



# Oracle (Active) Data Guard

Master Slide Deck - Updated 2025.03.21

---

## Ludovico Caldara

Senior Principal Product Manager

Oracle Database High Availability (HA), Scalability and  
Maximum Availability Architecture (MAA) Team

@ludodba



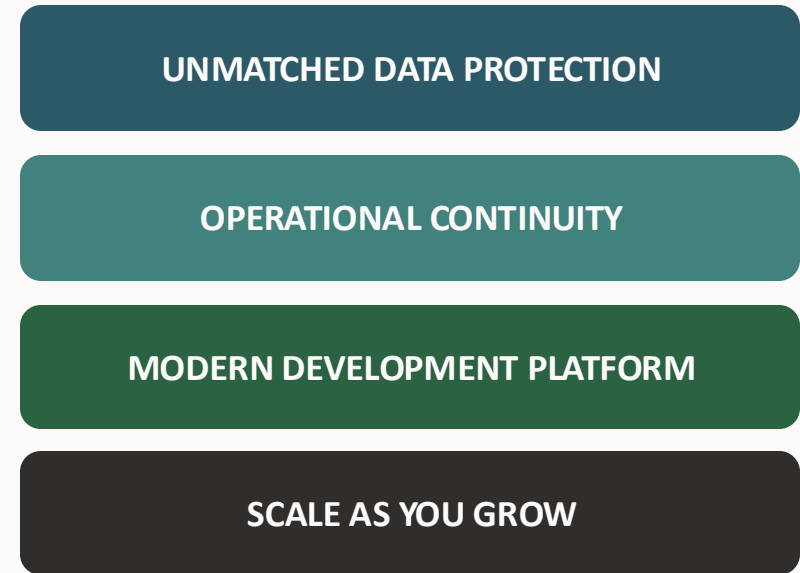
<http://www.linkedin.com/in/ludovicocaldara>



[www.ludovicocaldara.net](http://www.ludovicocaldara.net)

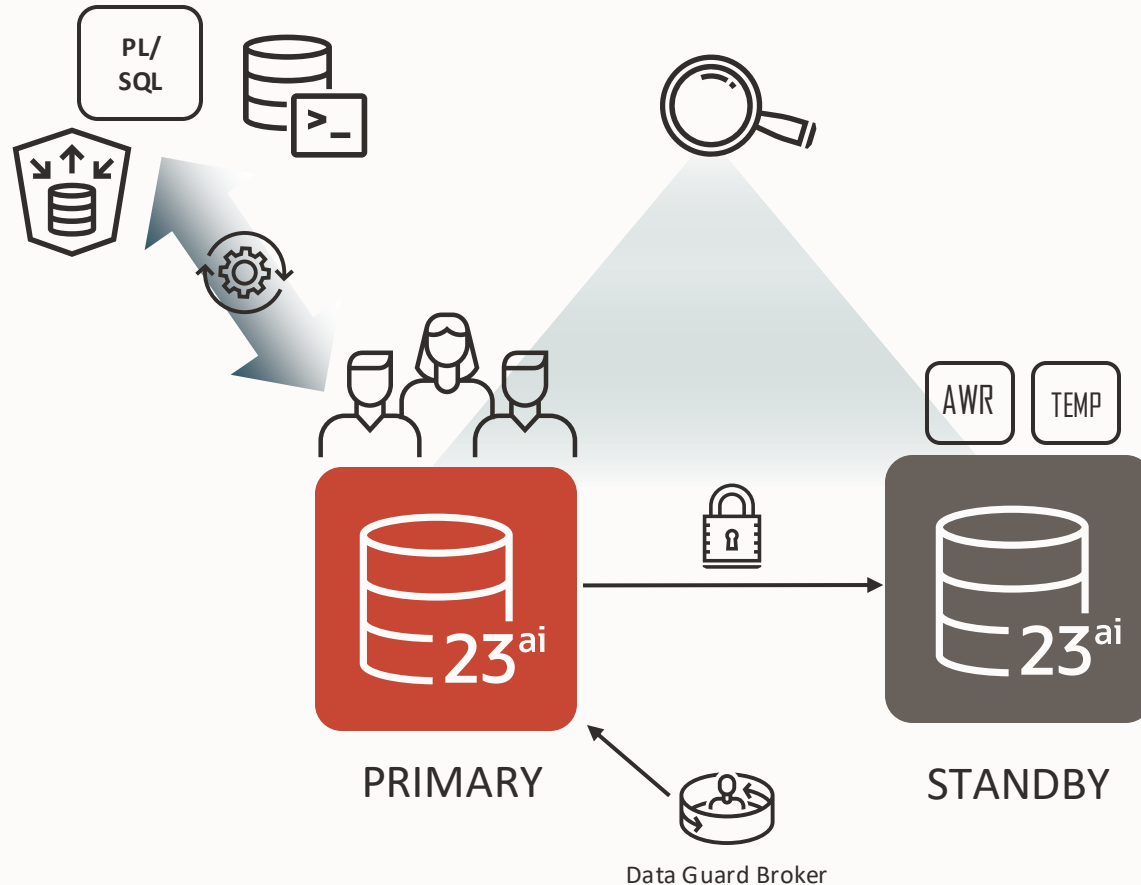










Come for the data protection, stay for the data management



# Active Data Guard 23<sup>ai</sup> furtherly enhances the experience

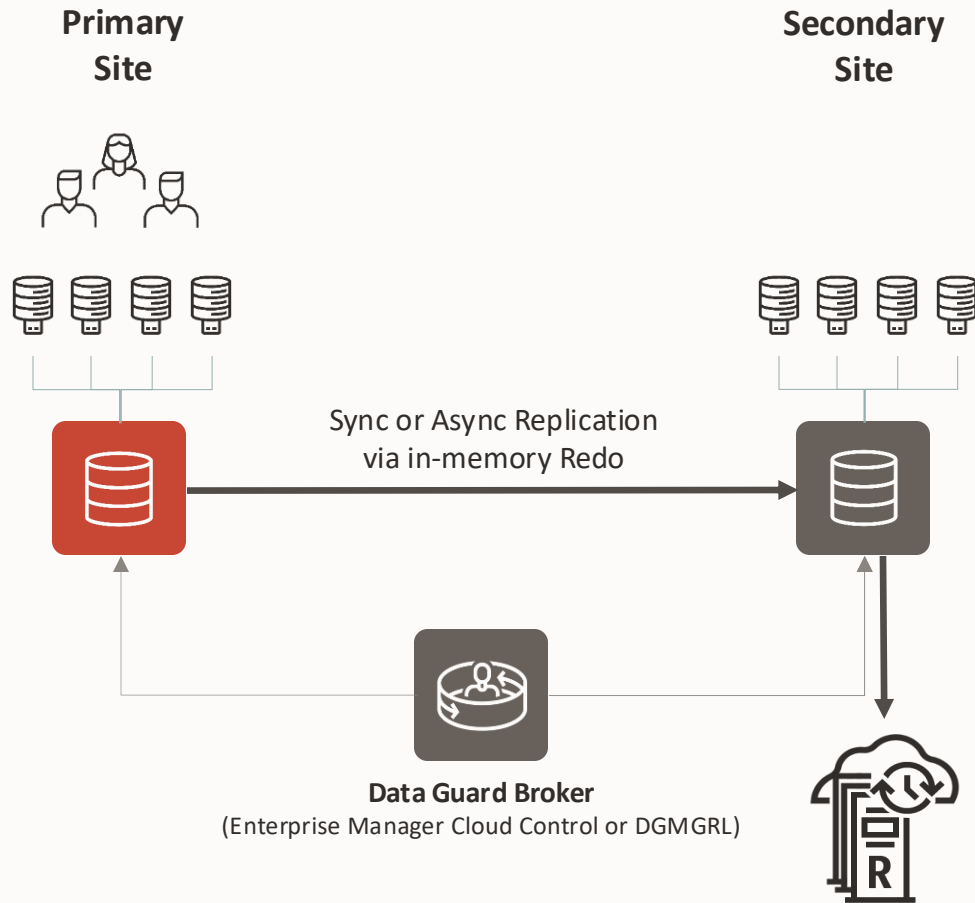
Enhanced performance, observability, and manageability



-  Automatic preparation of the primary
-  PL/SQL APIs for better automation
-  SQLcl command-line integration
-  REST APIs for easier DevOps integration
-  4 new SQL views to monitor Data Guard
-  Support for mixed Transparent Data Encryption
-  Automatic temp file creation on the standby
-  Simplified AWR snapshots on the standby DB

# Oracle Data Guard Overview

# Oracle Data Guard



- **Disaster Recovery (included with DB EE)**
  - License primary and secondary sites
- **Active-passive**
  - Standby is used mostly for failovers
- **Automatic failover to Standby site**
- **Zero / near-zero data loss**
- **Continuous data validation**
- **Simple migrations and upgrades**

<https://www.oracle.com/goto/dataguard>

# Data Guard

## Capabilities Included with Oracle Database Enterprise Edition (EE)



Zero or sub-second data loss protection

Strong isolation using continuous Oracle validation

Lost-write detection

Universal support – all data types and applications

Comprehensive monitoring with Enterprise Manager

Automatic database failover

Automatic client failover

Standby-first patch apply

Database rolling maintenance

Select platform migrations

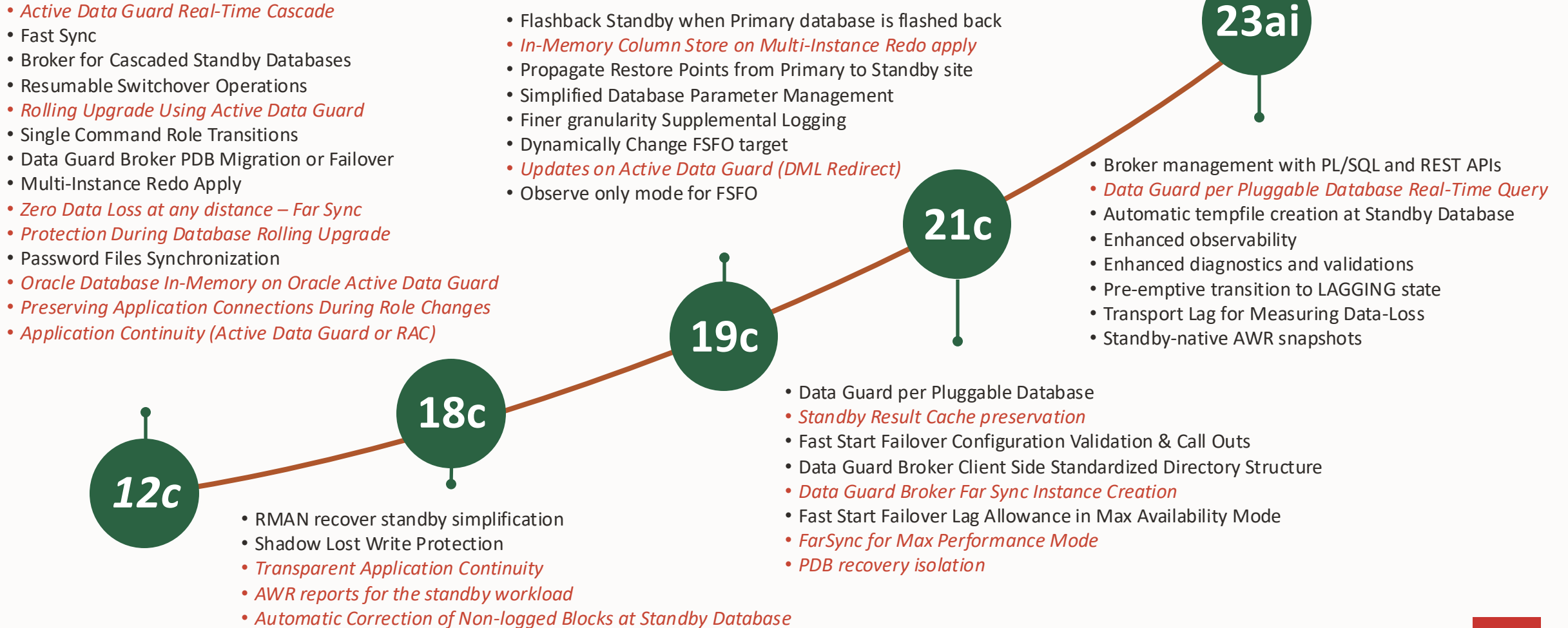
Extreme throughput - supports all workloads

Dual-purpose standby for development and test

Integrated management

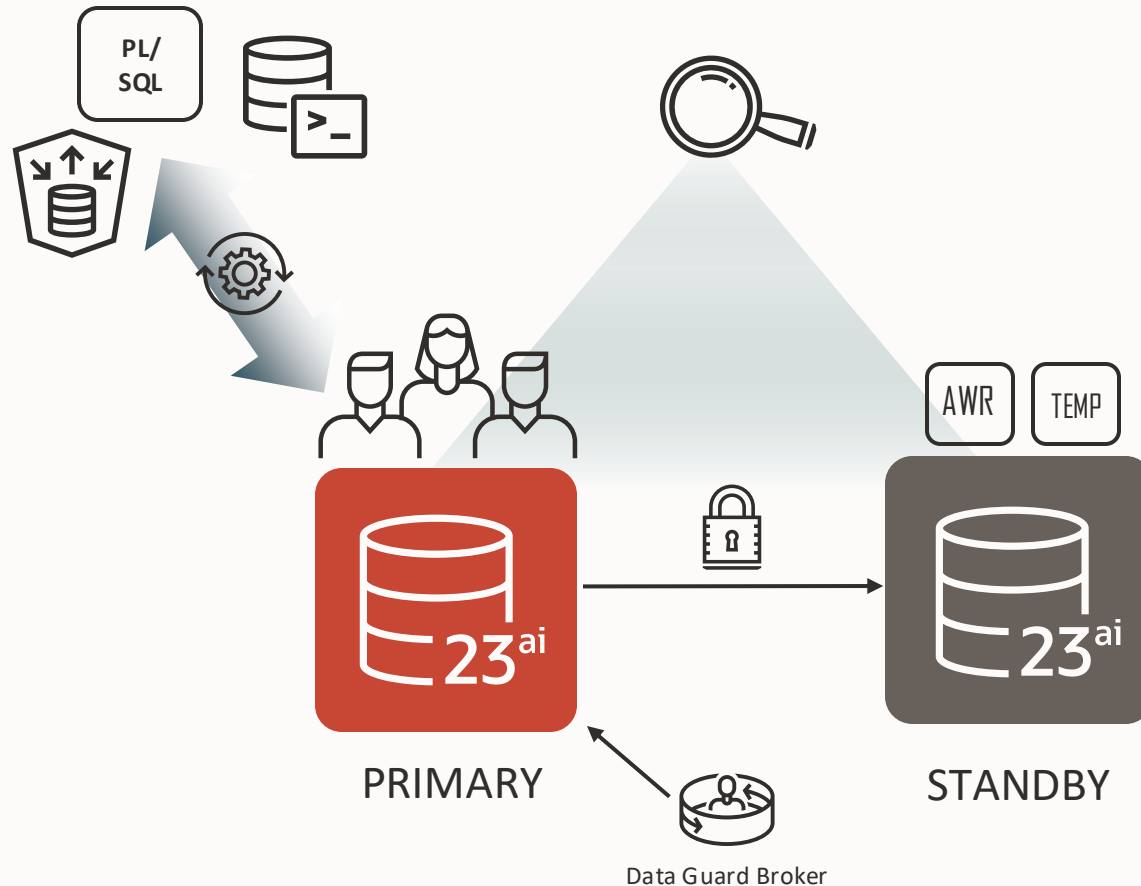
# Oracle *Active* Data Guard









Actively protecting data for the future *both* on-premises and in the cloud



## Active Data Guard 23ai furtherly enhances the experience

## Enhanced performance, observability, and manageability



-  Automatic preparation of the primary
-  PL/SQL APIs for better automation
-  SQLcl command-line integration
-  REST APIs for easier DevOps integration
-  4 new SQL views to monitor Data Guard
-  Support for mixed Transparent Data Encryption
-  Automatic temp file creation on the standby
-  Simplified AWR snapshots on the standby DB

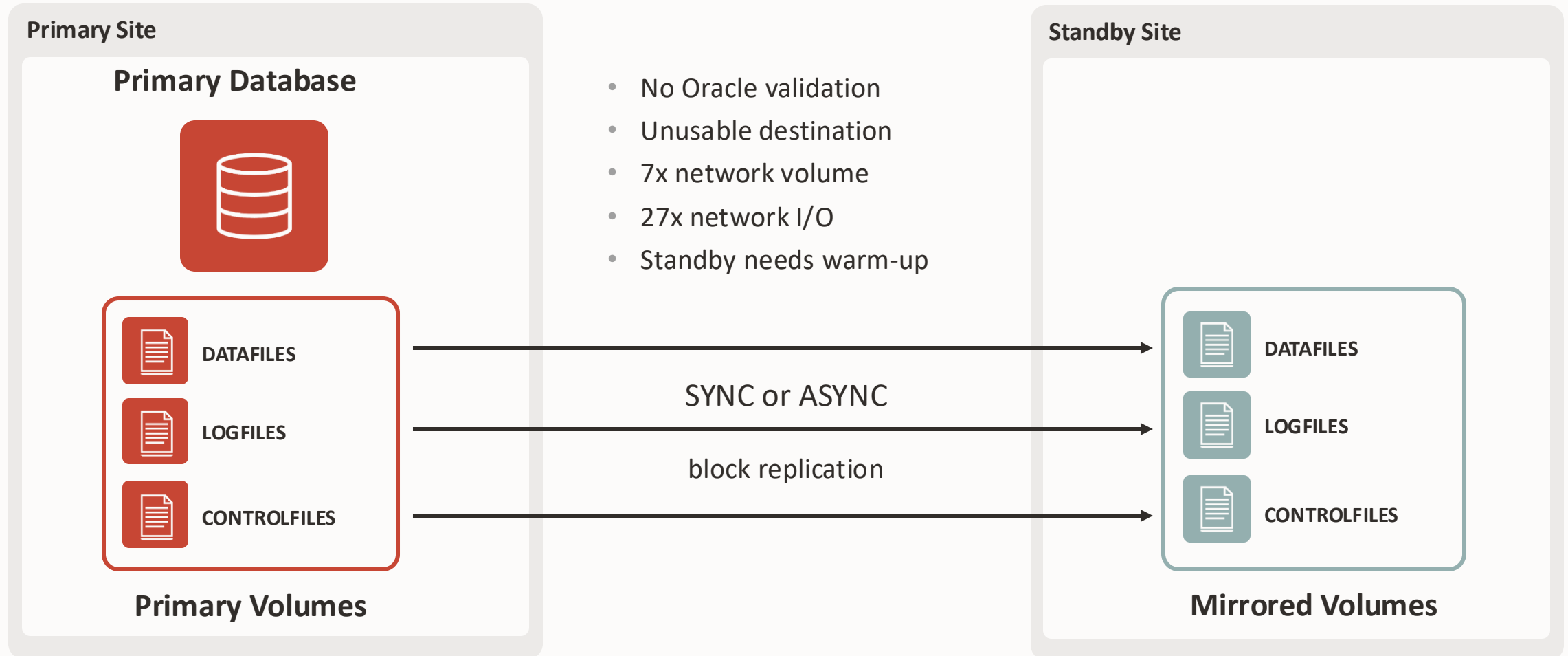




# Oracle Data Guard vs Storage Replication

# Storage Remote Mirroring Architecture

Mirrors every write to every file including those that are **corrupted or encrypted by ransomware**



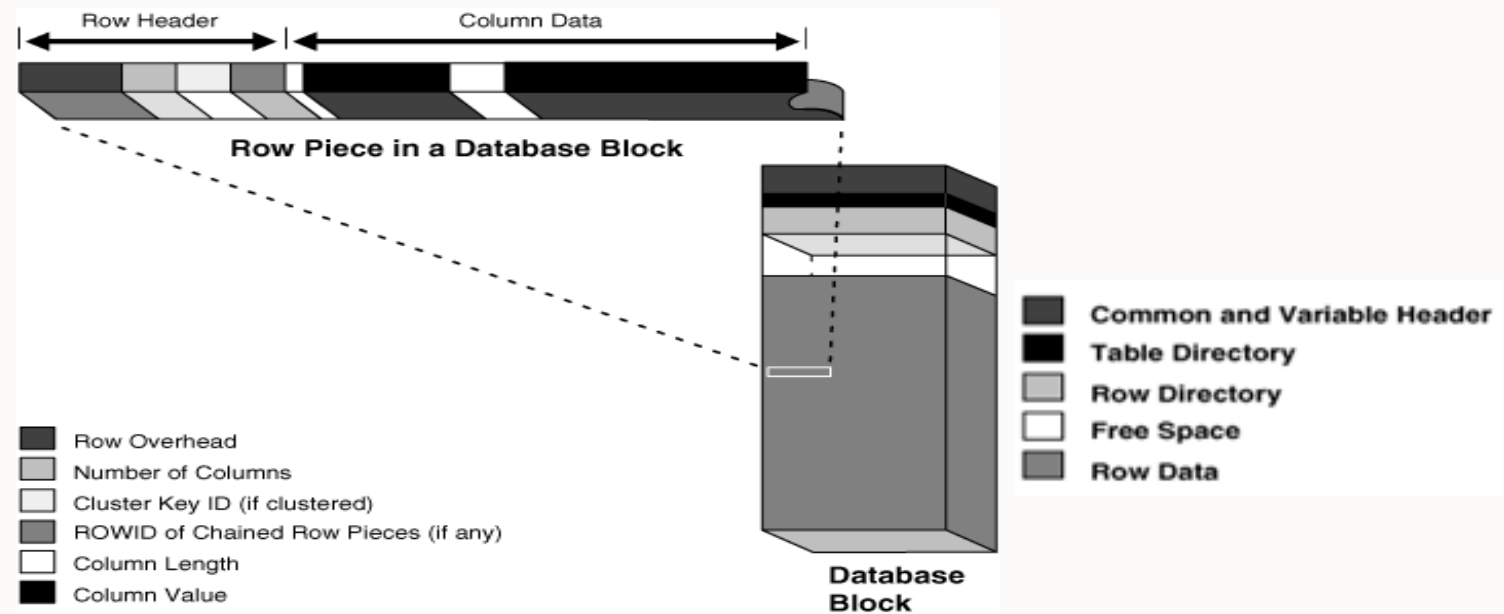
# Data Guard Does What Storage Mirroring Can't

Isolate Corruption, Protect Data, Maintain Availability

Storage Remote Mirroring...  
blocks are just bits on a disk

Data Guard uses physical and logical  
data consistency checks for end-to-end data integrity

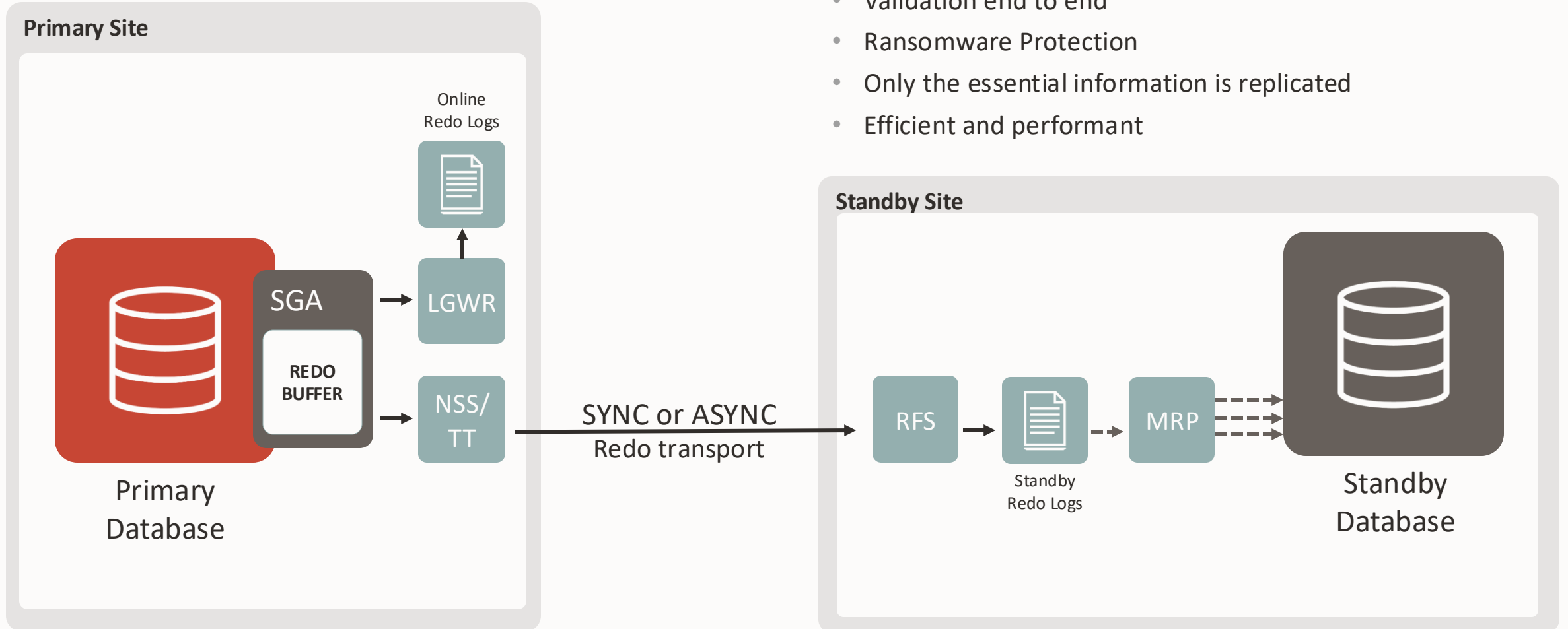
```
Block 3941 (0x0f65)
 0 1 2 3 4 5 6 7 8 9 a b c d e f
000 20 20 20 54 77 61 73 20 74 68 65 20 6e 69 67 68
010 74 20 62 65 66 6f 72 65 20 73 74 61 72 74 2d 75
020 70 20 61 6e 64 20 61 6c 6c 20 74 68 72 6f 75 67
030 68 20 74 68 65 20 6e 65 74 2c 0a 20 20 20 20 20
040 6e 6f 74 20 61 20 70 61 63 6b 65 74 20 77 61 73
050 20 6d 6f 76 69 6e 67 3b 20 6e 6f 20 62 69 74 20
060 6e 6f 72 20 6f 63 74 65 74 2e 0a 20 20 20 54 68
070 65 20 65 6e 67 69 6e 65 65 72 73 20 72 61 74 74
080 6c 65 64 20 74 68 65 69 72 20 63 61 72 64 73 20
090 69 6e 20 64 65 73 70 61 69 72 2c 0a 20 20 20 20
0a0 20 68 6f 70 69 6e 67 20 61 20 62 61 64 20 63 68
0b0 69 70 20 77 6f 75 6c 64 20 62 6c 6f 77 20 77 69
0c0 74 68 20 61 20 66 6c 61 72 65 2e 0a 20 20 20 54
0d0 68 65 20 73 61 6c 65 73 6d 65 6e 20 77 65 72 65
0e0 20 6e 65 73 74 6c 65 64 20 61 6c 6c 20 73 6e 75
0f0 67 20 69 6e 20 74 68 65 69 72 20 62 65 64 73 2c
100 0a 20 20 20 20 20 20 77 68 69 6c 65 20 76 69 73 69
110 6f 6e 73 20 6f 66 20 64 61 74 61 20 6e 65 74 73
120 20 64 61 6e 63 65 64 20 69 6e 20 74 68 65 69 72
130 20 68 65 61 64 73 2e 0a 20 20 20 41 6e 64 20 49
140 20 77 69 74 68 20 6d 79 20 64 61 74 61 73 63 6f
150 70 65 20 74 72 61 63 69 6e 67 73 20 61 6e 64 20
160 64 75 6d 70 73 0a 20 20 20 20 20 70 72 65 70 61
170 72 65 64 20 66 6f 72 20 73 6f 6d 65 20 70 72 65
180 74 74 79 20 62 61 64 20 62 72 75 69 73 65 73 20
190 61 6e 64 20 6c 75 6d 70 73 2e 0a 20 20 20 57 68
1a0 65 6e 20 6f 75 74 20 69 6e 20 74 68 65 20 68 61
1b0 6c 6c 20 74 68 65 72 65 20 61 72 6f 73 65 20 73
1c0 75 63 68 20 61 20 63 6c 61 74 74 65 72 2c 0a 20
1d0 20 20 20 20 49 20 73 70 72 61 6e 67 20 66 72 6f
1e0 6d 20 6d 79 20 64 65 73 6b 20 74 6f 20 73 65 65
1f0 20 77 68 61 74 20 77 61 73 20 74 68 65 20 6d 61
```



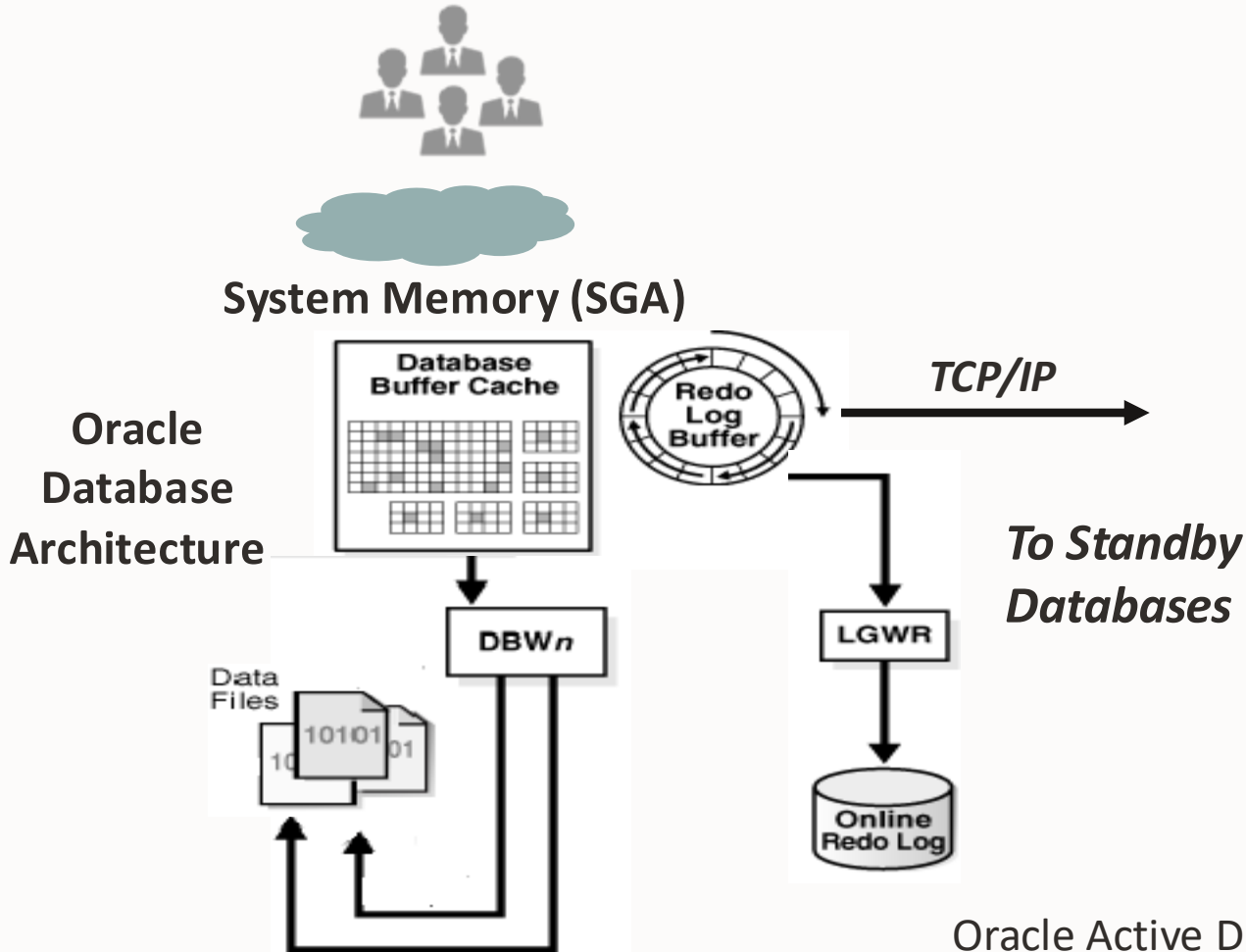
See My Oracle Support Note 1302539.1 for details

# Data Guard is optimized for the database

It efficiently maintains a **physical copy** of production and **guarantees its integrity**



# Data Guard Provides Strongest Fault Isolation and Best Performance



Data Guard transmits redo blocks directly from SGA: like a *memcpy* over the network

Redo received / applied by running Oracle instance: continuous Oracle-integrated data validation

- Best isolation from lower layer faults
- Best performance since no disk I/O
- Best network utilization: only redo sent
- Transactional consistency: always
- Corrupted blocks auto-repaired \*
- Database-integrated application failover

\* Requires Active Data Guard License

Oracle Active Data Guard Compared to Storage Remote Mirroring

<https://www.oracle.com/a/tech/docs/adg-vs-storage-mirroring.pdf>

Oracle Replication done right

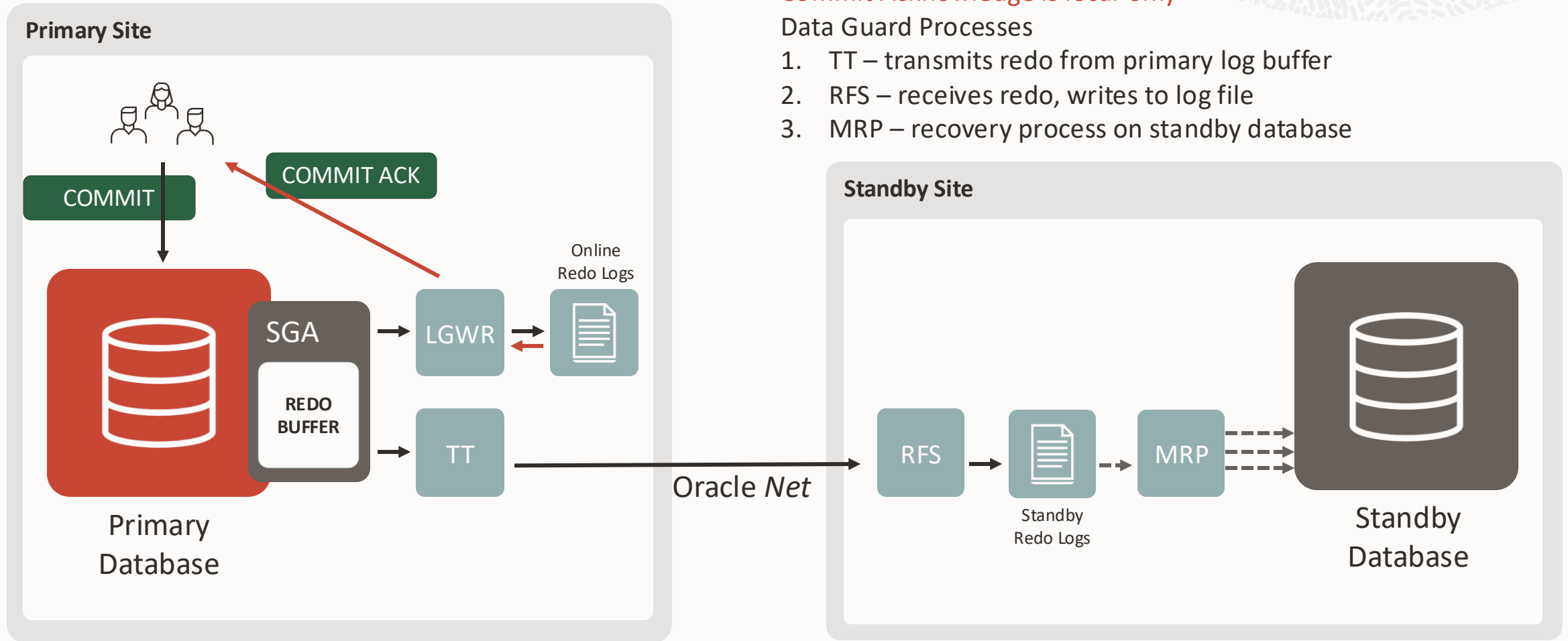
<https://blogs.oracle.com/maa/replication-done-right>

# Oracle Data Guard Redo Transport



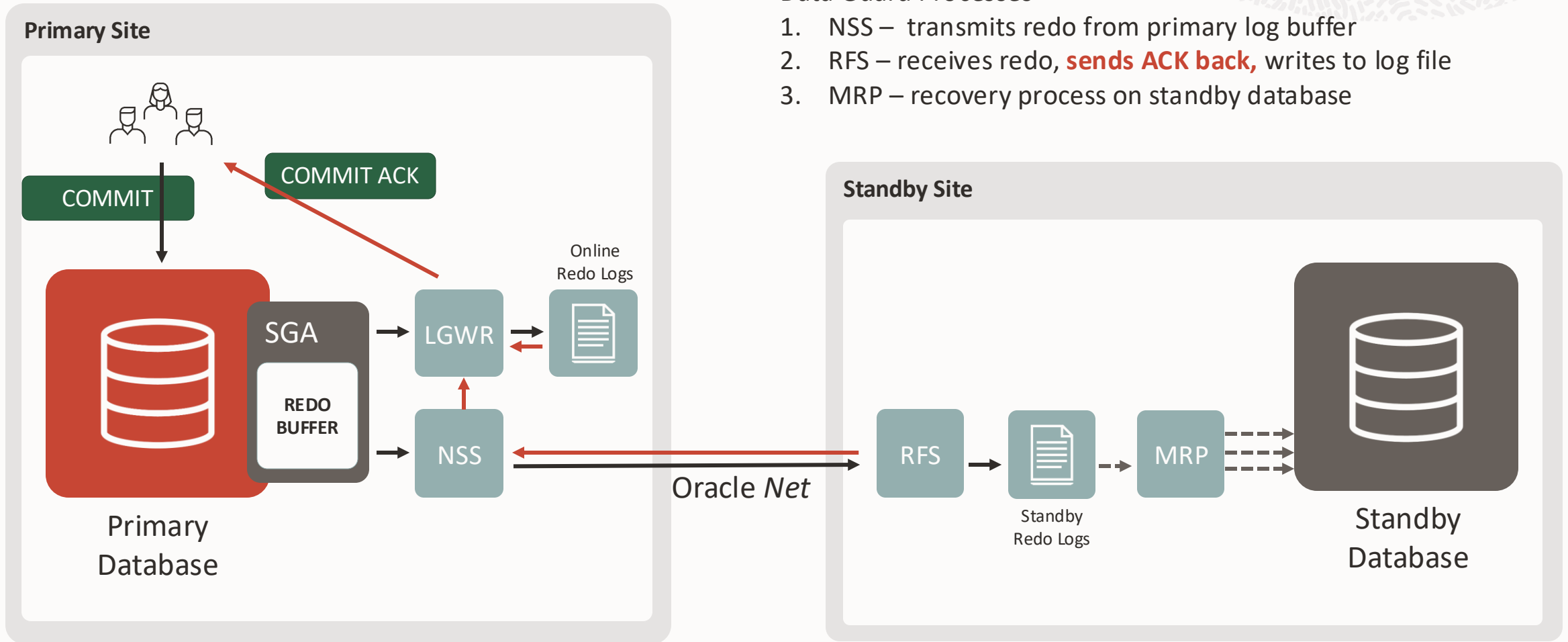
# Data Guard Transport for Best Performance

## Data Guard **ASYNC** Process Architecture



# Data Guard Transport for Zero Data Loss

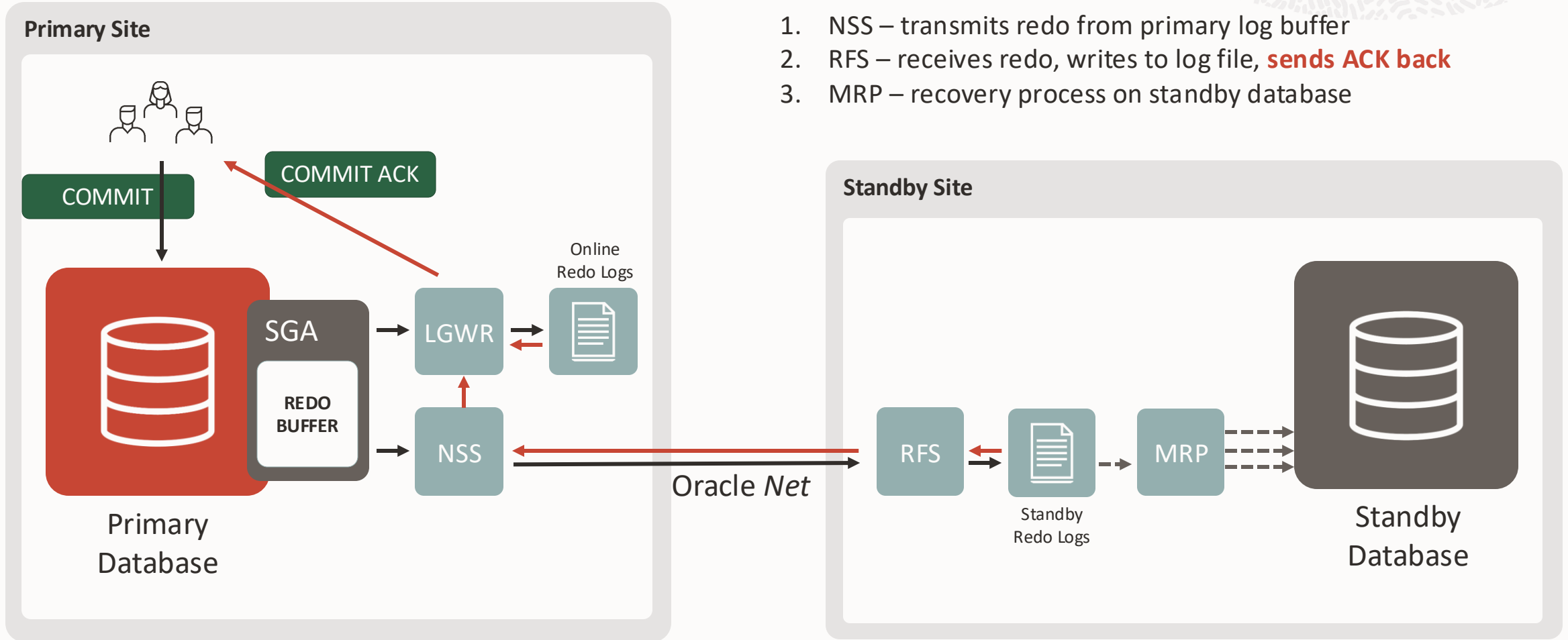
## Data Guard FASTSYNC Process Architecture





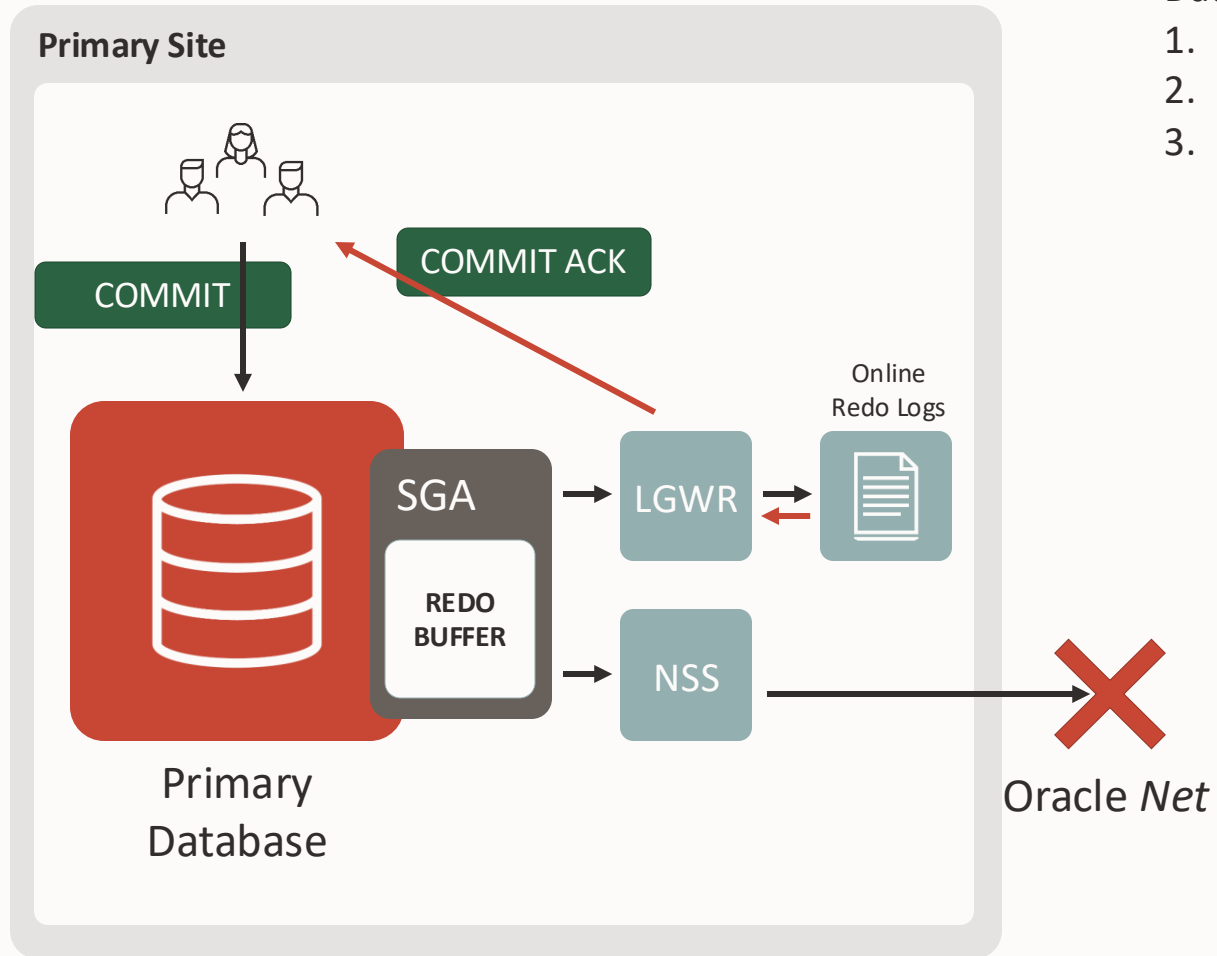
# Data Guard Transport for Zero Data Loss

## Data Guard SYNC Process Architecture



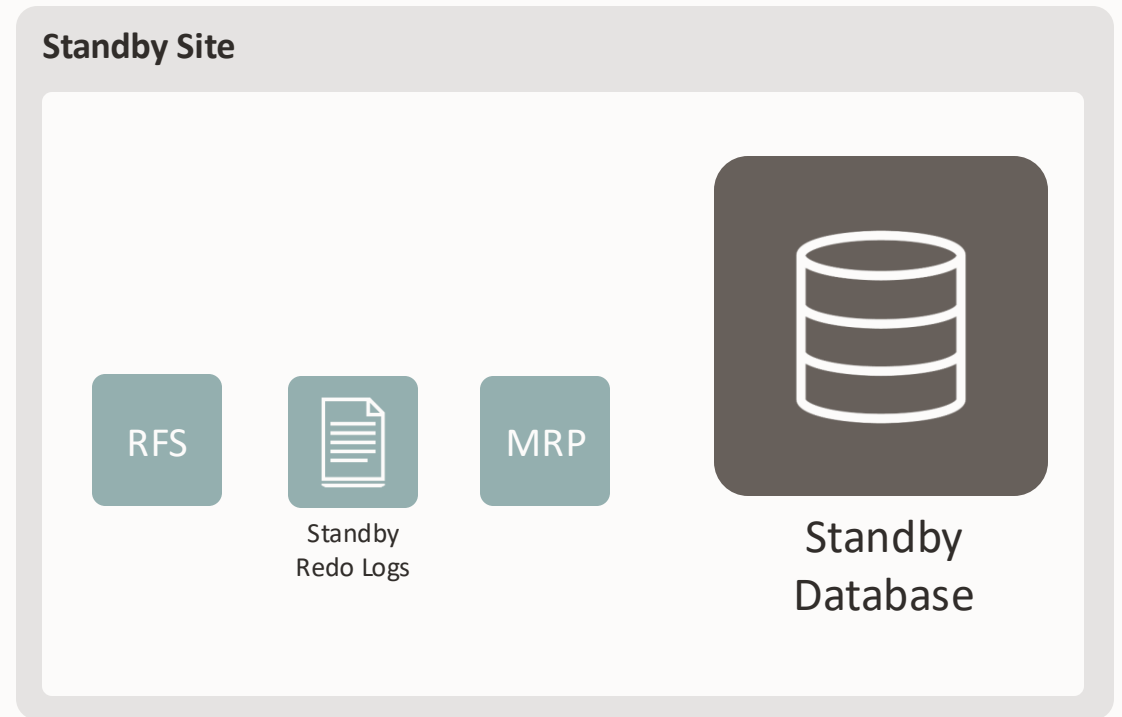
# Stalling Synchronous destinations

## Data Guard FASTSYNC/SYNC Process Architecture



### Data Guard Processes

1. NSS – tries to send the redo to the remote destination
2. The commits stall for NetTimeout seconds
3. The destination is **abandoned**, the commits resume



# The difference between receiving the redo late and not receiving it

## DATUM\_TIME vs TRANSPORT LAG vs LAST\_TIME

Standby **not receiving the redo** from the primary:

```
SQL> select value, datum_time, from v$dataguard_stats where name='transport lag';
```

VALUE	DATUM_TIME
-----	-----
+00 00:00:00	07/11/2024 08:28:46

Standby **receiving old redo** from the primary:

```
SQL> select value, datum_time, from v$dataguard_stats where name='transport lag';
```

VALUE	DATUM_TIME
-----	-----
+01 13:50:54	07/12/2024 21:48:10

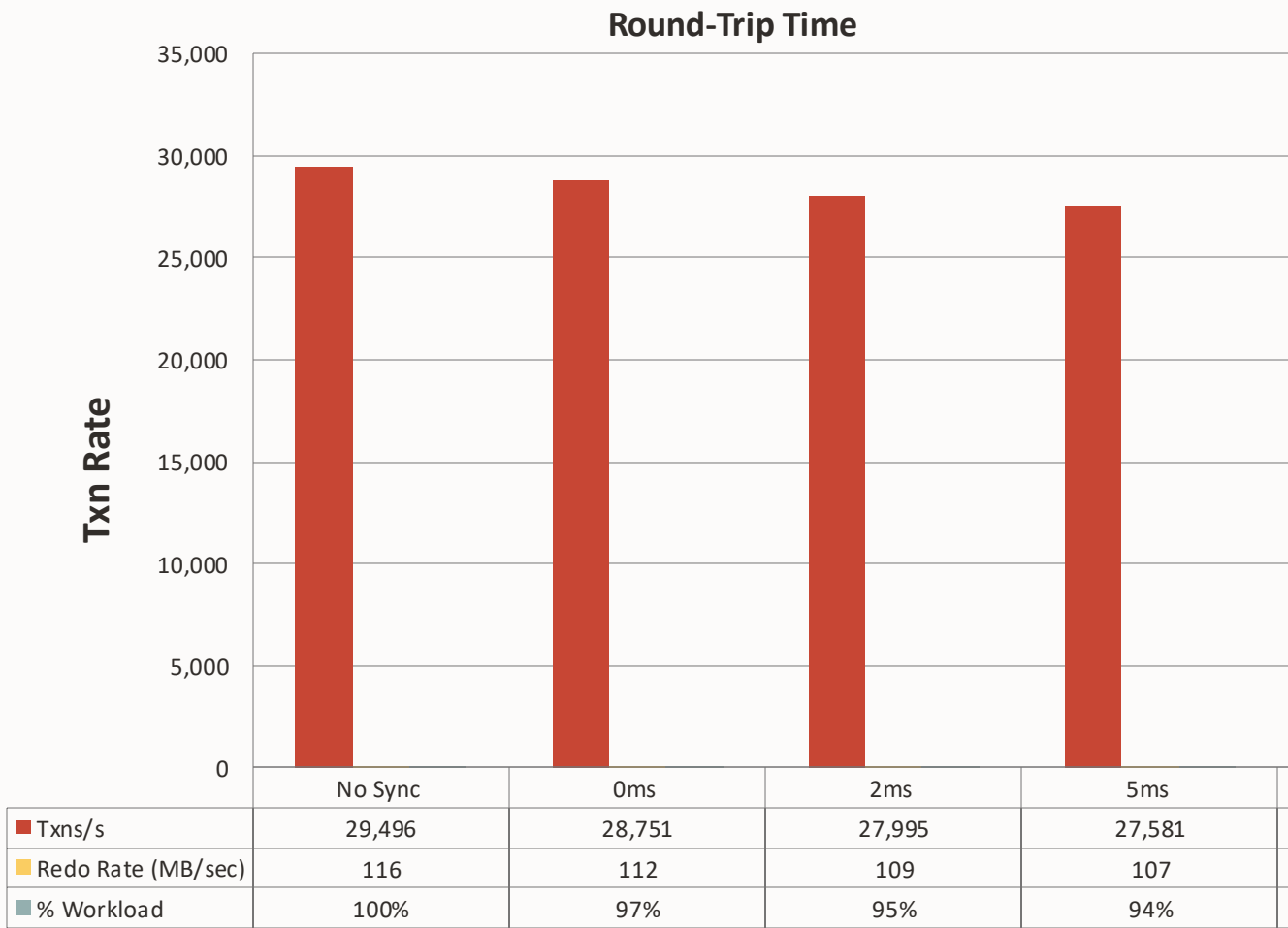
The last redo written in the standby logs:

```
SQL> select max(last_time) from v$standby_log where status='ACTIVE';
```

MAX(LAST_TIME)
-----
07/11/2024 08:28:46

# High Performance – Synchronous Redo Transport

Mixed OLTP workload with Metro-Area Network Latency



Note: 0ms latency on graph represents values <1ms

## Workload profile

- Simulated OLTP with large inserts
- **112 MB/s redo**

3% impact at < 1ms RTT

5% impact at 2ms RTT

6% impact at 5ms RTT

Use **oracptest** to assess your network bandwidth and latency



# Oracle Data Guard and Database Nologging



Enabling the FORCE LOGGING mode is a must for non-Engineered Systems.

On Exadata and Oracle Cloud Infrastructure, two modes are alternative to the FORCE LOGGING mode:

1. Standby Nologging **for Load Performance**
    - Ensures that standbys will receive the non-logged data changes with minimum impact on the speed of loading at the primary.
    - The standby can transiently have non-logged blocks. These non-logged blocks will be automatically resolved by managed standby recovery.
  2. Standby Nologging **for Data availability**
    - Ensures that all standbys have the data when the primary load commits, but at the cost of throttling the speed of loading data at the primary.
    - The standbys will never have any non-logged blocks.
- These new modes cause Multi-Instance Redo Apply to return an error
    - Single Instance Redo apply must be manually enabled to proceed past the nologging operation.

# Oracle Data Guard Best Practices – Transport and Apply Tuning

## Redo Apply Best Practices

<https://docs.oracle.com/en/database/oracle/oracle-database/23/haovw/tune-and-troubleshoot-oracle-data-guard.html#GUID-E8C27979-9D37-4899-9306-A5AE2B5CF6C0>

## Best Practices for Redo Transport Tuning

<https://docs.oracle.com/en/database/oracle/oracle-database/23/haovw/tune-and-troubleshoot-oracle-data-guard.html#GUID-A6963335-8C5A-4DD0-AD3F-22F4CBCE3DD0>

## Assessing Synchronous Redo Transport

<https://docs.oracle.com/en/database/oracle/oracle-database/23/haovw/tune-and-troubleshoot-oracle-data-guard.html#GUID-30CD6E1C-1CE2-4BB6-A404-896D5C06ECCE>

## How To Calculate The Required Network Bandwidth Transfer Of Redo In Data Guard (Doc ID 736755.1)

<https://support.oracle.com/rs?type=doc&id=736755.1>

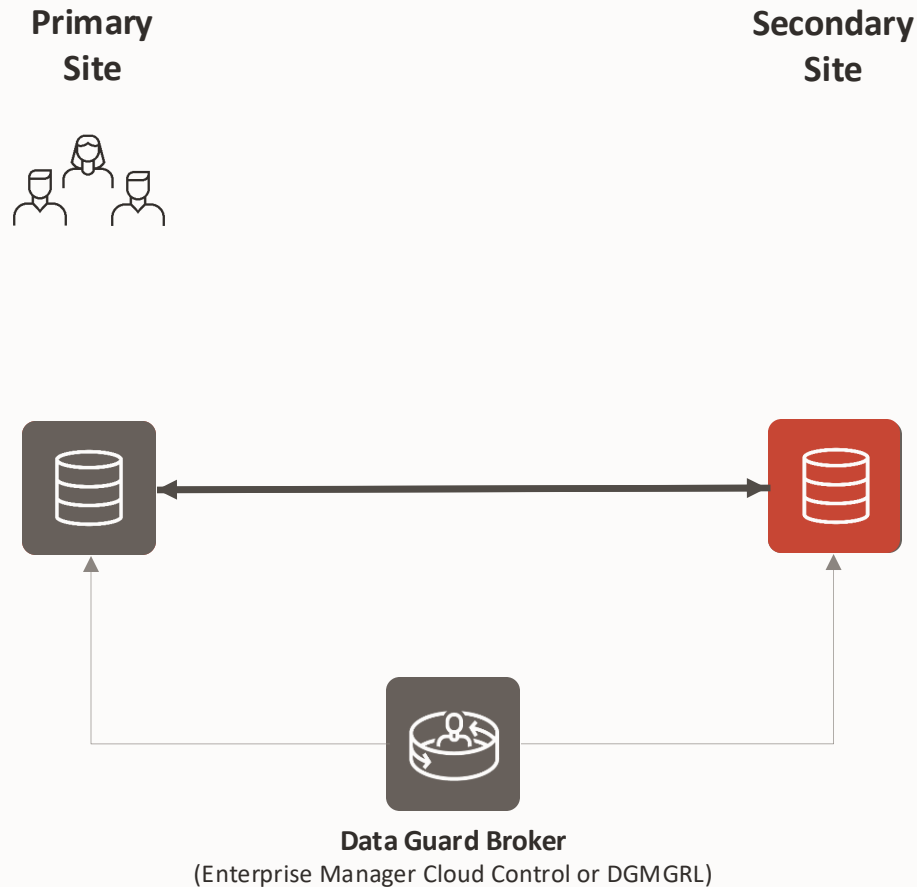
## Assessing and Tuning Network Performance for Data Guard and RMAN (Doc ID 2064368.1)

<https://support.oracle.com/rs?type=doc&id=2064368.1>

# Oracle Data Guard Role Transitions

# Oracle Data Guard Planned Role Transition

## Switchover: Planned role transition with Zero Data Loss

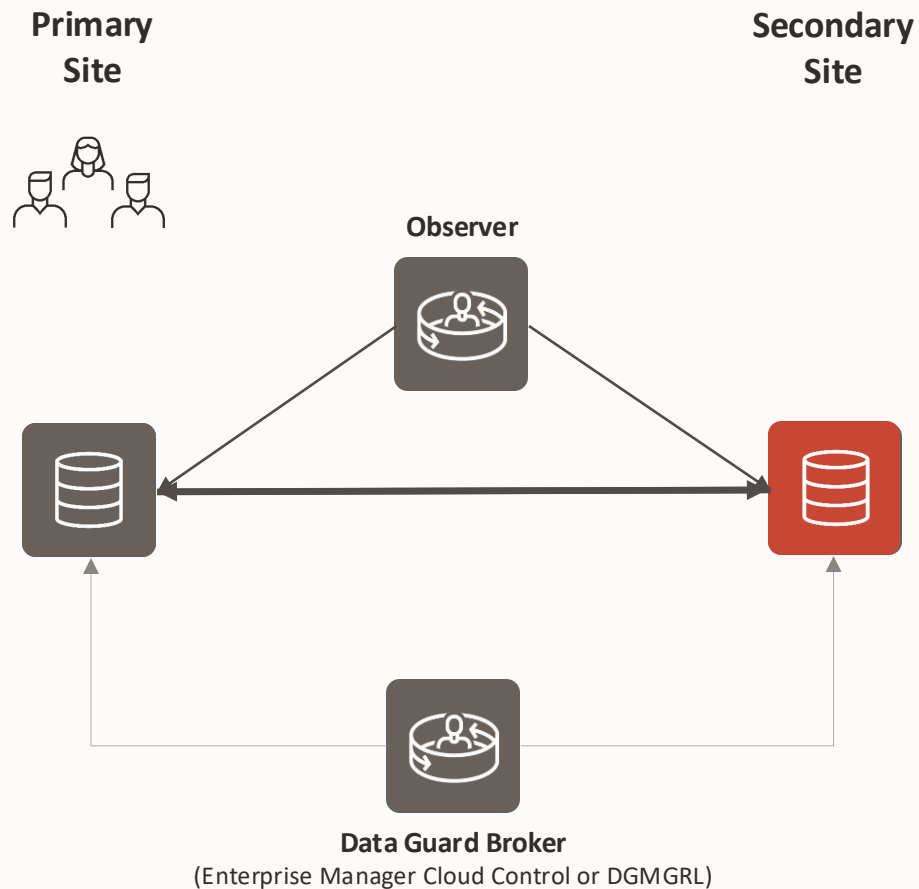


- **Switchover initiated**
- **The primary ends the transactions and stops the services**
- **All the transaction are synced to the standby**
- **The standby is converted to primary and the services are started**
  - The replication starts again
- **The applications reconnect transparently to the new primary**
  - If properly configured, the application experience just a freeze for 1-2 minutes or less



# Oracle Data Guard Unplanned Role Transition

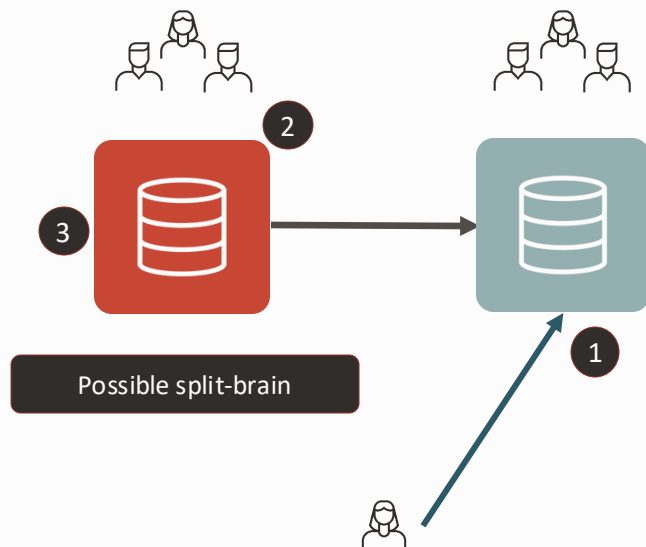
Failover: In case of failure the role transition *can* be without data loss



- **The observer detects the failure of the primary**
  - Depending on the protection mode and situation, the observer initiates the failover after `FastStartFailoverThreshold` seconds
- **The standby is converted to primary and the services are started**
  - Depending on the protection mode and situation, there might be some data loss (the tolerated amount is configurable)
- **The applications reconnect to the new primary**
  - The reinstatement of the primary requires a single broker command
- The **failover** can be initiated also **manually** (DGMGRL) or by the application (DBMS\_DG.INITIALIZE\_FS\_FAILOVER) . The amount of data loss is customer's responsibility in this case.

