# Referential Integrity Quality Metrics

Carlos Ordonez
University of Houston
Houston, TX 77204, USA

Javier García-García
UNAM
Mexico City, CU 04510, Mexico [*]

**Abstract**

Referential integrity is an essential global constraint in a relational database, that maintains it in a complete and consistent state. In this work, we assume the database may violate referential integrity and relations may be denormalized. We propose a set of quality metrics, defined at four granularity levels: database, relation, attribute and value, that measure referential completeness and consistency. Quality metrics are efficiently computed with standard SQL queries, that incorporate two query optimizations: left outer joins on foreign keys and early foreign key grouping. Experiments evaluate our proposed metrics and SQL query optimizations on real and synthetic databases, showing they can help detecting and explaining referential errors.

## 1 Introduction

Referential integrity is a fundamental global constraint in a relational database [8], that basically ensures a foreign key value exists in the referenced relation. Referential integrity issues are found in database integration, data quality assurance, data warehousing and data modeling. Referential integrity is violated or relaxed for practical reasons. Database integration represents a common scenario where similar tables coming from multiple source databases (OLTP systems) have different referential integrity constraints and each DBMS provides distinct mechanisms and rules to enforce referential integrity [20]. Therefore, source databases may violate referential integrity and their integration may uncover additional referential integrity problems. Performance is the second common reason, where referential integrity checking is disabled to allow fast insertions in batch mode. Finally, the logical data model behind a relational database evolves, incorporating new attributes and new relations not defined before, causing old data to violate new referential integrity constraints. In short, referential integrity is an important broad problem in modern relational databases.

The issues outlined above motivated us to revisit the fundamental concept of referential integrity. We propose several Quality Metrics (QMs) that measure completeness and consistency with respect to referential integrity . Our QMs not only cover normalized databases whose relations are incomplete when they have missing foreign key values, but also measure the inconsistency that arises when a relation is not normalized and an attribute value, determined by a foreign key, does not match the corresponding attribute value in the referenced relation. This is common when tables are denormalized, views are materialized or similar tables, from different source databases, are integrated. This article continues our work on referential integrity [20] (see Section 5).

The article is organized as follows. Section 2 introduces definitions on relational databases, referential integrity and denormalized databases. Section 3 presents our main contributions. We introduce QMs that

[*]© Elsevier, 2008. This is the author's version of the work. The official version of this article was published in Decision Support Systems (DSS Journal). 44(2):495-508, 2008. DOI: 10.1016/j.dss.2007.06.04

measure completeness and consistency in a relational database with respect to referential integrity. QMs quantify referential errors at the database, relation, attribute and value level, on both normalized and denormalized databases. We introduce a basic set of SQL queries to compute QMs and identify two main optimizations. Section 4 presents an experimental evaluation with real and synthetic databases. We demonstrate the usefulness of QMs on real databases and we study query optimization with large synthetic databases. Section 5 discusses related work. Section 6 presents the conclusions and directions for future research.

## 2 Definitions

We denote a relational database by $D(\mathcal{R}, I)$, where $\mathcal{R}$ is a set of $N$ relations $\mathcal{R} = \{R_1, R_2, \ldots, R_N\}$, $R_i$ is a set of tuples and $I$ a set of referential integrity constraints. A relation $R_i$ of degree $d_i$ is denoted by $R_i(A_{i1}, A_{i2}, \ldots, A_{id_i})$, where each attribute comes from some domain. One attribute from $R_i$ is the primary key (PK), called $K_i$, and the remaining attributes $A_{ij}$ are functionally dependent on $K_i$: denoted $K_i \rightarrow A_{ij}$. To simplify exposition we use simple PKs. Relations are manipulated with the standard relational algebra operators $\sigma, \Pi, \bowtie$ (i.e. with SPJ queries [7]) and aggregations. The cardinality (size) of $R_i$ is denoted by $|R_i|$ and $n_i$ (to avoid confusion with $N$).

A referential integrity constraint, belonging to $I$, between $R_i$ and $R_j$ is a statement of the form: $R_i(K) \rightarrow R_j(K)$, where $R_i$ is the referencing relation, $R_j$ is the referenced relation, $K$ is a *foreign key* (FK) in $R_i$ and $K$ is the primary key (PK) of $R_j$. To simplify exposition we assume the common attribute $K$ has the same name on both relations $R_i$ and $R_j$. In a valid database state with respect to $I$, the following two conditions hold: (1) $R_i.K$ and $R_j.K$ have the same domains. (2) for every tuple in $R_i$ there must exist a tuple in $R_j$ such that $R_i.K = R_j.K$. The primary key of a relation (i.e. $R_j.K$) is not allowed to have nulls. But in general, for practical reasons the foreign key $R_i.K$ is allowed to have nulls. We also study integrity over inclusion dependency (ID). An ID constraint between attributes $A_i$ and $A_j$ from relations $R_i$ and $R_j$ holds when $\Pi_{A_i}(R_i) \subseteq \Pi_{A_j}(R_j)$.

We relax the concept of referential integrity. Assume $D(\mathcal{R}, I)$ is an invalid state with respect to $I$. That is, some relations $R_i$ from $\mathcal{R}$ violate referential integrity constraints. We call the valid state a *strict* state. A database state where there exist referential errors is called *relaxed*. We assume $R_i$ may contain tuples having $R_i.K$ values that do not exist in $R_j.K$ or $R_i.K$ may be null ($\eta$). To make our definition more precise, if $R_i.K$ is null in some tuple we will consider such tuple incorrect. This is motivated by the fact that a null reference provides no information and the $\bowtie$ operator eliminates $R_i$ tuples with a null value on $K$. Our goal is to quantify referential integrity violations, by defining quality metrics of relaxed states.

We add one practical aspect: relations can be denormalized. We compute referential integrity metrics on attributes such that each attribute is either a *foreign key* (as defined above) or a *foreign attribute* (functionally dependent on some foreign key). We use $K$ to refer to a foreign key and $F$ for a foreign attribute, in a generic manner. Recall we assume primary and foreign keys consist of one attribute to simplify exposition, but in practice a foreign key may consist of two or more attributes (e.g. composite keys). Notice a foreign attribute introduces a potential source of inconsistency; this notion will be formalized in the next section. Relation $R_i$ has $k_i$ foreign keys and $f_i$ foreign attributes. Attributes that depend only on the primary key, and not on any foreign key, are assumed to be correct. Therefore, we do not define quality metrics on them.

## 3 Referential Integrity QMs

### 3.1 Operations violating Referential Integrity

Referential integrity can be violated basically for two reasons: (1) a relation (table) coming from a different source database is integrated into a central database, such as a data warehouse. (2) A database operation at the row level introduces a referential integrity violation when a constraint is disabled or non-existent.

