )
   $F_{total} \leftarrow \emptyset$ 
  for all  $j \leftarrow 1$  to  $|\hat{F}|$ 
    do  $\left\{ \begin{array}{l} \text{if } \exists C'_i (\hat{f}_j[C'_i] = \text{NULL}) \\ \text{then } \hat{f}_j[C'] \leftarrow \hat{p}[C] \\ F_{total} \leftarrow F_{total} \cup \{\hat{f}_j[C']\} \end{array} \right.$ 
  return ( $F_{total}$ )

main
 $\mathbf{F}_u \cup \mathbf{F}_m \leftarrow \text{output} (\text{DISCOVERY}(\mathbf{C}, \mathbf{T}, \mathbf{P}_u, \mathbf{P}_m, \theta, O))$ 
for all  $P = \{C_1, \dots, C_n\} \in \{F_m, P_m\} \cup \{F_u, P_u\}$ 
   $Q[\hat{P}] \leftarrow \text{Quantile}[\hat{P}]$ 
  for all  $F \in \{F_m, P_m\}$ 
    do  $\left\{ \begin{array}{l} \hat{F}_{total} := \left\{ \begin{array}{l} \text{output} (\text{PARTIAL}(\hat{F}, \hat{P})), \text{ if } O = \mathbf{p} \\ \text{output} (\text{SIMPLE}(\hat{F}, \hat{P})), \text{ if } O = \mathbf{s} \\ \hat{F}, \text{ if } O = \mathbf{u} \end{array} \right. \\ Q[\hat{F}_{total}] \leftarrow \text{Quantile}[\hat{F}_{total}] \\ L \leftarrow (F_m, \text{EMD}_n(Q[\hat{F}_{total}], Q[\hat{P}])) \end{array} \right.$ 
  for all  $F \in \{F_u, P_u\}$ 
    do  $\left\{ \begin{array}{l} Q[\hat{F}] \leftarrow \text{Quantile}[\hat{F}] \\ L \leftarrow (F_u, \text{EMD}_n(Q[\hat{F}], Q[\hat{P}])) \end{array} \right.$ 
  return ( $L$ )

```

Example 4. Applying PARTIAL from Algorithm 3.2 to Example 1 may yield the following completed sketch: $\{(1,2,1), (1,2,2), (3,3,3), (\text{NULL},5,3), (2,3,2), (3,6,3),$

$(4,2,2), (4,9,2), (2,5,2)$. An alternative is $\{(1,2,1), (4,2,2), (3,3,3), (\text{NULL},5,3), (2,3,2), (3,6,3), (2,5,1)\}$. In each completion, all partial tuples are replaced by complete tuples that subsume them. Tuple $(\text{NULL},5,3)$ is not subsumed by any primary key tuple and violates partial semantics. Applying SIMPLE produces a completed sketch in which all NULL occurrences of $(\text{NULL},2,2), (\text{NULL},5,3), (3,\text{NULL},3), (4,\text{NULL},2), (2,5,\text{NULL})$ are imputed by random domain values from the referenced primary key, for example by $(1,2,2), (3,5,3), (4,5,2), (2,5,3)$. Even $(\text{NULL},5,3)$ is replaced, in consistency with simple semantics. \square

4 Evaluation of Data-Driven Profiling Techniques

We evaluate the data-driven profiling of foreign keys from real world data sets.

Characteristics of Experiments. Our algorithms are evaluated on two benchmark databases TPC-C and TPC-H¹. The characteristics of the data sets are summarized as follows:

Data set	#tables	#rows	Data set size	#unary FKs	#multi-column FKs
TPC-C	9	2.41M	0.39G	3	7
TPC-H	9	9.42M	1.43G	7	1

Algorithms were implemented in C++ and run on an Intel Core i5 CPU 3.3 GHz with 8 GB RAM. The operating system was 64-bit Windows 7 Enterprise, Service pack 1. The database management system we used was MySQL version 5.5. Our algorithms are evaluated in terms of accuracy and time with respect to increasing levels of incompleteness in the data, starting with complete data. We use three well-known measures of accuracy: precision, recall and f-measure. For each data set, we consider the constraints which are explicitly declared in the schema as the golden standard for determining the accuracy of our algorithms. This ensures that we can compare our results to those of [17] who followed the same approach. In TPC-C, there are three single-column, three 2-column ($Cu \subseteq Di, Hi \subseteq Di, OL \subseteq St$) and four 3-column ($Hi \subseteq Cu, Or \subseteq Cu, OL \subseteq Or, NO \subseteq Or$) foreign keys (FKs). In TPC-H, 7 of the 8 foreign keys are single-column and the only composite foreign key has 2-columns. We randomly generated null marker occurrences in the foreign key columns. Starting from the given complete data set, null marker occurrences were randomly generated for foreign key columns, and in different percentages of tuples to pinpoint the impact of different levels of incompleteness on our measures. The names of the resulting data sets originate from the original data set augmented by the percentage of nullified tuples. For example, TPC-C2 results from TPC-C where 2% of the tuples had null markers in their foreign key columns.

