Synchronized Database Service (SDS) White Paper

VERSION 7.1

Jade Software Corporation Limited cannot accept any financial or other responsibilities that may be the result of your use of this information or software material, including direct, indirect, special or consequential damages, or loss of profits. There are no warranties extended or granted by this document or software material.

You should be very careful to ensure that the use of this software material and/or information complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Jade Software Corporation Limited.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

For details about other licensing agreements for third-party products, you must read the JADE **ReadMe.txt** file.

Contents

Synchronized Database Service (SDS)

JADE's Synchronized Database Service (SDS) offers an effective solution to protect a business's core data assets, providing access to these assets on a 24x7 basis in the face of disasters and system outages.

The facility provides an out-of-box software solution to automating a hot standby database in one or more locations. JADE's Synchronized Database Service facility provides hot standby secondary databases, which can be used for both disaster recovery and offloading query workloads. Multiple secondary databases are supported, enabling sophisticated solutions to balance query workloads.

Today, business is increasingly dependent on information technology that provides enhanced productivity and competitive advantage. However, this increased reliance on technology introduces new challenges. If availability is compromised, then the entire business is vulnerable.

For more details, see the following subsections.

Availability and Disaster Recovery

In order to understand how JADE is positioned in the high-availability and disaster recovery solution space, in this section we review some industry concepts and terminology. These concepts are discussed in further detail in Appendix A of this document.

Availability

Availability is a measure of the degree to which a system can be used for its intended purposes during the times required by the business.

Planned Outages

Planned outages are scheduled system outages due to an application upgrade, system or environmental software upgrade, hardware upgrade, or maintenance.

Unplanned Outages

An unplanned outages is any inadvertent act or failure that causes the system to become unavailable, such as hardware or software failure, user errors, floods, fire, and so on.

Exposure and Recovery Objectives

What does a system outage cost an organization per hour? In the event of a permanent failure of the primary system, how long would it take to recover? How much data would be lost in the 'worst case' scenario?

For more details about service-level requirements, see the following subsections.

Service-Level Requirements

What does a system outage cost an organization per hour? In the event of a permanent failure of the primary system, how long would it take to recover? How much data would be lost in the 'worst case' scenario?

- Maximum acceptable recovery time
- Maximum acceptable data loss

These give rise to the following disaster recovery objectives. Having established these objectives, an organization can evaluate the cost of each potential solution.

Recovery Time Objective (RTO)

What is the business cost-justified maximum elapsed time to restore the system to a functional level?

Recovery Point Objective (RPO)

When the recovery time objective is met, what is the tolerable amount of data or transactional loss the business can tolerate? How much data can you afford to recreate?

SDS Overview

JADE's Synchronized Database Service has been designed to address the critical database recovery component of an IT-dependent company's business continuity plans.

SDS is aimed at providing a hot standby server solution for disaster recovery with better efficiency and lower cost than hardware remote mirroring solutions.

The SDS terminology and concepts used in this document are defined in detail in Appendix B of this document.

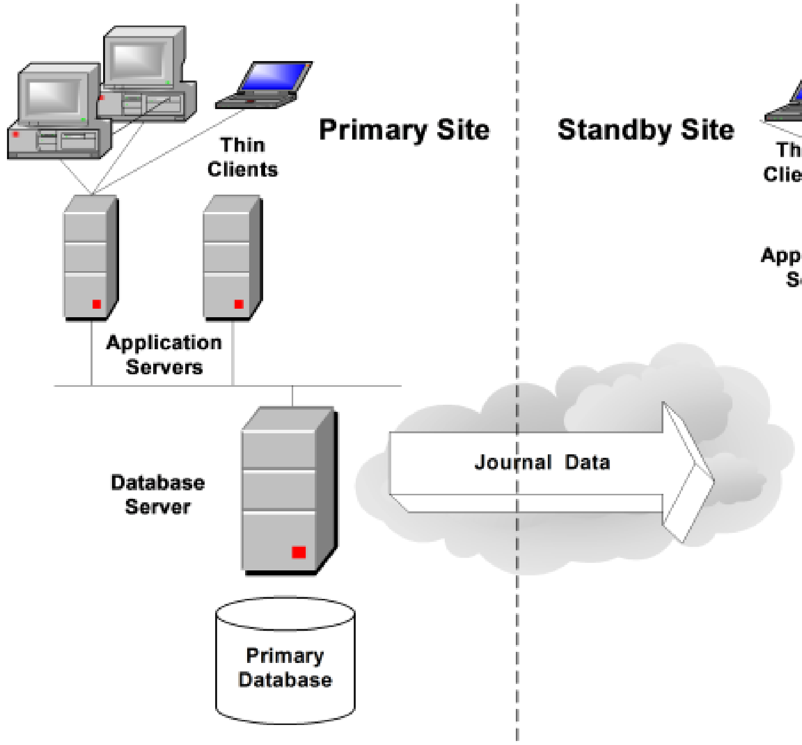
For more details, see the following subsections.

What is SDS?

The Synchronized Database Service is the name given to the software service in a JADE system that provides the functionality to keep one or more secondary databases synchronized with the primary database.

SDS automates the otherwise-manual process of maintaining a hot standby database that can be used if the primary database is taken offline for routine maintenance, becomes damaged, or is lost entirely.

The secondary database is synchronized by periodically applying the journal, data as shown in the following diagram.



Benefits of SDS

This section contains the benefits of SDS; that is:

- Efficient Solution for High Availability and Disaster Recovery
- Safeguard against Corruption
- Inbuilt Takeover Capabilities
- Centralized Management Interfaces

Efficient Solution for High Availability and Disaster Recovery

A JADE Synchronized Database Environment (SDE) combines loosely connected and geographically dispersed primary and secondary databases into a robust, easily managed disaster recovery solution. It is a superior data recovery solution compared to backup and recovery from tape, which is generally incapable of meeting high-availability recovery time objectives.

Safeguard against Corruption

Primary site physical corruptions, due to device failure, do not propagate through database journals that are transported and replayed by the secondary database.

Inbuilt Takeover Capabilities

The ability to perform hostile or negotiated takeovers provides for efficient role-reversals between primary and secondary databases, minimizing primary system downtime for planned or unplanned outages.

Centralized Management Interfaces

The JADE Database Administration framework and SDS user administration tool automate the management and operational tasks across multiple databases in an SDS environment.

A single SDS administration interface can monitor all databases participating in the environment.

SDS Functionality

A JADE SDS environment consists of a production database (referred to as the *primary* database) and one or more standby or secondary databases, usually hosted on different machines that may be in the same room or in entirely different geographic locations. Secondary databases are ‘transactionally’ consistent copies of the primary database.

The transactional consistency between primary and secondary databases is maintained using the database transaction journals. As application transactions modify objects in the primary database, the operations required to perform these changes are audited to the active transaction journal. This journal information is transferred to secondary destinations, where it is applied to the secondary databases by the database-tracking module of SDS.

For more details, see the following subsections.

Database Reorganizations

Changes to an application may require persistent objects to be converted to a new structure. This structural conversion, achieved by a process known as database reorganization, first takes place on the primary database. The operations required to achieve this are recorded in the database journal and are automatically applied to secondary databases.

