9

Diagnosing the Database

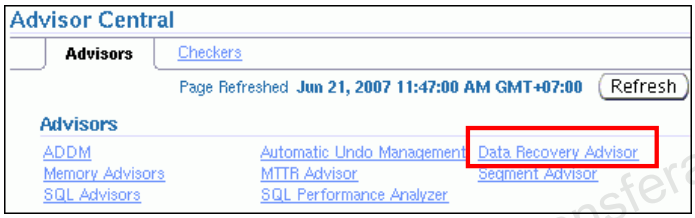
Objectives

After completing this lesson, you should be able to:

- Detect and repair database corruption
- Handle block corruption
- Set up Automatic Diagnostic Repository
- Run health checks

Data Recovery Advisor

- Fast detection, analysis, and repair of failures
- Minimizing disruptions for users
- Down-time and run-time failures
- User interfaces:
  - EM GUI interface (several paths)
  - RMAN command line



- Supported database configurations:
  - Single-instance
  - Not RAC
  - Supporting failover to standby, but not analysis and repair of standby databases

ORACLE

Data Recovery Advisor

The Data Recovery Advisor automatically gathers data failure information when an error is encountered. In addition, it can proactively check for failures. In this mode, it can potentially detect and analyze data failures before a database process discovers the corruption and signals an error. (Note that repairs are always under human control.)

Data failures can be very serious. For example, if your current log files are missing, you cannot start your database. Some data failures (such as block corruptions in data files) are not catastrophic, in that they do not take the database down or prevent you from starting the Oracle instance. The Data Recovery Advisor handles both cases: the one when you cannot start up the database (because some required database files are missing, inconsistent, or corrupted) and the one when file corruptions are discovered during run time.

User Interfaces

The Data Recovery Advisor is available from Enterprise Manager (EM) Database Control and Grid Control. When failures exist, there are several ways to access the Data Recovery Advisor. The following examples all begin on the Database Instance home page:

- Availability tabbed page > Perform Recovery > Advise and Recover
- Active Incidents link > on the Support Workbench “Problems” page: Checker Findings tabbed page > Launch Recovery Advisor
- Database Instance Health > click the specific link, for example, ORA 1578 in the Incidents section > Support Workbench, Problems Detail page > Data Recovery Advisor
- Database Instance Health > Related Links section: Support Workbench > Checker Findings tabbed page: Launch Recovery Advisor
- Related Link: Advisor Central > Advisors tabbed page: Data Recovery Advisor
- Related Link: Advisor Central > Checkers tabbed page: Details > Run Detail tabbed page: Launch Recovery Advisor

You can also use it via the RMAN command-line. For example:

```
rman target / nocatalog
rman> list failure all;
```

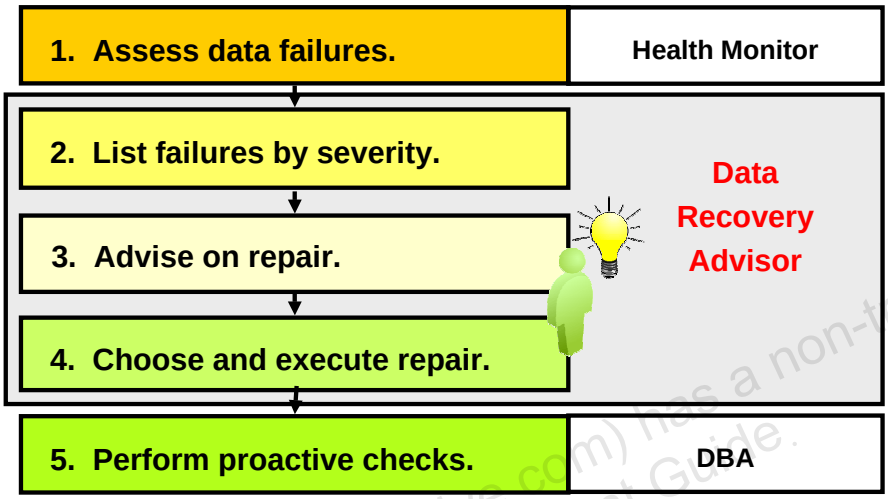
Supported Database Configurations

In the current release, the Data Recovery Advisor supports single-instance databases. Oracle Real Application Clusters (RAC) databases are not supported.