This is a common scenario in a data warehouse which stores tables coming from multiple source databases. In general, referential integrity constraints are disabled with two reasons in mind: Adding new data more efficiently and avoiding the elimination of tuples with invalid references. The database operations that can cause a relaxed state are: (a) Inserting or updating a foreign key value $R_i.K$ in $R_i$, such that the $R_i.K$ is null or it does not exist in $R_j.K$ at the time of insertion or updating (completeness issue); (b) Deleting or updating tuples from $R_j$, whose primary key $R_j.K$ leaves a set of referencing tuples in $R_i$ without a corresponding matching tuple with the same $R_i.K$ value (completeness issue); (c) Updating or inserting an attribute $F$ in $R_j$, functionally dependent on $R_j.K$, that is used as a foreign attribute in $R_i$ leaving both attributes with inconsistent values or, conversely, updating $F$ in $R_i$, but not on $R_j$ (inconsistency issue). This case arises when referencing tables are denormalized, when views are materialized or when query results are stored.

There are a variety of DBMS mechanisms to enforce referential integrity [10] in databases that are assumed to be in 3NF. A common mechanism is to forbid insert or update operations on a referencing relation that have tuples with inexistent foreign keys on the referenced relation. Orthogonal referential actions associated with delete or update operations on referenced relations commonly are: restrict, cascade, nullify, set a default value or no action at all. These actions are commonly used to neutralize the possible referential integrity violation. In general, a DBMS does not enforce referential integrity on non-key attributes because databases are assumed to be in 3NF and views are materialized. Triggers [10] can be used to detect referential violations.

## 3.2 Absolute and Relative Error

Let $X$ represent an attribute (foreign key $K$ or foreign attribute $F$), a relation or a database. We define the absolute error of object $X$ as $a(X)$, given by its number of incorrect references. Let $t(X)$ be the total count of references in $X$. The relative error of $X$ is defined as $r(X) = a(X)/t(X)$.

Absolute error is useful at fine granularity levels or when there are few referential violations. In contrast, relative error is useful at coarse granularity levels or when the number of referential violations is large.

## 3.3 Hierarchical Definition of QMs

We introduce referential integrity QMs at four granularities: (1) database; (2) relation; (3) attribute; (4) value. QMs are intended to give a global picture and a micro-view of the database. Database level QMs constitute the first step towards discovering referential errors. Relation level QM isolate those relations with referential problems. Attribute QMs provide specific information that may be used to explain and fix specific referential integrity errors. QMs are hierarchically defined starting with the attribute QMs and ending with the database QMs.

Attribute and attribute value QMs are defined over foreign keys and foreign attributes. QMs over foreign keys represent a measure of *completeness*. QMs over foreign attributes represent a measure of *consistency*. Therefore, denormalization introduces potential inconsistency.

**Attribute level QMs**

Given an attribute $K$ that is a foreign key, a referencing tuple in $R_i$ is considered correct if there exists a matching tuple in the referenced relation $R_j$ with the same $K$ value. For a foreign attribute $F$ in a referencing table that is denormalized, a referencing tuple is considered correct if $R_i.F = R_j.F$ for the matching tuple in the referenced table $R_j$, where $R_i.K = R_j.K$. Otherwise, a tuple is considered incorrect. Therefore, a tuple in $R_i$ with null values in $K$ or $F$ is considered incorrect. We even consider the case that a foreign attribute is also a foreign key, where we treat it as a foreign attribute so that inconsistency can be detected.

Summarizing, referential integrity for $K$ (a foreign key), requires just an existential check, but referential integrity for $F$ (foreign attribute) also requires an equality test.

We define the attribute level metric on $R_i$ with referential integrity $R_i(K) \rightarrow R_j(K)$ on foreign key $K$. We start by defining the attribute level absolute error metric, that is defined in terms of unmatched values and nulls. If $K$ is a foreign key in $R_i$ referencing $R_j$ then

$$a(R_i.K) = |R_i| - |R_i \bowtie_K R_j| \tag{1}$$

This QM is a measure of completeness. In this case, null values and unmatched foreign keys contribute to absolute error. Motivated by the fact that sometimes foreign keys are allowed to have nulls, especially in data warehouses, we provide a less restrictive definition, where $K$ is allowed to have nulls. In this case error is computed over a subset of $R_i$, where $K$ is not null. That is, tuples where $K$ is null are assumed correct.

$$a(R_i.K) = |\sigma_{\text{notnull}(K)}(R_i)| - |R_i \bowtie_K R_j| \tag{2}$$

Let $F$ be a foreign attribute in $R_i$ dependent on a foreign key attribute $K$ and let $P$ be the primary key of $R_i$ where $R_i$ references $R_j$. That is, $R_i.P \rightarrow R_i.K$, $R_i.P \rightarrow R_i.F$ and $R_j.K \rightarrow R_j.F$. But notice $R_i.K \rightarrow R_i.F$ is not necessarily valid because $\mathcal{R}$ can be in a relaxed (inconsistent) state. We define the absolute error for a foreign attribute as:

$$a(R_i.F) = |R_i| - |\sigma_{R_i.F=R_j.F}(R_i \bowtie_K R_j)|. \tag{3}$$

This QM is a measure of consistency. Notice the metric for a foreign attribute is strict in the sense that in order to consider a reference a correct one, not only must its foreign key exist, but also have the same value and such value must be different from null. A comparison between two values in which either value is null returns null by three-valued logic [8] and therefore it increases $a(R_i.F)$. Notice that in Equation 3 we adopted a conservative strategy in the sense that if the foreign key of the corresponding foreign attribute does not match, then it will be counted as an error, no matter the value of the foreign attribute. If $F$ is also a foreign key then we treat it as a foreign attribute so that inconsistency can be detected. If we only need the number of inconsistent values from $F$ where the foreign key is valid we can compute $a(R_i.F) - a(R_i.K)$. In an analogous manner, we provide a less restrictive definition, where $K$ can be null. Notice when $F$ is null it will still be considered incorrect since comparison with any value is undefined.

$$a(R_i.F) = |\sigma_{\text{notnull}(K)}(R_i)| - |\sigma_{R_i.F=R_j.F}(R_i \bowtie_K R_j)|. \tag{4}$$

We can now define the attribute level relative error QM. Notice this metric is undefined for empty relations. If $A_{ij}$ is either a foreign key or a foreign attribute in $R_i$ then

$$r(R_i.A_{ij}) = \frac{a(R_i.A_{ij})}{n_i} \tag{5}$$

The QM $r(R_i.K)$ over a foreign key $K$ is a measure of completeness and $r(R_i.F)$ on a foreign attribute $F$ is a measure of consistency.

**Value level QMs**

We define a value level QM for one foreign key value $k_p \in R_i.K$ as $a(R_i.K, k_p) = |\sigma_{K=k_p}(R_i)| - |\sigma_{K=k_p}(R_i \bowtie_K R_j)|$, for each key value $k_p$, where $k_p$ may be null. This QM provides the frequency of invalid key values, including null. This QM is a measure of completeness. The value QM for each foreign attribute value $f_q \in R_i.F$ is similarly defined counting those tuples that contain values $k_p \in K$ and $f_q \in F$ in $R_i$ and $R_i \bowtie_K R_j$. This QM measures consistency and provides the frequency of each non-matching foreign attribute value. If nulls in $K$ are considered correct then $f_q$ will be considered correct if it is null. When nulls are not considered correct a null value in $K$ will cause $f_q$ to be considered incorrect, regardless of null. Therefore, incompleteness in $K$ leads to inconsistency in $F$. Relative error for attribute values is defined dividing by $n_i = |R_i|$.

