|  |  |
| --- | --- |
|  | **Best Practices for Corruption Detection, Prevention, and Automatic Repair - in a Data Guard Configuration** |

**Purpose**

Oracle Active Data Guard is a data protection and availability solution for the Oracle Database.  The purpose of this document is to provide Database Administrators best practice recommendations for configuring key parameters, database features, system features and operational practices to enable best corruption detection, prevention, and automatic repair, in a MAA or Data Guard configuration. This note also provides additional background information on each parameter, performance considerations, and relevant Oracle documentation references.

**Scope**

This article is intended for Database Administrators wanting to learn how to prevent, detect and automatically repair from various data block corruptions.

A [corrupt block](http://docs.oracle.com/cd/E11882_01/backup.112/e10642/glossary.htm#BRADV90136) is a block that has been changed so that it differs from what Oracle Database expects to find. This note covers two data block corruption types:

* In a [**physical block corruption**](http://docs.oracle.com/cd/E11882_01/server.112/e17157/glossary.htm#CBAEHHIG), which is also called a media corruption, the database does not recognize the block at all: the [checksum](http://docs.oracle.com/cd/E11882_01/backup.112/e10642/glossary.htm#BRADV90125) is invalid or the block contains all zeros.  An example of a more sophisticated physical block corruption is when the block header and footer do not match.
* In a [**logical block corruption**](http://docs.oracle.com/cd/E11882_01/server.112/e17157/glossary.htm#CBAJAEHF), the contents of the block are physically sound and pass the physical block checks; however the block can be logically inconsistent. Examples of logical corruption include incorrect block type, incorrect data or redo block sequence number, corruption of a row piece or index entry or data dictionary corruptions.
* Block corruptions caused by **stray writes, lost writes or misdirected writes** can also cause havoc to your database availability. The data block may be physically or logically correct but in this case the block’s content is older or stale or in the wrong location.

Block corruptions can also be divided into interblock corruption and intrablock corruption:

* In intrablock corruption, the corruption occurs in the block itself and can be either a physical or a logical corruption.
* In an interblock corruption, the corruption occurs between blocks and can only be a logical corruption.

## Details

### Causes of corrupted blocks

Block corruptions can be caused by various failures including, but not limited to the following:

* Faulty disks and disk controllers
* Faulty memory
* Faulty network components
* Firmware, operating system, volume manager, NFS or third party software defects
* Oracle Database software defects

Block corruptions can be also be caused by operator errors such as copying backups over existing data files or restoring inconsistent database backups.    Since corruptions can happen anywhere in the system and software stack, MAA recommends a comprehensive list of architectural and configuration best practices.   Active Data Guard is strategic and important component to achieve the most comprehensive Oracle data protection.

### Corruption Summary

The table outlines block corruption checks for various manual operational checks and runtime and background corruption checks.    Manual checks are something that the DBA and operations team can incorporate such as running RMAN backups, RMAN "check logical" validations or running ANALYZE VALIDATE STRUCTURE command on important objects.  Manual checks are especially important to validate data that are rarely updated or queried.    Runtime checks are far superior in that it will catch corruptions almost immediately or during runtime for actively queried and updated data.   Runtime checks can prevent corruptions or automatically fix corruptions resulting in better data protection and higher application availability.  A new background check has been introduced in Exadata to automatically scan and scrub disks intelligently with no application overhead and to automatically fix physically corrupted blocks.