Management Facilities

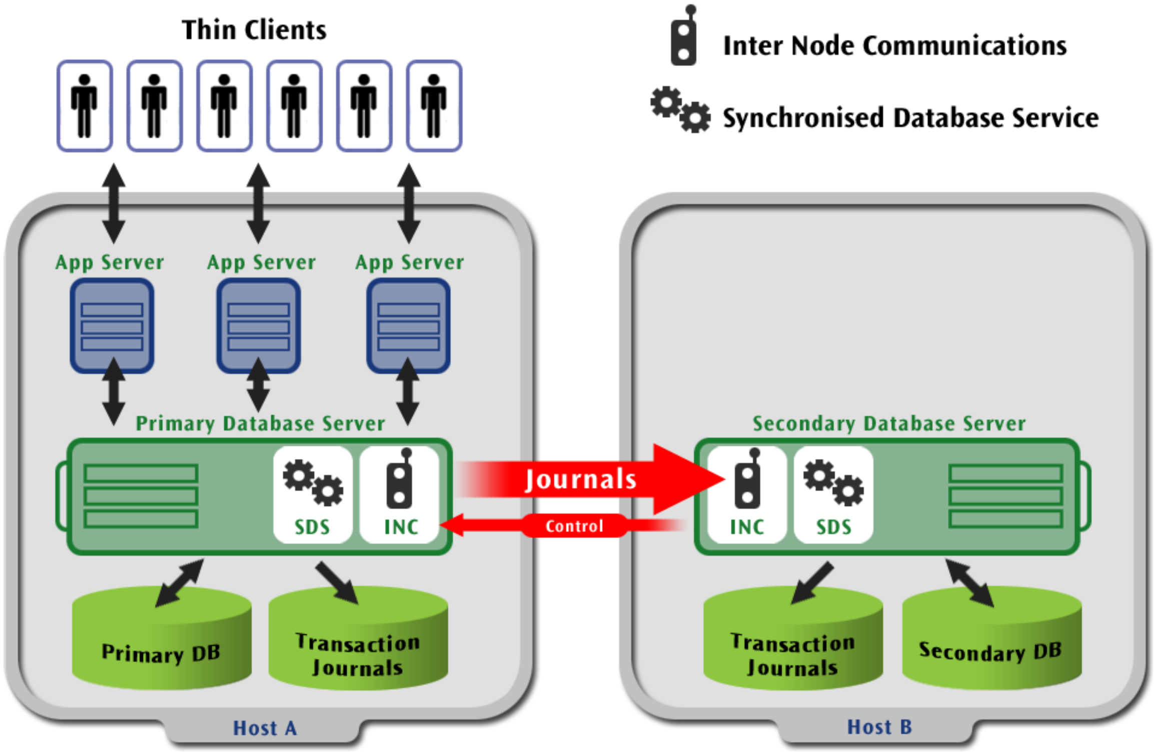
SDS provides several management interfaces, including an application programming interface or API, initialization file parameters, and the JADE SDS Administration utility, which presents a graphical user interface for monitoring and controlling an SDS environment.

Sample Usage Scenarios

This section contains the following sample usage scenarios.

- Example 1: Passive Secondary
- Example 2: Remote Active Secondary
- Example 3: Hybrid Configuration

Example 1: Passive Secondary



The above diagram of passive hot-standby configuration illustrates a single secondary database server operating in a passive ‘hot standby’ mode. SDS is an integral component of a database server node; the above diagram illustrates some of the key components of this service. The Inter Node Communications module (or INC) transports the messages used by SDS to communicate between database servers.

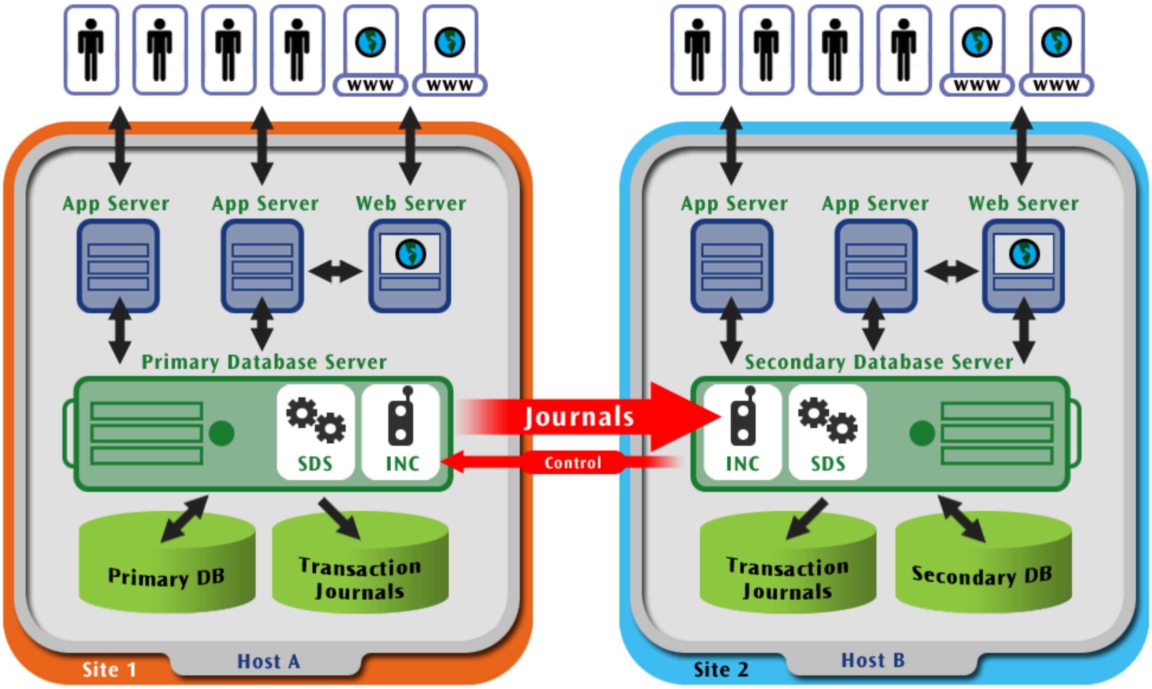
In this example, a secondary database is automatically synchronized with a primary updating copy in a hot standby role. The secondary database is termed *passive*, because there are no connected nodes supporting inquiry applications.

A passive secondary sever would typically be used for one of the following.

- An onsite local database backup strategy
- An offsite remote database backup strategy

Nothing special is required to convert a passive secondary system into an active secondary system supporting read-only applications. An active secondary can run locally at the same site as the primary, or remotely to support both disaster recovery and query offloading.

Example 2: Remote Active Secondary



The above diagram of active hot-standby configuration illustrates the deployment of a single secondary database server operating in an active 'hot standby' mode. The secondary server is installed at a different location to the primary. In this example, the secondary database provides a remote 'hot standby' database while at the same time supporting read-only database access.

