

VIII Konferencja PLOUG
Kościelisko
Październik 2002

Increase productivity and safety data of warehouse systems using Shareplex for Oracle

referat sponsorski

Quest Software

1. High speed replication for Oracle

by Eyal Aronoff, Chief Technology Officer of Database Products at Quest Software

This chapter discusses three critical issues when considering replication techniques for Oracle: (1) Impact on the performance of the primary (possibly OLTP) application and (2) Time criticality and accuracy of the data on the secondary system.

The increase in complexity and management overhead with growth in both the size and the number of the objects to replicate as well as the number of destinations In 1996 for the first time there were more Oracle-based applications in production than in development. Once an application is deployed to production, the organization becomes dependent on it. That causes Oracle-based production applications to become more “mission critical” than ever before. Production applications have the unwieldy profile of rapid data growth – some databases experienced doubling in size over the last year. This combination of rapid growth and “mission criticality” creates the need to find a high availability solution over and above the standard backup and recovery scheme. On the other hand, Data Warehouses, Executive Information Systems (EIS) and other data consolidation projects require data integration from multiple sources. Similarly data produced by one part of the organization may be used throughout. **High availability, data consolidation projects and data distribution projects all have one thing in common. They all require data redundancy through replication.** Oracle replication features address some of data replication challenges. However many users of Oracle replication are expressing concerns with issues of performance and manageability. This paper explains the main issues with data replication, how Oracle addresses them, and proposes an alternate method of data replication.

1.1. The Problem

As the organization becomes more dependent on data and applications, the importance of data distribution increases. The two most common purposes for data distribution are data sharing and off-site backup/standby. One of the most important aspects of data distribution is data replication. With the speed of database growth and the event of hundreds of users hitting the database, the workload on existing hardware is already close to peak capacity. To combat this effect, the investment in increased capacity often leads to shops with large machines that perform hundreds of database modifications per second. In environments like this, the intrinsic overhead of replication pushes the already overworked machines over the top, increasing response times to unacceptable levels. The speed of replication is generally not sufficient to support the volume of changes required. For these reasons organizations could not fully deploy the required replication scheme.

1.2. The Solution

At Quest Software, we are addressing these problems by focusing on specific issues encountered by users of Oracle replication. These are five of the areas we are addressing:

- Minimize the overhead on the source machine.
- Minimize the impact to the processes that generate database modification.
- Minimize the exposure to network failure
- Shorten the time delay between modifications on the source and their application on the destination machine (especially important for standby machines).
- Enforce read consistency across both source and destinations.

- Easily scale to a large number of replicas, some of which may not be accessible at all times.
- The solution: we are using a replication scheme based on the data stored in the Oracle log files.

The Oracle replication scheme contains many configuration options. For simplicity, I use in my examples only the two most frequently used configurations: incremental snapshots and updateable masters. Many of the points, though, apply for most configurations.

1.3. Performance Is Everything

Let's look at an example. A job that manipulated data in three tables took five hours to run. Once one of the tables was turned into a replicated master, that same job took 12 hours to complete - blowing through the nightly batch window. Although results vary significantly between different production examples, there is always significant overhead to transactions on the master table. The reason is that trigger-based replication causes insert, update and delete operations (DML) to be added to your own transactions. These operations are executed as part of the user (or batch job) session. So where initially the job had only 100,000 operations to complete, with trigger based replication, it might have the equivalent of 200,000 operations. Thus, transactions that manipulate replicated tables will take longer (and some times MUCH longer) to complete.