|  |  |  |  |
| --- | --- | --- | --- |
| **Checks** | **Capabilities** | **Physical Block Corruption** | **Logical Block Corruption** |
| Manual checks | Dbverify, Analyze | Physical block checks | Logical intra-block and inter-object consistency checks |
| Manual checks | RMAN | Physical block checks during backup and restore operations | Intra-block logical checks |
| Manual checks | ASM Scrub | Physical block checks | Some logical intra-block checks |
| Runtime checks | Active Data Guard | 1. Continuous physical block checking at standby during transport and apply  2. Strong database isolation eliminates single point database failure  3. Automatic repair of block corruptions  4. Automatic database failover | 1.  With DB\_LOST\_WRITE\_PROTECT enabled, detection of lost writes (11.2 and higher).   With 11.2.0.4 and Data Guard broker, ability to shutdown the primary when lost writes are detected on the primary database.  2. With DB\_BLOCK\_CHECKING enabled on the standby, additional intra-block logical checks |
| Runtime checks | Database | With DB\_BLOCK\_CHECKSUM, in-memory data block and redo checksum validation | With DB\_BLOCK\_CHECKING, in-memory intra-block check validation |
| Runtime checks | ASM | Implicit data corruption detection for reads and writes and automatic repair if good ASM extent block pair is available during writes |  |
| Runtime checks | DIX + T10 DIF | Checksum validation from operating system to HBA controller to disk (firmware). Validation for reads and writes for certified Linux, HBA and disks. |  |
| Runtime checks | Hardware and Storage | Limited checks due to lack of Oracle integration.  Checksum is most common. | Limited checks due to lack of Oracle integration.  Checksum is most common |
| Runtime checks | Exadata | Comprehensive HARD checks on writes | HARD checks on writes |
| Background checks | Exadata | Automatic HARD disk scrub and repair.  Detects and fixes bad sectors. |  |

### Configuration Details

#### Configure at Primary Database:

* DB\_BLOCK\_CHECKSUM=FULL
* DB\_BLOCK\_CHECKING=FULL or MEDIUM
* DB\_LOST\_WRITE\_PROTECT=TYPICAL
* Enable Flashback Technologies for fast point-in-time recovery from human errors (e.g. drop table, inadvertent or malicious data changes) and for fast reinstatement of a primary database following failover.

#### Configure at Data Guard Standby Database:

* DB\_BLOCK\_CHECKSUM=FULL
* DB\_BLOCK\_CHECKING=FULL or MEDIUM
* DB\_LOST\_WRITE\_PROTECT=TYPICAL
* Enable Flashback Technologies for fast point-in-time recovery from human errors (e.g. drop table, inadvertent or malicious data changes) and for fast reinstatement of a primary database following failover.
* Use Active Data Guard to enable Automatic Block Repair (Data Guard 11.2 onward).

 Review the additional background on each of these settings provided in the sections below, especially if tests show that any of the above recommendations have a greater than acceptable impact on the performance of your application.

#### Deploy Primary and Standby on Oracle Exadata Database Machine and Exadata Storage Servers

In addition to the settings above that provide optimal data protection for the Oracle Database on any platform supported by Oracle, the Exadata Database Machine and Sparc Supercluster also implements comprehensive Oracle Hardware Assisted Resilient Data (HARD) specifications, providing a unique level of validation for Oracle block data structures.   The Exadata HARD checks include support for spfiles, controlfiles, log files, Oracle data files and Data Guard broker file when residing on Exadata Storage and works during an ASM rebalance or during ASM resync operations.   Oracle Exadata Storage Server Software detects corruptions introduced into the I/O path between the database and storage. It stops corrupted data from being written to disk when a HARD check fails. This eliminates a large class of failures that the database industry had previously been unable to prevent.   Examples of the Exadata HARD checks include:  1) redo and block checksum, 2) correct log sequence, 3) block type validation, 4) block number validation, 5) Oracle data structures such as block magic number, block size, sequence#, and block header and tail data structures.   Exadata HARD checks are the most comprehensive list of Oracle data block checks initiating from the storage software (cellsrv) and works transparently after enabling database's DB\_BLOCK\_CHECKSUM parameter.   Except for the case of Exadata storage, the Oracle HARD initiative has ended.  Most past storage HARD implementations only provided checksums and very simple data block checks.