### Statistics and Correlations on Attribute QMs

We introduce univariate and bivariate statistics to understand the probability distribution behind referential errors in some attribute and also, to discover interrelationships among two unrelated attributes. We calculate univariate statistics on attribute QMs including the minimum, maximum, mean and standard deviation of absolute error (frequency) of invalid attribute values. Statistics on QMs are useful to explain why errors happen and to develop a database repair plan. As we shall see in Section 4, these statistics can give us specific information on the probability distribution behind invalid values in a relation. For instance, we can know how many tuples there are per invalid value on average. To have an idea about the scatter of frequencies of invalid values we compare their mean and standard deviation. The minimum and maximum frequencies help us detect critical values (outliers). We also introduce an error correlation measure between two attributes on the same relation. In general, the correlation between a foreign key and a functionally dependent foreign attribute is close to 1. Error correlation can reveal interesting facts among two attributes that do not have any functional dependency between them. A correlation between two unrelated foreign keys or two unrelated foreign attributes provides valuable insight to explain why referential errors happen.

### Relation level QMs

We define relation QMs for table $R_i$ with $k_i$ foreign keys and $f_i$ foreign attributes. When $R_i$ is normalized $f_i = 0$.

Based on attribute QMs we define independent error aggregations on foreign keys and foreign attributes. The following QM measures the completeness of $R_i$:

$$a_K(R_i) = \sum_{j=1}^{k_i} a(R_i.K_j) \tag{6}$$

This QM measures completeness since it tells us how many foreign key values are not matched in each $R_j$ or are simply null. Similarly, we define a consistency QM for $R_i$ as:

$$a_F(R_i) = \sum_{j=1}^{f_i} a(R_i.F_j) \tag{7}$$

We define relative error QMs for completeness and consistency, respectively:

$$r_K(R_i) = \frac{a_K(R_i)}{k_i n_i} \tag{8}$$

$$r_F(R_i) = \frac{a_F(R_i)}{f_i n_i} \tag{9}$$

$r_K()$ and $r_F()$ are $\eta$ (undefined) when $k_i = 0$, $f_i = 0$ or $n_i = 0$. Also, $a_F(R_i)$ is $\eta$ when $R_i$ is normalized. Observe that relation level QMs quantify completeness and consistency in a relation based on all its attributes.

| Data Quality dimension | absolute error | relative error |
|---|---|---|
| Completeness | $a(R_i.K)$ | $r(R_i.K)$ |
| Consistency | $a(R_i.F)$ | $r(R_i.F)$ |

Table 1: Attribute level QMs.

| $\mathcal{R}$ | $N$ | $r_K()$ | $r_F()$ |
|---|---|---|---|
| storeDB | 14 | 0.02% | 0.02% |

Table 2: Database level QMs

**Database level QMs**

We define database absolute error QMs, for completeness and consistency respectively.

$$a_K(\mathcal{R}) = \sum_{i=1}^{N} a_K(R_i) \tag{10}$$

$$a_F(\mathcal{R}) = \sum_{i=1}^{N} a_F(R_i) \tag{11}$$

Finally, we define global relative error QMs.

$$r_K(\mathcal{R}) = \frac{a_K(\mathcal{R})}{\sum_{i=1}^{N} k_i n_i} \tag{12}$$

$$r_F(\mathcal{R}) = \frac{a_F(\mathcal{R})}{\sum_{i=1}^{N} f_i n_i} \tag{13}$$

Our database level QMs exclude relations $R_i$ with $k_i = 0$ (e.g. lookup relations without references). If $k_i = 0$ for some $R_i$ then $R_i$ does not contribute to relative error.

## Discussion on QMs

Completeness and consistency are measured separately. In our proposal we do not give different weights to references coming from large or small relations. An incorrect reference in a small relation (e.g. a lookup table) is as important as an incorrect reference in a large relation (e.g. a transaction table). Correlations and further multidimensional statistical models can be used to explain relationships between referential errors in different attributes. QMs can be ranked by the cardinality of the referencing relation so that references in smaller or larger relations carry more or less weight based on user's preferences. Table 1 summarizes attribute-level QMs, which can be thought as the atomic metrics from which relation and database QMs are aggregated. Value QMs can be considered a zoomed view of attribute QMs. Figure 1 shows the hierarchical relationship among QMs; everything is derived from attribute level QMs.

## 3.4 Example

We now present examples of QMs using a store database schema. Since the goal of our QMs is to help a user discover and explain referential errors in a database, we present QMs going from the highest level

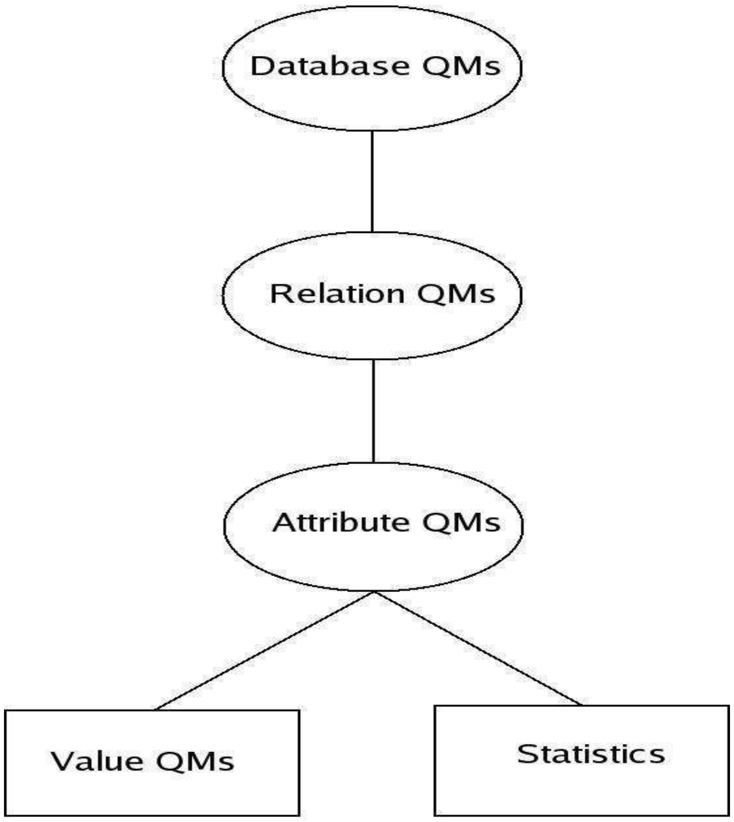

Figure 1: QMs diagram.

| $R_i$ | $k_i$ | $f_i$ | $n_i$ | $a_K()$ | $a_F()$ |
|---|---|---|---|---|---|
| categ | 0 | 0 | 20 | $\eta$ | $\eta$ |
| product | 4 | 5 | 100109 | 2444 | 3100 |
| txHeader | 2 | 2 | 2003569 | 0 | 220 |
| txLine | 4 | 2 | 19130964 | 7200 | 7495 |

