**DATABASE REPLICATION: A COMPARATIVE STUDY OF HETEROGENEOUS TOOLS FOR REPLICATION IN A DISTRIBUTED SYSTEM**

**Abiodun K. Moses,** **abbeykmos@yahoo.com**

**Aremu D. R.,** **draremu2006@gmail.com**

Department of Computer Science, University of Ilorin,

**ABSTRACT**

This paper presents a comparative study of Heterogeneous Tools for Replication in a Distributed System. The specific objectives of the study were to: investigate how to perform heterogeneous replication; analyse the different tools that can be used to achieve heterogeneous replication; and compare these tools with each other. The findings from the study show that tungsten replicator provides a very flexible and cost effective method, whereas the oracle golden gate would be the choice if money is not an hindrance. The MySQL native replication is also good, but lack the dexterity of tungsten replicator and oracle golden gate. By using replication mechanism of the different database system does not bring about high efficiency as it was envisage. The study helps in making sharing and accessing of vital information easy, even if stored in various database systems. This is very good for business all over the world where several organisations need to merge data for one reason or the other.

**Key Words:** Replication, Database-Replication; MySQL-Replicator, Tungsten-Replicator

**Introduction**

There is an increasing need to share resources across heterogeneous databases around the world. The challenges to get this done is enormous, because, data are stored in various database products such as MySQL, Oracle, DB2, SQL Server, and so on. Each of these databases has different database schema which needs to be brought together as one because users required data from various organizations. In the literature, replication has been widely employed for enhancing system accessibility (e.g. recovering data from disaster), data consolidation (for central audit or analysis), data distribution (for balancing access load, or for offline query), and so on. For database applications where users are geographically and widely distributed, replication is often the most efficient method of Database access [13]. A replication system maintains multiple copies of data objects, synchronizing any changes to that data. This paper presents an overview of replication technology in which users can replicate between heterogeneous database systems without any loss of data. The specific objectives of the study were to: (i) investigate how to perform heterogeneous replication; (ii) analyse the different tools that can be used to achieve heterogeneous replication; and (iii) compare these tools with each other. The rest part of the paper is organized as follows: Section 2 presents the related work; section 3 discussed the tools for Heterogeneous Replication; while section 4 Compare Tungsten Replicator and Oracle Golden Gate; and section 5 summarized and concluded the paper.

**2. Related Work**

Replication is the process of creation and maintenance of duplicate versions of database objects in a distributed database system [8]. Replication improves the performance and increases the availability of applications by providing alternate data access options. For example, users can access a local database rather than a remote server to minimize network traffic and provide location transparency. Furthermore, the application can continue to function if parts of the distributed database are down as replicas of the data might still be accessible. Database replication is needed in the case of a system failure where in if a primary copy of the database is failed the secondary copy will still be there to retain the data. A replication service is required to maintain data consistency across these diverse environments. Distribution reduces the network costs for query access, and it improves application availability and consistency.

**2.1 Types of Replication**

The replication tools may be selected based on type of replication it supports. The capabilities and performance characteristics varies from one type of replication to another. A replication strategy may be selected based on two basic characteristics: **Where**and **When**.

When the data is updated at one site, the updates have to be propagated to the respective replicas by Synchronous (eager) and Asynchronous (lazy) methods and where the updates can take place can be achieved by update everywhere and primary copy (master-slave) methods.

**Synchronous replication**(Master-Slave replication) works on the principle of Two-Phase commit protocol. In a two-phase commit protocol, when an update to the master database is requested, the master system connects to all other systems (slave databases), locks those databases at the record level and then updates them simultaneously. If one of the slaves is not available, the data may not be updated. The consistency of data is preserved; however it requires availability of all sites at the time of propagation of updates.

