Oracle9*i*

Data Guard Concepts and Administration

Release 2 (9.2)

March 2002

Part No. A96653-01

ORACLE

Oracle9*i* Data Guard Concepts and Administration, Release 2 (9.2)

Part No. A96653-01

Contents

2 Configurations and Considerations

3 Creating a Physical Standby Database

4 Creating a Logical Standby Database

5 Log Transport Services

6 Log Apply Services

7 Role Management Services

8 Managing a Physical Standby Database

9 Managing a Logical Standby Database

10 Data Guard Scenarios

Part II Reference

11 Initialization Parameters

12 LOG_ARCHIVE_DEST_n Parameter Attributes

13 SQL Statements

14 Views

List of Examples

List of Figures

List of Tables

Send Us Your Comments

Oracle9*i* Data Guard Concepts and Administration, Release 2 (9.2)

Part No. A96653-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: nedc-doc_us@oracle.com
- FAX: 603-897-3825 Attn: Oracle9*i* Data Guard Documentation
- Postal service:
  Oracle Corporation
  Oracle9*i* Data Guard Documentation
  One Oracle Drive
  Nashua, NH 03062-2804
  U.S.A.

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

A standby databases is the most effective disaster recovery solution for an Oracle database. A standby database can also be used to remedy problems caused by user errors, data corruption, and other operational difficulties.

This guide describes Oracle9*i* Data Guard concepts and helps you configure and implement standby databases. A standby database can be used to run your production system if your primary database becomes unusable.

This preface contains these topics:

Audience

Oracle9i Data Guard Concepts and Administration is intended for database administrators (DBAs) who administer the backup, restore, and recovery operations of an Oracle database system.

To use this document, you should be familiar with relational database concepts and basic backup and recovery administration. You should also be familiar with the operating system environment under which you are running Oracle.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle Corporation is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Organization

This document contains:

Part I, "Concepts and Administration"

Chapter 1, "Oracle9i Data Guard Concepts"

This chapter offers a general overview of the Oracle9*i* Data Guard architecture.

Chapter 2, "Configurations and Considerations"

This chapter explains how to assess and configure a Data Guard environment.

Chapter 3, "Creating a Physical Standby Database"

This chapter explains how to create a physical standby database and start applying redo logs to it.

Chapter 4, "Creating a Logical Standby Database"

This chapter explains how to create a logical standby database and start applying redo logs to it.

Chapter 5, "Log Transport Services"

This chapter introduces log transport services. It provides procedures and guidelines for configuring log transport services on a primary and standby database. It also provides guidelines and procedures for configuring data protection modes, network tuning for log transport services, and log transport services monitoring.

Chapter 6, "Log Apply Services"

This chapter introduces log apply services. It provides guidelines for managing a physical standby database in managed recovery mode and read-only mode. It also provides guidelines for managing the log apply services for a logical standby database.

Chapter 7, "Role Management Services"

This chapter introduces role management services. It provides information on database role transitions.

Chapter 8, "Managing a Physical Standby Database"

This chapter describes how to manage a physical standby database. It provides information on monitoring and responding to events that affect the database role.

Chapter 9, "Managing a Logical Standby Database"

This chapter describes how to manage a logical standby database. It provides information on applying redo logs, system tuning, and tablespace management.

Chapter 10, "Data Guard Scenarios"

This chapter describes common database scenarios such as creating, recovering, failing over, switching over, configuring, and backing up standby and primary databases.

Part II, "Reference"

Chapter 11, "Initialization Parameters"

This reference chapter describes initialization parameters for each Oracle instance, including the primary database and each standby database in the Data Guard environment.

Chapter 12, "LOG_ARCHIVE_DEST_n Parameter Attributes"

This reference chapter provides syntax and examples for the attributes of the LOG_ARCHIVE_DEST_*n* initialization parameter.

Chapter 13, "SQL Statements"

This reference chapter provides SQL statements that are useful for performing operations on a standby database.

Chapter 14, "Views"

This reference chapter lists views that contain useful information for monitoring the Data Guard environment. It summarizes the columns contained in each view and provides a description for each column.

Part III, "Appendixes and Glossary"

Appendix A, "Troubleshooting the Standby Database"

This appendix discusses troubleshooting for the standby database.

Appendix B, "Manual Recovery"

This appendix describes managing a physical standby database in manual recovery mode. It provides instructions for manually resolving archive gaps and renaming standby files not captured by conversion parameters.

Appendix C, "Log Writer Asynchronous Network I/O"

This appendix describes how to achieve asynchronous network I/O when using the log writer process (LGWR) to transmit primary database online redo log modifications to standby databases.

Appendix D, "Standby Database Real Application Clusters Support"

This appendix describes the primary and standby database configurations in a Real Application Clusters environment.

Appendix E, "Cascading Standby Databases"

This appendix describes how to implement cascading standby databases.

Glossary

Related Documentation

Every reader of *Oracle9i Data Guard Concepts and Administration* should have read:

- The beginning of the *Oracle9i Database Concepts* manual, which provides an overview of the concepts and terminology related to the Oracle database server and a foundation for the more detailed information in this guide. The rest of the *Oracle9i Database Concepts* manual explains the Oracle architecture and features in detail.
- The chapters in the *Oracle9i Database Administrator's Guide* that deal with managing the control file, online redo logs, and archived redo logs.

You will often need to refer to the following guides:

- *Oracle9i Data Guard Broker*
- *Oracle9i SQL Reference*
- *Oracle9i Database Reference*
- *Oracle9i User-Managed Backup and Recovery Guide*
- *Oracle9i Recovery Manager User's Guide*
- *Oracle9i Net Services Administrator's Guide*
- *SQL*Plus User's Guide and Reference*

In addition, refer to *Oracle9i Database Concepts* for information about other Oracle products and features that provide disaster recovery and high data availability solutions.

In North America, printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

Customers in Europe, the Middle East, and Africa (EMEA) can purchase documentation from

`http://www.oraclebookshop.com/`

Other customers can contact their Oracle representative to purchase printed documentation.

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://otn.oracle.com/admin/account/membership.html`

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://otn.oracle.com/docs/index.htm`

To access the database documentation search engine directly, please visit

`http://tahiti.oracle.com`

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- Conventions in Text
- Conventions in Code Examples

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table.**
Italics	Italic typeface indicates book titles or emphasis.	*Oracle9i Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.

Convention	Meaning	Example
UPPERCASE monospace (fixed-width font)	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a NUMBER column. You can back up the database by using the BACKUP command. Query the TABLE_NAME column in the USER_TABLES data dictionary view. Use the DBMS_STATS.GENERATE_STATS procedure.
lowercase monospace (fixed-width font)	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter sqlplus to open SQL*Plus. The password is specified in the orapwd file. Back up the datafiles and control files in the /disk/oracle/dbs directory. The department_id, department_name, and location_id columns are in the hr.departments table. Set the QUERY_REWRITE_ENABLED initialization parameter to true. Connect as oe user. The JRepUtil class implements these methods.
lowercase monospace (fixed-width font) italic	Lowercase monospace italic font represents placeholders or variables.	You can specify the *parallel_clause*. Run U*old_release*.SQL where *old_release* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (digits [ , precision ])`
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`
|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	`{ENABLE | DISABLE}` `[COMPRESS | NOCOMPRESS]`
...	Horizontal ellipsis points indicate either:	
	■ That we have omitted parts of the code that are not directly related to the example	`CREATE TABLE ... AS subquery;`
	■ That you can repeat a portion of the code	`SELECT col1, col2, ... , coln FROM employees;`

. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	`SQL> SELECT NAME FROM V$DATAFILE;`
		`NAME`
		`----------`
		`/fs1/dbs/tbs_01/dbf`
		`/fs1/dbs/tbs_02/dbf`
		`.`
		`.`
		`.`
		`/fs1/dbs/tbs_09/dbf`
		`9 rows selected.`

Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	`acctbal NUMBER(11,2);` `acct CONSTANT NUMBER(4) := 3;`
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/system_password` `DB_NAME = database_name`

UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

Convention	Meaning	Example

`lowercase`	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files.
	
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		`SELECT last_name, employee_id FROM employees;`
		
		`sqlplus hr/hr`
		
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

What's New in Data Guard?

This section describes the new Data Guard features in Oracle9*i* release 2 (9.2) and provides pointers to additional information. New features information from previous releases is also retained to help those users upgrading to the current release.

The following sections describe the new features in Data Guard:

Oracle9*i* Release 2 (9.2) New Features in Data Guard

The features and enhancements described in this section were added to Data Guard in Oracle9*i* release 2 (9.2).

- **Logical standby database**

  Until now, there has been only the physical standby database implementation, in which the standby database can be in either recovery mode or in read-only mode. A physical standby database is physically equivalent to the primary database, and, while the database is applying logs it cannot be opened for reporting and vice versa. A logical standby database has the same logical schema as the primary database but may have different physical objects, such as additional indexes. With logical standby databases, you can have the database available for reporting and applying the logs to the standby log at the same time.

- **Database protection modes**

The database administrator (DBA) can place the database into one of the following modes:

- Maximum protection
- Maximum availability
- Maximum performance

These modes replace the guaranteed, instant, rapid, and delayed modes of data protection available in Oracle9*i* release 1 (9.0.1).

See Also: Chapter 5, "Log Transport Services", Chapter 7, "Role Management Services", and Chapter 13, "SQL Statements"

- **Cascading standby databases**

  A cascading standby is a standby database that receives its redo information from another standby database and not from the original primary database. A physical or logical standby database can be set up to send the incoming redo information to other remote destinations in the same manner as the primary database, with up to one level of redirection.

  See Also: Appendix E

- **Oracle9*i* Data Guard broker**

  The broker now supports:

  - Up to nine physical or logical standby destinations
  - Failover and switchover operations

  See Also: *Oracle9i Data Guard Broker*

- **New keywords for the `REMOTE_ARCHIVE_ENABLE` initialization parameter include:**

  - `send`
  - `receive`

  See Also: Section 5.5.2.1, "Setting Permission to Archive Redo Logs"

- **New attributes for the `LOG_ARCHIVE_DEST_`*n* initialization parameter include:**
  - `[NO]TEMPLATE`

    Defines a directory specification and format for archived redo logs at the standby destination.
  - `[NO]NET_TIMEOUT`

    Specifies the number of seconds the log writer process will wait for status from the network server of a network operation issued by the log writer process.
  - `PARALLEL` qualifier to the `SYNC` attribute

    Indicates whether I/O operations to multiple destinations are done in parallel or in series.

    See Also: Chapter 12, "LOG_ARCHIVE_DEST_n Parameter Attributes"

- **New syntax added to the `ALTER DATABASE` statement includes:**
  - `ACTIVATE [PHYSICAL | LOGICAL] STANDBY DATABASE [SKIP [STANDBY LOGFILE]]`
  - `COMMIT TO SWITCHOVER TO {PHYSICAL | LOGICAL} {PRIMARY | STANDBY} [[WITH | WITHOUT] SESSION SHUTDOWN [WAIT | NOWAIT]]`
  - `[NO]FORCE LOGGING`
  - `RECOVER MANAGED STANDBY DATABASE [FINISH [SKIP [STANDBY LOGFILE] [WAIT | NOWAIT]]`
  - `RECOVER MANAGED STANDBY DATABASE [THROUGH {ALL | LAST | NEXT} SWITCHOVER]`
  - `RECOVER MANAGED STANDBY DATABASE [THROUGH ALL ARCHIVELOG | [ THREAD` *n* `] SEQUENCE` *n* `]`
  - `REGISTER [OR REPLACE] [PHYSICAL | LOGICAL] LOGFILE` *filespec*
  - `SET STANDBY DATABASE TO MAXIMIZE {PROTECTION | AVAILABILITY | PERFORMANCE}`
  - `START LOGICAL STANDBY APPLY`

- {STOP | ABORT} LOGICAL STANDBY APPLY

  See Also: Chapter 13, "SQL Statements"

- **New views have been added:**
  - DBA_LOGSTDBY_EVENTS
  - DBA_LOGSTDBY_LOG
  - DBA_LOGSTDBY_NOT_UNIQUE
  - DBA_LOGSTDBY_PARAMETERS
  - DBA_LOGSTDBY_PROGRESS
  - DBA_LOGSTDBY_SKIP
  - DBA_LOGSTDBY_SKIP_TRANSACTION
  - DBA_LOGSTDBY_UNSUPPORTED
  - V$DATAGUARD_STATUS
  - V$LOGSTDBY
  - V$LOGSTDBY_STATS

  See Also: Chapter 14, "Views"

- **New columns have been added to existing fixed views:**
  - V$ARCHIVE_DEST view:
    - NET_TIMEOUT
    - TYPE
  - V$ARCHIVE_DEST_STATUS view:
    - PROTECTION_MODE
    - SRL
  - V$DATABASE view:
    - GUARD_STATUS
    - SUPPLEMENTAL_LOG_DATA_MIN
    - SUPPLEMENTAL_LOG_DATA_PK

        - SUPPLEMENTAL_LOG_DATA_UI
        - FORCE_LOGGING
        - PROTECTION_LEVEL

        See Also: Chapter 14, "Views"

- **Existing columns have been renamed in existing fixed views:**
  - V$ARCHIVE_DEST view:
    - MANIFEST has been renamed to REGISTER and values have been changed to YES and NO.
    - REGISTER has been renamed to REMOTE_TEMPLATE.
  - V$DATABASE view:
    - STANDBY_MODE has been renamed to PROTECTION_MODE and values MAXIMUM PROTECTED, MAXIMUM AVAILABILITY, RESYNCHRONIZATION, MAXIMUM PERFORMANCE, and UNPROTECTED have been added.

    See Also: Chapter 14, "Views"

- **New values have been added to existing columns of existing fixed views:**
  - TRANSMIT_MODE column of the V$ARCHIVE_DEST view:
    - SYNC=PARALLEL
    - SYNC=NOPARALLEL
    - ASYNC
  - REMOTE_ARCHIVE column of the V$DATABASE view:
    - send
    - receive

    See Also: Chapter 14, "Views"

- **New integer values have been added for the LOG_ARCHIVE_TRACE parameter:**
  - 1024: RFS physical client tracking

– 2048: ARC*n* or RFS heartbeat tracking

See Also: Chapter 6, "Log Apply Services"

Oracle9*i* Release 1 (9.0.1) New Features in Data Guard

The features and enhancements described in this section were added to Data Guard in Oracle9*i* release 1 (9.0.1).

- **Oracle9*i* Data Guard**

  Oracle8*i* Standby Database has been renamed to Oracle9*i* Data Guard.

- **Oracle9*i* Data Guard broker**

  The broker is a new management framework that simplifies the configuration and control of a Data Guard configuration, and adds the ability to monitor the configuration.

  The broker provides two new interfaces:

- **Oracle9*i* Data Guard Manager (GUI)**

  Oracle9*i* Data Guard Manager is a graphical user interface that provides easy configuration of a two-site Data Guard environment, the most typical Data Guard configuration.

  – **Data Guard command-line interface**

    The Data Guard command-line interface allows you to control and monitor a Data Guard configuration directly from the command-line prompt or from within scripts.

    See Also: *Oracle9i Data Guard Broker*

- **No data loss**

  With the **no-data-loss** feature, the primary database will not acknowledge primary database modifications until they have been confirmed as being available on at least one standby database. Data is protected when primary database modifications occur while there is connectivity to the standby database. Data is unprotected when primary database modifications occur while connectivity to the standby database is not available.

  The database administrator (DBA) can place the database into one of the following modes:

- Guaranteed protection
- Instant protection
- Rapid protection
- Delayed protection

Note: In Oracle9*i* release 2, these protection modes have been replaced by the maximum protection, maximum availability, and maximum performance modes. See Oracle9i Release 2 (9.2) New Features in Data Guard.

- **New syntax has been added to the `ALTER DATABASE` statement to support the no-data-loss feature:**
  - `SET STANDBY DATABASE {PROTECTED | UNPROTECTED}`
  - `ADD [STANDBY] LOGFILE TO [THREAD` *`integer`*`] [GROUP` *`integer`*`]` *`filespec`*
  - `ADD [STANDBY] LOGFILE MEMBER '`*`filename`*`' [REUSE] TO GROUP` *`integer`*

Note: In Oracle9*i* release 2, this syntax was further revised. See Oracle9i Release 2 (9.2) New Features in Data Guard.

- **No data divergence**

  Data divergence occurs when the primary database is modified, but the modifications are not available to be applied to a corresponding standby database. Subsequent failover of the primary database to the standby database would result in the loss of previously committed transactions.

  Note that no data divergence is different from no data loss.

  Data divergence is prohibited through the concept of *protected* data. In this sense, the primary database's data is protected by the existence of one or more synchronized standby databases. The failure of the last standby database causes the primary database instance to shut down to avoid data divergence.

  Data integration is responsible for reapplying diverged (unprotected) data to the standby databases.

See Also: Section 5.2.5

- **Database switchover**

  The new database switchover feature provides the database administrator (DBA) with the ability to switch the role of the primary database to one of the available standby databases. The chosen standby database becomes the primary database and the original primary database then becomes a standby database. There is no need to reinstantiate any of the databases in the switchover operation. There is no data divergence between the original and the new primary database after the successful completion of the database switchover.

  See Also: Chapter 7, "Role Management Services"

- **Archive gaps are automatically detected and transmitted**

  Archive gaps are detected and transmitted automatically. This feature is built into log transport services and log apply services. The log transport services archive gap detection is always enabled in Oracle9*i* on the primary database. To enable the log apply services archive gap detection on the standby database, the DBA must define the values of two new initialization parameters: `FAL_CLIENT` and `FAL_SERVER`. Oracle Corporation recommends that you always set the log apply services archive gap detection. The log transport services archive gap detection is a best-attempt-effort and does not guarantee complete archive gap detection in some cases. The log apply services archive gap detection does guarantee complete archive gap detection and transmittal for the managed recovery process.

  See Also: Section 6.5

- **Adding new datafiles**

  You can add new datafiles to the primary database without having to manually add the corresponding datafile to the standby databases. The new datafile is created and added to the standby databases if the DBA has set up the `STANDBY_FILE_MANAGEMENT` and `DB_FILE_NAME_CONVERT` initialization parameters.

  See Also: Section 6.3.4 and Section 8.3.1

- **Background managed recovery mode**

You can now create a background process to perform managed recovery. This frees the foreground terminal session that you used to execute the managed recovery statement.

See Also: Section 6.3.2

- **Parallel recovery**

  You can execute parallel recovery using the `RECOVER MANAGED STANDBY DATABASE` clause of the SQL `ALTER DATABASE` statement. This allows faster recovery on your standby databases.

  See Also: Section 6.3.3.9

- **More archive destinations**

  You can specify up to 10 archive destinations in Oracle9*i*. Prior versions allowed only up to 5 archive destinations.

  See Also: Section 5.3.1

- **Incremental change to `LOG_ARCHIVE_DEST_`*n* initialization parameter**

  The DBA can incrementally modify individual attributes of the `LOG_ARCHIVE_DEST_`*n* initialization parameter, where *n* is a value from 1 to 10.

  See Also: Section 5.3.1

- **Standby redo logs are introduced**

  You can control where the standby database stores the redo data received from the primary site. This data can be stored on the standby site using either standby redo logs or archived redo logs.

  See Also: Section 5.8.4

- **Archiver process (ARC*n*) is available on standby databases**

  The ARC*n* process is used on standby databases to archive standby redo logs. It uses the `LOG_ARCHIVE_DEST_`*n* initialization parameter (where *n* is a value from 1 to 10). The archiver process on the standby database can archive locally and remotely to other standby databases.

  See Also: Section 5.3.1

- **Changes to archiving online redo logs**

  In prior releases, the current redo log had to be full before you could archive it to the standby site. In Oracle9*i*, it is possible to:

  – Archive the current redo log. In previous releases, you needed to perform a log switch first.

  – Archive an online redo log based on the SCN (system change number) value when the database is mounted, but not open.

  – Archive an online redo log when a backup control file is being used. In previous releases, a current control file was required.

  See Also: Section 5.5 and Section 10.1.8

- **New control options**

  The following control options have been added to this release:

  – `DELAY`

    Specifies an absolute apply delay interval to the managed recovery operation. The managed recovery operation waits the specified number of minutes before applying the archived redo logs.

  – `DISCONNECT`

    Starts managed recovery in background mode.

  – `EXPIRE`

    Specifies the number of minutes after which the managed recovery operation automatically terminates, relative to the current time. The managed recovery operation terminates at the end of the current archived redo log that is being processed.

  – `FINISH`

    Completes managed recovery in preparation for a failover from the primary database to the standby database.

  – `NEXT`

    Directs the managed recovery operation to apply a specified number of archived redo logs as soon as possible after the log transport services have archived them.

  – `NODELAY`

Directs the managed recovery operation to apply the archived redo logs as soon as possible after log transport services have archived them.

See Also: Section 6.3.3

- **Standby Database Real Application Clusters support**

  Oracle9*i* provides the ability to perform true database archiving from a primary database to a standby database when both databases reside in a Real Application Clusters environment. This allows you to separate the log transport services processing from the log apply services processing on the standby database, thereby improving overall primary and standby database performance.

  See Also: Appendix D, "Standby Database Real Application Clusters Support"

- **Archive log repository**

  Oracle9*i* introduces the archive log repository, which is a standalone standby database. An archive log repository is created by using the standby control file, starting the instance, and mounting it. This type of destination allows off-site archiving of redo log files.

  See Also: Section 1.4 and Section 5.2.2

- **Relationship defined between an archived redo log and an archive destination**

  The DBA can now determine the following by joining the `V$ARCHIVED_LOG` and `V$ARCHIVE_DEST` fixed views:

  – Archive destination information for an archived redo log

  – All archived redo logs generated from a destination

  See Also: Chapter 14, "Views"

- **New initialization parameters**

  The following initialization parameters have been added to this release for each Oracle instance:

  – `REMOTE_ARCHIVE_ENABLE`

- FAL_CLIENT
- FAL_SERVER
- STANDBY_FILE_MANAGEMENT
- ARCHIVE_LAG_TARGET

  See Also: Chapter 11, "Initialization Parameters"

- **New attributes for the LOG_ARCHIVE_DEST_*n* initialization parameter include:**
  - ARCH | LGWR
  - [NO]AFFIRM
  - [NO]ALTERNATE
  - [NO]DELAY
  - [NO]DEPENDENCY
  - [NO]MAX_FAILURE
  - [NO]QUOTA_SIZE
  - [NO]QUOTA_USED
  - [NO]REGISTER | REGISTER [=*location_format*]
  - NOREOPEN
  - SYNC | ASYNC

  See Also: Section 5.3.1 and Chapter 12, "LOG_ARCHIVE_DEST_n Parameter Attributes"

- **A new range of values and a keyword have been added to the LOG_ARCHIVE_DEST_STATE_*n* initialization parameter:**
  - The variable *n* can be a value from 1 to 10 (versus 1 to 5 in Oracle8*i*).
  - The new keyword, ALTERNATE, has been added.

  See Also: Section 5.3.1.2

- **One to ten destinations (versus one to five in Oracle8*i*) must archive successfully before the log writer process (LGWR) can overwrite the online redo logs**

  See Also: Section 5.5.2.4

- **New tracing levels have been added to the `LOG_ARCHIVE_TRACE` initialization parameter:**
  - Tracing level 128 allows you to track fetch archive log (FAL) server process activity.
  - Tracing level 256 will be available in a future release.
  - Tracing level 512 allows you to track asynchronous log writer activity.

  See Also: Section 6.4.8.3

- **New clauses have been added to the `ALTER DATABASE` statement:**
  - `ACTIVATE [PHYSICAL] STANDBY DATABASE [SKIP [STANDBY LOGFILE]]`
  - `ADD [STANDBY] LOGFILE TO [THREAD` *`integer`*`] [GROUP` *`integer`*`]` *`filespec`*
  - `ADD [STANDBY] LOGFILE MEMBER '`*`filename`*`' [REUSE] TO '`*`logfile-descriptor`*`'`
  - `COMMIT TO SWITCHOVER TO [PHYSICAL] {PRIMARY | STANDBY} [[NO]WAIT]`
  - `REGISTER [PHYSICAL] LOGFILE` *`filespec`*
  - `SET STANDBY DATABASE {PROTECTED | UNPROTECTED}`

  Note: In Oracle9*i* release 2, this syntax was further revised. See Oracle9i Release 2 (9.2) New Features in Data Guard.

- **New keywords have been added to the `RECOVER MANAGED STANDBY DATABASE` clause:**
  - `NODELAY`
  - `CANCEL [IMMEDIATE] [NOWAIT]`

  - [DISCONNECT [FROM SESSION]]
  - [FINISH [NOWAIT]]
  - [PARALLEL [*integer*]]
  - NEXT
  - EXPIRE
  - DELAY

Note: In Oracle9*i* release 2, this syntax was further revised. See Oracle9i Release 2 (9.2) New Features in Data Guard.

- **New fixed views have been added:**
  - V$ARCHIVE_DEST_STATUS
  - V$ARCHIVE_GAP
  - V$MANAGED_STANDBY
  - V$STANDBY_LOG

  See Also: Chapter 14, "Views"

- **New columns have been added to existing fixed views:**
  - V$ARCHIVE_DEST view:
    * AFFIRM
    * ALTERNATE
    * ARCHIVER
    * ASYNC_BLOCKS
    * DELAY_MINS
    * DEPENDENCY
    * FAILURE_COUNT
    * LOG_SEQUENCE
    * MANIFEST (This column name was new in Oracle9*i* release 1; it was changed to REGISTER in Oracle9*i* release 2.)

    - MAX_FAILURE
    - MOUNTID
    - PROCESS
    - QUOTA_SIZE
    - QUOTA_USED
    - REGISTER (This column name was new in Oracle9*i* release 1; it was changed to REMOTE_TEMPLATE in Oracle9*i* release 2.)
    - SCHEDULE
    - TRANSMIT_MODE
    - TYPE
    - New values, ALTERNATE and FULL, have been added to the STATUS column.
- V$ARCHIVED_LOG view:
    - APPLIED
    - BACKUP_COUNT
    - COMPLETION_TIME
    - CREATOR
    - DELETED
    - DEST_ID
    - DICTIONARY_BEGIN
    - DICTIONARY_END
    - REGISTRAR
    - STANDBY_DEST
    - STATUS
    - END_OF_REDO
    - ARCHIVAL_THREAD#
- V$LOG view:
    - A new value, INVALIDATED, has been added to the STATUS column.

- V$LOGFILE view:
  - TYPE
- V$DATABASE view:
  - ACTIVATION#
  - ARCHIVELOG_CHANGE#
  - DATABASE_ROLE
  - REMOTE_ARCHIVE
  - STANDBY_MODE
  - SWITCHOVER_STATUS
- V$ARCHIVE_DEST_STATUS view:
  - STANDBY_LOGFILE_COUNT
  - STANDBY_LOGFILE_ACTIVE

Note: In Oracle9*i* release 2, new values were added and existing columns renamed in the V$DATABASE and V$ARCHIVE_DEST fixed views. See Oracle9i Release 2 (9.2) New Features in Data Guard.

Oracle8*i* New Features in Data Guard

The features and enhancements described in this section were added to the standby database in Oracle8*i*.

- **Read-only mode**

  The read-only mode option can be used to make a standby database available for queries and reporting, even while redo logs are being archived from the primary database site.

  See Also: Section 6.3.5

- **Backing up a primary database from a standby database**

  You can use the Recovery Manager utility (RMAN) to back up the primary database using the standby database.

See Also: Section 8.1

Part I

Concepts and Administration

This part contains the following chapters:

1

Oracle9*i* Data Guard Concepts

Oracle9*i* Data Guard provides an extensive set of data protection and disaster recovery features to help you to survive disasters, human errors, and corruptions that can incapacitate a database. Based on business requirements, these features can be advantageously combined with other Oracle9*i* high-availability and disaster recovery features for enhanced levels of high availability and disaster protection.

This chapter explains Oracle9*i* Data Guard concepts. It includes the following topics:

1.1 Oracle9*i* Data Guard Overview

Oracle9*i* **Data Guard** is the management, monitoring, and automation software that works with a production database and one or more **standby databases** to protect your data against failures, errors, and corruptions that might otherwise destroy your database. It protects critical data by automating the creation, management, and monitoring of the databases and other components in a Data Guard configuration. It automates the otherwise manual process of maintaining a transactionally

consistent copy of an Oracle production database, called a standby database, that can be used if the production database is taken offline for routine maintenance or becomes damaged.

In a Data Guard configuration, a production database is referred to as a **primary database.** Using a backup copy of the primary database, you can create from one to nine physical and logical standby databases and incorporate them in a Data Guard configuration. Primary and standby databases can be running on a single node or in a Real Application Clusters environment. Every standby database is associated with only one primary database. However, a single primary database can support multiple physical standby databases, logical standby databases, or a mix of both in the same configuration:

- Physical standby databases

  A **physical standby database** is physically identical to the primary database, with on-disk database structures that are identical to the primary database on a block-for-block basis. The physical standby database is updated by performing recovery. It can either be recovering data or open for read-only reporting.

- Logical standby databases

  A **logical standby database** is logically identical to the primary database. The logical standby database is updated using SQL statements. The tables in a logical standby database can be used simultaneously for recovery and for other tasks such as reporting, summations, and queries.

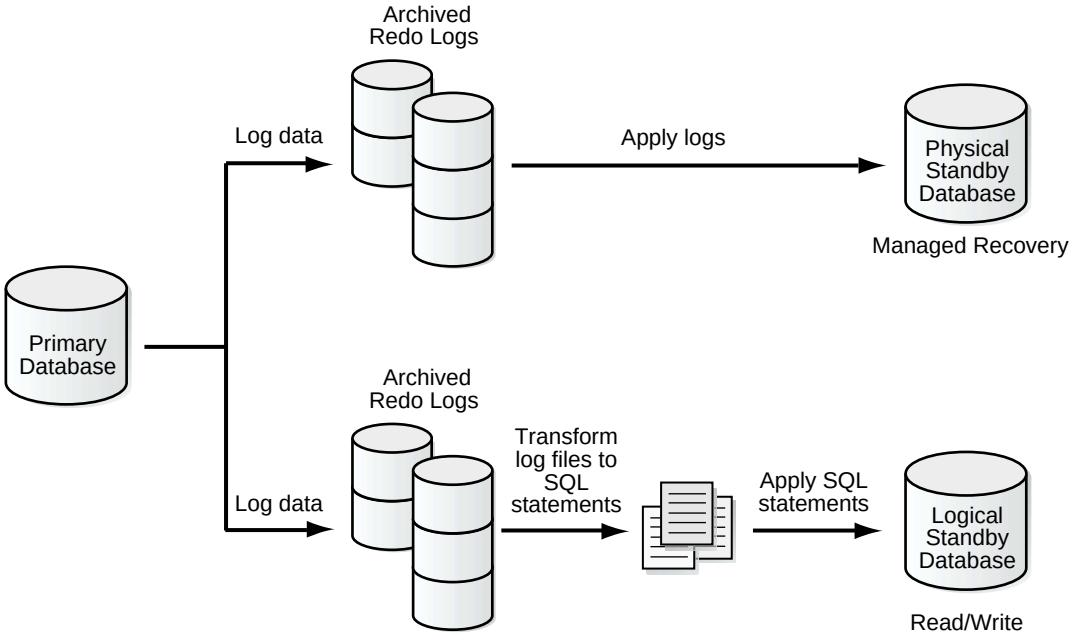
Oracle9*i* Data Guard maintains each standby database as a transactionally consistent copy of the primary database using standard Oracle online redo logs. As transactions add data or change information stored in the primary database, the changes are also written to the online redo logs. The redo logs associated with the primary database are archived to standby destinations by log transport services and applied to the standby database by log apply services.

- **Log transport services** control the automated transfer of **archived redo logs** from the primary database to one or more standby databases. Log transport services transmit redo logs to local and remote physical and logical standby databases. You can configure log transport services to balance data protection and availability against performance.
- **Log apply services** apply the archived redo logs differently for physical and logical standby databases. Archived redo logs are applied:
  - To a physical standby database when it is performing recovery (for example, in managed recovery mode), but not when it is open for reporting access (for example, in read-only mode).

– To a logical standby database by converting the redo data into SQL statements and applying them to an open logical standby database. Because the logical standby database remains open, tables that are maintained can be used simultaneously for other tasks such as reporting, summations, and queries.

Figure 1–1 shows logs being applied to both physical and logical standby databases in the same Data Guard configuration.

Figure 1–1 Log Apply Services for Physical and Logical Standby Databases



Role management services work with log transport services and log apply services to reduce downtime of the primary database for planned outages, such as operating system or hardware upgrades, by performing a **switchover** operation that is a graceful role reversal between the primary database and one of its standby databases. Role management services also minimize downtime from an unplanned failure of the primary database by facilitating the quick *fail over* to one of the standby databases through a **graceful failover** or **forced failover** operation.

Because the first priority of Data Guard is to ensure that copies of the data are elsewhere, usable, and correct when any data-destroying condition occurs on a primary database, you can use Oracle Net services to locate the standby databases on a geographically remote location (sometimes referred to as a site) over a wide area network (WAN), as well as making standby databases locally accessible on a local area network (LAN). By locating standby databases remotely from the primary system, you ensure that an event that disables the primary database will be unlikely to also disable the standby database. However, it can also be beneficial to locate standby databases on a LAN as another failover option to protect against user errors, system problems, and data corruptions.

Note: The primary purpose of Oracle9*i* Data Guard is to keep your data highly available against any event, including disasters; it is *complementary* to traditional backup, restore, and clustering techniques.

Finally, Data Guard allows you to configure and manage a Data Guard configuration through any of several interfaces, including SQL statements, initialization parameters, a PL/SQL package, and the Oracle9*i* Data Guard broker. The **broker** is a distributed management framework that automates the creation and management of Data Guard configurations through the Data Guard Manager graphical user interface or its command-line interface.

1.2 Benefits of Oracle9*i* Data Guard

Oracle9*i* Data Guard automates the tasks involved in setting up and managing the Data Guard environment, including one or more standby databases, the log transport services, log apply services, and role management services. Oracle9*i* Data Guard helps you survive events that might otherwise make your database unavailable by:

- Enhancing high **availability** and disaster protection using standby databases

  An Oracle9*i* Data Guard configuration is a collection of loosely connected systems that combine the primary database and the standby databases into a single, easily managed disaster recovery solution. The databases in a Data Guard configuration are connected by Oracle Net and may be dispersed geographically.

- Offering the maximum protection, maximum availability, and maximum performance modes to help you balance data availability against performance requirements
- Centralizing and simplifying management

  The Data Guard broker distributed management framework provides the Data Guard Manager graphical user interface and the Data Guard command-line interface to automate the management and operational tasks across the multiple databases in a Data Guard configuration. The broker also monitors all of the systems within a single standby database configuration.
- Providing failover and switchover capabilities

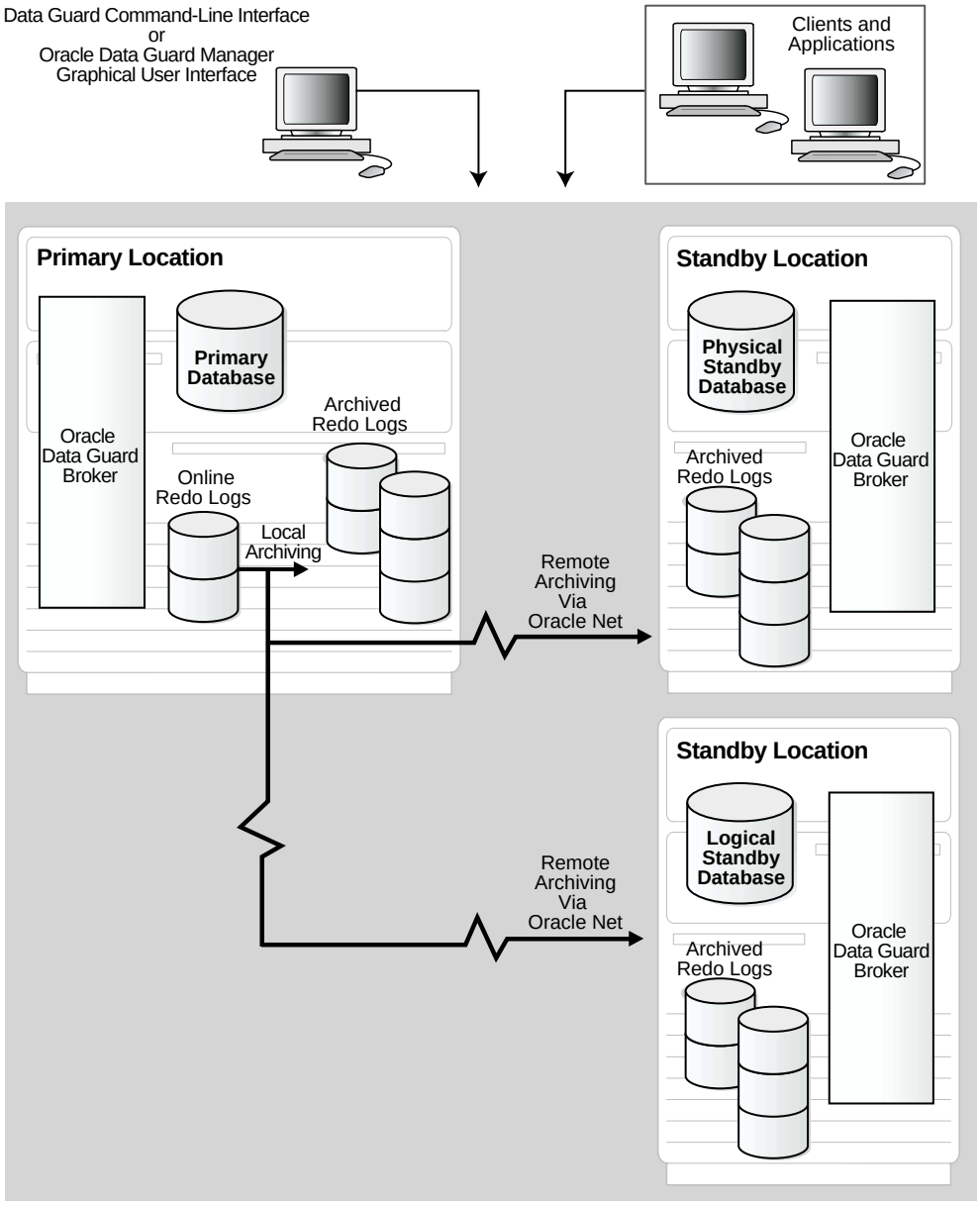
  These failover and switchover capabilities allow you to perform planned and unplanned role reversals between primary and standby databases.
- Safeguarding against physical corruptions

  For standby databases, the risk of physical corruptions is reduced. Primary-side physical corruptions, due to device failure, are unlikely to propagate through the archived redo logs that are transported to the standby database.
- Protecting against user errors

  You can resolve user errors or perform a failover operation when the primary database is permanently damaged. For example, if a critical table is accidentally dropped from the primary database and the change has not yet been applied to the standby database, you can fail over to a standby database or use it to transport the relevant tablespaces back to the primary database.

Figure 1–2 shows an example of a Data Guard configuration that allows failover to occur to either a physical standby database or a logical standby database. If the primary database becomes unavailable, the workload can fail over to either standby database.

Figure 1–2 Oracle9i Data Guard Configuration



1.3 Physical and Logical Standby Databases

The Data Guard configuration shown in Figure 1–2 consists of loosely connected systems, including a single primary database, a logical standby database, and a physical standby database. A configuration can include multiple (up to nine) standby databases, which can be all physical standby databases, all logical standby databases, or a mix of both.

1.3.1 Configuring Local and Remote Standby Databases

In Figure 1–2, the logical and physical standby databases are geographically distant from the primary database, connected by Oracle Net over a WAN. While configuring remote standby databases is the best strategy for disaster protection, there are many good reasons for configuring both local and remote standby databases:

- Local standby databases are connected to the primary database over a LAN, which provides an inexpensive and reliable network link with low latency. The highest data protection modes perform very well in this configuration, and also provide fast switchover operations when you need to perform routine maintenance.
- Remote standby databases are the best solution for disaster tolerance because of the greater distance between the primary and standby databases. Although performance can be affected by the WAN's lower bandwidth and higher latency, remote standby databases on WAN connections are the best solution for disaster protection.

When a primary database is open and active, both the local and remote standby databases are sent redo log data generated as transactions are made to the primary database. In a Data Guard environment, a physical standby database can be either recovering data or open for reporting access, and it can also be performing backup operations in either mode. However, because SQL is used to apply the changes to a logical standby database, users can access it for queries and reporting purposes at any time. This section describes the operational characteristics of physical and logical standby databases.

1.3.2 Physical Standby Databases

Oracle Corporation began shipping the standby database feature with Oracle release 8.0.4. Subsequent releases incorporated technical enhancements and user feedback that helped standby databases evolve as automated standby databases,

managed standby databases, and the physical standby database disaster recovery feature available today with the Oracle9*i* Data Guard.

A *physical* standby database has its roots in media recovery. A physical standby database must be physically identical to the primary database. Put another way, standby on-disk database structures must be identical to the primary database on a block-for-block basis, because a recovery operation applies changes block-for-block using the physical row ID. The database schema, including indexes, must be the same, and the database cannot be opened for updates. If opened for updates, the standby database would have different row IDs, making continued recovery impossible.

You can operate a physical standby database in either of the following modes:

- **Managed recovery mode**

  In managed recovery mode, the primary database archives redo logs to the standby database and log apply services automatically apply the redo logs to the standby database.

- **Read-only mode**

  To use the standby database for reporting purposes, open it in read-only mode. Log apply services cannot apply redo logs to the standby database when it is in this mode, but you can execute queries on the database. The primary database continues to archive to the standby site so long as the standby instance is mounted.

Although the standby database cannot be in more than one mode at the same time, you can change between the modes. For example, you can run in managed recovery mode, open in read-only mode to run reports, and then return to managed recovery mode to apply outstanding archived redo logs. You can repeat this cycle, alternating between reporting and recovery, as necessary.

In addition, you can perform online backup operations in either mode. Only backup copies from physical standby databases can be recovered to the primary database.

Physical standby databases provide the following benefits:

- Support for all DDL and DML

  A physical standby database provides data protection and disaster recovery capabilities for all Oracle databases, with no restrictions on datatypes, types of tables, or types of data definition language (DDL) and data manipulation language (DML) operations.

- Performance

  A physical standby database provides better apply performance than logical standby databases, especially when there are a large number of updates on the primary system and a high rate of redo data is generated. Log apply services are better able to keep physical standby databases up-to-date because the process of translating the redo data to SQL statements and reexecuting the SQL on a logical standby database takes more time and uses more system resources (memory and I/O operations) than using media recovery to apply the changes to a physical standby database.

- Backup off-loading capabilities

  A physical standby database can off-load backup operations from the primary system. Backup copies from physical standby databases can be recovered on the primary database.

- Report generation

  A physical standby database can be used to run reports while the standby database is in read-only mode.

1.3.3 Logical Standby Databases

Oracle9*i* introduces logical standby databases, which answers Oracle users' requests for a standby database that provides data protection while simultaneously satisfying additional reporting requirements.

A logical standby database is logically identical to the primary database and can be used as a production database if the primary database is taken offline for routine maintenance or is damaged through human error, corruption, or a disaster.

Synchronization of a logical standby database with a primary database is done in **SQL apply mode** using standard Oracle archived redo logs. Log apply services automatically apply archived redo log information to the logical standby database by transforming transaction information into SQL statements (using LogMiner technology) and applying the SQL statements to the logical standby database.

Because the logical standby database is updated using SQL statements (unlike a physical standby database), it must remain open, and the tables that are maintained can be used simultaneously for other tasks such as reporting, summations, and queries. Moreover, these tasks can be optimized by creating additional indexes and materialized views on the maintained tables to those used on the primary database. Even though a logical standby database may have a different physical layout than

its primary database, the logical standby database will support failover and switchover from the primary database.

The logical standby database is open in read / write mode, but the target tables for the regenerated SQL are available only in read-only mode for reporting purposes.

In addition to off-loading reporting and query processing from the primary database, logical standby databases provide the following benefits:

- Efficient use of system resources

  A logical standby database is an open production database and can be used concurrently for both data protection and reporting.

- Decision support

  Because the data in a logical standby database can be presented with a different physical layout, you can create additional indexes and materialized views to suit your reporting and query requirements.

1.4 Log Transport Services and Log Apply Services

The log transport services component of Oracle9*i* Data Guard controls the automated transmission of archived redo logs from the primary database to one or more standby locations. In addition, you can use log transport services to set up the following:

- Archive log repository

  This type of destination allows off-site archiving of redo logs. An archive log repository is created by using a standby control file, starting the instance, and mounting the database. This database contains no datafiles and cannot be used for primary database recovery.

- Cross-instance archival database environment

  A **cross-instance archival** database environment is possible on both the primary and standby databases. Within a Real Application Clusters environment, each instance directs its archived redo logs to a single instance of the cluster. This instance, known as the **recovery instance,** is typically the instance where managed recovery is performed. The recovery instance typically has a tape drive available for RMAN backup and restore support.

Prior to Oracle9*i*, **redo logs** were transferred from the primary database to the standby database as they were archived. However, you can customize log transport services and log apply services for a variety of standby database configurations.

Oracle9*i* Data Guard provides the DBA with great flexibility when defining archive destinations, archive completion requirements, I/O failure handling, and automated transmission restart capability.

For example:

- The DBA can choose to synchronously write redo log updates directly from the primary database to the standby database to provide for a comprehensive *no-data-loss* disaster-recovery solution. Should a disaster strike at the primary database, all redo logs necessary to preserve all transactions will be available for application at the standby databases.
- The DBA can choose to transfer archived redo logs in synchronous mode running in parallel to the standby database. Customization options provide the means to control the potential for data loss.
- The DBA can specify a delay for application of the redo log data once it arrives at the standby database. Immediate application means faster **failover** due to a more up-to-date standby database. However, delaying the application of redo logs allows the DBA a window of opportunity to halt the application of erroneous logs, preventing human errors or corruptions from propagating to the standby database.

See Also: Chapter 5, "Log Transport Services" and Chapter 6, "Log Apply Services"

1.5 Role Management Services

A database operates in one of two mutually exclusive roles: primary or standby. Role management services operate in conjunction with the log transport services and log apply services to change these roles dynamically as a planned transition called a switchover operation, or as a result of a database failure through either a graceful failover or a forced failover operation:

- Database switchover

  Switchover operations provide the means to transition database roles from a primary role to a standby role and from a standby role to a primary role.

- Graceful (no-data-loss) failover

  A no-data-loss failover is possible if the corresponding primary database is operating in either the maximum protection or maximum availability data protection mode.

- Forced (minimal-data-loss) failover

  Minimal-data-loss failover occurs when primary database modifications are not yet available on the standby site at the time the failover operation occurs. Minimal data loss is possible when operating in a data protection mode such as maximum performance mode, when the failover operation occurs.

You should not fail over to a standby database except in an emergency, because the failover operation is an unplanned transition that may result in data loss. If you need to fail over before the primary and standby databases are resynchronized, and if the standby database becomes inaccessible, data on the primary database may diverge from data on standby databases. You can prevent this by using forced logging (to force the generation of redo records of changes against the database), standby redo logs, and the maximum protection mode with physical standby databases. Data loss may occur when you fail over to a standby database whose corresponding primary database is not in the maximum protection or maximum availability mode.

The amount of data differences or data loss you incur during a failover operation is directly related to how you configure the overall Data Guard configuration and log transport services, and can be prevented entirely by using maximum protection mode.

See Also: Chapter 7, "Role Management Services"

1.6 Data Guard Architecture

Oracle9*i* Data Guard uses several processes to achieve the automation necessary for disaster recovery and high availability.

- On the primary location, log transport services use the following processes:
  - Log writer process (LGWR)

    The **log writer process (LGWR)** collects transaction redo and updates the **online redo logs.**
  - Archiver process (ARC*n*)

    The **archiver process (ARC*n*)** creates a copy of the online redo logs, either locally or remotely, for standby databases.
  - Fetch archive log (FAL) process (physical standby databases only)

    The **fetch archive log (FAL)** process provides a client/server mechanism for resolving gaps detected in the range of archived redo logs generated at the

primary database and received at the standby database. The **FAL client** requests the transfer of archived redo log files automatically when it detects a **gap** in the redo logs received by the standby database. The **FAL server** typically runs on the primary database and services the FAL requests coming from the FAL client. The FAL client and server are configured using the `FAL_CLIENT` and `FAL_SERVER` initialization parameters that are set on the standby location.

See Also: Section 5.1.3

- On the standby location, log transport services use the following processes:
  - Remote file server (RFS)

    The **remote file server (RFS)** process receives redo logs from the primary database.
  - ARC*n* process

    The archiver (ARC*n)* process archives the **standby redo logs** when standby redo logs and LGWR are used.
- On the standby location, log apply services use the following processes:
  - **Managed recovery process (MRP)**

    For physical standby databases only, the MRP applies archived redo log information to the physical standby database.
  - **Logical standby process (LSP)**

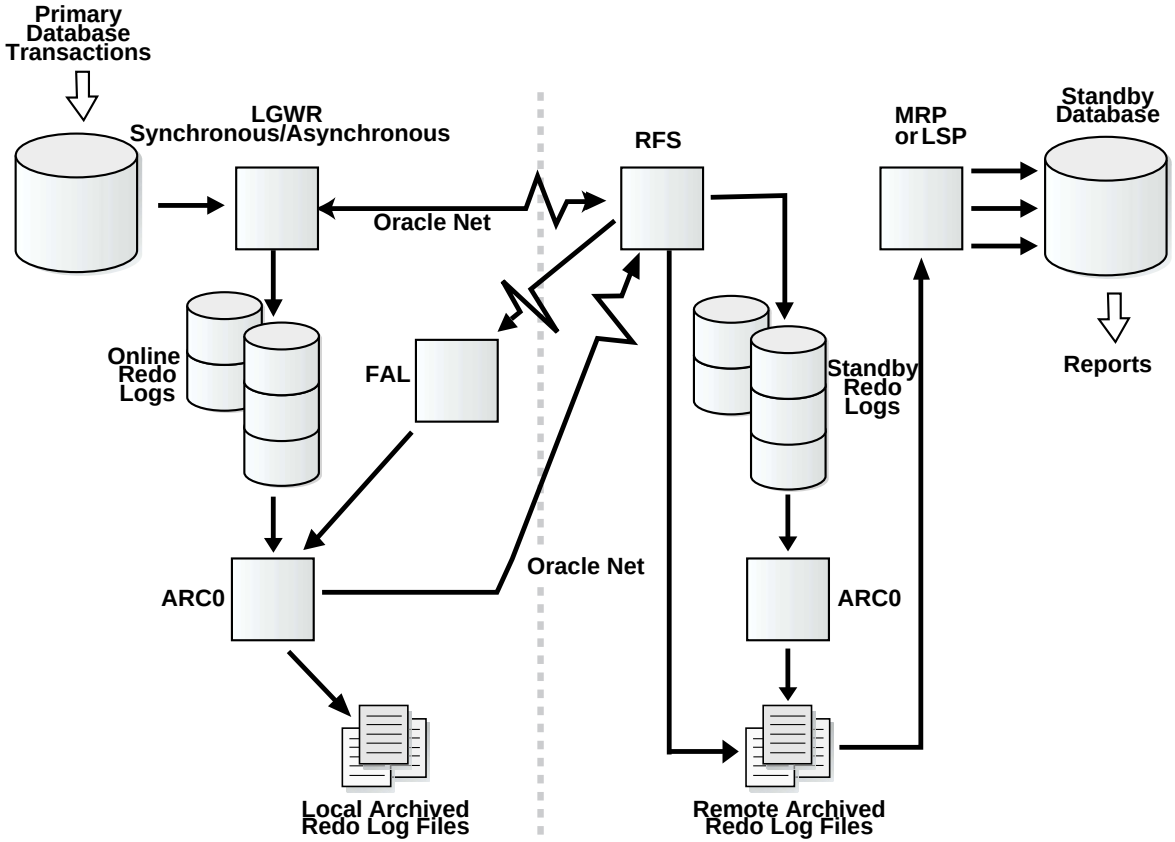
    For logical standby databases only, the LSP applies archived redo log information to the logical standby database, using SQL interfaces.
- On the primary and standby locations, the Data Guard broker uses the following processes:
  - Data Guard broker monitor (DMON) process

    The DMON processes work cooperatively to manage the primary and standby databases as a unified configuration. The DMON processes work together to execute switchover and failover operations, monitor the status of the databases, and manage log transport services and log apply services.

    See Also: *Oracle9i Data Guard Broker*

Figure 1–3 identifies the relationships of these processes to the operations they perform and the database objects on which they operate in the absence of the Data Guard broker. In the following figure, the standby redo logs are optionally configured for physical standby databases, except when running in maximum protection mode, which *requires* physical standby databases and standby redo logs. Logical standby databases do not use standby redo logs.

Figure 1–3 Data Guard Architecture



1.7 Data Guard Interfaces

You can use the following interfaces to configure, implement, and manage a standby database:

- SQL statements

  Several SQL statements use a STANDBY keyword to specify operations on a standby database. Other SQL statements do not include standby-specific syntax, but are useful for performing operations on a standby database. Chapter 13 describes the relevant statements.

- Initialization parameters

  Several initialization parameters are used to define the Data Guard environment. Table 11–1 describes relevant initialization parameters.

- Oracle9*i* Data Guard Manager

  The Oracle9*i* Data Guard Manager graphical user interface (GUI) is a broker interface that you can use to automate many of the tasks involved in configuring and monitoring a Data Guard environment.

  See Also: *Oracle9i Data Guard Broker* and the Oracle9*i* Data Guard Manager online help for information on the Data Guard Manager GUI and the Oracle9*i* Data Guard Manager Wizard

- Data Guard command-line interface

  The Data Guard command-line interface is an alternative to using the Oracle9*i* Data Guard Manager GUI. The command-line interface is useful if you want to use the broker to manage a Data Guard configuration from batch programs or scripts.

  See Also: *Oracle9i Data Guard Broker*

- Recovery Manager (RMAN)

  You can use **Recovery Manager (RMAN)** to create and back up a standby database.

  See Also: *Oracle9i Recovery Manager User's Guide*

See Chapter 3 for information on creating a physical standby database, Chapter 4 for information on creating a logical standby database, and Chapter 10 for scenarios.

1.8 Operational Requirements

Note the following operational requirements for maintaining a standby database:

- The primary database must run in **ARCHIVELOG mode.**
- Use the same database release on the primary and standby databases. The operating system on the primary and standby sites must be the same, but the operating system release does not need to be the same. In addition, the standby sites can use a different directory structure from the primary site.
- The hardware and operating system architecture on the primary and standby locations must be the same. For example, this means a Data Guard configuration with a primary database on a 32-bit Sun system must have a standby database that is configured on a 32-bit Sun system. Similarly, a primary database on a 64-bit HP-UX system must be configured with a standby database on a 64-bit HP-UX system, a primary database on a 32-bit Linux on Intel system must be configured with a standby database on a 32-bit Linux on Intel system, and so forth.
- The primary database can be a single instance database or a multi-instance Real Application Clusters database. The standby databases can be single instance databases or multi-instance Real Application Clusters databases, and these standby databases can be a mix of both physical and logical types.
- The hardware (for example, the number of CPUs, memory size, storage configuration) can be different between the primary and standby systems. If the standby system is smaller than the primary system, you may have to restrict the work that can be done on the standby system after a switchover operation.
- Each primary database and standby database must have its own **control file.**
- If you place your primary and standby databases on the same system, you must adjust the initialization parameters correctly.
- To protect against unlogged direct writes in the primary database that cannot be propagated to the standby database, turn on `FORCE LOGGING` at the primary database before taking datafile backups for standby creation. Keep the database in `FORCE LOGGING` mode as long as the standby database is required.
- If you are currently running Oracle8*i* Data Guard, see *Oracle9i Database Migration* for complete information to help you upgrade to Oracle9*i* Data Guard.
- In Microsoft Windows environments, single-instance primary or standby databases also can be configured with Oracle Fail Safe to provide enhanced

high availability. The standby databases can be logical, physical, or a mixture of both.

2

Configurations and Considerations

A **standby database environment** includes a primary database with from one to nine associated standby databases and potentially cascaded standby databases. This chapter describes the following main factors affecting the Data Guard configuration:

- Number of Standby Databases
- Standby Databases in the Data Guard Environment
- Method of Transmitting Redo Logs to the Standby Database
- Method of Applying Redo Logs or SQL Statements
- Location and Directory Structure of Primary and Standby Databases
- Types of Standby Databases
- Examples of Using Multiple Standby Databases

2.1 Number of Standby Databases

Although a standby database can be synchronized with one and only one primary database, a single primary database can directly support up to nine standby databases. These standby databases are separate and independent, and can reside on multiple systems or on a single system. Also, **cascading standby databases** can be set up to send incoming redo information to other locations in the same manner as the primary database, with up to one level of redirection.

2.2 Standby Databases in the Data Guard Environment

Typically, you create a standby database for one or more of the following reasons:

- To protect against the total destruction of the primary database

- To protect against data corruption (from applications or hardware) of the primary database
- To off-load the cost of backing up data contained in the primary database to the standby database
- To off-load the cost of running read-only reports to the standby databases

There are a number of factors to consider when weighing whether to implement a physical standby database, a logical standby database, or both in a Data Guard configuration:

Protection against total destruction of the primary database Standby databases can be located at the same location as the primary database to provide protection against hardware failures, or located at a remote location from the primary database to protect against electrical power loss, physical structure loss, or other catastrophic outages.

Protection against data corruption Both physical and logical standby databases protect against data corruption through failover capabilities. Both types of standby databases support the maximum availability and maximum performance modes of data protection. Physical standby databases go a step further by supporting standby redo logs and the maximum protection mode. (See Section 5.7 for complete information about the data protection modes.)

In addition, you can delay the application of redo logs already received at one or more standby databases. The ability to stagger the application of changes to standby databases enables not only the protection of production data from data center disasters, but also provides a window of protection from user errors or corruptions.

Off-loading backup operations Online backup operations can be run against physical standby databases to off-load work from the primary database. These backup copies can be applied to the primary database, if needed.

Reporting versus data protection Consider the following points for logical and physical standby databases:

- A physical standby database can be used for reporting or recovery, but not both at the same time. Physical standby databases must be maintained in managed recovery mode for data protection. To use a physical standby database for reporting purposes, you must take it out of managed recovery mode and open it in read-only mode. Log apply services cannot apply archived redo logs to the physical standby database when it is in read-only mode, but the primary

database continues to transmit archived redo logs to the standby database so long as the standby instance is started and mounted. Thus, the data from the primary database is on the standby database in log form, so no data loss can occur.

- Logical standby databases can be used concurrently for data protection and reporting. Although the logical standby database is open in read / write mode, its target tables for the regenerated SQL are available only in read-only mode for reporting purposes. Logical standby databases support reporting because changes are applied to the database from the redo logs using SQL statements. Although the tables built from regenerated SQL are available in read-only mode for reporting purposes, a logical standby database using additional indexes and materialized views can provide optimized query performance.

Datatype support and DML operations A physical standby database can be deployed against any Oracle production database to protect against data loss or corruption. All datatypes, tables, and DML operations are supported by a physical standby database. In this release of Oracle9*i*, some datatypes and DML operations cannot be maintained in a logical standby database. Before setting up a logical standby database for disaster recovery, you must make sure the logical standby database can maintain all of the datatypes and tables in your primary database.

> **See Also:** Section 4.1 for information about datatypes and database structures not supported by logical standby databases

Planned maintenance Both physical and logical standby databases provide switchover capabilities in which the standby database can take over processing for the primary database, and the original primary database can be transitioned into the standby role. A switchover operation can be performed without the need to re-create the new standby database from the new primary database. The primary database is then available for maintenance operations.

2.3 Method of Transmitting Redo Logs to the Standby Database

One crucial aspect of any Data Guard environment is **archiving** the redo logs from the primary database to the standby database. You configure the primary database to archive automatically to the standby database using log transport services.

When a primary database archives to a standby database, log transport services automatically transmit the online redo logs through Oracle Net to a directory on the standby system. As redo logs are generated on the primary database, log transport services automatically transmit them and log apply services apply them to each

standby database. This allows standby databases to remain synchronized with the primary database.

> **See Also:** Chapter 5, "Log Transport Services" and Chapter 6, "Log Apply Services"

2.3.1 Independence of Automatic Archiving and Log Apply Services

The mechanism for applying redo logs to a standby database is independent of the mechanism for automatic archiving of redo logs to the standby database. Thus, log transport services on the primary database can continue to archive to the standby database even if the log apply services on the standby database have been stopped, but only if the standby instance is mounted for physical standby databases or open for logical standby databases.

For example, you can take a physical standby database out of managed recovery mode and temporarily place it in read-only mode. While the physical standby database is in read-only mode, archived redo logs continue to accumulate on the standby system.

2.3.2 Optional Manual Copy of Archived Redo Logs

Even if you configure the primary database to archive automatically to the standby destination, you can still copy the completed archived redo logs manually if necessary.

For example, assume that a problem with the Oracle Net configuration prevents the copying of archived redo logs to the standby location. The primary database continues to archive locally, so you can copy the logs manually using operating system commands, then register the archived redo logs at the standby location. Log apply services automatically apply the archived redo logs to the standby database.

> **See Also:** Chapter 6, "Log Apply Services"

2.3.3 Functions of Log Transport Services

Log transport services provide the following functions:

- Control of different log archiving mechanisms
- Automatic log archiving
- Error handling and reporting
- Automatic archive gap detection and retrieval of lost logs

- Guaranteed no-data-loss data protection
- Selectable log transport modes that balance data protection needs with availability and performance
- Increased DBA and operator productivity (by eliminating the need for operator intervention)
- Ability to fetch archived logs from any number of servers, which reduces the risk that log archiving will stop because of data communication failures, out-of-space conditions, or other reasons
- Ability to configure a repository to receive but not apply archived logs

2.4 Method of Applying Redo Logs or SQL Statements

The log apply services component of the Data Guard environment is responsible for maintaining a physical standby database in either a managed recovery mode or in open read-only mode, and for applying SQL statements to a logical standby database. It coordinates activities with the log transport services.

2.4.1 Functions of Log Apply Services

Log apply services provide the following functions:

- Automatic application of archived logs
- Automatic archive gap resolution
- Enhanced monitoring capability
- Automatic delayed application of archived logs
- Partial archived log recovery (recovery from primary database failure)

2.5 Location and Directory Structure of Primary and Standby Databases

One crucial aspect of the Data Guard environment is the number and configuration of the databases involved. Of particular importance are whether:

- The primary and standby databases reside on the same computer system or on different systems

- The primary and standby systems have identical or different directory structures

2.5.1 Number and Location of Standby Databases

You can locate a standby database:

- On the same system as the primary database
- On a different system in the same data center
- On a different system in a different data center, but in the same metropolitan area
- On a different system in a data center in a geographically remote location, for example, on a different continent

The location of systems involved in the Data Guard environment has obvious implications for a disaster recovery strategy. For example, if the primary system in a data center is destroyed, then you cannot perform failover to a standby database unless it resides on a different system, which may or may not be in the same data center. In a worst case scenario, if the data center is completely destroyed, then you cannot perform a failover to a standby database unless the standby database is located on a different system in a remote location.

See Also: Section 10.1.5 for a disaster recovery scenario

2.5.2 Directory Structure of Standby Databases

The directory structure of the various standby databases is important because it determines the path names for the standby datafiles and redo logs. If you have a standby database on the same system as the primary database, you must use a different directory structure; otherwise, the standby database attempts to overwrite the primary database files.

For physical standby databases, use the same path names for the standby files if possible. This option eliminates the need to set filename conversion parameters. Nevertheless, if you need to use a system with a different directory structure or place the standby and primary databases on the same system, you can do so with a minimum of extra administration.

2.5.3 Configuration Options

The three basic configuration options are illustrated in Figure 2–1. These include:

- A standby database on the same system as the primary database that uses a different directory structure than the primary system (`Standby1`)
- A standby database on a separate system that uses the same directory structure as the primary system (`Standby2`)—this is the recommended method
- A standby database on a separate system that uses a different directory structure than the primary system (`Standby3`)

Figure 2–1 Some Possible Standby Configurations

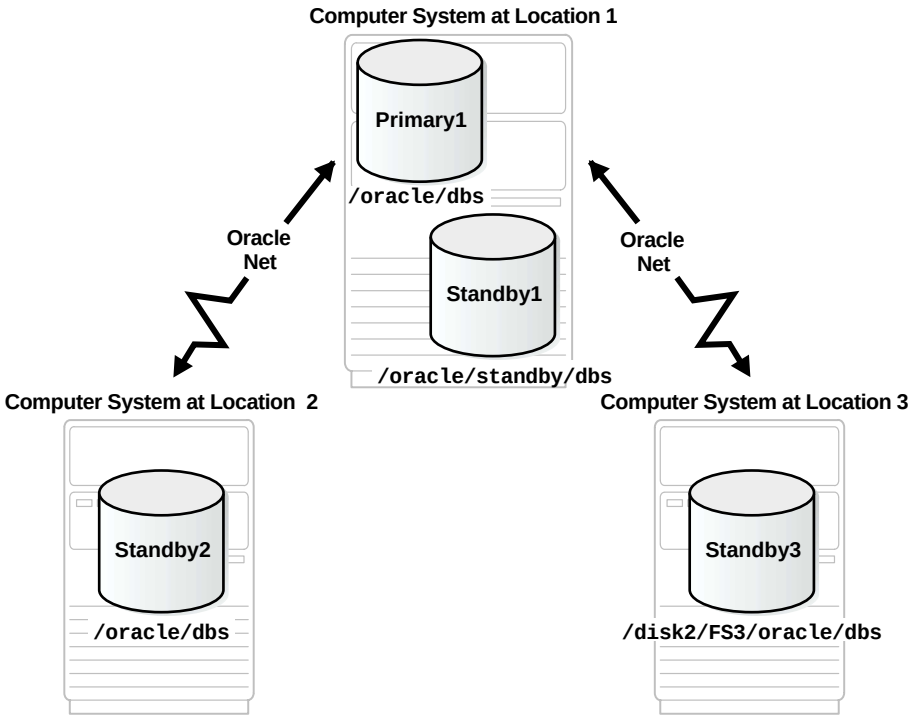


Table 2–1 describes possible configurations of primary and standby databases and the consequences of each.

Table 2–1 Primary and Standby Database Configurations

Standby System	Directory Structure	Consequences
Same as primary system	Different than primary system (required)	■ You must set the LOCK_NAME_SPACE initialization parameter. ■ You must rename primary database datafiles in the standby database control file. You can either manually rename the datafiles (see Section B.4) or set up the DB_FILE_NAME_CONVERT initialization parameter on the standby database to automatically rename the datafiles (see Section 6.3.4). ■ Some operating systems do not permit two instances with the same name to run on the same system. Refer to your Oracle operating system-specific documentation for more information. ■ The standby database does not protect against disaster.
Separate system	Same as primary system	■ You do not need to rename primary database filenames in the standby database control file, although you can still do so if you want a new naming scheme (for example, to spread the files among different disks). ■ Using separate physical media for your databases safeguards your primary data.
Separate system	Different than primary system	■ You must rename primary database datafiles in the standby database control file. You can either manually rename the datafiles (see Section B.4) or set up the DB_FILE_NAME_CONVERT initialization parameter on the standby database to automatically rename the datafiles (see Section 6.3.4). ■ Using separate physical media for your databases safeguards your primary data.

2.6 Types of Standby Databases

As database activity is recorded in the online redo logs on the primary database, the same information is transmitted to local and remote standby databases. You can set up standby databases in any of the following types of configurations:

- Physical standby database

  A physical standby database is an identical copy of the primary database. The physical standby database's on-disk database structures are identical to the primary database on a block-for-block basis. The logs are applied to physical standby databases by log apply services.

- Logical standby database

  A logical standby database is an Oracle database that contains the same user data as the primary database. However, the contents of the logical standby database may differ from the primary database by having only a subset of the tables or by including additional metadata, such as additional indexes or materialized views. The redo data in the logs is converted to SQL statements that are then applied to the database. The SQL apply mode allows for differences in the primary and standby databases.

- Archive log repository

  This type of standby database allows archival of redo logs to another system in the configuration. An archive log repository is created by using a standby control file, starting the instance, and mounting the database. This database contains no datafiles and cannot be used for primary database recovery. This alternative is useful as a way of holding redo logs for a short period of time, perhaps a day, after which the logs can then be deleted. This avoids most of the storage and processing expense of another fully configured standby database.

- Cross-instance archival database environment

  It is possible to set up a **cross-instance archival** database environment. Within a Real Application Clusters environment, each instance directs its archived redo logs to a single instance of the cluster. This instance, known as the recovery instance, is typically the instance where managed recovery is performed. The recovery instance typically has a tape drive available for RMAN backup and restore support. Typically, this type of usage requires a physical standby database.

2.7 Examples of Using Multiple Standby Databases

Because there can be several databases on one computer system, you can implement any combination of primary and standby databases in a variety of configurations. Only mission-critical databases require a standby database for disaster recovery. However, standby databases can also be used to off-load the primary database. This section provides some examples of Data Guard configurations from actual customer situations.

Figure 2–2 shows two large mission-critical databases configured on locations that are geographically separated to achieve a Data Guard disaster recovery environment. The primary database at location 1 has its standby database at location 2, and the primary database at location 2 has its standby database at location 1 to make efficient use of each system with no idle hardware. If either

primary database becomes incapacitated, the physical standby database on the other location can be failed over to the primary role so processing can continue. In addition, the use of Real Application Clusters provides the ability to scale each location with additional nodes in the future.

Figure 2–2 Standard Bidirectional Standby Database Configuration

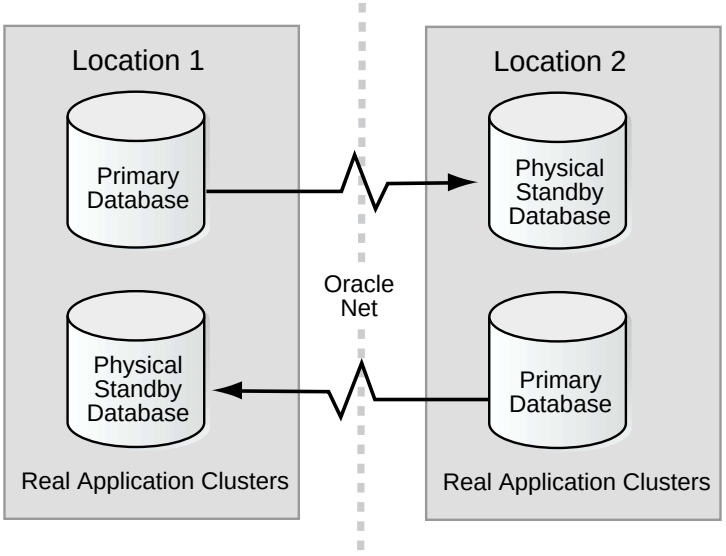


Figure 2–3 shows a Data Guard configuration that implements a primary database instance that transmits redo logs to physical and logical standby databases that are both remote from the primary database instance. In this configuration, a physical standby database is configured for disaster recovery and backup operations, and a logical standby database is configured primarily for reporting but it can also be used for disaster recovery. For straight backup and reporting operations, you could locate the standby database at the same location as the primary database. However, for disaster recovery, the standby databases must be located on remote systems.

Figure 2–3 Standby Locations Used to Off-load Backup Operations and Reporting

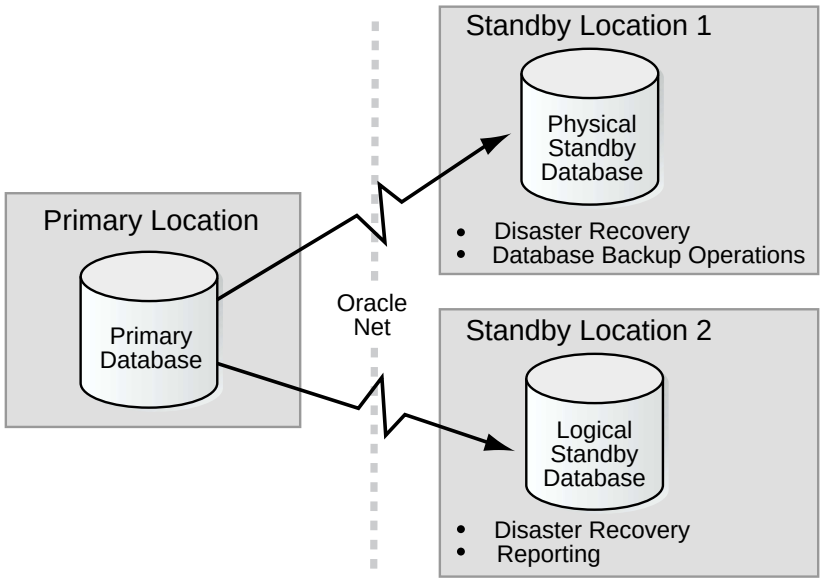
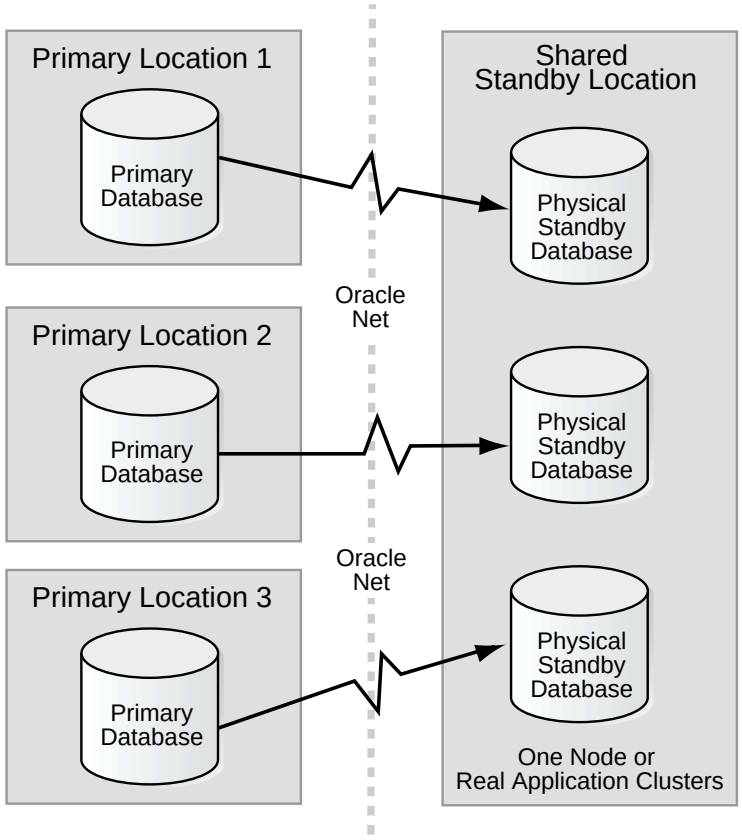


Figure 2–4 shows a configuration in which Data Guard and Real Application Clusters provide complementary Oracle features that together help you to implement high availability, disaster protection, and scalability. In the example, the organization has chosen to use one shared standby database system, hosting several standby databases for disaster recovery. This approach amortizes the hardware costs of standby systems, because several primary databases are served by one standby system.

Figure 2–4 A Single Location Shared by Multiple Standby Databases



3

Creating a Physical Standby Database

This chapter explains how to create a physical standby database and start applying redo logs to it. This chapter includes the following main topics:

- Checklist of Tasks
- Creating a Standby Database: Basic Tasks
- Creating the Standby Database Files
- Creating the Standby Initialization Parameter File

3.1 Checklist of Tasks

Table 3–1 provides a checklist of tasks that you perform to create a standby database and synchronize it so that it is ready to begin managed recovery. Each step includes a reference to a section that provides additional information.

> **Note:** Use the Create Configuration Wizard that comes with Oracle9*i* Data Guard Manager to automatically perform all of the steps described in this chapter.

Table 3–1 Task List: Preparing for Managed Recovery

Step	Task	Site	Reference
1	Either make a new backup of the primary database datafiles or access an old backup.	Primary	Section 3.3.2
2	Ensure the primary database is in ARCHIVELOG mode.	Primary	Section 3.3.3
3	Connect to the primary database and create the standby control file.	Primary	Section 3.3.3

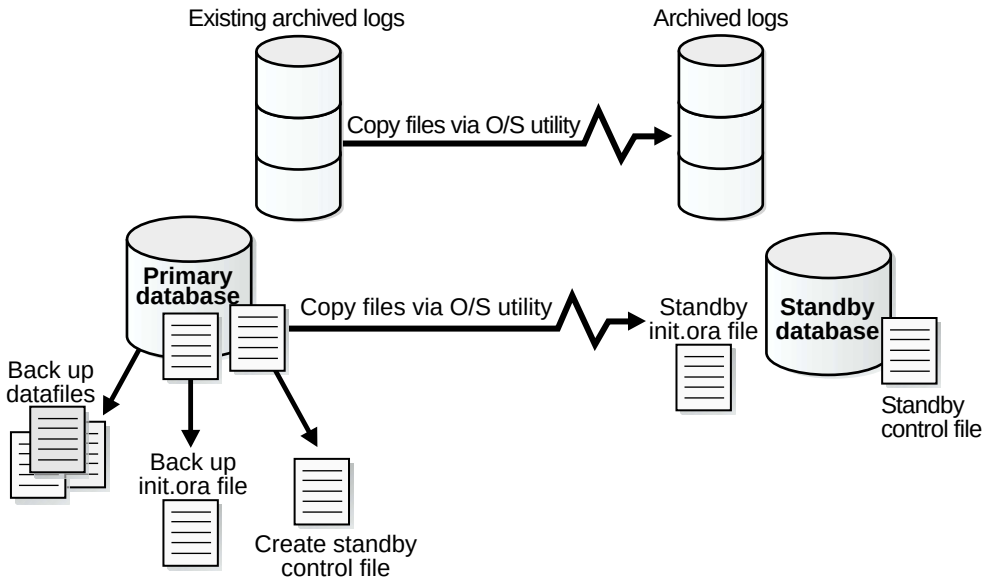
Table 3–1 (Cont.) Task List: Preparing for Managed Recovery

Step	Task	Site	Reference
4	Copy the backup datafiles and standby control file from the primary site to the standby site.	Primary	Section 3.3.4
5	Set the initialization parameters for the primary database.	Primary	Section 5.6.3
6	Create the standby initialization parameter file and set the initialization parameters for the standby database. Depending on your configuration, you may need to set filename conversion parameters.	Primary or Standby	Section 5.6.1 and Section 6.3.4
7	Use Oracle Net Manager to create a net service name that the standby database can use to connect to the primary database. The net service name must resolve to a connect descriptor that uses the same protocol, host address, port, and SID that you specified when you configured the listener on the primary database site. If you are unsure what values to use for these parameters, run Oracle Net Manager on the primary database site to display the listener configuration.	Standby	*Oracle9i Net Services Administrator's Guide*
8	Use Oracle Net Manager to create a net service name that the primary database can use to connect to the standby database. The net service name must resolve to a connect descriptor that uses the same protocol, host address, port, and SID that you specified when you configured the listener on the standby database site. If you are unsure what values to use for these parameters, run Oracle Net Manager on the standby database site to display the listener configuration.	Primary	*Oracle9i Net Services Administrator's Guide*
9	Use Oracle Net Manager to configure a **listener** on the standby database. If you plan to manage this standby database using the Data Guard broker, you must configure the listener to use the TCP/IP protocol and statically register the standby database service using its SID.	Standby	*Oracle9i Net Services Administrator's Guide*
10	Start the standby instance and mount the standby database.	Standby	Section 6.3.2
11	Create standby redo log files, if necessary.	Standby	Section 5.8.4
12	Manually change the names of the primary datafiles and redo logs in the standby control file for all files *not* automatically renamed using `DB_FILE_NAME_CONVERT` and `LOG_FILE_NAME_CONVERT` as noted in step 6.	Standby	Section B.4
13	Stop and restart the listener on the primary database, and start the listener on the standby database.	Primary and Standby	*Oracle9i Net Services Administrator's Guide*
14	Manually enable initialization parameter changes on the primary database so that it can initiate archiving to the standby site.	Primary	Section 5.3.2.2

3.2 Creating a Standby Database: Basic Tasks

Setting up a standby database for managed recovery requires you to perform a series of different tasks. After you have completed the preparation and initiated managed recovery, the standby database automatically and continuously applies redo logs as they are received from the primary database. Figure 3–1 shows the creation of a standby database.

Figure 3–1 Standby Database Creation



3.3 Creating the Standby Database Files

You can create a standby database on the same site as your primary database or on a separate site. If you create your standby database on the same site, follow the creation procedure carefully when creating the standby database files so that you do not overwrite files on the primary database.

The creation of the standby database files occurs in four stages:

1. Using Backups for Standby Creation
2. Creating the Standby Datafiles

3. Creating the Standby Control File
4. Copying Files to the Standby Site

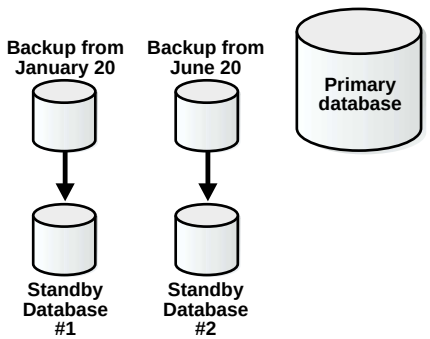
3.3.1 Using Backups for Standby Creation

Each standby database must be created from a backup of the primary database.

Note: To protect against unlogged direct writes in the primary database that cannot be propagated to the standby database, turn on `FORCE LOGGING` at the primary database before taking datafile backups for standby creation. Keep the database (or at least important tablespaces) in `FORCE LOGGING` mode as long as the standby database is active.

You can also use a single backup of the primary database to create up to nine standby databases, although the various standby databases in the environment do not have to be created from the same backup. (You can *create* an unlimited number of standby databases, but a single primary database can only *support* up to nine standby databases at a time.) Figure 3–2 shows that you can create one standby database from a backup of the primary database taken on January 20 and create another standby database from the backup taken on June 20. So long as you have the archived redo logs required to perform complete recovery of a backup, it can serve as the basis for a standby database.