Table 3: Relation level QMs

| $R_i$ | $A_{ij}$ | $K$ or $F$ | $a()$ | $r()$ |
|---|---|---|---|---|
| categ | categName | $\emptyset$ | 0 | 0.00% |
| city | stateId | $K$ | 12 | 9.01% |
| product | categId | $K$ | 2444 | 2.00% |
| product | categName | $F$ | 3100 | 3.00% |
| txLine | prodId | $K$ | 7200 | 0.02% |
| txLine | prodPrice | $F$ | 7495 | 0.02% |

Table 4: Attribute level QMs.

| $R_i$ | $stateId$ | $a(R_i.K)$ |
|---|---|---:|
| store | $\eta$ | 4 |
| store | 0 | 3 |
| store | blank | 2 |
| store | T. | 2 |
| store | XX | 1 |

Table 5: Value level QMs for foreign key $K$.

| $R_i$ | $cityId$ | $cityName$ | $a(R_i.K, R_i.F)$ |
|---|---|---|---:|
| store | LA | $\eta$ | 43 |
| store | SF | blank | 34 |
| store | SF | Los Angeles | 1 |
| store | SF | $\eta$ | 67 |
| store | $\eta$ | New York | 1 |

Table 6: Value level QMs for foreign attribute $F$.

down to the most detailed level, in opposite order from their definition. We start showing global measures of quality of referential integrity at the database level in Table 2. The first important observation is that $r_K()$ and $r_F()$ are not zero, which indicates there exist referential errors in the database. The second observation is that incompleteness and inconsistency have similar importance. If it is desired to maintain a database in a strict state this QM table can be periodically refreshed and monitored, by a set of SQL queries, as discussed above. Going down one level we can discover which specific relations are giving us problems. Table 3 helps us identify those relations with errors. For each relation there are completeness and consistency QMs with $a(K)$ and $a(F)$. Relation *product* has a mixture of completeness and consistency problems. Relation *txHeader* has only consistency problems in its foreign attributes (e.g. *customerId*). Relation *txLine* reveals completeness and consistency problems: some foreign keys have invalid values or referenced relations are incomplete.

The next level gives us more specific evidence about referential problems, as shown in Table 4. To make output more concise we only show absolute error, but relative error can be easily derived. Relation *categ* has no references, and it has attribute *categName* which is neither $K$ nor $F$; therefore, this attribute does not contribute to absolute/relative error. Relation *city* indicates there exists a serious completeness problem in the database with a high fraction of invalid foreign keys. Relation *product* indicates denormalization introduces important consistency problems since it has many inconsistent values in *categName*. In relation *txLine* we can see that the main source of referential problems are invalid foreign keys, but we can see there are 7495-7200=295 rows with inconsistent values for *prodPrice* even when the foreign key exists. Finally, at the finest storage granularity level, Tables 5 and 6 show value level QMs, exhibiting unmatched $K$ values and inconsistent $F$ values in a denormalized relation, as well as their respective frequencies. Values are sorted in descending order by their absolute error to quickly identify critical values in order to develop a database repair plan. In other words, when $a(K)$ or $a(F)$ have skewed distributions we can learn which values can help us fix most errors. Value level metrics for $F$ consider every existing combination between $K$ and $F$, which can help explain why denormalized values do not match and assess their relative importance. In this case we can see that the functional dependency between $K$ and $F$ does not hold. When a database is in a *strict* state, these value level tables are empty. Another important aspect is that these QM tables represent extended statistics on the database metadata. Therefore, QM tables can be queried by an end-user to explore a database for completeness and consistency.

## 3.5 QMs Calculation with SQL

Absolute error metrics are distributive [13] based on the fact that they are counts. Therefore, from a query evaluation perspective, most of the effort is spent on computing attribute level or value level QMs. But since relative error is a quotient it cannot be used to compute relative error at coarser granularity levels. This is a consequence of the relative QM being an algebraic [13] function.

We present two query optimizations that are particularly useful to compute QMs. The first optimization favors a left outer join over a set containment computation to compute absolute QMs. The second optimization evaluates error aggregations grouping by foreign keys either before or after joining $R_i$ and referenced relations $R_j$.

### Set containment and left outer join

The following query computes $a(R_i.K)$ as defined above, with the constraint $R_i(K) \to R_j(K)$, assuming the primary key of $R_i$ is $P$. We call this query "set containment". This query is generally computed with a nested loop algorithm.

SELECT count(*) FROM $R_i$ WHERE $R_i.K_i$ NOT IN
(SELECT $R_i.K_i$ FROM $R_i$ JOIN $R_j$ ON $R_i.K = R_j.K$)

We introduce an equivalent query to compute $a(R_i.K)$ that may be more efficient, when using a different join algorithm, like a merge-sort join. We call it "left outer join".

SELECT count(*)
FROM $R_i$ LEFT OUTER JOIN $R_j$ ON $R_i.K = R_j.K$
WHERE $R_j.K$ is null

A similar framework can be used for foreign attributes. The following query computes $a(R_i.K)$ and $a(R_i.F)$ in a single table scan over $R_i$. In general, the left outer join optimization will be applied by default.

SELECT
 sum(case when $R_j.K$ is null then 1 end) AS $a_K$
,sum(case when $R_j.K$ is null or $R_i.F <> R_j.F$ then 1 end)
 AS $a_F$
FROM $R_i$ LEFT OUTER JOIN $R_j$ ON $R_i.K = R_j.K$;

### Early or late foreign key grouping

This optimization evaluates a "group-by" clause by foreign keys either before or after joining $R_i$ and referenced relations $R_j$. The rationale behind this optimization is reducing the size of $R_i$ before joining with $R_j$. We also call this optimization the "cube" variant because we build a cube whose dimensions are foreign keys before joining. The early foreign key grouping query optimization with $R_i$ and $R_j$ computes $a(R_i.K)$ as follows:

INSERT INTO $RK$ SELECT $K$,count(*) AS $a$
FROM $R_i$ GROUP BY $K$;
SELECT sum($a$) FROM $RK$ LEFT OUTER JOIN $R_j$
ON $RK.K = R_j.K$ WHERE $R_j.K$ is null;

```
Input: $\mathcal{R}$
Output: attribute, relation and database QMs, stored in tabular form
1. Create and initialize output tables.
2. Compute attribute level QMs.
  FOR $i = 1$ TO $N$ DO /* $R_i$: referencing relation */
    Compute $R_i \bowtie_K R_j$ s.t. $i \neq j$ /* $R_i$ references $R_j$ */
    IF $k_i = 0$ THEN $a(R_i) = \eta$ /* $R_i$: zero references */
    ELSE /* $k_i \neq 0$ */
      FOR $J = 1$ TO $k_i$ DO
        Compute $a(R_i.A_{iJ})$ using Equation 1 and Eq. 3.
        Compute $r(R_i, A_{iJ})$ using Equation 5.
        Compute statistics on QMs
      END
    END
  END
3. Compute relation level QMs with Equations 6, 7, 8, 9.
4. Compute database level QMs with Equations 10, 11, 12, 13.
```

Figure 2: Algorithm to compute QMs.

