A Referential Integrity Browser for Distributed Databases

Carlos Ordonez[1] Javier García-García[2] Rogelio Montero-Campos[2] Carlos Garcia-Alvarado[1]

[1] University of Houston
Houston, TX, USA

[2]UNAM*
Mexico City, Mexico

ABSTRACT

We demonstrate a program that can inspect a distributed relational database on the Internet to discover and quantify referential integrity issues for integration purposes. The program computes data quality metrics for referential integrity at four granularity levels: database, table, column and value, going from a global to a detailed view, exhibiting specific evidence about referential errors. Two orthogonal data quality dimensions are considered: completeness and consistency. Each table is stored at one primary site and it can be replicated at multiple sites, having foreign key references to tables at the same site or at different sites. The user can choose alternative query evaluation strategies to efficiently compute referential error metrics. Our proposal can be used in data integration, data warehousing and data quality assurance.

1. INTRODUCTION

Referential integrity [1] is the glue that maintains relational database components together. In a database such components are tables and the fundamental link between two tables is a foreign key. Unfortunately, referential integrity may be violated for several reasons: tables with similar information, extracted from different source databases, are integrated, referential integrity constraints are disabled to improve performance, the logical data model evolves to include new columns or drop existing columns, or simply, some foreign key column value is not available when a row is inserted. To compound the problem, nowadays many organizations use several databases, residing at different locations, communicated by the Internet. In general, such databases are integrated into a data warehouse. Our program has the purpose of efficiently detecting tables and identifying columns with referential integrity problems on interrelated tables in a distributed database on the Internet.

*Universidad Nacional Autónoma de México. Authors Javier García-García and Rogelio Montero-Campos were sponsored by the UNAM project Macroproyecto de Tecnología para la Univ. de la Información y la Computación.

Twelfth International Workshop on the Web and Databases (WebDB 2009), June 28, 2009, Providence, Rhode Island, USA.

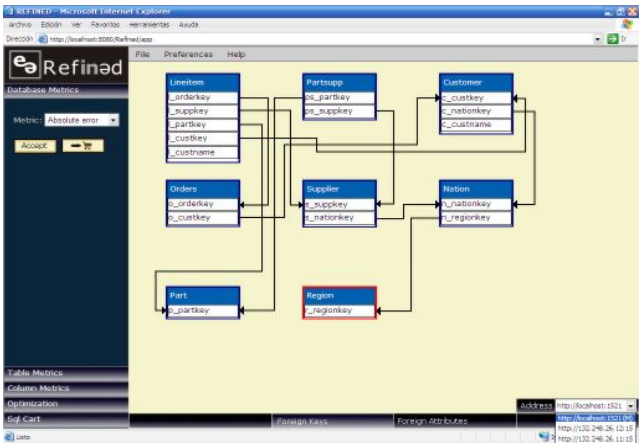

Figure 1: Selecting tables and columns from LDM

2. QUALITY METRICS

The program quantifies referential errors in a distributed database with absolute and relative error metrics. Absolute error is useful in tables where almost zero errors are acceptable. Such tables correspond to small lookup tables that are infrequently updated and which are generally replicated to improve performance and fault tolerance. Relative error is useful in large tables, where errors can appear frequently. Large tables typically correspond to transaction or historic tables; these tables are replicated to improve fault tolerance and they are periodically refreshed with new records.

Our QMs are based on foreign keys and foreign columns. Foreign columns are those which are functionally dependent on a foreign key in a denormalized table. QMs on foreign keys measure completeness, whereas QMs on foreign columns measure consistency.

