January 2014

AN INTEGRATION APPROACH FOR HETEROGENEOUS DISTRIBUTED DATABASE TRANSACTION PROTOCOLS

DHARAVATH RAMESH

Department of CSE, Indian School of Mines (ISM), ramsjworld@gmail.com

Saritha Vemuri

Pulipati Prasad Engineering College, Khammam, A.P, sarithak.vemuri@gmail.com

VIJAY KUMARV

Department of CSE, S V E C, Suryapet, India, vijayvallamdas@gmail.com

Follow this and additional works at: <https://www.interscience.in/ijcsi>

Part of the [Computer Engineering Commons](#), [Information Security Commons](#), and the [Systems and Communications Commons](#)

Recommended Citation

RAMESH, DHARAVATH; Vemuri, Saritha; and KUMARV, VIJAY (2014) "AN INTEGRATION APPROACH FOR HETEROGENEOUS DISTRIBUTED DATABASE TRANSACTION PROTOCOLS," *International Journal of Computer Science and Informatics*: Vol. 3 : Iss. 3 , Article 11.

DOI: 10.47893/IJCSI.2014.1146

Available at: <https://www.interscience.in/ijcsi/vol3/iss3/11>

This Article is brought to you for free and open access by the Interscience Journals at Interscience Research Network. It has been accepted for inclusion in International Journal of Computer Science and Informatics by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

AN INTEGRATION APPROACH FOR HETEROGENEOUS DISTRIBUTED DATABASE TRANSACTION PROTOCOLS

DHARAVATH RAMESH¹, RAVURI SARITHA² & VIJAY KUMARV³

¹Department of CSE, Indian School of Mines (ISM)

²Department of CSE, P P I T S, khammam, India

³Department of CSE, S V E C, Suryapet, India

Email: ramsjworld@gmail.com, sarithak.vemuri@gmail.com, vijayvallandas@gmail.com

Abstract— Integration is commonly used for summarizing information held in very large databases which are encountered in data warehousing, large scale transaction management, and statistical databases. Such applications often involve distributed databases that have developed independently and therefore may exhibit incompatibility, heterogeneity, and data inconsistency. We are here concerned with the integration that has heterogeneous classification schemes where local ontology's, in the form of such classification schemes, may be mapped onto a common ontology. Existing transaction protocols typically rely on compensating transactions to handle exceptional conditions. In this paper, we develop an approach that can handle data inconsistencies and identify a number of issues with compensation-based transaction protocols and describe another transaction protocol that addresses those issues. Moreover, we define a set of properties, analogous to the ACID properties of traditional transactions that are more appropriate for business activities that span multiple enterprises. For this approach we use distributed transaction commit algorithm with EM algorithmic approach which is associated with Kullback-Leibler information divergence.

Keywords – *Distributed databases, ontologies, ACID properties, Kullback-Leibler information divergence, and EM algorithm.*

I. INTRODUCTION

Diverse database systems are used in large organizations. The heterogeneous distributed database system (HDDBS) can provide a flexible integration of diverse databases for users and applications. This is because it allows for retrieval and update of distributed data under different data systems giving the illusion of accessing a single centralized database system.

A heterogeneous distributed database system (HDDBS) integrates a set of autonomous database systems to provide global database functions. In a HDDBS environment, transaction management is handled at both the global and local levels. As a confederation of pre-existing local databases, the overriding concern of any HDDBS must be the preservation of local autonomy. This is accomplished through the superimposition of a global transaction manager upon a set of local database systems. Global transactions are submitted to the global transaction manager, where they are parsed into a set of global sub transactions to be individually submitted to local transaction management systems at local sites. At the same time, local transactions are directly submitted to the local transaction management systems. Each local transaction management system maintains the correct execution of both local and global sub transactions at its site. It is left to the global transaction manager to maintain the correct execution of global transactions. The preservation of the atomicity and isolation of global transactions is fundamental in achieving the correct execution of global transactions. Preserving

the atomicity or semantic atomicity of global transactions in the HDDBS systems has been recognized as an open and difficult issue. The traditional two-phase commit protocol developed in distributed database environments has been shown to be inadequate to the preservation of the atomicity of global transactions in the HDDBS environment. For example, some local database systems may not support a visible prepare-to-commit state, in which a transaction has not yet been committed but is guaranteed the ability to commit. In such situations, a local database system that participates in a HDDBS environment may unilaterally abort a global sub transaction without agreement.

