



Using Physical Replication and Oracle® Database Standard Edition for Disaster Recovery

A Dbvisit White Paper

Contents

Executive Summary	1
Standby Concepts	2
What is a Standby database?	2
Business Continuity Metrics	3
Enabling a Standby Database	3
Full Business Continuity	5
Implementing a Physical Standby Database	6
Installation on standby	7
Creating a backup	7
Transferring backup to standby server	8
Configure standby server using backup	8
Setting up ongoing synchronization	9
Monitor and Test	10
Delivering a Physical Standby Database	11
Dbvisit Standby	14
Components	15
Log Extraction	16
Transportation	18
Log Application	18
Identifying & Responding to a Failure	20
Return on Investment	24
Reduced Costs	24
Increased Returns	25
Risk Management	25
Conclusion	27

Executive Summary

Companies in today's online business environment rely entirely on providing continued access to their data to a multitude of users and external systems. It is no longer acceptable for a failure of some component (hardware or software) to leave data unavailable at any time of the day or night.

Until only a few years ago, systems were typically only accessed by employees during office hours, but this has now all changed. Nowadays we see the proliferation of online systems including web-based self-service applications (B2C and B2B) and integration points (such as those delivered by web services), as well as around-the-clock access by employees and business partners. This continuous demand for system access means that applications, and their databases, simply cannot be offline for anything other than the shortest possible time.

The critical importance of disaster recovery is demonstrated by the following figures:

- 43% of US businesses never reopen again after a disaster and a further 29% close within two years (US Small Business administration)
- 93% of companies that suffer significant data loss are out of business within five years (US Bureau of Labor)

To achieve the required availability, IT departments demand that their database administrators provide reliable and timely replication of databases. The objective of this replication is to ensure that, in the event of a failure of the primary database, an up-to-date copy is available within a very short period of time, and that this copy can then be used to support the user base.

Several options exist to deliver data replication, including tools provided by database management system vendors (e.g. Oracle), tools and scripts developed and maintained in-house by database administrators, and third party tools.

The database vendor solutions, such as Oracle's Data Guard are typically a costly option, particularly when, as is the case with Data Guard, it dictates the use of an Enterprise Edition license for the database management system when this may otherwise be unnecessary.

In-house developed solutions can be unreliable, costly to support and maintain, and present a significant risk in terms of ongoing operation and maintenance.

Dbvisit Standby provides a cost-effective, reliable and proven physical replication solution for Oracle Standard Edition databases. Dbvisit Standby provides a complete tool for replication that removes the need for the development of custom scripts and unreliable, untested code.

This whitepaper explains the concepts, considerations and principles behind replication and provides details of various replication methods available, including physical replication. It then outlines a range of solutions that can be used to deliver physical replication for Oracle databases before explaining in detail the operation of one such solution, Dbvisit Standby.

Standby Concepts

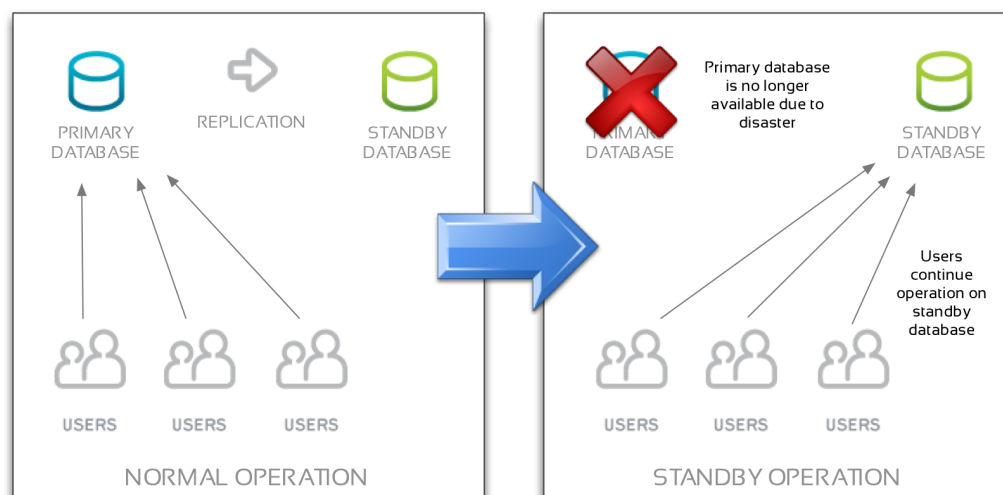
What is a Standby database?

Fundamentally, database replication entails making a duplicate copy of the “live” production database so that, in the event that this is no longer available, the duplicate copy can be used in its place, with minimal impact.

We are therefore concerned with two different database instances, deployed on two sets of infrastructure and most likely at two geographically dispersed locations:

- **Primary database** – Contains production data that must be protected against any kind of loss.
- **Standby database** – A byte-level copy of the production database that can be brought online to become the production database in the event of failure of the primary database.

The main purpose of a standby database is to support users when the primary database suffers a major outage. The following diagram illustrates this process, showing operation in both normal and standby (DR) modes:



Under normal operation, users access the primary database and the database replication process takes their changes and applies them to the standby database, keeping it updated. In the event of a failure of the primary database (and or other components at the primary site that render the primary database unusable), users are redirected to the standby database

Business Continuity Metrics

When considering business continuity and database replication, there are two key metrics used to evaluate options, make decisions and measure service levels:

- **Recovery Time Objective (RTO)** – RTO describes the maximum period of time that should pass before systems are up and running again in the event of a failure. This describes how long it is between failure of the primary site and the standby site being operational in its place.
- **Recovery Point Objective (RPO)** – RPO describes the maximum acceptable data loss (measured in time) in the event of a disaster. This is typically determined by how much data the company can afford to lose without serious impact to the business.

Implementing a business continuity plan with specific RTO and RPO metrics is a trade-off between the impact of system outage and data loss, and the cost of the disaster recovery solution.

Lower RTO and RPO values incur additional cost, as is illustrated in the following table which provides examples of a range of RTO and RPO metrics and the relative cost of each:

Tier	RPO	RTO	Solution Cost
I	No data loss	<30 min	\$\$\$\$\$\$\$\$\$\$
II	< 30 min	< 1 hour	\$\$\$
III	24+ hours	48+ hours	\$\$
IV	7+ days	3+ days	\$

Enabling a Standby Database

There are two distinct approaches to providing a standby database, physical and logical database replication.

A physical standby database maintains a copy of a production database but in a permanent state of recovery. If the production database fails then the standby database can be opened (or activated) and be ready for use very quickly. A physical standby database is a binary copy of the primary database. Changes are applied at the lowest level available within the DBMS, ensuring that the standby database is an *exact* replica of the primary database, including all internal database indexes, pointers and tables. A physical standby database is analogous with using a tool such as rsync to synchronize Word document, with rsync replicating the file at the binary file level.