Profiling Accuracy and Time. We applied Algorithm 3.2 to the benchmark data sets with increasing levels of incompleteness under simple, partial and full

¹ <http://www.tpc.org/tpcc/> and <http://www.tpc.org/tpch/>.

Table 2. Data profiling of SQL foreign keys from incomplete benchmark variants

TPC-C												
Dataset	Simple			Partial			Full			$T_{3.1}$	$T_{3.2}$	
	t_p	f_p	$T_{3.1}$	$T_{3.2}$	t_p	f_p	$T_{3.1}$	$T_{3.2}$	t_p			f_p
	hr:mn		hr:mn	hr:mn	hr:mn		hr:mn	hr:mn	hr:mn		hr:mn	
TPC-C	9	165	0:47	3:09	9	165	0:47	3:09	9	165	0:34	2:44
TPC-C2	9	166	0:41	3:29	9	162	0:58	3:47	3	24	0:33	1:14
TPC-C5	9	170	0:40	3:32	9	164	0:41	3:31	1	21	0:26	0:56
TPC-C10	8	160	0:45	3:09	8	155	1:02	3:24	1	21	0:29	0:42
TPC-C25	9	149	0:41	2:41	9	145	0:59	3:03	1	20	0:23	0:35
TPC-C50	9	167	0:36	3:22	9	162	0:52	3:38	1	20	0:30	0:57
TPC-C75	9	150	0:48	2:55	7	145	1:05	3:12	1	20	0:27	0:45

TPC-H												
Dataset	Simple			Partial			Full			$T_{3.1}$	$T_{3.2}$	
	t_p	f_p	$T_{3.1}$	$T_{3.2}$	t_p	f_p	$T_{3.1}$	$T_{3.2}$	t_p			f_p
	hr:mn		hr:mn	hr:mn	hr:mn		hr:mn	hr:mn	hr:mn		hr:mn	
TPC-H	7	40	0:50	1:36	7	40	0:50	1:36	7	40	0:49	1:35
TPC-H2	7	52	1:12	2:09	6	46	1:24	2:12	6	46	0:56	1:44
TPC-H5	7	46	1:14	2:04	6	40	1:22	2:07	5	39	0:55	1:40
TPC-H10	7	46	1:17	2:09	6	40	1:25	2:11	5	39	0:56	1:43
TPC-H25	7	46	1:13	2:04	6	40	1:25	2:06	5	39	0:54	1:40
TPC-H50	5	45	0:59	1:46	4	42	1:07	1:49	3	37	0:51	1:30
TPC-H75	5	39	1:02	1:48	4	38	1:06	1:44	3	37	0:51	1:28

semantics, respectively. As in [17] we applied $\theta = 0.9$, bottom-256 sketches, 256-quantiles for unary, and 16-quantiles for multi-column foreign key candidates. The results are summarized in Fig. 1 and Table 2. Here, t_p denotes the number of true positives, f_p the number of false positives, $T_{3.2}$ the overall time of running Algorithm 3.2 in hours and minutes (hr:mn), and $T_{3.1}$ indicates the time spent on Algorithm 3.1 as part of running Algorithm 3.2. *Partial-PK* denotes an optimization technique discussed in Sect. 5.

Our first observation is that on the original data sets TPC-C and TPC-H, respectively, all accuracy measures agree under all 3 different semantics. The sweet spot for balancing precision and recall is given by the top-20% ranked foreign keys discovered from TPC-C, which yield an f-measure of 0.36. For TPC-H, the f-measure is around 0.4 when the sweet spots of top-55% or -60% ranked foreign keys are considered, see Table 3.

