Active-Active Replication

Considerations for high availability

ABSTRACT

Many organizations are now using active-active replication to ensure high availability. In this white paper, we'll reveal how you can easily and successfully implement this strategy across your enterprise.

INTRODUCTION

As data becomes more and more critical, its availability should not be compromised by any type of outage – whether it's unscheduled due to a system crash or malfunction, or it's scheduled due to patches or upgrades to Oracle, the OS, or applications, and storage replacement. Because today, people view scheduled outages differently than 10 years ago. They don't really care if the outage is scheduled or unscheduled; an outage is an outage. With this in mind, all types of organizations are looking for more uptime: many are striving for five 9s, or only about 6 minutes of unscheduled downtime a year. This is, of course, very difficult to achieve.

Replication technology like SharePlex® for Oracle, a lower-cost alternative to Oracle GoldenGate, can help.

With replication, organizations can extend the database and server to minimize outages. One replication method, active-active replication (also known as peer-to-peer, master-to-master, active-active, or multi-active server replication), offers the most promise. Active-active replication is a horizontal scaling of the application over multiple servers that allows propagation of changes to more than one server. If everything is done properly, end users will not see any outages from the application.

This white paper discusses the key considerations to keep in mind when using active-active replication to ensure high availability.

CONSIDERATIONS FOR HIGH AVAILABILITY IN ACTIVE-ACTIVE REPLICATION

To ensure your application can be set up for horizontal scale with active-active replication, consider the following:

- Unique key collisions
- Triggers
- Update jobs
- Application deployment (DDL)
- Network connectivity
- Conflict resolution
- Conflict avoidance

Understanding seven key considerations will help you prevent outages and achieve your replication goals.

Unique key collisions

As you extend your application horizontally, you need to consider how to handle unique key collisions. Unique keys are usually generated by an Oracle sequence. Since the application is now split over multiple servers, a single sequence with the same value cannot be used, since each one could potentially generate the same key value.

Methods for creating unique keys

Here are two methods for creating different unique keys:

- **Use a range for each server system.** For example, you could assign the following ranges:
  Server 1 – Range from 1-999,999,999
  Server 2 – Range from 1,000,000,000-1,999,999,999
  Server 3 – Range from 2,000,000,000-2,999,999,999 Etc.

- **Use a distinct set of values for each server system.** For example, if you have two servers, you might have one use odd numbers and the other use even numbers, as follows:
  Server 1 – Use a sequence that generates odd numbers:
  CREATE SEQUENCE supplier_seq MINVALUE 1 START WITH 1 INCREMENT BY 2 CACHE 20;

  Server 2 – Use a sequence that generates even numbers:
  CREATE SEQUENCE supplier_seq MINVALUE 1 START WITH 2 INCREMENT BY 2 CACHE 20;

If you have more than two nodes, follow a similar convention. However, be sure to:

- Assign a node number for each server in your active-active setup.
- Consider the max number of nodes you will have in the active-active setup.
- When creating the sequence, use the following template:
  CREATE SEQUENCE sequence_name MINVALUE 1 START WITH node_number INCREMENT BY n CACHE 20;
  Where:
  node_number is the assigned node number in the active-active setup.
  n is the number of maximum nodes in the active-active setup.

Recovering the database and resetting the sequence

As you select a method to handle uniqueness collision, keep in mind how to recover the database and reset the sequence to its rightful values.

- With method 1, to recreate the sequence, scan the unique key for largest value in the range. For example:
  "max (col) where col > min_range and col < max_range"

- With method 2, find the largest value and depending on the node number, you will need to set the next available value. For example:
  "(Trunc ((max (col) + n) / n)) + node_number"
  Where:
  n is the number of nodes in replication.
  node_number is the assigned node number.
  The same formula can be used on any node.

Triggers

Row-level triggers

A row-level trigger that modifies data in replication should be prepped with a "WHEN" clause. When using SharePlex for Oracle, this is needed because when SharePlex applies the data on the target, it will not trigger the change. For any existing trigger, there is a script that will prep all triggers with the WHEN clause. For any new trigger, application deployment must ensure that the "WHEN" clause exists. SharePlex for Oracle provides a script to modify all triggers to add a "WHEN" clause to the trigger. The script is provided with the software called sp_add_trigger.sql.

Statement-level triggers

There is no good method to isolate the change from a particular user. Statement-level triggers that change replicated data must be avoided.

Update jobs

Consider where to run update jobs, since running jobs on multiple servers can cause collision. Make sure your failover process handles enabling and disabling of jobs; otherwise, conflicts may occur.

Application deployment (DDL)

In a downtime scenario, application and DDL changes are made at the same time on one system.

- A column is added, and then the application is updated to use the column.
- When a column is removed, first stop updates to the database, drop the column, and then update the application.