Starting with Oracle Exadata Software 11.2.3.3.0 and Oracle Database 11.2.0.4, Oracle Exadata Storage Server Software provides **Automatic Hard Disk Scrub and Repair.** This feature automatically inspects and repairs hard disks periodically when hard disks are idle. If bad sectors are detected on a hard disk, then Oracle Exadata Storage Server Software automatically sends a request to Oracle ASM to repair the bad sectors by reading the data from another mirror copy. By default, the hard disk scrub runs every two weeks.   It’s very lightweight and has enormous value add by fixing physical block corruptions even with infrequently access data.

#### Deploy Oracle Data Integrity eXtensions (DIX) with T10 Data Integrity Field (DIF) when not on Exadata

Oracle Linux team has collaborated with hardware vendors and Oracle database development to extend Oracle data integrity extensions from Oracle’s operating system (Linux) to various vendor’s host adapter down to the storage device.  With these extensions, DIX provides end to end data integrity for reads and writes through a checksum validation.   The prerequisite is to leverage certified storage, HBA and disk firmware.  An example of this partnership is DIX integration with Oracle Linux, Emulex or QLogic Host Bus Adapters and any T10 DIF capable storage arrays such as EMC VMAX.    Refer to the following documentation for more information.

* [https://oss.oracle.com/~mkp/docs/OOW2011-DI.pdf](https://oss.oracle.com/%7Emkp/docs/OOW2011-DI.pdf)
* <http://www.oracle.com/us/technologies/linux/prevent-silent-data-corruption-1852761.pdf>

### General Guidance on Performance Trade-offs

Performance implications are discussed in each of the sections below. In general, the processing that accompanies higher levels of corruption checking, automatic repair, or fast point in time recovery, will create additional overhead on primary and standby systems. While this overhead is reduced with every Oracle release as validation and repair algorithms are enhanced, the usual recommendation for conducting thorough performance testing still applies.

#### DB\_BLOCK\_CHECKSUM- Background

This parameter determines whether DBWn and the direct loader will calculate a checksum (a number calculated from all the bytes stored in the block) and store it in the cache header of every data block and redo log when writing to disk. The checksum is used to validate that a block is not physically corrupt, detecting corruptions caused by underlying disks, storage systems, or I/O systems. If checksum validation fails when it is set to FULL, Oracle will attempt to recover the block by reading it from disk (or from another instance) and applying the redo needed to fix the block. Corruptions are recorded as ORA-600 or ORA-01578 in the database or ASM alert logs.   
  
Checksums do not ensure logical consistency of the block contents (see DB\_BLOCK\_CHECKING). Checksum checks happen in memory when a process reads the data or redo block into the SGA or PGA. Prior to writing an updated or new data or redo block, a new checksum is created. Potential clients of DB\_BLOCK\_CHECKSUM include: all foregrounds, DBWR, LGWR, LNS, RFS, ARCH, MRP, and recovery slaves.   
  
Possible settings include:

* OFF or FALSE: When set to OFF, DBWn calculates checksums only for the SYSTEM tablespace, but not for user tablespaces. In addition, no log checksum is performed when this parameter is set to OFF.
* TYPICAL or TRUE: Checksums are verified when a block is read and the last write of the block stored a checksum.
* FULL: In addition to checks made by TYPICAL, Oracle also verifies the checksum before a change application from update/delete statements and recomputes it after the change is applied. In addition, Oracle gives every log block a checksum before writing it to the current log.
* Starting with Oracle Database 11g, CPU and cache efficiency has been improved by having the generating foreground processes perform most of the log block checksum, while the LGWR performs the remaining work. Prior to Oracle Database 11g, the LGWR solely performed the log block checksum, it would verify the checksum of each log block generated by the foreground processes before writing it to disk.
* Best practice is to set DB\_BLOCK\_CHECKSUM=FULL on both the primary and standby databases, which typically incurs from 4% to 5% overhead on the system. Overhead for a setting of TYPICAL ranges from 1% to 2% for an OLTP workload and has minimal effect on Redo Apply performance, but also provides less protection. If tests show that using FULL at the primary results in unacceptable performance impact, consider setting the primary to TYPICAL and the standby database to FULL fo an optimal tradeoff between protection and performance (and also accept the fact that settings will have to be changed following role transitions).