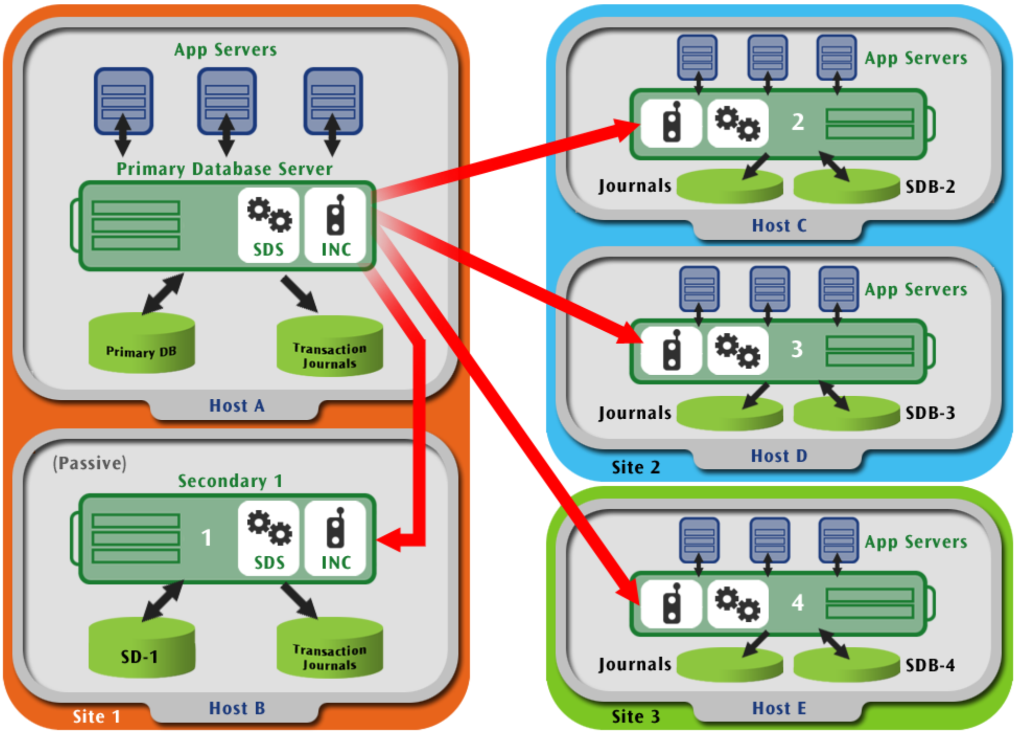
The main differences between this and the prior scenario are:

- The secondary database is hosted in a remote location to the primary host system
- There are active nodes connected to the secondary database server supporting inquiry transactions

Disaster recovery strategies require a means to quickly relocate processing from the primary processing center to a site in a different geographic location. In this scenario, the secondary database is located at Site 2 in a different geographic location to the primary database at Site 1, and can be used to restore database availability following a catastrophic failure that renders the primary processing site or hardware unavailable for any reason.

Example 3: Hybrid Configuration

The following diagram is an example of a hybrid configuration.



The above diagram of hybrid multiple secondary configuration illustrates an SDE comprised of multiple secondary database servers. This more-complex scenario combines the passive standby functionality from example 1 with the active remote standby functionality from example 2 and introduces the use of multiple read-only secondary servers for application load-balancing.

A single on-site passive 'hot-standby' server situated at Site 1 on Host B provides the local backup function. This can be used to restore database availability at the primary processing site (Site 1), by converting the secondary database on Host B to a primary and switching all update processing to the new primary located on site.

The three remote secondary databases are located at two different sites, providing triple site redundancy and the ability to partition inquiry processes and users across multiple servers for load-balancing.

Querying the Secondary Database

Running high volumes of online queries and reports on a live production database is often seen as a necessary evil. It will degrade application performance, but users require up-to-date information. On the other hand, running the same queries and reports on an SDS secondary database is an ideal alternative, because this activity no longer competes with updating transaction processing.

This means that even during busy reporting periods (for example, month- or quarter-end), application response time need not suffer from the additional burdens incurred from heavy reporting workloads.

For more details, see the following subsections.

Secondary Database Read Access

A key feature of SDS is the ability for query and reporting applications to access objects in a secondary database while the background tracking mechanism is updating it. The tracker process replays updates contained in database journals to keep the secondary database closely synchronized with the state of its primary.

Conceptually, the tracking mechanism can be viewed as a JADE node with multiple updating processes updating the database. In fact, special processes called *replay processes* are created in secondary servers to reflect and represent the original updating processes on the primary database. For convenience, these replay processes are associated with a single node called a *replay node.*

For more details, see the following subsections.

Transaction Isolation

The continuous updating activity performed by the tracker module means that at any time there can be uncommitted updates for one or more in progress transactions. JADE does not permit ‘dirty-reads’ (that is, access to object state changes made by a transaction before it is committed) on a primary database, and therefore dirty-reads are not permitted on a secondary database.

In order to provide inquiry processes with a transactionally consistent view of the database at all times, the effects of uncommitted transactions are hidden until the transaction commits. The state of an object in a ‘dirty state’ returned to the application is simply the last committed state.

The series of updates made within a transaction are hidden in their entirety until the transaction commits. However, the actual commit that allows the effects of the transaction to become visible may be delayed due to share locks held by reader processes.

Share Lock Semantics Preserved

In order to preserve object share lock semantics on a secondary, the effects of a transaction are not made visible until there are no conflicts between objects updated by the transaction and share locks held by reader processes.

In addition, since successive transactions can be dependent on the results of prior transactions, it is necessary to ensure that transactions commit in the same sequence on the secondary as they did on the primary. This means that a transaction commit that is deferred due to conflicts can also delay the commit of successive dependent transactions.

Effects of Transaction Isolation on Journal Retention

While transactions are being isolated, a secondary may need to retain access to database journals that have been replayed. In particular, the journal containing the oldest transaction being isolated and all subsequent journals are required to be available to the secondary.

The required journals are kept open and will not be released until they are no longer required; journal close actions are deferred until a journal is released.

Cache Coherency

Since a secondary system used for inquiry purposes can be made up of multiple nodes, each with its own object cache, JADE supports similar mechanisms to those available on the primary system to keep caches current. The simplest way to ensure caches are kept current is to enable automatic cache coherency, and JADE will take care of the task for you.

In addition to automatic cache coherency control, both system event and persistent user event notifications are supported on secondary systems.

System Event Notifications

When a transaction commits on a secondary database, system events are caused for objects that were created, updated, or deleted within the transaction and any subscribers to the events are notified. For example, inquiry applications can subscribe to system events in order to trigger a refresh of user interface elements.

Persistent User Events

Selected user events, termed *persistent events*, can be audited by a primary database for replay at all secondary databases.

In a primary system, subscribers are notified of non-immediate user events when the transaction that caused the event commits. Similarly, when a persistent user event is replayed on a secondary system, subscribers to the event are notified when the transaction containing the event commits on that secondary.

Persistent user events can be achieved in one of two ways.

- Globally, by setting the **AuditCauseEvents** parameter in the [SyncDbService] section of the JADE initialization file to **true**.
- Selectively, by leaving the **AuditCauseEvents** parameter disabled and using the **Object::sdsCauseEvent** method to explicitly cause user events within a transaction for secondary replay.

The global approach allows all user events to be made persistent in existing applications without changing the application, but will incur unnecessary auditing overhead for the events that are not used. The selective approach is preferable when only some user events are likely to be of interest to inquiry applications on the secondary.

Inter-System Event Notifications

The **Object** class **sdsCauseEvent** method can be used to achieve inter-system event notifications.

The two role-dependent usage scenarios for inter-system events are:

- From a primary system, to cause a persistent event that is audited by the primary database and replayed by secondary databases.
- From a secondary system, to cause an event that is notified to subscribers on the primary system.

Environmental Objects

Certain environmental objects (in particular, system, node, process, and Web session objects) have their own independent lifetime in a secondary database; they are also not the same objects as their counterparts on the primary database. Since these environmental objects are not part of the persistent database itself, user processes executing in the secondary system are allowed to update them.