On the other hand, the late group-by evaluation on $R_i$ and $R_j$ to get $a(R_i.K)$ is shown below. The derived table $T$ can be materialized to efficiently query value level QMs.

```
INSERT INTO $Rij$   SELECT $R_i.K$
 FROM $R_i$ LEFT OUTER JOIN $R_j$
ON $R_i.K = R_j.K$ WHERE $R_j.K$ is null;
SELECT sum($a$) FROM (
SELECT $K$,count(*) AS $a$ FROM $Rij$ GROUP BY $K$ )$T$;
```

The early foreign key grouping optimization can be generalized to $R_i$ being joined with $m$ relations $R_1, R_2, \ldots, R_m$ on foreign keys, $K_1, K_2, \ldots, K_m$, respectively.

### 3.6 Algorithm to get QMs

We represent referential integrity constraints by a directed graph $G = (V, E)$, where $V = \{1, \ldots, N\}$ and vertex $i$ corresponds to $R_i$ and $E = \{(i, j)|R_i(K) \to R_j(K)\}$. That is, each directed edge represents one reference. We introduce an algorithm that computes referential integrity QMs based on $G$. The algorithm traverses the logical data model behind the database, moving from referenced relations towards referencing relations, as specified by $G$. Each vertex $i$ corresponding to relation $R_i$, stores information about its primary key, foreign keys, foreign attributes and functional dependencies. Each edge (reference) makes the algorithm create an SQL query between $R_i$ and $R_j$. The algorithm produces metadata tables with QMs at different levels. The algorithm is shown in Figure 2.

## 4 Experimental evaluation

We implemented our program in the Java language that connected to different relational DBMSs through the JDBC interface. The program received $G$ (the referential integrity directed graph) as input, and generated SQL statements to create the four QM tables, one for each granularity level. We used three real databases

| $R_i$ | $K$ | $r_K$ | time |
|---|---|---|---|
| student | studentId | 19.0% | 707 |

Table 7: Educational institution: attribute level QMs.

| $R_i$ | $A_{ij}$ | min | $\mu$ | max | $\sigma$ |
|---|---|---|---|---|---|
| student | studentId | 1 | 3 | 9 | 1.52 |

Table 8: Educational institution: attribute QM statistics.

and one synthetic database, generated with the TPC-H DBGEN program. All times are reported in seconds, unless stated otherwise.

We used multiple relational DBMSs, from different software companies. To avoid discussion on each DBMS specific features, performance and SQL compliance, we omit their real name. This was also a request from end users to protect their privacy, while allowing us to report our findings. We used standard SQL (ANSI) in our QMs implementation, making it portable in all relational DBMSs.

## 4.1 Real Databases

Real databases were used to assess the usefulness of our approach and the relative importance of QMs. In the experiments presented below we asked several information technology organizations to give us permission to discover referential integrity problems in their databases. The real databases came from a university, a government organization and a retail company. In our discussion we change the actual table and column names to protect privacy. Organizations did not agree to reveal attribute value QMs since they revealed specific information about their operation. In general, query running times are given in seconds.

### Educational Institution

We now present interesting results on a database from a public higher educational institution. Since this was a production environment, other database processes were running concurrently with our program, but the workload was similar in all cases; we report the average running time. Due to security restrictions we could only explore one important historic table containing important student information, including student name, year, semester, course names, course grades, credits and grade point average. We were not allowed to explore value level QMs, showing specific referential errors for specific students. The experiment goal was to validate that student identifications were actually valid, according to a reference table containing biographic and demographic information for all students ever registered at the institution. Results were discussed with the Information Technology manager, two database developers and a system administrator. Results are presented in Table 7. The IT manager was aware there existed some referential integrity issues. Database developers were surprised there was such a high fraction of missing foreign keys for student identification numbers. The system administrator stated the historic table was constantly updated with new semester course grades, but he said this table had a few changes in its definition in the past. Table 8 gives

| $\mathcal{R}$ | $N$ | $r_K$ | $r_F$ |
|---|---|---|---|
| GovtDB | 23 | 0.43% | 0.01% |

Table 9: Government database; database level QMs.

| $R_i$ | $k_i$ | $n_i$ | $a$ | $r$ |
|---|---|---|---|---|
| docum | 3 | 4779940 | 2081465 | 14.52% |
| genId | 3 | 2493855 | 15198 | 0.14% |

Table 10: Government database; relation level QMs.

| $R_i$ | $A_{ij}$ | $KorF$ | $a$ | $r$ |
|---|---|---|---|---|
| docum | county | $K$ | 2076530 | 43.443% |
| docum | citizen | $K$ | 4935 | 0.103% |
| docum | paymts | $K$ | 0 | 0.000% |
| genId | brName | $F$ | 5628 | 0.226% |
| genId | model | $F$ | 4562 | 0.183% |

Table 11: Government database; attribute level QMs.

statistical information about the probabilistic distribution of invalid references. It is noteworthy that there exists a value that contributes significantly to $a()$ with 9 invalid references. The coefficient of variation $\sigma/\mu$ states there is not much variation around $\mu$: most values contribute equally to $a()$.

**Government Organization**

Applying our approach on a different database, we now present QMs on a driver's license database from a state in Mexico (state name omitted due to privacy restrictions). This database contained driver's license information and vehicle registration information for people living in certain counties. The database had historic tables containing historic personal information, driving records, traffic fines and payments. There were additional reference tables containing personal identifiers issued by government, similar to the US Social Security Number. These results were obtained during an early stage of our project, where we did not compute all QMs. Results were discussed with the IT manager, a database applications developer and a database administrator. The database level QMs are shown on Table 9, where we can see there are minor referential integrity problems. Incompleteness of foreign keys is clearly more important than consistency, even though the database is denormalized. This was good news for the IT manager, who did not anticipate having any important referential issues. Going down one level, Table 10 shows a couple of relation level QMs. Users became aware referential problems were prevalent specifically in relation *License*, which was the most important relation being continuously updated. To a lesser extent, there were both completeness and consistency problems in relation *person*. Then going to a more granular storage level, attribute level metrics are shown in Table 11. QMs $a()$ and $r()$ are significantly high for the *county* attribute. On the other hand, our prototype uncovered inconsistency problems in attributes *brName* and *model*. Before computing

| Attribute | $K/F$ | $a()$ | $r()$ |
|---|---|---|---|
| storeId | $K$ | 9468 | 7.56% |
| format | $F$ | 9672 | 7.72% |
| region | $F$ | 15036 | 12.01% |

Table 12: Retail database; attribute-level QMs.

| $\rho$ | $a(storeId)$ | $a(format)$ | $a(region)$ |
|---|---|---|---|
| $a(storeId)$ | 1.000 | | |
| $a(format)$ | 0.988 | 1.000 | |
| $a(region)$ | 0.774 | 0.764 | 1.000 |

Table 13: Retail database; $a()$ error correlation.

our QMs, users had told us there could be referential violations. Nevertheless, they had not imagined there were serious problems in many cases.