One of our goals is to create a replication scheme that has minimal impact on the source system. Log replication is done independent of the Oracle transaction. Hence, it does not make a difference from the user perspective whether a table is replicated or not. The response time of both online and batch transactions is minimally impacted by this type of replication. In addition to the direct impact on the transactions that are involved with trigger-based replicated tables, there is a constant overhead on systems that host a replicated database. As one can imagine, the overhead of managing the replication process by means of database triggers and insert/update/delete transactions can be very significant. Oracle uses background processes to determine what, when and where to replicate. These background processes scan what could become large tables. If rows are found, they are sent via SQL*Net to the remote sites. Once this is done, the rows are deleted from the local site. With database snapshots, the overhead is relatively low since only the ROWIDs of the modified rows are inserted and deleted. In symmetrical replication, the entire before and after image of the complete record is manipulated, which many times ends up causing chaining that further increases the overhead. Log replication does not manipulate the database for every change. The modification queues are stored in flat files that are a hundred times faster to scan than database tables. If the workload is moderate, the data flows in and out of the queue without being flushed to disk. You may be concerned that your site generates hundreds of megabytes of redo log data every day - "Surely log replication will cause a huge amount of network traffic." What we have found is that only a fraction of the log is user data. The log is written in blocked mode (so if the block is 2K and there were only 124 bytes to write before a commit, there will be 1900 bytes of wasted space). Additionally, Oracle writes index modifications and other "accounting" information to the log such as changes to the high-water-mark, adding and removing blocks from the free list, and more. These modifications do not require replication. Another advantage of log replication that increases its speed is the ability to use array operations on the remote database. Rather than performing the operation a row at a time, a whole batch of rows may be applied simultaneously.

1.4. Timing Is Everything

Many organizations require fast replication, whether for a standby machine or just because their business needs make them concerned about a lag between the time a transaction is completed on the source system and the time it is reflected on the target system. This is specifically important for

long running transactions because they generate a lot of data and they are the most difficult to recover. With trigger-based replication, the Oracle background process cannot “see” the data in the snapshot log or the replication queue until the transaction commits. Consider the previous example of that job that took 12 hours. Suppose that job had only one commit in the end. Thus, after 12 hours of work, the Oracle replica would be 12 hours behind. It could take another 5 minutes for the snapshot process to realize that there is data to replicate. The replication process itself and the network traffic might take another 8 hours (if there would be a network problems the entire transfer might have to restart).

At that point, the replica would be 20 hours behind. Here, we would have a standby machine that was almost one day behind! With log replication, transactions are replicated as soon as they hit the log (which is instantly). As long as the replica can keep up with the source, it will not be behind. Once a commit arrives in the log, there is little or no data to send, and the commit just flows through. Additionally, the unit of retry on a network failure is an individual change (or a small group of changes). So a network failure causes the resending of few records at the most, unlike the trigger-based replication that has one commit at the end (which means all or nothing). In our test that same job took only 5.5 hours to run and the replica reflected the changes 2 minutes after completion on the source system. Another aspect of time delay between the source and destination is the dependency of the replication process on the availability of the network. In Oracle the propagation process makes a direct SQL*Net connection to the remote database. The entire queue of changes is then applied as one transaction to the replica site. At the end of this transaction there is a two-phase commit that ensures all data was replicated to the remote site before deleting it from the queue tables in the source site. If the number of changes to replica is large, this transaction can take a very long time. During this entire time, the network connection has to be available. If at any time during the propagation transaction the network becomes unavailable, the entire propagation is rolled back and Oracle will have to start all over again. In Oracle8, parallel propagation overcomes some of the aspects of this problem by providing multiple simultaneous connections to the remote database. If the transaction is limited by the speed of the network traffic, multiple connections will not get the data any faster to the remote site than a single connection. Even if the network traffic is not the bottleneck, multiple connections only limit the window of exposure but do not solve the root of the problem. For example: assuming that in Oracle7 the propagation of changes took four hours to run. In Oracle8, with four processes doing the propagation it could take an hour to run. However, if after 40 minutes the network connection is lost, the four propagation transactions will roll back, and the entire data will have to be reapplied to the remote database.

In some situations the reliance on a single transaction to propagate the data to the remote site can bring replication to a halt. For example, let’s assume you have a replication schema that replicates an application from the US to Hong-Kong. Suppose that the network connection to Hong-Kong is not very reliable. You may get one network failure every hour of continuous connection. If the application generates about four hours of network transmission throughout the day, with normal operation the site in Hong-Kong will be very close behind.

The network reliability will not be an issue because the four hours network workload will be spread on a 24 hours span. However, let’s assume that one weekend the Oracle instance in Hong-Kong was shutdown, and stayed shutdown for two days before being restarted on Monday. During this time, the instance in the US would keep collecting changes for the Hong-Kong site. On Monday the connection finally got reestablished. The US instance had accumulated enough changes for eight hours of transmission. However, Oracle might never get a straight eight hours span to send the data out without an error that would cause the transmission to restart.

