Measuring Referential Integrity in Distributed Databases

Carlos Ordonez
University of Houston
Houston, TX, USA

Javier García-García
UNAM University
Mexico City, Mexico

Zhibo Chen
University of Houston
Houston, TX, USA

ABSTRACT

Distributed relational databases are used by different organizations located at multiple sites that work together on common projects. In this article, we focus on distributed relational databases with incomplete and inconsistent content. We propose to measure referential integrity errors in them for integration and interoperability purposes. We propose local and global referential integrity metrics at three levels: column, table and database. We assume each table can be asynchronously updated at any site and new records are periodically broadcasted to all sites. We explain several distributed query optimization issues. Our proposal is useful in database integration, multiple database interoperability and data quality assurance. We discuss applications of our proposal in distributed scientific databases.

1. INTRODUCTION

Referential integrity [3] represents the cement that keeps relational database components together. In a relational database such components are tables and the link between two tables is a foreign key. Unfortunately, in a modern environment, referential integrity may be violated because tables with similar content, but coming from different source databases, are integrated or need to inter-operate. To compound the problem, nowadays many organizations use multiple databases, residing at different locations, that are communicated through the Internet. In general, such databases are integrated into a data warehouse or they need to exchange records. Our goal is to efficiently identify tables and columns with referential integrity problems in a distributed database so that users detect and avoid inconsistency or incompleteness issues. In our approach there is the underlying assumption that it is expensive to maintain all databases synchronized with the latest updates. For a long time scientists have worked in isolation, creating private data repositories with their own experimental results. Nowadays, with the growing usage of the Internet and faster computers, more scientists are interested in collaborating with other scientists working at different locations and storing shared data in their local computers. We believe distributed databases combined with referential integrity verification represent a promising alternative to detect and fix data quality issues in order to maintain experimental data repositories in a collaborative environment.

http://doi.acm.org/10.1145/1317353.1317367

The article is organized as follows. Section 2 introduces basic definitions. Section 3 introduces referential integrity metrics in a distributed database and identifies query optimization issues. Section 4 provides a description of our program and GUI. Related work is discussed in Section 5. Section 6 concludes the article.

2. DEFINITIONS

Our proposal is based on a distributed database at n sites [8]. Let $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$ be a set of n relational databases. To have a uniform framework, all databases have the same tables (i.e. same structure), but potentially different content (i.e. inconsistent or incomplete content). Let D_i be defined as $D_i(\{T_1, T_2, \ldots, T_m\}, I)$, where T_i is a table and I a set of referential integrity constraints.

A referential integrity constraint [3] in I, between T_i and T_j is an assertion of the form $T_i(K) \rightarrow T_j(K)$, where T_i is called the referencing table, T_j is called the referenced table, K is a foreign key (FK) in T_i and K is the primary key (PK) of T_j. Since we deal with incomplete or inconsistent databases, we assume T_i may contain rows having $T_i.K$ values that are null or that do not exist in $T_j.K$. To make our proposal more practical, we also assume tables can be denormalized. Denormalization is generally used to reduce query evaluation time. We use K to refer to a foreign key and F for a foreign column in a denormalized table, that is functionally dependent on some foreign key.

Each table T_i can be updated at any site since all sites have local copies of T_i (replicas) that are periodically refreshed with sets of new records. Since the database is distributed any site can receive local updates and sets of new records. For instance, for a distributed medical database for hospitals each local database has patient and doctor information that is locally updated, where a batch of new records is periodically broadcasted to the other hospitals. The diagram in Figure 1 shows a distributed database with independent updates at each site and periodic updates broadcasting sets of new records.