**Retail Company**

We present error correlation results with a database from a retail company. We focused on a summary fact table, having $n_i = 125,208$ rows, used to perform store-level data mining analysis and to build monthly and annual OLAP reports. This fact table had been built from a database containing 6 billion transactions. From an analytical perspective this was one of the most important tables to perform OLAP and data mining analysis for this company. We ran our QMs and they took just 10 seconds; the correlation matrix was derived in 2 seconds. Table 12 shows attribute-level QMs only for those attributes with non-zero error. Clearly, completeness is a major issue since more than 7% of rows have an invalid FK for *storeId*. Looking at foreign attributes we can see $a(format)$ is close to $a(storeId)$ which indicates we cannot tell if $format$ is consistent or not with the value in the referenced relation; given our conservative definition we assume it is incorrect. On the other hand, $a(region)$ is far from $a(storeId)$, which reveals a serious consistency problem. In fact, 15036-9468=5568 values are inconsistent despite the fact that the FK *storeId* exists. Table 13 shows a high correlation between *storeId* and *format*, which is a consequence of the functional dependency. However, *region* shows a lower correlation to either attribute, which tells us that this attribute shows inconsistency in an independent manner. In practical terms, repairing referential errors in *storeId* takes care of *format*, but *region* requires a separate (independent) repair action.

**Users Feedback Summary**

In general, users expected their databases to be clean and they asked us to keep their specific organization names and specific record information confidential. In some cases, they also requested to keep the DBMS brand confidential as well. We requested getting access to critical databases that were updated and refreshed continuously. It did not matter if such databases were denormalized since our program was prepared to handle them. In particular, we focused on analyzing large historic tables whenever possible since those tables are used in a wide variety of queries and reports. Under the IT manager supervision we were allowed to compute our QMs with automatically generated SQL queries. When the referential integrity prototype had generated results, these results were discussed with the IT managers, database developers and database administrators. We presented results hierarchically going from the database level QMs down to attribute level QMs. Based on findings, users requested to browse a sample of records with referential errors. Some users were intrigued by results and asked us to explain how our QMs were computed with SQL queries.

We asked the IT managers the following questions. Are you surprised QMs indeed uncovered some referential problems?, what level of granularity of QMs would you compute on a frequent basis?, which is a more critical dimension in your database: completeness or consistency?, for which tables is it critical to maintain referential integrity? We asked database developers and database administrators the following questions. Why do you think table X has referential errors?, when table X is denormalized, how is it computed?, how frequently is database X updated?, is table X updated in batch or continuously, inserting one

| QM level | usefulness | application | user |
|---|---|---|---|
| database | low | detection | IT manager |
| relation | medium | detection | IT manager/DBA |
| attribute | high | diagnosis | DBA/developer |
| value | high | diagnosis/repair | DBA/developer |

Table 14: Users feedback.

record at a time?, is there an explanation for attribute Y to have high levels of referential errors, compared to other attributes?

In general, users feedback about our tool was positive. Users stated that QMs helped them to discover unexpected referential integrity problems and to ensure data quality policies and procedures were working properly. Since most tables were normalized, QMs on foreign keys were more interesting. Database level QMs were not particularly useful on the three real databases because relative error was high. Attribute level QMs on foreign keys (completeness) were particularly useful in OLTP systems or isolated databases in which referential integrity was routinely enforced. That is, completeness was more important for databases where records were inserted by transactions. Attribute level QMs on foreign attributes (consistency) were valuable to identify stale records in large fact tables and to detect inconsistent attributes in denormalized tables used for data mining purposes. That is, consistency was more important for a data warehouse where there exists a large fact table with historic information and where there are denormalized tables computing aggregations from the fact table. Database and relation level QMs helped detecting unforeseen referential integrity issues. Attribute and value level QMs helped diagnosing (explaining) referential errors and preparing a repair plan. Users feedback is summarized in Table 14.

We now present a more detailed discussion of users comments. A system manager stated that QMs revealed problems he was not aware of, and another IT manager stated that QMs helped him obtain a clear idea about data quality. A DBA for a data warehouse stated that QMs could enrich metadata to test data loading scripts in order to detect problems while tables are being refreshed. An IT manager said QMs could justify a plan to improve data quality when integrating databases into a central data warehouse. QMs provided varied usefulness depending on each type of user. QMs at the relation and database levels were more useful for IT managers since they give a high level referential integrity quality assessment. QMs at the attribute and value level were more useful for database application developers, who suggested integrating QMs with testing database application code. FK value level QMs (invalid FK frequencies) were interesting to all users (e.g. a FK appearing in many relations) because they provided evidence about problems, but users stated they preferred to run the prototype on their own. An IT manager and a DBA stated that QMs should be collected over a long period of time (e.g. every week), especially for large historic tables containing transaction data, to track data quality and prevent future data quality problems. Users feedback is summarized in Table 14.

## 4.2 Synthetic Databases

We conducted our experiments on a database server with one Intel Xeon CPU at 1 GHz with 256 MB of main memory and 108 GB on disk. The relational DBMS we used was Postgres V8.0.1.

Our synthetic databases were generated by the TPC-H DBGEN program, with scaling factors 1 and 2. We inserted referential integrity errors in the referencing fact table ($lineitem$) with different relative errors ($0.1\%$, $0.2\%,\ldots$, $1\%$, $2\%,\ldots$, $10\%$). The invalid values were inserted following several different probability distribution functions (pdfs) including geometric, uniform, zipf and normal, and in three foreign keys ($l\_orderkey$, $l\_partkey$ and $l\_suppkey$).

| pdf | function | Parameters |
|---|---|---|
| Uniform | $\frac{1}{h}$ | $h = 6000$ |
| Zipf | $\frac{1/k^s}{H_{M,s}}$ | $M = 40000$<br>$s = 1$ |
| Geometric | $(1-p)^{n-1}p$ | $p = 1/2$ |
| Normal | $\frac{1}{\sqrt{2\pi\sigma^2}}\ \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ | $\mu = 3000$<br>$\sigma^2 = 1000$ |

Table 15: Probability distributions used to insert invalid values.

| $n_i$ | Left outer join | Set containment |
|---:|---:|---:|
| 10000 | 25 | 60 |
| 20000 | 25 | 240 |
| 30000 | 29 | 600 |
| 50000 | 35 | 1560 |
| 100000 | 37 | 6120 |

Table 16: Query optimization: left outer join and set containment.

The results we present in this section, unless stated otherwise, use a default scale factor 1. The referencing table, $lineitem$ and the referenced tables, $orders$, $part$ and $supplier$ have the following sizes: 6M, 1.5M, 200k and 10k tuples, respectively. Invalid FK values were randomly inserted according to four different pdfs, as shown in Table 15. The minimum number of referential errors was approximately 6,000 and the maximum was 600,000. Elapsed times are indicated in seconds.

**Query Optimizations**

First, we evaluated the left outer join optimization on foreign keys, summarized in Table 16. Performance degrades significantly for the set containment query, as the cardinality of the referencing relation increases. On the other hand, it is noteworthy time grows more slowly for the left outer join query. To explain why set containment queries are slower than left outer join queries, we obtained the query plans for both types of queries on two DBMSs. We found that the query optimizer in DBMS X first produced a sequential scan for the nested subquery and then a nested loop join to determine the negated set containment. The optimizer from DBMS Y produced a nested loop join for the same query, which was more efficient. On the other hand, the left outer join was generally computed with a merge sort or hash join. These results supported using a left outer join by default.