Our second observation is that full semantics is inadequate for SQL data profiling of foreign keys. Since this is the semantics applied by [17], their good performance for complete data does not carry over to incomplete data. The poor performance is more pronounced for TPC-C, since there are more multi-column foreign keys. Just having null markers present in 2% of all tuples, lowers the recall for full semantics from 0.9 to 0.3. Introducing null markers in 5% of all tuples brings the recall down to 0.1, and this foreign key no longer features amongst the top-25%. On TPC-H with 7 unary foreign keys and 1 binary foreign key, the

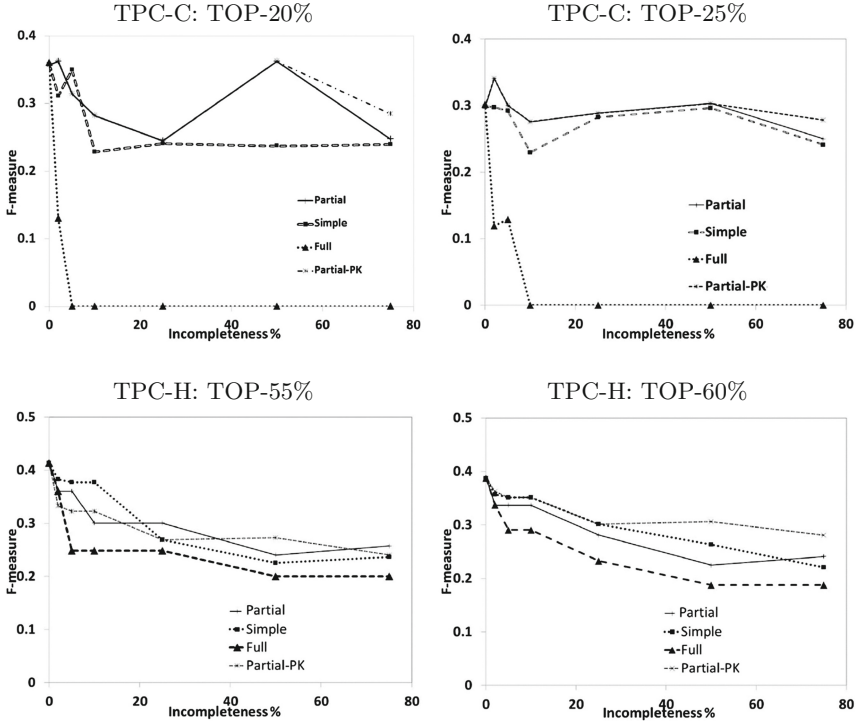


Fig. 1. F-measure: TPC-C and TPC-H

f-measure under full semantics also declines as soon as null markers occur. In particular, the binary foreign key is never profiled under full semantics on any data set with some level of incompleteness.

Our third observation is that both simple and partial semantics perform very well in profiling foreign keys. Their measures of accuracy remain robust under increasing levels of incompleteness and are competitive with the corresponding measures on complete data, as Fig. 1 and Table 2 show.

Our final observation is that the robustness of simple and partial semantics under incomplete data comes at a very reasonable increase in profiling time, when compared to the state of the art algorithm for complete data. It is natural that the profiling time increases with the recall. However, spending 3hr:47mn - which is already the worst case - instead of 2hr:44mn is well worth the increase in accuracy, considering the purpose of data profiling.

Comparing Precision of Simple and Partial Semantics. While simple semantics outperforms partial semantics in recall, the roles change for precision and f-measure. Table 3 shows for each data set and semantics, the lowest percentage that captures all discovered foreign keys from the golden standard. Hence, partial semantics ranks meaningful foreign keys higher.