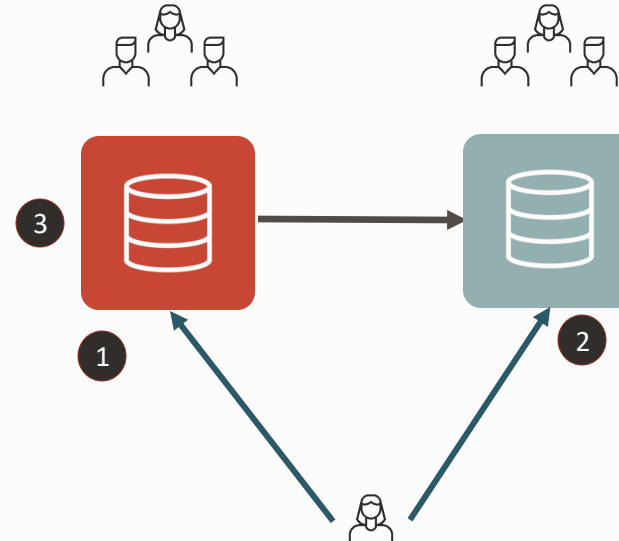
# "FAILOVER TO" vs "DBMS\_DG.INITIALIZE\_FS\_FAILOVER"

## FAILOVER TO stdby



- 1 The standby database becomes the new primary
- 2 The former primary shuts down if FSFO is in place
- 3 The former primary is automatically reinstated

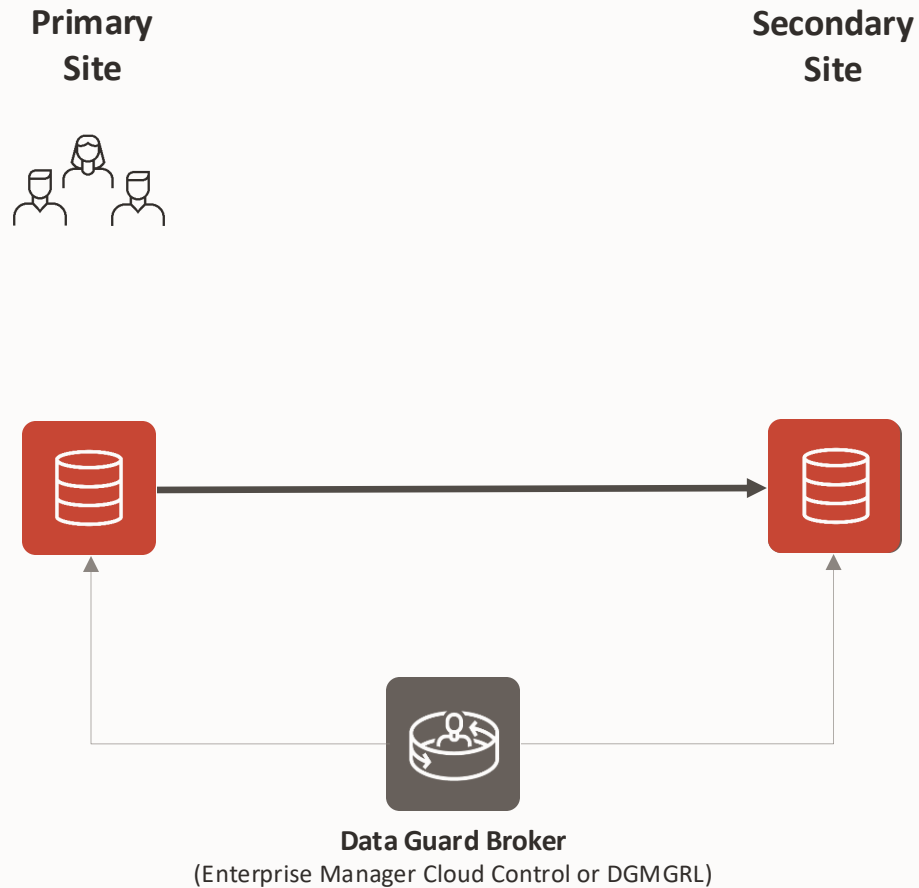
## DBMS\_DG.INITIALIZE\_FS\_FAILOVER



- 1 The procedure shuts down the primary first
- 2 The standby database becomes the new primary
- 3 The former primary is not automatically reinstated

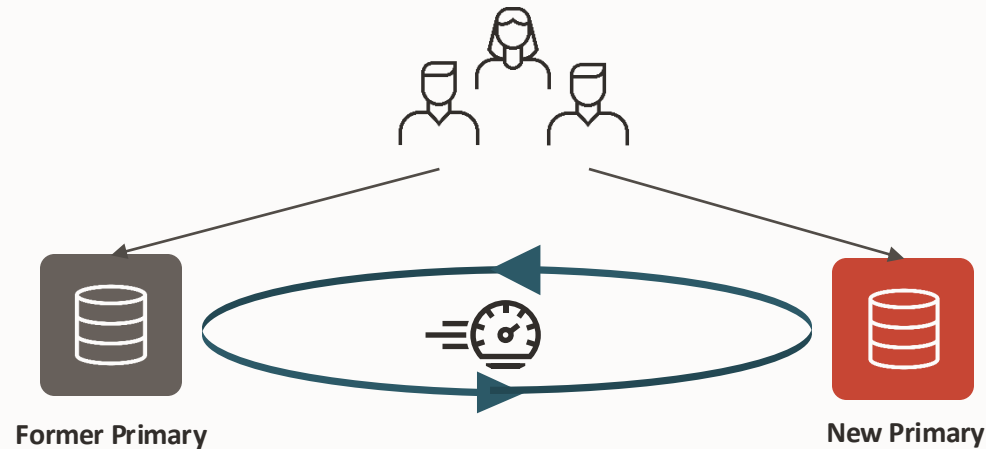
# Data Guard Snapshot Standby

Standby database temporarily in Read Write



- **The standby is converted to Snapshot Standby**
  - Standby open read write
- **Users and DBAs perform tests (Upgrade, Performance, etc)**
  - The primary is still protected by the redo transfer
- **When the tests are over, the standby is flashed back and converted to physical standby again**
- Note: the snapshot standby cannot relay the redo to a cascaded standby

# Tune the Switchover and Failover Operations



## ADMINISTRATION

- Stop any long running operations before switchover
- Reduce `FAST_START_MTTR_TARGET`
- Switchover to a MOUNTED database whenever possible

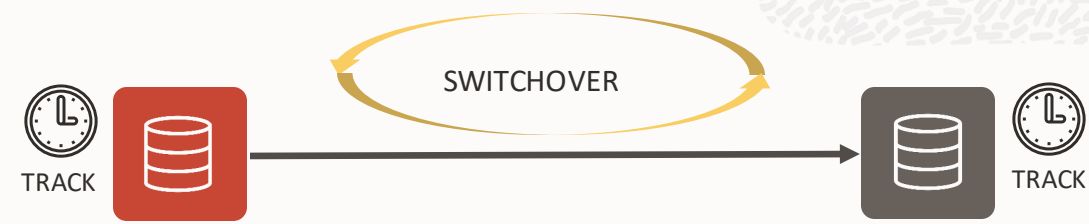
## DEVELOPMENT

- Use Oracle Connection Pools
- Make your transactions as light as possible
- Use FAN notifications
- Implement session draining
- Use the recommended connection strings

## IMPLEMENTATION

- Reduce the number of data files
- Reduce the workload on the database
- Don't over-consolidate PDBs

# Easier tracking of role transitions



The new fixed view **V\$DG\_BROKER\_ROLE\_CHANGE** tracks the last 10 role transitions

```
SQL> select * from V$DG_BROKER_ROLE_CHANGE;
```

EVENT	STANDBY_TYPE	OLD_PRIMARY	NEW_PRIMARY	FS_FAILOVER_REASON	BEGIN_TIME	END_TIME
-----						
Failover	Physical	mydb1	mydb1b	Manual Failover	30-SEP-2024 19:01:14	30-SEP-2024 19:01:35
Switchover	Physical	mydb1b	mydb1		30-SEP-2024 19:04:53	30-SEP-2024 19:05:15
Switchover	Physical	mydb1	mydb1b		30-SEP-2024 20:51:38	30-SEP-2024 20:52:03
Failover	Physical	mydb1b	mydb1	Manual Failover	30-SEP-2024 20:52:46	30-SEP-2024 20:53:04
Switchover	Physical	mydb1	mydb1c		30-SEP-2024 19:53:14	30-SEP-2024 19:54:14
Switchover	Physical	mydb1c	mydb1		30-SEP-2024 20:03:14	30-SEP-2024 20:04:04
Switchover	Logical	mydb1	mydb1d	Primary Disconnected	30-SEP-2024 20:24:46	30-SEP-2024 20:26:32
Switchover	Logical	mydb1d	mydb1		30-SEP-2024 20:35:27	30-SEP-2024 20:35:48
Fast-Start Failover	Physical	mydb1	mydb1b		30-SEP-2024 20:13:51	30-SEP-2024 20:14:53



# Strict validation of switchover readiness

## New command VALIDATE DATABASE STRICT

(\*) available in 23.6

```
VALIDATE DATABASE [VERBOSE] <database>  
[ STRICT { ALL | APPLY_PROPERTY | DATAFILES_OFFLINE (*) | FLASHBACK | FORCE_LOGGING | LOG_FILES_CLEARED  
| LOG_FILE_CONFIGURATION | PDBS_OFFLINE (*) | PDB_SAVE_STATE (*) | TRANSPORT_PROPERTY }];
```

```
DGMGRL> validate database chicago strict all;  
DGM-17567: Current database session was authenticated using operating system credentials.
```

```
Database Role:      Physical standby database  
Primary Database:   boston
```

```
Ready for Switchover: No  
  The primary or standby database does not have flashback database enabled. (*)
```

```
Ready for Failover:   Yes (Primary Running)
```

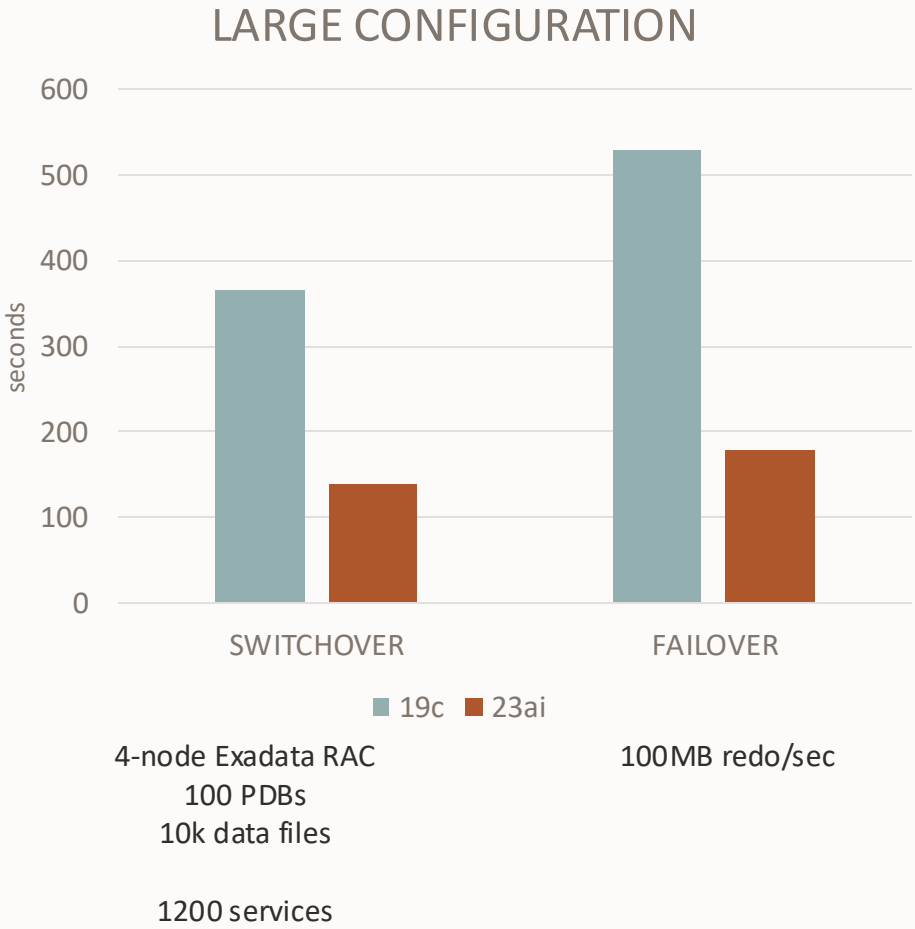
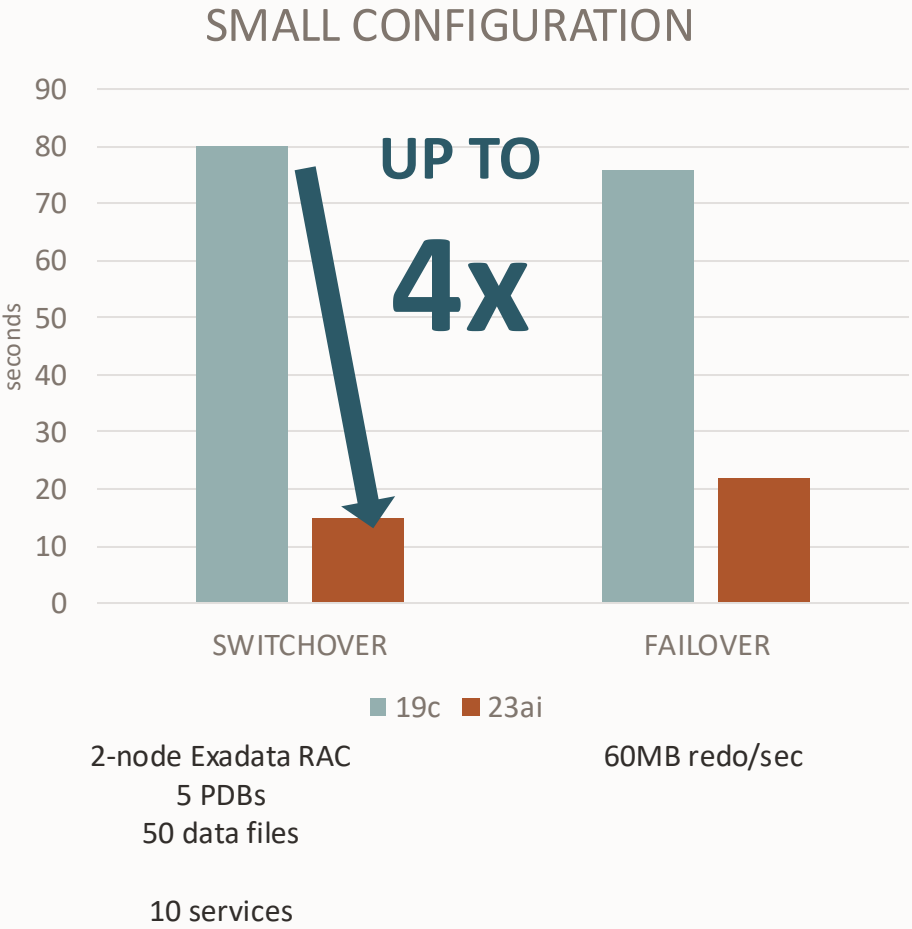
### Flashback Database Status:

Database	Status	Retention Target
boston	<b>Off</b>	1440
chicago	<b>Off</b>	1440

...

# Faster Role Transitions in Oracle Data Guard 23ai

Between 50% and 85% faster role transition



# Faster Observer acknowledgment in Maximum Performance

Pre-emptive actions to prevent FSFO stalls.

## Zero-stall observer acknowledgment



19c

23ai

in Fast-Start Failover Max Performance





# Oracle Data Guard Role Transitions – Read More



## Role Transition Assessment and Tuning

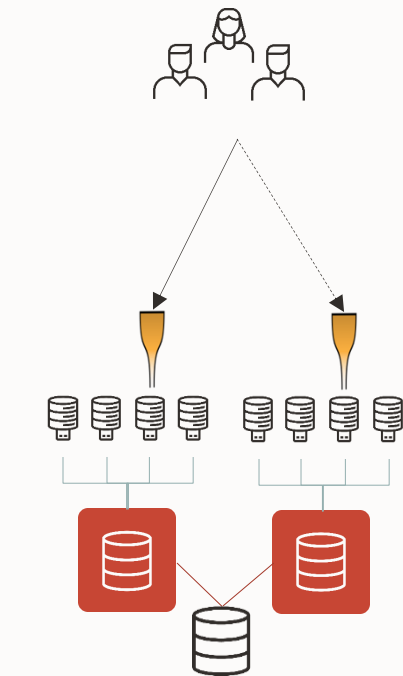
<https://docs.oracle.com/en/database/oracle/oracle-database/23/haovw/tune-and-troubleshoot-oracle-data-guard.html#GUID-CBA9FC61-9894-4D62-9569-EFBD7960267F>

# Client Failover and Application Continuity

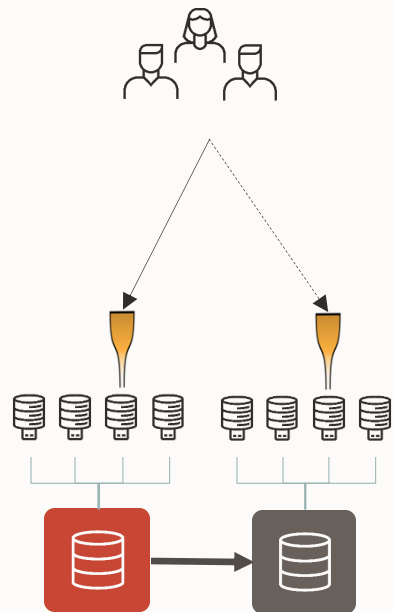
# Services for Location Transparency and High Availability

Services provide a “dial in number” for your application

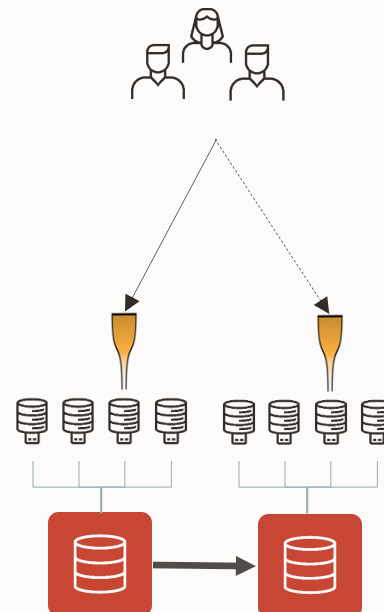
- Use Custom services with FAN notifications and Application Continuity
- Regardless of the location, the connection string does not change!
- Client failover best practices across the Oracle technology stack



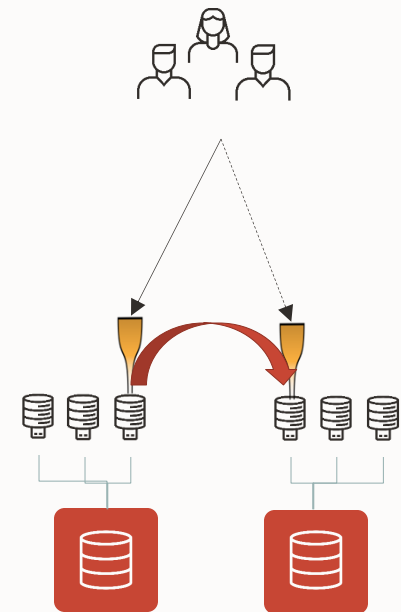
Real Application Clusters



Active Data Guard



GoldenGate



PDB Relocation

# Connections Appear Continuous

Standard for all drivers from 12.2

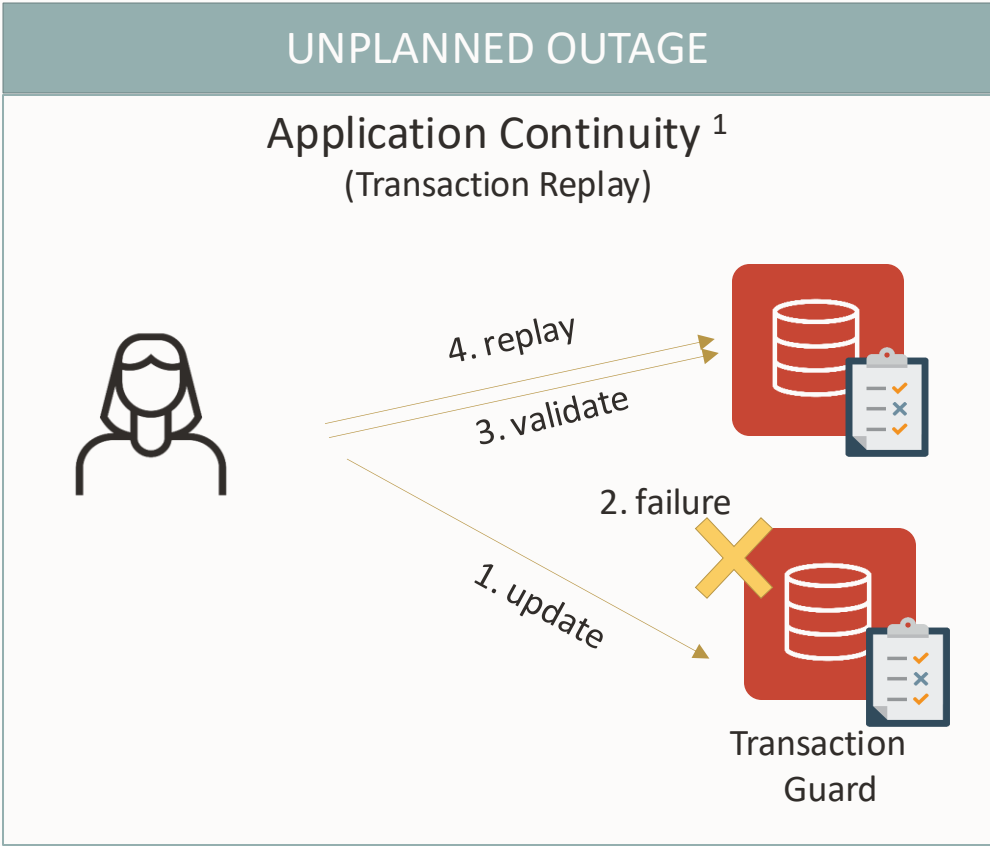
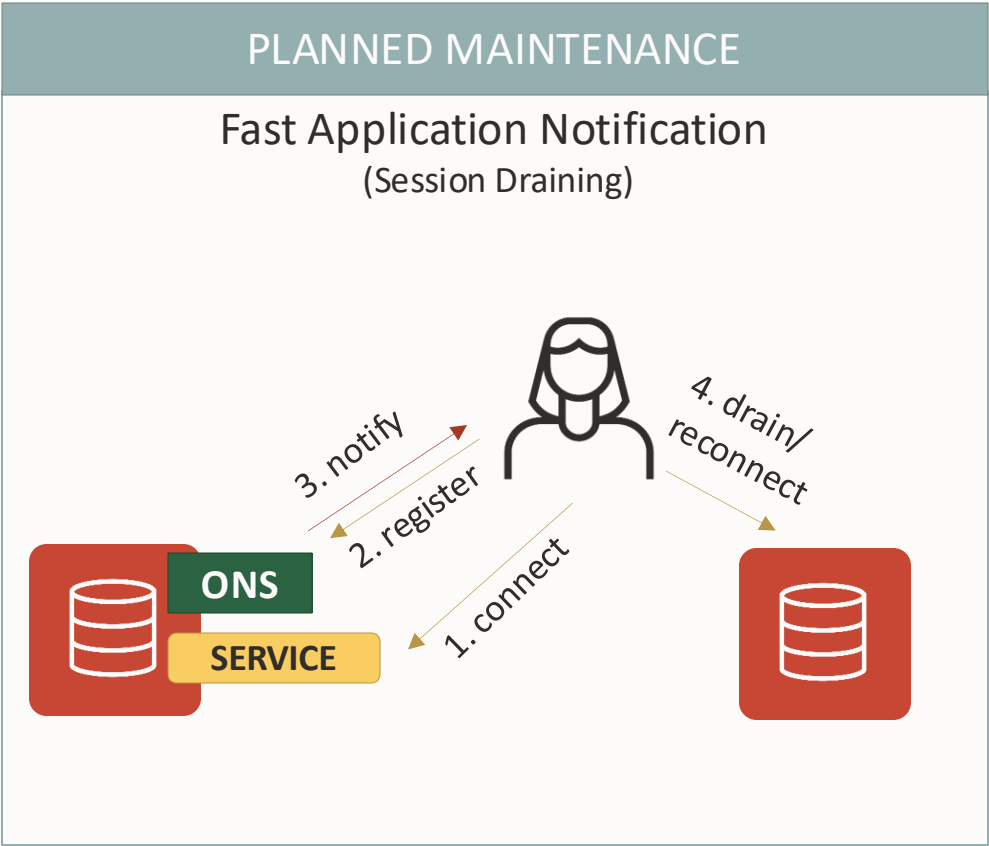
Automatic retries until the service is available

```
HR = (DESCRIPTION =  
  (CONNECT_TIMEOUT=120)(RETRY_COUNT=50)(RETRY_DELAY=3)  
  (TRANSPORT_CONNECT_TIMEOUT=3)  
  (ADDRESS_LIST =  
    (LOAD_BALANCE=on)  
    (ADDRESS=(PROTOCOL=TCP)(HOST=cluster1-scan)(PORT=1521)))  
  (ADDRESS_LIST =  
    (LOAD_BALANCE=on)  
    (ADDRESS=(PROTOCOL=TCP)(HOST=cluster2-scan)(PORT=1521)))  
  (CONNECT_DATA=(SERVICE_NAME = HR.oracle.com)))
```

Always use a custom service!  
Do NOT use PDB or DB Name

# Client-side required technologies

Client draining/failover is a crucial part of high availability for applications connecting to the database.



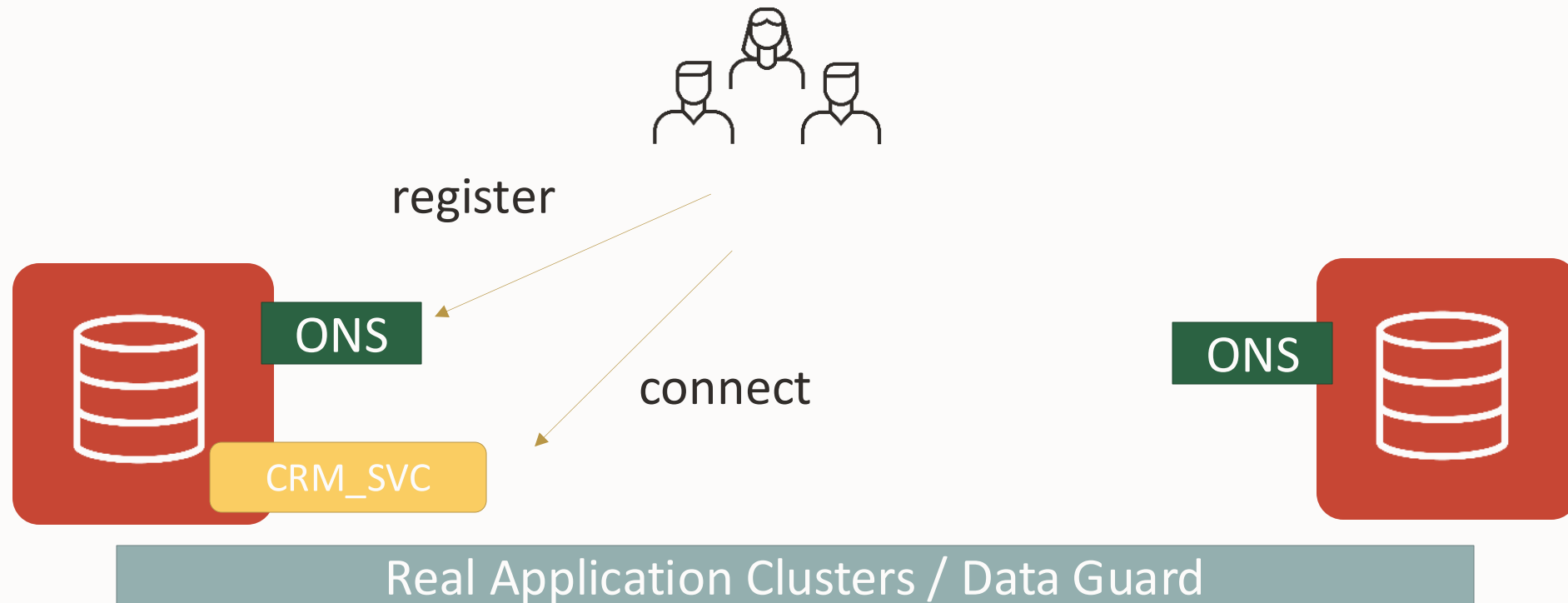
<sup>1</sup> Configuring Continuous Availability for Applications

<https://docs.oracle.com/en/database/oracle/oracle-database/23/haovw/configuring-continuous-availability-applicationsconfiguring-continuous-availability-applicationsco.html>



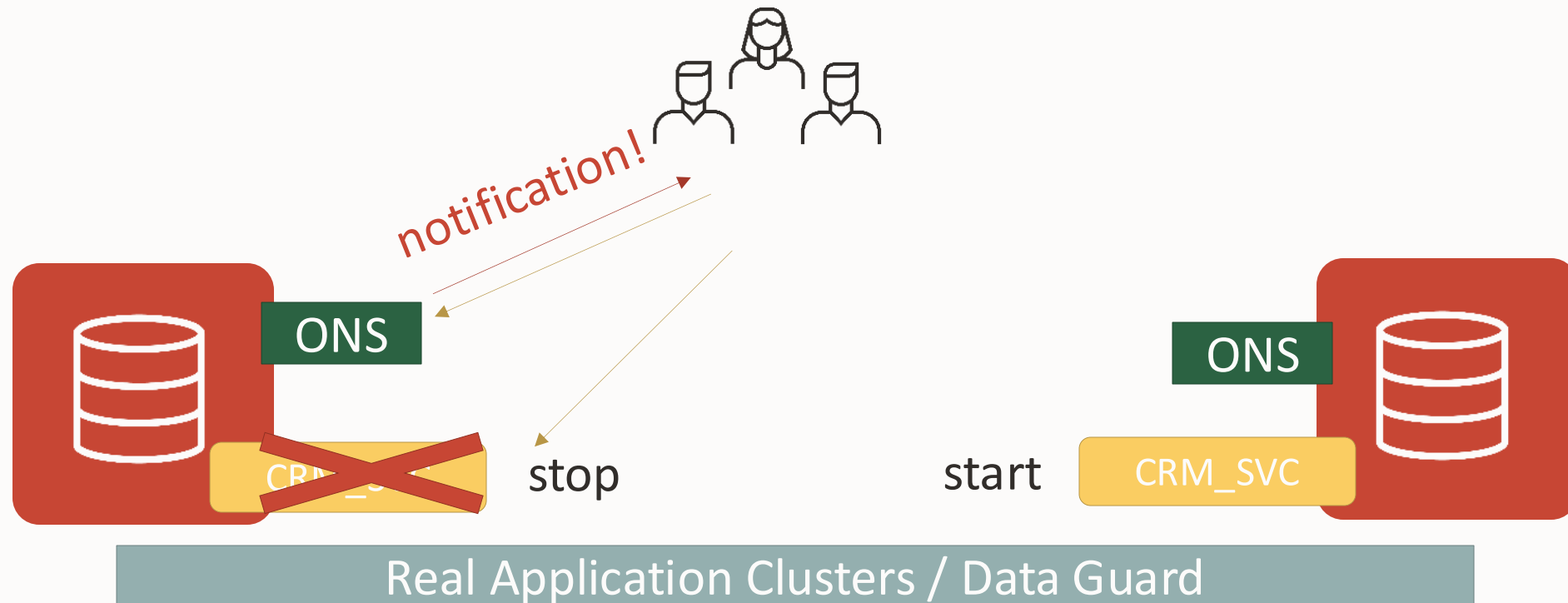
# Fast Application Notification

Session Draining for planned maintenance



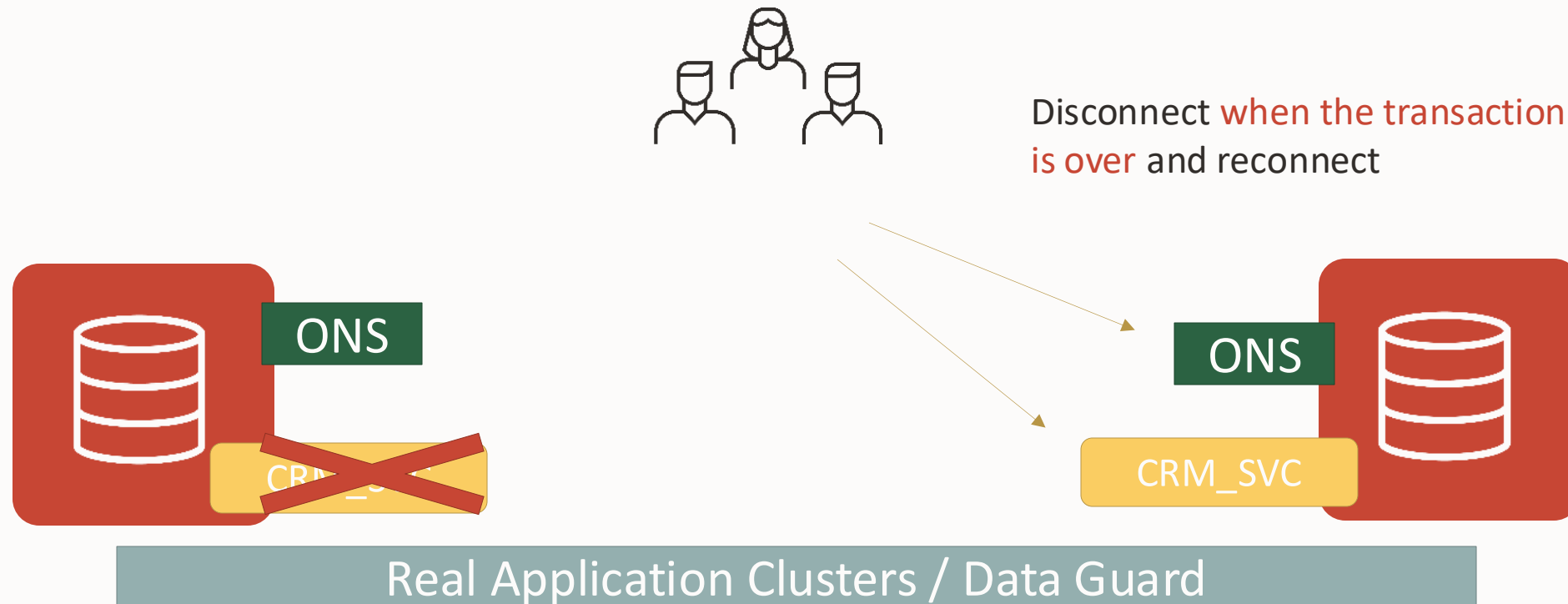
# Fast Application Notification

Session Draining for planned maintenance



# Fast Application Notification

Session Draining for planned maintenance





## Fast Connection Failover (FCF)

FAN integrated in connection pools



- Pre-configured FAN integration
- Uses connection pools
- The application must be pool aware
  - (borrow/release)
- The connection pool leverages FAN events to:
  - Remove quickly dead connections on a DOWN event
  - (opt.) Rebalance the load on a UP event

## Fast Connection Failover (FCF)

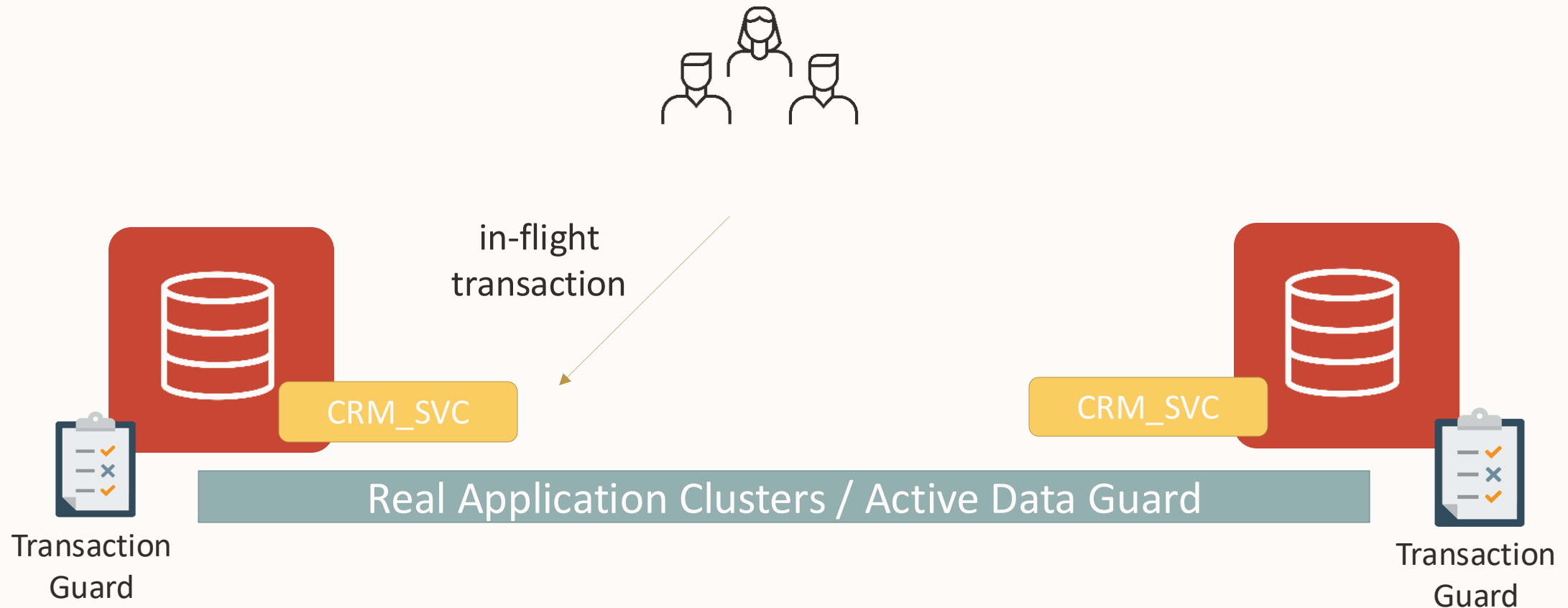
FAN integrated in connection pools



- **UCP** (Universal Connection Pool, ucp.jar) and WebLogic **Active GridLink** handle FAN out of the box.  
No code changes! Just enable **FastConnectionFailoverEnabled**
- Third-party connection pools can implement FCF
  - If JDBC driver version  $\geq 12.2$
  - simplefan.jar and ons.jar in CLASSPATH
  - Connection validation options are set in pool properties
  - Connection pool can plug **javax.sql.ConnectionPoolDataSource**
  - Connection pool checks connections at borrow/release

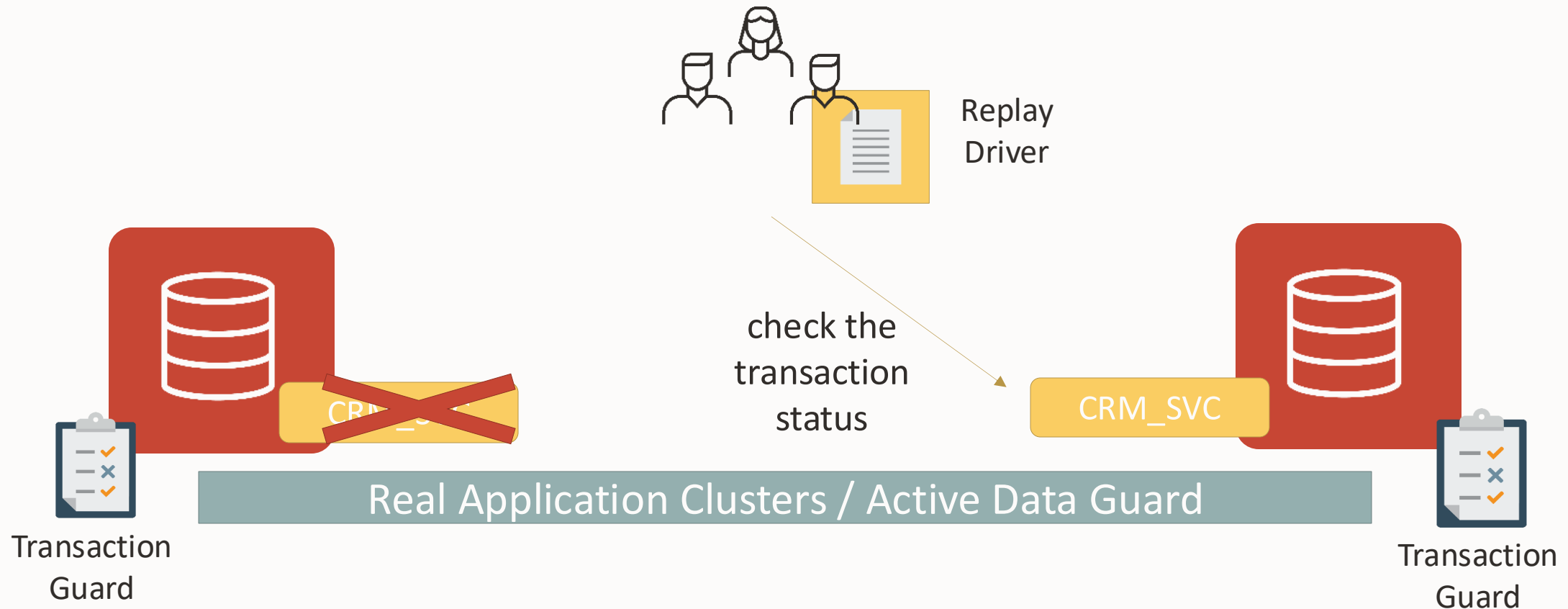
# Application Continuity (AC)

Protects the in-flight transaction from failures and disconnections



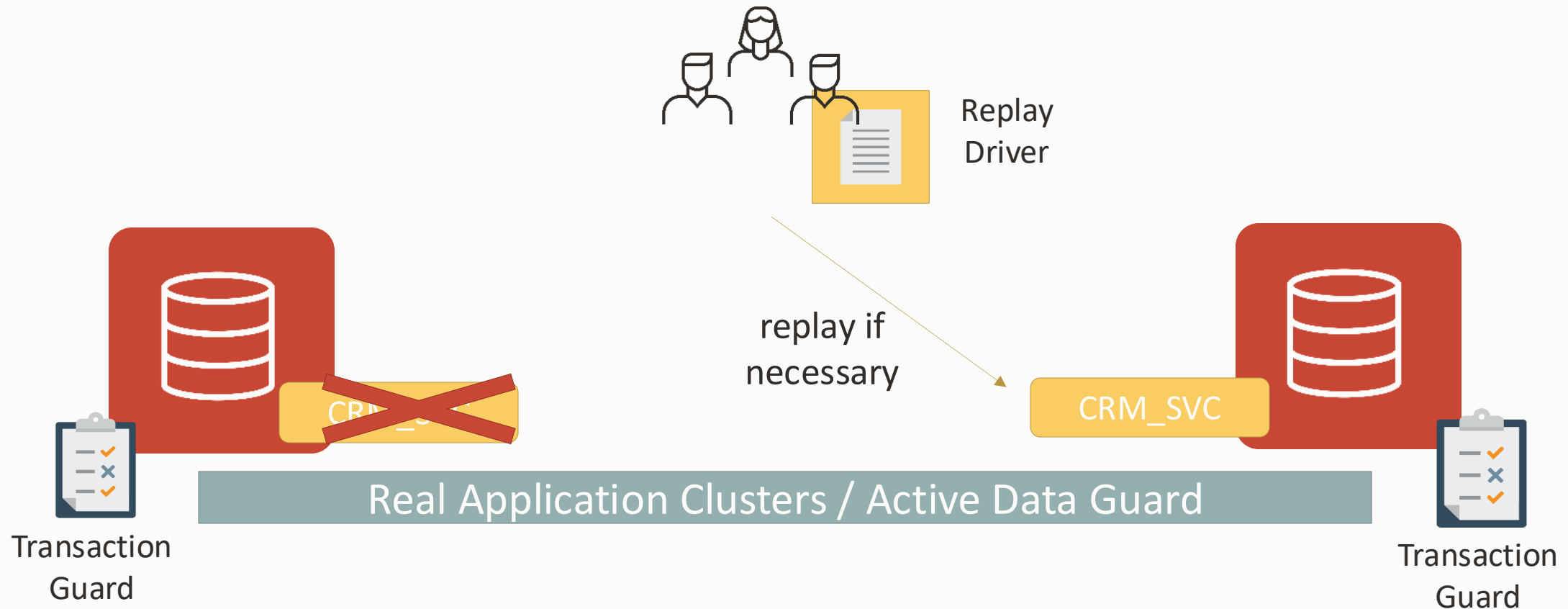
# Application Continuity (AC)

Protects the in-flight transaction from failures and disconnections



# Application Continuity (AC)

Protects the in-flight transaction from failures and disconnections



# Application Continuity (AC)

Protects the in-flight transaction from failures and disconnections

- AC with UCP: no code change

```
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
pds.setConnectionFactoryClassName("oracle.jdbc.replay.OracleDataSourceImpl");
...
conn = pds.getConnection();           // Implicit database request begin
// calls protected by Application Continuity
conn.close();                         // Implicit database request end
```

- AC without connection pool: code change

```
OracleDataSourceImpl ods = new OracleDataSourceImpl();
conn = ods.getConnection();
...
((ReplayableConnection) conn).beginRequest(); // Explicit database request begin
// calls protected by Application Continuity
((ReplayableConnection) conn).endRequest();   // Explicit database request end
```

# Transparent Application Continuity (TAC)

Application Continuity for every connection and application type

- Introduced in **18c** for JDBC thin, **19c** for OCI (Oracle Call Interface)
- Records session and transaction state server-side
- No application change
- Works without connection pools (although they are still recommended)
- Replayable transactions are replayed
- Non-replayable transactions raise exception
- Good driver coverage but check the doc!
- Side effects are never replayed

# Key Differences between FAN, AC, and TAC



	Best for	Since Version	Application Changes	Requires Connection Pool	Replay Side Effects	JDBC/OCI
FAN	Planned Maintenance	10g	Catch FAN events (or use UCP)	No, but recommended (FCF)	N/A	Both
AC	Unplanned Outage	12c	Use explicit boundaries (or use UCP)	Yes	Yes (Choose)	Both
TAC (Recommended)	Unplanned Outage	19c	No	No, but recommended	Never	Both





Help Center JDBC Developer's Guide

Database / Oracle / Oracle Database / Release 23

# JDBC Developer's Guide

- List of Tables
- Title and Copyright Information
- Preface
- Changes in This Release for Oracle Database JDBC Developer's Guide
- Part I Overview
- Part II Oracle JDBC
- Part III Connection and Security
- Part IV Data Access and Manipulation
- Part V Performance and Scalability
- Part VI High Availability**
  - 32 Transaction Guard for Java
  - 33 Application Continuity for Java
  - 34 Oracle JDBC Support for FAN Events
  - 35 Transparent Application Failover
  - 36 Single Client Access Name
- Part VII Transaction Management
- Part VIII Manageability
- Appendixes
- Index

## Documentation!

## Oracle Call Interface (OCI)

## JDBC

Help Center Oracle Call Interface Developer's Guide

Database / Oracle / Oracle Database / Release 23

# Developer's Guide

- List of Tables
- Title and Copyright Information
- Preface
- 1 OCI: Introduction
- 2 Building and Configuring OCI Applications
- 3 OCI Programming Basics
- 4 Data Types
- 5 Using SQL Statements in OCI
- 6 Binding and Defining in OCI
- 7 Describing Schema Metadata
- 8 LOB and BFILE Operations
- 9 Managing Scalable Platforms
- 10 Session Pooling and Connection Pooling in OCI
- 11 High Availability in OCI**
- 12 Notification Methods and Database Advanced Queuing
- 13 User-Defined Callback Functions in OCI
- 14 Performance Topics
- 15 Database Startup and Shutdown
- 16 Support for Pluggable Databases
- 17 OCI Interface for Using Shards

## 11 High Availability in OCI

This chapter describes high availability (HA) features in OCI.

This chapter includes the following topics:

- [Runtime Connection Load Balancing](#)
- [HA Event Notification](#)
- [Transparent Application Failover in OCI](#)
- [OCI and Transaction Guard](#)
- [OCI and Application Continuity](#)



# Fast-Start Failover: an overview

# Network partitioning

When did the primary disconnect?



```
-- THE LAST TIME THE STANDBY HEARD FROM THE PRIMARY (1 second tolerance)
SQL> select datum_time, (sysdate-to_date(datum_time,'MM/DD/YYYY HH24:MI:SS'))*86400 secs_ago
2>   from v$dataguard_stats where name='transport lag';
```

DATUM_TIME	SECS_AGO
07/11/2024 08:28:46	90361



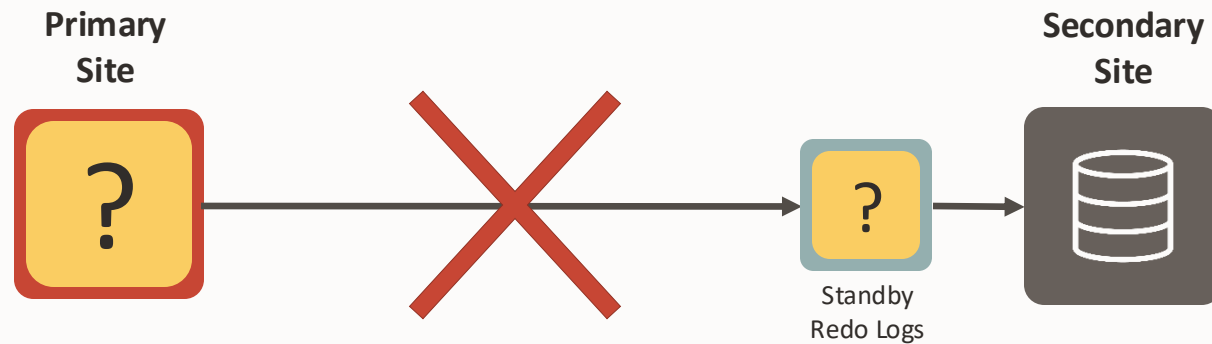
The columns might be null in some cases



Do not rely on the transport lag value during network partitioning

# Network partitioning

Up to which point can the standby recover?



```
-- THE TIMESTAMP OF THE LAST REDO ENTRY RECEIVED FROM THE PRIMARY
SQL> alter session set nls_date_format='MM/DD/YYYY HH24:MI:SS';
SQL> select coalesce(max(s.last_time), max(a.next_time)) as last_redo_from_prim,
2>    (sysdate-coalesce(max(s.last_time), max(a.next_time)))*86400 as secs_ago
3>   from v$standby_log s , v$archived_log a ;
```

LAST_REDO_FROM_PRIM	SECS_AGO
07/11/2024 08:28:46	91061



Is it a reliable way to calculate data loss?

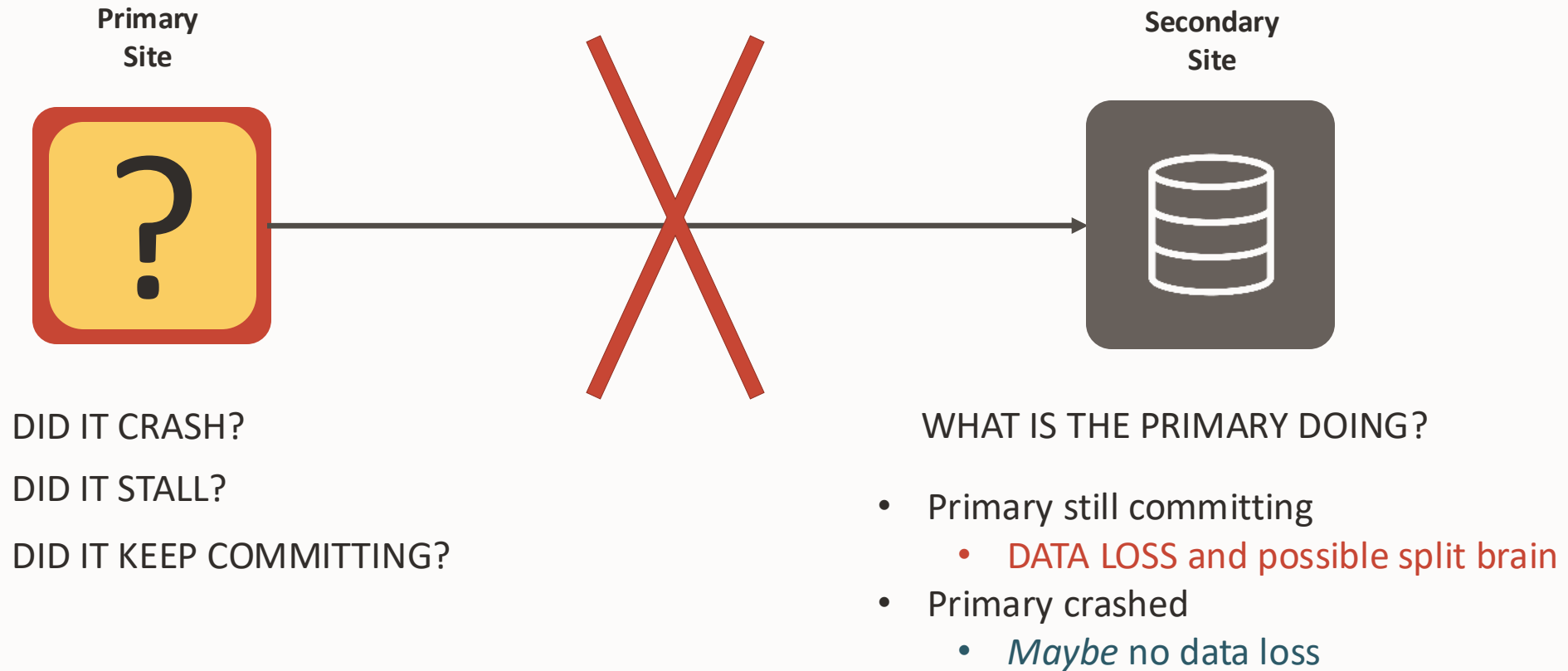


# Network partitioning

Is there a way to calculate the data loss upon failover?

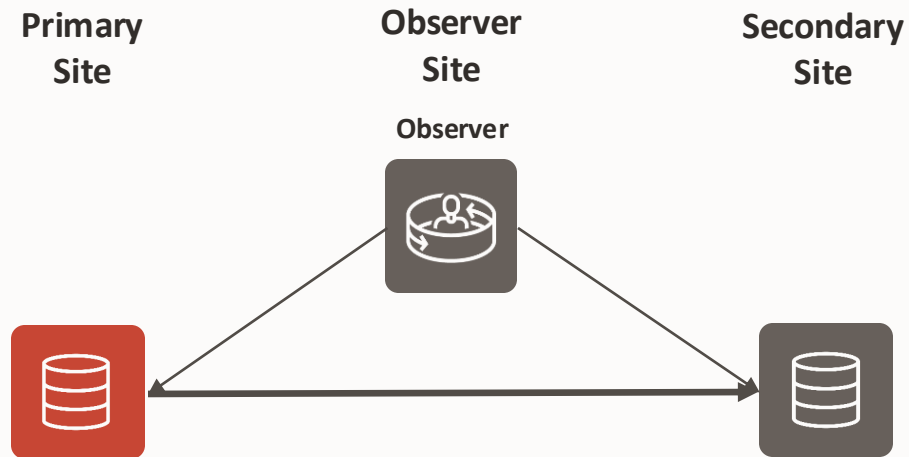


Can I safely fail over to the standby site?



# Oracle Fast-Start Failover introduces a quorum

Automatic failover when the Primary Database is unavailable



- The observer monitors both primary and standby
- **Primary has the quorum** (standby is isolated):
  - The primary keeps writing
- **Primary and standby have the quorum** (observer is isolated):
  - The configuration keeps working unobserved
- **Standby has the quorum** (primary is isolated):
  - Failover!  
The primary loses the quorum and stops committing
- The observer can work in “**OBSERVE ONLY**” mode
  - Reports a failure without failing over

# Fast-Start Failover automates the failover and solves important problems



## Recovery Point Objective is honored

No automatic failover if the data loss breaches your RPO.

RPO can be set to ZERO (no data loss).

## Recovery Time Objective improves

Automatic failover begins immediately after the primary is not reachable for more than a specified threshold.

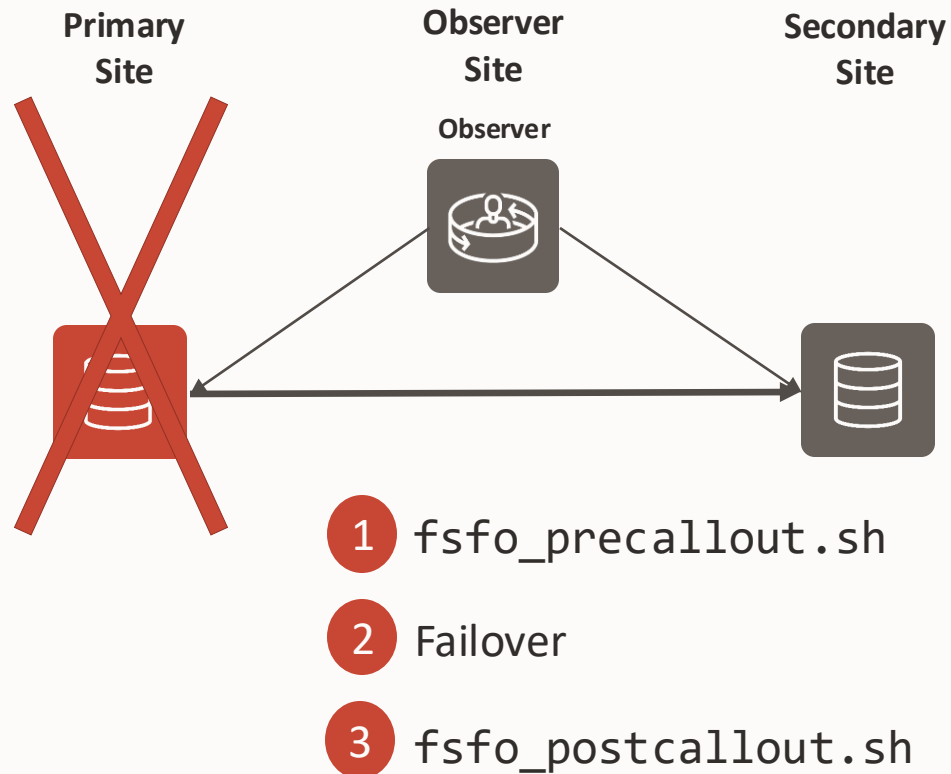
## Split-brain protection

The quorum mechanism intrinsically prevents having two primary databases after an automatic failover.

# Fast-Start Failover callouts

NEW IN  
21c

Execute custom actions before and after the automatic failover occurs



```
$ cat $DG_ADMIN/config_ConfigName/callout/fsfocallout.ora

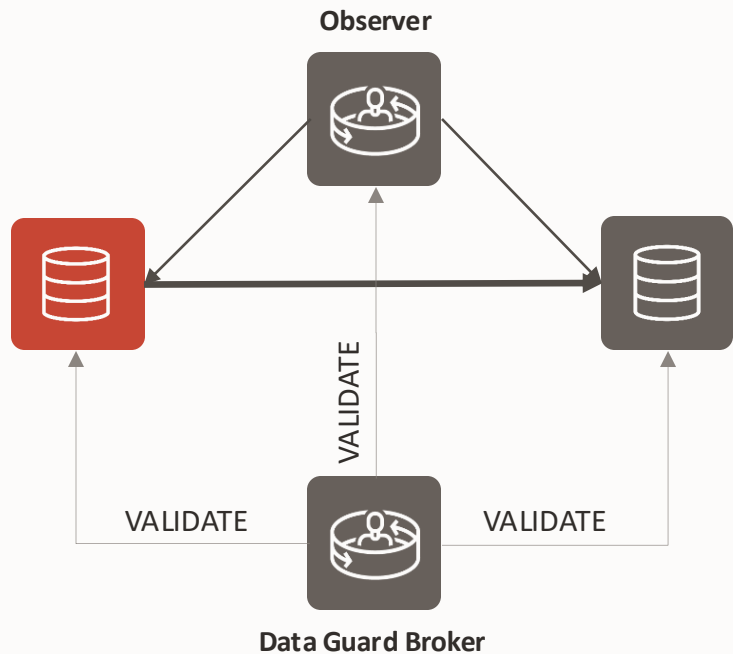
# The pre-callout script is run before failover
FastStartFailoverPreCallout=fsfo_precallout.sh
FastStartFailoverPreCalloutTimeout=1200
FastStartFailoverPreCalloutSucFileName=fsfo_precallout.suc
FastStartFailoverPreCalloutErrorFileName=precallout.err
FastStartFailoverActionOnPreCalloutFailure=STOP

# The post-callout script is run after failover succeeds
FastStartFailoverPostCallout=fsfo_postcallout.sh
$
```



# Fast Start Failover Configuration Validation

Ensure everything is configured properly for the automatic failover



```
DGMGRL> VALIDATE FAST_START FAILOVER;
```

```
Fast-Start Failover: Enabled in Potential Data Loss Mode
Protection Mode:      MaxPerformance
Primary:              North_Sales
Active Target:        South_Sales
```

Fast-Start **Failover Not Possible:**

```
Fast-Start Failover observer not started
```

Post Fast-Start **Failover Issues:**

```
Flashback database disabled for database 'dgv1'
```

**Other issues:**

```
FastStartFailoverThreshold may be too low for RAC databases.
```

Fast-start failover **callout configuration** file "fsfocallout.ora" has the following **issues:**

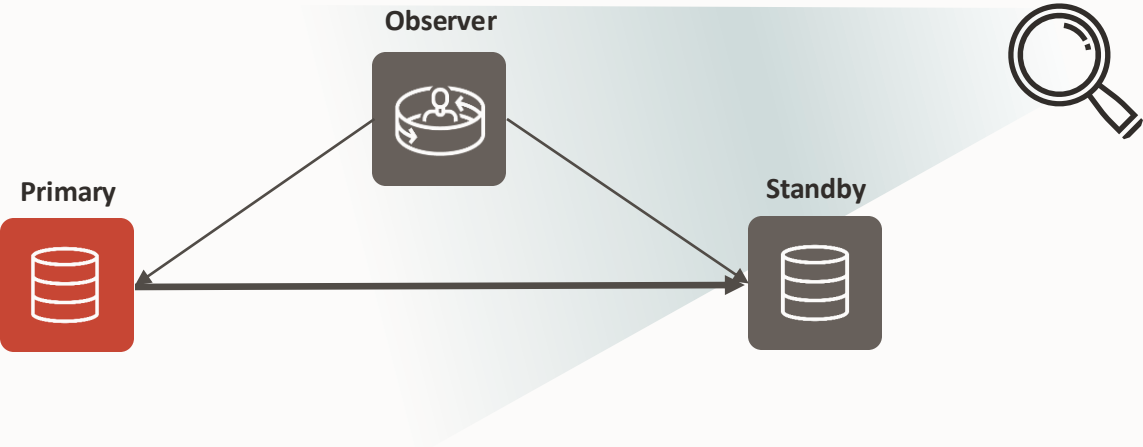
```
Invalid lines
```

```
foo=foo
```

```
The specified file "./precallout" contains a path.
```

# Easier checking of Fast-Start Failover configurations

The new fixed view **V\$FAST\_START\_FAILOVER\_CONFIG** shows the Fast-Start Failover settings and status



```
SQL> desc V$FAST_START_FAILOVER_CONFIG;
```

Name	Null?	Type
-----	-----	-----
FSFO_MODE		VARCHAR2(19)
STATUS		VARCHAR2(22)
CURRENT_TARGET		VARCHAR2(30)
THRESHOLD		NUMBER
OBSERVER_PRESENT		VARCHAR2(7)
OBSERVER_HOST		VARCHAR2(512)
PING_INTERVAL		NUMBER
PING_RETRY		NUMBER
PROTECTION_MODE		VARCHAR2(30)
LAG_LIMIT		NUMBER
AUTO_REINSTATE		VARCHAR2(5)
OBSERVER_RECONNECT		NUMBER
OBSERVER_OVERRIDE		VARCHAR2(5)
SHUTDOWN_PRIMARY		VARCHAR2(5)
CON_ID		NUMBER

```
SQL> SELECT fsfo_mode, status, current_target, threshold, observer_present, observer_host,
2> protection_mode, lag_limit, auto_reinstate, observer_override, shutdown_primary FROM V$FAST_START_FAILOVER_CONFIG;
```

FSFO_MODE	STATUS	CURRENT_TARGET	THRESHOLD	OBSERVE	OBSERVER_HOST	PROTECTION_MODE	LAG_LIMIT	AUTO_	OBSER	SHUTD
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
POTENTIAL DATA LOSS TARGET UNDER LAG LIMIT mydb_site2		180	YES	mydb-obs	MaxPerformance	300	TRUE	FALSE	TRUE	

Note: V\$DATABASE columns starting with FS\_FAILOVER\_ are therefore deprecated.





# Fast-Start Failover Lag Histogram

The view **V\$FS\_LAG\_HISTOGRAM** displays the frequency of Fast-Start Failover lags.

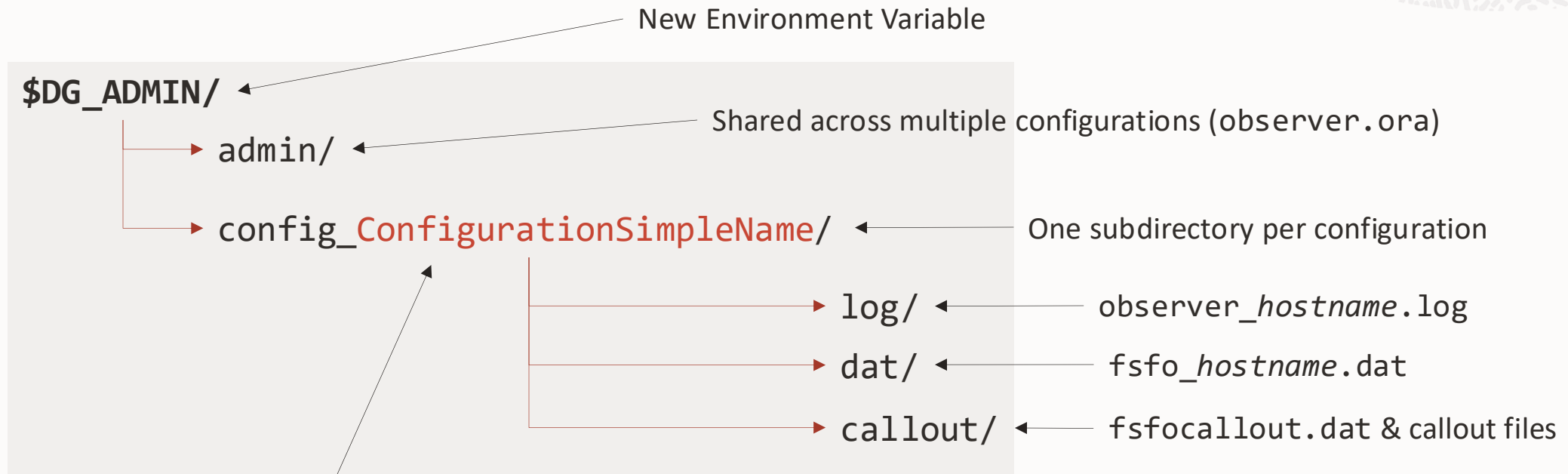
```
SQL> select * from v$fs_lag_histogram;
```

THREAD#	LAG_TYPE	LAG_TIME	LAG_COUNT	LAST_UPDATE_TIME	CON_ID
1	APPLY	5	122	01/23/2025 10:46:07	0
1	APPLY	10	5	01/02/2025 16:12:42	0
1	APPLY	15	2	12/25/2024 12:01:23	0
1	APPLY	30	0		0
1	APPLY	60	0		0
1	APPLY	120	0		0
1	APPLY	180	0		0
1	APPLY	300	0		0
1	APPLY	65535	0		0

- Useful to calculate the optimal **FastStartFailoverLagTime** property.
- It shows also the most recent occurrence for each bucket.
- **LAG\_TIME** is the upper bound of the bucket:
  - 5 -> between 0 and 5 seconds
  - 10 -> between 5 and 10 seconds
  - etc.
- It's calculated every minute, only when Fast-Start Failover is enabled (also in observe-only mode)

# Data Guard Broker Client Side Standardized Directory Structure

A single environment variable to define all the locations



New configuration property. It defaults to the Configuration Name

## Location of Client-side Broker Files

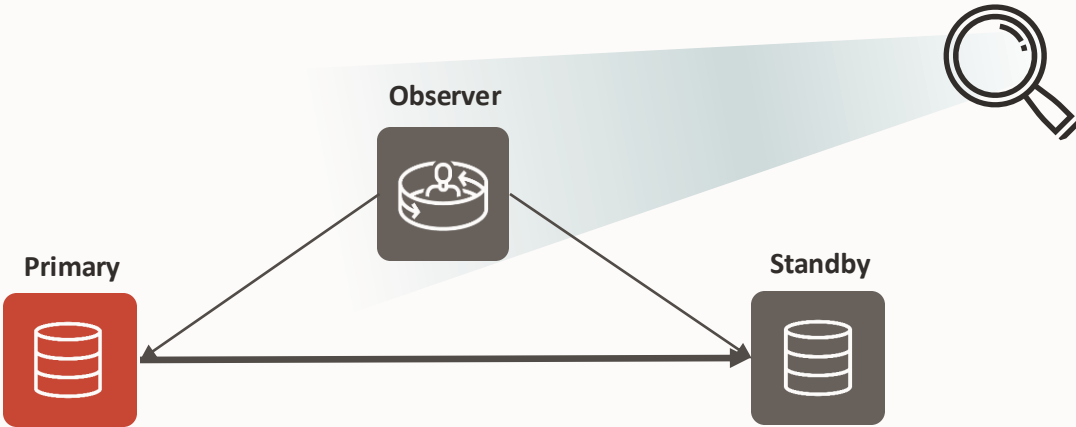
<https://docs.oracle.com/en/database/oracle/oracle-database/23/dgbkr/using-data-guard-broker-to-manage-switchovers-failovers.html#GUID-0C8473F6-33B5-479F-9208-9CA651F1B483>

# Enhanced observer diagnostic

New columns in V\$FS\_FAILOVER\_OBSERVERS with additional details

New columns:

- LAST\_PING\_PRIMARY
- LAST\_PING\_TARGET
- LOG\_FILE
- STATE\_FILE
- CURRENT\_TIME



```
SQL> select name, registered, host, ismaster, pinging_primary, pinging_target ,
2> last_ping_primary, last_ping_target, log_file, state_file, current_time
3> from V$FS_FAILOVER_OBSERVERS where host is not null;
```

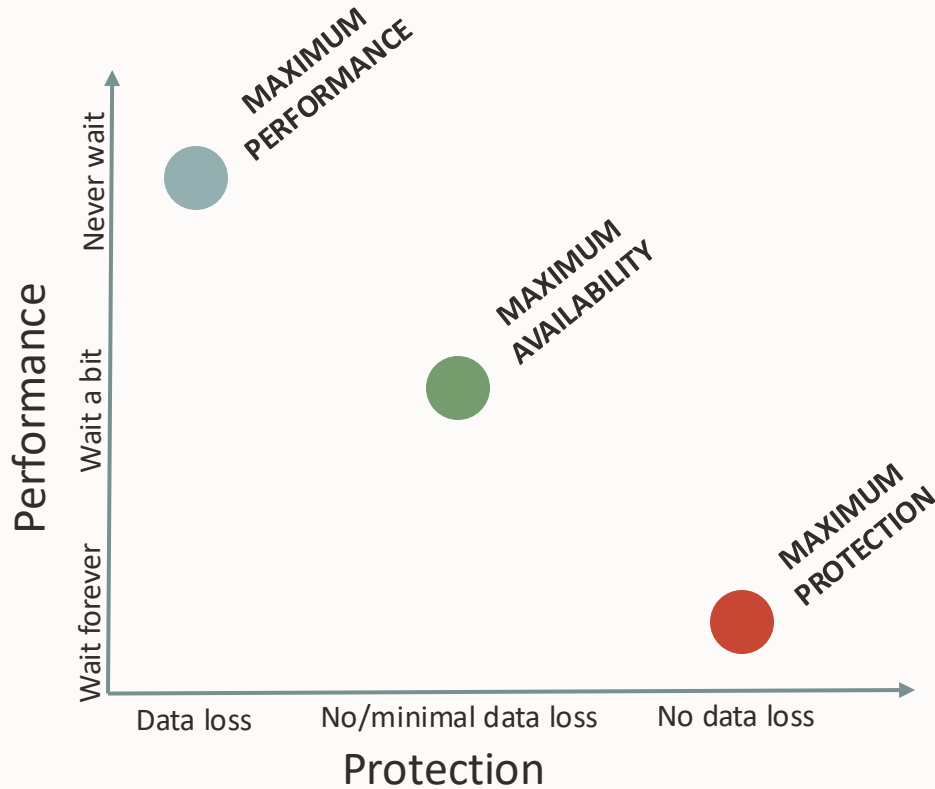
NAME	REGI	HOST	ISMA	PING	PING	LAST_PING_PRIMARY	LAST_PING_TARGET	LOG_FILE	STATE_FILE	CURRENT_TIME
host-obs	YES	host-obs	YES	YES	YES	0	2	/.../observer.lst	/.../observer.dat	06-OCT-24 06.38.14.000000000 AM



# Fast-Start Failover: Oracle Data Guard Protection Modes

# Data Guard Fast-Start Failover Protection Modes

Balance Data Protection with Performance and Availability



What does the primary do if the standby does not acknowledge the transaction?

## MAXIMUM PERFORMANCE

Never waits for acknowledge (**ASYNC**).  
Some transactions might get lost when primary fails.

## MAXIMUM AVAILABILITY

Waits until *NetTimeout* seconds (**SYNC** or **FASTSYNC**), then continue without standby. Data loss possible with manual failover or within specified limit (21c).

## MAXIMUM PROTECTION

Waits until the standby is available again (**SYNC only**).  
No transactions are lost, ever.

