

# Preventing, Detecting, and Repairing Data Corruption

*An Oracle White Paper*  
*May 2006*

# Preventing, Detecting, and Repairing Data Corruption

Introduction .....	3
Preventing Block Corruption .....	3
Data Guard .....	4
Detection of Corruption .....	4
Database Parameters .....	5
Recovery Manager .....	6
DBVERIFY .....	7
ANALYZE .. VALIDATE STRUCTURE .....	8
DBMS_REPAIR: Using the CHECK_OBJECT, ADMIN_TABLES, DUMP_ORPHAN_KEYS Procedures .....	9
Repairing Block Corruption.....	9
RMAN and Block Media Recovery.....	9
Repairing Corruption with Data Guard .....	10
RMAN and Media Recovery .....	11
Flashback Database .....	11
DBMS_REPAIR: SEGMENT_FIX_STATUS.....	11
Conclusion.....	11

# Preventing, Detecting, and Repairing Data Corruption

## INTRODUCTION

Data corruptions can stem from many different sources. For example, they can be caused by an errant bit-flip on a disk, or a bug in the operating system, storage area network (SAN), or storage system. If not prevented or repaired, data corruptions can bring down the database and even result in loss of key business data.

Oracle provides sophisticated techniques for detecting most block corruptions and recovering from them. These techniques include block-level recovery, automated backup and recovery, tablespace point-in-time recovery, remote standby databases, and transactional recovery. While these techniques make recovery possible, the process can take a long time (e.g. restoring from old tapes). Implementing techniques to prevent data corruptions can save much time, effort, and stress dealing with their possible consequences -- lost data and downtime.

This paper discusses the essential tools and techniques for prevention, detection, and repair of block corruptions in the Oracle database.

## PREVENTING BLOCK CORRUPTION

While 100% of all block corruptions cannot be completely prevented, there are a number of techniques that can be implemented to deter most of them. These include implementing H.A.R.D and using Data Guard.

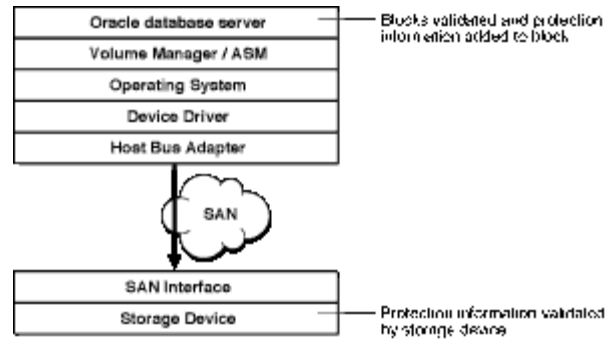
### H.A.R.D.

Oracle sponsors the Hardware Assisted Resilient Data (HARD) Initiative, a program designed to prevent data corruptions before they can be written to disk by the storage management software.

Under the HARD Initiative, Oracle works with selected system and storage vendors to build operating system and storage components that can detect corruptions early and prevent corrupted data from being written to disk. The key approach is block checking where the storage subsystem validates the Oracle block contents. Implementation of this feature is transparent to the end user and DBA, regardless of the hardware vendor.

To use HARD validation, datafiles and log files are placed on HARD-compliant storage. The user then enables the HARD validation feature on the storage. When

Oracle writes data to the storage, the storage system validates the data. If it appears to be corrupted, then the write is rejected with an error.



**Figure 1 Oracle Data Validation**

The HARD program defines multiple levels of protection. Oracle storage partners may choose to implement firmware or hardware checks that can prevent the following types of corruptions:

- Writes of corrupted blocks
- Writes of blocks to incorrect locations
- Erroneous writes by programs other than Oracle to Oracle data

For more information on HARD, refer to the [OTN HARD Overview](#).

### **Data Guard**

Oracle Data Guard ensures high availability, data protection, and disaster recovery for enterprise data. Data Guard provides a comprehensive set of services that create, maintain, manage, and monitor one or more standby databases to enable production Oracle databases to survive disasters and data corruptions.

A standby database provides a safeguard against data corruption and user errors:

- Storage level physical corruptions on the primary database are not propagated to the standby database.
- Logical corruptions or user errors that cause the primary database to be permanently damaged can be resolved by delaying the application of redo or by implementing Flashback Database at the standby site.
- Redo data is validated when it is applied to the standby database.