3. REFERENTIAL INTEGRITY METRICS IN A DISTRIBUTED DATABASE

In this section we present our main contributions. First, we introduce distributed referential integrity metrics. Then we explain several distributed query optimization issues. Finally, we conclude by summarizing potential application in scientific databases. Our proposal generalizes referential integrity quality metrics for a single database. We first define

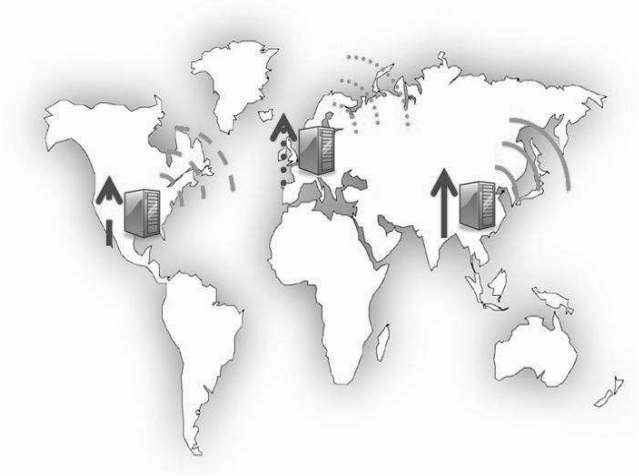

Figure 1: Distributed database environment.

Letter	Meaning
l	local
g	global
r	referential
com	completeness
con	consistency
cur	current

Table 1: Logic for acronyms of metrics.

a local metric for a single database and then such metric is generalized to n databases. We define completeness and consistency metrics so that they are in [0,1] (1 being the optimal). Table 1 summarizes the logic to name our metrics.

3.1 Assumptions

We make two assumptions:

1. Metadata has been integrated before; this is a separate and more difficult problem.
2. Database content may be inconsistent due to both local and global issues.

We assume tables structure is known and uniform across databases. Broadcasting updates happens independtly and asynchronously for each table in one database. Therefore, tables may violate referential integrity over time. We further assume that unique identifiers can be generated by concatenating the site identifier to avoid primary key duplication.

3.2 Column Metrics

An invalid foreign key or a foreign key with nulls is considered incorrect. In practice nulls are sometimes considered correct FK values. It is straightforward to extend our definitions to ignore nulls for metric computation. We first define a metric as a local measure on a single database. Notice $\bowtie$ automatically filters out invalid and null FKs. The following metric is a completeness measure for one column:

$$lrcom(T_i.K) = \frac{|T_i \bowtie_K T_j|}{|T_i|} \tag{1}$$

We generalize definitions to n databases. Let $\mathcal{T}_i$ be defined as the global union of all local copies of T_i:

$$\mathcal{T}_i = D_1.T_i \cup \cdots \cup D_n.T_i. \tag{2}$$

The next metric measures global completeness for one table and detects the existence of table replicas with missing foreign keys.

$$grcom(\mathcal{T}_i.K) = \frac{|\mathcal{T}_i \bowtie_K \mathcal{T}_j|}{|\mathcal{T}_i|} \tag{3}$$

This definition does not consider the case where there are inconsistent rows such that the K is the same, but functionally dependent columns on K are different, because different sites cannot generate duplicate record identifiers.

We now concentrate on consistency metrics. Let F be a foreign column in a denormalized table T_i. The following equation measures the local consistency:

$$lrcon(T_i.F) = \frac{|\sigma_{T_i.F=T_j.F}(T_i \bowtie_K T_j)|}{|T_i|}. \tag{4}$$

This equation can be computed in a simpler manner with one relational operator as:

$$lrcon(T_i.F) = \frac{|T_i \bowtie_{K,F} T_j|}{|T_i|}. \tag{5}$$

We generalize the previous definition to n databases reusing the global union tables $\mathcal{T}_i, \mathcal{T}_j$. The following metric detects foreign columns F that do not match their corresponding reference values in the referenced table.

$$grcon(\mathcal{T}_i.K, \mathcal{T}_i.F) = \frac{|\mathcal{T}_i \bowtie_{K,F} \mathcal{T}_j|}{|\mathcal{T}_i|} \tag{6}$$

