http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 12 (24): 8035-8039, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.8035.8039

Rewriting Queries for Inconsistent Data Based on Directed Join Graphs

[1]Dong Xie, [2,3]Song Tang, [2]Huijun Wang, [4]Guangjun Guo and [1]Yun Cheng
[1]Department of Computer Science and Technology, Hunan University of Humanities,
Science and Technology, Loudi, China
[2]Institute of Network Media and Public Sentiment,
Hunan Mass Media Vacational Technical College, Changsha, China
[3]Department of Network Media, Hunan Mass Media Vacational Technical College, Changsha, China
[4]Department of Electronics and Information Engineering,
Loudi Vocational and Technical College, Loudi, China

Abstract: Inconsistency may arise when data are integrated together with respect to a given set of integrity constraints. Many techniques can identify inconsistent data by ad-hoc ways but inconsistent data can not be managed effectively. To judge a query is rewritable, the work analyses whether the join graph for the query is directed. Also, the work presents an approach for computing tuple probabilities and a basic technique for query rewriting. The experiments use the data and queries of the TPC-H specification to compare performance for degrees of inconsistencies and the results show that the approach is effective and feasible.

Key words: Relational database, inconsistent data, rewriting query, directed join graph

INTRODUCTION

Integrity Constraint (IC) of relational database effectively ensures data integrity and effectiveness. For a given binding, multi-data sources are consistent respectively but they may be Inconsistent Database (IDB) (Ananthakrishna *et al*., 2002) when they are integrated. Traditionally, the data cleansing of inconsistent databases is one strategy to identify and correct data' errors, which is in order to restore consistency. For the cost of processing is expensive, useful data will be lost or users don't know how to store consistent data, it is impossible to modify the database to preserve the consistency of data under these circumstances. For some tasks, a user would like to adopt a different strategy for cleaning or wish to retain all the data and even inconsistent data. However, most of the data in the database remains consistent. As a result, maintaining the source data not to be changed is a way to alternative data cleansing but it queries to address inconsistency, identifies the consistency of data (Arenas *et al*., 1999).

Some data integration tools used cluster approach to divide data collection into several categories, making each type of internal data similar as far as possible and different types of data different as much as possible. As for different categories' queries and constraints, some of the existed SPJ queries are not rewritten, so they are NP completes (Cali *et al*., 2003). However, the actual number of queries, the computational complexity is PTIME (Cali *et al*., 2003). Based on dealing with the probability of relational database model (Fuhr *et al*., 1997) of the uncertainty information, a possible world semantics is presented (Dalvi *et al*., 2004) to analyze the cluster within potential multi-inconsistencies of the probability of tuple. A probabilistic approach is proposed (Andritsos *et al*., 2006) to obtain clean answers over dirty databases but the approach did not consider query rewriting. Our previous work (Xie *et al*., 2012a) presents a method to obtain certain answers by query rewriting but the method is based on the possible world semantics. In our other previous work (Xie *et al*., 2012b), we consider the theory and algorithm for polytype queries in inconsistent databases.

We propose a directed join graph concept to rewrite queries for extending our previous works. This paper, by analyzing the corresponding join to the query whether there are plans to connect to the graph, can determine whether two types of queries (non-key-key and key-key joins) are rewritable and give the tuple probability calculation and the basic way of query rewriting. Using the TPC-H decision support benchmark data and performance of queries, consider the non-uniform multi-granularity and clustering of data base to compare non-uniform implementation of the performance of data granularity.

Corresponding Author: Song Tang, Hunan Mass Media Vacational Technical College, Changsha, China

FORMAL FRAMEWORK

Database instances are first-order structures over the first-order language L in the standard way. A database schema R is a finite set of relation names and the associated arities and given built-in predicates (=,,,, <, >) have infinite, fixed extensions. Integrity constraints are closed first-order L-formulas.

In the sequel we will denote: relation symbols by P, P_1,..., Pm, atomic formulas by A_1,..., A_n, tuples of variables and constants by t', ***x***',... and quantifier-free formulas referring to built-in predicates by v. We consider the following basic classes of integrity constraints: universal integrity constraints, denial constraints, binary constraints, Functional Dependencies (FDs), referential integrity constraints (inclusion dependencies,INDs).

We adopt a semantics (Arenas *et al*., 1999), it relies on repair and consistent query answer. A repair satisfies the Ics which has a minimal distance to the inconsistent database. The formal definition of repair is the following.