The advantages and disadvantages of a physical standby database are as follows:

Advantages	Disadvantages
<ul style="list-style-type: none">• Easier to setup and maintain than a logical standby.• Less overhead than a logical standby,• 100% guaranteed to be consistent with the primary database (assuming best practices are followed).• DBA's are more familiar with them since they are more common.	<ul style="list-style-type: none">• It is necessary to have the same operating system and DBMS on both primary and standby servers.• Both the primary and standby servers must have the same amount of disk space.• The standby database can be opened read only, but data is not updated in real time (except with Active Data Guard).• Once the standby database is activated it cannot easily revert back to being a standby database. It is necessary to recreate the primary database and then rebuild the standby database using the new primary database.• The standby database must be licensed in-line with the primary database license metric

A logical standby database is an independent database that is kept in sync by a replication mechanism that applies updates at the logical level (e.g. via SQL statements). This means that while the data within a logical standby database may be the same as that in the primary database, the internal database-level structures will be different. This may have implications for some applications, and for the usage of the standby database in the event of a failure. This is important as the database must be viewed not only as a repository of application data, but also a container with its own management and administrative data. For example, if a password is changed in the source database it is not updated in the target, then when it comes to failover the system will not work because of an old password. It also means that, although internal linkages that support referential integrity may be in place in the standby database, these may be physically different than at the primary site, and as a result, may have an impact on the application (e.g. different automatically created foreign key values).

A logical standby database can be made available in read/write mode at all times, so can be used for purposes other than disaster recovery. While this can be useful, it also introduces the possibility of inconsistencies with the primary database based on unmanaged user changes to the standby database.

A logical standby database is analogous with manually updating a Word document by scanning for changes in the source file and copying them to the right location within the standby file.

The advantages and disadvantages of logical standby databases are as follows:

Advantages	Disadvantages
<ul style="list-style-type: none"> • Primary and standby can be on different platforms. • Primary and standby can be different DBMS versions. • The standby database can serve multiple purposes other than DR, for example for real-time reporting or to support data migration. • Can actually use the target/DR database as a read/write reporting server at any time - it is an open Oracle instance in its own right • Potential to provide zero data loss when it comes to failover time. • No rebuild is required after failover • Can switch back and forward between source and target as much as you like 	<ul style="list-style-type: none"> • The database is not a binary copy, therefore cannot be guaranteed to be consistent. • Data can be applied in a different sequence, resulting in data differences (e.g. data fields maintained by the database). • There are restrictions on the types of data that can be replicated. • It is more complex and time-consuming to administer • Data conflicts can occur, and it is necessary to set up resolution processes. • The DBA needs to understand the data and the application to allow them to resolve conflicts. • Standby databases maintained using logical replication can be opened for update access. • There can be performance issues which require additional hardware to be used at the standby site. Since SQL statements are being applied to the standby database, it can experience load levels similar to the production database.

On balance it is evident that a physical standby database is best suited for use as a disaster recovery solution. Its disadvantages are not significant when its sole purpose is to act as a disaster recovery copy for use in rare circumstances, and where a recovery to normal operation can be expected to take some period of time. A physical database can deliver a shorter RPO through its more efficient replication mechanism and a shorter RTO by enabling a faster switchover to the standby database.

A logical database is not well suited to DR, but can be used to deliver reliable data replication and data distribution, which can serve a range of functions including as a reporting database.

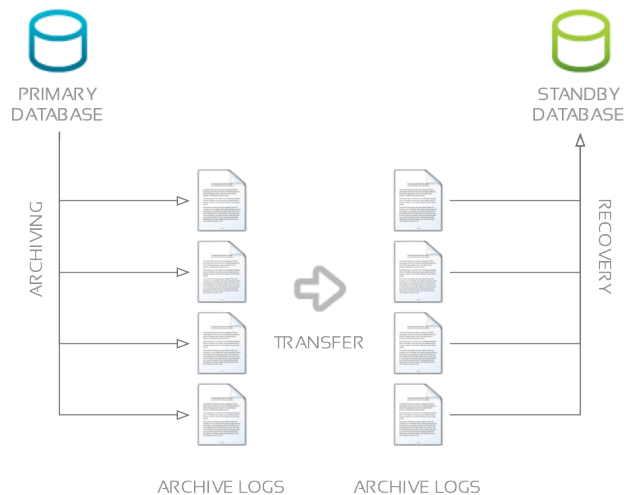
Full Business Continuity

Database replication is only part of the full business continuity plan. While a failure may occur within the database, resulting from an issue with either the hardware (e.g. SAN or database server) or software (some error within the DBMS itself), it may also be more widespread and may impact other components within the solution (e.g. web application, Internet connectivity from the primary site).

For this reason, it is important that any business continuity plan encompasses all of these components at all levels within the application stack (i.e. the entire “service” must be covered by business continuity).

Implementing a Physical Standby Database

The physical standby process works by keeping the standby database in a constant recovery state, and applying archive log files created by the primary database. These archive log files contain the changes made to the primary database, and allow these same changes to be made to the standby database. This is illustrated in the following diagram:



Prior to ongoing synchronization, the standby database must be established. It is then necessary to regularly transfer archive logs from one machine to the other, and apply these to the standby database in order to keep the standby database up to date.

While there are several options for implementing a physical standby database, the following steps are generally required, whether they are performed manually, or automated using available tools:

- 1) Installing Oracle software on the standby/DR server
- 2) Creating a backup
- 3) Transferring backup to standby server
- 4) Configure standby server using backup
- 5) Setting up ongoing synchronization
- 6) Monitor and Test

We will explain this process in more detail using Oracle as an example DBMS and using RMAN for backup and recovery services.

Before starting, it is critical that a number of pre-requisites have been met. The primary database must be in archive log mode and the standby database server must be configured as follows:

- 1) User accounts and groups must be identical to the primary database server.
- 2) The Oracle directory structure must match that in place on the primary database server

Installation on standby

The first step is to install the Oracle database management system on the standby server:

- Install Oracle software on standby server
- Patch Oracle to same version as production
- Create ASM instance (if needed)
- Create service (Windows only)
- Setup listener
- Update the oratab (/etc/oratab or /var/opt/oracle/oratab) file on the standby with the new standby database home location
- The pfile or spfile must be copied from the primary server

Note that, the standby database server should be licensed in the same way as the primary database and that the software installed on the standby must match the software version and edition on the primary.

Creating a backup

With the DBMS software installed on the standby server, the next step is to take a copy of the primary database for use in priming the standby database.

RMAN operates as follows:

- No need to shutdown primary database.
- You do not need to create an RMAN catalog.
- Most efficient.

The easiest way to use RMAN is to create the backup on local disk, however if your database is very large this will require significant free space. If tape is used, a tape or media manager is required as part of your RMAN configuration.

The following provides the RMAN command to create a backup to a local disk and creates the standby controlfile:

```
$ rman
RMAN> connect target /
RMAN> run
{
  change archivelog all crosscheck;
```

```
allocate channel ch1 type disk;

backup as compressed backupset database format '/u02/backup/devdb/db_%U' tag 'BKP1_DB';

sql "alter system archive log current";

backup archivelog all format '/u02/backup/devdb/arc_%U' tag 'BKP1_ARC';

backup current controlfile for standby format '/u02/backup/devdb/standby-ctl-%U';

release channel ch1;

}
```

It is critical that all archives created subsequent to the commencement of the backup are saved, since these are required to create an up-to-date standby database.

Transferring backup to standby server

The next step involves moving the backup, and archive log files, to the standby server, either in one piece or as a series of pieces.

This can be performed over the network using SSH (secure shell), FTP (sftp) or NFS. If the database file (or files) is too large, then media such as tape or an external drive (e.g. USB drive) can be utilized.