Table 3. Partial semantics ranks meaningful foreign keys higher

	Partial			Simple			Full		
	t_p	$X\%$	EMD	t_p	$X\%$	EMD	t_p	$X\%$	EMD
TPC-C2	9	25	0.026	9	29	0.028	3	37	0.004
TPC-C5	9	31	0.04	9	36	0.047	1	27	0.033
TPC-C10	8	35	0.05	8	37	0.05	1	32	0.001
TPC-C25	9	33	0.059	9	37	0.07	1	29	0.001
TPC-C50	9	35	0.05	9	36	0.05	1	29	0.001
TPC-C75	7	36	0.054	9	38	0.062	1	38	0.001

Table 4. Semantics with lower EMD for TPC-C multi-column foreign keys

	TPC-C2	TPC-C5	TPC-C10	TPC-C25	TPC-C50	TPC-C75
$Cu \subseteq Di$	S	P	P	S	P	P
$Hi \subseteq Di$	S	S	P	P	S	P
$OL \subseteq St$	S	S	-	S	S	S
$Hi \subseteq Cu$	P	P	P	P	P	P
$Or \subseteq Cu$	P	P	P	P	P	P
$OL \subseteq Or$	P	P	S	P	S	P

Table 5. Profiling benchmarks with different sample sizes under partial semantics

	Sample Size	t_p	f_p	$T_{3.1}$ hr:mn		Sample Size	t_p	f_p	$T_{3.1}$ hr:mn
TPC-C75	128	5	131	0:41	TPC-H75	128	3	36	0:36
TPC-C75	256	7	145	1:05	TPC-H75	256	4	38	1:06
TPC-C75	512	7	152	1:12	TPC-H75	512	4	38	1:53
TPC-C75	1024	7	156	1:20	TPC-H75	1024	4	42	2:33
TPC-C75	10K	8	162	19:16	TPC-H75	10K	5	45	19:27

Table 4 shows for each of the meaningful multi-column foreign keys of TPC-C, which of simple (S) and partial (P) semantics resulted in a lower EMD. In about two thirds of the cases (23 out of 35) this was partial semantics.

Size of Sketches. As in [17] we observe that bottom-256 sketches are a sweet spot for the experiments. Bottom-128 sketches lead to drops in recall, and bottom- k sketches for $k = 512, 1024, 10K$ lead to minor improvements in recall, more false positives, and higher profiling time, see Table 5 for example.

Other Experiments. We have applied our algorithms to the CM data set about employer costs for employee compensation from US government data <http://catalog.data.gov/>. It consists of 11 tables featuring 11 one-attribute foreign keys and two 3-column foreign keys. On total data, we obtain a recall of 0.85 and

Table 6. TPC-C and TPC-H with partial-PK, 256-bottom sketch

	Partial				Partial-PK					
	t_p	f_p	$T_{3.1}^a$	Recall	F-measure ^b	t_p	f_p	$T_{3.1}$	Recall	F-measure ^b
TPC-C75	7	145	1:05	0.7	0.25	9	152	1:07	0.9	0.28
TPC-H75	4	38	1:06	0.5	0.24	5	41	1:06	0.63	0.28

^ahr:mn: Running time (hour & minute) of Algorithm 3.1

^bIn top-20 % for TPC-C and top-60 % for TPC-H

precision of 0.3 under all semantics. Adding 2 %, 5 % and 10 % null values, respectively, simple and partial semantics maintain these rates, while full semantics drops to a recall of 0.31 and precision of 0.2.

5 Optimization Strategies

We propose and evaluate two strategies for improving our algorithms.

Increasing Recall. Recall under partial semantics can be improved to the level of recall for simple semantics. Under partial semantics, our algorithm is looking for a matching reference in the bottom-256 sketch of the primary key. If a match does not exist in the sketch, it may still exist in the whole table. Referring to this strategy as *Partial-PK*, Fig. 1 shows its increase in f-measure in comparison to scanning only the bottom-256 sketch of the referenced primary key. Moreover, Table 6 gives an indication of the very minor penalty in profiling time, resulting from the presence of the index for the referenced primary key.

Schema-Driven Strategies. We add two schema-driven techniques leading to significant improvements of our measures, as in [17].

The first technique concerns the golden standard. So far, it comprises only those foreign keys that were explicitly specified in the benchmark documentation. In reality, these foreign keys logically imply other foreign keys [3], which our current algorithms treat (incorrectly) as false positives. On the TPC-C schema, for example, the two foreign keys $\text{ORDER}[\text{o_c_id}, \text{o_d_id}, \text{o_w_id}] \subseteq \text{CUSTOMER}[\text{c_id}, \text{c_d_id}, \text{c_w_id}]$ and $\text{CUSTOMER}[\text{c_d_id}, \text{c_w_id}] \subseteq \text{DISTRICT}[\text{d_id}, \text{d_w_id}]$ logically imply the foreign key $\text{ORDER}[\text{o_d_id}, \text{o_w_id}] \subseteq \text{DISTRICT}[\text{d_id}, \text{d_w_id}]$. Adding all implied foreign keys to the golden standard is more realistic and leads to an improvement of our accuracy measures.

