[1]Rafea Mohammed Ibrahim

Data Synchronization for Distributed Heterogeneous Database

Abstract: **-** Heterogeneous distributed database concurrent data is one of the important topics of heterogeneous databases that have important meaning. This paper discusses the mechanism of optimizing data synchronization for distributed heterogeneous database and trying to propose real solutions.

Database synchronization is the exchange of records in databases on the condition that they are different. When modifications are made to a specific database, they will be applied to the database that has been synchronized in different environments and servers horizontally. The synchronization is either a one-way or two-way application and can be incidental, i.e. it is synchronous to the local network and asynchronous to the Internet due to its lagging technology.

Heterogeneous distributed database systems consisting of a row storage database and a column storage database will face many difficulties, due to the inconsistent storage type such as: synchronization speed mismatch in data synchronization compared to a homogeneously distributed database system.

Heterogeneous database management systems work on exchanging information for a package of various formats, accessing modification information in a heterogeneous database, maintaining the independence of the proposed system, and ensuring a good application of the proposed system in order to reach data synchronization between several packages of databases.

Keywords: Heterogeneous, Synchronization, Lagging Technology

1. Introduction

The data sources considered for information integration are distributed around the world. Therefore, not only can systems containing data sources be separate, they can exist in remote locations without being connected by only one common network. All data sources may be in a variety of formats depending on the underlying technology used in them.

The study aims to analyze several types of characteristics of heterogeneous distributed databases found in ready-made devices and to explain the practical aspects of the internal improvement of their characteristics; That is, what is accessible there and which models are potentially more efficient.

The constructive approach to designing a distributed environment from a database that is unified by a logically separated distributed database system (DDBMS). During this standardization process, it may not even allow many foreign restrictions to apply the current database system in all locations (eg database dependency restrictions, etc.). [1]

When creating a distributed database environment, if it is possible to use a diverse database management system for some nodes that participate in this distributed database, then it is called a heterogeneous distributed database environment, as in the following diagram:

[1] Lecturer. The Iraqia University / College of Islamic Sciences Specialization Computer Science

rafeasweet@gmail.com

rafea.Ibrahim@iraqia.edu.iq

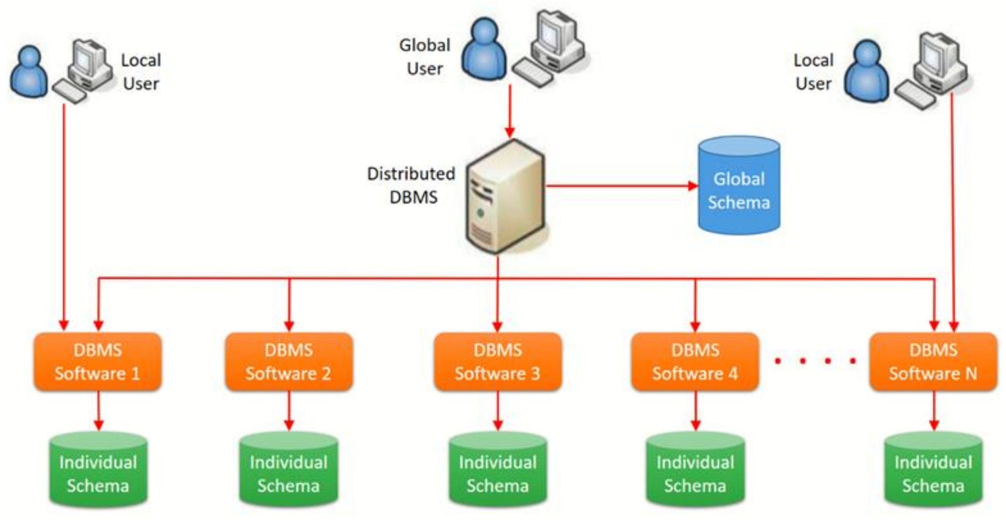

Figure 1: Heterogeneous distributed database environment

This paper proposes various methods for managing distributed and heterogeneous data and provides an explanation of the functions and structures of several of these systems, which are optimized for practical application. The goal is to perform a state-of-the-art analysis of target devices and manufacturing locations rather than research prototypes. [2]