A. Main Contributions

In a distributed database environment, heterogeneity may arise due to differences in the granularity of data stored in different distributed databases or may be due to differences in the underlying concepts. In this paper, integration is on the basis of local ontologies, in the form of classification schemes, which map onto a common ontology. If computable, the aggregates are then derived by minimization of the Kullback-Leibler information divergence using the EM (Expectation-Maximization) algorithm. The EM algorithm is used to provide an intuitive approach to the integration of aggregates by minimizing the Kullback-Leibler information divergence.

A significant contribution of this approach in extending the work [5], [6] is that inconsistent data can now be handled.

The novelty of this paper resides mainly in the provision of a scalable methodology for the integration of distributed aggregates by first determining the dynamic shared ontology and then carrying out the integration. The EM (Expectation-Maximization) algorithm is a widely used general class of iterative procedures used for learning in the presence of missing information; its use here for distributed database integration is also novel.

Compensating transactions can be difficult to design and program and even more difficult to test in a distributed Web Service environment, resulting in a rather high level of errors. Even if the compensating transactions are designed and programmed correctly, it is difficult to ensure that when they are needed, the compensating transactions are applied correctly (for example, the number of retries must be limited for practical purposes). The typical result of an error in the design, programming, or deployment of a compensating transaction is inconsistency in the databases, either within a single database or, more probably, between the databases of different enterprises. It is difficult and expensive to resolve such inconsistencies, particularly when the databases are spread across multiple enterprises, as is the intention of Web Services. We demonstrate that the use of compensating transactions results in a much higher probability of database inconsistency than does the reservation protocol. Indeed, for entirely reasonable parameters, the probability of inconsistency approaches unity when using compensating transactions.

II. THE DATA MODELS

A transaction is a set of operations on the application state that satisfies the following ACID properties [12]:

- *Atomicity.* Either all of the operations of the transaction succeed, in which case the transaction commits, or none of the operations is carried out, in which case the transaction aborts.
- *Consistency.* If the application state is consistent at the beginning of the transaction, the application state remains consistent after the transaction commits.
- *Isolation.* The transaction does not read or overwrite intermediate results produced by another transaction, that is, the transactions appear to execute serially.
- *Durability.* The updates to the application state become permanent (or persist) once the transaction is committed, even if a fault occurs.

A. System model for Computations

It is usually the case that, where data are distributed, there is a common ontology that specifies how the

local semantics correspond to the global meaning of the data. The relationship between the local and global schemes is held in a correspondence table. Thus, for example, using the Indian data on credit-worthiness discussed in [7], we have the common ontology for the attribute two different local classification schemes are shown in Table 2, along with aggregates produced as a result of extracting views of data from two different databases. Scheme 1 contains aggregates for a sample using the four high-level categories: "home/leisure," "office/professional," "car," and "other." Scheme 2 contains corresponding aggregates for a second sample differently classified into five high-level categories: "entertainment," "home," "skills," "work/travel," and "other."

Table 1 Common ontology for "loan" attributes

Attribute	Value	Attribute	Value
1.	Education	6.	Furniture
2.	Appliances	7.	New car
3.	Business	8.	Used car
4.	Repairs	9.	Radio/TV
5.	retraining	10.	other

Table 2 Two Different Local Classification Schemes for the "Reason for Loan Request" Attribute

Scheme1		
Scheme1 categories	Ontology categories	Cardinality
1.home	2,6,9	238
2.office	1,3,5	81
3.car	4,7,8	173
4.other	10	8

Scheme2		
Scheme2 categories	Ontology categories	Cardinality
1.entertainment	9	130
2.home	2,6	105
3.skills	1,5	26
4.work/travel	3,4,7,8	235
5.other	10	4

The correspondence table is presented in Table 3. For each relevant attribute, there is an associated correspondence table in which rows represent the local databases, together with associated tables that hold the corresponding aggregates for each classification scheme, as presented in Table 2.