#### DB\_BLOCK\_CHECKING - Background

This parameter specifies whether or not Oracle performs logical intra-block checking for database blocks (memory semantic check). Block checking will check block contents, including header and user data, when changes are made to the block and prevents in-memory corruptions from being written to disk. It performs a logical validation of the integrity of a block by walking through the data on the block, making sure it is self consistent. When DB\_BLOCK\_CHECKING is set at MEDIUM or FULL, block corruptions that are detected in memory are automatically repaired by reading the good block from disk and applying required redo. If for any reason the corruption cannot be repaired an error will be reported and the data block write will be prevented. All corruptions are reported as ORA-600 or ORA-01578 errors in the database or ASM alert logs.  
  
Possible settings include:

* OFF or FALSE - No block checking is performed for blocks in user tablespaces. However, semantic block checking for SYSTEM tablespace blocks is always turned on.
* LOW - Basic block header checks are performed after block contents change in memory (for example, after UPDATE or INSERT statements, on-disk reads, or inter-instance block transfers in Oracle RAC). Low is very limited in usefulness for detecting and preventing data block corruption as it does not perform any data layer checks.
* MEDIUM - All LOW checks and full semantic checks are performed for all objects except indexes (whose contents can be reconstructed by a drop and rebuild upon encountering a corruption).
* FULL or TRUE - All LOW and MEDIUM checks plus full semantic checks are performed for all objects.

Oracle recommends setting DB\_BLOCK\_CHECKING to FULL at both primary and standby databases. Workload specific testing is required to assess whether the performance overhead of FULL is acceptable.  If tests show unacceptable performance impact, then set DB\_BLOCK\_CHECKING to MEDIUM.

Performance testing is particularly important given that overhead is incurred on every block change. Block checking typically causes 1% to 10% overhead, but for update and insert intensive applications (such as Redo Apply at a standby database) the overhead can be much higher. OLTP compressed tables also require additional checks that can result in higher overhead depending on the frequency of updates to those tables.

If performance concerns prevent setting DB\_BLOCK\_CHECKING to either FULL or MEDIUM at a primary database, then it becomes even more important to enable this at the standby database. This protects the standby database from logical block corruption that would be undetected at the primary database.

Note: When DB\_BLOCK\_CHECKING is set on the primary database, end-to-end checksums introduced in Oracle Database 11g make it unnecessary to use DB\_BLOCK\_CHECKING at the standby to detect primary database corruption. Oracle, however, still recommends enabling this parameter on the standby database for the following reasons:

* The parameter is always enabled regardless of role, ensuring that the standby will have the correct settings when it assumes the primary database role
* Even though very rare, enabling the parameter on the standby will protect against logical corruptions that may be introduced on the standby database independent of the primary.

#### DB\_LOST\_WRITE\_PROTECT - Background

This parameter enables lost write detection. A data block lost write occurs when an I/O subsystem acknowledges the completion of the block write, while in fact the write did not occur in the persistent storage or in some cases an older version of the block was written out instead.   
  
Possible settings include:

* NONE: When set to NONE on either the primary database or the standby database, no lost write detection functionality is enabled.
* TYPICAL: When set on a primary database it will log buffer cache reads for read-write tablespaces in the redo log and will detect lost writes when the primary database performs media recovery. By also setting TYPICAL at a Data Guard physical standby database, the Data Guard MRP process will continuously check for lost writes that occurred to read-write tablespaces at the primary database, enabling immediate action to be taken and preventing corruptions from being applied to the standby database. It will also detect lost writes that may occur at a standby database independent of the primary.
* FULL: All logging, checking, and detecting that occurs with TYPICAL, but for both read-write and read-only tablespaces.