3.3 Table Metrics

We first define a metric to know if a table has all latest updates (i.e. if it is current). This global metric is applicable to both the referencing table T_i and the referenced table T_j and is similar to the Jaccard coefficient [4]:

$$gcur(T_i) = \frac{|D_1.T_i \cap D_2.T_i \cap \cdots \cap D_n.T_i|}{|D_1.T_i \cup D_2.T_i \cup \cdots \cup D_n.T_i|} \tag{7}$$

We now define global metrics and omit analogous definitions for local metrics. Given a global table $\mathcal{T}_i$ with k foreign keys $K_1, \ldots, K_k$ its referential completeness is measured by:

$$grcom(\mathcal{T}_i) = \frac{\sum_{j=1}^{k} |\mathcal{T}_i| grcom(\mathcal{T}_i.K_j)}{k|\mathcal{T}_i|} \tag{8}$$

Given the global table $\mathcal{T}_i$ with f foreign columns $K_1, \ldots, K_f$ its referential consistency is:

$$grcon(\mathcal{T}_i) = \frac{\sum_{j=1}^{f} |\mathcal{T}_i| grcon(\mathcal{T}_i.F_j)}{f|\mathcal{T}_i|} \tag{9}$$

3.4 Database Metrics

It makes sense to only define local referential metrics for each individual database. Given a database D_i its local completeness metric can be derived by aggregating table metrics:

$$lrcom(D_i) = \frac{\sum_{j=1}^{m} |T_j| lrcom(T_j)}{\sum_j |T_j|} \tag{10}$$

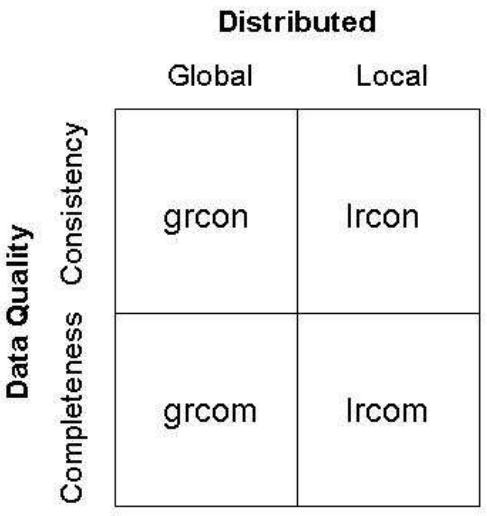

Figure 2: Classification of metrics.

For D_i its local metric for consistency is computed aggregating table consistency metrics:

$$lrcon(D_i) = \frac{\sum_{j=1}^{m} |T_j| lrcon(T_j)}{\sum_j |T_j|} \tag{11}$$

Finally, we define global metrics for the entire distributed database $\mathcal{D}$ for completeness and consistency, respectively:

$$grcom(\mathcal{D}) = \frac{\sum_{j=1}^{m} |\mathcal{T}_j| grcom(\mathcal{T}_j)}{\sum_j |\mathcal{T}_j|} \tag{12}$$

$$grcon(\mathcal{D}) = \frac{\sum_{j=1}^{m} |\mathcal{T}_j| grcon(\mathcal{T}_j)}{\sum_j |\mathcal{T}_j|} \tag{13}$$

The diagram in Figure 2 shows our proposed metrics classification according to data quality dimensions and being local or global.

3.5 Query Optimization

Referential integrity metrics at the three levels are computed with automatically generated SQL code. As expected, column metrics are the most demanding. On the other hand, table and database metrics are quickly aggregated from column metrics. For local metrics in a single database two query optimizations are useful [7]:

- Computing aggregations grouping by foreign keys before joins for a table having several FKs. This optimization reduces the size of intermediate tables when foreign keys have low selectivity and it is based on pushing aggregations before joins [2].
- Creating a secondary index on each foreign key for efficient access; the cost of index creation is amortized with frequent foreign key joins.

