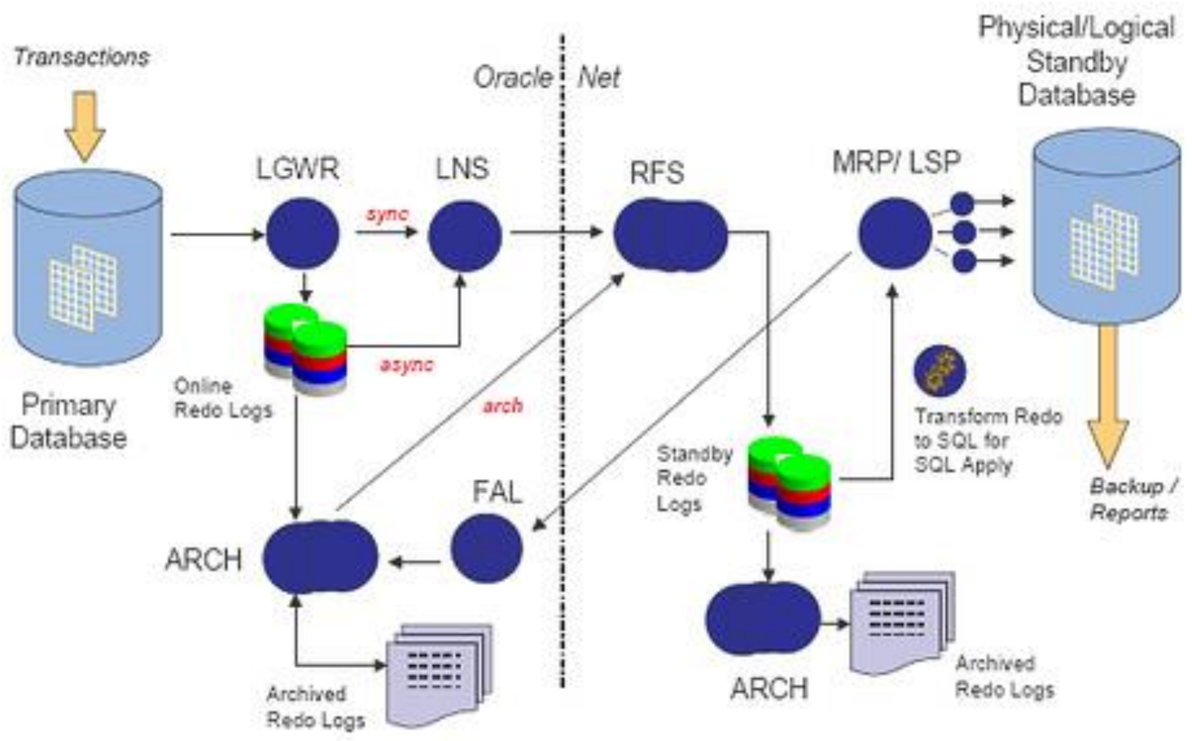
# Oracle Dataguard Architecture



## PHYSICAL STANDBY PROCESSES ARCHITECTURE (APPLY REDO LOGS)

On the primary database site, the **log writer process (LGWR)** collects transactions from the log buffer and writes to the online redo logs. The **archiver process (ARCH)** creates a copy of the online redo logs, and writes to the local archive destination. Depending on the configuration, the archiver process or log writer process can also transmit redo logs to standby database. When using the log writer process, you can specify synchronous or asynchronous network transmission of redo logs to remote destinations. Data Guard achieves ***asynchronous*** network I/O using **LGWR network server process (LNS)**. These network severs processes are deployed by LOG_ARCHIVE_DEST_n initialization parameter. Data Guard's asynchronous log transport (i.e. the Maximum Performance mode) is recommended for a configuration in which the network distance is up to thousands of miles, providing continual maximum performance, while minimizing the risks of transaction loss in the event of a disaster.