For more information on Data Guard and Flashback, refer to the [OTN Data Guard Overview](#) and [OTN Flashback Overview](#).

### **DETECTION OF CORRUPTION**

If corrupt data is written to disk, despite all preventative measures, or if a disk or controller error causes good data to become corrupt after it is written, it is

necessary to detect those blocks. There are two important database parameters that can be enabled to detect datafile corruptions, as reads and writes are occurring – `DB_BLOCK_CHECKSUM` and `DB_BLOCK_CHECKING`. In addition, Oracle provides four tools to validate data on demand: Recovery Manager (RMAN), `DBVERIFY`, `ANALYZE`, and `DBMS_REPAIR`.

## Database Parameters

Database administrators can use initialization parameters to:

- Optimize performance by adjusting memory structures, such as the number of database buffers in memory
- Set database-wide defaults, such as the amount of space initially allocated for a context area when it is created
- Set database limits, such as the maximum number of database users
- Specify names of files or directories required by the database

There are two parameters that can be set to detect block corruption in the database. They are `DB_BLOCK_CHECKSUM` and `DB_BLOCK_CHECKING`.

### DB\_BLOCK\_CHECKSUM

`DB_BLOCK_CHECKSUM` (default value is `TYPICAL`) determines whether the database will calculate a checksum for the block and store it in the block each time it is written to disk. When a block is read, if it contains a checksum that was stored the last time the block was written, the checksum is used to verify that the block is not corrupt. If this parameter is set to `OFF`, checksums are calculated only for the `SYSTEM` tablespace, but not for user tablespaces.

When the parameter is set to `FULL`, Oracle also verifies the checksum before a change application from update/delete statements and recomputes it after the change is applied. Also, Oracle gives every log block a checksum before writing it to the current log. Enabling `FULL` mode typically incurs an additional 4-5% additional overhead on the system.

While the database is operational and corruptions are detected, they will be recorded as `ORA-600` or `ORA-01578` in the alert log. Note that this checksum does not ensure logical consistency of the blocks contents, but only performs a lightweight checksum to catch on-disk errors, such as bit flips.

Checksums allow Oracle to detect corruption caused by underlying disks, storage systems, or I/O systems. Setting block checksum to `TYPICAL` should only incur an additional 1-2% overhead on the system. Therefore, this parameter is set to `TYPICAL` by default.

### DB\_BLOCK\_CHECKING

When `DB_BLOCK_CHECKING` is set to `FULL`, Oracle performs block checking for all data blocks, including block header and user data. When it is set to `OFF`

(default), Oracle does not perform block checking for blocks in the user tablespaces, with the exception of the SYSTEM tablespace, which is always checked. In addition, the space metadata (bitmap and segment header blocks) is always block-checked.

In addition, LOW and MEDIUM settings are provided. When DB\_BLOCK\_CHECKING is set to 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 a Real Application Cluster (RAC) configuration). When set to MEDIUM, all LOW checks are performed, as well as semantic block checking for all non-index-organized table blocks.

When DB\_BLOCK\_CHECKING is set to FULL, Oracle checks a block by going through the data on the block, making sure it is self-consistent. This means that the block header is coherent with data, rows are coherent, and column pieces look good (e.g. column lengths are not corrupt, rows do not overlap over each other). Block checking often catches memory and data corruption, that would otherwise pass through checksum verification. While the database is operational and corruptions are detected, they will be recorded as ORA-600 or ORA-01578 in the alert log.

Block checking typically incurs 1-10% overhead on the system, depending on workload. The more updates or inserts in a workload, the more expensive it is to turn on block checking. You should enable DB\_BLOCK\_CHECKING if the performance overhead is acceptable.

## Recovery Manager

Implementing a good backup and recovery strategy is required to ensure protection from media corruption. Oracle's Recovery Manager (RMAN) is the best utility to backup the Oracle Database. RMAN knows what files require a backup, but most importantly, it knows what files are needed for recovery.