Table 3 Correspondence Tables for the "Reason for Loan Request" Attribute

Common ontology	1	2	3	4	5	6	7	8	9	10
Local scheme1	2	1	2	3	2	1	3	3	1	4

Local scheme2	3	2	4	4	3	2	4	4	1	5
------------------	---	---	---	---	---	---	---	---	---	---

A task is a short-duration unit of work that is executed as a sequence of one or more local transactions. A task can be modelled as an operation on one or more resources and typically modifies some of the attributes of those resources. For example, the task of purchasing/selling certain kinds of goods (application-defined resources) such as automobiles can be modelled as an operation in which the owner attribute of the resource is changed from the supplier to the buyer. An operation can be read-only, in which case the task is a read-only task.

In a business activity, the ACID properties of a traditional transaction are not appropriate for the following reasons. First, it is not necessary that all of the participants see the same result, an observation that has been well recognized in practice. For example, if reservations are requested from several alternative participants, typically, one participant will see a confirmation, and the other participants will see a cancellation. The Web Services Business Activity (WS-BA) specification [5] defines two outcome types, atomic outcome and mixed outcome, to support different kinds of business activities, including those where only some of the participants agree on the same outcome. Thus, the failure of one task does not need to result in the rollback of the entire business activity. Moreover, the effect of executing a business activity might not be completely reversible, because of the business logic.

III. DATA INTEGRATION

We assume that the participants in a business activity are subject to crash faults but not arbitrary (Byzantine) faults. We present more details on faults within the participants in a business activity and discuss how those faults are handled later as part of the description of the reservation protocol instantiation. Moreover, we assume that the communication between different participants in a business activity is reliable. An implementation of a specification such as the Web Services Reliable Messaging (WS-RM) specification [16] can be used to achieve reliable communication.

We now develop an algorithm that allows us to integrate different aggregate views where the aggregates exhibit classification scheme heterogeneity. Such problems may not in general have a unique solution, but, in our case, the process of reduction to the dynamic shared ontology is guaranteed to lead to uniqueness.

A. Dynamic shared ontology

We consider an attribute A with corresponding domain $D = (v_1; \dots; v_k)$. Then, for aggregate view V_r ,

the domain D is partitioned into sets $(r_1 \dots r_g)$ for $r = 1 \dots m$, where $nrs = \{\text{cardinality of set } Srs\}$. Here, rg is the number of categories in the classification scheme of attributes A in view V_r .

The EM algorithm is an intuitively appealing approach that is here inherently reasonable since it maximizes both the likelihood and the information divergence. The algorithm proceeds by calculating those values of the integrated table that is unknown. If complete knowledge at the shared ontology granularity were available, the probabilities would be estimated by simply dividing the number of cases that take the respective values by the total cardinality. Where the number of cases that take a particular value is unknown, the number is replaced by apportioning the corresponding data from the local databases that might take that value.

B. Algorithmic Approach

We now derive the formulas of the EM algorithm for this problem following standard notations. We start with an initial estimate θ^0 (which can be either the ANOVA estimate or any other reasonable estimate). At the $(m + 1)^{\text{th}}$ iteration, we update the current estimate θ^m by completing the E-step and the M-step as follows.