There exists two variations of *Asynchronous replication* (Store and Forward replication) that is Periodic and Aperiodic. In Periodic replication, the updates to data items are done at specific intervals and in aperiodic replication the updates are propagated only when necessary (usually based on firing of event in a trigger). The time at which the copies are inconsistent is an adjustable parameter which is application dependent In U**pdate anywhere** method, the update propagation can be initiated by any of the sites. All sites are allowed to update the copy of the datum whereas in a *Primary Copy* method there is only one copy (primary copy or master) which can be updated and all other (secondary or slave) copies are updated reflecting the changes to the master.

**2.2 Replication Strategies**

**Snapshot Replication**: In snapshot replication, a snapshot or copy of data is taken from one server and moved to another server or to another database on the same server. After the initial synchronization, snapshot replication can refresh data in published tables periodically. Though snapshot replication is easiest form of replication, it requires copying all data items each time a table is refreshed.

**Transactional Replication**: In transactional replication, the replication agent monitors the server for changes to the database and transmits those changes to the other backup servers [18]. This transmission can take place immediately or on periodic basis. Transactional Replication is used for server-server scenarios.

**Merge Replication**: Merge replication allows the replicas to work independently [18]. Both entities can work offline. When they are connected, the merge replication agent checks for changes on both sets of data and modifies each database accordingly. If transaction conflict occurs, it uses a predefined conflict resolution algorithm to achieve consistency. Merge replication is used mostly in wireless environments.

**Statement based replication**: The statement based replication intercepts every SQL query and sends it to different replicas [15]. Each replica (server) operates independently. To resolve conflicts, Read-Write queries are sent to all servers whereas read only queries can be sent to only one server. This enables the read workload to be distributed. Statement based replication is applicable for optimistic approaches where each cache maintains the same replica.

**2.3 Replicating between heterogeneous databases**

Heterogeneous replication between different databases means the changes that happened on database A need to be happened on database B, while A and B are from different vendors. Figure 1 shows a way of replicating between heterogeneous databases.



**Oracle**

**MySQL**

Figure 1: Replication between heterogeneous databases (Damian Dang, 2008)

In Figure 1, OGSA-DAI controls both master and replica databases. It gets changes from the Oracle database and applies them to the MySQL database. One of the critical problems is, though different vendors follow the instructions of SQL standard, they have lots of differences on implementation of their specific extensions.

Replication is a good example, there is no information on how to replicate in the standard Database Language SQL 1992. So every vendor has its own commands. Fortunately, commands which manipulate data are almost standard SQL commands. So if we can map all the data manipulation commands to standard SQL commands (discarding the nonstandard ones), and then apply them to the destination database, we can achieve a heterogeneous replication.

According to Kemian Dang [4], there are 3 important parts for a heterogeneous replication:

**Capturing:** This part finds the changes in the master table and extracts them out from the database management system.

As mentioned before, to realize incremental replication, changes are described in SQL language. Commands for changing table content include INSERT, DELETE and UPDATE, this kind of commands is called Data Manipulation Language (DML).

There are also commands which change the table structure, they are called Data Definition Language (DDL). These two parts are very different in definition, and also in the way of affecting replication.

DML can be used to make an efficient incremental replication, whereas DDL is more complex as it does not only concern data, but also changes the data type by changing the structure of how data storing. DDL sometimes includes vendor specific changes to the database, this kind of changes make replication very difficult to realize since they are hard to map to standard commands. Considering that DDL may greatly change the data object and potentially change vendor specific data structure, it is better to do a complete replication when DDL is met in the replication. However, a low frequency of using DDL limits its effect.

**Transforming:** This part is not the most difficult one, but it is usually very complex and needs a lot of work. Because a transformation is a mapping from one vendor’s database to another vendor’s. There are many things in database management systems, and every two databases need a checking: finding differences and making schema mappings of them. Even the commands from different database versions of the same vendor may be different when doing the same thing.

Another problem is that some of the commands are hard to map to any other vendors’ databases. In this situation, there needs to be a complete synchronization. But once a mapping is done, it will be a big advantage to the heterogeneous database replication.

**Applying:** This part is the easiest one. After mapping, all the SQL commands are

ready for use and compatible with the replica databases.