Generalizing optimizations to n databases is interesting because updates can happen asynchronously at any site. We adapted distributed query optimization techniques [8] to our problem, assuming the cost to transfer a large table is high. For global metrics the most important relational operation is the global union of all local copies of one table. Such union can be optimized by computing pairwise intersections of tables projecting only the foreign key to determine which tables have new rows and then transfer only new rows. Metrics are based on foreign keys and foreign columns which can be projected before the global union. The second important operation is computing the global intersection to find out records existing at every database. The intersection operation is less demanding because intersections generally return smaller tables.

Global metrics for all tables (say m) and the database can be computed with several approaches:

- A static approach is to transfer $n-1$ copies to some site designated as the "central" site. When this is done for all m tables it becomes expensive.
- A second approach is to compute metrics at one site one time and then incrementally update them when new records are inserted; metrics are continuously recomputed for all m tables when they are refreshed. The challenge is to maintain a low overhead.
- A third approach is to compute metrics for each pair of tables linked by a foreign key on demand, when their referential integrity needs to be verified.

When tables are large sampling can be used to get an approximation of referential metrics. Iterated sampling are preferred over transferring large tables. To process a distributed join the program can take advantage of a replicated table, being locally stored at the same site, but which may be outdated. When there is a need to join tables stored at different sites the program projects only a minimum set of primary keys, foreign keys and foreign attributes, covering the generated query. When a join requires two tables stored at different sites the smallest table is transferred to the other site. We are currently exploring strategies to compute a distributed join with two large tables.

3.6 Application with Scientific Databases

Let us consider a distributed database to manage experiment data from researchers collaborating on a common project. We make the following assumptions. Researchers work at different locations and are connected by the Internet. Researchers have access to each other databases, but they need to frequently compare their results with everyone else. Hence, results must be integrated. Therefore, each researcher has a local database for his results and keeps copies of other researchers results in his database. That is, each researcher export data sets from the other databases, downloads them into local text files and imports those files into his local database. Referential integrity may be locally violated because experiment data sets are manually collected outside the database and because the local database does not have the latest version of referenced tables. Now consider a potential application scenario. Researcher A is browsing experimental results from other researches. Researcher A makes the following findings. There exists incomplete data in experiments (nulls) and there is inconsistent classification of results (similar experimental results have different foreign column values). There are missing foreign keys in rows (i.e. unavailable, invalid at time of experiment). It turns out that

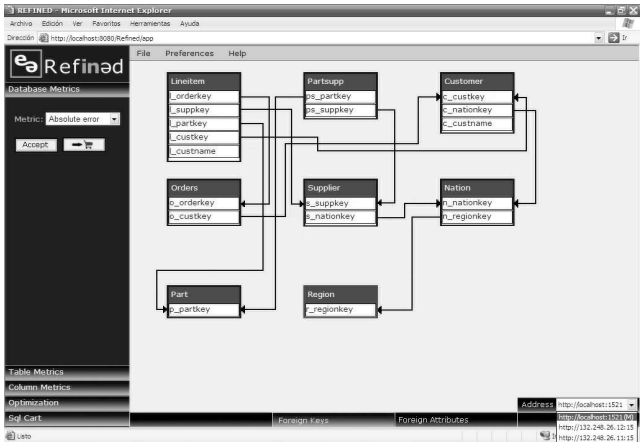

Figure 3: Selecting tables and columns from LDM.

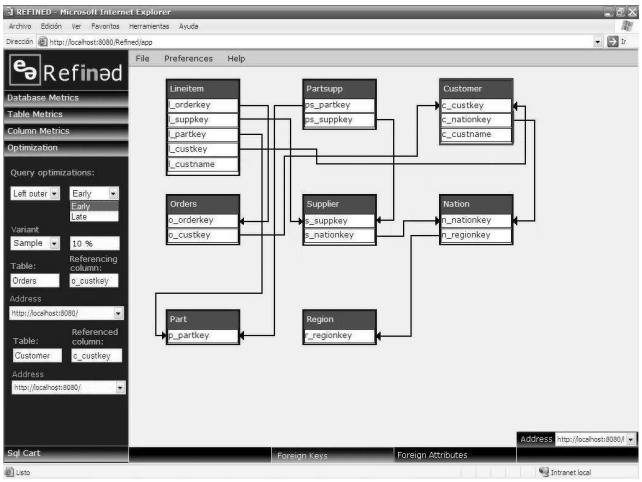

Figure 4: Query optimizations.