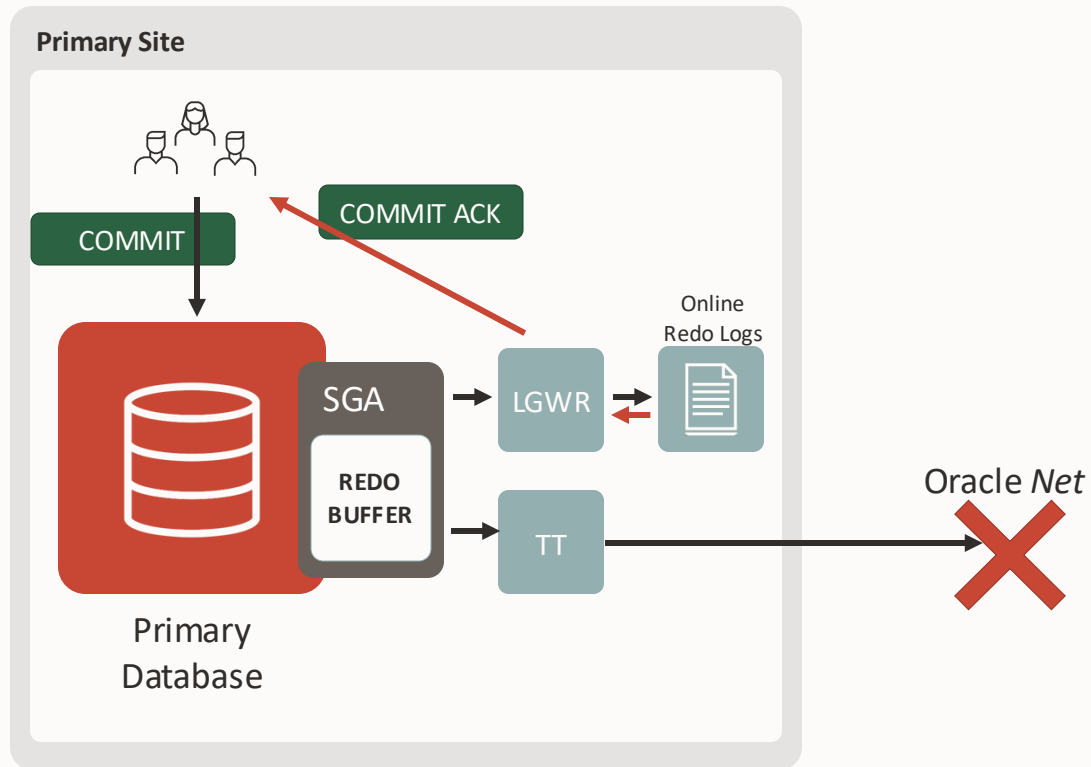


# Fast-Start Failover: Maximum Performance



# Max Performance without Fast-Start Failover

Data Guard **ASync** redo transport

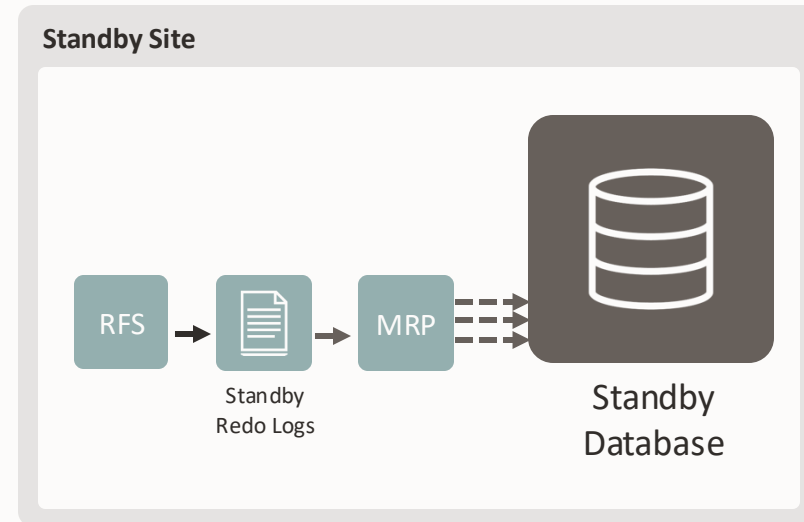


**If the standby is not reachable or is slow:**

- The primary keeps writing at its pace
- The lag (data loss exposure) increases

**If the primary fails:**

- The standby requires manual failover

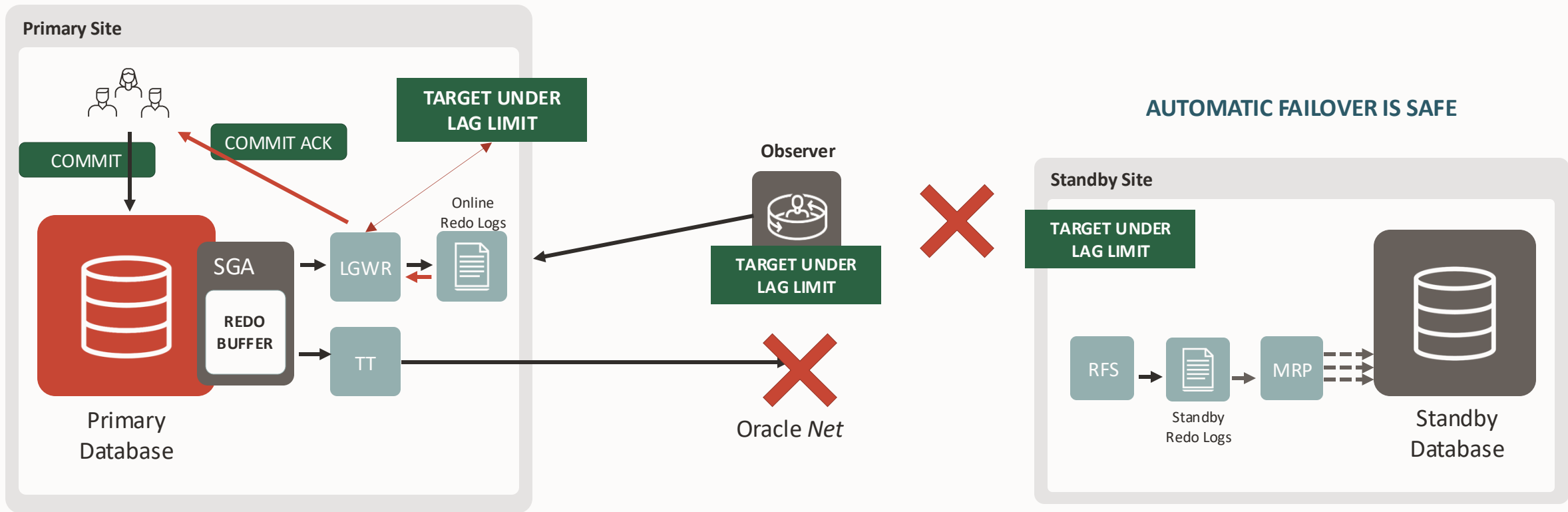


# Max Performance with Fast-Start Failover

The primary continuously computes the lag with the Fast-Start Failover Target

If the standby is not reachable or is slow:

- The primary keeps committing until reaching the lag limit

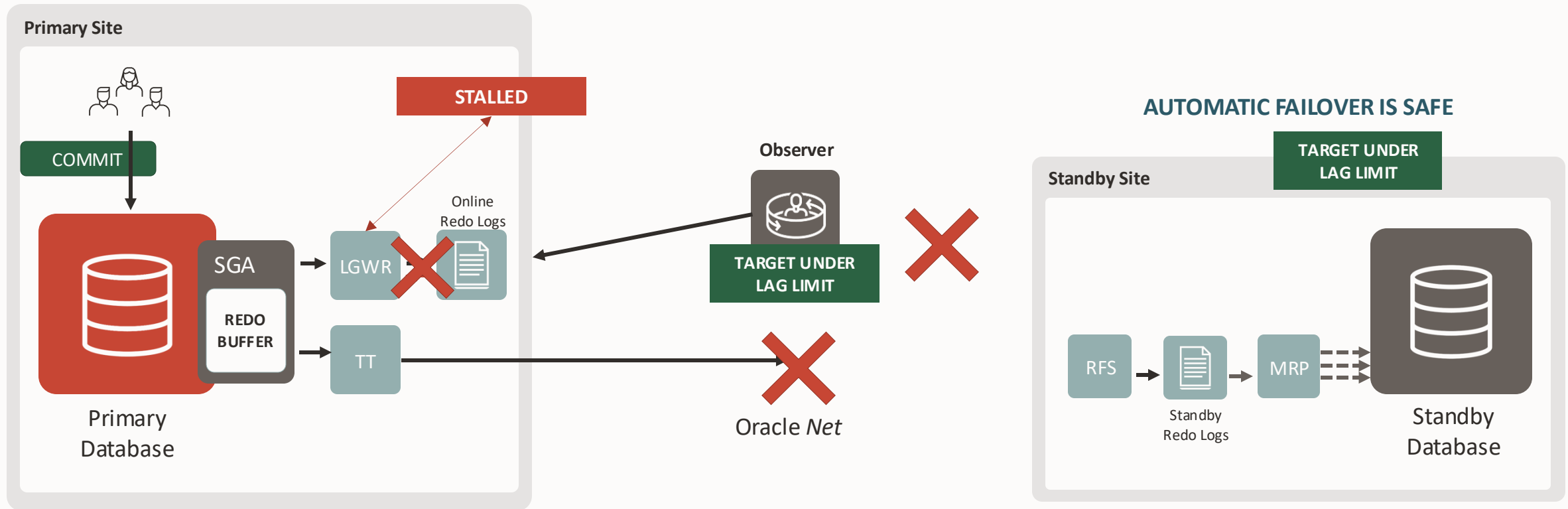


# Max Performance with Fast-Start Failover

The primary continuously computes the lag with the Fast-Start Failover Target

**If the standby is not reachable or is slow:**

- The primary keeps committing until reaching the lag limit
- Then it goes to STALLED mode (no commits possible)

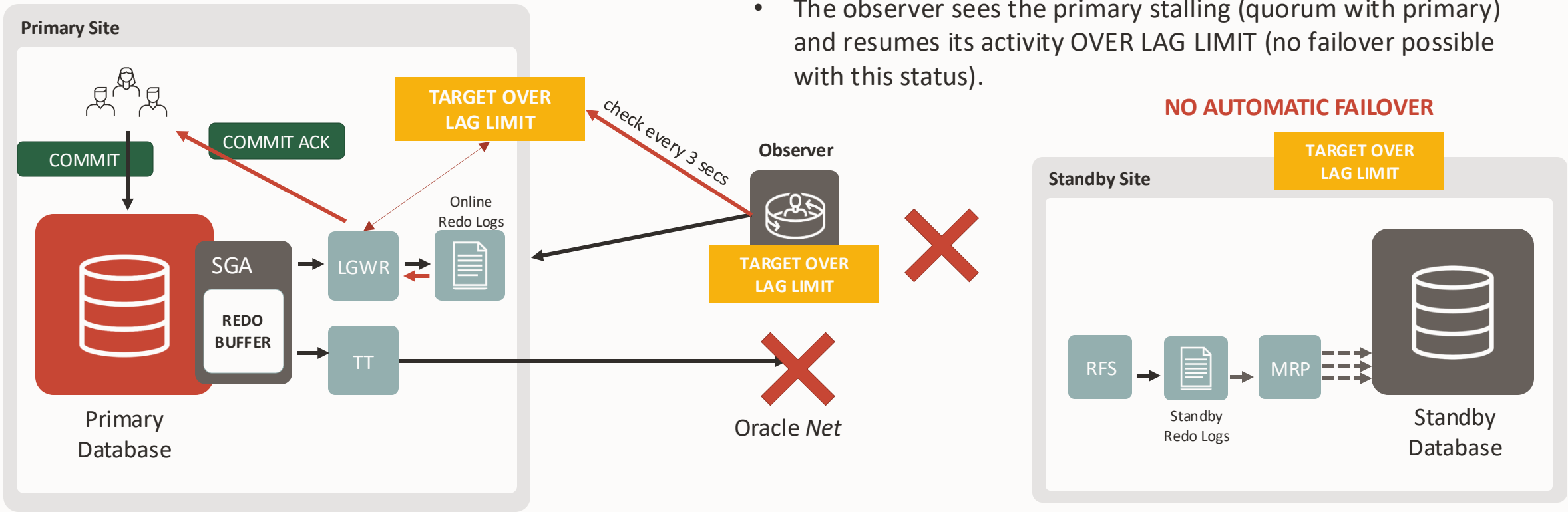


# Max Performance with Fast-Start Failover

The primary continuously computes the lag with the Fast-Start Failover Target

If the standby is not reachable or is slow:

- The primary keeps committing until reaching the lag limit
- Then it goes to STALLED mode (no commits possible)
- The observer sees the primary stalling (quorum with primary) and resumes its activity OVER LAG LIMIT (no failover possible with this status).

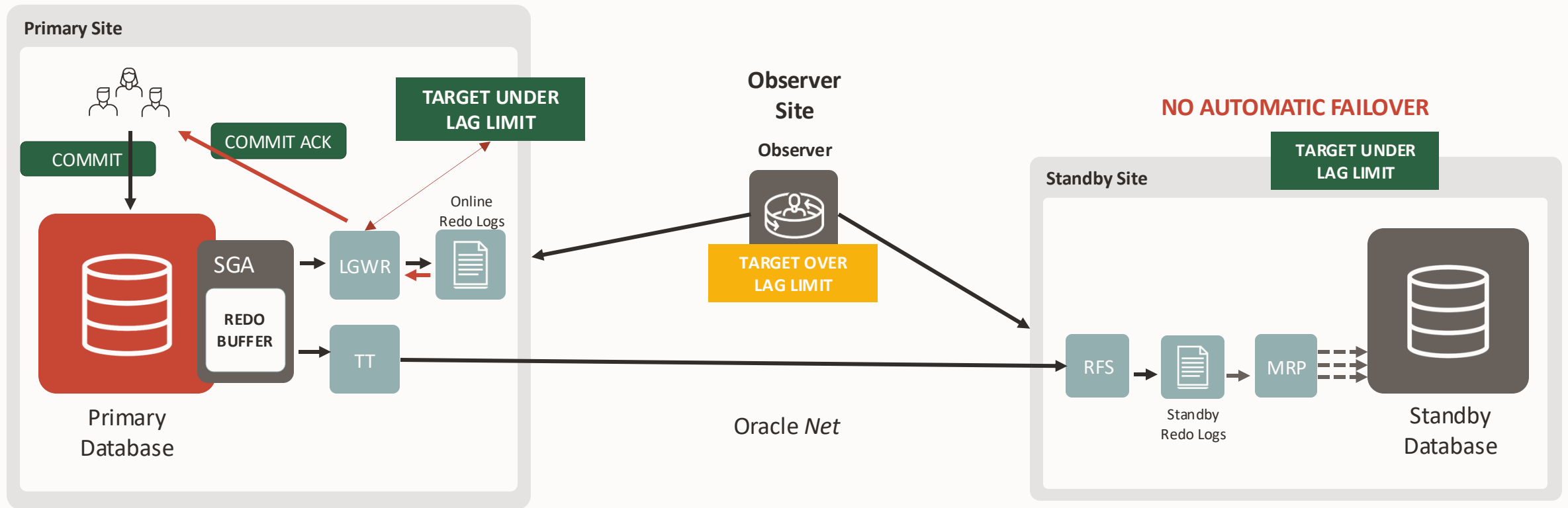


# Max Performance with Fast-Start Failover

The primary continuously computes the lag with the Fast-Start Failover Target

When the standby catches up with the primary

- The primary goes autonomously under lag limit

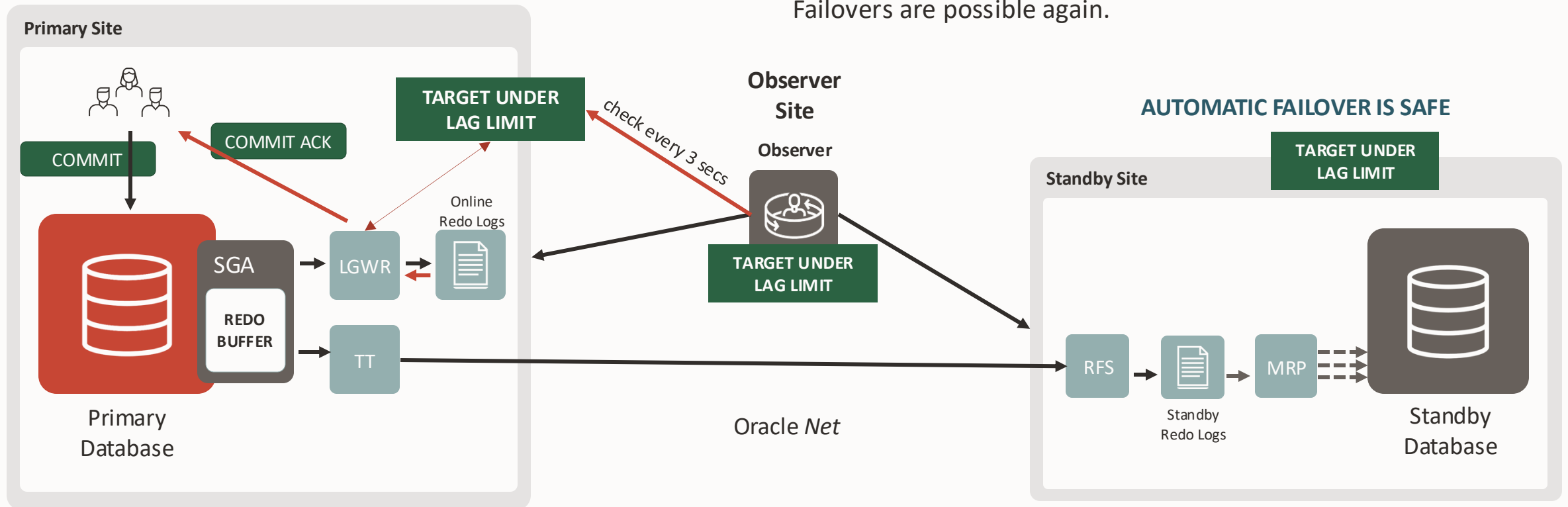


# Max Performance with Fast-Start Failover

The primary continuously computes the lag with the Fast-Start Failover Target

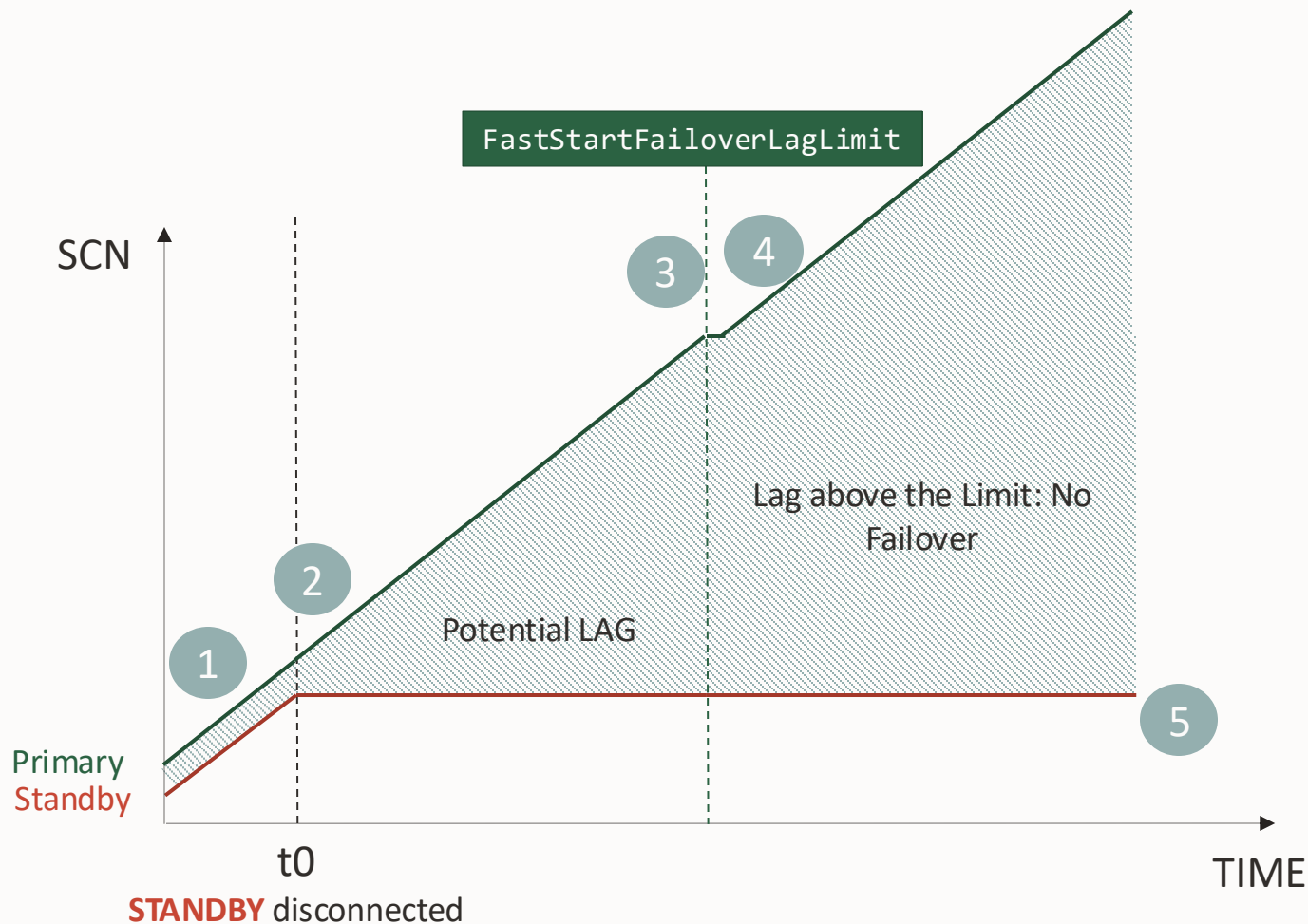
## When the standby catches up with the primary

- The primary goes autonomously under lag limit
- At the next check, the observer acknowledges the change. Failovers are possible again.



# Automatic Failover with MaxPerformance

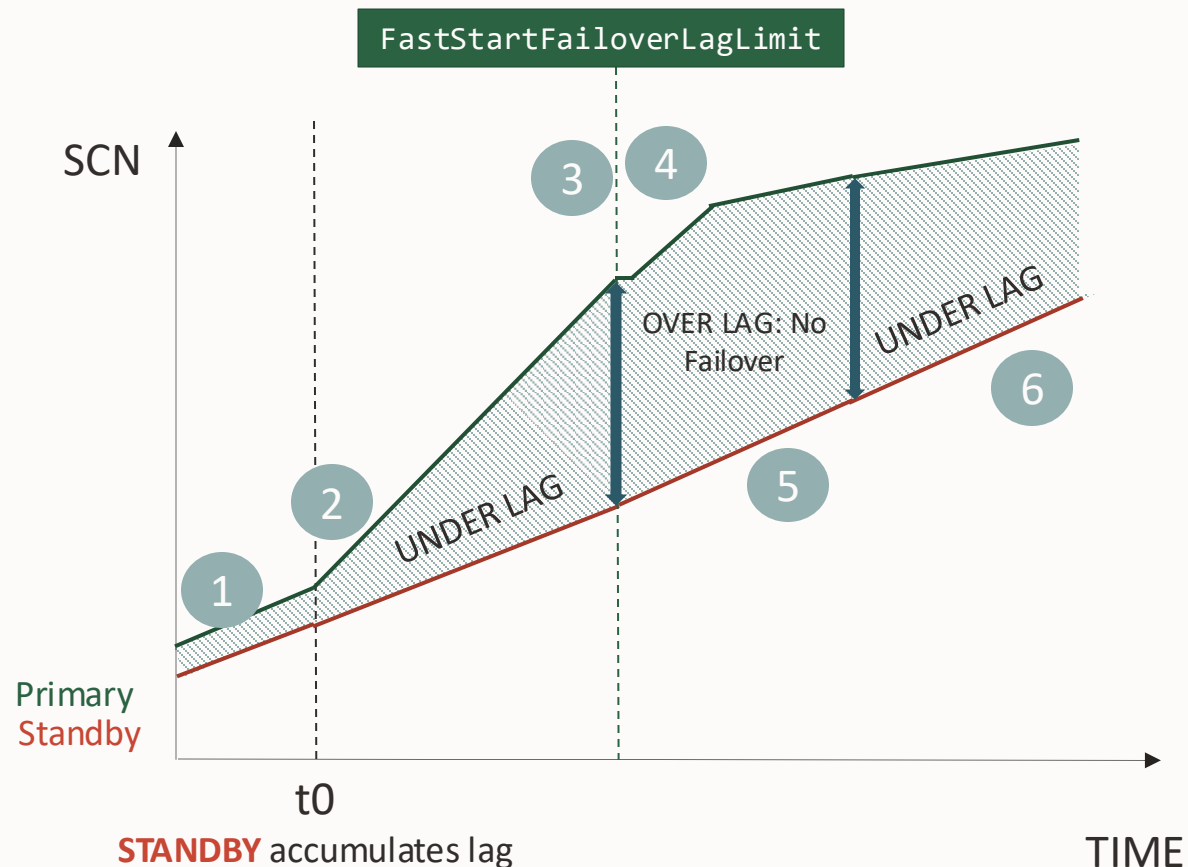
Choose how much data loss you can tolerate



- 1 ASYNC Transport. The Standby has a residual lag  
Status: TARGET UNDER LAG LIMIT
- 2 At  $t_0$  + ping time, the primary cannot contact the standby. The Primary keeps committing, the Standby lag increases.  
Status: TARGET UNDER LAG LIMIT
- 3 The primary reaches FastStartFailoverLagLimit. It temporarily stalls (~3 seconds) until it gets permission from the observer to continue.  
Status: STALLED
- 4 After the observer pings the primary and gives permission to continue, the primary resumes the commit activity. Status: TARGET OVER LAG LIMIT
- 5 The observer acknowledges that the lag is above the limit and will not permit a failover in case it loses connectivity with the primary. The standby is declared out of sync.  
Status: TARGET OVER LAG LIMIT

# Automatic Failover with MaxPerformance

Choose how much data loss you can tolerate

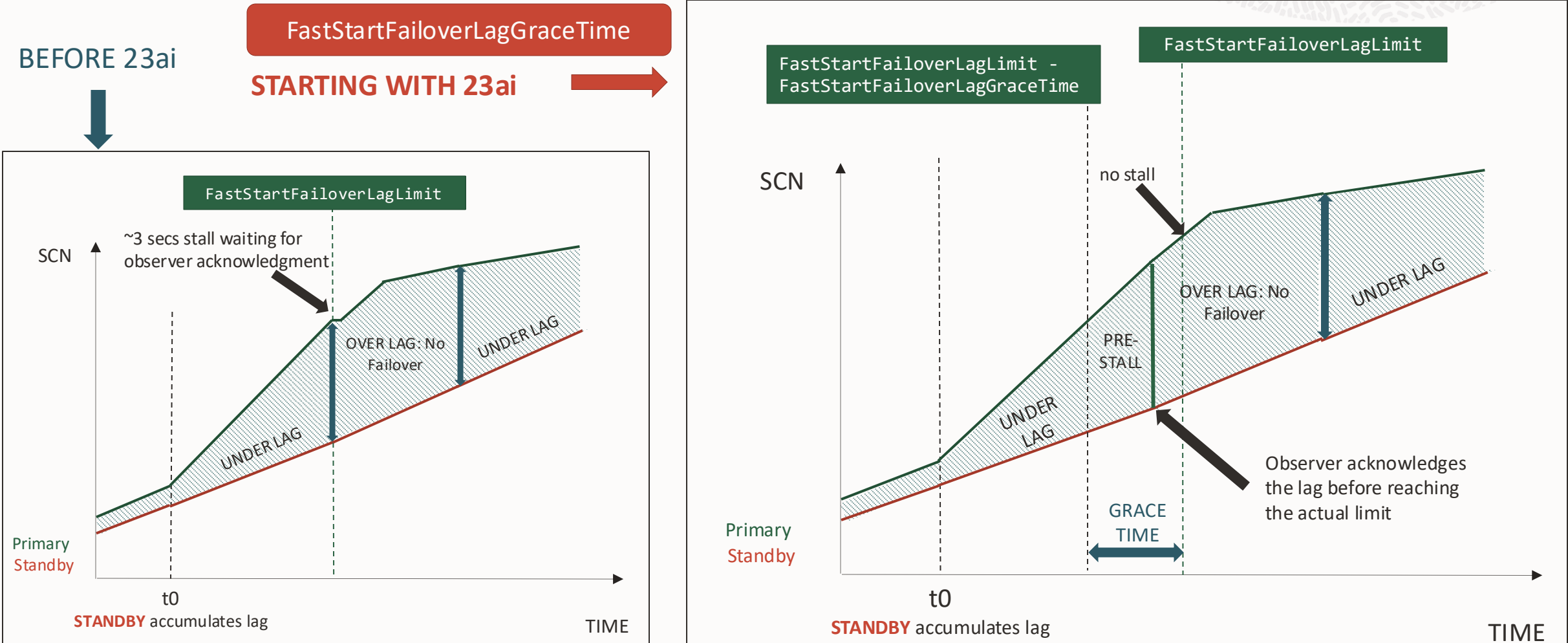


- 1 ASYNC Transport. The Standby has a residual lag  
Status: TARGET UNDER LAG LIMIT
- 2 At  $t_0$  the primary increases the activity rate.  
The Standby lag increases.  
Status: TARGET UNDER LAG LIMIT
- 3 The primary reaches `FastStartFailoverLagLimit`. It temporarily stalls (~3 seconds) until it gets permission from the observer to continue.  
Status: STALLED
- 4 After the observer pings the primary and gives permission to continue, the primary resumes the commit activity. Status: TARGET OVER LAG LIMIT
- 5 The observer acknowledges that the lag is above the limit and will not permit a failover in case it loses connectivity with the primary. The standby is declared out of sync.  
Status: TARGET OVER LAG LIMIT
- 6 The standby catches up with the primary and the lag goes under the limit. The observer can now failover if the primary fails.  
Status: TARGET UNDER LAG LIMIT



# Minimized Stall in Fast-Start Failover Maximum Performance

Reduce/avoid delays during FSFO state transition to "OVER LAG LIMIT"

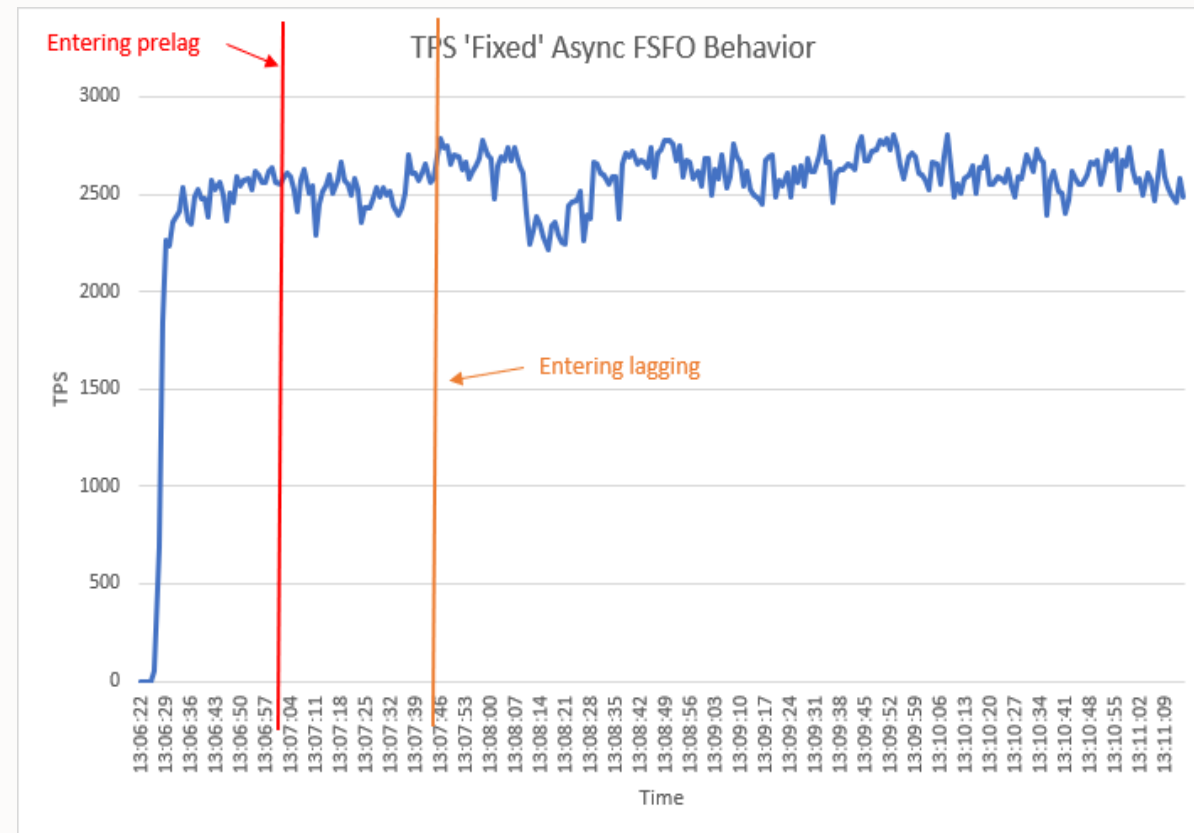
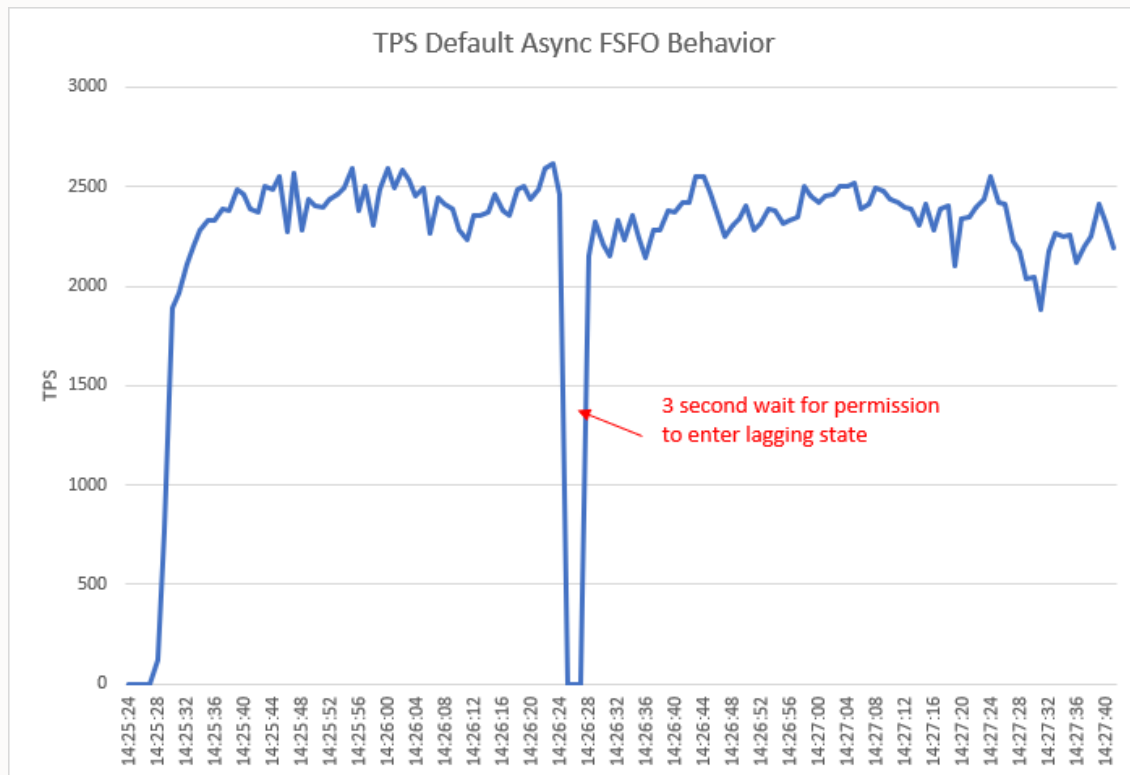


\* 23ai or 19.19 and later which include patch 34995066: MINIMIZE STALL IN DATA-LOSS FAST-START FAILOVER



# Minimized Stall in Fast-Start Failover Maximum Performance

Reduce/avoid delays during FSFO state transition to "OVER LAG LIMIT"



## Choose how much data loss you can tolerate

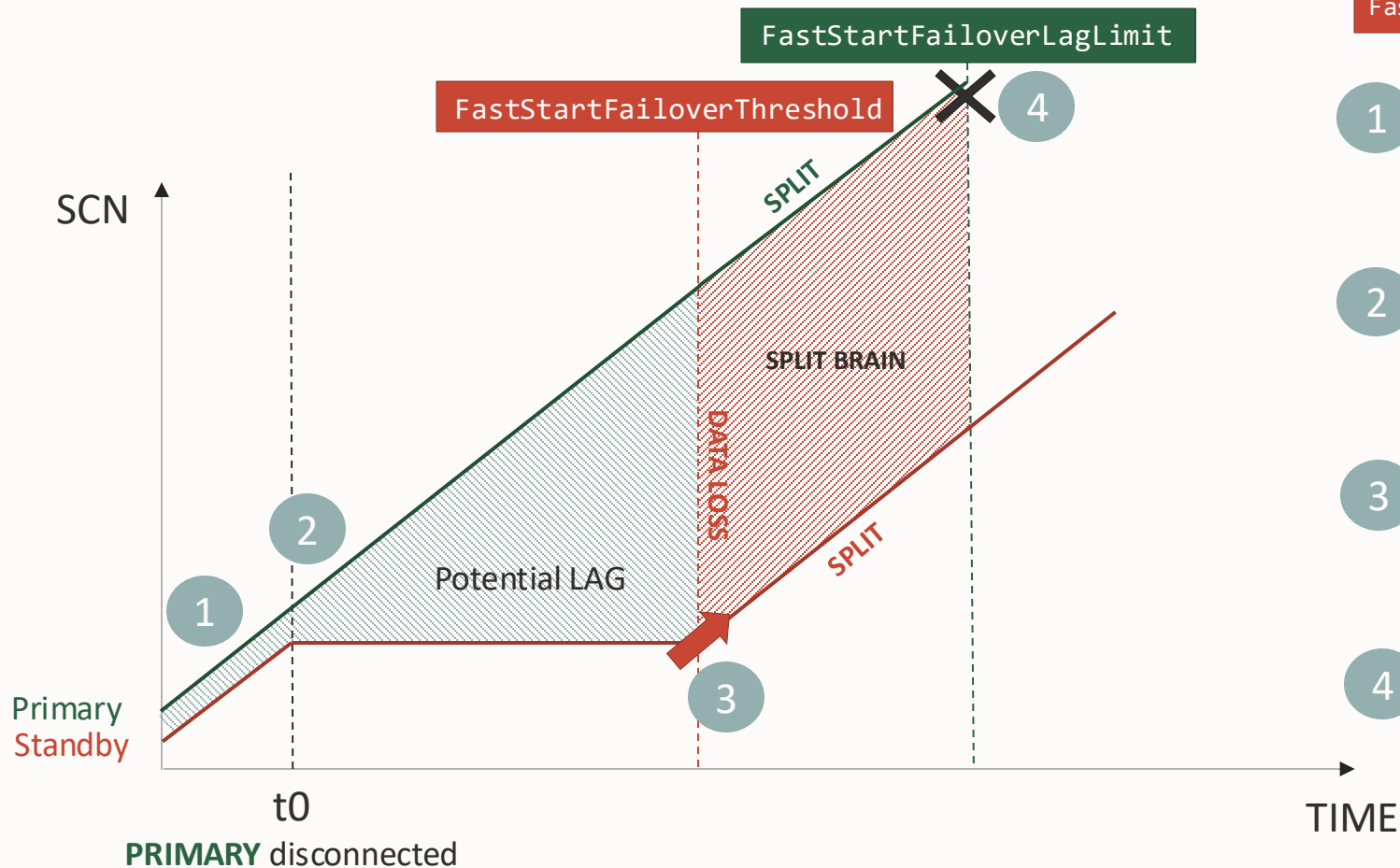
The diagram illustrates the FastStartFailover mechanism. The vertical axis represents the Sequence Number (SCN), and the horizontal axis represents time. A green line represents the primary's SCN, and a red line represents the standby's SCN. The primary disconnects at time  $t_0$ . The standby starts a redo apply. A 'Potential LAG' is shown as the gap between the lines. A 'STALL' occurs when the lag reaches the 'FastStartFailoverLagLimit'. A 'DATA LOSS' occurs when the lag reaches the 'FastStartFailoverThreshold'. A 'New primary' is elected after the data loss.

- 75 Copyright © 2025, Oracle and/or its affiliates



# Automatic Failover with MaxPerformance

Set the FastStartFailoverLagLimit wisely to avoid split-brain conditions



`FastStartFailoverThreshold` < `FastStartFailoverLagLimit`

- 1 ASYNC Transport. The Standby has a residual lag  
Status: TARGET UNDER LAG LIMIT
- 2 At  $t_0$  + ping time, the observer cannot contact the primary and starts a timer for `FastStartFailoverThreshold` seconds. The Primary keeps committing, the Standby lag increases.  
Status: TARGET UNDER LAG LIMIT
- 3 The observer timer reaches `FastStartFailoverThreshold`. Still no connection with the primary: it initiates the Failover  
Primary status: TARGET UNDER LAG LIMIT  
Standby status: REINSTATE REQUIRED
- 4 The Primary keeps committing until it reaches `FastStartFailoverLagLimit`, then stalls or shuts down. A Split-Brain condition may occur depending on timings and parameter values.  
Primary status: STALLED or REINSTATE REQUIRED  
Standby status: REINSTATE REQUIRED

# Choose the Lag Type for Maximum Performance Mode

## New Property FastStartFailoverLagType

The broker can now use the standby's transport lag to determine whether a data loss situation exists. The amount of tolerated data loss is still set with **FastStartFailoverLagLimit**.

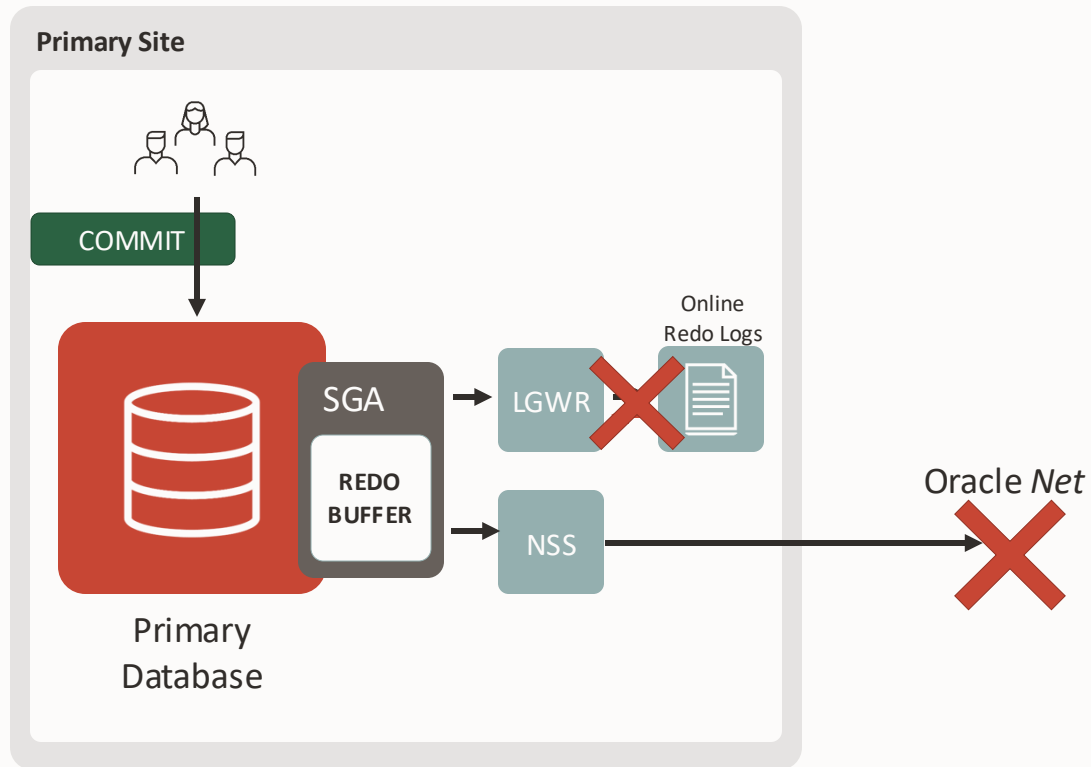
Before 23ai	Starting with 23ai
Only the <b>APPLY lag</b> is used to determine the data loss exposure.	<p>The new property <b>FastStartFailoverLagType</b> property can be used to choose which type of lag should be used.</p> <p>It can be <b>TRANSPORT</b> or <b>APPLY</b>.</p> <p><b>APPLY</b> is the default to keep the old behavior.</p>



# Fast-Start Failover: Maximum Availability

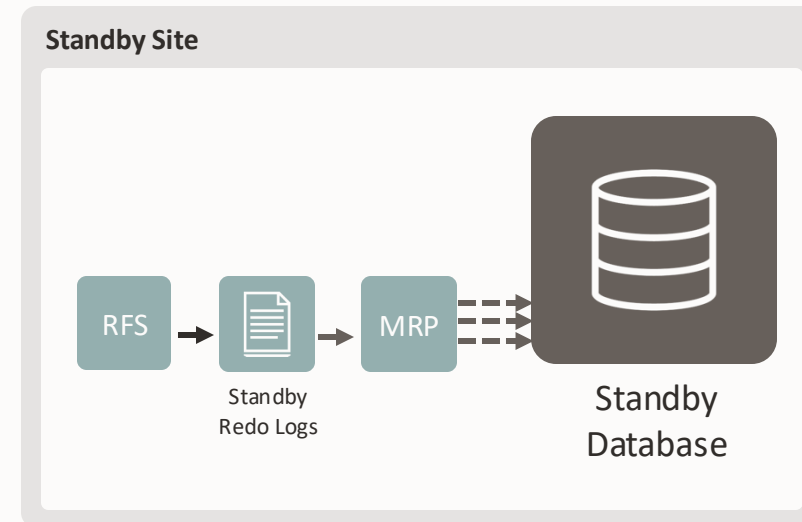
# Max Availability without Fast-Start Failover

Data Guard **SYNC** redo transport



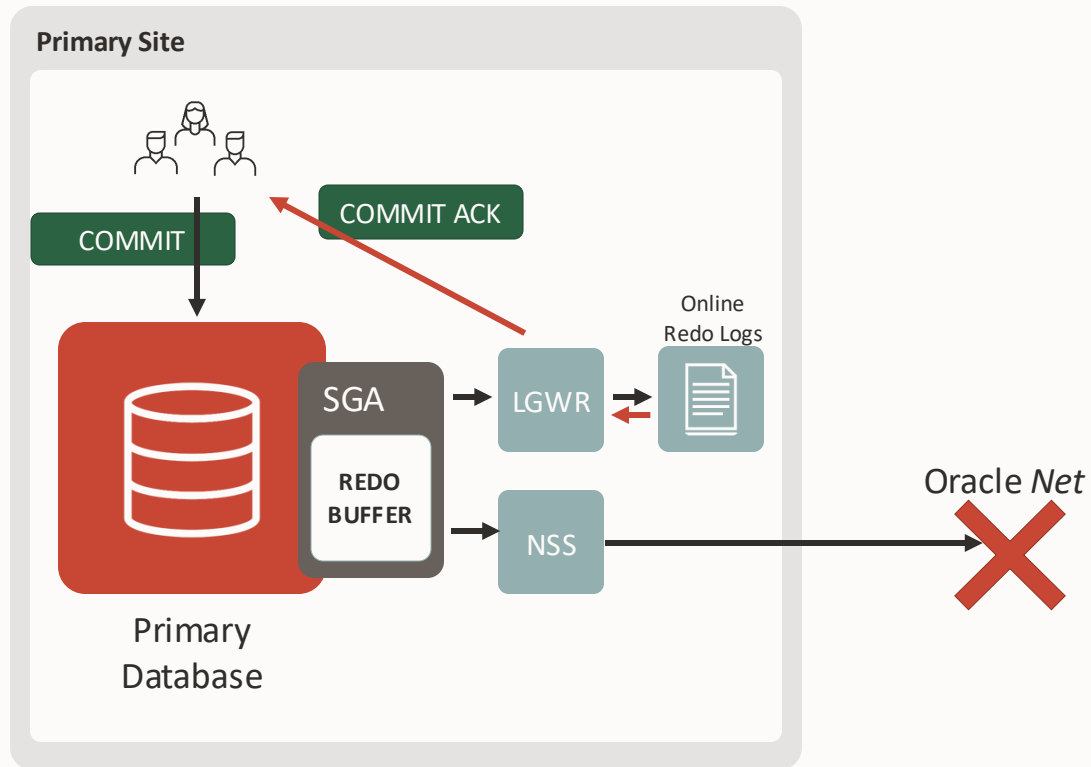
**If the standby is not reachable:**

- The primary stalls waiting for the SYNC destination (no commits possible)



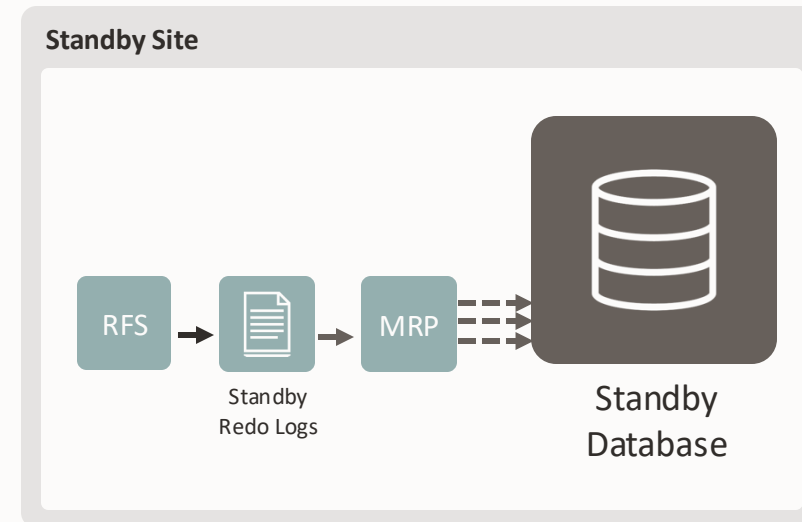
# Max Availability without Fast-Start Failover

Data Guard **SYNC** redo transport



## If the standby is not reachable:

- The primary stalls waiting for the SYNC destination (no commits possible)
- After NetTimeout seconds, it resumes the activity without protection



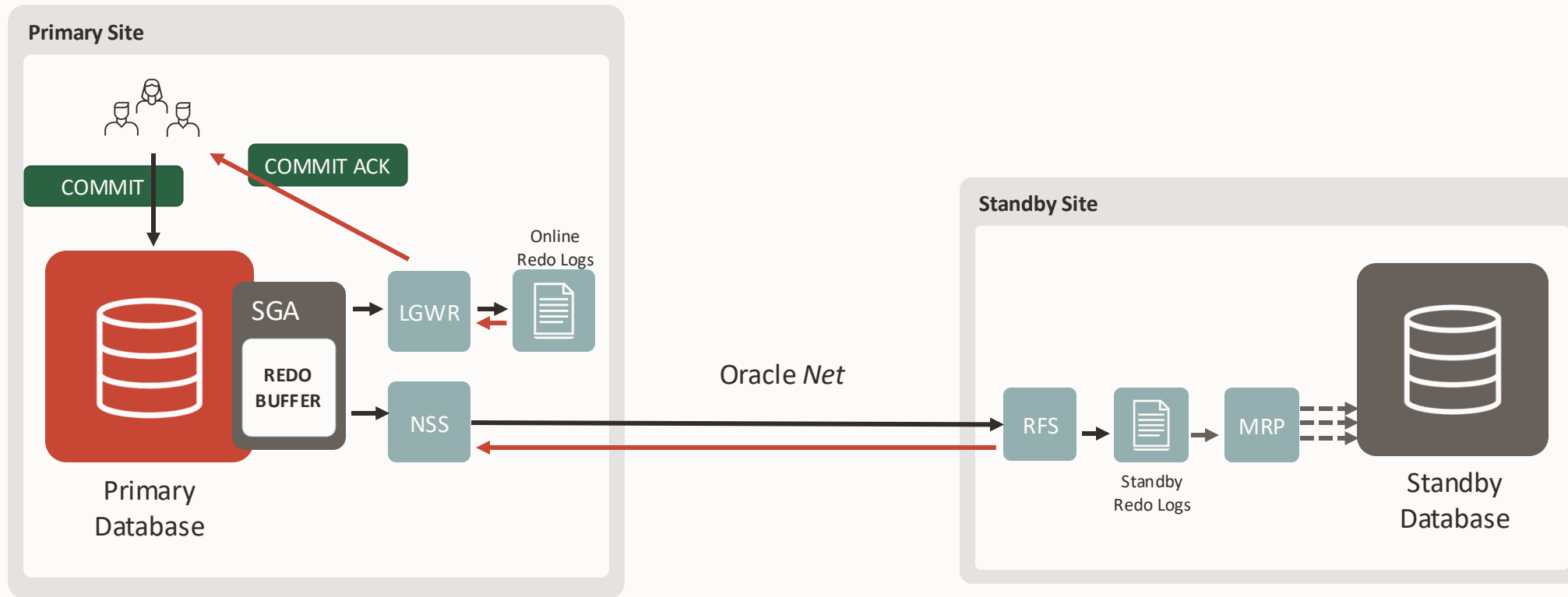


# Max Availability without Fast-Start Failover

Data Guard **SYNC** redo transport

## If the standby is slow:

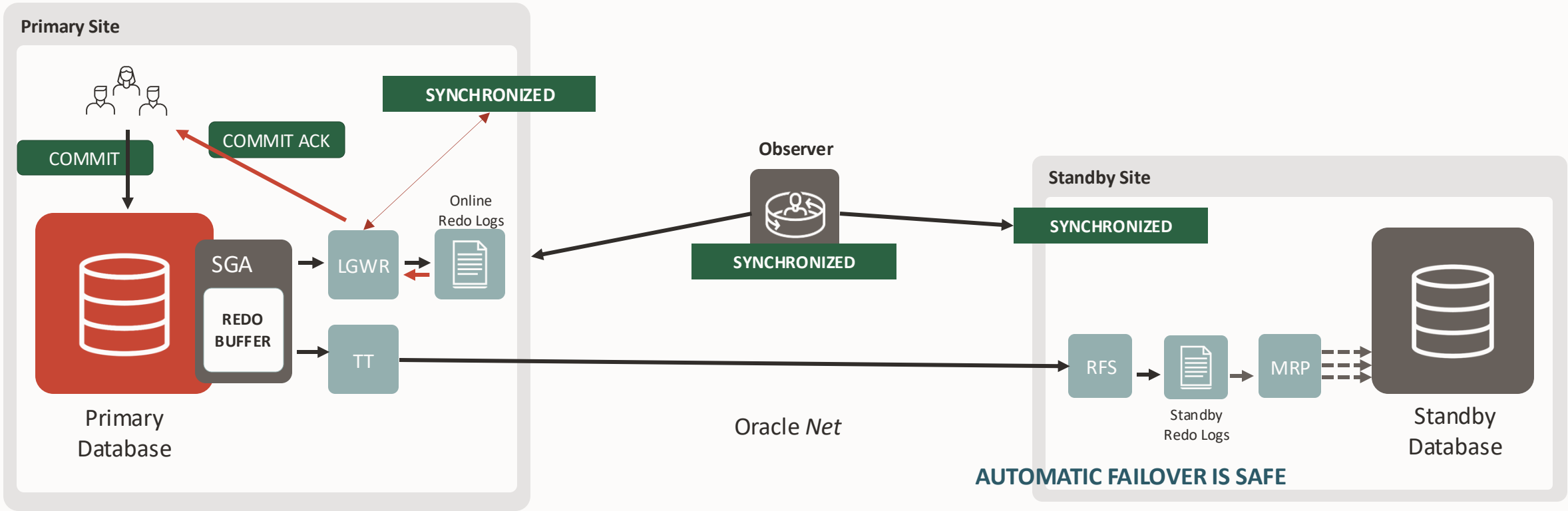
- The commits will take longer, decreasing the primary database performance
- Latencies (disk and/or network) have a crucial role



# Max Availability with Fast-Start Failover

The primary never commits without the observer quorum

In a normal situation, the status is SYNCHRONIZED

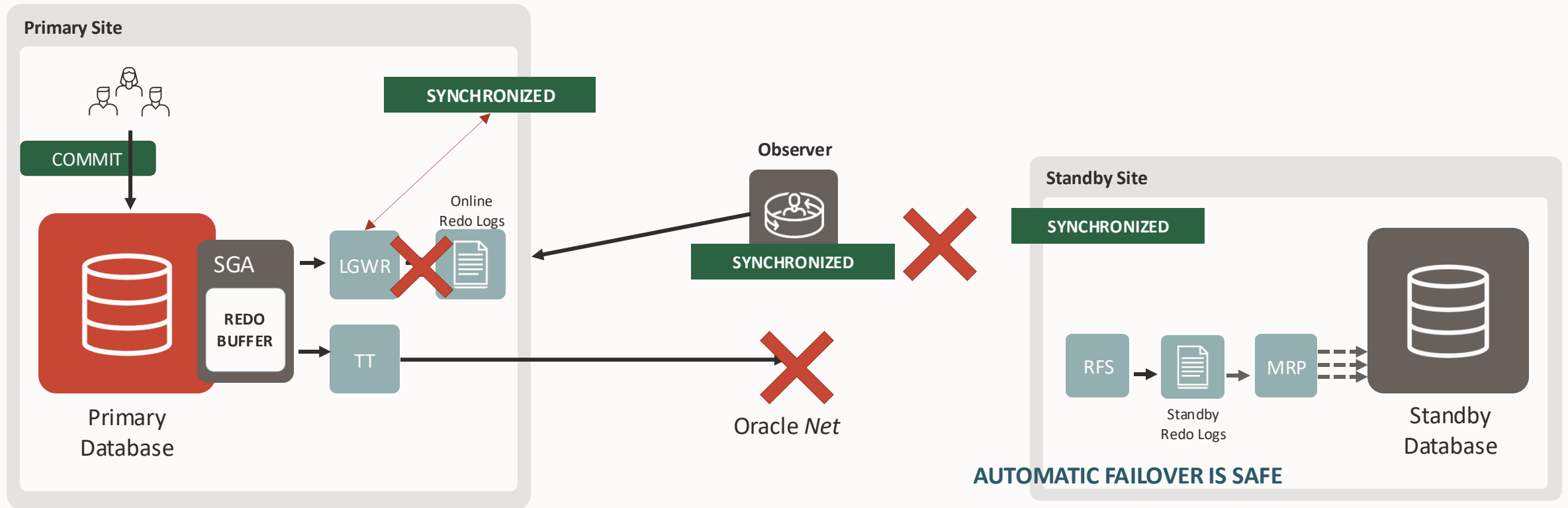


# Max Availability with Fast-Start Failover

The primary never commits without the observer quorum

## If the standby is not reachable:

- The primary waits for the SYNC destination (no commits possible) but keeps the status synchronized

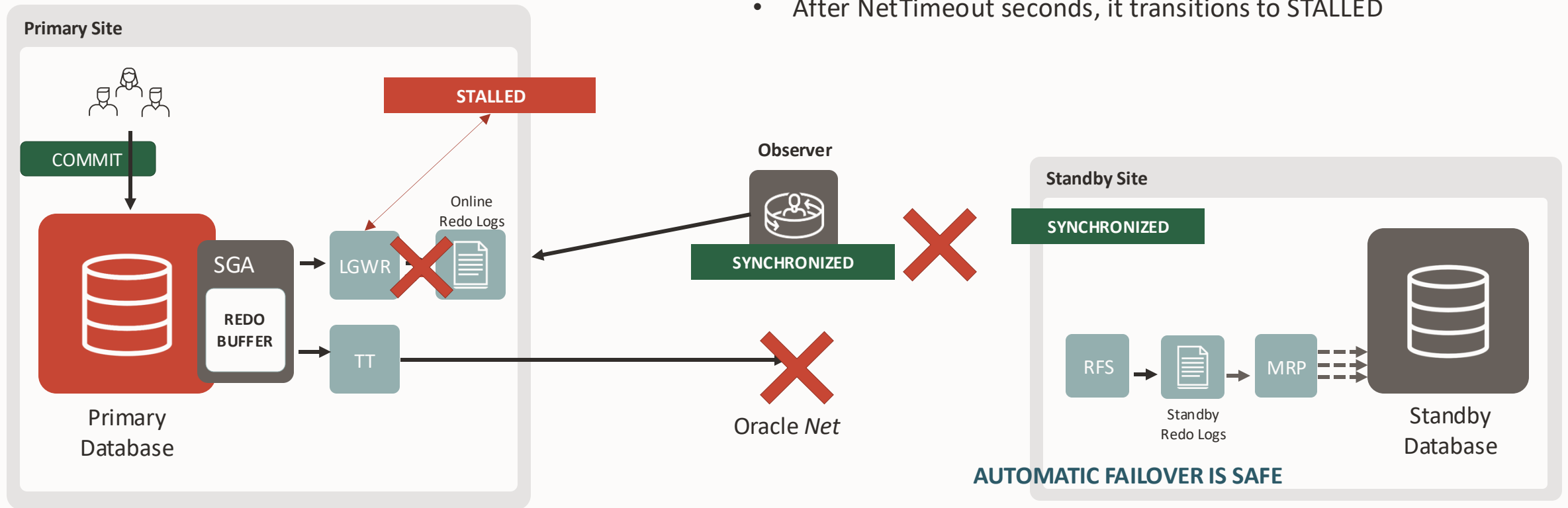


# Max Availability with Fast-Start Failover

The primary never commits without the observer quorum

## If the standby is not reachable:

- The primary waits for the SYNC destination (no commits possible) but keeps the status synchronized
- After NetTimeout seconds, it transitions to STALLED

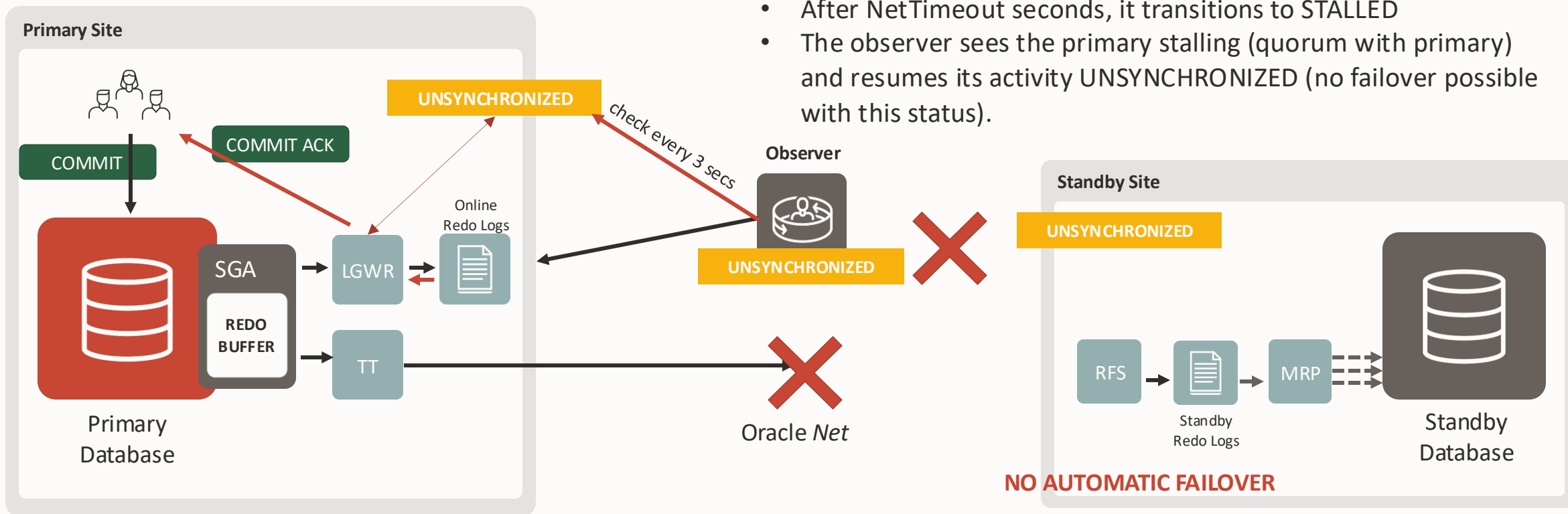


# Max Availability with Fast-Start Failover

The primary never commits without the observer quorum

## If the standby is not reachable:

- The primary waits for the SYNC destination (no commits possible) but keeps the status synchronized
- After NetTimeout seconds, it transitions to STALLED
- The observer sees the primary stalling (quorum with primary) and resumes its activity UNSYNCHRONIZED (no failover possible with this status).

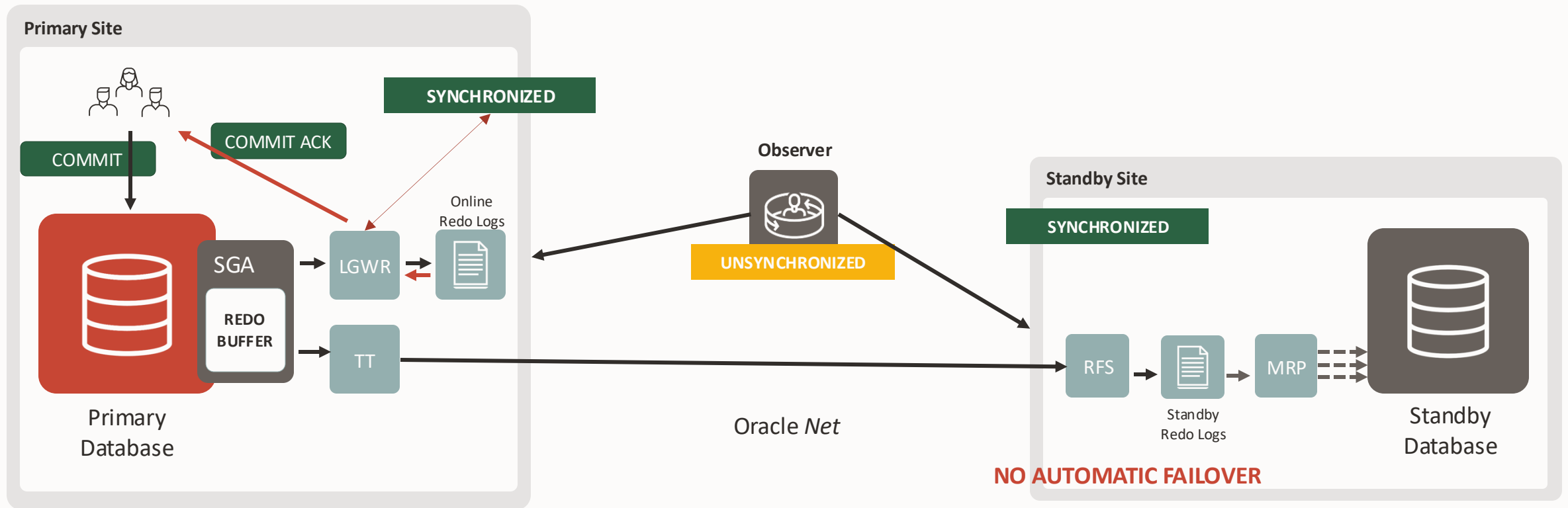


# Max Availability with Fast-Start Failover

The primary never commits without the observer quorum

## When the standby is in SYNC again

- The primary autonomously change the status to SYNCHRONIZED

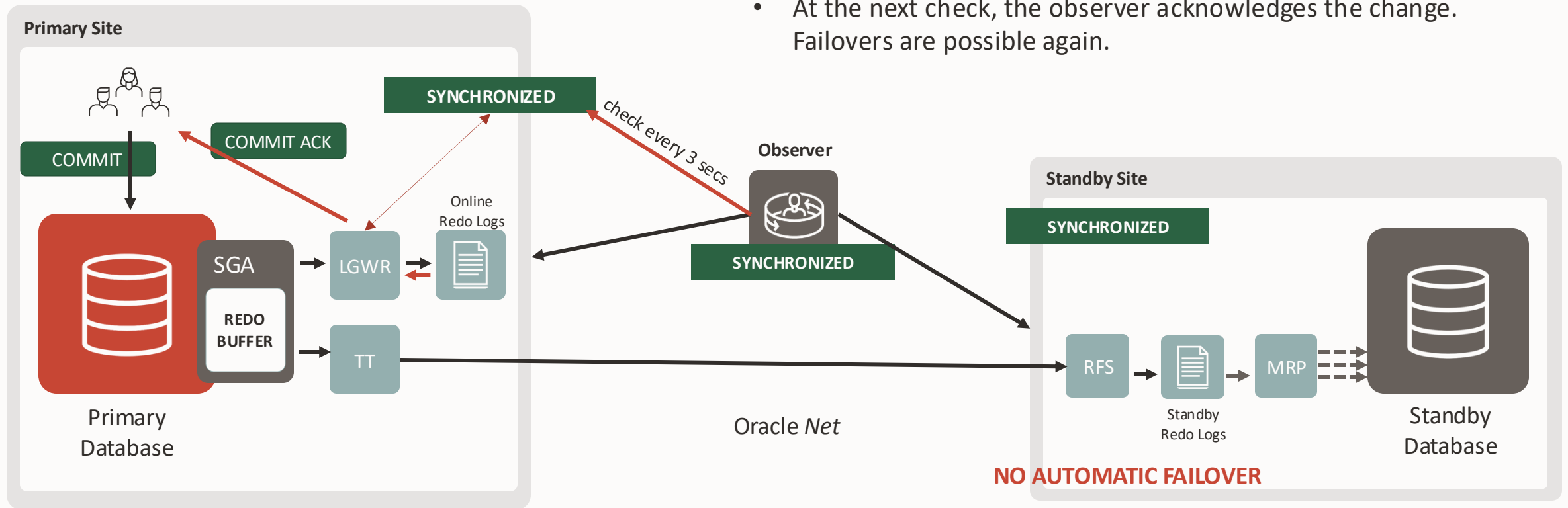


# Max Availability with Fast-Start Failover

The primary never commits without the observer quorum

## When the standby is in SYNC again

- The primary autonomously change the status to SYNCHRONIZED
- At the next check, the observer acknowledges the change. Failovers are possible again.