The procedure of capturing, transforming and applying are shown in Figure 2

.



Figure 2: Actions needed for a heterogeneous replication. (Damian Dang, 2008)

An actual implementation needs to extract SQL commands from database management systems. Some databases have already provided a way to extract history SQL commands. For example, Oracle has a tool called Log Miner, which is used to extract SQL languages.

However, most databases do not provide a direct way to extract SQL history. But because DBMS has to maintain atomicity, consistency, isolation, durability (ACID) itself, it has to record all the SQL commands that happened in the history. These records are usually stored in redo and undo logs, which are used in database management systems to recover from a crash. Because they contain all the SQL commands in the history, analyzing them will produce commands we need to carry out replication. Such a task is not easy, as not all the databases have an open guide on their log system, and every vendor has its own way of doing this. Another way of implementing heterogeneous replication is again re-using database vendor’s replication mechanisms.

Some large DBMSs have their own way of doing heterogeneous replications in a non-Grid environment. For example Oracle 11g’s stream replication method support heterogeneous replication; Sybase’s Replication Agent can control replication between MS SQL Server, Oracle and IBM DB2 to Sybase database; and IBM DB2 can use non-DB2 databases as replication source. Re-using such mechanisms with OGSADAI middle-ware makes it possible to perform heterogeneous replication through Grid environment.

The mapping replication method is a wide-compatible way, and it can do a 2-way replication between all the databases that already have a mapped schema; on the other hand, the re-using method is much easier and more efficient, but this method is more specific in that master databases are limited to some specific vendors’ database versions, which need to have the heterogeneous replication mechanism. If using mapping method in a cross domain environment, we only need to add a transformation action in the intermediate site. Though highly difficult, but if supported, the mapping method will surely be an easier way of doing heterogeneous Grid database replication

**3. Tools for Heterogeneous Replication**

There are several tools available in the market that can be used for heterogeneous replication. This section presents Oracle Golden Gate, Tungsten Replicator and MySQL replication for the purpose of this study.

**3.1 ORACLE GOLDENGATE**

Oracle Golden Gate allows change and manipulates data in transaction level in middle platform that is more than one and different. Oracle Golden Gate uses modular architecture that finally gives flexibility in extracting and replicating data notes that are chosen, change in transactional and changes in DLL (Data Definition Language) through various topologies, as can be seen on figure 3.


Figure 3: Supported Topologies by Oracle Golden Gate (Jeffries, 2012)

Oracle Golden Gate consists of these components: Extract, Data pump, Replicat, Trails or extract files, Checkpoints, Manager, and Collector, as can be seen on figure 4.



Figure 4: Components of Oracle Golden Gate (Jeffries, 2012)

Extract process runs in source system and is extraction mechanism from Oracle Golden Gate. Configuration of extract can be done by: (1) Initial load: for initial data loads, extracts a set of data directly from source object. (2) Changing synchronization: to keep synchronized data source with other sets of data, Extract catches changes done in data ( especially insert, update and delete transactional) after initial synchronization has been done, change in DDL and sorting is also extracted if it is according to type of used database. Extract process catches all changes that is done in configured objects to synchronize. Extract process saves all changes till stage has accepted commit records or rollbacks. When rollback is accepted, Extract throws away data for that transaction. When commit is received, Extract sends data for that transaction to trail process in target system. All log notes for a transaction is written on trail as a transaction unit arranged sequentially. This design holds responsible of data speed and integrity.

Data pump is additional extract configured in source system. If a data pump isn’t used, Extract process must send data into remote trail in target system. If data pump is configured, group of primary Extract will write in local trail which is available in target system. Data pump add flexibility of storage and also serve isolation process of primary Extract from TCP/IP activity. Replicat process runs in target system. Replicat reads change in data extracted and change in DDL (if there is change) specified in Replicat configuration and then replicates back to target database. Configuration of replicat is done by: (1) Initial load: for initial data loads, Replicat can apply data to target object or send it to high-speed bulk load utility. (2) Changing synchronization: to keep synchronization, Replicat applies change in extracted data in target object using native database interface or ODBC, depends on the types of database. DDL and orders replicated are also applied, if it is according to used database. Replicat applies changes that are replicated by the same order when those changes are committed in source database.