Configure standby server using backup

With the database backup created and transferred to the standby server, the next step is to configure the standby database using the backup.

Important – the examples below assume the primary and standby servers are using the same directory structure and layout.

Start the standby database in a nomount state:

```
oracle@dbvlin102[/home/oracle]: . oraenv
ORACLE_SID = [oracle] ? devdb
The Oracle base remains unchanged with value /u01/app/oracle
oracle@dbvlin102[/home/oracle]: sqlplus "/ as sysdba"

SQL> startup nomount
```

The following provides the RMAN command to create the standby database:

```
$rman
connect target /

RMAN> restore standby controlfile from '/u02/backup/devdb/standby-ctl-1qq7t60h_1_1';
RMAN> sql 'alter database mount';
RMAN> restore database from tag 'BKP1_DB';
RMAN> recover database from tag 'BKP1_ARC';
```

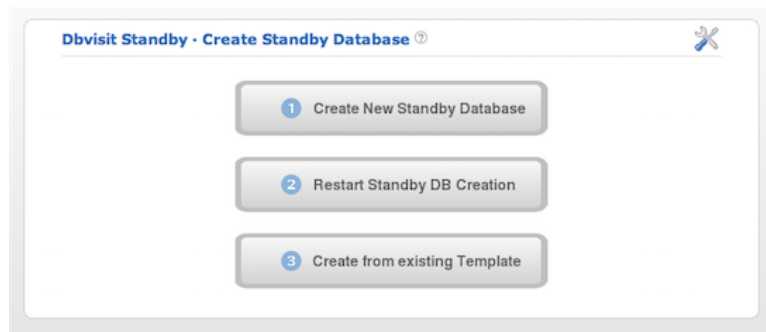
This runs from primary server, creates the standby controlfile and places all datafiles in the same location as primary database. This same command is used to create both ASM and RAC standby databases.

Irrespective of the method used (RMAN or traditional), it is necessary to apply all available archive log files to bring the database up to date with changes since the backup was taken

The following links provide the complete set of detailed steps for reference:

- 1) Manually creating a standby database for Windows:
http://www.dbvisit.com/content/freeTechGuides/Creating_a_standby_database_windows.pdf
- 2) Manually creating a standby database for Linux/Unix:
http://www.dbvisit.com/content/freeTechGuides/Creating_a_standby_database_linux.pdf

Note that Dbvisit Standby takes care of the entire creation of the standby database using a built-in wizard. The following screenshot illustrates the creation of a new standby database from the primary server:



Setting up ongoing synchronization

With the initial instance of the standby database in place, it is necessary to establish an ongoing process to keep the database up to date by applying changes that are made on the primary database.

There are several options available to deliver this:

- 1) Oracle Data Guard – Requires Oracle Enterprise Edition
- 2) Home grown solution (manual log shipping)
- 3) Third Party tools such as Dbvisit Standby, which work with all Oracle Editions and versions and provide a proven and reliable solution to implementing a physical standby database

Each organization will select their preferred approach based on their assessment of the options, and the impact of the pros of cons on each to their particular situation.

In summary, it would generally be agreed that the following attributes are key to a successful synchronization mechanism:

- Delivers high reliability and is a robust and proven solution
- Provides a high degree of resilience
- Supports Oracle RAC, OMF and ASM
- Creates the standby database
- Operates with low noise (i.e. only tells you when things go wrong)
- Is fast to setup, easy to use and provides a short learning curve
- Delivers a low Total Cost of Ownership (TCO)
- Provides peace of mind of technical staff, business stakeholders and management
- Has expert technical support available
- Provides a solution that the DBA's feel comfortable using

See the following section in this whitepaper for a discussion of the three options mentioned above.

Monitor and Test

Once the physical standby database is configured and operational, and is being updated on an ongoing basis, it is critical that the solution is constantly monitored and regularly tested. Best practice says that disaster recovery systems should be tested every quarter.

Key attributes of the implemented solution should include:

- Monitoring to ensure standby database is up to date.
- Exception reporting (only reports exception and errors).
- Daily heartbeat message to say everything is okay.
- Historical reporting on standby statistics.

Delivering a Physical Standby Database

As we outlined when discussing the steps to implement a physical standby database, a critical factor in the success of a standby system is the method for keeping the standby database up to date with changes made to the primary database.

In order to make the standby site useful it is critical that up-to-date and accurate data is maintained within the standby database. To minimize the potential for data loss, software can be used to periodically transfer data changes to the standby site, thus keeping the primary and standby databases synchronized.

Keeping the standby database up to date is critical in achieving the targeted Recovery Point Objective (RPO). In the event of a failure at the primary site and reversion to the standby site, any data not transferred from the primary site's production database to the standby site ahead of the failure will be lost, and will impact the RPO.

There are several options available to deliver physical replication:

- 1) Oracle Data Guard – Requires Oracle Enterprise Edition
- 2) Home grown solution (manual log shipping)
- 3) Third Party tools such as Dbvisit Standby, which work with all Oracle Editions and versions and provide a proven and reliable solution to implementing a physical standby database

Using a tool such as Oracle Data Guard provides certainty and comfort. Data Guard is a feature rich and proven solution that can be configured to provide various levels of data synchronization, each delivering a different balance of resilience and performance. Data Guard does however require Oracle Enterprise Edition, and for Oracle customers with Standard Edition who are otherwise satisfied with the features it offers, this can present a significant and often uneconomic increase in license fees.

Thankfully there are alternatives to Oracle Data Guard, including non-Oracle products developed and supported by third-party vendors, and systems developed and maintained by in house database support teams.

In assessing available options for DR, we need to assess the total cost of ownership of each solution, including implementation, operation and support. We also need to determine the level to which each will deliver an effective and reliable mechanism for:

- 1) Synchronization between the primary and standby sites
- 2) Providing a failover process to move operations from a failed primary site to the standby site in a timeframe that meets the agreed service levels
- 3) Restoring the full dual-site state once the primary site is operational again

While a system developed in-house may initially seem like a cost-effective solution, this may not be the case overall. The home grown solution seems pretty simple on the surface, starting by transferring the archive logs to standby server and then applying these once they arrive:

```
sqlplus /nolog
SQL> connect / as sysdba ;
SQL> startup nomount
SQL> alter database mount standby database ;
SQL> recover standby database ;
```

Beyond this simple command however, lies a myriad of considerations and risks related to the operation, maintenance, reliability and support of the solution:

- Is it robust/secure enough? Can it recover from all outages and glitches? Does it have a good locking and transport mechanism?
- Does the transportation of data between primary and standby sites secure the critical business data?
- Has it been tested under all scenarios? Has it been proven to work in a disaster recovery situation? Can the primary site be reinstated after the outage?
- Does it have solid notifications?
- Does it cover all the DBMS errors and exceptions?
- Will it support multiple databases?
- Does it scale as your data volumes increase?
- What will happen when you change, upgrade or patch the DBMS?
- What happens when you rebuild or refresh the standby database?
- Does it support DBMS feature such as Oracle RAC, OMF and ASM?
- Does it include support and documentation?
- Will other DBA's be comfortable using and maintaining it if the current DBA leaves?

Delivering on all of these critical features can add significant overhead to the total cost of the home-grown solution.