When a primary database lost write corruption is detected by a Data Guard physical standby database, Redo Apply (MRP) will stop and the standby will signal an ORA-752 error to explicitly indicate a primary lost write has occurred. Note that a Data Guard physical standby also detects other types of corruptions that can occur at either the primary or standby database and will likewise stop Redo Apply, in these cases signaling a more generic ORA-600 [3020] error. In cases where problems originated at the primary database this prevents corruptions from being applied to the standby database and indicates immediate action must be taken to prevent further corruption and downtime. While there is no threat to the primary database in cases where problems originate at the standby database, this provides immediate warning that corrective action must be taken to maintain data protection. For more information on lost write detection and instructions for repair from either ORA-752 or ORA-600 [3020] errors, see My Oracle Support [Document 1265884.1](https://support.oracle.com/DocumentDisplay?id=1265884.1).  
  
Primary performance overhead is negligible when lost write protection is enabled as the validation is performed by the standby database. For DSS and Data Warehouse applications, however, where there is a high volume of queries from non-READONLY tablespaces that are updated infrequently, the primary redo generation will increase. The redo apply throughput at the standby database can be impacted due to additional data block read I/Os. For most cases, however, the impact on the primary database and standby databases is negligible. The best way to estimate the impact on the DB\_LOST\_WRITE\_PROTECT on the primary or standby database is to enable it dynamically in your test environment and eventually in your production and standby databases.  
   
Monitor the change in the application or redo apply throughput and evaluate the database statistics described below. The primary redo rate and redo size will grow (as measured by database statistic redo size for lost write detection) after enabling this setting since block read redo (BRR) will be created to record the version of the block at subsequent reads. The size impact is mitigated since Oracle attempts to batch the BRRs in a single redo record if possible and a typical BRR redo change is only 144 bytes and redo header is 40 bytes in a 64 bit platform.  BRRs are recorded only for physical reads from disk since the goal is to detect lost writes or stale reads from disk. After enabling DB\_LOST\_WRITE\_PROTECT on a physical standby database, you will encounter an increase in physical reads especially for data that is updated very infrequently but read semi-frequently (forcing a physical read) on the primary database.  The additional physical reads on the physical standby database will be measured via database statistic recovery blocks read for lost write detection which can be extracted from the Standby Statspack report.

Starting with Oracle 11.2.0.4, there's a new Data Guard broker configuration-level property, [PrimaryLostWriteAction](http://docs.oracle.com/cd/E11882_01/server.112/e40771/dbpropref.htm#DGBKR3778). This property allows the user to choose whether the primary continues operation or is shutdown. The default value is CONTINUE.   Refer to **Oracle® Data Guard Broker 11g Release 2 (11.2)** documentation.

Setting DB\_LOST\_WRITE\_PROTECT on a non-Data Guard environment is still advantageous for better troubleshooting and debugging of lost writes problems.

Starting in Oracle 12.1, Active Data Guard with lost write protect can clearly detect lost writes on the standby.   When recovery process detects a lost write on the standby, the following error will be reported.

ORA-00753: recovery detected a lost write of a data block

Cause: A data block write to storage was lost during normal redo database operation on the standby database or during recovery on a primary database.

An example of the benefit of using DB\_LOST\_WRITE\_PROTECT, refer to  [**Data Guard Protection From Lost-Write Corruption demo**](http://www.oracle.com/technetwork/database/availability/lostwriteprotection-1867677.exe)at http://www.oracle.com/technetwork/database/features/availability/demonstrations-092317.html.

#### Oracle Automatic Storage Management (ASM) - Background

Read errors can be the result of a loss of access to the entire disk or media corruptions on an otherwise healthy disk. Oracle ASM tries to recover from read errors on corrupted sectors on a disk. When a read error by the database or Oracle ASM triggers the Oracle ASM instance to attempt bad block remapping, Oracle ASM reads a good copy of the extent and copies it to the disk that had the read error.

* If the write to the same location succeeds, then the underlying allocation unit (sector) is deemed healthy. This might be because the underlying disk did its own bad block reallocation.
* If the write fails, Oracle ASM attempts to write the extent to a new allocation unit on the same disk. If this write succeeds, the original allocation unit is marked as unusable. If the write fails, the disk is taken offline.