To support extraction and replication process continually from change in database, Oracle Golden Gate saves changes caught to disc in series file that is called as trail. A trail can be in source or even target system, or also in intermediate system, it depends on how the configuration of Oracle Golden Gate. In the local system, trail is known as extract trail or local trail. In remote system trail is known as remote trail. In using trail for storage, Oracle Golden Gate support accuracy of a data and error tolerance. Usage of trail also allows extraction and replication activity appearing freely with other trails. By this separated process, there is an opportunity to arrange how data is sent. For example, rather than extracting and replicating change continually, we can extract changes continually but save it in trail to replicate in target later, whenever target application needs it. Checkpoint saves position that is recently read and copied from a process to a disc for recovery. This checkpoint ensures that any changes in marked data for synchronization is extracted through Extract process and replicated by Replicat process, and also prevent repetitive or redundant process. Checkpoint provides error tolerance by preventing data loss, and requiring system, network, or a process of Oracle Golden Gate to start again. For complex configuration of synchronization, checkpoint allows Extract and Replicat process to be more than one to read set of same trail. Checkpoint works with inter-process acknowledgement to prevent data loss in network. Oracle Golden Gate has technology of incision of guaranteed message. Extract creates Checkpoint in its position in source data and in trail.
A checkpoint system is used by Extract and Replicat process operating continually, but it doesn’t need Extract and Replicat process running in batch mode. A batch process can be run back from its start point, where the process continually needs support to interrupt planned or unplanned that is provided by checkpoint.

Manager is a process controller from Oracle Golden Gate. Manager must run in both configured system by Oracle Golden Gate before Extract or Replicat can be started, and Manager must still run while those processes running till management function of resource is done. Manager does functions like these: (1) Monitor and restart process of Oracle Golden Gate. (2) Publishing beginning reports, like when throughput runs slowly or when hidden synchronization increases. (3) Keeping trail and log files (4) Allocating data storage room. (5) Reporting
mistake and incident. (6) Accepting and sending demand from user of interface
Collector is a process running in background in target system. Collector accepts change in extracted database sent through TCP/IP network, and writes it into trail file or extract. Specially, Manager starts Collector automatically when a network connection is needed. When Manager starts Collector, process is known as dynamic Collector and users of Oracle Golden Gate also can be run manually. This is known as static Collector. Not all configurations of Oracle Golden Gate use Collector process.
When a dynamic Collector is used, Collector can received information only from an Extract process, then it must be a dynamic Collector in each Extract process that is used. However, the comparison of one and one is optimal. Collector process is finished when combination of Extract process is finished. For default, the Extract process starts TCP/ IP connection from source system to Collector in target, but Oracle Golden Gate can be configured so Collector starts connection from target. Initiation of connection from target may be needed if target is in network area that can be trusted, but the source is in the area of network that can’t be trusted.

**3.2 TUNGSTEN REPLICATOR**

Tungsten Replicator is an open source high performance replication engine that works with a number of different source and target databases to provide high-performance and improved replication functionality over the native solution. With MySQL replication, for example, the enhanced functionality and information provided by Tungsten Replicator allows for global transaction IDs, advanced topology support such as multi-master, star, and fan-in, and enhanced latency identification.

In addition to providing enhanced functionality Tungsten Replicator is also capable of heterogeneous replication by enabling the replicated information to be transformed after it has been read from the data server to match the functionality or structure in the target server. This functionality allows for replication between MySQL, Oracle, and Vertica, among others.

Understanding the Tungsten Replicator works requires looking at the overall replicator structure. In the diagram below is the top-level overview of the structure of a replication service.