Figure 3–2 Creating Standby Databases Using Different Backups



3.3.2 Creating the Standby Datafiles

First, make backups of your primary database datafiles. You create the standby datafiles from these backups.

You can use any backup of the primary database so long as you have archived redo logs to completely recover the database. The backup can be old or new, consistent or inconsistent. **Hot backups** (or **open backups**) allow you to keep the database open while performing the backup. Nevertheless, you may prefer to make a new **closed, consistent backup** using the Recovery Manager utility (RMAN) to prevent the application of a large number of archived redo logs.

To make a consistent, whole database backup to serve as the basis for the standby database:

1. Start a SQL session on your primary database and query the V$DATAFILE fixed view to obtain a list of the primary datafiles. For example, enter:

```
SQL> SELECT NAME FROM V$DATAFILE;
NAME
----------------------------------------------------------------------------
/oracle/dbs/tbs_01.dbf
/oracle/dbs/tbs_02.dbf
/oracle/dbs/tbs_03.dbf
/oracle/dbs2/tbs_11.dbf
/oracle/dbs2/tbs_12.dbf
/oracle/dbs3/tbs_21.dbf
/oracle/dbs3/tbs_22.dbf
7 rows selected.
```

2. Shut down the primary database cleanly:

```
SQL> SHUTDOWN;
```

3. Make a consistent backup of the datafiles from your primary database using the Recovery Manager utility (RMAN) or an operating system utility.

4. Reopen the primary database. For example, enter:

```
SQL> STARTUP PFILE=initPRIMARY1.ora;
```

See Also: *Oracle9i Recovery Manager User's Guide* to learn how to create a standby database with Recovery Manager

3.3.3 Creating the Standby Control File

After you have created the backups that will be used as the standby datafiles, you can create the standby database **control file.** The control file must be created after the latest timestamp for the backup datafiles.

Note: You cannot use a single control file for both the primary and standby databases. The standby instance is independent from the primary instance and so requires exclusive possession of its database files.

To create the standby database control file:

1. Ensure that the primary database is in ARCHIVELOG mode and that archiving is enabled. Either issue the SQL*Plus `ARCHIVE LOG LIST` command or query the `V$DATABASE` view. Take the following steps:

   a. Start and mount the primary database without opening it. For example:

   ```
   SQL> STARTUP MOUNT PFILE=initPRIMARY1.ora
   ```

   b. Issue the SQL*Plus `ARCHIVE LOG LIST` command to determine if the database is in ARCHIVELOG mode. For example:

   ```
   SQL> ARCHIVE LOG LIST;
   Database log mode              No Archive Mode
   Automatic archival             Disabled
   Archive destination            /oracle/dbs/arch
   Oldest online log sequence     0
   Current log sequence           1
   ```

   c. If the database is not in ARCHIVELOG mode, as shown in step b, issue the following command to place the database in ARCHIVELOG mode:

   ```
   SQL> ALTER DATABASE ARCHIVELOG;
   ```

   d. You can issue the SQL*Plus `ARCHIVE LOG LIST` command again to verify the database has been placed in ARCHIVELOG mode. For example:

   ```
   SQL> ARCHIVE LOG LIST;
   Database log mode              Archive Mode
   ```

```
Automatic archival            Disabled
Archive destination           /oracle/dbs/arch
Oldest online log sequence    0
Next log sequence to archive  1
Current log sequence          1
```

To enable the automatic archiving of the online redo logs, you must set LOG_ARCHIVE_START=true in the initialization parameter file. However, this does not have to be done before you create the standby control file.

> **See Also:** *SQL*Plus User's Guide and Reference* for additional information on the ARCHIVE LOG LIST command and *Oracle9i Database Administrator's Guide* for additional information on the ALTER DATABASE ARCHIVELOG statement and the LOG_ARCHIVE_START initialization parameter

2. Connect to the primary database and create the control file for your standby database. For example, to create the standby control file as /oracle/dbs/stbycf.ctl on the primary site, enter the following:

```
SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS '/oracle/dbs/stbycf.ctl';
```

The filename for the created standby control file *must* be different from the filename of the **current control file** of the primary database. You can also use RMAN to create the standby database control file.

> **See Also:** *Oracle9i SQL Reference* for additional information on the ALTER DATABASE statement and *Oracle9i Recovery Manager User's Guide* for additional information on RMAN

3.3.4 Copying Files to the Standby Site

After you have successfully created the standby datafiles and control file, copy the files to the standby site using an operating system utility.

If the standby database is on	Then you
A separate site with the same directory structure as the primary database	Can use the same path names for the standby files as the primary files. In this way, you do not have to rename the primary datafiles in the standby control file.

If the standby database is on	Then you
The same site as the primary database, or the standby database is on a separate site with a different directory structure	Must rename the primary datafiles in the standby control file after copying them to the standby site. You can: ■ Set the filename conversion initialization parameters. See Section 6.3.4. ■ Rename the files manually using ALTER DATABASE statements. See Section B.4. ■ Use a combination of conversion parameters and manual renames.

Use operating system commands or utilities to perform the following copy operations to the standby location:

1. Copy the standby control file.
2. Copy the backup datafiles.
3. Copy all available archived redo logs to the standby site.
4. Copy the online redo logs. This is recommended for switchover and failover operations.

Do not copy temporary tablespaces. Use an appropriate method for copying binary files.

> **See Also:** Section 6.3.6 for more information about creating temporary tablespaces and Section 10.1.2 for a scenario showing how to copy files to a standby site

3.4 Creating the Standby Initialization Parameter File

Once you have configured the primary database initialization parameter file, you can duplicate the file for use by the standby database. The procedure for creating the standby initialization parameter file is as follows:

1. Copy the initialization parameter file for the primary database using an operating system utility.
2. Edit the initialization parameter file for use by the standby database.
3. Transfer the initialization parameter file to the standby site using an appropriate operating system utility.

> **See Also:** Section 5.6.1, Section 6.3.4, and Section 10.1.2

4

Creating a Logical Standby Database

This chapter explains how to create a logical standby database and start applying redo logs to it. This chapter includes the following main topics:

- Preparing to Create a Logical Standby Database
- Creating a Logical Standby Database

> **Note:** Use the Create Configuration Wizard that comes with Oracle Data Guard Manager to automatically perform all of the steps described in this chapter.

4.1 Preparing to Create a Logical Standby Database

This section describes the steps that you must perform before you can create a logical standby database. Table 4–1 provides a checklist of tasks that you need to perform.

Table 4–1 Checklist of Preparatory Tasks to Create a Logical Standby Database

Step	Task	Site
1	Determine if the primary database contains datatypes or tables that are not supported by a logical standby database.	Primary
2	Ensure that table rows in the primary database can be uniquely identified.	Primary
3	Ensure that the primary database is running in ARCHIVELOG mode.	Primary
4	Enable supplemental logging on the primary database.	Primary
5	Start the Resource Manager if you plan to perform a hot backup.	Primary

Table 4–1 (Cont.) Checklist of Preparatory Tasks to Create a Logical Standby

Step	Task	Site
6	Create an alternate tablespace in the primary database for logical standby system tables.	Primary

The following steps describe these tasks in more detail.

Step 1 Determine if the primary database contains datatypes or tables that are not supported by a logical standby database.

Before setting up a logical standby database, you must make sure the logical standby database can maintain the datatypes and tables in your primary database.

The following lists indicate which of the various database objects are supported in logical standby databases.

Supported Datatypes

CHAR
NCHAR
VARCHAR2 and VARCHAR
NVARCHAR2
NUMBER
DATE
TIMESTAMP
TIMESTAMP WITH TIME ZONE
TIMESTAMP WITH LOCAL TIME ZONE
INTERVAL YEAR TO MONTH
INTERVAL DAY TO SECOND
RAW
CLOB
BLOB

Unsupported Datatypes

Logical standby databases do not support columns of the following datatypes: NCLOB, LONG, LONG RAW, BFILE, ROWID, and UROWID, and user-defined type (including object types, REFs, varrays, and nested tables). An error is returned if a logical standby database attempts to apply DML changes for a table that contains a column of an unsupported datatype.

Unsupported Tables and Sequences

Tables and sequences in the SYS schema
Tables with unsupported datatypes
Tables used to support functional indexes
Tables used to support materialized views
Global temporary tables

To determine whether your primary database contains unsupported objects, use the DBA_LOGSTDBY_UNSUPPORTED view. For example, enter the following SELECT statement on the primary database to list the names of primary database tables (and columns and datatypes in those tables) that are not supported by logical standby databases:

```
SQL> SELECT * FROM DBA_LOGSTDBY_UNSUPPORTED;
```

If the primary database contains unsupported tables, log apply services automatically exclude the tables when applying redo logs to the logical standby database.

> **See Also:** Chapter 14, "Views" for more information about the DBA_LOGSTDBY_UNSUPPORTED view

Step 2 Ensure that table rows in the primary database can be uniquely identified.

To maintain data in a logical standby database, SQL apply operations must be able to identify the columns that uniquely identify each row that has been updated in the primary database. Oracle Corporation recommends that you use primary keys in your primary database to ensure that SQL apply operations can efficiently and correctly apply data updates to a logical standby database.

Most tables already have a primary key or a non-null unique index, and, if not, the **supplemental logging** feature (that you will enable in step 4) automatically gathers the information necessary to identify rows that have been updated in the primary database. With supplemental logging, information that will uniquely identify the row is added to every update transaction in the archived redo logs so that log apply services can properly maintain tables in the logical standby database.

To ensure that SQL apply operations can uniquely identify table rows, perform the following steps:

1. Use the DBA_LOGSTDBY_NOT_UNIQUE view to identify the tables that have no primary key or non-null unique constraints. For example:

   ```
   SQL> SELECT OWNER, TABLE_NAME, BAD_COLUMN FROM DBA_LOGSTDBY_NOT_UNIQUE;
   ```

This query returns the schema name, the table name, and a Y or N value in the BAD_COLUMN column:

- A value of Y indicates the table column is defined using an unbounded datatype, such as LONG. If two rows in the table match except in their LOB column, then the table cannot be maintained properly.

  SQL apply operations of log apply services maintain these tables, but you must take extra steps to ensure that the application does not provide uniqueness only in unbounded columns; applications must provide uniqueness in rows in columns other than in unbounded columns.

- A value of N indicates the table contains enough column information to maintain the table in the logical standby database. However, the log transport services and SQL apply operations will run more efficiently if you add a primary key. You should consider adding a disabled RELY constraint to these tables.

  See Also: Chapter 14, "Views" for more information about the DBA_LOGSTDBY_NOT_UNIQUE view and *Oracle9i SQL Reference* for more information about creating RELY constraints

2. If necessary, define a primary key to improve performance.

   To maintain a table in a logical standby database, the supplemental logging function automatically adds column data to the redo log for every update performed on that table, as follows:

   - If the table has a primary key or a unique index with a non-null column, the amount of information added to the redo log is minimal.
   - If the table does not have a primary key, supplemental logging automatically creates a unique key by adding all scalar values for each row to the redo log. However, this automated key creation will result in an increase in the amount of information written to the redo logs.

3. If necessary, create a disabled RELY constraint on the table.

   If the tables identified by the DBA_LOGSTDBY_NOT_UNIQUE view are updated frequently, you can improve redo log performance by creating a disabled RELY constraint to avoid the associated overhead of maintaining a primary key. A disabled RELY constraint provides more information to SQL apply operations about the table, but the constraint does not incur the overhead cost of an index on the primary database.

The following example shows how to create a disabled RELY constraint on a table named mytab where rows can be uniquely identified using the id and name columns.

```
SQL> ALTER TABLE mytab ADD PRIMARY KEY (id, name) RELY DISABLE;
```

The RELY constraint tells the system to log the id and name columns to identify rows in this table. Be careful to select columns for the disabled RELY constraint that will create a primary key. If the columns selected for the RELY constraint do not make a primary key for that table, SQL apply operations will fail to apply redo information to the logical standby database.

> **See Also:** Chapter 14, "Views" for more information about the DBA_LOGSTDBY_NOT_UNIQUE view and *Oracle9i SQL Reference* for more information about creating RELY constraints

Step 3 Ensure that the primary database is running in ARCHIVELOG mode.

Ensure that the primary database is in ARCHIVELOG mode and that archiving is enabled. Either issue the SQL*Plus ARCHIVE LOG LIST command or query the V$DATABASE view. Take the following steps:

1. Issue the SQL*Plus ARCHIVE LOG LIST command to determine if the database is in ARCHIVELOG mode. For example:

```
SQL> ARCHIVE LOG LIST;
Database log mode              No Archive Mode
Automatic archival             Disabled
Archive destination            /oracle/dbs/arch
Oldest online log sequence     0
Current log sequence           1
```

2. If the database is not in ARCHIVELOG mode, as shown in step 1, issue the following command to place the database in ARCHIVELOG mode:

```
SQL> ALTER DATABASE ARCHIVELOG;
```

3. Start and mount the primary database without opening it. For example:

```
SQL> SHUTDOWN
SQL> STARTUP MOUNT PFILE=initPRIMARY1.ora
```

4. Open the database:

```
SQL> ALTER DATABASE OPEN;
```

5. You can issue the SQL*Plus ARCHIVE LOG LIST statement again to verify the database has been placed in ARCHIVELOG mode. For example:

```
SQL> ARCHIVE LOG LIST;
Database log mode              Archive Mode
Automatic archival             Disabled
Archive destination            /oracle/dbs/arch
Oldest online log sequence     0
Next log sequence to archive   1
Current log sequence           1
```

To enable automatic archiving of the online redo logs, you must set LOG_ARCHIVE_START=true in the initialization parameter file.

> **See Also:** *SQL*Plus User's Guide and Reference* for additional information about the ARCHIVE LOG LIST command and *Oracle9i Database Administrator's Guide* for additional information about the ALTER DATABASE ARCHIVELOG statement and the LOG_ARCHIVE_START initialization parameter

Step 4 Enable supplemental logging on the primary database.

The supplemental information helps SQL apply operations to correctly maintain your tables in the logical standby database. To verify whether supplemental logging is enabled, start a SQL session and query the V$DATABASE fixed view. For example, enter:

```
SQL> SELECT SUPPLEMENTAL_LOG_DATA_PK, SUPPLEMENTAL_LOG_DATA_UI FROM V$DATABASE;
SUP SUP
--- ---
YES YES

SQL>
```

View the SUPPLEMENTAL_LOG_DATA_PK, and SUPPLEMENTAL_LOG_DATA_UI columns. If supplemental logging is not enabled, execute the following statements on the primary database to add primary key and unique index information to the archived redo logs, and to switch to a new redo log.

Note: You must enable supplemental logging before you create the logical standby database. This is because the logical standby database cannot use archived redo logs that contain both supplemental log data and nonsupplemental log data.

For example:

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY, UNIQUE INDEX) COLUMNS;
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

> **Note:** If you enable supplemental logging on your primary database and you have already created physical standby databases, then you must perform the same command on each physical standby database to ensure that future switchovers work correctly.

In addition to supplemental logging, you must also ensure that the LOG_PARALLELISM initialization parameter is set to 1 (this is the default). Set this parameter as shown in the following example to make sure that the correct redo information is sent to the logical standby database:

```
SQL> ALTER SYSTEM SET LOG_PARALLELISM=1 SCOPE=BOTH;
```

If you are not using the server parameter file (SPFILE) option, you must modify your initialization parameter file after executing the ALTER SET statement. You do not have to restart your primary database.

> **See Also:** Chapter 14, "Views" for more information about the V$DATABASE view and the *Oracle9i SQL Reference* for more information about the ALTER DATABASE ADD SUPPLEMENTAL LOG DATA and ALTER SET statements

Step 5 Start the Resource Manager if you plan to perform a hot backup.

If you plan to create the logical standby database using a hot backup copy of the primary database, you must start the Resource Manager. To do this, define the RESOURCE_MANAGER_PLAN initialization parameter to use a resource plan and then shut down and start up the primary database.

If you do not have a resource plan, you can use one of the supplied plans by defining the SYSTEM_PLAN attribute. If the RESOURCE_MANAGER_PLAN initialization parameter was not defined when you started the instance, you must restart your primary database to ensure that the Resource Manager is running. However, this can still be faster than performing a cold backup procedure.

The following example shows how to set the RESOURCE_MANAGER_PLAN initialization parameter:

```
SQL> ALTER SYSTEM SET RESOURCE_MANAGER_PLAN=SYSTEM_PLAN SCOPE=BOTH;
SQL> SHUTDOWN
```

```
SQL> STARTUP
```

Step 6 Create an alternate tablespace in the primary database for logical standby system tables.

This step is necessary only if you expect to perform switchover operations in which the primary site changes roles with a standby site in the configuration.

Logical standby databases use a number of tables defined in the SYS and SYSTEM schemas. By default, these tables are created in the SYSTEM tablespace.

> **Note:** Some of these tables can rapidly become very large. To prevent these tables from filling the entire SYSTEM tablespace, you must move the tables to a separate tablespace. Move the tables to the new tablespace before they are populated during the logical standby creation process.

On the primary database, use the SQL CREATE TABLESPACE statement to create a new tablespace for the logical standby tables and use the DBMS_LOGMNR_D.SET_TABLESPACE procedure to move the tables into the new tablespace. The following example shows how to create a new tablespace named logmnrts$ and move the logical standby tables into that tablespace:

```
SQL> CREATE TABLESPACE logmnrts$ DATAFILE '/usr/prod1/dbs/logmnrts.dbf'
  2> SIZE 25 M AUTOEXTEND ON MAXSIZE UNLIMITED;
SQL> EXECUTE DBMS_LOGMNR_D.SET_TABLESPACE('logmnrts$');
```

> **See Also:** *Oracle9i SQL Reference* for complete information about the CREATE TABLESPACE statement

4.2 Creating a Logical Standby Database

This section describes the tasks you must perform to set up and create a logical standby database. Table 4–2 provides a checklist of the tasks that you perform to create a logical standby database and the site on which you perform each step.

Table 4–2 Checklist of Tasks to Create a Logical Standby Database

Step	Task	Site
1	Identify primary database datafiles.	Primary
2	Make a copy of the primary database.	Primary

Table 4–2 (Cont.) Checklist of Tasks to Create a Logical Standby Database

Step	Task	Site
3	Modify the initialization parameter file for the logical standby site.	Standby
4	Start and mount the logical standby database.	Standby
5	Rename the datafiles and redo logs on the logical standby database.	Standby
6	Create the online log files.	Standby
7	Recover the logical standby database.	Standby
8	Turn on the database guard.	Standby
9	Reset the database name of the logical standby database.	Standby
10	Open the logical standby database.	Standby
11	Drop the current temporary files from the logical standby database.	Standby
12	Create a new temporary file for the logical standby database.	Standby
13	Register the starting archived redo log with log apply services.	Standby
14	Start applying redo log data to the logical standby database.	Standby
15	Use Oracle Net Manager to configure a listener on the logical standby database.	Standby
16	Create a net service name that the logical standby database can use to connect to the primary database.	Standby
17	Create a net service name that the primary database can use to connect to the logical standby database.	Primary
18	Stop and restart the listeners on both systems.	Primary and standby
19	Enable archiving to the logical standby database.	Primary
20	Start archiving the current redo logs.	Primary
21	Create a database link to the primary database.	Standby
22	Create a database link to the logical standby database.	Primary

The tasks to create a logical standby database are presented in step-by-step order, with steps alternating between the primary and logical standby databases.

On the primary database:

Step 1 Identify primary database datafiles.

Query the V$DATAFILE view to get a list of files that will be used to create the logical standby database.

```
SQL> SELECT NAME FROM V$DATAFILE;
NAME
--------------------------------------------------------------------------------
/oracle/oratmp/system01.dbf
/oracle/oratmp/undotbs01.dbf
/oracle/oratmp/cwmlite01.dbf
.
.
.
```

Step 2 Make a copy of the primary database.

If you can afford to shut down the primary database for the length of time it takes to copy the datafiles, perform a cold backup operation using the steps described in the following "Using a Cold Backup of the Primary Database" section. Otherwise, skip to "Using a Hot Backup of the Primary Database" on page 4-11.

Using a Cold Backup of the Primary Database

Perform the following steps to create a copy of the primary database:

1. Shut down the primary database:

   ```
   SQL> SHUTDOWN;
   ```

2. Copy the files that you identified in step 1 to a temporary location. For example:

   ```
   cp /oracle/oratmp/system01.dbf /disk1/stdbyhold/system01.dbf
   ```

   This will reduce the amount of time that the primary database remains shut down.

3. Restart the primary database in mount mode:

   ```
   SQL> STARTUP MOUNT;
   ```

4. Create a backup copy of the control file. For example:

   ```
   SQL> ALTER DATABASE BACKUP CONTROLFILE TO '/disk1/stdbyhold/ctlbckup01.ctl';
   ```

5. Determine the current SCN in the primary database and record the number for future use:

```
SQL> SELECT CHECKPOINT_CHANGE# FROM V$DATABASE;
CHECKPOINT_CHANGE#
------------------
            443582
```

6. Open the primary database and switch log files:

```
SQL> ALTER DATABASE OPEN;
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

7. Use an operating system copy utility to copy the following files to the logical standby site:
   - Copy the files from the temporary location on the primary database site to the proper locations on the logical standby site that you identified in step 1 "Identify primary database datafiles."
   - Copy the database initialization parameter file and database password files from the primary database site to the proper locations on the logical standby site.
   - Copy the last archived redo log file generated by the `ALTER SYSTEM SWITCH LOGFILE` statement in step 6 to the archived redo log directory on the standby site. If you change the archived redo log format for the standby database at a later time, you must also rename this file on the standby site to match.
8. Go to step 3 "Modify the initialization parameter file for the logical standby site."

Using a Hot Backup of the Primary Database

To perform a hot backup copy of the primary database, you must first ensure that the Resource Manager is running on the primary database when the instance is started. For information about starting the Resource Manager, see step 5 "Start the Resource Manager if you plan to perform a hot backup."

1. Back up your datafiles online by placing each tablespace in backup mode and copying the datafiles. For example;

```
SQL> ALTER TABLESPACE SYSTEM BEGIN BACKUP;
```

2. Copy the datafile `/oracle/oratmp/system01.dbf` from the primary database site to the proper location on the logical standby site. Use an operating system copy utility to copy the files.
3. Issue the following statement for each tablespace:

   ```
   SQL> ALTER TABLESPACE SYSTEM END BACKUP;
   ```

4. Create a backup copy of the control file. For example:

   ```
   SQL> ALTER DATABASE BACKUP CONTROLFILE TO '/disk1/stdbyhold/ctlbckup01.ctl';
   ```

5. Copy the backup control file, the database initialization parameter file, and the database password files from the primary database site to the proper locations on the logical standby site using an operating system copy utility.
6. Put the primary database in a quiesced state to obtain a starting point for building the standby database.

   Putting the database in a quiesced state allows you to perform administrative operations that cannot be safely performed in the presence of concurrent transactions, queries, or PL/SQL operations. This step may take some time depending on how long it takes the system to put all active sessions in this state. (See the *Oracle9i SQL Reference* for more information about the `ALTER SYSTEM` statement.)

   The following example shows the `ALTER SYSTEM QUIESCE` statement and a query being performed on the `V$DATABASE` view to obtain the starting SCN. Record the SCN number for later use. Then, take the database out of the quiesce state and switch log files, as shown:

   ```
   SQL> ALTER SYSTEM QUIESCE;
   SQL> SELECT CHECKPOINT_CHANGE# FROM V$DATABASE;

   CHECKPOINT_CHANGE#
   ------------------
               443582

   SQL> ALTER SYSTEM UNQUIESCE;
   SQL> ALTER SYSTEM SWITCH LOGFILE;
   ```

   In addition to the datafiles copied previously, you must also copy all archived redo logs that were created during the hot backup procedure (up to the end of the database quiesce) to the standby location. Use an operating system copy utility to copy the archived redo logs to the standby system and apply them to the backup copy of the database.

7. Go to step 3 "Modify the initialization parameter file for the logical standby site."

On the standby database:

Step 3 Modify the initialization parameter file for the logical standby site.

Most of the initialization parameters settings in the file that you copied from the primary database are also appropriate for the logical standby database. (Chapter 11 lists the initialization parameters that are specific to the Data Guard environment, most of which pertain to both physical and logical standby databases.) However, you should modify the initialization parameter file for the logical standby site to:

1. Change the path names and filenames of the primary control files to identify the locations of the control files for the logical standby database.
2. Add the STANDBY_ARCHIVE_DEST initialization parameter to specify a valid directory on the standby system for the archived redo logs that will be transmitted from the primary database.
3. Format the filenames of the archived redo logs to include a thread and sequence number. Specify the LOG_ARCHIVE_FORMAT initialization parameter to make the sequence number part of the filename for the archived redo log. For example, you might specify LOG_ARCHIVE_FORMAT=log%t_%s.arc. Later in the creation process, you will be asked to specify this filename as the starting archived redo log to be applied to the new logical standby database.
4. Configure initialization parameters properly for SQL apply operations. Refer to the initialization parameters described in Chapter 11.

   If you are creating your logical standby database on the same system as your primary database, you must change the INSTANCE_NAME parameter to something other than its original value and add in the LOCK_NAME_SPACE initialization parameter, setting it to the same value that you specified for the INSTANCE_NAME initialization parameter. You can later remove these parameters after you have completed step 9 "Reset the database name of the logical standby database." At this point, you cannot change the DBNAME initialization parameter.

Step 4 Start and mount the logical standby database.

On the standby database, use the STARTUP statement to start and mount the logical standby database. The database should remain closed to user access.

```
SQL> STARTUP PFILE=init$Log1.ora EXCLUSIVE MOUNT;
```

Step 5 Rename the datafiles and redo logs on the logical standby database.

Rename the datafiles that were copied from the primary database and the online redo logs to indicate their new location. On the logical standby database, use the ALTER DATABASE RENAME FILE statement to rename all datafiles and log files that were copied from the primary database:

```
SQL> ALTER DATABASE RENAME FILE '/oracle/oratmp/system01.dbf'
  2> TO '/oraHome/oratmp/system01.dbf';
SQL> ALTER DATABASE RENAME FILE '/oracle/oratmp/redo01.dbf'
  2> TO '/oraHome/oratmp/redo01.dbf';
```

Step 6 Create the online log files.

Because the online redo log files were not copied over from the primary database system, the following command will create them in the directory specified by the RENAME statement. Execute the following statement for each log file group in the database:

```
SQL> ALTER DATABASE CLEAR LOGFILE GROUP 1;
```

Step 7 Recover the logical standby database.

You must recover the archived redo logs that you copied from the primary site in step 2, including the last archived redo log that was generated by the ALTER SYSTEM SWITCH LOGFILE statement. For example, if you copied the archived redo log to the '/fs1/logi/arch' directory on the standby site, then perform the following command using the SCN you recorded previously:

```
SQL> ALTER DATABASE RECOVER AUTOMATIC FROM '/fs1/logi/arch' DATABASE
  2> UNTIL CHANGE 443582 USING BACKUP CONTROLFILE;
```

Step 8 Turn on the database guard.

Turn on the database guard to prevent users from updating objects in the logical standby database using the following SQL statements:

```
SQL> ALTER DATABASE GUARD ALL;
SQL> ALTER DATABASE OPEN RESETLOGS;
SQL> SHUTDOWN IMMEDIATE
```

You can bypass the database guard by using the DBMS_LOGSTDBY.GUARD_BYPASS_ON PL/SQL procedure.

Step 9 Reset the database name of the logical standby database.

Use the DBNEWID (nid) utility to change the database name of the logical standby database. Changing the name prevents any interaction between this copy of the

primary database and the original primary database. The following example starts and mounts the database, changes the database name, and shuts down the standby database:

```
SQL> STARTUP PFILE=init$Log1.ora EXCLUSIVE MOUNT;
nid TARGET=SYS/CHANGE_ON_INSTALL DBNAME=Log1 SETNAME=YES
Connected to database PRIM (DBID=1456557175)

Control Files in database:
    /private2/ade/oradata/log2/t_cf1.f
Change database ID and database name PRIM to LOG2? (Y/[N]) => y

Proceeding with operation
Changing database ID from 1456557175 to 416458362
Changing database name from PRIM to LOG2
    Control File /private2/ade/oradata/log2/t_cf1.f - modified
    Datafile /private2/ade/oradata/log2/t_db1.f - dbid changed, wrote new name
    Datafile /private2/ade/oradata/log2/log_mnrts.f-dbid changed, wrote new name
    Control File /private2/ade/oradata/log2/t_cf1.f-dbid changed, wrote new name

Database name changed to LOG2.
Modify parameter file and generate a new password file before restarting.
Database ID for database LOG2 changed to 416458362.
All previous backups and archived redo logs for this database are unusable.
Shut down database and open with RESETLOGS option.
Successfully changed database name and ID.
DBNEWID - Completed successfully.

SQL> SHUTDOWN IMMEDIATE
```

> **Note:** You must change the DBNAME initialization parameter to match the changed database name.

Step 10 Open the logical standby database.

Enter the following statement to start and open the database to user access:

```
SQL> STARTUP PFILE=init$Log1.ora EXCLUSIVE MOUNT;
SQL> ALTER DATABASE OPEN RESETLOGS;
```

Step 11 Drop the current temporary files from the logical standby database.

This step removes the current temporary files (tempfiles). The **tempfiles,** which were included as a part of the **cold backup** operation of the primary database, are

not viable on the logical standby database. To identify and drop obsolete tempfiles, perform the following steps on the logical standby database:

1. Identify the current temporary files for the standby database:

```
SQL> SELECT * FROM V$TEMPFILE;
```

2. Drop each current temporary file from the standby database:

```
SQL> ALTER DATABASE TEMPFILE 'tempfilename' DROP;
```

Step 12 Create a new temporary file for the logical standby database.

Perform the following SQL statements on the logical standby database to add a new temporary file to the tablespace on the logical standby database:

1. Identify the tablespace that should contain the tempfile:

```
SQL> SELECT TABLESPACE_NAME FROM DBA_TABLESPACES WHERE
  2> CONTENTS ='TEMPORARY';
```

2. Add a new tempfile:

```
SQL> ALTER TABLESPACE tablespacename ADD TEMPFILE 'tempfilename'
  2> SIZE 20M REUSE;
```

Step 13 Register the starting archived redo log with log apply services.

You must register the first redo log at which SQL apply operations of log apply services can begin to apply data to the logical standby database. Do this by executing the `ALTER DATABASE REGISTER LOGICAL LOGFILE` statement on the logical standby database. When you execute this statement, specify the filename and location of the most recently archived redo log that you copied to the logical standby site. For example:

```
SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE '/Oracle/remote_arc/db1/arc0.arc';
```

Step 14 Start applying redo log data to the logical standby database.

Use the following `ALTER DATABASE` statement and include the `INITIAL` keyword to begin SQL apply operations for the first time on the logical standby database. Include the starting SCN that you obtained during step 5 on page 4-11. For example:

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY INITIAL 443582;
```

Step 15 Use Oracle Net Manager to configure a listener on the logical standby database.

On the logical standby site, use Oracle Net Manager to configure a listener for the logical standby database.

See Also: *Oracle9i Net Services Administrator's Guide*

Step 16 Create a net service name that the logical standby database can use to connect to the primary database.

On the logical standby site, use Oracle Net Manager to create a net service name that the logical standby database can use to connect to the primary database.

The net service name must resolve to a connect descriptor that uses the same protocol, host address, port, and SID that you specified when you configured the listener on the primary database site. If you are unsure what values to use for these parameters, run Oracle Net Manager on the primary database site to display the listener configuration.

See Also: *Oracle9i Net Services Administrator's Guide*

On the primary database:

Step 17 Create a net service name that the primary database can use to connect to the logical standby database.

On the primary database, use Oracle Net Manager to create a net service name that the primary database can use to connect to the logical standby database. The net service name must resolve to a connect descriptor that uses the same protocol, host address, port, and SID that you specified when you configured the listener on the logical standby database site. If you are unsure what values to use for these parameters, run Oracle Net Manager on the logical standby database site to display the listener configuration.

See Also: *Oracle9i Net Services Administrator's Guide* and the *Oracle9i Database Administrator's Guide*

On both the primary and standby databases:

Step 18 Stop and restart the listeners on both systems.

First, start the standby listener by entering the following command on the standby site:

```
% LSNRCTL START standby1_listener
```

Normally, your default listener is already started on your primary site. Restart the default listener on the primary database to pick up the new definitions. Enter the following commands on your primary site:

```
% LSNRCTL STOP
% LSNRCTL START
```

On the primary database:

Step 19 Enable archiving to the logical standby database.

On the primary database, define and enable archiving by setting the LOG_ARCHIVE_DEST_*n* and the LOG_ARCHIVE_DEST_STATE_*n* initialization parameters. For example:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=Log1, LGWR';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

Step 20 Start archiving the current redo logs.

This step is optional on a busy system. However, to ensure that your configuration is correctly set up:

1. Issue the following statement on the primary database to start archiving redo logs:

   ```
   SQL> ALTER SYSTEM SWITCH LOGFILE;
   ```

2. Monitor the initialization step of log apply services to the logical standby database.

On the standby database:

Step 21 Create a database link to the primary database.

Perform the following tasks to define a database link to the primary database that will be used during switchover operations:

1. On the logical standby database, create a database link to the primary database. Begin by using the DBMS_LOGSTDBY.GUARD_BYPASS_ON procedure to bypass the database guard and allow modifications to the tables in the logical standby database. For example:

   ```
   SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_ON;
   SQL> CREATE DATABASE LINK primary CONNECT TO SYSTEM IDENTIFIED BY manager
   ```

```
  2> USING 'prod1';
SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_OFF;
```

See Also: *Oracle9i Supplied PL/SQL Packages and Types Reference* for more information about the DBMS_LOGSTDBY package

2. Verify that the database link has been configured correctly by executing the DBMS_LOGSTDBY.UNSKIP procedure, as follows:

```
SQL> EXECUTE DBMS_LOGSTDBY.UNSKIP@primary('DML','test','test');
```

If an error message is returned, it indicates there is a problem with the configuration.

On the primary database:

Step 22 Create a database link to the logical standby database.

Perform the following tasks to define a database link to the standby database that will be used during switchover operations:

1. On the primary database, create a database link to the standby database:

```
SQL> CREATE DATABASE LINK standby CONNECT TO SYSTEM IDENTIFIED BY manager
  2> USING 'prod2';
```

2. Verify that the database link has been configured correctly by executing the DBMS_LOGSTDBY.UNSKIP procedure, as follows:

```
SQL> EXECUTE DBMS_LOGSTDBY.UNSKIP@standby('DML','test','test');
```

If an error message is returned, it indicates there is a problem with the configuration.

5

Log Transport Services

This chapter explains how to set up and use log transport services to control automated archiving of redo logs from the primary database to one or more standby sites. It includes the following topics:

5.1 Introduction to Log Transport Services

The **log transport services** component of the Data Guard environment is responsible for automatic archiving of primary database online redo logs. Once archived, these logs are known as **archived redo logs.** Log transport services provide for the management of archived redo log permissions, destinations, transmission, reception, and transmission failure resolution. In a Data Guard environment, the log transport services component coordinates its activities with

log apply services and role management services for switchover and failover operations.

> **See Also:** Chapter 6, "Log Apply Services" and Chapter 7, "Role Management Services"

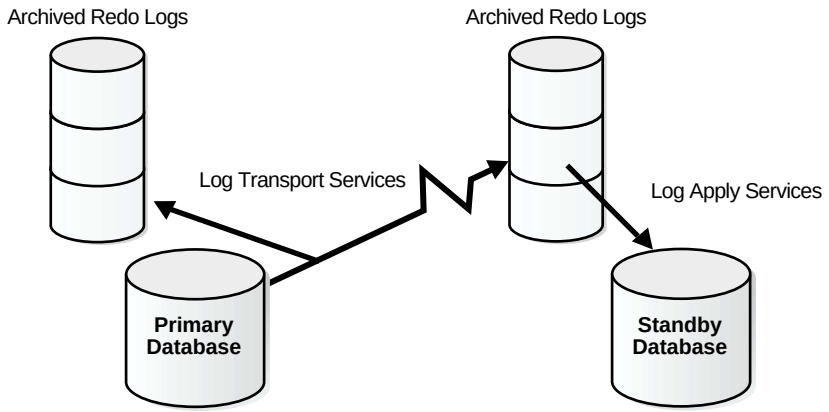
5.1.1 Background

Log transport services and log apply services automate the process of copying online redo logs to a remote node and applying them so that data from a primary database can be made available on a standby database. Log transport services provide the capability to archive the primary databases's redo log files to a maximum of 10 archiving locations called **destinations.**

Log transport services provide flexibility in defining destinations and archiving completion requirements. These services also provide I/O failure handling and transmission restart capability.

Figure 5–1 shows a simple Data Guard configuration with redo logs being archived from a primary database to a local destination and a remote standby database destination using log transport services.

Figure 5–1 Archiving Redo Logs



5.1.2 Functional Overview

The **archiver process (ARC*n*)** or **log writer process (LGWR)** can be configured to archive online redo logs to a maximum of 10 destinations. These destinations are

specified using the LOG_ARCHIVE_DEST_*n* (where *n* is a number from 1 to 10) database initialization parameter. One destination must be a local directory, but others can be at remote destinations. A remote destination is identified by an Oracle Net network service name, which is defined in the appropriate network services configuration files.

Archiving redo logs to a remote destination requires uninterrupted connectivity through Oracle Net. If the destination is a remote physical standby database, it must be mounted or open in read-only mode to receive the archived redo logs.

> **Note:** Log transport services to the physical standby database must be configured to use a dedicated server process in managed recovery mode. Although the read-only mode allows a shared server process, you must have a dedicated server once you open the database again in managed recovery mode.

Logical standby databases must be open in read / write mode. If a network connection to a standby database is broken, it must be reestablished to continue log archiving.

5.1.3 Log Transport Services Process Architecture

Log transport services use the following processes on the primary site:

- Log writer (LGWR)

  The **log writer process (LGWR)** collects transaction redo and updates the **online redo logs**. The log writer process can also create local archived redo logs and transmit online redo to standby databases.

- Archiver (ARC*n*)

  The **archiver process (ARC*n*)**, or a SQL session performing an archiving operation, creates a copy of the online redo logs, either locally or remotely, for standby databases.

- Fetch archive log (FAL) client

  The **fetch archive log (FAL) client** is the Oracle Net name of the standby site that is requesting a gap sequence. The initialization parameter for the FAL client is set on the standby site. The FAL client pulls archived redo log files from the primary site and initiates and requests the transfer of archived redo log files automatically when it detects an archive gap on the standby database.

- Fetch archive log (FAL) process (physical standby databases only)

  The **fetch archive log (FAL)** process provides a client/server mechanism for resolving gaps detected in the range of archived redo logs generated at the primary database and received at the standby database. The **FAL client** requests the transfer of archived redo log files automatically when it detects a **gap** in the redo logs received by the standby database. The **FAL server** typically runs on the primary database and services the FAL requests coming from the FAL client. The FAL client and server are configured using the FAL_CLIENT and FAL_SERVER initialization parameters that are set on the standby location. (The parameters can also be set on the primary database in preparation for a future switchover operation.)

5.2 Log Transport Services Capabilities

Log transport services capabilities allow you to determine how archiving online redo logs will be handled in a Data Guard configuration. These capabilities can be classified into the following categories:

- Permission
- Destination
- Transmission
- Reception
- Failure Resolution

5.2.1 Permission

You control whether the primary database can archive redo logs to a standby site, and whether the standby database can receive them from a primary site. Permission is set using the REMOTE_ARCHIVE_ENABLE database initialization parameter.

See Also: Section 5.5.2.1

5.2.2 Destination

Using log transport services, redo logs can be archived to any of the following destinations:

- Local

  A local file system location.

- Remote

  A remote location accessed using an Oracle Net service name. There are several types of remote destinations, as shown in the following table:

Remote Destination Types	Description
Physical standby database	An identical copy of a primary database used for disaster protection.
Logical standby database	Takes standard Oracle archived redo logs, transforms them into SQL transactions, and then applies them to an open standby database.
Archive log repository	An off-site storage location for archived redo logs. There is no database associated with this type of destination, just a control file.
Cross-instance archival database	Used within a Real Application Clusters configuration, another instance of the primary database. This is an archiving destination for all redo logs from the cluster and is on the node where recovery is performed.

See Also:

- Section 5.5.2 for more information about specifying destinations and destination options
- Section 8.4.1 for information about cross-instance archiving

5.2.3 Transmission

You can choose the method and form (online redo logs or archived redo logs) by which redo logs are transmitted from the primary database to each archive destination by specifying:

- The process that will perform the log transmission operation

  You can specify either the log writer process or the archiver process to transmit the redo logs.

- The method of transmitting redo logs over the network

  When using the log writer process, you can specify synchronous or asynchronous network transmission of redo logs to remote destinations. The log writer process sends redo logs in the form of **online redo logs.**

When you are using the archiver process, only synchronous transmission of redo logs to remote destinations is available. The archiver process sends redo logs in the form of **archived redo logs.**

Using transmission options, you can configure log transport services protection modes that give you the flexibility to balance data protection levels and application performance.

- The method of sending redo logs to a standby

  You can specify whether log archiving network I/O operations are to be performed synchronously or asynchronously. This applies to both the log writer and archiver processes.

  See Also: Section 5.7 for more information on log transport services data protection modes

5.2.4 Reception

Reception gives you control over determining where redo logs will be placed at remote destinations. The redo data can be stored on the standby site using either standby redo logs or archived redo logs.

Oracle9*i* introduces the concept of **standby redo logs.** Standby redo logs form a separate pool of log file groups. Although standby redo log file groups are only used when a database is operating in a physical standby role, you can specify them on the primary database in anticipation of a switchover operation.

See Also: Section 5.8.4 for more information about reception options

5.2.5 Failure Resolution

Failure resolution gives you control over what actions will occur on a primary database if log archiving from the primary database to the standby database fails. These actions include:

- Retrying the archiving operation to a failed destination after a specified period of time and specifying a limit to the number of times a destination will be retried
- Specifying an alternate or substitute destination
- Specifying that if connectivity to the standby database is lost, then the primary database will automatically shut down, ensuring that data in the primary

database is always protected by the presence of a standby database and ensuring that the data does not diverge between the primary database and at least one standby database

See Also: Section 5.5.2.5 and Section 5.8 for more information about failure resolution options

5.3 Log Transport Services Interfaces

Log transport services options are entered and changed using the following interfaces:

- Database Initialization Parameters
- SQL Interface

5.3.1 Database Initialization Parameters

You configure the primary database to perform remote archiving by setting destinations and associated states. You do this using LOG_ARCHIVE_DEST_*n* (where *n* is an integer from 1 to 10) initialization parameter and corresponding LOG_ARCHIVE_DEST_STATE_*n* (where *n* is an integer from 1 to 10) initialization parameter. You can also use these initialization parameters to set up cascading standby databases, as described in Appendix E.

Note: If you do not use the Data Guard broker, you must define the LOG_ARCHIVE_DEST_*n* and LOG_ARCHIVE_DEST_STATE_*n* parameters on all standby sites so that when a switchover or failover operation occurs, all of the standby sites continue to receive logs from the *new* primary database. Configurations that you set up with the Data Guard command-line interface or Data Guard Manager handle the LOG_ARCHIVE_DEST_*n* and LOG_ARCHIVE_DEST_STATE_*n* definitions automatically, including defining the LOG_ARCHIVE_DEST_*n* parameters to point back to the primary database and all the other standby databases.

5.3.1.1 LOG_ARCHIVE_DEST_*n* Initialization Parameter

LOG_ARCHIVE_DEST_*n* (where *n* is an integer from 1 to 10) initialization parameter allows you to specify up to 10 archive destinations, including 1 required local destination and up to 9 additional local or remote destinations. This parameter also

allows you to set a number of archiving options for each destination. These options are set using the attributes described in Table 5–1.

Table 5–1 LOG_ARCHIVE_DEST_n Initialization Parameter Attributes

Attribute	Description	Capability	More Information
[NO]AFFIRM	Ensures that archived redo log contents are written to disk successfully and are available immediately for database recovery. The AFFIRM attribute performs all I/O operations to disk synchronously, while NOAFFIRM performs I/O operations asynchronously.	Transmission	See Section 5.8.3 and see AFFIRM and NOAFFIRM in Chapter 12
[NO]ALTERNATE=*destination*	Specifies an alternate location that can be used as a destination for archival operations if archiving to the original destination fails	Failure resolution	See Section 5.5.2.5 and see ALTERNATE and NOALTERNATE in Chapter 12
ARCH	The archiver process (ARC*n*) will archive online redo logs to local and remote destinations	Transmission	See Section 5.8.1 and see ARCH and LGWR in Chapter 12
ASYNC[=*blocks*]	Network I/O operations for log transport services are to be done asynchronously. Specifies the size of the SGA network buffer to be used.	Transmission	See Section 5.8.2 and see SYNC and ASYNC in Chapter 12
[NO]DELAY[=*minutes*]	Specifies a time interval between archiving a redo log at a remote site and applying that archived redo log to the standby database	Reception	See Section 5.5.2.8 and see DELAY and NODELAY in Chapter 12
[NO]DEPENDENCY=*destination*	Archival operations to a destination are dependent upon the success or failure of archival operations to another local destination	Transmission	See Section 5.5.2.6 and see DEPENDENCY and NODEPENDENCY in Chapter 12
LGWR	The log writer process will archive current online redo logs to local and remote destinations	Transmission	See Section 5.8.1 and see ARCH and LGWR in Chapter 12

Table 5–1 (Cont.) LOG_ARCHIVE_DEST_n Initialization Parameter Attributes

Attribute	Description	Capability	More Information
LOCATION=*local_disk_directory*	Identifies a local disk directory location where archived redo logs will be stored	Destination	See Section 5.3.1.3 and see LOCATION and SERVICE in Chapter 12
MANDATORY	Archiving to this location must succeed before the online redo log can be overwritten	Failure resolution	See Section 5.5.2.3 and see MANDATORY and OPTIONAL in Chapter 12
[NO]MAX_FAILURE=*count*	Sets a limit on the number of consecutive times log transport services will retry archival operations to any location after a communication failure	Failure resolution	See Section 5.5.2.5 and see MAX_FAILURE and NOMAX_FAILURE in Chapter 12
[NO]NET_TIMEOUT=*seconds*	Specifies the number of seconds the log writer process will wait for status from the network server before assuming there is a network timeout error	Failure resolution	See NET_TIMEOUT and NONET_TIMEOUT in Chapter 12
OPTIONAL	Successful archiving to this location is not necessary	Failure resolution	See Section 5.5.2.3 and see MANDATORY and OPTIONAL in Chapter 12
[NO]QUOTA_SIZE=*blocks*	The maximum number of 512-byte blocks that can be consumed on the specified destination	Transmission	See Section 5.5.2.9 and see QUOTA_SIZE and NOQUOTA_SIZE in Chapter 12
[NO]QUOTA_USED=*blocks*	Identifies the size of all of the archived redo logs currently residing on the specified destination	Transmission	See QUOTA_USED and NOQUOTA_USED in Chapter 12
[NO]REGISTER	Specifies whether or not the archiving location is to be recorded in the standby database control file	Reception	See REGISTER and NOREGISTER in Chapter 12

Table 5–1 (Cont.) LOG_ARCHIVE_DEST_n Initialization Parameter Attributes

Attribute	Description	Capability	More Information
REGISTER=*location_format*	Specifies a fully qualified filename format template for archived redo logs that is different from the default filename format template defined in the primary and standby database initialization parameter files. The default filename format template is a combination of the database initialization parameters STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT.	Reception	See Section 5.5.2.7 and see REGISTER=location_format in Chapter 12
[NO]REOPEN[=*seconds*]	Specifies the minimum number of seconds before the archiver process (ARC*n*, foreground, or log writer process) should try again to access a previously failed destination	Failure resolution	See Section 5.5.2.5 and see REOPEN and NOREOPEN in Chapter 12
SERVICE=*net_service_name*	Specifies the net service name of a standby database where redo log files are to be archived	Destination	See Section 5.5.2.2 and see LOCATION and SERVICE in Chapter 12
SYNC[=PARALLEL|NOPARALLEL]	Indicates that network I/O operations for log transport services are to be done synchronously and in parallel or not for all LGWR destinations.	Transmission	See Section 5.8.2 and see SYNC and ASYNC in Chapter 12
[NO]TEMPLATE=*filename_template*	Defines a directory specification and format template for archived redo logs at the standby destination. You can specify this attribute in either the primary or standby initialization parameter file, but the attribute applies only to the database role that is archiving.	Reception	See TEMPLATE and NOTEMPLATE in Chapter 12

Note: The use of the LOG_ARCHIVE_DEST_*n* initialization parameter on the primary database replaces the use of LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST parameters on the primary database in previous versions of the Oracle database server.

5.3.1.2 LOG_ARCHIVE_DEST_STATE_*n* Initialization Parameter

The LOG_ARCHIVE_DEST_STATE_*n* (where *n* is an integer from 1 to 10) initialization parameter specifies the *state* of the corresponding destination indicated by the LOG_ARCHIVE_DEST_*n* initialization parameter (where *n* is the same integer). For example, the LOG_ARCHIVE_DEST_STATE_3 parameter specifies the state of the LOG_ARCHIVE_DEST_3 destination.

LOG_ARCHIVE_DEST_STATE_*n* parameter attributes are described in Table 5–2.

Table 5–2 LOG_ARCHIVE_DEST_STATE_n Initialization Parameter Attributes

Attribute	Description
ENABLE	Log transport services can archive redo logs at this destination.
DEFER	Log transport services will not archive redo logs to this destination. This is an unused destination.
ALTERNATE	This destination is not enabled but will become enabled if communication to another destination fails.
RESET	Functions the same as DEFER, but clears any error messages for the destination if it had previously failed.

5.3.1.3 Setting Destination Parameters in the Initialization Parameter File

Setting up log transport services requires modification of the database initialization file. When you set up log transport services parameters, you can specify attributes as:

- A single attribute on one line
- Multiple attributes on one line
- Single attributes on multiple sequential lines
- Attributes for multiple destinations on multiple lines

Example 5–1 shows how to specify a single attribute on one line.

Example 5–1 Specifying a Single Attribute on One Line

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/d1arch'
```

Example 5–2 shows how to set multiple attributes on a single line.

Example 5–2 Specifying Multiple Attributes on One Line

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/d1arch OPTIONAL'
```

Example 5–3 shows how to set multiple attributes incrementally on separate lines. SERVICE or LOCATION attributes must be specified on the first line. You can specify attributes incrementally even when the initialization parameter file is used to create a server parameter file (SPFILE).

Example 5–3 Specifying Multiple Attributes Incrementally

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/d1arch'
LOG_ARCHIVE_DEST_1='OPTIONAL'
LOG_ARCHIVE_DEST_1='REOPEN=5'
```

Example 5–4 shows how to specify attributes for multiple destinations. Incremental parameters such as the LOG_ARCHIVE_DEST_*n* initialization parameter must immediately follow each other in the initialization parameter file.

Example 5–4 Specifying Multiple Attributes for Multiple Destinations

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/d1arch OPTIONAL'
LOG_ARCHIVE_DEST_2='SERVICE=stby REOPEN=60'
```

Attributes specified on multiple lines for a single destination must be entered sequentially. Example 5–5 shows an entry for the LOG_ARCHIVE_DEST_1 that is not allowed.

Example 5–5 Incorrect Destination Specification

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/d1arch'
LOG_ARCHIVE_DEST_2='SERVICE=stby REOPEN=60'
LOG_ARCHIVE_DEST_1='OPTIONAL'
```

A destination specification can be redefined after another destination has been specified if the new specification includes LOCATION or SERVICE attributes. Example 5–6 shows how to replace the initial specification of LOG_ARCHIVE_DEST_1.

Example 5–6 Replacing a Destination Specification

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/d1arch'
LOG_ARCHIVE_DEST_2='SERVICE=stby REOPEN=60'
LOG_ARCHIVE_DEST_1='LOCATION=/disk3/d3arch MANDATORY'
```

A string containing a null value for parameter attributes will clear a previously entered destination specification. Example 5–7 shows how to clear the definition of LOG_ARCHIVE_DEST_1.

Example 5–7 Clearing a Destination Specification

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/d1arch'
LOG_ARCHIVE_DEST_2='SERVICE=stby REOPEN=60'
LOG_ARCHIVE_DEST_1=''
```

To specify service names that use embedded characters, such as equal signs (=) and spaces, use double quotation marks (") around the service name. Example 5–8 shows the specification of a service name that includes a space.

Example 5–8 Specifying a Service Name That Includes a Space

```
LOG_ARCHIVE_DEST_6='SERVICE="stdby arch" MANDATORY'
```

Caution: Oracle Corporation does not recommend creating service names that contain spaces.

5.3.2 SQL Interface

SQL statements can be used to set up most initialization parameters for log transport services and to monitor the environment. You can use SQL statements for the following tasks:

- Viewing Current Settings of Initialization Parameters
- Changing Initialization Parameter Settings
- Setting Log Transport Services Options Using SQL

5.3.2.1 Viewing Current Settings of Initialization Parameters

You can use SQL to query fixed views such as V$ARCHIVE_DEST to see current LOG_ARCHIVE_DEST_*n* initialization parameter settings. For example, to view current destination settings on the primary database, enter the following statement:

```
SQL> SELECT DESTINATION FROM V$ARCHIVE_DEST;
```

5.3.2.2 Changing Initialization Parameter Settings

At runtime, the LOG_ARCHIVE_DEST_*n* initialization parameter can be changed using ALTER SYSTEM and ALTER SESSION statements. You can specify the attributes in one or more strings in one statement or incrementally in separate statements.

Note: When incrementally changing the LOG_ARCHIVE_DEST_*n* parameters and you have an SPFILE system initialization parameter file, you must specify SCOPE IS MEMORY. Otherwise, you will not be able to restart the database.

Example 5–9 shows how to modify archive destination parameters at the system and session level.

Example 5–9 Modifying Destination Parameters at the System and Session Level

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_6='SERVICE=stby REOPEN=8';
SQL> ALTER SESSION SET LOG_ARCHIVE_DEST_6='SERVICE=stby2 REOPEN=10';
```

Unlike setting parameters in the database initialization file at startup, at runtime you can incrementally change the characteristics of a destination after modifying the settings of another destination. Example 5–10 shows how to make incremental changes to LOG_ARCHIVE_DEST_6 after LOG_ARCHIVE_DEST_7 has been defined.

Example 5–10 Adding Destination Attributes Incrementally

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_6='SERVICE=stby1 REOPEN=8';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_7='SERVICE=stby2 NOREOPEN';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_6='MANDATORY LGWR DELAY';
```

You can enter a null value for a destination to clear a previous definition. Example 5–11 shows how to clear the definition of LOG_ARCHIVE_DEST_6.

Example 5–11 Clearing a Destination Specification

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_6='SERVICE=stby1 REOPEN=8';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_7='SERVICE=stby2 NOREOPEN';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_6='';
```

To specify service names that use embedded equal signs (=) and spaces, use double quotation marks (") around the service name. Example 5–12 shows the specification of a service name that includes a space.

Example 5–12 Specifying a Service Name That Includes a Space

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_6='SERVICE="stdby arch" MANDATORY';
```

> **Caution:** Oracle Corporation does not recommend creating service names that contain spaces.

Table 5–3 lists the attributes that can be set for the LOG_ARCHIVE_DEST_*n* initialization parameter and indicates whether the attribute can be changed using an ALTER SYSTEM or ALTER SESSION statement.

Table 5–3 Changing Destination Attributes Using SQL

Attribute	ALTER SYSTEM	ALTER SESSION
[NO]AFFIRM	Yes	Yes
[NO]ALTERNATE=*destination*	Yes	Yes
ARCH	Yes	Yes
ASYNC[=*blocks*]	Yes	No
[NO]DELAY	Yes	Yes
[NO]DEPENDENCY=*destination*	Yes	No
LGWR	Yes	No
LOCATION=*local_disk_directory*	Yes	Yes
MANDATORY	Yes	Yes
[NO]MAX_FAILURE=*count*	Yes	No
OPTIONAL	Yes	Yes
[NO]NET_TIMEOUT[=*seconds*]	Yes	No
[NO]QUOTA_SIZE=*blocks*	Yes	No
[NO]QUOTA_USED=*blocks*	Yes	No
[NO]REGISTER	Yes	Yes

Table 5–3 (Cont.) Changing Destination Attributes Using SQL

Attribute	ALTER SYSTEM	ALTER SESSION
REGISTER=location_format	Yes	Yes
[NO]REOPEN[=*seconds*]	Yes	Yes
SERVICE=*net_service_name*	Yes	Yes
SYNC[=PARALLEL|NOPARALLEL]	Yes	Yes
[NO]TEMPLATE=*filename_template*	Yes	Yes

5.3.2.3 Setting Log Transport Services Options Using SQL

You can set specific options for log transport services using SQL statements. The following example sets the data protection mode to maximize availability.

```
SQL> ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE AVAILABILITY;
```

See Also: *Oracle9i SQL Reference* for more information about SQL statements

5.4 Configuring Log Transport Services: Basic Tasks

Log transport services manage the automatic archiving of primary database online redo logs to the standby site. Table 5–4 summarizes the basic tasks for configuring log transport services on the primary database and the standby database.

Table 5–4 Task List: Configuring Log Transport Services

Step	Task	See ...
1	Set the primary database to run in ARCHIVELOG mode.	Section 5.5.1.1 and the *Oracle9i Database Administrator's Guide*
2	Determine the appropriate number and size of primary database online redo logs.	Section 5.5.1.2
3	Set the CONTROL_FILE_RECORD_KEEP_TIME initialization parameter to prevent overwriting information in the archived redo logs.	Section 5.5.1.3 and the *Oracle9i Database Reference*
4	Monitor checkpointing and system performance by examining the V$SYSTAT view and adjust appropriately.	Section 5.5.1.4 and the *Oracle9i Database Administrator's Guide*

Table 5–4 (Cont.) Task List: Configuring Log Transport Services

Step	Task	See ...
5	Set permission for the archiving of online redo logs to remote destinations using the REMOTE_ARCHIVE_ENABLE initialization parameter.	Section 5.5.2.1
6	Set the initialization parameters for the primary database.	Section 5.5.2.2 through Section 5.5.2.9, Section 5.6.3.1, and Chapter 12 for details about these attributes
7	Duplicate and transfer the primary database initialization parameter file to the standby site. Depending on your configuration, you may need to set filename conversion parameters.	Section 5.6.2, Section 5.6.3.2, and Section 6.3.4
8	Choose a data protection mode.	Section 5.7
9	Configure the data protection mode for log transport services.	Section 5.8
9a	Specify a redo log writing process.	Section 5.8.1
9b	Specify a network transmission mode.	Section 5.8.2
9c	Specify a method of writing archived logs to disk.	Section 5.8.3
9d	Create standby redo logs, if necessary.	Section 5.8.4
9e	Set a failure resolution policy.	Section 5.8.5

5.5 Configuring Log Transport Services on the Primary Database

This section discusses what settings to make on the primary database to use the various options of log transport services in a Data Guard environment. The following topics are included in this section:

- Configuring Log Transport Services
- Setting Up the Log Transport Services Environment

5.5.1 Configuring Log Transport Services

Considerations when configuring log transport services on the primary database include:

5.5.1.1 Setting the Database to Run in ARCHIVELOG Mode

To use log transport services, you must set the primary database to run in ARCHIVELOG mode. When you run a database in **ARCHIVELOG mode**, you enable the **archiving** of the online redo logs. The database control file indicates that a group of filled online redo logs cannot be reused by the log writer process until the group is archived. A filled group is immediately available for archiving after a redo **log switch** occurs.

> **See Also:** *Oracle9i Database Administrator's Guide* for details about setting ARCHIVELOG mode

You can set the ARCHIVELOG mode at database creation or by using the SQL `ALTER DATABASE` statement.

> **Note:** Oracle Corporation does *not* recommend running in NOARCHIVELOG mode, because it severely limits the possibilities for recovery of lost data.

5.5.1.2 Configuring Online Redo Logs in the Primary Database

Both the size of the online redo logs and the frequency with which they switch affect the generation of archived redo logs at the primary site. In general, the most important factor when considering the size of an online redo log should be the amount of application data that needs to be applied to a standby database during a database failover operation. The larger the online redo log, the more data that needs to be applied to a standby database to make it consistent with the primary database.

Another important consideration should be the size of the archival media. Online redo logs should be sized so that a filled online redo log group can be archived to a single unit of offline storage media (such as a tape or disk), with the least amount of space on the medium left unused. For example, suppose only one filled online redo log group can fit on a tape, and 49% of the tape's storage capacity remains unused. In this case, it is better to decrease the size of the online redo log files slightly, so that two log groups can be archived per tape.

The best way to determine the appropriate number of online redo log files for a database instance is to test different configurations. The goal is to create the fewest groups possible without hampering the log writer process's ability to write redo log information.

In some cases, a database instance may require only two groups. In other situations, a database instance may require additional groups to guarantee that a recycled group is always available to the log writer process. During testing, the easiest way to determine if the current configuration is satisfactory is to examine the contents of the log writer process's trace file and the database's alert log. If messages indicate that the log writer process frequently must wait for a group because a checkpoint has not completed or a group has not been archived, add more groups.

Table 5–5 provides some guidelines to help in determining the number and sizes of primary database online redo logs.

Table 5–5 Guidelines for Online Redo Log Configuration

Online Redo Log Size	Advantages	Disadvantages
Small (50 megabytes)	■ Standby database lag time is minimized ■ Ideal when using the archiver process (ARC*n*) to exclusively archive online redo logs	■ Log switches occur more frequently ■ A larger number of online redo log groups may be required to allow log switches to occur smoothly ■ The large number of archived redo logs may be difficult to manage ■ Additional archiver processes may be required ■ More frequent database checkpoints may occur
Medium (200 megabytes)	■ Ideal when using a combination of archiver and log writer processes to archive online redo logs	■ The large number of archived redo logs may be difficult to manage

Table 5–5 (Cont.) Guidelines for Online Redo Log Configuration

Online Redo Log Size	Advantages	Disadvantages
Large (1000 megabytes)	■ Fewer online redo log groups are required ■ Archiving occurs less frequently and is more efficient ■ Archived redo logs are easier to manage ■ Recovery is simplified because of fewer archived redo logs to apply ■ Ideal when using the log writer process (LGWR) exclusively to archive online redo logs ■ Fewer database checkpoints	■ Standby database lag time can be high ■ Increased potential for data loss when using the archiver process ■ Archived redo logs may not fit efficiently on backup media ■ Instance recovery may take longer

5.5.1.3 Controlling the Reuse of Archived Redo Logs

The CONTROL_FILE_RECORD_KEEP_TIME initialization parameter specifies the minimum number of days that must pass before a reusable record in the control file can be reused. Setting this parameter prevents the ARCHIVELOG mechanism from overwriting a reusable record in the control file. (It applies only to records in the control file that are serially reusable.) Using this parameter helps to ensure that data is made available on the standby database. The range of values for this parameter is 0 to 365 days. The default value is 7 days.

> **See Also:** *Oracle9i Database Reference* for more details about the CONTROL_FILE_RECORD_KEEP_TIME initialization parameter

5.5.1.4 General Considerations for Configuring Log Transport Services

The Oracle database server will attempt a checkpoint at each log switch. Therefore, if the online redo log size is too small, frequent log switches will lead to frequent checkpointing and negatively affect system performance. Oracle Corporation recommends a checkpoint and log switch interval between 5 and 15 minutes during normal operations and at 5-minute intervals during peak load periods, such as during batch processing. To get started, examine the V$SYSSTAT view during a peak load period and then calculate 5 to 10 minutes of redo.

If you have remote hardware mirroring, you may want a minimum number of log groups, to ensure that no-data-loss failover can occur in case some logs are not archived. Furthermore, if some redo logs are not archived, they are not available to the standby database. This is another reason to avoid overly large online redo logs.

The sizing of the log files, along with the setting of the data protection modes, should factor in how much data loss your site can tolerate.

You should avoid situations where you cannot overwrite a redo log because you must wait for a checkpoint or archiving of that log. In these cases, there is usually a combination of problems with redo log configuration and database tuning. Oracle Corporation recommends that you solve the tuning problem first but, if that fails to correct the problem, add more log groups or increase log sizes.

Oracle Corporation generally recommends:

- Multiplexing of online redo logs and a minimum of four log groups
- Disk striping across as many spindles as possible with approximately a 16-kilobyte stripe size or smaller of the online redo logs, especially for OLTP and batch applications

> **See Also:** *Oracle9i Database Administrator's Guide* for more details about configuring online redo logs and online redo log groups

5.5.2 Setting Up the Log Transport Services Environment

The log transport services component of the Data Guard environment is configured primarily through the setting of database initialization parameters and options. You set parameters by editing the database initialization parameter file or using SQL statements.

> **See Also:** *Oracle9i Database Reference* for details about creating and editing database initialization parameter files

5.5.2.1 Setting Permission to Archive Redo Logs

Permission for the archiving of online redo logs to remote destinations is specified using the `REMOTE_ARCHIVE_ENABLE` initialization parameter. This parameter provides `true`, `false`, `send`, and `receive` options. In most cases, you should set this parameter to `TRUE` on both the primary and standby databases in a Data Guard environment. To independently enable and disable the sending and receiving of remote archived redo logs, use the `send` and `receive` values.

The following list describes how you might use each option to control archiving of online redo logs:

- Specify REMOTE_ARCHIVE_ENABLE=true on the primary database and standby databases to allow the primary database to send redo logs to the standby database and to allow the standby database to receive redo logs for archiving from the primary database.
- Specify REMOTE_ARCHIVE_ENABLE=false on the primary database and standby databases to disable both the sending and receiving of redo logs.
- Specify REMOTE_ARCHIVE_ENABLE=send on the primary database to enable the primary database to send redo logs to the standby database.
- Specify REMOTE_ARCHIVE_ENABLE=receive on the standby database to enable the standby database to receive redo logs from the primary database.

Specifying the send and receive values together is the same as specifying true. Every instance of the database must contain the same REMOTE_ARCHIVE_ENABLE value.

5.5.2.2 Specifying Archive Destinations for Redo Logs

In addition to setting up the primary database to run in ARCHIVELOG mode, you must configure the primary database to archive redo logs by setting destinations and associated states. You do this using the LOG_ARCHIVE_DEST_*n* initialization parameter and corresponding LOG_ARCHIVE_DEST_STATE_*n* parameter as discussed in Section 5.3.1.

This section describes the creation of a standby database named standby1 on a remote node named stbyhost based on the following assumptions:

- The prmyinit.ora file is the initialization parameter file for the primary database.
- You have backed up the primary database files.
- You have created the standby database control file.
- You have transferred the datafiles and control file to the standby site (stbyhost).
- You have copied the primary database initialization parameter file to stbyhost.
- You have used Oracle Net Manager to configure the **listener** on stbyhost to use the TCP/IP protocol, and to statically register the standby1 database

service using its SID, so that the standby1 standby database can be managed by the Data Guard broker.

- You have used the Oracle Net Manager to create the net service name standby1 that can be resolved to a connect descriptor that contains the same protocol, host address, port, and SID that were used to statically register the standby1 standby database with the listener on stbyhost.
- You have used the Listener Control utility to start the listener on stbyhost.

> **See Also:** *Oracle9i Net Services Administrator's Guide* for details about using Oracle Net Manager networking components

To set up log transport services to archive redo logs to the standby database, make the following modifications to the primary database initialization parameter file. These modifications will take effect after the instance is restarted:

1. Specify the archive destination by adding the following entry to the prmyinit.ora file:

```
LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1 MANDATORY REOPEN=60'
```

> **See Also:** LOCATION and SERVICE in Chapter 12

2. Enable the archive destination state by adding the following entry to the prmyinit.ora file:

```
LOG_ARCHIVE_DEST_STATE_2 = ENABLE
```

Parameter/Option	Meaning
LOG_ARCHIVE_DEST_2	Specifies a remote redo log archiving destination
SERVICE	Specifies the Oracle Net Manager service name of the remote destination
MANDATORY	Indicates that archiving to this location must succeed before the redo log can be reused
REOPEN	Indicates how many seconds the archiving process waits before reattempting to archive after a failed attempt
LOG_ARCHIVE_DEST_STATE_2	Indicates the state of the LOG_ARCHIVE_DEST_2 remote destination

Parameter/Option	Meaning
ENABLE	Indicates that the primary database can archive redo logs at this destination

To avoid having to restart the instance, you can issue the following SQL statements to ensure that the initialization parameters you have set in this step take effect immediately:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=standby1 MANDATORY REOPEN=60';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

5.5.2.3 Specifying a Reuse Policy for Online Redo Logs

You can specify a policy for the reuse of online redo logs using the OPTIONAL or MANDATORY attributes of the LOG_ARCHIVE_DEST_*n* initialization parameter. The archiving of log files to an OPTIONAL destination can fail, and the online redo log will be overwritten. If the archiving of redo log files to a MANDATORY destination fails, online redo log files will not be overwritten.

By default, one destination is mandatory even if all destinations are designated to be optional.

Example 5–13 shows how to set a mandatory local archiving destination and enable that destination.

Example 5–13 Setting a Mandatory Archiving Destination

```
LOG_ARCHIVE_DEST_3 = 'LOCATION=/arc_dest MANDATORY'
LOG_ARCHIVE_DEST_STATE_3 = ENABLE
```

See Also: MANDATORY and OPTIONAL in Chapter 12

5.5.2.4 Setting the Number of Destinations That Must Receive Logs

The LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* parameter (where n is an integer from 1 to 10) specifies the number of destinations that must receive redo logs successfully before the log writer process can reuse the online redo logs. All MANDATORY destinations and non-standby OPTIONAL destinations contribute to satisfying the LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* count. Example 5–14 shows how to specify that redo logs must be successfully archived to two destinations before the online redo logs can be reused.

Example 5–14 Specifying a Minimum Number of Successful Destinations

```
LOG_ARCHIVE_MIN_SUCCEED_DEST=2
```

See Also: *Oracle9i Database Administrator's Guide*

5.5.2.5 Specifying Archive Failure Policies

You use attributes of the LOG_ARCHIVE_DEST_*n* initialization parameter to specify what actions are to be taken when archiving to a destination fails. You can use LOG_ARCHIVE_DEST_*n* attributes to:

- Configure log transport services to archive again to a failed destination
- Control the number of archive destination failures allowed
- Configure alternate destinations

Use the REOPEN attribute of the LOG_ARCHIVE_DEST_*n* parameter to determine whether and when the archiver process or the log writer process attempts to archive redo logs again to a failed destination following an error.

The REOPEN=seconds attribute specifies the minimum number of seconds that must elapse following an error before the archiving process will try again to access a failed destination. The default value is 300 seconds. The value set for the REOPEN attribute applies to all errors, not just connection failures. You can turn off the option by specifying NOREOPEN, which will prevent the destination from being retried after a failure occurs.

You can use the REOPEN attribute in conjunction with the MAX_FAILURE attribute to limit the number of consecutive attempts that will be made to reestablish communication with a failed destination. Once the specified number of consecutive attempts has been exceeded, the destination is treated as if the NOREOPEN attribute had been specified.

If you specify REOPEN for an OPTIONAL destination, it is still possible for the Oracle database server to overwrite online redo logs even if there is an error. If you specify REOPEN for a MANDATORY destination, log transport services stalls the primary database when it cannot successfully archive redo logs. When this situation occurs, consider the following options:

- Change the destination by deferring the destination, specifying the destination as optional, or changing the service.
- Specify an alternate destination.
- Disable the destination.

When you use the REOPEN attribute, note that:

- The archiver or log writer process reopens a destination only when starting an archive operation from the beginning of the log file and never during a current operation. Archiving always starts the log copy from the beginning.
- If a value was specified or the default value used for the REOPEN attribute, the archiving process checks whether the time of the recorded error plus the REOPEN interval is less than the current time. If it is, the archival operation to that destination is retried.

See Also: REOPEN and NOREOPEN in Chapter 12

You can control the number of times a destination will be retried after a log archiving failure by specifying a value for the MAX_FAILURE=*count* attribute of the LOG_ARCHIVE_DEST_*n* initialization parameter.

The MAX_FAILURE attribute specifies the maximum number of contiguous archiving failures that will be allowed for the particular destination. This attribute is useful for archive destinations that you want to retry after a failure, but do not want to retry indefinitely. The REOPEN attribute is required when you use the MAX_FAILURE attribute. Example 5–15 shows how to set a retry time of 5 seconds and limit retries to 3 times.

Example 5–15 Setting a Retry Time and Limit

```
LOG_ARCHIVE_DEST_1='LOCATION=/arc_dest REOPEN=5 MAX_FAILURE=3'
```

See Also: MAX_FAILURE and NOMAX_FAILURE in Chapter 12

Using the ALTERNATE attribute of the LOG_ARCHIVE_DEST_*n* parameter, you can specify alternate archive destinations. An alternate archive destination can be used when the archiving of an online redo log to a standby site fails. If archiving fails and the NOREOPEN attribute has been specified, or the MAX_FAILURE attribute threshold has been exceeded, log transport services will attempt to archive redo logs to the alternate destination on the next archiving operation.

Use the NOALTERNATE attribute to prevent the original archive destination from automatically changing to an alternate archive destination when the original archive destination fails.

Example 5–16 shows how to set the initialization parameter file so that a single, mandatory, local destination will automatically fail over to a different destination if any error occurs.

Example 5–16 Specifying an Alternate Destination

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1 MANDATORY ALTERNATE=LOG_ARCHIVE_DEST_2'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_2='LOCATION=/disk2 MANDATORY'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
LOG_ARCHIVE_DEST_STATE_2=ALTERNATE
```

If the LOG_ARCHIVE_DEST_1 destination fails, the archiving process will automatically switch to the LOG_ARCHIVE_DEST_2 destination at the next log switch on the primary database.

See Also: ALTERNATE and NOALTERNATE in Chapter 12

5.5.2.6 Specifying Dependent Destinations

Archiving redo logs to a remote database can be defined as being dependent upon the success or failure of an archival operation for another destination. The dependent destination is known as the **child destination.** The destination on which the child depends is known as the **parent destination.**

Use the DEPENDENCY attribute of the LOG_ARCHIVE_DEST_*n* initialization parameter to define a child destination. This attribute indicates that this destination depends upon the successful completion of archival operations for the parent destination.

Specifying a destination dependency can be useful in the following situations:

- The standby database and the primary database are on the same node. Therefore, the archived redo logs are implicitly accessible to the standby database.
- Clustered file systems are used to provide remote standby databases with access to the primary database archived redo logs.
- Operating system-specific network file systems are used, providing remote standby databases with access to the primary database archived redo logs.
- Mirrored disk technology is used to provide transparent networking support across geographically remote distances.

- There are multiple standby databases on the same remote node, sharing access to common archived redo logs for staggered managed recovery.

In these situations, although a physical archival operation is not required, the standby database needs to know the location of the archived redo logs on the primary site. This allows the standby database to access the archived redo logs on the primary site when they become available for managed recovery. You must specify an archiving destination as being dependent on the success or failure of another (parent) destination. This is known as a **destination dependency.**

See Also: DEPENDENCY and NODEPENDENCY in Chapter 12

5.5.2.7 Registering the Location of Archived Redo Logs at the Remote Destination

The REGISTER attribute indicates that the fully qualified filename of the archived redo log at the remote destination is to be recorded with the destination database at the remote destination. The fully qualified filename is derived from the values entered in the STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT initialization parameters. This is the implicit default setting.

For a physical standby destination, the archived redo log filename is recorded in the destination database control file, which is then used by the managed recovery operation. Also, for a physical standby destination database, this archived redo log registry serves as the manifest for the standby recovery operation.

For a logical standby database, the archived redo log filename is recorded in a tablespace maintained by the logical standby database control file, which is then used by log apply services.

You can also use the optional TEMPLATE=*filename_template* attribute to specify a filename format that is different from the format defined by the STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT initialization parameters at the standby destination. Thus, you can use the TEMPLATE attribute to override the values entered in the STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT initialization parameters. This is the filename that will be recorded at the remote destination.

See Also: REGISTER and NOREGISTER, REGISTER=location_format, and TEMPLATE and NOTEMPLATE in Chapter 12

5.5.2.8 Specifying a Time Lag for the Application of Redo Logs

By default, a physical standby database automatically applies archived redo logs when they arrive from the primary database. A logical standby database automatically applies SQL statements once they have been transformed from the archived redo logs. But in some cases, you may want to create a time lag between the archiving of a redo log at the primary site and the applying of the redo log at the standby site. A time lag can protect against the application of corrupted or erroneous data from the primary site to the standby site.

Use the DELAY=*minutes* attribute of the LOG_ARCHIVE_DEST_*n* initialization parameter to specify a time lag for applying redo logs at the standby site. The default setting for this attribute is NODELAY. If the DELAY attribute is set with no value specified, then the value for this attribute is 30 minutes. Example 5–17 shows how to set up a destination with a time delay of 4 hours.

The DELAY interval is relative to when the archived redo log file is complete at the destination; it does not delay the transport of the redo log to the standby database.