Once developed these systems need to be supported, placing a critical dependency on the staff member or members who originally developed them. These systems also need to be maintained and tested with each change to the underlying environments, including updated operating systems and database management systems, as well as infrastructural changes.

It is also critical to assess the reliability of the solution in the areas outlined above. Without extensive testing of exception cases it is difficult to be confident that the home grown solution will operate as required. This level of testing can be difficult to perform in a customer environment with minimal test environments and infrastructure. The sole purpose of these data recovery solutions is to provide certainty in the event of a site failure, so if this cannot be assured then the approach needs to be questioned.

The risks of an in-house solution can be summarized as follows:

- 1) Development cost unknown – may or may not be cheaper to build than licenses for a third party product.
- 2) Reliability of maintaining synchronization, failover management and recovery is all unproven
- 3) The need to maintain staff skills in order to provide the ongoing support, maintenance, enhancement and operation of the solution

The option of third party tools provides an ideal balance between the in-house development and Data Guard approaches, delivering a proven, reliable and supported product in a cost effective manner.

The question then becomes selecting the right third-party tool. Dbvisit Standby is a proven, reliable and stable product used by customers in over 90 countries round the globe. It is used here as an example of a cost-effective third-party tool for providing physical data replication for Oracle Standard Edition.

The following section explains how Dbvisit Standby operates to deliver physical database replication.

Dbvisit Standby

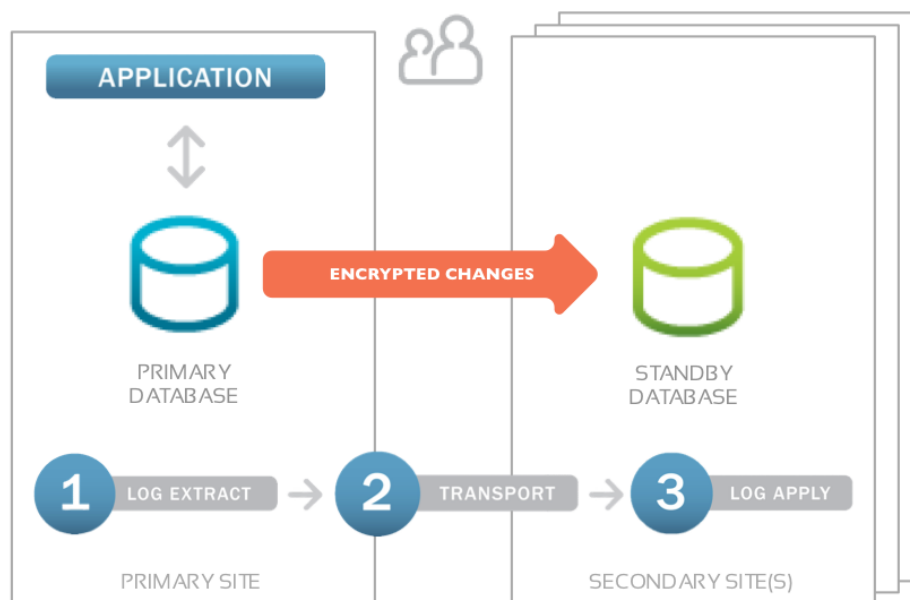
Dbvisit Standby provides a proven, reliable and cost effective physical replication solution for Oracle Standard Edition databases that can be used to deliver a disaster recovery solution.

A Dbvisit Standby configuration consists of a primary (production) database and one or more standby (backup) databases. The primary and standby databases connect using a secure mechanism (SSH) over TCP/IP. Databases can exist anywhere provided they can communicate with one another via TCP/IP.

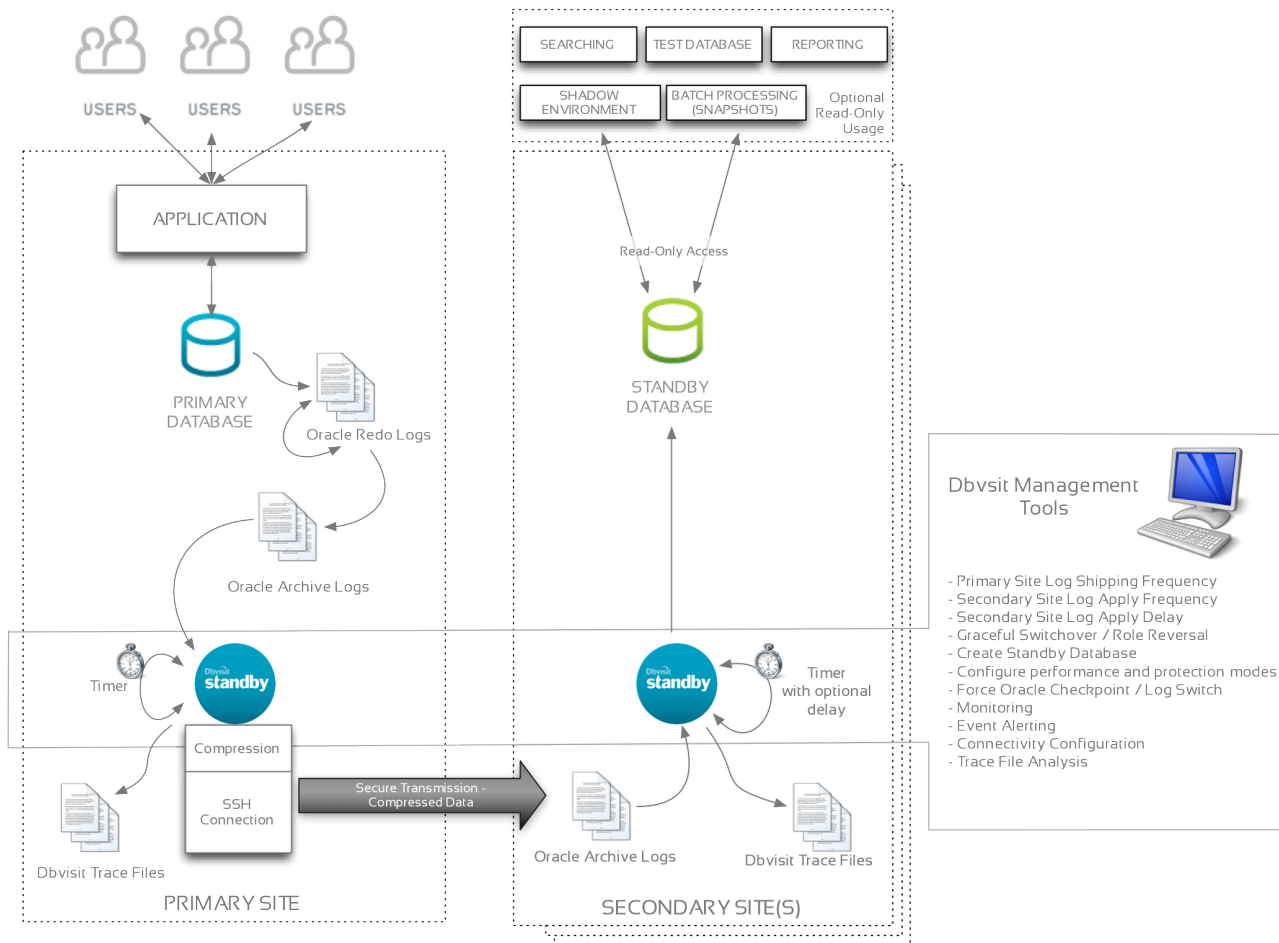
Once the standby database is available, Dbvisit Standby is then scheduled to synchronize the databases by transmitting the archive logs from the primary database and applying them to all of the standby databases.