# Automatic Failover with MaxAvailability

Failover only if Zero Data Loss is guaranteed

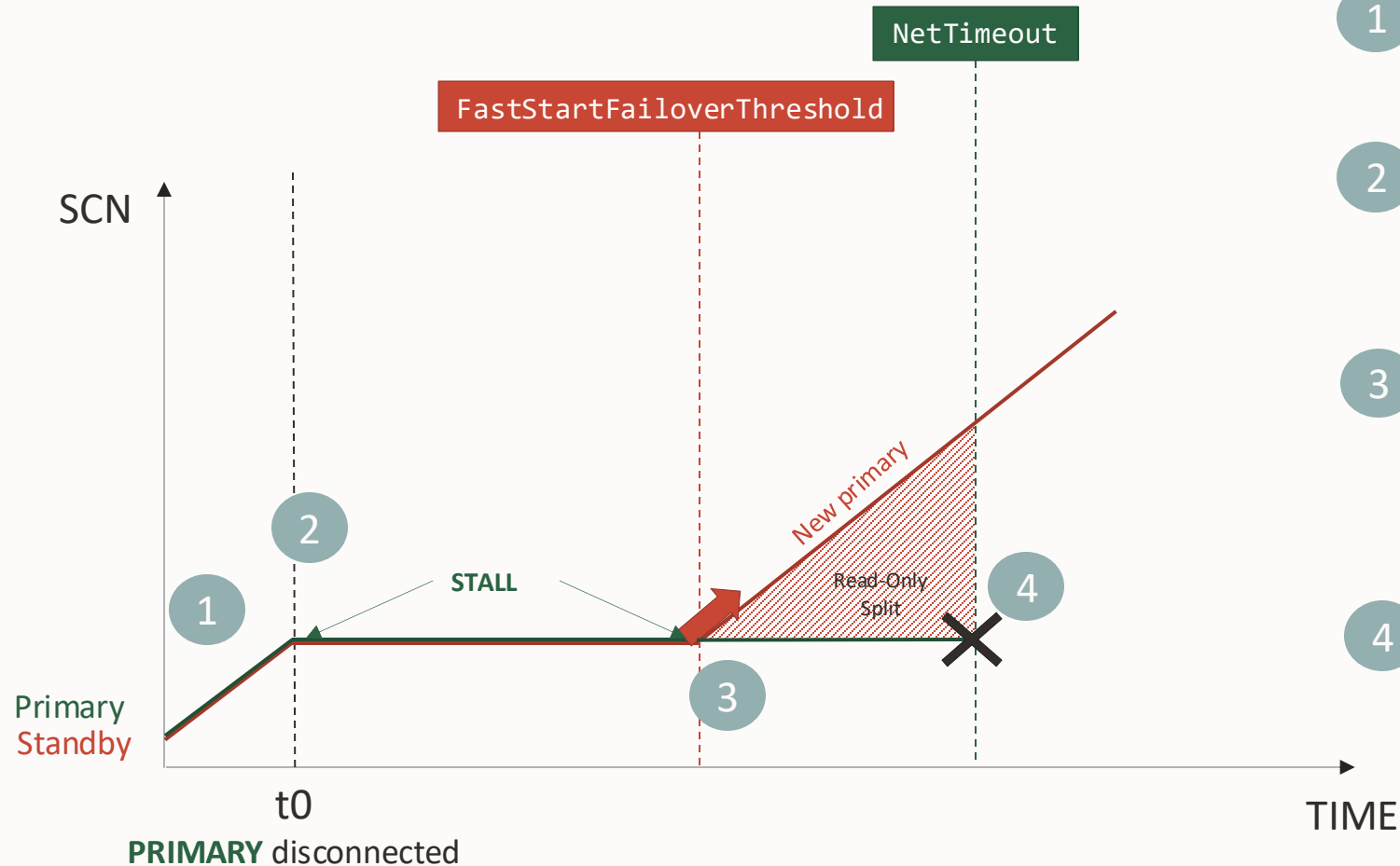


- 1 SYNC Transport. The Standby is synced.  
Status: SYNCHRONIZED
- 2 At  $t_0$  + ping time, the primary and observer cannot contact the standby. The primary stalls and keeps retrying for NetTimeout seconds.  
Status: STALLED
- 3 At NetTimeout seconds, the primary asks and obtain permission from the observer to stop the redo transport to the destination. The standby is declared unsynchronized.  
Status: UNSYNCHRONIZED
- 4 The Observer knows that the standby is out of sync. If later the connection with the primary is lost, and the standby is back, the observer will not initiate a failover because of the unsynched status, unless FastStartFailoverLagLimit is set.  
Status: UNSYNCHRONIZED



# Automatic Failover with MaxAvailability

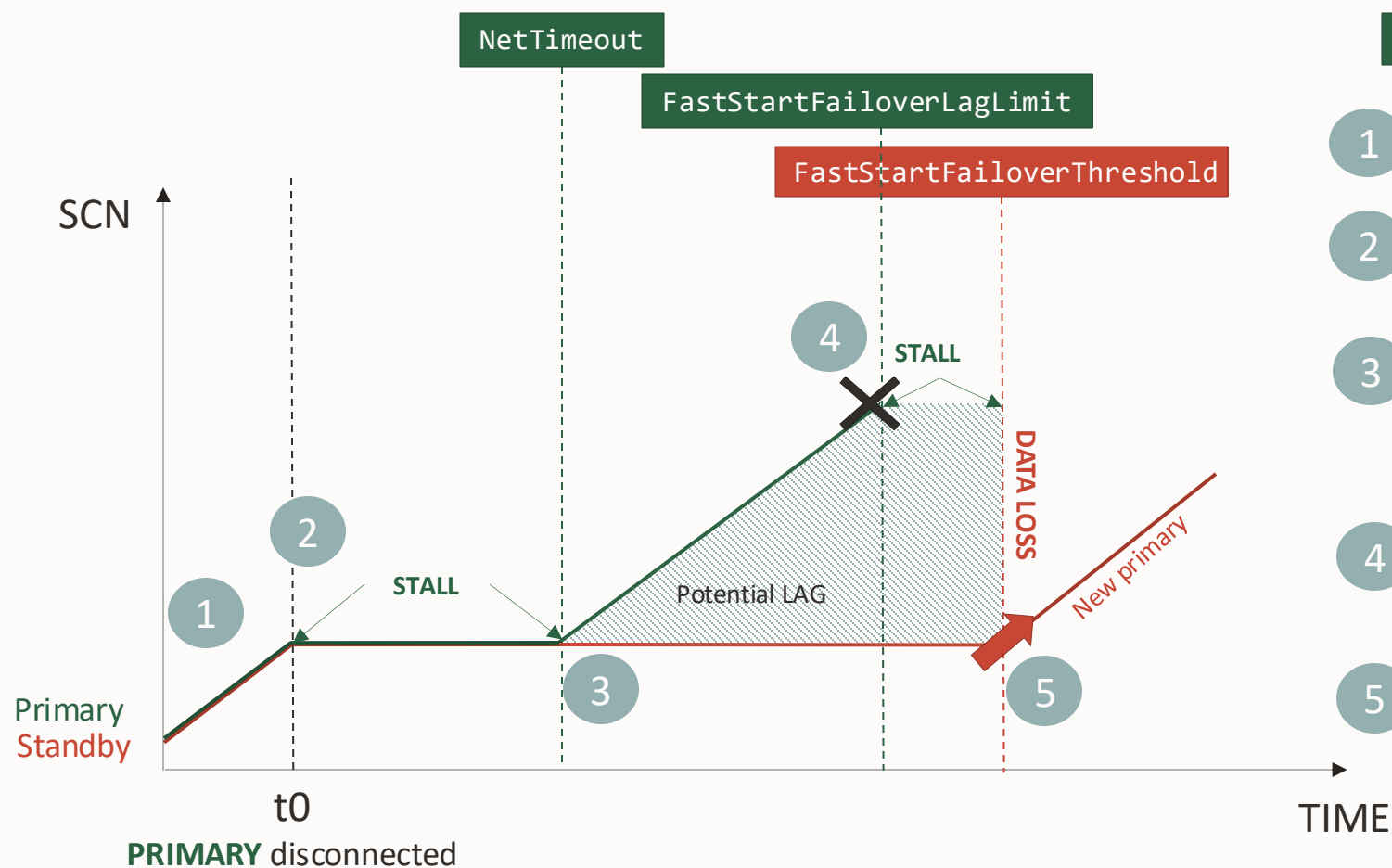
Failover only if Zero Data Loss is guaranteed



- 1 SYNC Transport. The Standby is synched.  
Status: SYNCHRONIZED
- 2 At  $t_0 + \text{ping time}$ , the observer cannot contact the primary and starts a timer for `FastStartFailoverThreshold` seconds. The Primary stalls and keeps retrying for `NetTimeout` Seconds.  
Status: STALLED
- 3 The observer timer reaches `FastStartFailoverThreshold`. Still no connection with the primary: it initiates the Failover. If `NetTimeout` is higher than the threshold, the Primary is reachable but not committing (read-only split).  
Primary status: STALLED  
Standby status: REINSTATE REQUIRED
- 4 The Primary keeps stalling or shuts down after `NetTimeout` because it cannot get the permission from the observer to abandon the destination. A lower `NetTimeout` reduces the potential of read-only split.  
Status: REINSTATE REQUIRED

# Automatic Failover with MaxAvailability

Optionally choose a limit to have an automatic failover with potential data loss



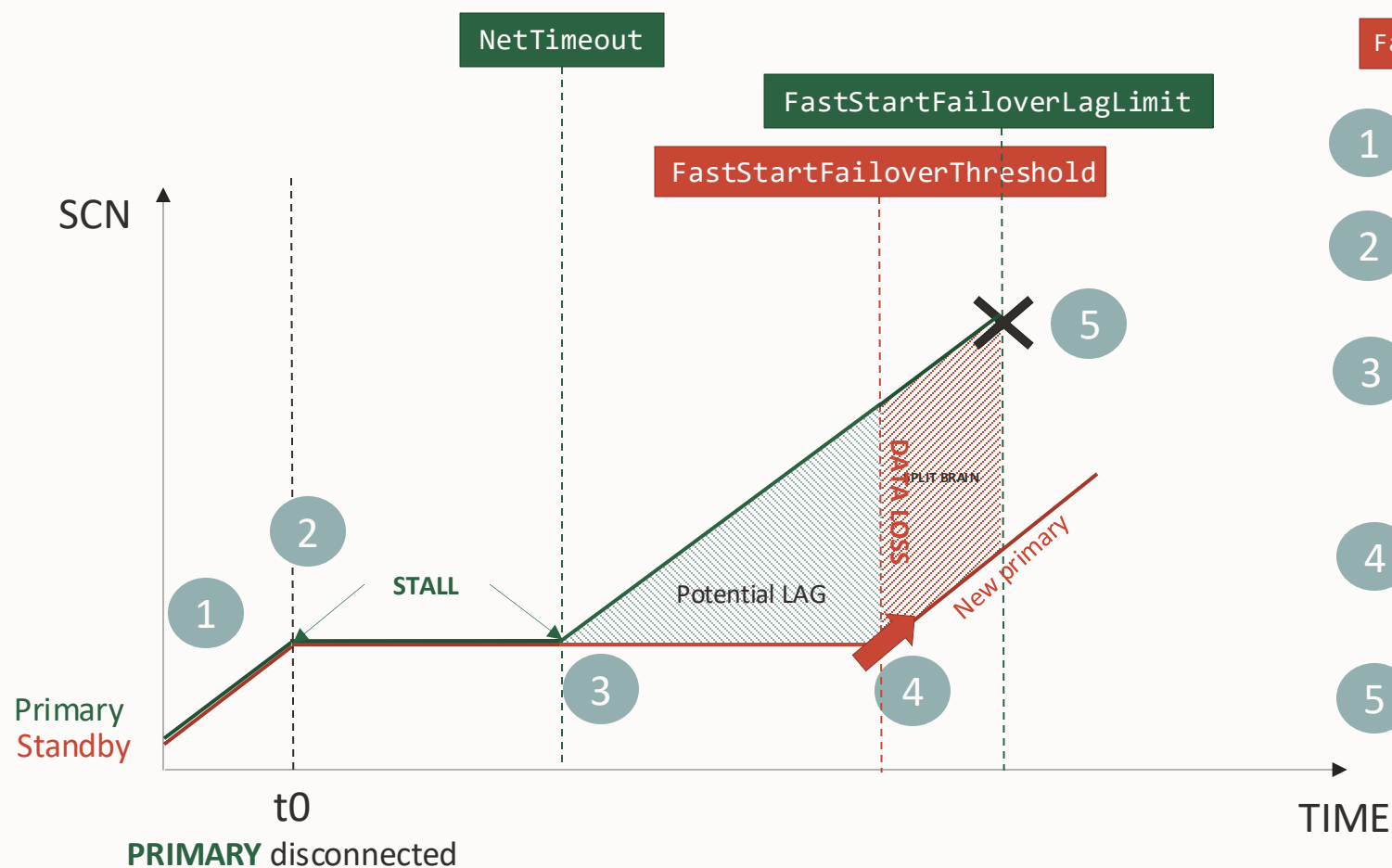
$$\text{FastStartFailoverLagLimit} \leq \text{FastStartFailoverThreshold}$$

- 1 SYNC Transport. The Standby is synched.
- 2 At  $t_0 + \text{ping time}$ , the primary cannot contact the standby. The primary stalls and keeps retrying for NetTimeout seconds. The observer starts the timer.
- 3 At NetTimeout seconds, the primary stops shipping the redo without asking permission to the observer, because FastStartFailoverLagLimit is set. The observer does not acknowledge the unsynched status.
- 4 The primary reaches FastStartFailoverLagLimit. It cannot obtain permission from the observer to continue. It stalls or shuts down depending on FastStartFailoverPmyShutdown
- 5 The observer timer reaches FastStartFailoverThreshold. Still no connection with the primary: it initiates the Failover.



# Automatic Failover with MaxAvailability

Set the FastStartFailoverLagLimit wisely to avoid split-brain conditions



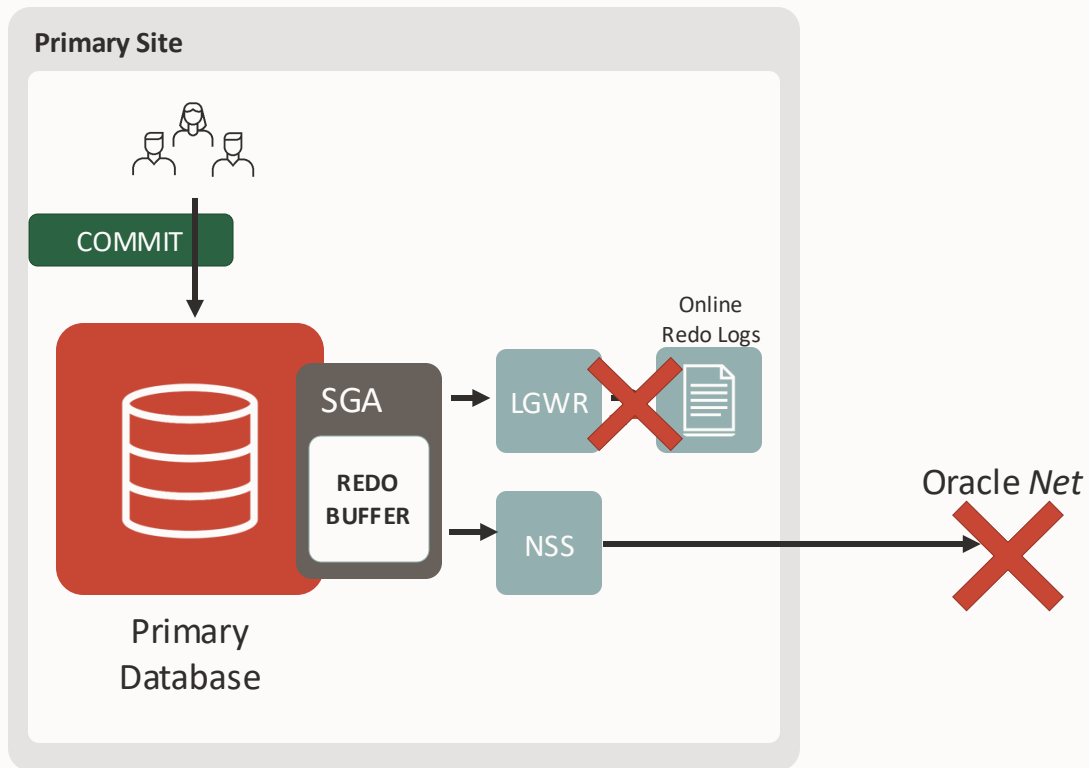
**FastStartFailoverThreshold** < **FastStartFailoverLagLimit**

- 1 SYNC Transport. The Standby is synched.
- 2 At  $t_0 + \text{ping time}$ , the primary cannot contact the standby. The primary stalls and keeps retrying for **NetTimeout** seconds. The observer starts the timer.
- 3 At **NetTimeout** seconds, the primary switches to ASYNC redo transport without requiring permission to the observer, because **FastStartFailoverLagLimit** is set. The observer does not acknowledge the unsynched status.
- 4 The observer timer reaches **FastStartFailoverThreshold**. Still no connection with the primary: it initiates the Failover.
- 5 The Primary keeps committing until it reaches **FastStartFailoverLagLimit**, then stalls or shuts down. A Split-Brain condition may occur depending on timings and parameter values.

# Fast-Start Failover: Maximum Protection

# Max Protection without Fast-Start Failover

Data Guard **SYNC** redo transport

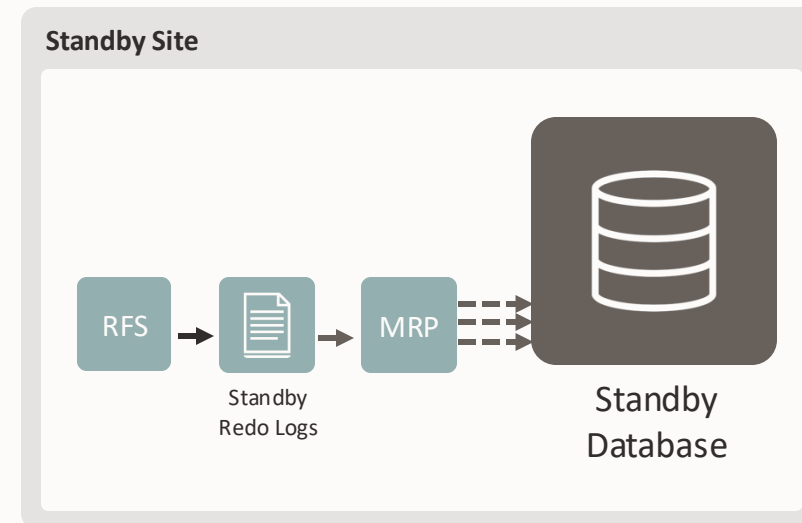


## If the standby is not reachable:

- The primary stalls waiting for the SYNC destination (no commits possible)
- Even after NetTimeout, the commits cannot happen

## If the primary is not reachable:

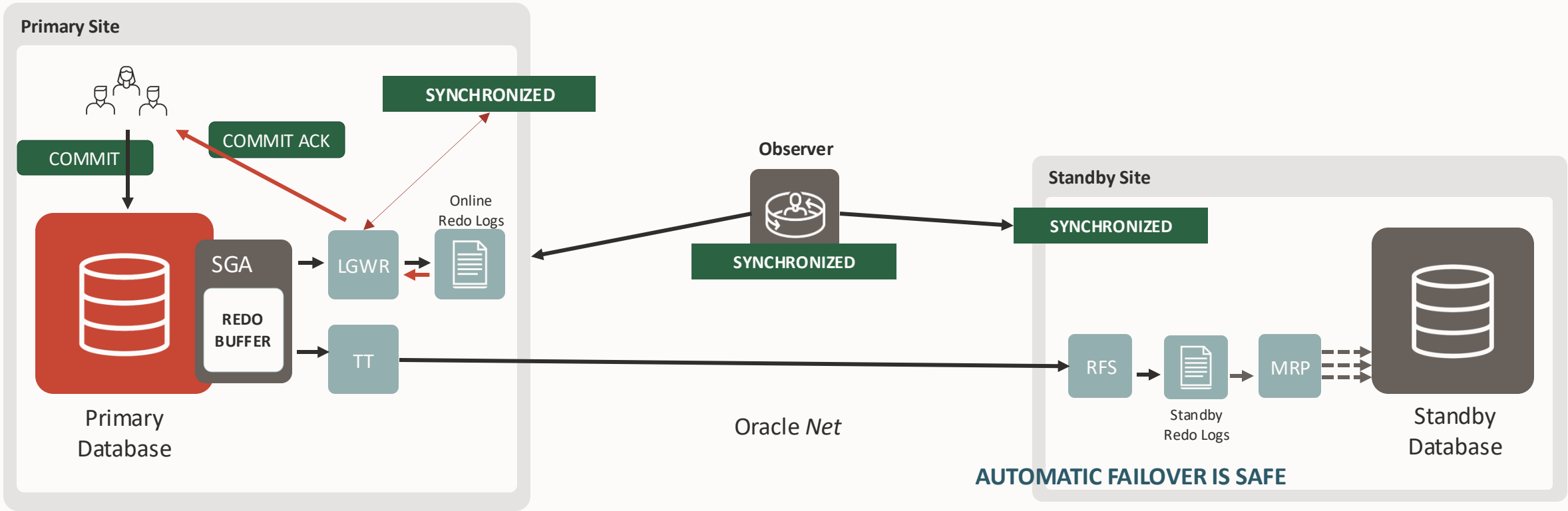
- Failovers, even manual, are safe if no other standbys are protecting the primary



# Max Protection with Fast-Start Failover

The primary never commits without the observer quorum

In a normal situation, the status is **SYNCHRONIZED**

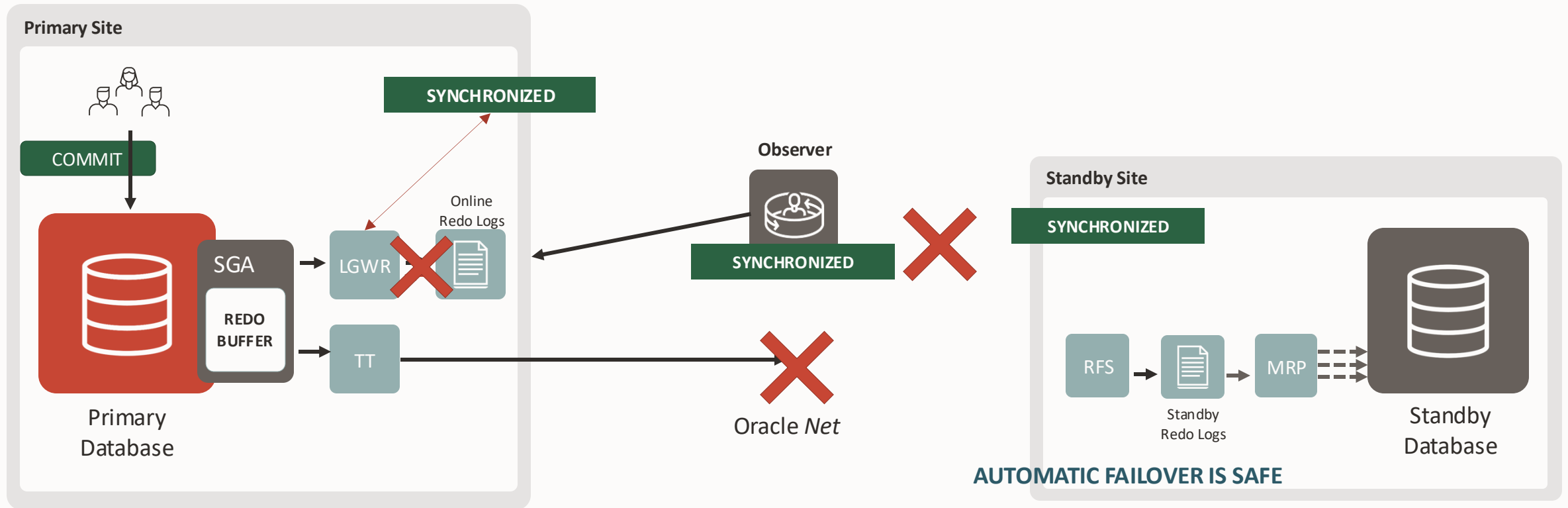


# Max Protection with Fast-Start Failover

The primary never commits without the observer quorum

## If the standby is not reachable:

- The primary waits for the SYNC destination (no commits possible) but keeps the status synchronized

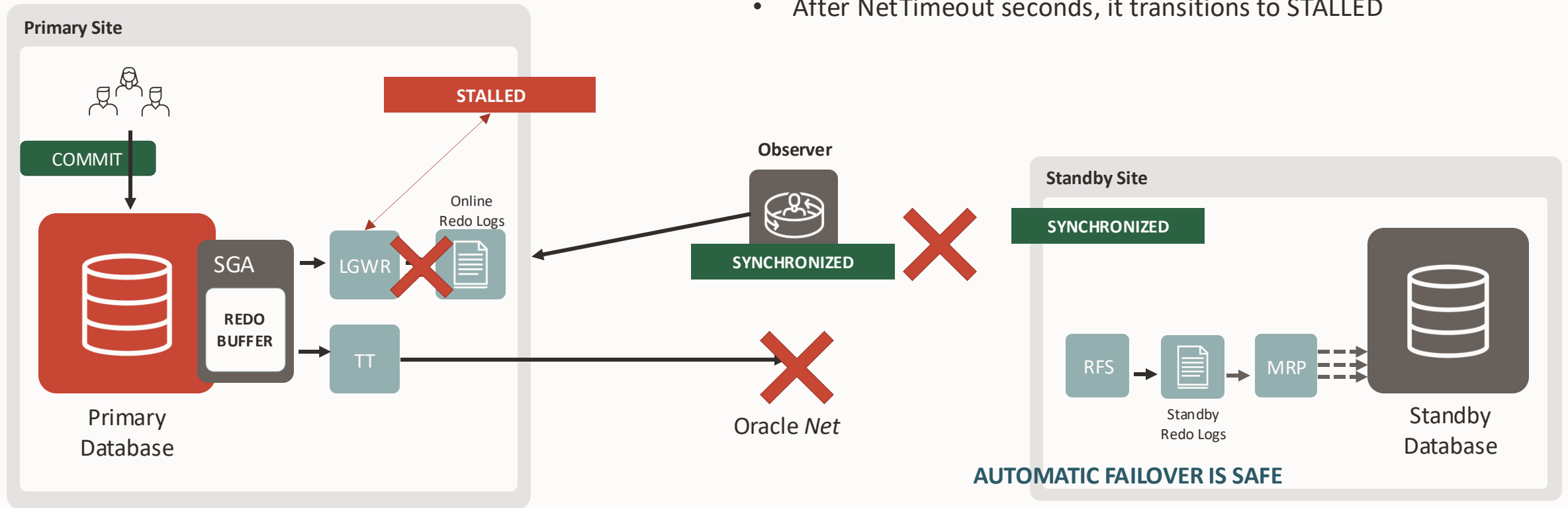


# Max Protection with Fast-Start Failover

The primary never commits without the observer quorum

## If the standby is not reachable:

- The primary waits for the SYNC destination (no commits possible) but keeps the status synchronized
- After NetTimeout seconds, it transitions to STALLED



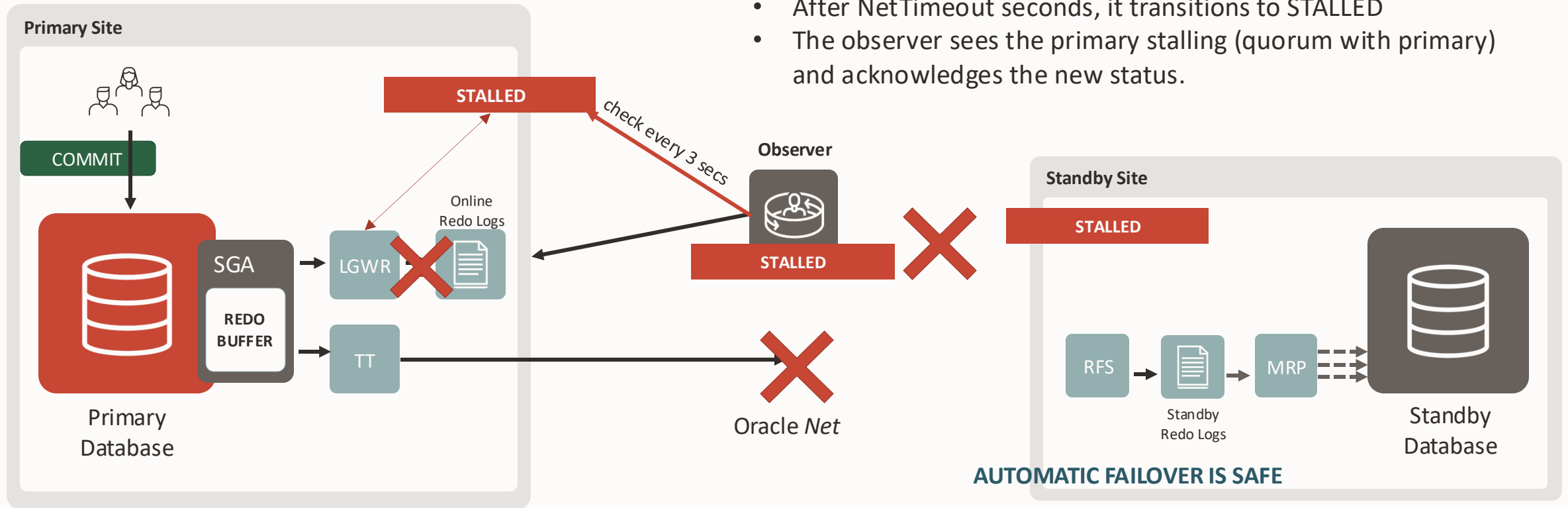


# Max Protection with Fast-Start Failover

The primary never commits without the observer quorum

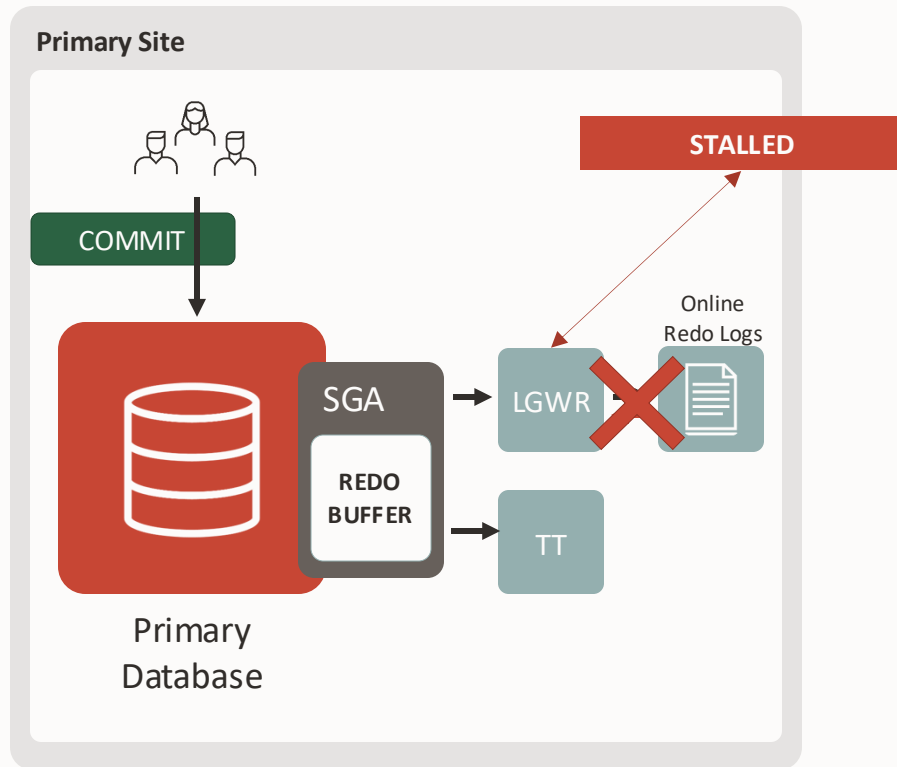
## If the standby is not reachable:

- The primary waits for the SYNC destination (no commits possible) but keeps the status synchronized
- After NetTimeout seconds, it transitions to STALLED
- The observer sees the primary stalling (quorum with primary) and acknowledges the new status.



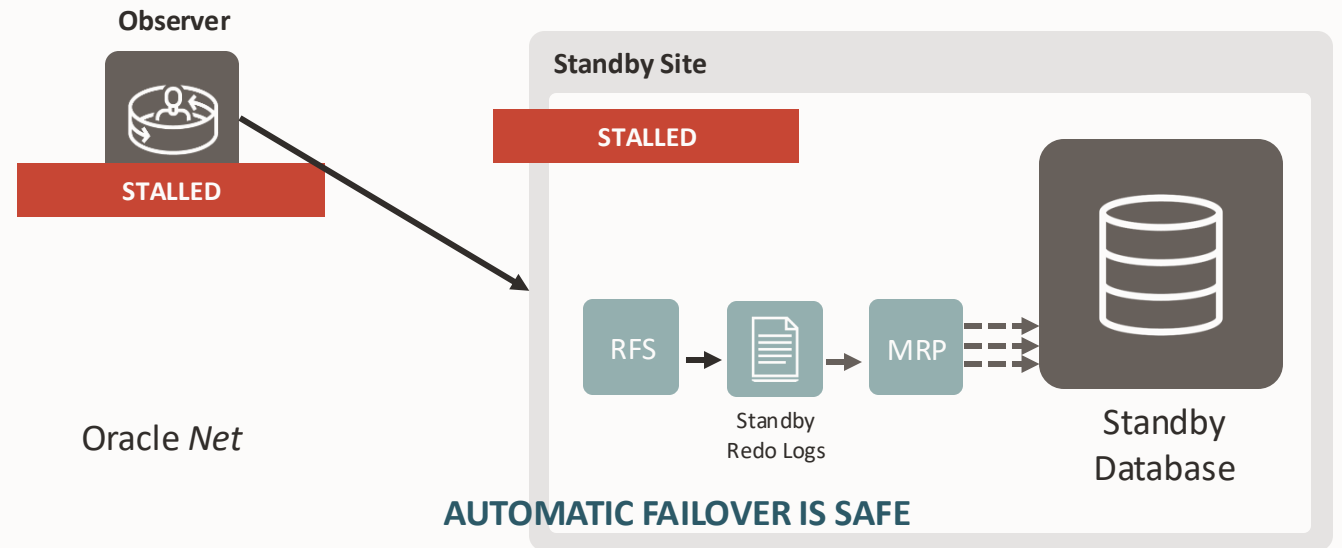
# Max Protection with Fast-Start Failover

The primary never commits without the observer quorum



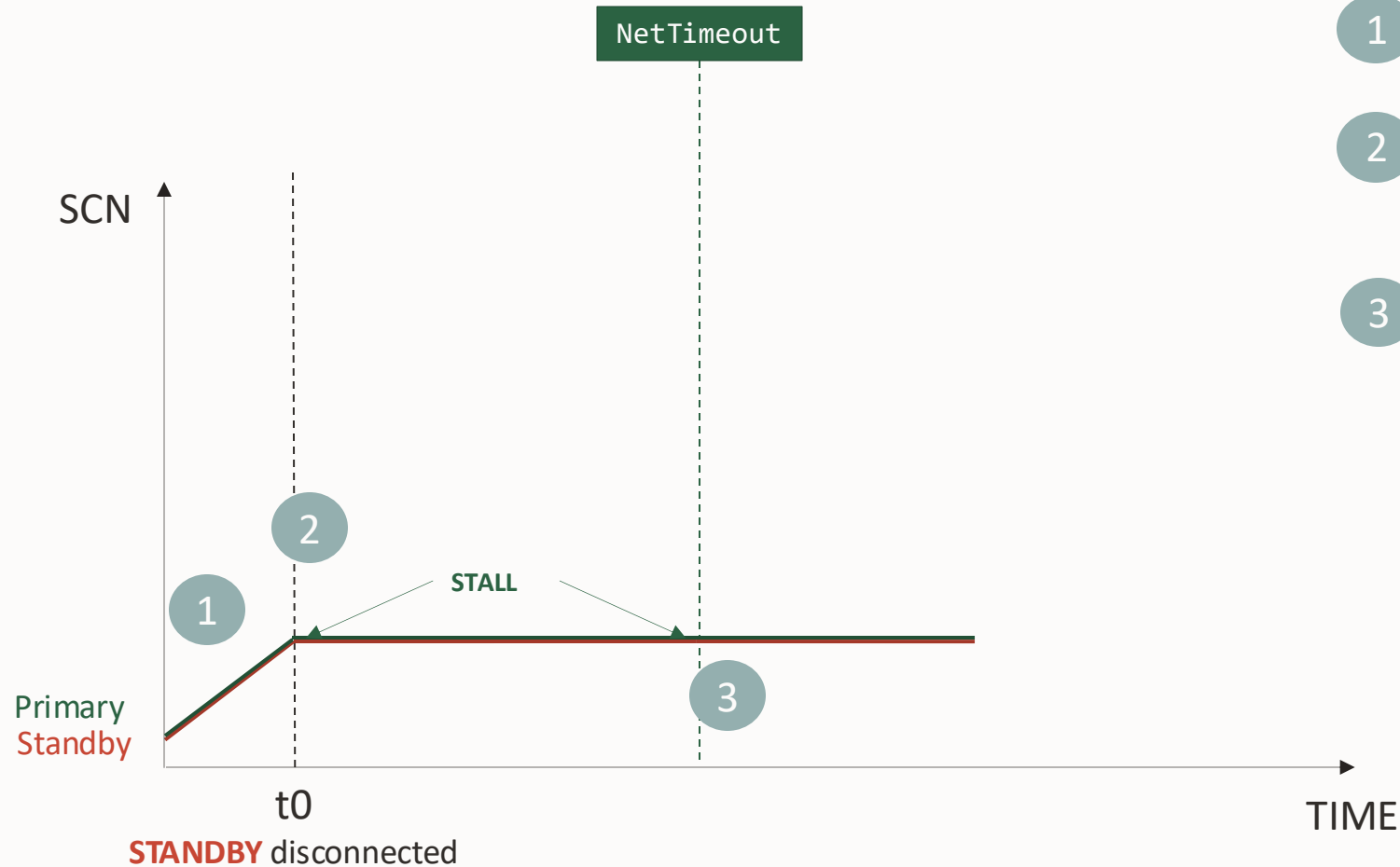
If the primary is not reachable:

- The primary never committed without a SYNC standby
- The automatic failover is always safe, but the new primary will require another standby to keep the current protection level



# Automatic Failover with MaxProtection

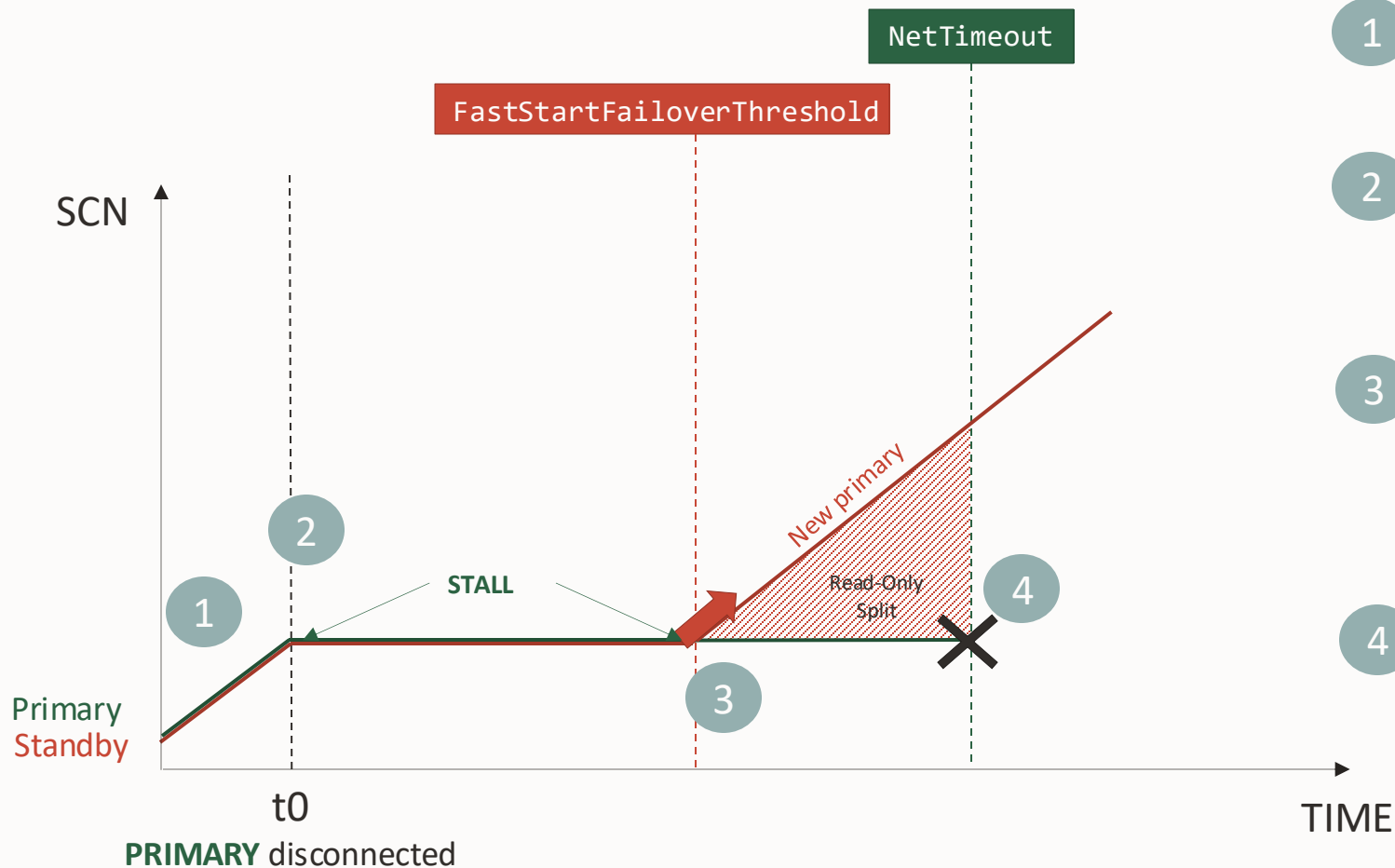
Zero Data Loss in any condition



- 1 SYNC Transport. The Standby is synched.  
Status: SYNCHRONIZED
- 2 At  $t_0$  + ping time, the primary cannot contact the standby and keeps retrying for NetTimeout Seconds.  
Status: STALLED
- 3 After NetTimeout, the primary still has connectivity with the observer. Knowing that a failover did not occur, it keeps trying forever.  
Status: STALLED

# Automatic Failover with MaxProtection

Zero Data Loss in any condition



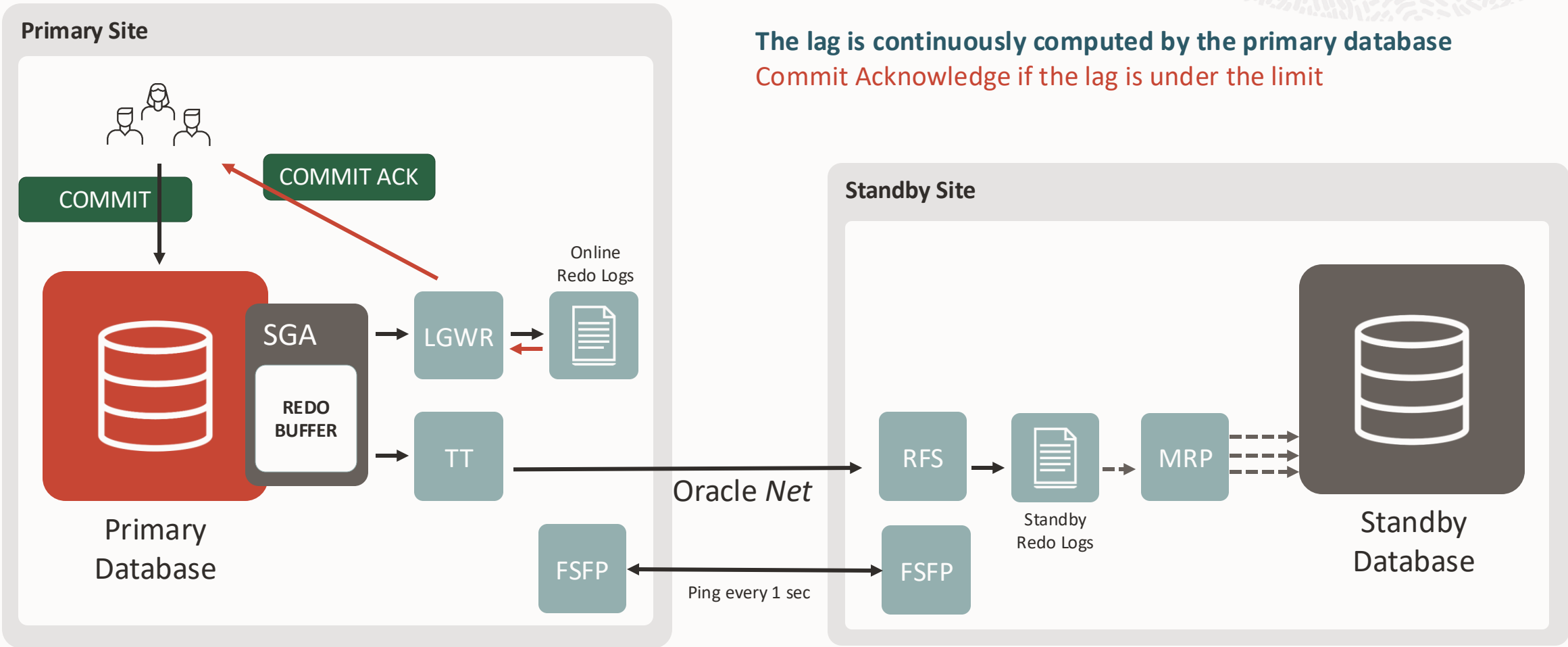
- 1 SYNC Transport. The Standby is synced.  
Status: SYNCHRONIZED
- 2 At  $t_0 + \text{ping time}$ , the observer cannot contact the primary and starts a timer for  $\text{FastStartFailoverThreshold}$  seconds. The Primary stalls and keeps retrying for  $\text{NetTimeout}$  Seconds.  
Status: STALLED
- 3 The observer timer reaches  $\text{FastStartFailoverThreshold}$ . Still no connection with the primary: it initiates the Failover. If  $\text{NetTimeout}$  is higher than the threshold, the Primary is reachable but not committing (read-only split).  
Former Primary Status: STALLED  
New Primary Status: REINSTATE\_REQUIRED
- 4 The primary keeps stalling or shuts down after  $\text{NetTimeout}$  depending on  $\text{FastStartFailoverPmyShutdown}$ . A lower  $\text{NetTimeout}$  reduces the potential of read-only split.  
Former Primary Status: STALLED or REINSTATE\_REQUIRED  
New Primary Status: REINSTATE\_REQUIRED

# Fast-Start Failover: Oracle Data Guard Observer

# How Fast-Start Failover limits data loss

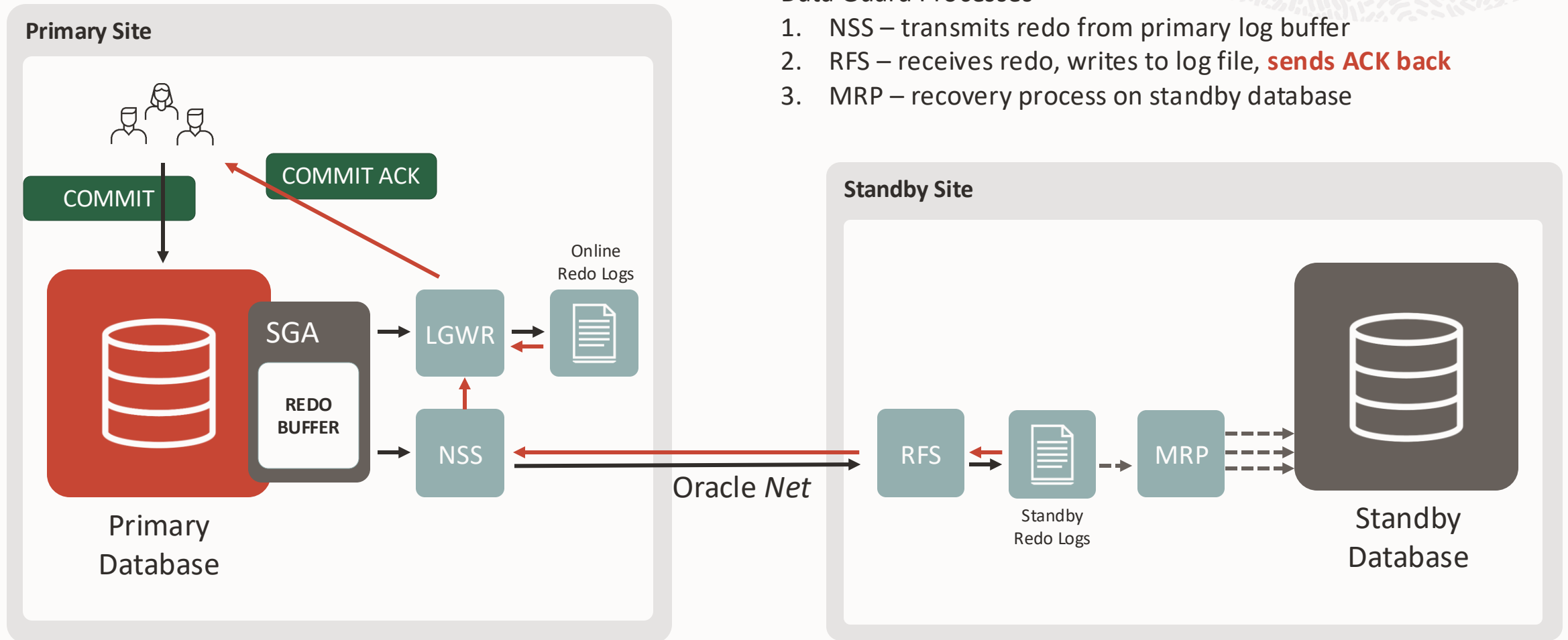
Data Guard **ASYNC** Process Architecture (Possible Data Loss)

The lag is continuously computed by the primary database  
Commit Acknowledge if the lag is under the limit



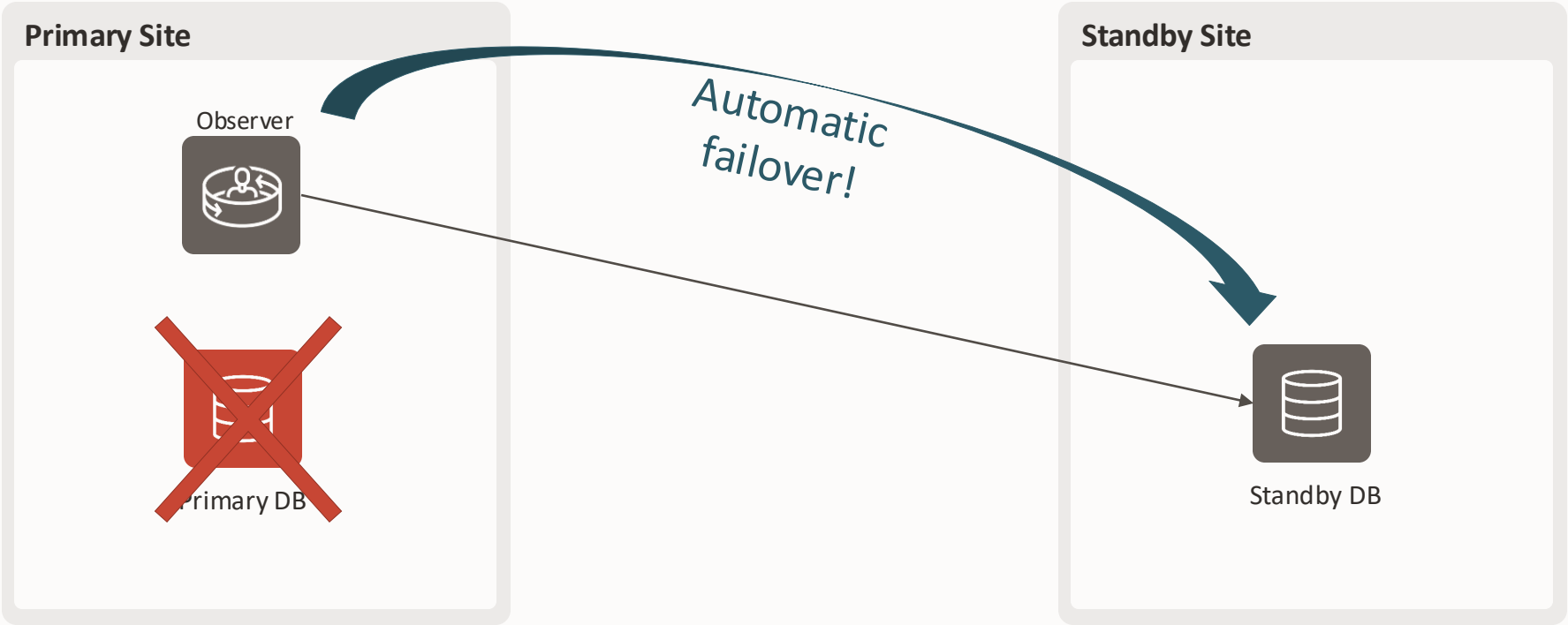
# Data Guard Transport for Zero Data Loss

## Data Guard SYNC Process Architecture



# Oracle Data Guard Observer Placement

Observer at the primary site



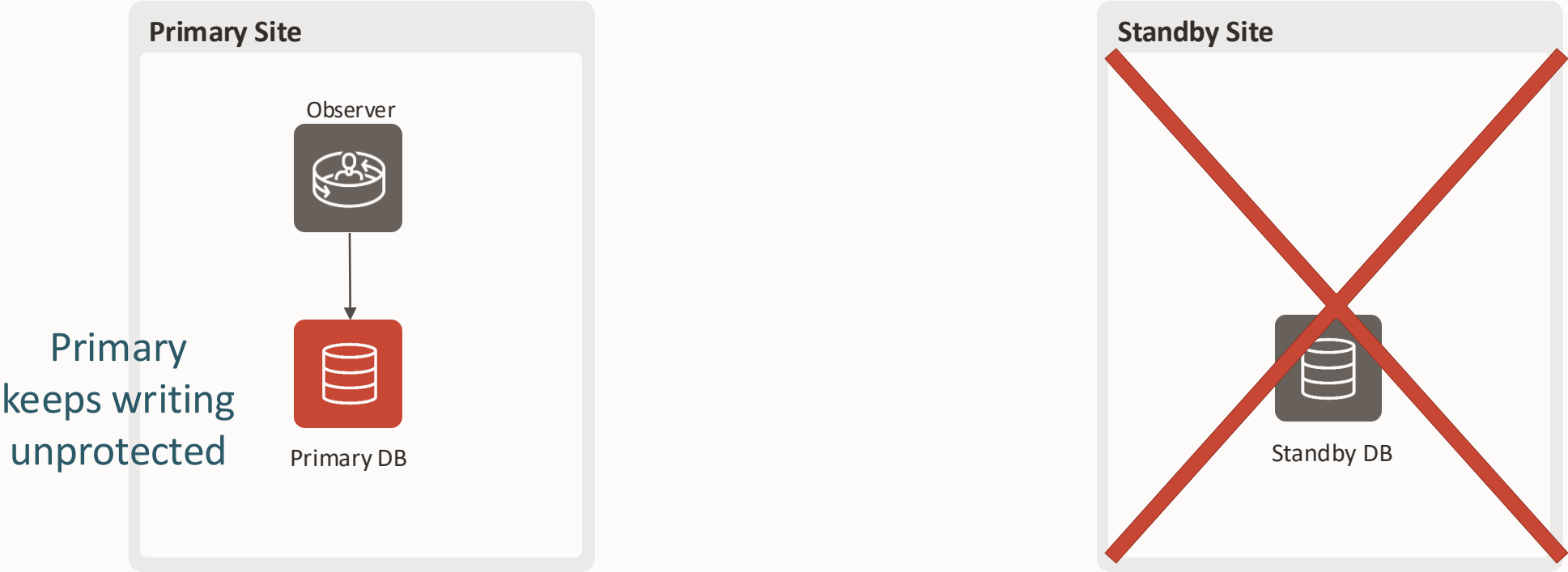
Failure:	Primary DB	Standby DB	Network	Primary Site	Standby Site	Observer
	✓	✓	✓	✗	✓	✓





# Oracle Data Guard Observer Placement

Observer at the primary site

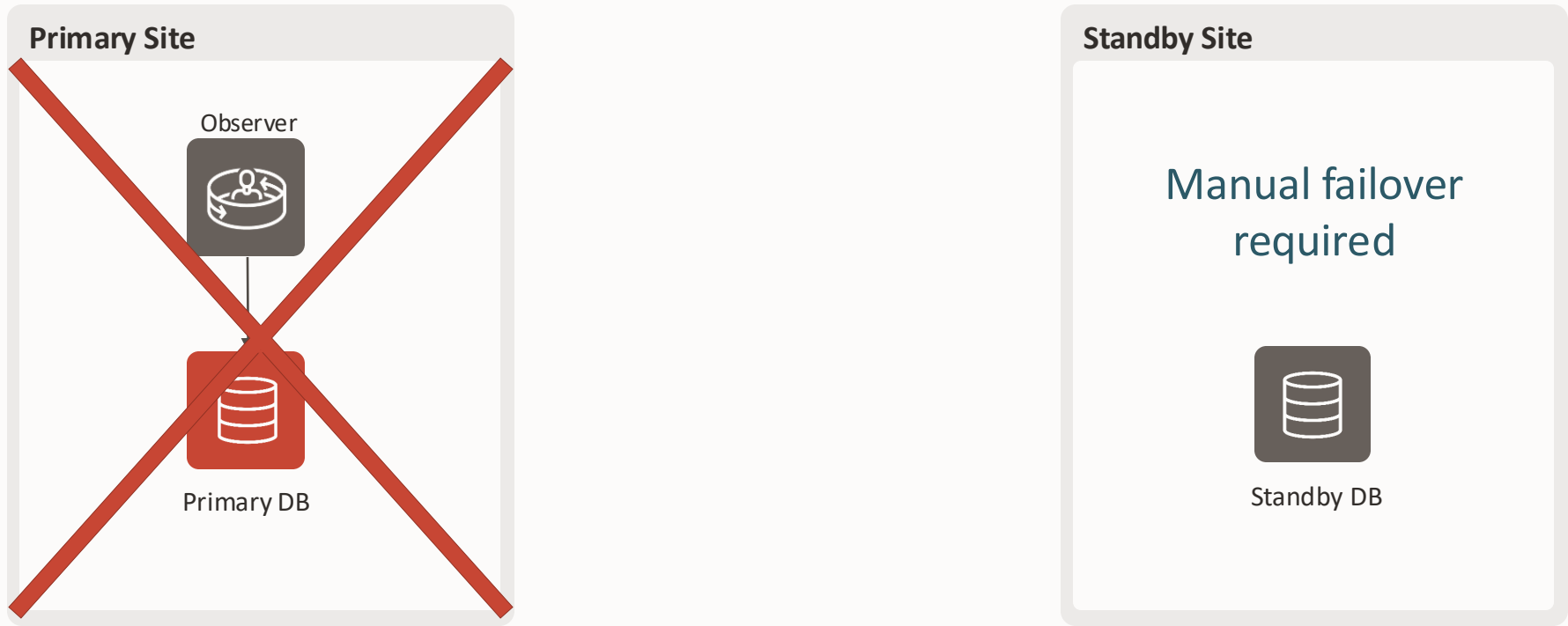


Failure:	Primary DB	Standby DB	Network	Primary Site	Standby Site	Observer
	✓	✓	✓	✗	✓	✓



# Oracle Data Guard Observer Placement

Observer at the primary site

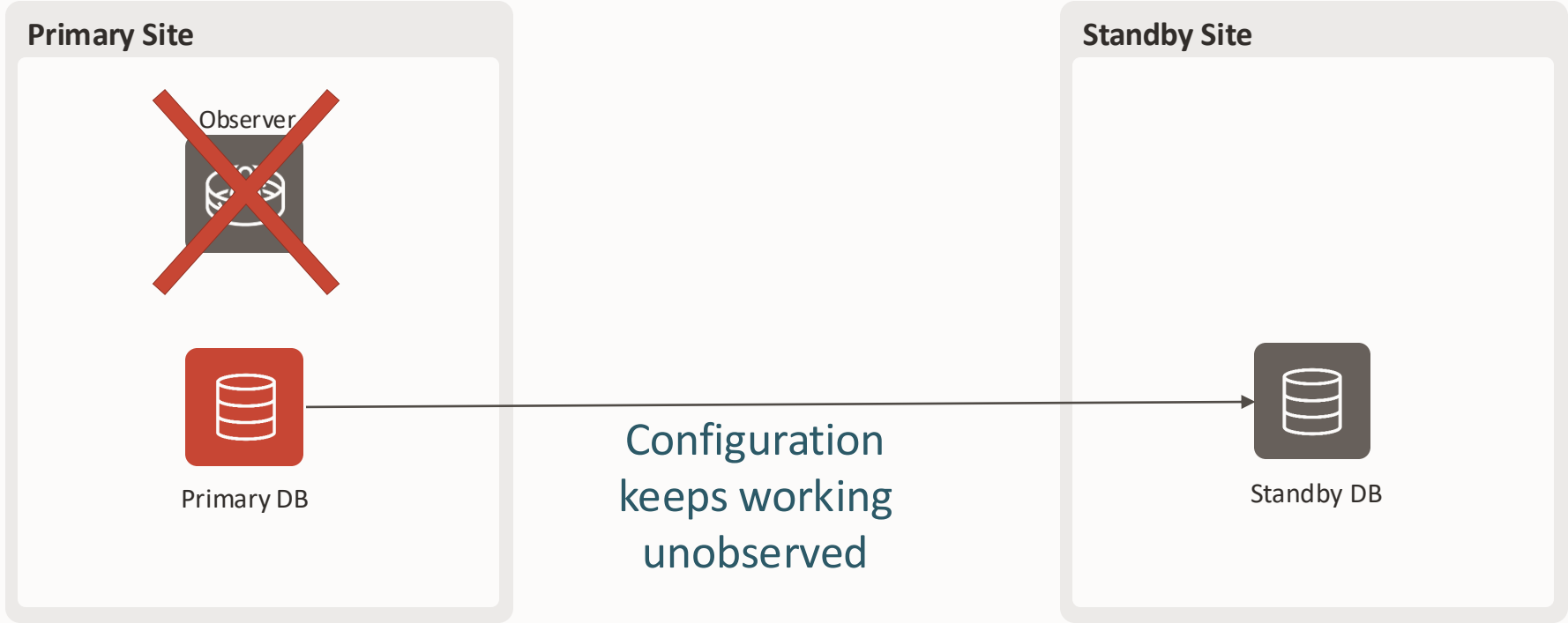


Failure:	Primary DB	Standby DB	Network	Primary Site	Standby Site	Observer
	✓	✓	✓	✗	✓	✓



# Oracle Data Guard Observer Placement

Observer at the primary site

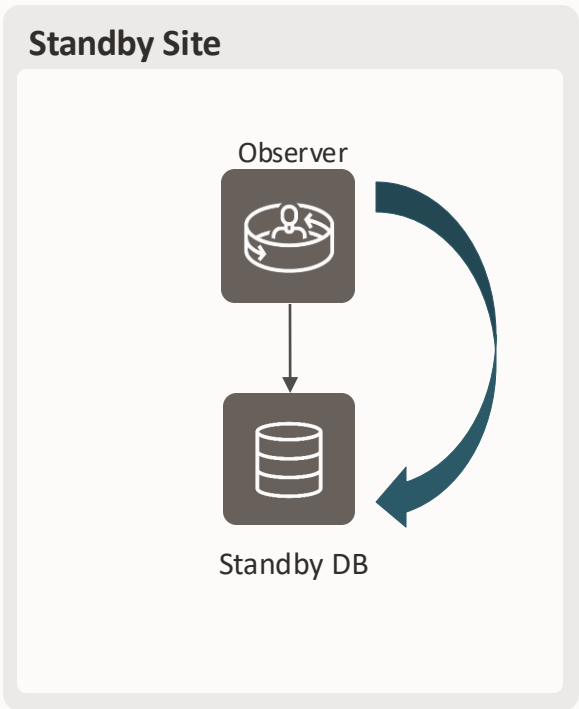
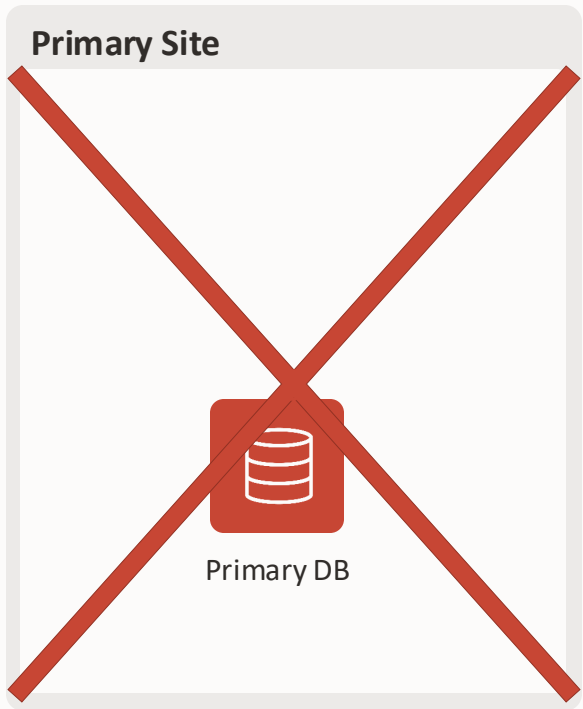


Failure:	Primary DB	Standby DB	Network	Primary Site	Standby Site	Observer
	✓	✓	✓	✗	✓	✓



# Oracle Data Guard Observer Placement

Observer at the standby site



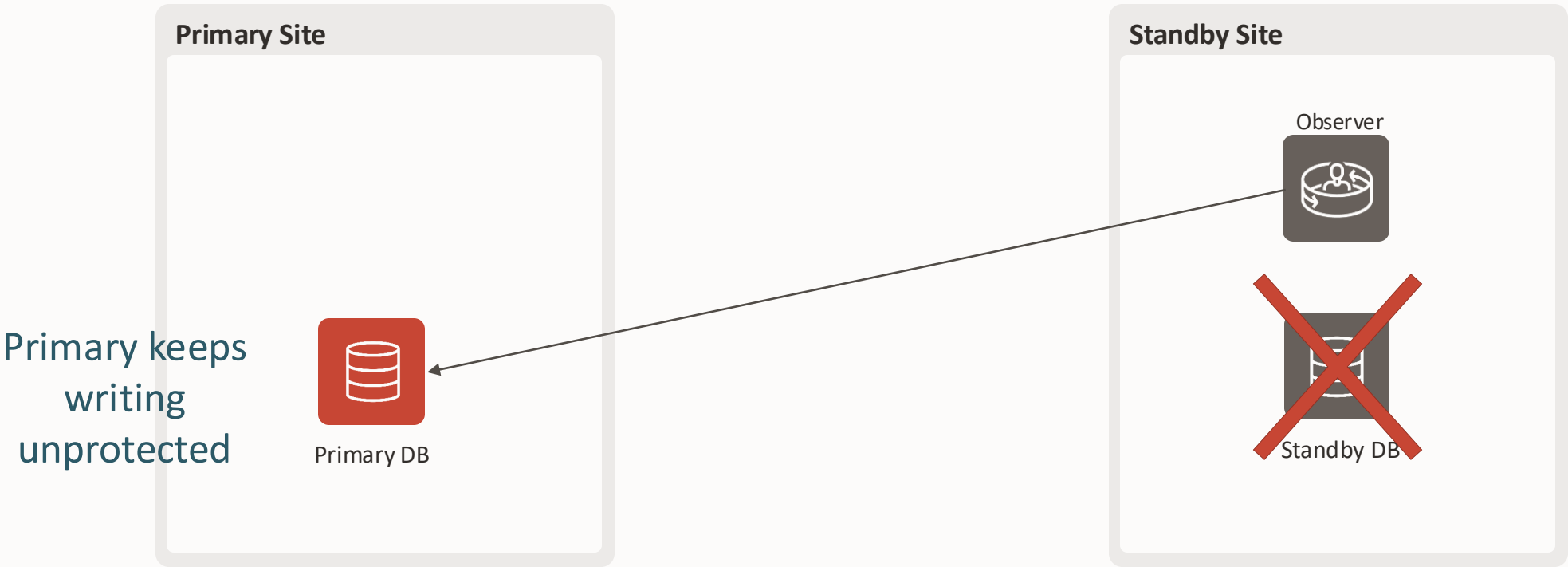
Automatic failover!