E Step:

$$Q(\theta, \theta^{(m)}) = E[L_r(\theta) | \theta^{(m)}, \text{observed data}]$$

The computation is simplified to:

$$Q(\theta, \theta^{(m)}) = \sum_{i=1}^n \sum_{j=1}^3 \sum_{k=1}^3 \sum_{l=1}^3 \delta_{i,jkl}^{(m+1)} l_i(\theta),$$

Where $\delta_{i,jkl}^{(m+1)} = \delta_{i,jkl}(\theta^{(m)})$ denotes the $\Pr(X_{1,i} = j, X_{2,i} = k, X_{3,i} = l | \text{observed data and } \theta^m)$. It can be computed according to the following formula: If Y_i is observed,

If Y_i is missing,

$$\delta_{i,jkl}^{(m+1)} = \frac{p_{j,k,l}^{(m)}}{\sum_{j \in X_{1,i}} \sum_{k \in X_{2,i}} \sum_{l \in X_{3,i}} p_{j,k,l}^{(m)}} \phi\{j \in X_{1,i}, k \in X_{2,i}, l \in X_{3,i}\}$$

M Step:

Update the parameter estimate to the value $\theta = \theta^{(m+1)}$ that maximizes $Q\theta, \theta^m$. The maximization over θ becomes rather simple if we further write out the expression

$$Q(\theta, \theta^{(m)}) = \sum_{i=1}^n \sum_{j=1}^3 \sum_{k=1}^3 \sum_{l=1}^3 \delta_{i,jkl}^{(m+1)} \log(p_{j,k,l}) - \sum_{i \in \text{obs}(Y)} \sum_{j=1}^3 \sum_{k=1}^3 \sum_{l=1}^3 \delta_{i,jkl}^{(m+1)} \frac{(Y_i - \mu_j)^2}{2\sigma^2} - n_{\text{obs}(Y)} \log(\sqrt{2\pi}\sigma)$$

Here, $\text{obs}(Y)$ denotes the set of i 's where Y_i is observed, and $n_{\text{obs}(Y)} = |\text{obs}(Y)|$. The maximization of the above expression is very similar to a linear model

and we find explicitly the following updating formula:

$$p_{j,k,l}^{(m+1)} = \sum_{i=1}^n \delta_{i,jkl}^{(m+1)} / n, j = 1, 2, 3$$

$$k = 1, 2, 3, l = 1, 2, 3.$$

Method: Distributed Transaction Commit:

```

SendMessages( round, std::vector<Process>
&processes ) {
for ( size_t i = 0 ; i < m[ round ][ mId ].size() ; i++ ) {
Path srce_node_path = m[ round ][ mId ][ i ];
srce_node_p = srce_n_p.substr( 0, srce_n_p.size() - 1 );
Node: srce_node = mNodes[ source_node_path ];
for ( size_t j = 0 ; j < mFaults.mN ; j++ )
if ( j != mNodes.mSource )
{
char value = mFaults.GetValue( srce_n.input_v, mId,
(int) j, mPathsByRank[ round ][ mId ][ i ] );
if ( mNodes.mDebug )
std::cout << "Sending from process " << mId << " to "
<< static_cast<unsigned int>( j )
<< ": { " << value << ", " << m[ round ][ mId ][ i ]
<< ", " << UNKNOWN << " } "
<< ", getting value from source_node " <<
srce_n_path << "\n";
proc[ j ].ReceiveMessg( m[ round ][ mId ][ i ],
Node( value, UNKNOWN ) );
} } }

```

Fig 1 Source Code for Transaction Commit Protocol

The results are presented in Table 4. We note that, where categories cannot be separated, however, such situations may be readily identified from the correspondence table since these categories have equal rows that are equal to no other row; collapsing could therefore be carried out in advance of computing the EM solution. The cardinalities in Table 4 are given by $N * \parallel i$, where $N (=1000)$ is the cardinality.

TABLE 4 Derived Cardinalities for the Indian Data

Common ontology category	Cardinality	Common ontology category	Cardinality
1.Education	25.7	6. Furniture	105.7
2.Appliances	105.7	7. New car	116.9
3. Business	112.9	8. Used car	116.9
4. Repairs	116.9	9.Radio/TV	261.7
5. retraining	25.7	10. other	12

IV. AVAILABILITY ANALYSIS

For the purposes of the availability and consistency analyses, we model a business activity as a sequence of one or more tasks, each of which is a traditional ACID transaction. The concurrency of tasks within the business activity has no effect on our availability and consistency analyses.