The following diagram provides an overview of the data replication process performed by Dbvisit Standby, highlighting the three key steps in the replication process:

- 1) The extraction of logs on the primary site
- 2) The transportation of logs to the secondary site
- 3) The application of logs at the secondary site



The following diagram illustrates the architecture of the Dbvisit Standby solution, including components and external environments:



Components

Dbvisit Standby consists of the following components.

Dbvisit software. The Dbvisit software includes the following executables:

- **dbvisit** - The main Dbvisit executable.
- **dbv_orastartstop** - Executable to start, stop, failover and switchover the databases.
- **dbvisit_setup** - Menu driven wizard to configure Dbvisit and to create the standby database.
- **dbv_functions** - Executable to provide extra tools and functionality.

Dbvnet. This component provides the network communication between the primary and standby servers. If using Unix based systems, SSH can also be used.

Dbvserver. This is Web server that provides the web interface to Dbvisit. It consists of the following executable:

- `dbvserverd` - The main Dbvserver executable for the Dbvisit Web server that provides the Dbvisit Standby graphical user interface.

Dbvisit Database Configuration (DDC) File. This file is similar to the `init.ora` parameter file. It is generated during setup for each database and contains the Dbvisit settings for the specific database. The DDC file should only be edited on the primary server, and can be edited with any text editor or by running `dbvisit_setup`. In addition, much of the configuration can be performed using the Dbvisit Standby graphical user interface.

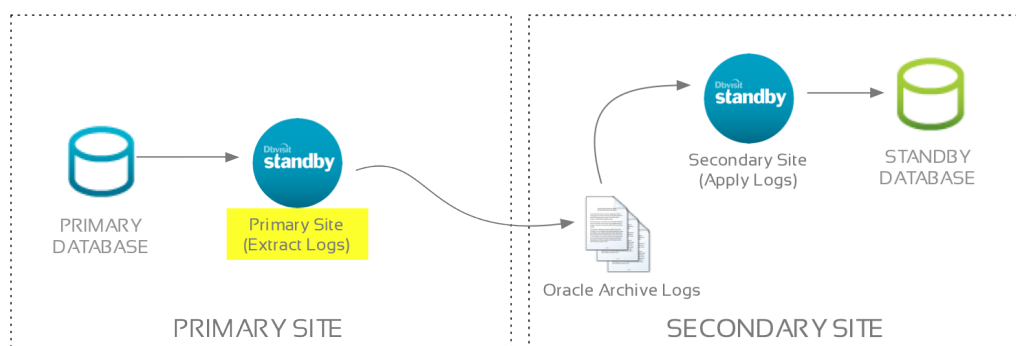
Dbvisit Database Repository (DDR). This schema holds the Dbvisit tables within the database and is created for each database during setup.

Dbvisit Trace Files. Dbvisit Standby generates a trace file each time it executes, containing information about Dbvisit processing and its environment. When an alert or error notification is sent by Dbvisit, the trace file will be attached to the email so that this can be forwarded to Dbvisit support if needed, allowing them to quickly diagnose issues.

Log Extraction

Key to the replication of data is the ability to extract updates from the primary database and make these available for delivery to the standby database (or databases). Dbvisit Standby is built on top of the tried and proven Oracle logging mechanism, and periodically picks up the Oracle archive logs and prepares these for transmission to the standby databases.

The following diagram illustrates the extraction of archive logs from the primary site, containing changes made to the production database:

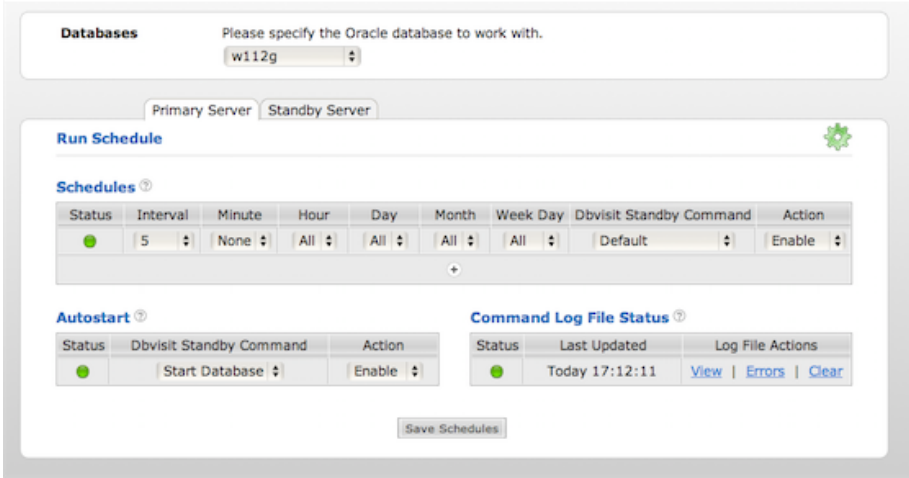


Dbvisit Standby is configured to perform a transmission of logs at a given frequency. The shortest frequency available is 60 seconds, with settings typically falling between 10 and 15 minutes.

Organizations set the extraction frequency based on their Recovery Point Objective (RPO), which describes the acceptable amount of data loss measured in time. The RPO is expressed in time and is generally a definition of what an organization determines to be an "acceptable loss" in a disaster situation.

While more frequent transmissions keep the standby database more up-to-date (and support a lower RPO), they come at the cost of added load on the production and standby servers, as well as on the communication network.

The following screenshot illustrates the maintenance of the transfer schedule within Dbvisit Standby. Note that scheduling can also be managed via cron (under Unix/Linux) or the Windows Task Scheduler:



The screenshot shows the Dbvisit Standby configuration interface. At the top, there's a 'Databases' section with a dropdown menu set to 'w112g'. Below this, there are tabs for 'Primary Server' and 'Standby Server'. The 'Run Schedule' section is active, showing a table of schedules. The table has columns for Status, Interval, Minute, Hour, Day, Month, Week Day, Dbvisit Standby Command, and Action. A single schedule is listed with a green status icon, an interval of 5 minutes, and the command 'Start Database'. Below the table, there's an 'Autostart' section with a green status icon and the command 'Start Database'. To the right, there's a 'Command Log File Status' section showing the last updated time as 'Today 17:12:11' and links for 'View', 'Errors', and 'Clear'. A 'Save Schedules' button is at the bottom.

Status	Interval	Minute	Hour	Day	Month	Week Day	Dbvisit Standby Command	Action
●	5	None	All	All	All	All	Default	Enable

Status	Dbvisit Standby Command	Action
●	Start Database	Enable

Status	Last Updated	Log File Actions
●	Today 17:12:11	View Errors Clear

Save Schedules

Database Update Management

Dbvisit Standby allows the administrator to configure the management of Oracle buffers and logs to aid in the synchronization and to ensure recent changes are included in each transfer to the standby server. Standby can be configured to:

- 1) Perform an Oracle Checkpoint to force the database to flush its buffers to disk
- 2) Perform an Oracle Log Switch to force the creation of up-to-date archive logs from the existing redo logs

Using the LOGSWITCH setting, Dbvisit Standby can be configured to either:

- (i) Perform these actions every time it runs an extract, or
- (ii) Only when necessary (i.e. when there are no existing extract logs available to process), or
- (iii) Leave it entirely to the Oracle database itself.