On the standby database site, the **remote file server process (RFS)** receives archived redo logs from the primary database. The primary site launches the RFS process during the first log transfer. The redo logs information received by the RFS process can be stored as either standby redo logs or archived redo logs. Data Guard introduces the concept of standby redo logs (separate pool of log file groups). Standby redo logs must be archived by the **ARCH process** to the standby archived destination before the **managed recovery process (MRP)** applies redo log information to the standby database.

The **fetch archive log (FAL) client** is the MRP process.  The **fetch archive log (FAL) server** is a foreground process that runs on the primary database and services the fetch archive log requests coming from the FAL client.  A separate FAL server is created for each incoming FAL client.

FAL_SERVER specifies the FAL (fetch archive log) server for a standby database. The value is an Oracle Net service name, which is assumed to be configured properly on the standby database system to point to the desired FAL server.
FAL_CLIENT specifies the FAL (fetch archive log) client name that is used by the FAL service, configured through the FAL_SERVER parameter, to refer to the FAL client. The value is an Oracle Net service name, which is assumed to be configured properly on the FAL server system to point to the FAL client (standby database).

Thanks to the FAL_CLIENT and FAL_SERVER parameters, the managed-recovery process in the physical database will automatically check and resolve gaps at the time redo is applied. This helps in the sense that you don't need to perform the transfer of those gaps by yourself.
FAL_CLIENT and FAL_SERVER only need to be defined in the initialization parameter file for the standby database(s). It is possible; however, to define these two parameters in the initialization parameter for the primary database server to ease the amount of work that would need to be performed if the primary database were required to transition its role.

Prior to Oracle 11g, Redo Apply only worked with the standby database in the MOUNT state, preventing queries against the physical standby whilst media recovery was in progress. This has changed in Oracle 11g.

When using Data Guard Broker (DG_BROKER_START = TRUE), the monitor agent process named **Data Guard Broker Monitor (DMON)** is running on every site (primary and standby) and maintain a two-way communication.

## LOGICAL STANDBY PROCESSES ARCHITECTURE (REDO LOGS CONVERTED TO SQL, CALLED SQL APPLY)

The major difference between the logical and physical standby database architectures is in its log apply services. On Logical Standby, you can query it while simultaneously applying transactions from the primary. This is ideal for business that requires a near real-time copy of your production DB for reporting.

The key advantage for logical standby databases is that **they're opened read/write**, even while they're in applied mode. That is, they can be used to generate reports and the like. It is indeed a fully functional database. Also, additional indexes, materialized views and so on can be created.

Oracle (or more exactly the log apply services) uses the primary database's redo log, transforms them into SQL statements and replays them on the logical standby database. SQL Apply uses LOGMINER technology to reconstruct DML statements from the redo generated on the primary.

The **logical standby process (LSP)** is the coordinator process for two groups of **parallel execution process (PX)** that work concurrently to read, prepare, build, and apply completed

SQL transactions from the archived redo logs sent from the primary database. The first group of PX processes read log files and extract the SQL statements by using Log Miner technology; the second group of PX processes apply these extracted SQL transactions to the logical standby database. The mining and applying process occurs in parallel. Logical standby database does not use standby online redo logs. Logical standby database does not have FAL capabilities in Oracle. All gaps are resolved by the proactive gap resolution mechanism running on the primary that polls the standby to see if they have a gap.

## Dataguard Services

a) Log Transport
b) Log Apply
c) Role Management

## Dataguard Process

a) RFS: Remote File Server ➔ RFS receives the redo records from the archiver or the log writer process of the primary database over Oracle Net and writes to filesystem on the standby site.

b) FAL: Fetch Archive Log ➔ The FAL process has two components: FAL Client and FAL Server. Both processes are used for archive gap resolution. If the Managed Recovery Process (MRP) on the standby db site detects an archive gap sequence, it initiates a fetch request to the FAL client on the standby site. This action, in turn, requests the FAL server process on the primary database to re-transmit the archived log files to resolve the gap sequence.

c) MRP: Managed Recovery Process ➔ the standby database server will use the Managed Recover Process (MRP) to apply the redo information if the standby database is a physical standby.

d) LSP: Logical Standby Process ➔ the standby database server will use the Logical Standby Process (LSP) to apply redo information if the standby database is a logical standby.