Figure 5: Tungsten Replicator Overview

At this level, there are three major components in the system that provide the core of the replication functionality:

**Extractor**

The extractor component reads data from a data server, such as MySQL or Oracle, and writes that information into the Transaction History Log (THL). The role of the extractor is to read the information from a suitable source of change information and write it into the THL in the native or defined format, either as SQL statements or row-based information.

For example, within MySQL, information is read directly from the binary log that MySQL produces for native replication; in Oracle, the Change Data Capture (CDC) information is used as the information source.

**Applier**

Appliers within Tungsten Replicator convert the THL information and apply it to a destination data server. The role of the applier is to read the THL information and apply that to the data server.

The applier works a number of different target databases, and is responsible for writing the information to the database. Because the transactional data in the THL is stored either as SQL statements or row-based information, the applier has the flexibility to reformat the information to match the target data server. Row-based data can be reconstructed to match different database formats, for example, converting row-based information into an Oracle-specific table row, or a MongoDB document.

**Transaction History Log (THL)**

The THL contains the information extracted from a data server. Information within the THL is divided up by transactions, either implied or explicit, based on the data extracted from the data server. The THL structure, format, and content provides a significant proportion of the functionality and operational flexibility within Tungsten Replicator.

As the THL data is stored additional information, such as the metadata and options in place when the statement or row data was extracted are recorded. Each transaction is also recorded with an incremental global transaction ID. This ID enables individual transactions within the THL to be identified, for example to retrieve their content, or to determine whether different appliers within a replication topology have written a specific transaction to a data server.

These components will be examined in more detail as different aspects of the system are described with respect to the different systems, features, and functionality that each system provides.

From this basic overview and structure of Tungsten Replicator, the replicator allows for a number of different topologies and solutions that replicate information between different services. Straightforward replication topologies, such as master/slave are easy to understand with the basic concepts described above. More complex topologies use the same core components. For example, multi-master topologies make use of the global transaction ID to prevent the same statement or row data being applied to a data server multiple times. Fan-in topologies allow the data from multiple data servers to be combined into one data server.

### **Extractor**

Extractors exist for reading information from the following sources:

MySQL

Oracle

### Appliers

The replicator commits transactions using block commit meaning it only commits on x transactions. This improves performance but when using a non-transactional engine it can cause the problems you have seen. By default this is set to 10 (The value is replicator.global.buffer.size in replicator.properties). It is possible to set this to 1 which will remove the problem with MyISAM tables but it will impact the performance of the replicators

Available appliers include:

MongoDB

MySQL

Oracle

Vertica

Tungsten Replicator operates by reading information from the source database (MySQL, Oracle) and transferring that information to the Tungsten History Log (THL).

Each transaction within the THL includes the SQL statement or the row-based data written to the database. The information also includes where possible transaction specific option and metadata, such as character set data, SQL modes and other information that may affect how the information is written when the data is applied. The combination of the metadata and the global transaction ID also enable more complex data replication scenarios to be supported, such as multi-master, without fear of duplicating statement or row data application because the source and global transaction ID can be compared.

In addition to all this information, the THL also includes a timestamp and a record of when the information was written into the database before the change was extracted. Using a combination of the global transaction ID and this timing information provides information on the latency and how up to date a dataserver is compared to the original datasource.

Depending on the underlying storage of the data, the information can be reformatted and applied to different data servers. When dealing with row-based data, this can be applied to a different type of data server, or completely reformatted and applied to non-table based services such as MongoDB.

THL information is stored for each replicator service, and can also be exchanged over the network between different replicator instances. This enables transaction data to be exchanged between different hosts within the same network or across wide-area-networks.

Tungsten Replicator 2.1 offers a set of features that surpass any open source replication solution available today: parallel, multi-master and heterogeneous replication. Tungsten Replicator benefits include:

1. Provides low-impact, real-time replication with up-to 5X throughput over native MySQL and over 100X reduction in slave lag.
2. Enables efficient multi-master operation across sites using a variety of replication topologies including bi-directional, all-masters, stars, and snowflakes.
3. Replicates heterogeneously between MySQL and Oracle as well as from Oracle to Oracle itself.
4. Loads data in real-time from Oracle and MySQL to high-performance data warehouses like Vertica and InfiniDB.
5. Publishes data in real-time from Oracle and MySQL to NoSQL implementations, such as MongoDB.
6. Permits flexible filtering and inspection of transactions during replication.
7. Cloud-ready with easy installation and management that works in both cloud as well as on-premise environments.