Definition 1: Arenas *et al*. (1999); the distance between two database instances I1 and I2 is their symmetric difference $\Delta(I1, I2) = (I1-I2)\cup(I2-I1)$.

Definition 2: Arenas *et al*. (1999); Given an instance I of a database schema R and a set of integrity constraints IC, we say that I is consistent if I||IC in the standard model-theoretic sense; inconsistent otherwise.

Definition 3: Arenas *et al*. (1999); Given a set of integrity constraints IC and database instances I and I', we say that I' is a repair of I with respect to IC if I' || IC and there is no instance I'' such that I''| IC and $\Delta((I, I'')\Delta(I, I')$.

Definition 4: Arenas *et al*. (1999). A tuple t is a consistent query answer to a query Q(x) in a database instance I with respect to a set of integrity constraints IC iff t is an answer to the query Q(x) in every repair I' of I with respect to IC. We can define true being a consistent answer to a Boolean query in a similar way.

It is inconsistent if it is scheduled to meet any IC (Arenas *et al*., 1999). Relationships are divided into several tuple subsets (cluster), which include several probability tuples, a member of the cluster tuple is called cluster base (Andritsos *et al*., 2006). We give a concept of cluster credible based on probability approach.

Definition 5: Believable cluster: C_i of relation R is a cluster which the base is more than 1, t_i is an arbitrary tuple of cluster C_i, Q is a query of R. If $t_i \notin Q(R)$, it is known as an incredible cluster, otherwise it is an absolutely credible cluster. Cluster probability of Q: $pr = \Sigma c_i \,|\, t \notin Q(c_i)\, pr(t)$.

We consider that the key relies in chief. Most of TPC-H queries are based on non-key and key link, so these are also the most ordinary circumstances of IC. It can use graph to show queries for showing the join of the relationship attributes in queries directly. Node means the relationship in queries and arc uses attribute to connect among relationships. Previous works do not illustrate the graph's direction, so this paper supplements the original definition.

Definition 6: Join graph (Andritsos *et al*., 2006): Assume that Q is a SPJ query, R_i and R_j stand for the relationships in Q. If G's vertex is the relationship in Q and a non-key attribute or key attribute in R_i equals to the key attribute of R_j, then there is join arc from R_i to R_j. Therefore, the graph G who corresponds to Q is a join graph.

However, the join graph (Andritsos *et al*., 2006) does not consider different attribute join and the same attribute join of inner relationship. In the worst cases, they lead to a problem that whether they can calculate effectively. The rewritten query, given in this study, is based on the directional join graph, which has two categories of joins: one can be connected between two key attributes; the other is connected from the relationship's non-key attribute (which may be a foreign key) to another primary relationship key.

Definition 7: Directed join graph: assume that Q is a SPJ query, if G meets the following conditions, then G is a directional join graph. (a) G's vertex is the relationship in Q, for any two vertices R_i or R_j, there can't be i = j. (b) For any vertex R_i or R_j, if a non-key attribute of R_i equals to the key attribute of R_j, then there exists a directional edge whose direction is R_i pointing R_j. (c) For any vertex R_i or R_j, if a non-key attribute of R_i equals to the key attribute of R_j, then there can't exists a non-key attribute of R_i which equals to any attribute of R_k. (d) For any vertex R_i, there is no ring.

The first condition means that each relationship can be used only once in the query; the second one means that its main consideration is equivalent to connect and rely on the key IC; the third one means that if a key attribute of a relationship equals to a non-key attribute of another relationship, then there can't be an arbitrary attribute of the third condition equals to a non-key attribute of the first condition. The fourth one means that there is no Auto-Connect.

Two types of directional joins are very common used in experiments (e.g., the standard TPC-H), it is PTIME (Cali *et al*., 2003) in computational complexity. Therefore, if Q is an SPJ query, linking map of Q is a directional join graph and then Q is a query which can be rewritten. For example, because only q_4 is a directional join graph in the following four queries, so only q_4 can be rewritten:

q_1 = $\exists x \exists z \exists y$: $R_1(x, y) \exists R_2(z, y)$
q_2 = $\exists x \exists y \exists z$: $R1(z, x, y)$ $\exists R_2(y, x)$
q_3 = $\exists x \exists y \exists z \exists w$: $R1(x, y) \exists R2(z, w) \exists R3(y, w)$
q_4 = $\exists x$: $R1(x) \exists R1(x)$
q_5 = $\exists x$: $R1(x) \exists R2(x)$
q_6 = $\exists x \exists y \exists z \exists w$: $R1(x, y) \exists R2(y, z)$ $\exists R3(z, w)$ $\exists R4(z,c) \exists R5(y,c)$