Transportation

The primary (production) server initiates the transfer of extracted logs at the same frequency as the extraction process. Dbvisit Standby uses a secure transport channel to send the logs to the standby server where they are available to be applied by a separate and independent process. Logs can be compressed prior to being transferred, thus reducing the bandwidth required, and reducing transfer times. This compression does require CPU cycles to perform, so compression can be disabled if preferred.

Secure Communication

Dbvisit Standby is making use of Dbvnet or SSH (if Unix is used) for secure communication between the primary and standby servers. By default 128bit encryption is used and if required 256bit encryption can be enabled. By default port 7890 (dbvnet) and 22 (ssh) will be used. Note that, if preferred, custom ports can be configured instead.

Compression

Dbvisit Standby uses world-leading data compression algorithms to allow synchronization to take place over standard ADSL lines if required. In our experience, these algorithms typically deliver compression ratios of between 70% and 80%. For example, a 2Mbps line will support transfers of up to 640GB per month and a 10Mbps link up to 3.2TB per month. Higher speed connections will support even larger transfers, with a 100Mbps link supporting up to 32TB per month (based on uncompressed Oracle Archive logs).

Communication Resilience

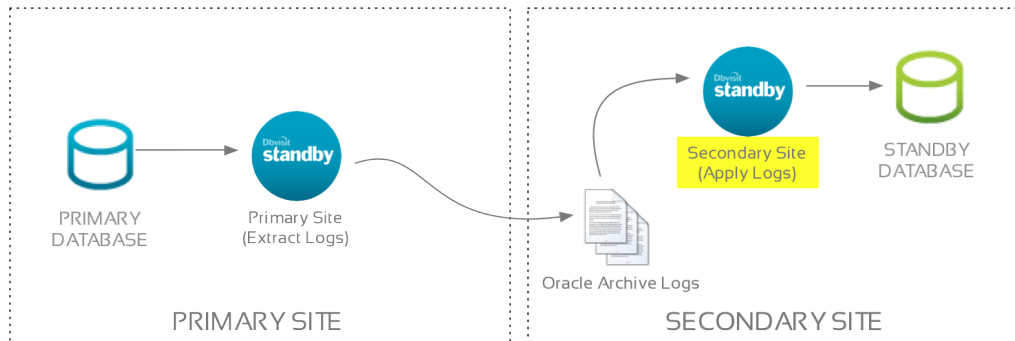
In the situation where communication with the standby server is lost for a period of time, the primary server continues to accumulate logs ready for transmission. Dbvisit Standby monitors the network connection with the standby server and transmits these logs to the standby server once communication is re-established.

Log Application

Dbvisit Standby running on the standby server is configured to periodically check for new archive logs delivered by the primary server. The standby server's process runs completely independently of the primary server's process, allowing each to operate in the absence of the other. This allows the servers to identify situations where the other has failed and provide the DBA with alerts.

At a defined frequency the standby server picks up and applies to the standby database all of the archive logs received since the last time it ran. While they would typically be in step, the primary and secondary processes can be configured with different frequencies (see Delaying Updates below).

The following diagram illustrates the application of the archive logs to the standby database:



Although independent and disconnected, Standby provides the option to initiate the application of the logs to the standby database at the conclusion of the log transfer process. This is controlled using the `INITIATE_DBVISIT_ON_STANDBY` parameter.

Standby supports the ability for a single primary site to push updates to multiple standby sites, allowing organizations to maintain multiple synchronized standby databases. This opens the opportunity to utilize secondary sites not only for DR purposes, but also for other read-only activities such as reporting and searching.

Delaying Updates

Dbvisit Standby allows the administrator to build a delay into applying the updates to the standby database.

Since RPO is only dependent on availability of the archive logs on the standby site, not on actual application of the logs to the standby database, this technique allows an organization to reduce the RPO whilst maintaining the option to use the standby database in the event of data corruption.

Recent backup data from the production database is available on the standby server should a disaster event occur, thereby maintaining RPO. Additionally, the application of data is delayed so the standby database can be used in the event of a data corruption in the primary database, assuming the corruption is identified within the timeframe of the delay.

The delay is managed within Standby using the `APPLY_DELAY_LAG_MINUTES` parameter. For example the administrator can set a synchronization frequency on the primary server of one minute, and a delay of 20 minutes on standby server. This achieves an RPO of one minute, but gives the organization 20 minutes to identify data corruption in the database and still make use of the standby database to recover.

Furthermore, different delays can be assigned to different secondary sites, allowing the administrator to tailor the timeliness of the data in each standby database to the specific role of the database at that site. For example, at a site intended for disaster recovery a delay of 20 minutes may be used, whereas at a site used for real-time searches and/or reporting, there may be no delay at all.

Alerting

When the application process runs on the standby server, it can alert the administrator if no updates have been received from the primary server within a defined timeframe. The `MAX_TIMES_TRIED` parameter is used to set to the number of empty update cycles after which an alert will be raised (e.g. a two minute frequency and `MAX_TIMES_TRIED` set to five, will have the effect of raising an alert when no updates have been received for ten minutes).

Identifying & Responding to a Failure

Key functionality of any disaster recovery solution is the operation required to switch processing from the primary database to the standby database in the event of a failure. In order to ensure full disaster recovery and continuity of service to users, as defined by the agreed service levels for the application, all components must be subject to redundancy. In this paper however, we deal solely with the database.

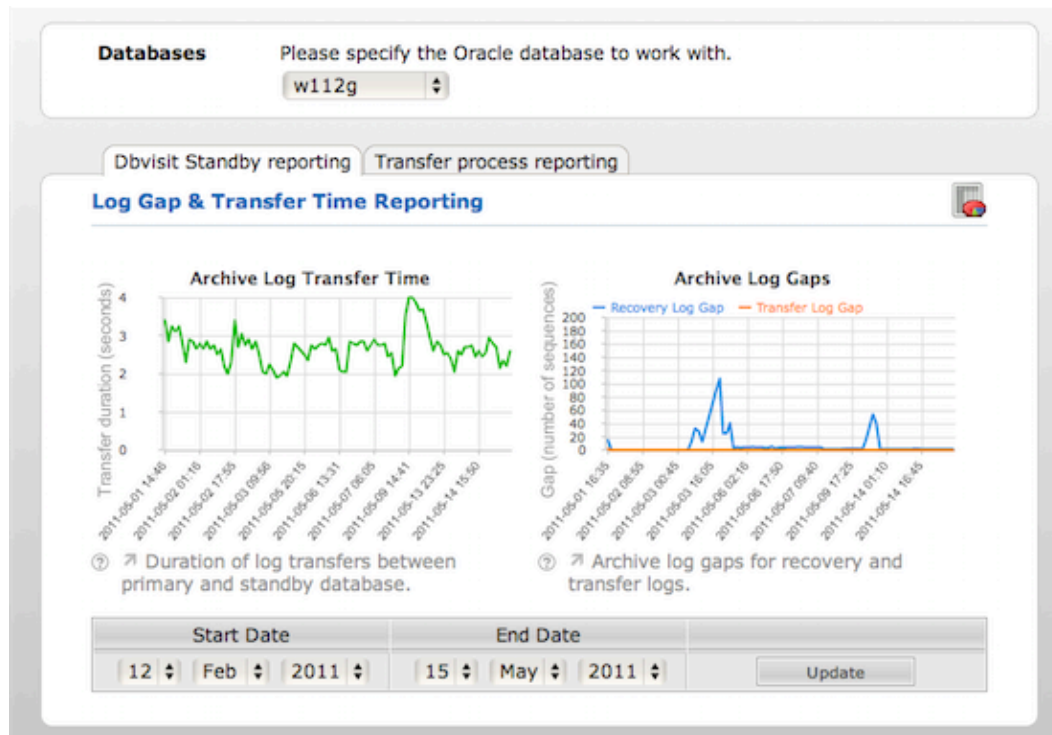
The end-to-end process from normal operation and disaster event to recovery to normal operations is illustrated in the following diagram:



Monitoring

Key in the provision of a disaster recovery solution is the ability to identify that a disaster has taken place. This identification can be as simple as awareness of a major physical event that renders the primary site inoperable, but typically requires more detailed monitoring and alerting to identify exceptions.

Dbvisit Standby provides monitoring of its replication operations, and the availability of the primary and secondary sites to participate in the replication. The following screenshots illustrate sample reporting from Standby:



This monitoring can then send alerts when it identifies exceptions outside of the configured acceptable bounds, providing advance warning to operators of a failure or disaster. Typically this alerting would be fed into an organization-wide network monitoring solution such as IBM Tivoli or Zabbix. These systems are able to collect observations from multiple sources and analyze them to report and alert operators based on pre-configured rules.

Having observed a disaster event that renders the primary site inoperable, the failover process must be initiated.

Failover

Dbvisit Standby provides alerting to assist a monitoring system in identifying a failure of the primary database. This includes failure by the standby server to receive updates from the primary server for a configured period of time. In addition to Standby, other monitoring and alerting within the application stack would also be used to identify failure of the primary server.

With a disaster identified, it is critical that an organization has the systems and processes in place to respond, and regain service in line with its agreed service levels. These will be measured in terms of the Recovery Time Objective (RTO) and the Recovery Point Objective (RPO). To meet the RTO the standby site must be operational as the new primary site within the agreed timeframe. In order to meet the RPO,

the restarted systems must be in service at the standby site with the agreed levels of data loss (measured in time).

Dbvisit Standby aids in the achievement of both RTO and RPO.

- Standby helps achieve RTO through monitoring, alerting and fast failover. Although failover can be automated through the implementation of a simple custom script, Dbvisit recommends that it is left as a manual process to retain a level of control over when it is invoked. This is important as there can be many non-critical events that could force a switchover unnecessarily.
- RPO is delivered through configuration of appropriate replication frequencies, and the provision of tools to allow the rapid application of pending logs on the standby server (particularly in the event of a configured delay in the log file application).

Once failure of the primary site is identified, the following command is used on the standby server to force it to take over as primary database. Note that including the “Yes” parameter overrides the confirmation prompt and immediately triggers activation of the standby database.

```
dbv_orasStartStop activate oracle_database [Yes]
```

As an alternative, Standby’s GUI can also be used to activate the standby database.

Once triggered, this activation cannot be reversed, and a standby database will have to be built before the replication service can be restarted.

Recovery using Graceful Switchover

Once disaster recovery has been invoked and the replicated (standby) systems are operable and stable, the task turns to one of recovery to the normal state where all environments (primary and standby) are again available. In the event of a true disaster striking the primary site, this will typically involve considerable effort and take some significant period of time.

Dbvisit Standby aids in the recovery steps through tools to enable the recreation of new standby databases from the operational production database. This process is highly automated and can be used to quickly establish standby databases at one or more sites.

The steps to restore normal operation typically involve the following:

- 1) Recreate the environment at the original (and future) primary site, leaving the production system running at the standby site.
- 2) Establish the future primary site as a standby site using the production database currently running at the standby site. This can be achieved by using Dbvisit Standby’s built in CSD utility, or by following the manual steps outlined above.
- 3) Establish the ongoing synchronization of the standby database from the production database.
- 4) Confirm readiness of new primary site to accept the role of production site

- 5) Use Dbvisit Standby's Graceful Switchover to switch the roles of the databases, so that the primary site's database switches from standby to production, and the database at the standby site switches from production to standby.

Graceful switchover is used to switch roles, transitioning the production database (operating at the standby site during DR operation) to a standby database and the standby database (newly created at the primary site) to the production database. There is no loss of data during the transition and the standby database does not have to be rebuilt.

The graceful switchover is initiated using the following command. This command must be given on both the primary and the standby servers, and the unique key is a user-defined value that must be identical on both the primary and standby servers

```
dbv_orasStartStop switchover oracle_database [unique_key]
```

As an alternative, Standby's GUI can also be used to perform the graceful switchover.

Supporting Planned Maintenance

Graceful switchover may be used to enable planned maintenance of servers, operating systems, database management systems and applications etc. The administrator can implement changes on the standby server and switch roles so the recently updated server becomes the primary database while the original primary is updated. Once the update of the original primary is completed, the switchover is repeated in reverse, restoring the original roles of the two servers. In this case the only downtime is the time required to perform the switchovers.

Return on Investment

The primary objective of a disaster recovery solution is to ensure continued service in the event of a disaster. The investment, or cost, for the DR solution is relatively easy to calculate as it is based on measurable items such as hardware and systems, as well as the ongoing support costs for the standby sites. The return, or value, however is largely measured in terms of costs that will be saved during an unplanned outage, including direct staff costs to recover from the failure and lost company revenue.

Return on Investment (ROI) can be improved through a combination of reduced costs and increased returns.

Reduced Costs

In terms of the investment, or cost, of a disaster recovery solution it is important to consider the total cost of ownership, including the establishment, operation and execution in a disaster:

- 1) Hardware costs (e.g. servers, network infrastructure)
- 2) System software costs (e.g. operating systems)
- 3) License costs (e.g. replication software, database management system)
- 4) Ongoing operational costs during normal operation (e.g. staff and equipment)
- 5) Cost of system downtime during failover to DR site (based on RTO)
- 6) Cost of lost data in the event of a disaster (based on RPO)
- 7) Cost of recovery from a disaster

A proven solution such as Dbvisit Standby reduces costs in a number of these areas:

- 1) Licensing costs for the Dbvisit Standby product are lower than alternative products.
- 2) Dbvisit Standby supports Standard Edition, unlike products such as Oracle Data Guard that require an upgrade to Enterprise Edition, saving Oracle license fees.
- 3) Dbvisit Standby is a proven product that reduces the operational costs compared with a home built system through reliability, automated operation and regular updates to support new Oracle releases.
- 4) Dbvisit Standby can assist in delivering a recovery point objective of as little as 60 seconds, thereby minimizing the cost of lost data
- 5) Dbvisit Standby provides tools to reduce the effort and increase the reliability of recovering from a disaster and reinstating the normal operational state. These tools reduce the cost of recovery from a disaster.