2. Related Work

A proprietary protocol called Buffer has been considered by Google as a free variant since 2011 (accredited by Berkeley Software Distribution). Apache also maintains the powerful THRIFT system [3] that was first implemented on the Facebook platform. The two systems do an efficient job of serialization and deserialization (serialization) times, which is more efficient than the XML protocols. In addition, the dual format of Proto-Buf and Thrift packets reduces network bandwidth consumption without increasing processing time. [4, 5]
A model by Hall and Yap discusses the mathematical theory for studying data representation in databases called the coordination model. A format model consists of formal objects called "formats" specifying how to use them as a template for structuring data. Formats are defined using basic types and three constructors, namely attainment; Category; and installation. Base types correspond to the attributes in the relational model. The collection constructor is used to form collections of objects of a certain type. A relationship can be defined by a configuration set of basic types. The hierarchical model can be defined in the coordination model using alternate layers for grouping and composition operations. [6]
In a detailed study of heterogeneous data management prototyping systems, Sheth and Larson categorized these applications as federated or non-federated architectures. The term unified architecture was first introduced by Heimbigner and McLeod to refer to a system based on an organizational model that uses equal and independent databases. In unified systems data sharing and definition are controlled by real interfaces. [7]
Some models have recently been developed to address the problem of heterogeneity in databases. Among them: Multi-base, DDTS (Honeywell), IMDAS (National Institute of Standards and Technology), MRDSM Prototypes (INRIA, France) and PRECI (University of Aberdeen (Scotland, UK). There are many differences between these systems with regard to the application for which they are designed, the local data models, the supported database management systems, and the query processing tools. [8, 9]
In [10] two components that communicate and co-author an implementation of the popular JSON template are introduced. This system suggests that for efficient execution of the model, most machines have their own times. These two timestamps are compared to start the process.
In [11], mobile service providers have been implemented as an interface for distributed monolithic databases. The communication between the forms takes place via mobile network providers, where Aglet is appended to different search queries and directed to the desired target. In this system, the processing of selecting information from various proposals has been considered while our focus is on sending and receiving data.

The Distributed Database Testing System (DDTS) uses the E-C-R model and GORDAS as a general high-level language. The system is based on the relational model and therefore cannot provide support and transparency for non-relational data. The definition of data in DDTS is based on a five-level model. It contains a single federation model called the global representation model. The user only sees the external level diagram. The semantic description of the entire database is described at the conceptual level of the model. DDTS develops the semantic query as the first step in processing. [12]
The Amoco Distributed Data System (ADDS) is a distributed heterogeneous database control model designed by Amoco. The system can combine heterogeneous data stored in hierarchical and relational data models. The system does not keep an outline. User queries are formulated and compiled using information stored in local export schemas. [13]

3. Problem formulation

A centralized database design is required to control most packets of data in a distributed model. The process of data synchronization takes place in which the following issues can be resolved: Access modification data in a heterogeneous database. Heterogeneous DBMS can transform data, to obtain various formats. To ensure the manual operation of the current system, and to maintain the independence of the proposed model to achieve data synchronization between multiple databases.
In Synchronization Using XML, the researchers copied a write-up about synchronizing data in heterogeneous places across HTTP and JMS systems but did not mention how the model could be applied to preserve database features. [14]
Studies show that Wisdom-Force Database-Sync is very expensive and can synchronize data in a heterogeneous environment. Cannot detect a modification in the sample parent database. It also requires permission from Oracle Corporation. This makes it more expensive. For this reason, the modified information is supposed to be picked from the source database to use this solution. This is because it is dealing with a heterogeneous database, some databases do not provide access to the modified data. [15]

It is best to stress the importance of synchronization between existing heterogeneous distributed databases. After discussing the discrepancy between these databases and the responsibilities they have in the distribution contract as the operating environment, I made data synchronization models very difficult. A range of methods have been achieved to solve the proposed issue, although most of the processing methods use databases that depend on data such as timestamps. Other solutions have one-way communication while others are one way.
The aim of the study is to propose treatment methods for the indicated issues to improve the efficiency of everyday database engine software applications. So a method has been presented which has the following features.
- Parallelism: that is, each stage is dealt with alone, which makes it work at the same moment without external interference. This means that the sending and receiving of information can be synchronized.
- Routing alternatives: A hybrid scheme has been introduced to speed up possible routing alternatives for data exchange if one of the directions fails.