As Oracle retries to send the data more data is accumulated at the source. The result is that the Hong-Kong site might never catch up to the US source. With our replication the unit of retransmission over the network is a few records at most. This means that if the network is not reliable, the

amount of retransmission will be very small. Additionally, the log based replication queues changes on both the source site and the destination. This means that as long as Hong-Kong server is “up” (although the database may be down), it can keep receiving changes and queue them locally. When the database is finally started, the transactions get applied from the local queue with no additional network traffic.

1.5. Size Is Everything

As the number of components in a replication scheme increases, so does their interdependency. To resolve this problem, Oracle introduced a new term - “quiesce”.

This means that while changes are made to the replication scheme, the entire network of replicated machines is locked (or “quiesced”) for the duration of the change.

However if you have a large number of computers and large volumes of transactions, it could become impossible to get to a quiesced situation. This is especially true if some of the computers that participate in the replication scheme are not connected to the network at all times. To resolve this challenge, we decided to include the configuration changes as part of the data in the data stream. This eliminates the need for a synchronized transaction. Each target machine makes the modifications to the replication scheme when it gets to that ‘point in time’. From that time on only data that represent the new scheme will be present in the data stream. Another challenge we overcame is the impact that reorganization of tables on the source system has on the target systems. In snapshots, for example, an import of data will cause a full refresh.

Since we use the content of the columns (primary key, unique key or the entire record) as our update key rather than the original ROWID, we do not require a full refresh for data reorganization.

Oracle replication uses SQL*Net and a direct TCP/IP connection between the source and each target machine participating in the replication. This is known as “two tier architecture”. When the number of target machines increases, so is the overhead of keeping all these TCP/IP connections. Each new machine means duplicating the data to that machine directly from the source. One hundred machines means sending the same data a 100 times through the network. This strains the network link from the source to the network service provider. Added to that the cost of “accounting” of what machine

got what portion of the data that, as shown, involves quite a number of database operations. The result is that the high overhead of adding more target machines limits the size of the replication scheme. Our replication paradigm supports the concept of a “multi-tier architecture”. You can define that the data will go through one or more intermediate machines. The data is only duplicated for each directly connected machine. This means that if you have a 100 target machines, the source could be configured to send the data to only 10 machines, and from there the data could be propagated to the remaining 90 machines. The workload on the source and each routing machine would be greatly reduced.

As the replication scheme grows, so increases the need for better control. The current Oracle replication scheme does not provide centralized information on how the entire network of replication is performing. Simple questions such as how behind the replicas are can be very difficult to answer. With our replication we provide constant feedback on the status of the replication to one or any number of monitoring stations. The administrator uses our GUI based tool to see the performance and status of the entire network of targets from one station.

1.6. It Is the Technique

There are many ways to replicate data throughout the organization. Each of these ways has advantages and disadvantages depending on the business requirement it fulfills. We believe that log replication is one of the fastest, most flexible, and scaleable way to achieve replication for a variety of applications. Additionally the generic SQL implementation of the product enables it to replicate across databases from multiple vendors, translate table names, column names, and even different formats.

Still to come are automatic propagation of data dictionary changes, automatic switch over to a (or several) standby machines, automatic incremental re-synchronization of out-of-sync data, and more.

2. Replication Methods in a Data Warehousing Environment

by Bill Brunt, Product Marketing Manager, Quest Software

2.1. Overview

Data warehousing is a rapidly growing segment of information management. In the beginning, data warehouse projects were undertaken only by large corporations or those businesses that could not run without business analytics. Today, with lower cost hardware and software to support data warehousing, end users from medium sized companies to multi-national corporations are using business analytics to improve decision-making.

However, IT professionals who have created data warehouses and data marts have become a victim of their own success with rapidly increasing expectations from their customers. Some of the most common expectations and their needs are listed below:

Currency of data

Customers are expecting the most accurate and available data, which is leading to daily or even more frequent loads. The idea being that the load should occur in an ever-decreasing window, which puts a drain on system performance.

An enterprise view of data

Many corporations will run different packaged applications while customers want to report across the entire enterprise. This becomes a driver for data consolidation.