When reading data for backup, RMAN always verifies the block checksums, if present. Optionally, RMAN can perform logical block validation by specifying the CHECK LOGICAL option during the backup. CHECK LOGICAL tests data and index blocks that pass physical corruption checks for logical corruption, for example, corruption of a row piece or index entry.

By default, RMAN stops the backup operation when 1 corrupt block is found. You can control this by setting MAXCORRUPT. Setting MAXCORRUPT greater than 1 allows a specified number of previously undetected block corruptions in datafiles during the execution of an RMAN BACKUP command. If RMAN detects more than this number of corruptions while taking the backup, then the command terminates.

There are some considerations to note if you utilize MAXCORRUPT. If the server session encounters a datafile block during a backup that has already been identified as corrupt by the database, then the server session copies the corrupt block into the

backup and the corrupt block is recorded in the control file as either a logical or media corruption. RMAN copies the block in case the user wants to try to salvage the contents of the block. If RMAN encounters a datafile block that is media corrupt but that has not already been identified as corrupt by the database, then it writes the block to the backup with a reformatted header indicating that the block has media corruption. The `V$DATABASE_BLOCK_CORRUPTION` view is updated with newly detected corruptions.

As a best practice, if a corrupt block is found during the backup, you should use the `RMAN BACKUP VALIDATE` command to check for any other possible corruptions. The `VALIDATE` option reads the entire datafile but does not write a physical backup. If corrupt blocks are found, `VALIDATE` will populate the view `V$DATABASE_BLOCK_CORRUPTION`.

Archivelogs can also be corrupted due to I/O errors or software bugs. RMAN validates archivelogs when they are backed up. During backup of archived redo logs, RMAN checks the file being backed up for corruption. If a corruption is found, RMAN automatically switches to reading another copy of the same archived redo log, if one exists. By validating archivelogs when they are backed up, RMAN ensures that complete recovery of the database is possible.

Once a backup has been created, you can use the `RMAN RESTORE VALIDATE` command to verify your backup is still available and reliable to use for recovery. RMAN decides which backup sets, datafile copies, and archived logs need to be restored and then scans them to verify their contents. No files will be restored. Use this option periodically to verify that the copies and backup sets required to restore your database are intact and usable.

Use the following practices to get the strongest possible corruption protection with your RMAN backups:

- In the initialization parameter file, set `DB_BLOCK_CHECKSUM=TRUE`
- In `BACKUP` and `RESTORE` commands, do *not* specify the `MAXCORRUPT` option, do *not* specify the `NOCHECKSUM` option, but *do* specify the `CHECK LOGICAL` option

For more information on RMAN, refer to the [Backup and Recovery Basics Documentation](#).

## DBVERIFY

`DBVERIFY` is a command-line utility that performs a physical data structure integrity check. It can be used on offline or online databases, and on backup copies of datafiles. You can use `DBVERIFY` when you need to ensure that a backup copy of the database (or set of datafiles) is valid before it is restored or as a diagnostic aid when you have encountered data corruption problems. `DBVERIFY` is only able to check datafile blocks and does not work with control files, redo logs, or archive logs.

Segments can be validated in DBVERIFY. In this mode, a table segment or index segment is specified. The tool checks to make sure that a correct row chain pointer is within the segment being verified. Furthermore, this mode requires SYSDBA privileges so that additional information about the segment can be retrieved from the database. During the operation, the segment is locked. If the specified segment is an index, the parent table is locked. If corruptions are detected, DBV will report them in its output.

For more information on DBVERIFY, refer to the [UTILITIES documentation](#).

## ANALYZE .. VALIDATE STRUCTURE

The ANALYZE command in SQL\*Plus provides capabilities to:

- collect or delete statistics about an index or index partition, table or table partition, index-organized table, cluster, or scalar object attribute.
- identify migrated and chained rows of a table or cluster.
- validate the structure of an index or index partition, table or table partition, index-organized table, cluster, or object reference (*REF*).

The VALIDATE STRUCTURE option of ANALYZE validates the structure of the analyzed object.