there is missing classification of experimental findings because there is work in progress. Based on referential metrics researcher A realizes certain databases, from other teams, tend to have more referential errors and when that happens referential errors tend to be localized in tables X, Y and Z. Researcher A can investigate which specific columns tend to have errors to prepare a solution plan. Researchers at other sites are probably unaware of referential issues. Two potential solutions to fixing referential errors could be the following. Referenced tables could be synchronized at every site, but this implies immediately propagating new records, which is inefficient. Keeping a central database, but this implies there is always a fast connection and the central database is always fast and available. In short, it is more flexible and faster to keep local databases, even though they may have referential errors.

4. DESCRIPTION OF PROGRAM

We now provide an overview of a prototype that can connect to several distributed databases with join-compatible tables to compute referential completeness and consistency metrics. The program takes as input the logical data model (LDM) behind the distributed database, which is assumed to be uniform across local databases. The LDM is represented as a directed graph, where each vertex represents a table and each edge represents a foreign key relationship between two tables. Tables can be stored at multiple sites and the LDM is enriched with the http addresses of the DBMSs storing each table, designating one as the master copy. The LDM is internally manipulated as a list of foreign key relationships between a referencing table and a referenced table. The user can inspect the entire distributed database, which requires running the program in batch mode, or interactively, by selecting one table or a few columns from one table; see Figure 3. The database may be denormalized. That is, some tables may not be in 3NF [3]. In such case the user can verify foreign attribute consistency by comparing denormalized column values against "reference"values.

Completeness and consistency metrics are initially computed for each column. Then they are aggregated at the table level and finally, they are aggregated at each database, in a bottom-up fashion. Finally, the program computes global referential metrics. The program provides several query optimization strategies, as shown in Figure 4. The program sampling feature can approximate metrics, providing fair accuracy guarantees, that may be improved through repeated sampling. As discussed above, a useful query optimization is performing an early "group by" on foreign keys before joining a large table to accelerate processing of tables with several foreign keys, each having a low number of distinct values; this optimization effectively compresses a table before joins. Response time depends on metric results stored from previous runs, at which sites tables are stored, network speed, tables sizes and query optimizations.

When results are ready, the user has the ability to explore local and global referential integrity metrics. When exploring local metrics the user can browse each LDM by clicking on desired tables or columns. The output is a set of metrics at three levels: database, table and column. Databases, tables and columns are marked green, yellow or red, indicating no errors, some errors and significant referential issues, respectively, as shown in Figure 5. When the user is interested in knowing why a certain column has a high error, he can "drill down"on such column, getting a list of values with referential errors, sorted by their frequency in descending order. This list explains why errors happened and it can help identifying critical values to fix referential errors.

5. RELATED WORK

In order to manage distributed scientific models and data, a distributed database system and an enhanced metadata support system are proposed in [1]. They focus on solving issues related to the use of scientific resources, programs and data, in environmental applications. By means of an extended architecture they manage distribution and heterogeneity of data and program sources. Additionally, they propose mechanisms to capture model descriptions and to monitor the distributed usage of models, programs and data. A middleware system called MOCHA is presented in [10]. This system is used to integrate distributed data sources and it is based on the idea that the code that implements user-defined types and functions should be automatically deployed to remote sites by the middleware system itself. This