2.2. Fast response time to queries

The more successful a data mart or data warehouse becomes, the more people will provide increasingly demanding questions. Additionally, when customers of a data mart/data warehouse work for a large company with several locations, the network also becomes a limiting factor in response time.

SharePlex can help optimize a data warehouse environment to meet each of the above challenges. Quest Software's SharePlex®, provides asynchronous, fault tolerant, streaming and near real time replication of the Oracle database.

2.3. Currency of Data

SharePlex can increase the currency of data by taking time out of the load process.

Information in a data warehouse usually travels from the transaction system, to a staging area, extracted from staging, transformed and then loaded into the data warehouse. In order to provide a consistent view of the data, referential integrity must be maintained as data moves from the transaction system to the staging area.

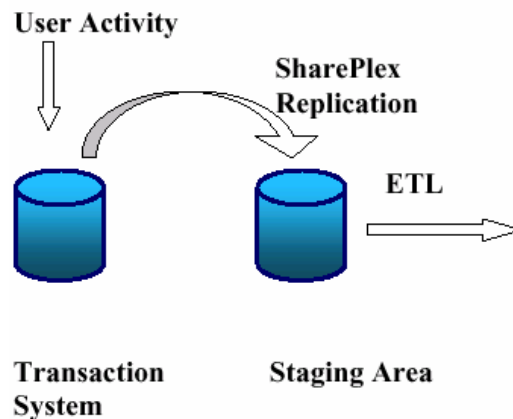
An example will help illustrate:

Assume three tables are being moved to the staging area and these tables are: CUSTOMER_MASTER, SALES_TERRITORIES and ORDERS.

While the load to staging has completed reading the CUSTOMER_MASTER and is in the midst of reading the SALES_TERRITORIES, a customer service representative enters a new customer and then an order for this customer. Once SALES_TERRITORIES has completed being read, ORDERS will be read and contain an order for a customer, which doesn't exist in the extracted CUSTOMER_MASTER file. To avoid this issue, data warehouse administrators take a consistent image and move this to staging.

SharePlex can provide this capability in a nearly instantaneous fashion.

There are many other ways to achieve a consistent image in staging but only SharePlex will do it as quickly and with so little overhead and administration. SharePlex also has the granularity to move only the tables needed and not the whole database thereby saving a considerable amount of disk.

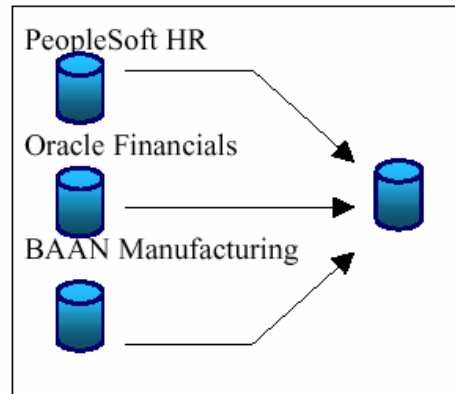


Above illustrates the data warehouse process from User Activity through to the staging area

In a matter of seconds, the SharePlex posting process on the staging area can be suspended. The ETL process can then immediately start without any issue of referential activity arising in the final destination.

2.4. An Enterprise View of Data

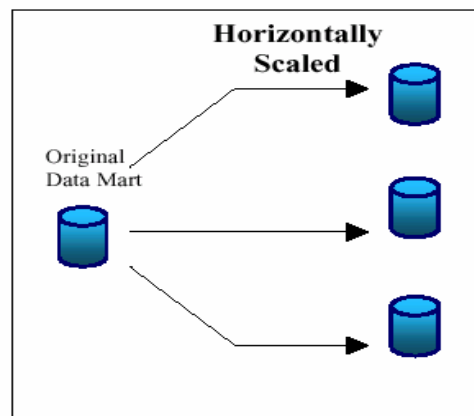
Seldom is the source data available in a single transaction system to feed the staging area. Often the staging area will act as a consolidation point from several systems. SharePlex can accommodate this requirement as well. Since SharePlex is not limited to being a target from only one source, multiple transaction systems can consolidate to a single staging area as shown below.