Example 5–17 Specifying a Time Delay for Archiving Redo Logs

```
LOG_ARCHIVE_DEST_3='SERVICE=stby1 DELAY=240 REOPEN=300'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

See Also:

- Chapter 12 about using the DELAY and the NODELAY attributes on the LOG_ARCHIVE_DEST_*n* parameter
- Section 10.1.9, "Scenario 9: Standby Database with a Time Lag"
- Section 6.3.3.3 for physical standby databases using the DELAY control option on the ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DELAY statement to supersede any apply delay interval specified on the primary database
- *Oracle9i Supplied PL/SQL Packages and Types Reference* for logical standby databases using the DBMS_LOGSTDBY.APPLY_SET procedure to supersede any apply delay interval specified on the primary database

5.5.2.9 Specifying Quotas for Archive Destinations

You can specify the amount of physical storage on a disk device to be available for an archiving destination using the QUOTA_SIZE attribute of the LOG_ARCHIVE_

DEST_*n* initialization parameter. The QUOTA_SIZE attribute indicates the maximum number of 512-byte blocks of physical storage on a disk device that may be consumed by a destination. The value is specified in 512-byte blocks even if the physical device uses a different block size. The optional suffix values K, M, and G represent thousand, million, and billion, respectively (the value "1K" means 1,000 512-byte blocks).

A destination can be designated as being able to occupy all or some portion of the physical disk represented by the destination. For example, in a Real Application Clusters environment, a physical archived redo log disk device may be shared by two or more separate nodes (through a clustered file system, such as is available with Sun Clusters). As there is no cross-instance initialization parameter knowledge, none of the Real Application Clusters nodes is aware that the archived redo log physical disk device is shared with other instances. This leads to substantial problems when the destination disk device becomes full; the error is not detected until every instance tries to archive to the already full device. This seriously affects database availability.

The QUOTA_SIZE value does not have to be the actual number of blocks on the disk device; the value represents the portion of the disk device that can be consumed by the archived redo log destination.

See Also: QUOTA_SIZE and NOQUOTA_SIZE in Chapter 12

5.6 Configuring Log Transport Services on the Standby Database

This section describes how to set up log transport services on the standby database in preparation for switchover. The following topics are presented:

- Configuring the Standby Initialization Parameter File
- Transferring the Initialization Parameter File to the Standby Database
- Setting Up the Initialization Parameter File

5.6.1 Configuring the Standby Initialization Parameter File

Most initialization parameters at the primary and standby databases should be identical, although some initialization parameters such as CONTROL_FILES and DB_FILE_NAME_CONVERT must differ. Differences in initialization parameters can cause performance degradation at a standby database and, in some cases, halt database operations completely. Change parameter values only when it is required for the functionality of the standby database or for filename conversions.

The initialization parameters in the following list play a key role in the configuration of the standby database.

- COMPATIBLE
- CONTROL_FILE_RECORD_KEEP_TIME
- CONTROL_FILES
- DB_FILE_NAME_CONVERT
- DB_FILES
- DB_NAME
- FAL_CLIENT (physical standby databases only)
- FAL_SERVER (physical standby databases only)
- LOCK_NAME_SPACE
- LOG_ARCHIVE_DEST_*n* (where *n* is a value from 1 to 10)
- LOG_ARCHIVE_DEST_STATE_*n* (where *n* is a value from 1 to 10)
- LOG_ARCHIVE_FORMAT
- LOG_ARCHIVE_MAX_PROCESSES
- LOG_ARCHIVE_MIN_SUCCEED_DEST (logical standby databases only)
- LOG_ARCHIVE_START
- LOG_ARCHIVE_TRACE
- LOG_FILE_NAME_CONVERT
- LOG_PARALLELISM (logical standby databases only)
- PARALLEL_MAX_SERVERS (logical standby databases only)
- REMOTE_ARCHIVE_ENABLE
- STANDBY_ARCHIVE_DEST
- STANDBY_FILE_MANAGEMENT

For more information about these and other database initialization parameters specific to the Data Guard environment, see Chapter 11.

5.6.2 Transferring the Initialization Parameter File to the Standby Database

Once you have configured the primary database initialization parameter file, you can duplicate the file for use by the standby database. The procedure for creating the standby initialization parameter file is as follows:

1. Copy the initialization parameter file for the primary database using an operating system utility.
2. Edit the initialization parameter file for use by the standby database.
3. Transfer the initialization parameter file to the standby site using an appropriate operating system utility.

5.6.3 Setting Up the Initialization Parameter File

Oracle Corporation suggests that you maintain two database initialization parameter files at both the primary and standby databases. This will allow you to easily change the databases from the primary role to the standby role or from the standby role to the primary role.

5.6.3.1 Primary Database Initialization Parameter Files

Example 5–18 and Example 5–19 show sections of initialization parameter files that you could maintain on the primary database. These examples show only parameters specific to log transport services. For complete examples of database initialization files, see Chapter 10.

Example 5–18 shows log transport services initialization parameters for a typical primary database. These log transport services parameter settings are used when the primary database is operating in the primary role.

Note: `LOG_ARCHIVE_DEST` and `LOG_ARCHIVE_DUPLEX_DEST` parameters have been set to null values because they are obsolete, and they are not used in the Data Guard environment.

Example 5–18 Primary Database: Primary Role Initialization Parameters

```
DB_NAME=primary1
CONTROL_FILES=primary.ctl
COMPATIBLE=9.0.1.0.0
LOG_ARCHIVE_START=TRUE
LOG_ARCHIVE_DEST=""
LOG_ARCHIVE_DUPLEX_DEST=""
```

```
LOG_ARCHIVE_DEST_1='LOCATION=/oracle/arc/ MANDATORY REOPEN=30'
LOG_ARCHIVE_DEST_2='SERVICE=standby MANDATORY REOPEN=15'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_STATE_2=ENABLE
LOG_ARCHIVE_FORMAT=log%t_%s.arc
REMOTE_ARCHIVE_ENABLE=TRUE
.
.
.
```

Example 5–19 shows log transport services parameters for an initialization parameter file to be maintained on the primary database and used if the primary database's role is changed to the standby role.

Example 5–19 Primary Database: Standby Role Initialization Parameters

```
DB_NAME=primary1
CONTROL_FILES=primary.ctl
COMPATIBLE=9.0.1.0.0
LOG_ARCHIVE_START=TRUE
# The following parameter is required only if the primary and standby databases
# are located on the same system.
LOCK_NAME_SPACE=primary1
FAL_SERVER=standby1
FAL_CLIENT=primary1
DB_FILE_NAME_CONVERT=('/standby','/primary')
LOG_FILE_NAME_CONVERT=('/standby','/primary')
STANDBY_ARCHIVE_DEST=/oracle/arc/
LOG_ARCHIVE_DEST_1='LOCATION=/oracle/arc/'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_STATE_2=DEFER
LOG_ARCHIVE_TRACE=127
LOG_ARCHIVE_FORMAT=log%t_%s.arc
STANDBY_FILE_MANAGEMENT=AUTO
REMOTE_ARCHIVE_ENABLE=TRUE
.
.
.
```

5.6.3.2 Standby Database Initialization Parameter Files

Example 5–20 and Example 5–21 show sections of initialization parameter files that you could maintain on the standby database. These examples show only parameters specific to log transport services. For complete examples of database initialization files, see Chapter 10.

Example 5–20 shows log transport services parameters for an initialization parameter file to be maintained on the standby database when it is operating in the standby role.

Example 5–20 Standby Database: Standby Role Initialization Parameters

```
DB_NAME=primary1
CONTROL_FILES=standby.ctl
COMPATIBLE=9.0.1.0.0
LOG_ARCHIVE_START=TRUE
# The following parameter is required only if the primary and standby databases
# are located on the same system.
LOCK_NAME_SPACE=standby1
FAL_SERVER=primary1
FAL_CLIENT=standby1
DB_FILE_NAME_CONVERT=("/primary","/standby")
LOG_FILE_NAME_CONVERT=("/primary","/standby")
STANDBY_ARCHIVE_DEST=/oracle/stby/arc
LOG_ARCHIVE_DEST_1='LOCATION=/oracle/stby/arc/'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_TRACE=127
LOG_ARCHIVE_FORMAT=log%t_%s.arc
STANDBY_FILE_MANAGEMENT=AUTO
REMOTE_ARCHIVE_ENABLE=TRUE
.
.
.
```

Example 5–21 shows transport services parameters for an initialization parameter file to be maintained on the standby database and used when it is to operate in the primary role.

Example 5–21 Standby Database: Primary Role Initialization Parameters

```
DB_NAME=primary1
CONTROL_FILES=standby.ctl
COMPATIBLE=9.0.1.0.0
LOG_ARCHIVE_START=true
LOG_ARCHIVE_DEST=""
LOG_ARCHIVE_DUPLEX_DEST=""
LOG_ARCHIVE_DEST_1='LOCATION=/oracle/stby/arc/ MANDATORY REOPEN=30'
LOG_ARCHIVE_DEST_2='SERVICE=primary1 MANDATORY REOPEN=15'
LOG_ARCHIVE_DEST_STATE_1=DEFER
LOG_ARCHIVE_DEST_STATE_2=DEFER
LOG_ARCHIVE_FORMAT=log%t_%s.arc
REMOTE_ARCHIVE_ENABLE=TRUE
```

.
.
.

5.7 Data Protection Modes

Data Guard provides three high-level modes of data protection that you can configure to balance cost, availability, performance, and transaction protection. Using the following SQL SET STANDBY DATABASE statement on the primary database, you can configure the Data Guard environment to maximize data protection, availability, or performance:

```
ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE {PROTECTION | AVAILABILITY | PERFORMANCE};
```

The default data protection mode is to maximize performance.

> **See Also:** Chapter 13 for additional information about SQL statements that are pertinent to Data Guard environments

To determine the appropriate data protection mode to use, you need to weigh your business requirements for data availability against user demands for response time and performance. Table 5–6 summarizes the data protection modes, the LOG_ARCHIVE_DEST_*n* settings for log transport services, and the implications of each one for primary database data divergence and standby database switchover and failover.

Table 5–6 Summary of Data Protection Modes

Primary Database Protection Mode	Maximum Protection	Maximum Availability	Maximum Performance
Database Protection Policy (Status)	Highest level of primary data availability - primary database downtime is possible.	Very high level of primary data availability - no primary database downtime during runtime.	High level of primary data availability - no primary database downtime or performance effect.
Archived Log Destination Attributes (Dynamic)	MANDATORY LGWR, SYNC, AFFIRM	MANDATORY LGWR, SYNC, AFFIRM	Any combination of: LGWR SYNC, LGWR ASYNC, or ARCH
Resulting Primary Database Operating State	When the standby network is connected, there is no data divergence. A disconnected standby network results in primary instance shutdown.	When the standby network is connected, there is no data divergence. When the network is disconnected, the protection mode is lowered temporarily to the maximum performance mode until the fault has been corrected and the standby database catches up with the primary database.	Whether the standby network is connected or disconnected, the data will differ between the primary and standby databases.
Standby Database Switchover Result	When the standby network is connected, there is no data loss. If disconnected, a switchover operation is not possible.	When the standby network is connected, there is no data loss. If disconnected, a switchover operation is not possible.	When the standby network is connected, there is no data loss. If disconnected, a switchover operation is not possible.
Standby Database Failover Result	When the standby network is disconnected, there is no data loss.	When the standby network is connected, there is no data loss. If disconnected, there may be some data loss.	Whether the standby network is connected or disconnected, there is possibly some data loss depending on the transport modes.

The following sections describe all of these aspects in more detail to help you determine the correct protection mode for your Data Guard environment.

5.7.1 Maximum Protection

Maximum protection mode offers the highest level of data availability for the primary database. When used with force logging, this protection mode *guarantees*

all data that has been committed on the primary database will be available for recovery on the standby site in the event of a failure. Also, if the last participating standby database becomes unavailable, processing automatically halts on the primary database as well. This ensures that no transactions are lost when the primary database loses contact with all of its standby databases.

> **Note:** Oracle Corporation recommends that you use multiple standby databases when your business requires maximum data protection. With multiple standby databases, if one standby database becomes unavailable, the primary database can continue operations as long as at least one standby database is participating in the configuration.

When operating in maximum protection mode, the log writer process (LGWR) transmits redo records from the primary database to the standby database, and a transaction is not committed on the primary database until it has been confirmed that the transaction data is available on at least one standby database. While this can potentially decrease primary database performance, it provides the highest degree of data protection at the standby site. The impact on performance can be minimized by configuring a network with sufficient throughput for peak transaction load and with low row trip latency. Stock exchanges, currency exchanges, and financial institutions are examples of businesses that require maximum protection.

Issue the following SQL statement on the primary database to define this level of protection for the overall Data Guard configuration:

```
ALTER DATABASE SET STANDBY TO MAXIMIZE PROTECTION;
```

Standby Online Redo Logs

The standby sites you want to participate as members of the maximum protection configuration must use standby online redo logs.

Note: Because logical standby databases cannot use standby online redo logs, you cannot configure logical standby sites to participate in a maximum protection configuration. However, logical standby sites can participate in a maximum availability configuration even though they do not use standby online redo logs. Logical standby databases will receive logs from a primary database that is in maximum protection mode.

Configuring Log Transport Services to Maximize Protection

You must define at least one standby site destination with these attributes of the LOG_ARCHIVE_DEST_*n* parameter: MANDATORY, LGWR, SYNC, and AFFIRM. Also, all destinations that will be part of a maximum protection configuration must be enabled (LOG_ARCHIVE_DEST_STATE_*n*=ENABLE) and reachable on the network when you start the primary database. For example:

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/d1arch OPTIONAL'
LOG_ARCHIVE_DEST_2='SERVICE=stby MANDATORY LGWR SYNC AFFIRM'
LOG_ARCHIVE_DEST_STATE_1 = ENABLE
LOG_ARCHIVE_DEST_STATE_2 = ENABLE
```

You can define other destinations differently in a maximum protection configuration. Destinations defined with other attributes will not be considered part of the requirements for this mode, but will continue to receive redo logs.

You can defer currently enabled sites using the LOG_ARCHIVE_DEST_STATE_*n*=DEFER parameter, and then reenable the sites later. This capability has the advantage of allowing you to physically move a site and easily make it a part of the maximum protection configuration by reenabling it using the LOG_ARCHIVE_DEST_STATE_*n*=ENABLE parameter.

See Also: Section 5.8 for more information about configuring log transport services

Configuring Real Application Clusters to Maximize Protection

In a Real Application Clusters configuration, any node that loses connectivity with a standby destination will cause all other members of the cluster to stop sending data to that destination (this maintains the data integrity of the data that has been transmitted to that destination and can be recovered).

When the failed standby destination comes back up, Data Guard runs the site in resynchronization mode until the primary and standby databases are identical (no

gaps remain). Then, the standby destination can participate in the maximum protection configuration.

If the lost destination is the *last* participating standby site, then the instance on the node that loses connectivity will be shut down. Other nodes in a Real Application Clusters configuration that still have connectivity to the last standby site will recover the lost instance and continue sending to their standby site. Only when every node in a Real Application Clusters configuration loses connectivity to the last standby site will the configuration, including the primary database, be shut down.

When a failover operation occurs to a site that is participating in the maximum protection configuration, all data that was ever committed on the primary database will be recovered on the standby site.

5.7.2 Maximum Availability

Maximum availability mode offers the next highest level of data availability for the primary database. If a standby database becomes unavailable, processing does not halt on the primary database *unless* a primary database failure occurs before recovery from a network outage. Then, no data is lost up to the last transaction that was transmitted to the site. (Transactions that continued on the primary site after the network went down could be lost.) The standby database may temporarily diverge from the primary database, but upon failover to the standby database, the databases can be synchronized, and no data will be lost.

Maximum availability mode makes a best effort to write the redo records to at least one physical standby database that is configured to use the SYNC log transport mode. The maximum availability mode does not shut down the primary database instance. Instead, the protection mode is lowered temporarily to maximum performance mode until the fault has been corrected and the standby database has caught up with the primary database.

As with the maximum protection mode, the log writer process (LGWR) transmits redo logs from the primary database to the standby database. The transaction is not complete on the primary database until it has been confirmed that the transaction data is either available on the standby database, or that the data could not be received by the standby database.

This protection mode presents a potential response time degradation and a potential bottleneck for throughput. These can be minimized by configuring a network with sufficient throughput for peak transaction load, and with low row trip latency.

An example of a business that can use this data protection mode is a manufacturing plant; the risks of having no standby database for a period of time and data divergence are acceptable as long as no data is lost if failover is necessary.

Use the following SQL statement on the primary database to define this level of protection for the overall Data Guard configuration:

```
ALTER DATABASE SET STANDBY TO MAXIMIZE AVAILABILITY;
```

Standby Online Redo Logs

Although the use of standby online redo log files is optional for this mode, Oracle Corporation recommends that the physical standby sites that you want to participate as members of the maximum availability configuration be configured to use standby online redo logs. You can configure logical standby sites to participate in a maximum availability configuration even though they do not use standby online redo logs.

Configuring Log Transport Services to Maximize Availability

You set up log transport services to maximize availability exactly the same way as described for the maximum protection mode. You must define at least one site destination with these attributes on the LOG_ARCHIVE_DEST_*n* parameter: MANDATORY, LGWR, SYNC, and AFFIRM. All destinations that will be part of a maximum availability configuration must be enabled (LOG_ARCHIVE_DEST_STATE_*n*=ENABLE) and reachable on the network when you start the primary database. For example:

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/d1arch OPTIONAL'
LOG_ARCHIVE_DEST_2='SERVICE=stby MANDATORY LGWR SYNC AFFIRM'
LOG_ARCHIVE_DEST_STATE_1 = ENABLE
LOG_ARCHIVE_DEST_STATE_2 = ENABLE
```

You can define other destinations differently in a maximum availability configuration. Destinations defined with other attributes will not be considered part of the requirements for this mode.

You can defer currently enabled sites using the LOG_ARCHIVE_DEST_STATE_*n*=DEFER parameter, and then reenable the sites later. This provides the ability to physically move a site (while it is deferred) but still easily make it a part of the maximum availability configuration by reenabling it using the LOG_ARCHIVE_DEST_STATE_*n*=ENABLE parameter.

> **See Also:** Section 5.8 for more information about configuring log transport services

Configuring Real Application Clusters to Maximize Availability

In a Real Application Clusters configuration, any node that loses connectivity with a standby destination will cause all other members of the Real Application Clusters configuration to stop sending data to that destination. When the failed standby destination comes back up, Data Guard runs the site in resynchronization mode until the primary and standby databases are identical (no gaps remain). Then, the standby destination can again participate in the maximum availability configuration. Losing the last standby destination does not cause the primary database instance to shut down.

When a failover operation occurs to a site that is participating in the maximum availability configuration, all data that was ever committed on the primary database and was successfully sent to the standby database will be recovered on the standby site.

5.7.3 Maximum Performance

Maximum performance mode is the default protection mode. It offers slightly less primary data protection than maximum availability mode. During normal operations, processing on the primary database continues without regard to data availability on any standby database. If any standby destination becomes unavailable at any time, processing does not halt on the primary database and there is little or no effect on performance.

You can use the log writer process (LGWR) or the archiver process (ARC*n*) to transmit archived redo logs to the standby sites, and you can specify a synchronous or asynchronous network transmission mode. Use the maximum performance mode when availability and performance on the primary database are more important than the risk of losing a small amount of data. This protection mode can be used if your business has some other form of backup or if the loss of data would have only a small effect on your business.

You use the following SQL statement to define this level of protection for the overall Data Guard configuration:

```
ALTER DATABASE SET STANDBY TO MAXIMIZE PERFORMANCE;
```

Standby Online Redo Logs

Standby online redo logs are not required for this protection mode; therefore, both physical and logical standby sites can participate in a maximum performance configuration. Physical standby databases can use the standby redo logs if the log transport services are set up using the `LGWR` attribute.

Configuring Log Transport Services to Maximize Performance

You do not need to define standby destinations to implement the maximum performance mode. However, Oracle Corporation recommends that you configure at least one standby site; availability cannot be achieved without the presence of at least one standby site in the configuration.

If you define any destinations, you can define them with any combination of log transport services attributes: `LGWR SYNC`, `LGWR ASYNC`, or `ARCH`. Destinations that will be part of the Data Guard configuration operating under this protection mode do not have to be enabled or reachable at the time you start the primary database instance.

You can defer currently enabled sites using the `LOG_ARCHIVE_DEST_STATE_`*n*`=DEFER` parameter, and then reenable the sites later. This provides the ability to physically move a site (while it is deferred) but still easily make it a part of the maximum performance configuration by reenabling it using the `LOG_ARCHIVE_DEST_STATE_`*n*`=ENABLE` parameter.

> **See Also:** Section 5.8 for more information about configuring log transport services

Configuring Real Application Clusters to Maximize Performance

In a Real Application Clusters environment, any node that loses connectivity with a standby destination will stop transmitting archived redo logs to that destination, but all other nodes in the Real Application Clusters configuration will continue sending data to that destination. When the failed standby destination comes back up, Data Guard runs the site in resynchronization mode until the primary and standby databases are identical (no gaps remain). Then, the standby destination can again participate in the maximum performance configuration. Losing the last standby destination does not cause the primary database instance to shut down.

When a failover operation occurs to any standby site, data that was received from the primary database will be recovered on the standby database up to the last transactionally consistent point in time. In a single-instance configuration, this means all data received will be recovered. In a failover situation, it is possible to lose some transactions from one or more logs that have not yet been transmitted.

5.8 Configuring Log Transport Services for Data Protection

Log transport services supports three data protection modes, as explained in Section 5.7. You configure data protection modes by:

- Specifying LGWR or ARCH for the Redo Log Writing Process

- Specifying SYNC or ASYNC for the Network Transmission Mode
- Specifying AFFIRM or NOAFFIRM for the Disk Write Option
- Comparing Network and Disk I/O Methods
- Setting the Data Protection Mode for an Overall Failure Resolution Policy

Table 5–7 lists the protection modes and the settings required to implement each mode. You can set other variations of archive destination attributes and log transport services options; however, the following table identifies the most common settings.

Table 5–7 Configuring Log Transport Services Protection Modes

Data Protection Mode	Log Writing Process Option	Network Transmission Mode	Disk Write Option	Redo Log Reception Option	Supported on ...
Maximum Protection	LGWR	SYNC	AFFIRM	Standby redo logs are required for physical standby databases	Physical standby databases
Maximum Availability	LGWR	SYNC	AFFIRM	Standby redo logs are optional, but recommended for physical standby databases	Physical and logical standby databases
Maximum Performance	LGWR or ARCH	ASYNC or SYNC	AFFIRM or NOAFFIRM	Standby redo logs are optional	Physical and logical standby databases

5.8.1 Specifying LGWR or ARCH for the Redo Log Writing Process

Timely protection of application data requires use of the log writer process (LGWR) to propagate primary database modifications to one or more standby databases. This is achieved using the LGWR attribute of the LOG_ARCHIVE_DEST_*n* initialization parameter.

Attribute	Example	Default
{LGWR | ARCH}	LOG_ARCHIVE_DEST_3='SERVICE=stby1 LGWR'	ARCH

Choosing the LGWR attribute indicates that the log writer process (LGWR) will concurrently create the archived redo logs as the online redo log is populated. Depending on the configuration, this may require the log writer process to also transmit redo logs to remote archival destinations.

Choosing the ARCH attribute indicates that one of the archiver processes (ARC*n*) will create archived redo logs on the primary database and also transmit redo logs for archiving at specified destinations. This is the default setting.

When you set the LOG_ARCHIVE_DEST_*n* parameter to specify LGWR instead of ARCH, the behavior is as follows. Instead of waiting for the online redo log to switch at the primary database and then let the archiver process write the entire archived redo log at the standby destination all at once, the log writer process creates a new redo log at the standby site that reflects the log sequence number (and size) of the current online redo log of the primary database. Then, as redo is generated at the primary database, it is also propagated to the standby database redo log file. Based on whether you specify the SYNC or ASYNC network transmission mode, the propagation will either be immediate or asynchronous.

The LGWR and ARCH attributes are mutually exclusive. Therefore, you cannot specify both attributes for the same destination. However, you can specify either the LGWR or the ARC*n* attribute for individual destinations. This allows you to specify that the log writer process writes to redo logs and archives for some destinations while the archiver process archives redo logs to other destinations.

See Also: ARCH and LGWR in Chapter 12

Note: The log writer process will transmit log buffers to the archival destination only if the primary database is in ARCHIVELOG mode.

5.8.2 Specifying SYNC or ASYNC for the Network Transmission Mode

When using the log writer process to archive redo logs, you can specify synchronous (SYNC) or asynchronous (ASYNC) network transmission of redo logs to archiving destinations using the SYNC or ASYNC=*blocks* attributes. If you do not specify either the SYNC or ASYNC attribute, the default is the SYNC network transmission mode.

The SYNC Network Transmission Method

If you specify the SYNC attribute, all network I/O operations are performed synchronously, in conjunction with each write operation to the online redo log. Control is not returned to the executing application or user until the redo information is received by the standby site. This attribute has the potential to affect primary database performance adversely, but provides the highest degree of data

protection at the destination site. Synchronous transmission is required for no-data-loss environments.

When you specify the LGWR and SYNC attributes to transmit redo logs to multiple standby destinations, you can specify either the SYNC=PARALLEL or SYNC=NOPARALLEL option for each destination:

- If you specify SYNC=NOPARALLEL, the LGWR process initiates an I/O operation to the first destination and waits until the operation completes before initiating an I/O operation to the next destination, and so on.
- If you specify SYNC=PARALLEL, the LGWR process initiates an I/O operation to each destination at the same time. However, the LGWR process waits for each of the I/O operations to complete before continuing, which is the same as performing multiple, synchronous I/O operations simultaneously.

> **Note:** Because the PARALLEL and NOPARALLEL options make a difference only when multiple destinations are defined, Oracle Corporation recommends that all destinations use the same PARALLEL or NOPARALLEL option.

If you specify the SYNC attribute without specifying an option, the default option depends on whether the location is local or remote, and on whether you chose the ARCH or LGWR process for the destination. Because the PARALLEL qualifier is not supported for the ARCH process, the default will always be NOPARALLEL. Table 5–8 shows the default PARALLEL and NOPARALLEL options.

Table 5–8 Default PARALLEL and NOPARALLEL Options

Local or Remote Destination	Log Writing Process	Network Transmission Mode	Default SYNC Option
LOCATION=*location*	ARCH	SYNC	NOPARALLEL
SERVICE=*service*	ARCH	SYNC	NOPARALLEL
LOCATION=*location*	LGWR	SYNC	NOPARALLEL
SERVICE=*service*	LGWR	SYNC	PARALLEL

The ASYNC Network Transmission Method

If you specify the ASYNC=*blocks* attribute, all network I/O operations are performed asynchronously, and control is returned to the executing application or user immediately. You can specify a block count to determine the size of the SGA

network buffer to be used. Block counts from 0 to 20,480 are allowed. The attribute allows the optional suffix value K to represent 1,000 (the value "1K" indicates 1,000 512-byte blocks). In general, for slower network connections, use larger block counts.

See Also: Appendix C, "Log Writer Asynchronous Network I/O"

When you use the ASYNC attribute, there are several events that cause the network I/O to be initiated:

- If the LGWR request exceeds the currently available buffer space, the existing buffer is transmitted to the standby database. The LGWR process stalls until sufficient buffer space can be reclaimed, but this should seldom occur.
- A primary database log switch forces any buffered redo data to be transmitted to the standby database before the log switch operation completes.
- The primary database is shut down normally. An immediate shutdown of the primary database results in the buffered redo data being discarded. A standby database shutdown also causes the buffered redo data to be discarded.
- The primary database has no redo activity for a period of time. The duration of database inactivity is determined by the system and cannot be modified by the user.
- If the rate of redo generation exceeds the runtime network latency, then the LGWR request will be buffered if sufficient space is available. Otherwise, the existing buffer is transmitted to the standby database. The LGWR process stalls until sufficient buffer space can be reclaimed, but this should seldom occur.

Attribute	Example	Default
{SYNC [=NOPARALLEL | PARALLEL] | ASYNC [=blocks]}	LOG_ARCHIVE_DEST_3= 'SERVICE=stby1 LGWR SYNC'	SYNC

See Also: SYNC and ASYNC in Chapter 12

Table 5–9 identifies the attributes of the LOG_ARCHIVE_DEST_*n* initialization parameter that are used to specify the transmission mode.

Table 5–9 Transmission Mode Attributes

Process Attribute	Method Attribute	Description
LGWR	SYNC[=NOPARALLEL | PARALLEL]	The primary database log writer process will synchronously transmit the online redo log contents. Control will not be returned to the application until the data safely resides on the standby site. This mode has the best guarantee of data protection on the destination database, but the highest performance effect on the primary database.
LGWR	ASYNC=blocks	The primary database log writer process will asynchronously transmit the online redo log contents to the destination database. Control will be returned to the application processes immediately, even if the data has not reached the destination. Block counts up to 20,480 blocks are allowed. This mode has a reasonable degree of data protection on the destination database, with minimal performance effect on the primary database.
ARCH	SYNC[=NOPARALLEL]	The primary database archiver process will synchronously transmit the online redo log contents. This mode has the lowest degree of data protection on the destination database, and a slight performance effect on the primary database.

5.8.3 Specifying AFFIRM or NOAFFIRM for the Disk Write Option

The [NO]AFFIRM attribute of the LOG_ARCHIVE_DEST_*n* initialization parameter is used to specify whether log archiving disk write I/O operations on the standby database are to be performed synchronously or asynchronously. By default, disk write operations are performed asynchronously.

It is necessary for the primary database to receive acknowledgment of the availability of the modifications on the standby database in a maximum protection or maximum availability environment. This is achieved using both the SYNC and AFFIRM attributes of the LOG_ARCHIVE_DEST_*n* initialization parameter. The SYNC attribute indicates that synchronous network transmission is necessary, and the AFFIRM attribute indicates that synchronous archived redo log disk write I/O

operations are necessary. Together, these attributes ensure that primary database modifications are available on the standby database.

This attribute applies to local and remote archive destination disk I/O operations and standby redo log disk write I/O operations.

Attribute	Example	Default
[NO]AFFIRM	LOG_ARCHIVE_DEST_3= 'SERVICE=stby1 LGWR SYNC AFFIRM'	NOAFFIRM

Note: The [NO]AFFIRM attribute has no effect on primary database online redo log disk I/O operations.

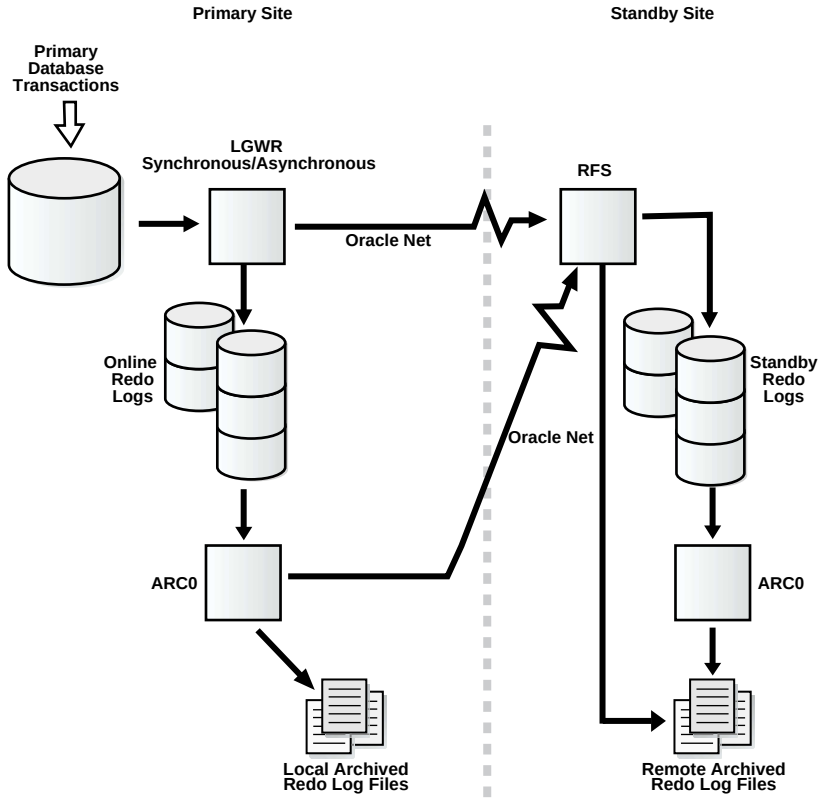
See Also: AFFIRM and NOAFFIRM in Chapter 12

5.8.4 Comparing Network and Disk I/O Methods

Note: This section applies to physical standby databases only.

Online redo logs transferred from the primary database are received by the remote file server process (RFS) on the physical standby site and can be stored as either standby redo logs or archived redo logs as shown in Figure 5–2.

Figure 5–2 Redo Log Reception Options



Oracle9*i* Data Guard introduces the concept of **standby redo logs.** Standby redo logs are required for physical standby databases running in the overall maximum protection mode. Standby redo logs form a separate pool of log file groups and provide the following advantages over archived online redo logs:

- Because standby redo logs are preallocated files, they avoid the operating system overhead of file system metadata updates common with sequential files.
- Standby redo logs can reside on raw devices, thus providing I/O write performance equal to that of the primary database.
- Standby redo logs can be multiplexed using multiple members, improving reliability over archived redo logs.

- If primary database failure occurs, redo data written to standby redo logs can be fully recovered.
- Standby redo logs are required to implement the maximum protection level of data protection (no-data-loss) disaster recovery solution with physical standby databases.
- Standby redo logs are required for cascading standby databases. (See Appendix E for more information.)

You should also configure standby redo logs on the primary database. Even though the standby redo logs are not used when the database is running in the primary role, configuring the standby redo logs on the primary database is recommended in preparation for an eventual switchover operation.

5.8.4.1 Standby Redo Logs

Standby redo logs are created using the `ADD STANDBY LOGFILE` clause of the `ALTER DATABASE` statement. Although standby redo logs are only used when the database is running in the standby role, Oracle Corporation recommends that you create standby redo logs so that the primary database can switch roles easily and quickly without additional DBA intervention.

To verify that standby redo logs have been created, query the `V$STANDBY_LOG` view (displays standby redo log status as `ACTIVE` or `INACTIVE`) or the `V$LOGFILE` view. The following example queries the `V$LOGFILE` view:

```
SQL> SELECT * FROM V$LOGFILE WHERE TYPE = 'STANDBY';
```

Additional standby log group members can be added later to provide another level of reliability against disk failure on the standby site.

Once you have created the standby redo logs, they are automatically selected as the repository of redo data received from the primary site, if all of the following conditions apply:

- The primary database archive redo log destination uses the log writer process (LGWR) to archive online redo logs.
- The archiver process (ARC*n*) is active on the standby database.
- The size of a standby redo log exactly matches at least one of the primary database online redo logs.
- The standby redo log has been successfully archived. In other words, the standby redo logs must be empty of primary database redo data.

For example, if the primary database uses two online redo log groups whose log size is 100K and 200K, respectively, then the standby database should have standby redo log groups with those same sizes. However, it may be necessary to create additional standby log groups on the standby database, so that the archival operation has time to complete before the standby redo log is used. If the primary database is operating in maximum protection mode, and a standby redo log cannot be allocated, the primary database instance may be shut down immediately. (In some cases, the primary database may wait for the standby redo log to become available.) Therefore, be sure to allocate an adequate number of standby redo logs.

When you use Real Application Clusters, the various standby redo logs are shared among the various primary database instances. Standby redo log groups are not dedicated to a particular primary database thread.

Standby redo logs must be archived before the data can be applied to the standby database. The standby archival operation occurs automatically, even if the standby database is not in ARCHIVELOG mode. However, the archiver process must be started on the standby database. Note that the use of the archiver process (ARC*n*) is a requirement for selection of a standby redo log.

Because of this additional archival operation, using standby redo logs may cause the standby database to lag further behind the primary database than when archiving directly from the primary database to standby destinations. However, the use of standby redo logs ultimately improves redo data availability on the standby database.

5.8.4.2 Choosing the Number of Standby Redo Log Groups

The best way to determine the appropriate number of standby redo log groups for a database instance is to test different configurations. The minimum configuration should have one more standby redo log group than the primary database. The optimum configuration has a few more groups than the primary database.

In some cases, a standby database instance may require only two groups. In other situations, a database may require additional groups to guarantee that a recycled group is always available to receive redo information from the primary database. During testing, the easiest way to determine if the current standby log configuration is satisfactory is to examine the contents of the RFS process trace file and the database alert log. If messages indicate that the RFS process frequently has to wait for a group because archiving has not completed, add more standby groups.

Consider the parameters that can limit the number of standby redo log groups before setting up or altering the configuration of the standby redo log groups. The

following parameters limit the number of standby redo log groups that you can add to a database:

- The MAXLOGFILES parameter of the CREATE DATABASE statement for the primary database determines the maximum number of groups of standby redo logs per standby database. The only way to override this limit is to re-create the primary database or control file.
- The LOG_FILES parameter can temporarily decrease the maximum number of groups of standby redo logs for the duration of the current instance.
- The MAXLOGMEMBERS parameter of the CREATE DATABASE statement used for the primary database determines the maximum number of members per group. The only way to override this limit is to re-create the primary database or control file.

See Also: *Oracle9i SQL Reference*

5.8.4.3 Creating Standby Redo Log Groups

Plan the standby redo log configuration of a database and create all required groups and members of groups after you have instantiated the standby database. However, there are cases where you might want to create additional groups or members. For example, adding groups to a standby redo log configuration can correct redo log group availability problems. To create new standby redo log groups and members, you must have the ALTER DATABASE system privilege. A database can have as many groups as the value of MAXLOGFILES.

To create a new group of standby redo logs, use the ALTER DATABASE statement with the ADD STANDBY LOGFILE clause.

The following statement adds a new group of standby redo logs to the database:

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE
  2> ('/oracle/dbs/log1c.rdo','/oracle/dbs/log2c.rdo') SIZE 500K;
```

You can also specify a number that identifies the group using the GROUP option:

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE GROUP 10
  2> ('/oracle/dbs/log1c.rdo','/oracle/dbs/log2c.rdo') SIZE 500K;
```

Using group numbers can make administering standby redo log groups easier. However, the group number must be between 1 and the value of the MAXLOGFILES parameter. Do not skip redo log file group numbers (that is, do not number groups 10, 20, 30, and so on), or you will consume additional space in the standby database control file.

5.8.4.4 Adding Standby Redo Log Members to an Existing Group

In some cases, it might not be necessary to create a complete group of standby redo logs. A group could already exist, but not be complete because one or more members were dropped (for example, because of disk failure). In this case, you can add new members to an existing group.

Use fully qualified filenames of new log members to indicate where the file should be created. Otherwise, files will be created in either the default or current directory of the database server, depending upon your operating system.

To add new standby redo log group members, use the ALTER DATABASE statement with the ADD STANDBY LOGFILE MEMBER parameter. The following statement adds a new member to redo log group number 2:

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE MEMBER '/oracle/dbs/log2b.rdo'
  2> TO GROUP 2;
```

> **See Also:** *Oracle9i SQL Reference* for a complete description of the ALTER DATABASE statement

5.8.4.5 Specifying Storage Locations for Archived Redo Logs and Standby Redo Logs

When archived redo logs are used, use the STANDBY_ARCHIVE_DEST initialization parameter on the standby database to specify the directory in which to store the archived redo logs. Log transport services use this value in conjunction with the LOG_ARCHIVE_FORMAT parameter to generate the archived redo log filenames on the standby site.

Parameter	Indicates	Example
STANDBY_ARCHIVE_DEST	Directory in which to place archived online redo logs	STANDBY_ARCHIVE_DEST= /arc_dest/

Parameter	Indicates	Example
LOG_ARCHIVE_FORMAT	Format for filenames of archived online redo logs	LOG_ARCHIVE_FORMAT = "log_%t_%s.arc" **Note:** The %s corresponds to the sequence number. The %t, which is required for Real Application Clusters configurations, corresponds to the thread.

Log transport services store the fully qualified filenames in the standby control file. Log apply services use this information to perform recovery operations on the standby database. You can access this information through the V$ARCHIVED_LOG view on the standby database:

```
SQL> SELECT NAME FROM V$ARCHIVED_LOG;
NAME
--------------------------------------------------------------------------------
/arc_dest/log_1_771.arc
/arc_dest/log_1_772.arc
/arc_dest/log_1_773.arc
/arc_dest/log_1_774.arc
/arc_dest/log_1_775.arc
```

Note: If you use the optional TEMPLATE attribute of the LOG_ARCHIVE_DEST_*n* parameter to define a directory specification and format for archived redo logs at the standby destination, the TEMPLATE attribute overrides any values entered in the STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT initialization parameters. You can specify this attribute in either the primary or standby initialization parameter file, but the attribute applies only to the database role that is archiving.

When standby redo logs are used, the LOG_ARCHIVE_DEST_*n* initialization parameter (where *n* is a value from 1 to 10) on the standby database specifies the directory in which to store standby redo logs.

Parameter	Indicates	Example
LOG_ARCHIVE_DEST_*n*	The directory for storage of standby redo logs on the standby site	LOG_ARCHIVE_DEST_1= 'LOCATION=/oracle/stby/arc/' **Note:** If this parameter is not defined, the value of the STANDBY_ARCHIVE_DEST parameter is used.
LOG_ARCHIVE_FORMAT	Format for filenames of archived online redo logs	LOG_ARCHIVE_FORMAT = "log_%t_%s.arc" **Note:** The %s corresponds to the sequence number. The %t, which is required for Real Application Clusters configurations, corresponds to the thread.

Note: When using standby redo logs, you must enable the archiver process (ARC*n*) on the standby database. Oracle Corporation recommends that you always set the LOG_ARCHIVE_START initialization parameter to true on the standby database.

5.8.5 Setting the Data Protection Mode for an Overall Failure Resolution Policy

A failure resolution policy determines what actions will occur on a primary database when the last standby destination fails to archive redo logs.

To set the highest level of data protection, place the primary database in maximum protection mode using the SET STANDBY DATABASE clause of the ALTER DATABASE statement as shown in the following example:

```
SQL> ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE PROTECTION;
```

Note: This statement must be issued on the primary database.

If connectivity between the primary and standby database should now be lost, the primary database will shut down. When using this level of data protection, standby databases must have two or more standby redo log groups. Also, one or more primary database archived redo log destinations must have LGWR and SYNC attributes specified. The functionality of the AFFIRM attribute is implicitly set.

You can revert to a mode that allows some data divergence by placing the primary database in MAXIMIZE AVAILABILITY mode using the following statement:

```
SQL> ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE AVAILABILITY;
```

See Also: *Oracle9i SQL Reference* for complete ALTER DATABASE syntax

5.9 Comparing Network and Disk I/O Methods

How you set the destination disk write I/O attributes NOAFFIRM and AFFIRM, and the SYNC and ASYNC network transmission attributes, affects performance on the primary database and data availability on the standby database. (The SYNC and ASYNC attributes affect network I/O performance only when the log writer process is used.) Table 5–10 shows a comparison of primary database performance to data protection on the standby database when various combinations of archive process, network I/O, and disk I/O attribute settings are used.

Table 5–10 Comparing Network and Disk I/O Methods

Archive Process Attribute Setting	Network I/O Attribute Setting	Disk I/O Attribute Setting	Primary Database Performance	Data Protection
LGWR	ASYNC	NOAFFIRM	High	Redo logs are archived asynchronously on the network as well as to disk, providing high primary database performance but lower data protection on the standby database.
LGWR	ASYNC	AFFIRM	High	Redo logs are archived asynchronously on the network but synchronously to disk, providing high primary database performance and slightly higher data protection on the standby database than when the NOAFFIRM attribute is used.
LGWR	SYNC	NOAFFIRM	Fair	Redo logs are archived synchronously on the network but asynchronously to disk, providing fair primary database performance and higher data protection on the standby database.

Table 5–10 (Cont.) Comparing Network and Disk I/O Methods

Archive Process Attribute Setting	Network I/O Attribute Setting	Disk I/O Attribute Setting	Primary Database Performance	Data Protection
LGWR	SYNC	AFFIRM	Low	Redo logs are archived synchronously on the network and synchronously to disk, providing low primary database performance but the highest data protection on the standby database.
ARCH	SYNC	NOAFFIRM	Fair	Redo logs are archived synchronously on the network and asynchronously to disk, providing fair primary database performance and the lowest data protection on the standby database.
ARCH	SYNC	AFFIRM	Low	Redo logs are archived synchronously on the network and to disk, providing lower primary database performance but high data availability on the standby database.

5.10 Network Tuning for Log Transport Services

The process of archiving redo logs involves reading a buffer from the redo log and writing it to the archive log location. When the destination is remote, the buffer is written to the archive log location over the network using Oracle Net services.

The default archive log buffer size is 1 megabyte. The default transfer buffer size for Oracle Net is 2 kilobytes. Therefore, the archive log buffer is divided into units of approximately 2 kilobytes for transmission. These units could get further divided depending on the maximum transmission unit (MTU) of the underlying network interface.

The Oracle Net parameter that controls the transport size is **session data unit (SDU).** This parameter can be adjusted to reduce the number of network packets that are transmitted. This parameter allows a range of 512 bytes to 32 kilobytes.

For optimal performance, set the Oracle Net SDU parameter to 32 kilobytes for the associated SERVICE destination parameter.

The following example shows a database initialization parameter file segment that defines a remote destination netserv:

```
LOG_ARCHIVE_DEST_3='SERVICE=netserv'
SERVICE_NAMES=srvc
```

The following example shows the definition of that service name in the `tnsnames.ora` file:

```
netserv=(DESCRIPTION=(SDU=32768)(ADDRESS=(PROTOCOL=tcp)(HOST=host) (PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=srvc)(ORACLE_HOME=/oracle)))
```

The following example shows the definition in the `listener.ora` file:

```
LISTENER=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)
(HOST=host)(PORT=1521))))

SID_LIST_LISTENER=(SID_LIST=(SID_DESC=(SDU=32768)(SID_NAME=sid)
(GLOBALDBNAME=srvc)(ORACLE_HOME=/oracle)))
```

If you archive to a remote site using high-latency / high-bandwidth connections, you can improve performance by increasing the TCP send and receive window sizes. Use caution, however, because this may adversely affect networked applications that do not exhibit the same characteristics as archiving. This method consumes a large amount of system resources.

See Also: *Oracle9i Net Services Administrator's Guide*

You can also use cascading standby databases to off-load network processing from the primary database to a standby database. See Appendix E for more information.

5.11 Log Transport Services Monitoring

This section describes manual methods of monitoring redo log archival activity for the primary database.

The Oracle9*i* Data Guard Manager graphical user interface automates many of the tasks involved in monitoring a Data Guard environment. See *Oracle9i Data Guard Broker* and the Data Guard Manager online help for more information.

5.11.1 Gathering Redo Log Archival Information

Enter the following query on the primary database to determine the current redo log sequence numbers:

```
SQL> SELECT THREAD#, SEQUENCE#, ARCHIVED, STATUS FROM V$LOG;

THREAD#   SEQUENCE#  ARC  STATUS
--------  ---------  ---  ------
       1        947  YES   ACTIVE
```

```
       1        948  NO    CURRENT
```

Enter the following query at the primary database to determine the most recently archived redo log file:

```
SQL> SELECT MAX(SEQUENCE#) FROM V$ARCHIVED_LOG;

MAX(SEQUENCE#)
--------------
          947
```

Enter the following query at the primary database to determine the most recently archived redo log file to each of the archive destinations:

```
SQL> SELECT DESTINATION, STATUS, ARCHIVED_THREAD#, ARCHIVED_SEQ#
  2> FROM V$ARCHIVE_DEST_STATUS
  3> WHERE STATUS <> 'DEFERRED' AND STATUS <> 'INACTIVE';

DESTINATION         STATUS  ARCHIVED_THREAD#  ARCHIVED_SEQ#
------------------  ------  ----------------  -------------
/private1/prmy/lad   VALID                 1            947
standby1             VALID                 1            947
```

The most recently archived redo log file should be the same for each archive destination listed. If it is not, a status other than VALID may identify an error encountered during the archival operation to that destination.

You can issue a query at the primary database to find out if a log has not been sent to a particular site. Each archive destination has an ID number associated with it. You can query the DEST_ID column of the V$ARCHIVE_DEST fixed view on the primary database to identify archive destination IDs.

Assume the current local archive destination is 1, and one of the remote standby archive destination IDs is 2. To identify which logs have not been received by this standby destination, issue the following query:

```
SQL> SELECT LOCAL.THREAD#, LOCAL.SEQUENCE# FROM
  2> (SELECT THREAD#, SEQUENCE# FROM V$ARCHIVED_LOG WHERE DEST_ID=1)
  3> LOCAL WHERE
  4> LOCAL.SEQUENCE# NOT IN
  5> (SELECT SEQUENCE# FROM V$ARCHIVED_LOG WHERE DEST_ID=2 AND
  6> THREAD# = LOCAL.THREAD#);

  THREAD#  SEQUENCE#
---------  ---------
  1        12
```

```
1          13
1          14
```

See Also: Appendix A, "Troubleshooting the Standby Database" to learn more about monitoring the archiving status of the primary database

5.11.2 Setting Archive Tracing

To see the progression of the archiving of redo logs to the standby site, set the LOG_ARCHIVE_TRACE parameter in the primary and standby initialization parameter files. See Section 6.4.8 for complete details and examples.

6

Log Apply Services

This chapter describes how to manage a standby database. It includes the following topics:

6.1 Introduction to Log Apply Services

Log apply services automatically apply archived redo logs to maintain transactional synchronization with the primary database, and allow transactionally consistent read-only access to the data.

The main difference between physical and logical standby databases is the manner in which log apply services apply the archived redo logs. For physical standby databases, log apply services maintain the standby database in managed recovery mode or open read-only mode. For logical standby databases, log apply services maintain the standby database in SQL apply mode. The following list summarizes these modes:

- **Managed recovery mode** (physical standby databases only)

  In this mode, log transport services archive logs to the standby site, and log apply services automatically apply these logs. If you want maximum protection against data loss or corruption, then maintain the standby database in managed recovery mode in a Data Guard environment.

- **Read-only mode** (physical standby databases only)

  Use read-only mode for supplemental reporting of data contained in the primary database. If you want to use the standby database for reporting purposes, then open it in read-only mode in a Data Guard environment. Log apply services cannot apply archived redo logs to the standby database when it is in this mode, but you can still execute queries on the database. While the standby database is in read-only mode, it can continue to receive archived redo logs from the primary database.

- **SQL apply mode** (logical standby databases only)

  Log apply services manage logical standby databases in SQL apply mode only. Only logical standby databases can be opened in read / write mode, but the target tables for the regenerated SQL statements are available only in read-only mode for reporting purposes. The SQL apply mode supports the application of SQL statements and reporting activities simultaneously to the logical standby database.

The sections in this chapter describe these modes and log apply services in more detail.

6.2 Applying SQL Statements to Logical Standby Databases

The main difference between physical and logical standby databases is the manner in which log apply services apply the archived redo logs. For a logical standby database, log apply services manage the application of log information in archived redo logs from the primary database by transforming transaction information into SQL syntax.

This is done in SQL apply mode, which converts the redo data into SQL statements and applies the SQL statements to an open logical standby database. Because the logical standby database remains open, tables that are maintained can be used simultaneously for other tasks such as reporting, summations, and queries.

The logical standby database uses the following processes:

- Remote file server (RFS)

  The remote file server (RFS) process receives archived redo logs from the primary database.

- Archiver (ARC*n*)

  The ARC*n* process archives the archived redo logs to be applied by the logical standby process (LSP).

- Logical standby process (LSP)

  The logical standby process is the coordinator process for a collection of parallel server processes (P*nnn*) that work concurrently to read, prepare, build, analyze, and apply completed SQL transactions from the archived redo logs. The number of processes used must be at least five and is controlled by the PARALLEL_MAX_SERVERS initialization parameter.

Logical standby databases use supplemental logging, an Oracle feature that adds information into the redo stream, so that LogMiner (and technologies like logical standby databases that use LogMiner) can correctly interpret the changes in the archived redo log. Logical standby databases require that primary key and unique index columns be supplementally logged so that row changes can be properly found and scheduled.

> **See Also:** Section 4.1 and Section 13.4 for more information about supplemental logging and the ALTER DATABASE ADD SUPPLEMENTAL LOG DATA statement

The following sections describe how log apply services work with logical standby databases:

- Managing SQL Apply Operations
- Summary of the DBMS_LOGSTDBY PL/SQL Supplied Package
- Delaying the Application of Archived Redo Logs
- Ensuring That Redo Logs Are Being Applied

6.2.1 Managing SQL Apply Operations

Using the DBMS_LOGSTDBY PL/SQL supplied package, you can:

- Allow controlled access to tables in the standby database that may require maintenance
- Provide a way to skip applying archived redo logs to selected tables or entire schemas in the standby database, and describe how exceptions should be handled
- Manage initialization parameters used by log apply services
- Ensure supplemental logging is enabled properly
- Describe a set of operations that should not be applied to the logical standby database

- Describe what to do in the event of a failure

You can use procedures in the DBMS_LOGSTDBY package to stop applying SQL statements or continue applying SQL statements after filtering out unsupported datatypes or statements. For example, DDL statements that are applied to the primary database can be filtered or *skipped* so that they are not applied on the logical standby database. You can use the procedures in the DBMS_LOGSTDBY package to:

- Avoid applying DML or DDL changes for temporary tables
- Avoid applying any CREATE, ALTER, or DROP INDEX operations
- Log the error and continue applying archived redo logs to the logical standby database when an error occurs on a DDL statement
- Stop log apply services and SQL apply operations, and wait for the DBA to specify what action should be taken when an error occurs on a DDL statement

Some, all, or none of the preceding actions can be made for the same logical standby database.

6.2.2 Summary of the DBMS_LOGSTDBY PL/SQL Supplied Package

The DBMS_LOGSTDBY PL/SQL package provides procedures to manage tasks on logical standby databases.

Table 6–1 summarizes the procedures of the DBMS_LOGSTDBY PL/SQL package. See *Oracle9i Supplied PL/SQL Packages and Types Reference* for complete information about the DBMS_LOGSTDBY package.

Table 6–1 Procedures of the DBMS_LOGSTDBY PL/SQL Package

Subprograms	Description
APPLY_SET	Allows you to set the values of specific initialization parameters to configure and maintain SQL apply operations.
APPLY_UNSET	Resets the value of specific initialization parameters to the system default values.
BUILD	Ensures supplemental logging is enabled properly and builds the LogMiner dictionary.
GUARD_BYPASS_OFF	Reenables the database guard that you bypassed previously with the GUARD_BYPASS_ON procedure.
GUARD_BYPASS_ON	Allows the current session to bypass the database guard so that tables in a logical standby database can be modified.

Table 6–1 (Cont.) Procedures of the DBMS_LOGSTDBY PL/SQL Package

Subprograms	Description
INSTANTIATE_TABLE	Creates and populates a table in the standby database from a corresponding table in the primary database.
SKIP	Allows you to specify what database operations that are done on the primary database will not be applied to the logical standby database.
SKIP_ERROR	Specifies criteria to follow if an error is encountered. You can stop SQL apply operations or ignore the error.
SKIP_TRANSACTION	Specifies transaction identification information to skip (ignore) while applying specific transactions to the logical standby database.
UNSKIP	Modifies the options set in the SKIP procedure.
UNSKIP_ERROR	Modifies the options set in the SKIP_ERROR procedure.
UNSKIP_TRANSACTION	Modifies the options set in the SKIP_TRANSACTION procedure.

6.2.3 Delaying the Application of Archived Redo Logs

Specifying an apply delay interval (in minutes) on the primary database is the same for both logical and physical standby databases. However, you can also set up an apply delay on the standby databases that overrides the delay interval that you had set on the primary database. On a logical standby database, if the primary database is no longer available, you can cancel the apply delay by specifying the following command:

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_UNSET('APPLY_DELAY');
```

6.2.4 Ensuring That Redo Logs Are Being Applied

In addition to using the DBMS_LOGSTDBY package to perform management tasks, you can verify information about the status of SQL apply operations to a logical standby database using views.

For example, to verify that the archived redo logs are being applied, query the V$LOGSTDBY view. This view provides information about the processes that are reading redo log information and applying it to the logical standby databases. You can also query the DBA_LOGSTDBY_PROGRESS view to find out the progress of SQL apply operations. The V$LOGSTDBY_STATS view shows the state of the coordinator

process and information about the SQL transactions that have been applied to the logical standby database.

> **See Also:** Chapter 9, "Managing a Logical Standby Database" and Chapter 14, "Views"

6.3 Applying Logs to Physical Standby Databases

The physical standby database uses several processes to achieve the automation necessary for disaster recovery and high availability. On the standby database, log apply services use the following processes:

- Remote file server (RFS)

  The remote file server (RFS) process receives redo logs from the primary database either in the form of archived redo logs or standby redo logs.

- Archiver (ARC*n*)

  If standby redo logs are being used, the ARC*n* process archives the standby redo logs that are to be applied by the managed recovery process (MRP).

- Managed recovery process (MRP)

  The managed recovery process (MRP) applies information from the archived redo logs or standby redo logs to the standby database.

Log apply services, in coordination with log transport services, manage the standby database by automatically applying archived redo logs to maintain transactional synchronization with the primary database.

Log apply services can apply logs to a physical standby database when the database is performing recovery (for example, in managed recovery mode), but not when it is open for reporting access (for example, in read-only mode).

For physical standby databases, you can easily change between managed recovery mode and read-only mode. In most implementations of a Data Guard environment, you may want to make this change at various times to either:

- Synchronize a physical standby database used primarily for reporting
- Check that data is correctly applied to a physical standby database that is used primarily for disaster protection

Note: Log transport services to a physical standby database require that the configuration uses a dedicated server process in managed recovery mode. Although the read-only mode allows a shared server process, you must have a dedicated server again once you change from read-only mode to managed recovery mode.

Table 6–2 summarizes the basic tasks for configuring log apply services.

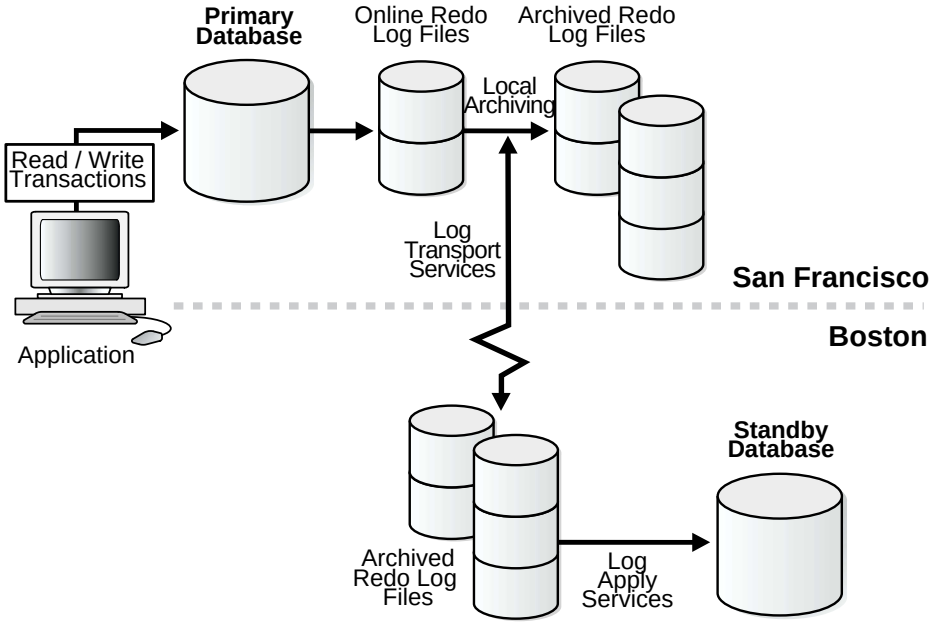
Table 6–2 Task List: Configuring Log Apply Services

Step	Task	See ...
1	Start the standby instance and mount the standby database.	Section 6.3.2
2	Enable managed recovery or read-only mode.	Section 6.3.2 or Section 6.3.5, respectively
3	If using managed recovery mode, set initialization parameters to automatically resolve archive gaps.	Section 6.5 and the *Oracle9i Net Services Administrator's Guide*
4	Monitor log apply services.	Section 6.4

6.3.1 Managed Recovery Mode

Log transport services automate archiving to a standby database. Log apply services keep the standby database synchronized with the primary database by waiting for archived logs from the primary database and then automatically applying them to the standby database, as shown in Figure 6–1.

Figure 6–1 Automatic Updating of a Standby Database



6.3.2 Starting Managed Recovery for Physical Standby Databases

After all necessary parameter and network files have been configured, you can start the standby instance. If the instance is not started and mounted, the standby database cannot receive archived redo logs that are automatically copied to the standby site from the primary database by log transport services.

Step 1 Start the standby instance.

To start the physical standby database instance, perform the following steps:

1. Connect to the physical standby database instance. For example:

```
SQL> CONNECT SYS/CHANGE_ON_INSTALL@standby1 AS SYSDBA
```

2. Start the Oracle instance at the physical standby database without mounting the database. (Starting a physical standby database instance requires the NOMOUNT qualifier.) For example:

```
SQL> STARTUP NOMOUNT PFILE=initSTANDBY.ora;
```

3. Mount the physical standby database using the following statement:

```
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

Step 2 Initiate log apply services.

As log transport services copy the archived redo logs to the standby site, log apply services can automatically apply them to the standby database.

Log apply services can run as a foreground session or as a background process with control options such as DELAY, DEFAULT DELAY, [NO]EXPIRE, FINISH, NEXT, NODELAY, [NO]PARALLEL, SKIP, THROUGH...SWITCHOVER, and [NO]TIMEOUT.

- To start a foreground session, issue the SQL statement with or without the TIMEOUT or NO TIMEOUT keywords. For example:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE;
```

  If you started a foreground session, by default, control is not returned to the command prompt after you execute the RECOVER statement unless you include the DISCONNECT keyword; this is the expected foreground behavior.

  You can start a detached server process and immediately return control to the user by using the DISCONNECT FROM SESSION option to the ALTER DATABASE statement. Note this does not disconnect the current SQL session. For example:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT
  2> FROM SESSION;
```

- To start a background session, you *must* use the DISCONNECT [FROM SESSION] keyword on the SQL statement, and you can optionally include the NO TIMEOUT keyword. For example:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION NO TIMEOUT;
```

  This statement starts a detached server process and immediately returns control to the user. While the managed recovery process is performing recovery in the background, the foreground process that issued the RECOVER statement can continue performing other tasks. This does not disconnect the current SQL session.

  When you start managed recovery using either of these SQL statements, you can also include any of the keywords shown in Table 13–10 except for the CANCEL control option.

See Also: Section 6.3.3 and Chapter 13

Step 3 Monitor the recovery process.

You can query views to monitor log apply services as follows:

1. To verify that you have correctly initiated log apply services, query the V$MANAGED_STANDBY fixed view on the standby database. This view monitors the progress of a standby database in managed recovery mode. For example:

```
SQL> SELECT PROCESS, STATUS, THREAD#, SEQUENCE#, BLOCK#, BLOCKS
  2> FROM V$MANAGED_STANDBY;

PROCESS STATUS       THREAD#    SEQUENCE#  BLOCK#     BLOCKS
------- ------------ ---------- ---------- ---------- ----------
MRP0    APPLYING_LOG 1          946        10         1001
```

   If you did not start a detached server process, you need to execute this query from another SQL session.

2. To monitor activity on the standby database, query the V$ARCHIVE_DEST_STATUS fixed view.

See Also: Section 6.4

6.3.3 Controlling Managed Recovery Mode

This section describes controlling managed recovery operations using the following control options:

- CANCEL Control Option
- DEFAULT DELAY Control Option
- DELAY Control Option
- DISCONNECT Control Option
- EXPIRE and NO EXPIRE Control Options
- FINISH Control Option
- NEXT Control Option
- NODELAY Control Option
- PARALLEL and NOPARALLEL Control Options
- THROUGH ALL ARCHIVELOG Control Option

- THROUGH...SWITCHOVER Control Option
- THROUGH...SEQUENCE Control Option
- TIMEOUT and NO TIMEOUT Control Options

6.3.3.1 CANCEL Control Option

Cancel the managed recovery operation at any time by issuing the CANCEL option of the ALTER DATABASE RECOVER MANAGED STANDBY DATABASE statement.

Format

```
CANCEL
CANCEL IMMEDIATE NOWAIT
CANCEL NOWAIT
```

The CANCEL option directs log apply services to stop the managed recovery operation after completely processing the current archived redo log. The managed recovery operation is terminated on an archived log boundary.

The CANCEL IMMEDIATE option, however, directs log apply services to stop the managed recovery operation either before reading another block from the archived redo log or before opening the next archived redo log, whichever occurs first. The managed recovery operation is terminated on an I/O boundary or on an archived log boundary, whichever occurs first. Stopping the recovery on an I/O boundary will leave the database in an inconsistent state and, therefore, unable to be opened. Note the following scenarios:

If you cancel recovery	Then
Before recovery opens the next archived redo log	CANCEL IMMEDIATE is equivalent to CANCEL.
While the standby database is processing an archived redo log	CANCEL IMMEDIATE leaves the database in an inconsistent state. The Oracle database server does not allow a database to be opened in an inconsistent state, although you can still initiate manual or managed recovery.

By default, the CANCEL option waits for the managed recovery operation to terminate before control is returned. If you specify the NOWAIT option, control is returned to the process that issued the CANCEL option without waiting for the managed recovery operation to terminate.

6.3.3.2 DEFAULT DELAY Control Option

The DEFAULT DELAY control option clears any value specified using the DELAY option, and directs the managed recovery operation to use any delay interval that may have been specified with the LOG_ARCHIVE_DEST_*n* (where *n* is a number from 1 to 10) initialization parameter on the primary database.

Format

```
DEFAULT DELAY
```

6.3.3.3 DELAY Control Option

Use the DELAY control option to specify an absolute apply delay interval to the managed recovery operation. The apply delay interval begins once the archived redo logs have been selected for recovery.

The apply delay interval set by the DELAY control option supersedes any apply delay interval specified for the standby database by the primary database's corresponding LOG_ARCHIVE_DEST_*n* initialization parameter. The DELAY control option pertains to archived redo logs being applied by this managed recovery operation. (The DEFAULT DELAY control option returns the managed recovery operation behavior to the value that you set for the DELAY option.)

You can use the DELAY control option when starting a managed recovery operation or to alter the mode of an active managed recovery operation. A DELAY control option value of zero (0) directs the managed recovery operation to revert to default behavior.

Format

```
DELAY minutes
```

In the following example, log apply services specify an absolute apply delay interval of 30 minutes to a foreground managed recovery operation:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DELAY 30;
```

6.3.3.4 DISCONNECT Control Option

Use the DISCONNECT control option to start managed recovery in background mode.

Format

```
DISCONNECT
```

```
DISCONNECT [FROM SESSION]
```

Using this option creates a **managed recovery process (MRP)** to perform the recovery in the background while the foreground process that issued the RECOVER statement continues performing other tasks. The optional FROM SESSION keywords can be added for clarity but do not change the behavior of the DISCONNECT option.

6.3.3.5 EXPIRE and NO EXPIRE Control Options

Use the EXPIRE control option to specify the number of minutes after which the managed recovery operation automatically terminates, relative to the current time. The managed recovery operation terminates at the end of the current archived redo log that is being processed. This means that the value of the EXPIRE control option is the earliest amount of time that the managed recovery operation will terminate, but it could be significantly later than the specified value.

The managed recovery operation expiration is always relative to the time the statement was issued, not when the managed recovery operation was started. To cancel an existing managed recovery operation, use the CANCEL control option.

Specifying a new EXPIRE control option overrides any previously specified value. Use the NO EXPIRE control option to cancel a previously specified expiration value.

When you use a detached managed recovery process, the MRP makes an alert log entry when it terminates. A foreground managed recovery process simply returns control to the SQL prompt.

Format

EXPIRE *minutes*
NO EXPIRE

In the following example, log apply services automatically terminates two hours from now:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE EXPIRE 120;
```

The EXPIRE control option cannot be specified in conjunction with the CANCEL control option.

The NO EXPIRE control option turns off a previously specified EXPIRE control option.

6.3.3.6 FINISH Control Option

Use the FINISH control option to complete managed recovery in preparation for a failover from the primary database to the standby database.

Format

```
FINISH
FINISH NOWAIT
FINISH [SKIP [STANDBY LOGFILE]] [NOWAIT | WAIT]
```

Managed recovery, directed by the FINISH option, first applies all available archived redo logs and then recovers any available standby redo logs (but a recovery without the FINISH control option only applies the archived redo logs). This extra step brings the standby database up-to-date with the last committed transaction on the primary database. You can use the FINISH option when starting managed recovery operations or to alter the mode of an ongoing managed recovery operation. If you use the FINISH option to alter the mode of an ongoing managed recovery operation, use the NOWAIT option to allow control to be returned to the foreground process before the recovery completes. You can specify the following keywords:

- SKIP [STANDBY LOGFILE] to indicate that it is acceptable to skip applying the contents of the standby redo logs (if used). The FINISH SKIP option recovers to the first unarchived SCN.
- NOWAIT to return control to the foreground process before the recovery completes.
- WAIT to return control to the foregrouond process after recovery completes.

Once the RECOVER...FINISH statement has successfully completed, you must issue the COMMIT TO SWITCHOVER TO PRIMARY statement to convert the standby database to the primary database. You can no longer use this database as a standby database.

See Also: Section 7.3.3.1 and Section 13.12

6.3.3.7 NEXT Control Option

Use the NEXT control option to direct the managed recovery operation to apply a specified number of archived redo logs as soon as possible after log transport services have archived them. Any apply delay interval specified by the DELAY attribute of the LOG_ARCHIVE_DEST_*n* initialization parameter set on the primary database, or by the DELAY control option, is ignored by the managed recovery

operation until the specified number of archived redo logs have been applied. Then, the managed recovery operation reverts to the default behavior.

You can use the NEXT control option when starting a managed recovery operation or to alter the mode of an active managed recovery operation.

Format

NEXT *count*

In the following example, log apply services direct a foreground managed recovery operation to apply the next 5 archived redo logs without delay:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE NEXT 5;
```

6.3.3.8 NODELAY Control Option

Use the NODELAY control option to direct the managed recovery operation to apply the archived redo logs as soon as possible after log transport services have archived them. Any apply delay interval specified by the DELAY attribute of the LOG_ARCHIVE_DEST_*n* initialization parameter on the primary database, or by a DELAY control option, is ignored by the managed recovery operation when the NODELAY control option is specified. By default, the managed recovery operation respects any apply delay interval specified for the standby database by the primary database's corresponding LOG_ARCHIVE_DEST_*n* initialization parameter.

Format

NODELAY

In the following example, log apply services direct a foreground managed recovery operation to apply archived redo logs without delay:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE NODELAY;
```

6.3.3.9 PARALLEL and NOPARALLEL Control Options

When running in managed recovery mode, log apply services apply the changes generated on the primary database by many concurrent processes. Therefore, applying the archived redo logs on the standby database can take longer than the time it took to initially generate the changes on the primary database. By default, log apply services use a single process to apply all of the archived redo logs sequentially. This is the same thing as specifying the NOPARALLEL control option,

which disables a previously specified PARALLEL option, so that log apply services use a single process to apply all of the archived redo logs sequentially.

When using the parallel recovery option, several processes are able to apply the archived redo logs simultaneously.

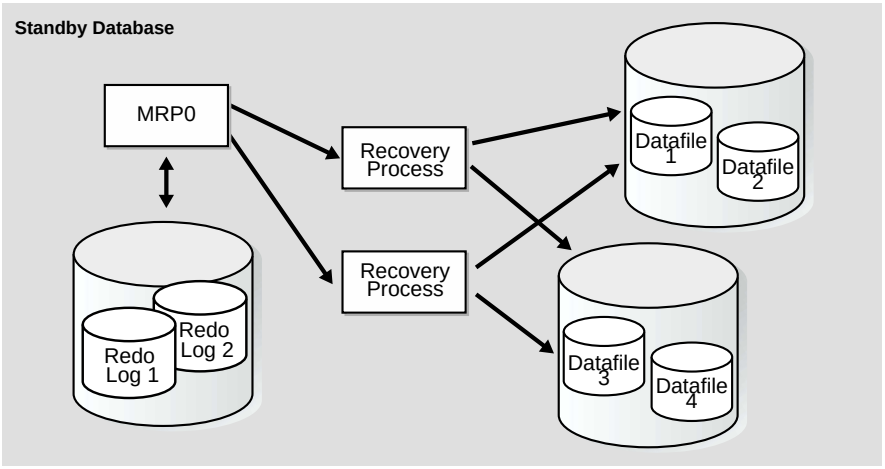
In general, using the parallel recovery option is most effective at reducing recovery time when several datafiles on several different disks are being recovered concurrently. The performance improvement from the parallel recovery option is also dependent upon whether the operating system supports asynchronous I/O. If asynchronous I/O is not supported, the parallel recovery option can dramatically reduce recovery time. If asynchronous I/O is supported, the recovery time may be only slightly reduced by using parallel recovery.

> **See Also:** Your Oracle operating system-specific documentation to determine whether the system supports asynchronous I/O

In a typical parallel recovery situation, one process is responsible for reading and dispatching archived redo logs. This is the dedicated managed recovery server process (either the foreground SQL session or the background MRP0 process) that begins the recovery session. The managed recovery server process reading the archived redo logs enlists two or more recovery processes to apply the changes from the archived redo logs to the datafiles.

Figure 6–2 illustrates a typical parallel recovery session.

Figure 6–2 Parallel Recovery Session



Standby database recovery is a very disk-intensive activity (as opposed to a CPU-intensive activity). Therefore, the number of recovery processes needed is dependent entirely upon how many disk drives are involved in recovery. In most situations, one or two recovery processes per disk drive containing datafiles needing recovery are sufficient. In general, a minimum of eight recovery processes is needed before parallel recovery can show improvement over a serial recovery.

Format

PARALLEL *number*
NOPARALLEL

In the following example, log apply services specify a parallel managed recovery operation utilizing 8 background child processes:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE PARALLEL 8;
```

6.3.3.10 THROUGH ALL ARCHIVELOG Control Option

Use the THROUGH ALL ARCHIVELOG control option to specify the default behavior for managed recovery mode, which is to continue managed recovery operations until the operations are explicitly stopped. This clause is useful in altering a managed recovery operation that is currently running with the THROUGH THREAD

n SEQUENCE *n* option so that it does not stop after applying the specified archived redo log.

Format

```
THROUGH ALL ARCHIVELOG
```

6.3.3.11 THROUGH...SWITCHOVER Control Option

Once started, log apply services continue to apply redo data until canceled or applied through the switchover indicator, which is an end-of-redo marker added to archived redo logs. The switchover indicator is added to archived redo logs in any of the following situations:

- When a switchover operation occurs, because this operation adds an end-of-redo marker to the header of the last log file being archived
- When you perform a graceful failover operation

Use the THROUGH...SWITCHOVER control option to keep the managed recovery process operational after the archived redo log containing the switchover indicator has been applied. By using this control option, other standby databases not participating in the switchover operation can continue to receive and apply archived redo logs from the new primary database, rather than stopping the recovery process and then starting it again after the standby database has failed over to become the new primary database.

Format

```
THROUGH ALL SWITCHOVER
THROUGH LAST SWITCHOVER
THROUGH NEXT SWITCHOVER
```

To use the THROUGH...SWITCHOVER control option, you must specify the ALL, LAST, or NEXT option to control the behavior of log apply services when encountering an end-of-redo marker:

- Use the THROUGH ALL SWITCHOVER control option to recover through all end-of-redo markers and continue applying archived redo logs received from the new primary database after a switchover operation. The ALL option is very beneficial when there are many other standby databases (that are not involved in the switchover operation) in the Data Guard configuration and you do not want to stop and restart the recovery process on each one.

- Use the THROUGH LAST SWITCHOVER control option to continue managed recovery through all end-of-redo markers, stopping managed recovery only when an end-of-redo marker is encountered in the last archived redo log received.

  Note: If the last primary database online redo log is not archived to the original standby database, or if the SQL ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY statement is issued before the archived log has been applied to the original standby database, log apply services will not wait for the *last* archived redo to arrive. Log apply services will finish processing the current archived redo log, and then start receiving and applying logs from the new primary database.

- Use the THROUGH NEXT SWITCHOVER control option to cancel managed recovery at each switchover indicator (end-of-redo marker). This is the default.

In the following example, log apply services specify that other standby databases (not involved in the switchover operation) should continue to receive and apply all archived redo logs after the switchover operation:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE THROUGH ALL SWITCHOVER;
```

6.3.3.12 THROUGH...SEQUENCE Control Option

Use the THROUGH...SEQUENCE control option to specify the thread number and sequence number of the archived redo log through which you want to recover. Once the specified archived redo log has been applied, managed recovery terminates. The THREAD keyword is optional. If you do not specify THREAD *n*, it defaults to thread 1.

Format

```
THROUGH [THREAD integer] SEQUENCE integer
```

6.3.3.13 TIMEOUT and NO TIMEOUT Control Options

Use the TIMEOUT control option to specify the number of minutes that foreground log apply services wait for log transport services to complete the archiving of the redo log required by the managed recovery operation. If the specified number of minutes passes without receiving the required archived redo log, the managed recovery operation is automatically terminated. By default, log apply services wait indefinitely for the next required archived redo log. The managed recovery

operation is terminated only through use of the CANCEL option, a CTRL+C key combination, or an instance shutdown.

Format

```
TIMEOUT minutes
NO TIMEOUT
```

In the following example, log apply services initiate a foreground managed recovery operation with a timeout interval of 15 minutes:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE TIMEOUT 15;
```

The TIMEOUT control option cannot be used in conjunction with the DISCONNECT control option.

The NO TIMEOUT control option turns off a previously specified TIMEOUT control option.

6.3.4 Datafile Management

If a physical standby system uses the same directory naming structure as the primary system, you do not have to rename the primary database files in the physical standby database control file. If the primary and standby databases are located on the same site, however, or if the primary and standby sites use different directory naming structures, then you must rename the database files in the standby control file so that the archived redo logs can be applied.

You can set initialization parameters so that your standby database automatically converts datafile and archived redo log filenames based on data in the standby database control file. If you cannot rename all primary database files automatically using these parameters, then you must rename them manually.

Note: If the standby and primary databases are located on the same system, you must define the LOCK_NAME_SPACE parameter. If you do not set the LOCK_NAME_SPACE parameter differently when the standby and primary databases are located on the same system, you will receive an ORA-1102 error.

The initialization parameters in Table 6–3 perform automatic filename conversions.

Table 6–3 Filename Conversion

Parameter	Function
DB_FILE_NAME_CONVERT	Converts primary database datafile filenames to standby datafile filenames, for example, from tbs_* to standbytbs_*.
LOG_FILE_NAME_CONVERT	Converts primary database redo log filenames to standby database redo log filenames, for example, from log_* to standbylog_*.
STANDBY_FILE_MANAGEMENT	When set to auto, this parameter automates the creation and deletion of datafile filenames on the standby site using the same filenames as the primary site.

Use the DB_FILE_NAME_CONVERT parameter to convert the filenames of one or more sets of datafiles on the primary database to filenames on the standby database; use the LOG_FILE_NAME_CONVERT parameter to convert the filename of a new redo log on the primary database to a filename on the standby database.

When the standby database is updated, the DB_FILE_NAME_CONVERT parameter is used to convert the datafile name on the primary database to a datafile name on the standby database. The file must exist and be writable on the standby database or the recovery process will halt with an error.

The DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT parameters must have two strings. The first string is a sequence of characters to be looked for in a primary database filename. If that sequence of characters is matched, it is replaced by the second string to construct the standby database filename.

Adding a datafile or log to the primary database necessitates adding a corresponding file to the standby database. Use the STANDBY_FILE_MANAGEMENT initialization parameter with the DB_FILE_NAME_CONVERT initialization parameter to automate the process of creating files with identical filenames on the standby database and primary database.

The DB_FILE_NAME_CONVERT initialization parameter allows multiple pairs of filenames to be specified. For example:

```
DB_FILE_NAME_CONVERT="/private1/prmy1/df1", "/private1/stby1/df1", \
                     "/private1/prmy1", "/private1/stby1"
STANDBY_FILE_MANAGEMENT=auto
```

Note: When you specify pairs of files, be sure to specify supersets of path names before subsets.

This section contains the following topics:

- Setting the STANDBY_FILE_MANAGEMENT Initialization Parameter
- Restrictions on ALTER DATABASE Operations

6.3.4.1 Setting the STANDBY_FILE_MANAGEMENT Initialization Parameter

When you set the STANDBY_FILE_MANAGEMENT initialization parameter to auto, it automatically creates on the standby database any datafiles that have been newly created on the primary database, using the same name that you specified on the primary database.

The STANDBY_FILE_MANAGEMENT initialization parameter works with the DB_FILE_NAME_CONVERT parameter to take care of datafiles that are spread across multiple directory paths on the primary database.

6.3.4.2 Restrictions on ALTER DATABASE Operations

You cannot rename the datafile on the standby site when the STANDBY_FILE_MANAGEMENT initialization parameter is set to auto. When you set the STANDBY_FILE_MANAGEMENT initialization parameter to auto, the following operations are no longer necessary:

- ALTER DATABASE RENAME
- ALTER DATABASE ADD/DROP LOGFILE
- ALTER DATABASE ADD/DROP LOGFILE MEMBER
- ALTER DATABASE CREATE DATAFILE AS

If you attempt to use any of these statements on the standby database, an error is returned. For example:

```
SQL> ALTER DATABASE RENAME FILE '/private1/stby1/t_db2.dbf' to 'dummy';
alter database rename file '/private1/stby1/t_db2.dbf' to 'dummy'
*
ERROR at line 1:
ORA-01511: error in renaming log/data files
ORA-01270: RENAME operation is not allowed if STANDBY_FILE_MANAGEMENT is auto
```

See Also: Section 8.3.1 to learn how to add datafiles to a database

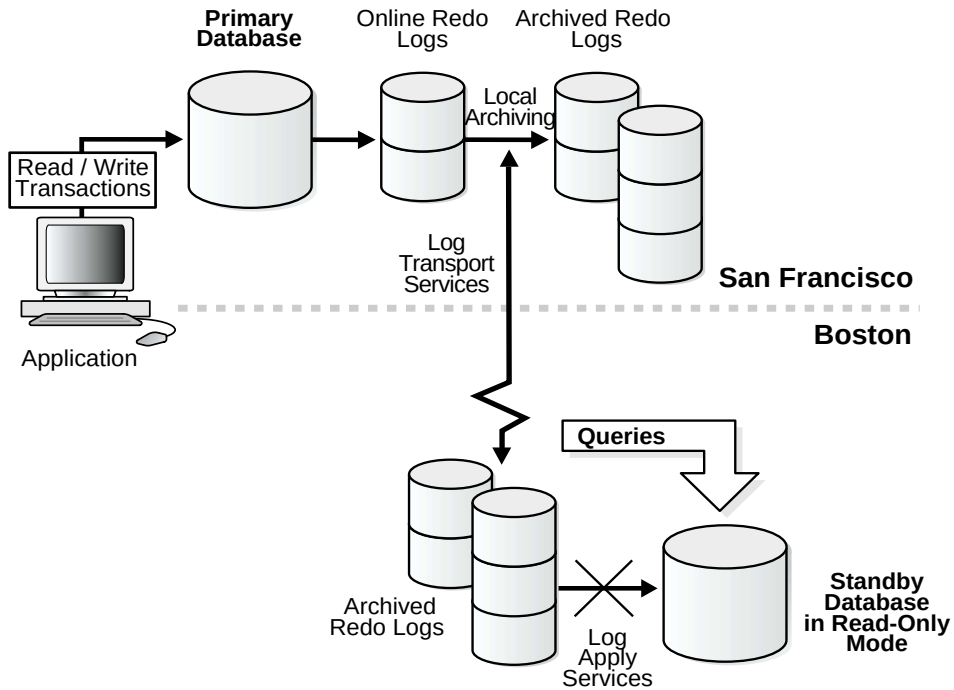
6.3.5 Read-Only Mode

Read-only mode allows users to open and query a standby database without the potential for online data modifications. This mode reduces system overhead on the primary database by using the standby database for reporting purposes. You can periodically open the standby database in read-only mode to:

- Run reports
- Perform ad hoc queries to ensure that log apply services are updating the standby database properly

Figure 6–3 shows a standby database in read-only mode.

Figure 6–3 Standby Database in Read-Only Mode



This section contains the following topics:

6.3.5.1 Assessing Whether to Run in Read-Only Mode

As you decide whether to run the standby database in read-only mode, consider the following:

- Having the standby database open in read-only mode available for queries makes it unavailable for managed recovery. The archived redo logs are received by the standby database site but are not applied. At some point, you need to put the standby database back in managed recovery mode and apply the archived redo logs to resynchronize the standby database with the primary database. Therefore, running the standby database in read-only mode may prolong a failover operation if one is required for disaster recovery.
- If you need the standby database both for protection against disaster and reporting, then you can maintain multiple standby databases, some in read-only mode and some in managed recovery mode. However, you will need to synchronize the **read-only database** periodically. A database in managed recovery mode gives you immediate protection against disaster.