This is a sufficient condition for a query whose join graph is a forest to be first-order rewritable. We show a class of queries such that the problem of computing the consistent answers is coNP-complete, the queries does not satisfy the conditions of our algorithm. The work (Xie *et al*., 2012a) establishes a dichotomy between first-order rewritability and coNP-completeness, one intractable query in order to conclude that the entire class is coNP-complete (Hernandez and Stolfo, 1998).

We will present a class of queries,which are not a forest and it is not difficult to find a rewriting for it whose two literals have the same key. It is easy to find a rewriting for this query; its join graph contains a self-loop.

Definition 8: A conjunctive query q is in class C_{np} if for every pair R and R' of literals of q one of the following conditions: (a) there is not a key-key join between R and R'; (b) there is a nonkey-nonkey join between R and R'; (c) there are literals R1,...,Rm in q such that there is a nonkey- key join but the join graph of q is a cycle.

If a query $q \epsilon C_{np}$, then the problem of computing the consistent answers for q is coNP-complete in data complexity.

CQA with a single nonkey-nonkey join is coNP-complete (Cali *et al*., 2003). Their result used a query with repeated relation symbols. We can use their insight to show that the problem of computing consistent answers for the following query without repeated relation symbols but with a single nonkey-nonkey join is also coNP-complete. Given a query q such that $q \epsilon C_{np}$. The problem of CQA for q can be reduced from the problem of obtaining the consistent answers for one of three particular query families. Since $q \epsilon C_{np}$, there is a PTIME reduction from q' to q where q' is one of the following queries:

- x, x',y: $S1(x,y) \wedge S2(x',y)$
- x,y:$T1(x, y) \wedge T2(y, x)$

PROCESSING INCONSISTENT DATA

To deal with inconsistent data, it must consider from several aspects. First of all, the probability of clusters must be calculated. Secondly, judge the query whether can be rewritten. Thirdly, calculate the query rewriting. Finally, perform the last rewritten query on the existing DBMS.

Adding an attribute marked as a clustering in the IDB to distinguish clusters, there can introduce the probability mark Pr to indicate its uncertainty. The greater the base cluster is, the more complex of the calculation of the inner cluster-tuple is. If $V = (V_1, V_2,..., V_m)$ means a possible value of a cluster attribute (where, $t \epsilon \{C_1,C_2,..., C_n\}$, $v \epsilon V$), then the conditional probability distribution is $pr(t|C_n) = |v| / |C_n|$. $t(C_i)$ means the number of attribute value of *t*th tuple in the cluster, C_i (A_i) means the number of attribute value in the inner clustering tuple. If the cluster has only a tuple, that is, $|C_n| = 1$, then $pr(t \mid C_n) = 1$. As for a large cluster (when $|C_n| > 1$), the probability of t is $pr(t \mid Ci) = \Sigma pr(t(A_i)) / \Sigma pr(ci(Ai))$. If an attribute value of the t hree tuples of a cluster appeared once, then $pr(t|C_n) = 0.333$. Therefore, the probability of tuple's calculation and human's intuition are the same in the processing of designating probability.

Algorithm 1
Input: t clusters
Output: tuple probability
1. For I = 1 to t do
2. {Calculate the number of cluster m of inner clustering tuple in the value of each attribute.
3. The attribute of the probability is m // cluster base belonging to the inner clustering tuple.
4. If the cluster base is 1
5. Then the tuple probability is also 1.
6. Else it is all the value of attributes in the cluster // the number of attribute value of inner clustering tuple.
7. }

Based on most of queries of TPC-H are joins between non-key and key, so it is the most common IC. As for different categories of queries and constraints, some of the existing queries are not based on the rewritten query and they are NP-complete (Cali *et al*., 2003). However, the actual numbers of queries are easy to calculate. In the calculation of the complexity, 20 of the 22 TPC-H queries are the PTIME (Cali *et al*., 2003). A given Method (Arenas *et al*., 1999): if given $q = \exists x, y, z: R_1(x, y) \exists R_2(y, z)$, it can get are written queries $Q = \exists x, y, z : R_1(x, y) \exists R_2(y, z) \exists \exists y': (R_1(x, y') \exists z': R_2(y', z'))$. The method mentioned above is based on the rewritten query and focuses on key IC. Therefore, it must judge whether the query can be rewritten or not before the query is given. As the join of queries can be rewritten, so there is no need to judge again, it only needs to consider SPJ queries.