In a no-downtime, active-active scenario, you can use SharePlex to propagate the changes. SharePlex lets you choose either to replicate DDLs or to control the DDL yourself. Full control can be obtained without downtime; however, the deployment of changes has to be split into two phases and the steps depend on whether the transaction is an add or delete:

- When using a DDL to add a column or table, do not add data (the application should not utilize the new column). This is to avoid replication of the new column data to the target server(s) because the column does not yet exist on the target server. The second phase is when all servers are updated with the new column, allowing the application to use the new column.
- When using a DDL to drop a column or table, the first phase is to modify the application to not use the column and then update all systems to use the same application version. The second phase is to drop the column on each server.

Network connectivity

Application routing should be in place for both normal and outage situations. Connections from customer requests should have some routing factor, such as geographic or IP persistence. A secondary mechanism should also be in place in case of outages. When you use multiple servers, the application needs to know that it must connect to a different box when there is an outage on one server.

SharePlex will utilize the network connections to send data between the servers. Network bandwidth considerations for replication should be determined. In addition, network isolation and routing for SharePlex activities must be considered as well. If network bandwidth is not taken into consideration, a backlog could occur.

Application routing should be in place for both normal and outage situations.

Conflict resolution

In data replication, conflicts can occur on three types of operations: INSERT, UPDATE and DELETE.

- **INSERT** – With active-active replication, collisions on an INSERT operation should not occur if the database design is correct; that is, it considers UNIQUE key violations. In most cases, UNIQUE keys are generated by a SEQUENCE statement. There are mechanisms for handling sequence generation in a peer-to-peer environment to ensure uniqueness.
- **DELETE** – Collisions on a DELETE operation will be ignored since the DELETE operation is the end cycle for the data.
- **UPDATE** – The only conflict that has proven to be problematic occurs with the UPDATE operation. Here are some methods that organizations are using to resolve UPDATE conflicts:

  Timestamp resolution – The update that occurred last should be the one that wins. This means that the server ignores UPDATES with an older timestamp than the last UPDATE for the same row. If the timestamp on the UPDATE is newer than the last one applied to that row then the record will be applied.

  Host resolution – In an active-active environment, you can dictate the priority of operations based upon where the data is originated. In this scenario, the data from a lower-priority host will be ignored if there is a conflict. If the data came from a server with higher priority, it will be applied.

  Business logic resolution – A record's fate depends on the business logic applied. Since everyone's business logic is different, the mechanism for resolving conflict in this scenario should be provided by the organization's own code.

SharePlex for Oracle has built-in conflict-resolution procedures for timestamp resolution and host resolution. It also provides the ability to write business logic conflict resolution procedures using PL/SQL. With the power of PL/SQL, organizations can apply any business logic to resolve the data. Remember that the goal is to resolve the conflict as well as to bring the data back in sync on all nodes.

Conflict avoidance

Avoiding conflicts is cheaper than resolving them, of course; in fact, if you can avoid conflicts, the integrity of the data is preserved without any cost. Sometimes, conflict resolution on the target cannot be done automatically due to the complexity of the logic. In this case, avoiding conflicts is the best scenario.

Here are some methods for avoiding conflicts:

- In web-based applications, you can persist one connection throughout the transaction. This will isolate changes from one customer to one server, reducing conflicts.
- In some applications, the isolation can be done by the login: you can isolate a user's changes to a certain machine, avoiding conflict altogether.

Avoidance mechanisms cannot always prevent all conflicts, but they can reduce the conflicts to a small, manageable number. By using conflict avoidance, you may narrow down your actual conflicts to a handful of tables; then conflict-resolution procedures need to be applied to only those tables.

Conclusion

Organizations strive to minimize outages, with goals as stringent as just six minutes of unscheduled downtime a year. Active-active replication scales the application over multiple servers to eliminate ALL outages—scheduled and unscheduled—if implemented properly. That means considering a variety of factors, including unique key collisions, scheduling of update jobs, and methods of conflict resolution and conflict avoidance. Understanding the key considerations detailed in this paper, and using a powerful data replication solution like SharePlex for Oracle, will enable you to easily achieve your uptime goals with active-active replication.

SharePlex for Oracle has built-in conflict-resolution procedures for timestamp resolution and host resolution.

ABOUT QUEST

Quest helps our customers reduce tedious administration tasks so they can focus on the innovation necessary for their businesses to grow. Quest® solutions are scalable, affordable and simple-to-use, and they deliver unmatched efficiency and productivity. Combined with Quest's invitation to the global community to be a part of its innovation, as well as our firm commitment to ensuring customer satisfaction, Quest will continue to accelerate the delivery of the most comprehensive solutions for Azure cloud management, SaaS, security, workforce mobility and data-driven insight.

If you have any questions regarding your potential use of this material, contact:

Quest Software Inc.
Attn: LEGAL Dept
4 Polaris Way
Aliso Viejo, CA 92656

Refer to our Web site (www.quest.com) for regional and international office information.