The illustration above shows how the staging area can become the consolidation area for many points in the enterprise

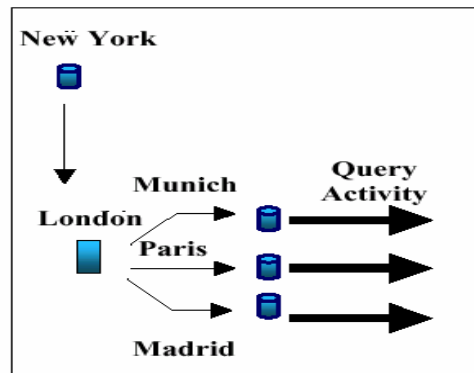
2.5. Fast Response Time to Queries

When a data mart has achieved a certain degree of success, response time can begin to suffer as many users submit one or more queries. Since the data mart is a snapshot of information in time, the data mart can be scaled horizontally by adding additional machines whose databases are an exact copy. SharePlex can accomplish while the load completes via a broadcasting of data as shown below.



The illustration shows how the user load can be horizontally distributed throughout the data mart. SharePlex can be used to broadcast data to all of these points in the data mart

When the volume of data being extracted from a database through queries, which bring down large datasets in a wide area network, exceeds the amount of change being applied to the database, SharePlex can improve response time and reduce network traffic. This can happen with many popular front query tools such as the full client version of Business Objects, SAS or SPSS. Imagine a case where an organization has hundreds or even thousands of analytic users across the globe. For instance, if the database is filled in New York via an ETL process, the remote database then broadcasts data by replicating only the changes to the database. Therefore less network traffic will be used.



In this case, the data warehouse is loaded in New York and then replicated to a machine without an Oracle database in London. If the machine that will cascade data will not need to access the data through Oracle, then a database is not required. SharePlex will simply store and forward the information. The resource requirements for this system will be a fraction of those where databases do reside. The cascade machine then broadcasts this information. While three databases are available in Europe for query, the transatlantic network requirements are for only one third of the data being populated.

2.6. Transformation

For companies looking for a streaming, near real time method of loading the data warehouse, SharePlex is the answer. SharePlex works by performing change data capture on the source database. A dedicated process called capture reads the information from the online redo logs and turns these into change messages, which are then transported to the target database. On the target database, a process called the poster takes these change messages and converts them into SQL to be applied to the target database.

In the discussion so far, SharePlex has simply been replicating data from the source to the target in the same form. The posting process is very flexible in that PL/SQL routines can be designed to be fired when a message propagates to the target. Despite considerable inroads today by ETL vendors with advanced GUIs, PL/SQL remains a mainstay of transformation. Its ultimate flexibility, relative ease to get skilled developers of and pervasiveness allow PL/SQL to remain a viable transformation technology.

Lets examine a data mart, which reports on sales from point of sale (POS) terminals in the store. The POS transaction system populates a sales order and detail lines containing information about the date, time, location, customer, total amount of sales and the individual items sold. This data resides in a second, third, possibly third normal form on line transaction processing data model. SharePlex will pick up this information from the source and transport it to the target. Assuming there were five items sold, this would involve six change data capture records. The sales order header and the five detailed lines. Lets assume this is the first sale of the day and it is for a new customer.

The data model in the data mart is a simple star schema with aggregate summary tables for product group, region and date. The fact table is sales detail while the supporting dimensions are: date/time, product, location and customer. The PL/SQL routines will insert a new record into the fact table. Look up in each of the dimensions if the keys to the dimension exist and if so, move on to aggregate processing. Since the keys to the dimension table don't exist for this customer and this date, new records will be inserted into the dimension tables. The totals for product group, region and date will then be added to the supporting aggregates. Of course, the aggregate record will be checked to see if it exists and if not, an insert will be performed. Finally, the entire transaction is

committed. With SharePlex for Oracle, analysis and data exploration can be conducted in the near real time. In this example, it allows a retailer to judge the effectiveness of different store displays designed to promote select products.

Companies around the globe are recognizing the effectiveness of decision support technologies to provide competitive advantage. SharePlex for Oracle takes this to a new level by providing near real time analytics.

3. Conclusions

SharePlex can be an invaluable tool in data warehouse to collect, distribute data, improve performance and reduce network traffic. These are but a sample of how SharePlex can be used in a data warehouse environment. For example, if loads are constrained in a window but can split to run as parallel jobs, SharePlex can consolidate the loads as they occur to a query data mart.