We compared the performance between the late and the early foreign key grouping optimization for small groups of invalid foreign key values. If the number of distinct values in the foreign key was similar to the relation cardinality (i.e. large) applying early foreign key grouping was counterproductive. But if the number of distinct values on foreign keys was smaller, then this optimization produced a significant speedup. In Table 17 we present performance for different cardinalities, considering foreign key groups of size 4 and 5 (meaning each invalid value appeared in 4 or 5 tuples on average). Times become better for group size 5. Small referenced relations or large referencing relations having few distinct FK values make early foreign key grouping an essential optimization, since it significantly reduces the size of the relation to

| $n_i$ | Late | Early FK grouping | |
|---|---|---|---|
| | FK group | Size 4 | Size 5 |
| 1200000 | 54 | 67 | 56 |
| 2500000 | 91 | 136 | 85 |
| 5000000 | 165 | 172 | 134 |

Table 17: Query optimization: early and late foreign key grouping.

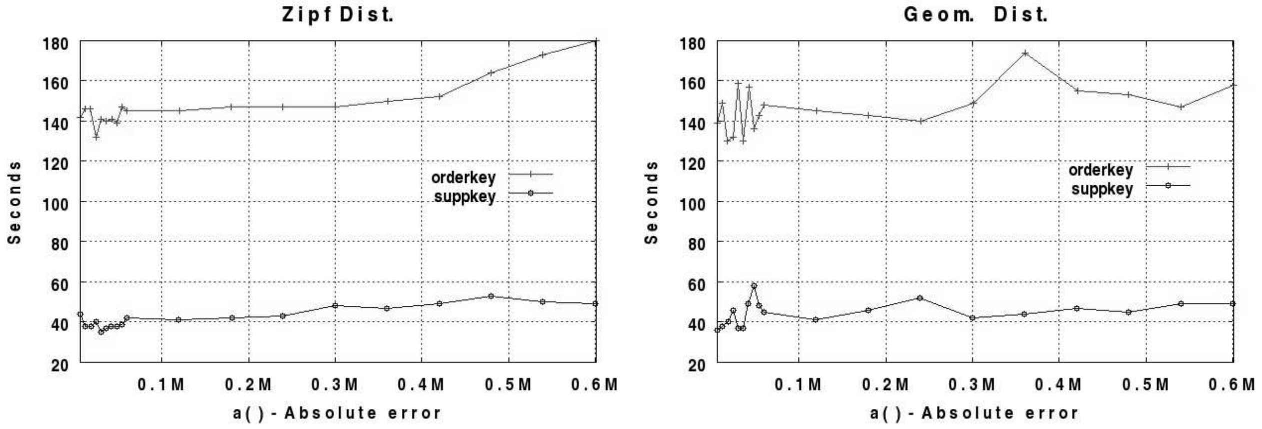

Figure 3: Early FK grouping variant with two pdfs.

be joined with all its referenced relations.

Figure 3 analyzes the impact of referential errors on time performance with two representative pdfs. We can see $a()$ has a marginal impact on performance as it increases. The impact is more important for the zipf distribution. Summarizing, the size of a relation is far more important than the number of invalid values.

## Statistics on QMs

Table 18 shows a summary of statistics of QMs with a fixed $r() = 1\%$ for the *lineItem* table and the *partId* column. Observe that for the geometric pdf one invalid value produces many referential errors. On the other hand, for the uniform pdf all invalid values contribute evenly to referential errors. The geometric pdf has the highest standard deviation, meaning there are values far from the mean, that significantly contribute to error.

| pdf | min | $\mu$ | max | $\sigma$ |
|---|---|---|---|---|
| Uniform | 1 | 10 | 22 | 3 |
| Zipf | 1 | 7 | 6002 | 82 |
| Geometric | 1 | 3505 | 29873 | 7825 |
| Normal | 1 | 120 | 372 | 121 |

Table 18: Univariate statistics with different probability distributions.

## 5 Related Work

There is extensive work on maintaining referential integrity. For instance, [17, 19] identify conditions to avoid referential problems during data manipulation. Reference [14] presents a model for referential integrity maintenance during run-time execution. There is a proposal to check deferred referential integrity using a metadata network [5], where inclusion dependencies are considered and each foreign key value is verified to have a corresponding matching key value. No analysis is done on measuring inconsistency, nor on algorithm performance. Implementation of update and delete referential actions, full and partial types, reference types, are features that have not been fully completed in commercial DBMSs [24].

The SQL language has supported referential integrity by defining foreign keys and referential actions. A survey on SQL query optimization is given in [7], explaining when to perform a "group-by" operation before a join operation (early or eager), instead of joining tables first and then performing the "group-by" operation (late or lazy). This optimization is applicable when the number of result groups is small and when a different evaluation order does not change result correctness. Our early foreign key grouping is a particular case of this optimization, but we also consider null key values and we generalize it to build a cube of foreign keys. SQL aggregations are extended to return approximately consistent answer sets when there exist invalid FKs [20]; this approach dynamically detects referential errors and improves answer sets in two ways: (1) by distributing aggregated values in a measure attribute from invalid FK values among valid FK values; (2) by exploiting valid FK values in another attribute to make distribution more accurate.

To our knowledge, data quality metrics have not been proposed with respect to referential integrity. A proposal of simple metrics for data quality in relational databases is given in [18], where completeness and soundness metrics are introduced. These metrics compare the state of the database to the data from the real world such database is supposed to represent. Our proposed QMs measure the completeness of references among relations and to some extent QMs on foreign attributes measure soundness. We do not deal with the problem of measuring if a valid reference is indeed valid. Concerning this aspect, [25] introduces an attribute-based model that incorporates data quality indicators and focuses on two dimensions: interpretability and believability. In [22] the authors introduce an algebra to measure completeness in several data quality dimensions, considering basic relational set operations, null values and the open/closed world assumptions, but ignoring referential integrity.

A model linking data quality assessment to user requirements is proposed in [6]. In [21] a classification of objective quality measures is presented. In particular, their simple ratio function measures the ratio of desired outcomes to total outcomes. The authors suggest that consistency could be measured by these means. This is related to our referential integrity error definitions. In [16] the authors investigate the correct integration of relationship instances integrated from different source databases, focusing in detecting semantic conflicts and reconciling them. This proposal can be used as pre-processing to get QMs, since semantic conflicts must be solved before quantifying referential integrity errors. Referential integrity metrics have been studied considering the model design point of view. Reference [4] introduces two metrics to aid model designers make better decisions by analyzing referential paths and the number of foreign keys in a schema. The authors do not consider invalid keys or denormalized relations. In contrast, our approach is applicable after the database logical data model has evolved from its original design or for database integration. Authors in [23] introduce metrics to evaluate the effectiveness of conceptual models for a data warehouse, which complement logical and physical design tools.

We now explain our approach from a broader perspective. Data quality problems related to referential integrity are described in [15]. The authors distinguish between operational and diagnostic data quality metrics. In this work referential violations between two tables are called poor join paths. Our metrics at the database and relation levels are diagnostic and frequencies of invalid foreign key values are operational since they provide insight into how to fix referential errors.

Healthcare data warehouses represent a prominent example, where data quality (missing information, inconsistency), data integration from diverse sources (structured and semistructured) and privacy (confidentiality of patient records) make database management difficult [2]; several of such issues are related to referential integrity maintenance among tables coming from different sources. On a closely related topic, [1, 11] study data quality issues in data warehouses considering the time dimension for aggregate queries; in our work we simply detect referential integrity errors on the current state of the database, but we do not track when referential errors were introduced. Therefore, discovering and explaining referential errors back in time is an interesting issue for future work.