## On the primary database server 4 processes are involved -

2. LGWR - LGWR is used for SYNC and ASYNC modes (maximum protection and maximum availability)
3. ARCH - ARCH is used only for ASYNC mode (maximum performance) (you can choose between LGWR and ARCH. For the same primary database you can use LGWR to send redo to some standby dbs and ARCH to other dbs)
4. LNS (Log-write network server)
5. FAL (fetch archive log) this process runs on the primary database server and only executes to resolve gaps.

## On the standby database server 4 processes are involved -

1. RFS - Remote File server process that receives redo from primary database
2. ARCH - writes the standby redo log files to archive redo log files.
3. MRP (managed recovery process - for physical standby database only)
4. LSP (logical standby process - for logical standby database only)

## Difference between logical & physical standby database?

| **Physical standby database** | **Logical standby database** |
|---|---|
| Physical standby schema matches exactly the source database. Provides a physically identical copy of the primary database, with on disk database structures that are identical to the primary database on a block-for-block basis. | Logical standby database does not have to match the schema structure of the source database. Contains the same logical information as the production database, although the physical organization and structure of the data can be different |
| A physical standby database is kept synchronized with the primary database by recovering the redo data received from the primary database. | The logical standby database is kept synchronized with the primary database by transforming the data in the redo received from the primary database into SQL statements and then executing the SQL statements on the standby database. |
| A physical standby is mounted and generally cannot be used for any other purposes. You can however switch temporarily to read-only mode and query it but synchronization with the primary will be paused until you return to the recovery mode. In 10g R2 you can also open it read-write and then flashback to its original state | A logical standby database can be used concurrently for data protection, reporting, and database upgrades. |

## Real Time Apply

With Data Guard's new Real Time Apply feature in Oracle Database 10*g*, redo data can be applied on the standby database (whether Redo Apply or SQL Apply) as soon as the redo data is written to a Standby Redo Log (SRL). Prior releases of Data Guard require this redo data to be archived at the standby database in the form of archive logs before they can be applied.

```
-------- Physical Standby ------------------
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING
     CURRENT LOGFILE;

-------- Logical Standby --------------------
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

## Data Protection Modes

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

| **Protection Mode** | **Risk of Data Loss In the Event of a Disaster** | **Redo Transport Mechanism** |
|---|---|---|
| *Maximum Protection* | Zero data loss; Double failure protection | LGWR SYNC |
| *Maximum Availability* | Zero data loss; Single failure protection | LGWR SYNC |
| *Maximum Performance (default)* | Minimal data loss – usually 0 to few seconds | LGWR ASYNC or ARCH |

| | Maximum Protection | Maximum Availability | Maximum Performance |
|---|---|---|---|
| Redo Archival process | LGWR | LGWR | LGWR or ARCH |
| Network Transmission Mode | SYNC | SYNC | ASYNC when using LGWR process. SYNC when using ARCH process |
| Disk write option | AFFIRM | AFFIRM | AFFIRM or NOAFFIRM |
| Standby Redo log required | YES | YES | NO, but its recommended |

### Maximum Protection

1. No data loss
2. Redo has to be written to both Primary redo logs and standby redo logs (of atleast one standby database) before transaction commits
3. Primary database shuts down if redo stream is prevented to write at standby redo logs of atleast one standby database
4. Configure standby redo logs on at least one standby database
5. Attribute to use in log_archive_dest_n : LGWR, SYNC and AFFIRM for at least one standby DB

```
LOG_ARCHIVE_DEST_2='SERVICE=standby SYNC AFFIRM
DB_UNIQUE_NAME=standby VALID_FOR=(ALL_LOGFILES,PRIMARY_ROLE)'
```

Maximum Availability

1. Redo has to be written to both Primary redo logs and standby redo logs (of atleast one standby database) before transaction commits
2. If redo stream is prevented to write at standby redo logs of at least one standby database then Primary database does not shuts down unlike Maximum protection mode, instead primary database operates in Maximum Performance mode.
3. Primary database automatically resumes to operate in maximum availability mode once all gaps are resolved.
4. Configure standby redo logs on at least one standby database
5. Attribute to use in log_archive_dest_n : LGWR, SYNC and AFFIRM for at least one standby DB