Scalability and Load Balancing

A single JADE system allows you to distribute a workload across multiple nodes in the system. JADE nodes can be located on different physical machines. SDS brings an extra dimension to workload partitioning, which allows inquiry workloads to be further distributed across multiple database servers.

For more details, see the following subsections.

Scalability

Multiple secondary servers can be configured in an SDE to allow inquiry workloads to be spread across several discrete JADE systems. Furthermore, additional secondary servers can be brought online at any time without restarting the environment.

Adding new secondary database servers to an SDE is a simple process of installing the software and a recent database backup on a new host, setting up the required configuration parameters in the JADE initialization file, and starting the server. If it is necessary to upgrade the primary licence key to support additional secondary servers, the licence upgrade can be performed online. When a new secondary database server joins an SDE, it will start receiving and replaying journals in catch-up mode until it is synchronized with the primary.

Inquiry processing can commence on a new secondary while it is still catching up, as soon as any interrupted transactions have been resolved.

Workload Partitioning

There are various ways to partition inquiry workloads across multiple secondary servers. Workloads can be partitioned statically, dynamically, or a combination of statically and dynamically.

Static Partitioning

With static partitioning, each secondary is statically assigned a specific workload function. For example, one secondary might be dedicated to a passive standby role for backup purposes alone, another performs offline reporting, and a third supports online inquiry processing from thin client users. Any of the three secondary databases could take over the primary role in a disaster situation.

Dynamic Partitioning

Dynamic partitioning can be employed where it is desirable to load-balance inquiry users or transactions across multiple secondary systems. This type of partitioning is commonly employed in Web server farms, where Web queries are load-balanced across a number of servers while update transactions are directed to a central updating server.

Load-Balancing Options

Load-balancing is a means to distribute connections or Web messages across multiple processing nodes. There are various ways to implement a load-balancing solution, including but not limited to:

- Using Domain Name Server (DNS) to round-robin target Internet Protocol (IP) addresses
- A front-end application that distributes thin client connections or Web messages across nodes
- A load-balancing network router

Third-party load-balancing hardware can be used to load-balance HyperText Transfer Protocol (HTTP) requests, for example. Typically, load-balancing routers map a single virtual IP address obtained from DNS to a target IP addresses from an address pool using heuristic algorithms and user-selectable criteria. The heuristics take into account metrics such as network link congestion, server health, proximity to client, response times, and so on, to select a target IP address.

Web interactions in the form of Web services SOAP messages or plain HyperText Markup Language (HTML) requests can be load-balanced as long as they don't depend on server-maintained state. This means that if you are considering load-balancing Web services requests to providers across multiple database servers, any application dependency on session-state handling must be avoided.

SDS Operational Considerations

Secondary Database Synchronization Modes

The secondary synchronization mode is a user-configurable option that determines how a secondary database is kept synchronized with the primary. The parameter is configured for each SDS secondary and the setting can differ between secondary databases in an SDE.

The two secondary synchronization modes available are:

- Journal Switch
- Journal Block Write

For more details, see the following subsections.

Journal Switch Synchronization Mode

The goal of the Journal Switch synchronization mode is to establish and maintain periodic synchronization between the primary and secondary audit trails. Synchronization can be within one completed journal.

Under the Journal Switch synchronization mode, SDS automatically transfers a completed journal to connected secondary databases after writing has switched to a new journal on the primary. The SDS software automatically acknowledges receipt of the journal, and the tracker process reads and applies the audited updates from the journal to the database.

The Journal Switch synchronization mode makes intermittent use of the network communications for the transmission of journals and control information.

Benefits

The Journal Switch mode provides journal shipping with a minimum demand on your network and processor capacity. Journal file size can be controlled using the **JournalMaxSize** and **JournalMinSize** JADE initialization file parameters and the **JournalSwitchInterval** parameter can be used to limit the number of transactions that have not been sent to secondaries to transactions committed within a specified number of minutes.

Journal Block Write Synchronization Mode

The goal of the Journal Block Write synchronization mode is to transfer journal blocks to secondary databases as they are generated and written to the primary database journal. Under this mode, SDS can establish and maintain very close synchronization between primary and secondary journals, and hence a close synchronization of the databases.

When Journal Block Write mode is configured for a secondary database and it is synchronized with the primary, audit blocks are transferred and written to the secondary database journal when they are written to the journal on the primary. In effect, primary journal writes are mirrored to the secondary journal. These mirrored journal writes are performed asynchronously, so that primary database transaction processing is not held up waiting for acknowledgments from secondary databases. In journal block write mode, the database tracker thread continuously reads data from the end of the current journal applying the updates to the database.

If a secondary database becomes unavailable at any time, transaction processing can continue on the primary database. The current asynchronous mode of mirroring journal writes offers a relatively high degree of performance with a low impact on primary database processing but does not *guarantee* zero transaction loss.

Benefits

The principal benefits of using the Journal Block Write synchronization mode are:

- Synchronization of journals on the primary and secondary databases is closer than is possible with Journal Switch mode.
- Loss of audit data is minimized so that in a disaster recovery situation most, if not all, transactions that committed on the primary can be recovered.
- Following a takeover, restoration of database access can be faster than with Journal Switch mode.

Synchronization Mode and Disaster Recovery

The mode of database synchronization that you choose is tied to the following key disaster recovery objectives.

- The length of time required to restore database access following an interruption (Recovery Time Objective).
- The amount of data that will be lost as a result of an interruption (Recovery Point Objective).

Journal Block Write Mode

If you have a secondary database that is synchronized with its primary database when the primary becomes unavailable, you will be in a very good position to recover the database quickly and with minimal loss of data. Since all audit data written to the primary database is sent to the secondary as it is written, in a best-case disaster recovery scenario zero transaction loss can be achieved. However, since transaction commits on the primary do not wait for acknowledgments that the journal block containing the commit is stable on disk on the secondary, this asynchronous synchronization mode does not *guarantee* zero transaction loss.

Journal Switch Mode

You can still achieve quite fast recovery with a minimal loss of data if you operate SDS with a delayed level of synchronization by using the Journal Switch mode. Because you have a database already set up to take over the operations of the primary database, you can apply outstanding journals as quickly as possible, and be back online within a predictable length of time.

Considerations about Synchronization Levels

Before choosing a synchronization mode, you need to consider the impact of losing data if the primary database becomes unavailable. The more-closely synchronized the journals, the smaller the transaction loss will be in a Disaster Recovery (D/R) takeover situation. If most of your transactions are easily reproducible, having closely synchronized databases might not be so important to you.

The workload and performance of the database, the host machines, and the network are tightly integrated. A heavy workload in any one component can impact the performance of any of the other components. The more-closely synchronized the databases, the more-sensitive your environment becomes to heavy workloads in any one component.

Configuring Synchronization Modes

The Synchronization Mode is configured for each secondary database by using the **SyncMode** parameter specified in the [SyncDbService] section in the JADE initialization file. The valid values for this parameter are **JournalBlockWrite** (the default value) and **JournalSwitch**.