6.3.5.2 Placing the Database in Read-Only Mode

You can change from the managed recovery mode to read-only mode (and back again) using the following procedures. The following modes are possible for a standby database:

- Started
- Mounted
- Managed recovery mode
- Read-only mode

To open a standby database in read-only mode when the database is shut down:

1. Start the Oracle instance for the standby database without mounting it:

   ```
   SQL> STARTUP NOMOUNT PFILE=initSTANDBY.ora;
   ```

2. Mount the standby database:

   ```
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

3. Open the database in read-only mode:

```
SQL> ALTER DATABASE OPEN READ ONLY;
```

To open the standby database in read-only mode when in managed recovery mode:

1. Cancel log apply services:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
```

2. Open the database in read-only mode:

```
SQL> ALTER DATABASE OPEN READ ONLY;
```

To change the standby database from read-only mode back to managed recovery mode:

1. Terminate all active user sessions on the standby database.
2. Restart log apply services:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE;
```

Note: The log apply services component resumes from the time when it was last canceled.

6.3.6 Read-Only Mode Considerations

Before you put your standby database in read-only mode, consider the following topics:

- Receiving Archived Redo Logs While in Read-Only Mode
- Sorting While in Read-Only Mode
- Sorting Without Temporary Tablespaces

6.3.6.1 Receiving Archived Redo Logs While in Read-Only Mode

While the standby database is in read-only mode, the standby site can still receive archived redo logs from the primary site. However, these archived redo logs are not automatically applied to the standby database until the database is in managed recovery mode. Consequently, a read-only standby database is not synchronized

with the primary database at the archive level. You should not fail over to the standby database unless all archived redo logs have been applied.

> **See Also:** Section 5.5.2 for examples of initialization parameter settings you need to define to automatically archive from the primary site to the standby site

6.3.6.2 Sorting While in Read-Only Mode

To perform queries on a read-only standby database, the Oracle database server must be able to perform on-disk sorting operations. You cannot allocate space for sorting operations in **tablespaces** that cause Oracle to write to the data dictionary.

Temporary tablespaces allow you to add tempfile entries in read-only mode for the purpose of making queries. You can then perform on-disk sorting operations in a read-only database without affecting dictionary files or generating redo entries.

Note the following requirements for creating temporary tablespaces:

- The tablespaces must be temporary, locally managed, and contain only tempfiles.
- User-level allocations and permissions to use the locally managed temporary tablespaces must be in place on the primary database. You cannot change these settings on the standby database.

You should also follow these guidelines:

- Minimize data validation time on the standby database so that you can change to managed or **manual recovery mode** when necessary.
- Minimize runtimes for reports.
- Implement desired optimizations on the primary database only.

> **See Also:** *Oracle9i Database Administrator's Guide* for more information about using tempfiles and temporary tablespaces

To create a temporary tablespace for use on a read-only standby database:

On the primary database, perform the following steps:

1. Enter the following SQL statement:

```
SQL> CREATE TEMPORARY TABLESPACE temp1
     TEMPFILE '/oracle/dbs/temp1.dbf'
     SIZE 20M REUSE
```

```
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 16M;
```

2. Switch the log to send the redo data to the standby database.

On the physical standby database, perform the following steps:

1. Start managed recovery, if necessary, and apply the archived redo logs by entering the following SQL statement:

   ```
   SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE;
   ```

2. Cancel managed recovery and open the physical standby database in read-only mode using the following SQL statements:

   ```
   SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
   SQL> ALTER DATABASE OPEN READ ONLY;
   ```

   Opening the physical standby database in read-only mode allows you to add a tempfile. Because adding a tempfile does not generate redo data, it is allowed for a read-only database.

3. Create a temporary tablespace. For example:

   ```
   SQL> ALTER TABLESPACE temp1
          ADD TEMPFILE '/oracle/dbs/s_temp1.dbf'
          SIZE 10M REUSE;
   ```

The redo data that is generated on the primary database automatically creates the temporary tablespace in the standby control file after the archived redo log is applied to the physical standby database. However, you must use the ADD TEMPFILE clause to actually create the disk file on the standby database.

> **See Also:** *Oracle9i SQL Reference* for information about the CREATE TEMPORARY TABLESPACE syntax

6.3.6.3 Sorting Without Temporary Tablespaces

If you attempt to sort without temporary tablespaces by executing a SQL SELECT * FROM V$PARAMETER statement when the database is not open, you will get an error. For example:

```
SQL> SELECT * FROM V$PARAMETER;

select * from v$parameter

              *
```

```
ERROR at line 1:

ORA-01220: file based sort illegal before database is open
```

To look at the parameters when the database is not open:

- Set the SORT_AREA_SIZE parameter to a suitable value in your initialization parameter file. This enables you to execute the SELECT * FROM V$PARAMETER statement when the database is open. SORT_AREA_SIZE is a static parameter. Specify the parameter in your initialization parameter file or SPFILE prior to starting the instance.
- If you do not set the SORT_AREA_SIZE parameter to a sufficient value, you cannot select all of the columns from the V$PARAMETER view. If you need to select only a few columns, you can execute the SELECT statement in any database state.

6.4 Monitoring Log Apply Services

To determine the status of archived redo logs on the standby database, query the V$MANAGED_STANDBY, V$ARCHIVE_DEST_STATUS, V$ARCHIVED_LOG, V$LOG_HISTORY fixed views, and the DBA_LOGSTDBY_LOG and DBA_LOGSTDBY_PROGRESS views. You can also monitor the standby database using Oracle9*i* Data Guard Manager.

See Also: Appendix A, "Troubleshooting the Standby Database"

This section contains the following topics:

- Accessing the V$MANAGED_STANDBY Fixed View (Physical Standby Databases Only)
- Accessing the V$ARCHIVE_DEST_STATUS Fixed View
- Accessing the V$ARCHIVED_LOG Fixed View
- Accessing the V$LOG_HISTORY Fixed View
- Accessing the V$DATAGUARD_STATUS Fixed View
- Accessing the DBA_LOGSTDBY_LOG View (Logical Standby Databases Only)
- Accessing the DBA_LOGSTDBY_PROGRESS View (Logical Standby Databases Only)
- Setting Archive Tracing

6.4.1 Accessing the V$MANAGED_STANDBY Fixed View (Physical Standby Databases Only)

Query the physical standby database to monitor log apply and log transport services activity at the standby site.

```
SQL> SELECT PROCESS, STATUS, THREAD#, SEQUENCE#, BLOCK#, BLOCKS
  2> FROM V$MANAGED_STANDBY;

PROCESS STATUS       THREAD#    SEQUENCE#  BLOCK#     BLOCKS
------- ------------ ---------- ---------- ---------- ----------
RFS     ATTACHED     1          947        72         72
MRP0    APPLYING_LOG 1          946        10         72
```

The previous query output shows an RFS process that has completed the archiving of redo log file sequence number 947. The output also shows a managed recovery operation that is actively applying archived redo log sequence number 946. The recovery operation is currently recovering block number 10 of the 72-block archived redo log file.

See Also: V$MANAGED_STANDBY (Physical Standby Databases Only) in Chapter 14

6.4.2 Accessing the V$ARCHIVE_DEST_STATUS Fixed View

To quickly determine the level of synchronization for the standby database, issue the following query:

```
SQL> SELECT ARCHIVED_THREAD#, ARCHIVED_SEQ#, APPLIED_THREAD#, APPLIED_SEQ#
  2> FROM V$ARCHIVE_DEST_STATUS;

ARCHIVED_THREAD# ARCHIVED_SEQ# APPLIED_THREAD# APPLIED_SEQ#
---------------- ------------- --------------- ------------
1                947           1               945
```

The previous query output shows the standby database is two archived log files behind in applying the redo logs received from the primary database. This may indicate that a single recovery process is unable to keep up with the volume of archived redo logs being received. Using the PARALLEL option may be a solution.

See Also: V$ARCHIVE_DEST_STATUS in Chapter 14

6.4.3 Accessing the V$ARCHIVED_LOG Fixed View

The V$ARCHIVED_LOG fixed view on the standby database shows all the archived redo logs received from the primary database. This view is only useful after the standby site has started receiving logs, because before that time the view is populated by old archived log records generated from the primary control file. For example, you can execute the following SQL*Plus script (sample output included):

```
SQL> SELECT REGISTRAR, CREATOR, THREAD#, SEQUENCE#, FIRST_CHANGE#,
  2> NEXT_CHANGE# FROM V$ARCHIVED_LOG;

REGISTRAR CREATOR THREAD#    SEQUENCE#  FIRST_CHANGE# NEXT_CHANGE#
--------- ------- ---------- ---------- ------------- ------------
RFS       ARCH    1          945        74651         74739
RFS       ARCH    1          946        74739         74772
RFS       ARCH    1          947        74772         74774
```

The previous query output shows three archived redo logs received from the primary database.

See Also: V$ARCHIVED_LOG in Chapter 14

6.4.4 Accessing the V$LOG_HISTORY Fixed View

The V$LOG_HISTORY fixed view on the standby database shows all the archived redo logs that have been recovered. For example:

```
SQL> SELECT THREAD#, SEQUENCE#, FIRST_CHANGE#, NEXT_CHANGE#
  2> FROM V$LOG_HISTORY;

THREAD#    SEQUENCE#  FIRST_CHANGE# NEXT_CHANGE#
---------- ---------- ------------- ------------
1          945        74651         74739
```

The previous query output shows that the most recently recovered archived redo log was sequence number 945.

See Also: V$LOG_HISTORY in Chapter 14

6.4.5 Accessing the V$DATAGUARD_STATUS Fixed View

The V$DATAGUARD_STATUS fixed view displays and logs events that would typically be triggered by any message to the alert log or server process trace files.

The following example shows V$DATAGUARD_STATUS output from a primary database:

```
SQL> SELECT MESSAGE FROM V$DATAGUARD_STATUS;

MESSAGE
--------------------------------------------------------------------------------

ARC0: Archival started
ARC1: Archival started
Archivelog destination LOG_ARCHIVE_DEST_2 validated for no-data-loss
recovery
Creating archive destination LOG_ARCHIVE_DEST_2: 'dest2'
ARCH: Transmitting activation ID 0
LGWR: Completed archiving log 3 thread 1 sequence 11
Creating archive destination LOG_ARCHIVE_DEST_2: 'dest2'
LGWR: Transmitting activation ID 6877c1fe
LGWR: Beginning to archive log 4 thread 1 sequence 12
ARC0: Evaluating archive   log 3 thread 1 sequence 11
ARC0: Archive destination LOG_ARCHIVE_DEST_2: Previously completed
ARC0: Beginning to archive log 3 thread 1 sequence 11
Creating archive destination LOG_ARCHIVE_DEST_1:
'/oracle/arch/arch_1_11.arc'

ARC0: Completed archiving  log 3 thread 1 sequence 11
ARC1: Transmitting activation ID 6877c1fe

15 rows selected.
```

The following example shows the contents of the V$DATAGUARD_STATUS view on a physical standby database:

```
SQL> SELECT MESSAGE FROM V$DATAGUARD_STATUS;

MESSAGE
--------------------------------------------------------------------------------

ARC0: Archival started
ARC1: Archival started
RFS: Successfully opened standby logfile 6: '/oracle/dbs/sorl2.log'

ARC1: Evaluating archive   log 6 thread 1 sequence 11
ARC1: Beginning to archive log 6 thread 1 sequence 11
Creating archive destination LOG_ARCHIVE_DEST_1:
```

```
'/oracle/arch/arch_1_11.arc'

ARC1: Completed archiving  log 6 thread 1 sequence 11
RFS: Successfully opened standby logfile 5: '/oracle/dbs/sorl1.log'

Attempt to start background Managed Standby Recovery process
Media Recovery Log /oracle/arch/arch_1_9.arc

10 rows selected.
```

See Also: V$DATAGUARD_STATUS in Chapter 14

6.4.6 Accessing the DBA_LOGSTDBY_LOG View (Logical Standby Databases Only)

The DBA_LOGSTDBY_LOG view provides dynamic information about what is happening to log apply services. This view is very helpful when you are diagnosing performance problems during the SQL application of archived redo logs to the logical standby database, and it can be helpful for other problems.

For example:

```
SQL> SELECT SUBSTR(FILE_NAME,1,25) FILE_NAME, SUBSTR(SEQUENCE#,1,4)"SEQ#",
  2> FIRST_CHANGE#, NEXT_CHANGE#, TO_CHAR(TIMESTAMP, 'HH:MM:SS')  TIMESTAMP,
  3> DICT_BEGIN BEG, DICT_END END, SUBSTR(THREAD#,1,4) "THR#"
  4> FROM DBA_LOGSTDBY_LOG ORDER BY SEQUENCE#;

SQL> SELECT FILE_NAME, SEQUENCE#, FIRST_CHANGE#, NEXT_CHANGE#,
  2> TIMESTAMP, DICT_BEGIN, DICT_END, THREAD# FROM DBA_LOGSTDBY_LOG
  3> ORDER BY SEQUENCE#;

FILE_NAME                 SEQ# FIRST_CHANGE# NEXT_CHANGE# TIMESTAM BEG END THR#
------------------------- ---- ------------- ------------ -------- --- --- ----
/oracle/dbs/hq_nyc_2.log  2           101579       101588 11:02:58 NO  NO  1
/oracle/dbs/hq_nyc_3.log  3           101588       142065 11:02:02 NO  NO  1
/oracle/dbs/hq_nyc_4.log  4           142065       142307 11:02:10 NO  NO  1
/oracle/dbs/hq_nyc_5.log  5           142307       142739 11:02:48 YES YES 1
/oracle/dbs/hq_nyc_6.log  6           142739       143973 12:02:10 NO  NO  1
/oracle/dbs/hq_nyc_7.log  7           143973       144042 01:02:11 NO  NO  1
/oracle/dbs/hq_nyc_8.log  8           144042       144051 01:02:01 NO  NO  1
/oracle/dbs/hq_nyc_9.log  9           144051       144054 01:02:16 NO  NO  1
/oracle/dbs/hq_nyc_10.log 10          144054       144057 01:02:21 NO  NO  1
/oracle/dbs/hq_nyc_11.log 11          144057       144060 01:02:26 NO  NO  1
/oracle/dbs/hq_nyc_12.log 12          144060       144089 01:02:30 NO  NO  1
/oracle/dbs/hq_nyc_13.log 13          144089       144147 01:02:41 NO  NO  1
```

The output from this query shows that a LogMiner dictionary build starts at log file sequence 5. The most recent archive log file is sequence 13 and it was received at the logical standby database at 01:02:41.

> **See Also:** DBA_LOGSTDBY_LOG (Logical Standby Databases Only) in Chapter 14

6.4.7 Accessing the DBA_LOGSTDBY_PROGRESS View (Logical Standby Databases Only)

The DBA_LOGSTDBY_PROGRESS view describes the progress of SQL apply operations on the logical standby databases.

For example:

```
SQL> SELECT APPLIED_SCN, APPLIED_TIME, READ_SCN, READ_TIME, NEWEST_SCN, NEWEST_TIME FROM
  2> DBA_LOGSTDBY_PROGRESS;

APPLIED_SCN       APPLIED_TIME   READ_SCN          READ_TIME       NEWEST_SCN NEWEST_TIME
----------- ----------------- ---------- ----------------- ----------------------------
     165285 22-FEB-02 17:00:59     165285 22-FEB-02 17:12:32
```

The previous query output shows that transactions up to and including SCN 165285 have been applied.

> **See Also:** DBA_LOGSTDBY_PROGRESS (Logical Standby Databases Only) in Chapter 14

6.4.8 Setting Archive Tracing

To see the progression of the archiving of redo logs to the standby site, set the LOG_ARCHIVE_TRACE parameter in the primary and standby initialization parameter files.

LOG_ARCHIVE_TRACE on the	Causes Oracle to write	In trace file
Primary database	Audit trail of archiving process activity (ARC*n* and foreground processes) on the primary database	Whose filename is specified in the USER_DUMP_DEST initialization parameter

LOG_ARCHIVE_TRACE on the	Causes Oracle to write	In trace file
Standby database	Audit trail of the RFS and the ARC*n* process activity relating to archived redo logs on the standby database	Whose filename is specified in the USER_DUMP_DEST initialization parameter

6.4.8.1 Determining the Location of the Trace Files

The trace files for a database are located in the directory specified by the USER_DUMP_DEST parameter in the initialization parameter file. Connect to the primary and standby instances using SQL*Plus and issue a SHOW statement to determine the location, for example:

```
SQL> SHOW PARAMETER user_dump_dest
NAME                                 TYPE    VALUE
------------------------------------ ------- ------------------------------
user_dump_dest                       string  ?/rdbms/log
```

6.4.8.2 Setting the Log Trace Parameter

The format for the archiving trace parameter is as follows, where *trace_level* is an integer:

```
LOG_ARCHIVE_TRACE=trace_level
```

To enable, disable, or modify the LOG_ARCHIVE_TRACE parameter in a primary database, do one of the following:

- Shut down the primary database, modify the initialization parameter file, and restart the database.
- Issue an ALTER SYSTEM SET LOG_ARCHIVE_TRACE=*trace_level* statement while the database is open or mounted.

To enable, disable, or modify the LOG_ARCHIVE_TRACE parameter in a standby database in read-only or recovery mode, issue a SQL statement similar to the following:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_TRACE=15;
```

In the previous example, specifying 15 sets trace levels 1, 2, 4, and 8 as described in Section 6.4.8.3.

Issue the ALTER SYSTEM statement from a different standby session so that it affects trace output generated by the remote file service (RFS) and ARC*n* processes

when the next archived log is received from the primary database. For example, enter:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_TRACE=32;
```

6.4.8.3 Choosing an Integer Value

The integer values for the LOG_ARCHIVE_TRACE parameter represent levels of tracing data. In general, the higher the level, the more detailed the information. The following integer levels are available:

Level	Meaning
0	Disables archived redo log tracing - default setting.
1	Tracks archiving of redo log file.
2	Tracks archival status per archived redo log destination.
4	Tracks archival operational phase.
8	Tracks archived redo log destination activity.
16	Tracks detailed archived redo log destination activity.
32	Tracks archived redo log destination parameter modifications.
64	Tracks ARC*n* process state activity.
128	Tracks FAL server process activity.
256	Supported in a future release.
512	Tracks asynchronous LGWR activity.
1024	Tracks the RFS physical client.
2048	Tracks the ARC*n* or RFS heartbeat.

You can combine tracing levels by setting the value of the LOG_ARCHIVE_TRACE parameter to the sum of the individual levels. For example, setting the parameter to 6 generates level 2 and level 4 trace output.

The following are examples of the ARC0 trace data generated on the primary site by the archiving of redo log 387 to two different destinations: the service standby1 and the local directory /oracle/dbs.

Note: The level numbers do not appear in the actual trace output: they are shown here for clarification only.

```
Level   Corresponding entry content (sample)
-----   -------------------------------
( 1)    ARC0: Begin archiving log# 1 seq# 387 thrd# 1
( 4)    ARC0: VALIDATE
( 4)    ARC0: PREPARE
( 4)    ARC0: INITIALIZE
( 4)    ARC0: SPOOL
( 8)    ARC0: Creating archive destination 2 : 'standby1'
(16)    ARC0:  Issuing standby Create archive destination at 'standby1'
( 8)    ARC0: Creating archive destination 1 : '/oracle/dbs/d1arc1_387.dbf'
(16)    ARC0:  Archiving block 1 count 1 to : 'standby1'
(16)    ARC0:  Issuing standby Archive of block 1 count 1 to 'standby1'
(16)    ARC0:  Archiving block 1 count 1 to :  '/oracle/dbs/d1arc1_387.dbf'
( 8)    ARC0: Closing archive destination 2  : standby1
(16)    ARC0:  Issuing standby Close archive destination at 'standby1'
( 8)    ARC0: Closing archive destination 1  :  /oracle/dbs/d1arc1_387.dbf
( 4)    ARC0: FINISH
( 2)    ARC0: Archival success destination 2 : 'standby1'
( 2)    ARC0: Archival success destination 1 : '/oracle/dbs/d1arc1_387.dbf'
( 4)    ARC0: COMPLETE, all destinations archived
(16)    ARC0: ArchivedLog entry added: /oracle/dbs/d1arc1_387.dbf
(16)    ARC0: ArchivedLog entry added: standby1
( 4)    ARC0: ARCHIVED
( 1)    ARC0: Completed archiving log# 1 seq# 387 thrd# 1

(32)  Propagating archive 0 destination version 0 to version 2
         Propagating archive 0 state version 0 to version 2
         Propagating archive 1 destination version 0 to version 2
         Propagating archive 1 state version 0 to version 2
         Propagating archive 2 destination version 0 to version 1
         Propagating archive 2 state version 0 to version 1
         Propagating archive 3 destination version 0 to version 1
         Propagating archive 3 state version 0 to version 1
         Propagating archive 4 destination version 0 to version 1
         Propagating archive 4 state version 0 to version 1

(64) ARCH: changing ARC0 KCRRNOARCH->KCRRSCHED
        ARCH: STARTING ARCH PROCESSES
        ARCH: changing ARC0 KCRRSCHED->KCRRSTART
        ARCH: invoking ARC0
```

```
ARC0: changing ARC0 KCRRSTART->KCRRACTIVE
ARCH: Initializing ARC0
ARCH: ARC0 invoked
ARCH: STARTING ARCH PROCESSES COMPLETE
ARC0 started with pid=8
ARC0: Archival started
```

The following is the trace data generated by the RFS process on the standby site as it receives archived log 387 in directory `/stby` and applies it to the standby database:

```
level    trace output (sample)
----    ------------------
( 4)      RFS: Startup received from ARCH pid 9272
( 4)      RFS: Notifier
( 4)      RFS: Attaching to standby instance
( 1)      RFS: Begin archive log# 2 seq# 387 thrd# 1
(32)      Propagating archive 5 destination version 0 to version 2
(32)      Propagating archive 5 state version 0 to version 1
( 8)      RFS: Creating archive destination file: /stby/parc1_387.dbf
(16)      RFS:  Archiving block 1 count 11
( 1)      RFS: Completed archive log# 2 seq# 387 thrd# 1
( 8)      RFS: Closing archive destination file: /stby/parc1_387.dbf
(16)      RFS: ArchivedLog entry added: /stby/parc1_387.dbf
( 1)      RFS: Archivelog seq# 387 thrd# 1 available 04/02/99 09:40:53
( 4)      RFS: Detaching from standby instance
( 4)      RFS: Shutdown received from ARCH pid 9272
```

6.5 Managing Archive Gaps

An **archive gap** is a range of archived redo logs created whenever you are unable to receive the next archived redo log generated by the primary database at the standby database. For example, an archive gap occurs when the network goes down and automatic archiving from the primary database to the standby database stops. When the network is up and running again, automatic archiving of the archived redo logs from the primary database to the standby database resumes. However, if the standby database is specified as an optional archive destination, and one or more log switches occurred at the primary site, the standby database has an archive gap. The archived redo logs that were not transmitted represent the gap. The gap is automatically detected and resolved when the missing archived redo logs are transmitted to the standby database to resolve the gap.

Setting Initialization Parameters to Automatically Resolve Archive Gaps

In Oracle9*i*, you can set initialization parameters so that log apply services automatically identify and resolve archive gaps as they occur.

> **Note:** To be able to place the physical standby database in managed recovery mode prior to Oracle9*i*, you would first manually apply logs in the archive gap to the standby database. After you had performed this manual recovery, you could then issue the `ALTER DATABASE RECOVER MANAGED STANDBY DATABASE` statement, at which point log apply services would apply subsequent logs to the standby database automatically.

For log apply services to automatically identify and resolve archive gaps, you must:

1. Use Oracle Net Manager to configure the listener on the standby site. Use the TCP/IP protocol and statically register the standby database service with the listener using the service name so that the standby database can be managed by Data Guard.
2. Use Oracle Net Manager to create a net service name that the standby database can use to connect to the FAL server. The net service name should resolve to a connect descriptor that uses the same protocol, host address, port, and service name that you specified when you configured the listener on the FAL server system, which is typically the same as the primary system. If you are unsure what values to use for these parameters, use Oracle Net Manager to display the listener configuration on the FAL server system.
3. Use Oracle Net Manager to create a net service that the FAL server can use to connect to the standby database. The net service name should resolve to a connect descriptor that uses the same protocol, host address, port, and SID that you specified when you configured the listener on the standby database site. If you are unsure what values to use for these parameters, use Oracle Net Manager to display the listener configuration on the standby database site.
4. In the initialization parameter file of the standby database, assign the net service name that you created for the standby database to the `FAL_CLIENT` initialization parameter, and assign the net service name that you created for the FAL server to the `FAL_SERVER` initialization parameter.

Log apply services automatically detect, and the FAL server process running on the primary database resolves, any gaps that may exist when you enable managed

recovery with the ALTER DATABASE RECOVER MANAGED STANDBY DATABASE statement.

If the FAL process cannot resolve an archive gap, then you must resolve it manually. To manually identify archive gaps:

- On a physical standby database, use the V$ARCHIVE_GAP fixed view to identify an archive gap.
- On a logical standby database, use the DBA_LOGSTDBY_LOG and the DBA_LOGSTDBY_PROGRESS views to identify an archive gap.

> **See Also:** Section B.3 for a description of the manual steps and *Oracle9i Net Services Administrator's Guide* for information about Oracle Net

Define the FAL_CLIENT and FAL_SERVER initialization parameters only for physical standby databases in the initialization parameter file:

Parameter	Function	Syntax
FAL_CLIENT	This parameter specifies the net service name that the FAL server should use to connect to the standby database.	**Syntax**: FAL_CLIENT=*net_service_name* **Example**: FAL_CLIENT='standby1_db'
FAL_SERVER	This parameter specifies the net service name that the standby database should use to connect back to the FAL server.	**Syntax**: FAL_SERVER=*net_service_name* **Example**: FAL_SERVER='my_primary_db'

The FAL server is a background Oracle process that services the incoming requests from the FAL client. In most cases, the FAL server is located on a primary database. However, it can be located on another standby database.

Note: The FAL server automatically detects and resolves archive gaps for logical standby databases without the need to set the FAL_SERVER and FAL_CLIENT initialization parameter. A logical standby database can only request a primary database to resolve its gap. A logical standby database cannot request another standby database (physical or logical) to resolve an archive gap.

7

Role Management Services

This chapter describes how to manage the Data Guard environment. Oracle9*i* Data Guard provides the means to easily manage, manipulate, and change the standby database in many ways.

This chapter contains the following topics:

7.1 Database Roles and Role Transitions

A database operates in one of the following mutually exclusive roles: primary or standby. Data Guard allows you to change these roles dynamically as a planned transition called a **switchover** operation, or in response to a primary database failure, through either a **graceful failover** or a **forced failover** operation.

> **Note:** Switchover and failover operations are not invoked automatically. You must initiate switchover or failover operations using a SQL statement or a Data Guard broker interface.

The main difference between a switchover operation and a failover operation is that the switchover operation does not result in data loss. A failover operation may result in data loss, and requires that you shut down and restart the new primary instance and some or all of the standby database instances.

You must consider the different physical aspects of each role. For example, a database in the primary role uses a current control file, while a database in the

physical standby role uses a standby control file, and a database in the logical standby role uses a dictionary. Using different control files or a dictionary, depending on the current role of the database, requires careful coordination of the initialization parameter files.

Switchover Operations

A **switchover** operation transitions the primary database to the standby role and transitions the standby database to the primary role. You initiate a switchover operation on the primary database. Note the following about switchover operations:

- No data is lost.
- If the switchover operation involves a physical standby database, both the primary database and the physical standby database that is switching over to the primary role must be shut down and restarted. However, there is no need to shut down and restart any other standby databases that are not participants of the switchover operation.
- If the switchover operation involves a logical standby database, there is no need to shut down and restart either the primary database or any of the standby databases. Logical standby databases do not need to be shut down and restarted.

Thus, a switchover operation allows the primary database to transition into its role as a standby database. As a result, scheduled maintenance can be performed more frequently with less system downtime.

For many configurations, you can lessen the frequency of failures by using the switchover operation to perform a regular program of preventive maintenance tasks. You can schedule time for hardware and software maintenance such as the following, without interrupting processing:

- Hardware--repairs, cleaning, adjustments, upgrades, installations. Hardware failures repaired or prevented through this approach can extend the apparent **reliability** of the hardware.
- Software--upgrades, maintenance releases, patch installations, disk defragmentation, cleaning up error logs, verifying status and operation of fault management tools, and collecting data to analyze your needs for additional maintenance can be done during this preventive maintenance time.

See Also: Section 7.2 for information about switchover operations

Failover Operations

A **failover operation** changes one of the standby databases into the role of primary database, and discards the original primary database. You initiate the failover operation on the standby database that you want to fail over to the primary role. You should perform a standby database failover only when a software or system failure results in the loss of the primary database. For a failover operation:

- Data loss is possible unless the standby database is running in maximum protection or maximum availability mode.
- The original primary database is presumed to be lost. You must re-create the original primary database using a backup copy of the new primary database.
- Standby databases that are online at the time of the failover operation, but are not involved in the role transition, do not need to be shut down and restarted.

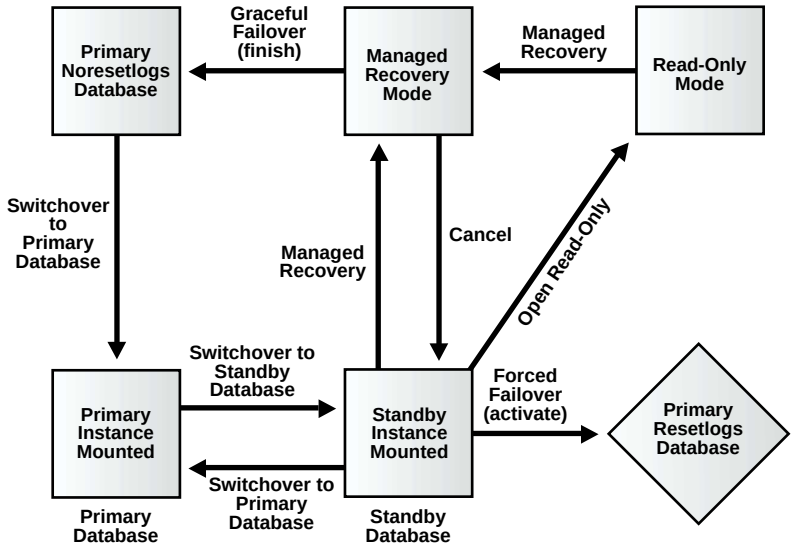
Note: It is not always necessary to perform a failover operation. In some cases, recovery of the primary database may be faster. See Section 7.3.2.

In a failure of the primary database, such as a system or software failure, you may need to perform a failover operation to change one of the standby databases to the primary role. This unplanned transition may result in the loss of application data. The potential loss of data is dependent on the data protection mode that is set for the configuration. This type of transition often requires you to shut down and restart the original primary database and configure it as a new standby database. There are two types of failover operations: *graceful failover* or *forced failover*.

See Also: Section 7.3 for information about graceful and forced failover operations

Figure 7–1 shows a road map of the operations that are discussed in this chapter.

Figure 7–1 Standby Database Road Map



7.2 Database Switchover

A switchover operation transitions the primary database to the standby role and transitions one of the standby databases to the primary role, without resetting the online redo logs of the new primary database. No data is lost regardless of the protection mode that is set for the configuration. However, the start up and shut down requirements are different for physical and logical standby databases:

- After a switchover operation that involves a logical standby database, all databases continue to run normally; there is no need to shut down and restart the primary database or any of the standby databases.
- After a switchover operation that involves a physical standby database, the primary database and the physical standby database that is involved in the switchover operation must be shut down and restarted. There is no need to restart other standby databases that are not involved in the switchover operation.

A switchover operation is performed in two phases: first, the primary database role is switched to the standby role; then, a standby database is selected to assume the primary database role.

This section contains the following topics:

7.2.1 Preparing to Perform a Successful Switchover Operation

Carefully plan each switchover operation so that the primary and standby databases involved have as small a transactional lag as possible.

Note: Do not use a switchover operation to perform a rolling upgrade of Oracle software. However, it may be possible to use a switchover operation to perform a hardware-based rolling upgrade. It may be possible to perform an application software rolling upgrade if necessary schema modifications and data updates have been made to the primary database prior to the switchover operation.

Before starting a switchover operation, verify that:

- Initialization parameter files or server parameter files (SPFILEs) are set up properly

  You should maintain two database initialization parameter files or SPFILEs for each database: one should contain initialization parameter settings for when the database is in the primary role, and the other should contain settings for the standby role. Having two initialization parameter files or SPFILEs allows you to easily start the databases after a switchover operation has occurred.

  Although most initialization parameters will be identical for both the primary and standby databases, some initialization parameters (such as `CONTROL_FILES`, `LOCK_NAME_SPACE`, `LOG_FILE_NAME_CONVERT`, and `DB_FILE_NAME_CONVERT`) may differ.

  See Also: Section 5.6.3 provides sample primary and standby initialization parameter files

- There is network connectivity between the primary and standby databases

  All primary and standby databases in the configuration should have network connectivity to all other databases in the configuration, and for each primary and standby database in the configuration, you should have entries in the `tnsnames.ora` file to identify the other databases in the configuration. You must also have corresponding entries in the `listener.ora` file for each database.

  See Also: *Oracle9i Net Services Administrator's Guide* for information about configuring and administering the `listener.ora` and `tnsnames.ora` initialization files

- All but one primary instance and one standby instance in a Real Application Clusters configuration are shut down

  For a database using Real Application Clusters, only one primary instance and one standby instance can perform the switchover operation. Shut down all other instances prior to the switchover operation.

- The primary database instance is open and the standby database instance is mounted

  The standby database that you plan to transition to the primary role must be mounted before you begin the switchover operation. A physical standby database instance must be mounted and in managed recovery mode before you start the switchover operation. If the physical standby database is open for read-only access, the standby switchover operation will take longer. Ideally, the standby database will also be actively recovering archived redo logs when the database roles are switched.

  See Also: Section 6.3.1

- The standby database that will become the new primary database is in ARCHIVELOG mode

For switchover operations involving a physical standby database, see Section 7.2.2. For switchover operation involving a logical standby database, see Section 7.2.3.

7.2.2 Switchover Operations Involving a Physical Standby Database

This section describes how to perform a switchover operation that changes roles between a primary database and a physical standby database. (See Section 7.2.3 for information about switchover operations involving a logical standby database.)

Always initiate the switchover operation on the primary database and complete it on the physical standby database. The following steps describe how to perform a switchover operation.

On the original primary database:

Step 1 Verify it is possible to perform a switchover operation.

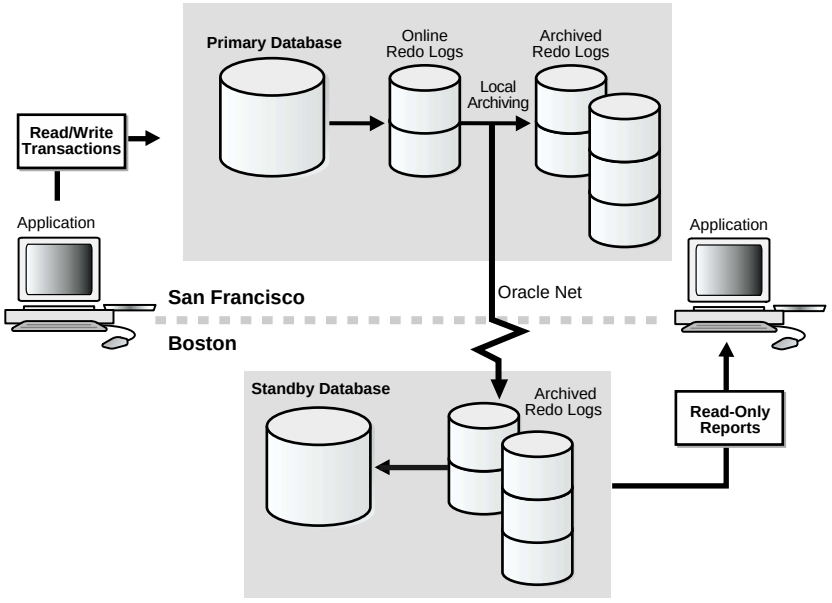
Query the SWITCHOVER_STATUS column of the V$DATABASE fixed view on the primary database to verify that it is possible to perform a switchover operation. For example:

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
 SWITCHOVER_STATUS
 -----------------
 TO STANDBY
 1 row selected
```

The TO STANDBY value in the SWITCHOVER_STATUS column indicates it is possible to switch the primary database to the standby role.

Figure 7–2 shows a two-site Data Guard configuration before the roles of the databases are switched.

Figure 7–2 Data Guard Configuration Before a Switchover Operation



Step 2 Initiate the switchover operation on the primary database.

To transition the primary database to a *physical* standby database role, use the following SQL statement syntax on the primary database:

```
ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY [ {WITH | WITHOUT}
SESSION SHUTDOWN [WAIT | NOWAIT] ];
```

You can include the optional clauses for this statement as follows:

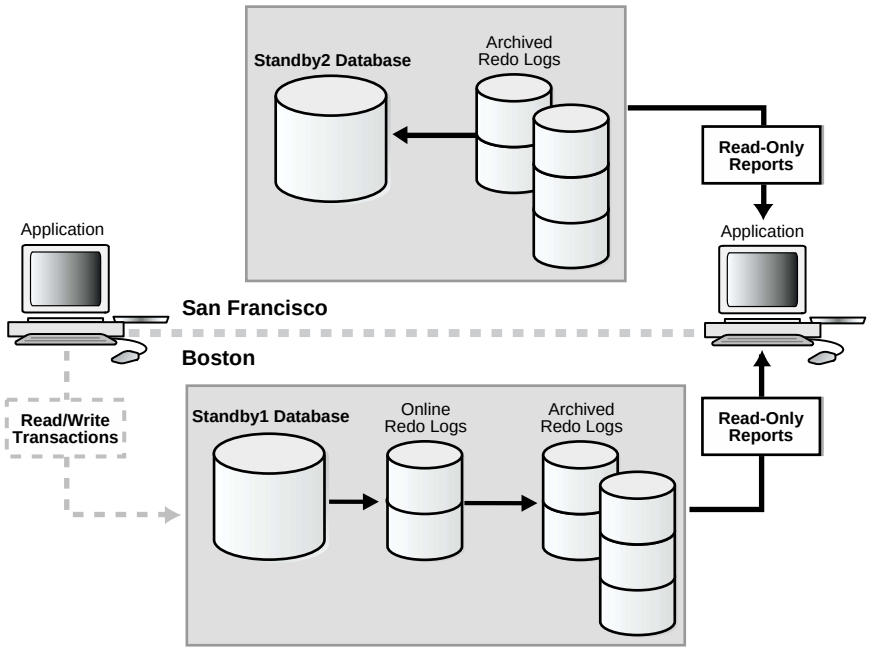
- The switchover operation requires that only one SQL session be active on the primary instance. To terminate all other SQL sessions, include the `WITH SESSION SHUTDOWN` clause on the SQL statement.
- By default, the switchover operation must complete before control is returned to the SQL prompt (this is the same as the `WAIT` clause). To return control before the switchover operation is complete, specify the `NOWAIT` clause.

The primary database role is converted into a standby database. The current control file will be backed up to the current SQL session trace file prior to the switchover operation; this makes it possible to reconstruct a current control file, if necessary.

For example, you may want to reverse the switchover (sometimes referred to as a switchback) and revert the original primary database to the primary role.

Figure 7–3 shows the Data Guard environment after the primary database has been switched over to a standby database and before the original standby database has become the new primary database. At this stage, the Data Guard configuration temporarily has two standby databases.

Figure 7–3 Standby Databases Before Switchover to the New Primary Database



Step 3 Shut down and restart the primary instance.

Shut down the primary instance and restart it without mounting the database:

```
SQL> SHUTDOWN NORMAL;
SQL> STARTUP NOMOUNT;
```

Mount the database as a physical standby database:

```
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

On the original physical standby database:

Step 4 Verify the switchover status in the V$DATABASE view.

After you switch the primary database to the standby role and the switchover notification has been received by the standby database, you should verify whether the switchover notification has been processed by the standby database by querying the SWITCHOVER_STATUS column of the V$DATABASE fixed view on the standby database.

> **Note:** Ensure the physical standby database is in managed recovery mode to allow the primary database switchover notification to be processed by the physical standby database.

For example:

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
SWITCHOVER_STATUS
-----------------
SWITCHOVER PENDING
 1 row selected
```

The SWITCHOVER PENDING value of the SWITCHOVER_STATUS column indicates the standby database is about to switch from the standby role to the primary role.

Step 5 Switch the physical standby database role to the primary role.

You can switch a physical standby database from the standby role to the primary role when the standby database instance is either mounted in managed recovery mode or open for read-only access. It must be mounted in one of these modes so that the primary database switchover operation request can be coordinated.

On the physical standby database that you want to run in the primary role, use the following SQL statement syntax:

```
ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY [ {WITH | WITHOUT} SESSION
SHUTDOWN ] [WAIT | NOWAIT];
```

Note: The `COMMIT TO SWITCHOVER TO PRIMARY` clause automatically creates online redo logs. This may significantly increase the time required to complete the `COMMIT` operation. Oracle Corporation recommends that you always manually add the online redo logs to the standby database prior to starting the switchover operation. Also, define the `LOG_FILE_NAME_CONVERT` initialization parameter to correlate the standby database path names to the online redo logs.

Step 6 Shut down and restart the new primary database.

Shut down the original standby instance and restart it as the new primary database.

```
SQL> SHUTDOWN;
SQL> STARTUP;
```

The selected physical standby database is now transitioned to the primary database role.

Note: There is no need to shut down and restart any standby databases that are online at the time of the switchover operation, but are not involved in it. These standby databases will continue to function normally after the switchover completes.

On the new physical standby database:

Step 7 Start managed recovery operations and log apply services.

Issue the following command to begin managed recovery operations on the new physical standby database:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;
```

On the new primary database:

Step 8 Begin archiving logs to the physical standby database.

Issue the following statements on the new primary database:

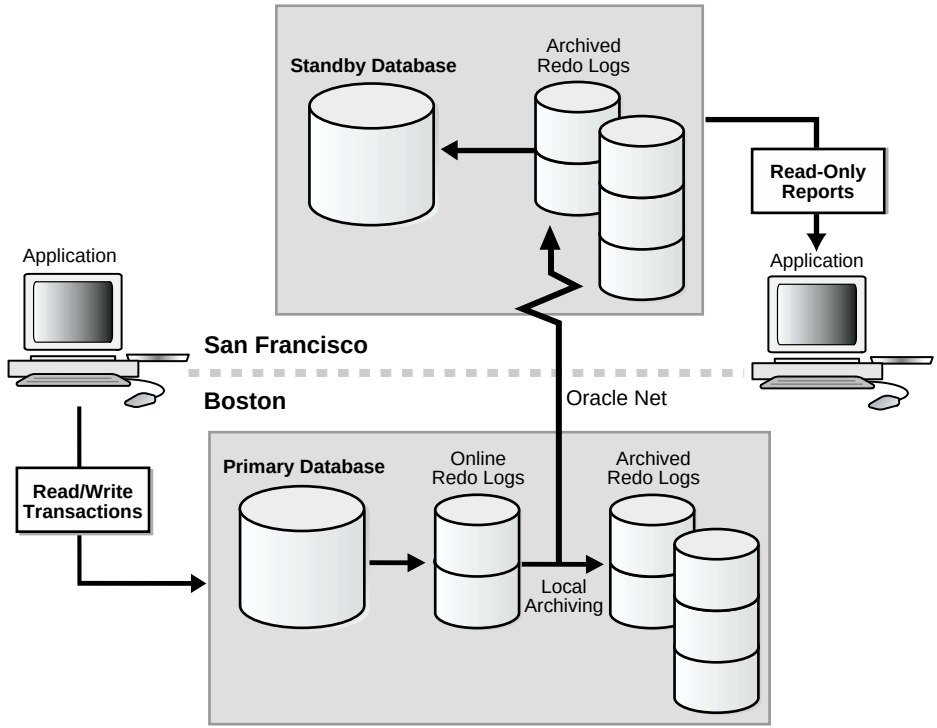
```
SQL> ALTER SYSTEM ARCHIVE LOG START;
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

When you perform a switchover operation involving a physical standby database, the existing control file type is converted in place. That is, the primary database's current control file is converted into a physical standby control file, and a physical standby control file is converted into a primary control file.

The current primary control file is backed up to the trace file before it is converted to be a standby control file. The current standby control file is not backed up to the trace file before becoming a primary control file.

Figure 7–4 shows a Data Guard environment after a switchover has taken place.

Figure 7–4 Data Guard Environment After Switchover



See Also: Chapter 13 and the *Oracle9i SQL Reference* for more information about the SQL statement syntax

7.2.3 Switchover Operations Involving a Logical Standby Database

This section describes how to perform a switchover operation that changes roles between a primary database and a logical standby database. (See Section 7.2.2 for information about switchover operations involving a physical standby database.)

Always initiate the switchover operation on the primary database and complete it on the logical standby database. The following steps outline the SQL statements to perform the switchover operation on databases located on separate nodes.

On the primary database:

Step 1 Verify it is possible to perform a switchover operation.

Begin by verifying it is possible to perform a switchover operation by querying the SWITCHOVER_STATUS column of the V$DATABASE fixed view on the primary database. For example:

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
 SWITCHOVER_STATUS
 -----------------
 TO STANDBY
 1 row selected
```

The TO STANDBY value in the SWITCHOVER_STATUS column indicates it is possible to switch the primary database to the standby role.

Step 2 Place the primary database in a quiesced state.

Ask users to log off the primary database or place it in a quiesced state to ensure there is no update activity on the database.

> **See Also:** *Oracle9i Database Administrator's Guide* for more information on quiescing an Oracle database

Step 3 Switch the primary database to the logical standby database role.

To transition the primary database to a logical standby database role, use the following SQL statement syntax:

```
ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY [WAIT | NOWAIT];
```

This statement ends current update operations on the primary database and prevents any new users from starting update operations. It also puts a marker in the redo log to provide a synchronization point for logical standby database operations.

Executing this statement will also prevent users from making any changes to the data being maintained in the logical standby database.

You can include the optional WAIT or NOWAIT clauses on this statement. By default, the switchover operation must complete before control is returned to the SQL prompt (this is the same as the WAIT clause). To return control before the switchover operation is complete, specify the NOWAIT clause.

The primary database is transitioned to run in the standby database role.

Note: When you transition a primary database to a logical standby database role, you do not have to shut down and restart it.

Step 4 Verify the switchover status in the V$DATABASE view.

After you switch the primary database to the standby role and the switchover notification has been received by the standby database, you should verify whether the switchover notification has been processed by the standby database by querying the SWITCHOVER_STATUS column of the V$DATABASE fixed view on the standby database.

For example:

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
SWITCHOVER_STATUS
-----------------
SWITCHOVER PENDING
 1 row selected
```

The SWITCHOVER PENDING value of the SWITCHOVER_STATUS column indicates the standby database is about to switch from the standby role to the primary role.

Step 5 Defer archiving redo logs.

Defer archiving redo logs on the new logical standby database. For example:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=DEFER;
```

On the original logical standby database:

Step 6 Switch the logical standby database to the primary database role.

On the logical standby database that you want to run in the primary role, use the following SQL statement to switch the logical standby database to the primary role:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY [WAIT | NOWAIT];
```

Note: There is no need to shut down and restart any logical standby databases that are in the Data Guard configuration. Logical standby databases continue to function normally after a switchover operation completes.

Step 7 Enable archiving redo logs.

Enable archiving redo logs to the new logical standby database. For example:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

On the new logical standby database:

Step 8 Begin SQL apply operations.

Enter the following statement to begin SQL apply operations on the new logical standby database.

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY NEW PRIMARY dblink;
```

See Also: Chapter 13 and the *Oracle9i SQL Reference* for more information about the SQL statement syntax

7.2.4 Transitioning Multiple Standby Databases to the Primary Role

Typically, you choose only one standby database to become the new primary database. That is, you switch the primary database to the standby role and choose one standby database to be switched to the primary role.

In some cases, however, you may choose to switch multiple databases from the standby role to the primary role. If you transition multiple standby databases to run in a primary role, each will become a separate and distinct primary database. Any remaining standby databases can be connected to a single primary database only. They cannot be changed later to another primary database.

7.2.5 Validating the Switchover Transition (Physical Standby Databases Only)

After the SQL `ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY` statement has been issued on the primary database, the resulting physical standby database is automatically placed in managed recovery mode. This is necessary to ensure that all unapplied redo logs have been applied to the new standby database. Upon completion of the recovery operation, the physical standby database instance is dismounted.

Examine the new standby alert log to ensure that recovery has finished on the new standby database. Subsequent standby database switchover operations cannot take place until this happens and the original standby database acknowledges that it has received all available archived logs.

The following example shows a successful managed recovery operation in the standby alert log:

```
MRP0: Media Recovery Complete: End-Of-REDO
Resetting standby activation ID 3830496333 (0xe450bc4d)
MRP0: Background Media Recovery process shutdown
```

If the primary database finished archiving the last online redo log, you can query the SEQUENCE# column in the V$ARCHIVED_LOG view to see if the last archived log has been applied on the physical standby database. If the last log has not been applied, you can manually copy the archived log from the primary database to the physical standby database and register it with the SQL ALTER DATABASE REGISTER LOGFILE *filespec* statement. If you then start up the managed recovery process, the archived log will be applied automatically. Query the SWITCHOVER_STATUS column in the V$DATABASE view. The TO PRIMARY value in the SWITCHOVER_STATUS column verifies that switchover to the primary role is now possible.

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
SWITCHOVER_STATUS
-----------------
TO PRIMARY
1 row selected
```

In situations where an error has occurred, it may still be possible to revert the new physical standby database to the original primary database using the following steps:

1. When the switchover procedure to change the role from primary to standby was initiated, a trace file was written in the log directory. This trace file contains the SQL statements to re-create the original primary control file. Locate the trace file and extract the SQL statements into a temporary file. Execute the temporary file from SQL*Plus. This will convert the new physical standby database back to the original primary role.
2. Create a new physical standby control file. This is necessary to resynchronize the primary and physical standby databases. Copy the standby control file to the physical standby sites.
3. Shut down the original physical standby instance and restart it.

If this procedure is successful and archive gap management is enabled, the FAL processes will start and re-archive any missing archived logs to the physical standby database. Force a log switch on the primary database and examine the alert logs on both the primary and physical standby databases to ensure that the archived log sequence numbers are correct.

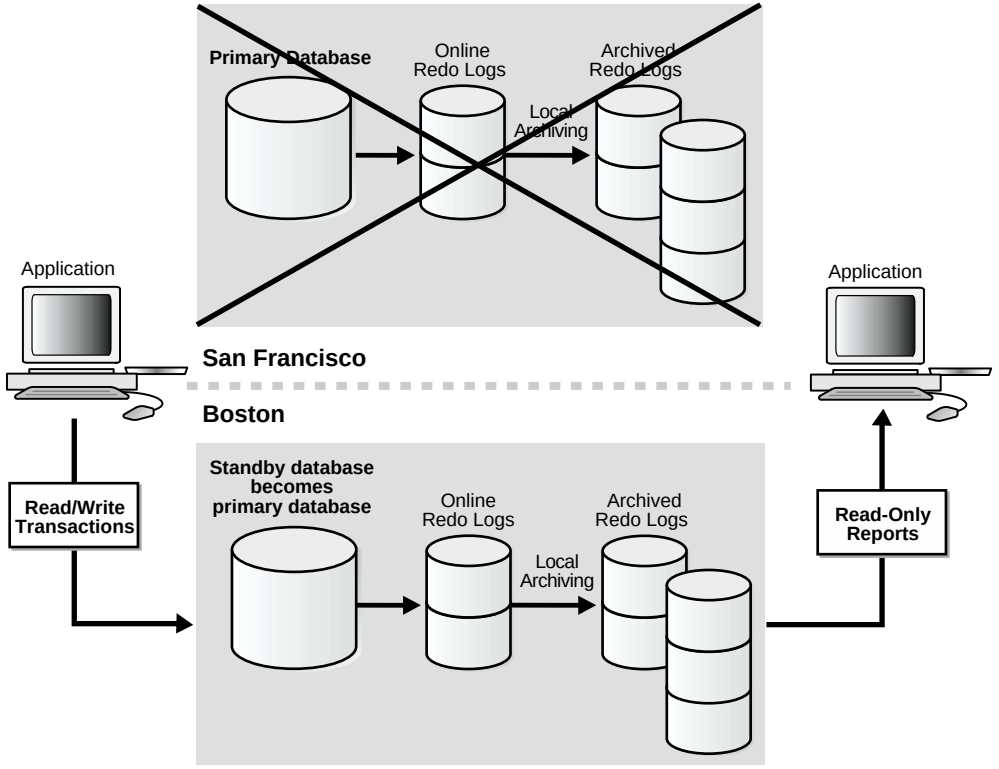
> **See Also:** Section 6.5 for information about archive gap management and Section 6.4.8 for information about locating the trace files

7.3 Database Failover

You invoke a **failover** operation when a catastrophic failure occurs on the primary database, and there is no possibility of recovering the primary database in a timely manner. During a failover operation, the incapacitated primary database is removed from the Data Guard environment and a standby database assumes the primary database role. You invoke the failover operation on the standby database that you want to fail over to the primary role.

Figure 7–5 shows the result of a failover operation from a primary database in San Francisco to a physical standby database in Boston.

Figure 7–5 Failover to a Standby Database



You should not fail over to a standby database except in an emergency, because the failover operation is an unplanned transition that may result in the loss of application data. Once you perform a failover operation, there is no going back. This is because the original primary database is incapacitated and the standby database that you fail over to the primary role is no longer capable of returning to being a standby database in the original configuration.

Most failures can be resolved at a primary site within a reasonable amount of time without failover.

Note: It is not always necessary to fail over to the standby database. In some cases, recovery of the primary database may be faster. See Section 7.3.2.

However, a database that needs complete restoration and recovery (or a database that is partially restored and recovered) can be resolved quickly by failing over to the standby database. Failures such as the following are good reasons to fail over to the standby database:

- Loss of the current online redo log
- Loss of system or rollback tablespace
- Loss of hardware and the database
- Inadvertent database operations that cannot be recovered from quickly, such as dropping a tablespace

There are two types of failover operations: *graceful failover* and *forced failover*. Which to use depends on whether a physical or a logical standby database is participating in the failover operation, and if standby redo logs are used.

This section contains the following topics:

7.3.1 Planning for Database Failover

It is possible for the standby database to have lost transactions committed on the original primary database. This occurs because the current log of the primary database contains database modifications that may not be available to the standby database. Therefore, if the primary database current log is unavailable, data loss may be possible on the standby database.

Depending on the log transport services destination attributes, a graceful failover may provide no data loss or minimal data loss. A forced failover will result in intentional data loss. The database role transition options and requirements are summarized in Table 7–1.

7.3.1.1 No-Data-Loss Failover

On a standby database, no-data-loss failover is possible only when the corresponding primary database is operating in a data protection mode that provides no-data-loss semantics, such as the maximum protection and maximum availability modes. This means that all primary database archived redo logs

necessary for no-data-loss failover are available on the standby system. However, it is possible that the archived redo logs have not yet been applied to the standby database. No-data-loss failover requires that all archived redo logs are first applied. If the archived redo logs are not applied, failover to a standby database will result in data loss relative to the primary database.

7.3.1.2 Minimal-Data-Loss Failover

Standby database minimal-data-loss failover occurs when primary database modifications have not all been applied, or when the corresponding primary database is operating in a data protection mode that allows data-divergence semantics, such as maximum performance mode. This results in standby database data loss relative to the primary database. The amount of data loss can be controlled by primary database archived log destination attributes and the availability of standby redo logs at the standby database.

7.3.1.3 Physical Standby Databases and Standby Redo Logs

The presence of standby redo logs on a physical standby database determines whether a graceful failover with no data loss is possible. To avoid or minimize data loss during a database failover operation involving a physical standby database, you should always use standby redo logs. (See Section 5.8.4 for information about creating standby redo logs.) If the primary database is lost, the use of standby redo logs allows primary database modifications to be automatically recovered on the physical standby database. Depending on the log transport services options used, this may provide no-data-loss or minimal-data-loss failover.

Note: You should not fail over to a physical standby database to test whether it is being updated correctly. Instead, open the standby database in read-only mode.

If the physical standby database does not use standby redo logs, database failover will almost always result in data loss. However, if the primary database is not completely lost, it may be possible to manually recover the primary database online redo logs and apply them on the standby database. (Section 7.3.2 describes no-data-loss recovery without performing a failover operation.) In both cases, the original primary database is incompatible with the resulting new primary database.

Table 7–1 summarizes the database role transition options and requirements. Also, see Chapter 5 for more information about setting up log transport services.

Table 7–1 Summary of Role Transitions

Summary of Role Transitions	Database Switchover	Graceful Database Failover - No Data Loss	Graceful Database Failover - Minimal Data Loss	Forced Database Failover
Data Loss Potential	No data loss	No data loss	Minimal data loss	Probable data loss
Data Divergence Potential	Not applicable	No divergence if in maximum protection mode Possible divergence if in maximum performance mode	Possible divergence	Probable divergence
Requirements to Shut Down and Restart Database	For switchovers with a physical standby database: the participating physical standby database and the original primary database For switchovers with a logical standby database: None	New primary database	New primary database	New primary database and all other standby databases that were not the target of the failover operation

Required Archived Log Destination Attributes	`SERVICE`
	`REGISTER`
		`SERVICE`
		`LGWR`
		`SYNC`
		`AFFIRM`
		`REGISTER`
		`MANDATORY`
			`SERVICE`
			`LGWR`
			`ASYNC`
			`REGISTER`
				`SERVICE`

Table 7–1 (Cont.) Summary of Role Transitions

Summary of Role Transitions	Database Switchover	Graceful Database Failover - No Data Loss	Graceful Database Failover - Minimal Data Loss	Forced Database Failover
Standby Redo Logs	Optional	Required for physical standby databases	Required for physical standby databases	Optional for physical standby databases
SQL Statement Issued on the Primary Database	ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY	Not applicable	Not applicable	Not applicable
SQL Statement Issued on the Standby Database	ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY	Physical standby: ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH Logical standby: ALTER DATABASE ACTIVATE LOGICAL STANDBY DATABASE Physical and Logical: ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY	Physical standby: ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH Logical standby: ALTER DATABASE ACTIVATE LOGICAL STANDBY DATABASE Physical and Logical: ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY	ALTER DATABASE ACTIVATE STANDBY DATABASE

7.3.2 Primary Database No-Data-Loss Recovery

It may be faster to recover the primary database, even when you are using a no-data-loss environment, than to fail over a standby database to the primary role.

When the primary database is operating in no-data-loss mode and a system failure occurs, you can restart the primary database without incurring data loss on the corresponding standby databases. The following failure scenarios are automatically recovered by log transport services:

- Instance recovery, which is only possible in a Real Application Clusters configuration, occurs in an open primary database when one instance discovers that another instance has failed. A surviving instance automatically uses the online redo log to recover the committed data in the database buffers that was lost when the instance failed. Further, the Oracle database server undoes any

transactions that were in progress on the instance when it failed, and then clears any locks held by the failed instance after recovery is complete.

- Failure recovery occurs when either a single-instance database fails or all instances of a multi-instance primary database fail. In failure recovery, an instance must first open the database and then execute recovery operations. In general, the first instance to open the database after a failure or `SHUTDOWN ABORT` statement automatically performs failure recovery.

In both of these scenarios, the standby databases participating in the no-data-loss environment will be automatically resynchronized with the recovered primary database. However, for the resynchronization to occur, the primary database archived log destinations must be properly established to identify the standby databases, and network connectivity must exist between the primary database and the standby databases.

Note: When the primary database is running in a mode other than no-data-loss, the archiver (ARC*n*) process is responsible for performing resynchronization activities to the primary database.

7.3.3 Graceful Failover

Graceful database failover automatically recovers some (minimal data loss) or all (no data loss) of the original primary database application data. There is also graceful failover for unavoidable data loss for physical standby databases whose standby redo logs either are not available or contain corruption. The data protection modes that you set act as failure policies that dictate how to manage the primary database if a standby database becomes inaccessible. Depending on whether you have defined the maximum protection, maximum availability, or maximum performance protection mode for the configuration and what attributes you have chosen for log transport services, it may be possible to automatically recover some or all of the primary database modifications.

One of the consequences of a graceful database failover is that you must shut down and restart the new primary database. However, standby databases not participating in the failover operation do not need to be shut down and restarted.

This section contains the following topics:

- Graceful Failover with No Data Loss
- Graceful Failover with Unavoidable Data Loss
- Graceful Failover with Minimal Data Loss

7.3.3.1 Graceful Failover with No Data Loss

No-data-loss failover to the standby database can be achieved when the primary database log transport services has been previously set up to provide this capability. The primary database must be configured with the maximum protection (physical standby databases only) or maximum availability mode. These modes require that the primary database have the following archived log destination attributes on the LOG_ARCHIVE_DEST_*n* initialization parameter:

- LGWR - archived by the LGWR process
- SYNC - synchronous network transmission
- AFFIRM - synchronous archived log disk I/O
- REGISTER - archived log registration on the standby database

Furthermore, the archived log destinations cannot have the DEPENDENCY attribute defined, which provides for archived log destination dependencies. In addition, physical standby databases participating in the no-data-loss environment must have standby redo logs available.

See Also: Section 5.8.4 for an overview of standby redo logs

Graceful Database Failover for Physical Standby Databases

Depending on how you have defined the protection mode for the configuration and the attributes for log transport services, it may be possible to automatically recover some or all of the primary database modifications.

Step 1 Initiate the graceful failover operation on the standby database.

If a failure occurs at the primary site, a graceful failover allows you to salvage incomplete standby redo logs using the following statement:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH;
```

Note: To issue the SQL ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH statement, you must set the COMPATIBLE initialization parameter to 9.0.1.0.0 (or higher) in the initialization parameter file. For example, add the following line to the initialization parameter file:

```
COMPATIBLE=9.0.1.0.0
```

Once you set the COMPATIBLE initialization parameter to 9.0.1.0.0 (or higher), its value must remain at this release number from this point forward. The only way to revert to an 8.0 or 8.1 release is to cancel the SQL ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH statement before the standby redo logs have been applied.

Note that when you start managed recovery, you can specify the FINISH control option to define the mode of the recovery operation or to alter the mode of an ongoing managed recovery operation. If the FINISH control option is being used to alter the mode of an ongoing managed recovery operation, specifying the NOWAIT option allows control to be returned to the foreground process before the recovery operation completes.

When the NOWAIT option is used, you can query the V$MANAGED_STANDBY fixed view to verify when the managed recovery operation is finished. There should be no rows selected by the following statement after recovery has completed:

```
SQL> SELECT PROCESS FROM V$MANAGED_STANDBY;
no rows selected
```

Step 2 Convert the physical standby database to the primary role.

Once the RECOVER...FINISH statement has successfully completed, you must issue the COMMIT TO SWITCHOVER TO PRIMARY statement to convert the physical standby database to the primary database role. You can no longer use this database as a standby database that is compatible with the old primary database.

After a graceful failover, subsequent redo logs from the original primary database cannot be applied to the standby database. The standby database is *on hold*. The standby redo logs have been archived and should be copied to, registered, and recovered on other standby databases derived from the original primary database. This will happen automatically if the standby destinations are defined on the new primary database.

You must now change the standby database into the new primary database by issuing the following statement:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
```

There is no need to shut down and restart other standby databases in the configuration that were not participants in the failover operation.

Step 3 Register the missing redo logs.

Once the archived standby redo logs have been received and recovered on all destinations, the other standby databases are ready to receive redo logs from the new primary database. To register the copied archived redo logs, issue the following statement on each standby database:

```
SQL> ALTER DATABASE REGISTER LOGFILE '/standby/arch_dest/arch_1_101.arc';
```

To use the original primary database, you must re-create it from a backup copy of the new primary database.

Graceful Database Failover for Logical Standby Databases.

When a failover occurs, perform the following steps to stop SQL apply operations on the logical standby database, register missing redo logs, and switch the logical standby database to the primary database role.

Step 1 Copy redo logs to the logical standby site.

If redo logs exist on the primary database that have not yet been applied on the logical standby database, manually copy the redo logs to that standby database.

Step 2 Register the missing redo logs.

Specify the `REGISTER LOGFILE` clause from the standby database to register log files from the failed primary database. This operation is required unless missing log files from the failed primary database have been copied to the directory specified in the `STANDBY_ARCH_DEST` initialization parameter.

Specify the following statements on the logical standby database to register the missing redo logs that you copied manually from the original primary database:

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE 'filespec';
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
```

Step 3 Ensure that all redo logs have been applied to the new primary database.

On the new primary database, ensure that the remaining redo logs have been applied by checking the V$LOGSTDBY view for idle processes.

> **See Also:** Chapter 14 for information about the V$LOGSTDBY view

Step 4 Activate the new primary database.

Execute the following statements on the new primary database to stop SQL apply operations and activate the database in the primary database role:

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
SQL> ALTER DATABASE ACTIVATE LOGICAL STANDBY DATABASE;
```

Step 5 Prepare the new logical standby database.

Prepare the new logical standby database to apply all remaining redo logs from the old primary database and begin applying redo logs from the primary database.

On the new logical standby database, apply the remaining redo logs from the old primary database and begin SQL apply operations from the new primary database:

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY NEW PRIMARY dblink;
```

When this statement completes, all remaining archived redo logs will have been applied. Depending on the volume of work to be done, this operation may take some time to complete. A database link to an account with access to the logical standby system views is necessary.

Step 6 Shut down and restart the original primary database.

To return the system to its original database configuration, re-create the original primary database as a logical standby database, add it to the new Data Guard configuration, and perform a switchover operation to return it to the primary database role.

7.3.3.2 Graceful Failover with Unavoidable Data Loss

For physical standby databases only, you can perform a graceful failover operation that allows you to fail over to a physical standby database whose standby redo logs either are not available or contain corruption. This type of graceful failover operation, called unavoidable-data-loss failover, is performed using the FINISH

SKIP STANDBY LOGFILE clause of the ALTER DATABASE RECOVER MANAGED STANDBY DATABASE SQL statement.

Unavoidable-data-loss failover allows you to transition a physical standby database to the primary role even though some data loss may occur. The FINISH SKIP STANDBY LOGFILE clause directs managed recovery operations to:

- Ignore the contents of the standby redo logs, even if they are available.
- Apply as much archived redo as possible before ending the recovery operation.

  When you issue the FINISH SKIP STANDBY LOGFILE clause, log apply services apply archived redo logs until the first unarchived redo log is encountered. All archived redo logs beyond this point are not recovered and are unavoidably lost. In a Real Application Clusters environment, this may result in greater data loss due to the dependencies on the redo data by multiple instances.

To perform an unavoidable-data-loss graceful failover operation, perform these steps:

Step 1 Set attributes for the LOG_ARCHIVE_DEST_n parameter.

For a standby database to be eligible for graceful failover, it must be a physical standby database and the following attributes must be set on the LOG_ARCHIVE_DEST_*n* initialization parameter:

- SERVICE=*net_service_name*

  Specifies a valid Oracle Net service name identifying the standby database

- REGISTER

  Indicates that the location of the archived redo log is to be recorded at the corresponding destination

Step 2 Begin the graceful failover and managed recovery operations.

Issue the following SQL statements to perform a graceful failover to a standby database without standby redo logs:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH SKIP
  2> STANDBY LOGFILE;
```

After the graceful failover operation completes successfully, archived redo logs containing the failover information are automatically archived to the destinations defined by the LOG_ARCHIVE_DEST_*n* initialization parameter on the physical standby database that will be transitioned to the primary database role. These

archived log destinations were defined when the database was still running in the standby database role, in anticipation of the standby database assuming the primary database role.

Step 3 Optionally, copy archived redo logs to the other standby destinations.

If the standby database does not define archived log destinations, you may need to manually copy and register the resulting failover archived redo logs on each of the remaining physical standby databases in the configuration. Use the ALTER DATABASE REGISTER LOGFILE statement to register a copied archived redo log at a standby destination.

Step 4 Switch the physical standby database to the primary role.

Once the RECOVER...FINISH SKIP STANDBY LOGFILE statement has successfully completed, you must issue the COMMIT TO SWITCHOVER TO PRIMARY statement to transition the physical standby database to run in the primary database role.

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
```

Until you issue the COMMIT TO SWITCHOVER TO PRIMARY statement, the database is on hold. While on hold, the database is not usable as a standby database. After you issue the COMMIT TO SWITCHOVER TO PRIMARY statement, the database must be shut down and restarted in the primary role.

Step 5 Drop old standby redo logs and create a standby database from the new primary database.

There is no need to shut down and restart other standby databases in the configuration. Once the archived redo logs containing the failover information have been received and recovered on all destinations, the other standby databases are ready to receive redo logs from the new primary database.

Note: Some standby redo logs may be created during a graceful failover operation. Thus, you may want to consider dropping them after completion of the failover operation.

A graceful failover assumes that the primary database is incapacitated; therefore, performing a graceful failover transitions the standby database to the primary role and eliminates the original primary database from participating in the standby configuration. To reuse the old primary database in the new configuration, you

must create it as a standby database, using a backup copy of the new primary database.

7.3.3.3 Graceful Failover with Minimal Data Loss

Minimal-data-loss failover to the standby database can be achieved when the primary database log transport services have been previously set up to provide this capability. Standby databases participating in the minimal-data-loss environment must have the following archived log destination attributes on the LOG_ARCHIVE_DEST_*n* initialization parameter:

- LGWR - archived by the LGWR process
- ASYNC - asynchronous network transmission
- SERVICE - standby database
- REGISTER - archived log registration on the standby database

Using a lower block count for the archived log destination ASYNC attribute minimizes the amount of potential data loss but possibly decreases primary database throughput. The archived log destination AFFIRM attribute can also be used to further reduce potential data loss. Furthermore, the archived log destinations cannot have the DEPENDENCY attribute defined, which provides for archived log destination dependencies.

In addition, it is recommended that physical standby databases participating in the minimal-data-loss environment should have standby redo logs available. If a failure occurs at the primary site, there will be incomplete standby redo logs on the physical standby database. The standby redo log contents can be salvaged with the RECOVER...FINISH statement, for example:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH;
```

Note: This is the same statement used for no-data-loss recovery in which a physical standby database is participating. This statement finishes recovery of the standby redo logs, applying as much data to the physical standby database as possible.

Once the RECOVER...FINISH statement has successfully completed, you must issue the COMMIT TO SWITCHOVER TO PRIMARY statement to transition the physical standby database to run in the primary database role. You can no longer use this database as a standby database.

After a graceful failover, subsequent redo logs from the original primary database cannot be applied to the physical standby database. The standby database is *on hold*. The standby redo logs have been archived and should be copied to, registered, and recovered on other physical standby databases derived from the original primary database. This will happen automatically if the standby destinations are defined on the new primary database.

You must now change the standby database to be the new primary database by issuing the following statement:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
```

There is no need to shut down and restart other standby databases in the configuration that were not participants in the failover operation. Once the archived standby redo logs have been registered and recovered on all destinations, the other standby databases are ready to receive redo logs from the new primary database. To register the copied archived redo logs, issue the following statement on each standby database:

```
SQL> ALTER DATABASE REGISTER LOGFILE '/standby/arch_dest/arch_1_101.arc';
```

The original primary database must be re-created from a backup copy of the new primary database.

7.3.4 Forced Failover

Forced database failover forces a standby database into the primary role, possibly resulting in lost application data. (This is true even when standby redo logs are configured on the physical standby databases.) Do not perform a forced failover operation unless a software or system failure results in the loss of the primary database. Perform a graceful failover whenever possible.

The syntax for physical and logical standby database environments is the same. You invoke a forced failover using the SQL `ALTER DATABASE ACTIVATE {PHYSICAL | LOGICAL} STANDBY DATABASE` statement. This changes the state of a standby database to an active database in preparation for it to become the primary database.

One consequence of a forced database failover is that you must re-create the old primary database and other standby databases from a copy of the new primary database. Another consequence is that application data may be lost unless the standby database is running in maximum protection or maximum availability mode at the time of the primary database failure. Depending on the nature of the emergency, you may not have access to your primary database files. If you do have access, you should attempt to archive the **current online redo log** on the primary

database manually, and then copy and apply all available archived redo logs to the standby database that you plan to change to the primary database role.

Before you can execute a forced failover, Oracle Corporation recommends that you try to perform a graceful failover operation. If you must execute a forced failover operation, you must first stop the primary database archiving operations to the standby database.

Note: You cannot perform a forced failover operation to a physical standby database on which standby redo logs are present unless you indicate that it is acceptable to skip applying the contents of the standby redo log with the `FINISH SKIP STANDBY LOGFILE` keywords of the `RECOVER MANAGED STANDBY DATABASE` clause of the SQL `ALTER DATABASE` statement.

See Section 5.3.1.2 for information about disabling archived log destinations using the `DEFER` attribute of the `LOG_ARCHIVE_DEST_STATE_`*n* initialization parameter.

This section contains the following topics:

- Recovering Primary Database Modifications
- Standby Database Failover
- Intentional Data Loss (Physical Standby Databases Only)

7.3.4.1 Recovering Primary Database Modifications

Prior to performing the standby database failover operation, you should transfer as much of the missing primary database contents as possible to the standby database. The more database modifications that can be transferred to the standby site, the closer the new primary database will match the original primary database.

To move modifications from the primary database to the standby database, take the following steps:

1. If possible, archive the current log group of the primary database locally; for example:

   ```
   SQL> ALTER SYSTEM LOG CURRENT;
   ```

   In a true database failover situation, this is seldom possible. However, if you are able to sufficiently recover the primary database to permit archiving to online redo logs, then you can probably recover the entire primary database.

2. For physical standby databases running in maximum protection mode only: By default, you cannot fail over to a physical standby database whose control file was created from a primary database operating in the maximum protection mode, even if the primary database is no longer available. To perform a forced failover operation, you must put the standby database in maximum performance mode by issuing the following statement on the physical standby database:

   ```
   SQL> ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE PERFORMANCE;
   ```

3. If possible, manually copy all primary database archived log groups to the standby database.

   Use an appropriate operating system utility for transferring binary data; for example:

   ```
   % cp /oracle/arch_dest/*.arc /standby/arch_dest
   ```

4. Register the copied archived logs with the standby database using the SQL `ALTER DATABASE REGISTER LOGFILE` statement. Each archived log must be registered individually, for example:

   ```
   SQL> ALTER DATABASE REGISTER LOGFILE '/standby/arch_dest/arch_1_101.arc';
   SQL> ALTER DATABASE REGISTER LOGFILE '/standby/arch_dest/arch_1_102.arc';
   ```

5. Apply all available archived logs to the standby database; for example:

   - On a physical standby database, use the following statement to apply archived redo logs:

     ```
     SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH;
     ```

     Note: Do not use the `RECOVER...FINISH` clause when applying manually copied archived logs to the standby database.

   - On a logical standby database, specify the following statements to register the missing redo logs that you copied manually from the original primary database:

     ```
     SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
     SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE 'filespec';
     SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
     ```

Once you complete all of these steps, the standby database is rolled forward as close to the time of the primary database failure as possible.

7.3.4.2 Standby Database Failover

Perform the following steps to fail over to the standby database:

1. Ensure that your standby database is mounted in EXCLUSIVE mode by executing the following query:

```
SQL> SELECT NAME,VALUE FROM V$PARAMETER WHERE NAME='cluster_database';
NAME                    VALUE
------------------------  -------------------------------------
cluster_database          FALSE
1 row selected.
```

   If the value is `true`, then the database is *not* mounted exclusively and you must shut down, set the parameter to `false`, and restart the instance. If the value is `false`, then the database is mounted exclusively.

2. Fail over to the standby database:

   Note: This statement resets the online redo logs.