Failure:	Primary DB	Standby DB	Network	Primary Site	Standby Site	Observer
	✓	✓	✗	✓	✗	✓



# Oracle Data Guard Observer Placement

Observer at the standby site

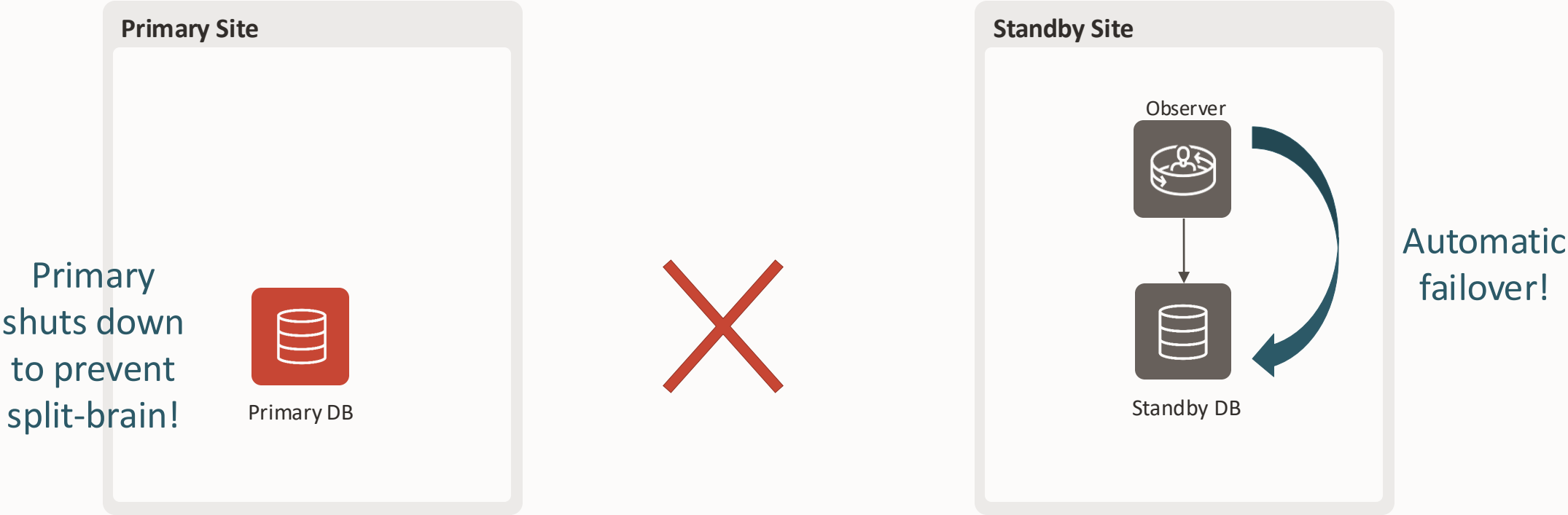


Failure:	Primary DB	Standby DB	Network	Primary Site	Standby Site	Observer
	✓	✓	✗	✓	✗	✓



# Oracle Data Guard Observer Placement

Observer at the standby site

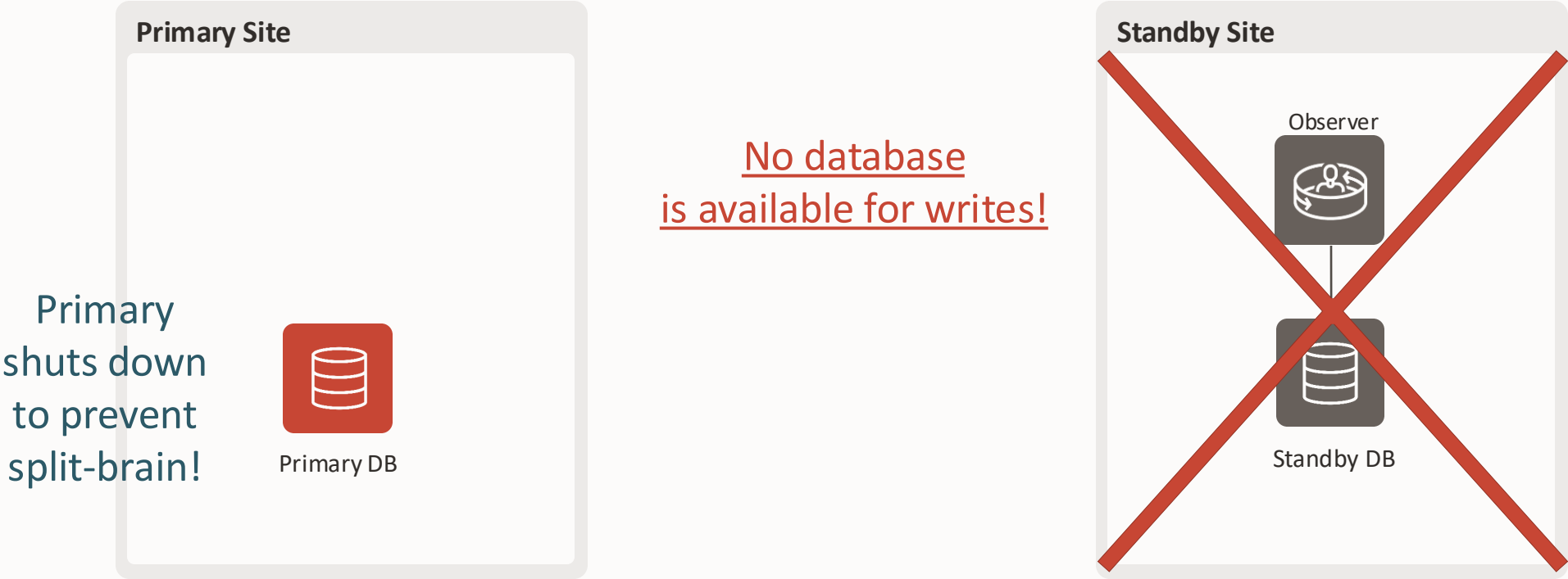


Failure:	Primary DB	Standby DB	Network	Primary Site	Standby Site	Observer
	✓	✓	✗	✓	✗	✓



# Oracle Data Guard Observer Placement

Observer at the standby site

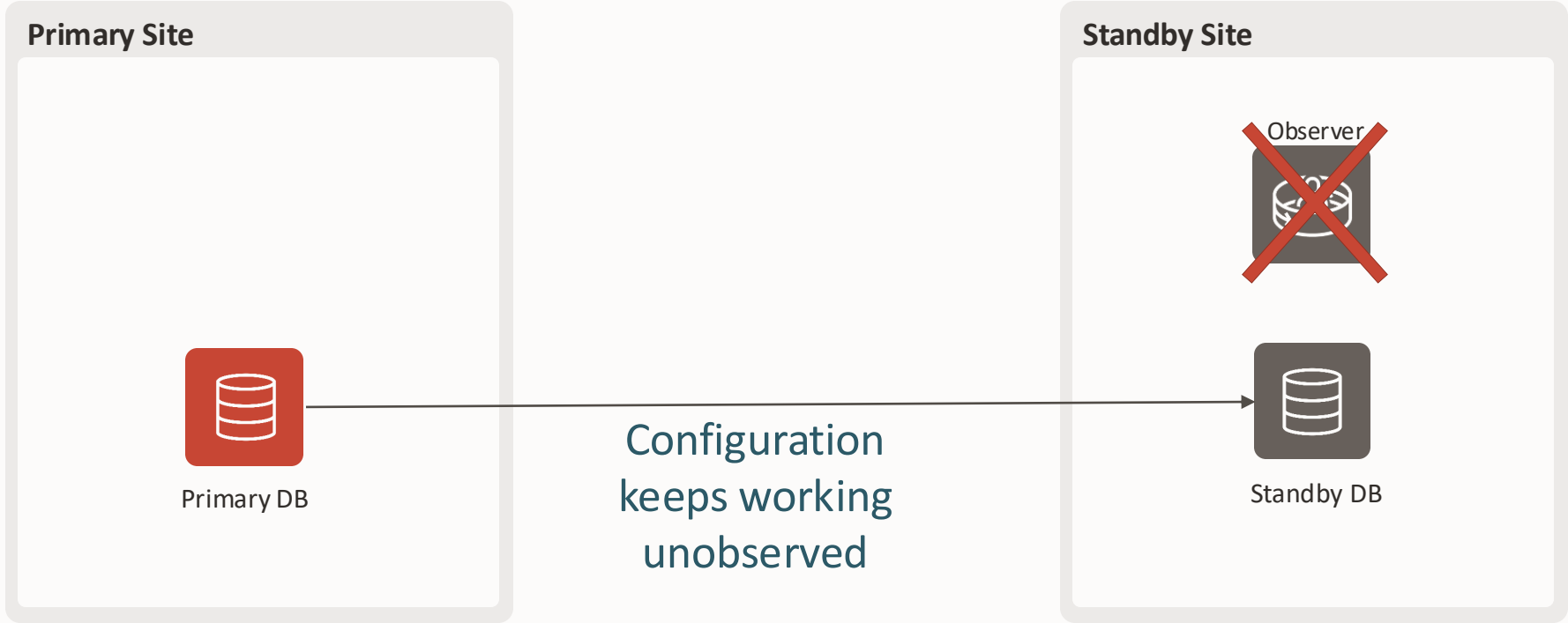


Failure:	Primary DB	Standby DB	Network	Primary Site	Standby Site	Observer
	✓	✓	✗	✓	✗	✓



# Oracle Data Guard Observer Placement

Observer at the standby site



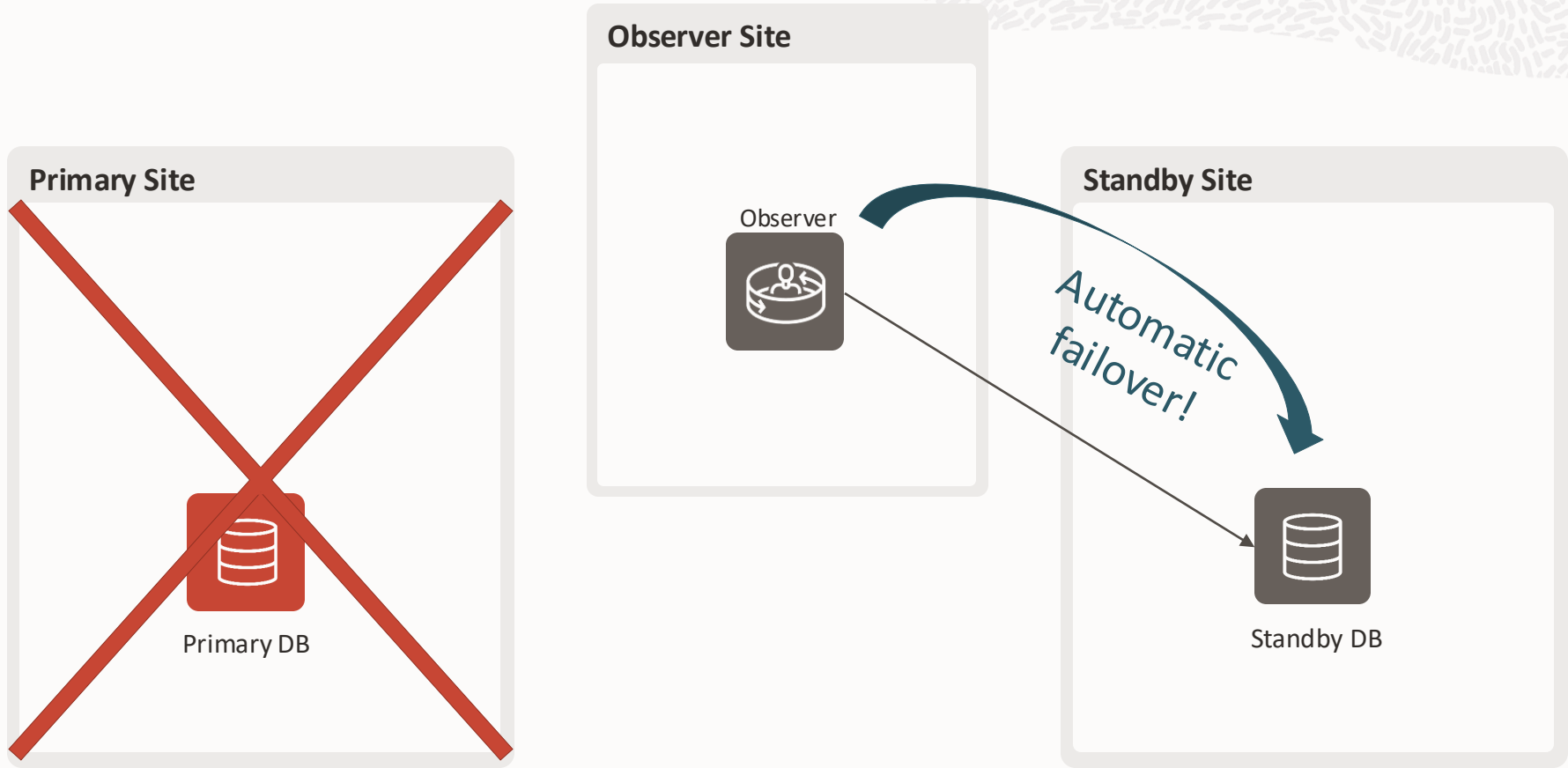
Failure:	Primary DB	Standby DB	Network	Primary Site	Standby Site	Observer
	✓	✓	✗	✓	✗	✓





# Oracle Data Guard Observer Placement

Observer at an external size

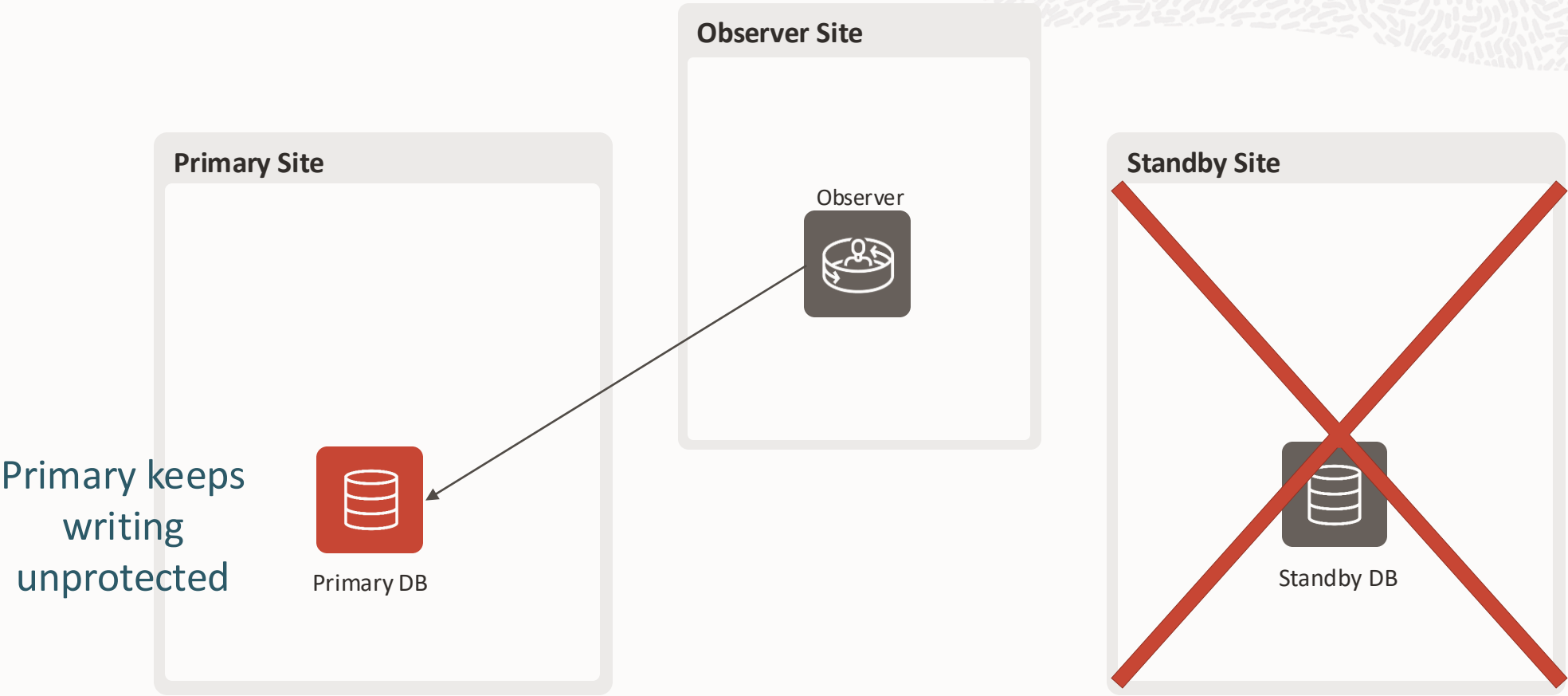


Failure:	Primary DB	Standby DB	Network	Primary Site	Standby Site	Observer
	✓	✓	✓	✓	✓	✓



# Oracle Data Guard Observer Placement

Observer at an external size

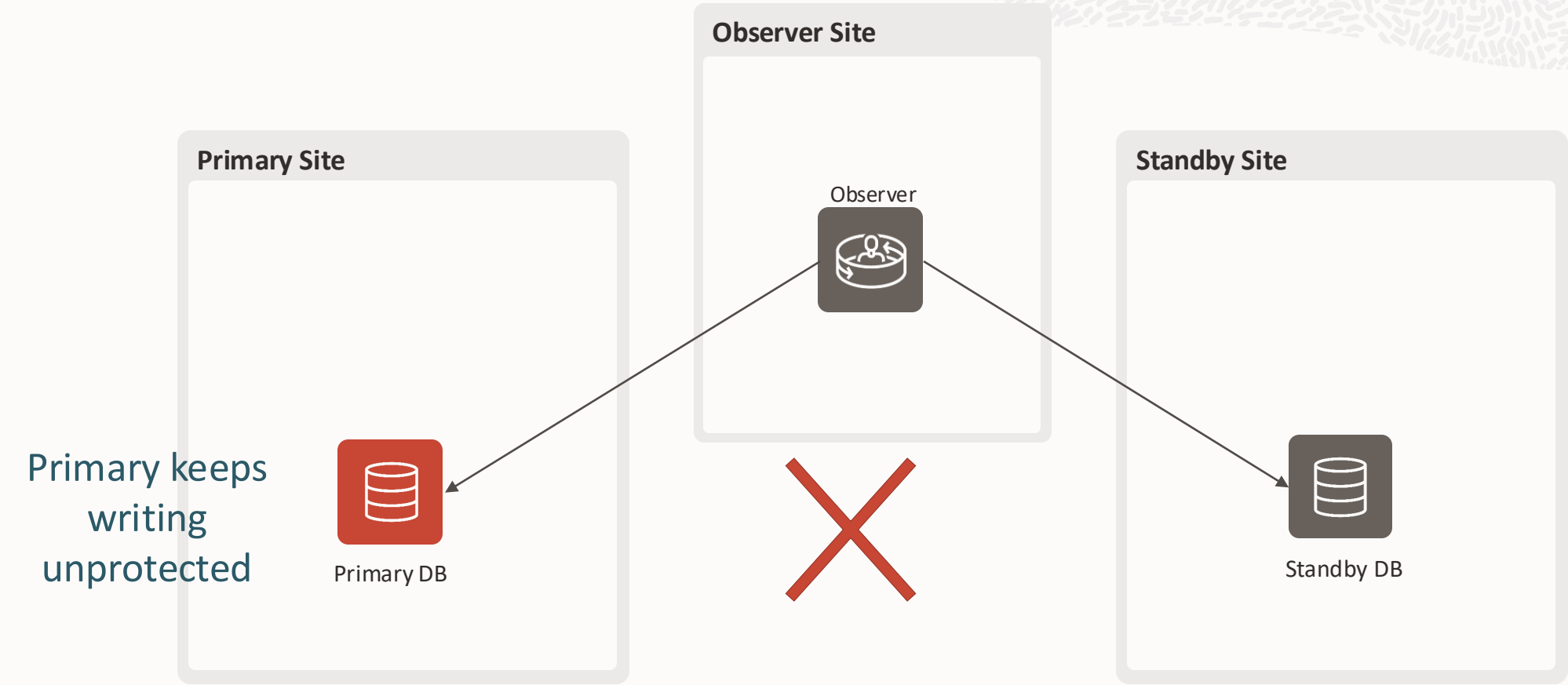


Failure:	Primary DB	Standby DB	Network	Primary Site	Standby Site	Observer
	✓	✓	✓	✓	✓	✓



# Oracle Data Guard Observer Placement

Observer at an external size

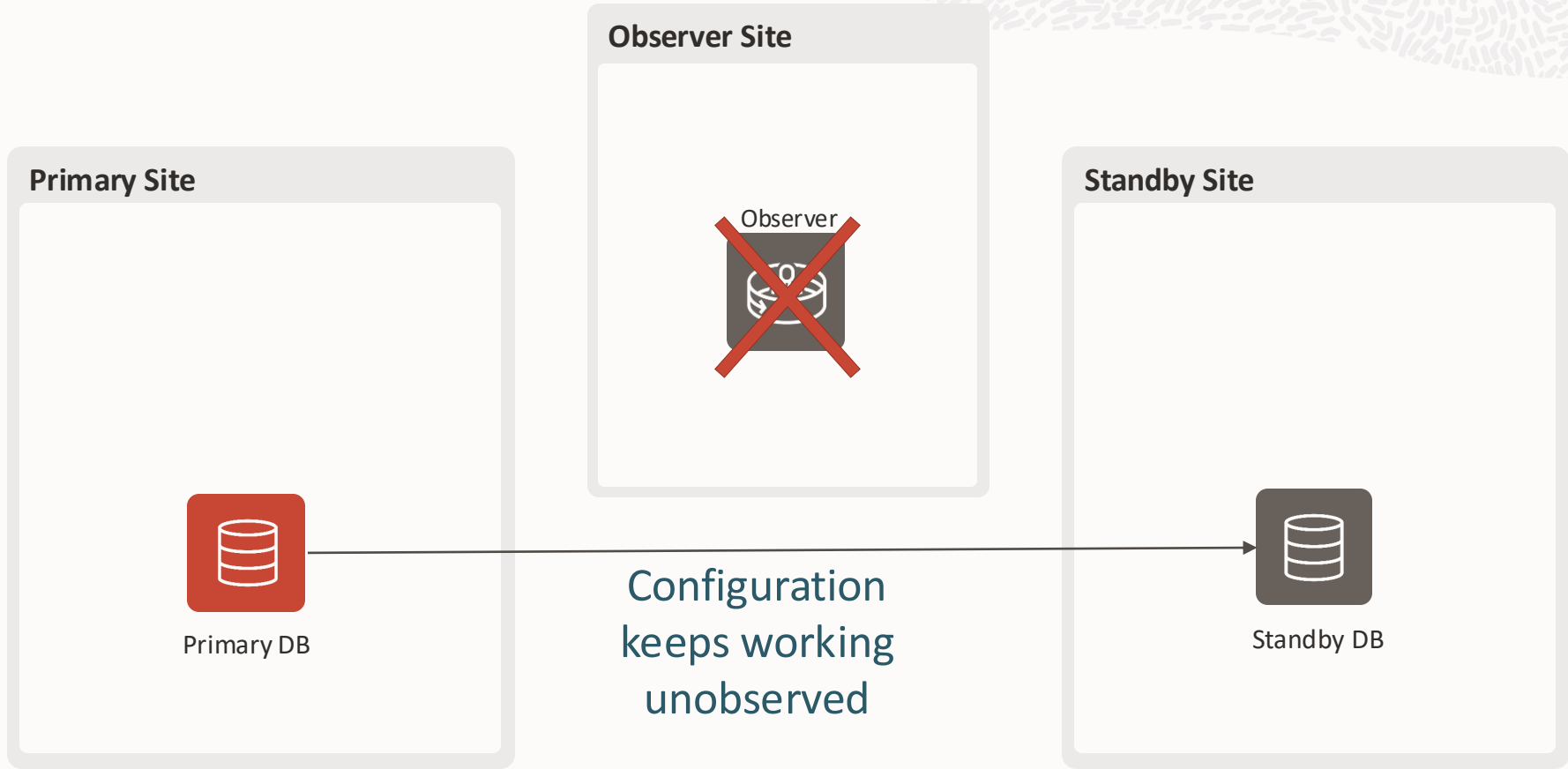


Failure:	Primary DB	Standby DB	Network	Primary Site	Standby Site	Observer
	✓	✓	✓	✓	✓	✓



# Oracle Data Guard Observer Placement

Observer at an external size

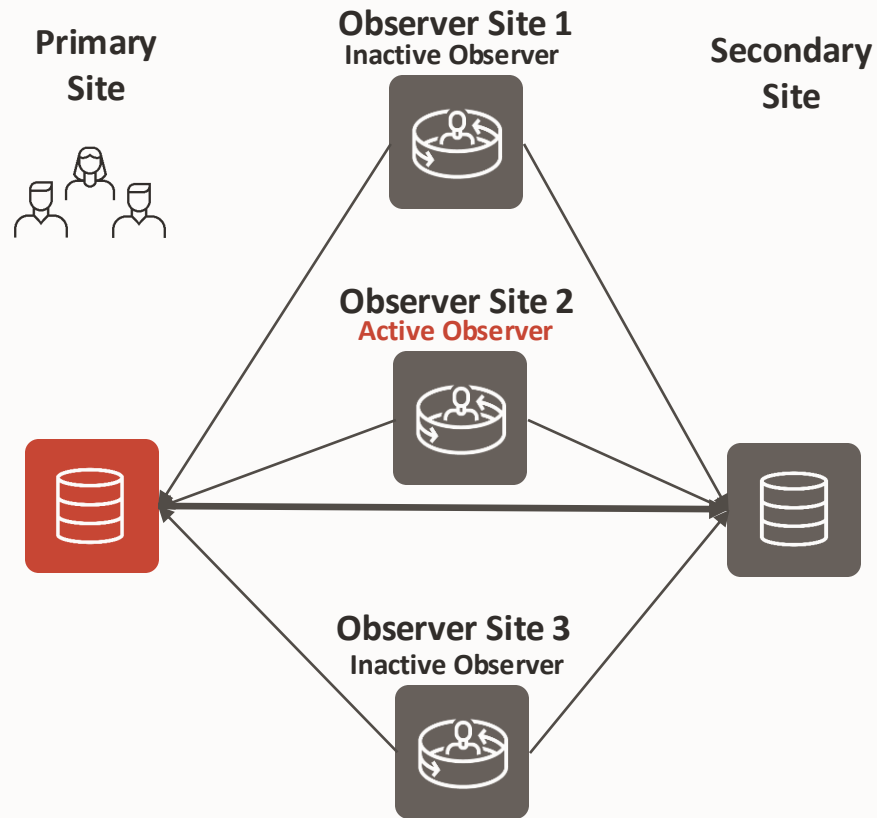


Failure:	Primary DB	Standby DB	Network	Primary Site	Standby Site	Observer
	✓	✓	✓	✓	✓	✓



# Oracle Data Guard Observer High Availability

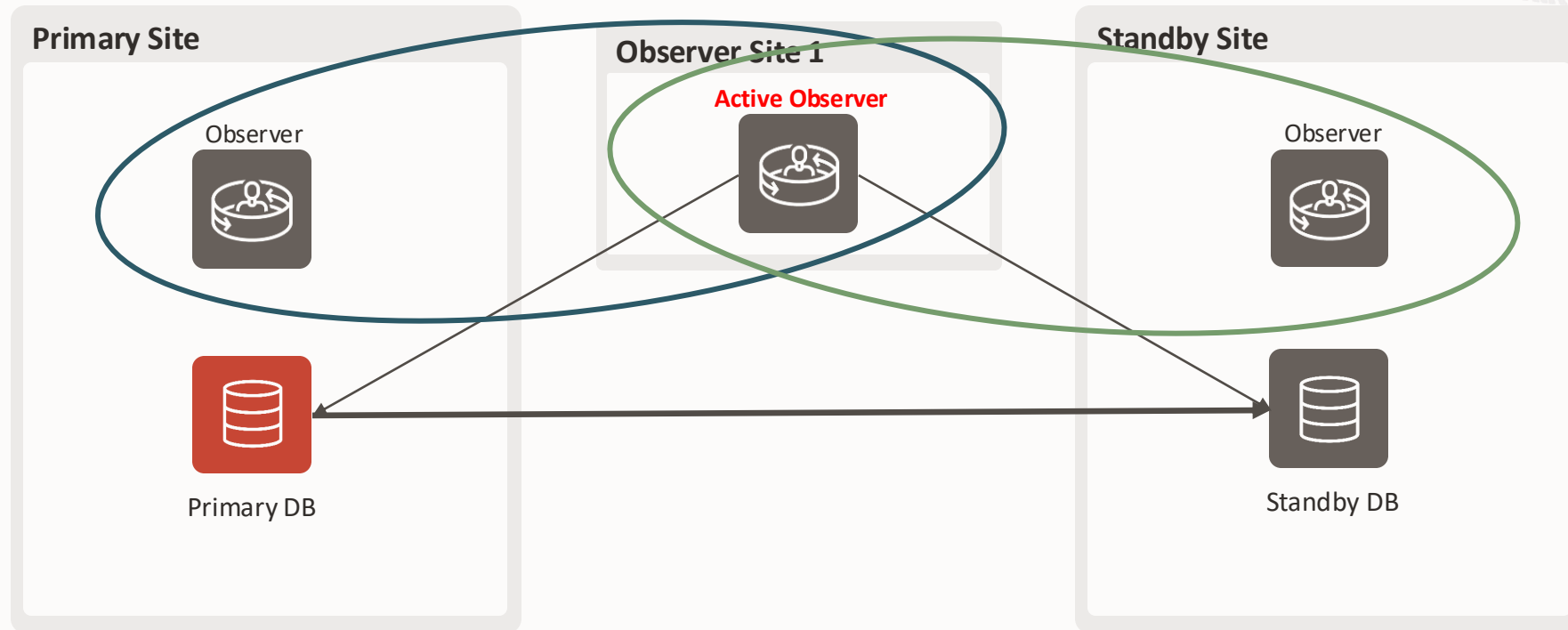
Up to four observers configured (one active at a time)



- **Optimal: 2 or 3 different Regions/Data Centers/Ads**
  - Ensure there are no SPOFs (network, power...)
- If one observer fails, another is promoted

# Oracle Data Guard Observer High Availability

Tolerate observer site failure

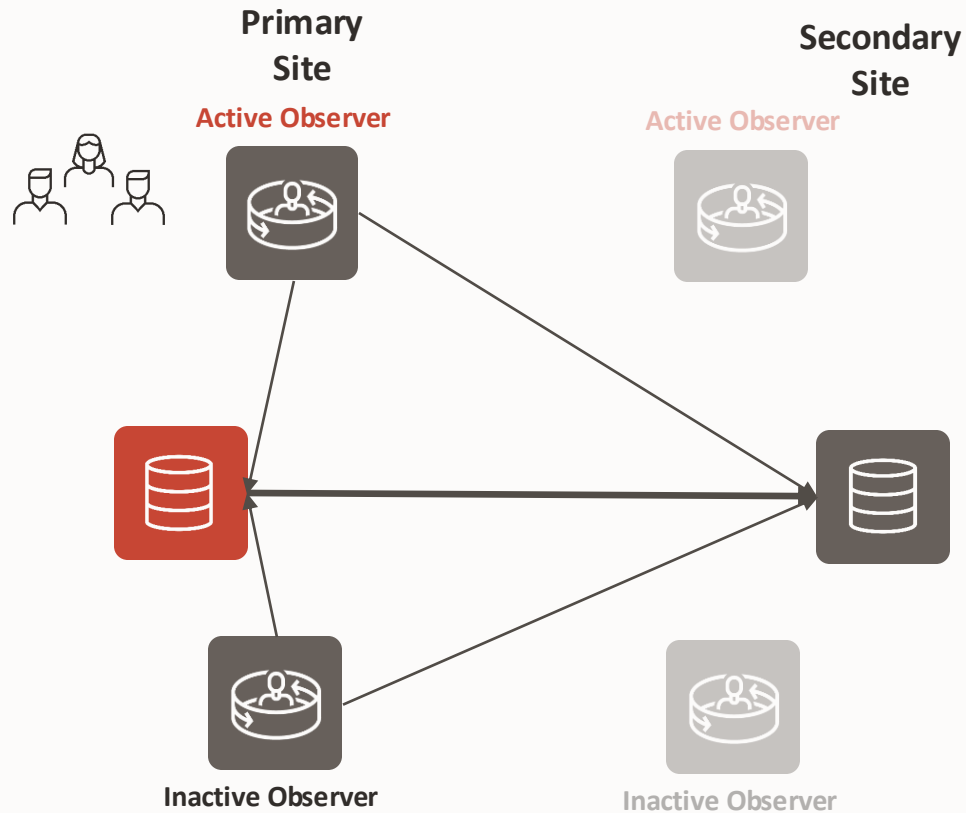


```
edit database db_site1 set property PreferredObserverHosts='obs_ext:1,obs_site1:2';
```

```
edit database db_site2 set property PreferredObserverHosts='obs_ext:1,obs_site2:2';
```

# Oracle Data Guard Observer High Availability

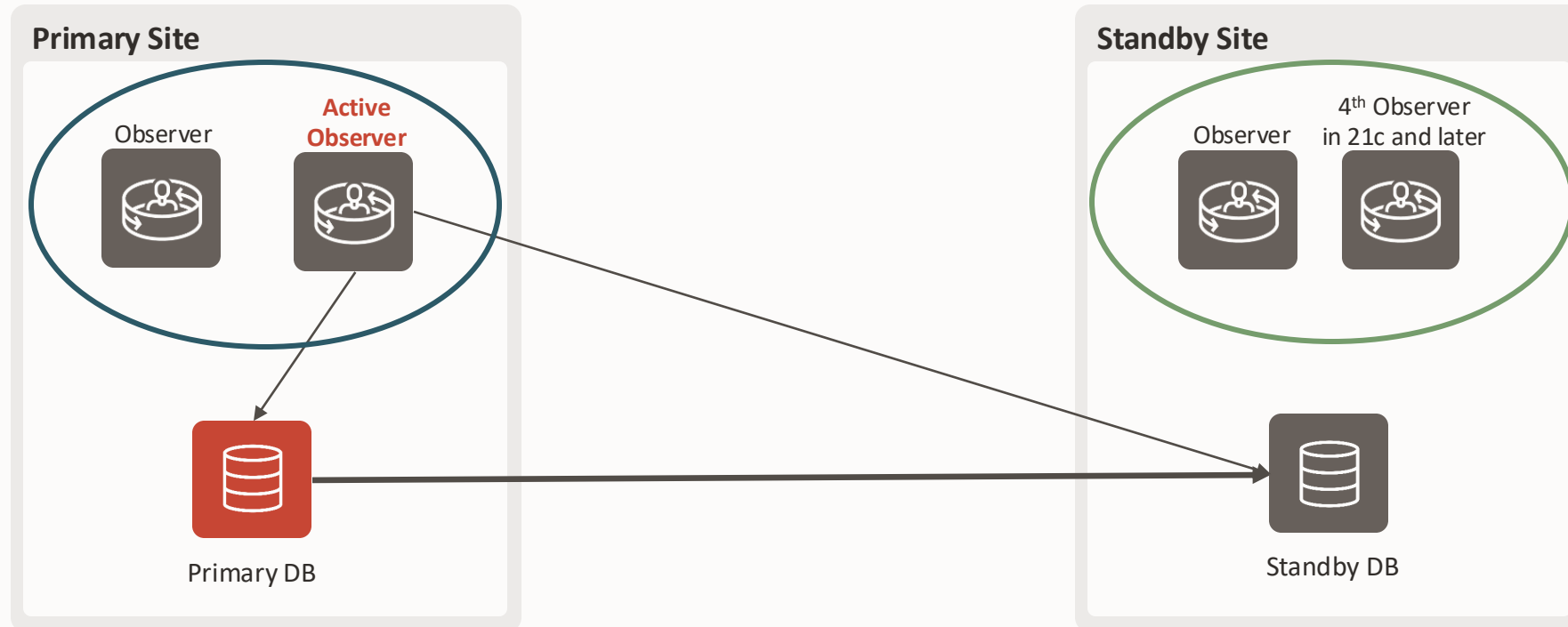
Up to four observers configured (one active at a time)



- **No external site?**
  - Configure HA observers at the primary site
  - Ideally, at least one in the application network
  - When role change occurs, have two observers ready at the secondary site

# Oracle Data Guard Observer High Availability

Optimal configuration with two sites



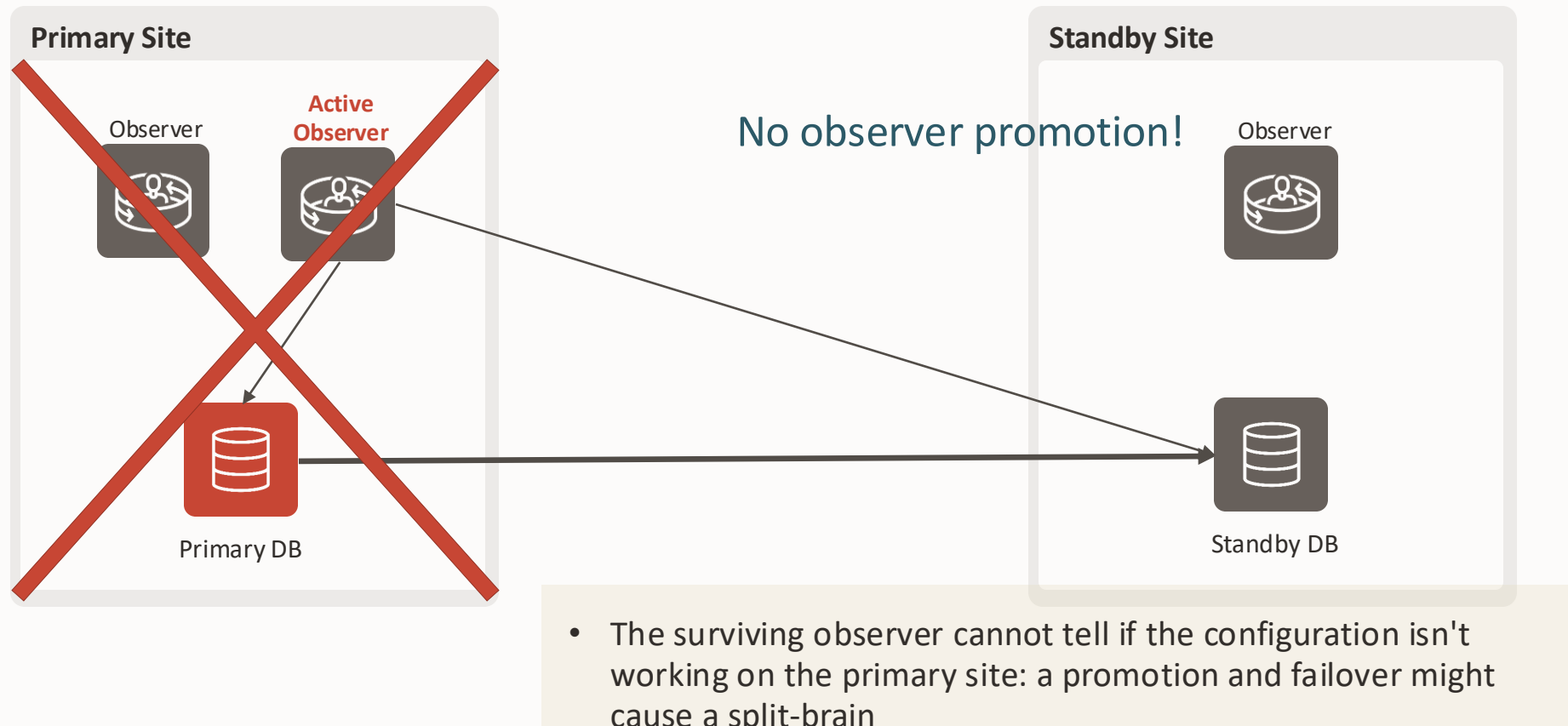
```
edit database db_site1 set property PreferredObserverHosts='obs1_site1,obs2_site1';
```

```
edit database db_site2 set property PreferredObserverHosts='obs1_site2,obs2_site2';
```



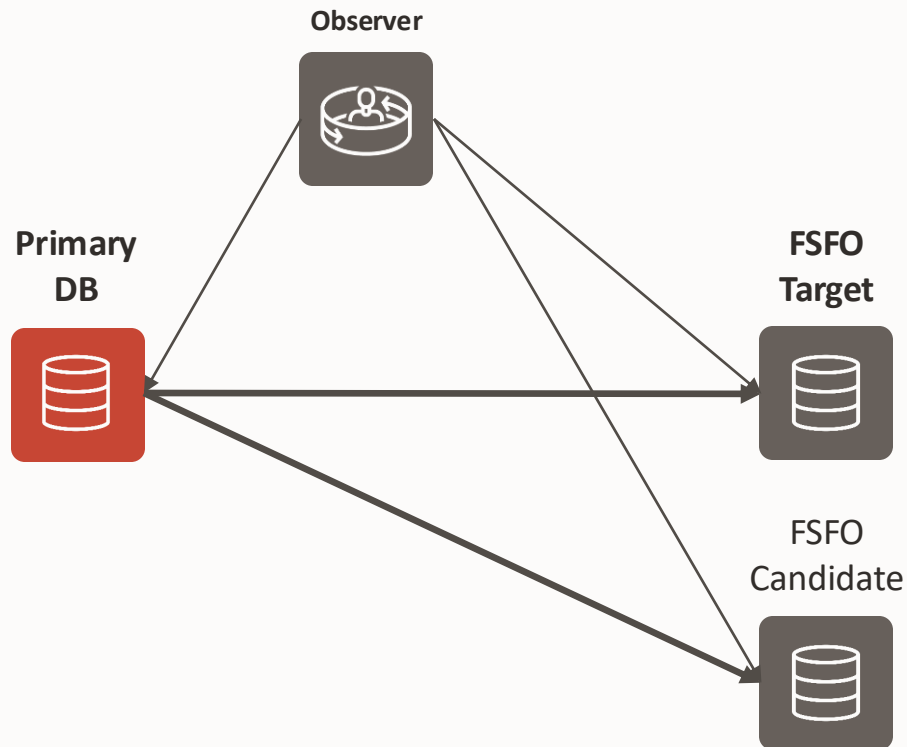
# Oracle Data Guard Observer High Availability

Observer promotion requires both primary and standby databases



# Multiple Fast-Start Failover Targets

Don't let a database failure compromise your protection



```
DGMGRL> edit database BOSTON set property
FastStartFailoverTarget='NASHUA,NEWYORK';
```

```
DGMGRL> edit database NASHUA set property
FastStartFailoverTarget='BOSTON,NEWYORK';
```

```
DGMGRL> edit database NEWYORK set property
FastStartFailoverTarget='BOSTON,NASHUA';
```

```
DGMGRL> show fast_start failover;
```

...

Active Target: NASHUA

Potential Targets: NEWYORK

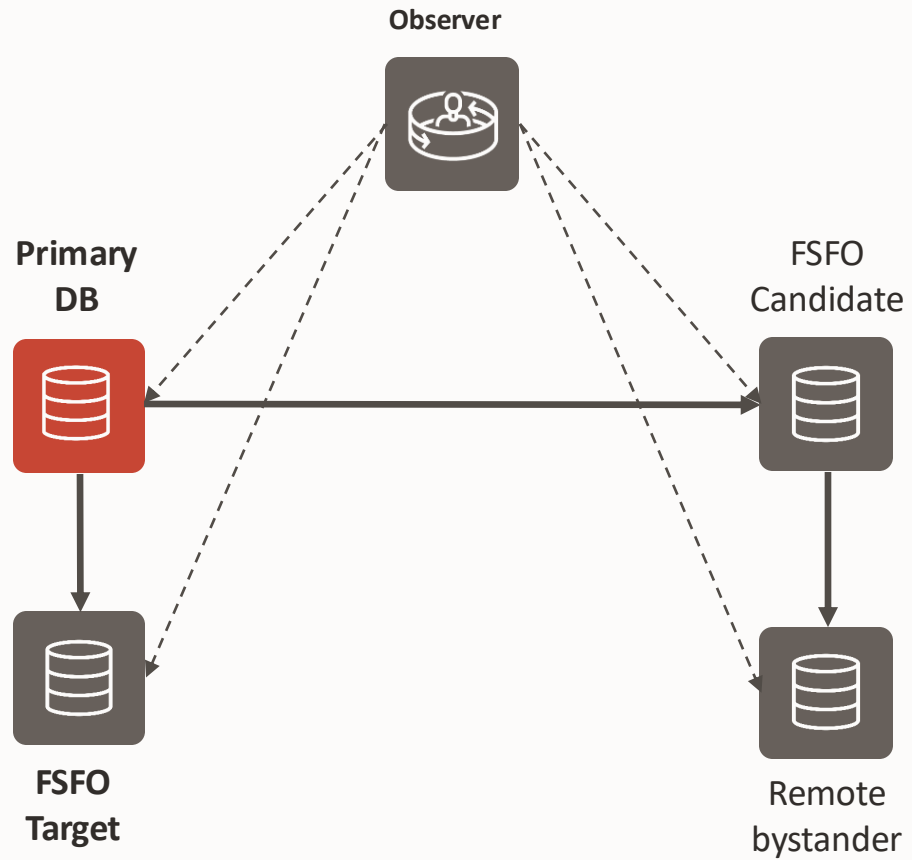
NEWYORK valid

...

Always use at least two standbys in Max Protection!

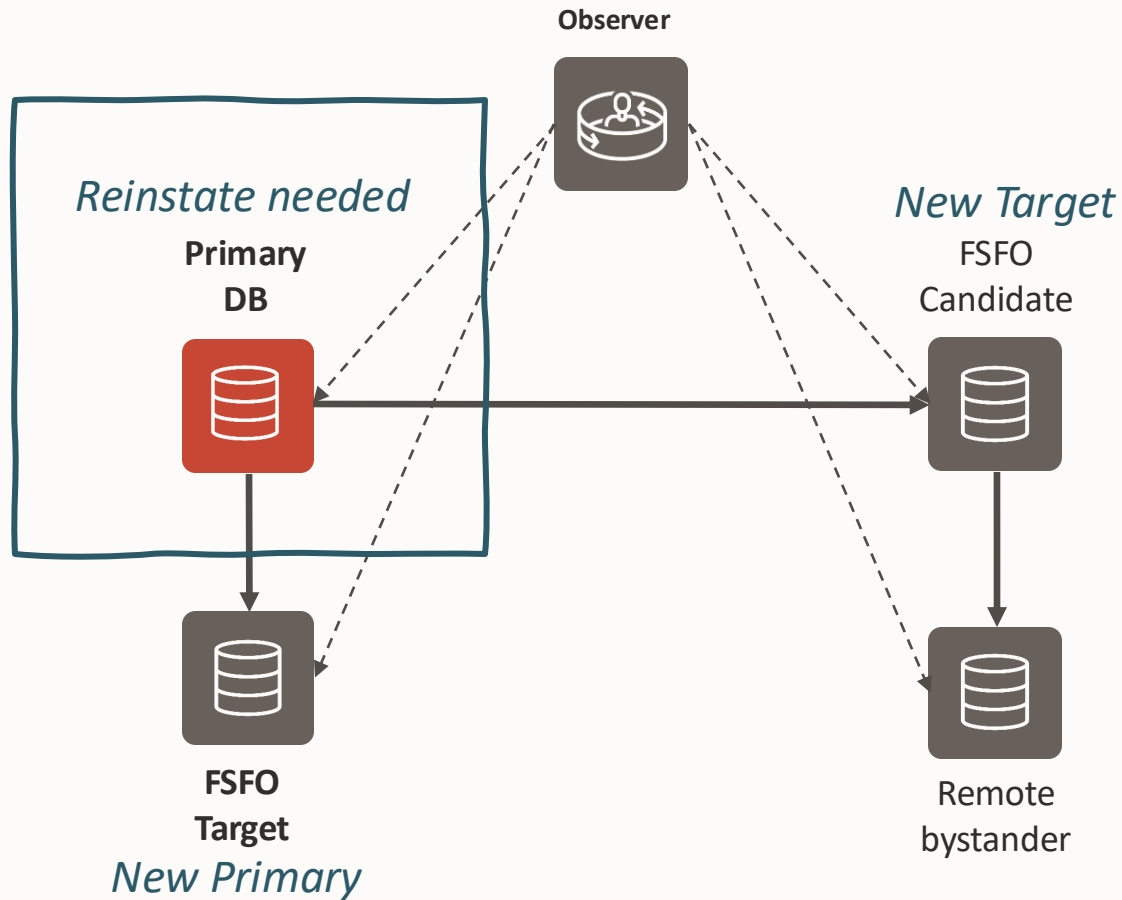
# Network partitions and consistency

## Multiple Fast-Start Failover Targets



# Network partitions and consistency

## Multiple Fast-Start Failover Targets



### Primary Isolated

- The observer can still contact the FSFO target.

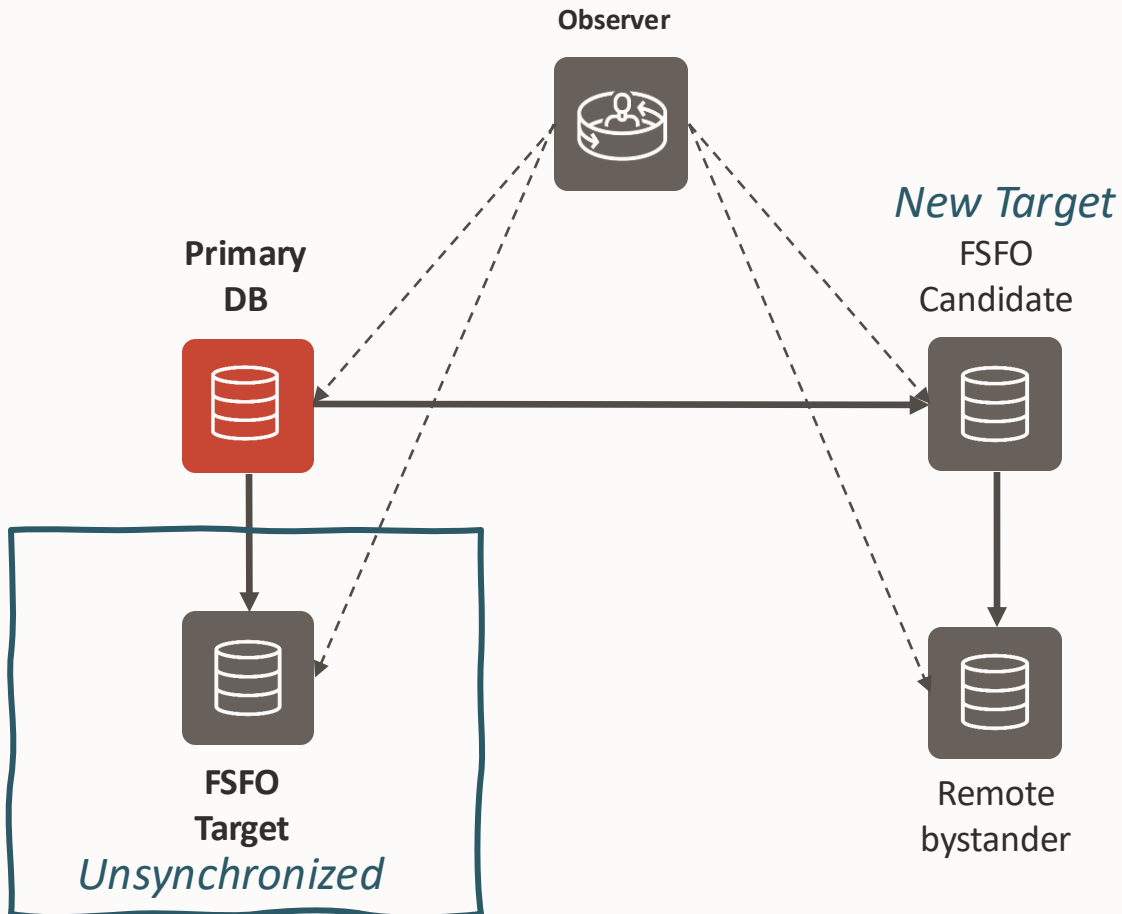
What happens:

1. The primary is STALLED.
2. The observer initiates the failover to FSFO Target.
3. The FSFO Target becomes primary.
4. The new primary and the observer agree to a new FSFO Target.
5. The former Primary DB will require a reinstate.

### Execution of Fast Start Failover

# Network partitions and consistency

## Multiple Fast-Start Failover Targets



### FSFO Target Isolated

- The observer can still contact the primary.

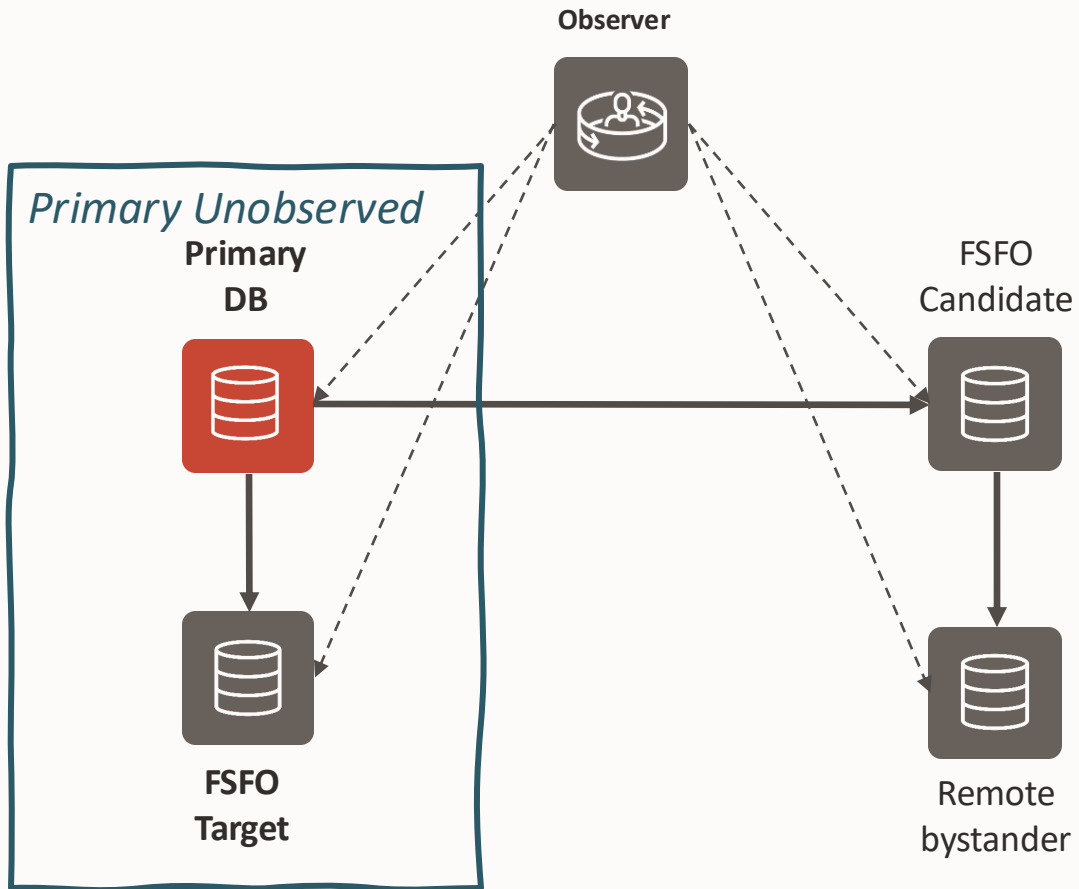
### What happens:

1. The primary goes temporarily UNSYNCHRONIZED (no FSFO, unless Max Protection).
2. The new primary and the observer agree to a new FSFO Target.
3. As soon as the new FSFO target is ready, FSFO is possible again.

*Fast Start Failover not possible for the time of target switch, then possible again*

# Network partitions and consistency

## Multiple Fast-Start Failover Targets



### Primary and FSFO Target Isolated

- The observer cannot contact the primary nor the FSFO target

### What happens:

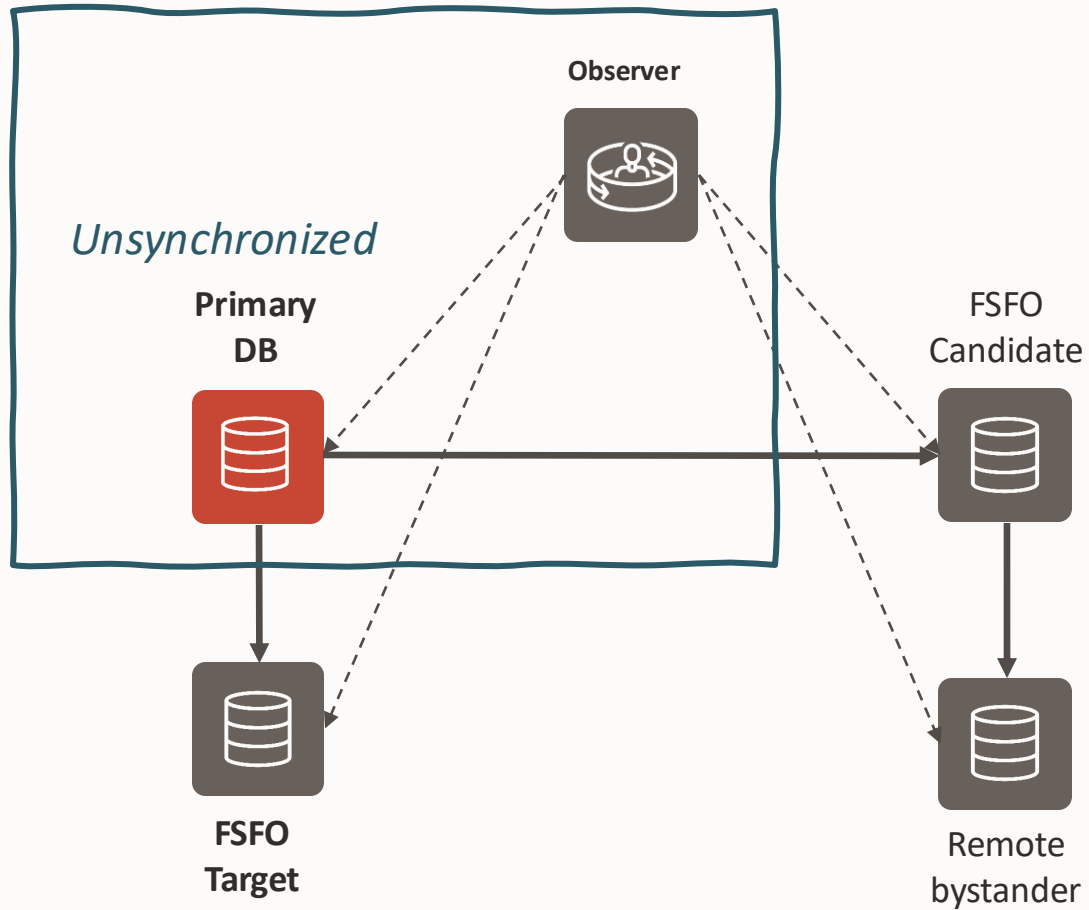
1. The observer cannot tell if the network is unreachable, or the whole site is down. The primary might still write to the standby (valid LAD destination).
2. The primary and FSFO targets keep working, the configuration is UNOBSERVED.
3. The observer cannot initiate a failover, as it would lead to split-brain and data loss.

*Fast Start Failover not possible*



# Network partitions and consistency

## Multiple Fast-Start Failover Targets



### Primary and Observer Isolated

- No FSFO target or candidates can be contacted.

### What happens:

1. The primary keeps working without protection (unless Max Protection).

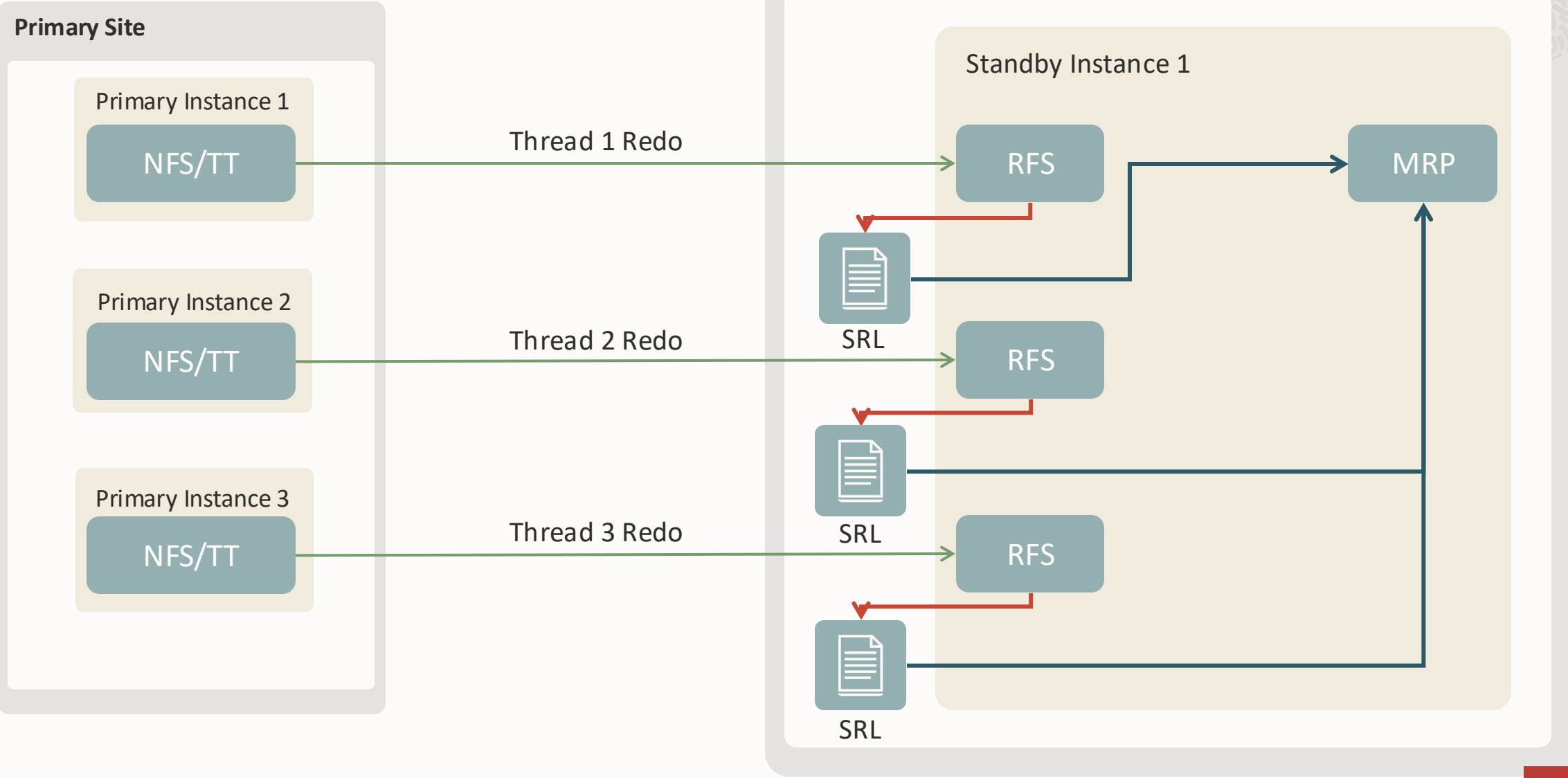
*Fast Start Failover not possible*



# Multi-Instance Redo Apply (MIRA)

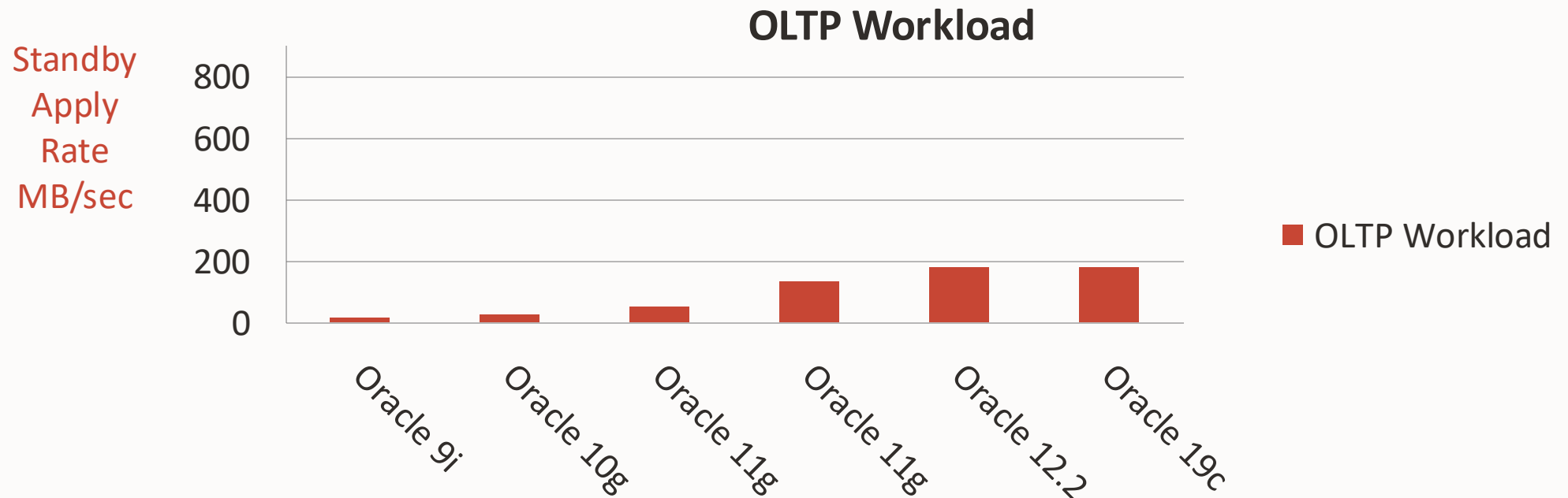


# Single-Instance Redo Apply (SIRA)

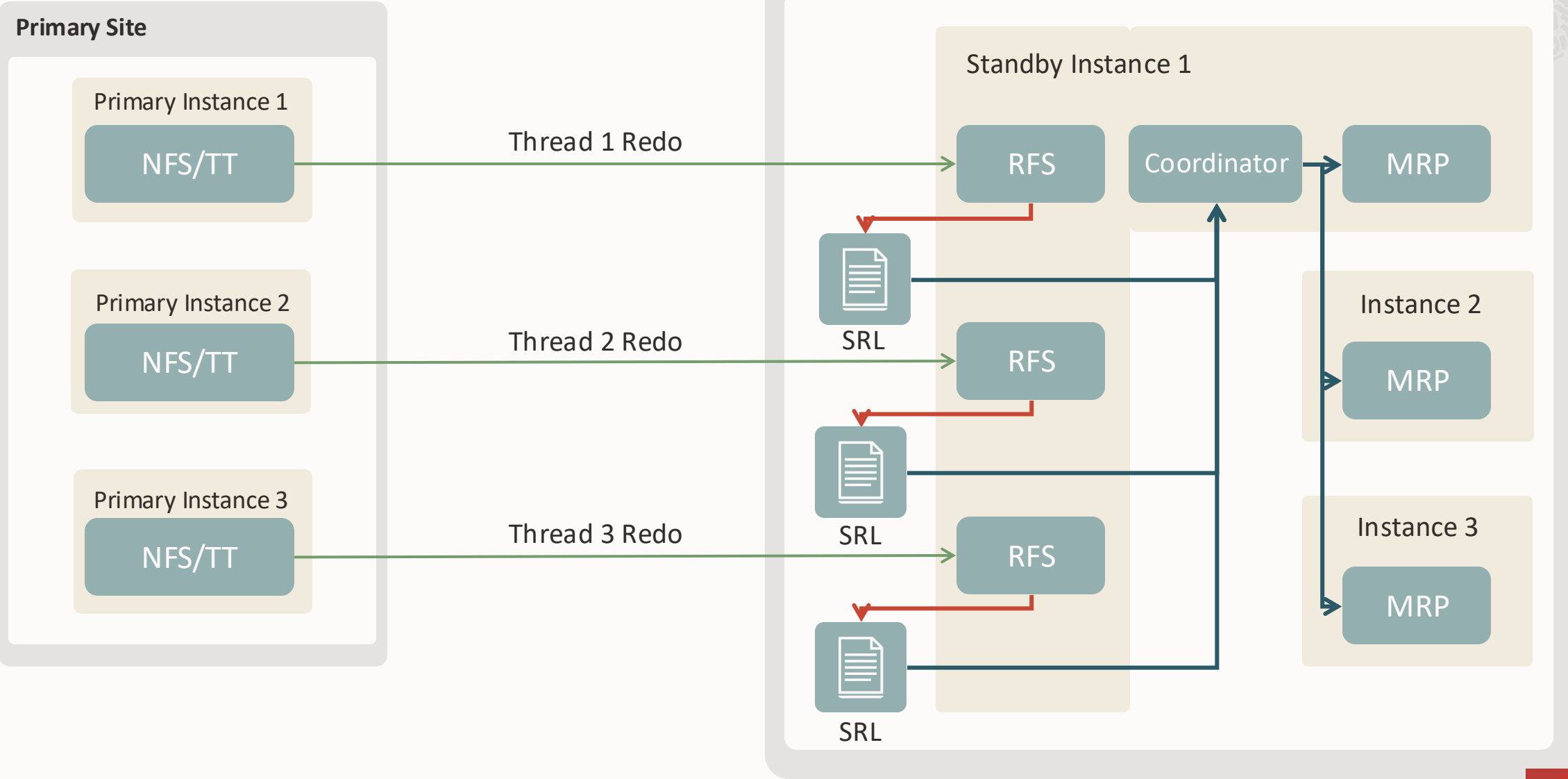


## Single-Instance Redo Apply (SIRA)

- The MRP and its redo apply servers run on one node of a Physical Standby RAC.
- Single-Instance Redo Apply performance generally meets all use cases.
- Before considering Multi-Instance Apply, make sure you apply the best practices for Redo Apply
- On Exadata, the standby can keep up with redo rates of 1GB/s



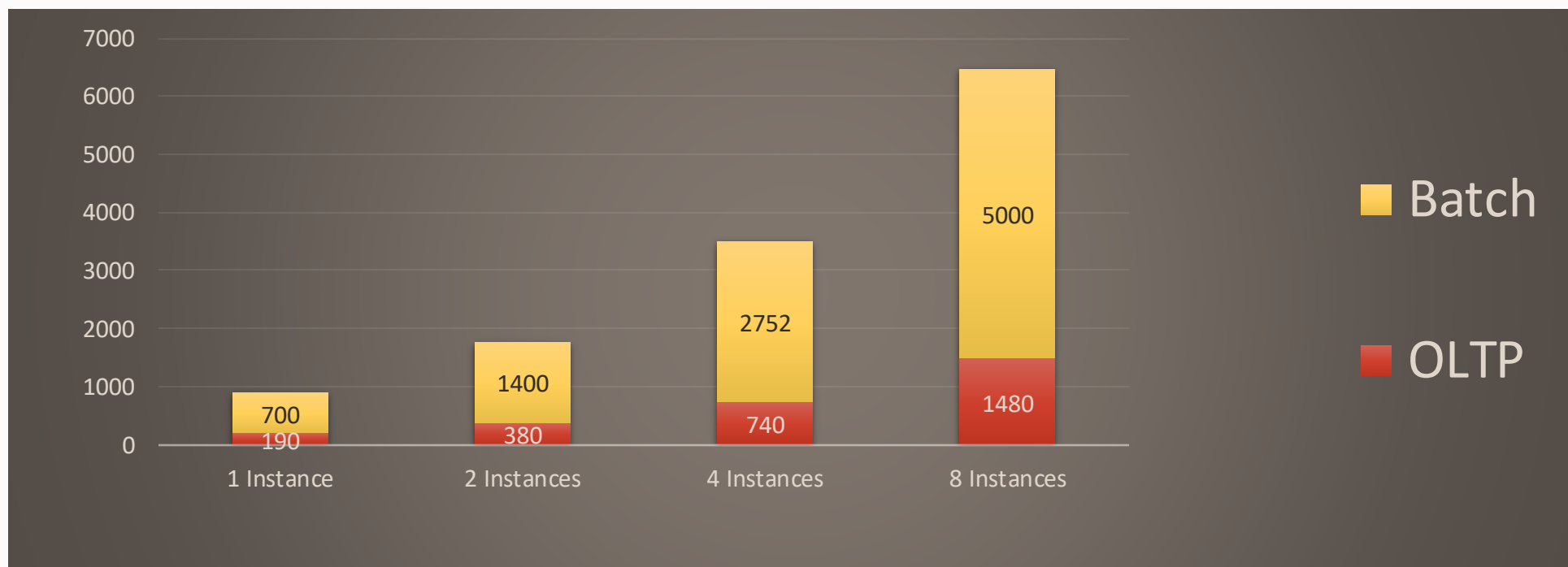
# Multi-Instance Redo Apply



# Multi-Instance Redo Apply

- Utilizes all RAC nodes on the Standby database to parallelize recovery
- OLTP workloads on Exadata show great scalability
- When the bottleneck is SIRA, generally 30% improvement or more, depending on the workload

Standby  
Apply  
Rate  
MB/sec



# When to consider Multi-Instance Redo Apply



- **IO bottlenecks** and **database wait events** affecting SIRA, will affect MIRA as well!
- Consider MIRA only if SIRA *is* the bottleneck and cannot meet the SLA
- Always use a recent Oracle Database 19c/23ai release update
- CPU-bound apply coordinator or worker are the best indicators that MIRA is needed
- **Oracle Data Guard Configuration Best Practices**  
<https://docs.oracle.com/en/database/oracle/oracle-database/23/haovw/configure-and-deploy-oracle-data-guard.html#GUID-97769612-4980-42C2-A28C-4C5E49FE2824>
- **Redo Apply Troubleshooting and Tuning**  
<https://docs.oracle.com/en/database/oracle/oracle-database/23/haovw/tune-and-troubleshoot-oracle-data-guard.html#GUID-E8C27979-9D37-4899-9306-A5AE2B5CF6C0>

# How to Enable Multi-Instance Redo Apply

With the Data Guard Broker:

```
DGMGRL> edit database <standby> set property ApplyInstances=<#|ALL>;
```

From SQL\*Plus:

```
SQL> alter database recover managed standby database disconnect from session instances <#|ALL>;
```

Entries in the alert log:

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION INSTANCES ALL
2024-05-23T11:37:09.937690+01:00
Attempt to start background Managed Standby Recovery process (<db_unique_name>)
...
2024-05-23T11:37:15.111518+01:00
Started logmerger process on instance id 1
Started logmerger process on instance id 2
Starting Multi Instance Redo Apply (MIRA) on 2 instances
...
2024-05-23T11:37:16.027775+01:00
Started 24 apply slaves on instance id 1
2024-05-23T11:37:16.545221+01:00
Started 24 apply slaves on instance id 2
```

# Multi-Instance Redo Apply on Exadata



- Exadata prerequisites to enable MIRA

Exadata Systems	RDBMS version	Steps
With PMEM	19.13 and higher	No additional steps
Without PMEM	19.13 and higher	Set dynamic parameter on all instances: "_cache_fusion_pipelined_updates_enable"=FALSE (*)
Any Exadata System	19.12 and lower	Apply Patch 31962730 and set dynamic parameter on all instances: "_cache_fusion_pipelined_updates_enable"=FALSE (*)

(\*) MIRA can recover only redo generated with the "\_cache\_fusion\_pipelined\_updates\_enable" set to FALSE

- Using ExaWatcher Charts to Monitor Exadata Database Machine Performance  
<https://docs.oracle.com/en/engineered-systems/exadata-database-machine/dbmmn/exadata-general-maintenance.html#GUID-5AEB3139-333D-453F-91D6-8EB09CB6E6EB>



## Tuning Multi-Instance Redo Apply

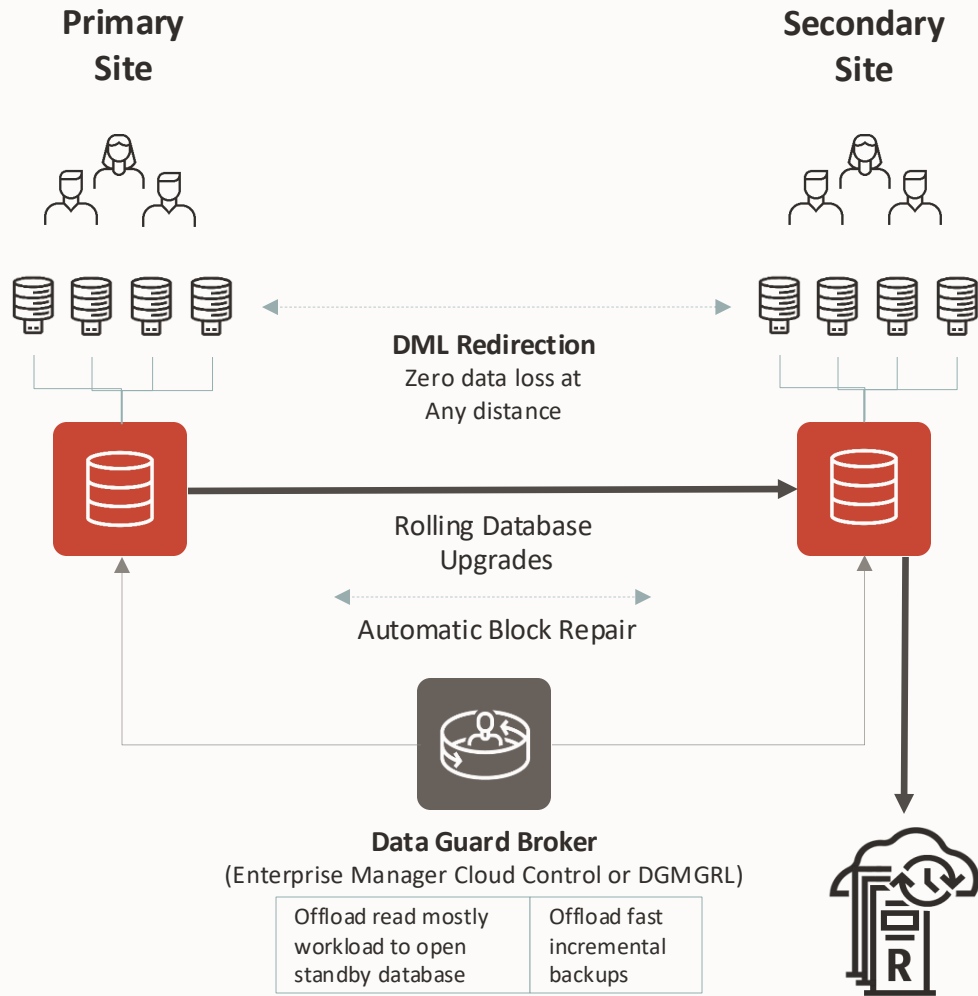


- Tune Redo Apply by evaluating Database **wait events**
- If **recovery apply pending** and/or **recovery receive buffer free** are among the top wait events:
  - Incrementally increase `_mira_num_receive_buffers` and `_mira_num_local_buffers` by 100
  - Additionally set “`_mira_rcv_max_buffers`”=10000
  - The additional memory requirements for each participating MIRA RAC instance:  
 $(\_mira\_num\_receive\_buffers + \_mira\_num\_local\_buffers) * (\#instances * 2MB)$
- If **parallel recovery change buffer free** is among the top wait events:
  - Increase `_change_vector_buffers` to 2 or 4.



# Oracle **Active** Data Guard Overview

# Oracle Active Data Guard



## Oracle Data Guard features, plus:

- **Active-active**
  - Queries, reports, backups
  - Occasional updates (19c)
  - Assurance of knowing system is operational
- **Automatic block repair**
- **Application Continuity**
- **Zero data loss across any distance**
- **Rolling Upgrades and Maintenance**
- **Real-time cascaded standbys**
- **Global Data Services**

<https://www.oracle.com/database/technologies/high-availability/dataguard-activedataguard-demos.html>

# Active Data Guard

Option of Oracle Database for Advanced Capabilities and Protection



Zero data loss at any distance

Real-time cascade

Automatic Block Repair

Automatic block repair

Automated rolling database maintenance

Application continuity

Service management for replicated databases

Rolling Upgrade

Extreme throughput - supports all workloads

Dual-purpose standby for development and test

Integrated management

Offload network compression

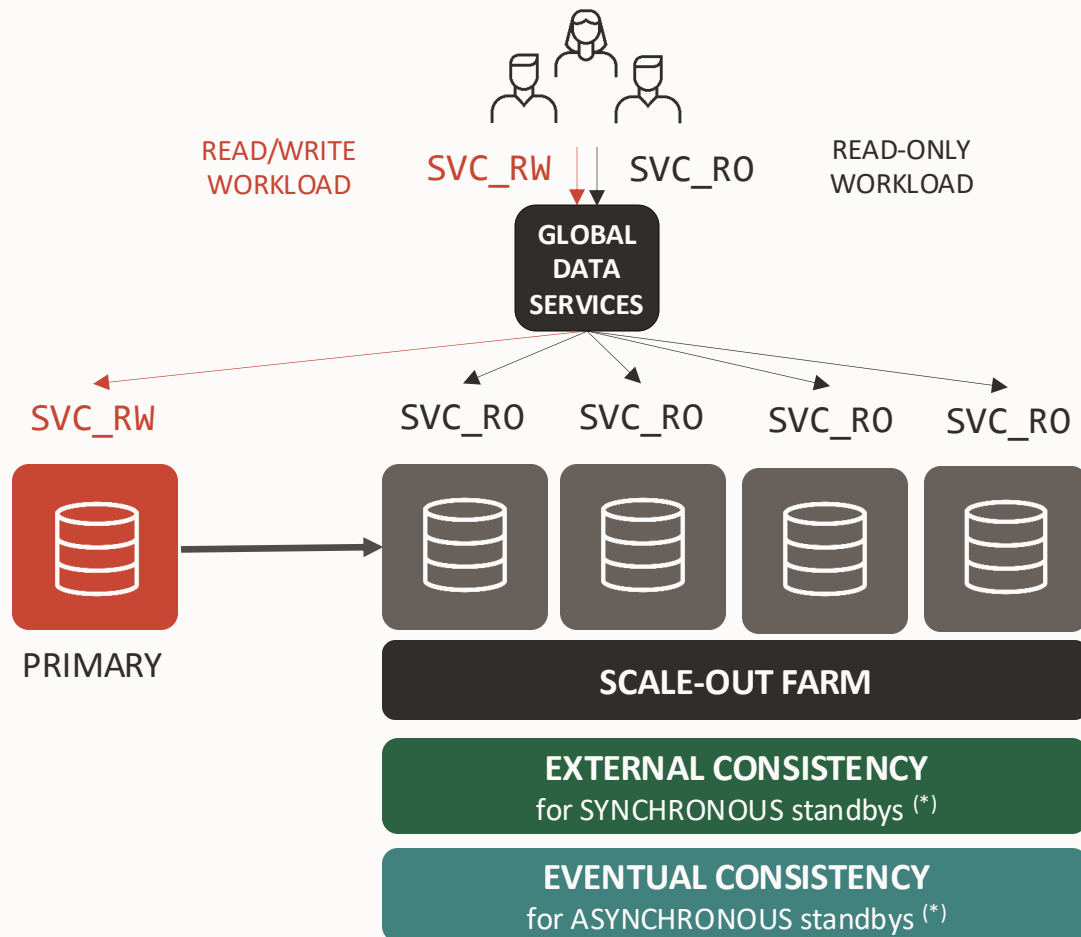
Intelligent load balancing for replicated databases

Active Standby DML redirection

# Oracle **Active** Data Guard Real-Time Query

# Scale and Improve ROI of your Active Data Guard Environments

Scale linearly by opening standbys read-only for additional processing power



## Benefits of offloading read-only workload:

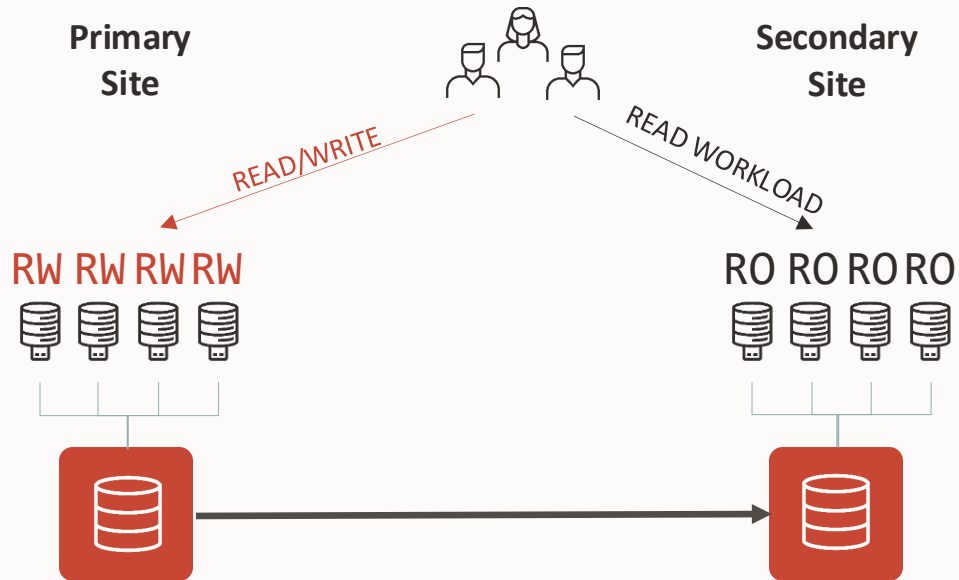
- Linear scalability of read-only (RO) workloads
- Isolation of primary from heavy queries
- RO sessions uninterrupted during role changes
- Reports and data extractions off the primary
- Fast incremental backups on the standby databases

## Global Data Service (optional):

- Simplifies the application configuration with a single, highly-available endpoint
- Sessions are redirected to the best standby based on locality and load
- Problematic or lagging standbys are excluded automatically

# Active Data Guard Real-Time Query

## Read-only Standby while Recovery is Active



### Activation

With Data Guard Broker:

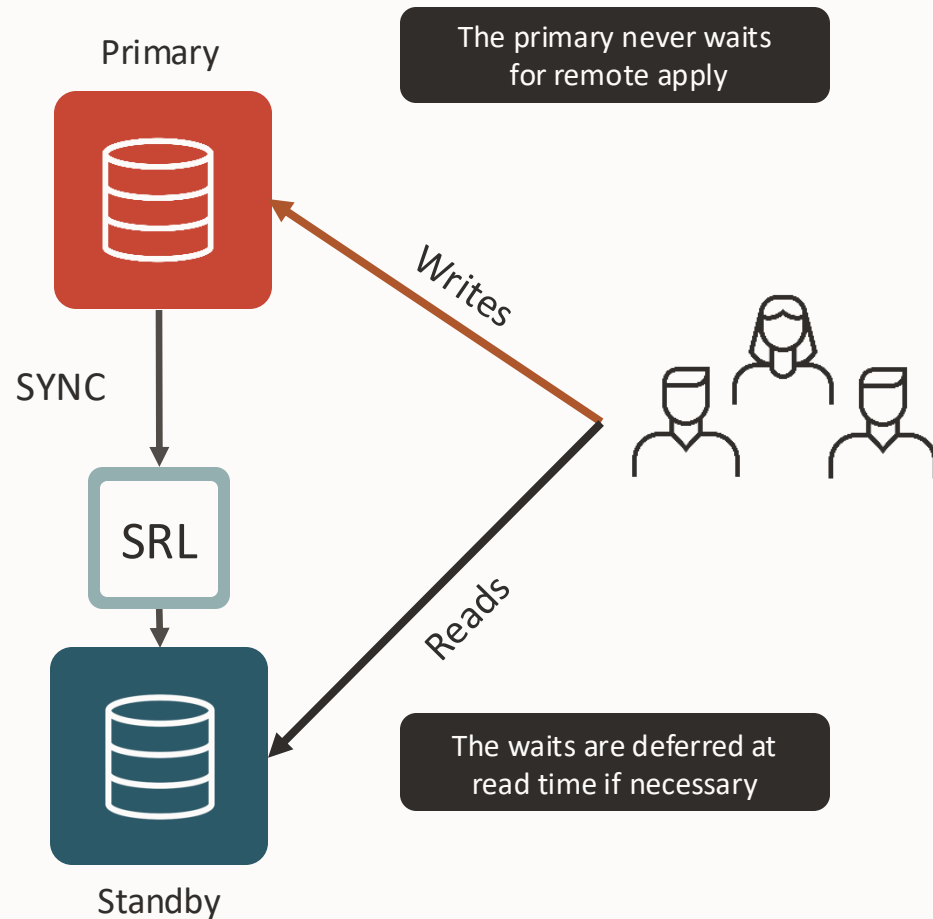
```
SQL> ALTER DATABASE OPEN;
```

Without Data Guard Broker:

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;  
ALTER DATABASE OPEN;  
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT;
```

# Real-Time Query Apply Lag Limit

Full external consistency at the session level



```
-- log transport to the standby must be synchronous
DGMGRL> edit database prim
  set property LogXptMode='SYNC';
```

```
-- write on the primary
insert into emp values (...);
commit;
```

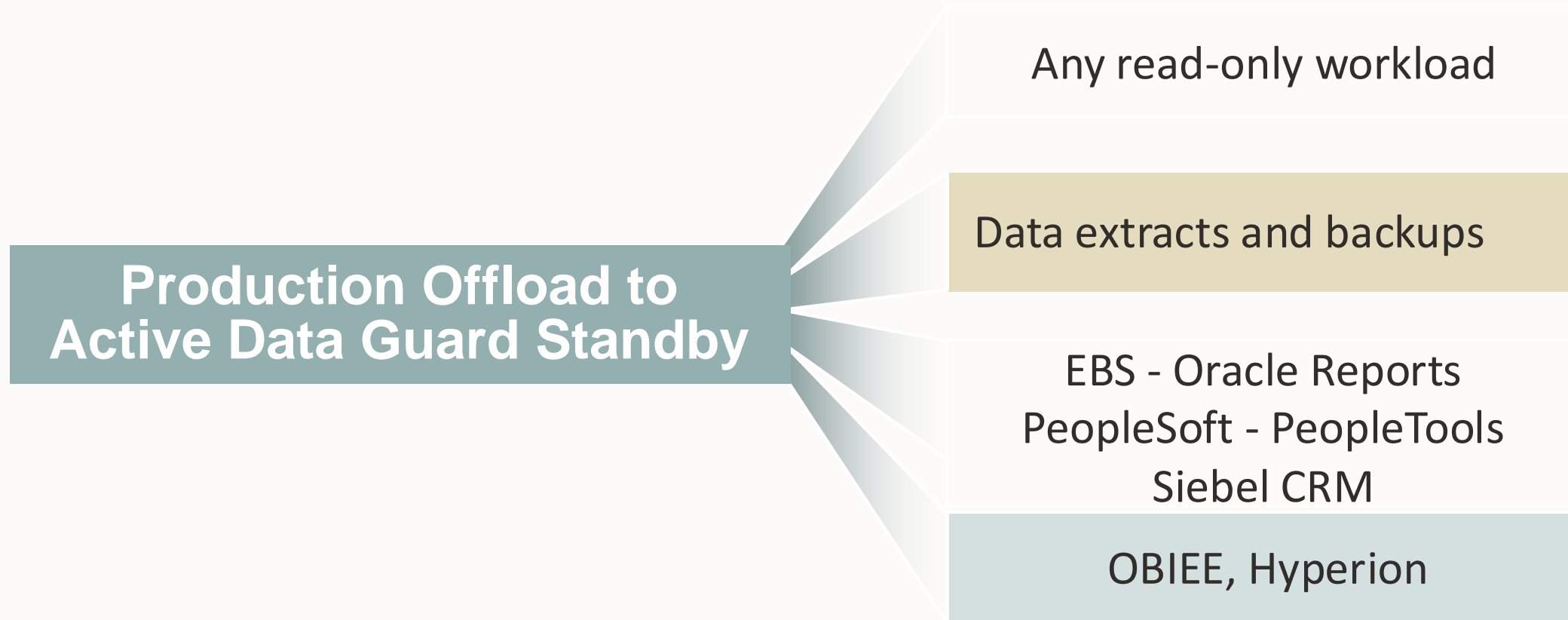
```
-- read on the standby
-- wait once until current SCN is applied
alter session sync with primary;
```

```
-- or always have READ COMMITTED in the session
alter session set standby_max_data_delay=0;
```

```
select first_name from emp where ...;
```

# Offload Read-Only Workloads

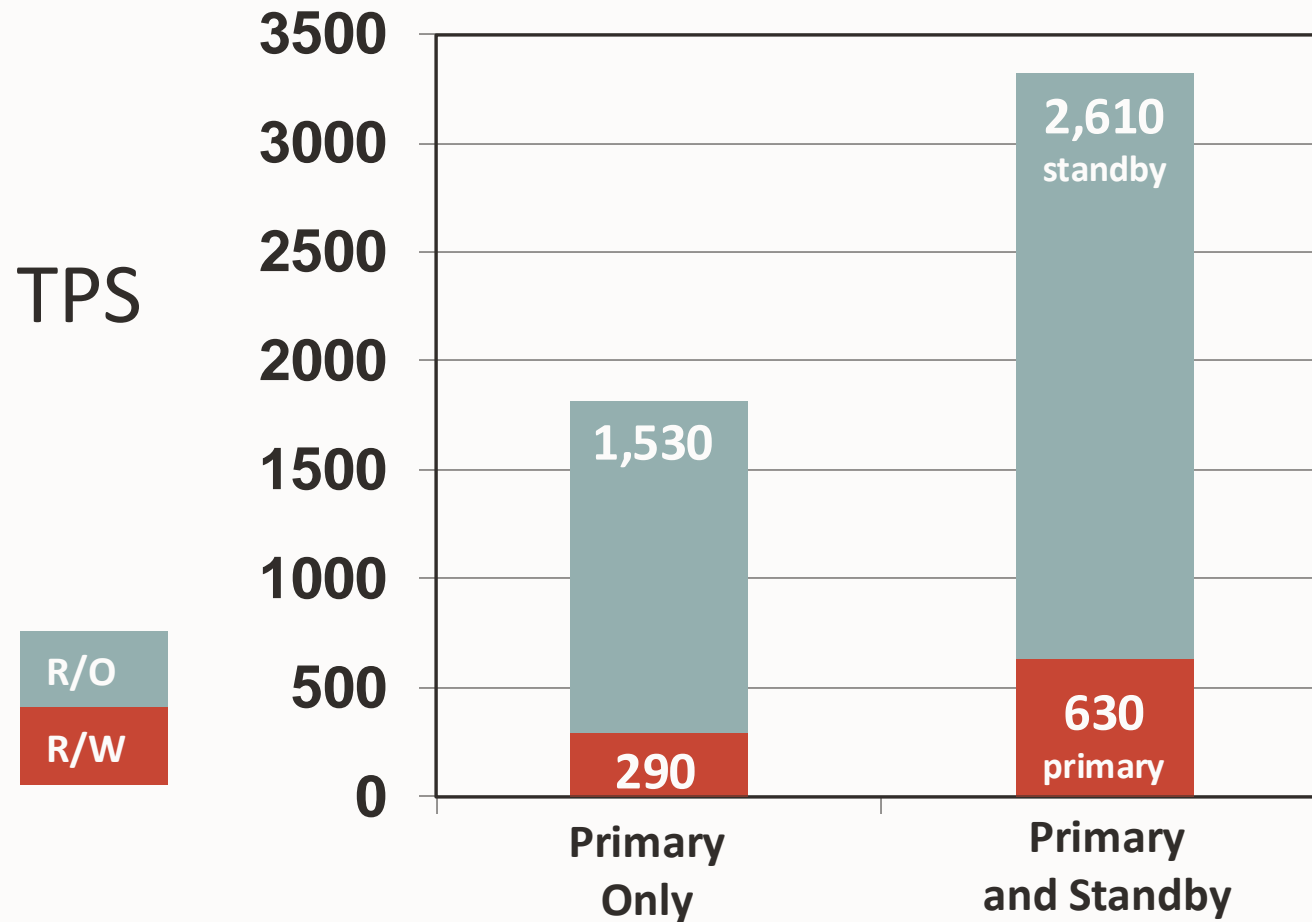
Increase Performance and ROI – Standby is a Production System





# Standby Offload Increases Performance for all Workloads

Bring Idle Capacity Online



Double read-write throughput

Increase read-only throughput by 70% with a single standby database

Eliminate contention between read-write and read-only workloads

# Real-Time Query

Not just Selects for your Application Workloads!



SQL Performance Analyzer

Oracle Database In-Memory \*

Global Temporary Tables

R/O Connections Preserved

Sequences

Updates on Active Data Guard

Standby Result Cache preservation

Simplified AWR snapshots

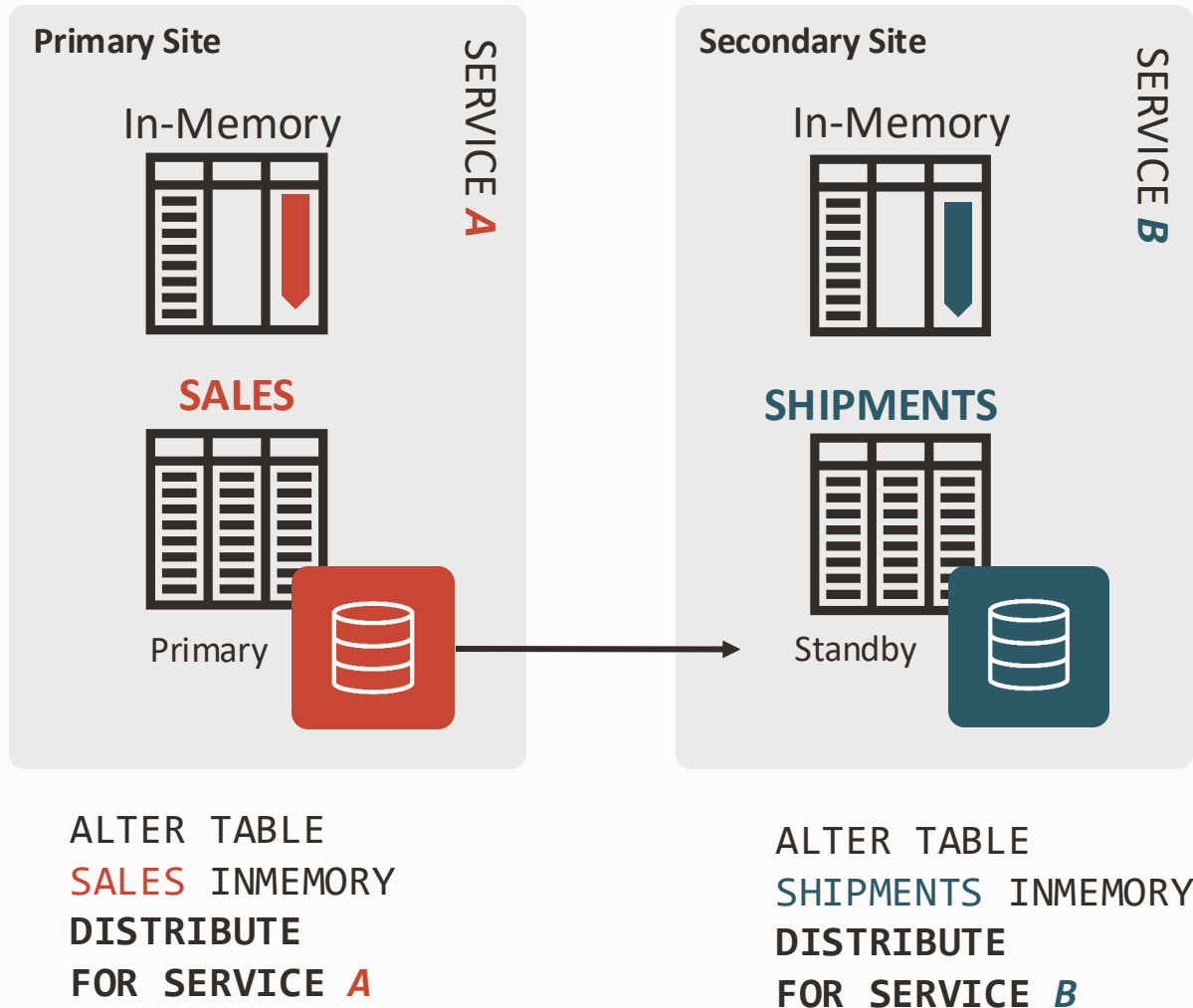
**NEW in 21c**

**NEW in 23ai**

\* Only on Engineered Systems or Oracle Cloud Infrastructure

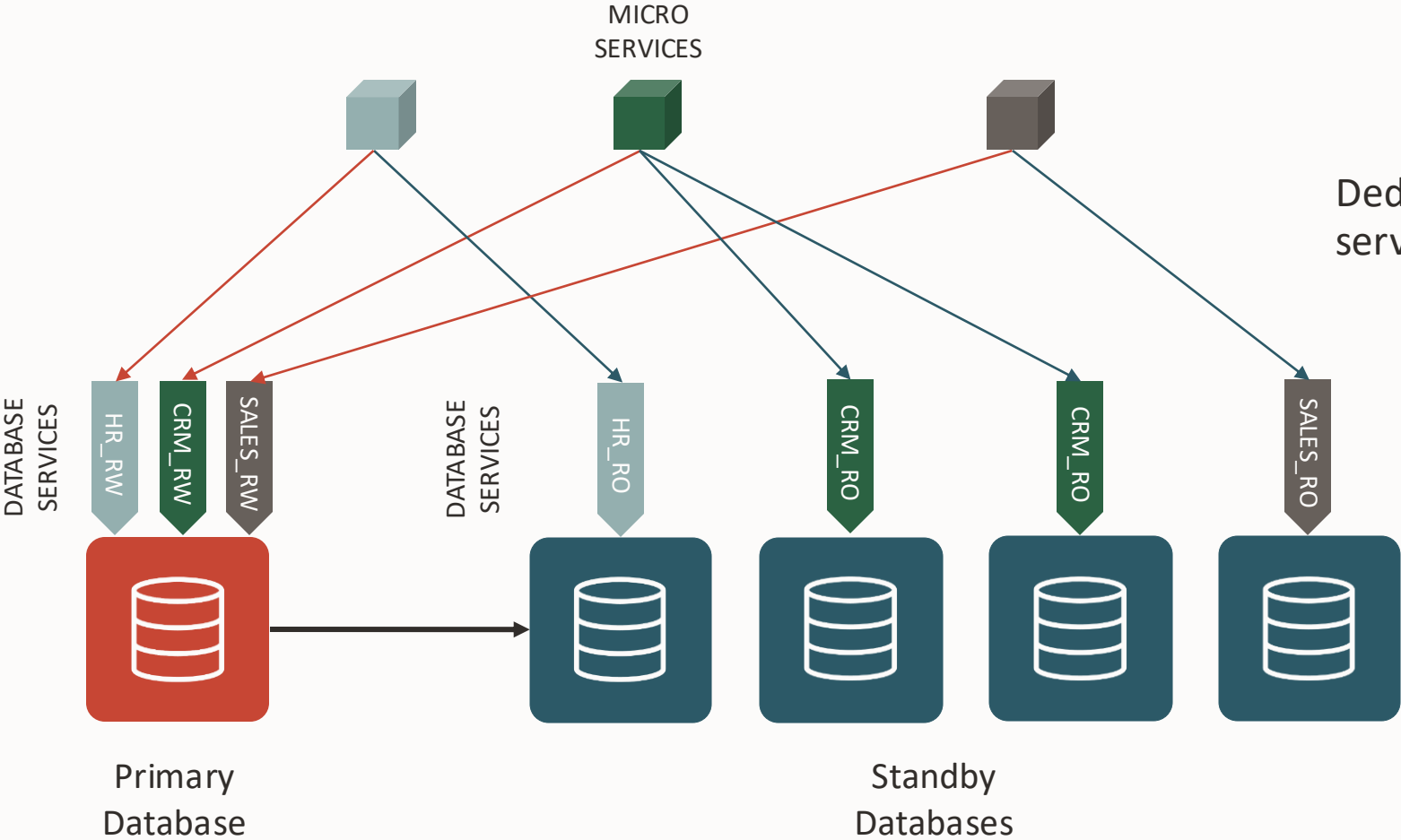


# Active Data Guard and Database In-Memory



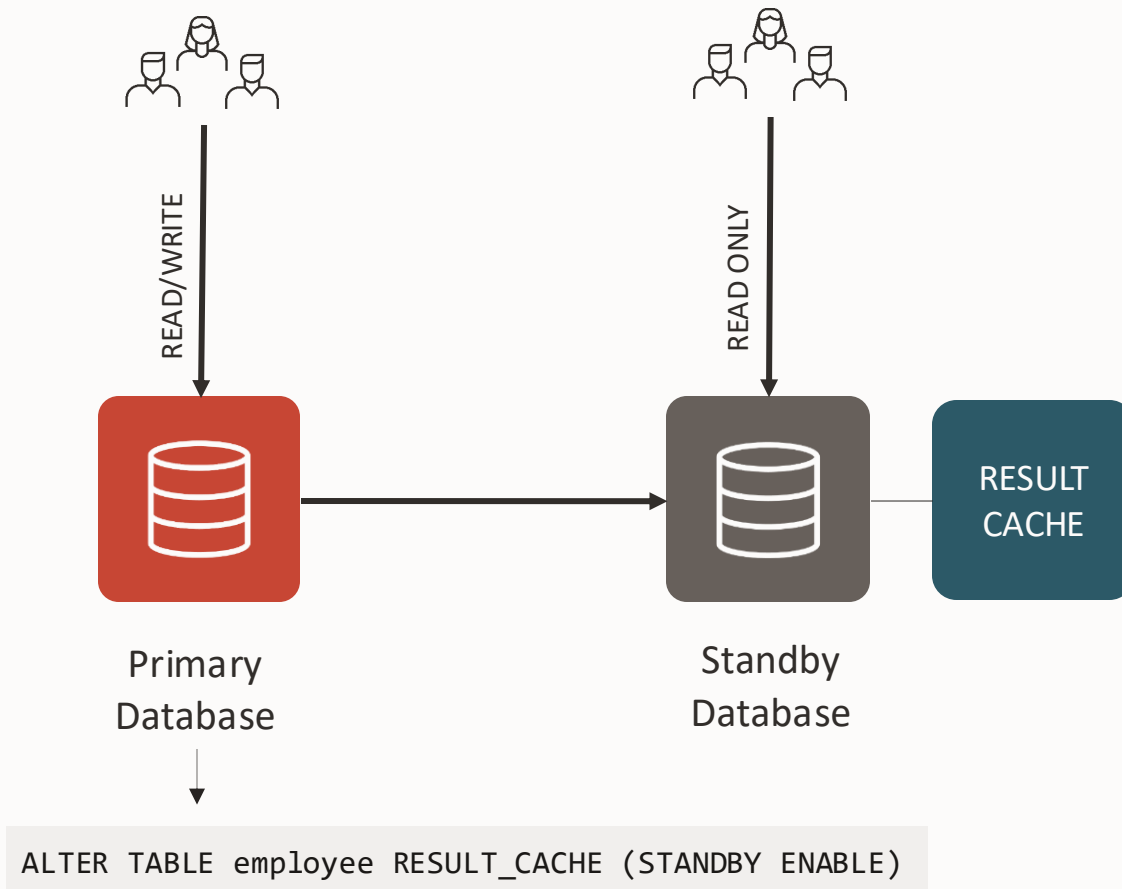
- In-Memory queries run on standby
  - No impact on the primary database
  - Full use of standby database resources
- Standby can have different in-memory contents from Primary
  - **DISTRIBUTE FOR SERVICE** subclause used to determine data placement
  - Increases total effective in-memory columnar **capacity**
  - Increases column store **availability**:
    - Reporting workload on standby unaffected by primary site outage

# Service Distribution to Optimize the Buffer Cache



# Standby Buffer Cache and Result Cache preservation

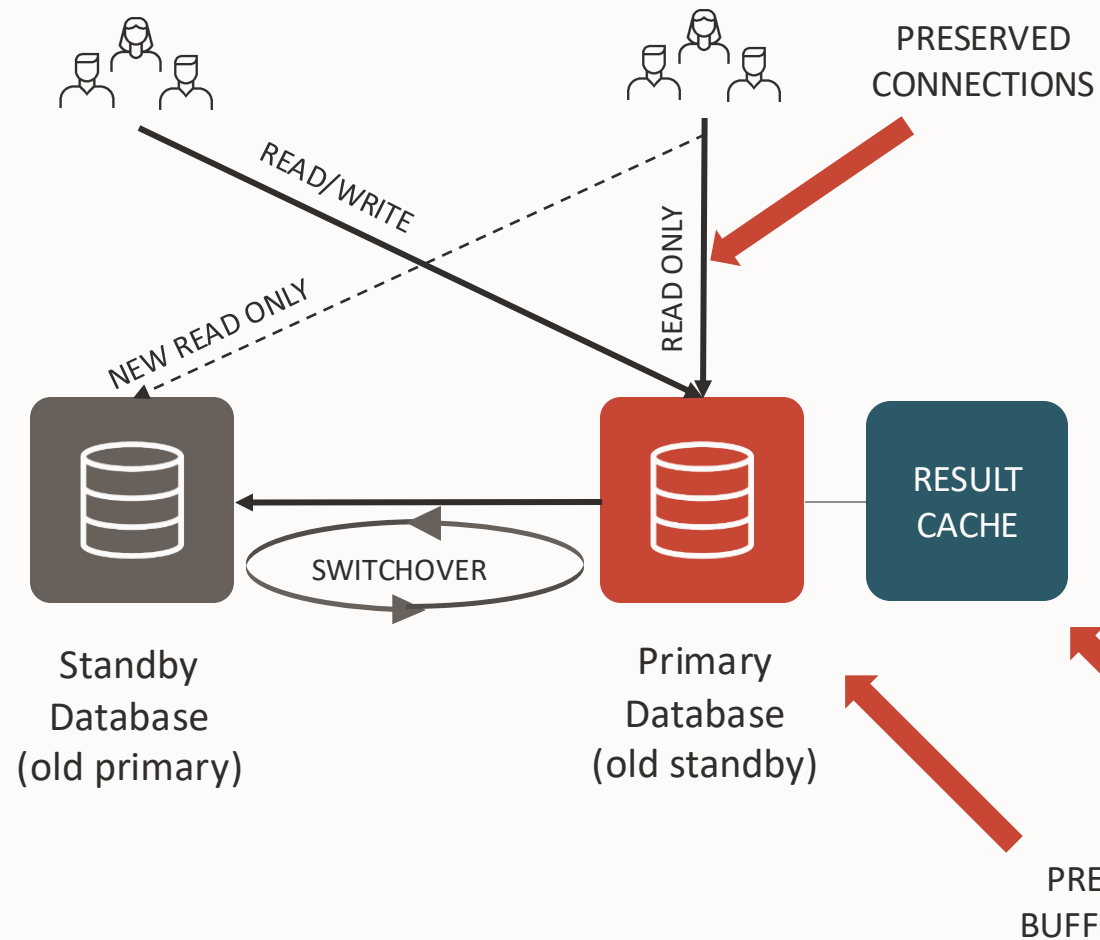
Keep the Caches warm after a role transition



- Real-Time Query supports the Result Cache for queries run on the standby database (tables only)
- Result Cache improves query performance for recurring queries and reduces resource usage (CPU, I/O)

# Standby Buffer Cache and Result Cache preservation

Keep the Caches warm after a role transition



- Real-Time Query supports the Result Cache for queries run on the standby database (tables only)
- Result Cache improves query performance for recurring queries and reduces resource usage (CPU, I/O)
- In **21c**, after a role transition (switchover or failover), the result cache is preserved along with the buffer cache
  - Query performance not impacted
  - No cache warm-up required

# Simplified AWR snapshots on Active Data Guard

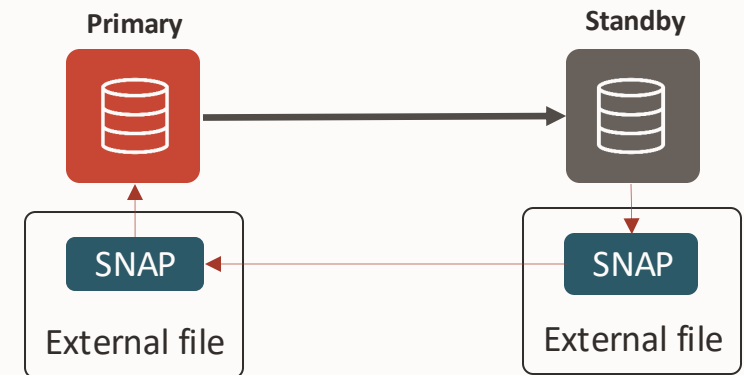
Just create the snapshot on the standby, and you are ready to go.

```
-- at the PDB or CDB level
alter session set container = PDB1;

-- the snapshot can be created without additional configuration
select dbms_workload_repository.create_snapshot from dual;

-- create the report at the PDB or CDB level
@?/rdbms/admin/awrrpti
```

```
-- databases already using SYS$UMF can set this
-- to use the new framework instead:
alter system set "_umf_remote_enabled"=FALSE;
```

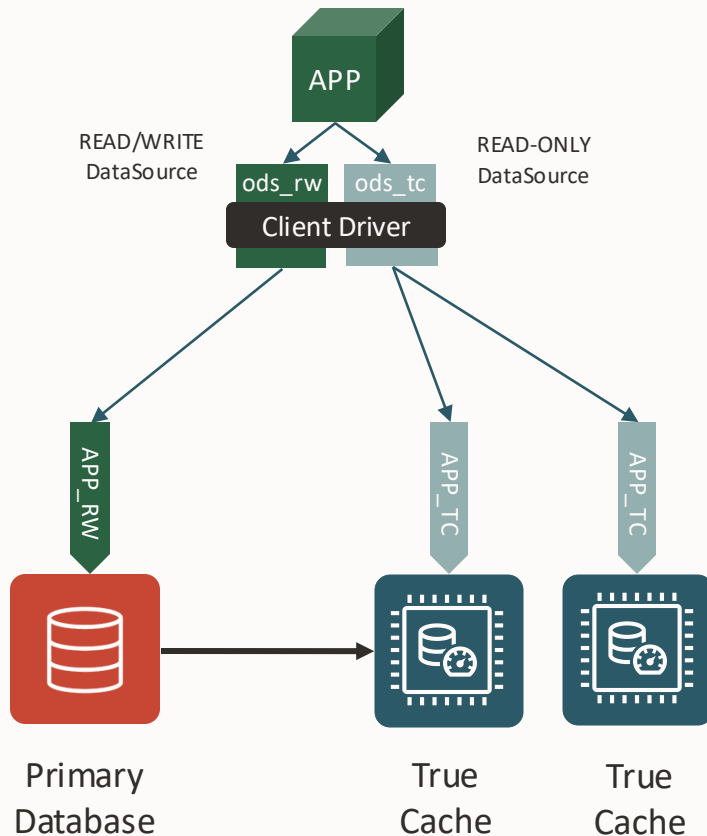


# Oracle **Active** Data Guard True Cache Driver Redirection



# Scaling out read-mostly workloads using two (or more) data sources

One connection pool per service, maintained by the application



```
import oracle.jdbc.datasource.impl.OracleDataSource;

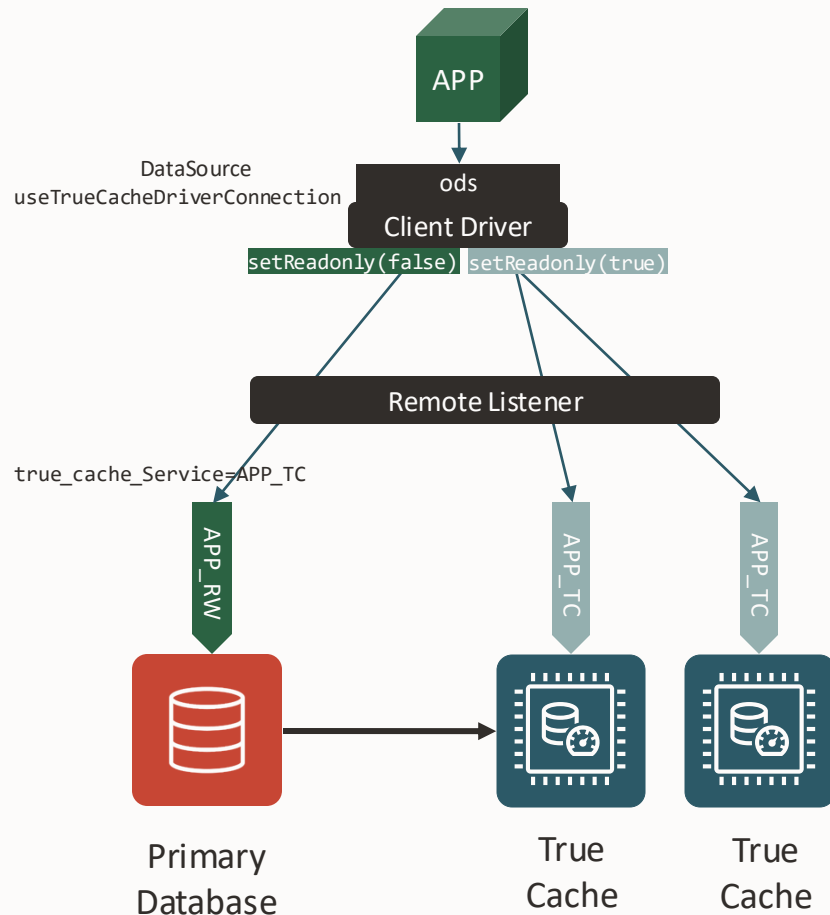
OracleDataSource ods_rw = new OracleDataSource();
ods_rw.setURL(url_app_rw);
...
try (Connection conn_rw = ods_rw.getConnection()) {
    // This is connected to the Primary
} catch (SQLException sqex) {
    ...
}

OracleDataSource ods_tc = new OracleDataSource();
ods_tc.setURL(url_app_tc);
...
try (Connection conn_tc = ods_tc.getConnection()) {
    // This is connected to the True Cache
} catch (SQLException sqex) {
    ...
}
```

- The application maintains two data sources
- The connection string and listeners load-balance the connections in the pool across the available instances and replicas
- Available with any driver

# Scaling out using the JDBC Thin True Cache driver

One connection pool with transparent automatic redirection



```
import oracle.jdbc.datasource.impl.OracleDataSource;

OracleDataSource ods = new OracleDataSource();
ods.setURL(url_app);
...
ods.setConnectionProperty(
    "oracle.jdbc.useTrueCacheDriverConnection", "true");
try (Connection conn = ods.getConnection()) {
    // This is connected to the Primary

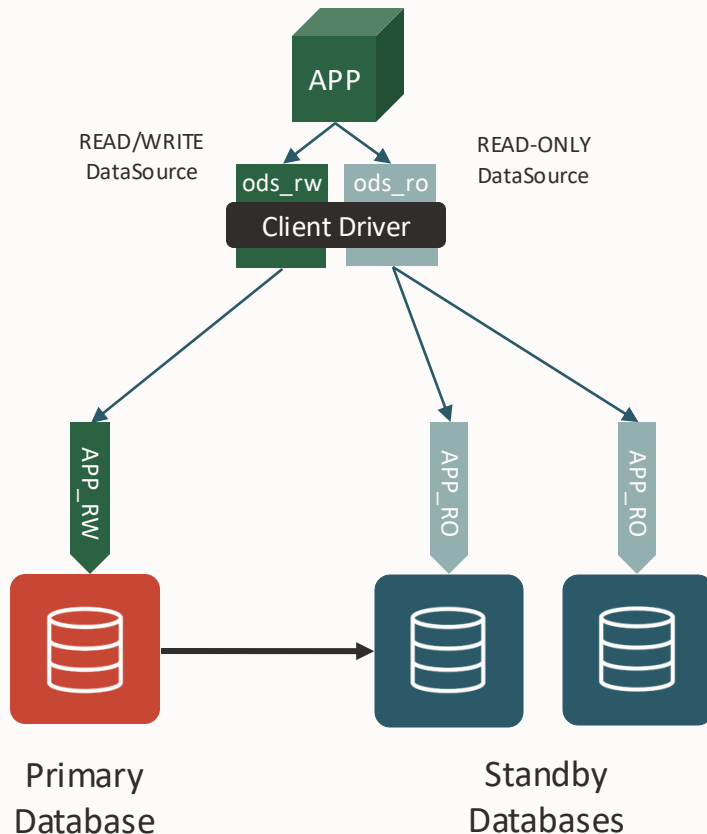
    conn.setReadOnly(true);
    // This is connected to the True Cache

} catch (SQLException sqex) {
    ...
}
```

- The application maintains **just one data source**
- The driver transparently redirects the ReadOnly context to the True Cache-enabled service
- The remote listener load balances the read-only connections
- Available for 23+ JDBC Thin drivers only

# Scaling out read-mostly workloads using two (or more) data sources

One connection pool per service, maintained by the application



```
import oracle.jdbc.datasource.impl.OracleDataSource;

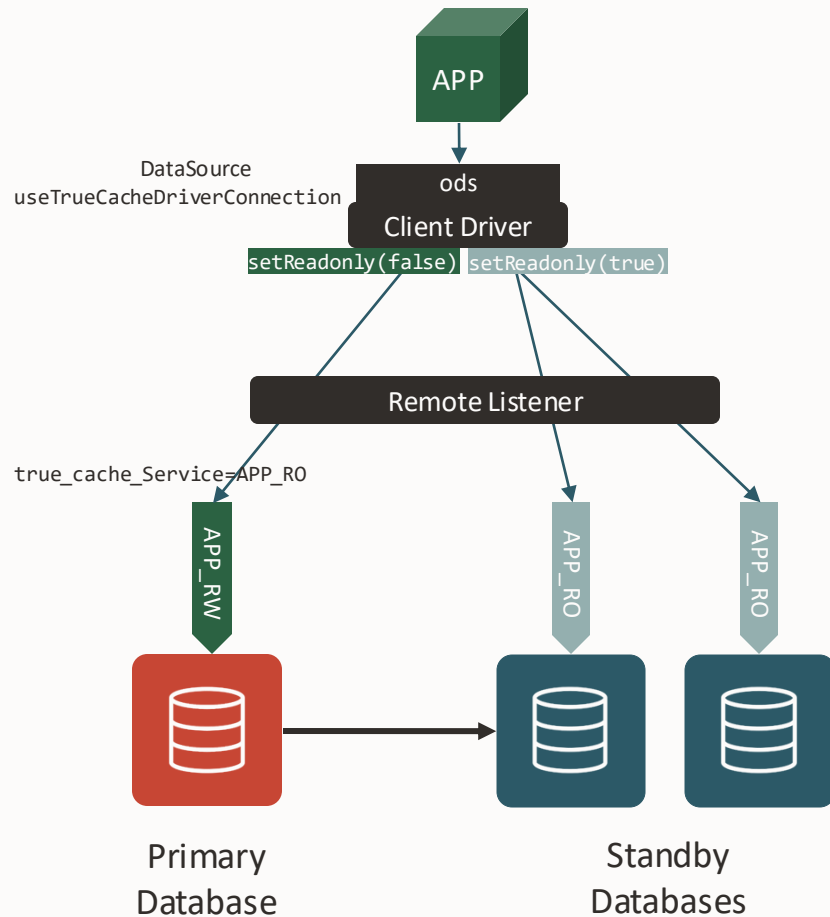
OracleDataSource ods_rw = new OracleDataSource();
ods_rw.setURL(url_app_rw);
...
try (Connection conn_rw = ods_rw.getConnection()) {
    // This is connected to the Primary
} catch (SQLException sqex) {
    ...
}

OracleDataSource ods_ro = new OracleDataSource();
ods_ro.setURL(url_app_ro);
...
try (Connection conn_ro = ods_ro.getConnection()) {
    // This is connected to the standby
} catch (SQLException sqex) {
    ...
}
```

- The application maintains two data sources
- The connection string and listeners load-balance the connections in the pool across the available instances and replicas
- Available with any driver

# Scaling out using the JDBC Thin True Cache driver

One connection pool with transparent automatic redirection



```
import oracle.jdbc.datasource.impl.OracleDataSource;

OracleDataSource ods = new OracleDataSource();
ods.setURL(url_app);
...
ods.setConnectionProperty(
    "oracle.jdbc.useTrueCacheDriverConnection", "true");
try (Connection conn = ods.getConnection()) {
    // This is connected to the Primary

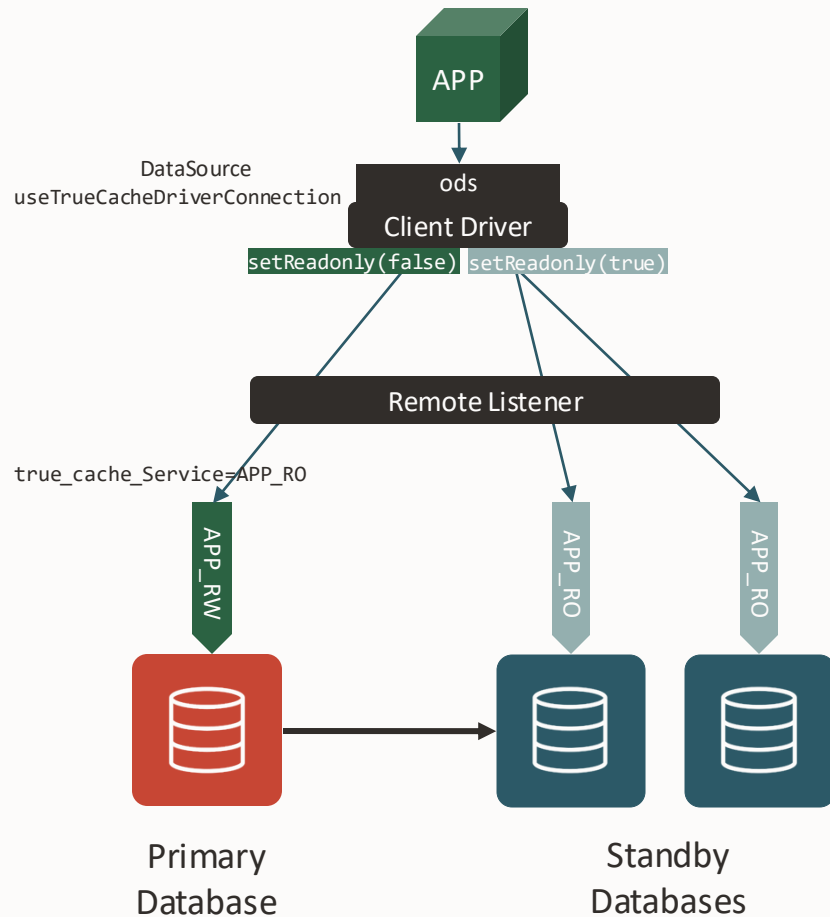
    conn.setReadOnly(true);
    // This is connected to the standby

} catch (SQLException sqex) {
    ...
}
```

- The application maintains **just one data source**
- The driver transparently redirects the ReadOnly context to the True Cache-enabled service
- The remote listener load balances the read-only connections
- Available for 23+ JDBC Thin drivers only

# Scaling out using the JDBC Thin True Cache driver

One connection pool with transparent automatic redirection



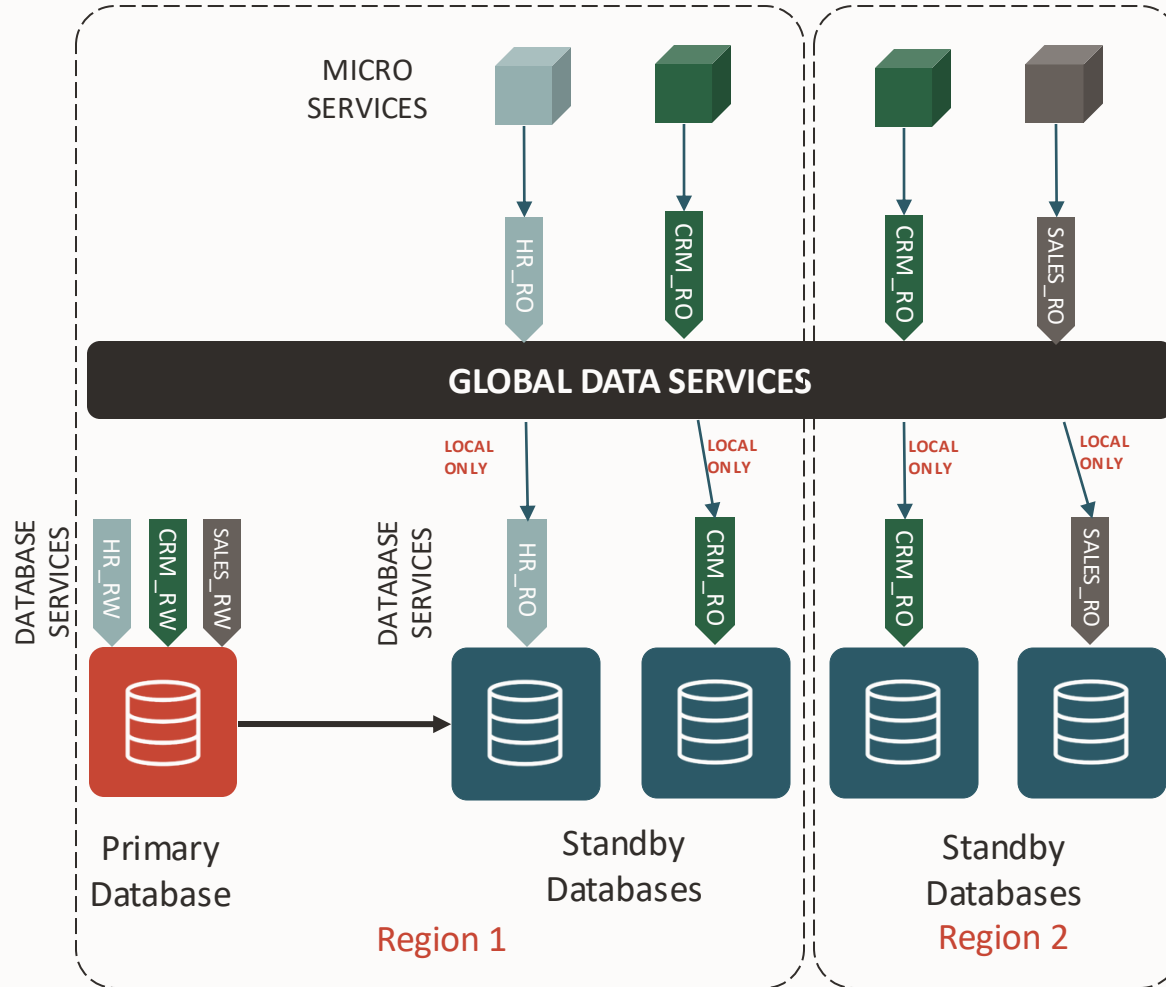
```
import oracle.jdbc.datasource.impl.OracleDataSource;

OracleDataSource ods = new OracleDataSource();
ods.setURL(url_app);
...
ods.setConnectionProperty(
    "oracle.jdbc.useTrueCacheDriverConnection", "true");
try (Connection conn = ods.getConnection()) {
    // This is connected to the Primary

    conn.setReadOnly(true);
    // This is connected to the standby
} catch (SQLException sqex) {
    ...
}
```

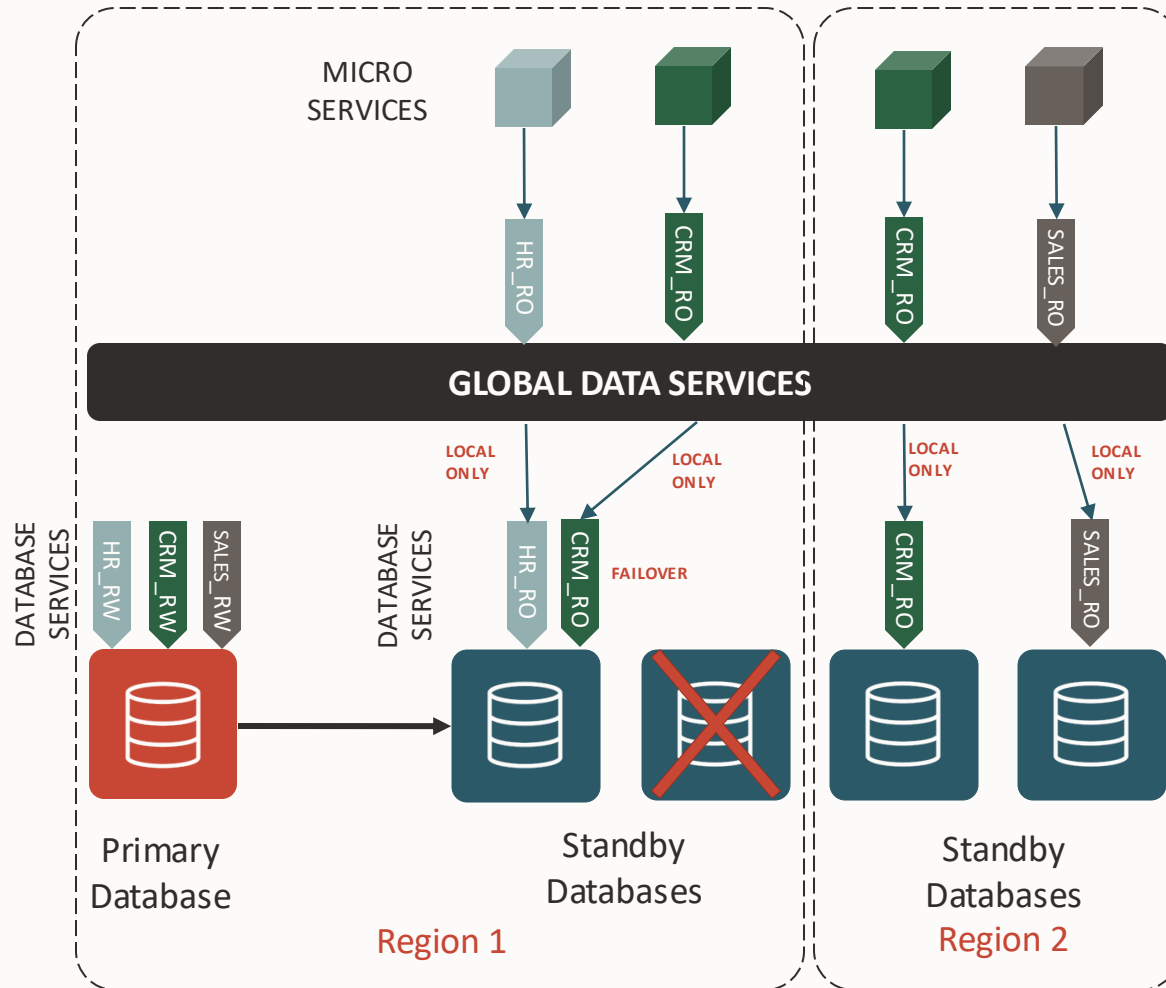
# Oracle **Active** Data Guard Global Data Services

# Oracle Global Data Services (GDS)



- Automatic and transparent client workload management across replicas
- Extends the concept of services across clusters
- Capabilities
  - Workload routing based on standby **load**, **locality**, or **lag**
  - Service failover across replicas
- Benefits
  - Maximize application performance
  - Mitigate downtime during planned and unplanned outages
  - Centrally manage services and resources of replicas

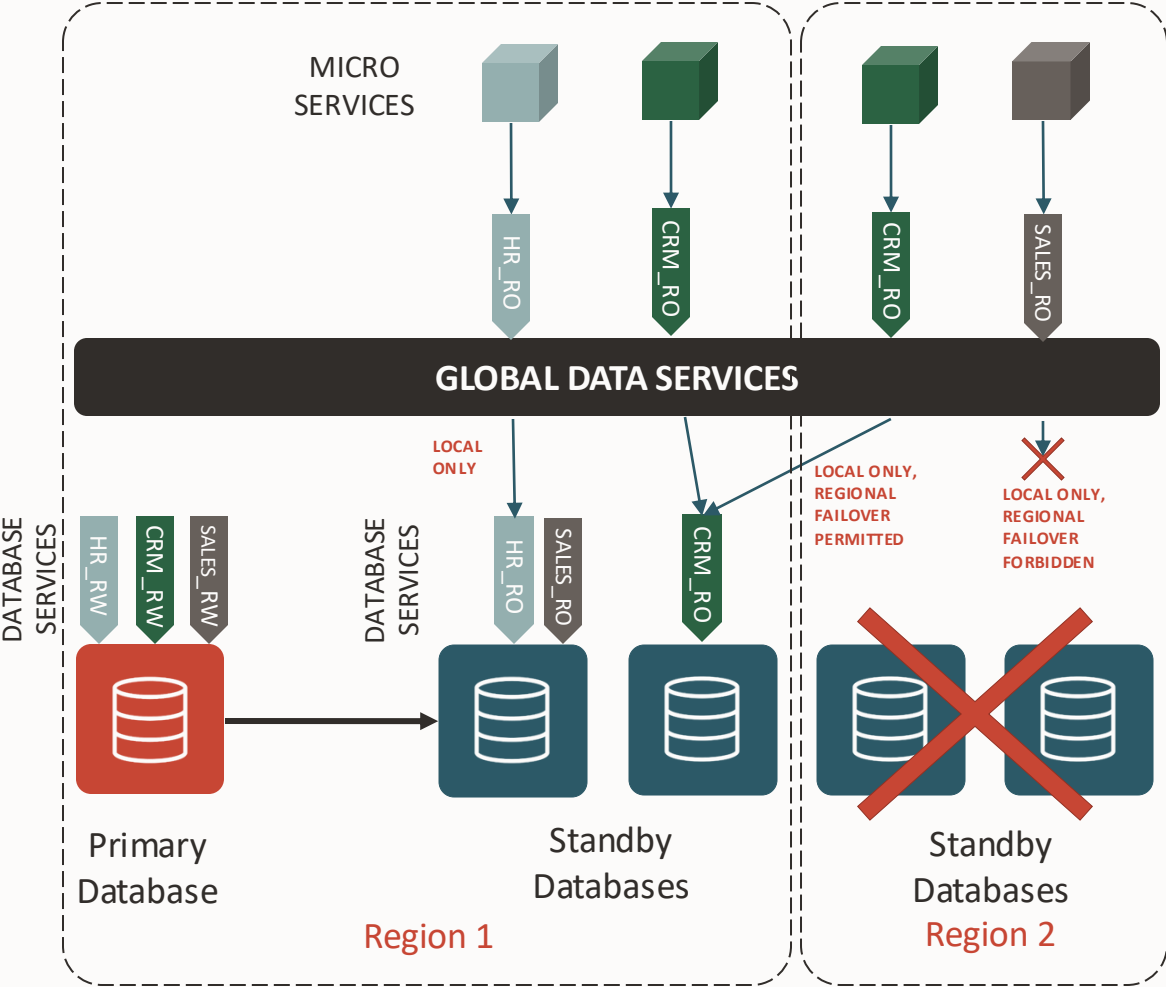
# Oracle Global Data Services (GDS)



- **Inter-database service failover**
  - If a cluster fails, the service is restarted automatically on another cluster
  - Clients reconnect automatically where the service is available
- Workload routing (region-based and lag-based)
- Load balancing (connect-time & run-time)



# Oracle Global Data Services (GDS)



- Region-based services can be configured to accept (fail over) deny requests from remote clients should a region become unavailable



# Oracle **Active** Data Guard DML Redirection

# Bigger Footprint of ADG Applications

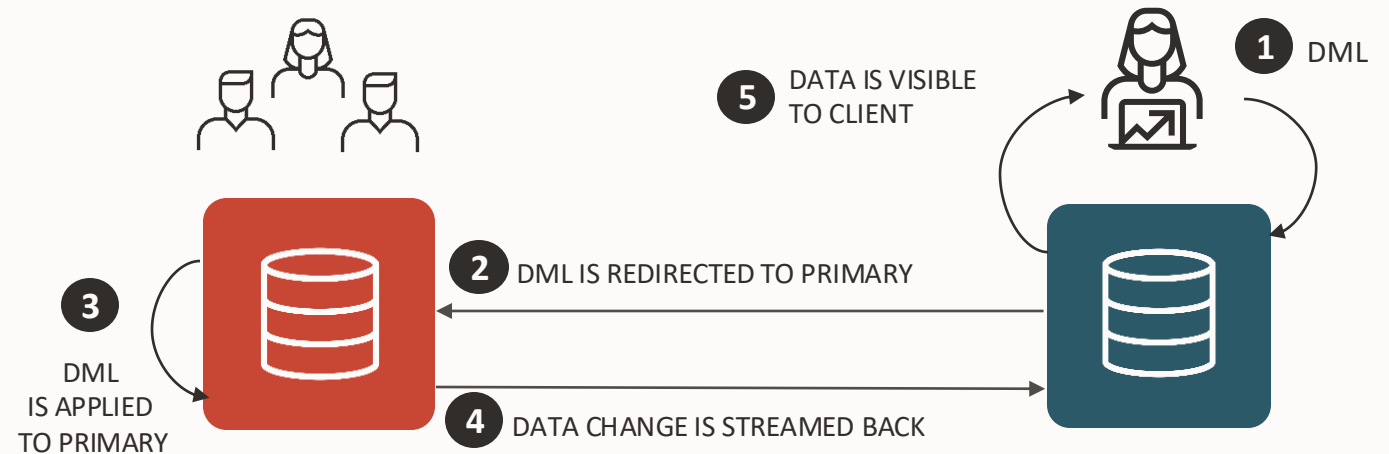
## DML on Active Data Guard

DML Re-direction is automatically performed from an Active Data Guard standby to the primary without compromising ACID compliance

- The parameter `ADG_REDIRECT_DML` controls DML Redirection
- `alter system set ADG_REDIRECT_DML | alter session enable ADG_REDIRECT_DML`
- The parameter `ADG_REDIRECT_PLSQL` controls PL/SQL redirection

Targeted for “Read-Mostly, **Occasional Updates**” applications

DDLs to create Global Temporary Tables are also transparently redirected



# Active Data Guard DML Replication

Easy and ready to use



By default DMLs are not possible on the standby

```
SQL> update hr.employee set salary=salary+100 where employee_id=1;  
ERROR at line 1:  
ORA-16000: database or pluggable database open for read-only access
```

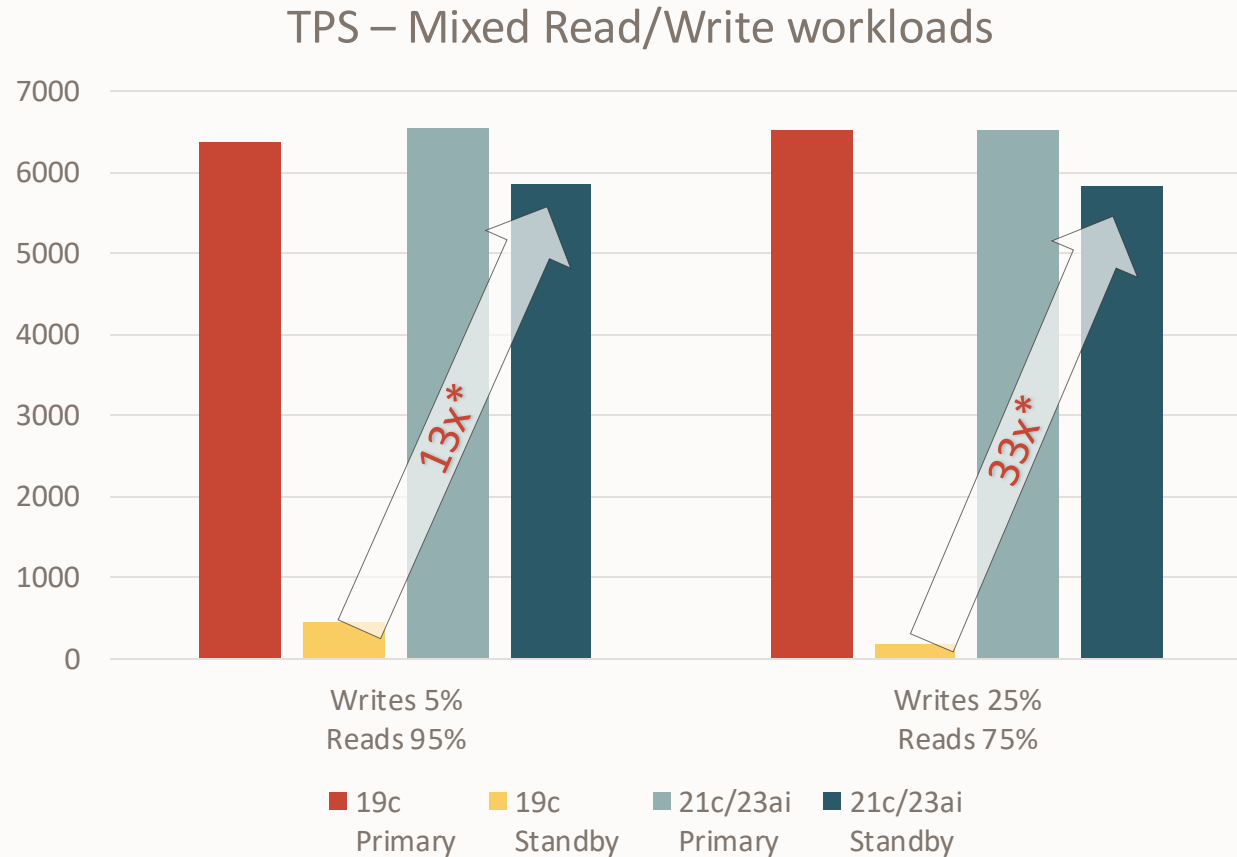
Enable DML redirection

```
SQL> alter session enable ADG_REDIRECT_DML;
```

DMLs work seamlessly

```
SQL> update hr.employee set salary=salary+100 where employee_id=1;  
1 row updated.  
SQL> commit;  
Commit complete.
```

# Improved Performance of Redirected Transactions



- 19c redirected statements wait for the DML to be applied on the standby before returning to the application.
  - Wait event: `"standby query scn advance"`
- From **21c** onwards, the statement returns as soon as it's executed on the primary.
  - The session waits only at commits or when the modified data is needed for consistent reads.
  - The non-documented parameter: `"_alter_adg_redirect_behavior"` can be set to `"sync_each_dml"` to restore the previous behavior.