Bi-directional: data exchange can be in all paths, that is, all databases send and receive data.
- No timestamp: In the current model, synchronization extends from the sender to the receiver or vice versa as the information used is de-application to the database and the processing method is generally heterogeneous. [16]

3. Methodology

Database logs are an effective way to maintain and restore database integrity. It's where all playback packets of information that were sent are stored. The file is analyzed by studying database log data for updates in synchronizers. The database file contains all application information that was correctly transmitted. There is an approach to implementing a custom log study or interface to analyze the database logical file, because some database log formats cannot be played, to perform the query back into an SQL query and store the log in a special file. This file contains query time data, SQL data, etc. [17]
Microsoft has designed a database capture API to facilitate sharing of data between heterogeneous databases. To make the model work with several classes of data files, it is necessary to link flexibly with another engine, in order to complete API properties and to communicate with the data file. This method includes a combination of software packages, recovery of data resources, and this increases the price of the model software and hardware, and greatly increases the duration and effort of the system's technical support. [18]
In this study, the exchange message containing an update to the registry is called "Dsync". Dsync maintains the development root server, timestamp, and serialized object. All results are configured for Dsync after the sync results stage.
Each class in the database is supposed to be implemented at the program level by implementing a custom Dsync record. A Dsync record is accessed when a record is created, updated, or deleted. The generated Dsync record is also stored in a special queue of the database to ensure a proper step of serialization and to give the possibility of rebuilding. By creating a static queue, the system administrator can retrieve the position of the database at a specific point and enter most of the "Dsync" registers accurately, after correcting the issue that caused the defect. [19]

```
Require: Subscribed Nodes List LL
1. for (Li in LL)
   a. serialized_packet ← request_for_updates(Li)
   b. updated_rows_ li← deserialize(serialized_packet)
   c. updated_rows.append(updated_rows_ i)
   d. ACK(Li, updated_rows_i.size())
2. end For
3. arrange_Items_by_timestamp(updated_rows)
4. save(updated_rows)
```

Algorithm 1: Leader RFN - CRON Running

The main packages send queries to all shared packages requesting any updates (Algorithm 1). When the dependent package receives a query from the system administrator, it takes the "Dsync" qualifier from its database. To prevent overburdening the form with data, the maximum number of Dsync files to send in one message is a given build number. The Dsync record allowed to be sent to the sys-admin is all Dsync records designed internally. When solutions are submitted to the sys-admin, "Dsync" is stored in the waiting stack, sorted by time description. Then the sys-admin of the root server recognizes the accepted master components. These components are considered to be sent to the system administrator in the source server waiting packet (algorithm 2). [20]

```
Require:
1. updated_rows ←Fetch at most MPS rows from local
Dsync where “target server ids” contains L.getId()
and “ source node” equals S.getId()
2. packet_to_send ←serialize (updated_rows)
3. ack←answer_leader(packet_to_send)
4. if(ack.count() equals updated_rows.count())
   a. mark_as_delivered(updated_rows)
```

Algorithm 2: Sample sync workflow in Sync history

The same process used on each server will be used for adding, updating, and deleting. All of these actions must be performed using the same implementation and not using database applications directly.

At this point, the serialized data may be fast and efficient, and a long time can be added to each exchange sequence when converting the binary data into a modified version of the original information. This conversion is effective when there is a need to convert the collation to convert a column from the root package collation to the desired database collation.