```
SQL> ALTER DATABASE ACTIVATE LOGICAL STANDBY DATABASE;
```

   Note: The ACTIVATE STANDBY DATABASE clause automatically creates online redo logs. This may significantly increase the time required to complete the failover operation. Oracle Corporation recommends that you always manually add the online redo logs to the standby database prior to starting the failover operation. Also, define the LOG_FILE_NAME_CONVERT initialization parameter to correlate the standby database path names to the online redo logs.

3. Shut down the standby instance:

```
SQL> SHUTDOWN IMMEDIATE;
```

4. At this point, the former standby database is now your primary database. Back up your new primary database. This task, while not required, is a

recommended safety measure, because you cannot recover changes made after failover without a backup.

5. Start the new primary instance in read / write mode:

```
SQL> STARTUP PFILE=initPRIMARY.ora
```

Now that you have switched the database role, use the proper initialization parameter file.

> **See Also:** Section 5.6.3 for information about setting up initialization parameter files for the standby and primary roles

7.3.4.3 Intentional Data Loss (Physical Standby Databases Only)

If the physical standby database has standby redo logs, you should always attempt a graceful failover operation. However, this may result in undesirable modifications being applied to the physical standby database. The following example shows the error produced if you attempt to fail over a physical standby database without applying all received archived logs:

```
SQL> ALTER DATABASE ACTIVATE STANDBY DATABASE;
ERROR at line 1:
ORA-16140: standby online logs have not been recovered
```

You can force the failover to the physical standby database, which will result in lost data. For example:

```
SQL> ALTER DATABASE ACTIVATE STANDBY DATABASE
  2> SKIP STANDBY LOGFILE;
```

> **Note:** Oracle Corporation recommends that you always apply all data to the standby database prior to the failover operation.

8

Managing a Physical Standby Database

This chapter describes how to manage the physical standby environment. Oracle9*i* Data Guard provides the means to easily manage, manipulate, and change the physical standby database in many ways.

This chapter contains the following topics:

8.1 Backing Up the Primary Database Using the Standby Database

You can use the Recovery Manager utility (RMAN) to back up the primary database at the standby site. This allows the standby database to off-load the task of database backup from the primary database. Using RMAN at the standby site, you can back up the datafiles and the archived redo logs while the standby database is in managed recovery mode. These backups can be later restored to the primary database with RMAN.

The primary control file, however, still must be backed up at the primary site. This is not a problem, because the control file is significantly smaller than the datafiles.

> **See Also:** *Oracle9i Recovery Manager User's Guide* for more details about RMAN backup and recovery of a primary database using a standby database and Section 10.1.10 for a relevant scenario

8.2 Monitoring Events That Affect the Standby Database

To prevent possible problems, you should be aware of events that affect a standby database and learn how to monitor them. Most changes to a primary database are automatically propagated to a standby database through archived redo logs and thus require no user intervention. Nevertheless, some changes to a primary database require manual intervention at the standby site.

This section contains the following topics:

8.2.1 Dynamic Performance Views (Fixed Views)

The Oracle database server contains a set of underlying views that are maintained by the server. These views are often called **dynamic performance views** because they are continuously updated while a database is open and in use, and their contents relate primarily to performance. These views are also called **fixed views,** because they cannot be altered or removed by the database administrator.

These view names are prefixed with either V$ or GV$, for example, `V$ARCHIVE_DEST` or `GV$ARCHIVE_DEST`.

Standard dynamic performance views (V$ fixed views) store information on the local instance. In contrast, global dynamic performance views (GV$ fixed views), store information on all open instances. Each V$ fixed view has a corresponding GV$ fixed view.

The following fixed views contain useful information for monitoring the Data Guard environment:

- `V$ARCHIVE_DEST`

  Describes the archived redo log destinations associated with the current instance on the primary site. You can use this view to find out the current settings of your archived redo log destinations.

- `V$ARCHIVE_DEST_STATUS`

  This view, an extension of the `V$ARCHIVE_DEST` view, displays runtime and configuration information for the archived redo log destinations. You can use

this view to determine the progress of archiving to each destination. It also shows the current status of the archive destination.

- V$ARCHIVE_GAP

  Provides information about archive gaps on a standby database. You can use this view to find out the current archive gap that is blocking recovery.

- V$ARCHIVED_LOG

  Displays archived redo log information from the control file, including archived log names. This view gives you information on which log has been archived to where on your primary database. On the primary database, this view describes the logs archived to both the local and remote destinations. On a standby database, this view provides information about the logs archived to this standby database. You can use this fixed view to help you to track archiving progress to the standby system by viewing the applied field.

- V$DATABASE

  Contains database information from the control file. You can use this view to quickly find out if your database is a primary or a standby database, as well as its switchover status and standby database protection mode.

- V$DATAFILE

  Contains datafile information from the control file. You can query this fixed view to verify that the standby datafiles are correctly renamed after your standby database is re-created.

  See Also: Table 10–1 and Section 10.1.3.2 for information on renaming standby datafiles when re-creating a standby database

- V$DATAGUARD_STATUS

  Displays and logs events related to Data Guard since the instance was started.

- V$LOG

  Contains log file information from the online redo logs. You can use the information about the current online log on the primary database from this view as a reference point to determine how far behind your standby database is in receiving and applying logs.

- V$LOGFILE

  Contains static information about the online redo logs and standby redo logs.

- V$LOG_HISTORY

  Contains log history information from the control file, including a record of the latest archived log that was applied.

- V$MANAGED_STANDBY

  Displays current and status information for some Oracle database server processes related to Data Guard. This view can show both foreground and background processes. You can use this view to monitor the various Data Guard recovery and archiving processes on the standby system.

- V$STANDBY_LOG

  Provides information about the standby redo logs. Standby redo logs are similar to online redo logs, but they are only used on a standby database receiving logs from the primary database using the log writer process.

  See Also: Chapter 14, "Views" and the *Oracle9i Database Reference* for additional information on view columns

8.2.2 Monitoring the Primary and Standby Databases

Table 8–1 indicates whether a primary database event is automatically administered by log transport services and log apply services or requires additional intervention by the database administrator (DBA) to be propagated to the standby database. It also describes how to respond to these events.

Table 8–1 Troubleshooting Primary Database Events

Primary Database Event	Primary Site Problem Report Location	Standby Site Problem Report Location	Recommended Response
Archiving errors	■ ERROR column of V$ARCHIVE_DEST view ■ Alert log ■ ARCHIVED column of V$LOG view ■ Archiving trace files	Remote file server (RFS) process trace file	Fix the problem reported in the alert log file or trace file and resume archiving to the destination. If the problem persists or archiving performance is degraded, you can create scripts to send the archived redo logs.
Thread events	■ Alert log ■ V$THREAD view	Alert log	Thread events are automatically propagated through archived logs, so no extra action is necessary.
Redo log changes	Alert log V$LOG view and STATUS column of V$LOGFILE view	Alert log	Redo log changes do not affect the standby database unless a redo log is cleared or lost. In these cases, you must rebuild the standby database. See Section 3.3. Clears the logs on the standby database with the ALTER DATABASE CLEAR LOGFILE statement. See Section 8.3.9.
Issue a CREATE CONTROLFILE statement	Alert log	Database functions normally until it encounters redo depending on any parameter changes.	Re-create the standby control file. See Section 8.3.8. Re-create the standby database if the primary database is opened with the RESETLOGS option. See Section 3.3.
Managed recovery performed	Alert log	Alert log	Re-create the standby database if the RESETLOGS option is utilized. See Section 3.3.
Tablespace status changes (made read / write or read-only, placed online or offline)	■ DBA_TABLESPACES view ■ Alert log	■ Verify that all datafiles are online. ■ V$RECOVER_FILE view	Status changes are automatically propagated, so no response is necessary. Datafiles remain online.

Table 8–1 (Cont.) Troubleshooting Primary Database Events

Primary Database Event	Primary Site Problem Report Location	Standby Site Problem Report Location	Recommended Response
Add datafile or create tablespace	■ DBA_DATA_FILES view ■ Alert log	■ ORA-283, ORA-1670, ORA-1157, ORA-1110 ■ Standby recovery stops.	If you have not set the STANDBY_FILE_MANAGEMENT initialization parameter to auto, you must re-create the control file on the standby database. See Section 8.3.8.
Drop tablespace	■ DBA_DATA_FILES view ■ Alert log	Alert log	If you have not set the STANDBY_FILE_MANAGEMENT initialization parameter to auto, you must refresh the control file on the standby database. See Section 8.3.8.
Tablespace or datafile taken offline, or datafile is deleted offline	■ V$RECOVER_FILE view ■ Alert log The tablespace or datafile requires recovery when you attempt to bring it online.	■ Verify that all datafiles are online. ■ V$RECOVER_FILE view	Datafiles remain online. The tablespace or datafile is fine after standby database activation.
Rename datafile	Alert log	Alert log	See Section 8.3.2.
Unlogged or unrecoverable operations	■ Direct loader invalidates block range redo entry in online redo log. Check V$DATAFILE view. ■ V$DATABASE view	Alert log. File blocks are invalidated unless they are in the future redo, in which case they are not touched.	Unlogged changes are not propagated to the standby database. See Section 8.3.7 on how to apply these changes.
Recovery progress	■ V$ARCHIVE_DEST_STATUS view ■ Alert log	■ V$ARCHIVED_LOG view ■ V$LOG_HISTORY view ■ V$MANAGED_STANDBY view ■ Alert log	Check the views on the primary site and the standby site for recovery progress. See Section 6.4.

Table 8–1 (Cont.) Troubleshooting Primary Database Events

Primary Database Event	Primary Site Problem Report Location	Standby Site Problem Report Location	Recommended Response
Autoextend a datafile	Alert log	May cause operation to fail on standby database because it lacks disk space.	Ensure that there is enough disk space for the expanded datafile.
Issue `OPEN RESETLOGS` or `CLEAR UNARCHIVED LOGFILES` statements	Alert log	Standby database is invalidated.	Rebuild the standby database. See Section 3.3.
Change initialization parameter	Alert log	May cause failure because of redo depending on the changed parameter.	Dynamically change the standby parameter or shut down the standby database and edit the initialization parameter file. See Chapter 11, "Initialization Parameters".

8.2.3 Determining Which Logs Have Been Applied to the Standby Database

Query the `V$LOG_HISTORY` view on the standby database, which records the latest log sequence number that has been applied. For example, issue the following query:

```
SQL> SELECT THREAD#, MAX(SEQUENCE#) AS "LAST_APPLIED_LOG"
  2> FROM V$LOG_HISTORY
  3> GROUP BY THREAD#;

THREAD# LAST_APPLIED_LOG
------- ----------------
      1              967
```

In this example, the archived redo log with log sequence number 967 is the most recently applied log.

You can also use the `APPLIED` column in the `V$ARCHIVED_LOG` fixed view on the standby database to find out which log is applied on the standby database. The column displays `YES` for the log that has been applied. For example:

```
SQL> SELECT THREAD#, SEQUENCE#, APPLIED FROM V$ARCHIVED_LOG;

   THREAD#  SEQUENCE# APP
---------- ---------- ---
         1          2 YES
```

```
         1          3 YES
         1          4 YES
         1          5 YES
         1          6 YES
         1          7 YES
         1          8 YES
         1          9 YES
         1         10 YES
         1         11 NO

10 rows selected.
```

8.2.4 Determining Which Logs Have Not Been Received by the Standby Site

Each archive destination has a destination ID assigned to it. You can query the DEST_ID column in the V$ARCHIVE_DEST fixed view to find out your destination ID. You can then use this destination ID in a query on the primary database to discover logs that have not been sent to a particular standby site.

For example, assume the current local archive destination ID on your primary database is 1, and the destination ID of one of your remote standby databases is 2. To find out which logs have not been received by this standby destination, issue the following query on the primary database:

```
SQL> SELECT LOCAL.THREAD#, LOCAL.SEQUENCE# FROM
  2> (SELECT THREAD#, SEQUENCE# FROM V$ARCHIVED_LOG WHERE DEST_ID=1) LOCAL
  3>  WHERE
  4> LOCAL.SEQUENCE# NOT IN
  5> (SELECT SEQUENCE# FROM V$ARCHIVED_LOG WHERE DEST_ID=2 AND
  6> THREAD# = LOCAL.THREAD#);

   THREAD#  SEQUENCE#
---------- ----------
  1        12
  1        13
  1        14
```

The preceding example shows the logs that have not yet been received by standby destination 2.

8.3 Responding to Events That Affect the Standby Database

Changes to the primary database structure always affect a standby database. In cases such as the following, the standby database is updated automatically through applied redo data:

- Taking a primary tablespace offline or bringing it online
- Changing tablespace status from read / write to read-only and the reverse
- Renaming a datafile
- Dropping a tablespace

However, you might need to perform maintenance on the standby database when you add a datafile to the primary database or create a tablespace on the primary database, or perform unlogged DML operations.

In addition, you must re-create the standby database entirely when you restore a **backup control file** on the primary database or open the primary database with the RESETLOGS option.

This section contains the following topics:

8.3.1 Adding or Dropping Tablespaces and Adding or Deleting Datafiles in the Primary Database

Adding or dropping a tablespace or adding or deleting a datafile in the primary database generates redo logs that, when applied to the standby database, automatically add or delete the datafile name or add or drop the tablespace in the

standby control file. If the standby database locates the file with the filename specified in the control file, then recovery continues. If the standby database is unable to locate a file with the filename specified in the control file, then recovery terminates.

Data Guard automatically updates tablespaces and datafiles on the standby site if you use the STANDBY_FILE_MANAGEMENT parameter with the auto option. For example:

```
STANDBY_FILE_MANAGEMENT=auto
```

> **See Also:** Section 10.1.3.1 for more information on adding datafiles and tablespaces and Section 10.1.3.3 for more information on deleting datafiles and dropping tablespaces

8.3.2 Renaming Datafiles on the Primary Database

If you rename a datafile on your primary database, the new name does not take effect at the standby database until you refresh the standby database control file. To keep the datafiles at the primary and standby databases synchronized when you rename primary database datafiles, you must update the standby database control file.

> **See Also:** Section 10.1.3.2 for a scenario that explains renaming a datafile on the primary database

8.3.3 Adding or Deleting Redo Logs on the Primary Database

You can add redo log file groups or members to the primary database without affecting the standby database. Similarly, you can drop log file groups or members from the primary database without affecting your standby database. Enabling and disabling threads at the primary database has no effect on the standby database.

Consider whether to keep the online redo log configuration the same at the primary and standby databases. Although differences in the online redo log configuration between the primary and standby databases do not affect the standby database functionality, they do affect the performance of the standby database after activation. For example, if the primary database has 10 redo logs and the standby database has 2, and you then failover to the standby database so that it functions as the new primary database, the new primary database is forced to archive more frequently than the original primary database.

See Also: Section 8.3.8 for instructions on refreshing the standby database control file

8.3.4 Resetting or Clearing Unarchived Redo Logs on the Primary Database

If you clear log files at the primary database by issuing the `ALTER DATABASE CLEAR UNARCHIVED LOGFILE` statement, or open the primary database using the `RESETLOGS` option, you invalidate the standby database. Because both of these operations reset the primary log sequence number to 1, you must re-create the standby database in order to be able to apply archived logs generated by the primary database.

See Also: Section 3.3 and Section 10.1.8 for information on re-creating the standby database

8.3.5 Altering the Primary Database Control File

If you use the `CREATE CONTROLFILE` statement at the primary database to perform any of the following operations, you may invalidate the control file for the standby database:

- Change the maximum number of redo log file groups or members
- Change the maximum number of instances that can concurrently mount and open the database

Using the `CREATE CONTROLFILE` statement with the `RESETLOGS` option on your primary database will force the next open of the primary database to reset the online logs, thereby invalidating the standby database.

If you have invalidated the control file for the standby database, re-create the file using the procedures in Section 8.3.8.

8.3.6 Taking Datafiles in the Standby Database Offline

You can take standby database datafiles offline as a means to support a subset of your primary database's datafiles. For example, to skip recovery of datafiles that were not copied to the standby database, take the missing datafiles offline using the following statement on the standby database:

```
SQL> ALTER DATABASE DATAFILE 'MISSING00004' OFFLINE DROP;
```

If you execute this statement, then you must drop the tablespace containing the offline files after opening the standby database.

8.3.7 Detecting Unlogged or Unrecoverable Operations

Caution: Unlogged or unrecoverable operations invalidate the standby database and may require substantial DBA administrative activities.

When you perform a direct load originating from any of the following, the performance improvement applies *only* to the primary database (there is no corresponding recovery process performance improvement on the standby database):

- Direct path load
- `CREATE TABLE` through subquery
- `CREATE INDEX` on the primary database

8.3.7.1 Propagating Unrecoverable Operations Manually

Primary database operations using the `UNRECOVERABLE` option are not propagated to the standby database because these operations do not appear in the archived redo logs. If you perform an unrecoverable operation at the primary database and then recover the standby database, you do not receive error messages during recovery; instead, the Oracle database server writes error messages in the standby database `alert log` file. The following error message is displayed:

```
Errors in file /oracle/rdbms/log/rcv12_ora_150.trc:
ORA-01578: ORACLE data block corrupted (file # 1, block # 36031)
ORA-01110: data file 1: '/oracle/dbs/s1_t_db1.dbf'
ORA-26040: Data block was loaded using the NOLOGGING option
```

To correct your standby database following an unrecoverable operation, you will need to perform one or more of the following tasks:

- Take the affected datafiles offline in the standby database and drop the tablespace after failover. See Section 8.3.6.
- Re-create the standby database from a new database backup. See Section 3.3.
- Back up the affected tablespace and archive the current logs in the primary database, copy the datafiles to the standby database, and resume standby recovery. This is the same procedure that you would perform to guarantee ordinary database recoverability after an unrecoverable operation.

See Also: Section 10.1.4

8.3.7.2 Determining Whether a Backup Is Required After Unrecoverable Operations

If you have performed unrecoverable operations on your primary database, determine whether a new backup is required.

To determine whether a new backup is necessary:

1. Query the V$DATAFILE view on the primary database to determine the **system change number (SCN)** or time at which the Oracle database server generated the most recent invalidation redo data.
2. Issue the following SQL statement on the primary database to determine whether you need to perform another backup:

```
SELECT UNRECOVERABLE_CHANGE#,
       TO_CHAR(UNRECOVERABLE_TIME, 'mm-dd-yyyy hh:mi:ss')
FROM   V$DATAFILE;
```

3. If the query in the previous step reports an unrecoverable time for a datafile that is more recent than the time when the datafile was last backed up, then make another backup of the datafile in question.

See Also: V$DATAFILE in Chapter 14, "Views" and the *Oracle9i Database Reference* for more information about the V$DATAFILE view

8.3.8 Refreshing the Standby Database Control File

The following steps describe how to refresh, or create a copy, of changes you have made to the primary database control file. Refresh the standby database control file after making major structural changes to the primary database, such as adding or deleting files.

Caution: Do this only if all archived logs have been applied to the standby database. Otherwise, the archived logs are lost when you create the new standby control file.

To refresh the standby database control file:

1. Start a SQL session on the standby database and issue the CANCEL statement on the standby database to halt its recovery process.

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
```

2. Shut down the standby database:

```
SQL> SHUTDOWN IMMEDIATE;
```

3. Start a SQL session on the primary database and create the control file for the standby database:

```
SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS 'stbycf.ctl';
```

4. Copy the standby control file and archived log files to the standby site using an operating system utility appropriate for binary files.
5. Connect to the standby database and mount (but do not open) the standby database:

```
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

6. Restart the recovery process on the standby database:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;
```

8.3.9 Clearing Online Redo Logs

After creating the standby database, you can clear online redo logs on the standby site to optimize performance by issuing the following statement, where 2 is the number of the log group:

```
SQL> ALTER DATABASE CLEAR LOGFILE GROUP 2;
```

This statement optimizes failover to the standby database because it is no longer necessary for the Oracle database server to clear the logs before failover. Clearing involves writing zeros to the entire contents of the redo log and then setting a new header to make the redo log look like it was when it was created.

If you clear the logs manually, the Oracle database server realizes at activation that the logs already have zeros and skips the clearing step. This optimization is important because it can take a long time to write zeros into all of the online logs. If you prefer not to perform this operation during maintenance, the Oracle database server clears the online logs automatically during failover.

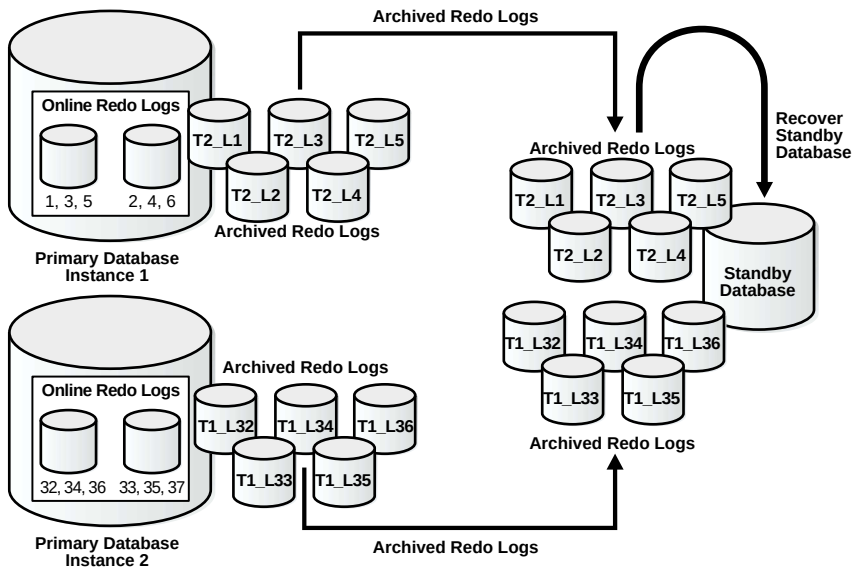
8.4 Standby Databases in an Oracle Real Application Clusters Configuration

You can configure a standby database to protect a primary database using Real Application Clusters. The following table describes the possible combinations of instances in the primary and standby databases:

Instance Combinations	Single-instance Standby Database	Multi-instance Standby Database
Single-instance Primary Database	Yes	Yes (for read-only queries)
Multi-instance Primary Database	Yes	Yes

In each scenario, each instance of the primary database archives its own online redo logs to the standby database. For example, Figure 8–1 illustrates a Real Application Clusters database with two primary database instances (a multi-instance primary database) archiving redo logs to a single-instance standby database.

Figure 8–1 Archiving Redo Logs from a Multi-instance Primary Database



In this case, Instance 1 of the primary database transmits logs 1, 2, 3, 4, 5 while Instance 2 transmits logs 32, 33, 34, 35, 36. If the standby database is in managed recovery mode, it automatically determines the correct order in which to apply the archived redo logs.

If both your primary and standby databases are in a Real Application Clusters configuration, and the standby database is in managed recovery mode, then a single instance of the standby database applies all sets of logs transmitted by the primary instances. In this case, the standby instances that are *not* applying redo cannot be in read-only mode while managed recovery is in progress; in most cases, the nonrecoverable instances should be shut down, although they can also be mounted.

> **See Also:** *Oracle9i Real Application Clusters Setup and Configuration* for information about configuring a database for Real Application Clusters

This section contains the following topic:

- Setting Up a Cross-Instance Archival Database Environment

8.4.1 Setting Up a Cross-Instance Archival Database Environment

It is possible to set up a cross-instance archival database environment. Within a Real Application Clusters configuration, each instance directs its archived redo logs to a single instance of the cluster. This instance is called the **recovery instance** and is typically the instance where managed recovery is performed. This instance typically has a tape drive available for RMAN backup and restore support. Example 8–1 shows how to set up the LOG_ARCHIVE_DEST_*n* initialization parameter for archiving redo logs across instances. Execute this example on all instances except the recovery instance.

Example 8–1 Setting Destinations for Cross-Instance Archiving

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_1 = 'LOCATION=archivelog MANDATORY REOPEN=120';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1 = enable;
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = 'SERVICE=prmy1 MANDATORY REOPEN=300';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = enable;
```

Destination 1 is the repository containing the local archived redo logs required for instance recovery. This is a mandatory destination. Because the expected cause of failure is lack of adequate disk space, the retry interval is 2 minutes. This should be adequate to allow the DBA to purge unnecessary archived redo logs. Notification of

destination failure is accomplished by manually searching the primary database alert log.

Destination 2 is the primary database on the instance where RMAN is used to back up the archived redo logs from local disk storage to tape. This is a mandatory destination, with a reconnect threshold of 5 minutes. This is the time needed to fix any network-related failures. Notification of destination failure is accomplished by manually searching the primary or standby database alert log.

Cross-instance archiving is available using the ARC*n* process only. Using the LGWR process for cross-instance archiving results in the RFS process failing and the archive log destination being placed in the Error state.

> **See Also:** Appendix D, "Standby Database Real Application Clusters Support"

9

Managing a Logical Standby Database

This chapter describes how to manage logical standby databases. This chapter contains the following topics:

- Configuring and Managing Logical Standby Databases
- Tuning Logical Standby Databases

The topics in this chapter describe how to use SQL statements, initialization parameters, views, and the DBMS_LOGSTDBY PL/SQL package to manage logical standby databases.

> **See Also:** *Oracle9i Data Guard Broker*, which describes how the Data Guard broker automates the management tasks described in this chapter through the Data Guard Manager graphical user interface and the Data Guard command-line interface

9.1 Configuring and Managing Logical Standby Databases

Chapter 6 described how you use the DBMS_LOGSTDBY PL/SQL package to control SQL apply operations to the logical standby database. The DBMS_LOGSTDBY PL/SQL package also provides procedures to help you configure and manage logical standby databases. You can use the DBMS_LOGSTDBY PL/SQL package to perform management tasks such as the following on logical standby databases:

- Controlling User Access to Tables in a Logical Standby Database
- Skipping Tables in a Logical Standby Database
- Adding Tables to a Logical Standby Database
- Finding Unsupported Database Objects
- Viewing and Controlling Logical Standby Events

- Viewing SQL Apply Operations Activity
- Determining How Much Redo Log Data Has Been Applied
- Recovering from Errors

> **See Also:** Table 6–1 and *Oracle9i Supplied PL/SQL Packages and Types Reference* for complete information about the DBMS_LOGSTDBY package

In addition, see Section 10.2 for scenarios that show how to skip a transaction, instantiate a table, and perform switchover and failover operations in a logical standby environment.

9.1.1 Controlling User Access to Tables in a Logical Standby Database

The ALTER DATABASE GUARD SQL statement controls user access to tables in logical standby databases. Until you start SQL apply operations, users can modify the logical standby database. However, once you start SQL apply operations to the logical standby database, the database guard is set to ALL by default, which prevents nonprivileged users from executing DML or DDL statements on the logical standby database.

This command allows the following keywords:

- ALL

  Specify ALL to prevent all users other than SYS from making changes to any data in the database.

- STANDBY

  Specify STANDBY to prevent all users other than SYS from making changes to any database object being maintained by logical standby databases. This setting is useful if you want report operations to be able to modify data as long as it is not being replicated by the logical standby database.

- NONE

  Specify NONE if you want typical security for all data in the database.

Users with DBMS_LOGSTDBY access may be able to perform DML transactions after executing the DBMS_LOGSTDBY.GUARD_BYPASS_ON PL/SQL procedure. For example, this GUARD_BYPASS_ON PL/SQL procedure allows users to create additional indexes and materialized views.

Also, it is possible to set the database guard to the STANDBY keyword, which allows users to submit DML transactions except for tables maintained in SQL apply mode. In other words, a user can create new tables and modify them in SQL apply mode.

Finally, you can *skip* certain operations affecting tables that are maintained by the SQL apply operation of log apply services. In this case, users can execute DML statements against the skipped objects if the database guard is set to the STANDBY keyword.

9.1.2 Skipping Tables in a Logical Standby Database

By default, all SQL statements executed on a primary database are applied to a logical standby database. If only a subset of activity on a primary database is of interest for replication, use the DBMS_LOGSTDBY.SKIP procedure to define filters that prevent the application of SQL statements on the logical standby database.

9.1.3 Adding Tables to a Logical Standby Database

Initially, you may decide to leave tables out of the logical standby database that you later want to add.

This following list and Example 9–1 show how to add a table to a logical standby database:

1. Stop log apply services to stop applying SQL statements to the database.

   Stopping log apply services on a logical standby database does not compromise the standby database. Log transport services on the primary database continue to transmit archived redo logs to the logical standby database. However, stopping log apply services affects the time it takes to fail over to the logical standby database because of the remaining log data that must be applied before failing over.

2. Remove the skip table entry using the DBMS_LOGSTDBY.UNSKIP procedure.

3. Re-create the table in the logical standby database using the DBMS_LOGSTDBY.INSTANTIATE_TABLE procedure.

4. Restart log apply services.

Example 9–1 demonstrates how to add the emp table to a logical standby database.

Example 9–1 Adding a Table to a Logical Standby Database

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
SQL> EXECUTE DBMS_LOGSTDBY.UNSKIP('table','scott','emp',null);
```

```
SQL> EXECUTE DBMS_LOGSTDBY.INSTANTIATE_TABLE('emp','scott','dbone');
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
```

> **See Also:** *Oracle9i Supplied PL/SQL Packages and Types Reference* for information about the DBMS_LOGSTDBY.UNSKIP and the DBMS_LOGSTDBY.INSTANTIATE_TABLE procedures

Prior to running reports that use this table, you should archive the current redo log on the primary database and ensure that it is applied to the logical standby database.

9.1.4 Finding Unsupported Database Objects

Logical standby databases maintain user tables, sequences, and jobs. To maintain other objects, you must reissue the DDL statements seen in the redo stream. Tables in the SYS schema are never maintained, because only Oracle metadata is maintained in the SYS schema.

To see other tables that are not being maintained, query the following views:

- DBA_LOGSTDBY_UNSUPPORTED view

  Lists all of the tables that contain datatypes not supported by logical standby databases in the current release. These tables are not maintained (will not have DML applied) by the logical standby database. Query this view on the primary database to ensure that those tables necessary for critical applications are not in this list. If the primary database includes unsupported tables that are critical, consider using a physical standby database.

- DBA_LOGSTDBY_SKIP view

  Lists the DML and DDL that is to be skipped. This view reflects the options chosen by the DBMS_LOGSTDBY.SKIP and the DBMS_LOGSTDBY.SKIP_ERROR procedures. The DBA_LOGSTDBY_SKIP view does not list tables.

  You select which DML and DDL to skip using the DBMS_LOGSTDBY.SKIP procedure. If there are schemas or tables with contents that are not important during a failover operation, you should avoid applying changes for those tables. Doing so may improve the overall performance of the system because it avoids performing unnecessary work. The same applies to sequences. You can skip DML for a sequence in the same way that you skip DML for a table.

  ```
  EXECUTE DBMS_LOGSTDBY.SKIP('dml','scott','scottseq');
  ```

Another important skip tool is DBMS_LOGSTDBY.SKIP_ERROR. Depending on how important a table is, you may want to ignore failures for a table or specific DDL. Doing so prevents the SQL apply operations from stopping. Later you can view the DBA_LOGSTDBY_EVENTS view to find and correct any problems that exist.

9.1.5 Viewing and Controlling Logical Standby Events

When you query the DBA_LOGSTDBY_EVENTS view, it displays a table of events that contains interesting activity from SQL apply operations. In particular, DDL execution or anything that generates an error is recorded in the events table. You can control what and how much activity is recorded in the events table. By default, 100 records are stored in this table, but you can increase that to 200 by specifying the following:

```
DBMS_LOGSTDBY.APPLY_SET('max_events_recorded', 200);
```

Additionally, you can indicate what type of events you want recorded. By default, everything is recorded in the table. However, you can set the RECORD_SKIP_DDL, RECORD_SKIP_ERRORS, and RECORD_APPLIED_DDL parameters to the APPLY_SET procedure to `false` to avoid recording these events.

Errors that cause SQL apply operations to stop are always recorded in the events table (unless access to the system tablespace is an issue). These events are always put into the ALTER.LOG file as well, with the phrase 'LOGSTDBY event' included in the text. When querying the view, select the columns in order by EVENT_TIME, COMMIT_SCN, and CURRENT_SCN. This ordering ensures that a shutdown failure appears last in the view.

9.1.6 Viewing SQL Apply Operations Activity

SQL apply operations for logical standby databases use a collection of parallel execution servers and background processes to perform a number of different tasks. The V$LOGSTDBY view shows what each process is currently doing; the TYPE column describes the task being performed:

- The READER process reads redo records from the archived redo logs.
- The PREPARER processes do the heavy computing required to convert the block changes into table changes.
- The BUILDER process assembles completed transactions.

- The ANALYZER process examines the records, possibly dumping transactions and performing some dependency computation.
- The APPLIER processes apply the completed transactions.

When querying the V$LOGSTDBY view, pay special attention to the HIGH_SCN column. This is an activity indicator. As long as it is changing each time you query the V$LOGSTDBY view, progress is being made. The STATUS column gives a text description of the current activity.

Another place to gain information of current activity is the V$LOGSTDBY_STATS view, which provides state and status information. All of the options for the DBMS_LOGSTDBY.APPLY_SET procedure have default values, and those values (default or set) can be seen in the V$LOGSTDBY_STATS view. In addition, a count of the number of transactions applied or transactions ready will tell you if transactions are being applied as fast as they are being read. Other statistics include information on all parts of the system.

9.1.7 Determining How Much Redo Log Data Has Been Applied

An important difference between a logical standby database and a physical standby database is that a logical standby database applies transactions while a physical standby database applies blocks of redo. This means that when a physical standby database has applied an archived redo log, that log will not be read again. However, for a logical standby database, all committed transactions from a given redo log may be applied, but there is always data in a redo log for transactions that commit in the next log. For this reason, logical standby databases use an SCN range of redo data, rather than individual archived redo logs.

The DBA_LOGSTDBY_PROGRESS view displays APPLIED_SCN, NEWEST_SCN, and READ_SCN information. The APPLIED_SCN indicates that committed transactions at or below that SCN have been applied. The NEWEST_SCN is the maximum SCN to which data could be applied if no more logs were received. This is usually the MAX(NEXT_CHANGE#)-1 from DBA_LOGSTDBY_LOG when there are no gaps in the list.

Log files with a NEXT_CHANGE# below READ_SCN are no longer needed. The information in those logs has been applied or persistently stored in the database. The time values associated with these SCN values are only estimates based on log times. They are not meant to be accurate times of when those SCN values were written on the primary database.

The DBA_LOGSTDBY_LOG view contains a list of all the log files that have been registered. If you prefer using sequence numbers, rather than SCNs, you can

combine the DBA_LOGSTDBY_LOG view and the DBA_LOGSTDBY_PROGRESS view to get that information. For example:

```
SELECT SEQUENCE# FROM DBA_LOGSTDBY_LOG l, DBA_LOGSTDBY_PROGRESS p
WHERE l.FIRST_CHANGE# <= p.APPLIED_SCN AND l.NEXT_CHANGE# > p.APPLIED_SCN;

SELECT SEQUENCE# FROM DBA_LOGSTDBY_LOG l, DBA_LOGSTDBY_PROGRESS p
WHERE l.FIRST_CHANGE# <= p.NEWEST_SCN AND l.NEXT_CHANGE# > p.NEWEST_SCN;

SELECT SEQUENCE# FROM DBA_LOGSTDBY_LOG l, DBA_LOGSTDBY_PROGRESS p
WHERE l.FIRST_CHANGE# <= p.READ_SCN AND l.NEXT_CHANGE# > p.READ_SCN;
```

9.1.8 Recovering from Errors

If the SQL apply operation fails, an error is recorded in the DBA_LOGSTDBY_EVENTS table. If the error was caused by a DDL transaction that contained a file specification that does not match in the logical standby database environment, perform the following steps to fix the problem:

1. Use the DBMS_LOGSTDBY.GUARD_BYPASS_ON procedure to bypass the database guard so you can make modifications to the logical standby database:

   ```
   SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_ON;
   ```

2. Execute the DDL command, using the correct file specification, and then reenable the database guard. For example:

   ```
   SQL> ALTER TABLESPACE t_table ADD DATAFILE 'dbs/t_db.f' SIZE 100M REUSE;
   SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_OFF;
   ```

3. Query the DBA_LOGSTDBY_EVENTS view to find the XIDUSN, XIDSLT, and XIDSQN values for the failed DDL, and provide the values to the DBMS_LOGSTDBY.SKIP_TRANSACTION.

   ```
   SQL> SELECT XIDUSN, XIDSLT, XIDSQN FROM DBA_LOGSTDBY_EVENTS
     2> WHERE EVENT_TIME = (SELECT MAX(EVENT_TIME) FROM DBA_LOGSTDBY_EVENTS);
   SQL> EXECUTE DBMS_LOGSTDBY.SKIP_TRANSACTION( /*xidusn*/, /*xidslt*/,
   /*xidsqn*/);
   ```

4. Start SQL apply operations on the logical standby database.

   ```
   SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
   ```

When the logical standby database restarts, it will attempt to reexecute the transaction, but the transaction will be skipped. Because you have compensated for

the failure by executing the DDL properly, anything else that depends on the DDL should execute without problems.

Although the SKIP_TRANSACTION procedure can be very helpful, you should be cautious when using it to filter DML failures. Not only is the DML that is seen in the events table skipped, but so is all the DML associated with the transaction. Thus, multiple tables may be damaged by such an action.

DML failures usually indicate a problem with a specific table. Assume the failure is an out-of-storage error that you cannot resolve immediately. One way to bypass the table but not the transaction is to add the table to the skip list.

```
SQL> EXECUTE DBMS_LOGSTDBY.SKIP('dml','scott','emp');
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
```

From this point on, DML activity for the scott.emp table will not be applied. After you have corrected the storage problem, you can fix the table provided you have set up a database link to the primary database that has administrator privileges to run procedures in the DBMS_LOGSTDBY package. With that database link, you can re-create the table and pull the data over to the standby database.

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
SQL> EXECUTE DBMS_LOGSTDBY.INSTANTIATE_TABLE('scott','emp','primarydb');
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
```

The local scott.emp table is dropped, then re-created using the database link. Table scott.emp will contain records as of when the INSTANTIATE_TABLE procedure was performed. That means it is possible for the scott.emp table to contain records for a department not in the scott.dept table. To remedy this, log on to the primary database and execute the following statements:

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
SQL> SELECT FIRST_CHANGE# FROM V$LOG WHERE STATUS = 'CURRENT';
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

When DBA_LOGSTDBY_PROGRESS.APPLIED_SCN is greater than the value selected from the query of the V$LOG view, the database is consistent, and you can safely run reports again.

9.2 Tuning Logical Standby Databases

Take the following actions to increase system performance:

- On the primary database, provide additional identification key information, indicating which columns uniquely define the rows for tables without primary

keys. This makes logging more efficient. You also have some control over tuning the system through your ability to affect what DDL is logged. Use the DBMS_LOGSTDBY.APPLY_SET procedure to affect SGA and process usage.

- Application programmers should provide primary key information to aid in efficient logging of the application data. Also, programmers can provide primary keys with RELY constraints.

  See Also: Section 4.1 and *Oracle9i SQL Reference* for more information about RELY constraints

- Transaction consistency

  Use the TRANSACTION_CONSISTENCY parameter of the DBMS_LOGSTDBY.APPLY_SET procedure to set the level of transaction consistency.

  Specify one of the following values:

  – FULL

    Transactions are applied to the logical standby database in the exact order in which they were committed on the primary database. This option results in the lowest performance. This is the default parameter setting.

    Reporting applications can run trouble-free in this mode, even if they use temporary tables to summarize results.

  – READ_ONLY

    Transactions are committed out of order from how they were committed on the primary database. The READ_ONLY value provides better performance than the FULL value, and SQL SELECT statements return read-consistent results. This is particularly beneficial when you are using the logical standby database to generate reports.

    Note: DML statements involving standby tables are not allowed in this mode.

  – NONE

    Transactions are committed out of order from how they were committed on the primary database, and no attempt is made to provide read-consistent results. This results in the best performance of the three values. If

applications that are reading the logical standby database make no assumptions about transaction order, this option works well.

- PARALLEL_MAX_SERVERS

  Use the PARALLEL_MAX_SERVERS initialization parameter to specify the maximum number of parallel execution processes and parallel recovery processes for an instance. The default value for this parameter is derived from the values of the CPU_COUNT, PARALLEL_AUTOMATIC_TUNING, and PARALLEL_ADAPTIVE_MULTI_USER initialization parameters. This parameter must not be set to a value less than 5 on a logical standby database.

  The MAX_SERVERS parameter of the DBMS_LOGSTDBY.APPLY_SET procedure can be used to limit the number of parallel servers used by log apply services. The default value of this parameter is set equal to the value of the PARALLEL_MAX_SERVERS parameter. If explicitly set, this parameter should not be set to a value less than 5 or greater than the value of the PARALLEL_MAX_SERVER parameter.

- SHARED_POOL_SIZE

  Use the SHARED_POOL_SIZE initialization parameter to specify the size (in bytes) of the shared pool. The shared pool contains shared cursors, stored procedures, and control structures. The shared pool is also used to hold redo data while it is processed by log apply services.

  The MAX_SGA parameter of the DBMS_LOGSTDBY.APPLY_SET procedure can be used to set the maximum amount of shared pool space used by log apply services. By default, log apply services will use up to one quarter of the shared pool. Generally speaking, increasing the size of the shared pool or the amount of shared pool space used by log apply services will improve the performance of a logical standby database.

10

Data Guard Scenarios

This chapter is divided into the following sections:

- Physical Standby Database Scenarios
- Logical Standby Database Scenarios

10.1 Physical Standby Database Scenarios

This section contains the following scenarios relating to physical standby databases:

- Scenario 1: Creating a Physical Standby Database on the Same System
- Scenario 2: Creating a Physical Standby Database on a Remote Site
- Scenario 3: Accommodating Physical Changes in the Primary Database
- Scenario 4: Recovering After the NOLOGGING Clause Is Specified
- Scenario 5: Deciding Which Standby Database to Fail Over to in a Multiple Physical Standby Database Configuration
- Scenario 6: Switching Over a Primary Database to a Standby Database
- Scenario 7: Recovering After a Network Failure
- Scenario 8: Re-creating a Physical Standby Database
- Scenario 9: Standby Database with a Time Lag
- Scenario 10: Using a Standby Database to Back Up the Primary Database

10.1.1 Scenario 1: Creating a Physical Standby Database on the Same System

This scenario describes the creation of a physical standby database `standby1` on the same system as the primary database `primary1`. This is a UNIX system with

three file systems, each mounted on a separate disk configuration on a different controller. By placing the physical standby database on a different file system from the primary database, you protect the primary database from a hard disk failure. By running the same-site standby database in managed recovery mode, you can keep it continuously up-to-date.

After you set up the physical standby database on the local site, you plan to create a physical standby database on a remote system for total disaster protection. In this way, even if all disks of the primary database fail or are destroyed in a disaster, you can fail over to the remote standby database and keep the database open.

Step 1 Plan the physical standby database.

Because the site uses three file systems, each on its own set of disks with its own controller, you decide to maintain the primary database files on the first file system, the standby database files on the second file system, and the `ORACLE_HOME` binaries on the third file system. If the primary database disks fail, you can switch to the standby database; if the `ORACLE_HOME` disks fail, you can switch to the remote standby database.

To host the physical standby database on the same system as the primary database, you must set the following parameters in the standby database initialization parameter file:

- `CONTROL_FILES`
- `LOCK_NAME_SPACE`
- `DB_FILE_NAME_CONVERT`
- `LOG_FILE_NAME_CONVERT`

Because the primary database is shut down every Sunday for an hour for maintenance, you decide to use that time to make a closed, consistent backup. You can then restart the database while you make the necessary configurations for the standby database.

Step 2 Create the physical standby database.

The next step in the procedure is to create the backup that will form the basis for the physical standby database. You know that you can use either an inconsistent or consistent backup, but because the database is shut down every Sunday for maintenance, you decide to make a consistent backup then and use it for the standby database.

1. Determine the database files.

On Sunday, before shutting down the primary database, you query the database to determine which datafiles it contains:

```
SQL> SELECT NAME FROM V$DATAFILE;
NAME
--------------------------------------------------------------------------------
/fs1/dbs/tbs_01.dbf
/fs1/dbs/tbs_02.dbf
/fs1/dbs/tbs_11.dbf
/fs1/dbs/tbs_12.dbf
/fs1/dbs/tbs_21.dbf
/fs1/dbs/tbs_22.dbf
/fs1/dbs/tbs_13.dbf
/fs1/dbs/tbs_23.dbf
/fs1/dbs/tbs_24.dbf
/fs1/dbs/tbs_31.dbf
/fs1/dbs/tbs_32.dbf
/fs1/dbs/tbs_41.dbf
/fs1/dbs/tbs_42.dbf
/fs1/dbs/tbs_51.dbf
/fs1/dbs/tbs_52.dbf
/fs1/dbs/tbs_03.dbf
/fs1/dbs/tbs_14.dbf
/fs1/dbs/tbs_25.dbf
/fs1/dbs/tbs_33.dbf
/fs1/dbs/tbs_43.dbf
/fs1/dbs/tbs_53.dbf
21 rows selected.
```

2. Back up the datafiles.

After determining which datafiles are in the primary database, you shut down the database with the `IMMEDIATE` option:

```
SQL> SHUTDOWN IMMEDIATE;
```

You can copy the datafiles from the primary file system to the physical standby file system. Because the transfer of datafiles can take a long time, you may want to copy the datafiles and then proceed to other tasks (such as network configuration). For example, enter the following at the UNIX command shell:

```
% cp /fs1/dbs/tbs* /fs2/dbs
```

After you perform some other routine maintenance operations, restart the database as follows:

```
SQL> STARTUP PFILE=PRMYinit.ora;
```

3. Ensure that the primary database is in ARCHIVELOG mode and that archiving is enabled.

   You can confirm the primary database is in ARCHIVELOG mode either by viewing the output from the SQL*Plus `ARCHIVE LOG LIST` command or by querying the `V$DATABASE` view.

4. Create the physical standby database control file.

   After a few minutes, you create the standby database control file in the same directory in which you stored the consistent backup:

   ```
   SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS '/fs2/dbs/cf1.ctl';
   ```

Step 3 Configure Oracle Net.

To run a physical standby database in a Data Guard environment, you must configure an Oracle Net connection between the primary and physical standby databases so that you can archive the redo logs to the standby service.

You use the IPC protocol to connect the primary database to the physical standby database because both databases are on the same site. Because you are using the local naming method, you must create new entries in the `tnsnames.ora` file. You must also add corresponding entries in the `listener.ora` file.

1. Configure the `tnsnames.ora` file.

   Currently, only one service name entry exists in your configuration, a TCP/IP connection to the `primary1` database.

   Using Oracle Net Manager, define an IPC connection between the primary and the standby database using `standby1` as the service name, `stdby1` as the SID, and `kstdby1` as the IPC key.

2. Configure the `listener.ora` file.

   Using Oracle Net Manager, modify the listener and add a listening address using IPC as the protocol and `kstdby1` as the IPC key. Add your standby database `stdby1` as a database service for this listener. Alternatively, you can add a new listener.

   See Also: *Oracle9i Net Services Administrator's Guide*

Step 4 Configure the primary database parameter file.

Now that you have configured the network files, you can edit the primary database initialization parameter file. The primary database is now up and running, so these changes will only be enabled if you restart the instance or issue `ALTER SESSION` or `ALTER SYSTEM` statements.

The only changes you need to make to the file involve archiving to the standby service. Currently, the primary database initialization parameter file looks as follows:

```
#
#parameter file PRMYinit.ora
#
db_name=primary1
control_files=(/fs1/dbs/cf1.ctl,/fs1/dbs/cf2.ctl)
compatible=9.0.1.0.0
log_archive_start = true
log_archive_dest_1 = 'LOCATION=/fs1/arc_dest/ MANDATORY REOPEN=60'
log_archive_dest_state_1 = ENABLE
log_archive_format = log_%t_%s.arc
audit_trail=FALSE
o7_dictionary_accessibility=false
global_names=false
db_domain=us.oracle.com
remote_login_passwordfile = exclusive

# default parameters for instance 1
processes=30
sessions=30
transactions=21
transactions_per_rollback_segment=21
db_block_buffers=1000
db_files=200
shared_pool_size=10000000
```

1. Specify standby archive destinations.

   Currently, you archive to only one location: a local directory. Because you want to maintain the local standby database in managed recovery mode, you must specify a new archiving location using a service name.

   Open the primary database initialization parameter file with a text editor and examine the current archiving location and format:

   ```
   log_archive_dest_1 = 'LOCATION=/fs1/arc_dest/ MANDATORY REOPEN=60'
   log_archive_dest_state_1 = ENABLE
   ```

```
log_archive_format = log_%t_%s.arc
```

Parameter/Option	Meaning
LOG_ARCHIVE_DEST_1	Indicates an archiving destination.
LOCATION	Indicates a local directory.
LOG_ARCHIVE_DEST_STATE_1	Indicates the state of the LOG_ARCHIVE_DEST_1 archiving destination.
ENABLE	Indicates that the Oracle database server can archive to the destination.
LOG_ARCHIVE_FORMAT	Indicates the format for filenames of archived redo logs.

2. Because you want to archive to the standby database with service `standby1`, you edit the file, adding the following entries:

```
log_archive_dest_2 = 'SERVICE=standby1 OPTIONAL REOPEN=180'
log_archive_dest_state_2 = ENABLE
```

Parameter/Option	Meaning
LOG_ARCHIVE_DEST_2	Indicates a new archiving destination. LOG_ARCHIVE_DEST_1 is already reserved for local archiving to /fs1/arc_dest/.
SERVICE	Indicates the service name of the standby database.
OPTIONAL	Indicates that the Oracle database server can reuse online redo logs even if this destination fails.
REOPEN	Indicates how many seconds the archiving process waits before reattempting to archive to a previously failed destination.
LOG_ARCHIVE_DEST_STATE_2	Indicates the state of the LOG_ARCHIVE_DEST_2 archiving destination.
ENABLE	Indicates that the Oracle database server can archive to the destination.

After editing the primary database initialization parameter file, create a copy for use by the standby database:

```
% cp /fs1/PRMYinit.ora /fs3/oracle/dbs/STBYinit.ora
```

If the primary database initialization parameter file contains the IFILE parameter, you also need to copy the file referred to by the IFILE parameter to the standby site and, if necessary, make appropriate changes to it.

Step 5 Configure the physical standby database parameter file.

You know that the initialization parameters shown in Table 10–1 play a key role in the standby database recovery process, and decide to edit them.

Table 10–1 Configuring Standby Database Initialization Parameters

Parameter	Setting
COMPATIBLE	This parameter must be the same at the primary and standby databases. Because it is already set to 9.0.1.0.0 in the primary database parameter file, you can leave the standby setting as it is.
CONTROL_FILES	This parameter must be different between the primary and standby databases. You decide to locate the control files in the /fs2/dbs directory.
DB_FILE_NAME_CONVERT	You must rename your standby datafile filenames to direct the standby database to correctly access its datafiles. A convenient way to do so is to use this parameter. If you have more than one set of files mapping from your primary datafiles to your standby datafiles, you need to specify multiple mapping pairs as values to this parameter. For this example, you decide to map primary datafiles from /fs1/dbs to /fs2/dbs, from /fs1/dbs2 to /fs2/dbs, and from /fs1/dbs3 to /fs2/dbs.
DB_FILES	DB_FILES must be the same at both databases so that you allow the same number of files at the standby database as you allow at the primary database. Consequently, you leave this parameter alone. An instance cannot mount a database unless DB_FILES is equal to or greater than MAXDATAFILES.
DB_NAME	This value should be the same as the DB_NAME value in the primary database parameter file. Consequently, you leave this parameter alone.
FAL_SERVER	This parameter specifies the net service name that the standby database should use to connect to the FAL server. You decide to set the name to primary1.
FAL_CLIENT	This parameter specifies the net service name that the FAL server should use to connect to the standby database. You decide to set the name to standby1.
LOCK_NAME_SPACE	This parameter specifies the name space that the distributed lock manager (DLM) uses to generate lock names. Set this value if the standby and primary databases share the same site. You decide to set the name to standby1.

Table 10–1 (Cont.) Configuring Standby Database Initialization Parameters

Parameter	Setting
LOG_ARCHIVE_DEST_1	This parameter specifies the location of the archived redo logs. You must use this directory when performing manual recovery. You decide to set the value to /fs2/arc_dest/.
LOG_FILE_NAME_CONVERT	You must rename your online log filenames on the standby database to allow correct access of its online log file. Even though the online log files are not used until you fail over to the standby database, it is good practice to rename your online log files. Much like the DB_FILE_NAME_CONVERT initialization parameter, you can specify more than one mapping pair as values to this parameter. For this example, you decide to map the primary online log file from /fs1/dbs to the standby online log file location /fs2/dbs.
STANDBY_ARCHIVE_DEST	The Oracle database server uses this value to create the name of the logs received from the primary site. You decide to set it to /fs2/arc_dest/.

Edit the standby database initialization parameter file as follows:

```
#
#parameter file STBYinit.ora
#
db_name = primary1                #The same as PRMYinit.ora
control_files = (/fs2/dbs/cf1.ctl)
compatible = 9.0.1.0.0
log_archive_start = true
log_archive_dest_1='LOCATION=/fs2/arc_dest/'
log_archive_dest_state_1 = ENABLE
log_archive_format = log_%t_%s.arc
standby_archive_dest = /fs2/arc_dest/
db_file_name_convert = ('/fs1/dbs','/fs2/dbs',
                        '/fs1/dbs2','/fs2/dbs',
                        '/fs1/dbs3','/fs2/dbs')
log_file_name_convert = ('/fs1/dbs','/fs2/dbs')
lock_name_space = standby1
fal_server=primary1
fal_client=standby1
audit_trail=false
o7_dictionary_accessibility=false
global_names=false
db_domain=us.oracle.com
remote_login_passwordfile = exclusive

# default parameters for instance 1
```

```
processes=30
sessions=30
transactions=21
transactions_per_rollback_segment=21
db_block_buffers=1000
db_files=200
shared_pool_size=10000000
```

Step 6 Start the standby database in preparation for managed recovery.

Now that you have configured all network and parameter files, you can enable archiving to the standby database.

1. Set the ORACLE_SID environment variable to the same value as the `service name` parameter in the `tnsnames.ora` file on the primary site and the `listener.ora` file on the standby site as follows:

   ```
   % setenv ORACLE_SID stdby1
   ```

2. Start the instance.

   First, you start the standby database instance without mounting the standby database control file, as the following example shows:

   ```
   SQL> CONNECT SYS/CHANGE_ON_INSTALL@standby1 AS SYSDBA
   SQL> STARTUP NOMOUNT PFILE=/fs3/oracle/dbs/STBYinit.ora;
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

3. Enable changes to the primary database parameter file.

   Finally, you enable the changes you made to the primary database parameter file so that the standby database can begin receiving archived redo logs:

   ```
   SQL> CONNECT SYS/CHANGE_ON_INSTALL@primary1 AS SYSDBA
   SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1
     2> OPTIONAL REOPEN=180';
   SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = ENABLE;
   ```

Step 7 Place the standby database in managed recovery mode.

You can now start managed recovery using the ALTER DATABASE RECOVER MANAGED STANDBY DATABASE statement. You decide to use the TIMEOUT option of the RECOVER statement to specify a time interval of 20 minutes that log apply services will wait until log transport services write the requested archived log entry to the directory of the standby database control file. If the requested archived log entry is not written to the standby database control file directory within the specified time interval, the recovery operation is canceled.

While connected to the standby database using SQL*Plus, place the standby database in managed recovery mode:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE TIMEOUT 20;
```

The standby database is now in managed recovery. When you start managed recovery, log apply services automatically identify and resolve any archive gaps that may exist. As the primary database archives redo logs to the standby site, the standby database automatically applies them.

10.1.2 Scenario 2: Creating a Physical Standby Database on a Remote Site

This scenario describes the creation of a physical standby database `standby1` on a remote site. The following assumptions are being made:

- You can perform a consistent backup.
- TCP/IP is used to connect to primary and standby databases.
- The physical standby database is part of a Data Guard environment.
- The primary database site name is `prmyhost`.
- The remote site name is `stbyhost`.
- `PRMYinit.ora` is the initialization parameter file for the primary database.
- `STBYinit.ora` is the initialization parameter file for the standby database.

Step 1 Back up the primary database datafiles.

Create the backup that will form the basis for the physical standby database.

1. Query the primary database to determine the datafiles. Query the `V$DATAFILE` view to obtain a list of the primary database datafiles, as the following example shows:

   ```
   SQL> SELECT NAME FROM V$DATAFILE;
   NAME
   ----------------------------------------------------------------------
   /oracle/dbs/dbf_1.dbf
   1 row selected.
   ```

2. Shut down the primary database to make a consistent backup of the datafiles:

   ```
   SQL> SHUTDOWN IMMEDIATE;
   ```

3. Copy the primary database datafiles to a temporary location (`/backup`):

```
% cp /oracle/dbs/dbf_1.dbf /backup
```

4. Reopen the primary database as follows:

```
SQL> STARTUP PFILE=PRMYinit.ora;
```

Step 2 Create the physical standby database control file.

1. Before you create the standby database control file, ensure that the primary database is in ARCHIVELOG mode and that automatic archiving is enabled. Issue the SQL*Plus `ARCHIVE LOG LIST` command, as follows:

```
SQL> ARCHIVE LOG LIST
```

   If the output from the `ARCHIVE LOG LIST` statement displays "`No Archive Mode`," perform the following steps:

   a. Shut down the primary database as follows:

```
SQL> SHUTDOWN IMMEDIATE;
```

   b. Start and mount the primary database instance without opening it:

```
SQL> STARTUP MOUNT PFILE=PRMYinit.ora;
```

   c. Set the ARCHIVELOG mode as follows:

```
SQL> ALTER DATABASE ARCHIVELOG;
```

   d. Open the primary database:

```
SQL> ALTER DATABASE OPEN;
```

2. Create the standby database control file by issuing the following statement:

```
SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS '/backup/stbycf.ctl'
```

   The standby database control file and the primary database datafiles are in the same temporary location at the primary site to make copying to the standby site easier.

Step 3 Transfer the datafiles and control file to the physical standby site.

Transfer the primary database datafiles and the standby control file from the temporary location at the primary site to the standby site, as the following example shows:

```
% rcp /backup/* stbyhost:/fs2/oracle/stdby
```

Step 4 Configure Oracle Net.

This scenario assumes that the TCP/IP network protocol is used to connect to the primary and the physical standby databases. To achieve remote archiving of redo log files, Oracle Net must be set up correctly from the primary database to the standby database. To allow the physical standby database to fetch archive gaps from the primary database and to facilitate switchover operations, Oracle Net must again be set up correctly from the standby database to the primary database.

This step involves editing the following files:

- `tnsnames.ora` file on the primary and standby sites
- `listener.ora` file on the primary and standby sites

1. On the primary system, use Oracle Net Manager to add a service called `standby1` that uses the TCP protocol, add the host name of your standby system, and add the service name parameter `stdby1`. Verify that the listener is configured to receive requests for the primary database. If it is not, add the primary database to the listener as a new database service.
2. On the standby system, use Oracle Net Manager to add a service called `primary1` with the TCP protocol that points to your primary database system and the service name parameter `prmy`. Also, add a listener called `stdby1_listener` and add the standby database `stdby1` as a database service.

> **See Also:** *Oracle9i Net Services Administrator's Guide* for detailed directions on using the Oracle Net Manager

Step 5 Start the listener on the primary and standby sites.

Start the standby listener on the standby site. For example:

```
% lsnrctl start stdby1_listener
```

Normally, your default listener is started on your primary site. Restart the default listener on the primary database to pick up the new definitions. For example:

```
% lsnrctl stop
% lsnrctl start
```

Step 6 Configure the physical standby database initialization parameter file.

To configure the initialization parameter file for the standby database:

1. Copy the primary database initialization parameter file from the primary site to the physical standby site. From the primary site, issue a command similar to the following:

```
% rcp /oracle/dbs/PRMYinit.ora stbyhost:/fs2/oracle/stdby/STBYinit.ora
```

2. Edit the standby initialization parameter file (STBYinit.ora). Edit the following parameters:

Parameter	Value
CONTROL_FILES	stbycf.ctl
STANDBY_ARCHIVE_DEST	/fs2/oracle/stdby/
LOG_ARCHIVE_DEST_1	/fs2/oracle/stdby/
LOG_ARCHIVE_FORMAT	stdby_%t_%s
DB_FILE_NAME_CONVERT	('/oracle/dbs','/fs2/oracle/stdby')
LOG_FILE_NAME_CONVERT	('/oracle/dbs','/fs2/oracle/stdby')
LOG_ARCHIVE_START	true
FAL_SERVER	standby1
FAL_CLIENT	primary1

The STBYinit.ora initialization parameter file looks as follows:

```
#
#parameter file STBYinit.ora
#

db_name=primary1                    # The same as PRMYinit.ora

control_files=/fs2/oracle/stdby/stbycf.ctl
audit_trail=false
o7_dictionary_accessibility=false
global_names=false
db_domain=us.oracle.com
commit_point_strength=1
processes=30
sessions=30
transactions=21
transactions_per_rollback_segment=21
db_block_buffers=100
shared_pool_size=4000000
ifile=/oracle/work/tkinit.ora # Verify that file exists on the standby site
                                  # and that the file specification is valid
```

```
# specific parameters for standby database
log_archive_format = stdby_%t_%s.arc
standby_archive_dest=/fs2/oracle/stdby/
log_archive_dest_1='LOCATION=/fs2/oracle/stdby/'
db_file_name_convert=('/oracle/dbs','/fs2/oracle/stdby')
log_file_name_convert=('/oracle/dbs','/fs2/oracle/stdby')
log_archive_start=true
log_archive_trace=127
fal_server=standby1
fal_client=primary1
```

Step 7 Copy the physical standby database initialization parameter file.

1. Make a copy of the `STBYinit.ora` file by issuing the following command:

   ```
   % cp STBYinit.ora Failover.ora
   ```

   Edit `Failover.ora` so if you fail over to the `stdby1` standby database, you can use the `Failover.ora` file as the initialization parameter file for the new primary database. Make sure you use appropriate values for the `LOG_ARCHIVE_DEST_`*n* parameters.

2. Edit the `tnsnames.ora` file on the standby site in case failover to the standby database occurs. See step 4 on page 10-12 for information on how to configure the `tnsnames.ora` file.

Step 8 Start the physical standby database.

Start the physical standby database to start archiving.

1. Set the `ORACLE_SID` environment variable to the same value as the `service name` parameter in the `tnsnames.ora` file on the primary site and the `listener.ora` file on the standby site as follows:

   ```
   % SETENV ORACLE_SID stdby1
   ```

2. Start SQL*Plus:

   ```
   SQL> CONNECT SYS/SYS_PASSWORD AS SYSDBA
   ```

3. Start the standby database instance without mounting the database:

   ```
   SQL> STARTUP NOMOUNT PFILE=STBYinit.ora;
   ```

4. Mount the standby database:

   ```
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

Step 9 Configure the primary initialization parameter file.

1. Specify the archive destination by adding the following entry to the `PRMYinit.ora` file:

```
LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1 MANDATORY REOPEN=60'
```

2. Enable the archive destination state by adding the following entry to the `PRMYinit.ora` file:

```
LOG_ARCHIVE_DEST_STATE_2 = ENABLE
```

3. Issue the following statements to ensure that the initialization parameters you have set in this step take effect:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=standby1
  2> MANDATORY REOPEN=60';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

Step 10 Place the standby database in managed recovery mode.

On the standby database, enable managed recovery by issuing the following SQL statement:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;
```

10.1.3 Scenario 3: Accommodating Physical Changes in the Primary Database

This scenario describes the procedures you should follow when a physical change is made in the primary database. The following topics are covered:

- Adding a Datafile to the Primary Database
- Renaming a Datafile in the Primary Database
- Deleting a Datafile or Dropping a Tablespace in the Primary Database
- Adding or Dropping Online Redo Logs
- Physical Changes That Require You to Rebuild the Standby Database

10.1.3.1 Adding a Datafile to the Primary Database

To maintain consistency when you add a datafile to the primary database, you must also add a corresponding datafile to the standby database. Otherwise, changes in the online redo logs that relate to the new datafile in the primary database will not be applied to the standby database. The steps you perform depend on how the

datafile was created on the primary database and how the initialization parameter STANDBY_FILE_MANAGEMENT is defined on the standby database.

If	Then
You create a new datafile on the primary database with the STANDBY_FILE_MANAGEMENT initialization parameter set to auto on the standby database	The new datafile is automatically created on the standby database.
You create a new datafile on the primary database with the STANDBY_FILE_MANAGEMENT initialization parameter undefined or set to manual on the standby database	You must manually copy the new datafile to the standby database and re-create the standby control file.
You copy an existing datafile from another database to primary database	You must also copy the new datafile to the standby database and re-create the standby control file.

The following sections describe the steps to enable the automatic creation of datafiles on the standby database, and the manual steps necessary if standby file management is not automatic.

Enabling the Automatic Creation of Datafiles on the Standby Database

When you create a new datafile on the primary database, the datafile is automatically created on the standby database if the initialization parameter STANDBY_FILE_MANAGEMENT is set to auto in the standby database initialization file.

The following steps describe how to set up your standby database to automate the creation of datafiles.

1. Add the following line to your standby database initialization file.

   ```
   STANDBY_FILE_MANAGEMENT=auto
   ```

2. Shut down the standby database and restart it to pick up the changes in the initialization file and then restart managed recovery.

   ```
   SQL> SHUTDOWN;
   SQL> STARTUP NOMOUNT PFILE=STBYinit.ora
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT;
   ```

Alternatively, you can dynamically set the STANDBY_FILE_MANAGEMENT initialization parameter with an ALTER SYSTEM statement to avoid shutting down and restarting the instance. For example:

```
SQL> ALTER SYSTEM SET STANDBY_FILE_MANAGEMENT=AUTO;
```

3. Add a new tablespace to the primary database.

```
SQL> CREATE TABLESPACE new_ts DATAFILE 't_db2.dbf'
  2> SIZE 1m AUTOEXTEND ON MAXSIZE UNLIMITED;
```

4. Archive the current redo log so it will get copied to the standby database.

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

5. Verify that the new datafile was added to the primary database.

```
SQL> SELECT NAME FROM V$DATAFILE;
NAME
----------------------------------------------------------------------
/oracle/dbs/t_db1.dbf
/oracle/dbs/t_db2.dbf
```

6. Verify that the new datafile was added to the standby database.

```
SQL> SELECT NAME FROM V$DATAFILE;
NAME
----------------------------------------------------------------------
/oracle/dbs/s2t_db1.dbf
/oracle/dbs/s2t_db2.dbf
```

Manually Creating New Datafiles on the Standby Database

The following steps describe how to manually create datafiles on the standby database when standby file management is manual.

1. Add a new tablespace to the primary database.

```
SQL> CREATE TABLESPACE new_ts DATAFILE 't_db2.dbf'
  2> SIZE 1m AUTOEXTEND ON MAXSIZE UNLIMITED;
```

2. Verify that the new datafile has been added to the primary database.

```
SQL> SELECT NAME FROM V$DATAFILE;
NAME

----------------------------------------------------------------------
/oracle/dbs/t_db1.dbf
```

```
/oracle/dbs/t_db2.dbf
```

3. Set the new tablespace offline, copy the new datafile to the standby database with an operating system command, and then set the new tablespace back online.

```
SQL> ALTER TABLESPACE new_ts OFFLINE;

% cp t_db2.dbf s2t_db2.dbf

SQL> ALTER TABLESPACE new_ts ONLINE;

%rcp s2t_db2.dbf standby_location
```

4. Archive the current redo log on the primary database so it will get copied to the standby database.

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

5. Verify that the datafile has been added on the standby database when the log file has been applied on the standby database.

```
SQL> SELECT NAME FROM V$DATAFILE;
NAME

----------------------------------------------------------------------
/oracle/dbs/s2t_db1.dbf
/oracle/dbs/s2t_db2.dbf
```

10.1.3.2 Renaming a Datafile in the Primary Database

When you rename one or more datafiles in the primary database, you also need to rename the corresponding datafiles in the standby database.

> **Note:** Unlike adding or deleting datafiles, the STANDBY_FILE_MANAGEMENT initialization parameter has no effect when renaming datafiles.

1. To rename the datafile at the primary site, you must take the tablespace offline:

```
SQL> ALTER TABLESPACE tbs_4 OFFLINE;
```

2. Exit SQL and execute the following command to rename the datafile on the system:

```
% mv tbs_4.dbf tbs_x.dbf
```

3. Go back into SQL and execute the following statements to rename the datafile in the database and to bring the tablespace back online:

```
SQL> ALTER TABLESPACE tbs_4 RENAME FILE 'tbs_4.dbf' TO 'tbs_x.dbf';
SQL> ALTER TABLESPACE tbs_4 ONLINE;
```

4. Rename the datafile at the standby site:

```
% mv tbs_4.dbf tbs_x.dbf
```

5. On the standby database, restart managed recovery by issuing the following statement:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM
  2> SESSION;
```

If you do not rename the corresponding file at the standby site, and then attempt to refresh the standby database control file, the standby database will attempt to use the renamed datafile, but it will not find the renamed datafile. Consequently, you will see error messages similar to the following in the alert log:

```
ORA-00283: recovery session canceled due to errors
ORA-01157: cannot identify/lock data file 4 - see DBWR trace file
ORA-01110: data file 4: '/oracle/dbs/stdby/tbs_x.dbf'
```

10.1.3.3 Deleting a Datafile or Dropping a Tablespace in the Primary Database

When you delete one or more datafiles or drop one or more tablespaces in the primary database, you also need to delete the corresponding datafiles in the standby database.

1. Drop the tablespace at the primary site:

```
SQL> DROP TABLESPACE tbs_4;
SQL> ALTER SYSTEM SWITCH LOGFILE;
% rm tbs_4.dbf
```

2. Delete the corresponding datafile on the standby site after the archived redo log has been applied to the standby database. For example:

```
% rm tbs_4.dbf
```

10.1.3.4 Adding or Dropping Online Redo Logs

One method of tuning available to the database administrator (DBA) is changing the size and number of online redo logs. Consequently, when you add or drop an online redo log at the primary site, it is important that you refresh the standby database control file.

1. Add or drop an online redo log as follows:

```
SQL> ALTER DATABASE ADD LOGFILE 'prmy3.log' SIZE 100K;
```

or

```
SQL> ALTER DATABASE DROP LOGFILE 'prmy3.log';
```

2. Repeat this same command on each standby database to add or drop the online redo log.

10.1.3.5 Physical Changes That Require You to Rebuild the Standby Database

Some physical changes you make to the primary database can invalidate the standby database. Once a standby database is invalidated, your only option is to rebuild it.

The following clauses of the ALTER DATABASE statement invalidate the standby database:

- CLEAR UNARCHIVED LOGFILE
- OPEN RESETLOGS

See Also: Section 3.3

10.1.4 Scenario 4: Recovering After the NOLOGGING Clause Is Specified

In some SQL statements, the user has the option of specifying the NOLOGGING clause, which indicates that the database operation is not logged in the redo log file. Even though the user specifies the NOLOGGING clause, a redo log record is still written to the redo log. However, when the redo log file is copied to the standby site and applied to the standby database, a portion of the datafile is unusable and marked as being unrecoverable. When you either fail over to the standby database, or open the physical standby database with the read-only option, and attempt to read the range of blocks that are marked as UNRECOVERABLE, you will see error messages similar to the following:

```
ORA-01578: ORACLE data block corrupted (file # 1, block # 2521)
ORA-01110: data file 1: '/oracle/dbs/stdby/tbs_1.dbf'
```

```
ORA-26040: Data block was loaded using the NOLOGGING option
```

> **Note:** To avoid this problem, Oracle Corporation recommends that you always specify the FORCE LOGGING clause in the CREATE DATABASE statement when creating standby databases. See the *Oracle9i Database Administrator's Guide.*

To recover after the NOLOGGING clause is specified, you need to copy the datafile that contains the unjournaled data from the primary site to the physical standby site. Perform the following steps:

1. Determine which datafiles should be copied.

   a. Issue the following query in the primary database:

```
SQL> SELECT NAME, UNRECOVERABLE_CHANGE# FROM V$DATAFILE;
NAME                                               UNRECOVERABLE
-------------------------------------------------- -------------
/oracle/dbs/tbs_1.dbf                                       5216
/oracle/dbs/tbs_2.dbf                                          0
/oracle/dbs/tbs_3.dbf                                          0
/oracle/dbs/tbs_4.dbf                                          0
4 rows selected.
```

   b. Issue the following query in the standby database:

```
SQL> SELECT NAME, UNRECOVERABLE_CHANGE# FROM V$DATAFILE;
NAME                                               UNRECOVERABLE
-------------------------------------------------- -------------
/oracle/dbs/stdby/tbs_1.dbf                                 5186
/oracle/dbs/stdby/tbs_2.dbf                                    0
/oracle/dbs/stdby/tbs_3.dbf                                    0
/oracle/dbs/stdby/tbs_4.dbf                                    0
4 rows selected.
```

   c. Compare the query results from the primary and the standby databases.

      Compare the value of the UNRECOVERABLE_CHANGE# column in both query results. If the value of the UNRECOVERABLE_CHANGE# column in the primary database is greater than the same column in the standby database, then the datafile needs to be copied from the primary site to the standby site.

In this example, the value of the UNRECOVERABLE_CHANGE# in the primary database for the `tbs_1.dbf` datafile is greater, so you need to copy the `tbs_1.dbf` datafile to the standby site.

2. On the primary site, back up the datafile that you need to copy to the standby site as follows:

```
SQL> ALTER TABLESPACE system BEGIN BACKUP;
SQL> EXIT;
% cp tbs_1.dbf /backup
SQL> ALTER TABLESPACE system END BACKUP;
```

3. Copy the datafile and the standby database control file from the primary site to the standby site as follows:

```
% rcp /backup/* stbyhost:/fs2/oracle/stdby/
```

4. On the standby database, restart managed recovery by issuing the following statement:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM
  2>SESSION;
```

You may get the following error messages (possibly in the alert log) when you try to restart managed recovery:

```
ORA-00308: cannot open archived log 'standby1'
ORA-27037: unable to obtain file status
SVR4 Error: 2: No such file or directory
Additional information: 3
ORA-01547: warning: RECOVER succeeded but OPEN RESETLOGS would get error
below
ORA-01152: file 1 was not restored from a sufficiently old backup
ORA-01110: data file 1: '/oracle/dbs/stdby/tbs_1.dbf'
```

If you get the ORA-00308 error, cancel recovery by issuing the following statement from another terminal window:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
```

These error messages are issued when one or more logs in the archive gap have not been successfully applied. If you receive these errors, manually resolve the gaps and repeat step 4.

See Also: Section B.3 for information on resolving an archive gap

10.1.5 Scenario 5: Deciding Which Standby Database to Fail Over to in a Multiple Physical Standby Database Configuration

Every standby database is associated with one and only one primary database. A single primary database can, however, support multiple standby databases. This scenario identifies the kind of information you need in order to decide which of the multiple standby databases is the best target for the failover operation.

One of the important things to consider in a multiple standby database configuration is whether the archive destination is mandatory or optional. The following table lists an advantage and disadvantage for each destination:

Destination	Advantage	Disadvantage

MANDATORY	All archived redo logs are archived to the mandatory archive destination. After you apply the archived redo logs at the standby site, you can ensure that the standby database is up-to-date. Furthermore, you can fail over the standby database as the new primary database with minimum loss of data.	In some cases, (such as network failure), the archived redo logs cannot reach the mandatory archive destination, causing the archiving of the redo log to stop. In the worst case, if all online redo logs are full and cannot be archived, the primary database instance will stop working.
		You can issue the following SQL query to determine whether the primary database stopped because it was not able to switch to an online redo log:
		`SELECT decode (count(*), 0, 'NO', 'YES') "switch_ possible"`
		`FROM V$LOG`
		`WHERE ARCHIVED='YES';`
		If the output from the query displays "Yes," a log switch is possible; if the output displays "NO," a log switch is not possible.
		See Section 10.1.7.

Destination	Advantage	Disadvantage
OPTIONAL	The primary database continues to operate normally when archival of the redo logs to the optional archive destination at the standby site is interrupted.	An archive gap may cause data loss because archive logs that are required to be applied to the standby database are unavailable. This results in the managed recovery operation terminating before all primary data has been applied to the standby database.

Consider the following recommendations in a multiple standby database configuration:

- Specify at least one remote standby database as a mandatory archive destination. Ideally, choose the standby database with the most stable network link and the most stable system configuration.
- Specify a local archive destination as mandatory. This allows the primary database to archive locally in a network failure.
- Use standby redo logs on your physical standby databases.

Suppose the primary database is located in San Francisco and supports five standby databases as follows:

Standby	Location	Type	Description
1	Local directory	Mandatory	Local copy of the archived redo logs.
2	San Francisco	Mandatory	Fail over to this standby database when there is physical damage at the primary site. This standby site is connected to the primary site by a local area network.
3	Boston	Optional	Fail over to this standby database when San Francisco is no longer available.
4	Los Angeles	Optional	This standby site receives archived redo logs, but does not apply them.

Standby	Location	Type	Description
5	San Francisco	Optional	This standby site receives archived redo logs, and applies them after an 8-hour time lag. See Section 10.1.9 for a description of this type of configuration.

Assume that something occurs in San Francisco where the primary site is located, and the primary site is damaged. You must fail over to one of the standby databases. You cannot assume that the database administrator (DBA) who set up the multiple standby database configuration is available to decide which standby database to fail over to. Therefore, it is imperative to have a disaster recovery plan at each standby site, as well as at the primary site. Each member of the disaster recovery team needs to know about the disaster recovery plan and be aware of the procedures to follow. This scenario identifies the kind of information that the person who is making the decision would need when deciding which standby database should be the target of the failover operation.

One method of conveying information to the disaster recovery team is to include a ReadMe file at each standby site. This ReadMe file is created and maintained by the DBA and should describe how to:

- Log on to the local database server as a DBA
- Log on to each system where the standby databases are located

  There may be firewalls between systems. The ReadMe file should include instructions for going through the firewalls.
- Log on to other database servers as a DBA
- Identify the most up-to-date standby database
- Perform the standby database failover operation
- Configure network settings to ensure that client applications access the new primary database instead of the original primary database

The following example shows the contents of a sample ReadMe file:

```
---------------Standby Database Disaster Recovery ReadMe File---------------

Warning:
********************************************************************************
Perform the steps in this procedure only if you are responsible for failing over
to a standby database after the primary database fails.
```

```
If you perform the steps outlined in this file unnecessarily, you may corrupt
the entire database system.
*******************************************************************************

Multiple Standby Database Configuration:

No.     Location       Type     IP Address
--- --------------- --------- --------------
 1   San Francisco   Primary   128.1.124.25
 2   San Francisco   Standby   128.1.124.157
 3   Boston          Standby   136.132.1.55
 4   Los Angeles     Standby   145.23.82.16
 5   San Francisco   Standby   128.1.135.24

You are in system No. 3, which is located in Boston.

Perform the following steps to fail over to the most up-to-date and available
standby database:

1.  Log on to the local standby database as a DBA.

    a)  Log on with the following user name and password:

                username: Standby3
                password: zkc722Khn

    b)  Invoke SQL*Plus as follows:

        % sqlplus

    c)  Connect as the DBA as follows:

        CONNECT sys/s23LsdIc AS SYSDBA

2.  Connect to as many remote systems as possible. You can connect to a maximum
    of four systems. System 4 does not have a firewall, so you can connect to it
    directly. Systems 1, 2, and 5 share the same firewall host.  You need to go
    to the firewall host first and then connect to each system.  The IP address
    for the firewall host is 128.1.1.100.  Use the following user name and
    password:
               username: Disaster
               password: 82lhsIW32

3.  Log on to as many remote systems as possible with the following user names
```

```
and passwords:

Login information:

No.     Location     IP Address    username   password
--- -------------- ------------ ---------- ----------
1   San Francisco   128.1.124.25   Oracle9i   sdd290Ec
2   San Francisco   128.1.124.157  Standby2   ei23nJHb
3                    (L o c a l)
4   Los Angeles     145.23.82.16   Standby4   23HHoe2a
5   San Francisco   128.1.135.24   Standby5   snc#$dnc
```

4. Invoke SQL*Plus on each remote system you are able to log on to as follows:

```
% sqlplus
```

5. Connect to each remote database as follows:

```
CONNECT sys/password AS SYSDBA

The DBA passwords for each location are:

No.     Location      Password
--- -------------- -----------
1   San Francisco     x2dwlsd91
2   San Francisco     a239s1DAq
3          (L o c a l)
4   Los Angeles       owKL(@as23
5   San Francisco     sad_KS13x
```

6. If you are able to log on to System 1, invoke SQL*Plus and issue the following statements:

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP PFILE=PRMYinit.ora;

Note: If you are able to execute the STARTUP statement successfully, the
      primary database has not been damaged.  Do not continue with this
      procedure.
```

7. Issue the following SQL statements on each standby database (including the one on this system) that you were able to connect to:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
SQL> SHUTDOWN IMMEDIATE;
```

```
    SQL> SELECT THREAD#, MAX(SEQUENCE#) FROM V$LOG_HISTORY GROUP BY THREAD#;

    Compare the query results of each standby database. Fail over to the
    standby database with the largest sequence number.

8.  Fail over to the standby database with the largest sequence number.

    On the standby database with the largest sequence number, invoke SQL*Plus
    and issue the following SQL statements:

    SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH;
    SQL> SHUTDOWN IMMEDIATE;
    SQL> STARTUP PFILE=Failover.ora;

------------End of Standby Database Disaster Recovery ReadMe File-------------
```

10.1.6 Scenario 6: Switching Over a Primary Database to a Standby Database

Reversing the roles of a primary and standby database is referred to as a switchover operation. The following steps outline what commands must be issued to perform a switchover operation.

For this discussion, `boston` is initially the primary database and `lonoon` is initially the standby database. During the following switchover scenario, `lonoon` will become the primary database and `boston` will become the standby database.

This scenario assumes that the primary and standby databases have been previously created and initialized.

> **See Also:** Section 7.2.2, Section 10.1.1, and Section 10.1.2

Step 1 End read or update activity on the primary and standby databases.

Exclusive database access is required by the DBA before beginning a switchover operation. Ask users to log off the primary and standby databases or query the `V$SESSION` view to identify users that are connected to the databases and close all open sessions except the SQL*Plus session from which you are going to issue the switchover command.

> **See Also:** *Oracle9i Database Administrator's Guide* for more information on managing users

Step 2 Switch the primary database over to the physical standby role.

On the primary database, `boston`, execute the following statement:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY WITH SESSION
SHUTDOWN;
```

This statement does the following:

- Closes the primary database, terminating any active sessions
- Archives any unarchived log files and applies them to the standby database, lonoon
- Adds an end-of-redo marker to the header of the last log file being archived
- Creates a backup of the current control file
- Converts the current control file into a standby control file

Step 3 Shut down and start up the former primary instance without mounting the database.

Execute the following statement on boston:

```
SQL> SHUTDOWN NORMAL;
SQL> STARTUP NOMOUNT;
```

Step 4 Mount the former primary database in the physical standby database role.

Execute the following statement on boston:

```
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

Step 5 Switch the former physical standby database over to the primary database role.

Execute the following statement on lonoon after the final logs have been received and applied:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
```

This statement does the following:

- Makes sure the last archived redo log file has been received and applied through the switchover (end-of-redo) marker. If not, Data Guard returns an error.
- Closes the database if it has been opened for read-only transactions
- Converts the standby control file to the current control file

Step 6 Shut down the database.

Execute the following statement on lonoon:

```
SQL> SHUTDOWN;
```

Step 7 Start up the database in the primary role.

Execute the following statement on lonoon:

```
SQL> STARTUP;
```

Step 8 Put the physical standby database in managed recovery mode.

Execute the following ALTER DATABASE statement on the standby database, boston, to place it in managed recovery mode. The following statement includes the DISCONNECT FROM SESSION clause, which starts a detached server process and immediately returns control to the user. (Note that this does not disconnect the current SQL session.)

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;
```

If you want to start a detached server process and immediately return control to the user, add the DISCONNECT FROM SESSION option to the ALTER DATABASE statement. Note that this does not disconnect the current SQL session.

Step 9 Start archiving logs from the primary database to the physical standby database.

Execute the following statement on the new primary database, lonoon:

```
SQL> ALTER SYSTEM ARCHIVE LOG START;
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

10.1.7 Scenario 7: Recovering After a Network Failure

When a standby database is in managed recovery mode, the standby database automatically applies archived redo logs as it receives them from the primary database. When the network goes down, automatic archival from the primary database to the standby database stops.

If the standby database is specified as an optional archive destination, then the primary database continues to operate normally.

When the network is up and running again, automatic archival of the archived redo logs from the primary database to the standby database resumes. However, if the standby database is specified as an optional archive destination, and a log switch occurred at the primary site, the standby database has an archive gap for the time

when the network was down. The archive gap is automatically detected and resolved.

See Also: Section 6.5 for information on archive gaps

If the standby database is specified as a mandatory archive destination, then the primary database will not overwrite any redo logs until the network failure is resolved and the primary database is able to archive to the standby site.

The primary database may eventually stall if the network problem is not fixed in a timely manner, because the primary database will not be able to switch to an online redo log that has not been archived. You can issue the following SQL query to determine whether the primary database stalled because it was not able to switch to an online redo log:

```
SELECT decode(COUNT(*),0,'NO','YES') "switch_possible"
FROM V$LOG
WHERE ARCHIVED='YES';
```

If the output from the query displays "Yes," a log switch is possible; if the output displays "No," a log switch is not possible.

It is important to specify a local directory at the primary site as a mandatory archive destination, so that all of the archived redo logs reside on the same system as the primary database. When the primary system is unreachable, and the primary database is part of a multiple standby database configuration, you can try to identify the archived redo logs at the other standby sites.

This scenario describes how to recover after a network failure.

Step 1 Identify the network failure.

The V$ARCHIVE_DEST view contains the network error and identifies which standby database cannot be reached. On the primary database, issue the following SQL statement for the archived log destination that experienced the network failure. For example:

```
SQL> SELECT DEST_ID, STATUS, ERROR FROM V$ARCHIVE_DEST WHERE DEST_ID = 2;

DEST_ID    STATUS
---------- ---------
ERROR
----------------------------------------------------------------
         2  ERROR
ORA-12224: TNS:no listener
```

The query results show there are errors archiving to the standby database, and the cause of the error as TNS:no listener. You should check whether the listener on the standby site is started. If the listener is stopped, then start it.

Step 2 Prevent the primary database from stalling.

If you cannot solve the network problem quickly, and if the physical standby database is specified as a mandatory destination, try to prevent the database from stalling by doing one of the following:

- Disable the mandatory archive destination:

  ```
  SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = DEFER;
  ```

  When the network problem is resolved, you can enable the archive destination again:

  ```
  SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = ENABLE;
  ```

- Change the archive destination from mandatory to optional:

  ```
  SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1
    2> OPTIONAL REOPEN=60';
  ```

  When the network problem is resolved, you can change the archive destination from optional back to mandatory:

  ```
  SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1
    2> MANDATORY REOPEN=60';
  ```

  In some cases, you may not be the person responsible for correcting the problem. You can periodically query the V$ARCHIVE_DEST view to see if the problem has been resolved.

Step 3 Archive the current redo log.

On the primary database, archive the current redo log:

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

When the network is back up again, log apply services can detect and resolve the archive gaps automatically if you place the standby database in managed recovery mode.

10.1.8 Scenario 8: Re-creating a Physical Standby Database

This scenario describes the case where you have failed over to a physical standby database and have begun using it as a normal primary database. After a period of time, you decide you want to switch back to the original primary system and make the primary database the standby database again.

Step 1 Create a physical standby database at the original primary site.

1. Copy the original standby initialization parameter file from the original standby site to the original primary site as follows:

   ```
   % rcp STBYinit.ora PRMYHOST:fallback.ora
   ```

   The `fallback.ora` file will become the standby initialization parameter file for the standby database at the original primary site.

2. On the original primary site, configure the `fallback.ora` file. You need to modify the following parameters:

Parameter	Value
CONTROL_FILES	/fs2/oracle/stdby/cf1.ctl
LOCK_NAME_SPACE	fallback
LOG_ARCHIVE_FORMAT	r_%t_%s.arc
STANDBY_ARCHIVE_DEST	/oracle/dbs/
LOG_ARCHIVE_DEST_1	'LOCATION=/oracle/dbs/'
DB_FILE_NAME_CONVERT	('/fs2/oracle/stdby','/oracle/dbs')
LOG_FILE_NAME_CONVERT	('/fs2/oracle/stdby','/oracle/dbs')

The `fallback.ora` file looks as follows:

```
#
#parameter file fallback.ora
#

db_name=primary1               #The same as PRMYinit.ora

control_files=/fs2/oracle/stdby/cf1.ctl
lock_name_space=fallback;

audit_trail=false
```

```
o7_dictionary_accessibility=false
global_names=false
db_domain=us.oracle.com
commit_point_strength=1

processes=30
sessions=30
transactions=21
transactions_per_rollback_segment=21
db_block_buffers=100
shared_pool_size=4000000
ifile=/oracle/work/tkinit.ora

# specific parameters for standby database
log_archive_format = r_%t_%s.arc
standby_archive_dest=/oracle/dbs/
log_archive_dest_1='LOCATION=/oracle/dbs/'
log_archive_dest_state_1 = ENABLE
db_file_name_convert=('/fs2/oracle/stdby','/oracle/dbs')
log_file_name_convert=('/fs2/oracle/stdby','/oracle/dbs')
log_archive_start=true
log_archive_trace=127
```

3. On the original primary site, create or modify the primary initialization parameter file.

   You need to supply appropriate values for the LOG_ARCHIVE_DEST_1 and LOG_ARCHIVE_DEST_STATE_1 initialization parameters.

   Note: If you made a copy of the standby database initialization parameter file, then you can modify the copy in this step.

4. On the original standby site, back up the primary database datafiles.
5. On the original standby site, create the standby database control file.
6. Copy the primary database datafiles and the standby database control file from the original standby site to the original primary site.
7. Archive the current online redo log and shut down the primary database to prevent further modifications as follows:

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
SQL> SHUTDOWN IMMEDIATE;
```

8. Copy the archived redo logs that were generated after the primary database datafiles were backed up from the original standby site to the original primary site, as follows:

```
% rcp /fs2/oracle/stdby/stdby_1_102.arc prmyhost:/oracle/dbs/r_1_102.arc
% rcp /fs2/oracle/stdby/stdby_1_103.arc prmyhost:/oracle/dbs/r_1_103.arc
```

Step 2 Switch over to the standby database at the original primary site.