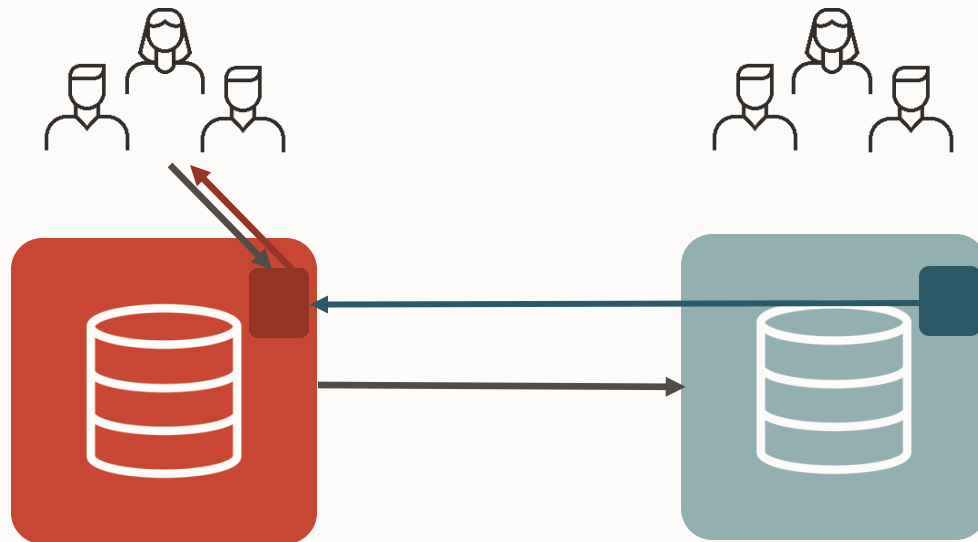
\* 16 concurrent Order Entry sessions simulated with Swingbench with mixed 'NewCustomerProcess' and 'BrowseProducts' transactions.

# Oracle **Active** Data Guard Automatic Block Repair

# Oracle Active Data Guard Automatic Block Repair

## Transparently repairs corrupted blocks

- Oracle detects if a block is corrupted when reading it
- The corruption is automatically repaired using a good copy
  - From the standby when the corruption is on the primary
  - From the primary when the corruption is on the standby



This statistic in v\$sysstat shows the recovered blocks:  
db corrupt blocks recovered



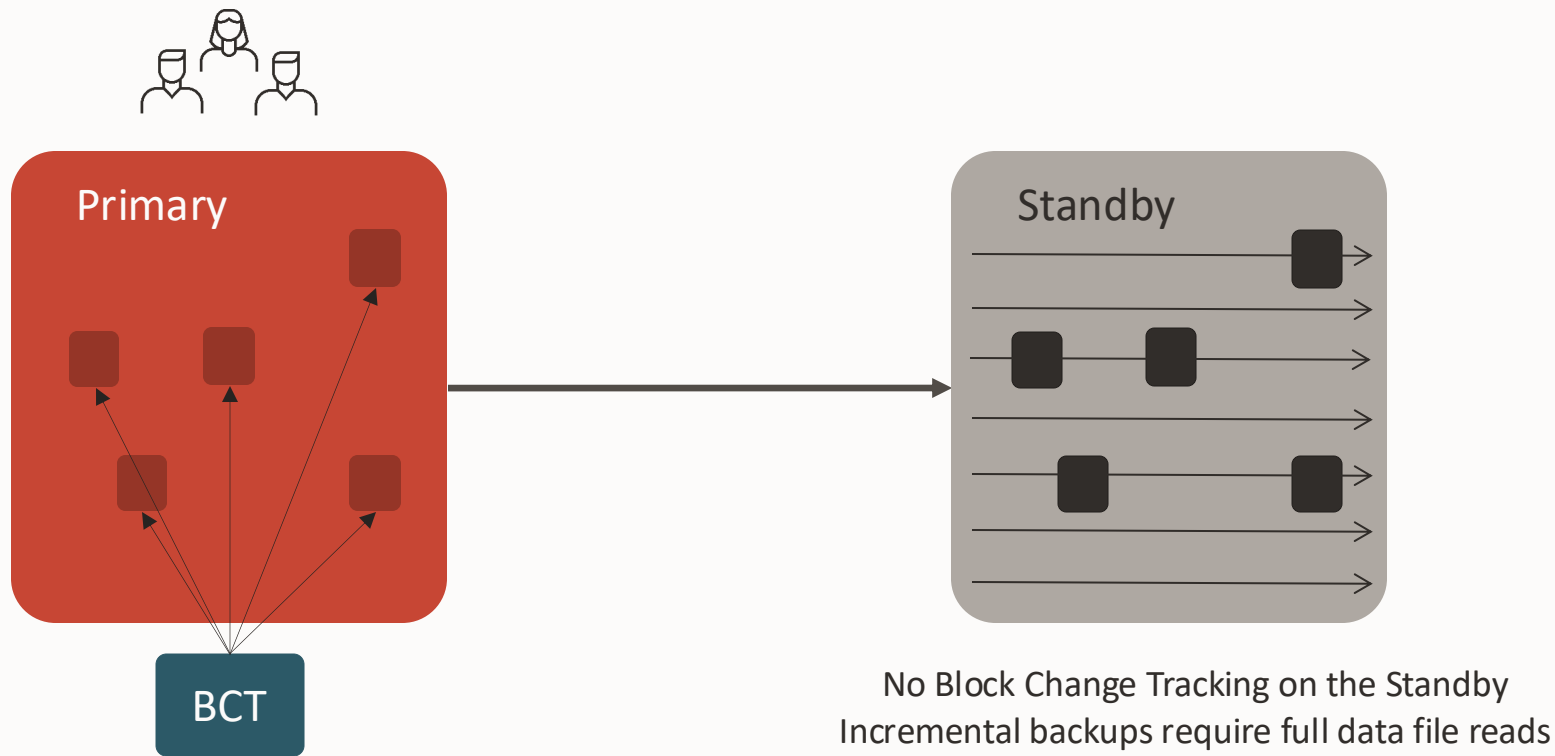
# Oracle **Active** Data Guard

## Fast Incremental Backup on Physical Standby



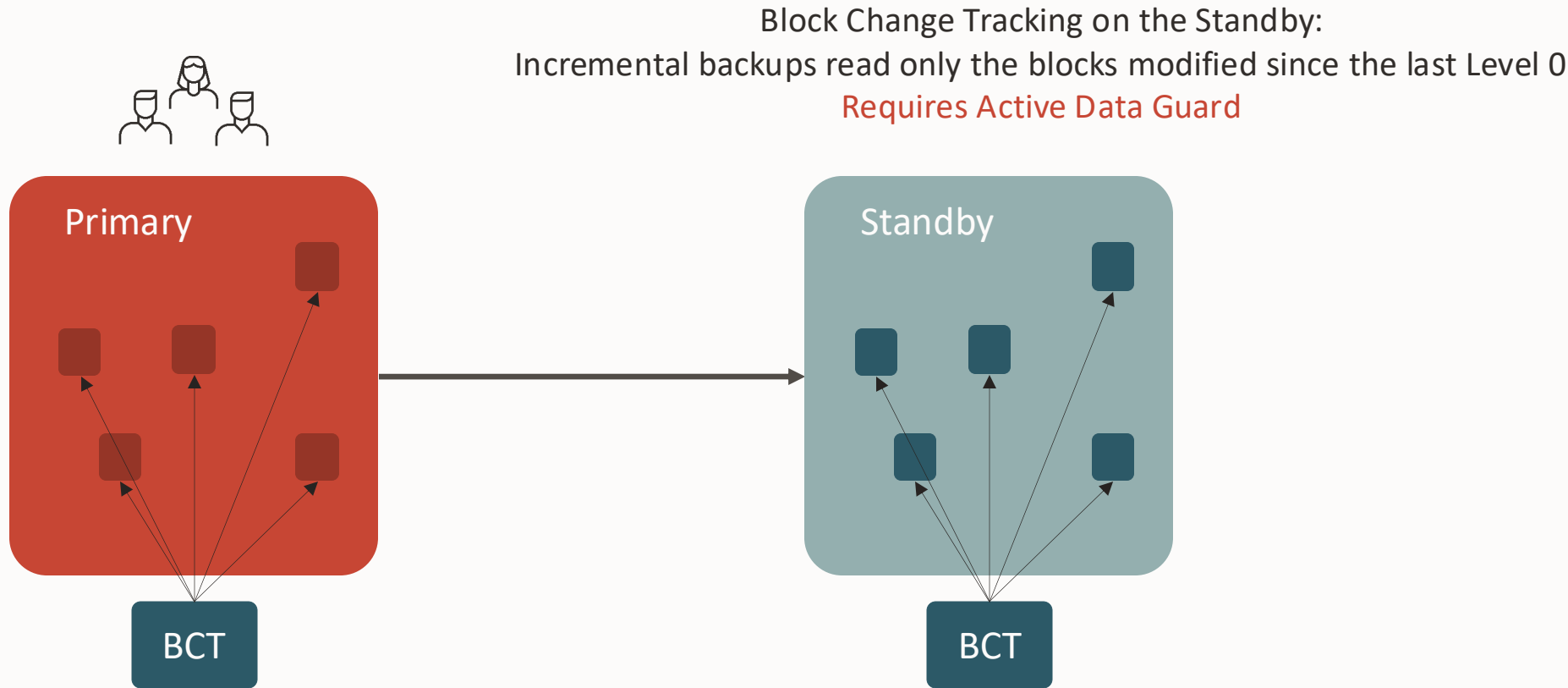
# Fast Incremental Backup on Physical Standby

Enable the Block Change Tracking to speed up backups and avoid unnecessary I/O



# Fast Incremental Backup on Physical Standby

Enable the Block Change Tracking to speed up backups and avoid unnecessary I/O



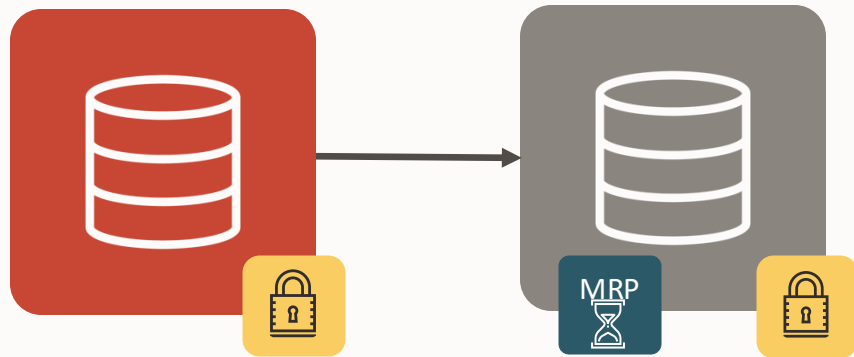
# Oracle **Active** Data Guard Online Tablespace Encryption

# Oracle Data Guard Online Tablespace Encryption before 23ai

The recovery pauses while the standby is encrypted



```
SQL> -- on the primary database
SQL> ALTER TABLESPACE bigtbsp ENCRYPTION ONLINE ENCRYPT;
```



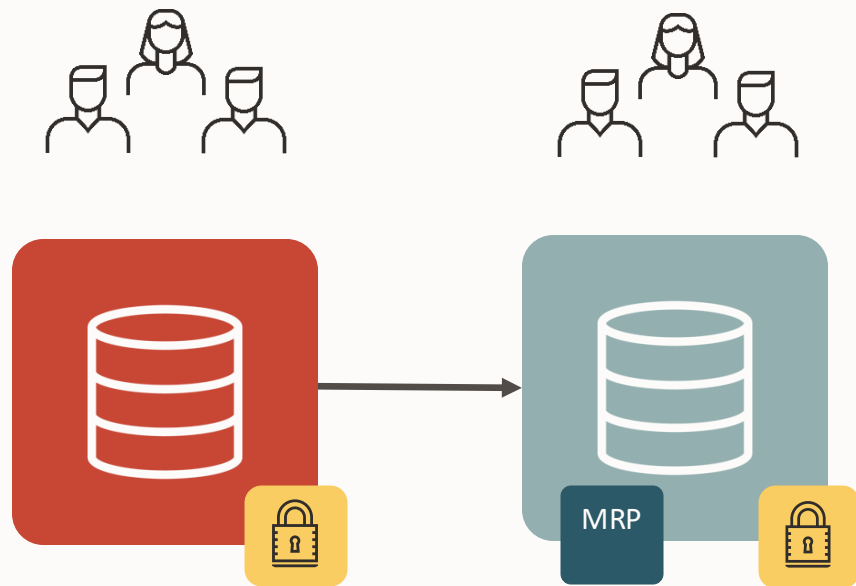
- 1 The tablespace encryption starts on the primary.
- 2 The recovery process "sees" the encryption command.
- 3 The recovery process suspends the recovery.
- 4 The recovery process starts encrypting the tablespace on the standby.
- 5 When the standby encryption is over, the recovery resumes.



Very large databases (100s of TB) can accumulate days of apply lag.

# Oracle Active Data Guard Online Tablespace Encryption before 23ai

Encrypt primary and standby databases online and without recovery lag



```
SQL> -- on the standby database
```

```
SQL> ALTER SYSTEM SET "_AUTO_REKEY_DURING_MRCV"=FALSE;
```

```
SQL> -- on the primary database
```

```
SQL> ALTER TABLESPACE bigtbsp ENCRYPTION ONLINE ENCRYPT;
```

- 1 The tablespace encryption starts on the primary.
- 2 The recovery process marks the tablespace as ENCRYPTING on the standby.
- 3 The recovery continues without interruptions.

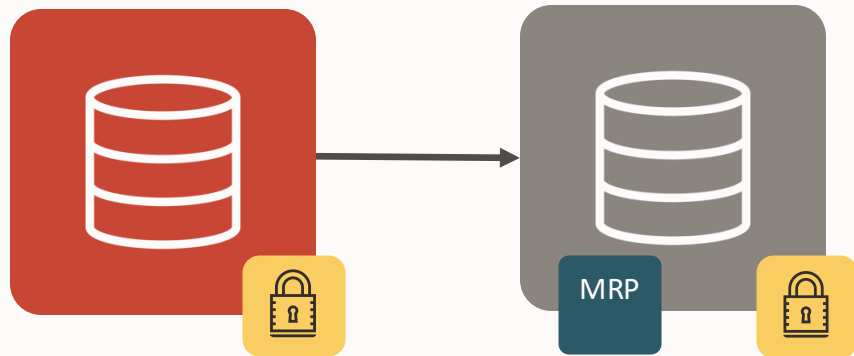
```
SQL> -- on the standby database with PDB open read-only
```

```
SQL> ALTER TABLESPACE bigtbsp ENCRYPTION ONLINE FINISH ENCRYPT;
```

- 4 The user process encrypts the tablespace online on the standby.

# Oracle Data Guard Online Tablespace Encryption

Encrypt primary and standby databases online and without recovery lag



```
SQL> -- on the standby database
SQL> ALTER SYSTEM SET DB_RECOVERY_AUTO_REKEY=OFF;

SQL> -- on the primary database
SQL> ALTER TABLESPACE bigtbsp ENCRYPTION ONLINE ENCRYPT;
```

- 1 The tablespace encryption starts on the primary.
- 2 The recovery process marks the tablespace as ENCRYPTING on the standby.
- 3 The recovery continues without interruptions.

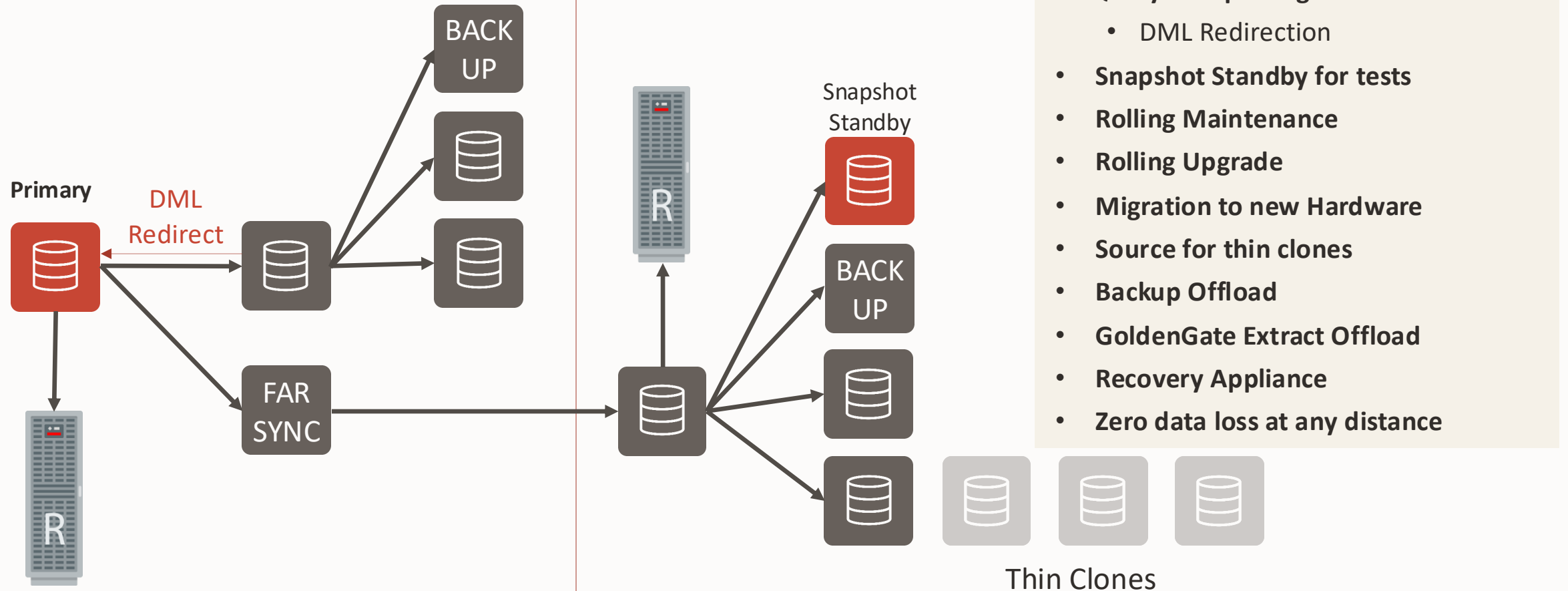
```
SQL> -- on the standby database with PDB mounted or open
SQL> ALTER TABLESPACE bigtbsp ENCRYPTION ONLINE FINISH ENCRYPT;
```

- 4 The user process encrypts the tablespace online on the standby.

# Oracle **Active** Data Guard Real-Time Cascade Standbys

# Active Data Guard: up to 30 direct standbys and 253 total members

Far Sync and Cascading Standby open endless possibilities





# Active Data Guard Real-Time Cascade Standby

Offload multiple redo transports to a first-level standby

BOSTON



RedoRoutes=

```
(LOCAL : ( NASHUA SYNC PRIORITY=1, NEWYORK ASYNC PRIORITY=8, NEWARK ASYNC PRIORITY=8 ))  
(NASHUA : NEWYORK ASYNC, NEWARK ASYNC ) )
```

NASHUA



RedoRoutes=

```
(LOCAL : ( BOSTON SYNC PRIORITY=1, NEWYORK ASYNC PRIORITY=8, NEWARK ASYNC PRIORITY=8 ))  
(BOSTON : NEWYORK ASYNC, NEWARK ASYNC ) )
```

NEWYORK



NEWARK

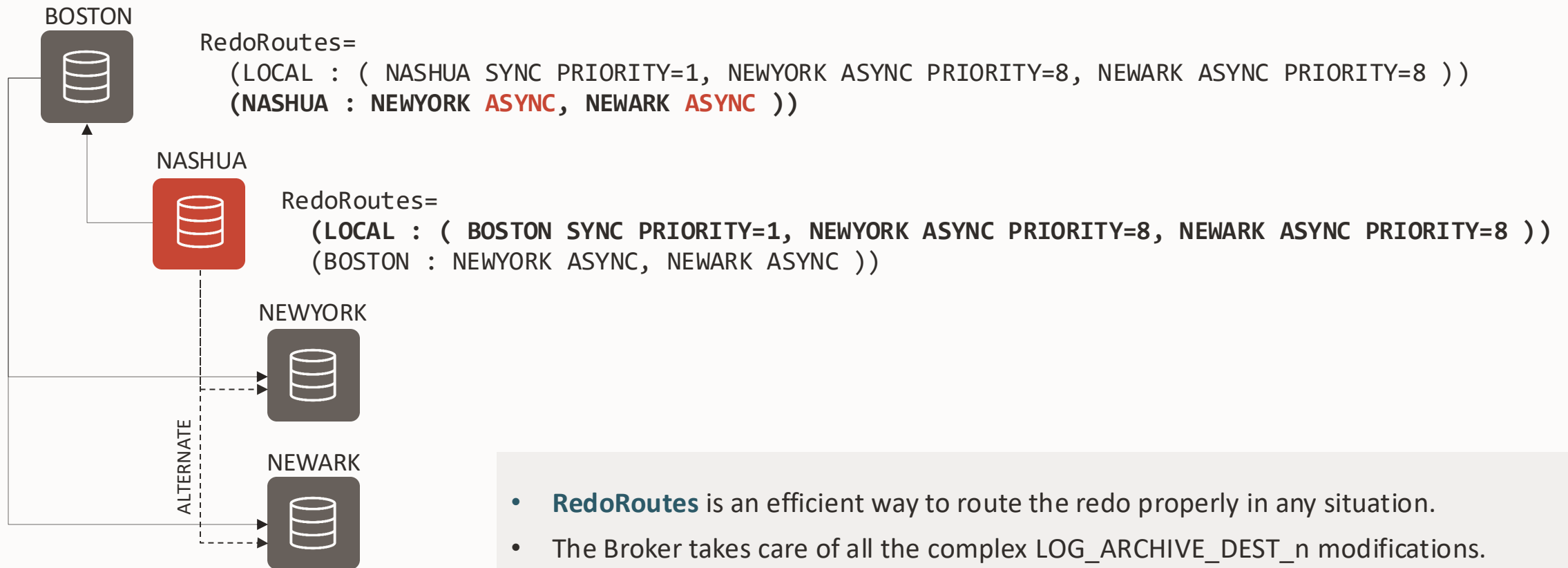


ALTERNATE

- Explicit “ASYNC” in the cascading member means “Real-Time Cascade”. Such configuration requires **Active Data Guard**.
- If not specified, the redo is shipped at log switch.

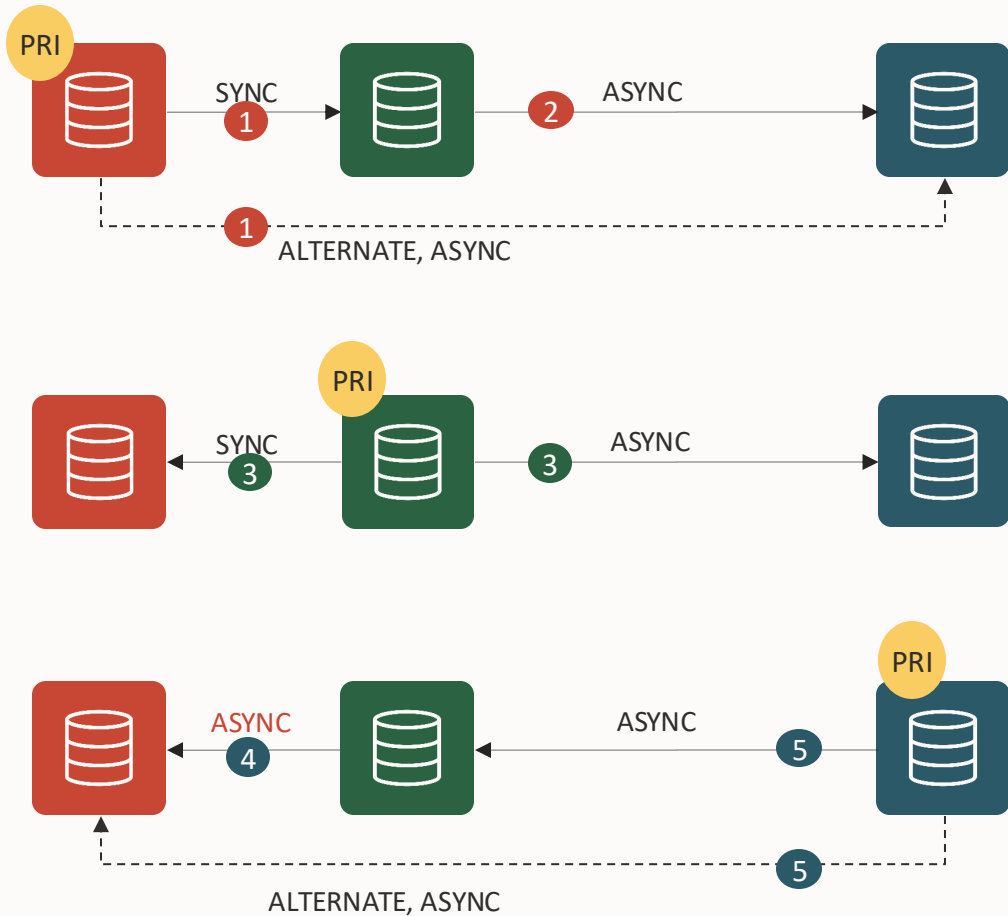
# Active Data Guard Real-Time Cascade Standby

Offload multiple redo transports to a first-level standby



# Understanding RedoRoutes

Start by drawing all the permutations and describe what you want



## Where does RED send the redo?

- 1 When RED is primary:
  - to GREEN (SYNC)
  - or BLUE (ASYNC) if GREEN is not available

## Where does GREEN send the redo?

- 2 When RED is primary:
  - to BLUE (ASYNC) (real-time cascade)
- 3 When GREEN is primary:
  - to RED (SYNC)
  - and to BLUE (ASYNC)
- 4 When BLUE is primary:
  - to RED (ASYNC) (real-time cascade)


## Where does BLUE send the redo?


- 5 When BLUE is primary:
  - to GREEN (ASYNC)
  - or RED (ASYNC) if GREEN is not available

R  
e  
d  
o  
R  
o  
u  
t  
e  
s




# Understanding RedoRoutes


RedoRoutes tells where a DB (or Far Sync) should send the redo, depending on the primary


EDIT DATABASE RED SET PROPERTY RedoRoutes =  Where does RED send the redo?

- ① (RED: ( GREEN SYNC PRIORITY=1, BLUE ASYNC PRIORITY=2 ))  ① When RED is primary:
- to GREEN (SYNC)  
or BLUE (ASYNC) if GREEN is not available

EDIT DATABASE GREEN SET PROPERTY RedoRoutes =  Where does GREEN send the redo?

- ② (RED: BLUE ASYNC)  ② When RED is primary:
- to BLUE (ASYNC) (real-time cascade)
- ③ (GREEN: RED SYNC, BLUE ASYNC)  ③ When GREEN is primary:
- to RED (SYNC)  
and to BLUE (ASYNC)
- ④ (BLUE: RED ASYNC)  ④ When BLUE is primary:
- to RED (ASYNC) (real-time cascade)

EDIT DATABASE BLUE SET PROPERTY RedoRoutes =  Where does BLUE send the redo?

- ⑤ (BLUE: ( GREEN ASYNC PRIORITY=1, RED ASYNC PRIORITY=2 ))  ⑤ When BLUE is primary:
- to GREEN (ASYNC)  
or RED (ASYNC) if GREEN is not available

# Verifying the RedoRoutes configuration

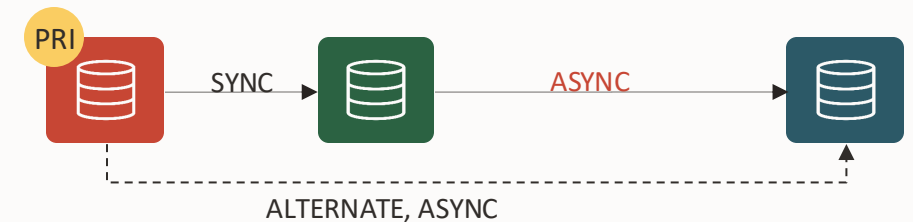
```
EDIT DATABASE RED SET PROPERTY RedoRoutes = '(RED: ( GREEN SYNC PRIORITY=1, BLUE ASYNC PRIORITY=2 ))';  
EDIT DATABASE GREEN SET PROPERTY RedoRoutes = '(RED:BLUE ASYNC)(GREEN:RED SYNC,BLUE ASYNC)(BLUE:RED ASYNC)';  
EDIT DATABASE BLUE SET PROPERTY RedoRoutes = '(BLUE: ( GREEN ASYNC PRIORITY=1, RED ASYNC PRIORITY=2 ))';
```

```
DGMGRL> show configuration when primary is red ;
```

Configuration when red is primary - redoroutes\_demo

Members:

- red - Primary database
- green - Physical standby database
- blue - Physical standby database (receiving current redo)
- blue - Physical standby database (alternate of green)



# Verifying the RedoRoutes configuration



```
EDIT DATABASE RED SET PROPERTY RedoRoutes = '(RED: ( GREEN SYNC PRIORITY=1, BLUE ASYNC PRIORITY=2 ))';  
EDIT DATABASE GREEN SET PROPERTY RedoRoutes = '(RED:BLUE ASYNC)(GREEN:RED SYNC,BLUE ASYNC)(BLUE:RED ASYNC)';  
EDIT DATABASE BLUE SET PROPERTY RedoRoutes = '(BLUE: ( GREEN ASYNC PRIORITY=1, RED ASYNC PRIORITY=2 ))';
```

```
DGMGRL> show configuration when primary is green ;
```

Configuration when green is primary - redoroutes\_demo

Members:

green - Primary database

red - Physical standby database

blue - Physical standby database



# Verifying the RedoRoutes configuration

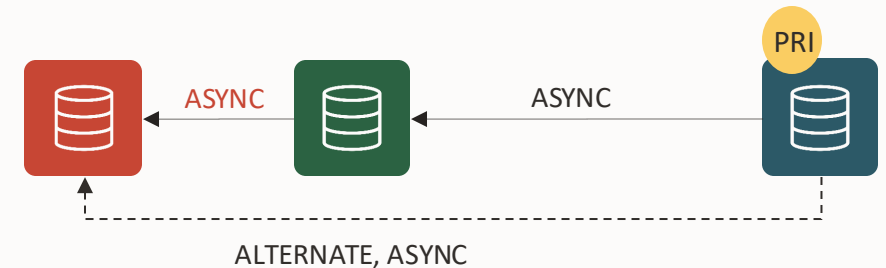
```
EDIT DATABASE RED SET PROPERTY RedoRoutes = '(RED: ( GREEN SYNC PRIORITY=1, BLUE ASYNC PRIORITY=2 ))';  
EDIT DATABASE GREEN SET PROPERTY RedoRoutes = '(RED:BLUE ASYNC)(GREEN:RED SYNC,BLUE ASYNC)(BLUE:RED ASYNC)';  
EDIT DATABASE BLUE SET PROPERTY RedoRoutes = '(BLUE: ( GREEN ASYNC PRIORITY=1, RED ASYNC PRIORITY=2 ))';
```

```
DGMGRL> show configuration when primary is blue ;
```

Configuration when blue is primary - redoroutes\_demo

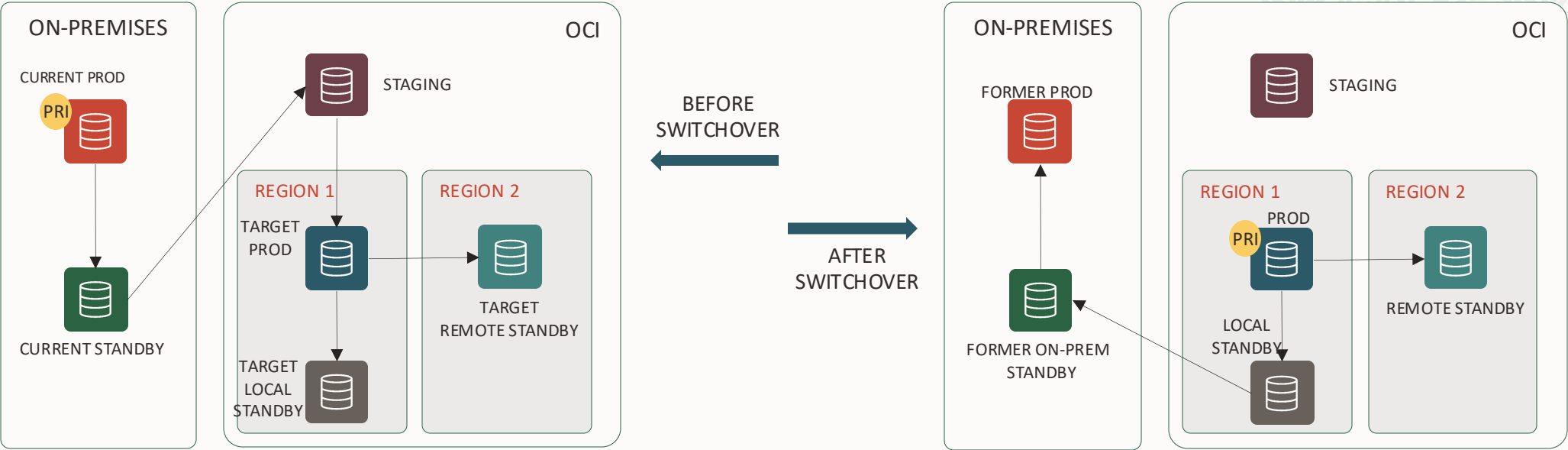
Members:

- blue - Primary database
- green - Physical standby database
- red - Physical standby database (receiving current redo)
- red - Physical standby database (alternate of green)



# Real Life Use Case #1

## Highly available migration to the Oracle Cloud Infrastructure



```
DGMGRL> show configuration when primary is red;
```

Configuration when blue is primary - redoroutes\_demo

Members:

- red - Primary database
- green - Physical standby database
- purple - Physical standby database (receiving current redo)
- blue - Physical standby database (receiving current redo)
- gray - Physical standby database (receiving current redo)
- turquoise - Physical standby database (receiving current redo)

```
DGMGRL> show configuration when primary is blue ;
```

Configuration when blue is primary - redoroutes\_demo

Members:

- blue - Primary database
- turquoise - Physical standby database
- gray - Physical standby database
- green - Physical standby database (receiving current redo)
- red - Physical standby database (receiving current redo)

Members Not Receiving Redo:

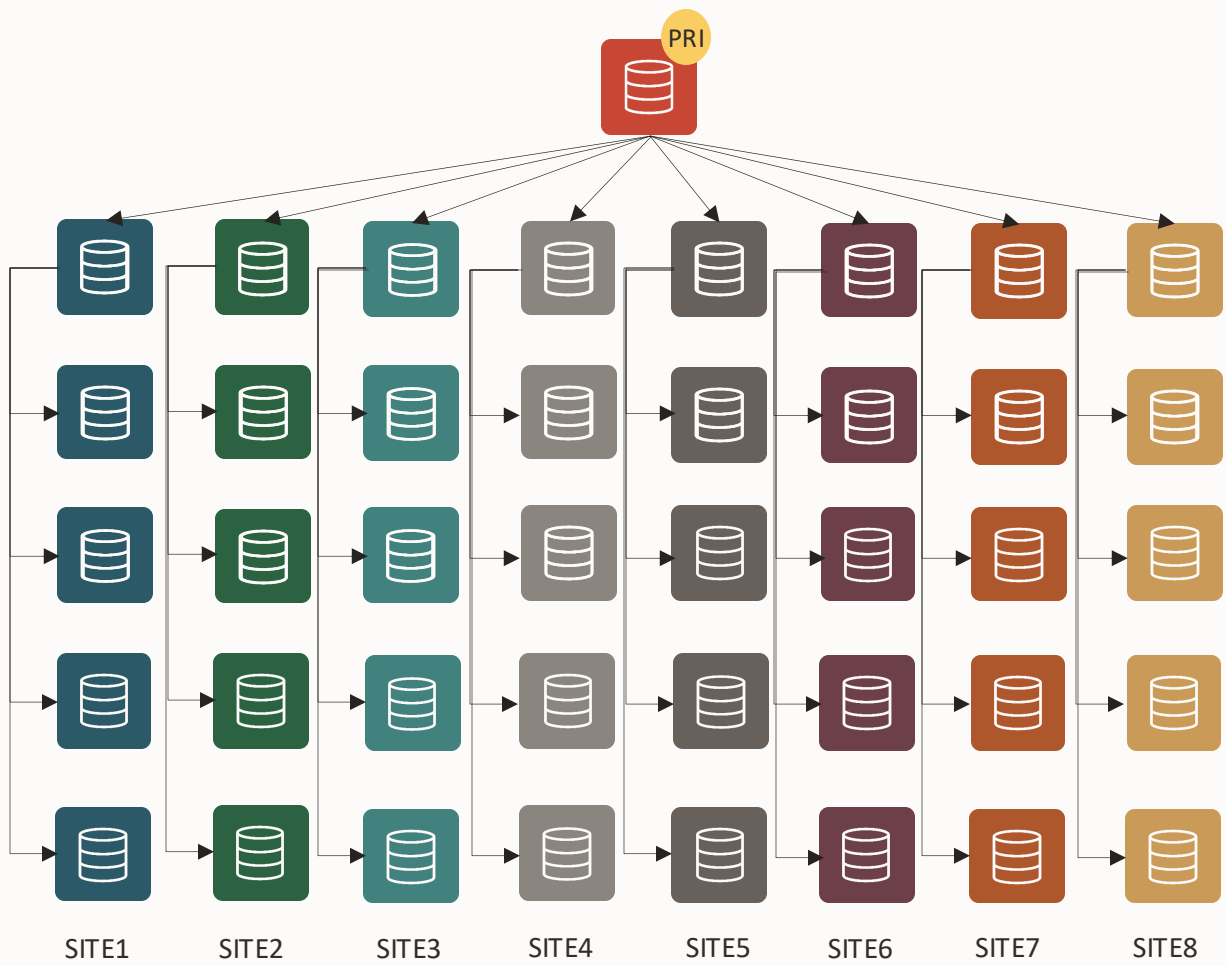
- purple - Physical standby database





# Real Life Use Case #2

Read-only farm for intensive, latency-sensitive workloads

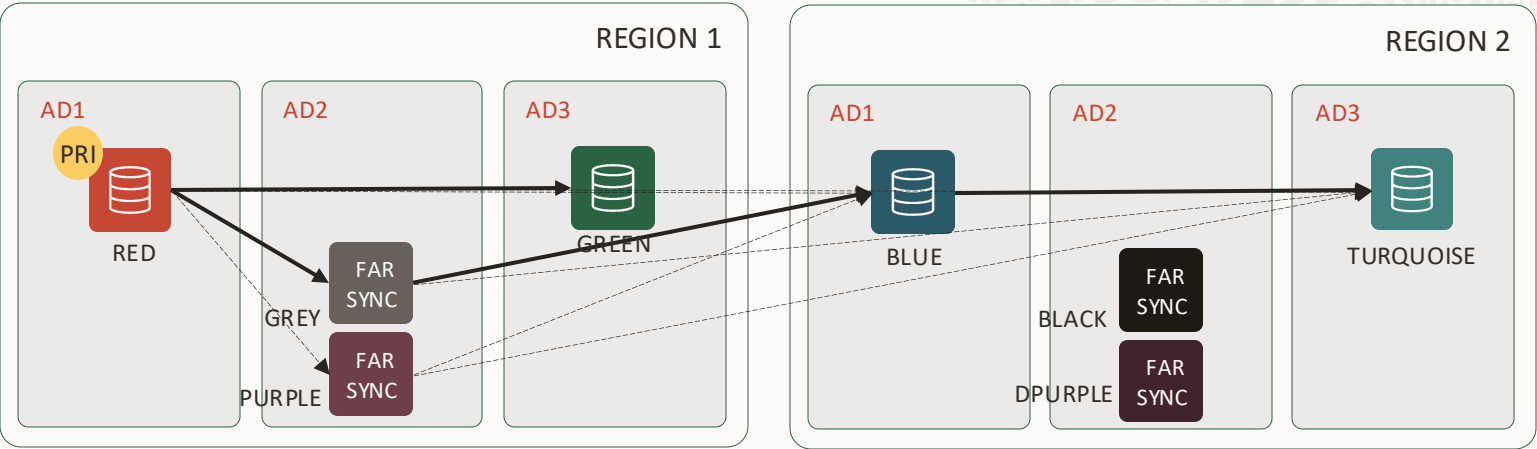


- Members:
- prod** - Primary database
  - site1** - Physical standby database
    - site101 - Physical standby database (receiving current redo)
    - site102 - Physical standby database (receiving current redo)
    - site103 - Physical standby database (receiving current redo)
    - site104 - Physical standby database (receiving current redo)
  - site2** - Physical standby database
    - site201 - Physical standby database (receiving current redo)
    - site202 - Physical standby database (receiving current redo)
    - site203 - Physical standby database (receiving current redo)
    - site204 - Physical standby database (receiving current redo)
  - site3** - Physical standby database
    - site301 - Physical standby database (receiving current redo)
    - site302 - Physical standby database (receiving current redo)
    - site303 - Physical standby database (receiving current redo)
    - site304 - Physical standby database (receiving current redo)
  - site4** - Physical standby database
    - site401 - Physical standby database (receiving current redo)
    - site402 - Physical standby database (receiving current redo)
    - site403 - Physical standby database (receiving current redo)
    - site404 - Physical standby database (receiving current redo)
  - site5** - Physical standby database
    - site501 - Physical standby database (receiving current redo)
    - site502 - Physical standby database (receiving current redo)
    - site503 - Physical standby database (receiving current redo)
    - site504 - Physical standby database (receiving current redo)
  - site6** - Physical standby database
    - site601 - Physical standby database (receiving current redo)
    - site602 - Physical standby database (receiving current redo)
    - site603 - Physical standby database (receiving current redo)
    - site604 - Physical standby database (receiving current redo)
  - site7** - Physical standby database
    - site701 - Physical standby database (receiving current redo)
    - site702 - Physical standby database (receiving current redo)
    - site703 - Physical standby database (receiving current redo)
    - site704 - Physical standby database (receiving current redo)
  - site8** - Physical standby database
    - site801 - Physical standby database (receiving current redo)
    - site802 - Physical standby database (receiving current redo)
    - site803 - Physical standby database (receiving current redo)
    - site804 - Physical standby database (receiving current redo)



# Real Life Use Case #3

## Highly available cloud blueprint for multi-AD regions

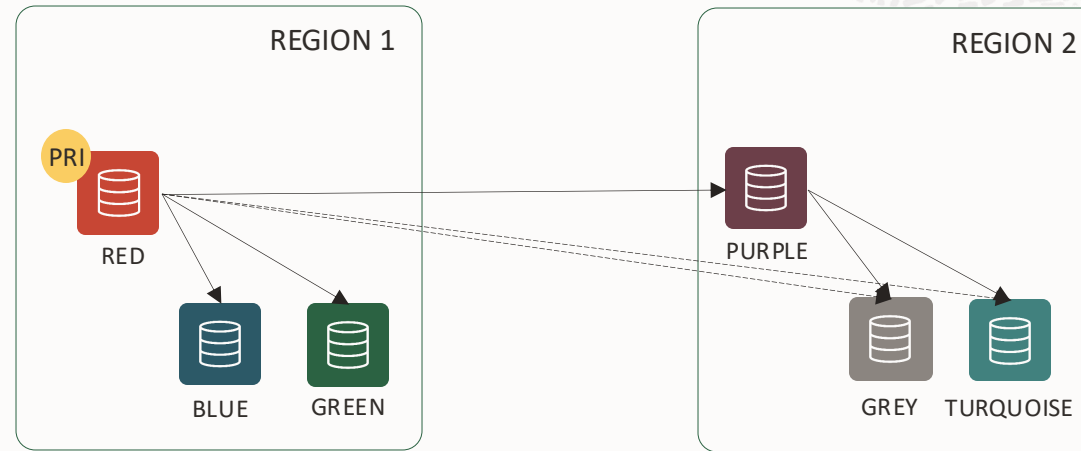


```
DGMGRL> show configuration when primary is RED;
Configuration - HADB
Protection Mode: MaxAvailability
Members:
RED      - Primary database
GREEN    - Physical standby database
GREY     - Far sync instance
BLUE     - Physical standby database
          TURQUOISE - Physical standby database (receiving current redo)
          TURQUOISE - Physical standby database (alternate of BLUE)
PURPLE   - Far sync instance (alternate of GREY)
BLUE     - Physical standby database
          TURQUOISE - Physical standby database (receiving current redo)
          TURQUOISE - Physical standby database (alternate of BLUE)
BLUE     - Physical standby database (alternate of GREY)
          TURQUOISE - Physical standby database (receiving current redo)
          TURQUOISE - Physical standby database (alternate of BLUE)
Members Not Receiving Redo:
BLACK    - Far sync instance
DPURPLE  - Far sync instance
```



## Real Life Use Case #4

Two local Active Data Guard standbys and a symmetric region for DR



```
DGMGRL> show configuration
```

Configuration - HADB

Protection Mode: MaxPerformance

Members:

RED - Primary database

GREEN - (\*) Physical standby database

BLUE - Physical standby database

PURPLE - Physical standby database

TURQUOISE - Physical standby database (receiving current redo)

GREY - Physical standby database (receiving current redo)

Fast-Start Failover: Enabled in Potential Data Loss Mode

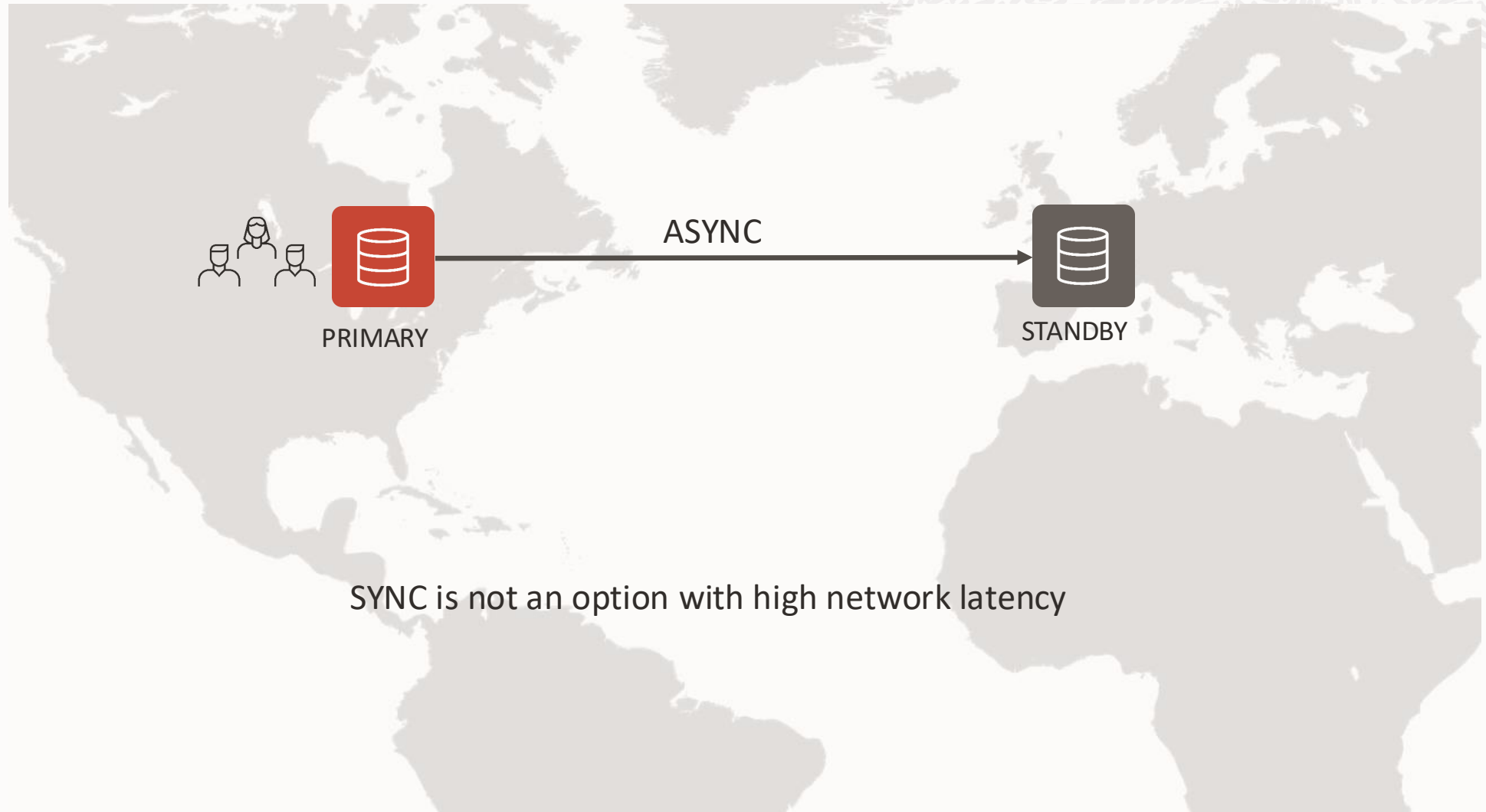
Configuration Status:

SUCCESS (status updated 29 seconds ago)

# Oracle **Active** Data Guard Far Sync

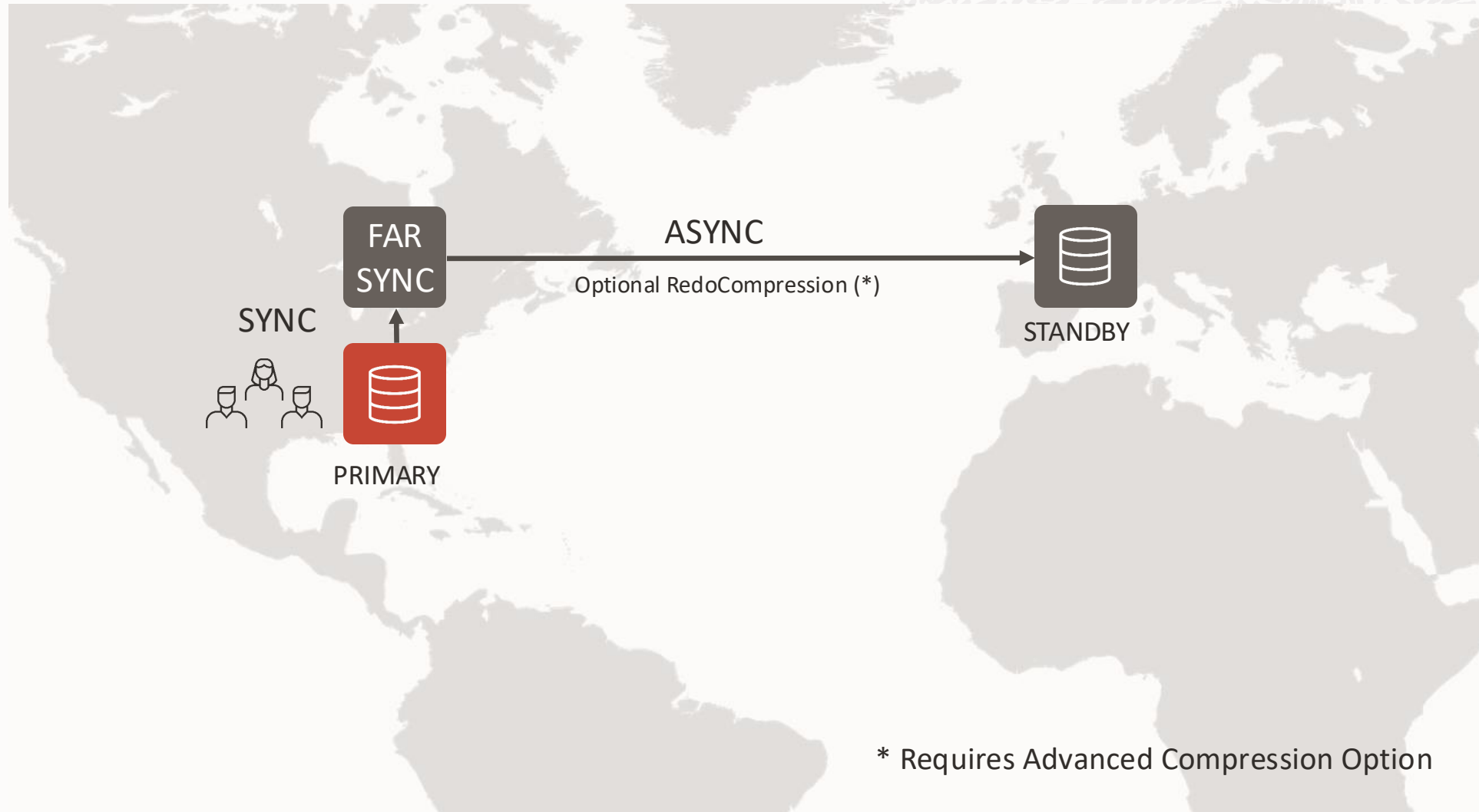
# The Zero Data Loss Challenge

## Trade-off Performance for Protection



# Active Data Guard Far Sync

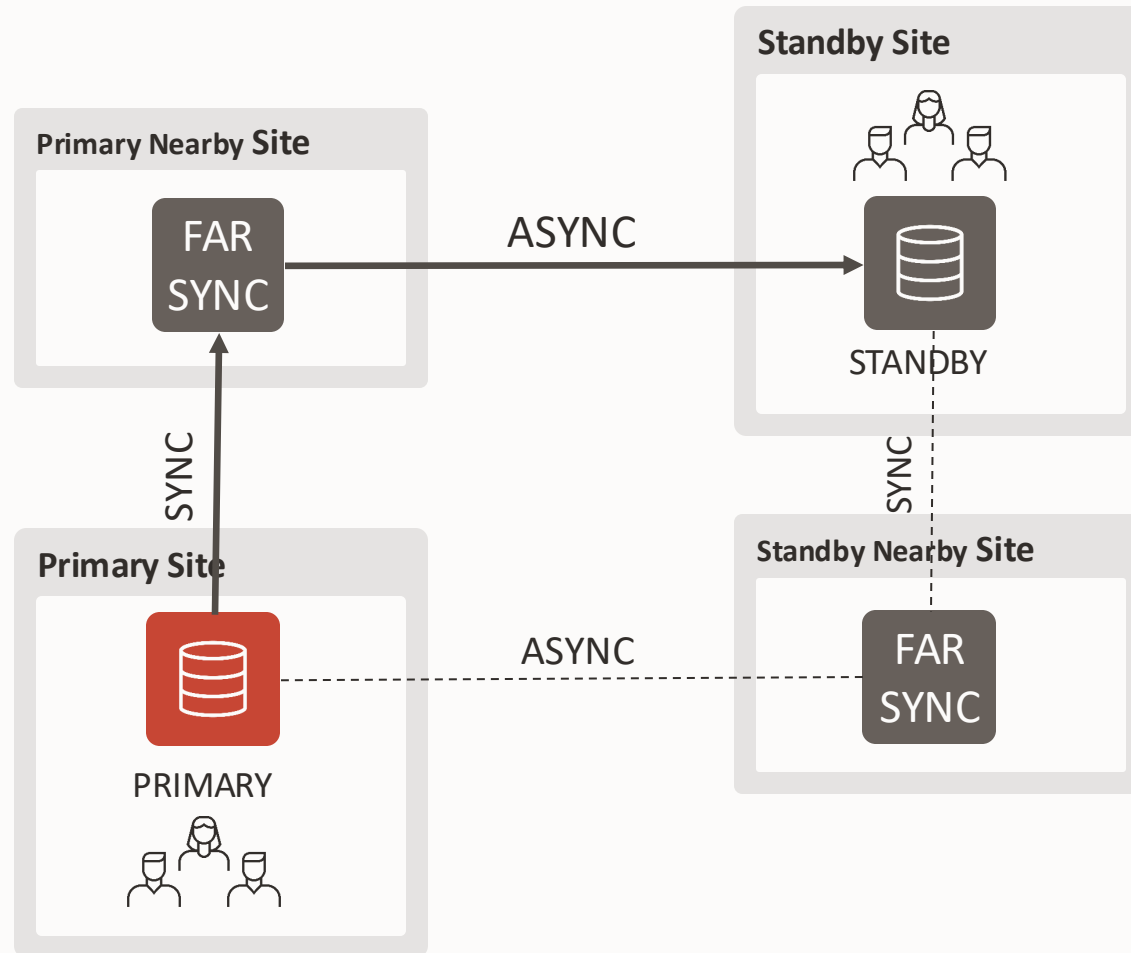
Zero Data Loss Protection at Any Distance



\* Requires Advanced Compression Option

# Active Data Guard Far Sync

Do not Trade-off Protection for Performance



## Far Sync

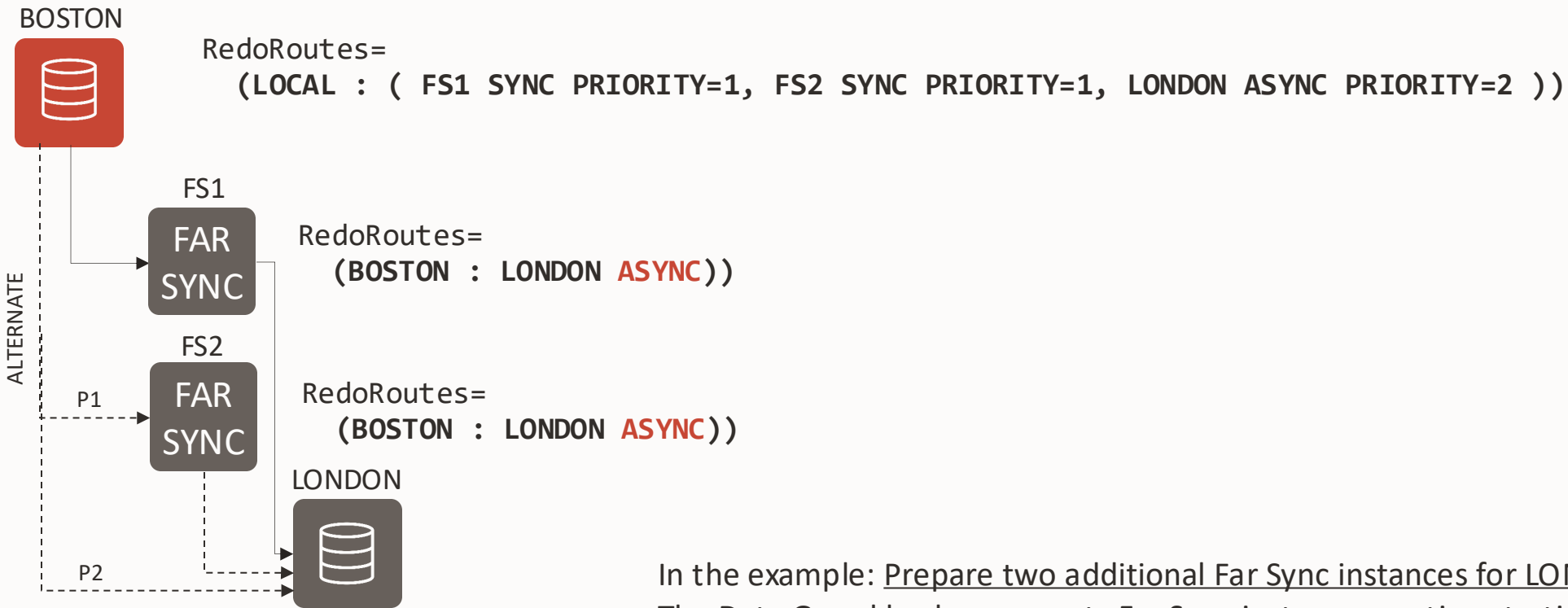
- Special instance:
  - No datafiles
  - No Media Recovery
  - Only control files, archives and standby logs
- Up to 30 direct destinations
- Offload transport compression (Advanced Compression)
- Supports FSFO in MaxAvailability
- Supports FSFO in MaxPerformance (**new in 21c**)

## Use different Datacenters or Availability Domains!

- Upon failover, the standby will fetch the very last redo from the Far Sync

# Active Data Guard Far Sync

## Use RedoRoutes for Far Sync High Availability



In the example: Prepare two additional Far Sync instances for LONDON!  
The Data Guard broker supports Far Sync instance creation starting with 21c.





# Benefits and Downsides of Far Sync

## When to consider Far Sync?

### Benefits

- Increased **performance** for existing Sync configurations
- Increased **protection** for existing Async configurations
- Zero Data Loss (Max Availability) across distant regions
- Fully integrated with the broker
- Automatic gap resolution through the Far Sync

### Downsides

- Additional server(s) or VM(s) and components
- A Far Sync co-located with the primary might not prevent data loss in case of full site failure



# Far Sync and Fast Start Failover

Which Fast Start Failover protection modes are compatible with Far Sync?

FSFO and FAR SYNC	Maximum Performance	Maximum Availability	Maximum Protection
ASync	✓ (21c+)	✗	✗
FAST SYNC	✗	✓	✗
SYNC	✗	✓	✗

FSFO without FAR SYNC	Maximum Performance	Maximum Availability	Maximum Protection
ASync	✓	✗	✗
FAST SYNC	✗	✓	✗
SYNC	✗	✓	✓

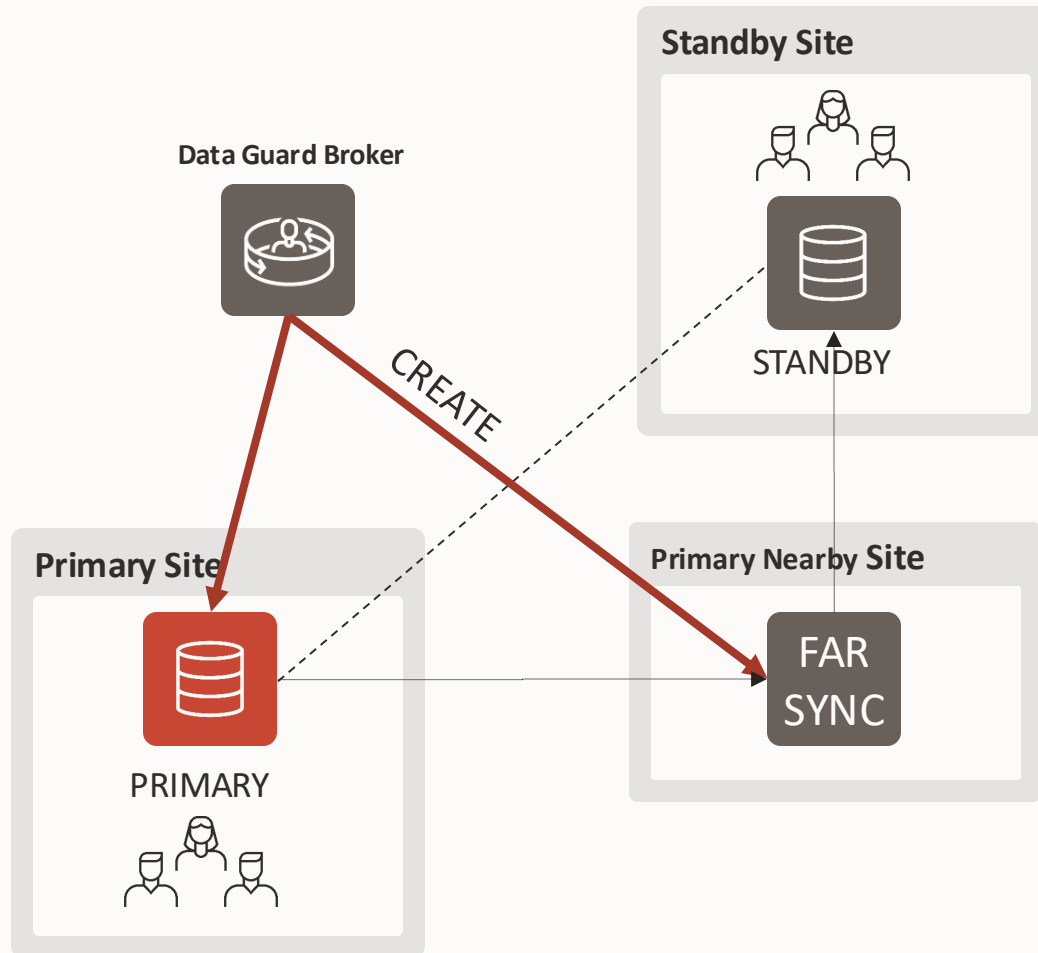
FAR SYNC without FSFO	Maximum Performance	Maximum Availability	Maximum Protection
ASync	✓	✗	✗
FAST SYNC	✗	✓	✗
SYNC	✗	✓	✗

<https://docs.oracle.com/en/database/oracle/oracle-database/21/dgbrk/using-data-guard-broker-to-manage-switchovers-failovers.html#GUID-7423C774-27DF-49F9-BB43-7D547BCE7762>



# Data Guard Broker Far Sync Instance Creation

One step further automated by the broker



```
DGMGRL> CREATE FAR_SYNC bostonfs
```

```
AS CONNECT IDENTIFIER IS "bostonfs_conn_str"  
PARAMETER_VALUE_CONVERT "boston", "bostonfs"  
SET LOG_FILE_NAME_CONVERT "boston", "bostonfs"  
SET DB_RECOVERY_FILE_DEST "$ORACLE_HOME/dbs/"  
SET DB_RECOVERY_FILE_DEST_SIZE "100G"  
RESET UNDO_TABLESPACE;
```

- Automated SPFILE and controlfile creation
- The Far Sync is created, started and added to the configuration

# Oracle **Active** Data Guard Rolling Maintenance and Upgrades

# Solutions for Database Rolling Maintenance and Upgrades

## Manual

Part of Enterprise Edition

Source >= 11.1.0.7 && <= 12.1.0.2

Manual approach

Limited feature support

## DBMS\_ROLLING

Requires Active Data Guard

Source >= 12.1.0.2

Automated

Comprehensive feature support

## GoldenGate

Requires GoldenGate

Source >= 11.2.0.4 (for OCI GG)

Manual approach

Best feature support

Fallback mechanism

Using SQL Apply to Upgrade the Oracle Database

<https://docs.oracle.com/en/database/oracle/oracle-database/19/sbydb/using-sql-apply-to-perform-rolling-upgrade.html>

Using DBMS\_ROLLING to Perform a Rolling Upgrade

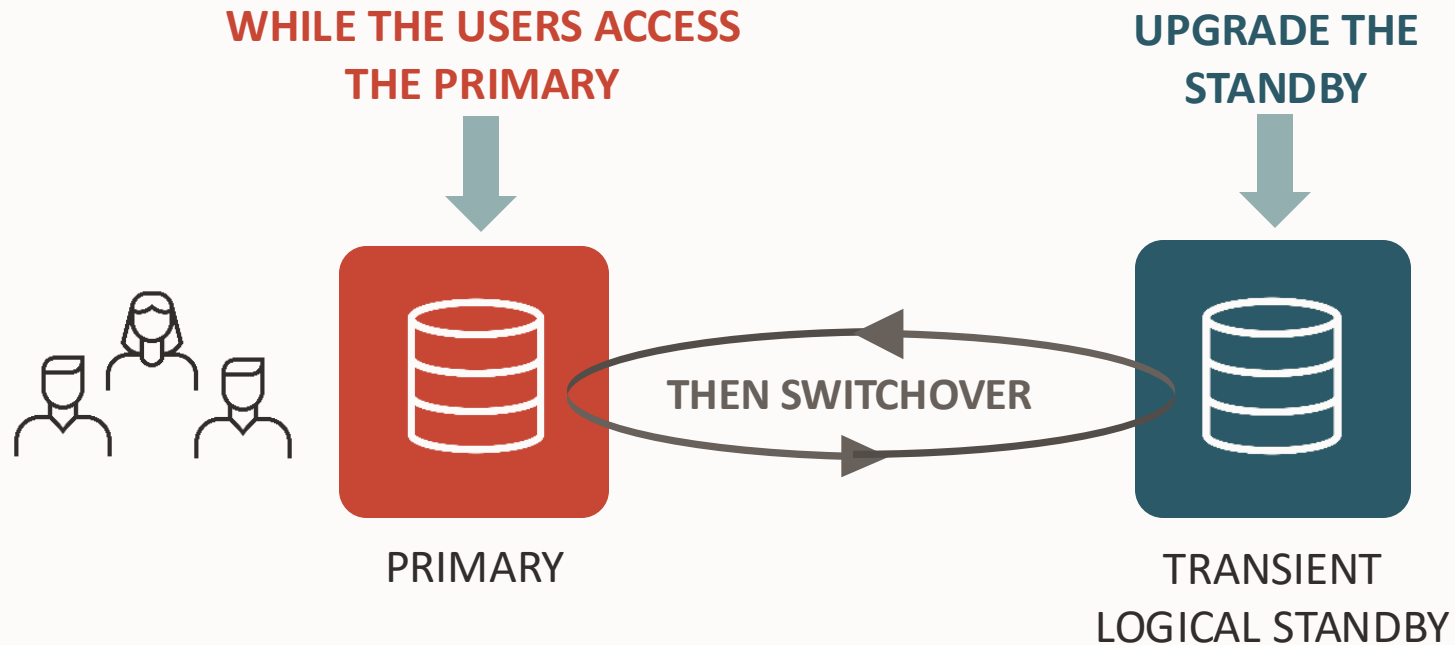
[https://docs.oracle.com/en/database/oracle/oracle-database/19/sbydb/using-DBMS\\_ROLLING-to-perform-rolling-upgrade.html](https://docs.oracle.com/en/database/oracle/oracle-database/19/sbydb/using-DBMS_ROLLING-to-perform-rolling-upgrade.html)

Overview of Steps for Upgrading Oracle Database Using Oracle GoldenGate

<https://docs.oracle.com/en/database/oracle/oracle-database/19/upgdrd/converting-databases-upgrades.html#GUID-8E029631-8265-497C-983B-B8A4ACD47B98>

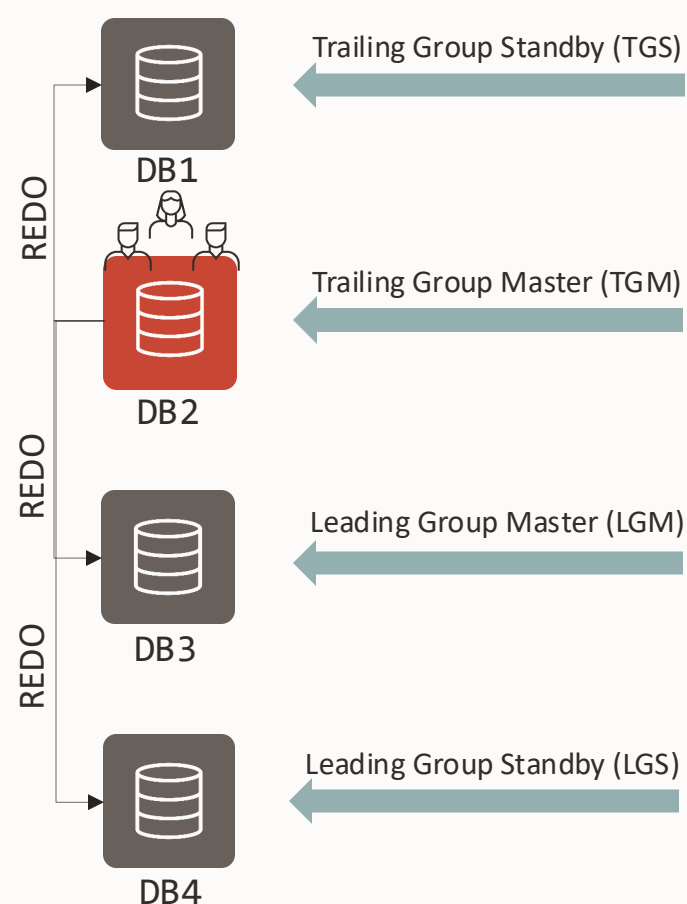
# Active Data Guard Rolling Maintenance and Upgrades

Using `DBMS_ROLLING` package



- Use a transient logical standby database to upgrade with very little downtime.
- The only downtime is as little as it takes to perform a switchover.