- For a table, the integrity of each data block and row is verified.
- For a cluster, the structure of the cluster tables is also verified.
- For a partitioned table, Oracle Database also verifies that each row belongs to the correct partition. If a row does not correlate to its partition correctly, then its rowid is inserted into the INVALID\_ROWS table.
- For a temporary table, Oracle Database validates the structure of the table and its indexes during the current session.
- For an index, Oracle Database verifies the integrity of each data block in the index and checks for block corruption. This clause does not confirm that each row in the table has an index entry or that each index entry points to a row in the table. You can perform these operations by validating the structure of the table with the CASCADE clause.

If Oracle Database encounters corruption in the structure of the object, then an error message is returned to you.

You can specify that you want to perform structure validation online while DML is occurring against the object being validated. There can be a slight performance impact when validating with ongoing DML affecting the object, but this is offset by the flexibility of being able to perform ANALYZE online. For example, the following statement validates the emp table and all associated indexes online:

```
ANALYZE TABLE emp VALIDATE STRUCTURE CASCADE ONLINE;
```



While database is operational and corruptions are detected by `ANALYZE`, they will be recorded as `ORA-600` or `ORA-01578` in the alert log.

For more information on `ANALYZE . . . VALIDATE STRUCTURE` refer to the [SQL Reference Documentation](#).

### **DBMS\_REPAIR: Using the CHECK\_OBJECT, ADMIN\_TABLES, DUMP\_ORPHAN\_KEYS Procedures**

The `DBMS_REPAIR` package contains procedures to detect corrupt blocks in tables and indexes. The `CHECK_OBJECT` procedure checks and reports block corruptions for a specified object. This procedure is similar to the `ANALYZE . . . VALIDATE STRUCTURE` statement for indexes and tables.

Before `CHECK_OBJECT` is run, a repair table must first be created using the `ADMIN_TABLES` procedure. All corrupted blocks that are detected are recorded in this table. In addition, any recommended actions are recorded, e.g. marking a block soft corrupt, so that the block can be skipped and the object can still be accessed. Before a block is marked soft corrupt, it may still be possible to extract any good data. These actions can then be carried out by running the `FIX_CORRUPT_BLOCKS` procedure on the object.

The `DUMP_ORPHAN_KEYS` procedure reports on index entries that point to rows in corrupt data blocks. All such index entries are inserted into an orphan key table (previously created with the `ADMIN_TABLES` procedure) that stores the key and rowid of the corruption. The index entry information can be used for rebuilding lost rows in the table, and rebuilding the index using the `ALTER INDEX . . . REBUILD ONLINE` statement. This procedure should be run for each index associated with a table listed in the repair table.

For more information on the procedures contained in the `DBMS_REPAIR` package, refer to the [Administrator's Guide](#).

### **REPAIRING BLOCK CORRUPTION**

Once a corrupt block is found, Oracle provides options that will ensure that data in the corrupted block can be recovered. Whatever method is used to recover corrupted blocks, you need to analyze the type and degree of corruption to perform the recovery. For example, if the object that is corrupted is an index, just drop and recreate the index. If a table is affected and blocks have been marked soft corrupt with `DBMS_REPAIR`, performing `CREATE TABLE . . . AS SELECT` from the corrupt table will create a new table and the affected rows can be manually re-inserted.

The recovery methods discussed below can only be used to recover data blocks.

### **RMAN and Block Media Recovery**

Although datafile media recovery is the principal form of recovery, you can also use the `RMAN BLOCKRECOVER` command to perform block media recovery (BMR).

BMR recovers an individual corrupt datablock or set of datablocks within a datafile. In cases when a small number of blocks require media recovery, you can just restore and recover damaged blocks identified by the database, rather than whole datafiles, using one command:

```
BLOCKRECOVER CORRUPTION LIST;
```

Block media recovery provides several advantages over datafile media recovery.

- Lowers the Mean Time to Recovery (MTTR) because only blocks needing recovery are restored and only the necessary corrupt blocks undergo recovery.
- Block media recovery minimizes redo application time and avoids I/O overhead during recovery.
- Allows affected datafiles to remain online during recovery of the blocks.

Block Media Recovery can only be performed by RMAN using a full RMAN backup or image copy of the datafile. In addition, you can only perform complete recovery of individual blocks. In other words, you cannot stop recovery before all redo has been applied to the block. The `V$DATABASE_BLOCK_CORRUPTION` view indicates which blocks in a file were marked corrupt since the most recent `BACKUP` or `BACKUP . . . VALIDATE` command was run against the file.