If you change, or are considering a change, from **JournalSwitch** to **JournalBlockWrite** mode and all secondary databases in an SDE will be configured to use the **JournalBlockWrite** value for the **SyncMode** parameter, we recommend that you ensure the **JournalSwitchInterval** parameter is set to zero (**0**), because in **JournalBlockWrite** mode, switching journals on a time-based interval is no longer of value in terms of minimizing Recovery Point or Recovery Time objectives, and use of this increases processing and management overheads. For similar reasons, we also recommend that you consider increasing the **JournalMaxSize** value if this is set lower than the default 64M bytes.

Takeovers

SDS supports administrative operations designed to handle planned and unplanned outages of the primary system. These operations are called *takeovers*. A takeover is the process that allows a nominated secondary database to assume or take over the role of the primary database.

The two principal forms of takeover are:

- Hostile takeovers
- Negotiated takeovers

For more details, see the following subsections.

Hostile Takeovers

A *hostile takeover* is the conversion of a secondary database to a primary database role without involving the current primary. *Failover* is an alternative industry term used to describe this type of operation.

A hostile takeover is used when a catastrophic failure occurs on the primary database and there is no possibility of recovering the primary in a timely manner. A hostile takeover is invoked on the secondary database that will assume the primary role.

Negotiated Takeovers

A *negotiated takeover* is the planned role-reversal of the primary and a nominated secondary database.

The main difference between a negotiated and a hostile takeover is that a negotiated takeover retains database synchronicity, allowing the primary to resume a secondary role almost immediately. Furthermore, the operation can be reversed. As a result, negotiated takeovers can be performed routinely or more frequently. A typical reason for performing a negotiated takeover would be to allow for scheduled maintenance of the primary host where you will move processing to the secondary, take the old primary database offline, and once the maintenance is complete, revert back to the original database roles.

Considerations before Performing a Takeover

Sites might consider performing a takeover when the primary database will be unavailable for an extended period. However, in some instances it might be less-disruptive to wait for the original primary to become available again rather than to perform a takeover.

The decision to perform a takeover operation should be based on several considerations.

- If the primary host or the network has failed, the time required to restore them or to revert to backup hardware at the primary site
- If the primary database has been damaged, the time required to restore the database from backup using standard restore and recovery techniques
- If the secondary replay process is lagging behind the primary, the time taken to replay the outstanding journals
- If the journals cannot be synchronized due to a primary host or network failure, the impact of losing transactions or having to reprocess transactions

Preparing for a Takeover Operation

Prior to performing a takeover, the following steps are recommended.

1. If the reason for the takeover is to perform scheduled maintenance on the primary host, it is preferable to shut down all nodes connected to the primary *before* initiating the takeover.
2. Where possible, shut down all update-capable processes on the primary database. Stopping update processing ensures there are no active transactions when the takeover is initiated; a database quiet point on the primary is a prerequisite to the takeover action. If updating applications are left running on the primary system, they will encounter exceptions next time they attempt to update non-environmental objects once the primary database has assumed a read-only secondary database role.

On the other hand, there is no real need to stop inquiry processes on the secondary that is to assume the primary role. As soon as the takeover is accomplished and the secondary has changed roles to a primary, the inquiry processes can begin to perform updating transactions without the need to restart.

Re-Establishing Thin Client Connections

Following a takeover, thin client users will need to be moved to the new primary system; there are various logistic issues to consider.

- In a failover at a remote site, a network reconfiguration may be required to switch user access to the remote site. This is less of an issue if users access the systems via a Virtual Private Network (VPN) delivered across the Internet.

  Switching a dedicated network may require intervention by the communications provider; alternatively, redundant ‘always on’ networks can be configured.

- The next issue is to move thin client connections to a new application server host, which can be achieved in several ways.

  If the clients connect using Domain Name Server (DNS) host name, a DNS update to redirect the host name to a new IP address will be required.

Outline of a Negotiated Takeover

In a negotiated takeover, the primary database prepares to relinquish its role, requests the nominated secondary database to assume the primary database role, and waits for a positive or negative acknowledgment from the secondary. The primary database will not relinquish its role until a go-ahead is received from the secondary.

Before a primary database relinquishes its role, it must achieve a database quiet point - a point at which no transactions are active. If a database quiet point is not achieved within the configured ‘maximum wait for quiet point interval’, the takeover is abandoned.

Once the primary database is ready to relinquish its role, it audits a *change-to-primary* request in the current journal and closes the journal. The journal containing the *change-to-primary* request is available for transfer to all connected secondary databases, but only the nominated secondary actions the request.

When the secondary database nominated to take the primary role has replayed all transactions up to and including the *change-to-primary* audit record, the secondary is synchronized with the primary database at the takeover quiet point. If there are no reader processes running on the secondary at this point, the secondary can assume the primary role immediately.

On the secondary, if inquiry processes are executing, these processes may hold locks that are causing the effects of one or more replayed transactions to be isolated (effects hidden from all readers). In order for the secondary database to assume a primary role, these isolated transactions must be made visible, since hiding committed transactions is not a function of a primary database. This creates a dilemma of whether to give precedence to the takeover operation or to the reader processes. The resolution of this dilemma is delegated to the administrative user initiating the takeover.

When initiating a negotiated takeover, the user must decide between a *conditional takeover* and a *forced takeover*. The decision should be based on what is more important at the time: allowing long-running processes to complete or the takeover itself because maintenance on the primary host has been scheduled to occur at a specific time and cannot wait.

For more details, see the following subsections.

Conditional Takeover

In a conditional takeover, inquiry processes are given precedence. If an inquiry process is holding share locks that conflict with an applied but not yet-committed transaction, the secondary returns a negative acknowledgment to the primary; the takeover is abandoned, and the primary retains the primary role.

Forced Takeover

In a forced takeover, the takeover operation is given precedence. When the secondary database processes a ‘change to primary’ request with the **force** parameter and conflicts with reader processes exist, the secondary attempts to force through the visibility of currently isolated transactions one at a time. This is achieved by terminating user processes holding locks that conflict with isolated transactions until no conflicts remain. When all isolated transactions have been committed (made visible), the secondary database assumes the primary database role and returns a positive acknowledgment to the primary, allowing it to relinquish its role.

Takeovers in a Multiple Secondary Environment

In an environment with more than one secondary database, when the new primary has processed the *change-to-primary* request, other secondary databases can be lagging behind in either replay or journals received. The takeover process does not wait for them to catch up; instead, when the current primary receives the go-ahead from the new primary, a *change-primary* message is broadcast to the other connected secondary servers to inform them of the role change and the name of the new primary system.

On receipt of a *change-primary* message, secondary servers complete any in-progress replay jobs, update their configuration parameters, close their connection to the old primary, and connect to the new primary database server.

Once reconnected, secondary databases receive journals from the new primary and the journal transfer, and replay processes resume from where they left off.

Configuration Options

Secondary databases can be remote (connected over a Wide Area Network (WAN)) or local (connected on a Local Area Network (LAN)).

Remote secondary databases are the best solution for disaster recovery because an event that disables the primary database will be unlikely to also disable the secondary. However, a WAN's lower bandwidth and higher latency can impact recovery point objectives.

Local standby databases are better-suited to address outages related to human errors or data corruptions. Since a LAN provides an inexpensive and reliable network link with low latency, having a dedicated LAN segment to support a standby server is an ideal environment for providing a high degree of data protection. In scheduled takeovers that require an orderly shutdown of the primary system, shipping the last journal and application of it to the secondary database can be very fast operations if running locally.