In this study, all the actual operations that are applied to a given database in the application layer are kept within a queue for each packet as in Figure 2. The dynamically selected packet controls the object messaging protocol with a system to distribute most of the components of the queue packets to all shared servers. This packet attacks the network. A messaging protocol is a deliberate and ordered set of works, which are executed either on the primary end or in secondary packets. [21]

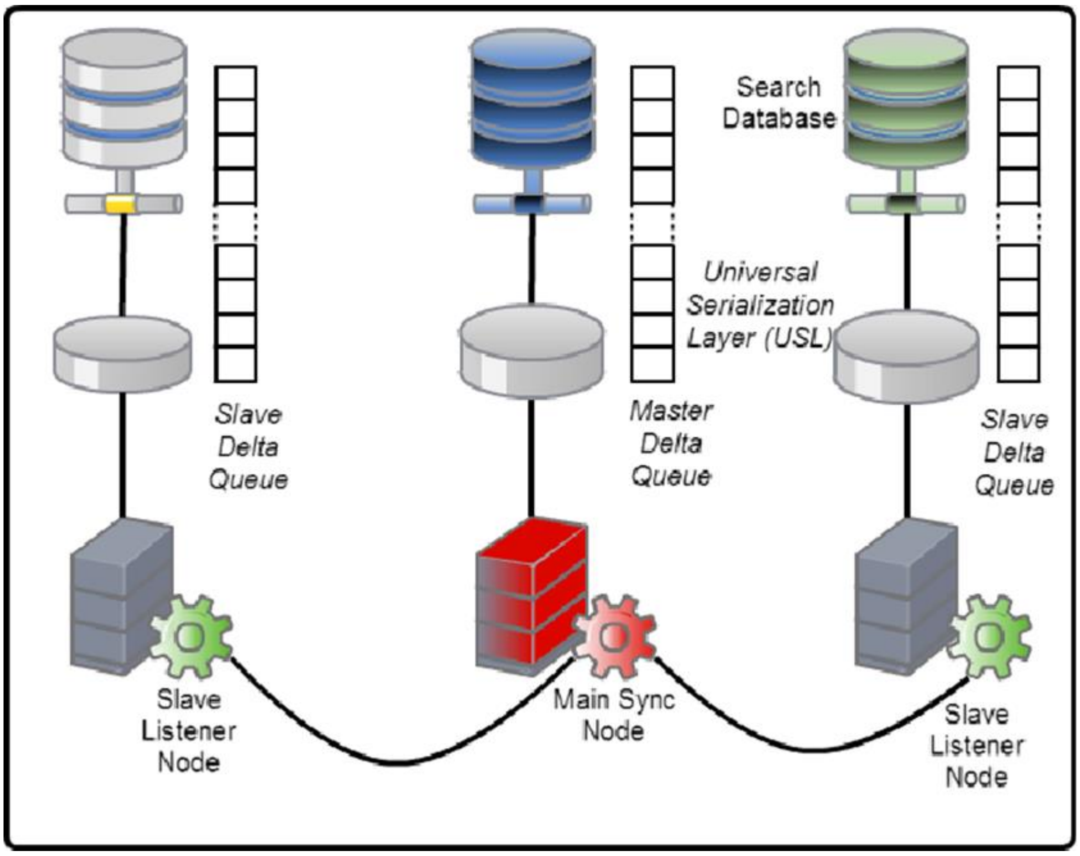

Figure 2: Dsync register based method

A central database was created to produce a model for the heterogeneous database, merging all the shared data of the member database, to create unified authentication methods and access management. The whole model is a star-shaped diagram, in the middle of which is a data transmission center that acts as a server. A data conversion center server is a medium for data exchange and conversion. One web service interface per package can be used to communicate data transformation support.

All data transformation forms of the data transformation package can be integrated with the transformation website by web developers. All data transformation models in a data transformation package can share a specific function in exchanging data with information system administrators for a heterogeneous source database. [22]

Conclusions

This study proposes a scheme for developing heterogeneous distributed database synchronization based on several aspects. Several approaches were used to access the data according to the log algorithm. The heterogeneous database integration method, and the synchronization method using web services, can resynchronize the master database and the root database.