```
LOG_ARCHIVE_DEST_2='SERVICE=standby SYNC AFFIRM NET_TIMEOUT=30
DB_UNIQUE_NAME=standby VALID_FOR=(ALL_LOGFILES,PRIMARY_ROLE)'
```

Maximum Performance

1. Default mode
2. Asynchronous redo data is written to at least one standby database
3. Attributes on log_archive_dest_n to set either LGWR and ASYNC or ARCH for standby DB destination

```
LOG_ARCHIVE_DEST_2='SERVICE=standby ASYNC NOAFFIRM
DB_UNIQUE_NAME=standby VALID_FOR=(ALL_LOGFILES,PRIMARY_ROLE)'
```

## AFFIRM/NOAFFIRM, SYNC/ASYNC of log_archive_dest_n

According to the documentation, affirm means right after RFS writes redo into SRLs, the standby notifies the primary the redo has been received. Noaffirm means make this acknowledgement even before writing to SRLs (which could fail, hence the name noaffirm).

So affirm or noaffirm is about acknowledging receipt of redo write on the standby redo logs after or before redo write. This is different from sync and async. Sync means the primary is not allowed to commit unless the standby sends back the signal saying it has received the redo. Async means commit can proceed regardless whether the standby has acknowledged that or not.

So the two concepts, (no)affirm and (a)sync, are both related to receipt of redo on the standby. However, (no)affirm is an attribute to control the disk I/O behavior of the standby, while (a)sync is to control the primary to standby network behavior.

## Initialization parameters

```
---Primary Initialization parameters: stg_dg1

db_name='STG'
db_unique_name= STG_DG1
LOG_ARCHIVE_CONFIG='DG_CONFIG= (stg_dg1, stg_dg2)'

LOG_ARCHIVE_DEST_1='LOCATION=/u01/app/oracle/admin/ ecmstgdg1/arch
VALID_FOR=(ALL_LOGFILES,ALL_ROLES) DB_UNIQUE_NAME=stg_dg1'

LOG_ARCHIVE_DEST_2='SERVICE=stg VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=stg_dg2'

LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_STATE_2=ENABLE
LOG_ARCHIVE_MAX_PROCESSES=2

log_archive_format='%t_%s_%r.dbf' #SPECIFIC TO STANDBY ROLE

STANDBY_FILE_MANAGEMENT=AUTO
STANDBY_ARCHIVE_DEST='/u01/app/oracle/admin/stgdg1/arch'

FAL_SERVER=stg_dg2
FAL_CLIENT=stg_dg1




---Standby Initialization parameters

db_name='STG'
db_unique_name= STG_DG2



---COMMON TO BOTH PRIMARY AND STANDBY ROLES
LOG_ARCHIVE_CONFIG='DG_CONFIG= (stg_dg1, stg_dg2)'
LOG_ARCHIVE_DEST_1='LOCATION=/u01/app/oracle/admin/stgdg2/arch VALID_FOR=
(ALL_LOGFILES, ALL_ROLES) DB_UNIQUE_NAME=stg_dg2'
LOG_ARCHIVE_DEST_2='SERVICE=ecmstg LGWR ASYNC VALID_FOR= (ONLINE_LOGFILES,
PRIMARY_ROLE) DB_UNIQUE_NAME=stg_dg1'

LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_STATE_2=ENABLE
LOG_ARCHIVE_MAX_PROCESSES=2
log_archive_format='%t_%s_%r.dbf' #SPECIFIC TO STANDBY ROLE

STANDBY_FILE_MANAGEMENT=AUTO
STANDBY_ARCHIVE_DEST='/u01/app/oracle/admin/stgdg2/arch'

FAL_SERVER=stg_dg1
FAL_CLIENT=stg_dg2
```