Secondary Database Recovery

Restarting a Secondary Following a Crash

When a secondary database server is stopped or crashes with transactions in progress and it is subsequently restarted, the database remains in a 'recovery required' state until all interrupted transactions have been resolved. Interrupted transactions are resolved when the commit or abort audit records are eventually replayed. As long as interrupted transactions exist, read access to the secondary database is not permitted.

Roll-Forward Recovery

Roll-forward recovery can be manually initiated to recover from a backup through available journals before opening the database. However, SDS offers a straightforward alternative to roll-forward recovery; simply restore an online or offline backup and all available journals, then instead of performing a roll-forward recovery, open the database by starting the secondary server node and the tracker process will begin replaying these journals automatically.

In either recovery mode (that is, restart or roll-forward), the secondary databases must avoid undoing transactions that originally committed on the primary, otherwise the databases would lose synchronicity and diverge. For that reason, roll-forward recovery to a specific time on a secondary database is not permitted.

Delayed Auto-Start of Server Applications

In a secondary system, the automatic initiation of initialization file-specified database server and application server 'server applications' will be delayed until read access to the secondary database has been granted. In practice, this will occur when:

- The value of the **ReadAccessDisabled** parameter is **false**
- All interrupted transactions have been resolved by journal replay

SDS and Backup Strategies

Recovery Time Issue

More and more enterprises want to maintain business continuity with minimal downtime. In the event of a service failure, the paramount issue is the time it takes to recover the system; in other words the 'recovery time objective', an expectation of which may be embodied in a service-level agreement.

The predominant IT recovery strategies for databases utilize some form of backup and restore mechanism.

The time taken to recover a system includes the time to:

- Restore hardware infrastructure
- Restore the operating system
- Restart the system
- Establish connectivity
- Restore application software and data
- Recover application data

Everywhere, databases are growing in size. In the year 2000, approximately 2500 Petabytes of disk were shipped. In 2003, over 13,000 Petabytes of disk were shipped.

Large databases can no longer be backed up within backup windows. All complete database restore and recover mechanisms take longer than it took to perform the backup, and except in rare specific circumstances, a full backup must be loaded before applying updates from partial or incremental backups.

Hot Standby Database

An SDS secondary database in a ‘hot standby’ role provides a realistic alternative to taking regular full-backups of the primary.

The secondary database is in its own right a database backup and is kept synchronized with the primary database as journals are completed on the primary and transferred to the secondary database. In the event of a catastrophic failure that results in the partial or complete loss of the primary host system, processing can be switched to the secondary database, minimizing downtime without the need to restore a backup from archival media.

The secondary database can be converted to a primary database to support updating processing and all available journals can be applied using roll-forward recovery so that the database is brought as up-to-date as possible. Having service restored *in minutes* is achievable.

Additionally, a hot standby server eliminates the time to:

- Restore hardware infrastructure
- Restore the operating system
- Restart system elements

Backing Up a Secondary Database

Since it is always a good idea to keep multiple database backups, and preferably some offsite, it is also possible to backup an SDS secondary database while the SDS synchronization process is updating it. The ability to backup a secondary database can be used to avoid ever performing a direct backup of the primary database so there is no longer any impact or issues with ‘Backup Windows’ affecting processing on the primary system.

The online backup routines are cognizant of the fact they are running on a secondary database, and they process checkpoint procedures and control file updates appropriately.

Offline Backups

A regular offline backup of a secondary database can be performed after stopping the secondary database server.

Restoring a Secondary Database Backup

When a secondary database is restored, the database image retains its secondary role. Roll-forward recovery can be performed, but rolling forward to a specific time is not permitted. Roll-forward recovery on a secondary database that terminates with transactions outstanding will not *undo* the incomplete transactions; any active transactions at the end of recovery remain in an incomplete state.

Delayed Replay

While a primary database is open and active with updating transactions in progress, journals continue to be generated and transferred to connected secondary servers. It is possible to delay the application of transactions at one or more secondary databases by disabling tracking. This can be very useful in scenarios such as an application deployment, especially one involving a database reorganization that cannot be guaranteed to succeed. In scenarios involving such upgrades, it is advisable to disable tracking on at least one secondary database.

When invoked, the schema upgrade and reorganization processes on the primary will append redo information to journals, which continue to be transferred to attached secondary servers even when tracking is disabled.

Once the application deployment has completed successfully on the primary system, tracking can be re-enabled, allowing the secondary to replay the upgrade and reorganization processes to regain synchronization with the primary database. If an application upgrade should fail for any reason, the secondary database remains synchronized with the primary at the point prior to the upgrade.

Other Considerations

Journal Close Actions

The journal close action is triggered whenever a journal is released, to enable sites to automate the management and archiving of journals. These automated actions are independent of SDS activity but there are certain ramifications in the context of a SDS environment that should be considered. Journal close actions are configured in the JADE initialization file and include:

- Move
- Copy
- Remove
- CopyAndCompress
- MoveAndCompress.

The **Copy** and **Move** actions are used to automate transferring journal files to a specified journal archive directory. Different journal close actions can be configured on the primary and each secondary database. The **JournalCloseAction=Remove** action is not valid for a primary database, since this would make it impossible to synchronize a secondary that came online and required historical journals to catch up. On the other hand, a **JournalCloseAction=Remove** action may be acceptable on a non-critical secondary deployed for reporting purposes.

Automatic Resynchronization

SDS can handle network issues that result in the temporary loss of connectivity between primary and secondary databases as long as the required journals remain accessible to the primary database.

When connectivity from a secondary to the primary is re-established, journal shipping or mirroring continues from where the secondary database left off. The primary database will first look for required journals in the journal current directory and then in the journal archive directory.

When a required journal is not available, automatic journal shipping will be temporarily suspended. Operator action is then required to restore the missing journals and to resume shipping to the secondary.

Technology Comparison

SDS and Remote Volume Mirroring

Although remote volume-mirroring solutions can offer simple and complete data protection, SDS with its journal-based approach is inherently more efficient, less expensive, and better optimized for protecting a JADE database.

There are no third-party remote mirroring solutions that offer integration with a JADE database, which places constraints (see the following subsection) on how they must operate.

For more details, see the following subsections.

Better Network Efficiency

When a volume-mirroring solution is used to implement a remote database backup, the database files as well as the transaction journals must be mirrored at the remote site. In addition, *all* writes to *both* volumes must be synchronous in order to guarantee the database can be recovered to a consistent state in the event of a failover. With a synchronous mode of operation, every physical write to the database and journal must wait for an acknowledgment from the remote site. These constraints can have a significant performance impact on the primary database, impacting both transaction throughput and response time.

The need to achieve acceptable levels of performance on the primary will generally restrict the separation between primary and secondary sites to campus or metropolitan distances; beyond this, the impact on primary system performance will become intolerable. On the other hand, if both sites are located in the same metropolitan area, they can both be impacted by a disaster caused by weather, power grid failures, faulty telephone company circuits, and so on.

Some remote volume-mirroring solutions support an asynchronous copy mode, which does not suffer an impact from network latency issues. However, in this mode, the remote copy of the database is a fuzzy copy and will require periodically quiescing the primary database and synchronizing the remote target. Such a solution would entail regular and periodic downtime on the primary, which is unacceptable for high-availability systems.

Improved Resilience and Protection

The SDS journal transfer and tracker processes verify the structure and validity of audited data and ensure that transactions are applied in the correct sequence. In addition, SDS allows transaction replay to be delayed to provide windows of protection. This higher-level of checking and update propagation can help prevent many human errors and environmental data corruption incurred at the primary site from propagating to secondary databases.