**3.3 MYSQL REPLICATION**

Replication enables data from one MySQL database server (the master) to be replicated to one or more MySQL database servers (the slaves). Replication is asynchronous by default, therefore slaves do not need to be connected permanently to receive updates from the master. This means that updates can occur over long-distance connections and even over temporary or intermittent connections such as a dial-up service. Depending on the configuration, you can replicate all databases, selected databases, or even selected tables within a database.

Advantages of replication in MySQL include:

1. Scale-out solutions - spreading the load among multiple slaves to improve performance. In this environment, all writes and updates must take place on the master server. Reads, however, may take place on one or more slaves. This model can improve the performance of writes (since the master is dedicated to updates), while dramatically increasing read speed across an increasing number of slaves.
2. Data security - because data is replicated to the slave, and the slave can pause the replication process, it is possible to run backup services on the slave without corrupting the corresponding master data.
3. Analytics - live data can be created on the master, while the analysis of the information can take place on the slave without affecting the performance of the master.
4. Long-distance data distribution - if a branch office would like to work with a copy of your main data, you can use replication to create a local copy of the data for their use without requiring permanent access to the master.

Replication in MySQL features support for one-way, asynchronous replication, in which one server acts as the master, while one or more other servers act as slaves. In MySQL 5.5, an interface to semi-synchronous replication is supported in addition to the built-in asynchronous replication. With semi-synchronous replication, a commit performed on the master side blocks before returning to the session that performed the transaction until at least one slave acknowledges that it has received and logged the events for the transaction.

There are two core types of replication format, Statement Based Replication (SBR), which replicates entire SQL statements, and Row Based Replication (RBR), which replicates only the changed rows. You may also use a third variety, Mixed Based Replication (MBR).

Replication is controlled through a number of different options and variables. These control the core operation of the replication, timeouts, and the databases and filters that can be applied on databases and tables.

You can use replication to solve a number of different problems, including problems with performance, supporting the backup of different databases, and as part of a larger solution to alleviate system failures.

**4. Comparing Tungsten Replicator and Oracle Golden Gate**

Among the different replication software available in the market, Tungsten Replicator from Continuent and Oracle Golden gate from Oracle are the most popular. They are both used in the market more than any other software for replication in a large scale environment.

Oracle GoldenGate is a great solution, and has been around almost 20 years, it is robust and mature. Oracle GoldenGate covers multitude of replication needs:

1. Disaster Recovery and Data Protection,
2. Zero Downtime Upgrades and Migration,
3. Operational Reporting,
4. Operational Real-Time Business Intelligence and
5. EDA/SOA.

The expectation is set that everything works like magic and Golden Gate will satisfy all of your enterprise data replication needs. It does that, with very high cost and complexity. Continuent Tungsten Replicator offers you a real alternative. It is much more light-weight, simpler to operate, less expensive, and yet very flexible and powerful. Tungsten Replicator, it’s an open source [database](https://www.pythian.com/databases/) replication engine that can be used to complement or completely replace native [MySQL](https://www.pythian.com/mysql/) Replication.  In addition to providing standard replication functionality, Tungsten Replicator introduces exciting new features such as global transaction IDs, heterogeneous replication from MySQL to [Oracle](https://www.pythian.com/oracle-dba-services/) and Postgres, parallel replication, and the ability to replicate from multiple masters to a single slave.