After a corrupt block is repaired, the row identifying this block is deleted from the `V$DATABASE_BLOCK_CORRUPTION` view.

For more information on RMAN's BMR, refer to the [Backup & Recovery Advanced User's Guide](#).

## Repairing Corruption with Data Guard

Data Guard not only detects a logical corruption, but the physical standby database can also be used to repair corruption on the primary database. If you cannot use RMAN's BMR to repair the corrupted block, the data may be extracted from the standby database. There are two options using a standby database that can be used to repair block corruption on the primary database:

- Extract the rows from the block that is corrupted on the primary by using Data Pump or other means to select the data from a table. The data is then re-inserted into the primary database.
- Copy the standby database datafile(s) to the primary database. Once the file is restored on the primary database, archive logs are applied to bring it consistent with the rest of the database.

If the primary database corruption is widespread due to a bad controller or other hardware/software problem, then you may want to switchover to the standby database while repairs to the primary database server are made.

For more information on Data Guard switchovers, refer to the [Data Guard Concepts and Administration](#).

## **RMAN and Media Recovery**

Media recovery is the 'traditional' type of database repair, in which a backup copy of one or more files is restored, and then changes from archived redo logs are applied to the restored files to bring the database back to the current time or a prior point-in-time. If corruption is widespread in a datafile, then restoring a backup of the datafile and applying redo may repair the corruptions.

For more information on RMAN and Media Recovery, refer to the [Backup and Recovery Basics](#) documentation.

## **Flashback Database**

For database-wide logical corruptions caused by human or application logic errors, Oracle provides Flashback Database, which can take a database back to a point-in-time in the past within a matter of minutes. This is much less time than the hours that might be incurred with traditional restore and recovery from tape. Refer to the [Backup and Recovery Basics](#) documentation for more information on Flashback Database.

## **DBMS\_REPAIR: SEGMENT\_FIX\_STATUS**

As mentioned previously, the DBMS\_REPAIR package can be used to detect block corruptions, and mark them soft corrupt so that they can be skipped on table and index scans, allowing the objects themselves to still be usable.

The SEGMENT\_FIX\_STATUS procedure can be used to repair the corrupted state of a bitmap entry, if the free space in segments is being managed with bitmaps (SEGMENT SPACE MANAGEMENT AUTO). The procedure recalculates the state based on the current contents of the blocks in a specified segment. This procedure is useful for tables and indexes whose blocks have been marked soft corrupt or manually rebuilt and reinstated.

For more information on using the DBMS\_REPAIR package, refer to the [PL/SQL Packages and Types Reference](#).

## **CONCLUSION**

The best medicine for addressing block corruption is prevention. Oracle sponsors the HARD Initiative in which leading storage vendors implement Oracle block validation algorithms at the storage level, to prevent a corrupt write to disk from occurring. Data Guard provides an added level of protection by preventing storage-level corruptions from propagating to the standby database and re-validating redo before it is applied to the standby database.

Even if corruptions occur, setting the two database parameters DB\_BLOCK\_CHECKSUM and DB\_BLOCK\_CHECKING will allow the server to flag them to the user on normal reads and writes. Data file corruptions are also detected by RMAN backup and restore operations, DBVerify, ANALYZE, and the DBMS\_REPAIR package.

Once corruptions are isolated, Oracle offers BMR, Data Guard, and media recovery to recover the data. Database-wide logical corruptions caused by human or application errors can be easily undone with Flashback Database. Ultimately, your corruption recovery strategy is only as good as you have tested it. Whatever method is employed to prevent or recover from corruptions, all recovery strategies should always be thoroughly and regularly tested.



Preventing, Detecting, and Repairing Data Corruption

May 2006

Authors: Tammy Bednar, Dheeraj Pandey, Steve Wertheimer, Bharat Baddepudi, Timothy Chien

Contributing Authors:

Oracle Corporation  
World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065  
U.S.A.

Worldwide Inquiries:

Phone: +1.650.506.7000

Fax: +1.650.506.7200

[oracle.com](http://oracle.com)

Copyright © 2005, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission. Oracle, JD Edwards, and PeopleSoft are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.