Another benefit with Oracle ASM based mirroring is that the database instance is aware of the mirroring. For many types of physical block corruptions such as a bad checksum, the database instance proceeds through the mirror side looking for valid content and proceeds without errors. If the process in the database that encountered the read can obtain the appropriate locks to ensure data consistency, it writes the correct data to all mirror sides.

When encountering a write error, a database instance sends the Oracle ASM instance a disk offline message.

* If database can successfully complete a write to at least one extent copy and receive acknowledgment of the offline disk from Oracle ASM, the write is considered successful.
* If the write to all mirror side fails, database takes the appropriate actions in response to a write error such as taking the tablespace offline.

When the Oracle ASM instance receives a write error message from a database instance or when an Oracle ASM instance encounters a write error itself, the Oracle ASM instance attempts to take the disk offline. Oracle ASM consults the Partner Status Table (PST) to see whether any of the disk's partners are offline. If too many partners are offline, Oracle ASM forces the dismounting of the disk group. Otherwise, Oracle ASM takes the disk offline.

The ASMCMD remap command was introduced to address situations where a range of bad sectors exists on a disk and must be corrected before Oracle ASM or database I/O. For information about the remap command.

When ASM detects any block corruptions, ASM logs the error to the ASM alert.log file.  The same corruption error may not appear in the database alert.log or application if ASM can correct the corruption automatically.

Starting Oracle 12c, Oracle ASM disk scrubbing checks logical data corruptions and repairs the corruptions automatically in normal and high redundancy disks groups. The feature is designed so that it does not have any impact to the regular input and output (I/O) operations in production systems. The scrubbing process repairs logical corruptions using the Oracle ASM mirror disks. Disk scrubbing uses Oracle ASM rebalancing to minimize I/O overhead.

The scrubbing process is visible in fields of the V$ASM\_OPERATION view.    Refer to **Oracle® Automatic Storage Management Administrator's Guide 12c Release 1 (12.1).**

These ASM benefits are available for all databases using ASM.   Since every Exadata Database Machine uses ASM, all these benefits are always available for Exadata customers.

#### Oracle Flashback Technologies - Background

Flashback Database is used for fast point-in-time recovery to recover from human errors that cause widespread damage to a production database. It is also used for fast reinstatement of a failed primary database as a new standby database following a Data Guard failover. Flashback Database uses flashback logs to rewind an Oracle Database to a previous point in time. See My Oracle Support [Document 565535.1](https://support.oracle.com/DocumentDisplay?id=565535.1) for Flashback Database best practices. See Section 13.2 of Data Guard Concepts and Administration for Fast Reinstatement using Flashback Database. Note that new optimizations in Oracle Database 11.2.0.2 reduce the impact on load operations when Flashback Database enabled.  
  
Flashback Database can also be used on a Data Guard standby database to enable point-in-time recovery of the primary database at a schema level without requiring that the entire primary database be flashed back. For example, take the case of a primary database hosting a number of applications and different schemas where an incorrect batch job was run for one of the applications. Assuming no object dependencies between schemas, you can perform a granular repair of the primary database in the following manner:

* Stop the application where point-in-time recovery is required and let other applications continue to run.
* Flashback the standby database to the desired point in time and extract a good copy of the affected schema.
* Repair the primary database by replacing the schema using the good copy from the standby, and restart the application.
* Restart redo apply on the standby database. It will resynchronize the standby by reapplying the bad batch job run and the reload that repaired it.

More granular levels of repair from human errors (e.g. drop table, inadvertent or malicious updates or DDLs) are enabled by other Oracle Flashback technologies that include: Flashback Query, Flashback Version Query, Flashback Transaction, Flashback Transaction Query, Flashback Table and Flashback Drop. Flashback Drop only requires configuring a Recycle Bin. All other features use automatic undo management. See Section 4.2.7 of Oracle Database HA best practices for detailed best practices on the full complement of Oracle Flashback Technologies.