We consider only faults that are detected immediately so that a transaction can be aborted and retried immediately. Detection might involve the operating system, the database, the transaction or communication middleware, and custom coded application data validity checks. We do not consider faults that allow a transaction to appear to complete even though they yield incorrect results that were not detected. Following the detection of a fault in a transaction, the transaction or the entire business activity is aborted and retried. We assume that the standard transaction commits and abort mechanisms operate correctly. We consider only a single retry of the business activity and a single retry of compensating transactions when they are used.

We investigate the probability that all or a large number of business activities will complete successfully. Many businesses must process thousands or millions of activities every day. Each activity that does not complete successfully can involve difficult and expensive manual intervention. We investigate the probability that the databases of the business activity might be left in an inconsistent state, an inconsistency that might spread across multiple enterprises. Even a potential inconsistency can require difficult and expensive manual intervention. There is no intention that the calculations presented here provide accurate probabilities for any particular business application. They are intended only to investigate the effects on availability and consistency for the particular architecture chosen.

V. EVALUATING THE PERFORMANCE

Scalability of the proposed algorithm for use in problem domains involving the integration of large numbers of databases is of critical importance. Here, we quantify heterogeneity by an overlap metric:

Fig 2 Execution times by number of categories for 15 schemes.

The tests were performed on a PC with a 133 MHz processor and a summary of the results for the test runs are presented in Figs. 1, 2, and 3. Here, the initial probabilities are uniform and each result is an average taken from 100 test runs. Figs. 1 and 2 shows that the time complexity is approximately linear with respect to the number of categories in both the

common ontology and the number of schema. Also, as the overlap increases, the execution time also increases. However, this increase is modest for realistic overlap values. Further, in Fig. 3, we demonstrate that the efficiency of the algorithm is not reduced by the presence of inconsistent data (see Fig. 3).

Fig 3 Execution times by number of schemes for 20 categories.

Fig 4 Consistent and inconsistent data for 15 schemes and overlap 0.3.

VI. CONCLUSIONS AND FUTURE SCOPE

The nature of work has been implemented new criterion on the execution of local and flexible transactions in the HDDBS environment. We have implemented a theory which facilitates the maintenance of global transactions, a concurrency control criterion that is stricter than global serializability in that it prevents the flexible transactions which are serialized Betties a flexible transaction and its compensating sub transactions to affect any data items that have been updated by the flexible transaction.

In this paper, we have presented an approach to data set integration that is capable of accommodating the allocation of dynamic shared ontology category values in the presence of aggregate inconsistencies and is therefore potentially appropriate for problems involving the integration of large numbers of databases. Our current methodology allows a user to

answer a query involving heterogeneous aggregate views by first determining whether the information in the participating views is sufficient to answer the query, either fully or partially, and, if it is, then providing appropriate solutions. Such an approach has potential for application to interactive query processing in a distributed environment and could therefore, for example, assist users submitting queries over the internet that require retrieval of data from multiple data sources.

ACKNOWLEDGEMENTS

This paper is an extended and revised version of a paper, with the title, that appears in the proceedings of the *IEEE International Conference on A Scalable approach to Integrating Heterogeneous Aggregate views of Distributed Databases*. The authors would like to thank Sally McClean and Bryan Scotney, and Kieran Greer for their contribution. We are also very grateful to the reviewers for their helpful presentation suggestions on the earlier version of this paper.