**4.1 ADVANTAGES AND DISADVANTAGES OF TUNGSTEN REPLICATOR**

Table 1. Advantages and Disadvantages of Tungsten Replicator

|  |  |
| --- | --- |
| **ADVANTAGES** | **DISADVANTAGES** |
| Open Source Free of Charge | External to MySQL Process |
| One-to-Many/Many-to-One Topologies | Free version is limited in what it can do. |
| Parallel thread by Schema |  |
| Has complex filtering |  |
| Support Heterogeneous Replication: Can Extract from MySQL and Oracle and apply to Oracle, MySQL, Redshift, Vertical, Hadoop, Mango DB, Kafka etc. |  |
| Has simple role switching |  |
| Data loss is eliminated |  |

**4.2 ADVANTAGES AND DISADVANTAGES OF ORACLE GOLDEN GATE**

 Table 2. Advantages and Disadvantages of Oracle Golden Gate

|  |  |
| --- | --- |
| **ADVANTAGES** | **DISADVANTAGES** |
| Golden Gate is able to quickly replicate transactions in a method that maintains transactional integrity and give you a like to like replica on the other side. | Setting Up Oracle GoldenGate. I've found that specific Oracle GoldenGate and Oracle RAC experience is needed in order to make this happen. That level of experience does not exist everywhere. |
| Transformation. Oracle GoldenGate is able to do some data transformation as it replicates, this is very key if you need your data to appear a certain way, especially important in Reporting. | Monitoring Oracle GoldenGate. Currently there are two main options for monitoring Oracle GoldenGate, the first, GoldenGate Monitor, does not scale well as it becomes very non-performant when you have more than 10-15 GoldenGate Managers reporting to it, the second, Oracle Enterprise Manager, while getting better still requires a lot of manual configuration to have Oracle GoldenGate talk to the OEM agent. While this has got a lot better with 12c GoldenGate, I still feel that there is room to improve here. |
| Recoverability. Oracle GoldenGate has many fail safes to ensure data is not lost, some examples of this are trail files its own system of recording changes that works with the DB SCN. | Unable to replicate certain Data Types. Again, less of an issue in 12c Oracle GoldenGate, but there are still some Data Types that Oracle GoldenGate is unable to replicate. |
| Multiple Data Sources/Targets. Oracle GoldenGate is not limited to just Oracle Databases as a target it can support most of the popular databases in the industry today such as SQL Server, [MySQL](https://www.trustradius.com/products/mysql/reviews), [DB2](https://www.trustradius.com/products/db2/reviews) and more. | High cost of ownership |
| Data Recovery synchronization is very fast |  |

**4.3 COMPARISON BETWEEN THE THREE REPLICATOR**

Table 3. Comparison between MySQL, Tungsten Replicator and Oracle Golden Gate

|  |  |  |
| --- | --- | --- |
| **MySQL** | **Tungsten Replicator** | **Oracle Golden Gate** |
| Replication is very fast since it is embedded in the MySQL Engine | Replication also very fast, but not compared to MySQL | Replication is bit fast but not as compared to Tungsten and MySQL |
| Totally free with any support you want. | It is free and most of the target database can be gotten free as well. | It's expensive. And I believe you have to purchase a license for each type of source and/or target DB that you're trying to replicate. |
| Can Replicate between MySQL to MySQL | Can Replicate between much more different database system | Can Replicate between different database system |
| Very high support for MySQL | Support for Tungsten replication has been outstanding. | Support for Oracle Golden Gate is available and very impressive. |
| MySQL is not good at providing fail safe. If there are any issues, you might lose your data. | Tungsten is very good in ensuring that data is available all the time, avoiding single point of failure. And that is what it does perfectly. | Oracle Golden Gate has some fail safe that ensures that data is not lost |
| Data migration is not too good with MySQL. | You can seamlessly migrate to Hosted datacentre with noDowntime. It also nearly eliminate downtime caused by A DB problem. | The database upgrade and migration was possible with zero downtime because of having GoldenGate in place. |
| MySQL topologies are only Master-Slave and Master-Master if configured | Tungsten has a lot of topologies to offer. Fan-In, Fan-Out, Fan-In and Out, Direct, All-Master, Hub-Spoke and Heterogeneous | Oracle has Unidirection, Bi-direction, peer-to peer, broadcast, consolidation and cascading topologies |

**5. SUMMARY AND CONCLUSION**

This paper presents a comparative study of different heterogeneous replication tools that can be used to move data across different database systems.