Remote mirroring does not have that advantage; any error such as the accidental removal of a database file or corruptions introduced by a ‘rolling disaster’ as discussed in Appendix A will instantly propagate to the remote copy of the database.

Higher Return on Investment (ROI)

With an SDS solution, secondary databases can be used for inquiry and reporting while updates are still propagating. That is not possible with a remote mirroring solution, since hardware-based remote mirroring cannot provide a transactionally consistent view of the database.

There are no third-party remote mirroring solutions that offer the required integration with a JADE database.

SDS and Hardware Clustering

SDS and hardware clustering are complementary technologies. Hardware cluster solutions address system failures. Clusters can also provide scalability options for an application; for example, JADE application server nodes can be allocated across machine nodes in the cluster. However, clusters do not provide any data protection since there is a single copy only of the database within the cluster, typically resident on a disk shared between the physical machine nodes.

SDS can be deployed to incorporate the missing level of database protection into an existing clustered environment.

Summary

The Synchronized Database is a comprehensive high-availability and disaster recovery solution. It provides an out-of-box software solution to automating a hot standby database in one or more locations.

SDS secondary databases can be used for both disaster recovery and offloading query workloads. Inquiry workloads can be further partitioned and load-balanced across multiple secondary databases. Deploying one or more SDS secondary databases is a realistic alternative to tape-intensive backup and restore strategies, especially when time to recover a system to a functional state is crucial to an organization’s business continuity strategy.

Appendix A Disaster Recovery Concepts

In this appendix, we review some industry concepts and terminology. Disaster recovery is a complex subject and detailed coverage is beyond the scope of this paper. Consider this a whirl-wind tour of the main concepts.

For more details, see the following subsections.

Disasters

Disasters do not make good house-guests. They do not call ahead of time and they do not knock, but like it or not, disasters happen. One may happen to your organization tomorrow or perhaps one has happened already. Some are those that everyone hears about: earthquakes, hurricanes, floods, and fires. Others no-one hears about – often because they are embarrassingly avoidable.

The increasing reliance on computer systems is a double-edged sword. While computing systems can significantly enhance the productivity and efficiency of a business, it is more vulnerable to a profound negative impact of a catastrophe, measured in terms of lost revenue and lost opportunity.

Today more than ever, the concepts of high availability have an increasing role to play in an organization’s Information Technology delivery. Moreover, high availability does not end within the concepts of protecting local hardware against failure; it now has a major role to play in an organization's disaster recovery strategy.

Rolling Disasters

A disaster such as a flood, earthquake, or hurricane does not result in an instantaneous data center failure. In these scenarios, a disaster occurs over a period of several minutes - disk drives, tape drives, servers, and network controllers fail randomly over a short period of time.

Typically, database files and journals are hosted by different storage subsystems that will not fail at exactly the same time. It is during the several minutes of a disaster happening (called a *rolling disaster*) that data gets corrupted. Certain types of disaster recovery infrastructure, such as those based on remote volume mirroring, can in fact duplicate physical corruption at the remote site.

Availability

Availability is a measure of the degree to which a system can be used for its intended purposes during the times required by the business.

System availability is affected by the components described in the following subsections.

Planned Outages

Planned outages are scheduled system outages due to an application upgrade, system or environmental software upgrade, or hardware upgrade or maintenance.

Unplanned Outages

Unplanned outages are any inadvertent act or failure that causes the system to become unavailable, such as hardware or software failure, user errors, floods, fire, and so on.

The ideal paradigm is continuous availability where no outage ever occurs. Even if man were able to design and manufacture flawless commercial computers, they would probably be prohibitively expensive.

Fault-tolerant systems protect against unplanned outages by replicating hardware and using specialized operating systems to manage transparent cut-over from a failed component to its redundant partner.

With high-availability systems, on the other hand, you *do* expect both planned and unplanned outages. However, you expect the downtime associated with such outages to be small. In high-availability systems like fault-tolerant systems, hardware is replicated but cut-over from a failed component usually requires an outage.

Disaster Recovery (D/R)

It is important to consider the following issues.

- How would a shutdown of your IT systems affect your business?
- Is your data and business-critical processing protected from a site disaster?
- What does a system outage cost an organization per hour?
- In the event of a permanent failure of the primary system, how long would it take to recover?
- How much data would be lost in the ‘worst case’ scenario?

For more details, see the following subsections.

Business Continuity

Today, what is required is a business continuity plan. Such a plan ensures business functions can be restored following all potential disruptions within a tolerable outage window. For many business functions, that window can be large (greater than 72 hours). For the most-critical functions, however, the window must be non-existent.

While often thought of as being one and the same, there is a critical distinction between disaster recovery and business continuity. A disaster recovery plan should be just one component of a broader business continuity strategy to keep the organization’s operation continuing as usual, no matter what kind of disruption occurs – planned or otherwise.

Exposure and Recovery Objectives

How do you determine the appropriate recovery window for your business functions? You begin by understanding the exposure caused by that business function. This exposure must be quantified in real terms that apply to your business. Typically, that means calculating lost revenue per hour of disrupted service. Only when you know how much you might lose can you formulate an appropriate business continuity plan.

Assessing risk associated with IT systems and planning to mitigate that risk is ultimately a business decision, not a technical one. Important information is captured in this Business Impact Analysis that then enables you to devise a set of requirements for a D/R plan. This is done in conjunction with what you already know; that is, your:

- Business requirements
- Service-level requirements

For more details, see the following subsections.

Business Requirements

Business requirements should include overall revenue and profit objectives, customer satisfaction, and so on.

Service-Level Requirements

Service-level requirements are commitments to provide service to customers and other dependent parts of the organization, within specified time frames.

The results of this analysis should reveal:

- Maximum acceptable recovery time
- Maximum acceptable data loss

These give rise to the following disaster recovery objectives.

- Recovery time objective
- Recovery point objective

Recovery Time Objective (RTO)

What is the business cost-justified maximum elapsed time to restore the system to a functional level?

Recovery Point Objective (RPO)

When the recovery time objective is met, what is the tolerable amount of data or transactional loss the business can tolerate? How much data can you afford to recreate?

Planned outages are just as effective at removing service from the end-users as unplanned outages, and they are much more frequent yet typically, disaster recovery solution cost-justification is attempted based on the unplanned outage cost alone.

A realistic return on investment analysis should also consider the affects of planned outages.

Disaster Recovery Tiers

Once recovery time and recovery point objectives have been established, the next step is to select a recovery solution that can match those objectives. Unfortunately, disaster recovery facilities do come at a cost. A D/R plan should not be so expensive that it outweighs any benefit gained by doing it at all. It is, after all, like an insurance policy and should be cost-justified. In a perfect world, all data and applications would be treated equally in disaster recovery planning. Unfortunately, this is financially impractical.

To assist with this selection, analysts generally aim to categorize business functions into recovery classes or tiers, with different recovery point and recovery time objectives for each tier.