Algorithm 2
Input: SPJ query
Output: whether it is a can be rewritten query
1. C← predicate set;
2. p← | C |;
3. FOR k = 1 TO p DO
4. IF C [k] is not a key-key join or non-key-key join
5. RETRUN false; End the procedure;
6. ENDFOR
7. RETRUN true;

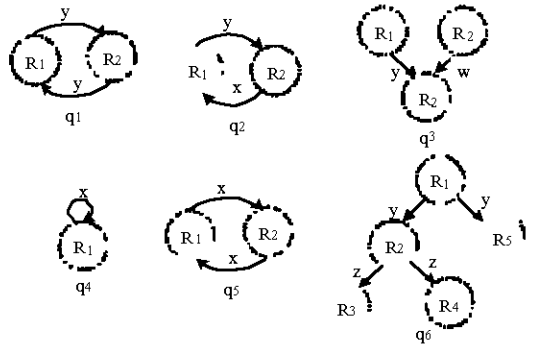

Fig. 1: Directional join graph

Algorithm 2 initializes SPJ query's the number of predicates and the predicate set first. By judging the predicate whether is a key-key join or non-key-key join predicate (the most common IC), it can determine whether it is a can be rewritten SPJ query.

The relationship of Fig. 1: customer means the account balance of store customer (balance attribute), cluster attribute means cluster and pr attribute means tuple probability.

In customer relation, the probability's summation of the inner clustering tuple is 1, the tuple range's probability is (0, 1]. If query the customers that their accounts are less than 80, it can get {(bc1, 0.7), (bc1, 0.1), (bc2, 1)}. Since bc1 appears twice, it is inconsistent. If grouping the clusters and calculating the cluster probability, then it can get ((bc1, 0.8), (bc2, 1)), both of bc1 and bc2 are credible clusters. Therefore, the written query of the credible cluster probability only needs to group the clusters. For example, if input a statement as the following:

- select E from R where W

where, E for expression, R for relations and W for a chosen predicate.

Then, the rewritten query is the following statement:

- select E, cluster, sum(pr) from R where W group by cluster, E

Because aggregation queries and SPJ queries are relatively complex, they will be researched in future works.

EXPERIMENTAL ANALYSIS

Experimental environment of evaluation the rewriting method: CPU is Core i3, Memory is 2GB, operating system is windows 7 and DBMS is SQL Server2008. Experiments

Table 1: An Inconsistent of customer relation

Cluster	Balance	pr
bc1	50	0.7
bc1	10	0.1
bc1	90	0.2
bc2	70	1

using standard TPC-H query 1 and 6, aggregation function is removed in the query and the use of standards that the query advised recommends the query parameters. It can establish a non-uniqueness index outside of the main foreign key of the cluster indicators and the relationship. Assume that the main key of the database model TPC-H specification is consistent, so in order to produce different parameters of the application of methods of assessment data, it can use a UIS data generated tool (Hernandez and Stolfo, 1998). Given parameters to control and contain the number of tuple of data may be inconsistent with the number of cluster; the resulting data will be designated in line with the TPC-H model (TPC, 2013).

- Scaling Factor (SF): Generate a designated TPC-H model to be used to control relations size, e.g., SF = 1, gets data 1GB, it is about 8,000,000 tuples
- Inconsistency factor (IF): used to determine the cluster base
- Inconsistency percent (IP): it is used to determine the percentage of non-uniformity of the database. E.g., among the 8,000,000 tuples, if IP is 0.5, then the number of inconsistent tuples is 4,000,000

IF and IP decide the level of inconsistency of database together. In the experiment, SF = 0.1, 0.5,1 and 2, respectively, IF = 1, 2, 4, 16, IP value is 0, 0.1, 0.2, 0.5. It should be noted that if the cluster base is set to 1, then the number of the inner clustering is 1, that is, the data tuple is certain, its probability is 1, meaning that all data is consistent with the aim of a reference to the baseline.