Discovering database relationships and repairing inconsistent databases are important related problems. In [9] the authors propose techniques to automatically identify PK/FK relationships between tables coming from different databases; in our case we assume such relationships are manually specified by the user or come from a logical data model. Therefore, this approach can be applied as a pre-processing phase before computing referential integrity QMs. In [3] the authors introduce a cost framework to aid in the restoration of a database with integrity constraints (user-defined rather than referential) violations via value modifications. Constraints in an inconsistent database can be satisfied by incorporating constraint checks in equivalent rewritten queries to given queries [12]. Both approaches apply heuristics to repair databases, either statically (by updating tables) or dynamically (by rewriting queries). In contrast, our work diagnoses problems and gives detailed information to the user in order to explain and repair referential errors. Therefore, our proposal can serve as a pre-processing step before applying any heuristic to repair invalid values.

## 6 Conclusions

We proposed a comprehensive set of quality metrics (QMs) for referential integrity, which can be applied in data warehousing, database integration and data quality assurance. Our QMs measure completeness in the case of foreign keys and consistency in the case of foreign attributes in denormalized databases. QMs are hierarchically defined at four granularities: database, relation, attribute and attribute value. Quality metrics are of two basic types: absolute and relative error. Absolute error is useful at fine granularity levels or when there are few referential violations. Relative error is adequate at coarser granularity levels or when the number of referential violations is relatively large. We introduced univariate and bivariate statistics on attribute level QMs to further understand the probability distribution of invalid foreign key values. We explained how to efficiently calculate QMs with SQL queries. Specifically, we presented two query optimizations. The first optimization favors a left outer join over a set containment to use a hash or merge-sort join algorithm instead of a nested loop algorithm. The second optimization performs a group-by operation on foreign keys before a join (pushing aggregation, early group-by) to reduce the size of the referencing relation. This optimization is effective for large relations with many foreign keys, where the number of distinct values per foreign key is small. Experiments evaluate referential integrity QMs with real and synthetic databases on different DBMSs. We got interesting results on real databases and end-users opinion was positive. QMs at the database and relation level were more useful for managers, whereas value and attribute level QMs were more interesting to DBAs and application programmers. We studied quality metrics for attributes following four different probability distributions. Attribute values with a high relative error in skewed distributions can be used to fix critical referential problems. Univariate statistics and correlations can help understand the probability distribution and co-occurrence of referential errors. On the time performance side, a left outer join evaluation was generally more efficient than a set containment due to fast join evaluation algorithms in different DBMSs. On the other hand, early foreign key grouping was always more efficient than late foreign key grouping.

Our work can be extended to repair a database considering the frequency of matching values for foreign keys and foreign attributes. Our statistics on attribute level metrics can be extended to apply multidimensional data mining techniques, such as clustering and factor analysis. We want to study how to efficiently repair a denormalized database to leave it in a strict state, based on a plan derived from quality metrics. Incremental computation is needed to keep quality metrics up to date in ever-growing data warehouses.

Finally, we believe our ideas may be applicable in semistructured data, such as text or XML.

## References

[1] D.J. Berndt and J.W. Fisher. Understanding dimension volatility in data warehouses. In *INFORMS CIST Conference*, 2001.

[2] D.J. Berndt, J.W. Fisher, and J. Studnicki A.R. Hevner. Healthcare data warehousing and quality assurance. *IEEE Computer*, 34(12):56–65, 2001.

[3] P. Bohannon, W. Fan, M. Flaster, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *ACM SIGMOD Conference*, pages 143–154, 2005.

[4] C. Calero, M. Piattini, and M. Genero. Empirical validation of referential integrity metrics. *Information & Software Technology*, 43(15):949–957, 2001.

[5] S.J. Cammarata, P. Ramachandra, and D. Shane. Extending a relational database with deferred referential integrity checking and intelligent joins. In *ACM SIGMOD Conference*, pages 88–97, 1989.

[6] C. Cappiello, C. Francalanci, and B. Pernici. Data quality assessment from the users perspective. In *Proc. ACM IQIS Workshop*, pages 68–73, 2004.

[7] S. Chaudhuri. An overview of query optimization in relational systems. In *Proc. ACM PODS Conference*, pages 84–93, 1998.

[8] E.F. Codd. Extending the database relational model to capture more meaning. *ACM TODS*, 4(4):397–434, 1979.

[9] T. Dasu, T. Johnson, S. Muthukrishnan, and V. Shkapenyuk. Mining database structure; or, how to build a data quality browser. In *ACM SIGMOD Conference*, pages 240–251, 2002.

[10] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems*. Addison/Wesley, Redwood City, California, 3rd edition, 2000.

[11] J.W. Fisher and D.J. Berndt. Creating false memories: Temporal reconstruction errors in data warehouses. In *Eleventh Workshop on Technologies and Systems*, New Orleans, 2001.

[12] A. Fuxman, E. Fazli, and R.J. Miller. Conquer: efficient management of inconsistent databases. In *ACM SIGMOD Conference*, pages 155–166, 2005.

[13] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-total. In *ICDE Conference*, pages 152–159, 1996.

[14] B.M. Horowitz. A run-time execution model for referential integrity maintenance. In *IEEE ICDE Conference*, pages 548–556, 1992.

[15] T. Johnson, A. Marathe, and T. Dasu. Database exploration and Bellman. *IEEE Data Engineering Bulletin*, 26(3):34–39, 2003.

[16] E.P. Lim and R.H. Chiang. The integration of relationship instances from heterogeneous databases. *Decis. Support Syst.*, 29-2(3-4):153–167, 2000.

[17] V.M. Markowitz. Safe referential structures in relational databases. In *VLDB*, pages 123–132, 1991.

[18] A. Motro and I. Rakov. Estimating the quality of data in relational databases. In *IQ*, pages 94–10, 1996.

[19] D. Mukhopadhyay and G. Thomas. Practical approaches to maintaining referential integrity in multi-database systems. In *RIDE-IMS*, pages 42–49, July 1993.

[20] C. Ordonez and J. García-García. Consistent aggregations in databases with referential integrity errors. In *ACM IQIS Workshop*, pages 80–89, 2006.

[21] L. Pipino, Y.W. Lee, and R.Y. Wang. Data quality assessment. *ACM CACM*, 45(4):211–218, 2002.

[22] M. Scannapieco and C. Batini. Completeness in the relational model: A comprehensive framework. In *IQ Conference*, 2004.

[23] M. Serrano, C. Calero, J. Trujillo, S. Lujan-Mora, and M. Piattini. Empirical validation of metrics for conceptual models of data warehouses. In *CAISE*, pages 506–520, 2004.

[24] C. Türker and M. Gertz. Semantic integrity support in SQL: 1999 and commercial (object-)relational database management systems. *VLDBJ*, 10(4):241–269, 2001.

[25] R.Y. Wang, M.P. Reddy, and H.B. Kon. Toward quality data: an attribute-based approach. *Decis. Support Syst.*, 13(3-4):349–372, 1995.