# The DBMS\_ROLLING.INIT\_PLAN phase



Trailing Group (\*)

```

-- check DBA_ROLLING_UNSUPPORTED for incompatible data types
-- initialize the plan and set the future primary
DBMS_ROLLING.INIT_PLAN(future_primary=>'DB3');

-- add the required standbys to the TRAILING GROUP
DBMS_ROLLING.SET_PARAMETER('DB1','MEMBER','TRAILING');

-- add the required standbys to the LEADING GROUP
DBMS_ROLLING.SET_PARAMETER('DB4','MEMBER','LEADING');
  
```

Leading Group (\*)

\* Each group can be composed of just one database, with a total of 2 databases in the configuration. Having two pairs provides continuous protection during the upgrade process.



# The DBMS\_ROLLING parameters



ACTIVE\_SESSIONS\_TIMEOUT  
ACTIVE\_SESSIONS\_WAIT  
BACKUP\_CONTROLFILE  
DGBROKER  
DICTIONARY\_LOAD\_TIMEOUT  
DICTIONARY\_LOAD\_WAIT  
DICTIONARY\_PLS\_WAIT\_INIT  
DICTIONARY\_PLS\_WAIT\_TIMEOUT  
EVENT\_RECORDS  
FAILOVER  
GRP\_PREFIX  
IGNORE\_BUILD\_WARNINGS  
IGNORE\_LAST\_ERROR  
LAD\_ENABLED\_TIMEOUT  
LOG\_LEVEL

MEMBER  
READY\_LGM\_LAG\_TIME  
READY\_LGM\_LAG\_TIMEOUT  
READY\_LGM\_LAG\_WAIT  
SWITCH\_LGM\_LAG\_TIME  
SWITCH\_LGM\_LAG\_TIMEOUT  
SWITCH\_LGM\_LAG\_WAIT  
SWITCH\_LGS\_LAG\_TIME  
SWITCH\_LGS\_LAG\_TIMEOUT  
SWITCH\_LGS\_LAG\_WAIT  
UPDATED\_LGS\_TIMEOUT  
UPDATED\_LGS\_WAIT  
UPDATED\_TGS\_TIMEOUT  
UPDATED\_TGS\_WAIT



# The DBMS\_ROLLING parameters



Example:

```
-- Activate full logging
exec DBMS_ROLLING.SET_PARAMETER (scope=>null, name=>'LOG_LEVEL', value=>'FULL');

-- Wait for the SQL Apply Lag to go below 1 minute before initiating the switchover
exec DBMS_ROLLING.SET_PARAMETER('SWITCH_LGM_LAG_WAIT', '1');
exec DBMS_ROLLING.SET_PARAMETER('SWITCH_LGM_LAG_TIME', '60');
```

## Final touches before starting




```
$ # The standby must be mounted  
$ srvctl stop database -d DB3  
$ srvctl start database -d DB3 -o mount
```

```
SQL> -- The PDBs must be open  
SQL> alter pluggable database all open;
```


```
DGMGRL> # no FSFO or MaxProtection  
DGMGRL> disable fast_start failover  
DGMGRL> edit configuration set protection mode as MaxAvailability;
```


# The DBMS\_ROLLING.BUILD\_PLAN phase

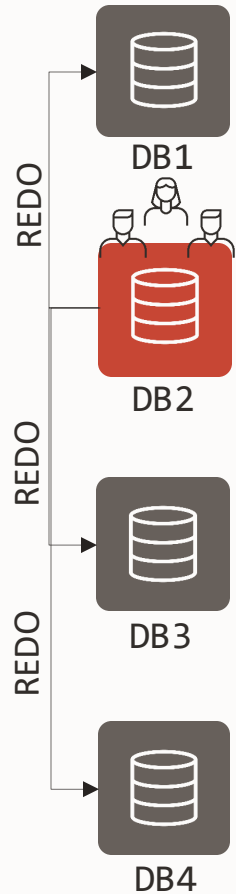
 User sessions

 +1 Upgraded

 Primary

 Physical Standby

 Logical Standby



```
-- build the plan
DBMS_ROLLING.BUILD_PLAN();

-- check for any errors or warnings
SELECT * FROM DBA_ROLLING_EVENTS;

-- review the plan
SELECT * FROM DBA_ROLLING_PLAN ORDER BY INSTID;
```


# The DBMS\_ROLLING.BUILD\_PLAN phase

```
1 START    Notify Data Guard broker that DBMS_ROLLING has started
2 START    Notify Data Guard broker that DBMS_ROLLING has started
3 START    Verify database is a primary
4 START    Verify MAXIMUM PROTECTION is disabled
5 START    Verify database is a physical standby
6 START    Verify physical standby is mounted
7 START    Verify future primary is configured with standby redo logs
8 START    Verify server parameter file exists and is modifiable
9 START    Verify server parameter file exists and is modifiable
10 START   Verify Data Guard broker configuration is enabled
11 START   Verify Data Guard broker configuration is enabled
12 START   Verify Fast-Start Failover is disabled
13 START   Verify Fast-Start Failover is disabled
14 START   Verify fast recovery area is configured
15 START   Verify available flashback restore points
16 START   Verify fast recovery area is configured
17 START   Verify available flashback restore points
18 START   Stop media recovery
19 START   Drop guaranteed restore point DBMSRU_INITIAL
20 START   Create guaranteed restore point DBMSRU_INITIAL
21 START   Drop guaranteed restore point DBMSRU_INITIAL
22 START   Create guaranteed restore point DBMSRU_INITIAL
23 START   Start media recovery
24 START   Verify media recovery is running
25 START   Verify user_dump_dest has been specified
26 START   Backup control file to rolling_change_backup.f
27 START   Verify user_dump_dest has been specified
28 START   Backup control file to rolling_change_backup.f
29 START   Get current supplemental logging on the primary database
30 START   Get current redo branch of the primary database
31 START   Wait until recovery is active on the primary's redo branch
32 START   Reduce to a single instance if database is a RAC
33 START   Verify only a single instance is active if future primary is RAC
34 START   Stop media recovery
35 START   Execute dbms_logstdby.build
36 START   Convert into a transient logical standby
37 START   Open database including instance-peers if RAC
38 START   Verify logical standby is open read/write
39 START   Get redo branch of transient logical standby
40 START   Get reset scn of transient logical redo branch
41 START   Configure logical standby parameters
42 START   Start logical standby apply
43 START   Enable compatibility advance despite presence of GRPs

44 START   Log pre-switchover instructions to events table
45 START   Record start of user upgrade of DB3
46 SWITCH  Verify database is in OPENRW mode
47 SWITCH  Record completion of user upgrade of DB3
48 SWITCH  Scan LADs for presence of DB2 destination
49 SWITCH  Test if DB2 is reachable using configured TNS service
50 SWITCH  Call Data Guard broker to enable redo transport to DB3
51 SWITCH  Archive all current online redo logs
52 SWITCH  Archive all current online redo logs
53 SWITCH  Stop logical standby apply
54 SWITCH  Start logical standby apply
55 SWITCH  Wait until apply lag has fallen below 600 seconds
56 SWITCH  Notify Data Guard broker that switchover to logical standby database is starting
57 SWITCH  Log post-switchover instructions to events table
58 SWITCH  Switch database to a logical standby
59 SWITCH  Notify Data Guard broker that switchover to logical standby database has completed
60 SWITCH  Wait until end-of-redo has been applied
61 SWITCH  Archive all current online redo logs
62 SWITCH  Notify Data Guard broker that switchover to primary is starting
63 SWITCH  Switch database to a primary
64 SWITCH  Notify Data Guard broker that switchover to primary has completed
65 SWITCH  Enable compatibility advance despite presence of GRPs
66 SWITCH  Synchronize plan with new primary
67 FINISH  Reduce to a single instance for FINISH
68 FINISH  Verify only a single instance is active
69 FINISH  Verify database is mounted
70 FINISH  Flashback database
71 FINISH  Convert into a physical standby
72 FINISH  Verify database is open
73 FINISH  Save the DBID of the new primary
74 FINISH  Save the logminer session start scn
75 FINISH  Wait until transient logical redo branch has been registered
76 FINISH  Start media recovery
77 FINISH  Wait until apply/recovery has started on the transient branch
78 FINISH  Wait until upgrade redo has been fully recovered
79 FINISH  Prevent compatibility advance if GRPs are present
80 FINISH  Prevent compatibility advance if GRPs are present
81 FINISH  Drop guaranteed restore point DBMSRU_INITIAL
82 FINISH  Drop guaranteed restore point DBMSRU_INITIAL
83 FINISH  Purge logical standby metadata from database if necessary
84 FINISH  Notify Data Guard broker that DBMS_ROLLING has finished
85 FINISH  Notify Data Guard broker that DBMS_ROLLING has finished
86 FINISH  Restore Supplemental Logging
```





# The DBMS\_ROLLING.START phase

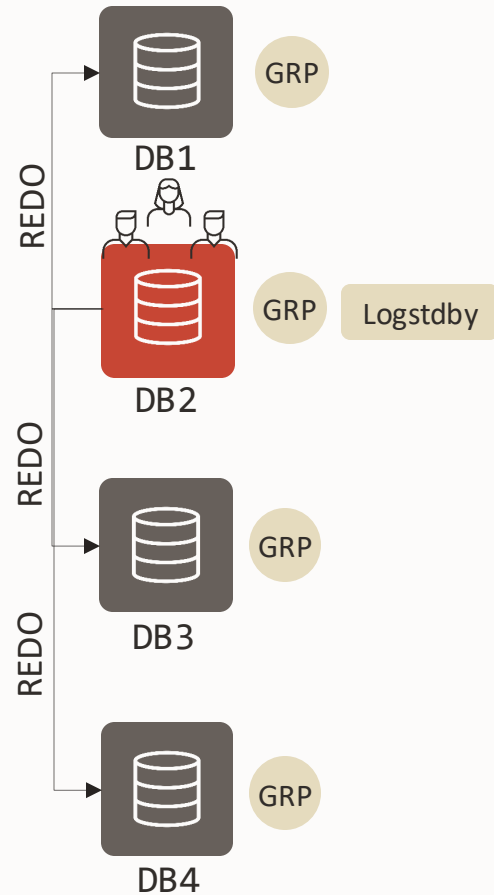
 User sessions

 +1 Upgraded

 Primary

 Physical Standby

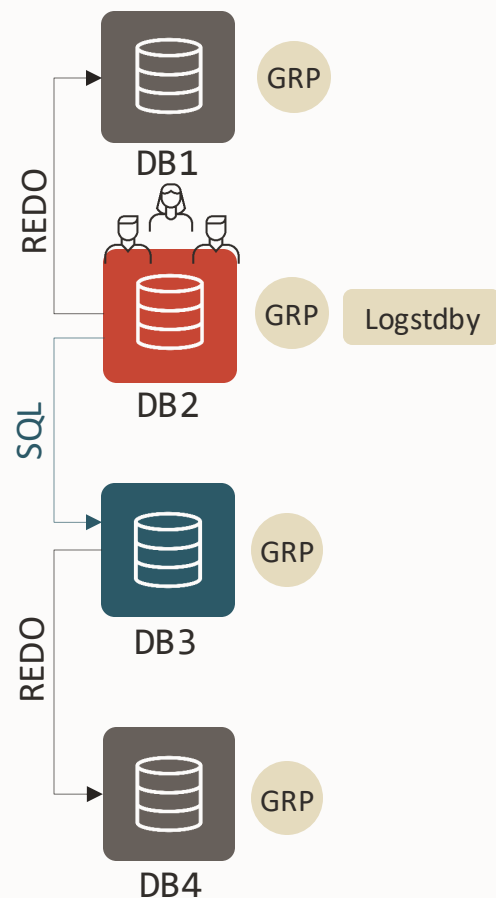
 Logical Standby



```
-- start the plan  
DBMS_ROLLING.START_PLAN();
```

- Creates the Guaranteed Restore Point (GRP)
- Builds the logical standby metadata (`dbms_logstdby.build`)

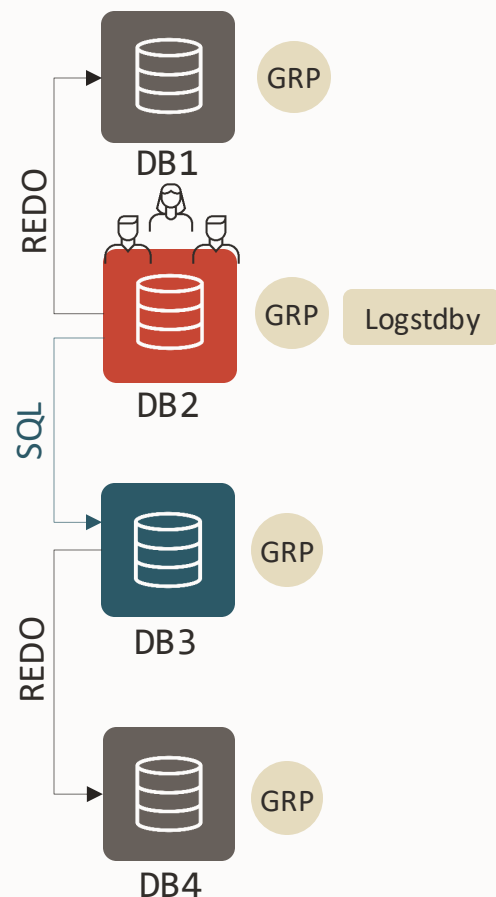
# The DBMS\_ROLLING.START phase



```
-- start the plan
DBMS_ROLLING.START_PLAN();
```

- Creates the Guaranteed Restore Point (GRP)
- Builds the LogMiner directory (dbms\_logstdby.build)
- Converts the LGM to Logical Standby
- Starts SQL Apply
- With a configuration composed of 4 databases, the LGM and TGM are still protected by a physical standby

# The DBMS\_ROLLING.START phase



```
DGMGRL> show configuration;
```

```
Configuration - geneva
```

```
Protection Mode: MaxAvailability
```

```
Members:
```

```
DB1 - Primary database
```

```
DB3 - Physical standby database
```

```
Warning: ORA-16854: apply lag could not be determined
```

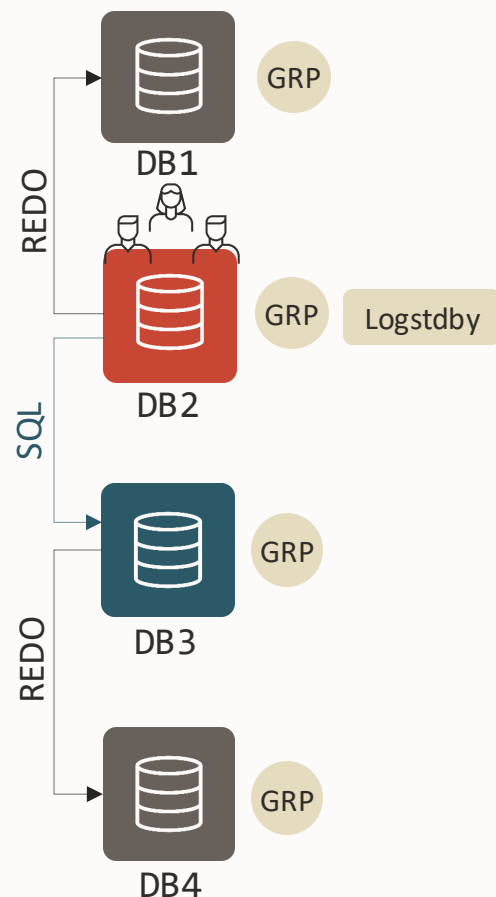
```
Fast-Start Failover: DISABLED
```

```
Configuration Status:
```

```
ROLLING DATABASE MAINTENANCE IN PROGRESS
```



# The DBMS\_ROLLING.START phase



```
DGMGRL> show database DB3
```

```
...
```

```
Role: PHYSICAL STANDBY
```

```
Intended State: APPLY-ON
```

```
Transport Lag: 0 seconds (computed 0 seconds ago)
```

```
Apply Lag: 3 minutes 18 seconds (computed 0 seconds ago)
```

```
...
```

```
Database Warning(s):
```

```
ORA-16866: database converted to transient logical standby database for rolling database maintenance
```

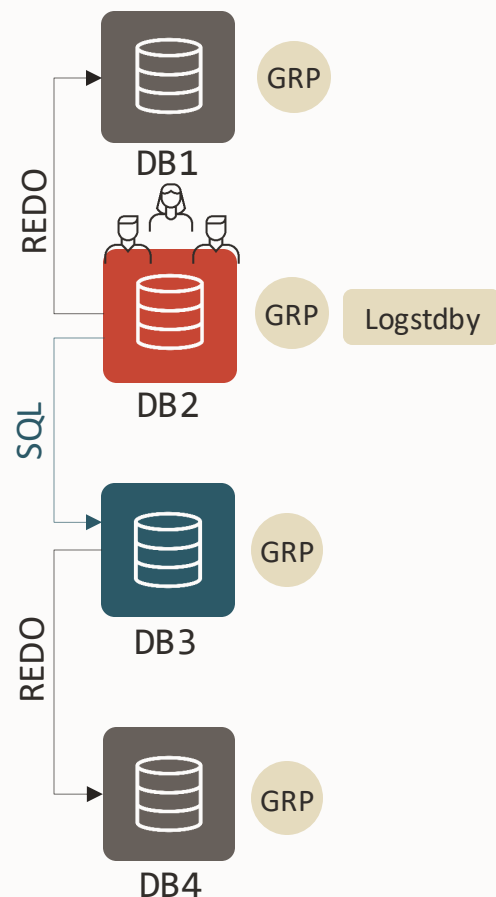
```
Database Status:
```

```
WARNING
```





# The DBMS\_ROLLING.START phase



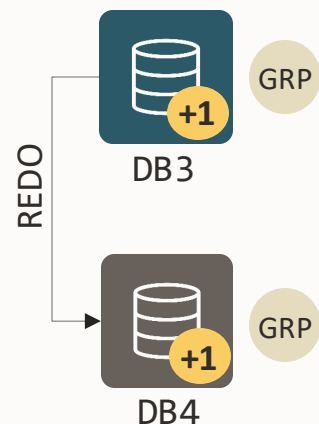
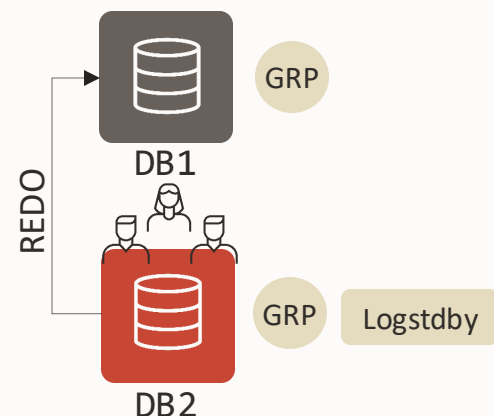
```
-- check the status of the SQL apply:
SQL> select * from V$LOGSTDBY_PROGRESS;
```

```
-- use SQL apply commands if you need
SQL> alter database start logical standby apply immediate;
```

```
-- check for logical standby error messages
SQL> select * from DBA_LOGSTDBY_EVENTS
2>      order by event_timestamp;
```

```
22-NOV-21 06.41.12  DML on "AUDSYS"."AUD$UNIFIED"
ORA-16129: unsupported DML encountered
22-NOV-21 06.41.13  truncate table wri$_adv_addm_pdb
ORA-16247: DDL skipped on internal schema
```

# The Upgrade/Maintenance phase



- Stop the SQL Apply on the Leading Group Master
- Do the maintenance on the Leading Group Master

```
-- e.g. upgrade to a major version with AutoUpgrade
$ java -jar autoupgrade.jar -config CDB1.cfg -mode deploy
```

- This is out of DBMS\_ROLLING scope (it is a manual step)
- Don't forget to align the Leading Group Standbys if necessary
- Use it for any major maintenance that requires longer downtimes (change of physical layout, structure changes, offline operations)





## The Upgrade/Maintenance phase (2)

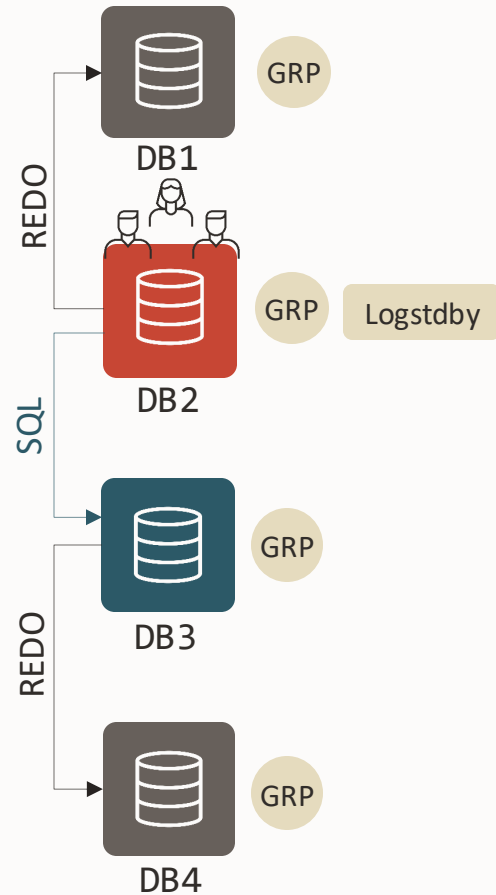
 User sessions

 +1 Upgraded

 Primary

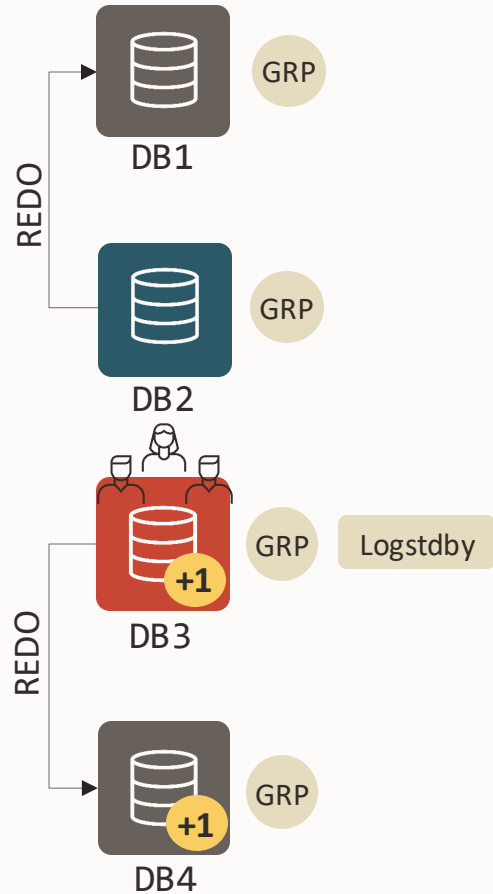
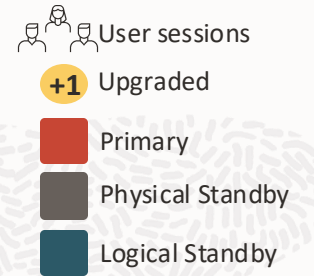
 Physical Standby

 Logical Standby



- Start the SQL Apply on the Leading Group Master to catch up with the primary database changes

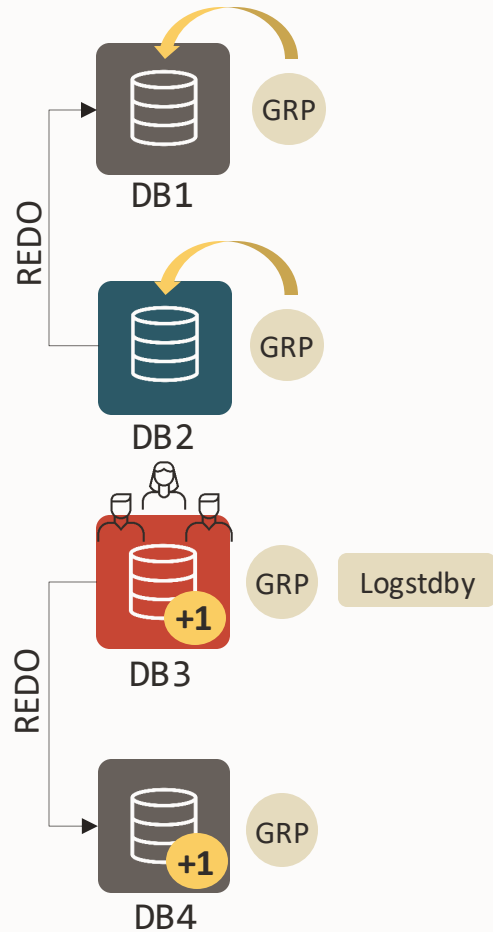
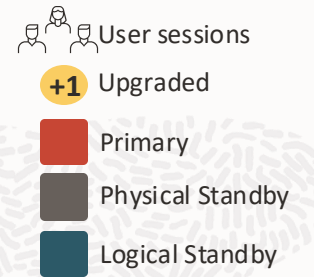
# The DBMS\_ROLLING.SWITCHOVER phase



```
-- switchover to the upgraded database  
DBMS_ROLLING.SWITCHOVER()
```

- Depending on the source version and HA configuration, the old connections get FAN notifications and drain automatically
- New connections go to the new primary. Application downtime is minimal.
- The former primary becomes unusable at this point

# The DBMS\_ROLLING.SWITCHOVER phase

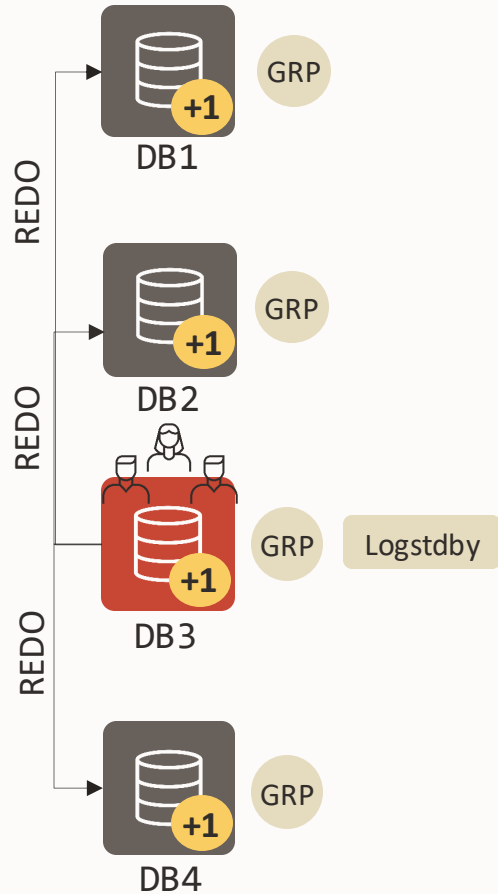
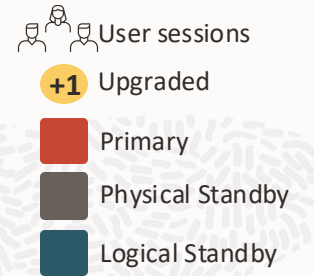


- Start the Trailing Group members with the new binaries (manual)

```
-- run the final part of the plan  
DBMS_ROLLING.FINISH_PLAN()
```

- Flashes back the Trailing Group Master and Standby to the GRP

# The DBMS\_ROLLING.SWITCHOVER phase



- Start the Trailing Group members with the new binaries (manual)

```
-- run the final part of the plan  
DBMS_ROLLING.FINISH_PLAN()
```


- Flashes back the Trailing Group Master and Standby to the GRP
- Converts the Trailing Group Master to a physical standby
- Starts redo apply and catches up with the primary
- Drops the guaranteed restore points and logical standby metadata


# The DBMS\_ROLLING.SWITCHOVER phase

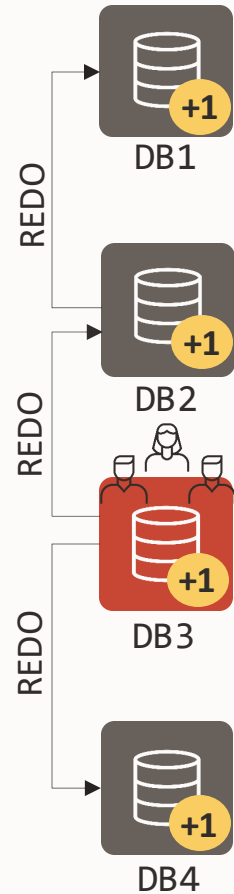
 User sessions

 +1 Upgraded

 Primary

 Physical Standby

 Logical Standby



```
-- destroy the plan to clean up everything
DBMS_ROLLING.DESTROY_PLAN()
```

## DBMS\_ROLLING catalog views



Evaluate DBA\_ROLLING\_UNSUPPORTED

Check here for unsupported data types!

Initialize DBA\_ROLLING\_PARAMETERS

Get the current parameters before building

Build DBA\_ROLLING\_DATABASES  
DBA\_ROLLING\_PLAN

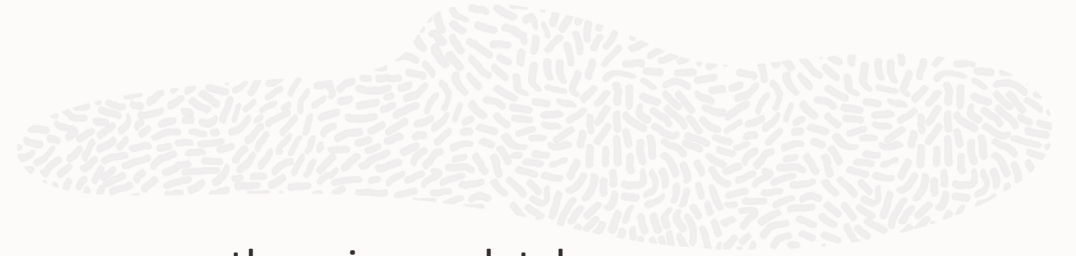
Verify the plan before and during the execution

Monitor DBA\_ROLLING\_EVENTS  
DBA\_ROLLING\_STATISTICS  
DBA\_ROLLING\_STATUS

Warning and errors are visible here



## DBMS\_ROLLING points of attention



Do not create the logical standby on the **same** server as the primary database



Supplemental logging is enabled automatically which introduces an overhead and increases the amount of redo generated



When supplemental logging is enabled all DML cursors are invalidated



Not all data types and partitioning types are supported



For optimal performance all tables should have primary keys or unique keys

# Important DBMS\_ROLLING milestones

The driver is the SOURCE database!



SOURCE VERSION

12.1

- First version of DBMS\_ROLLING for upgrades from 12.1 to higher versions

12.2

- Integration with the **Data Guard broker**
- **FAN events** for Clusterware-backed databases
- Support for **Identity columns**

19c

- Planned in future RU: Support for **Application Continuity** and **Transparent Application Continuity** (backport from 23ai)

21c

- **FAN events without Clusterware**
- Support for **JSON datatype**

23ai

- Support for **Application Continuity** and **Transparent Application Continuity**
- Support for Blockchain tables
- Support for new Boolean data type
- Support for SQL domains

# DBMS\_ROLLING and client failover

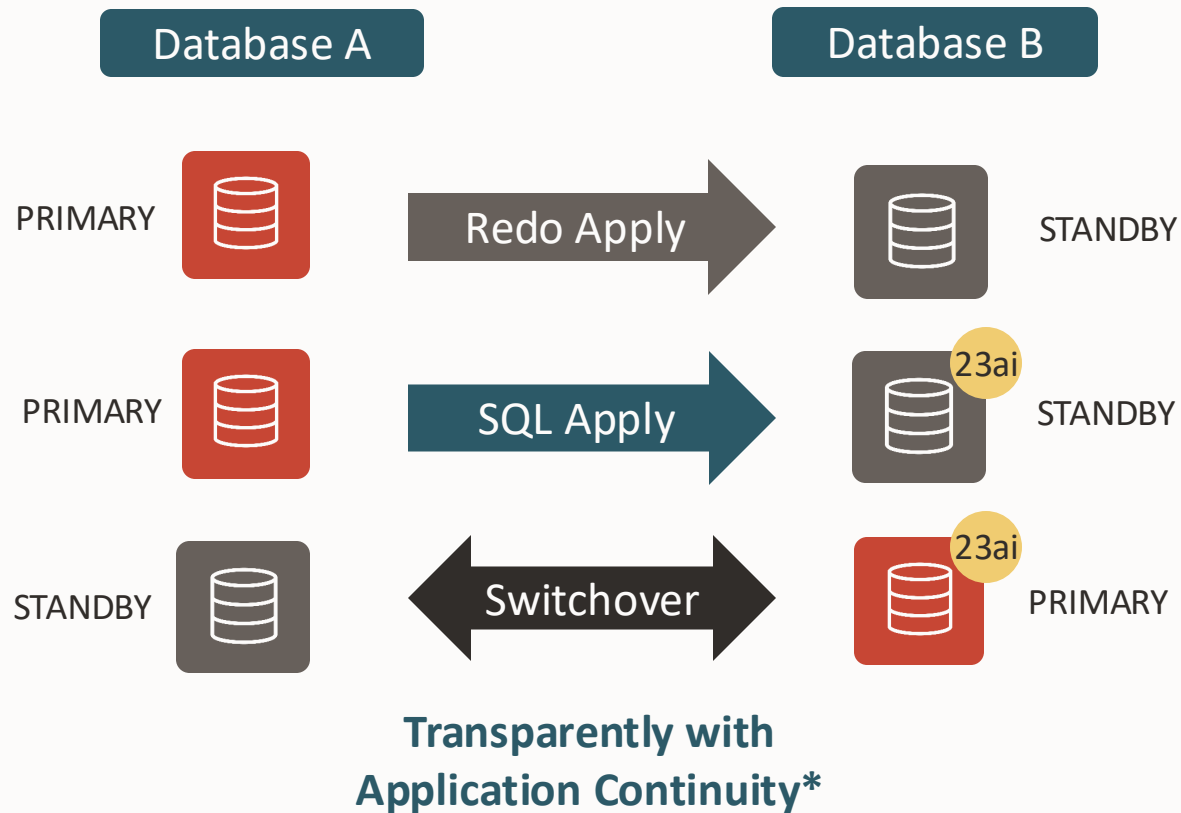


DBMS_ROLLING.SWITCHOVER	Broker + OCW	Broker Only
12.1	Broker Not supported	Broker Not supported
12.2	FAN events	No FAN events
19c	FAN events (AC/TAC backport planned)	No FAN events
21c	FAN events	FAN events
23ai	FAN events + AC/TAC	FAN events + AC/TAC



# Zero Application Downtime for Database Release Upgrades

Minimizes application impact throughout the entire database upgrade process



## (Transparent) Application Continuity

- Hides database downtime from your users
  - It rebuilds the session state
  - It replays in-flight transactions

## DBMS\_ROLLING

- Enables the automated rolling application of version-changing upgrades and patch sets.

**Together they hide the final switchover needed at the end of the automated process.**

## DBMS\_ROLLING – Read More



Using DBMS\_ROLLING to Perform a Rolling Upgrade

[https://docs.oracle.com/en/database/oracle/oracle-database/23/sbydb/using-DBMS\\_ROLLING-to-perform-rolling-upgrade.html](https://docs.oracle.com/en/database/oracle/oracle-database/23/sbydb/using-DBMS_ROLLING-to-perform-rolling-upgrade.html)

DBMS\_ROLLING - PL/SQL Packages and Types Reference

[https://docs.oracle.com/en/database/oracle/oracle-database/23/arpls/DBMS\\_ROLLING.html#GUID-097F1B39-E623-43B5-BA30-DF377BFE05CF](https://docs.oracle.com/en/database/oracle/oracle-database/23/arpls/DBMS_ROLLING.html#GUID-097F1B39-E623-43B5-BA30-DF377BFE05CF)

Automated Database Upgrades using Oracle Active Data Guard and DBMS\_ROLLING

<https://www.oracle.com/technetwork/database/availability/database-upgrade-dbms-rolling-4126957.pdf>

Oracle Database Rolling Upgrades (without DBMS\_ROLLING)

<https://www.oracle.com/technetwork/database/availability/database-rolling-upgrade-3206539.pdf>

# DBMS\_ROLLING – Read More



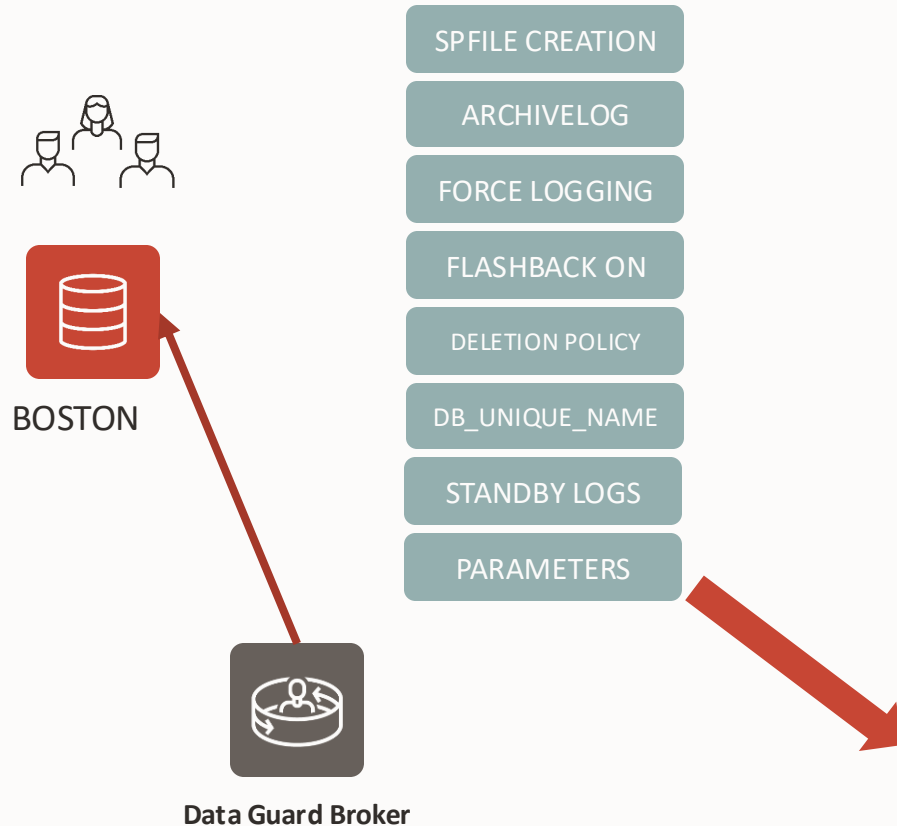
## MOS Notes:

- [Transient Rolling Upgrade Using DBMS\\_ROLLING - Beginners Guide](#)
- [Rolling upgrade using DBMS\\_ROLLING - Complete Reference \(Doc ID 2086512.1\)](#)
- [MAA Whitepaper: SQL Apply Best Practices \(Doc ID 1672310.1\)](#)
- [Step by Step How to Do Switchover/Failover on Logical Standby Environment \(Doc ID 2535950.1\)](#)
- [How To Skip A Complete Schema From Application on Logical Standby Database \(Doc ID 741325.1\)](#)
- [How to monitor the progress of the logical standby \(Doc ID 1296954.1\)](#)
- [How To Reduce The Performance Impact Of LogMiner Usage On A Production Database \(Doc ID 1629300.1\)](#)
- [Handling ORA-1403 ora-12801 on logical standby apply \(Doc ID 1178284.1\)](#)
- [Troubleshooting Example - Rolling Upgrade using DBMS\\_ROLLING \(Doc ID 2535940.1\)](#)
- [DBMS Rolling Upgrade Switchover Fails with ORA-45427: Logical Standby Redo Apply Process Was Not Running \(Doc ID 2696017.1\)](#)
- [SRDC - Collect Logical Standby Database Information \(Doc ID 1910065.1\)](#)
- [MRP fails with ORA-19906 after Flashback of Transient Logical Standby used for Rolling Upgrade \(Doc ID 2069325.1\)](#)
- [What Causes High Redo When Supplemental Logging is Enabled \(Doc ID 1349037.1\)](#)

## Other 21c features for Data Guard and Broker

# Automatic Primary Database Preparation

Faster and easier creation of Data Guard environments



```
DGMGRL> PREPARE DATABASE FOR DATA GUARD
WITH DB_UNIQUE_NAME IS boston
DB_RECOVERY_FILE_DEST IS "+FRA"
DB_RECOVERY_FILE_DEST_SIZE IS "400G"
BROKER_CONFIG_FILE1 IS "+DATA/BOSTON/dg1.dat"
BROKER_CONFIG_FILE2 IS "+FRA/BOSTON/dg2.dat"
RESTART;
```

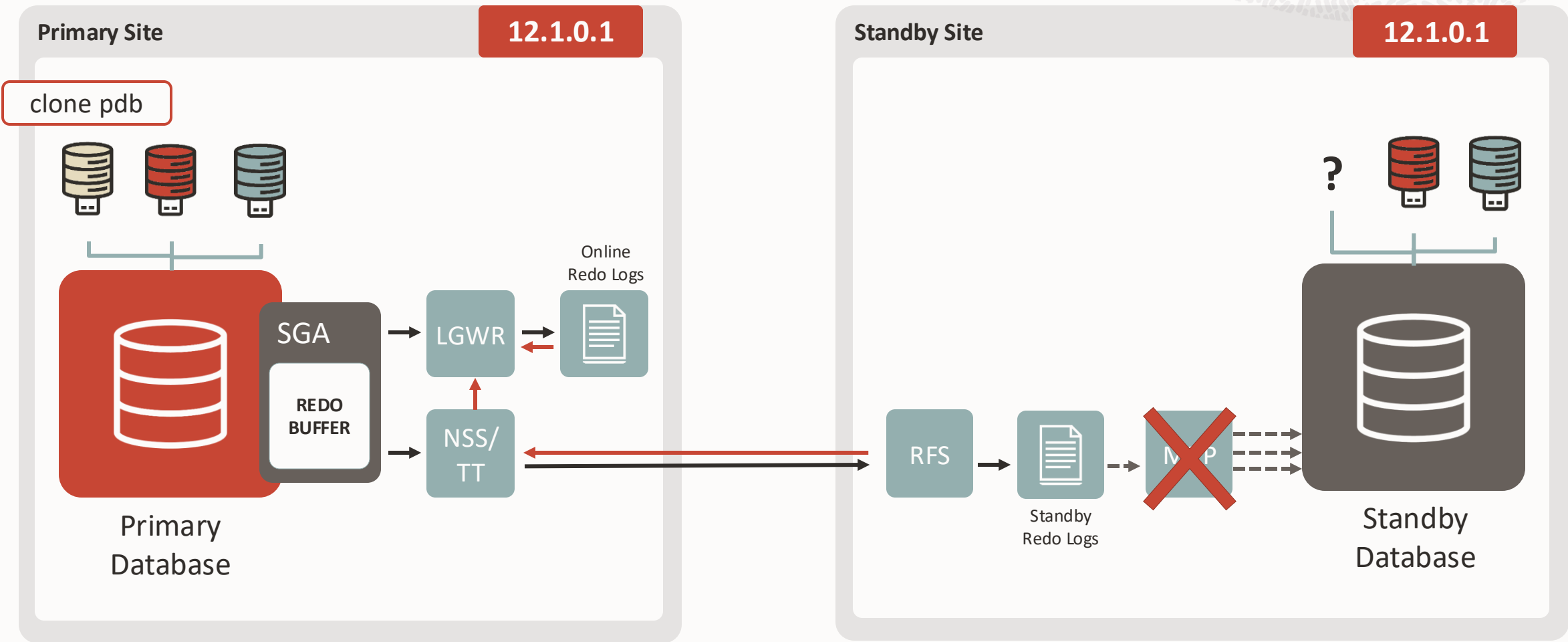
- If the parameters are good enough, they are not modified
- It restarts the database for:
  - Changes to static parameters
  - Enabling the Archivelog mode

DB_FILES	= 1024
LOG_BUFFER	= 256M
DB_BLOCK_CHECKSUM	= TYPICAL
DB_LOST_WRITE_PROTECT	= TYPICAL
DB_FLASHBACK_RETENTION_TARGET	= 120
PARALLEL_THREADS_PER_CPU	= 1
STANDBY_FILE_MANAGEMENT	= AUTO
DG_BROKER_START	= TRUE



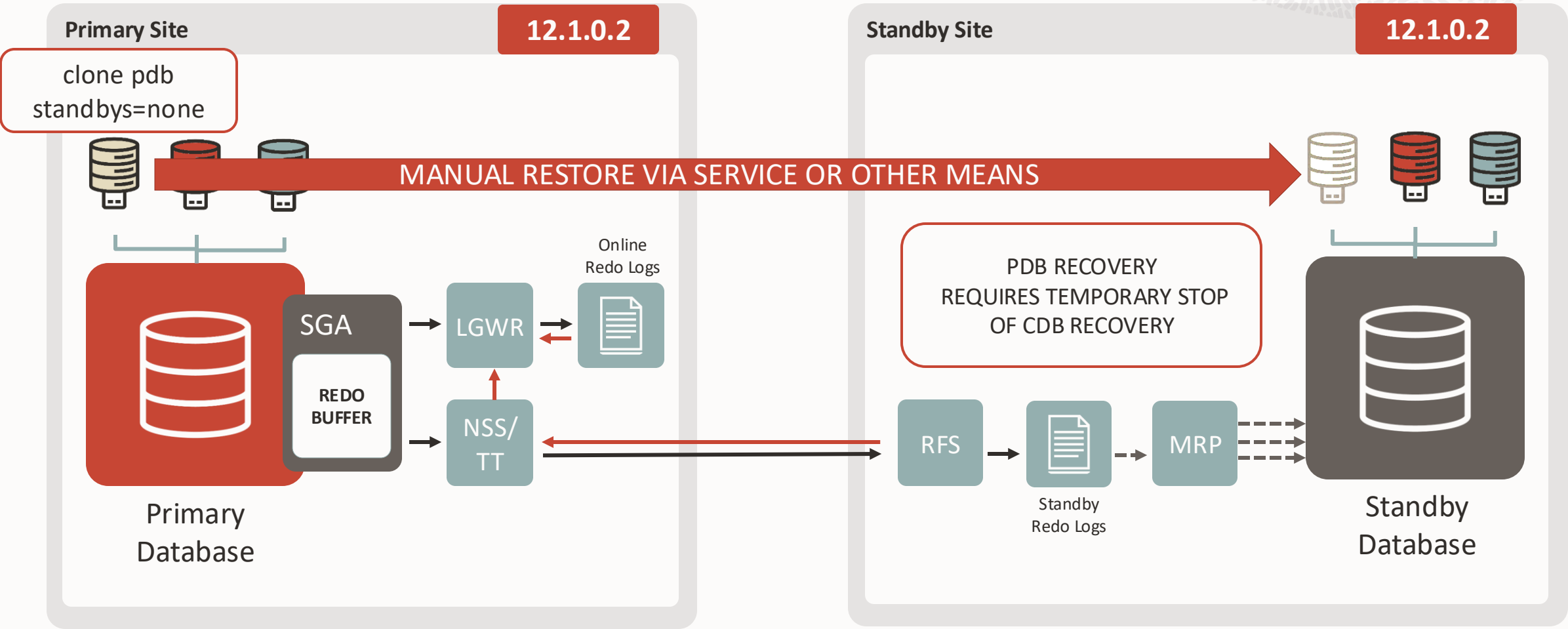
# Pluggable Database Recovery Isolation

Simplified PDB cloning in Data Guard configurations



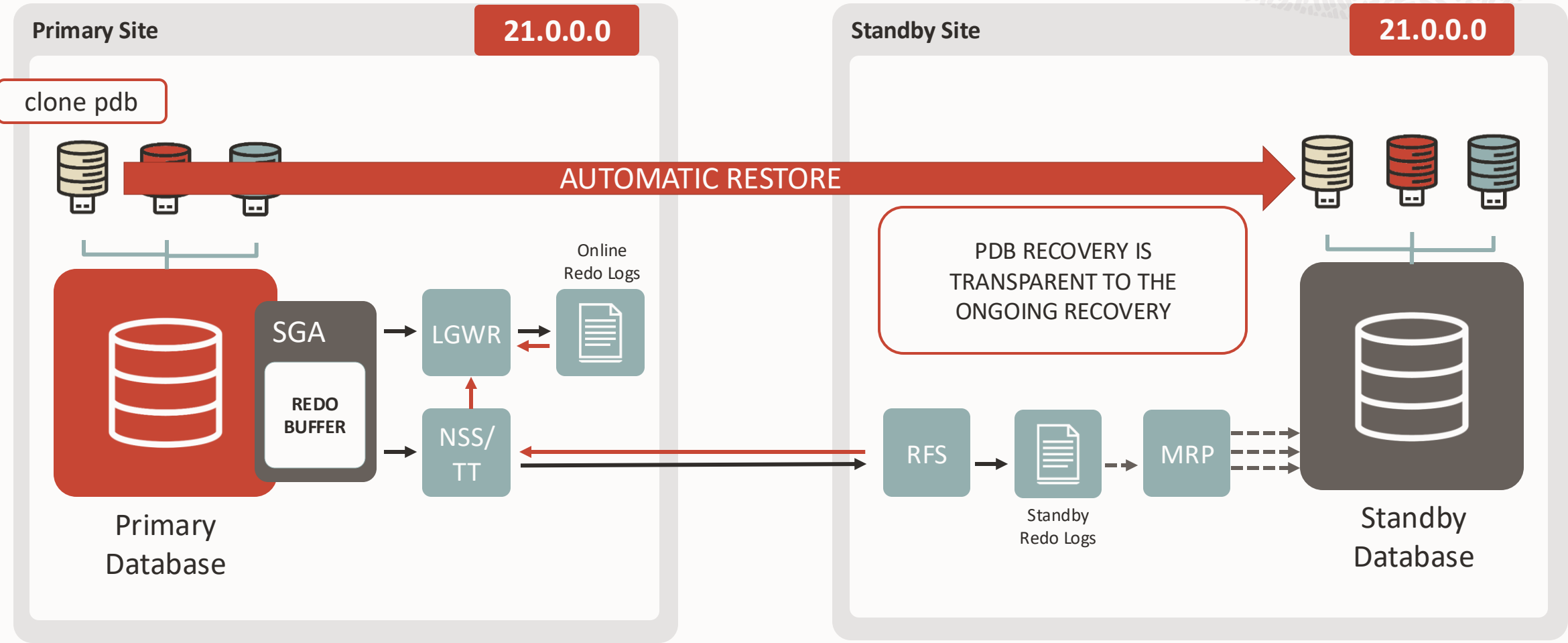
# Pluggable Database Recovery Isolation

Simplified PDB cloning in Data Guard configurations



# Pluggable Database Recovery Isolation

Simplified PDB cloning in Data Guard configurations



# ORDS REST API for Data Guard management

Ready for modern DevOps deployment

New in ORDS 21.4 for 21c databases

```
POST /database/dataguard/configuration/
{
  "primary_connection_identifier": "site1-scan:1521/mydb",
  "primary_database": "mydb_site1"
}
```

Create the configuration

```
POST /database/dataguard/databases/
{
  "connection_identifier": "site2-scan:1521/mydb",
  "database_name": "mydb_site2"
}
```

Add the standby databases

```
PUT /database/dataguard/configuration/
{
  "operation": "ENABLE"
}
```

Enable the configuration

Oracle REST Data Services API - Data Guard REST Endpoints

<https://docs.oracle.com/en/database/oracle/oracle-rest-data-services/21.4/orrst/api-data-guard.html>

# Data Guard management from SQLcl

Everything under control with a single command-line tool

New in SQLcl 22.1 for 21c databases

```
SQL> help dg
```

```
DG
```

```
-----
```

```
Run DG commands
```

```
DG ADD DATABASE "<database name>" AS CONNECT IDENTIFIER IS <connect identifier> [ INCLUDE CURRENT DESTINATIONS ];
DG CREATE CONFIGURATION "<config_name>" AS PRIMARY DATABASE IS <database name> CONNECT IDENTIFIER IS <connect_identifier>
    [ INCLUDE CURRENT DESTINATIONS ];
DG DISABLE CONFIGURATION;
DG DISABLE { DATABASE | RECOVERY_APPLIANCE | FAR_SYNC | MEMBER } <member name>;
DG EDIT CONFIGURATION SET PROPERTY <property name> = '<property value>';
DG EDIT { DATABASE | RECOVERY_APPLIANCE | FAR_SYNC | MEMBER } <member name> SET PROPERTY <property name> = '<property value>';
DG ENABLE CONFIGURATION;
DG ENABLE { DATABASE | RECOVERY_APPLIANCE | FAR_SYNC | MEMBER } <member name>;
DG FAILOVER TO <database name> [IMMEDIATE];
DG REINSTATE DATABASE <database name>;
DG REMOVE CONFIGURATION [PRESERVE DESTINATIONS];
DG REMOVE { DATABASE | RECOVERY_APPLIANCE | FAR_SYNC | MEMBER } <name> [PRESERVE DESTINATIONS];
DG SHOW CONFIGURATION [<property name>];
DG SHOW DATABASE <database name> [<property name>];
DG SWITCHOVER TO <database name> [WAIT [<timeout in seconds>]];
```

## Other changes in Oracle Data Guard 21c

- **Far Sync can now be used with Fast-Start Failover in Max Performance mode (Active Data Guard)**  
Primary can send redo asynchronously to Far Sync.
- **The broker configuration now supports up to four observers**  
Before 21c, the limit was three observers.
- **The PreferredObserverHosts property now supports priorities**  
Example: PreferredObserverHosts='host-a:1, host-b:2'
- **Properties deprecated in 19c are now **desupported****

ArchiveLagTarget	DbFileNameConvert	LsbyPreserveCommitOrder
DataGuardSyncLatency	LogArchiveFormat	LsbyRecordAppliedDdl
LogArchiveMaxProcesses	LogFileNameConvert	LsbyRecordSkipDdl
LogArchiveMinSucceedDest	LsbyMaxEventsRecorded	LsbyRecordSkipErrors
LogArchiveTrace	LsbyMaxServers	LsbyParameters
StandbyFileManagement	LsbyMaxSga	

## Other 23ai features for Data Guard and Broker

# Different Ways to Configure Oracle Data Guard

dgmgrl

```
DGMGRL> create configuration mydb  
> as primary database is mydb  
> connect identifier is 'clu-scan:1521/mydb'
```



```
SQL> DG create configuration mydb as primary database  
is mydb connect identifier is 'clu-scan:1521/mydb'
```

PL/  
SQL

```
DECLARE  
    severity BINARY_INTEGER;  
    retcode  BINARY_INTEGER;  
BEGIN  
    retcode := DBMS_DG.CREATE_CONFIGURATION (  
        config_name      => 'mydb'  
        primary_ci       => 'clu-scan:1521/mydb'  
        severity         => severity  
    );  
END;
```



```
POST /database/dataguard/configuration/  
{  
    "primary_connection_identifier": "clu-  
scan:1521/mydb",  
    "primary_database": "mydb_site1"  
}
```



# Easier Integration Thanks to Many SQL Additions

Some useful new views

```
SQL> select member, property, value from V$DG_BROKER_PROPERTY where value is not null;
```

MEMBER	PROPERTY	VALUE
mydb	FastStartFailoverThreshold	180
mydb	OperationTimeout	30
...		
mydb_site1	DGConnectIdentifier	mydb_site1
mydb_site1	FastStartFailoverTarget	mydb_site2
mydb_site1	LogShipping	ON
...		
mydb_site1	StaticConnectIdentifier	(DESCRIPTION=<...>))
mydb_site2	DGConnectIdentifier	mydb_site2
mydb_site2	FastStartFailoverTarget	mydb_site1
...		

66 rows selected.

```
SQL> desc V$FAST_START_FAILOVER_CONFIG;
```

Name	Null?	Type
FSFO_MODE		VARCHAR2(19)
STATUS		VARCHAR2(22)
CURRENT_TARGET		VARCHAR2(30)
THRESHOLD		NUMBER
OBSERVER_PRESENT		VARCHAR2(7)
OBSERVER_HOST		VARCHAR2(512)
PING_INTERVAL		NUMBER
PING_RETRY		NUMBER
PROTECTION_MODE		VARCHAR2(30)
LAG_LIMIT		NUMBER
AUTO_REINSTATE		VARCHAR2(5)
OBSERVER_RECONNECT		NUMBER
OBSERVER_OVERRIDE		VARCHAR2(5)
SHUTDOWN_PRIMARY		VARCHAR2(5)

```
SQL> select * from V$DG_BROKER_ROLE_CHANGE;
```

EVENT	STANDBY_TYPE	OLD_PRIMARY	NEW_PRIMARY	FS_FAILOVER_REASON	BEGIN_TIME	END_TIME
Failover	Physical	mydb1	mydb1b	Manual Failover	30-AUG-2024 19:01:14	30-AUG-2024 19:01:35
Switchover	Physical	mydb1b	mydb1		30-AUG-2024 19:04:53	30-AUG-2024 19:05:15
Switchover	Physical	mydb1	mydb1b		30-AUG-2024 20:51:38	30-AUG-2024 20:52:03
Failover	Physical	mydb1b	mydb1	Manual Failover	30-AUG-2024 20:52:46	30-AUG-2024 20:53:04
Switchover	Logical	mydb1d	mydb1		30-AUG-2024 20:35:27	30-AUG-2024 20:35:48
Fast-Start Failover	Physical	mydb1	mydb1b	Primary Disconnected	30-AUG-2024 20:13:51	30-AUG-2024 20:14:53



# Strict Database Validation

More checks, better explanations, increased operational security

(\*) available from 23.6

```
VALIDATE DATABASE [VERBOSE] <database>  
[ STRICT { ALL | APPLY_PROPERTY | DATAFILES_OFFLINE (*) | FLASHBACK | FORCE_LOGGING | LOG_FILES_CLEARED  
| LOG_FILE_CONFIGURATION | PDBS_OFFLINE (*) | PDB_SAVE_STATE (*) | TRANSPORT_PROPERTY } ];
```

```
DGMGRL> validate database chicago strict all;  
DGM-17567: Current database session was authenticated using operating system credentials.
```

```
Database Role:      Physical standby database  
Primary Database:   boston
```

```
Ready for Switchover: No  
The primary or standby database does not have flashback database enabled. (*)
```

```
Ready for Failover:   Yes (Primary Running)
```

```
Flashback Database Status:
```

Database	Status	Retention Target
boston	<b>Off</b>	1440
chicago	<b>Off</b>	1440

```
...
```

# Switchover and Failover Readiness

Checking if the database is ready for a role transition is as easy as selecting a column

Two new columns, SWITCHOVER\_READY and FAILOVER\_READY, computed every minute by the broker.

```
SQL> select database, dataguard_role, status, severity, switchover_ready, failover_ready, transport_mode
2> from v$dg_broker_config;
```

DATABASE	DATAGUARD_ROLE	STATUS	SEVERITY	SWITCHOVER_READY	FAILOVER_READY	TRANSPORT_MODE
boston	PRIMARY	0	SUCCESS	YES	UNKNOWN	-N/A-
chicago	PHYSICAL STANDBY	0	SUCCESS	YES	YES	ASync

The checks done by the broker are a superset of the ALTER DATABASE SWITCHOVER VERIFY command:

```
SQL> alter database switchover to chicago verify;
alter database switchover to chicago verify
*
ERROR at line 1:
ORA-16470: Redo Apply is not running on switchover target
```



# PL/SQL API for Data Guard broker management

Manage Data Guard configurations from any SQL\*Net connection

30+ new functions in DBMS\_DG PL/SQL package

```
DECLARE
    severity BINARY_INTEGER;
    retcode BINARY_INTEGER;
BEGIN

    retcode := DBMS_DG.CREATE_CONFIGURATION (
        config_name      => 'mydb'
        primary_ci       => 'site1-scan:1521/mydb'
        severity         => severity
    );

    IF retcode != 0 THEN
        /* handle error code */
    END IF;

    retcode := DBMS_DG.ADD_DATABASE (
        database_name    => 'mydb_site2'
        database_ci      => 'site2-scan:1521/mydb'
        severity         => severity
    );

END;
```

Create the configuration

Add the standby databases

PL/SQL Packages and Types Reference - DBMS\_DG

[https://docs.oracle.com/en/database/oracle/oracle-database/23/dgbr/oracle-data-guard-dbms\\_dg-api-reference.html](https://docs.oracle.com/en/database/oracle/oracle-database/23/dgbr/oracle-data-guard-dbms_dg-api-reference.html)

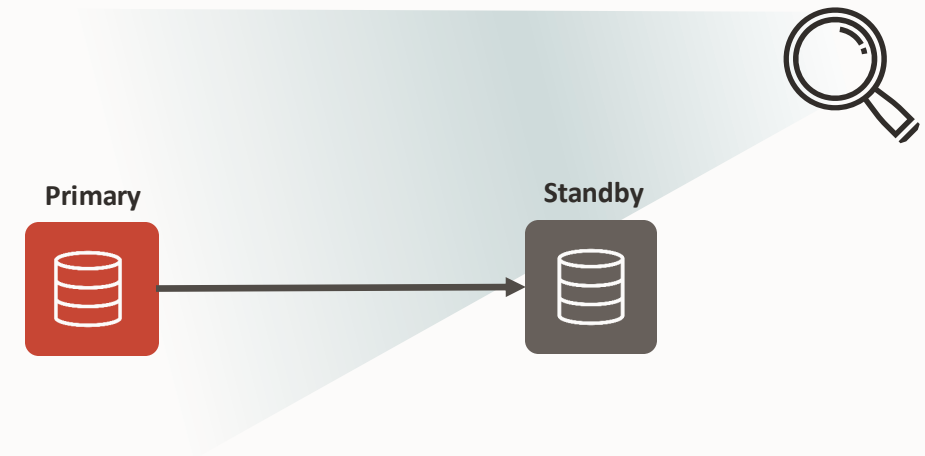
# Easier checking of Data Guard configurations

The new fixed view **V\$DG\_BROKER\_PROPERTY** contains the properties of the configuration and all the members

```
SQL> select member, property, value from V$DG_BROKER_PROPERTY where value is not null;
```

MEMBER	PROPERTY	VALUE
mydb	FastStartFailoverThreshold	180
mydb	OperationTimeout	30
mydb	TraceLevel	USER
mydb	FastStartFailoverLagLimit	300
mydb	CommunicationTimeout	180
mydb	ObserverReconnect	0
mydb	ObserverPingInterval	0
mydb	ObserverPingRetry	0
mydb	FastStartFailoverAutoReinstate	TRUE
mydb	FastStartFailoverPmyShutdown	TRUE
...		
mydb_site1	DGConnectIdentifier	mydb_site1
mydb_site1	FastStartFailoverTarget	mydb_site2
mydb_site1	LogShipping	ON
mydb_site1	LogXptMode	ASYNC
mydb_site1	DelayMins	0
...		
mydb_site1	StaticConnectIdentifier	(DESCRIPTION=<...>)))
mydb_site1	TopWaitEvents	(monitor)
mydb_site1	SidName	(monitor)
mydb_site2	DGConnectIdentifier	mydb_site2
mydb_site2	FastStartFailoverTarget	mydb_site1
...		

66 rows selected.



# New command: **VALIDATE DGConnectIdentifier**

Check network resolution, connectivity, password, service name from the database

```
DGMGRL> validate dgconnectidentifier mydb_site2;
```

```
At instance 'mydb' of member 'mydb_site1'
```

```
Connect Descriptor:
```

```
(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)(HOST=host2)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=mydb_site2.mydomain)(SERVER=DEDICATED)))
```

```
Environment Variables:
```

```
TNS_ADMIN: /u01/app/oracle/product/23.1.0.0/network/admin
```

```
ORACLE_HOME: /u01/app/oracle/product/23.1.0.0
```

```
ORACLE_BASE: /u01/app/oracle
```

```
Initialization Parameters:
```

```
LOCAL_LISTENER: host1:1521
```

```
Connected to instance 'mydb' at member 'mydb_site2'
```

```
At instance 'mydb' of member 'mydb_site2'
```

```
Connect Descriptor:
```

```
(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)(HOST=host2)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=mydb_site2.mydomain)(SERVER=DEDICATED)))
```

```
Environment Variables:
```

```
TNS_ADMIN: /u01/app/oracle/product/23.1.0.0/network/admin
```

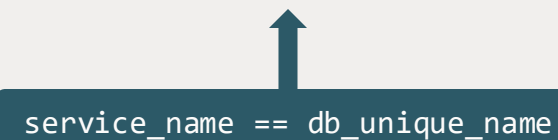
```
ORACLE_HOME: /u01/app/oracle/product/23.1.0.0
```

```
ORACLE_BASE: /u01/app/oracle
```

```
Initialization Parameters:
```

```
LOCAL_LISTENER: host2:1521
```

```
Connected to instance 'mydb' at member 'mydb_site2'
```



service\_name == db\_unique\_name

# New commands: SHOW|EDIT ALL MEMBERS

Easier management of member's properties and parameters

## -- DATA GUARD PROPERTIES

```
DGMGRL> SHOW ALL MEMBERS logxptmode
```

```
mydb_site1: logxptmode = 'ASYNC'
```

```
mydb_site2: logxptmode = 'ASYNC'
```

```
DGMGRL> EDIT ALL MEMBERS SET PROPERTY logxptmode = 'SYNC';
```

```
Property "logxptmode" updated for member "mydb_site1".
```

```
Property "logxptmode" updated for member "mydb_site2".
```

```
DGMGRL> SHOW ALL MEMBERS logxptmode
```

```
mydb_site1: logxptmode = 'SYNC'
```

```
mydb_site2: logxptmode = 'SYNC'
```

## -- DB PARAMETERS

```
DGMGRL> SHOW ALL MEMBERS PARAMETER fast_start_mttr_target
```

```
mydb_site1: fast_start_mttr_target = '0'
```

```
mydb_site2: fast_start_mttr_target = '0'
```

```
DGMGRL> EDIT ALL MEMBERS SET PARAMETER fast_start_mttr_target=15;
```

```
Parameter "fast_start_mttr_target" updated for member "mydb_site1".
```

```
Parameter "fast_start_mttr_target" updated for member "mydb_site2".
```

```
DGMGRL> SHOW ALL MEMBERS PARAMETER fast_start_mttr_target
```

```
mydb_site1: fast_start_mttr_target = '15'
```

```
mydb_site2: fast_start_mttr_target = '15'
```

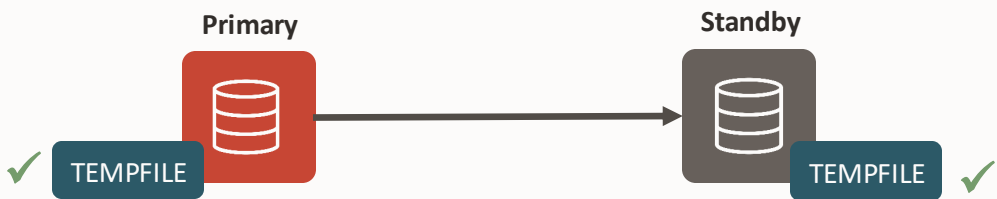
Get and set Data Guard broker properties

Get and set Database parameters

# Automatic tempfile creation on the standby database

Temporary tablespace creation during recovery:

PRIMARY	STANDBY	Non-OMF	OMF
Non-OMF		<b>Standby_file_management AUTO:</b> ✓ Creates <b>one</b> tempfile with the default size using db_file_name_convert.  <b>Standby_file_management MANUAL:</b> ✗ Does not create a tempfile.	<b>Standby_file_management AUTO:</b> ✓ Creates <b>one</b> tempfile with the default size with OMF naming.  <b>Standby_file_management MANUAL:</b> ✗ Does not create a tempfile.(?)
OMF		✗ Does not create a tempfile.	✓ Creates <b>one</b> tempfile with the default size with OMF naming.



When the standby opens and a temporary tablespace has no tempfiles:

Non-OMF	✗ Does not create a tempfile.
OMF	✓ Creates <b>one</b> tempfile with the default size with OMF naming.





# Questions & Answers

# Thank you



ORACLE