Firstly, the calculation of the probability of the tuple should be evaluated and then have a study of the implementation of the query rewriting. It can employ a relation Table 1 lineitem, its size is SF = 2, IF parameters for 1, 2, 5 and 25. As shown in Table 2, the implementation time is less than 14 min; this is an acceptable execute time. Obviously, the probability calculation of the time is increasing with the growth of IF parameters, which are mainly because the time spent in the difference of calculation of the cluster's core tuples and that of other tuple's. Therefore, when IF = 1 or 2, the calculating time has no significant growth; but when IF = 5 or 25, there is a significant increase in calculating time.

In Table 3, implementation of the parameters of Q1 and Q6 is SF = 1, IF = 2 and IP = 0.1. The execution time of

Table 2: Probability calculation of lineitem

Clustering base	1	2	5	25
Execution time (seconds)	405	500	550	799

Table 3: The execution time of rewritten query

Query	Q1	Rewritten query Q1	Q6	Rewritten query Q6
Execution time (seconds)	47	187	45	131

rewritten queries Q1 and Q6 have a large but acceptable difference in time, if compared with the initial queries. The reason is that Q1 has 6 projection attributes, Q6 has 2 projection attributes and the load of the rewritten query is decided by the projection attributes.

CONCLUSION

There is a combination of the probability of the original database technology in this paper. By analyzing the join graph of corresponding query whether is a directional join graph£¬and based on the cluster that generated by tuple match, it can judge whether the query is in two type rewritten queries (key to non-key join and key to key join). Moreover, it gives the methods of the probability calculation and the basic query rewriting. This approach complements and also extends previous approaches on the probability and CQA researching, the experiment shows that our approach is intuitive and effective for query performance by using TPC-H decision supporting benchmark data. The next step in the expansion of query needs to consider extending the semantic relationship for aggregation queries.

ACKNOWLEDGEMENTS

This study is supported by the Research Foundation of Education Committee of Hunan Province, China (09A046, 11B104, 12C0999, 13A046), the Hunan Provincial Natural Science Foundation of China (12JJ3057, 12JJ2040), the Hunan Science and Technology Plan of China (2010TT2055, 2011FJ3033, 2012GK3073, 2013NK3090), the Hunan Provincial Philosophy and Social Sciences Foundation of China (12YBA267), the Aid program for Science and Technology Innovative Research Team in Higher Educational Institute of Hunan Province and the Construct Program of the Key Discipline in Hunan Province, China.

REFERENCES

Ananthakrishna, R., S. Chaudhuri and V. Ganti, 2002. Eliminating fuzzy duplicates in data warehouses. Proceedings of the 28th VLDB Conference, August 20-23, 2002, Hong Kong, China, pp: 586-597.

Andritsos, P., A. Fuxman and R.J. Miller, 2006. Clean answers over dirty databases: A probabilistic approach. Proceeding of the IEEE 22nd International Conference on Data Engineering Workshops, April 3-7, 2006, Washington, DC, USA., 30.

Arenas, M., L. Bertossi and J. Chomicki, 1999. Consistent query answers in inconsistent databases. Proceedings of the 18th ACM Sigmod-Sigact-Sigart Symposium on Principles of Database Systems, May 31-June 3, 1999, Philadelphia, PA, USA., pp: 68-79.

Cali, A., D. Lembo and R. Rosati, 2003. On the decidability and complexity of query answering over inconsistent and incomplete databases. Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, New York, pp: 260-271.

Dalvi, N. and D. Suciu, 2004. Efficient query evaluation on probabilistic databases. Proceedings of the 30th Very Large Data Bases Conference, (VLDB'04), Toronto, Canada, pp: 864-875.

Fuhr, N. and T. Rolleke, 1997. A probabilistic relational algebra for the integration of information retrieval and database systems. ACM Trans. Inform. Syst., 15: 32-66.

Hernandez, M.A. and S.J. Stolfo, 1998. Real-world data is dirty: Data cleansing and the merge/purge problem. Data Mining Knowledge Discov., 2: 9-37.

TPC., 2013. TPC Benchmark™ H standard specification revision 2.16.0. Transaction Processing Performance Council, San Francisco, CA. http://www.tpc.org/tpch/ default.asp

Xie, D., Chen, X.B. and Y. Zhu, 2012. Obtaining certain results by query rewriting over uncertain database. Advance. Intell. Soft Comput., 162: 401-405.

Xie, D., Chen, X.B. and Y. Zhu, 2012. Tackling polytype queries in inconsistent databases: Theory and algorithm. J. Software, 7: 1861-1866.