Starting in Oracle 11g, manual RMAN block media recovery will automatically search flashback logs for good copies of blocks to help repair from physical data block corruptions quickly.

#### Oracle Data Guard

Oracle Data Guard ensures high availability, data protection, and disaster recovery for enterprise data.   One of the key benefits of Oracle Data Guard is its continuous Oracle-aware validation of all changes using multiple checks for physical and logical consistency of structures within an Oracle data block and redo before updates are applied to a standby database. This isolates the standby database from being impacted by data corruptions that can occur on the primary system.

##### Active Data Guard Automatic Block Repair - Background

If Oracle detects a physical block corruption on either the primary or a standby database in a configuration that uses Active Data Guard, Oracle will automatically repair the corrupt block using a valid copy from the other database. This repair is transparent to primary database applications and users. No application modifications are necessary. If the nature of the corruption makes it impossible to be repaired automatically (e.g. file header corruption, max block repair timeout of 60 seconds for one block repair or number of outstanding block corruptions reaching 100 block corruption incidents), an ORA-1578 error is returned to the application.    Manual block media recovery can be executed at a later time.  Automatic Block Repair requires an Active Data Guard standby and Data Guard real-time apply. The following database initialization parameters must be configured on the standby database: the following database initialization parameters are configured on the standby database:

* The LOG\_ARCHIVE\_CONFIG parameter is configured with a DG\_CONFIG list and a LOG\_ARCHIVE\_DEST\_n parameter is configured for the primary database
* The FAL\_SERVER parameter is configured and its value contains an Oracle Net service name for the primary database.

Active Data Guard auto block repair can fix physical block corruptions which is the most common type of block corruptions.    It does not address logical block corruptions which are normally prevented by setting DB\_BLOCK\_CHECKING on the primary and standby.   If enabling DB\_BLOCK\_CHECKING incurs an unacceptable performance impact on the primary, we recommend enabling on your standby database.  
  
Starting in Oracle 11g Release 2, manual RMAN block media recovery on the primary will automatically try to use a real-time query physical standby database.  
  
For an example of the benefits of using Active Data Guard auto block repair mechanisms, refer to http://www.oracle.com/technetwork/database/features/availability/demonstrations-092317.html

* **Automatic Block Repair at a Primary Database - Active Data Guard demo**
* **Automatic Block Repair at a Standby Database - Active Data Guard demo**

## Additional Operational Practices to detect block corruptions

1. Query v$database\_block\_corruption periodically on the primary and standby databases for any persistent data block corruptions
2. Scan and alert on any ORA-1578s in the  primary, standby and ASM alert.logs for every instance
3. Scan for IO errors in the database alert.log,  ASM alert.log and cell alert.log files    (Example: “IO Error”  or “I/O error”)
4. Execute RMAN backup and restore operations which will automatically check for physical block corruptions.   This is particularly helpful for data that is rarely queried or updated.
5. Periodically execute RMAN commands with “check logical” option to detect logical block corruptions.
6. Periodically execute low impact ASM disk scrubbing in the background.   This is not necessary for Exadata 11.2.3.3 and higher since Exadata’s Automatic HARD scrub and repair is automatically enabled
7. Use ANALYZE statement with the VALIDATE STRUCTURE option if you suspect corruptions due to inter-object or inter-block corruptions.    Refer to below for more information.

To verify the integrity of the structure of a table, index, cluster, or materialized view, use the ANALYZE statement with the VALIDATE STRUCTURE option. If the structure is valid, no error is returned. However, if the structure is corrupt, you receive an error message.

For example, in rare cases such as hardware or other system failures, an index can become corrupted and not perform correctly. When validating the index, you can confirm that every entry in the index points to the correct row of the associated table. If the index is corrupt, you can drop and re-create it.

If a table, index, or cluster is corrupt, you should drop it and re-create it. If a materialized view is corrupt, perform a complete refresh and ensure that you have remedied the problem. If the problem is not corrected, drop and re-create the materialized view.