1. On the primary database, execute the following statement:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY WITH SESSION SHUTDOWN;
```

   This statement does the following:

   - Closes the primary database, terminating any active sessions
   - Archives any unarchived log files and applies them to the standby database
   - Adds a switchover marker to the header of the last log file being archived
   - Creates a backup of the current control file
   - Converts the current control file into a standby control file

2. Shut down and start up the former primary instance without mounting the database.

   Execute the following statements on the former primary instance:

```
SQL> SHUTDOWN NORMAL;
SQL> STARTUP NOMOUNT;
```

3. Mount the former primary database in the physical standby database role.

   Execute the following statement on the primary database:

```
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

4. On the original primary site, start and mount the physical standby database:

```
SQL> CONNECT SYS/SYS_PASSWORD AS SYSDBA
SQL> STARTUP NOMOUNT PFILE=fallback.ora;
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

5. Switch over to the physical standby database:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
SQL> SHUTDOWN IMMEDIATE;
```

6. Start the primary database at the original primary site:

```
SQL> STARTUP PFILE=PRMYinit.ora;
```

7. Configure the network settings to enable client applications to access the primary database.

10.1.9 Scenario 9: Standby Database with a Time Lag

By default, in managed recovery mode, the standby database automatically applies redo logs when they arrive from the primary database. But in some cases, you may not want the logs to be applied immediately, because you want to create a time lag between the archiving of a redo log at the primary site and the application of the log at the standby site. A time lag can protect against the transfer of corrupted or erroneous data from the primary site to the standby site.

For example, suppose you run a batch job every night on the primary database. Unfortunately, you accidently ran the batch job twice and you did not realize the mistake until the batch job completed for the second time. Ideally, you need to roll back the database to the point in time before the batch job began. A primary database that has a standby database with a time lag (for example, 8 hours) could help you to recover. You could fail over the standby database with the time lag and use it as the new primary database.

To create a standby database with a time lag, use the DELAY attribute of the LOG_ARCHIVE_DEST_*n* initialization parameter in the primary database initialization parameter file. The archived redo log files are still automatically copied from the primary site to the standby site, but the log files are not immediately applied to the standby database. The log files are applied when the specified time interval has expired.

This scenario use a 4-hour time lag and covers the following topics:

- Creating a Standby Database with a Time Lag
- Managing a Standby Database with a Time Lag
- Rolling Back the Database to a Specified Time
- Bypassing the Time Lag and Switching Over to the Standby Database

Readers of this scenario are assumed to be familiar with the procedures for creating a typical standby database. The details have been omitted from the steps outlined in this scenario.

See Also: Section 10.1.2 for details about standby database setup

10.1.9.1 Creating a Standby Database with a Time Lag

To create a standby database with a time lag, modify the LOG_ARCHIVE_DEST_*n* initialization parameter on the primary database to set a delay for the standby database. For example, to specify a 4-hour delay, set the parameter as follows:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='DELAY=240';
```

The DELAY attribute indicates that the archived redo logs at the standby site are not available for recovery until the 4-hour time interval has expired. The time interval (expressed in minutes) starts when the archived redo logs are successfully transmitted to the standby site. The redo information is still sent to the standby database and written to the disk as normal.

10.1.9.2 Managing a Standby Database with a Time Lag

The runtime scenario of a standby database with a time lag is slightly different from a standby database with no time lag, because the application of the online redo logs lags behind.

As the DBA, you need to keep the following tasks in mind when managing a standby database with a time lag:

- Define the STANDBY_FILE_MANAGEMENT initialization parameter in the primary database initialization parameter file to ensure that when a datafile is created on the primary site, it is also created on the standby site. The section titled "Enabling the Automatic Creation of Datafiles on the Standby Database" on page 10-16 shows the step-by-step procedure for automatically creating a datafile in the standby database.
- When a datafile from another database is copied to the primary site, you need to copy the datafile to the standby site. The section titled "Manually Creating New Datafiles on the Standby Database" on page 10-17 explains this procedure.

10.1.9.3 Rolling Back the Database to a Specified Time

In the case stated at the beginning of this scenario, you may want to take advantage of the time lag, and get a primary database whose status is a specified time (for example, 4 hours) before the current primary database. You can fail over to the appropriate time-lagged standby database as follows:

1. Fail over the standby database by issuing the following statement:

   ```
   SQL> ALTER DATABASE ACTIVATE STANDBY DATABASE;
   ```

2. Shut down the standby database instance:

```
SQL> SHUTDOWN IMMEDIATE;
```

3. Start the new primary instance:

```
SQL> STARTUP PFILE=Failover.ora;
```

10.1.9.4 Bypassing the Time Lag and Switching Over to the Standby Database

If you do not want to take advantage of the time lag, you can switch over to the standby database as a normal standby database with no time lag as follows:

1. Switch over the primary database first. Then restart the original primary database as a physical standby database.
2. Apply all of the archived redo logs on this standby database:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE NODELAY DISCONNECT FROM SESSION
THROUGH LAST SWITCHOVER;
```

3. Switch over to the physical standby database by issuing the following statement:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY DATABASE;
```

4. Shut down the standby database instance:

```
SQL> SHUTDOWN;
```

5. Start the new primary instance:

```
SQL> STARTUP PFILE=Failover.ora;
```

10.1.10 Scenario 10: Using a Standby Database to Back Up the Primary Database

This scenario describes using a physical standby database to back up the primary database. This allows the standby site to off-load the task of backup at the primary site. The backup at the standby site can be done while the standby database is in the managed recovery mode. When the primary database needs to be restored, you can use the backup created at the standby site if you choose not to fail over to the standby database.

Step 1 Back up the standby database.

The standby database can be used to back up the datafiles and the archived redo logs. The primary control file must be backed up at the primary site because the standby database uses a different standby control file. Because the control file is

usually much smaller than the datafiles, backing up the control file at the primary site does not significantly affect the performance of the primary database.

To back up a database at the standby site:

1. At the standby site, start the Recovery Manager utility (RMAN) with the NOCATALOG option to back up the standby datafiles and log files. Assume that stby is the connect string of the standby database. For example:

```
% rman target sys/change_on_install@stby nocatalog

connected to target database: ADE3 (DBID=1417165739)
using target database controlfile instead of recovery catalog
```

2. Back up the standby datafiles. You can perform the backup while the standby database is still in managed recovery mode.

```
RMAN> run {
2> allocate channel c1 type disk;
3> backup database;
4> }

allocated channel: c1
channel c1: sid=13 devtype=DISK

Starting backup at 07-NOV-00
channel c1: starting full datafile backupset
channel c1: specifying datafile(s) in backupset
input datafile fno=00001 name=/oracle/dbs/s3t_db1.dbf
channel c1: starting piece 1 at 07-NOV-00
channel c1: finished piece 1 at 07-NOV-00
piece handle=/oracle/dbs/04c9t2ki_1_1 comment=NONE
channel c1: backup set complete, elapsed time: 00:00:35
Finished backup at 07-NOV-00

Starting Control File Autobackup at 07-NOV-00
warning - controlfile is not current, controlfile autobackup skipped
Finished Control File Autobackup at 07-NOV-00
released channel: c1
```

3. Back up the archived redo logs.

   After you back up a database, Oracle Corporation recommends that you record the current sequence number and thread number applied at the standby site. This is important because the archived logs, after this backup, may be needed to

restore the database at the primary site if the primary database loses its local archived logs.

For example, suppose there is only one thread and the sequence number at the time of the database backup is 15. After several new logs are archived to the standby site, you can back up these archived logs using the following commands:

```
RMAN> run {
2> allocate channel c1 type disk;
3> backup archivelog from logseq 15;
4> }

allocated channel: c1
channel c1: sid=9 devtype=DISK

Starting backup at 07-NOV-00
channel c1: starting archive log backupset
channel c1: specifying archive log(s) in backup set
input archive log thread=1 sequence=15 recid=15 stamp=413043150
input archive log thread=1 sequence=16 recid=16 stamp=413043168
input archive log thread=1 sequence=17 recid=17 stamp=413043433
input archive log thread=1 sequence=18 recid=18 stamp=413043442
input archive log thread=1 sequence=19 recid=19 stamp=413043450
input archive log thread=1 sequence=20 recid=20 stamp=413043454
channel c1: starting piece 1 at 07-NOV-00
channel c1: finished piece 1 at 07-NOV-00
piece handle=/oracle/dbs/05c9t3c9_1_1 comment=NONE
channel c1: backup set complete, elapsed time: 00:00:03
Finished backup at 07-NOV-00

Starting Control File Autobackup at 07-NOV-00
warning - controlfile is not current, controlfile autobackup skipped
Finished Control File Autobackup at 07-NOV-00
released channel: c1
```

4. View the **backup set** using the `list` command:

```
RMAN> list backup;

List of Backup Sets
===================

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
4       Full    104M       DISK        00:00:29     07-NOV-00
```

```
        BP Key: 4   Status: AVAILABLE   Tag:
        Piece Name: /oracle/dbs/04c9t2ki_1_1
  List of Datafiles in backup set 4
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1       Full 73158      07-NOV-00
/oracle/dbs/s3t_db1.dbf

BS Key  Device Type Elapsed Time Completion Time
------- ----------- ------------ ---------------
5       DISK        00:00:02     07-NOV-00
        BP Key: 5   Status: AVAILABLE   Tag:
        Piece Name: /oracle/dbs/05c9t3c9_1_1

  List of Archived Logs in backup set 5
  Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
  ---- ------- ---------- --------- ---------- ---------
  1    15      73139      07-NOV-00 73156      07-NOV-00
  1    16      73156      07-NOV-00 73158      07-NOV-00
  1    17      73158      07-NOV-00 73163      07-NOV-00
  1    18      73163      07-NOV-00 73165      07-NOV-00
  1    19      73165      07-NOV-00 73166      07-NOV-00
  1    20      73166      07-NOV-00 73167      07-NOV-00
```

5. You can also use the RMAN `BACKUP TABLESPACE` and `BACKUP DATAFILE` commands to back up individual tablespaces and datafiles.

Step 2 Restore the backup at the primary site.

To restore the backup at the primary site, take the following steps:

1. Set up a **recovery catalog database** on the primary site.

   See Also: *Oracle9i Recovery Manager User's Guide* for instructions on setting up a catalog database

2. Ensure that the primary database is in the mounted state.
3. Move the standby control file and the **backup pieces** to the primary site.

   Note: If the backup is on disk, the backup pieces must reside under the same directory at the primary site as they do at the standby site. This is a current restriction of RMAN. This is not an issue if the backup is on tape, which is the preferred method.

4. Resynchronize the catalog database with the standby control file.

   Assume that `prmy` is the connect string of the primary database, and `rcat` is the connect string of the recovery catalog database.

```
% rman target sys/change_on_install@prmy catalog rman/rman@rcat

connected to target database: ADE3 (DBID=1417165739)
connected to recovery catalog database

RMAN> resync catalog from controlfilecopy 'scf3.ctl';
```

5. List the backup set obtained by the catalog database after the resynchronization:

```
RMAN> list backup;

starting full resync of recovery catalog
full resync complete

List of Backup Sets
===================

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
182     Full    104M       DISK        00:00:29     07-NOV-00
        BP Key: 184   Status: AVAILABLE   Tag:
        Piece Name: /oracle/dbs/04c9t2ki_1_1
  List of Datafiles in backup set 182
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1       Full 73158      07-NOV-00 /oracle/dbs/t_db1.dbf

BS Key  Device Type Elapsed Time Completion Time
------- ----------- ------------ ---------------
183     DISK        00:00:02     07-NOV-00
        BP Key: 185   Status: AVAILABLE   Tag:
        Piece Name: /oracle/dbs/05c9t3c9_1_1

  List of Archived Logs in backup set 183
  Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
  ---- ------- ---------- --------- ---------- ---------
  1    15      73139      07-NOV-00 73156      07-NOV-00
  1    16      73156      07-NOV-00 73158      07-NOV-00
  1    17      73158      07-NOV-00 73163      07-NOV-00
  1    18      73163      07-NOV-00 73165      07-NOV-00
  1    19      73165      07-NOV-00 73166      07-NOV-00
```

```
  1    20      73166      07-NOV-00 73167      07-NOV-00
```

6. Restore the primary database:

```
RMAN> run {
2> allocate channel c1 type disk;
3> restore database;
4> }

allocated channel: c1
channel c1: sid=11 devtype=DISK

Starting restore at 07-NOV-00

channel c1: starting datafile backupset restore
channel c1: specifying datafile(s) to restore from backup set
restoring datafile 00001 to /oracle/dbs/t_db1.dbf
channel c1: restored backup piece 1
piece handle=/oracle/dbs/04c9t2ki_1_1 tag=null
params=NULL
channel c1: restore complete
Finished restore at 07-NOV-00
released channel: c1
```

7. Restore the archived log.

After restoring the database, you can try to open the database:

```
SQL> ALTER DATABASE OPEN;
alter database open
*
ERROR at line 1:
ORA-01113: file 1 needs media recovery
ORA-01110: data file 1: '/oracle/dbs/t_db1.dbf'
```

The ORA-01113 error indicates that media recovery is required on the database before you can open it. Issue the manual recovery statement, RECOVER DATABASE, as follows:

```
SQL> RECOVER DATABASE;
ORA-00279: change 73158 generated at 11/07/2000 14:12:47 needed for
thread 1
ORA-00289: suggestion : /oracle/dbs/db11_17.dbf
ORA-00280: change 73158 for thread 1 is in sequence #17

Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
```

The resulting error messages indicate that the database needs log sequence #17 for recovery. You can provide logs in several ways:

- If the primary database did not lose its local archived logs, then you can use these log files.
- You can move the standby archived logs back to the primary database. In this case, some renaming may be necessary.
- If you already backed up the archived log at the standby site, you can move the backup piece to the primary site, and restore these archived logs, as shown in the following example:

```
RMAN> run {
2> allocate channel c1 type disk;
3> restore archivelog from logseq 15;
4> }

allocated channel: c1
channel c1: sid=11 devtype=DISK

Starting restore at 07-NOV-00

channel c1: starting archive log restore to default destination
channel c1: restoring archive log
archive log thread=1 sequence=15
channel c1: restoring archive log
archive log thread=1 sequence=16
channel c1: restoring archive log
archive log thread=1 sequence=17
channel c1: restoring archive log
archive log thread=1 sequence=18
channel c1: restoring archive log
archive log thread=1 sequence=19
channel c1: restoring archive log
archive log thread=1 sequence=20
channel c1: restored backup piece 1
piece handle=/oracle/dbs/05c9t3c9_1_1 tag=null
params=NULL
channel c1: restore complete
Finished restore at 07-NOV-00
released channel: c1
```

After restoring the archived logs, you can recover the database and then open it.

10.2 Logical Standby Database Scenarios

This section presents several scenarios describing the configuration and maintenance of a logical standby database. The scenarios discussed in this section include the following:

- Scenario 1: Skipping a Transaction
- Scenario 2: Creating or Re-creating a Table
- Scenario 3: Failover Operations When In Maximum Availability Mode
- Scenario 4: Switchover Operations

The scenarios in this section assume that a primary and logical standby database have been configured according to the guidelines in Chapter 4 unless otherwise noted. Also, assume that the logical standby databases described in the scenarios have been configured to support the maximum protection mode.

Table 10–2 lists the identifier values used in the scenario examples.

Table 10–2 Identifiers for Logical Standby Database Scenarios

Identifier	Primary Database	Logical Standby Database
Location	San Francisco	Seattle
Database name	HQ	SAT
Instance name	HQ	SAT
Initialization parameter file	hq_init.ora	sat_init.ora
Control file	hq_cf1.f	sat_cf1.f
Datafile	hq_db1.f	sat_db1.f
Online redo log file 1	hq_log1.f	sat_log1.f
Online redo log file 2	hq_log2.f	sat_log2.f
Database link (client-defined)	hq_link	sat_link
Net service name (client-defined)	hq_net	sat_net
Listener	hq_listener	sat_listener

10.2.1 Scenario 1: Skipping a Transaction

As you plan your new logical standby database configuration, you must take into consideration the database objects in the primary database that cannot be supported

by the logical standby database. For example, a table that is defined with a LONG datatype is an unsupported datatype in a logical standby database.

> **See Also:** Section 4.1 for a list of unsupported datatypes, tables, and other information about database objects

Operating a logical standby database when the primary database contains unsupported objects should be considered seriously, because ignoring unsupported objects can stop log apply services on the standby database. In addition to identifying unsupported datatypes, you should identify transactions that depend on other transactions that reference unsupported database objects, and define actions to correct these dependencies. Assuming that the dependencies have been identified, you might handle these transactions by defining filters that ignore (skip) all activity to the dependent objects, or by defining filters that ignore all activity in the entire schema.

This section provides the following scenarios that define filters to accomplish these tasks:

- Section 10.2.1.1, "Skipping Activity on All Objects in a Schema"
- Section 10.2.1.2, "Skipping Activity on Specific Objects in a Schema"

These scenarios make the following assumptions:

- The MOVIES schema contains information about:
  - A movie in the INFO table
  - Video for a given movie in the VIDEO table
  - Language-specific audio for a given video in the AUDIO table.

    Unfortunately, the AUDIO table is not supported on the logical standby database because its DATA column is of type LONG RAW, which is an unsupported datatype.
- An application makes use of a stored PL/SQL procedure called GET_AVDATA that is used for the purposes of streaming movies.
- These objects have already been defined on both the HQ primary database and the SAT logical standby database. The DML transactions for unsupported objects (in this case, the AUDIO table) are automatically skipped.

10.2.1.1 Skipping Activity on All Objects in a Schema

The logical standby database in this scenario uses the GET_AVDATA procedure to off-load the media streaming workload from the HQ database. The DBA needs to filter redo data coming from the primary database because the AUDIO table in the MOVIES schema contains the LONG RAW datatype (an unsupported datatype). Thus, the following scenario describes how to skip activity occurring to the MOVIE schema. Note that any DML transaction involving the AUDIO table is automatically skipped.

To work around this datatype problem, perform the following steps on the logical standby database:

1. Stop log apply services:

   ```
   SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
   ```

2. Define a filter that skips DML transactions involving all objects in the MOVIES schema:

   ```
   SQL> EXECUTE DBMS_LOGSTDBY.SKIP('DML', 'MOVIES', '%', null);
   ```

3. Define a filter that skips all DDL transactions that create or modify objects in the MOVIES schema:

   ```
   SQL> EXECUTE DBMS_LOGSTDBY.SKIP('SCHEMA_DDL', 'MOVIES', '%', null);
   ```

4. Resume log apply services to start using these new settings:

   ```
   SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
   ```

10.2.1.2 Skipping Activity on Specific Objects in a Schema

The logical standby database in this scenario off-loads the primary database by handling queries consisting solely of movie information. Because there is no reason to maintain video data without the audio, this scenario skips all activity involving the VIDEO table. To do this, perform the following steps on the logical standby database:

1. Stop log apply services.

   ```
   SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
   ```

2. Use the SKIP procedure to define a filter that skips DML transactions involving the VIDEO table in the MOVIES schema:

   ```
   SQL> EXECUTE DBMS_LOGSTDBY.SKIP('DML', 'MOVIES', 'VIDEO', null);
   ```

3. Define a filter that skips all DDL that modifies the VIDEO table in the MOVIES schema.

```
SQL> EXECUTE DBMS_LOGSTDBY.SKIP('SCHEMA_DDL', 'MOVIES', 'VIDEO', null);
```

4. Define a filter that skips all DDL transactions that modify the AUDIO table in the MOVIES schema.

```
SQL> EXECUTE DBMS_LOGSTDBY.SKIP('SCHEMA_DDL', 'MOVIES', 'AUDIO', null);
```

5. Start log apply services to begin using these settings.

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
```

This scenario does not define a DML filter against the AUDIO table, because the DML is automatically ignored by the logical standby database. However, you must define a filter for the DDL transactions, because the schema DDL transactions are permitted by default on unsupported objects. This could introduce a problem for a logical standby database in a case where the AUDIO table is modified on the primary database such that a new column CONVERTED_DATA of type BLOB is added. The contents of the DATA column is migrated to the CONVERTED_DATA column, and the DATA column is dropped. This would qualify the AUDIO table for support on logical standby database.

However, during the conversion process, the standby database's AUDIO table was unsupported, meaning that the CONVERTED_DATA column is empty. Subsequent DML transactions on the now-supported AUDIO table will fail, because the transaction would not find data in the CONVERTED_DATA column. Defining a filter that skips all DDL transactions prevents redefinitions of the AUDIO table. To enable support for a table in a case such as this, see Section 10.2.2, "Scenario 2: Creating or Re-creating a Table"

10.2.2 Scenario 2: Creating or Re-creating a Table

To create a table on a logical standby database, you use the DBMS_LOGSTDBY.INSTANTIATE_TABLE procedure.

> **See Also:** *Oracle9i Supplied PL/SQL Packages and Types Reference* for more information about the DBMS_LOGSTDBY package.

This procedure creates or re-creates a table on a logical standby database from an existing table definition on a primary database. In addition to creating a table, the procedure also imports data from the primary table.

Typically, you use table instantiation to recover a table that has experienced an invalid DML operation, such as a NOLOGGING DML transaction, but you can also use the procedure to enable support on a table that was formerly unsupported. That is, the DBMS_LOGSTDBY.INSTANTIATE_TABLE procedure offers a way to configure or reconfigure a table for support by a logical standby database, without having to back up and restore the database.

Before you can create a table, it must meet the requirements described in Section 4.1 that explain:

- How to determine if the primary database contains datatypes or tables that are not supported by a logical standby database
- How to ensure that table rows in the primary database can be uniquely identified

The steps described in this scenario ensure that logical standby database will be able to support a table on the primary database. This scenario creates the AUDIO table in the MOVIES schema on the SAT standby database using the definitions and data as found on the HQ primary database. The scenario assumes the metadata definition for the AUDIO table remains unmodified throughout the creation process.

To create a table, perform the following steps on the logical standby database:

1. Stop log apply services:

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
```

2. Remove any existing filters on the AUDIO table, if any:

```
SQL> SELECT * FROM DBA_LOGSTDBY_SKIP;
ERROR  STATEMENT_OPT         OWNER          NAME            PROC
-------------------------------------------------------------------
N      SCHEMA_DDL            MOVIES         AUDIO

SQL> EXECUTE DBMS_LOGSTDBY.UNSKIP('SCHEMA_DDL', 'MOVIES', 'AUDIO');
```

3. Create the AUDIO table in the MOVIES schema using the database link to the HQ primary database:

```
SQL> EXECUTE DBMS_LOGSTDBY.INSTANTIATE_TABLE('AUDIO', 'MOVIES', 'HQ_NET');
```

4. Resume log apply services to start support on the table:

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
```

10.2.3 Scenario 3: Failover Operations When In Maximum Availability Mode

This scenario describes a failover operation in a configuration with the maximum availability mode. This mode is the highest grade of data protection supported by a Data Guard configuration that contains logical standby databases.

In a disaster, the biggest task is to determine which logical standby database is the best target for the failover operation. While there are many environmental factors that affect which is the best target standby database, this scenario assumes these things to be equal for the purpose of emphasizing data loss assessment. Once you identify a target standby database, initiating a failover operation is a matter of issuing a command.

> **See Also:** Section 5.7.2 for more information about the maximum availability protection mode for logical standby databases

This scenario starts out with a Data Guard configuration consisting of the HQ primary database and the SAT standby database, and adds a logical standby database named NYC. The following table provides information about the values used for the new NYC logical standby database.

Identifier	Values on the Standby Database
Location	New York City
Database name	NYC
Instance name	NYC
Initialization parameter file	nyc_init.ora
Control file	nyc_cf1.f
Datafile	nyc_db1.f
Online redo log file 1	nyc_log1.f
Online redo log file 2	nyc_log2.f
Database link (client-defined)	nyc_link
Net service name (client-defined)	nyc_net
Listener	nyc_listener

Step 1 Determine the standby site with the least amount of lost data

To do this, perform the following steps:

1. Connect to the SAT logical standby database.

```
SQL> CONNECT SYS/CHANGE_ON_INSTALL AS SYSDBA;
```

2. Determine the highest applied SCN and highest (newest) applicable SCN on the SAT database by querying the following columns in the DBA_LOGSTDBY_PROGRESS view. For example:

```
SQL> SELECT APPLIED_SCN, NEWEST_SCN FROM DBA_LOGSTDBY_PROGRESS;
APPLIED_SCN NEWEST_SCN
----------- ----------
     144059     144059
```

3. Obtain a list of the archived redo logs that have been applied or are currently pending application to the SAT database by querying the DBA_LOGSTDBY_LOG view. For example:

```
SQL> SELECT SUBSTR(FILE_NAME,1,25) FILE_NAME, SUBSTR(SEQUENCE#,1,4) "SEQ#",-
FIRST_CHANGE#, NEXT_CHANGE#, TO_CHAR(TIMESTAMP, 'HH:MM:SS') TIMESTAMP,-
DICT_BEGIN BEG, DICT_END END, SUBSTR(THREAD#,1,4) "THR#"-
FROM DBA_LOGSTDBY_LOG ORDER BY SEQUENCE#;

FILE_NAME                 SEQ# FIRST_CHANGE# NEXT_CHANGE# TIMESTAM BEG END THR#
------------------------- ---- ------------- ------------ -------- --- --- ----
/oracle/dbs/nq_sat_2.log  2           101579       101588 11:02:57 NO  NO  1
/oracle/dbs/nq_sat_3.log  3           101588       142065 11:02:01 NO  NO  1
/oracle/dbs/nq_sat_4.log  4           142065       142307 11:02:09 NO  NO  1
/oracle/dbs/nq_sat_5.log  5           142307       142739 11:02:47 YES YES 1
/oracle/dbs/nq_sat_6.log  6           142739       143973 12:02:09 NO  NO  1
/oracle/dbs/nq_sat_7.log  7           143973       144042 01:02:00 NO  NO  1
/oracle/dbs/nq_sat_8.log  8           144042       144051 01:02:00 NO  NO  1
/oracle/dbs/nq_sat_9.log  9           144051       144054 01:02:15 NO  NO  1
/oracle/dbs/nq_sat_10.log 10          144054       144057 01:02:20 NO  NO  1
/oracle/dbs/nq_sat_11.log 11          144057       144060 01:02:25 NO  NO  1
/oracle/dbs/nq_sat_13.log 13          144089       144147 01:02:40 NO  NO  1
```

Notice the gap in the sequence numbers in the SEQ# column; in the example, the gap indicates that SAT database is missing archived redo log number 12.

4. Connect to the NYC database:

```
SQL> CONNECT SYS/CHANGE_ON_INSTALL AS SYSDBA;
```

5. Determine the highest applied SCN and highest applicable SCN on the NYC database by querying the following columns in the DBA_LOGSTDBY_PROGRESS view:

```
SQL> SELECT APPLIED_SCN, NEWEST_SCN FROM DBA_LOGSTDBY_PROGRESS;

APPLIED_SCN NEWEST_SCN
----------- ----------
     143970     144146
```

6. Obtain a list of the log files that have been processed or are currently pending processing on the NYC database. For example:

```
SQL> SELECT SUBSTR(FILE_NAME,1,25) FILE_NAME, SUBSTR(SEQUENCE#,1,4) "SEQ#", -
FIRST_CHANGE#, NEXT_CHANGE#, TO_CHAR(TIMESTAMP, 'HH:MM:SS')  TIMESTAMP, -
DICT_BEGIN BEG, DICT_END END, SUBSTR(THREAD#,1,4) "THR#" -
FROM DBA_LOGSTDBY_LOG ORDER BY SEQUENCE#;

FILE_NAME                 SEQ# FIRST_CHANGE# NEXT_CHANGE# TIMESTAM BEG END THR#
------------------------- ---- ------------- ------------ -------- --- --- ----
/oracle/dbs/hq_nyc_2.log  2           101579       101588 11:02:58 NO  NO  1
/oracle/dbs/hq_nyc_3.log  3           101588       142065 11:02:02 NO  NO  1
/oracle/dbs/hq_nyc_4.log  4           142065       142307 11:02:10 NO  NO  1
/oracle/dbs/hq_nyc_5.log  5           142307       142739 11:02:48 YES YES 1
/oracle/dbs/hq_nyc_6.log  6           142739       143973 12:02:10 NO  NO  1
/oracle/dbs/hq_nyc_7.log  7           143973       144042 01:02:11 NO  NO  1
/oracle/dbs/hq_nyc_8.log  8           144042       144051 01:02:01 NO  NO  1
/oracle/dbs/hq_nyc_9.log  9           144051       144054 01:02:16 NO  NO  1
/oracle/dbs/hq_nyc_10.log 10          144054       144057 01:02:21 NO  NO  1
/oracle/dbs/hq_nyc_11.log 11          144057       144060 01:02:26 NO  NO  1
/oracle/dbs/hq_nyc_12.log 12          144060       144089 01:02:30 NO  NO  1
/oracle/dbs/hq_nyc_13.log 13          144089       144147 01:02:41 NO  NO  1
```

7. To apply archived redo logs to bring the SAT database to its most current state, including any partially archived log files, perform the following steps:

   a. Manually recover any missing archived redo logs using an operating system utility. In this case, the SAT database is missing archived redo log 12. Because the NYC received this archived redo log, you can copy it from the NYC database to the SAT database, as follows:

```
%cp /net/nyc/oracle/dbs/hq_nyc_12.log
/net/sat/oracle/dbs/hq_sat_12.log
```

   b. Determine if a partial archived redo log exists for the next sequence number. In this example, the next sequence number should be 14. The

following UNIX command shows the directory on the SAT database, looking for the presence of an archived redo log named hq_sat_14.log.

```
%ls -l /net/sat/oracle/dbs/hq_sat_14.log
-rw-rw----   1 oracle    dbs  333280 Feb 12  1:03 hq_sat_14.log
```

c. Register both the recovered archived redo log and the partial archived redo log.

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE '/oracle/dbs/hq_sat_12.log';
SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE '/oracle/dbs/hq_sat_14.log';
```

d. Start log apply services to apply to the most current log.

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
```

8. Bring the NYC database up to its most current state, including any partial archived redo logs by performing the following steps:

a. Determine if a partial archived redo log exists for the next sequence number. The following UNIX command shows the directory on the NYC database, looking for the presence of an archived redo log named with the next sequence named (hq_nyc_14). For example:

```
%ls -l /net/nyc/oracle/dbs/hq_nyc_14.log
-rw-rw----   1 oracle    dbs  333330 Feb 12  1:03 hq_nyc_14.log
```

b. Register the partial archive log file on the NYC database:

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE '/oracle/dbs/hq_nyc_14.log';
```

c. Start log apply services to apply to the most current log.

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
```

9. Determine the highest applied SCN on the SAT database. To be sure that log apply services have completed applying all archived redo logs, query the DBA_LOGSTDBY_PROGRESS view to see if the value of the APPLIED_SCN column is equal to the value of the NEWEST_SCN column:

```
SQL> SELECT APPLIED_SCN, NEWEST_SCN FROM DBA_LOGSTDBY_PROGRESS;

APPLIED_SCN NEWEST_SCN
----------- ----------
     144200     144200
```

Because the SCN values match, you can be assured that there is no longer a delay (lag) between the primary database's current log and the last log received and applied by the SAT database.

10. Determine the highest applied SCN on the NYC database by querying the DBA_LOGSTDBY_PROGRESS view to see if the value of the APPLIED_SCN column is equal to the value of the NEWEST_SCN column:

```
SQL> SELECT APPLIED_SCN, NEWEST_SCN FROM DBA_LOGSTDBY_PROGRESS;

APPLIED_SCN NEWEST_SCN
----------- ----------
     144205     144205
```

Because the SCN values match, you can be assured that there is no longer a delay (lag) between the primary database's current log and the last log received and applied by the NYC database.

In most cases, the logical standby database you choose as a failover target should be a balance between data loss and performance. As you analyze this information to make a decision about the best failover candidate in this scenario, consider the following:

- For minimal data loss during the failover operation, you should choose the NYC database as the best target standby database because steps 9 and 10 revealed that the NYC database contained more recoverable redo.
- For minimal performance effect during the failover operation, you should choose the SAT database as the best target standby database. This database is a more appropriate candidate because the queries prior to steps 3, 4, 5, and 6 reveal that the SAT database had applied 5 archived redo logs more than the NYC database (even though there was only a 1-second delay (lag) in the receipt of archived redo logs by the NYC database).

Step 2 Perform the failover operation.

Once you determine which standby database is going to be the failover target, initiate a failover operation by connecting to the standby database and issuing the following statements:

1. Stop log apply services:

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
```

2. Issue the following statement to fail over to the logical standby database that will become the new primary database:

```
SQL> ALTER DATABASE ACTIVATE LOGICAL STANDBY DATABASE;
```

10.2.4 Scenario 4: Switchover Operations

This scenario presents the requirements and procedures to ensure a successful switch over of roles between a primary database and a logical standby database. During a switchover operation, Data Guard takes measures to prevent data divergence during the role transition.

The scenario includes information about setting up initialization parameter files prior to initiating the switchover operation. In some cases, a switchover operation requires that you reconfigure the original primary database as a standby database to the new primary database. However, it is possible to eliminate this requirement in configurations with multiple standby databases without violating the database protection mode.

> **See Also:** Section 5.7 for more information about data protection modes

Performing a switchover operation without this requirement is the most likely scenario, because this allows a former primary database to undergo routine maintenance operation without downtime or data protection penalties.

Step 1 Modify the initialization parameter files.

1. Modify the SAT database's initialization parameter file, `sat_init.ora`, to ensure that the SAT database can function as a primary database, as follows:

   - Configure an additional archive destination using the Oracle Net service name for the primary database, which specifies the same database protection mode that is currently specified for the primary database.
   - Defer the new destination.

   The following example shows the `LOG_ARCHIVE_DEST_`*n* and `LOG_ARCHIVE_DEST_STATE_`*n* initialization parameters in the `sat_init.ora` parameter file:

```
LOG_ARCHIVE_DEST_1 = 'LOCATION=SAT MANDATORY REOPEN=5'
LOG_ARCHIVE_DEST_2 = 'SERVICE=HQ_NET LGWR SYNC AFFIRM MANDATORY NOREGISTER'
LOG_ARCHIVE_DEST_STATE_1 = ENABLE
LOG_ARCHIVE_DEST_STATE_2 = DEFER
```

```
LOG_ARCHIVE_FORMAT = "_SAT_%s"
STANDBY_ARCHIVE_DEST = 'HQ'
```

2. Modify the initialization parameter file, `hq_init.ora`, for the HQ database. to ensure that the HQ database can function as a logical standby database. Modify this file:

   - To ensure the file names for locally archived log files and received archived log files do not collide if they reside in the same directory
   - To tune the initialization parameters appropriately for log apply services

   The following example shows the relevant parameters in the `hq_init.ora` initialization parameter file:

```
LOG_ARCHIVE_DEST_1 = 'LOCATION=HQ MANDATORY REOPEN=5'
LOG_ARCHIVE_DEST_2 = 'SERVICE=SAT_NET LGWR SYNC AFFIRM MANDATORY NOREGISTER'
LOG_ARCHIVE_DEST_STATE_1 = ENABLE
LOG_ARCHIVE_DEST_STATE_2 = ENABLE
LOG_ARCHIVE_FORMAT = "_HQ_%s"
STANDBY_ARCHIVE_DEST = 'SAT'
LOG_PARALLELISM = 1
PARALLEL_MAX_SERVERS = 9
SHARED_POOL_SIZE = 167772160
```

Step 2 Switch over the HQ (primary) database to the logical standby role.

To switch over from the HQ database to the SAT database, perform the following steps:

1. Verify the database is currently a primary database by querying the V$DATABASE view. For example:

```
SQL> SELECT NAME, DATABASE_ROLE FROM V$DATABASE;
NAME      DATABASE_ROLE
--------- ----------------
HQ        PRIMARY
```

2. Prepare the primary database to become a logical standby database:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY;
```

3. Disable log transport services from transmitting archived redo logs to the SAT logical standby database:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = DEFER;
```

4. Set any desired initialization parameters for the logical standby database. To view the current settings on the SAT database, query the database link using the following SQL statements:

```
SQL> SELECT NAME, SUBSTR(VALUE,1,30) "Value" -
FROM SYSTEM.LOGSTDBY$PARAMETERS@SAT_LINK ORDER BY NAME;

NAME                          Value
----------------------------- -----------------------------
FIRST_SCN                     101579
LMNR_SID                      1
PRIMARY                       1457871294
TRANSACTION_CONSISTENCY       FULL
_SYNCPOINT_INTERVAL           0
```

5. To have the HQ database operate with transaction consistency, set it using the following PL/SQL procedure.

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('TRANSACTION_CONSISTENCY', 'FULL');
```

Step 3 Switch over the SAT (standby) database to the primary role.

1. Wait for a COMPLETED_SESSION record to display in the DBA_LOGSTDBY_PARAMETERS table. This indicates that the last of the archived redo logs from the primary database have been applied. For example:

```
SQL> SELECT NAME, SUBSTR(VALUE,1,30) "VALUE" FROM -
   > DBA_LOGSTDBY_PARAMETERS WHERE NAME='COMPLETED_SESSION';
NAME                           VALUE
------------------------------ ------------------------------
COMPLETED_SESSION              SWITCHOVER
```

2. Verify the SAT database is a logical standby database by querying the V$DATABASE view, as follows:

```
SQL> SELECT NAME, DATABASE_ROLE FROM V$DATABASE;
NAME      DATABASE_ROLE
--------- ----------------
SAT       LOGICAL STANDBY
```

3. Enable archiving to the new standby destination, which is the HQ database:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = ENABLE;
```

4. Prepare the SAT database to become the new primary database using the following SQL statement.

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL PRIMARY;
```

5. Verify the SAT database has changed to the role of a primary database by querying the V$DATABASE view:

```
SQL> SELECT NAME, DATABASE_ROLE FROM V$DATABASE;
NAME      DATABASE_ROLE
--------- ----------------
SAT       PRIMARY
```

Step 4 Start log apply services on the HQ database

On the HQ (logical standby) database, start log apply services on the new standby database using a database link to the new primary database, as follows:

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY NEW PRIMARY SAT_LINK;
```

Part II

Reference

This part provides reference material to be used in conjunction with the Oracle Data Guard standby database features. For more complete reference material, refer to the Oracle9*i* documentation set.

This part contains the following chapters:

11

Initialization Parameters

This chapter describes Oracle initialization parameters for Oracle instances, including the primary database and each standby database in a Data Guard configuration.

All initialization parameters are either contained in an initialization file, the name of which is a variation of `init.ora` depending on your operating system, or a server parameter file (SPFILE). As an alternative to specifying parameters in the initialization file or if you are using an SPFILE, you can modify dynamic parameters at runtime using the `ALTER SYSTEM SET` or `ALTER SESSION SET` statements.

> **See Also:** *Oracle9i Database Reference* and your Oracle operating system-specific documentation for more information about setting initialization parameters

Table 11–1 shows the initialization parameters that affect instances in a Data Guard environment. Some parameters are directly related to Data Guard, while others have additional implications when the instance is part of a Data Guard environment.

Each parameter description indicates whether the parameter applies to the primary database role, the standby database role, or both. For parameters that apply to the standby database role, most of the parameters pertain to both physical and logical standby databases. Any differences are noted in the table descriptions.

Note: You must use a server parameter file (SPFILE) if you use the Data Guard broker.

Table 11–1 Initialization Parameters Specific to the Oracle9i Data Guard Environment

Parameter	Description	For More Information
ARCHIVE_LAG_TARGET	Applies to the primary database role. Limits the amount of data that can be lost and effectively increases the availability of the standby database by forcing a log switch after the amount of time you specify (in seconds) elapses. That way, the standby database will not miss redo logs generated from a time range longer than a value of the ARCHIVE_LAG_TARGET parameter.	See *Oracle9i Database Reference*
COMPATIBLE	Applies to the primary database and standby database roles. Set to 9.0.0.0.0 or higher to use the Data Guard broker, logical standby databases, and the enhanced features of physical standby databases. Always set this parameter to the same value on the primary database and standby databases. If the values differ, you may not be able to archive the redo logs from your primary database to the standby database.	See *Oracle9i Database Reference*
CONTROL_FILE_RECORD_KEEP_TIME	Applies to the primary database and standby database roles. Prevents reusing archived redo log control file information for a specified period of time.	See Section 5.5.1.3
CONTROL_FILES	Applies to the primary database and standby database roles. Uniquely names the database control file. Always set this parameter on the standby database to a different value from the CONTROL_FILES parameter for the primary database if on the same system. The filenames you specify with the CONTROL_FILES parameter for the standby database must exist at the standby site.	See *Oracle9i Database Reference*
DB_FILE_NAME_CONVERT	Applies to the physical standby database role. Set this parameter to distinguish standby datafile filenames from primary datafile filenames. Because the standby database control file is a copy of the primary database control file, you must convert the standby database filenames if the standby database is on the same system as the primary database or on a separate system using different path names. This parameter must be set for all physical standby databases that share the same site as the primary database or use different path names.	See Section 6.3.4

Table 11–1 (Cont.) Initialization Parameters Specific to the Oracle9i Data Guard Environment

Parameter	Description	For More Information
DB_FILES	Applies to the primary database and standby database roles. Specifies the maximum number of database files that can be open for this database.	See *Oracle9i Database Reference*
DB_NAME	Applies to the primary database and standby database roles. Specifies a database identifier of up to eight characters. For physical standby databases, set it to the same value in the standby and primary database initialization files. For logical standby databases, set it using the DBNEWID (nid) utility, as specified in Section 4.2.	See *Oracle9i Database Reference*
FAL_CLIENT	Applies to the physical standby database in managed recovery mode. Assigns the FAL (fetch archive log) client name used by the FAL server to refer to the FAL client. This is the Oracle Net service name that the FAL server should use to refer to the standby database. This parameter is set on the standby site. This parameter is not used with logical standby databases.	See Section 6.5
FAL_SERVER	Applies to the physical standby database in managed recovery mode. Assigns the Oracle Net service name that the standby database should use to connect to the FAL server. This parameter is set on the standby site. This parameter is not used with logical standby databases.	See Section 6.5

Table 11–1 (Cont.) Initialization Parameters Specific to the Oracle9i Data Guard Environment

Parameter	Description	For More Information
LOCK_NAME_SPACE	Applies to the primary database and standby database roles. Specifies the name space that the distributed lock manager (DLM) uses to generate lock names. Set this parameter if the standby database has the same name as the primary database and is on the same system or cluster. **Note:** When multiple (primary or standby) databases with the same database name (DBNAME) are configured on the same system, you must set this initialization parameter to a unique value on each database.	See *Oracle9i Database Reference*
LOG_ARCHIVE_DEST_*n* [1]	Applies to the primary database and standby database roles. Defines a destination and attributes for the log transport services.	See Section 5.3.1.1
LOG_ARCHIVE_DEST_STATE_*n* = {ENABLE | DEFER | ALTERNATE | RESET}	Applies to the primary database and standby database roles. Specifies the state of the destination specified by the LOG_ARCHIVE_DEST_*n* parameter. Possible attributes are: ■ ENABLE specifies that a valid log archive destination can be used for a subsequent archiving operation (automatic or manual). This is the default. ■ DEFER specifies that valid destination information and attributes are preserved, but the destination is excluded from archiving operations until you reenable archiving with the ENABLE option. ■ ALTERNATE specifies that the destination is not enabled, but will become enabled if communication to another destination fails. ■ RESET functions the same as DEFER, but clears any error messages for the destination if it had previously failed.	See Section 5.3.1.2

Table 11–1 (Cont.) Initialization Parameters Specific to the Oracle9i Data Guard Environment

Parameter	Description	For More Information
LOG_ARCHIVE_FORMAT	Applies to the primary database and standby database roles. Indicates the format for log filenames. LOG_ARCHIVE_FORMAT and STANDBY_ARCHIVE_DEST are used to generate standby database log filenames.	See Section 5.5.2.7 and Section 5.8.4.5
LOG_ARCHIVE_MAX_PROCESSES	Applies to the primary database and standby database roles. Specifies the number of archiver processes to be invoked. This value is evaluated at instance startup if the LOG_ARCHIVE_START parameter has the value true; otherwise, this parameter is evaluated when the archiver process is invoked.	See *Oracle9i Database Reference*
LOG_ARCHIVE_MIN_SUCCEED_DEST	Applies to the primary database and standby database roles. Defines the minimum number of destinations that must succeed in order for the online log file on the primary database to be available for reuse.	See Section 5.5.2.4
LOG_ARCHIVE_START	Applies to the primary database and standby database roles. Enables automatic archiving of filled groups when LOG_ARCHIVE_START is set to true. To disable the automatic archiving of filled online redo log groups, set the LOG_ARCHIVE_START initialization parameter of a database's initialization parameter file to false.	See *Oracle9i Database Administrator's Guide*
LOG_ARCHIVE_TRACE	Applies to the primary database and standby database roles. Optionally, set this parameter to an integer value to see the progression of the archiving of redo logs to the standby site. The Oracle database server writes an audit trail of the redo logs received from the primary database into a trace file. This parameter controls output generated by the ARC*n*, LGWR, and foreground processes on the primary database, and the RFS and FAL server processes on the standby database.	See Section 6.4.8

Table 11–1 (Cont.) Initialization Parameters Specific to the Oracle9i Data Guard Environment

Parameter	Description	For More Information
LOG_FILE_NAME_CONVERT	Applies to the physical standby database role. Distinguishes the redo log filenames on the standby database from primary database redo log filenames. The parameter value converts the filename of a new log file on the primary database to the filename of a log file on the standby database. Adding a log file to the primary database necessitates adding a corresponding file to the standby database. When the standby database is updated, this parameter is used to convert the log filename from the primary database to the log filename on the standby database. This parameter is necessary if the standby database is on the same system as the primary database or on a separate system that uses different path names.	See Section 6.3.4
LOG_PARALLELISM	Applies to the logical standby database role only. Specifies the level of concurrency for redo allocation. Set this value to 1, which is the default.	See *Oracle9i Database Reference*
PARALLEL_MAX_SERVERS	Applies to the primary database and logical standby database roles. Log apply services use parallel query processes to perform processing, and use parallel apply algorithms to maintain a high level of database apply performance. A minimum of 5 parallel query processes is required for a logical standby database. Thus, the value of the PARALLEL_MAX_SERVERS parameter must be 5 or greater. This parameter specifies the maximum number of parallel servers working on log apply services on the logical standby database. This parameter is not used with physical standby databases.	See *Oracle9i Database Reference*

Table 11–1 (Cont.) Initialization Parameters Specific to the Oracle9i Data Guard Environment

Parameter	Description	For More Information
REMOTE_ARCHIVE_ENABLE	Applies to the primary database and standby database roles. Set this parameter to `true` on the primary database and standby databases in the Data Guard environment to allow the primary database to send redo logs to the standby database and to allow the standby database to receive redo logs for archiving from the primary database. Set this parameter to `false` to disable both the sending and receiving of redo logs. To independently enable and disable the sending and receiving of remote archived redo logs, use the `send` and `receive` values. `Send` enables the primary database to send redo logs to the standby database. `Receive` enables the standby database to receive redo logs from the primary database. The `send` and `receive` values together are the same as specifying `true`. Every instance of a multi-instance database must contain the same REMOTE_ARCHIVE_ENABLE value.	See Section 5.5.2
SHARED_POOL_SIZE	Set on the primary database and logical standby databases. Log apply services of logical standby databases use shared pool system global area (SGA) to stage the information read from the redo logs. The more SGA that is available, the more information that can be staged. By default, one quarter of the value set for the SHARED_POOL_SIZE parameter will be used by log apply services. This can be changed using the DBMS_LOGSTDBY.APPLY_SET PL/SQL procedure.	See *Oracle9i Database Reference* and *Oracle9i Supplied PL/SQL Packages and Types Reference*
SORT_AREA_SIZE	Applies to the primary database and standby database roles. Set this parameter to execute the `SELECT * FROM V$PARAMETER` statement when the database is open. This prevents errors when you attempt to sort without temporary tablespaces when the database is not open.	See Section 6.3.6.3

Table 11–1 (Cont.) Initialization Parameters Specific to the Oracle9i Data Guard Environment

Parameter	Description	For More Information
STANDBY_ARCHIVE_DEST	Applies to the standby database role. Used by a standby database to determine the archive location of online redo logs received from the primary database. The RFS process uses this value in conjunction with the LOG_ARCHIVE_FORMAT value to generate the fully qualified standby database log filenames. LOG_ARCHIVE_FORMAT and STANDBY_ARCHIVE_DEST are used to generate standby database log filenames. Note that the generated filename is overridden by the TEMPLATE attribute of the LOG_ARCHIVE_DEST_*n* parameter.	See Section 5.5.2.7 and Section 5.8.4.5
STANDBY_FILE_MANAGEMENT	Applies to the primary database and standby database roles. When set to auto, this parameter automates the creation and deletion of datafile filenames on the standby site using the same filenames as the primary site. When set to manual, datafile creation and deletion will not be automated and may cause managed recovery to terminate. Use this parameter with the DB_FILE_NAME_CONVERT initialization parameter (if necessary) to automate the process of creating files with identical filenames on the standby database and primary database.	See Section 6.3.4
USER_DUMP_DEST	Applies to the primary database and standby database roles. Determines the location of trace files for a database.	See *Oracle9i Database Reference* and Section 6.4.8

[1] The standby database assumes that the archived log file group is in the location specified by either the LOG_ARCHIVE_DEST or the LOG_ARCHIVE_DEST_*n* parameter in the standby initialization parameter file; both parameters cannot be specified.

12

LOG_ARCHIVE_DEST_n Parameter Attributes

This chapter provides syntax, values, and information on validity for the archival attributes of the LOG_ARCHIVE_DEST_*n* initialization parameter. These attributes include:

- AFFIRM and NOAFFIRM
- ALTERNATE and NOALTERNATE
- ARCH and LGWR
- DELAY and NODELAY
- DEPENDENCY and NODEPENDENCY
- LOCATION and SERVICE
- MANDATORY and OPTIONAL
- MAX_FAILURE and NOMAX_FAILURE
- NET_TIMEOUT and NONET_TIMEOUT
- QUOTA_SIZE and NOQUOTA_SIZE
- QUOTA_USED and NOQUOTA_USED
- REGISTER and NOREGISTER
- REGISTER=location_format
- REOPEN and NOREOPEN
- SYNC and ASYNC
- TEMPLATE and NOTEMPLATE

In addition, this chapter includes the following topics:

Note: Each destination *must* identify either a local disk directory or a remotely accessed database. See Chapter 5 for additional information about log transport services and using these initialization parameters. See Appendix E for information about using these initialization parameters to set up cascading standby databases.

12.1 About LOG_ARCHIVE_DEST_*n* Parameter Attributes

You should specify at least two LOG_ARCHIVE_DEST_*n* (where *n* is an integer from 1 to 10) parameters; one for the required local destination and another for a local or remote destination.

The following sections describe the attributes of the LOG_ARCHIVE_DEST_n initialization parameter. See Chapter 5 for additional information.

All LOG_ARCHIVE_DEST_*n* parameters must contain, at a minimum, either a LOCATION or SERVICE attribute. In addition, you must have a LOG_ARCHIVE_DEST_STATE_*n* parameter for each defined destination.

The LOG_ARCHIVE_DEST_STATE_*n* (where *n* is an integer from 1 to 10) initialization parameter specifies the state of the corresponding destination indicated by the LOG_ARCHIVE_DEST_*n* initialization parameter. For example, the LOG_ARCHIVE_DEST_STATE_3 parameter specifies the state of the LOG_ARCHIVE_DEST_3 destination.

AFFIRM and NOAFFIRM

Category	AFFIRM	NOAFFIRM
Datatype of the attribute	Keyword	Keyword
Minimum value	Not applicable	Not applicable
Maximum value	Not applicable	Not applicable
Default value	Not applicable	Not applicable
Requires attributes ...	Not applicable	Not applicable
Conflicts with attributes ...	NOAFFIRM	AFFIRM
Attribute class	ALTER SESSION and ALTER SYSTEM	ALTER SESSION and ALTER SYSTEM
Corresponding V$ARCHIVE_DEST column	AFFIRM	AFFIRM
Related V$ARCHIVE_DEST column	ASYNC_BLOCKS	ASYNC_BLOCKS

Purpose

Use the AFFIRM and the NOAFFIRM attributes to ensure that archived redo log contents have been successfully written to disk. This applies to both local and remote destinations.

Defaults

If the AFFIRM or the NOAFFIRM attribute is not specified with the LOG_ARCHIVE_DEST_n parameter, the default is NOAFFIRM.

Attributes

AFFIRM

The AFFIRM attribute indicates that all archived redo log I/O operations are to be performed synchronously even on a remote standby database. This attribute has the potential to affect primary database performance. When you use the LGWR and AFFIRM attributes to indicate that the log writer process synchronously write the locally archived redo log contents to disk, control is not returned to the users and does not continue until the disk I/O operation has completed. When you use the

ARCH and AFFIRM attributes to indicate that the ARC*n* process will synchronously write the archived redo logs to disk, the archival operation may take longer, and online redo logs may not be reusable until archiving is complete.

Using the AFFIRM attribute does not affect performance when you use the ASYNC attribute.

Query the AFFIRM column of the V$ARCHIVE_DEST fixed view to see whether or not the AFFIRM attribute is being used for the associated destination.

The following table identifies the various combinations of these attributes and their potential for affecting primary database performance and data availability. For example, the AFFIRM attribute used in conjunction with the SYNC and ASYNC attributes provides the highest degree of data protection but results in negative performance on the primary database.

Network I/O Attribute	Archived Redo Log Disk I/O Attribute	Potential Primary Database Performance	Standby Database Data Protection
SYNC	AFFIRM	Lowest	Highest
SYNC	NOAFFIRM	Low	High
ASYNC	AFFIRM	High	Low
ASYNC	NOAFFIRM	Highest	Lowest

The highest degree of data availability also has the potential for the lowest primary database performance.

Note: When the primary database is in the MAXIMIZE PROTECTION or MAXIMIZE AVAILABILITY mode, redo log archiving destinations using the log writer process are also automatically placed in AFFIRM mode.

See Also: SYNC and ASYNC on page 12-43

NOAFFIRM

The NOAFFIRM attribute indicates that all archived redo log disk I/O operations are to be performed asynchronously; the log writer process does not wait until the disk I/O has completed before continuing.

Examples

The following example shows the AFFIRM attribute with the LOG_ARCHIVE_DEST_*n* parameter.

```
LOG_ARCHIVE_DEST_3='SERVICE=stby1 LGWR SYNC AFFIRM'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

ALTERNATE and NOALTERNATE

Category	ALTERNATE=*destination*	NOALTERNATE
Datatype of the attribute	String value	Keyword
Minimum value	Not applicable	Not applicable
Maximum value	Not applicable	Not applicable
Default value	None[1]	Not applicable
Requires attributes ...	Not applicable	Not applicable
Conflicts with attributes ...	NOALTERNATE	ALTERNATE
Attribute class	ALTER SYSTEM	ALTER SESSION and ALTER SYSTEM
Corresponding V$ARCHIVE_DEST column	ALTERNATE	ALTERNATE
Related V$ARCHIVE_DEST column	STATUS	STATUS

[1] If the NOALTERNATE attribute is specified, or if no alternate destination is specified, the destination does not automatically change to another destination upon failure.

Purpose

The ALTERNATE and the NOALTERNATE attributes of the LOG_ARCHIVE_DEST_*n* parameter define an alternate archiving destination or prevent archiving to an alternate destination when the original archiving destination fails.

Defaults

If the ALTERNATE or the NOALTERNATE attribute is not specified with the LOG_ARCHIVE_DEST_*n* parameter, the default is NOALTERNATE.

Attributes

ALTERNATE=*destination*

Use the ALTERNATE attribute of the LOG_ARCHIVE_DEST_*n* parameter to define an alternate archiving destination to be used if archiving to the original archiving destination fails.

An archiving destination can have a maximum of one alternate destination specified. An alternate destination is used when the transmission of an online redo log from the primary site to the standby site fails. If archiving fails and the REOPEN attribute is specified with a value of zero (0), or NOREOPEN is specified, the Oracle database server attempts to archive online redo logs to the alternate destination on the next archival operation.

An alternate destination can reference a local or remote archiving destination. An alternate destination cannot be self-referencing.

A destination can also be in the ALTERNATE state; this state is specified using the LOG_ARCHIVE_DEST_STATE_*n* initialization parameter. The ALTERNATE state defers processing of the destination until such time as another destination failure automatically enables this destination, if the alternate destination attributes are valid. See Section 5.3.1.2 for more details about the LOG_ARCHIVE_DEST_STATE_*n* parameter.

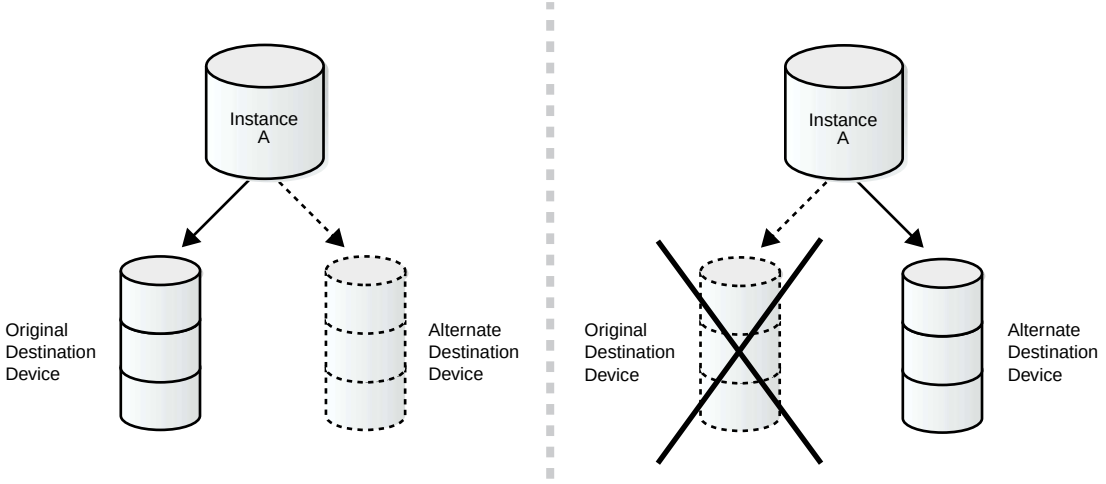
The ALTERNATE attribute cannot be modified at the session level.

In the following example, if the LOG_ARCHIVE_DEST_1 destination fails, the archiving process automatically switches to the LOG_ARCHIVE_DEST_2 destination.

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1 MANDATORY ALTERNATE=LOG_ARCHIVE_DEST_2'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_2='LOCATION=/disk2 MANDATORY'
LOG_ARCHIVE_DEST_STATE_2=ALTERNATE
```

Figure 12–1 shows a scenario where online redo log files are archived to a local disk device. If the original destination device becomes full or unavailable, the archival operation is automatically redirected to the alternate destination device.

Figure 12–1 Archiving Operation to an Alternate Destination Device



The REOPEN attribute takes precedence over the ALTERNATE attribute. The alternate destination is used only if one of the following is true:

- The NOREOPEN attribute is specified.
- A value of zero (0) is specified for the REOPEN attribute.
- A nonzero REOPEN attribute and a nonzero MAX_FAILURE count have been exceeded.

The ALTERNATE attribute takes precedence over the MANDATORY attribute. This means that a destination fails over to a valid alternate destination even if the current destination is mandatory.

The following is the archived redo log destination attribute precedence table:

Precedence[1]	Attribute
1	MAX_FAILURE
2	REOPEN
3	ALTERNATE
4	MANDATORY

[1] 1 indicates highest; 4 indicates lowest.

The use of a standby database as the target of an alternate destination should be carefully handled. Ideally, a standby alternate destination should only be used to specify a different network route to the same standby database system.

If no enabled destination references the alternate destination, the alternate destination is implied to be deferred, because there is no automatic method of enabling the alternate destination.

An alternate destination can be manually enabled at runtime. Conversely, an alternate destination can be manually deferred at runtime. See Section 5.3.2.2 for more information about changing initialization parameter settings using SQL at runtime.

There is no general pool of alternate archived redo log destinations. Ideally, for any enabled destination, the database administrator should choose an alternate destination that closely mirrors that of the referencing destination, although that is not required.

Each enabled destination can have its own alternate destination. Conversely, several enabled destinations can share the same alternate destination. This is known as an overlapping set of destinations. Enabling the alternate destination determines the set to which the destination belongs.

Increasing the number of enabled destinations decreases the number of available alternate redo log archiving destinations.

Note: An alternate destination is enabled for the next archival operation. There is no support for enabling the alternate destination in the middle of the archival operation, because that would require rereading already processed blocks, and so forth. This is identical to the `REOPEN` attribute behavior.

Any destination can be designated as an alternate given the following restrictions:

- At least one local mandatory destination is enabled.
- The number of enabled destinations must meet the defined `LOG_ARCHIVE_MIN_SUCCEED_DEST` parameter value.
- A destination cannot be its own alternate; however, this does not generate an error.

Destinations defined using the SQL ALTER SESSION statement do not activate an alternate destination defined at the system level. Conversely, system-defined destinations do not activate an alternate destination defined at the session level.

If the REOPEN attribute is specified with a nonzero value, the ALTERNATE attribute is ignored. If the MAX_FAILURE attribute is also specified with a nonzero value, and the failure count exceeds the specified failure threshold, the ALTERNATE destination is enabled. Therefore, the ALTERNATE attribute does not conflict with a nonzero REOPEN attribute value.

NOALTERNATE

Use the NOALTERNATE attribute of the LOG_ARCHIVE_DEST_*n* parameter to prevent the original destination from automatically changing to an alternate destination when the original destination fails.

Examples

In the sample initialization parameter file in Example 12–1, LOG_ARCHIVE_DEST_1 automatically fails over to LOG_ARCHIVE_DEST_2 on the next archival operation if an error occurs or the device becomes full.

Example 12–1 Automatically Failing Over to an Alternate Destination

```
LOG_ARCHIVE_DEST_1=
'LOCATION=/disk1 MANDATORY NOREOPEN ALTERNATE=LOG_ARCHIVE_DEST_2'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_2='LOCATION=/disk2 MANDATORY'
LOG_ARCHIVE_DEST_STATE_2=ALTERNATE
```

The sample initialization parameter file in Example 12–2 shows how to define an alternate Oracle Net service name to the same standby database.

Example 12–2 Defining an Alternate Oracle Net Service Name to the Same Standby Database

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1 MANDATORY'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_2='SERVICE=stby1_path1 NOREOPEN OPTIONAL ALTERNATE=LOG_ARCHIVE_DEST_3'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
LOG_ARCHIVE_DEST_3='SERVICE=stby1_path2 NOREOPEN OPTIONAL'
LOG_ARCHIVE_DEST_STATE_3=ALTERNATE
```

ARCH and LGWR

Category	ARCH	LGWR
Datatype of the attribute	Keyword	Keyword
Minimum value	Not applicable	Not applicable
Maximum value	Not applicable	Not applicable
Default value	Not applicable	Not applicable
Requires attributes ...	Not applicable	Not applicable
Conflicts with attributes ...	LGWR, ASYNC, NET_TIMEOUT	ARCH
Attribute class	ALTER SESSION and ALTER SYSTEM	ALTER SYSTEM only
Corresponding V$ARCHIVE_DEST column	ARCHIVER	ARCHIVER
Related V$ARCHIVE_DEST columns	PROCESS, SCHEDULE	PROCESS, SCHEDULE

Purpose

The optional ARCH and LGWR attributes are used to specify that either the archiver or log writer process is responsible for transmitting online redo logs to local and remote archival destinations.

When you change the archiving process from the ARC*n* process to the LGWR process using the ARCH and LGWR attributes for an archive destination, the LGWR process does not start archiving until the next log switch operation. Conversely, when you change the archiving process from the LGWR process to the ARC*n* process, the LGWR process continues to archive until the next log switch operation.

Defaults

If the ARCH or the LGWR attribute is not specified with the LOG_ARCHIVE_DEST_n parameter, the default is ARCH.

Attributes

ARCH

The ARCH attribute indicates that redo logs are transmitted to the destination during an archival operation. The background archiver processes (ARC*n*) or a foreground archival operation serves as the redo log transport service.

LGWR

The LGWR attribute indicates that redo log files are transmitted to the destination concurrently as the online redo log is populated. The background log writer process (LGWR) serves as the redo log transport service. When transmitting redo logs to remote destinations, the LGWR process establishes a network connection to the destination instance. Because the redo log files are transmitted concurrently, they are not retransmitted to the corresponding destination during the archival operation. If a LGWR destination fails, the destination automatically reverts to using the archiver (ARCn) process until the error is corrected.

Examples

The following example shows the LGWR attribute with the LOG_ARCHIVE_DEST_*n* parameter.

```
LOG_ARCHIVE_DEST_3='SERVICE=stby1 LGWR'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

DELAY and NODELAY

Category	DELAY[=*minutes*]	NODELAY
Datatype of the attribute	Numeric	Keyword
Minimum value	0 minutes	Not applicable
Maximum value	Unlimited	Not applicable
Default value	30 minutes	Not applicable
Requires attributes ...	SERVICE	Not applicable
Conflicts with attributes ...	LOCATION, NODELAY	DELAY
Attribute class	ALTER SESSION and ALTER SYSTEM	ALTER SESSION and ALTER SYSTEM
Corresponding V$ARCHIVE_DEST column	DELAY_MINS	DELAY_MINS
Related V$ARCHIVE_DEST column	DESTINATION	DESTINATION

Purpose

When the standby database is in managed recovery mode, redo logs are automatically applied when they arrive from the primary database. However, to protect against the transfer of corrupted or erroneous data from the primary site to the standby site, you may want to create a time lag between archiving a redo log at the primary site and applying that archived redo log at the standby site.

Defaults

In managed recovery mode, if the DELAY or the NODELAY attribute is not specified with the LOG_ARCHIVE_DEST_n parameter, the default is NODELAY.

If the DELAY attribute is specified without a time interval, the default time interval is 30 minutes.

Attributes

DELAY[=*minutes*]
Use the DELAY attribute of the LOG_ARCHIVE_DEST_*n* initialization parameter to specify a time lag for the application of redo logs at the standby site. The DELAY attribute does not affect the transmittal of redo logs to the standby site.

Note: Changes to the DELAY attribute take effect on the next archival operation. In-progress archival operations are not affected.

The DELAY attribute indicates that the archived redo logs at the standby site are not available for recovery until the specified time interval has expired. The time interval is expressed in minutes, and it starts when the redo log is successfully transmitted and archived at the standby site.

You can use the DELAY attribute to set up a configuration where multiple standby databases are maintained in varying degrees of synchronization with the primary database. For example, assume primary database A supports standby databases B, C, and D. Standby database B is set up as the disaster recovery database and therefore has no time lag. Standby database C is set up to protect against logical or physical corruption, and is maintained with a 2-hour delay. Standby database D is maintained with a 4-hour delay and protects against further corruption.

You can override the specified delay interval at the standby site. To immediately apply an archived redo log to the standby database before the time interval has expired, use the NODELAY keyword of the RECOVER MANAGED STANDBY DATABASE clause; for example:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE NODELAY;
```

NODELAY
When you specify the NODELAY attribute and the standby database is in managed recovery mode, redo logs are automatically applied when they arrive from the primary database.

See Also: Chapter 13, "SQL Statements"

Examples

The following example shows the DELAY attribute with the LOG_ARCHIVE_DEST_*n* parameter.

```
LOG_ARCHIVE_DEST_3='SERVICE=stby1 DELAY=240'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

DEPENDENCY and NODEPENDENCY

Category	DEPENDENCY=*destination*	NODEPENDENCY
Datatype of the attribute	String value	Keyword
Minimum value	Not applicable	Not applicable
Maximum value	Not applicable	Not applicable
Default value	Not applicable	Not applicable
Requires attributes ...	SERVICE, REGISTER	Not applicable
Conflicts with attributes ...	NODEPENDENCY, LOCATION, NOREGISTER, QUOTA_SIZE, QUOTA_USED	DEPENDENCY
Attribute class	ALTER SYSTEM only	ALTER SYSTEM only
Corresponding V$ARCHIVE_DEST column	DEPENDENCY	DEPENDENCY
Related V$ARCHIVE_DEST columns	Not applicable	Not applicable

Purpose

Archiving redo logs to a remote database can be defined as being dependent upon the success or failure of an archival operation to another destination. The dependent destination is known as the child destination. The destination on which the child depends is known as the parent destination. Specify the DEPENDENCY attribute with the LOG_ARCHIVE_DEST_*n* parameter to define a local destination, physical standby database, or logical standby database.

Defaults

If the DEPENDENCY or the NODEPENDENCY attribute is not specified with the LOG_ARCHIVE_DEST_n parameter, the default is NODEPENDENCY.

Attributes

DEPENDENCY=*destination*

Specifying a destination dependency can be useful in the following configurations:

- The standby database and the primary database are on the same node. Therefore, the archived redo logs are implicitly accessible to the standby database.
- Clustered file systems provide remote standby databases with access to the primary database archived redo logs.
- Operating system-specific network file systems provide remote standby databases with access to the primary database archived redo logs.
- Mirrored disk technology provides transparent networking support across geographically remote distances.
- Multiple standby databases are on the same remote site, sharing access to common archived redo logs.

In these situations, although a physical archival operation does not occur for the dependent destination, the standby database needs to know the location of the archived redo logs. This allows the standby database to access the archived redo logs when they become available for managed recovery. The DBA must specify a destination as being dependent on the success or failure of a parent destination.

Consider the case of a two-node cluster where a primary node shares access to the destination with the standby node through a mirrored disk device. This configuration, where you maintain a local standby database, is useful for off-loading ad hoc queries and reporting functions.

The primary database archives a redo log locally and, upon successful completion, the archived redo log is immediately available to the standby database for managed recovery. This does not require a physical remote archival operation for the standby destination. In this case, two destinations are used: one for local archiving, and another for archiving at the standby site. The standby destination is not valid unless the primary destination succeeds. Therefore, the standby destination has a dependency upon the success or failure of the local destination.

The `DEPENDENCY` attribute has the following restrictions:

- Only standby destinations can have a dependency.
- The parent destination can be either a local or standby destination.
- The `DEPENDENCY` attribute cannot be modified at the session level.
- The `REGISTER` attribute is required.
- The `SERVICE` attribute is required.
- The alternate destination cannot be self-referencing.

When one or more destinations are dependent upon the same parent destination, all attributes of the dependent destinations still apply to that destination. It appears as if the archival operation were performed for each destination, when only one archival operation actually occurred.

Consider, for example, that two standby databases are dependent upon the archived redo log of a parent destination. You can specify different DELAY attributes for each destination, which allows you to maintain a staggered time lag between the primary database and each standby database.

Similarly, a dependent destination can specify an alternate destination, which itself may or may not be dependent upon the same parent destination.

Note: Dependent destinations do not participate in a standby no-data-loss environment.

NODEPENDENCY
Specifies that there is no dependency on the success or failure of an archival operation to another destination.

Examples

One reason to use the DEPENDENCY attribute is if the standby database is on the same site as the primary database. Using this configuration, you only need to archive the redo logs once and, because the standby database resides on the local system, it can access the same redo logs. The following is an example of the LOG_ARCHIVE_DEST_*n* parameters in this scenario:

```
# Set up the mandatory local destination:
#
LOG_ARCHIVE_DEST_1='LOCATION=/oracle/dbs/ MANDATORY'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
#
# Set up the dependent standby database that resides on the local system:
#
LOG_ARCHIVE_DEST_2='SERVICE=dest2 DEPENDENCY=LOG_ARCHIVE_DEST_1 OPTIONAL'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

Another reason to use the DEPENDENCY attribute is if two standby databases reside on the same system. The parent and child standby databases can be any mix of physical and logical standby databases. The following is an example of this scenario:

```
# Set up the mandatory local destination:
#
LOG_ARCHIVE_DEST_1='LOCATION=/oracle/dbs/ MANDATORY'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
#
# Set up the remote standby database that will receive the logs:
#
LOG_ARCHIVE_DEST_2='SERVICE=dest2 OPTIONAL'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
#
# Set up the remote standby database that resides on the same system as, and is
# dependent on, the first standby database:
#
LOG_ARCHIVE_DEST_3='SERVICE=dest3 DEPENDENCY=LOG_ARCHIVE_DEST_2 OPTIONAL'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

LOCATION and SERVICE

Category	LOCATION=*local_disk_directory*	SERVICE=*net_service_name*
Datatype of the attribute	String value	String value
Minimum value	Not applicable	Not applicable
Maximum value	Not applicable	Not applicable
Default value	Not applicable	Not applicable
Requires attributes...	Not applicable	Not applicable
Conflicts with attributes ...	SERVICE, DELAY, DEPENDENCY, REGISTER=*location_format*, NOREGISTER, ASYNC, TEMPLATE, NET_TIMEOUT	LOCATION, QUOTA_USED, NOQUOTA_USED
Attribute class	ALTER SESSION and ALTER SYSTEM	ALTER SESSION and ALTER SYSTEM
Corresponding V$ARCHIVE_DEST column	DESTINATION	DESTINATION
Related V$ARCHIVE_DEST column	TARGET	TARGET

Purpose

Each destination *must* either identify a local disk directory or a remotely accessed database.

You use either the LOCATION or the SERVICE attribute to specify a destination where the log transport services archive redo log files. For each Data Guard configuration, you must identify at least one local disk directory (LOCATION=*local_disk_directory*) where redo logs are archived. You can specify up to nine additional local or remote destinations. You identify remote destinations by specifying a valid Oracle Net service name (SERVICE=*net_service_name*).

Note: If you are specifying multiple attributes, specify either the LOCATION or SERVICE attribute on the first line of the initialization parameter file.

Use remote archived redo logs to maintain a transactionally consistent copy of the primary database. Do not use locally archived redo logs to maintain a transactionally consistent standby database. However, when you configure log transport services, you must specify at least one local destination. This ensures that the locally archived redo logs are accessible should managed recovery of the primary database be necessary.

To verify the current settings, query the V$ARCHIVE_DEST fixed view:

- The TARGET column of the V$ARCHIVE_DEST fixed view identifies whether the destination is local or remote to the primary database.
- The DESTINATION column of the V$ARCHIVE_DEST fixed view identifies the values that have been specified for a destination. For example, the destination parameter value specifies the Oracle Net service name identifying the remote Oracle instance where the archived log files are located.

Defaults

One of these attributes must be specified. There is no default.

Note: When changing incremental parameters, there are no default attribute values assumed. For example, LOG_ARCHIVE_DEST_1='SERVICE=stby1 LGWR' assumes the SYNC=PARALLEL option. LOG_ARCHIVE_DEST_1='ARCH' fails because SYNC=PARALLEL conflicts with ARCH even though the default for ARCH is SYNC=NOPARALLEL.

Attributes

LOCATION=*local_disk_directory*

If you use the LOCATION attribute, specify a valid path name for a disk directory on the system that hosts the primary database. Each destination that specifies the LOCATION attribute must identify a unique directory path name. This is the local destination for archiving redo logs.

Local destinations indicate that the archived redo logs are to reside within the file system that is accessible to the primary database. Locally archived redo logs remain physically within the primary database namespace. The destination parameter value specifies the local file system directory path to where the archived redo logs are copied.

SERVICE=*net_service_name*

If you specify the SERVICE attribute, specify a valid Oracle Net service name.

Archiving redo logs to a remote destination requires a network connection and an Oracle database instance associated with the remote destination to receive the incoming archived redo logs.

The destination parameter value specifies the Oracle Net service name identifying the remote Oracle instance to which the archived log files are copied.

The *net_service_name* is translated into a connection descriptor that contains the information necessary for connecting to the remote database.

> **See Also:** *Oracle9i Net Services Administrator's Guide* for details about setting up Oracle Net service names

Examples

The following example shows the LOCATION attribute with the LOG_ARCHIVE_DEST_*n* parameter:

```
LOG_ARCHIVE_DEST_2='LOCATION=/arc_dest'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

The following example shows the SERVICE attribute with the LOG_ARCHIVE_DEST_*n* parameter:

```
LOG_ARCHIVE_DEST_3='SERVICE=stby1'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

MANDATORY and OPTIONAL

Category	MANDATORY	OPTIONAL
Datatype of the attribute	Keyword	Keyword
Minimum value	Not applicable	Not applicable
Maximum value	Not applicable	Not applicable
Default value	Not applicable	Not applicable
Requires attributes ...	Not applicable	Not applicable
Conflicts with attributes ...	OPTIONAL	MANDATORY
Attribute class	ALTER SESSION and ALTER SYSTEM	ALTER SESSION and ALTER SYSTEM
Corresponding V$ARCHIVE_DEST column	BINDING	BINDING
Related V$ARCHIVE_DEST columns	Not applicable	Not applicable

Purpose

You can specify a policy for reuse of online redo logs using the attributes OPTIONAL or MANDATORY with the LOG_ARCHIVE_DEST_*n* parameter. The archival operation of an optional destination can fail, and the online redo logs are overwritten. If the archival operation of a mandatory destination fails, online redo logs are not overwritten.

The LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* parameter (where *n* is an integer from 1 to 10) specifies the number of destinations that must archive successfully before the log writer process can overwrite the online redo logs. All mandatory destinations and non-standby optional destinations contribute to satisfying the LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* count. For example, you can set the parameter as follows:

```
# Database must archive to at least two locations before
# overwriting the online redo logs.
LOG_ARCHIVE_MIN_SUCCEED_DEST = 2
```

When determining how to set your parameters, note that:

- This attribute does not affect the destination protection mode.
- You must have at least one local destination, which you can declare OPTIONAL or MANDATORY.

  At least one local destination is operationally treated as mandatory, because the minimum value for the LOG_ARCHIVE_MIN_SUCCEED_DEST parameter is 1.
- The failure of any mandatory destination, including a mandatory standby destination, makes the LOG_ARCHIVE_MIN_SUCCEED_DEST parameter irrelevant.
- The LOG_ARCHIVE_MIN_SUCCEED_DEST value cannot be greater than the number of destinations, nor greater than the number of mandatory destinations plus the number of optional local destinations.
- If you defer a mandatory destination, and the online log is overwritten without transferring the redo log to the standby site, then you must transfer the redo log to the standby site manually.

The BINDING column of the V$ARCHIVE_DEST fixed view specifies how failure affects the archival operation.

Defaults

If the MANDATORY or the OPTIONAL attribute is not specified with the LOG_ARCHIVE_DEST_n parameter, the default is OPTIONAL.

At least, one destination must succeed even if all destinations are designated to be optional.

Attributes

MANDATORY
Specifies that archiving to the destination must succeed before the redo log file can be made available for reuse.

OPTIONAL
Specifies that successful archiving to the destination is not required before the redo log file can be made available for reuse. If the "must succeed count" set with the LOG_ARCHIVE_MIN_SUCCEED_DEST parameter is met, the redo log file is marked for reuse.

Examples

The following example shows the MANDATORY attribute with the LOG_ARCHIVE_DEST_*n* parameter.

```
LOG_ARCHIVE_DEST_1='LOCATION=/arc/dest MANDATORY'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_3='SERVICE=stby1 MANDATORY'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

MAX_FAILURE and NOMAX_FAILURE

Category	MAX_FAILURE=*count*	NOMAX_FAILURE
Datatype of the attribute	Numeric	Keyword
Minimum value	0	Not applicable
Maximum value	None	Not applicable
Default value	None	Not applicable
Requires attributes ...	REOPEN	Not applicable
Conflicts with attributes ...	NOMAX_FAILURE	MAX_FAILURE
Dynamically changed by SQL statement . . .	ALTER SYSTEM	ALTER SYSTEM
Corresponding V$ARCHIVE_DEST column	MAX_FAILURE	Not applicable
Related V$ARCHIVE_DEST columns	FAILURE_COUNT, REOPEN_SECS	Not applicable

Purpose

The MAX_FAILURE and the NOMAX_FAILURE attributes allow you to control the number of times log transport services attempts to reestablish communication and resume archival operations with a failed destination.

Defaults

If you do not specify either the MAX_FAILURE or the NOMAX_FAILURE attribute, the default is NOMAX_FAILURE, which allows an unlimited number of consecutive attempts to transport archived redo logs to the failed destination.

Attributes

MAX_FAILURE=*count*

The MAX_FAILURE attribute specifies the maximum number of consecutive times that log transport services reattempts archival operations to a failed destination. Using this attribute, you can provide failure resolution for archiving destinations to which you want to retry archival operations after a failure, but not retry indefinitely. When you specify the MAX_FAILURE attribute, you must also set the REOPEN

attribute to specify how often archival operations are retried to the particular destination.

To Limit the Number of Archival Attempts: If you set both the MAX_FAILURE and REOPEN attributes to nonzero values, log transport services limits the number of archival attempts to the number of times specified by the MAX_FAILURE attribute. Each destination contains an internal failure counter that tracks the number of consecutive archival failures that have occurred. You can view the failure count in the FAILURE_COUNT column of the V$ARCHIVE_DEST fixed view. The related column REOPEN_SECS identifies the REOPEN attribute value.

If an archival operation fails for any reason, the failure count is incremented until:

- The failure no longer occurs and archival operations resume
- The failure count is greater than or equal to the value set for the MAX_FAILURE attribute

> **Note:** Once the destination's failure count reaches the specified MAX_FAILURE attribute value, the only way to reuse the destination is to modify the MAX_FAILURE attribute value or some other attribute.