REFERENCES

- [1] F.M. Malvestuto, "A Universal-Scheme Approach to Statistical Databases Containing Homogeneous Summary Tables," *ACM Trans. Database Systems*, vol. 18, pp. 678-708, 1993.
- [2] S.I. McClean, B.W. Scotney, and C.M. Shapcott, "Incorporating Domain Knowledge into Attribute-Oriented Data Mining," *Int'l J. Intelligent Systems*, vol. 6, pp. 535-548, 2000.
- [3] S.I. McClean, B.W. Scotney, and C.M. Shapcott, "Aggregation of Imprecise and Uncertain Information in Databases," *IEEE Trans. Knowledge and Data Eng.*, vol. 13, no. 6, pp. 902-912, Nov./Dec. 2001.
- [4] B.W. Scotney and S.I. McClean, "Efficient Knowledge Discovery through the Integration of Heterogeneous Data," *Information and Software Technology*, (special issue on knowledge discovery and data mining), vol. 41, pp. 569-578, 1999.
- [5] B.W. Scotney, S.I. McClean, and M.C. Rodgers, "Optimal and Efficient Integration of Heterogeneous Summary Tables in a Distributed Database," *Data and Knowledge Eng.*, v A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc., Series B*, vol. 39, pp. 1-38, 1977. vol. 29, pp. 337-350, 1999.
- [6] The Statlog Partners, "Dataset Descriptions and Results," *Machine Learning, Neural and Statistical Classification*, pp. 131-174, 1994.
- [7] Y. Vardi and D. Lee, "From Image Deblurring to Optimal Investments: Maximum Likelihood Solutions for Positive Linear Inverse Problems (with discussion)," *J. Royal Statistical Soc., Series B*, pp. 569-612, 1993.
- [8] P. Ammann and S. Jajodia, "A Timestamp Ordering Algorithm for Secure, Single-Version, Multi-Level Databases," *Database Security, II: Status and Prospects*, C. Landihr and S. Jajodia, eds., pp. 23-25. NorthHolland, 1992.
- [9] V. Atluri and S. Jajodia, "ISR-Consistency: A New Notion of Correctness for Multilevel Secure, Multiversion Database Management Systems," *Proc. 19th Latin Am. Informatics Conf.*, pp. 195-214, Buenos Aires, Argentina, Aug. 1993

-
- [10] D.E. Bell and L.J. LaPadula, "Secure Computer Systems: Unified Exposition and Multics Interpretation," Technical Report MTR-2997, The Mitre Corp., Bedford, Mass., Mar. 1976.
 - [11] P. Bernstein, V. Hadzilacos, and N. Goodman, *Concurrency Control and Recovery in Database Systems*. Addison-Isley, 1987.
 - [12] J. Gray and A. Reuter, *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993.
 - [13] Committee on Multilevel Data Management Security, *Multilevel Data Management Security*. Washington, D.C.: Air Force Studies Board, Nat'l Research Council, 1983
 - [14] A. Elmagarmid, *Database Transaction Models for Advanced Applications*. Morgan-Kaufmann Publishing Company, 1992.
 - [15] S. McFarling, "Procedure Merging with Instruction Caches," *Proc. SIGPLAN 1991 Conf. Programming Language Design and Implementation*, pp. 71-79, June 1991.
 - [16] H. Garcia-Molina and B. Kogan, "Achieving High Availability in Distributed Databases," *IEEE Trans. Software Eng.*, vol. 14, no. 7, pp. 886-896, July 1988.
 - [17] J.N. Gray, R.A. Lorie, G.R. Putzolu, and I.L. Traiger, "Granularity of Locks and Degrees of Consistency in a Shared Data Base," *Modelling in Data Base Management Systems*, pp. 365-394, 1976.
 - [18] S. Jajodia and B. Kogan, "Transaction Processing in Multilevel Secure Databases Using Replicated Architecture," *Proc. IEEE Symp. Research in Security and Privacy*, pp. 360-368, May 1990.
 - [19] T.F. Keefe and W.T. Tsai, "Multiversion Concurrency Control for Multilevel Secure Database Systems," *Proc. 10th IEEE Symp. Research in Security and Privacy*, pp. 369-383, May 1990.
 - [20] B. Kogan and S. Jajodia, "Secure Concurrency Control," *Proc. Third RADC Workshop Multilevel Database Security*, Castille, N.Y., June 1990. Published as MITRE Technical Report MTP 385, B. Thuraisingham, ed.
 - [21] W.T. Maimone and I.B. Greenberg, "Single-level multiversion schedulers for multilevel secure database systems," *Proc. Sixth Ann. Computer Security Applications Conf.*, pp. 137-147, Tucson, Ariz., Dec. 1990.