Our second technique applies a pruning strategy from [17] regarding the similarity of column names. Finding good measures for string similarity in database schemata is an open problem and can easily lead to false positives [17]. However, the documentation for the benchmark databases is extensive and explains the naming conventions. For TPC-C, for example, we can apply string identity after removing table name prefixes from column names. The resulting names are identical only if the candidate is an implied foreign key.

Adding these techniques to our algorithms results in a perfect precision score of 1, and more than doubles our f-measure in comparison to applying the data-driven algorithms only, see Fig. 2.

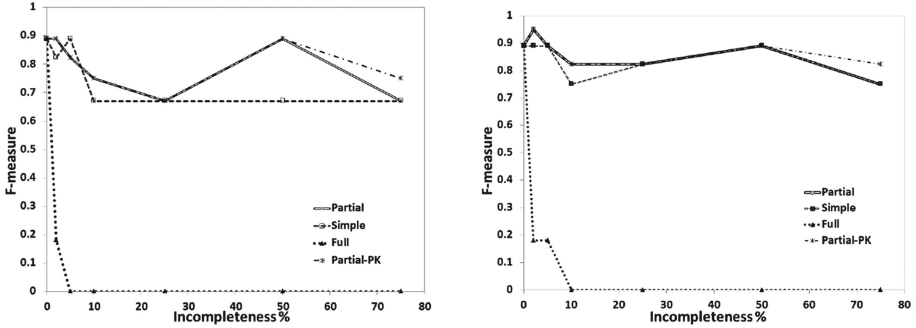


Fig. 2. F-measures after adding schema-driven techniques to TPC-C using the Top-20 % of candidates on the left and the Top-25 % of candidates on the right

6 Related Work

Previous algorithms for profiling referential integrity constraints do not consider incomplete data. The state of the art algorithm that considers dirty data is presented in [17]. The authors established a randomness test that reduces a large number of false positives and can profile foreign keys efficiently from complete data. Our evaluation on the original TPC-C data set, without introducing null markers, provides further evidence for the efficiency of those techniques [17], which did not consider TPC-C despite its high number of multi-column foreign keys compared to TPC-H and TPC-E.

State of the art algorithms for efficiently profiling uniqueness constraints are described in [7, 16]. Profiling inclusion dependencies is an effective pre-cursor for profiling foreign keys [14, 17]. In fact, the discovery of unary foreign keys from unary inclusion dependencies has been studied by several authors [2, 6, 8, 11]. Data summaries and random sampling, cliques, min-hash sketches and Jaccard coefficient all constitute pruning techniques from previous work [2, 4, 6, 9, 17]. Lopes et al. [10] profile foreign keys from given SQL query workloads. Approximate inclusion dependencies constitute a different popular approach to accommodate dirty data within the data profiling framework [2, 6, 17]. The profiling of data quality semantics is important for big data [15].

7 Conclusion and Future Work

We have proposed the first algorithms for profiling foreign keys from dirty and incomplete data. Our techniques target the three different semantics for null

marker occurrences as proposed by the SQL standard. On complete data, all three semantics coincide with the strictly relational semantics. Our detailed evaluation demonstrates several insights. Firstly, full semantics is not suitable for profiling foreign keys from incomplete data, even when only a few null markers occur. Secondly, our purely data-driven algorithms for simple and partial semantics perform as well as the state of the art algorithm does for complete data. Thirdly, the robustness of profiling with simple and partial semantics under different levels of incompleteness incurs very reasonable penalties in profiling time. Finally, combining data- with schema-driven techniques optimizes performances significantly, which is consistent with previous work on complete data.

In future work we will look at other strategies for imputing null markers during EMD calculation. Identifying thresholds of inclusion coefficients that work well for our profiling techniques is an interesting goal. We will also consider the less frequently observed case in which null markers also occur in referenced columns. Our preliminary observations suggest only marginal changes to the overall trend we have described. The extension of our techniques to profiling conditional foreign keys from dirty and incomplete data is an interesting goal for data cleaning. So far, all approaches to this problem do not address dirty data or different semantics for null markers [1].