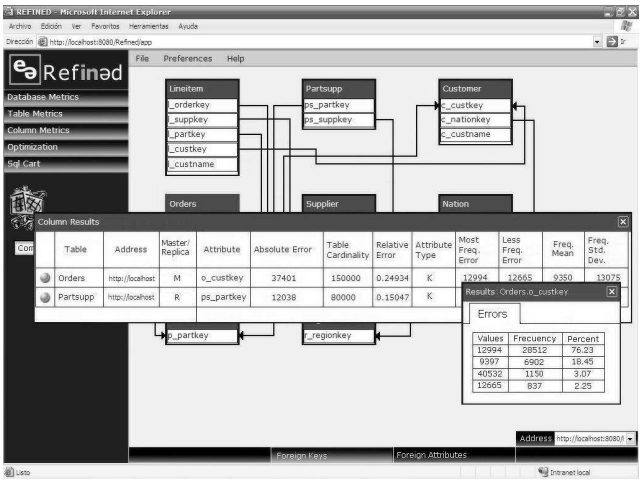

Figure 5: Displaying referential integrity issues.

is done by means of an XML-based framework to specify metadata about data sites, data sets, user-defined types and functions.

A prototype system to manage large result files from numerical simulations is presented in [9]. The proposed interface was specifically designed for users with a scientific background and not familiar with SQL. By using an intuitive searching and browsing mechanism the user can store, search and retrieve large, distributed, files resulting from scientific simulations. The interface construction is automated and dynamic so that it requires little database/web development experience to install and use. This is achieved by allowing the user interface specification to be defined in an XML file used to initialize the system and by providing a program that can generate default XML specifications. An architecture for creating a collaborative centralized infrastructure for sharing scientific data is proposed in [5]. The architecture uses a search scheme that allows users to locate data sets by exploiting metadata information.

Our proposal extends quality metrics defined on a single database [7]. This article presents metrics that measure absolute and relative error with respect to referential integrity. Two new aspects in our generalization are that we measure completeness and consistency instead of error and that we contrast a local database state with a global state where all referential issues are reduced or eliminated. Also, our proposal identifies new issues related to distributed query optimization. Our work is also related to getting consistent aggregations under referential integrity issues [6]; this work goes a step beyond detecting errors: it dynamically builds a repaired answer set to a query.

6. CONCLUSIONS

We proposed metrics to discover and quantify referential integrity problems in a distributed database, considering normalized and denormalized schemas. Referential completeness and consistency metrics are defined in a hierarchical fashion at the column, table and database levels. We discussed research issues on distributed query optimization to efficiently compute metrics. We presented potential application with scientific databases. As a data quality diagnostic tool, our referential integrity distributed browser represents a complement to programs that measure data quality in each table individually or in a single centralized database.

Acknowledgments

The second author is with Universidad Nacional Autónoma de México (UNAM) and was sponsored by the UNAM IT project "Macroproyecto de Tecnologías para la Universidad de la Información y la Computación". We would like to thank René Villeda-Ruz and the students from the database laboratory from the UNAM University for their help implementing our system.

7. REFERENCES

[1] M.C. Cavalcanti, M. Mattoso, M.L. Campos, E. Simon, and F. Llirbat. An architecture for managing distributed scientific resources. In *14th International Conference on Scientific and Statistical Database Management*, pages 47 – 55, 2002.

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

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

[4] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, 1st edition, 2001.

[5] A. R. Jaiswal, C. L. Giles, P. Mitra, and J. Z. Wang. An architecture for creating collaborative semantically capable scientific data sharing infrastructures. In *ACM WIDM Workshop*, pages 75–82, 2006.

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

[7] C. Ordonez and J. García-García. Referential integrity quality metrics. *Decision Support Systems Journal*, 44(2):495–508, 2008.

[8] M.T. Ozsu and P. Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, 2nd edition, 1999.

[9] M. Papiani, J.L. Wason, A.N. Dunlop, and D.A. Nicole. A distributed scientific data archive using the Web, XML and SQL/MED. *SIGMOD Rec.*, 28(3):56–62, 1999.

[10] M. Rodriguez-Martinez, N. Roussopoulos, J. M. McGann, S. Kelley, J. Mokwa, B. White, and J. JaJa. Integrating distributed scientific data sources with MOCHA and Xroaster. In *SSDBM Conference*, pages 263–266, 2001.