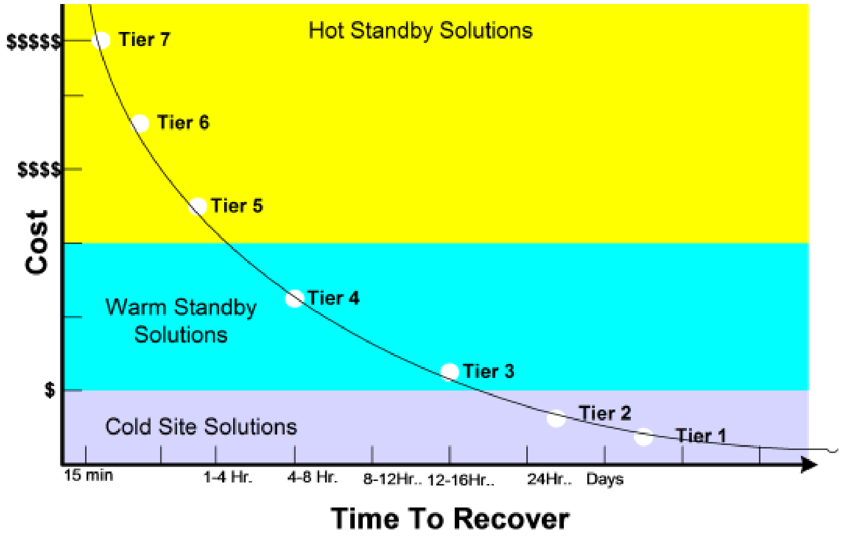
The RPOs and RTOs are used to guide the disaster recovery strategy of each tier. There is no industry standard classification. However, for purposes of comparison, we will use a tier classification first presented at an IBM SHARE user conference in 1992 that’s since been extended and is still in common use in IBM circles today.

At one extreme, the highest tier, the business function and data are deemed to be so critical that the recovery point objective is to the "*last committed transaction*" and Recovery Time Objective is "*immediate recovery*".

At the other extreme is tier zero (no recovery strategy). This tier provides no preparation in saving information, establishing a backup hardware platform, or developing a contingency plan. The length of time for recovery is unpredictable. Your data can be regarded as unprotected and if you are in this tier, you really do not care about your data.

The diagram later in this topic of tier classification for disaster recovery solutions based on IBM SHARE definitions illustrates the various tiers of recovery on a chart of cost versus recovery time. In general, there are three main bands of application criticality; within each band, there are tiers. The tiers can be summarized as follows.

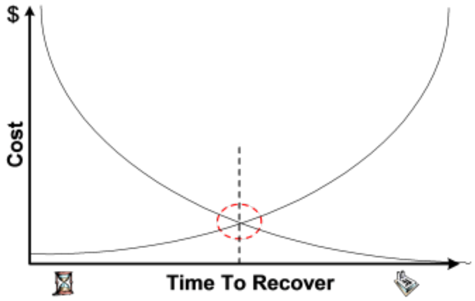
Tier	Description
Tier 0	No recovery capability
Tier 1	Offsite backup
Tier 2	Offsite backup, tape intensive recovery, business recovery service
Tier 3	Electronic vaulting
Tier 4	Warm standby site, fuzzy disk mirroring, point-in-time copies
Tier 5	Hot standby site, software database-specific recovery
Tier 6	Near zero or zero transaction loss
	Automated takeover
Tier 7	Near zero or zero transaction loss
	Highly automated takeover



Cost Assessment

There are fairly simple relationships between the cost of recovery versus time and the cost of an outage versus time. Generally speaking, the higher the cost of the D/R strategy the quicker you can recover, thus making the cost of the outage low. The lower the cost of the strategy, the longer it takes to recover, making the cost of the outage higher.

In striking the right balance, the aim is to find an effective compromise between the cost of recovery and the cost of the outage. If you could graph the two metrics, the compromise point is where the two curves intersect. This is the point at which both costs are at their minimum. Movement either side of the point sees one cost fall with the other rising at a greater rate.



Appendix B Terminology and Concepts

This appendix outlines some of the components of the SDS architecture, associated terminology, and concepts.

Synchronized Database Environment (SDE)

A *Synchronized Database Environment* is the collective term that describes an environment comprising of one or more connected JADE systems that provide or subscribe to a Synchronized Database Service. An SDE consists of a primary database (where updating occurs) and one or more secondary databases, which are transactionally consistent copies of the primary database.

Database Role

In a Synchronized Database Environment, the terms *primary* and *secondary* signify the intended function or role of each copy of the database.

The valid roles are ‘Primary Database Role’ and ‘Secondary Database Role’. For more details, see the following subsections.

Primary Database Role

The database with the primary role is the only database in an SDE where user applications can update persistent objects.

For more details, see the following subsections.

Primary Database Server

The primary database server is the database server node hosting the JADE database with the primary role.

Primary System

The primary system is the JADE system comprising nodes attached to the primary database server, including the primary database server node itself.

User applications executing in the primary system can create and update persistent objects, which reside in the primary database.

Secondary Database Role

Secondary databases are kept synchronized with a primary database by replaying journals automatically shipped from the primary system. User applications cannot update persistent objects (apart from environmental objects) in a secondary database.

A secondary database can assume the role of the primary if the primary database becomes unavailable. For more details, see the following subsections.

Secondary Database Server

A secondary database server is a database server node that hosts a database with a secondary database role.

Secondary System

A secondary system is the JADE system comprising nodes attached to a secondary database server. The secondary system includes a secondary database server node. User applications in a secondary system cannot directly create, update, or delete persistent objects (apart from environmental objects).

Nodes and Processes

A primary system is comprised of primary nodes. A secondary system is comprised of secondary nodes.

Secondary processes execute in a secondary system. In addition, special processes called *replay processes* are created in secondary servers to reflect and represent the original updating processes on the primary database. These replay processes are associated with a single secondary node called a *replay node.*

Inter-Node Communications (INC)

Inter-node communication is the name given to the module that implements the peer-to-peer transport protocol used to transfer journal information and commands between server nodes participating in a Synchronized Database Environment.

Initialization and the Catch-Up Process

When a secondary database server initially connects to a primary database, it will almost certainly be behind in terms of applied transactions and will need to catch up with the primary. This synchronization is achieved by replaying transactions from journals received from the primary. A secondary database server remains in catch-up mode until it has reached synchronization by replaying to the end of the journal prior to the one currently being written.

A secondary database can be taken offline at any time for any purpose; when it reconnects to the primary, the synchronization process will be repeated.

For more details, see the following subsections.

Primary Journal Transfer

On the primary database server, a separate journal transfer worker executes for each connected secondary database. The transfer workers handle shipping of the latest journal to synchronized secondaries and handle shipping of archived journals to secondaries that are catching up.

Secondary Journal Transfer

The journal transfer process is a background thread executing in a secondary server that is responsible for receiving journal blocks from the network and writing these to a ‘mirror copy’ of the journal on disk.

Database Tracker

The database tracker is the process that executes in a secondary database server when tracking is enabled.

The tracker process replays object updates and other audited actions such as persistent cause events, user sign-ons, file compaction, and reorganization operations.

Non-SDS-Managed Journal Replay

It is possible to operate secondary databases in a standalone warm standby mode; in this mode, SDS handles the tracking and replay function but does not automate journal shipping. The goal of this mode is to achieve periodic database synchronization that is completely under user control. At the optimum, synchronization can be within one completed journal.

Under this mode, SDS functions as though there were no communications link between the servers. Users determine the method of journal file transfers (network, tape, other media, and so on) and their timing.

Using the SDS Administration utility or administration API, a site can acknowledge receipt of journal files transferred to the secondary server's journal directory. The acknowledgment signals that the entire journal is present and ready to process.