Increased Returns

A solution such as Dbvisit Standby can increase the returns available from a DR solution by allowing secondary sites to serve additional roles that add value during normal operation (for example, activities such as test and shadow environments, reporting systems and searching). These additional environments, kept continuously synchronized, can add significant value to an organization, including:

- 1) Improved solution quality through production-scale test environments
- 2) Improved solution scalability through segmentation of non-core activities (such as reporting) to standby servers
- 3) Improved production performance through the offloading of read-only processing to standby databases
- 4) Extended life of production hardware through reduced load on production servers

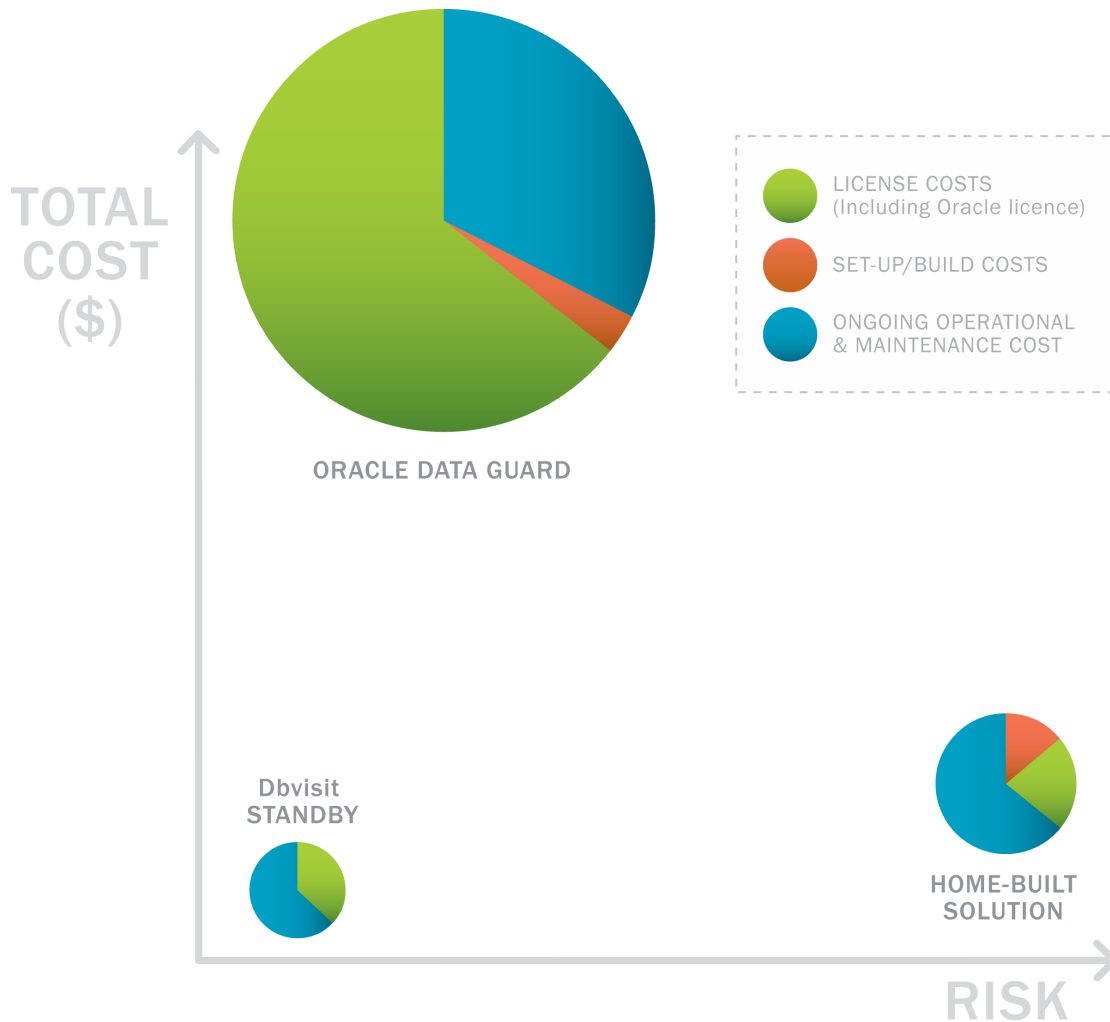
Risk Management

By its nature, disaster recovery is all about managing risk. Reducing risk typically increases cost, leading organizations to balance the cost of their DR solution against the benefit that it provides.

Some solutions may come at a lower cost, but if these are unproven then their value must be questioned. For this reason it is critical when evaluating disaster recovery options to consider the risk profile for each, both in absolute terms and relative to the cost. For example, no matter how “cheap” it may appear, a high risk DR solution has little value if an organization cannot be certain it will work in a disaster.

As described above, Dbvisit Standby delivers a proven and low risk solution at a relatively low total cost, including licensing, implementation and ongoing operation.

The following diagram demonstrates this concept, with axis for cost and risk, and the size of the individual circles illustrating relative cost of the solutions¹.



¹ Cost estimates include Oracle RAC database licensing, setup and training effort and annual ongoing operation and maintenance. Estimates are based on dual-CPU primary server and include Oracle Database licenses for primary & secondary servers. All pricing is US Dollars and assumes internal staff rate of \$75ph.

Conclusion

This paper started by discussing the evolving requirement within businesses for around-the-clock access to systems and data, and the resulting need for a business continuity plan. We introduced metrics such as RTO and RPO used to design and measure DR solutions.

Disaster Recovery requires a remote site with automated and regular synchronization from the production database to the standby databases and secondary sites. We discussed the physical and logical methods for database replication and presented the advantages and disadvantages of each, particularly in the context of delivering a DR solution.

Physical replication creates a 100% exact copy of the production database at the standby site, with a lower overhead than logical replication, and without the need for the DBA to understand the database structures involved.

While logical replication does provide some advantages, such as the ability to support different operating systems and database management systems at the different sites, these are largely irrelevant when delivering DR, and are offset by disadvantages such as performance implications and potential data inconsistencies resulting from the logical replication process. Based on this, we demonstrated that physical database replication is the best option for delivering DR.

Having demonstrated the advantages of physical replication, we next discussed the steps involved in establishing and maintaining physical database replication, and presented three options including Oracle Data Guard, a home built solution, or a third party product. We demonstrated the high total cost for Oracle Data Guard, including the requirement to deploy Oracle Enterprise Edition. We discussed how home built systems are unproven, risky, dependent on the continuity of in-house resources, and are likely to incur higher than expected overall costs, including development, operation and maintenance.

We discussed the approach provided by third party tools, and outlined in detail the way in which Dbvisit Standby delivers reliable and fully functional physical replication, including a series of best practice guidelines for physical replication on Oracle in general and specifically around Dbvisit Standby.

Finally we discussed return on investment, and looked at the benefits and costs of the three options, showing how a cost effective DR implementation is achievable using Dbvisit Standby with Oracle Standard Edition. We showed how this solution delivers opportunities for reduced cost and for increased value, thereby significantly improving the return on investment that can be demonstrated for a DR solution.



Dbvisit Software Limited

www.dbvisit.com

info@dbvisit.com

USA: 1-800-933-8007

Rest of the world: +64 9 950 3301

Fax: +64 9 950 3302

Dbvisit Software Limited
P.O. Box 48180, Blockhouse Bay
Auckland 0604, New Zealand