- You issue the ALTER SYSTEM SET statement to dynamically change the MAX_FAILURE attribute (or any other destination attribute). The failure count is reset to zero (0) whenever the destination is modified by an ALTER SYSTEM SET statement. This avoids the problem of setting the MAX_FAILURE attribute to a value less than the current failure count value.

> **Note:** Runtime modifications made to the destination, such as changing the QUOTA_USED attribute, do not affect the failure count.

Once the failure count is greater than or equal to the value set for the MAX_FAILURE attribute, the REOPEN attribute value is implicitly set to the value zero (0), which causes log transport services to transport archived redo logs to an alternate destination (defined with the ALTERNATE attribute) on the next archival operation.

To Attempt Archival Operations Indefinitely: Log transport services attempt to archive to the failed destination indefinitely if you do not specify the MAX_FAILURE attribute (or if you specify MAX_FAILURE=0 or the NOMAX_FAILURE attribute), and

you specify a nonzero value for the REOPEN attribute. If the destination has the MANDATORY attribute, the online redo log is not reclaimable in the event of a repeated failure.

NOMAX_FAILURE
Specify the NOMAX_FAILURE attribute to allow an unlimited number of archival attempts to the failed destination.

The NOMAX_FAILURE attribute is equivalent to specifying MAX_FAILURE=0.

Examples

The following example allows log transport services up to three consecutive archival attempts, tried every 5 seconds, to the arc_dest destination. If the archival operation fails after the third attempt, the destination is treated as if the NOREOPEN attribute had been specified.

```
LOG_ARCHIVE_DEST_1='LOCATION=/arc_dest REOPEN=5 MAX_FAILURE=3'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

NET_TIMEOUT and NONET_TIMEOUT

Category	NET_TIMEOUT=*seconds*	NONET_TIMEOUT
Datatype of the attribute	Numeric	Not applicable
Minimum value	15	Not applicable
Maximum value	1200	Not applicable
Default value	Not applicable	Not applicable
Requires attributes ...	LGWR with SYNC=PARALLEL or LGWR with ASYNC > 0	Not applicable
Conflicts with attributes ...	ARCH, LOCATION, NONET_TIMEOUT, LGWR with SYNC=NOPARALLEL, LGWR with ASYNC=0	NET_TIMEOUT
Attribute class	ALTER SYSTEM	ALTER SYSTEM
Corresponding V$ARCHIVE_DEST column	NET_TIMEOUT	NET_TIMEOUT=0
Related V$ARCHIVE_DEST column	Not applicable	Not applicable

Purpose

The NET_TIMEOUT and NONET_TIMEOUT attributes of the LOG_ARCHIVE_DEST_*n* parameter specify the number of seconds the log writer process waits for status from the network server of a network operation issued by the log writer process.

Defaults

If the NET_TIMEOUT or the NONET_TIMEOUT attribute is not specified with the LOG_ARCHIVE_DEST_*n* parameter, the default is NONET_TIMEOUT.

Attributes

NET_TIMEOUT=*seconds*

The NET_TIMEOUT attribute is used only when the log writer process archives logs using a network server process, and when either the ASYNC or SYNC=PARALLEL

attribute is specified. The log writer process waits for the specified amount of time to hear from the network server process regarding the status of a network operation. If the network server process does not respond within the specified amount of time, the log writer process assumes there is a network timeout error.

The NET_TIMEOUT attribute allows a DBA to bypass the default network timeout interval established for the system on which the primary database resides. Without the NET_TIMEOUT attribute (or if NONET_TIMEOUT is explicitly specified), the primary database can potentially stall for the default network timeout period. By specifying a smaller, nonzero value for NET_TIMEOUT, the DBA can allow the primary database to continue operation after the user-specified timeout interval expires when waiting for status from the network server.

A timeout error causes an error archiving the current online log at the standby database. Archive gap recovery eventually enables archival of this log at the standby database.

NONET_TIMEOUT

The NONET_TIMEOUT attribute implies that the log writer process waits for the default network timeout interval established for the system. The default network timeout interval differs from system to system. On some systems, the default TCP/IP network timeout can be between 10 and 15 minutes.

Examples

The following example shows the NET_TIMEOUT attribute with the LOG_ARCHIVE_DEST_*n* parameter.

```
LOG_ARCHIVE_DEST_2='SERVICE=stby1 LGWR NET_TIMEOUT=40 SYNC=PARALLEL'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

Caution: Be careful to specify a reasonable value when running in maximum protection mode. A *false* network failure detection may cause the primary instance to shut down.

QUOTA_SIZE and NOQUOTA_SIZE

Category	QUOTA_SIZE=*blocks*	NOQUOTA_SIZE
Datatype of the attribute	Numeric	Keyword
Minimum value	0 blocks	Not applicable
Maximum value	Unlimited blocks	Not applicable
Default value	Not applicable	Not applicable
Requires attributes ...	LOCATION	Not applicable
Conflicts with attributes ...	NOQUOTA_SIZE, DEPENDENCY, SERVICE	QUOTA_SIZE
Attribute class	ALTER SYSTEM only	ALTER SYSTEM only
Corresponding V$ARCHIVE_DEST column	QUOTA_SIZE	QUOTA_SIZE
Related V$ARCHIVE_DEST column	QUOTA_USED	QUOTA_USED

Purpose

The QUOTA_SIZE and the NOQUOTA_SIZE attributes of the LOG_ARCHIVE_DEST_*n* parameter indicate the maximum number of 512-byte blocks of physical storage on a disk device that may be consumed by a local destination.

Defaults

If the QUOTA_SIZE or the NOQUOTA_SIZE attribute is not specified with the LOG_ARCHIVE_DEST_n parameter, the default is NOQUOTA_SIZE.

Attributes

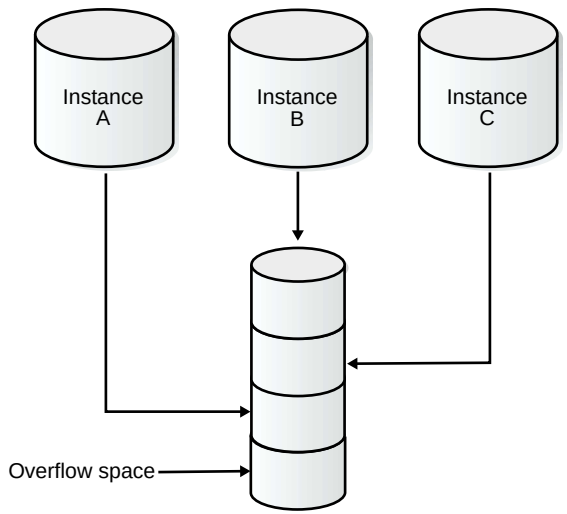
QUOTA_SIZE=*blocks*

The QUOTA_SIZE attribute indicates the maximum number of 512-byte blocks of physical storage on a disk device that may be consumed by a local destination. The value is specified in 512-byte blocks even if the physical device uses a different block size. The optional suffix values K, M, and G represent thousand, million, and billion, respectively (the value "1K" means 1,000 512-byte blocks).

A local archiving destination can be designated as being able to occupy all or some portion of the physical disk. For example, in a Real Application Clusters environment, a physical archived redo log disk device may be shared by two or more separate nodes (through a clustered file system, such as is available with Sun Clusters). As there is no cross-instance initialization parameter knowledge, none of the Real Application Clusters nodes is aware that the archived redo log physical disk device is shared with other instances. This can lead to significant problems when the destination disk device becomes full; the error is not detected until every instance tries to archive to the already full device. This seriously affects database availability.

For example, consider an 8-gigabyte (GB) disk device `/dev/arc_dest` that is further subdivided into node-specific directories `node_a`, `node_b`, and `node_c`. The DBA could designate that each of these instances is allowed to consume a maximum of 2 GB, which would allow an additional 2 GB for other purposes. This scenario is shown in Figure 12–2.

Figure 12–2 Specifying Disk Quota for a Destination



No instance consumes more than its allotted quota.

The quota is common to all users of the destination, including foreground archival operations, the archiver process, and even the log writer process.

Oracle Corporation highly recommends that the ALTERNATE attribute be used in conjunction with the QUOTA_SIZE attribute. However, this is not required.

See Also: ALTERNATE and NOALTERNATE on page 12-6

NOQUOTA_SIZE

Use of the NOQUOTA_SIZE attribute, or the QUOTA_SIZE attribute with a value of zero (0), indicates that there is unlimited use of the disk device by this destination; this is the default behavior.

Examples

The following example shows the QUOTA_SIZE attribute with the LOG_ARCHIVE_DEST_*n* parameter.

```
LOG_ARCHIVE_DEST_4='QUOTA_SIZE=100K'
```

QUOTA_USED and NOQUOTA_USED

Category	QUOTA_USED=*blocks*	NOQUOTA_USED
Datatype of the attribute	Numeric	Keyword
Minimum value	0 blocks	Not applicable
Maximum value	Unlimited blocks	Not applicable
Default value	Not applicable	Not applicable
Requires attributes ...	LOCATION	Not applicable
Conflicts with attributes ...	NOQUOTA_USED, DEPENDENCY, SERVICE	QUOTA_USED
Attribute class	ALTER SYSTEM only	ALTER SYSTEM only
Corresponding V$ARCHIVE_DEST column	QUOTA_USED	QUOTA_USED
Related V$ARCHIVE_DEST column	QUOTA_SIZE	QUOTA_SIZE

Purpose

The QUOTA_USED and the NOQUOTA_USED attributes of the LOG_ARCHIVE_DEST_*n* parameter identify the number of 512-byte blocks of data that have been archived on a specified destination.

Defaults

If the QUOTA_USED or the NOQUOTA_USED attribute is not specified with the LOG_ARCHIVE_DEST_n parameter, the default is NOQUOTA_USED.

The QUOTA_USED attribute has a default value of zero (0) for remote archiving destinations.

Attributes

QUOTA_USED=*blocks*
The QUOTA_USED attribute identifies the number of 512-byte blocks of data that have been archived on the specified local destination. The value is specified in 512-byte blocks even if the physical device uses a different block size. The optional

suffix values K, M, and G represent thousand, million, and billion, respectively (the value "1K" means 1,000 512-byte blocks).

This attribute cannot be modified at the session level.

If you specify a QUOTA_SIZE attribute value greater than zero (0) for a destination, but do not specify a QUOTA_USED attribute value in the database initialization parameter file, the QUOTA_USED attribute value is automatically determined when the database is initially mounted. The QUOTA_USED attribute value defaults to the actual number of blocks residing on the local archiving destination device. If the calculated QUOTA_USED attribute value exceeds the QUOTA_SIZE attribute value, the QUOTA_SIZE attribute value is automatically adjusted to reflect the actual storage consumed.

This automatic calculation of the QUOTA_USED value applies only to local archiving destinations.

Note: The runtime value of the QUOTA_USED attribute changes automatically as online redo log archival operations are started. The QUOTA_USED attribute value is optimistically pre-incremented against the destination quota size. You do not need to change the value of this attribute. In the event of archival failure, the QUOTA_USED value is adjusted.

If, at runtime, you dynamically modify the QUOTA_SIZE attribute value, but not the QUOTA_USED attribute value, the QUOTA_USED attribute value is not automatically recalculated.

For local destinations, the QUOTA_USED attribute value is incremented at the start of an archival operation. If the resulting value is greater than the QUOTA_SIZE attribute value, the destination status is changed to FULL and the destination is rejected before the archival operation begins.

The QUOTA_SIZE and QUOTA_USED attributes are very important because they can be used together to detect a lack of disk space before the archival operation begins.

Consider the case where the QUOTA_SIZE attribute value is "100K" and the QUOTA_USED attribute value is "100K" also. The destination status is VALID at this point. However, an attempt to archive 1 block results in the QUOTA_USED attribute value being changed to "101K", which exceeds the QUOTA_SIZE attribute value. Therefore, the destination status is changed to FULL, and the destination is rejected before the archival operation begins.

NOQUOTA_USED
Specifies that an unlimited number of blocks of data can be archived on a specified destination.

Examples

Oracle9*i* Data Guard automatically sets this value. You do not need to change the value of the attribute.

REGISTER and NOREGISTER

Category	REGISTER	NOREGISTER
Datatype of the attribute	Keyword	Keyword
Minimum value	Not applicable	Not applicable
Maximum value	Not applicable	Not applicable
Default value	Not applicable	Not applicable
Requires attributes ...	Not applicable	SERVICE
Conflicts with attributes ...	NOREGISTER, NOALTERNATE	REGISTER, LOCATION
Attribute class	ALTER SESSION and ALTER SYSTEM	ALTER SESSION and ALTER SYSTEM
Corresponding V$ARCHIVE_DEST column	DESTINATION	DESTINATION
Related $ARCHIVE_DEST column	TARGET	TARGET

Purpose

The REGISTER and the NOREGISTER attributes of the LOG_ARCHIVE_DEST_*n* parameter indicate whether the location of the archived redo log is to be recorded at the destination site.

Defaults

If the REGISTER or the NOREGISTER attribute is not specified with the LOG_ARCHIVE_DEST_*n* parameter, the default is REGISTER.

Attributes

REGISTER

The REGISTER attribute indicates that the location of the archived redo log is to be recorded at the corresponding destination.

For a physical standby destination, the archived redo log filename is recorded in the destination database control file, which is then used by the managed recovery operation.

For a logical standby database, the archived redo log filename is recorded in tablespace maintained by the logical standby database control file which is then used by SQL apply operations.

The REGISTER attribute implies that the destination is a Data Guard standby database.

By default, the location of the archived redo log, at a remote destination site, is derived from the destination instance initialization parameters STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT.

Note: You can also set the REGISTER attribute by executing the SQL ALTER DATABASE REGISTER LOGFILE *filespec* statement on each standby database. See Section 7.3.3.3 for an example of this SQL statement.

NOREGISTER

The optional NOREGISTER attribute indicates that the location of the archived redo log is not to be recorded at the corresponding destination. This setting pertains to remote destinations only. The location of each archived redo log is always recorded in the primary database control file.

The NOREGISTER attribute is required if the destination is a non-Data Guard standby database.

Examples

The following example shows the REGISTER attribute with the LOG_ARCHIVE_DEST_*n* parameter.

```
LOG_ARCHIVE_DEST_5='REGISTER'
```

REGISTER=location_format

Category	REGISTER=*location_format*
Datatype of the attribute	String value
Minimum value	Not applicable
Maximum value	Not applicable
Default value	Not applicable
Requires attributes ...	DEPENDENCY
Conflicts with attributes ...	NOREGISTER, LOCATION, TEMPLATE
Attribute class	ALTER SESSION and ALTER SYSTEM
Corresponding V$ARCHIVE_DEST column	DESTINATION
Related V$ARCHIVE_DEST column	TARGET

Purpose

The optional REGISTER=*location_format* attribute is used to specify a filename format template for archived redo logs that is different from the default filename format template defined in the primary and standby database initialization parameter files.

This attribute is for use on physical standby databases only.

Defaults

There is no default for this attribute.

Attributes

REGISTER=*location_format*
The optional REGISTER=*location_format* attribute is used to specify a fully qualified filename format template for archived redo logs that is different from the default filename format template defined in the primary and standby database initialization parameter files. The default filename format template is a combination

of the database initialization parameters STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT.

See Also: REGISTER and NOREGISTER on page 12-37

The REGISTER=*location_format* attribute is valid with remote destinations only.

Examples

The following example shows the REGISTER=*location_format* attribute with the LOG_ARCHIVE_DEST_*n* parameter.

```
LOG_ARCHIVE_DEST_4='REGISTER=/dsk1/dir1/arc_%t_%s.arc'
```

REOPEN and NOREOPEN

Category	REOPEN [=*seconds*]	NOREOPEN
Datatype of the attribute	Numeric	Keyword
Minimum value	0 seconds	Not applicable
Maximum value	Unlimited seconds	Not applicable
Default value	300 seconds	Not applicable
Requires attributes ...	Not applicable	Not applicable
Conflicts with attributes ...	NOREOPEN	REOPEN
Attribute class	ALTER SESSION and ALTER SYSTEM	ALTER SESSION and ALTER SYSTEM
Corresponding V$ARCHIVE_DEST column	REOPEN_SECS	REOPEN_SECS
Related V$ARCHIVE_DEST column	MAX_FAILURE	MAX_FAILURE

Purpose

The REOPEN and the NOREOPEN attributes of the LOG_ARCHIVE_DEST_*n* parameter specify the minimum number of seconds before the archiver process (ARC*n*, foreground, or log writer process) should try again to access a previously failed destination. You can turn off the attribute by specifying NOREOPEN.

Defaults

If the REOPEN or the NOREOPEN attribute is not specified with the LOG_ARCHIVE_DEST_n parameter, the default is REOPEN. If the REOPEN attribute is specified without an integer value, the default is 300 seconds.

Attributes

REOPEN[=*seconds*]
REOPEN applies to all errors, not just connection failures. These errors include, but are not limited to, network failures, disk errors, and quota exceptions.

NOREOPEN

If you specify NOREOPEN, the failed destination remains disabled until:

- You manually reenable the destination
- You issue an ALTER SYSTEM SET or an ALTER SESSION SET statement with the REOPEN attribute
- The instance is restarted

Examples

The following example shows the REOPEN attribute with the LOG_ARCHIVE_DEST_*n* parameter.

```
LOG_ARCHIVE_DEST_3='SERVICE=stby1 MANDATORY REOPEN=60'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

SYNC and ASYNC

Category	SYNC[=*parallel_option*]	ASYNC[=*blocks*]
Datatype of the attribute	Keyword	Numeric
Minimum value	Not applicable	0 blocks
Maximum value	Not applicable	20,480 blocks
Default value	Not applicable	2,048
Requires attributes ...	Not applicable	LGWR
Conflicts with attributes ...	ASYNC	SYNC, LOCATION, ARCH
Attribute class	ALTER SESSION and ALTER SYSTEM	ALTER SYSTEM only
Corresponding V$ARCHIVE_DEST column	TRANSMIT_MODE	TRANSMIT_MODE
Related V$ARCHIVE_DEST column	Not applicable	ASYNC_BLOCKS

Purpose

The SYNC and the ASYNC attributes of the LOG_ARCHIVE_DEST_*n* parameter specify that network I/O operations are to be done synchronously or asynchronously when using the log writer process (LGWR).

> **Note:** When the primary database is in one of the three protection modes, standby redo log archiving destinations using the log writer process are automatically placed in SYNC mode.

Defaults

If SYNC or ASYNC attribute is not specified, the default is SYNC. If the destination defaults to SYNC, or the SYNC attribute is specified without specifying the parallel qualifier, the default for the parallel qualifier depends on which transmitter process is chosen for the destination. When you specify the LGWR attribute, the default parallel qualifier is PARALLEL. Because the PARALLEL qualifier is not allowed with the ARCH attribute, when you specify the ARCH attribute, the default parallel qualifier is NOPARALLEL.

If the ASYNC attribute is specified without an integer value, the default is 2048 blocks.

Attributes

SYNC=PARALLEL
SYNC=NOPARALLEL
The SYNC attribute specifies that network I/O is to be performed synchronously for the destination, which means that once the I/O is initiated, the archiving process waits for the I/O to complete before continuing. The SYNC attribute is one requirement for setting up a no-data-loss environment, because it ensures that the redo records have been successfully transmitted to the standby site before continuing.

If the log writer process is defined to be the transmitter to multiple standby destinations that use the SYNC attribute, the user has the option of specifying SYNC=PARALLEL or SYNC=NOPARALLEL for each of those destinations. If SYNC=NOPARALLEL is used, the log writer process performs the network I/O to each destination in series. In other words, the log writer process initiates an I/O to the first destination and waits until it completes before initiating the I/O to the next destination. If SYNC=PARALLEL is used, the network I/O is initiated asynchronously, so that I/O to multiple destinations can be initiated in parallel. However, once the I/O is initiated, the log writer process waits for each I/O operation to complete before continuing. This is, in effect, the same as performing multiple, synchronous I/O operations simultaneously. The use of SYNC=PARALLEL is likely to perform better than SYNC=NOPARALLEL.

Because the PARALLEL and NOPARALLEL qualifiers only make a difference if multiple destinations are involved, Oracle Corporation recommends that all destinations use the same value.

ASYNC[=*blocks*]
The ASYNC attribute specifies that network I/O is to be performed asynchronously for the destination. Once the I/O is initiated, the log writer continues processing the next request without waiting for the I/O to complete and without checking the completion status of the I/O. Use of the ASYNC attribute allows standby environments to be maintained with little or no performance effect on the primary database. The optional block count determines the size of the SGA network buffer to be used. In general, the slower the network connection, the larger the block count should be.

See Also: Appendix C, "Log Writer Asynchronous Network I/O"

Examples

The following example shows the SYNC attribute with the LOG_ARCHIVE_DEST_*n* parameter.

```
LOG_ARCHIVE_DEST_3='SERVICE=stby1 LGWR SYNC'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

TEMPLATE and NOTEMPLATE

Category	TEMPLATE=*filename_template*	NOTEMPLATE
Datatype of the attribute	String value	Not applicable
Minimum value	Not applicable	Not applicable
Maximum value	Not applicable	Not applicable
Default value	Not applicable	Not applicable
Requires attributes ...	SERVICE	Not applicable
Conflicts with attributes ...	NOTEMPLATE, LOCATION, REGISTER=*location_format*	TEMPLATE
Attribute class	ALTER SESSION and ALTER SYSTEM	ALTER SESSION and ALTER SYSTEM
Corresponding V$ARCHIVE_DEST column	REMOTE_TEMPLATE	REMOTE_TEMPLATE
Related V$ARCHIVE_DEST column	REGISTER	REGISTER

Purpose

The TEMPLATE and the NOTEMPLATE attributes of the LOG_ARCHIVE_DEST_*n* parameter define a directory specification and format template for archived redo logs at the standby destination. You can specify this attribute in either the primary or standby initialization parameter file, but the attribute applies only to the database role that is archiving.

The TEMPLATE attribute overrides the STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT initialization parameter settings at the remote archive destination.

The TEMPLATE and NOTEMPLATE attributes are valid only with remote destinations.

Note: If used on a LGWR destination, rearchival by the ARC*n* process does not use the TEMPLATE specification. This is important for protected destinations.

Defaults

There is no default for this attribute.

Attributes

TEMPLATE=*filename_template*
Use the optional TEMPLATE attribute to define a directory specification and format for archived redo logs at the standby destination. The definition is used to generate a filename that is different from the default filename format defined by the STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT initialization parameters at the standby destination.

The *filename_template* value of the TEMPLATE attribute must contain a thread or sequence number directive. The following table provides a definition for each directive.

Thread or Sequence Number Directive	Description
%a	Substitute the database activation ID.
%A	Substitute the database activation ID zero-filled.
%d	Substitute the database ID.
%D	Substitute the database ID zero-filled.
%t	Substitute the instance thread number.
%T	Substitute the instance thread number zero-filled.
%s	Substitute the log file sequence number.
%S	Substitute the log file sequence number zero-filled.

The *filename_template* value is transmitted to the standby destination, where it is translated and validated prior to creating the filename.

If you do not specify the TEMPLATE attribute, the setting is the same as REGISTER.

NOTEMPLATE
Use the optional NOTEMPLATE attribute to allow the filename format template defined by the STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT initialization parameters to take effect.

Examples

The following example shows the TEMPLATE attribute with the LOG_ARCHIVE_DEST_*n* parameter.

```
LOG_ARCHIVE_DEST_1='SERVICE=stby1 MANDATORY REOPEN=5
    TEMPLATE=/usr/oracle/prmy1/p1_%t_%s.dbf'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

Prmy1 archived redo logs at the remote destination stby1 is located in the directory /usr/oracle/prmy1 with the filename format of p1_<thread#>_<sequence#>.dbf.

12.2 Attribute Compatibility for Archive Destinations

The LOG_ARCHIVE_DEST_*n* initialization parameter has many attributes. Some of these attributes conflict with each other. Some of the attributes require that other attributes are also defined. Table 12–1 lists the supported attributes and the requirements associated with each one.

Table 12–1 LOG_ARCHIVE_DEST_n Attribute Compatibility

Attribute	Requires...	Conflicts with...
AFFIRM	Not applicable	NOAFFIRM
NOAFFIRM	Not applicable	AFFIRM
ALTERNATE=*destination*	Not applicable	NOALTERNATE
NOALTERNATE	Not applicable	ALTERNATE
ARCH	Not applicable	LGWR ASYNC NET_TIMEOUT
ASYNC[=*blocks*]	LGWR	SYNC LOCATION ARCH
DELAY	SERVICE	LOCATION NODELAY
NODELAY	Not applicable	DELAY
DEPENDENCY	SERVICE REGISTER	LOCATION NODEPENDENCY NOREGISTER QUOTA_SIZE QUOTA_USED
NODEPENDENCY	Not applicable	DEPENDENCY
LGWR	Not applicable	ARCH
LOCATION	Not applicable	SERVICE DEPENDENCY REGISTER=*location_format* NOREGISTER DELAY ASYNC NET_TIMEOUT TEMPLATE

Table 12–1 (Cont.) LOG_ARCHIVE_DEST_n Attribute Compatibility

Attribute	Requires...	Conflicts with...
MANDATORY	Not applicable	OPTIONAL
MAX_FAILURE	REOPEN	NOMAX_FAILURE
NOMAX_FAILURE	Not applicable	MAX_FAILURE
NET_TIMEOUT	LGWR with SYNC=PARALLEL or LGWR with ASYNC > 0	ARCH LOCATION NONET_TIMEOUT LGWR with SYNC=NOPARALLEL LGWR with ASYNC=0
NONET_TIMEOUT	Not applicable	NET_TIMEOUT
OPTIONAL	Not applicable	MANDATORY
QUOTA_SIZE	LOCATION	DEPENDENCY SERVICE NOQUOTA_SIZE
NOQUOTA_SIZE	Not applicable	QUOTA_SIZE
QUOTA_USED	LOCATION	DEPENDENCY SERVICE NOQUOTA_USED
NOQUOTA_USED	Not applicable	QUOTA_USED
REGISTER	Not applicable	NOALTERNATE NOREGISTER
NOREGISTER	SERVICE	LOCATION REGISTER
REGISTER= *location_format*	DEPENDENCY	LOCATION NOREGISTER TEMPLATE
REOPEN	Not applicable	NOREOPEN
NOREOPEN	Not applicable	REOPEN
SERVICE	Not applicable	LOCATION QUOTA_USED QUOTA_SIZE
SYNC[= *parallel_option*]	Not applicable	ASYNC

Table 12–1 (Cont.) LOG_ARCHIVE_DEST_n Attribute Compatibility

Attribute	Requires...	Conflicts with...
TEMPLATE	SERVICE	NOTEMPLATE LOCATION REGISTER=*location_format*
NOTEMPLATE	Not applicable	TEMPLATE

13

SQL Statements

This chapter summarizes the SQL statements that are useful for performing operations on standby databases in a Data Guard environment. These include:

```
ALTER DATABASE ACTIVATE STANDBY DATABASE
ALTER DATABASE ADD [STANDBY] LOGFILE
ALTER DATABASE ADD [STANDBY] LOGFILE MEMBER
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA
ALTER DATABASE COMMIT TO SWITCHOVER
ALTER DATABASE CREATE STANDBY CONTROLFILE AS
ALTER DATABASE DROP [STANDBY] LOGFILE
ALTER DATABASE DROP [STANDBY] LOGFILE MEMBER
ALTER DATABASE [NO]FORCE LOGGING
ALTER DATABASE MOUNT STANDBY DATABASE
ALTER DATABASE OPEN READ ONLY
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE
ALTER DATABASE REGISTER LOGFILE
ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE {PROTECTION | AVAILABILITY | PERFORMANCE}
ALTER DATABASE START LOGICAL STANDBY APPLY
ALTER DATABASE {STOP|ABORT} LOGICAL STANDBY APPLY
```

See Also: *Oracle9i SQL Reference* for additional information about these and other SQL statements

13.1 ALTER DATABASE ACTIVATE STANDBY DATABASE

This statement performs a forced failover operation, in which the primary database is removed from the Data Guard environment and a standby database assumes the primary database role. The standby database must be mounted before it can be activated with this statement. The SQL statement syntax is:

```
ALTER DATABASE ACTIVATE [PHYSICAL | LOGICAL] STANDBY DATABASE [SKIP [STANDBY LOGFILE]];
```

Table 13–1 describes the keywords for this statement.

Table 13–1 Keywords for the ACTIVATE STANDBY DATABASE Clause

Keyword	Description
PHYSICAL	Activates a physical standby database. This is the default.
LOGICAL	Activates a logical standby database. If you have more than one logical standby database, you should first ensure that the same log data is available on all the logical standby sites.
SKIP [STANDBY LOGFILE] (Physical standby databases only)	Forces the failover operation to proceed even if standby redo logs contain data that could be recovered using the RECOVER MANAGED STANDBY DATABASE FINISH clause. Using the SKIP [STANDBY LOGFILE] clause indicates that it is acceptable to discard the contents of the standby redo log.

See Section 7.3.4 for more information and examples of this SQL statement.

> **Note:** Oracle Corporation recommends that you perform a graceful failover operation using the ALTER DATABASE RECOVER MANAGED STANDBY DATABASE statement with the FINISH or FINISH SKIP keywords rather than a forced failover operation whenever possible. A forced failover operation renders other standby databases that are not participating in the failover operation unusable as standby databases to the newly activated primary database.

13.2 ALTER DATABASE ADD [STANDBY] LOGFILE

This statement adds one or more redo log file groups to the specified thread, making the logs available to the instance assigned the thread. The SQL statement syntax is:

```
ALTER DATABASE ADD [STANDBY] LOGFILE [THREAD integer] [GROUP integer] filespec;
```

Table 13–2 describes the keywords for this statement.

Table 13–2 Keywords for the ADD LOGFILE Clause

Keyword	Description
STANDBY	Indicates that the redo log file created is for use by standby databases only.

Table 13–2 (Cont.) Keywords for the ADD LOGFILE Clause

Keyword	Description
THREAD *integer*	Applicable only if you are using the Real Application Clusters option in parallel mode. The *integer* variable is the thread number.
GROUP *integer*	Uniquely identifies the redo log file group among all groups in all threads and can range from 1 to the MAXLOGFILES value. You cannot add multiple redo log groups having the same GROUP value.
filespec	Specifies a redo log file group containing one or more members.
REUSE	If the file already exists, you can specify REUSE to allow Data Guard to overwrite the header information in the log file.
SIZE	Specify the size of the log file in bytes. Use K or M to specify the size in kilobytes or megabytes.

See Section 5.8.4.3 for more information about this SQL statement.

13.3 ALTER DATABASE ADD [STANDBY] LOGFILE MEMBER

This statement adds new members to existing redo log file groups. The SQL statement syntax is:

```
ALTER DATABASE ADD [STANDBY] LOGFILE MEMBER 'filename' [REUSE] TO logfile-descriptor;
```

Table 13–3 describes the keywords for this statement.

Table 13–3 Keywords for the ADD LOGFILE MEMBER Clause

Keyword	Description
STANDBY	Indicates that the log file member is for use only by a standby database. The STANDBY keyword is not required, but you can use it for symmetry in scripts, if necessary.
LOGFILE MEMBER 'filename'	Adds new members to existing redo log groups. Each new member is specified by 'filename'.
REUSE	If the file specified by 'filename' already exists, the log must be the same size as the other group members, and you must specify REUSE to allow Oracle to overwrite the existing file. If the file does not already exist, Oracle creates a log of the correct size.

Table 13–3 (Cont.) Keywords for the ADD LOGFILE MEMBER Clause

Keyword	Description
logfile_descriptor	Specify an existing log file group using either of these ways: ■ Specify the GROUP *integer* parameter, where *integer* is the number of the existing log file group. If this log file group was added for standby database use, all members of the log file group will be used only for standby databases. ■ List the full filename specification for each member of the redo log file group. Specify the filename according to the conventions for your operating system.

See Section 5.8.4.4 for more information about this SQL statement.

13.4 ALTER DATABASE ADD SUPPLEMENTAL LOG DATA

This statement is for logical standby databases only.

You must enable full supplemental logging before you create the logical standby database. This is because supplemental logging is the source of change to a logical standby database. To implement full supplemental logging, you must specify either PRIMARY KEY COLUMNS or UNIQUE INDEX COLUMNS. The SQL statement syntax is:

```
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA {PRIMARY KEY | UNIQUE INDEX} COLUMNS;
```

Table 13–4 describes the keywords for this statement.

Table 13–4 Keywords for the ADD SUPPLEMENTAL LOG DATA Clause

Keyword	Description
PRIMARY KEY	Ensures, for all tables with a primary key, that all columns of the primary key are placed into the redo log whenever an update operation is performed. If no primary key is defined, Oracle places into the redo log a set of columns that uniquely identifies the row.
UNIQUE INDEX	Ensures, for all tables with a unique index, that if any unique index columns are modified, all other columns belonging to the unique index are also placed into the redo log.

See Section 4.1 (step 4) for more information about this SQL statement.

13.5 ALTER DATABASE COMMIT TO SWITCHOVER

Use this statement to perform a switchover operation to change the current primary database to the standby role and to change one standby database to the primary role. The SQL statement clauses you specify differ depending on whether you issue the statement on the primary database, a physical standby database, or a logical standby database:

- On the primary database that you want to change to run in a physical standby database role, use the following SQL statement syntax:

```
ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY [[WITH | WITHOUT]
SESSION SHUTDOWN ] [WAIT | NOWAIT];
```

- On the primary database that you want to change to run in a logical standby database role, use the following SQL statement syntax:

```
ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY [WAIT | NOWAIT];
```

- On a physical standby database that you want to change to run in the primary role, use the following SQL statement syntax:

```
ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY [[WITH | WITHOUT] SESSION
SHUTDOWN ] [WAIT | NOWAIT];
```

- On a logical standby database that you want to change to run in the primary role, use the following SQL statement syntax:

```
ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY [WAIT | NOWAIT];
```

Table 13–5 describes the keywords for this statement.

Table 13–5 Keywords for the COMMIT TO SWITCHOVER Clause

Keyword	Description
`COMMIT TO SWITCHOVER TO PHYSICAL STANDBY`	Transitions the primary database to run in the role of a physical standby database.The physical standby database must be mounted and may be open in `READ ONLY` mode.
`COMMIT TO SWITCHOVER TO LOGICAL STANDBY`	Transitions the primary database to run in the role of a logical standby database. This option must be followed by an `ALTER DATABASE START LOGICAL STANDBY APPLY` statement.

Table 13–5 (Cont.) Keywords for the COMMIT TO SWITCHOVER Clause

Keyword	Description
`COMMIT TO SWITCHOVER TO [PHYSICAL | LOGICAL] PRIMARY`	Transitions the standby database to run in the primary database role. For physical standby databases only, the standby database must be mounted and may be open in `READ ONLY` mode. You can specify the `PHYSICAL` or `LOGICAL` parameters for symmetry (in scripts, for example), but these keywords are not required.
`WITH SESSION SHUTDOWN` (Physical standby databases only)	Shuts down any open application sessions and rolls back uncommitted transactions as part of the execution of this statement. Logical standby databases do not support the `WITH SESSION SHUTDOWN` option.
`WITHOUT SESSION SHUTDOWN` (Physical standby databases only)	Causes the switchover operation to fail if any application sessions are open. This is the default. Logical standby databases do not support the `WITHOUT SESSION SHUTDOWN` option.
`WAIT`	Waits for the completion of the switchover operation before returning control. This is the default.
`NOWAIT`	Returns control before the switchover operation is complete.

See Section 7.2 for additional information about this SQL statement.

13.6 ALTER DATABASE CREATE STANDBY CONTROLFILE AS

This statement is for physical standby databases only.

This statement creates a standby control file. Issue this statement on the primary database. The SQL statement syntax is:

```
ALTER DATABASE CREATE STANDBY CONTROLFILE AS 'filename' [REUSE];
```

Table 13–6 describes the keywords for this statement.

Table 13–6 Keywords for the CREATE STANDBY CONTROLFILE AS Clause

Keyword	Description
`CONTROLFILE AS 'filename'`	Specifies the name of the control file to be created and used to maintain a standby database.

Table 13–6 (Cont.) Keywords for the CREATE STANDBY CONTROLFILE AS Clause

Keyword	Description
REUSE	If the control file specified by the '*filename*' parameter already exists, you must specify REUSE to allow Oracle to overwrite the existing file.

See Section 3.3.3 for more information about this SQL statement.

13.7 ALTER DATABASE DROP [STANDBY] LOGFILE

This clause drops all members of a redo log group. The SQL statement syntax is:

```
ALTER DATABASE DROP [STANDBY] LOGFILE logfile_descriptor;
```

Table 13–7 describes the keywords for this statement.

Table 13–7 Keywords for the DROP [STANDBY] LOGFILE Clause

Keyword	Description
STANDBY	Drops all members of a redo log file group. You can specify STANDBY for symmetry, but this keyword is not required.
logfile_descriptor	Specify an existing log file group using either of these ways: ■ Specify the GROUP *integer* parameter, where *integer* is the number of the existing log group. ■ List the full filename specification for the redo log file group. Specify the filename according to the conventions for your operating system.

See Section 10.1.3.4 for an example using this SQL statement.

13.8 ALTER DATABASE DROP [STANDBY] LOGFILE MEMBER

This statement drops one or more redo log members. The SQL statement syntax is:

```
ALTER DATABASE DROP [STANDBY] LOGFILE MEMBER 'filename';
```

Table 13–8 describes the keywords for this statement.

Table 13–8 Keywords for the DROP LOGFILE MEMBER Clause

Keyword	Description
STANDBY	Drops one or more standby redo log members. You can specify STANDBY for symmetry, but this keyword is not required.
'filename'	Specifies filenames, separated by commas, for one or more log members. Each '*filename*' must specify the fully qualified file specification according to the conventions for your operating system.

13.9 ALTER DATABASE [NO]FORCE LOGGING

Controls whether or not the Oracle database server logs all changes in the database except for changes to temporary tablespaces and temporary segments. The [NO]FORCE LOGGING clause:

- Is required for physical standby databases to prevent inconsistent standby databases
- Is recommended for logical standby databases to ensure data availability at the standby database

> **Note:** Oracle Corporation recommends setting the FORCE LOGGING clause before performing the backup operation to create the standby database and remaining in force logging mode for as long as the standby database is active. Also, to prevent performance degradation when in force logging mode, the database should be running in ARCHIVELOG mode.

The primary database must be mounted but not open when you issue this statement. The SQL statement syntax is:

```
ALTER DATABASE [NO]FORCE LOGGING;
```

Table 13–9 describes the keywords for this statement.

Table 13–9 Keywords for the [NO]FORCE LOGGING Clause

Keyword	Description
FORCE LOGGING	Logs all changes in the database except for changes in temporary tablespaces and temporary segments. This setting takes precedence over and is independent of any NOLOGGING or FORCE LOGGING settings you specify for individual tablespaces and any NOLOGGING setting you specify for individual database objects. All ongoing, unlogged operations must finish before forced logging can begin.
NOFORCE LOGGING	Cancels the force logging mode. NOFORCE LOGGING is the default.

13.10 ALTER DATABASE MOUNT STANDBY DATABASE

Mounts a physical standby database, allowing the standby instance to receive archived redo logs from the primary instance. The SQL statement syntax is:

```
ALTER DATABASE MOUNT STANDBY DATABASE;
```

See Section 6.3.2 for another example of this SQL statement.

13.11 ALTER DATABASE OPEN READ ONLY

This statement is required for physical standby databases. It can be used for logical standby databases.

Opens a physical standby database in read-only mode. This SQL statement restricts users to read-only transactions, preventing them from generating redo logs. You can use this clause to make a physical standby database available for queries even while archive logs are being copied from the primary database site.

You must mount the physical standby database before you can open it. The SQL statement syntax is:

```
ALTER DATABASE OPEN READ ONLY;
```

See Section 6.3.5 for more information about this SQL statement.

13.12 ALTER DATABASE RECOVER MANAGED STANDBY DATABASE

This statement is for physical standby databases only.

Use this statement to start, control, and cancel managed recovery operations and log apply services for physical standby databases. You can use the RECOVER MANAGED STANDBY DATABASE clause on a database that is mounted, open or closed. Although this SQL statement does not require any additional clauses, it provides many options to help you control the managed recovery process. The SQL statement syntax is:

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE [ startup_clause | modify_clause | cancel_
clause ];
```

startup_clause

When you start managed recovery operations, you can start log apply services in a foreground or a background session:

- To start a foreground session, the SQL statement syntax is:

  ```
  ALTER DATABASE RECOVER MANAGED STANDBY DATABASE [TIMEOUT | NO TIMEOUT];
  ```

- To start a background session, the SQL statement syntax is:

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT [FROM SESSION] [NO TIMEOUT];
```

modify_clause

The RECOVER MANAGED STANDBY DATABASE clause provides a wealth of options for controlling the managed recovery process, switchover operations, and failover operations. These keywords work the same whether managed recovery operations were started in a foreground or a background session, with the exception of graceful failover and switchover operations.

Keywords can be placed in any order in the SQL statement except when you start a graceful failover operation using the FINISH keyword. This keyword must be specified last in the SQL statement.

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE [
[ NO TIMEOUT | TIMEOUT [integer] ]
[ NODELAY | DELAY [integer] ]
[ DEFAULT DELAY ]
[ NO EXPIRE | EXPIRE [integer] ]
[ NEXT [integer] ]
[ NOPARALLEL | PARALLEL [integer]]
[ THROUGH { ALL | NEXT | LAST } SWITCHOVER ]
[ THROUGH ALL ARCHIVELOG  [ THREAD n ] SEQUENCE n ]
[ FINISH [ SKIP [STANDBY LOGFILE] [NOWAIT | WAIT] ] ]
]
```

cancel_clause

To stop a managed recovery session, the SQL statement syntax is:

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL [IMMEDIATE] [NOWAIT];
```

Table 13–10 describes all of the keywords.

Table 13–10 Keywords for the RECOVER MANAGED STANDBY DATABASE Clause

Keyword	Description	Incompatible with
CANCEL [IMMEDIATE] [NOWAIT]	Terminates managed recovery. By default, log apply services will finish applying the current archived redo log before terminating. See Section 6.3.3.1. Specify IMMEDIATE to terminate managed recovery either before reading another block from the archived redo log or before opening the next archived redo log, whichever occurs first. Specify NOWAIT to return control to the process that issued the CANCEL statement without waiting for the managed recovery process to terminate.	All other keywords
DELAY *integer*	Specifies an absolute delay interval (in minutes) that log apply services will wait before applying the individual archived redo logs. The apply delay interval begins once the archived redo logs have been selected for recovery. See Section 6.3.3.3.	CANCEL, FINISH, NODELAY
DEFAULT DELAY	Reverts the delay interval to the number of minutes that was specified in the LOG_ARCHIVE_DEST_*n* initialization parameter on the primary database, if any. See Section 6.3.3.2.	CANCEL, DELAY, FINISH, NODELAY
DISCONNECT [FROM SESSION]	Starts the managed recovery process (MRP) and log apply services in a background server process. See Section 6.3.3.4.	CANCEL, TIMEOUT
NEXT *integer*	Specifies the number of delayed archived redo logs that log apply services should apply as soon as possible after the log transport services have archived them. See Section 6.3.3.7.	CANCEL, FINISH
EXPIRE *integer*	Specifies the number of minutes, relative to the current time, after which the managed recovery operation will automatically terminate. Log apply services will finish applying the current archived redo log before stopping. See Section 6.3.3.5.	CANCEL, FINISH, NO EXPIRE

Table 13–10 (Cont.) Keywords for the RECOVER MANAGED STANDBY DATABASE Clause

Keyword	Description	Incompatible with
FINISH [SKIP [STANDBY LOGFILE]] [NOWAIT | WAIT]	Invokes a graceful failover operation that first applies all available archived redo logs and then recovers available standby redo logs. See Section 6.3.3.6. Specify SKIP [STANDBY LOGFILE] to indicate that it is acceptable to skip applying the contents of the standby redo logs. Specify NOWAIT to return control to the foreground process before the recovery completes. Specify WAIT to return control after recovery completes.	CANCEL, DELAY, EXPIRE, NEXT, THROUGH..., TIMEOUT
NODELAY	Disables a previously specified DELAY option so that log apply services will apply the archived redo logs to the standby database without delay. See Section 6.3.3.8.	CANCEL, DELAY
NO EXPIRE	Disables a previously specified EXPIRE option. See Section 6.3.3.5.	CANCEL, EXPIRE
NO TIMEOUT	Disables a previously specified TIMEOUT option. See Section 6.3.3.13.	CANCEL
NOPARALLEL	Disables a previously specified PARALLEL option so that log apply services use a single process to apply all of the archived redo logs sequentially. This is the default. See Section 6.3.3.9.	CANCEL
PARALLEL [*integer*]	Starts additional parallel recovery processes to spread the workload of applying the archived redo logs simultaneously to the standby datafiles. By default, Oracle selects a number of parallel processes that equal the number of CPUs available on all active instances times the value of the PARALLEL_THREADS_PER_CPU initialization parameter. However, you can specify *integer* to indicate the number of parallel processes to use in the parallel operation. Each parallel thread may use one or two parallel execution servers. See Section 6.3.3.9.	CANCEL
THROUGH [THREAD *n*] SEQUENCE *n*	Specifies the thread number and sequence number of the archived redo log through which you want to recover. Once the specified archived redo log has been applied, managed recovery terminates. The THREAD *n* keyword is optional. If you do not specify THREAD *n*, it defaults to thread 1. See Section 6.3.3.12.	CANCEL, FINISH

Table 13–10 (Cont.) Keywords for the RECOVER MANAGED STANDBY DATABASE Clause

Keyword	Description	Incompatible with
THROUGH ALL ARCHIVELOG	Specifies the default behavior for managed recovery mode, which is to continue managed recovery until it is explicitly stopped. This clause is useful to alter a managed recovery that is currently running with a THROUGH THREAD *n* SEQUENCE *n* keyword so that it does not stop after applying the specified archived redo log. See Section 6.3.3.10.	CANCEL, FINISH
THROUGH {ALL|NEXT|LAST} SWITCHOVER	Keeps log apply services actively applying archived redo logs received from the new primary database after a switchover operation. (By default, log apply services stop after encountering a switchover (end-of-redo) marker within an archived redo log.) See Section 6.3.3.11 for more information. ALL - Continues managed recovery until you explicitly cancel it. Managed recovery continues through (ignoring) all switchover (end-of-redo) indicators. The ALL option is useful when there are other standby databases that are not participating in the switchover operation and you do not want to stop and restart the recovery process on each one. NEXT - Stops managed recovery at the first switchover (end-of-redo) indicator encountered. This is the default behavior. LAST - Continues managed recovery through all switchover (end-of-redo) indicators, stopping only when an end-of-redo marker is encountered in the last archived redo log received. (See Section 6.3.3.11 for exceptions.)	CANCEL, FINISH
TIMEOUT *integer*	Specifies the number of minutes that the managed recovery process waits for the next archived redo log to arrive from the primary database. If another log does not arrive within the specified time, log apply services automatically stop. Specify TIMEOUT only when starting a managed recovery in a foreground session. See Section 6.3.3.13.	CANCEL, DISCONNECT, FINISH

See Also: Section 6.3.1 for complete information about controlling log apply services and the managed recovery process.

13.13 ALTER DATABASE REGISTER LOGFILE

This clause allows the registration of manually archived redo logs. The SQL statement syntax is:

```
ALTER DATABASE REGISTER [OR REPLACE] [PHYSICAL | LOGICAL] LOGFILE filespec;
```

Table 13–11 describes the keywords for this statement.

Table 13–11 Keywords for the REGISTER LOGFILE Clause

Keyword	Description
`OR REPLACE`	Allows updates to an existing archived redo log entry for the standby database (for example, when the location or file specification for the archived redo log changes). The SCNs of the entries must match exactly, and the original entry must have been created by log transport services.
`PHYSICAL`	Indicates that the archived redo log will be registered in the control file for the physical standby database.
`LOGICAL`	Indicates that the archived redo log will be registered in the dictionary for the logical standby database.

See Section 7.3.3.3 for an example using this SQL statement.

13.14 ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE {PROTECTION | AVAILABILITY | PERFORMANCE}

Use this statement to specify the level of protection for the data in your database environment. Using one of these protection levels, you can protect the primary database against data loss and data divergence. The SQL statement syntax is:

```
ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE {PROTECTION | AVAILABILITY | PERFORMANCE};
```

You execute this statement on the primary database, which must be stopped and in the mount state. Table 13–12 describes the keywords for this statement.

Table 13–12 Keywords for the SET STANDBY TO MAXIMIZE Clause

Keyword	Description
`PROTECTION` (Physical standby databases only)	Offers the highest level of data protection. A transaction does not commit until all data needed to recover that transaction has been written to at least one physical standby database that is configured to use the `SYNC` log transport mode. If the primary database is unable to write the redo records to at least one such standby database, the primary database is shut down. This mode guarantees zero data loss, but it has the highest potential impact on the performance and availability of the primary database.

Table 13–12 (Cont.) Keywords for the SET STANDBY TO MAXIMIZE Clause

Keyword	Description
AVAILABILITY	Offers the next highest level of data protection. This mode guarantees there will be no data loss between the primary site and at least one standby site in the configuration unless a primary database failure occurs before recovery from a network outage. Then, no data is lost up to the last transaction that was shipped to the site. (Transactions that continued on the primary site after the network went down could be lost.) Unlike maximum protection mode, the primary database will not shut down if it is unable to write the redo records to at least one such standby database. Instead, the protection will be lowered to maximum performance mode until the situation has been corrected and the standby database has caught up with the primary database. This mode guarantees zero data loss unless the primary database fails while in maximum performance mode. Maximum availability mode provides the highest level of data protection that is possible without affecting the availability of the primary database.
PERFORMANCE	This is the default protection mode. A transaction commits before the data needed to recover the transaction has been written to a standby database. Therefore, some transactions may be lost if the primary database fails and you are unable to recover the redo records from the primary database. This mode provides the highest level of data protection that is possible without affecting the performance of the primary database.

See Section 5.7 for additional information about this SQL statement.

13.15 ALTER DATABASE START LOGICAL STANDBY APPLY

This statement is for logical standby databases only.

This statement starts log apply services on the logical standby database. The SQL statement syntax is:

```
ALTER DATABASE START LOGICAL STANDBY APPLY [INITIAL [scn-value] ] [NEW PRIMARY dblink];
```

Table 13–13 describes the keywords for this statement.

Table 13–13 Keywords for the START LOGICAL STANDBY APPLY Clause

Keyword	Description
`INITIAL [scn-value]`	Specify this keyword the first time you apply the logs to the logical standby database. It recovers the database to a transaction-consistent state immediately before the system change number (SCN) specified by integer.
`NEW PRIMARY dblink`	Starts log apply services after a database switchover has taken place. This statement ensures that all transactions in the archived redo logs are properly applied to the logical standby database. Specify this keyword after the `ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY` statement or when a logical standby database has completed processing logs from one primary database and a new database becomes the primary database. It uses the database link (dblink) provided in the command line to access tables on the primary database. The link must reference a privileged account that can read and lock the table on the primary database.

See Section 4.2 (step 14) for additional information about this SQL statement.

13.16 ALTER DATABASE {STOP|ABORT} LOGICAL STANDBY APPLY

This statement is for logical standby databases only.

This clause stops log apply services on a logical standby database. The SQL statement syntax is:

```
ALTER DATABASE { STOP | ABORT } LOGICAL STANDBY APPLY;
```

Table 13–14 describes the keywords for this statement.

Table 13–14 Keywords for the {STOP | ABORT} LOGICAL STANDBY APPLY Clause

Keyword	Description
`STOP`	Stops log apply services in an orderly fashion so that you can make changes to logical standby settings or perform planned maintenance. This clause is useful for refreshing materialized views or function-based indexes. Log transport services will continue to send archived redo logs to the logical standby database.

Table 13–14 (Cont.) Keywords for the {STOP | ABORT} LOGICAL STANDBY APPLY

Keyword	Description
ABORT	Stops log apply services immediately. If the DBA_LOGSTDBY_PARAMETERS view shows the TRANSACTION_CONSISTENCY option is set to READ_ONLY or NONE, an ABORT may leave transactions in an inconsistent manner. Only use the ABORT keyword when an immediate shutdown is necessary.

14

Views

This chapter describes the views that are used in a Data Guard environment. This is a subset of the views that are available for use in a database. This chapter contains the following sections:

- V$LOGFILE
- V$LOG_HISTORY
- V$LOGSTDBY (Logical Standby Databases Only)
- V$LOGSTDBY_STATS (Logical Standby Databases Only)
- V$MANAGED_STANDBY (Physical Standby Databases Only)
- V$STANDBY_LOG

About Views

An Oracle database contains a set of underlying views that are maintained by the server and accessible to the database administrator. These **fixed views** are also called **dynamic performance views** because they are continuously updated while a database is open and in use, and their contents relate primarily to performance.

Although these views appear to be regular database tables, they are not. These views provide data on internal disk structures and memory structures. You can select from these views, but you can never update or alter them.

Fixed view names are usually prefixed with either V$, or GV$, for example, `V$ARCHIVE_DEST` or `GV$ARCHIVE_DEST`. The views that are prefixed with DBA_ display all relevant information in the entire database. Standard dynamic performance views (V$ fixed views) store information on the local instance. In contrast, global dynamic performance views (GV$ fixed views) store information on all open instances. Each V$ fixed view has a corresponding GV$ fixed view. DBA_ views are intended only for administrators. They can be accessed only by users with the `SELECT_ANY_TABLE` privilege. (This privilege is assigned to the DBA role when the system is initially installed.)

In most cases, the information available in fixed views persists across instance shutdowns. However, certain fixed view information is reset when the instance is shut down; these views are specifically identified in this chapter.

For additional information about views, see *Oracle9i Database Reference.*

DBA_LOGSTDBY_EVENTS (Logical Standby Databases Only)

The DBA_LOGSTDBY_EVENTS view contains information about the activity of the logical standby database system. It can be used to determine the cause of failures that occur when log apply services apply redo logs. This view contains the following columns:

Column	Datatype	Description
EVENT_TIME	DATE	Time the event was logged
CURRENT_SCN	NUMBER	Change vector SCN for the event. If a failure occurred, examine this column to determine which archived log file contains the source of the failure (for example, an unsupported record).
COMMIT_SCN	NUMBER	SCN value for which the change was committed
XIDUSN	NUMBER	Transaction ID undo segment number
XIDSLT	NUMBER	Transaction ID slot number
XIDSQN	NUMBER	Transaction ID sequence number
EVENT	CLOB	The statement that was being processed when the failure occurred
STATUS_CODE	NUMBER	Status (or Oracle error code) belonging to the STATUS message
STATUS	VARCHAR2(2000)	Description of the current activity of the process or the reason why the apply operation stopped

DBA_LOGSTDBY_LOG (Logical Standby Databases Only)

The DBA_LOGSTDBY_LOG view shows the logs registered for a logical standby database. The view contains the following columns:

Column	Datatype	Description
THREAD#	NUMBER	Thread ID of the archived redo log. The THREAD number is 1 for a single instance. For Real Application Clusters, this column will contain different numbers.
SEQUENCE#	NUMBER	Sequence number of the archived redo log file
FIRST_CHANGE#	NUMBER	SCN of the current archived redo log
NEXT_CHANGE#	NUMBER	SCN of the next archived redo log
FIRST_TIME	DATE	Date of the current archived redo log
NEXT_TIME	DATE	Date of the next archived redo log
FILE_NAME	VARCHAR2(3)	Name of the archived redo log
TIMESTAMP	DATE	Time when the archived redo log was registered
DICT_BEGIN	VARCHAR2(3)	Value Y or N, where Y indicates that the beginning of the dictionary build is in this particular archived redo log
DICT_END	VARCHAR2(3)	Value Y or N, where Y indicates that the end of the dictionary build is in this particular archived redo log

Note: The SCN values in this view correlate to the SCN values in the DBA_LOGSTDBY_PROGRESS (Logical Standby Databases Only) view.

DBA_LOGSTDBY_NOT_UNIQUE (Logical Standby Databases Only)

The DBA_LOGSTDBY_NOT_UNIQUE view identifies tables that have no primary and no non-null unique indexes. Most of the tables displayed in this view are supported because their columns contain enough information to be maintained in a logical standby database. Some tables, however, cannot be supported because their columns do not contain the necessary information. Unsupported tables usually contain a column defined using an unsupported datatype. This view contains the following columns:

Column	Datatype	Description
OWNER	VARCHAR2(30)	Schema name
TABLE_NAME	VARCHAR2(30)	Name of the table
BAD_COLUMN	VARCHAR2(1)	This column contains a value of Y or N: ■ Y indicates the table column is defined using an unbounded datatype, such as LONG or BLOB. If two rows in the table match except in their LOB column, then the table cannot be maintained properly. Log apply services will attempt to maintain these tables, but you must ensure the application does not allow uniqueness only in the unbounded columns. ■ N indicates that enough column information is present to maintain the table in the logical standby database, but the log transport services and log apply services would run more efficiently if you added a primary key. You should consider adding a disabled RELY constraint to these tables.

DBA_LOGSTDBY_PARAMETERS (Logical Standby Databases Only)

The DBA_LOGSTDBY_PARAMETERS view contains the list of parameters used by the log apply services for logical standby databases. This view contains the following columns:

Column	Datatype	Description
NAME	VARCHAR2 (30)	Name of the parameter. Possible values are: ■ MAX_SGA - System global area (SGA) allocated for the log apply services cache in megabytes ■ MAX_SLAVES - Number of parallel query servers specifically reserved for log apply services ■ MAX_EVENTS_RECORDED - Number of events stored in the DBA_LOGSTDBY_EVENTS view ■ TRANSACTION_CONSISTENCY - Shows the level of transaction consistency maintained: FULL, READ_ONLY, or NONE ■ RECORD_SKIP_ERRORS - Indicates records that are skipped ■ RECORD_SKIP_DDL - Indicates skipped DDL statements ■ RECORD_APPLIED_DDL - Indicates applied DDL statements ■ FIRST_SCN - SCN at which log transport services will begin applying redo information ■ PRIMARY - Database ID of the database to which logs are being applied ■ LMNR_SID - LogMiner Session ID. This internal value indicates which LogMiner session is in use. ■ UNTIL_SCN - SCN value at which log apply services will shut down until all transactions have been applied ■ END_PRIMARY_SCN - During a switchover, this value indicates the last SCN applied by the new primary database from the old primary database ■ NEW_PRIMARY_SCN - During a switchover, this value indicates the starting SCN for the new primary database ■ COMPLETED_SESSION - Indicates that the log apply services session has concluded. The value will indicate SWITCHOVER or FAILOVER, as appropriate.
VALUE	VARCHAR2 (2000)	Value of the parameter

DBA_LOGSTDBY_PROGRESS (Logical Standby Databases Only)

The DBA_LOGSTDBY_PROGRESS view describes the progress of log apply services on the logical standby database. This view contains the following columns:

Column	Datatype	Description
APPLIED_SCN	NUMBER	Shows the newest SCN at which all changes have been applied. The values in the APPLIED_SCN and NEWEST_SCN columns will match if all available redo log data has been processed.
APPLIED_TIME	DATE	Estimate of the time and date of the APPLIED_SCN
READ_SCN	NUMBER	All log data greater than this SCN has been read and saved
READ_TIME	DATE	Estimate of the time and date of the READ_SCN
NEWEST_SCN	NUMBER	Most recent SCN available on the standby system. If no more logs are being transmitted to the standby database, changes could be applied to this SCN. The values in the APPLIED_SCN and NEWEST_SCN columns will match if all available redo log data has been processed.
NEWEST_TIME	DATE	Estimate of the time and date of the NEWEST_SCN

Note: The SCN values in this view correlate to the SCN values in the DBA_LOGSTDBY_LOG (Logical Standby Databases Only) view.

DBA_LOGSTDBY_SKIP (Logical Standby Databases Only)

The DBA_LOGSTDBY_SKIP view lists the tables that will be skipped by log apply services. The DBA_LOGSTDBY_SKIP view contains the following columns:

Column	Datatype	Description
ERROR	BOOLEAN	Indicates if the statement should be skipped or just return errors for the statement
STATEMENT_OPT	VARCHAR(30)	Specifies the type of statement that should be skipped. It must be one of the SYSTEM_AUDIT statement options.
SCHEMA	VARCHAR(30)	Name of the schema under which this skip option should be used
NAME	VARCHAR(30)	Name of the option under which this skip option should be used
PROC	VARCHAR(98)	Name of a stored procedure that will be executed when processing the skip option

DBA_LOGSTDBY_SKIP_TRANSACTION (Logical Standby Databases Only)

The DBA_LOGSTDBY_SKIP_TRANSACTION view lists the skip settings chosen. This view contains the following columns:

Column	Datatype	Description
XIDUSN	NUMBER	Transaction ID undo segment number
XIDSLT	NUMBER	Transaction ID slot number
XIDSQN	NUMBER	Transaction ID sequence number

DBA_LOGSTDBY_UNSUPPORTED (Logical Standby Databases Only)

The DBA_LOGSTDBY_UNSUPPORTED view identifies the schemas and tables (and columns in those tables) that contain unsupported datatypes. Use this view when you are preparing to create a logical standby database. This view contains the following columns:

Column	Datatype	Description
OWNER	VARCHAR2(30)	Schema name of the unsupported table
TABLE_NAME	VARCHAR2(30)	Name of the unsupported table
COLUMN_NAME	VARCHAR2(30)	Name of the unsupported column
DATA_TYPE	VARCHAR2(106)	Datatype of the unsupported column

V$ARCHIVE_DEST

The V$ARCHIVE_DEST view describes, for the current instance, all the archived redo log destinations, their current value, mode, and status.

Note: The information in this view does not persist across an instance shutdown.

The V$ARCHIVE_DEST view contains the following columns:

Column	Description
DEST_ID	Identifies the log archive destination parameter
STATUS	Identifies the current status of the destination. Possible values are: ■ VALID - initialized and available ■ INACTIVE - no destination information ■ DEFERRED - manually disabled by the user ■ ERROR - error during open or copy ■ DISABLED - disabled after error ■ BAD PARAM - parameter has errors ■ ALTERNATE - destination is in an alternate state ■ FULL - exceeded quota size for the destination
BINDING	Specifies how failure will affect the archival operation. Possible values are: ■ OPTIONAL - successful archival operation is not required ■ MANDATORY - successful archival operation is required
NAME_SPACE	Identifies the scope of parameter setting. Possible values are: ■ SYSTEM - system definition ■ SESSION - session definition
TARGET	Specifies whether the archive destination is local or remote to the primary database. Possible values are: ■ PRIMARY - local ■ STANDBY - remote

Column	Description
ARCHIVER	Identifies the archiver process relative to the database where the query is issued. Possible values are: ■ ARC*n* ■ FOREGROUND ■ LGWR ■ RFS
SCHEDULE	Indicates whether the archiving of this destination is INACTIVE, PENDING, ACTIVE, or LATENT
DESTINATION	Displays the location where the archived redo logs are to be archived
LOG_SEQUENCE	Identifies the sequence number of the last archived redo log to be archived
REOPEN_SECS	Identifies the retry time, in seconds, after error
DELAY_MINS	Identifies the delay interval, in minutes, before the archived redo log is automatically applied to a standby database
PROCESS	Identifies the archiver process relative to the primary database, even if the query is issued on the standby database. Possible values are: ■ ARC*n* ■ FOREGROUND ■ LGWR
REGISTER	Indicates whether the archived redo log is registered in the remote destination control file. If the archived redo log is registered, it is available to the managed recovery operation. Possible values are: ■ YES ■ NO
FAIL_DATE	Indicates the date and time of error
FAIL_SEQUENCE	Indicates the sequence number of the archived redo log being archived when the last error occurred
FAIL_BLOCK	Indicates the block number of the archived redo log being archived when the last error occurred
FAILURE_COUNT	Identifies the current number of consecutive archival operation failures that have occurred for the destination
MAX_FAILURE	Allows you to control the number of times log transport services will attempt to reestablish communication and resume archival operations with a failed destination
ERROR	Displays the error text

Column	Description
ALTERNATE	Identifies the alternate destination, if any
DEPENDENCY	Identifies the dependent archive destination, if any
REMOTE_TEMPLATE	Displays the template to be used to derive the location to be recorded
QUOTA_SIZE	Identifies the destination quotas, expressed in bytes
QUOTA_USED	Identifies the size of all archived redo logs currently residing on the specified destination
MOUNTID	Identifies the instance mount identifier
AFFIRM	Displays disk I/O mode
ASYNC_BLOCKS	Specifies the number of blocks for the ASYNC attribute
TRANSMIT_MODE	Displays network transmission mode. Possible values are: ■ SYNC=PARALLEL ■ SYNC=NOPARALLEL ■ ASYNC[=blocks]
TYPE	Indicates whether the archived log destination definition is PUBLIC or PRIVATE. Only PUBLIC destinations can be modified at runtime using the ALTER SYSTEM SET or ALTER SESSION SET statements. By default, all archived log destinations are PUBLIC.
NET_TIMEOUT	Specifies the number of seconds the log writer process will wait for status from the network server of a network operation issued by the log writer process

V$ARCHIVE_DEST_STATUS

The V$ARCHIVE_DEST_STATUS view displays runtime and configuration information for the archived redo log destinations.

Note: The information in this view does not persist across an instance shutdown.

The V$ARCHIVE_DEST_STATUS view contains the following columns:

Column	Description
DEST_ID	Identifies the log archive destination parameter
STATUS	Identifies the current status of the destination. Possible values are: ■ VALID - initialized and available ■ INACTIVE - no destination information ■ DEFERRED - manually disabled by the user ■ ERROR - error during open or copy ■ DISABLED - disabled after error ■ BAD PARAM - parameter has errors ■ ALTERNATE - destination is in an alternate state ■ FULL - exceeded quota size for the destination
TYPE	Identifies the type of archival destination database. Possible values are: ■ LOCAL - local to primary instance ■ PHYSICAL - physical standby database ■ CROSS-INSTANCE - an instance of the primary database
DATABASE_MODE	Identifies the current mode of the archival destination database. Possible values are: ■ STARTED - instance started, not mounted ■ MOUNTED - mounted ■ MOUNTED-STANDBY - mounted standby ■ OPEN - open read / write ■ OPEN_READ-ONLY - open read-only

Column	Description
RECOVERY_MODE	Identifies the current mode of media recovery at the archival destination database. Possible values are: ■ IDLE - managed recovery is not active ■ MANUAL - manual media recovery is active ■ MANAGED - managed recovery is active
DESTINATION	Displays the location where the archived redo logs are to be archived
ARCHIVED_THREAD#	Identifies the thread number of the most recent archived redo log received at the destination
ARCHIVED_SEQ#	Identifies the log sequence number of the most recent archived redo log received at the destination
APPLIED_THREAD#	Identifies the thread number of the most recent applied redo log received at the destination
APPLIED_SEQ#	Identifies the log sequence number of the most recent applied redo log received at the destination
ERROR	Displays the error text
STANDBY_LOGFILE_COUNT	Indicates the total number of standby redo logs created on the standby database
STANDBY_LOGFILE_ACTIVE	Indicates the total number of standby redo logs on the standby database that are active and contain primary database online redo log information
PROTECTION_MODE	Indicates whether the database is protected. Possible values are: ■ MAXIMUM PROTECTED ■ MAXIMUM AVAILABILITY ■ RESYNCHRONIZATION ■ MAXIMUM PERFORMANCE ■ UNPROTECTED
SRL	Indicates the use of standby redo logs on the standby database. Possible values are: ■ YES ■ NO

V$ARCHIVE_GAP

The V$ARCHIVE_GAP view displays information to help you identify an archive gap. The V$ARCHIVE_GAP view contains the following columns:

Column	Description
THREAD#	Specifies the thread number
LOW_SEQUENCE#	Specifies the low number of the log file
HIGH_SEQUENCE#	Specifies the high number of the log file

V$ARCHIVED_LOG

The V$ARCHIVED_LOG view displays archived redo log information from the control file, including archived log names. This view contains the following columns:

Column	Description
RECID	Archived log record ID
STAMP	Archived log record stamp
NAME	Archived log filename. If set to NULL, the log file was cleared before it was archived.
DEST_ID	The original destination from which the archived log was generated. Value is 0 if the destination identifier is not available.
THREAD#	Redo thread number
SEQUENCE#	Redo log sequence number
RESETLOGS_CHANGE#	Resetlogs change number of the database when this log was written
RESETLOGS_TIME	Resetlogs time of the database when this log was written
FIRST_CHANGE#	First change number in the archived log
FIRST_TIME	Timestamp of the first change
NEXT_CHANGE#	First change in the next log
NEXT_TIME	Timestamp of the next change
BLOCKS	Size of the archived log in blocks
BLOCK_SIZE	Redo log block size. This is the logical block size of the archived log, which is the same as the logical block size of the online log from which this archived log was copied. The online log logical block size is a platform-specific value that is not adjustable by the user.
CREATOR	Identifies the creator of the archived log
REGISTRAR	Identifies the registrar of the entry
STANDBY_DEST	Indicates if the entry is an archived log destination

Column	Description
ARCHIVED	Indicates that the online redo log was archived or that RMAN only inspected the log and created a record for future application of redo logs during recovery
APPLIED	Indicates whether the archived log has been applied to its corresponding standby database
DELETED	Specifies whether an RMAN DELETE command has physically deleted the archived log file from disk, as well as logically removing it from the control file of the **target database** and from the **recovery catalog**
STATUS	The status of this archived log. Possible values are: ■ A - Available ■ D - Deleted ■ U - Unavailable ■ X - Expired
COMPLETION_TIME	Indicates the time when the archiving completed
DICTIONARY_BEGIN	Indicates whether this log contains the start of a LogMiner dictionary
DICTIONARY_END	Indicates whether this log contains the end of a LogMiner dictionary
BACKUP_COUNT	Indicates the number of times this file has been backed up. Values range from 0 to 15. If the file has been backed up more than 15 times, the value remains 15.
END_OF_REDO	Indicates whether this archived redo log contains the end of all redo information from the primary database. Values are YES and NO.
ARCHIVAL_THREAD#	Indicates the redo thread number of the instance that performed the archival operation. This column differs from the THREAD# column only when a closed thread is archived by another instance.
ACTIVATION#	Indicates the number assigned to the database instantiation.

V$DATABASE

The V$DATABASE view provides database information from the control file. This view contains the following columns:

Column	Description
DBID	The database identifier that is calculated when the database is created. This identifier is stored in all file headers.
NAME	Name of the database
CREATED	Creation date
RESETLOGS_CHANGE#	Change number at open resetlogs
RESETLOGS_TIME	Timestamp of open resetlogs
PRIOR_RESETLOGS_ CHANGE#	Change number at prior resetlogs
PRIOR_RESETLOGS_ TIME	Timestamp of prior resetlogs
LOG_MODE	Archive log mode
CHECKPOINT_CHANGE#	Last SCN checkpointed
ARCHIVE_CHANGE#	Last SCN archived
CONTROLFILE_TYPE	The type of control file. Possible values are: ■ STANDBY - indicates database is in standby mode ■ LOGICAL - indicates the database is a logical standby database ■ CLONE - indicates a clone database ■ BACKUP | CREATED - indicates database is being recovered using a backup or created control file ■ CURRENT - indicates the database is available for general use
CONTROLFILE_CREATED	Control file creation timestamp
CONTROLFILE_ SEQUENCE#	Control file sequence number incremented by control file transactions
CONTROLFILE_CHANGE#	Last change number in the backup control file. Set to NULL if the control file is not a backup.

Column	Description
CONTROLFILE_TIME	Last timestamp in the backup control file. Set to NULL if the control file is not a backup.
OPEN_RESETLOGS	Indicates whether the next database open allows or requires the resetlogs option
VERSION_TIME	The version time
OPEN_MODE	Open mode information
PROTECTION_MODE	Indicates whether the database is protected. Possible values are: ■ MAXIMUM PROTECTED ■ MAXIMUM AVAILABILITY ■ RESYNCHRONIZATION ■ MAXIMUM PERFORMANCE ■ UNPROTECTED
PROTECTION_LEVEL	Displays the aggregated protection mode currently in effect on the primary or standby database. Possible values are: ■ MAXIMUM PROTECTION ■ MAXIMUM AVAILABILITY ■ RESYNCHRONIZATION ■ MAXIMUM PERFORMANCE ■ UNPROTECTED
REMOTE_ARCHIVE	The value of the REMOTE_ARCHIVE_ENABLE initialization parameter. Possible values are: ■ true ■ false ■ send ■ receive
ACTIVATION#	Number assigned to the database instantiation.
DATABASE_ROLE	Current role of the database; either primary or standby
ARCHIVELOG_CHANGE#	Highest NEXT_CHANGE# (from the V$ARCHIVED_LOG view) for an archived log

Column	Description
SWITCHOVER_STATUS	Specifies whether switchover is allowed. Possible values are: ■ NOT ALLOWED - Either this is a standby database and the primary database has not been switched first or this is a primary database and there are no standby databases. ■ SESSIONS ACTIVE - Indicates that there are active SQL sessions attached to the primary or standby database that need to be disconnected before the switchover operation is permitted. Query the V$SESSION view to identify the specific processes that need to be terminated. ■ SWITCHOVER PENDING - This is a standby database and the primary database switchover request has been received but not processed. ■ SWITCHOVER LATENT - The switchover was in pending mode, but did not complete and went back to the primary database. ■ TO PRIMARY - This is a standby database and is allowed to switch over to a primary database. ■ TO STANDBY - This is a primary database and is allowed to switch over to a standby database. ■ RECOVERY NEEDED - This is a standby database that has not received the switchover request.
GUARD_STATUS	Protects data from being changed. Possible values are: ■ ALL - Indicates all users other than SYS are prevented from making changes to any data in the database. ■ STANDBY - Indicates all users other than SYS are prevented from making changes to any database object being maintained by logical standby. ■ NONE - Indicates normal security for all data in the database.
SUPPLEMENTAL_LOG_DATA_MIN	Ensures that LogMiner will have sufficient information to support chained rows and various storage arrangements such as cluster tables. See *Oracle9i SQL Reference* for additional information about the ALTER DATABASE ADD SUPPLEMENTAL LOG DATA statement.
SUPPLEMENTAL_LOG_DATA_PK	For all tables with a primary key, ensures that all columns of the primary key are placed into the redo log whenever an update operation is performed. See *Oracle9i SQL Reference* for additional information about the ALTER DATABASE ADD SUPPLEMENTAL LOG DATA statement.
SUPPLEMENTAL_LOG_DATA_UI	For all tables with a unique key, ensures that if any unique key columns are modified, all other columns belonging to the unique key are also placed into the redo log. See *Oracle9i SQL Reference* for additional information about the ALTER DATABASE ADD SUPPLEMENTAL LOG DATA statement.

Column	Description
FORCE_LOGGING	Redo generation is forced even for NOLOGGING operations. Possible values are: ■ YES ■ NO
DATAGUARD_BROKER	Indicates whether the Data Guard configuration is being managed by the broker. Possible values are: ■ ENABLED indicates the configuration is under the control of the broker ■ DISABLED indicates the configuration is not under the control of the broker

V$DATAFILE

The V$DATAFILE view provides datafile information from the control file. This view contains the following columns:

Column	Description
FILE#	File identification number
CREATION_CHANGE#	Change number at which the datafile was created
CREATION_TIME	Timestamp of the datafile creation
TS#	Tablespace number
RFILE#	Tablespace relative datafile number
STATUS	Type of file (system or user) and its status. Possible values are: ■ OFFLINE - cannot be written to ■ ONLINE - can be written to ■ SYSTEM - system datafile ■ RECOVER - needs recovery ■ SYSOFF - offline system
ENABLED	Describes how accessible the file is from SQL. Possible values are: ■ DISABLED - no SQL access allowed ■ READ ONLY - no SQL updates allowed ■ READ WRITE - full access allowed
CHECKPOINT_CHANGE#	SCN at last checkpoint
CHECKPOINT_TIME	Timestamp of the last checkpoint
UNRECOVERABLE_ CHANGE#	Last unrecoverable change number made to this datafile. This column is always updated when an unrecoverable operation completes.
UNRECOVERABLE_TIME	Timestamp of the last unrecoverable change
LAST_CHANGE#	Last change number made to this datafile. Set to NULL if the datafile is being changed.
LAST_TIME	Timestamp of the last change
OFFLINE_CHANGE#	Offline change number of the last offline range. This column is updated only when the datafile is brought online.

Column	Description
ONLINE_CHANGE#	Online change number of the last offline range
ONLINE_TIME	Online timestamp of the last offline range
BYTES	Current datafile size in bytes; 0 if inaccessible
BLOCKS	Current datafile size in blocks; 0 if inaccessible
CREATE_BYTES	Size when created, in bytes
BLOCK_SIZE	Block size of the datafile
NAME	Datafile name
PLUGGED_IN	Describes whether the tablespace is plugged in. The value is 1 if the tablespace is plugged in and has not been made read / write; 0 if not.
BLOCK1_OFFSET	The offset from the beginning of the file to where the Oracle generic information begins. The exact length of the file can be computed as follows: BYTES + BLOCK1_OFFSET
AUX_NAME	The auxiliary name that has been set for this file

V$DATAGUARD_STATUS

The V$DATAGUARD_STATUS view displays and logs events that would typically be triggered by any message to the alert log or server process trace files.

Note: The information in this view does not persist across an instance shutdown.

The V$DATAGUARD_STATUS view contains the following columns:

Column	Description
INST_ID	The ID of the instance encountering the event. This column is present in the GV$DATAGUARD_STATUS view and not in the V$DATAGUARD_STATUS view.
FACILITY	Facility that encountered the event. Possible values are: ■ CRASH RECOVERY ■ LOG TRANSPORT SERVICES ■ LOG APPLY SERVICES ■ ROLE MANAGEMENT SERVICES ■ REMOTE FILE SERVER ■ FETCH ARCHIVE LOG ■ DATA GUARD ■ NETWORK SERVICES
SEVERITY	The severity of the event. Possible values are: ■ INFORMATIONAL - informational message ■ WARNING - warning message ■ ERROR - indicates the process has failed ■ FATAL ■ CONTROL - an expected change in state, such as the start or completion of an archival, log recovery, or switchover operation
DEST_ID	The destination ID number to which the event pertains. If the event does not pertain to a particular destination, the value is 0.
MESSAGE_NUM	A chronologically increasing number giving each event a unique number

Column	Description
ERROR_CODE	The error ID pertaining to the event
CALLOUT	Indicates whether the current entry is a callout event. Possible values are: ■ YES ■ NO A YES value indicates that this event may require the DBA to perform some action. Examine the ERROR_CODE and MESSAGE columns for more information. A NO value generally corresponds to an INFORMATIONAL or WARNING event that does not require any action by the DBA.
TIMESTAMP	The date and time when the entry was created.
MESSAGE	A text message describing the event

V$LOG

The V$LOG view contains log file information from the online redo logs. This view contains the following columns:

Column	Description
GROUP#	Log group number
THREAD#	Log thread number
SEQUENCE#	Log sequence number
BYTES	Size of the log in bytes
MEMBERS	Number of members in the log group
ARCHIVED	Archive status
STATUS	Indicates the log status. Possible values are: ■ UNUSED - The online redo log has never been written to. This is the status of a redo log that was just added, or just after specifying a RESETLOGS option when it is not the current redo log. ■ CURRENT - This is the current redo log. This implies that the redo log is active. The redo log could be open or closed. ■ ACTIVE - The log is active but is not the current log. It is needed for failure recovery. It may be in use for block recovery. It might or might not be archived. ■ CLEARING - The log is being re-created as an empty log after an ALTER DATABASE CLEAR LOGFILE statement. After the log is cleared, the status changes to UNUSED. ■ CLEARING_CURRENT - The current log is being cleared of a closed thread. The log can stay in this status if there is some failure in the switch, such as an I/O error writing the new log header. ■ INACTIVE - The log is no longer needed for instance recovery. It may be in use for managed recovery. It might or might not be archived. ■ INVALIDATED - Archived the current redo log without a log switch
FIRST_CHANGE#	Lowest SCN in the log
FIRST_TIME	Time of first SCN in the log

V$LOGFILE

The V$LOGFILE view contains information about the online redo logs. This view contains the following columns:

Column	Description
GROUP#	Redo log group identifier number
STATUS	Status of this log member. Possible values are: ■ INVALID - file is inaccessible ■ STALE - contents are incomplete ■ DELETED - file is no longer used ■ blank (no value listed) - file is in use
MEMBER	Redo log member name
TYPE	Specifies whether this is a standby log or an online log. Possible values are: ■ STANDBY ■ ONLINE

V$LOG_HISTORY

The V$LOG_HISTORY view contains log history information from the control file. This view contains the following columns:

Column	Description
RECID	Control file record ID
STAMP	Control file record stamp
THREAD#	Thread number of the archived log
SEQUENCE#	Sequence number of the archived log
FIRST_CHANGE#	Lowest SCN in the log
FIRST_TIME	Time of first entry (lowest SCN) in the log
NEXT_CHANGE#	Highest SCN in the log

V$LOGSTDBY (Logical Standby Databases Only)

The V$LOGSTDBY view provides dynamic information about what is happening to log apply services. This view is very helpful when you are diagnosing performance problems during the logical application of archived redo logs to the standby database, and it can be helpful for other problems. The V$LOGSTDBY view contains the following columns:

Column	Datatype	Description
SERIAL#	NUMBER	Contains the SQL session serial number. This data is used when joining this view with V$SESSION and V$PX_SESSION views.
LOGSTDBY_ID	NUMBER	Contains the parallel query slave ID
PID	VARCHAR2(9)	Contains the process ID
TYPE	VARCHAR2(30)	Indicates the task being performed by the process: COORDINATOR, APPLIER, ANALYZER, READER, PREPARER, BUILDER
STATUS_CODE	NUMBER	Contains the status number (or Oracle error code) belonging to the STATUS message
STATUS	VARCHAR2(256)	Description of the current activity of the process
HIGH_SCN	NUMBER	Contains the highest SCN seen by the process. This column is used to confirm the progress of the individual process.

V$LOGSTDBY_STATS (Logical Standby Databases Only)

The V$LOGSTDBY_STATS view displays LogMiner statistics, current state, and status information for a logical standby database during SQL apply operations. If log apply services are not running, the values for the statistics are cleared. This view contains the following columns:

Column	Datatype	Description
NAME	VARCHAR2(64)	Name of the statistic, state, or status: **Note:** Many of the following statistics are subject to change or deletion; programmers should write application code to tolerate missing or extra statistics. ■ Number of preparers ■ Number of appliers ■ Maximum SGA for LCR cache ■ Parallel servers in use ■ Transaction consistency ■ Coodinator state ■ Transactions scheduled ■ Transactions applied ■ Preparer memory allocation failures ■ Builder memory allocation failures ■ Attempts to handle low memory ■ Successful low memory recovery ■ Memory spills avoided ■ Rollback attempts ■ Successful rollbacks ■ Memory spill attempts ■ Successful memory spills ■ Preparer ignored memory low water mark ■ Builder ignored memory low water mark ■ Mining resumed
VALUE	VARCHAR2(64)	The value of the statistic or state information

V$MANAGED_STANDBY (Physical Standby Databases Only)

The V$MANAGED_STANDBY view displays current and status information for Oracle database server processes related to the Data Guard environment. Use this view to query physical standby database only. The information in this view does not persist after an instance shutdown. The V$MANAGED_STANDBY view contains the columns shown in the following table:

Column	Description
PROCESS	Type of process whose information is being reported. Possible values are: ■ ARCH - archiver process ■ RFS - remote file server ■ MRP0 - detached recovery server process ■ MR(fg) - foreground recovery session
PID	Operating system identifier of the process

Column	Description
STATUS	Current process status. Possible values are: ■ UNUSED - no active process ■ ALLOCATED - process is active but not currently connected to a primary database ■ CONNECTED - network connection is established to a primary database ■ ATTACHED - process is attached to, and communicating with, a primary database ■ IDLE - process is not performing any activities ■ ERROR - process has failed ■ OPENING - process is opening the archived redo log ■ CLOSING - process has completed the archival operation and is closing the archived redo log ■ WRITING - process is actively writing archived redo log data ■ RECEIVING - process is receiving network communication ■ ANNOUNCING - process is announcing the existence of a potential dependent archived redo log ■ REGISTERING - process is registering the existence of a completed dependent archived redo log ■ WAIT_FOR_LOG - process is waiting for the archived redo log to be completed ■ WAIT_FOR_GAP - process is waiting for the archive gap to be resolved ■ APPLYING_LOG - process is applying the archived redo log to the standby database
CLIENT_ PROCESS	Identifies the corresponding primary database process. Possible values are: ■ ARCHIVAL - foreground (manual) archival process (SQL) ■ ARCH - background ARC*n* process ■ LGWR - background LGWR process
CLIENT_PID	Operating system identifier of the client process
CLIENT_DBID	Database identifier of the primary database
GROUP#	Standby redo log group
THREAD#	Archived redo log thread number
SEQUENCE#	Archived redo log sequence number
BLOCK#	Last processed archived redo log block number
BLOCKS	Size of the archived redo log in 512-byte blocks
DELAY_MINS	Archived redo log delay interval in minutes

Column	Description
KNOWN_ AGENTS	Total number of standby database agents processing an archived redo log
ACTIVE_ AGENTS	Number of standby database agents actively processing an archived redo log

V$STANDBY_LOG

The V$STANDBY_LOG view contains the following columns:

Column	Description
GROUP#	Log group number
THREAD#	Log thread number
SEQUENCE#	Log sequence number
BYTES	Size of the log in bytes
USED	Number of bytes used in the log
ARCHIVED	Archive status
STATUS	Indicates the log status. Possible values are: ■ UNUSED - The online redo log has never been written to. This is the status of a redo log that was just added, or just after specifying a RESETLOGS option when it is not the current redo log. ■ CURRENT - This is the current redo log. This implies that the redo log is active. The redo log could be open or closed. ■ ACTIVE - The log is active but is not the current log. It is needed for failure recovery. It may be in use for block recovery. It might or might not be archived. ■ CLEARING - The log is being re-created as an empty log after an ALTER DATABASE CLEAR LOGFILE statement. After the log is cleared, the status changes to UNUSED. ■ CLEARING_CURRENT - The current log is being cleared of a closed thread. The log can stay in this status if there is some failure in the switch, such as an I/O error writing the new log header. ■ INACTIVE - The log is no longer needed for instance recovery. It may be in use for managed recovery. It might or might not be archived. ■ INVALIDATED - Archived the current redo log without a log switch.
FIRST_CHANGE#	Lowest SCN in the log
FIRST_TIME	Time of first SCN in the log
LAST_CHANGE#	Last change number made to this datafile. Set to NULL if the datafile is being changed.
LAST_TIME	Timestamp of the last change

Part III

Appendixes and Glossary

This part contains the following:

A

Troubleshooting the Standby Database

This appendix provides help troubleshooting a standby database. This appendix contains the following sections:

- Problems During Standby Database Preparation
- Problems Switching Over to a Standby Database
- What to Do If SQL Apply Operations to a Logical Standby Database Stop

A.1 Problems During Standby Database Preparation

If you encounter a problem during standby database preparation, it will probably be one of the following:

- The Standby Archive Destination Is Not Defined Properly
- The Standby Site Does Not Receive Logs Archived by the Primary Database
- You Cannot Mount the Physical Standby Database

A.1.1 The Standby Archive Destination Is Not Defined Properly

If the STANDBY_ARCHIVE_DEST initialization parameter is not defined as a valid directory name on the standby site, the Oracle database server will not be able to determine the directory in which to store the archived redo logs. Check the DESTINATION and ERROR columns in the V$ARCHIVE_DEST view. For example, enter:

```
SQL> SELECT DESTINATION, ERROR FROM V$ARCHIVE_DEST;
```

Make sure the destination is valid.

A.1.2 The Standby Site Does Not Receive Logs Archived by the Primary Database

If the standby site is not receiving the logs, the first thing you should do is obtain information about the archiving status of the primary database by querying the V$ARCHIVE_DEST view. Check especially for error messages. For example, enter:

```
SQL> SELECT DEST_ID "ID",
  2> STATUS "DB_status",
  3> DESTINATION "Archive_dest",
  4> ERROR "Error"
  5> FROM V$ARCHIVE_DEST;

ID DB_status Archive_dest                    Error
-- --------- ------------------------------
-------------------------------------
 1  VALID     /vobs/oracle/work/arc_dest/arc
 2  ERROR     standby1          ORA-16012: Archivelog standby database
identifier mismatch
 3  INACTIVE
 4  INACTIVE
 5  INACTIVE
5 rows selected.
```

If the output of the query does not help you, check the following list of possible issues. If any of the following conditions exist, the primary database will fail to archive to the standby site:

- The service name for the standby instance is not configured correctly in the tnsnames.ora file at the primary site.
- The service name listed in the LOG_ARCHIVE_DEST_*n* parameter of the primary initialization parameter file is incorrect.
- The LOG_ARCHIVE_DEST_STATE_*n* parameter specifying the state of the standby archiving destination has the value DEFER.
- The listener.ora file has not been configured correctly at the standby site.
- The listener is not started.
- The standby instance is not started.
- You have added a standby archiving destination to the primary initialization parameter file, but have not yet enabled the change.
- You used an invalid backup as the basis for the standby database (for example, you used a backup from the wrong database, or did not create the standby control file using the correct method).

A.1.3 You Cannot Mount the Physical Standby Database

If any of the following conditions exist, you cannot mount the physical standby database:

- The standby instance is not started in `NOMOUNT` mode. You must first start the instance and *then* mount the database.
- The standby control file was not created with the `ALTER DATABASE CREATE STANDBY CONTROLFILE ...` statement or RMAN. You cannot use the following types of control file backups:
  - An operating system-created backup
  - A backup created using an `ALTER DATABASE` statement *without* the `STANDBY` option

A.2 Problems Switching Over to a Standby Database

If you encounter a problem switching over from a primary database to a standby database, it will probably be one of the following:

- Switchover Fails
- Startup of Second Physical Standby Database Fails
- Archived Redo Logs Are Not Applied to the Standby Database After Switchover
- Switchover Fails in a Real Application Clusters Configuration
- Switchover Fails When SQL Sessions Are Active

A.2.1 Switchover Fails

`ALTER DATABASE COMMIT TO SWITCHOVER` failed with ORA-01093 error "Alter database close only permitted with no sessions connected."

This error occurs because the `COMMIT TO SWITCHOVER` statement implicitly closed the database and, if there are any other user sessions connected to the database, the close fails.

Action: Make sure all user sessions are disconnected from the database. You can query the `V$SESSION` fixed view to see what sessions are still around. For example:

```
SQL> SELECT SID, PROCESS, PROGRAM FROM V$SESSION;

       SID PROCESS   PROGRAM
---------- --------- -------------------------------------------------
         1 26900     oracle@dbuser-sun (PMON)
         2 26902     oracle@dbuser-sun (DBW0)
         3 26904     oracle@dbuser-sun (LGWR)
         4 26906     oracle@dbuser-sun (CKPT)
         5 26908     oracle@dbuser-sun (SMON)
         6 26910     oracle@dbuser-sun (RECO)
         7 26912     oracle@dbuser-sun (ARC0)
         8 26897     sqlplus@dbuser-sun (TNS V1-V3)
        11 26917     sqlplus@dbuser-sun (TNS V1-V3)

9 rows selected.
```

In the previous example, the first seven sessions are all server background processes. Among the two SQL*Plus sessions, one is the current SQL*Plus session issuing the query, and the other is an extra session that should be disconnected before the switchover operation.

A.2.2 Startup of Second Physical Standby Database Fails

Suppose the standby database and the primary database reside on the same site. After both the `ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY` and the `ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY` statements are successfully executed, shut down and restart the physical standby database and the primary database. However, the startup of the second database fails with ORA-01102 error "cannot mount database in EXCLUSIVE mode."

This could happen during the switchover if you forget to set the `LOCK_NAME_SPACE` parameter in the initialization parameter file that is used by the standby database (that is, the original primary database). If the `LOCK_NAME_SPACE` parameter of the standby database is not set, the standby and the primary databases both use the same mount lock and cause the ORA-01102 error during the startup of the second database.

Action: Add `LOCK_NAME_SPACE=`*`unique_lock_name`* to the initialization parameter file used by the physical standby database and shut down and restart both the standby and the primary databases.

A.2.3 Archived Redo Logs Are Not Applied to the Standby Database After Switchover

The archived redo logs are not applied to the standby database after the switchover.

This may happen because some environment or initialization parameters have not been properly set after the switchover.

Action:

- Check the `tnsnames.ora` file at the primary site and the `listener.ora` file at the standby site. There should be entries for a listener at the standby site and a corresponding tnsname at the primary site.
- Start the listener at the standby site if it has not been started.
- Check if the `LOG_ARCHIVE_DEST_`*n* initialization parameter has been set to properly archive logs from the primary site to standby site. For example, query the `V$ARCHIVE_DEST` fixed view at the primary site as follows:

  ```
  SQL> SELECT DEST_ID, STATUS, DESTINATION FROM V$ARCHIVE_DEST;
  ```

  If you do not see an entry corresponding to the standby site, you need to set `LOG_ARCHIVE_DEST_`*n* and `LOG_ARCHIVE_DEST_STATE_`*n* initialization parameters.
- Set the `STANDBY_ARCHIVE_DEST` and `LOG_ARCHIVE_FORMAT` initialization parameters correctly at the standby site so that the archived redo logs are applied to the desired location.
- At the standby site, set the `DB_FILE_NAME_CONVERT` and `LOG_FILE_NAME_CONVERT` initialization parameters. Set the `STANDBY_FILE_MANAGEMENT` initialization parameter to `auto` if you want the standby site to automatically add new datafiles that are created at the primary site.

A.2.4 Switchover Fails in a Real Application Clusters Configuration

When your database is using Real Application Clusters, active instances prevent a switchover from being performed. When other instances are active, an attempt to switch over fails with the following error message:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY;
ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY *
ORA-01105: mount is incompatible with mounts by other instances
```

Action: Query the GV$INSTANCE view as follows to determine which instances are causing the problem:

```
SQL> SELECT INSTANCE_NAME, HOST_NAME FROM GV$INSTANCE
  2> WHERE INST_ID <> (SELECT INSTANCE_NUMBER FROM V$INSTANCE);
INSTANCE_NAME HOST_NAME
------------- ---------
INST2         standby2
```

In the previous example, the identified instance must be manually shut down before the switchover can proceed. You can connect to the identified instance from your instance and issue the SHUTDOWN statement remotely, for example:

```
SQL> CONNECT SYS/CHANGE_ON_INSTALL@standby2 AS SYSDBA
SQL> SHUTDOWN;
SQL> EXIT
```

A.2.5 Switchover Fails When SQL Sessions Are Active

If you do not include the WITH SESSION SHUTDOWN clause as a part of the ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY statement, active SQL sessions may prevent a switchover from being processed. Active SQL sessions may include other Oracle processes.

When sessions are active, an attempt to switch over fails with the following error message:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY;
ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY *
ORA-01093: ALTER DATABASE CLOSE only permitted with no sessions connected
```

Action: Query the V$SESSION view to determine which processes are causing the error. For example:

```
SQL> SELECT SID, PROCESS, PROGRAM FROM V$SESSION
  2> WHERE TYPE = 'USER'
  3>  AND SID <> (SELECT DISTINCT SID FROM V$MYSTAT);
SID        PROCESS   PROGRAM
---------  --------  ------------------------------------------------
        7      3537  oracle@nhclone2 (CJQ0)
       10
       14
       16
       19
       21
 6 rows selected.
```

In the previous example, the JOB_QUEUE_PROCESSES parameter corresponds to the CJQ0 process entry. Because the job queue process is a user process, it is counted as a SQL session that prevents switchover from taking place. The entries with no process or program information are threads started by the job queue controller.

Verify that the JOB_QUEUE_PROCESSES parameter is set using the following SQL statement:

```
SQL> SHOW PARAMETER JOB_QUEUE_PROCESSES;
NAME                                 TYPE      VALUE
------------------------------------ -------   --------------------
job_queue_processes                  integer   5
```

Then set the parameter to 0. For example:

```
SQL> ALTER SYSTEM SET JOB_QUEUE_PROCESSES=0;
Statement processed.
```

Because JOB_QUEUE_PROCESSES is a dynamic parameter, you can change the value and have the change take effect immediately without having to restart the instance. You can now retry the switchover procedure.

Do not modify the parameter in your initialization parameter file. After you shut down the instance and restart it after switchover has completed, the parameter will be reset to the original value. This applies to both primary and physical standby databases.

Table A–1 summarizes the common processes that prevent switchover and what corrective action you need to take.

Table A–1 Common Processes That Prevent Switchover

Type of Process	Process Description	Corrective Action
CJQ0	The Job Queue Scheduler Process	Change the JOB_QUEUE_PROCESSES dynamic parameter to the value 0. The change will take effect immediately without having to restart the instance.
QMN0	The Advanced Queue Time Manager	Change the AQ_TM_PROCESSES dynamic parameter to the value 0. The change will take effect immediately without having to restart the instance.
DBSNMP	The Oracle Enterprise Manager Intelligent Agent	Issue the `agentctl stop` command from the operating system prompt.

A.3 What to Do If SQL Apply Operations to a Logical Standby Database Stop

Log apply services cannot apply unsupported DML statements, DDL statements, and Oracle supplied packages to a logical standby database in SQL apply mode.

When an unsupported statement or package is encountered, SQL apply operations stop. You can take the actions described in Table A–2 to correct the situation and start applying SQL statements to the logical standby database again.

Table A–2 Fixing Typical SQL Apply Operations Errors

If...	Then...
You suspect an unsupported statement or Oracle supplied package was encountered.	Find the last statement in the DBA_LOGSTDBY_EVENTS view. This will indicate the statement and error that caused SQL apply operations to fail. If an incorrect SQL statement caused SQL apply operations to fail, transaction information, as well as the statement and error information, can be viewed. The transaction information can be used with other Oracle9*i* LogMiner tools to understand the cause of the problem.
An error requiring database management occurred, such as running out of space in a particular tablespace.	Fix the problem and resume SQL apply operations using the ALTER DATABASE START LOGICAL STANDBY APPLY statement.
An error occurred because a SQL statement was entered incorrectly, such as an incorrect standby database filename being entered in a tablespace command.	Enter the correct SQL statement and use the DBMS_LOGSTDBY.SKIP_TRANSACTION procedure to ensure that the incorrect statement is ignored the next time SQL apply operations are run. Then restart SQL apply operations using the ALTER DATABASE START LOGICAL STANDBY APPLY statement.
An error occurred because skip parameters were incorrectly set up, such as specifying that all DML for a given table be skipped but CREATE, ALTER, and DROP TABLE statements were not specified to be skipped.	Issue a DBMS_LOGSTDBY.SKIP('TABLE','schema_name','table_name',null) call, then restart SQL apply operations.

See Also: Chapter 14 for information about querying the DBA_LOGSTDBY_EVENTS view to determine the cause of failures

B

Manual Recovery

Although Oracle Corporation recommends that you use the managed recovery mode for your physical standby databases, you can also use **manual recovery mode.** You might choose manual recovery mode for any of the following reasons:

- Manual recovery mode allows you to have more than 10 standby databases, which is the limit with managed recovery.
- If you do not want to connect to the primary and standby databases over the Oracle Net network, then open the database in manual recovery mode.
- If you are using managed recovery mode and, for some reason, that mode no longer works, you can change to manual recovery mode.

This appendix explains how to work in manual recovery mode. It includes the following topics:

B.1 Preparing a Standby Database for Manual Recovery: Basic Tasks

Table B–1 summarizes the basic tasks for setting up a standby database in preparation for manual recovery. This procedure assumes that you plan to connect to the standby database through Oracle Net. If you do not wish to use Oracle Net to connect to the standby database, skip steps 4 and 5.

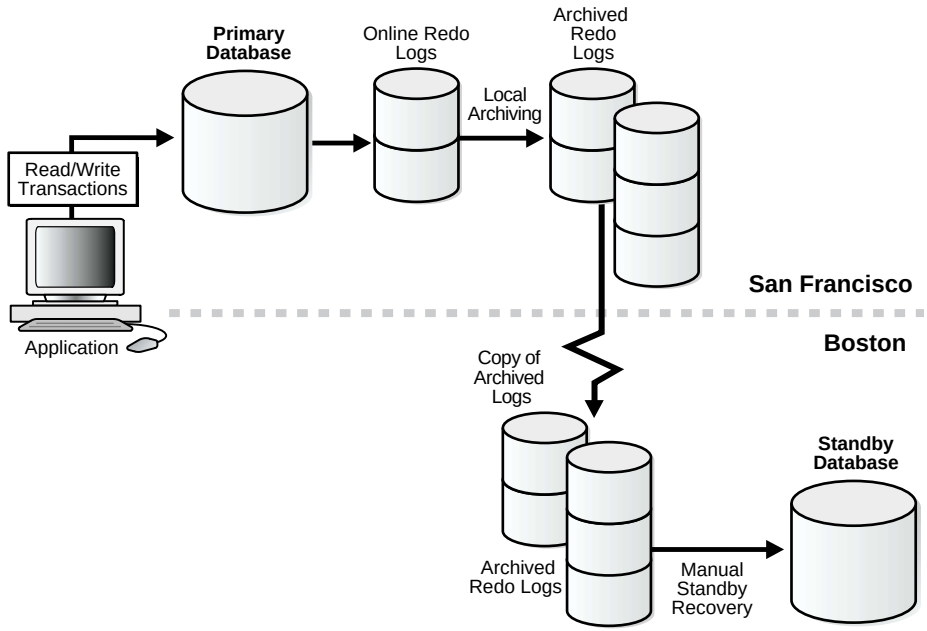
Table B–1 Task List: Preparing for Manual Recovery

Step	Task	Procedure
1	Either make a new backup of the primary database datafiles or access an old backup.	Section 3.3.2
2	Connect to the primary database and create the standby control file.	Section 3.3.3
3	Copy the backup datafiles and standby control file from the primary site to the standby site.	Section 3.3.4
4	If you want to create an Oracle Net connection to the standby database, create a service name.	Section 6.5 and LOCATION and SERVICE in Chapter 12
5	If you want to create an Oracle Net connection to the standby database, configure the listener on the standby site so that it can receive requests for connections to the standby instance.	Section 6.5
6	Create the standby initialization parameter file on the standby site and set the initialization parameters for the standby database. Optionally, set `DB_FILE_NAME_CONVERT` and `LOG_FILE_NAME_CONVERT` to automatically rename primary files in the standby control file.	Section 5.6
7	Start the standby instance and mount the standby database.	Section 6.3.2
8	While connected to the standby database, manually change the names of the primary datafiles and redo logs in the standby control file for all files *not* automatically renamed using `DB_FILE_NAME_CONVERT` and `LOG_FILE_NAME_CONVERT` in step 6. If step 6 renamed all files, skip this step.	Section B.4

B.2 Placing the Standby Database in Manual Recovery Mode

After you have started and mounted the standby database, you can place it in manual recovery mode. To keep the standby database current, you must manually apply archived redo logs from the primary database to the standby database. Figure B–1 shows a database in manual recovery mode.

Figure B–1 Standby Database in Manual Recovery Mode



This section contains the following topics:

- Initiating Manual Recovery Mode
- When Is Manual Recovery Required?

B.2.1 Initiating Manual Recovery Mode

Archived logs arrive at the standby site in one of the following ways:

- The primary database automatically archives the logs (only if you implement a Data Guard environment).
- You manually transfer logs using an operating system utility or some other means.

The standby database assumes that the archived log file group is in the location specified by either of the following parameters in the standby initialization parameter file:

- First valid disk location specified by LOG_ARCHIVE_DEST_*n* (where *n* is an integer from 1 to 10)
- Location specified by LOG_ARCHIVE_DEST_*n*

If the archived logs are not in the location specified in the initialization parameter file, you can specify an alternative location using the FROM option of the RECOVER statement.

To place the standby database in manual recovery mode:

1. Use SQL*Plus to connect to the standby instance and then start the Oracle instance at the standby database. For example, enter:

   ```
   STARTUP NOMOUNT pfile=initSTANDBY.ora
   ```

2. Mount the standby database:

   ```
   ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

3. If log transport services are not archiving logs automatically to the standby site, then manually copy the logs to the desired location on the standby site using an appropriate operating system utility for transferring binary data. For example, enter:

   ```
   % cp /oracle/arc_dest/*.arc /standby/arc_dest
   ```

4. Issue a RECOVER statement to place the standby database in manual recovery mode.

   Note: Specify the FROM 'location' option only if the archived log group is *not* in the location specified by the LOG_ARCHIVE_DEST_*n* parameter (where *n* is an integer from 1 to 10) or the LOG_ARCHIVE_DEST_*n* parameter in the standby initialization parameter file.

   For example, execute one of the following statements:

   ```
   RECOVER STANDBY DATABASE # uses location for logs specified in
                    # initialization parameter file
   RECOVER FROM '/logs' STANDBY DATABASE # specifies nondefault location
   ```

As the Oracle database server generates archived redo logs, you must continually copy and apply them to the standby database to keep it current.

B.2.2 When Is Manual Recovery Required?

Manual recovery mode is required in a non-Data Guard environment. A non-Data Guard environment is one in which you manually:

- Transfer the archived redo logs from the primary site to the standby site
- Apply the archived redo logs to the standby database

Even if you implement a Data Guard environment, you may occasionally choose to perform manual recovery on the standby database. For example, you may choose to manually resolve an existing archive gap by using manual recovery mode.

B.3 Resolving Archive Gaps Manually

An **archive gap** is a range of archived redo logs created whenever you are unable to apply the next archived redo log generated by the primary database to the standby database. This section contains the following topics:

- What Causes Archive Gaps?
- Determining Whether an Archive Gap Exists
- Manually Transmitting the Logs in the Archive Gap to the Standby Site
- Manually Applying the Logs in the Archive Gap to the Standby Database

---

Note: Typically, archive gaps are resolved automatically without the need for manual intervention. See Section 6.5 for more information about how log apply services automatically recover from gaps in the redo logs.

B.3.1 What Causes Archive Gaps?

An archive gap can occur whenever the primary database archives a log, but the log is not archived to the standby site. Because the standby database requires the sequential application of redo logs, media recovery stops at the first missing log encountered.

Archive gaps can occur in the following situations:

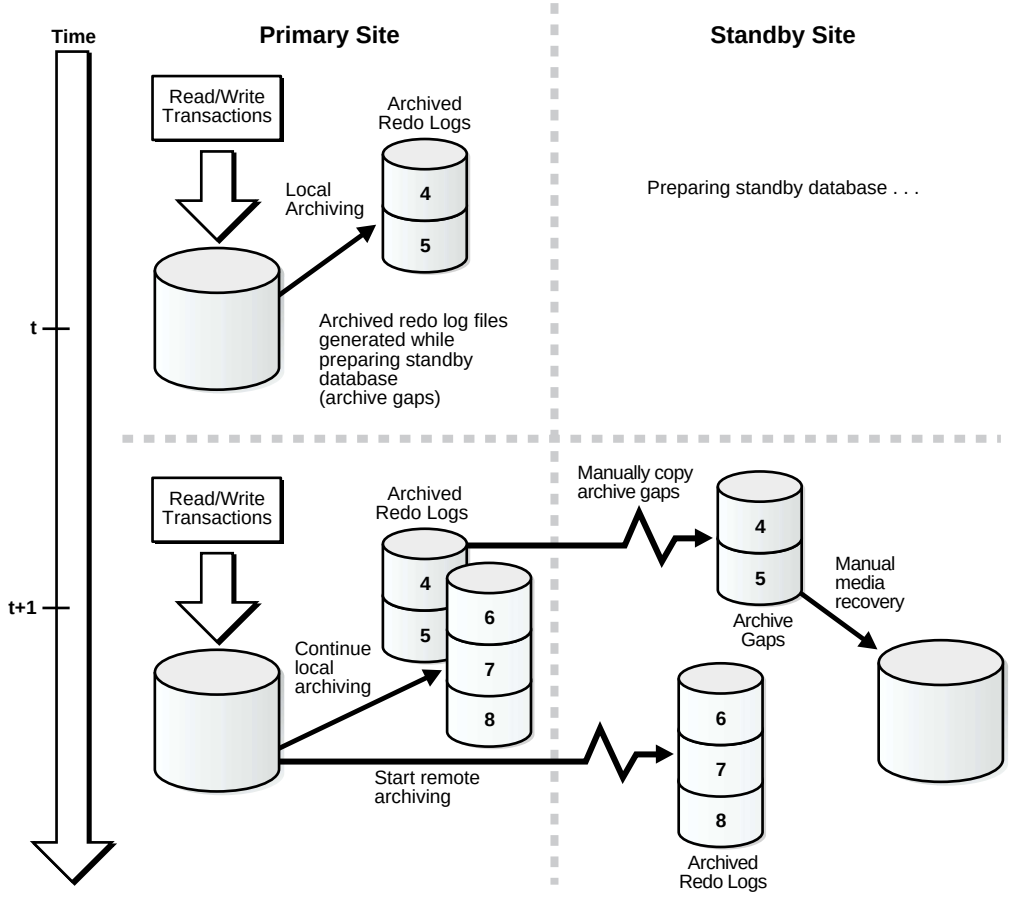
- Creation of the Standby Database
- Shutdown of the Standby Database When the Primary Database Is Open
- Network Failure Preventing the Archiving of Logs to the Standby Site

B.3.1.1 Creation of the Standby Database

One example of an archive gap occurs when you create the standby database from an old backup. For example, if the standby database is made from a backup that contains changes through log 100, and the primary database currently contains changes through log 150, then the standby database requires that you apply logs 101 to 150.

Another typical example of an archive gap occurs when you generate the standby database from a hot backup of an open database. For example, assume the scenario illustrated in Figure B–2.

Figure B–2 Manual Recovery of Archived Logs in an Archive Gap



The following steps occur:

1. You take a hot backup of database `primary`.
2. At time *t*, while you are busy configuring the network files, `primary` archives log sequences 4 and 5.
3. At time *t + 1*, you start the standby instance.
4. `primary` archives log sequences 6, 7, and 8 to both the primary site and the standby site.

Archived log sequences 4 and 5 are now part of an archive gap, and these logs must be applied to the standby database.

B.3.1.2 Shutdown of the Standby Database When the Primary Database Is Open

You may be required to shut down the standby database to resolve maintenance issues. For example, you must shut down the standby database when you change a control file parameter, such as MAXDATAFILE, in the primary database.

> **Note:** Performing a RESETLOGS operation on the primary database invalidates the standby database. If you reset the logs on the primary database, you must rebuild the standby database.

To avoid creating archive gaps, follow these rules:

- Start the standby databases and listeners *before* starting the primary database.
- Shut down the primary database *before* shutting down the standby database.

If you violate either of these two rules, then the standby database is down while the primary database is open and archiving. Consequently, the Oracle database server can create an archive gap.

> **Note:** If the standby site is specified as MANDATORY in one of the LOG_ARCHIVE_DEST_*n* parameters of the primary initialization parameter file, dynamically change it to OPTIONAL before shutting down the standby database. Otherwise, the primary database eventually stalls because it cannot archive its online redo logs.

B.3.1.3 Network Failure Preventing the Archiving of Logs to the Standby Site

If you maintain a Data Guard environment, and the network goes down, the primary database may continue to archive to disk but be unable to archive to the standby site. In this situation, archived logs accumulate as usual on the primary site, but the standby instance is unaware of them.

To prevent this problem, you can specify that the standby destination have mandatory status. If the archiving destination is mandatory, then the primary database will not archive any logs until it is able to archive to the standby site. For example, you can set the following in the primary initialization parameter file to make standby1 a mandatory archiving destination:

```
LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1 MANDATORY'
```

One consequence of this configuration is that unless the network problem is fixed, the primary database eventually stalls because it cannot switch into an unarchived online redo log. This problem is exacerbated if you maintain only two online redo logs in your primary database.

See Also:

- Section 5.5.2.3 for a detailed account of the significance of the OPTIONAL and MANDATORY attributes for standby archival
- Section 10.1.7 for a related scenario

B.3.2 Determining Whether an Archive Gap Exists

To determine whether there is an archive gap, query the V$ARCHIVE_GAP view. If an archive gap exists, the output of the query specifies the thread number and log sequence number of all logs in the archive gap. If there is *no* archive gap for a given thread, the query returns no rows.

To identify the logs in the archive gap:

1. Query the V$ARCHIVE_GAP view on the standby database as follows:

```
SQL> SELECT THREAD#, LOW_SEQUENCE#, HIGH_SEQUENCE#
  2> FROM V$ARCHIVE_GAP;
```

2. Examine the output of the query to determine the archive gap. For example, the output may look like:

```
THREAD#    LOW_SEQUENCE#   HIGH_SEQUENCE#
---------- -------------   -------------
       1             460             463
       2             202             204
       3             100             100
```

In this example, the LOW_SEQUENCE# and HIGH_SEQUENCE# for thread 3 are identical, indicating that thread 3 is missing 1 log, sequence 100. However, not every thread has an archive gap. If there are no gaps, then the output for this view will be empty.

B.3.3 Manually Transmitting the Logs in the Archive Gap to the Standby Site

After you have obtained the log sequence numbers of the logs in the archive gap, you can obtain their filenames by querying the V$ARCHIVED_LOG view on the primary site. The archived log filenames on the standby site are generated by the STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT parameters in the standby initialization parameter file.

If the standby database is on the same site as the primary database, or the standby database is on a remote site with a different directory structure than the primary database, the filenames for the logs on the standby site cannot be the same as the filenames of the logs archived by the primary database. Before transmitting the archived logs to the standby site, determine the correct filenames for the logs at the standby site.

To copy logs in an archive gap to the standby site:

1. Review the list of archive gap logs that you obtained earlier. For example, assume you have the following archive gap:

```
THREAD#    LOW_SEQUENCE#   HIGH_SEQUENCE#
---------- -------------   --------------
       1             460              463
       2             202              204
       3             100              100
```

   If a thread appears in the view, then it contains an archive gap. You need to copy logs from threads 1, 2, and 3.

2. Determine the filenames of the logs in the archive gap that were archived by the primary database. After connecting to the primary database, issue a SQL query to obtain the name of a log in each thread. For example, use the following SQL statement to obtain filenames of logs for thread 1:

```
SQL> SELECT NAME FROM V$ARCHIVED_LOG WHERE THREAD#=1 AND DEST_ID=1
  2> AND SEQUENCE# > 459 AND SEQUENCE# < 464;

NAME
-------------------------------------------------------------------
/primary/thread1_dest/arcr_1_460.arc
/primary/thread1_dest/arcr_1_461.arc
/primary/thread1_dest/arcr_1_462.arc
/primary/thread1_dest/arcr_1_463.arc
4 rows selected
```

   Perform similar queries for threads 2 and 3.

3. On the standby site, review the settings for STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT in the standby initialization parameter file. For example, you discover the following:

```
STANDBY_ARCHIVE_DEST = /standby/arc_dest/
LOG_ARCHIVE_FORMAT = log_%t_%s.arc
```

   These parameter settings determine the filenames of the archived redo logs at the standby site.

4. On the primary site, copy the archive gap logs from the primary site to the standby site, renaming them according to values for STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT. For example, enter the following copy commands to copy the archive gap logs required by thread 1:

```
% cp /primary/thread1_dest/arcr_1_460.arc /standby/arc_dest/log_1_460.arc
% cp /primary/thread1_dest/arcr_1_461.arc /standby/arc_dest/log_1_461.arc
% cp /primary/thread1_dest/arcr_1_462.arc /standby/arc_dest/log_1_462.arc
% cp /primary/thread1_dest/arcr_1_463.arc /standby/arc_dest/log_1_463.arc
```

   Perform similar copy commands to copy archive gap logs for threads 2 and 3.

5. On the standby site, if the LOG_ARCHIVE_DEST and STANDBY_ARCHIVE_DEST parameter values are *not* the same, then copy the archive gap logs from the STANDBY_ARCHIVE_DEST directory to the LOG_ARCHIVE_DEST directory. If these parameter values *are* the same, then you do not need to perform this step.

   For example, assume the following standby initialization parameter settings:

```
STANDBY_ARCHIVE_DEST = /standby/arc_dest/
LOG_ARCHIVE_DEST = /log_dest/
```

   Because the parameter values are different, copy the archived logs to the LOG_ARCHIVE_DEST location:

```
% cp /standby/arc_dest/* /log_dest/
```

   When you initiate manual recovery, the Oracle database server looks at the LOG_ARCHIVE_DEST value to determine the location of the logs.

Now that all required logs are in the STANDBY_ARCHIVE_DEST directory, you can proceed to Section B.3.4 to apply the archive gap logs to the standby database.

See Also: Section 6.4.3 and V$ARCHIVED_LOG in Chapter 14

B.3.4 Manually Applying the Logs in the Archive Gap to the Standby Database

After you have copied the logs in the archive gap to the standby site, you can apply them using the RECOVER AUTOMATIC statement.

To apply the archived redo logs in the archive gap:

1. Start up and mount the standby database (if it is not already mounted). For example, enter:

   ```
   SQL> STARTUP NOMOUNT PFILE=/oracle/admin/pfile/initSTBY.ora
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

2. Recover the database using the AUTOMATIC option:

   ```
   SQL> ALTER DATABASE RECOVER AUTOMATIC STANDBY DATABASE;
   ```

   The AUTOMATIC option automatically generates the name of the next archived redo log file needed to continue the recovery operation.

   After recovering the available logs, the Oracle database server prompts for the name of a log that does not exist. The reason is that the recovery process does not know about the logs archived to the standby site by the primary database. For example, you might see:

   ```
   ORA-00308: cannot open archived log '/oracle/standby/standby_logs/arcr_1_
   540.arc'
   ORA-27037: unable to obtain file status
   SVR4 Error: 2: No such file or directory
   Additional information: 3
   Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
   ```

3. Cancel recovery after the Oracle database server has applied the available logs, by executing the following statement (or typing CTRL+C):

   ```
   SQL> CANCEL
   Media recovery cancelled.
   ```

   The following error messages are acceptable after recovery cancellation and do not indicate a problem:

   ```
   ORA-01547: warning: RECOVER succeeded but OPEN RESETLOGS would get error
   below
   ORA-01194: file 1 needs more recovery to be consistent
   ORA-01110: data file 1: 'some_filename'
   ORA-01112: media recovery not started
   ```

Oracle Corporation recommends automatically applying the logs in the archive gap using the RECOVER MANAGED STANDBY DATABASE clause of the ALTER DATABASE statement.

See Also: Section 6.5 for additional information

B.4 Renaming Standby Database Files Manually

Sometimes all of the primary datafiles and redo log files cannot be renamed in the standby control file by conversion parameters. For example, assume that your database has the following datafiles, which you want to rename as shown in the following table:

Primary Filename	Standby Filename
/oracle/dbs/df1.dbf	/standby/df1.dbf
/oracle/dbs/df2.dbf	/standby/df2.dbf
/data/df3.dbf	/standby/df3.dbf

You can set DB_FILE_NAME_CONVERT as follows to convert the filenames for the first two datafiles:

```
DB_FILE_NAME_CONVERT = '/oracle/dbs', '/standby'
```

Nevertheless, this parameter will not capture the renaming of /data/df3.dbf. You must rename this datafile manually in the standby database control file by issuing a SQL statement as follows:

```
SQL> ALTER DATABASE RENAME FILE '/data/df3.dbf' to '/standby/df3.dbf';
```

To rename a datafile manually:

1. Start up and mount the standby database (if it is not already started) and then mount the database:

   ```
   SQL> STARTUP NOMOUNT PFILE=initSTANDBY1.ora;
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

2. Issue an ALTER DATABASE statement for each datafile requiring renaming, where *old_name* is the old name of the datafile as recorded in the control file and *new_name* is the new name of the datafile that will be recorded in the standby control file:

```
SQL> ALTER DATABASE RENAME FILE 'old_name' TO 'new_name';
```

When you manually rename all of the datafiles that are not captured by the DB_FILE_NAME_CONVERT parameter, the standby database control file can correctly interpret the log stream during the recovery process.

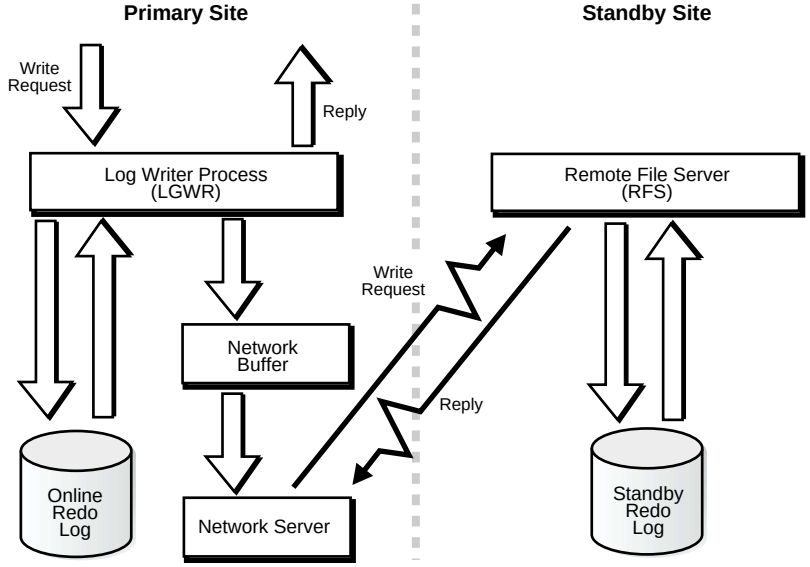
C

Log Writer Asynchronous Network I/O

This appendix describes the Data Guard algorithms for asynchronous network I/O when you use the log writer process (LGWR) to transmit primary database online redo log modifications to standby databases. This information is provided to help database administrators and support analysts know how asynchronous I/O is achieved, and how to diagnose and interpret asynchronous I/O-related network issues.

Figure C–1 shows the various processes involved in the asynchronous network I/O operation.

Figure C–1 Asynchronous Network I/O Processes



Data Guard achieves asynchronous network I/O using transparent network server processes. These network server processes are deployed by the LGWR process when the ASYNC attribute is specified for the LOG_ARCHIVE_DEST_*n* initialization parameter.

> **Note:** LGWR network server processes are different from LGWR I/O slave processes. However, both types of background processes are compatible with each other.

The network server processes are transparent because the remote file server (RFS) processes on the standby database do not know that network server processes are initiating the network I/O operations on the primary database. The network server processes mimic the same network I/O operations that occur when the LGWR process is set up for synchronous I/O operations. The network server process name is ora_lns*nn*_SID where *nn* is the arbitrary server number. For example, ora_lns5_prmy is background network server process number 5 for the database whose SID is prmy.

There is one network server process for each LGWR network connection to an asynchronous standby database. A network server process is created when the network connection to the standby database is established. A network server process remains active as long as the network connection to the standby database remains active. The abnormal termination of a network server process does not lead to instance shutdown. For example, if the system monitor (SMON) background process is unable to function, the Oracle instance ceases operation and fails. However, if one of the network server processes is terminated, the process monitor (PMON) background process wakes up and cleans up the resources that were being occupied by that network server process, and the Oracle instance continues to function. The PMON activity in cleaning up the resources is recorded in the corresponding alert log and trace files.

However, if a network server process terminates abnormally, the LGWR process discards the network connection for the remainder of the processing of the current online redo log. The network connection is reestablished, if possible, on the next log switch operation. Each network server process manages an internal network buffer, whose size is specified by the ASYNC attribute of the LOG_ARCHIVE_DEST_*n* initialization parameter. The network buffer resides in the private memory of the network server process. See Chapter 12 for information on the LOG_ARCHIVE_DEST_*n* initialization parameter.

The LGWR process sends the online redo modifications to the network server and the network server caches the online redo log modifications in the network buffer, if possible. If the online redo log modifications can be cached, the LGWR process continues operating as if the network I/O operation was successfully performed, when, in fact, it was not.

Because the network buffer resides on the primary database, the contents of this cache are susceptible to loss due to primary database system failure. To minimize this risk, the network server processes attempt to transmit the data to the standby database as long as the network does not become saturated.

Several events can also cause the network buffer to be transmitted to the standby database by the network server. These events are:

- If the LGWR request size exceeds the currently available buffer space, the existing network buffer is transmitted to the standby database. The LGWR process stalls until sufficient buffer space can be reclaimed, but this seldom occurs.
- A primary database online log switch forces the network buffer to be transmitted to the standby database.
- The primary database is shut down normally.

Note: An immediate shutdown of the primary database results in the network buffer contents being discarded because the network server process is shut down.

A standby database shutdown also causes the network buffer contents to be discarded.

- The primary database has no online redo log activity for a period of time. The duration of database inactivity cannot be controlled by the user.
- If the rate of redo generation exceeds the runtime network latency, the LGWR request is buffered if sufficient space is available. Otherwise, the existing buffer is transmitted to the standby database. The LGWR process stalls until sufficient buffer space can be reclaimed, but this seldom occurs.

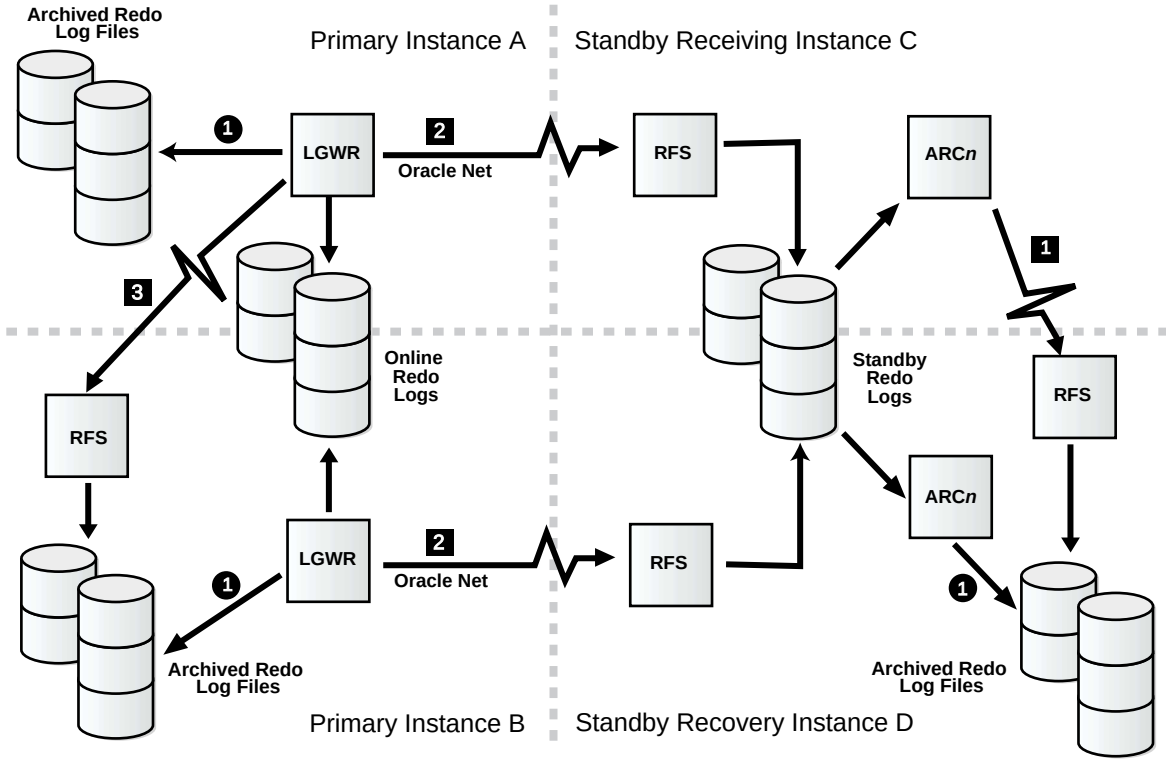
The network server process always performs synchronous network I/O; however, the LGWR process interprets this as asynchronous network I/O, because it is not blocked by the actual network I/O operation.

D

Standby Database Real Application Clusters Support

Oracle9*i* provides the ability to perform true database archiving from a primary database to a standby database when both databases reside in a Real Application Clusters environment. This allows you to separate the log transport services processing from the log apply services processing on the standby database, thereby improving overall primary and standby database performance. Figure D–1 illustrates a standby database configuration in a Real Application Clusters environment.

Figure D–1 Standby Database in Real Application Clusters



In Figure D–1, the numbers within circles indicate local connections, and the numbers within boxes indicate remote connections.

When you use the standby database in a Real Application Clusters environment, any instance can receive archived logs from the primary database; this is the **receiving instance.** However, the archived logs must ultimately reside on disk devices accessible by the node on which the managed recovery operation is performed; this is the **recovery instance.** Transferring the standby database archived logs from the receiving instance to the recovery instance is achieved using the cross-instance archival operation, performed on the standby database.

See Also: Section 8.4.1

The standby database cross-instance archival operation requires use of standby redo logs as the temporary repository of primary database archived logs. Using the

standby redo logs not only improves standby database performance and reliability, but also allows the cross-instance archival operation to be performed. However, because standby redo logs are required for the cross-instance archival operation, the primary database must use the log writer process (LGWR) to perform the primary database archival operation.

To set up a standby database in a Real Application Clusters environment:

Perform the following steps to set up log transport services on the standby database:

1. Create the standby redo logs. In a Real Application Clusters environment, the standby redo logs must reside on disk devices shared by all instances, such as raw devices.
2. On the recovery instance where the managed recovery process (MRP) is to operate, define the archived log destination to archive locally, because cross-instance archiving is not necessary. This is accomplished using the `LOCATION` attribute of the `LOG_ARCHIVE_DEST_1` initialization parameter.

   Note: Unlike a primary database, a standby database is not required to have a local archived log destination.

3. On the receiving instance, define the archived log destination to archive to the node where the MRP is to operate. This is accomplished using the `SERVICE` attribute of the `LOG_ARCHIVE_DEST_1` initialization parameter.
4. Start the ARC*n* process on all standby database instances.
5. Start the MRP on the recovery instance.

To set up a primary database in a Real Application Clusters environment:

Perform the following steps to set up log transport services on the primary database:

1. On all instances, designate the LGWR process to perform the archival operation.
2. Designate the standby database as the receiving node. This is accomplished using the `SERVICE` attribute of the `LOG_ARCHIVE_DEST_`*n* initialization parameter.

Ideally, each primary database instance should archive to a corresponding standby database instance. However, this is not required.

E

Cascading Standby Databases

To reduce the load on your primary system, you can implement cascading standby databases. A **cascading standby database** is a standby database that receives its redo logs from another standby database, not from the original primary database. You can configure:

- A physical standby database to send the incoming redo logs to other remote destinations in the same manner as the primary database, with up to one level of redirection.
- A logical standby database (because it is open in read / write mode) to send the redo logs it generates to its own set of standby (physical or logical) databases. In this manner, performance issues with network latency versus workload on the primary system can be spread out over the network and databases.

This appendix contains the following sections:

E.1 Background Information

To help you understand cascading standby databases and how to configure them, this section provides background information about configuring standby databases in the Data Guard environment.

A Data Guard configuration typically contains a primary database and a remote standby database. As a part of setting up the Data Guard configuration, you must define and enable the `LOG_ARCHIVE_DEST_`*n* (where *n* is a number from 1 to 10) initialization parameter on the primary database. Once this destination is enabled,

log transport services on the primary database send the redo logs to be applied on the standby database. You configure the destination attributes the same way for physical or logical standby databases, and you can continue to set up additional standby databases until you have the maximum of nine remote standby databases.

To create a seamless environment in which standby databases continue to operate through a switchover or failover operation, define the destination parameters on the various standby databases to point back to the primary database. In addition, you can configure the LOG_ARCHIVE_DEST_*n* initialization parameter:

- On physical standby databases to point to all other physical and logical standby databases
- On logical standby databases to point back to the primary database and possibly to other logical standby databases

For example:

On the primary database, you have three standby databases that are configured as follows:

```
LOG_ARCHIVE_DEST_2='SERVICE=PHYSICAL1'
LOG_ARCHIVE_DEST_3='SERVICE=PHYSICAL2'
LOG_ARCHIVE_DEST_4='SERVICE=LOGICAL3'
```

On the physical standby database called PHYSICAL1, you need to define the following:

```
LOG_ARCHIVE_DEST_2='SERVICE=PRIMARY'
LOG_ARCHIVE_DEST_3='SERVICE=PHYSICAL2'
LOG_ARCHIVE_DEST_4='SERVICE=LOGICAL3'
```

On the physical standby database called PHYSICAL2, you need to define the following:

```
LOG_ARCHIVE_DEST_2='SERVICE=PRIMARY'
LOG_ARCHIVE_DEST_3='SERVICE=PHYSICAL1'
LOG_ARCHIVE_DEST_4='SERVICE=LOGICAL3'
```

On the logical standby database called LOGICAL3, you need to define the following:

```
LOG_ARCHIVE_DEST_2='SERVICE=PRIMARY'
```

With the configuration defined in this way, a switchover operation from the primary database to a physical standby database will allow that standby database, as the new primary database, to send new redo log information to all of the other standby databases. Performing a switchover operation to a logical standby database

renders the other physical standby databases unable to accept redo logs from the new primary database in this case. However, other logical standby databases can accept the redo logs from the new primary database.

During a failover operation, these parameters are even more important as they enable the system to automatically send the final redo log (which contains the end-of-redo marker) to the other standby databases, allowing them to accept new redo information from the new primary database. For example, if you fail over to PHYSICAL1 when it finishes applying redo logs, PHYSICAL1 can send the redo log containing the end-of-redo marker to PHYSICAL2 and LOGICAL3 to signal that a new primary database will soon begin sending redo logs.

Although you can define up to nine remote standby databases per primary database, each standby destination that must be sent redo logs puts additional load on the primary system. You can reduce this load by using the archiving process (ARCH) transport method or the log writer (LGWR) with asynchronous (ASYNC) network protocol, but these methods increase the possibility of missing data in a failure situation. You could also define the DEPENDENCY attribute for some destinations, but that requires one of the following:

- The standby database that is dependent on another standby database must be located on the same system as the other standby database
- The archived redo logs sent to the other standby database must be visible to the dependent standby database

E.2 Configuring Cascading Standby Databases

The following sections describe how to set up the Data Guard configuration to use cascading standby databases:

- Cascading Physical Standby Databases
- Cascading Logical Standby Databases

E.2.1 Cascading Physical Standby Databases

To enable a physical standby database to send the incoming redo logs to another set of destinations, you must define the following items:

- Define the LOG_ARCHIVE_DEST_*n* initialization parameter on the primary database to set up a physical standby database that will be the starting point for a cascade to use the LGWR transport method. Use either SYNC or ASYNC network protocols depending on your requirements.

- On the receiving physical standby database, define sufficient standby redo log files and ensure that archiving is enabled.

At this point, you can begin defining the LOG_ARCHIVE_DEST_*n* initialization parameter on the physical standby database that will define the end points of the cascade. Remember, as part of the original setup of the physical standby database, you should have defined a local archive destination that will be used for local archiving when the physical standby database transitions to the primary role. For example, you might define the LOG_ARCHIVE_DEST_1 initialization parameter to be the'LOCATION=/physical1/arch' location. When the physical standby database switches roles, any archived redo logs will be put into that directory with the same format that you defined with the LOG_ARCHIVE_FORMAT initialization parameter. This local archiving destination can be the same as the one defined in the parameter STANDBY_ARCHIVE_DEST, but this is not required.

A side effect of this configuration is that the archive process on the standby database will now try to send the log files not only to the cascading end points but also to the other standby databases and the primary database if they are defined and enabled. This is not a problem, because the receiving database will either reject it if it is the primary database or a standby database that has already received the same log successfully. If the destination is another standby database and it has not received the log successfully, then this acts as a proactive gap resolution. You can avoid this by setting the state to DEFER for any destinations not involved in the cascade. However, you will have to remember to enable them again if you do a switchover or failover operation.

If you want to have one initialization parameter file to handle both the cascading standby databases and the original primary and standby configuration, define the destinations for the primary database and other standby databases as well as the cascading standby databases. However, the total remote destinations still cannot exceed 10, including the local archiving destination.

Because it is the archiver process and not the log writer process that (during the archiving of the standby online redo logs) is sending the redo information to the cascading standby databases, you are limited to one set of cascading standby databases per standby database that is connected directly to the primary database.

E.2.2 Cascading Logical Standby Databases

A logical standby database is similar to a *primary* database in that you can define and create standby databases for it in the same manner as for the original configuration. However, unlike cascading physical standby databases, logical standby databases will not be exact copies of the original primary database.

- Physical standby databases that cascade from a logical standby database will be a block-for-block copy of the logical standby database and a logical copy of the original primary database.
- Logical standby databases that cascade from a logical standby database will be logical copies of the parent logical standby database and bear only partial resemblance to the original primary database. This is because the original primary database's data is there but so is anything else stored in the parent logical standby database as well as any other changes such as different indexes or materialized views.

Cascading standby databases from a logical standby database require that you perform the same setup tasks as for a normal physical or logical standby database. You can use any transport mode (LGWR or ARCH) and network protocol (SYNC or ASYNC). If you use the LGWR network protocol, you can optionally use standby online redo logs on your physical standby databases.

E.3 Examples of Cascading Standby Databases

The following scenarios demonstrate set up and uses for cascading standby databases.

E.3.1 Scenario 1

You have a primary database in your corporate offices and you want to create a standby database in another building on your local area network (LAN). In addition, you have a legal insurance requirement to keep the redo information and backup copies off-site at a geographically distant location outside of your LAN but on your wide area network (WAN).

You could define two destinations on your primary database so that redo logs could be transmitted to both of these sites, but this would put an extra workload on your primary database throughput due to the network latency of sending the redo logs over the WAN.

To solve this problem, you could define a tight connection between your primary and physical standby databases in your LAN using the LGWR and SYNC network transports and standby online redo logs. This would protect against losing access to the primary database and provide an alternate site for production when maintenance is required on the primary database. The secondary location on the WAN could be serviced by the physical standby database, ensuring that the redo information is stored off-site. Nightly backup operations on the production

database could then be moved to the WAN remote standby database, which removes the requirement to ship tapes to the off-site storage area.

Finally, in a worst case scenario where you lose access to both the primary database and the physical standby database on the LAN, you could fail over to the remote standby database with minimal data loss. If you can gain access to the last standby database's online redo log from the original standby database, you could recover it on the remote standby database, incurring no data loss.

The only time you would incur problems by sending the information over the WAN is during a switchover or failover operation, when the physical standby database has transitioned to the primary role. However, this configuration would still meet your insurance requirements.

E.3.2 Scenario 2

You have a primary database in a remote city and you would like to have access to its data locally for reporting purposes. The primary database already has a standby database set up for failure protection in an off-site location on the LAN. Putting a destination on the primary database to send the information to your site would adversely affect the performance of the primary database.

Solving this problem is similar to the solution that is described in scenario 1, except that you are sending the redo logs to a logical standby database, because you already have a physical standby database. First, ensure that:

- The physical standby database is receiving its redo logs from the log writer of the primary database
- Standby redo log files are defined and being used

  If standby redo logs are not defined, you can define them dynamically on the standby database. The standby database will begin using the standby redo logs at the next log switch on the primary database. If the `LGWR` network transport is not being used, you can dynamically set log transport services on the primary database, and the primary database will start using the log writer at the next log switch.

Next, perform the normal setup tasks for a logical standby database. Any of the steps required to prepare to use a logical standby database must be done at the primary location, but once complete, you can copy the primary database to the standby location. After the logical standby database is up and running, define your destination parameter on the physical standby database to send the redo logs over the WAN, where they will be applied to the logical standby database.

E.3.3 Scenario 3

A primary database located in a manufacturing site already is configured with two physical standby databases. One standby database is located on the LAN in another building, and the second standby database is more remotely located on a WAN in the corporate offices. You cannot use cascading standby databases for the standby database on the WAN, because there is a requirement to have two standby databases in no-data-loss mode. Also, the marketing department has requested access to the manufacturing data for sales predictions. The marketing department needs access to the data on a daily basis, and they want to combine sales data with manufacturing data to better understand sales versus the actual manufacturing times.

One solution would be to allow marketing to access a physical standby database in the corporate offices using read-only mode. However, putting the standby database in read-only mode requires stopping managed recovery operations. This means that the physical standby database can only catch up with the primary database at night, while it is still receiving data from the second and third shifts at the manufacturing plant. In addition, the standby database would always be at least 12 hours behind in applying redo logs. You could add another destination to the primary database to send the redo logs to a new logical standby database in the corporate offices. Because the systems used in the corporate office are different for the physical standby database and the proposed logical standby database, you cannot use the `DEPENDENCY` attribute when defining the standby destinations. Because redo logs need to be transmitted over a WAN, it would degrade performance on the primary database to send the redo data twice, which has been deemed to be unacceptable.

Cascading standby databases can solve this problem. To set this up, you would create a logical standby database following the instructions in Chapter 4, but you would also set up the corporate physical standby database to transmit the redo logs over the corporate LAN to the new logical standby database. In this way, the primary database is only sending the data once over the WAN. The logical standby database could then be modified with new materialized views so that the marketing group can manage the data more efficiently. Because the logical standby database is open for read / write operations, the marketing group can add new schemas and load in sales data without affecting performance on the primary database, or the viability and current state of the physical standby database.

E.3.4 Scenario 4

You have five Sales offices around the world, each with its own primary database. You would like to implement a failure protection strategy for all of them, as well as a way to get timely access to all data with minimal effect on each primary database.

To solve this problem, you would first implement a no-data-loss environment for each of the five offices by creating a physical standby database (with LGWR and SYNC attributes) local to each office. The physical standby databases could be on a LAN or a WAN. Then, create a logical standby database from each of the five primary databases and locate the logical standby databases in your corporate office. However, instead of having log transport services on each of the five primary databases send the redo logs, you would configure each of the five standby databases to send the redo logs to its logical standby database over the WAN. At one logical standby database (or all of them), you would define database links to each of the other logical standby databases and use them to access all of the sales data. If you decide that you do not need all of the information from each of the five primary databases, but only certain tables, you can use the SKIP routines to stop applying data that you do not need on each of the logical standby databases.

E.3.5 Scenario 5

You have a primary database that is currently protected only by nightly backup operations. You have been told that you must implement a major failure recovery strategy immediately. You have another system of the same hardware type in-house, but it does not have enough power to serve as a standby database for failover purposes, and it does not have enough disks for the entire database. The only other system available to you that is large enough to hold the entire database is too far away to be put on the LAN, and the WAN that connects to it is extremely slow. The deadline for implementing the strategy is well before any network upgrades can occur. Adding a destination (on the primary database) to send the redo logs to the remote location would severely affect performance.

The interim solution to this problem would be to create a physical standby database on the remote system and create a distribution repository on the local smaller system. A distribution repository comprises only the standby control file and the standby database online redo logs, not the data files. You would configure the primary database to send the redo information to the repository locally using the log writer process (LGWR) in synchronous mode (SYNC). Because the connection is over the LAN, the effect on performance would be minimal. The repository would then be configured to send the data onwards over the WAN to the real standby database.

The risk with this configuration is that while the primary database has transmitted all of its data to a standby database, it is possible that the repository has not completed sending the data to the remote standby database at the time of a failure at the primary database. In this environment, as long as both systems do not fail at

the same time, the remote standby database should receive all the data sent up to the last log switch. You would have to send the current online redo log manually.

Once the WAN is upgraded to permit a direct connection to the remote standby database, you can either redirect the destination to the repository to point to the remote standby database directly or create a new destination to the remote standby database and continue transmitting to the repository as an archive log repository.

Glossary

ARCH

See archiver process (ARCn)

archive gap

A range of archived redo logs created whenever you are unable to apply the next archived redo log generated by the primary database to the standby database.

archived redo log

A copy of one of the filled members of an online redo log group made when the database is in ARCHIVELOG mode. As the LGWR process fills each online redo log with redo records, log transport services copy the log to one or more offline archive log destinations. This copy is the archived redo log, also known as the *offline redo log*.

See also ARCHIVELOG mode, online redo log, *and* redo log

ARCHIVELOG mode

The mode of the database in which log transport services archive filled online redo logs to disk. Specify the mode at database creation or by using the SQL `ALTER DATABASE ARCHIVELOG` statement. You can enable automatic archiving either dynamically using the SQL `ALTER SYSTEM ARCHIVE LOG START` statement or by setting the initialization parameter `LOG_ARCHIVE_START` to `true`.

Running your database in ARCHIVELOG mode has several advantages over NOARCHIVELOG mode. You can:

- Back up your database while it is open and being accessed by users
- Recover your database to any desired point in time

To protect your database that is in ARCHIVELOG mode in case of failure, back up your archived logs.

See also archived redo log, NOARCHIVELOG mode, *and* redo log

archiver process (ARC*n*)

On the primary database location, the process (or a SQL session performing an archival operation) that creates a copy of the online redo logs, either locally or remotely, for standby databases. On the standby database location, the ARC*n* process archives the standby redo logs to be applied by the managed recovery process (MRP).

archiving

The operation in which the ARC*n* background process copies filled online redo logs to offline destinations. You must run the primary database in ARCHIVELOG mode to archive redo logs.

ARC*n*

See archiver process (ARCn)

availability

The measure of the ability of a system or resource to provide the desired service when required. Measured in terms of the percentage of time the device is accessible out of the total time it is needed. Businesses that require uninterrupted computing services have an availability goal of 100%, or 24x365. The sum of availability and downtime equals 100%.

See also downtime *and* switchover

backup control file

A backup of the control file. Make the backup by:

- Using the Recovery Manager utility (RMAN) `backup` or `copy` command. Never create a backup control file by using operating system commands.
- Using the SQL statement `ALTER DATABASE BACKUP CONTROLFILE TO 'filename'`.

Typically, you restore backup control files when all copies of the current control file are damaged; sometimes you restore them before performing certain types of point-in-time recovery.

See also control file *and* current control file

backup piece

A physical file in a format specific to RMAN. The file belongs to only one backup set. A backup set usually contains only one backup piece. The only time RMAN creates more than one backup piece is when you limit the piece size using the `MAXPIECESIZE` option of the RMAN `ALLOCATE` or `CONFIGURE` command.

See also backup set *and* Recovery Manager (RMAN)

backup set

An RMAN-specific logical grouping of one or more physical files called *backup pieces*. The output of the RMAN `backup` command is a backup set. Extract the files in a backup set by using the RMAN `restore` command. You can multiplex files into a backup set, that is, intermingle blocks from input files into a single backup set.

There are two types of backup sets:

- Datafile backup sets, which are backups of any datafiles or a control file. This type of backup set is compressed, which means that it only contains datafile blocks that have been used; unused blocks are omitted.
- Archive log backup sets, which are backups of archived redo logs.

See also backup piece *and* Recovery Manager (RMAN)

broker

A distributed management framework that automates and simplifies most of the operations associated with the creation, control, and monitoring of a Data Guard configuration. The broker includes two user interfaces: Oracle Data Guard Manager (a graphical user interface) and the Data Guard command-line interface.

cascading standby database

A standby database that receives its redo logs from another standby database, not from the original primary database.

child destination

A log transport services archiving destination that is configured to receive redo logs from the primary database and depends on the successful completion of archival operations for the parent destination.

See also destination dependency, destinations, *and* parent destination

closed backup

A backup of one or more database files taken while the database is closed. Typically, a closed backup is also a whole database backup (a backup of the control file and all datafiles that belong to a database). If you closed the database cleanly, then all the files in the backup are consistent. If you shut down the database using a SHUTDOWN ABORT statement or the instance terminated abnormally, then the backups are inconsistent.

See also consistent backup

cold backup

See closed backup

consistent backup

A whole database backup (a backup of the control file and all datafiles that belong to a database) that you can open with the RESETLOGS option without performing media recovery. In other words, you do not need to apply redo logs to datafiles in this backup for it to be consistent. All datafiles in a consistent backup must:

- Have the same checkpoint system change number (SCN) in their headers, unless they are datafiles in tablespaces that are read-only or offline normal (in which case they will have a clean SCN that is earlier than the checkpoint SCN)
- Contain no changes past the checkpoint SCN
- Match the datafile checkpoint information stored in the control file

You can only make consistent backups after you have made a clean shutdown of the database. The database must not be opened until the backup has completed.

See also closed backup

control file

A binary file associated with a database that maintains the physical structure and timestamps of all files in that database. The Oracle database server updates the control file continuously during database use and must have it available for writing whenever the database is mounted or open.

See also backup control file *and* current control file

cross-instance archival environment

The environment in which each instance in a Real Application Clusters configuration directs its archived redo logs to a single instance of the cluster. This single instance is known as the *recovery instance*.

See also recovery instance

current control file

The control file on disk for a primary database; it is the most recently modified control file for the current incarnation of the database. For a control file to be considered current during recovery, it must not have been restored from backup.

See also backup control file *and* control file

current online redo log

The online redo log file in which the LGWR background process is currently logging redo records. Those files to which LGWR is not writing are called inactive.

When LGWR gets to the end of the file, it performs a log switch and begins writing to a new log file. If you run the database in ARCHIVELOG mode, then the ARC*n* process or processes copy the redo data into an archived redo log.

See also online redo log *and* redo log

Data Guard

The management, monitoring, and automation software that works with a production database and one or more standby databases to protect your data against errors, failures, and corruptions that might otherwise destroy your database.

See also primary database *and* standby database

data loss

The loss of data. Data loss occurs when you fail over to a standby database whose corresponding primary database is in the data divergence state.

See also graceful database failover

datafile

A physical operating system file on disk that was created by the Oracle database server and contains data structures such as tables and indexes. A datafile can only belong to one database.

See also tablespace

destination dependency

Configuring log transport services so that archiving redo logs to a specified destination is dependent upon the success or failure of archiving redo logs to another destination.

See also child destination, destinations, *and* parent destination

destinations

Log transport services allow the primary database to be configured to archive redo logs to up to 10 local and remote locations called destinations.

See also child destination, destination dependency, *and* parent destination

downtime

The measure of the inability of a system or resource to provide the desired service when required. Measured in terms of the percentage or amount of time the device is not accessible out of the total time it is needed. The sum of availability and downtime equals 100%.

A period of time for performing routine maintenance tasks, such as hardware and software upgrades and value-added services is *planned downtime*. The computing system is not available for productive operations during such scheduled maintenance tasks.

See also availability *and* switchover

failover

An unplanned event resulting from system or software failure.

If standby redo logs are present on the standby database, the failover is graceful and can occur with no data loss or with minimal data loss. If standby redo logs are not present on the standby database, the failover is forced and may result in the loss of application data.

See also forced database failover, graceful database failover, role transition, *and* switchover

FAL client

See fetch archive log (FAL) client

FAL server

See fetch archive log (FAL) server

fetch archive log (FAL)

See fetch archive log (FAL) client *and* fetch archive log (FAL) server

fetch archive log (FAL) client

A background Oracle database server process. The initialization parameter for the FAL client is set on the standby database. The FAL client pulls archived redo logs from the primary location and initiates and requests the transfer of archived redo logs automatically when it detects an archive gap on the standby database.

See also fetch archive log (FAL) server

fetch archive log (FAL) server

A background Oracle database server process that runs on the primary or other standby databases and services the fetch archive log (FAL) requests coming from the FAL client. For example, servicing a FAL request might include queuing requests (to send archived redo logs to one or more standby databases) to an Oracle database server that runs the FAL server. Multiple FAL servers can run on the same primary database at one time. A separate FAL server is created for each incoming FAL request. The initialization parameter for the FAL server is set on the standby database.

See also fetch archive log (FAL) client

forced database failover

Failover is an unplanned event resulting from system or software failure. If standby redo logs are not present, the failover is forced. A forced database failover changes one of the standby databases into the role of primary database, but may result in the loss of application data even when standby redo logs are configured on the standby database. You must reinstantiate the original primary database and all other standby databases after a forced database failover.

See also failover *and* graceful database failover

full supplemental logging

Ensures supplemental logging is set up correctly by performing log switches and then invoking the `DBMS_LOGMNR_D.BUILD` procedure.

See also supplemental logging

graceful database failover

Failover is an unplanned event resulting from system or software failure. If standby redo logs are present, the failover is graceful. A graceful database failover changes

one of the standby databases into the role of primary database, and automatically recovers some or all of the original primary data. Therefore, it is necessary to reinstantiate only the primary database. The other standby databases in the configuration do not need to be reinstantiated. A graceful database failover can occur with no data loss or with minimal data loss.

See also data loss, failover, forced database failover, *and* no data loss

hot backup

See open backup

LGWR

See log writer process (LGWR)

listener

An application that receives requests by clients and redirects them to the appropriate server.

log apply services

The component of the Data Guard environment that is responsible for maintaining the physical standby database in either a managed recovery mode or an open read-only mode and for maintaining the logical standby database in SQL apply mode.

See also log transport services, role management services, *and* SQL apply mode

log switch

The point at which LGWR stops writing to the active redo log and switches to the next available redo log. LGWR switches when either the active log is filled with redo records or you force a switch manually.

If you run your database in ARCHIVELOG mode, log transport services archive the redo data in inactive logs into archived redo logs. When a log switch occurs and LGWR begins overwriting the old redo data, you are protected against data loss because the archived redo log contains the old data. If you run in NOARCHIVELOG mode, log transport services overwrite old redo data at a log switch without archiving it. Hence, you lose all old redo data.

See also redo log

log transport services

The component of the Data Guard environment that is responsible for the automated transfer of primary database online redo logs. Log transport services provide for the management of archived redo log permissions, destinations, transmission, reception, and failure resolution. In a Data Guard environment, the log transport services component coordinates its activities with the log apply services component.

See also log apply services, no data divergence, no data loss, *and* role management services

log writer process (LGWR)

The background process that collects transaction redo and updates the online redo logs. The log writer process can also create local archived redo logs and transmit online redo logs to standby databases.

logging

See full supplemental logging *and* supplemental logging

logical standby database

A standby database that is logically identical to the primary database and can be used to take over processing if the primary database is taken offline. A logical standby database is the logical equivalent of the primary database; they share the same schema definition.

See also physical standby database *and* standby database

logical standby process (LSP)

The LSP applies archived redo log information to the logical standby database.

managed recovery mode

An environment in which the primary database automatically archives redo log files to the standby location, initiated by entering the following SQL statement:

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE;
```

When a standby database runs in managed recovery mode, it automatically applies redo logs received from the primary database.

managed recovery process (MRP)

The process that applies archived redo log information to the standby database.

managed standby environment

See standby database environment

manual recovery mode

An environment in which the primary database does not automatically archive redo log files to the standby location. In this environment, you must manually transfer archived log files to the standby location and manually apply them by issuing the following SQL statement:

```
ALTER DATABASE RECOVER STANDBY DATABASE;
```

This mode allows you to recover a standby database manually.

maximum availability mode

A log transport services data availability mode that can be set to ensure that redo logs are available at a standby database before the current database transaction is committed. Maximum availability does not protect against data divergence.

See also no data loss

maximum performance mode

A log transport services data availability mode that offers the lowest level of data protection. In this mode, the archiver process writes redo log data asynchronously to a standby database with as little effect as possible on the performance of the primary database. In a failover operation, it is possible to lose data from one or more logs that have not yet been transmitted. This is the default protection mode.

See also no data loss

maximum protection mode

A log transport services data availability mode that can be set to ensure that redo logs are available at a standby database before primary database processing can continue. Maximum protection mode protects against data divergence. This is also known as *protected mode*. This mode is not supported on logical standby databases because standby redo logs are required for this mode.

See also no data divergence *and* no data loss

MRP

See managed recovery process (MRP)

no data divergence

A log transport services option that can be configured to ensure that the primary and standby databases are always synchronized by prohibiting access to the primary database if connectivity to at least one standby database becomes unavailable.

See also log transport services, maximum availability mode, *and* maximum protection mode

no data loss

A log transport services option that can be configured to ensure that data modifications made to the primary database are not acknowledged until those data modifications are also available on (but not necessarily applied to) the standby database.

See also graceful database failover, log transport services, maximum availability mode, *and* maximum protection mode

NOARCHIVELOG mode

The mode of the database in which log transport services do not require filled online redo logs to be archived to disk. Specify the mode at database creation or change it by using the SQL ALTER DATABASE statement. Oracle Corporation does not recommend running in NOARCHIVELOG mode because it severely limits the possibilities for recovery of lost data.

See also ARCHIVELOG mode

node

See site

non-managed recovery mode

See manual recovery mode

offline redo log

See archived redo log

online redo log

A set of two or more files that records all changes made to datafiles and control files. Whenever a change is made to the database, the Oracle database server generates a redo record in the redo buffer. The LGWR process flushes the contents of the redo buffer into the online redo log.

Every database must contain at least two online redo log files. If you are multiplexing your online redo log, LGWR concurrently writes the same redo data to multiple files. The individual files are called members of an online redo log group.

See also archived redo log, current online redo log, redo log, *and* standby redo log

open backup

A backup of one or more datafiles taken while a database is open. This is also known as *hot backup*.

parent destination

A log transport services archiving destination that has a child destination associated with it.

See also child destination, destination dependency, *and* destinations

physical standby database

A standby database that is physically identical to the primary database because recovery applies changes block-for-block using the physical row ID.

See also logical standby database *and* standby database

planned downtime

See availability, downtime, *and* switchover

primary database

In a Data Guard configuration, a production database is referred to as a primary database. A primary database is used to create a standby database. Every standby database is associated with one and only one primary database. A single primary database can, however, support multiple standby databases.

See also standby database

protected mode

See maximum protection mode *and* no data divergence

read-only database

A database opened with the SQL statement `ALTER DATABASE OPEN READ ONLY`. As their name suggests, read-only databases are for queries only and cannot be modified. A standby database can be run in read-only mode, which means that it can be queried while still serving as an up-to-date emergency replacement for the primary database.

See also read-only mode

read-only mode

A standby database mode initiated by issuing the following SQL statement:

```
ALTER DATABASE OPEN READ ONLY;
```

This mode allows you to query the standby database, but not to make changes to it.

See also read-only database

receiving instance

When you use a standby database in a Real Application Clusters configuration, any instance can receive archived logs from the primary database; this is the receiving instance.

See also recovery instance

recovery catalog

A set of tables and views used by Recovery Manager (RMAN) to store information about Oracle databases. RMAN uses this data to manage the backup, restore, and recovery of Oracle databases. If you choose not to use a recovery catalog, RMAN uses the target database control file. You should not store the recovery catalog in your target database.

See also recovery catalog database *and* Recovery Manager (RMAN)

recovery catalog database

An Oracle database that contains a recovery catalog schema.

See also recovery catalog

recovery instance

The node where managed recovery is performed. Within a Real Application Clusters configuration, each primary instance directs its archived redo logs to this node of the standby cluster.

See also cross-instance archival environment *and* receiving instance

Recovery Manager (RMAN)

A utility that backs up, restores, and recovers Oracle databases. You can use it with or without the central information repository called a *recovery catalog*. If you do not use a recovery catalog, RMAN uses the database's control file to store information

necessary for backup and recovery operations. You can use RMAN in conjunction with a media manager, which allows you to back up files to tertiary storage.

See also backup piece, backup set, *and* recovery catalog

redo log

A file containing redo records. There are three types of redo logs: online redo logs, standby redo logs, and archived redo logs.

The online redo log is a set of two or more files that records all changes made to datafiles and control files. The LGWR process records the redo records in the log. The current online redo log is the one to which LGWR is currently writing.

The standby redo log is an optional location where the standby database can store the redo data received from the primary database. This redo data can be stored on the standby location using either standby redo logs or archived redo logs.

The archived redo log, also known as the *offline redo log*, is a copy of the online redo log that has been copied to an offline destination. If the database is in ARCHIVELOG mode, the ARC*n* process or processes copy each online redo log to one or more archive log destinations after it is filled.

See also archived redo log, ARCHIVELOG mode, current online redo log, log switch, online redo log, *and* standby redo log

reliability

The ability of a computing system or software to operate without failing.

remote file server (RFS)

The remote file server process on the standby location receives archived redo logs from the primary database.

RMAN

See Recovery Manager (RMAN)

role management services

The component of the Data Guard environment that is responsible for the changing of database roles. Database role transitions include switchover and switchback, as well as failover if the primary database is unavailable due to an unplanned shutdown.

See also log apply services *and* log transport services

role transition

A database can be in one of two mutually exclusive roles: primary or standby. You can change these roles dynamically as a planned transition (switchover) or you can change these roles as a result of an unplanned failure (failover).

See also failover *and* switchover

rolling upgrade

A software installation technique that allows a clustered system to continue to provide service while the software is being upgraded to the next release. This process is called a rolling upgrade because each database or system in the cluster is upgraded and rebooted in turn, until all databases or systems have been upgraded.

site

In a Data Guard configuration, this term is sometimes used to refer to the local or geographically remote location of a primary or standby database.

In a Data Guard broker configuration, a site is a managed unit of failover.

SQL apply mode

The mode in which log apply services automatically apply archived redo log information to the logical standby database by transforming transaction information into SQL statements (using LogMiner technology) and applying the SQL statement to the logical standby database.

See also log apply services

standby database

An identical copy of a primary database that you can use for disaster protection. You can update your standby database with archived redo logs from the primary database to keep it current. Should a disaster destroy or compromise the primary database, you can fail over to the standby database and make it the new primary database. A standby database has its own initialization parameter file, control file, and datafiles.

See also logical standby database, physical standby database, *and* primary database

standby database environment

The physical configuration of the primary and standby databases. The environment depends on many factors, including the:

- Number of standby databases associated with a primary database

- Number of host systems used by the databases
- Directory structures of the databases
- Network configuration

A configuration in which a primary database automatically archives redo logs to a standby location is a *managed standby environment*. If the standby database is in managed recovery mode, it automatically applies the logs received from the primary database to the standby database. Note that in a managed standby environment, a primary database continues to transmit archived logs even if the standby database is not in managed recovery mode.

standby redo log

The standby redo log is an optional set of log files where the standby database can store the redo data received from the primary database. (Redo data can also be stored on the standby location using archived redo logs.) Standby redo logs are created using the `ADD STANDBY LOGFILE` clause of the SQL `ALTER DATABASE` statement. Additional log group members can be added later to provide another level of reliability against disk failure on the standby location. Standby redo logs are required if you are using the maximum protection mode. Standby redo logs are not supported for logical standby databases.

See also redo log

supplemental logging

The ability to log additional information in the redo log stream to enable LogMiner to group and merge the redo streams related to a row change during log mining, and also to be able to identify the row using an identification key.

See also full supplemental logging

switchback

A switchover performed in reverse that results in the original primary database becoming a new primary database. Once you have performed a database switchover operation, you can switch back to your original Data Guard configuration.

See also switchover

switchover

The process of intentionally switching a database role from primary to standby, as well as from standby to primary, without resetting the online redo logs of the new primary database. This is a switchover operation, instead of a failover operation,

because there is no loss of application data, and there is no need to reinstantiate the standby databases, including other standby databases not involved in the switchover operation. You cannot use a switchover operation to perform a rolling upgrade of Oracle software. However, it may be possible to use a switchover operation to perform a hardware-based rolling upgrade.

See also availability, downtime, failover, role transition, *and* switchback

system change number (SCN)

A stamp that defines a committed version of a database at a point in time. The Oracle database server assigns every committed transaction a unique SCN.

tablespace

One or more logical storage units into which a database is divided. Each tablespace has one or more physical datafiles exclusively associated with it.

See also datafile

TAF

See transparent application failover (TAF)

target database

In RMAN, the database that you are backing up or restoring.

tempfile

A file that belongs to a temporary tablespace, and is created with the `TEMPFILE` option. Temporary tablespaces cannot contain permanent database objects such as tables, and are typically used for sorting. Because tempfiles cannot contain permanent objects, RMAN does not back them up.

See also temporary tablespace

temporary tablespace

Tablespace of temporary tables created during the processing of a SQL statement. This allows you to add tempfile entries in read-only mode for the purpose of making queries. You can then perform on-disk sorting operations in a read-only database without affecting dictionary files or generating redo entries.

See also tablespace *and* tempfile

transparent application failover (TAF)

The ability of client applications to automatically reconnect to a database and resume work after a failover occurs.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V