The findings from the study show that tungsten replicators provides a very flexible and cost effective method, whereas the oracle golden gate would be the choice if money is not an hindrance. The MySQL native replication is also good, but lack the dexterity of tungsten replicator and oracle golden gate. The using of replication mechanism of the different database system does not bring about high efficiency as it was envisage. The study helps in making sharing and accessing of vital information easy, even if stored in various database systems. This is very good for business all over the world where several organisations need to merge data for one reason or the other.

Future work will need to be carried out to ascertain the best heterogeneous replication tools by widening the scope of these tools as there are others more capable of doing the same work.

**REFERENCE**

1. Chen Y, Berry D., and Dantressangle P. (2007) “Transaction-Based Grid Database Replication.” National e-Science Centre, Edinburgh, UK.

2. Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., And Tuecke, S. (2000). The Data Grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *J. Net. Comput. Appl. 23*, 3, 187–200.

3. Chervenak A., Schuler R., Kesselman C., Koranda S., & Moe B. (2005) “Wide Area Data Replication for Scientific Collaborations.” 6th IEEE/ACM International Workshop on Grid Computing, 2005.

4. Kemian, Dang (2008) “Towards Heterogeneous Grid Database Replication" Master of Science Computer Science School of Informatics University of Edinburgh.

5. Kunszt P., Laure, E. Stockinger, H. & Stockinger, K. (2003). “Data Grid Replica Management with Reptor.” 5th international conference on parallel processing and applied mathematics.

6. Liming L., (2005). Large-Scale Data Replication for LIGO, ClusterWorld, July 2005.

7. Maurizio Lenzerini (2002). *"Data Integration: A Theoretical Perspective"* (PDF). PODS2002. pp. 233–246.

8. May Mar Oo, (2010). “Fault Tolerance by Replication of Distributed Database in P2P System using Agent Approach”, International Journal of Computers, Issue 1. vol.4, 2010.

9. Mell, P., & Grance, T. (2011). The NIST definition of cloud computing.

10. Morales T, *Oracle9i Database Migration, Release 2 (9.2) Tony Morales 2002*, United States of America: Oracle Corporation, 2002.

11. Munoz J. & Syracuse N, *Replication and Migration*, 2000, retrieved from:
 https://www.ietf.org/ proceedings/49/slides/NFSV4-4.pdf

12. Murthy, P.K., S. Agarwal, T. Garbyal and P. Chakrobarty, 2012. Secure migration of tables across different database management systems over a cross-platform environment. Int. J. Inform. Electron. Eng., 2: 387-390. DOI: 10.7763/IJIEE.2012.V2.121

13 Nicholson C., *Dynamic Data Replication in LCG 2008*, UK e-Science All HandsConference, Nottingham, September 2006.

14. Pratt M., (2001) *Oracle9i Replication,* Oracle White Paper, 2001.

15. “PostgreSQL 8.4 Server Administration”, Volume II- The PostgreSQL Global Development Group, 1996.

16. Silberschatz, A., H.F. Korth and S. Sudarshan, 2011. Database System Concepts. 6th Edn., McGraw-Hill, New York, ISBN-10: 0071289593, pp: 1349.

17. Saravanan, R. and N. Ramaraj, 2009. Providing reliability in replicated middleware applications. J. Comput. Sci., 5: 11-22. DOI: 10.3844/jcssp.2009.11.22

18. Sujoy P.Paul, “Pro SQL Server 2008 Replication”

19. Jeffries, J.P., (2012). Oracle GoldenGate 11g Implementer's Guide. 1st Edition.Packet

 Publishing Ltd., Birmingham, ISBN-10: 1849682011, pp: 280

.