Acknowledgement. This research is supported by the Marsden fund council from Government funding, administered by the Royal Society of New Zealand.

References

1. Bauckmann, J., Abedjan, Z., Leser, U., Müller, H., Naumann, F.: Discovering conditional inclusion dependencies. In: Chen, X., Lebanon, G., Wang, H., Zaki, M.J. (eds.) 21st ACM International Conference on Information and Knowledge Management, CIKM 2012, Maui, HI, USA, October 29–November 02, 2012, pp. 2094–2098. ACM (2012)
2. Bauckmann, J., Leser, U., Naumann, F., Tietz, V.: Efficiently detecting inclusion dependencies. In: Chirkova, R., Dogac, A., Özsu, M.T., Sellis, T.K. (eds.) Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, 15–20 April 2007, pp. 1448–1450. IEEE (2007)
3. Casanova, M.A., Fagin, R., Papadimitriou, C.H.: Inclusion dependencies and their interaction with functional dependencies. *J. Comput. Syst. Sci.* **28**(1), 29–59 (1984)
4. Chen, Z., Narasayya, V.R., Chaudhuri, S.: Fast foreign-key detection in Microsoft SQL server PowerPivot for Excel. *Proc. VLDB* **7**(13), 1417–1428 (2014)
5. Codd, E.: A relational model of data for large shared data banks. *Commun. ACM* **13**(6), 377–387 (1970)
6. De Marchi, F., Lopes, S., Petit, J.M.: Unary and n-ary inclusion dependency discovery in relational databases. *Int. J. Intell. Syst.* **32**(1), 53–73 (2009)
7. Heise, A., Quiané-Ruiz, J., Abedjan, Z., Jentzsch, A., Naumann, F.: Scalable discovery of unique column combinations. *Proc. VLDB* **7**(4), 301–312 (2013)
8. Kantola, M., Mannila, H., Räihä, K.J., Siirtola, H.: Discovering functional and inclusion dependencies in relational databases. *Int. J. Intell. Syst.* **7**(7), 591–607 (1992)

9. Koeller, A., Rundensteiner, E.A.: Heuristic strategies for the discovery of inclusion dependencies and other patterns. In: Spaccapietra, S., Atzeni, P., Chu, W.W., Catarci, T., Sycara, K. (eds.) *Journal on Data Semantics V. LNCS*, vol. 3870, pp. 185–210. Springer, Heidelberg (2006)
10. Lopes, S., Petit, J., Toumani, F.: Discovering interesting inclusion dependencies: application to logical database tuning. *Inf. Syst.* **27**(1), 1–19 (2002)
11. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. *Data Min. Knowl. Disc.* **1**(3), 241–258 (1997)
12. Melton, J., Simon, A.R.: *SQL:1999: Understanding Relational Language Components*. Morgan Kaufmann, Boston (2001)
13. Naumann, F.: Data profiling revisited. *SIGMOD Rec.* **42**(4), 40–49 (2013)
14. Rostin, A., Albrecht, O., Bauckmann, J., Naumann, F., Leser, U.: A machine learning approach to foreign key discovery. In: *12th International Workshop on the Web and Databases, WebDB 2009*, Providence, RI, USA, 28 June 2009 (2009)
15. Saha, B., Srivastava, D.: Data quality: the other face of big data. In: Cruz, I.F., Ferrari, E., Tao, Y., Bertino, E., Trajcevski, G. (eds.) *30th International Conference on Data Engineering, ICDE 2014*, Chicago, IL, USA, 31 March–4 April 2014, pp. 1294–1297. IEEE (2014)
16. Sismanis, Y., Brown, P., Haas, P.J., Reinwald, B.: GORDIAN: efficient and scalable discovery of composite keys. In: Dayal, U., Whang, K., Lomet, D.B., Alonso, G., Lohman, G.M., Kersten, M.L., Cha, S.K., Kim, Y. (eds.) *Proceedings of the 32nd International Conference on Very Large Data Bases*, Seoul, Korea, 12–15 September 2006, pp. 691–702. ACM (2006)
17. Zhang, M., Hadjieleftheriou, M., Ooi, B.C., Procopiuc, C.M., Srivastava, D.: On multi-column foreign key discovery. *Proc. VLDB* **3**(1), 805–814 (2010)