3. DESCRIPTION OF PROGRAM

The program takes as input the logical data model (LDM) behind the distributed database. 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

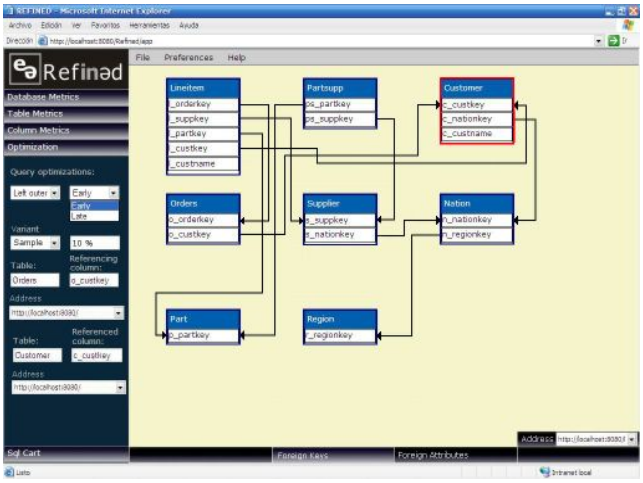

Figure 2: Query optimizations

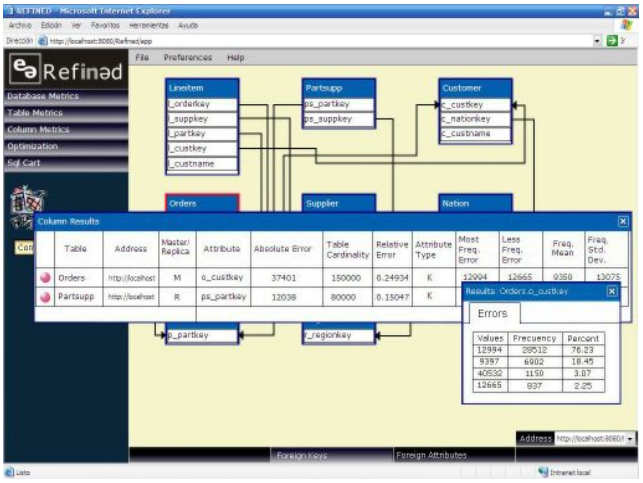

Figure 3: Displaying referential integrity issues

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 1. The database may be denormalized: some tables may not be in 3NF [1]. In such case the user can verify consistency by comparing denormalized column values against "reference" values; we call such columns "foreign columns".

Absolute and relative error QMs are initially computed for each column. Then they are aggregated at the table level and finally, they are aggregated at the database level, in a bottom-up fashion. The program provides several query optimization strategies, as shown in Figure 2. We adapted common distributed query processing techniques [2] to our problem, assuming the cost to transfer a large table is high. The program favors iterated sampling 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 via JDBC to the other site. We are currently exploring strategies to compute a distributed join with two large tables. The program sampling feature can approximate QMs, providing fair accuracy guarantees, that may be improved through repeated sampling. A particular 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 QM 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 referential integrity by traversing the LDM by clicking on desired tables or columns. The output is a set of QMs at three levels: database, table and column. Tables and columns are marked green, yellow or red, indicating no errors, some errors and signicant referential issues, respectively, as shown in Figure 3. 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.

4. DEMO WITH A SYNTHETIC DATABASE

Our prototype was programmed in the Java language for portability and web access. Queries to compute QMs are dynamically generated in ANSI SQL. The program is a web-based application that can connect to any relational DBMS through the JDBC interface.

We will demonstrate our prototype with a small distributed database on the Internet, created with the TPC-H generator. A few tables will be replicated on different DBMSs, residing at distinct geographical locations. The user can pick specific tables and columns to compute QMs. We will show interactive operation, where different options can be tried to compare speed and accuracy of our different query optimization strategies. Tables and columns having referential errors will be marked accordingly. Finally, the user can "drill down" on the specific foreign key/column values which contribute to referential error QMs.

5. CONCLUSIONS

Our system can inspect a distributed relational database to discover and quantify referential integrity problems, considering normalized and denormalized databases. Quality metrics are hierarchically computed applying several query optimizations. As a data quality diagnostic tool, our referential integrity detective program represents a complement to programs that measure data quality in each table individually or in a single centralized database.

6. REFERENCES

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

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