The current model in this study introduces two-way exchange and data messaging. Particularly parallelism is a priority as data exchange takes place simultaneously on a specified server without collision.

Because of the variety of heterogeneous databases and their dependent characteristics in distribution packages, many solutions have been proposed to address the issues mentioned, though, some solutions are proprietary and use database-driven data such as temporal model and implementation.

In order to work with distributed models, the implementation of serialization is proposed at the same time. In order to aim at the shortcomings of the column storage database itself in the rendering method, the idea of INSERT rendering is proposed. The proposed method discusses the problem that real-time data synchronization cannot be implemented between heterogeneous databases with multiple storage methods, and solves the synchronization problem from distributed row storage databases to column storage databases.

References

1. M. Raynal, (2015) Distributed Algorithms for Message-Passing Systems. Berlin, Heidelberg: Springer Berlin Heidelberg.
2. Xing L, Mishra Y, Tian Y, Ledwich G, Zhou C, Du W, et al. (2019) Distributed state-of-charge balance control with event-triggered signal transmissions for multiple energy storage systems in smart grid.
3. M. Slee, A. Agarwal, and M. Kwiatkowski, “Thrift: Scalable cross-language services implementation,” (2015). [Online]. Available: http://trillian42.homelinux.org/dir/pdfs/thrift-20070401.pdf.
4. J. Delgado, “Service interoperability in the Internet of Things,” (2015) Stud. Comput. Intell., vol. 460.
5. M. Elbushra and J. Lindström, (2014) “Eventual Consistent Databases: State of the Art,” Open J. Databases, vol. 1.
6. Hall R., Yap C. (1994) "The Format Model: A Theory of Database Organization" Journal of the Vol. 31.
7. Sheth A., AND LARSON J. A. (1999). Federated databases: Architectures and integration. ACM Computer.
8. Shipman D. (1991) The functional data model and the data language DAPLEX. ACM Trans.
9. Xu Zijian et al. (2019): Data Synchronization Tool for Distributed Heterogeneous Databases.
10. Sethia, D., Mehta, S., Chodhary, A., Bhatt, K., Bhatnagar, S. (2015): MRDMS-Mobile Replicat¬ed Database Management Synchronization.
11. Gajjam, N., Apte, S.S. (2014): Mobile Agent based Communication Platform for Heterogeneous Distributed Database.
12. ELMASRI, R. (1991). GORDAS: A data definition, query and update language for the entity-category-relationship model of data.
13. Breitbart Y. J. , Silberschatz A., and Thompson G. R. (1997). An update mechanism for multidatabase systems.
14. W. Xiaoli and Y. Yuan, (2012) “Xml-based heterogeneous database integration system design and implementation,” in 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT).
15. “Wisdom-Force Database-Sync Real Time Change Data Capture and Replication,” (2018) http://www.wisdomforce.com.
16. R. Silhavy et al. (eds.),Software Engineering in Intelligent Systems, (2017) Advances in Intelligent Systems and Computing, Springer International Publishing Switzerland.
17. Chatterjee, N. Krishna, M. Semantic.(2009) Integration of Heterogeneous Databases on the Web. Computing Theory and Applications.
18. Yuanping Lei, (2010) Message mechanism to achieve synchronization of heterogeneous databases updated. Application Research of Computers.
19. W. Zhao and Z. Xiaohong, (2013) “The Transaction Processing of Heterogeneous Databases Application in Web Service.
20. Y. Wang and Y. Li, (2016) “Heterogeneous Data Sources Synchronization Based on Man-in-the-Middle Attack,” Proc. 4th Int. Conf. Comput. Eng. Networks.
21. C. M. Krishna, (2015) “Fault-tolerant scheduling in homogeneous real-time systems,” ACM Comput. Surv., vol. 46.
22. F. Meyer, B. Etzlinger, F. Hawatsch, and A. Springer, (2014) “A distributed particle-based belief propagation algorithm for cooperative simultaneous localization and synchronization,” in 2013 Asilomar Conference on Signals, Systems and Computers.