The Data Recovery Advisor cannot use blocks or files transferred from a standby database to repair failures on a primary database. Also, you cannot use the Data Recovery Advisor to diagnose and repair failures on a standby database. However, the Data Recovery Advisor does support failover to a standby database as a repair option (as mentioned above).

Data Recovery Advisor

Reducing down time by eliminating confusion:



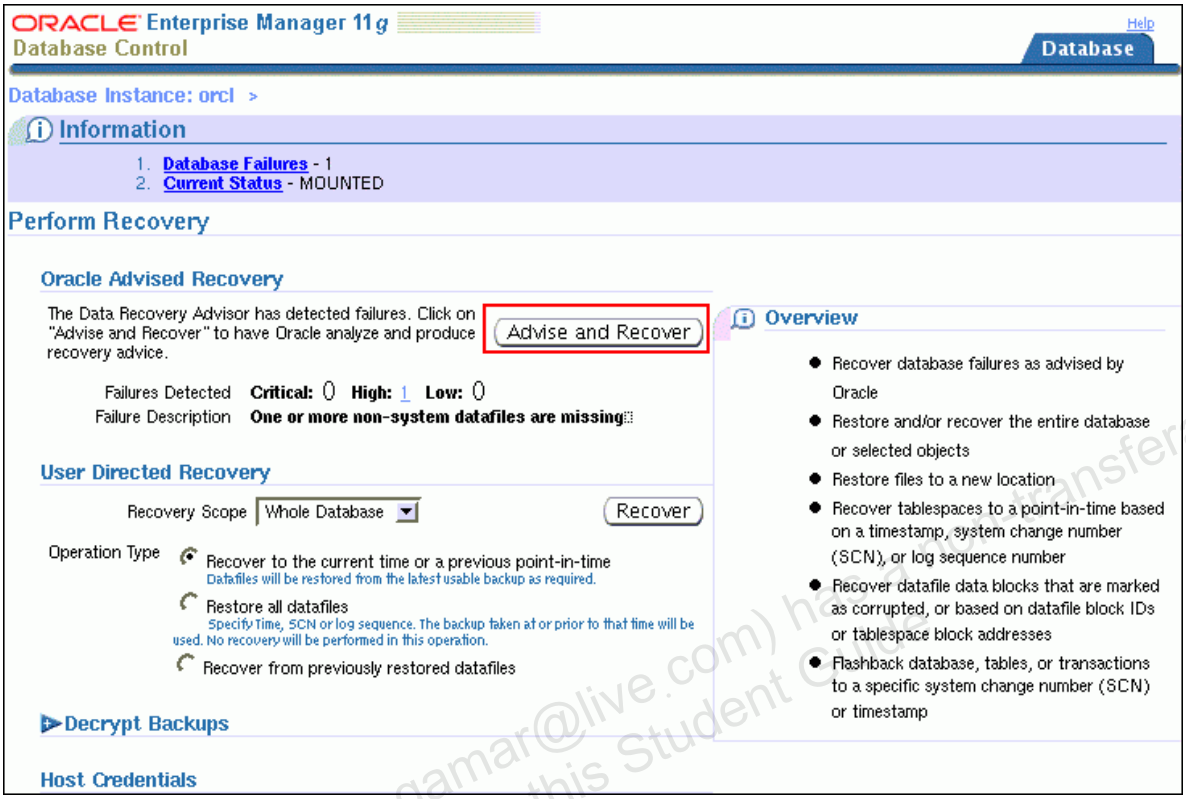
ORACLE

Data Recovery Advisor

The automatic diagnostic workflow in Oracle Database 11*g* performs as follows. With the Data Recovery Advisor, you only need to initiate an advice and a repair.

1. The Health Monitor automatically executes checks and logs failures and their symptoms as "findings" into the Automatic Diagnostic Repository (ADR).
2. The Data Recovery Advisor consolidates findings into failures. It lists the results of previously executed assessments with failure severity (critical or high).
3. When you ask for repair advice on a failure, the Data Recovery Advisor maps failures to automatic and manual repair options, checks basic feasibility, and presents you with the repair advice.
4. You can choose to manually execute a repair or request the Data Recovery Advisor to do it for you.
5. In addition to the automatic, primarily "reactive" checks of the Health Monitor and Data Recovery Advisor, Oracle recommends to additionally use the `VALIDATE` command as a "proactive" check.

Data Failures



Data Failures

Data failures are detected by checks, which are diagnostic procedures that assess the health of the database or its components. Each check can diagnose one or more failures, which are mapped to a repair.

Checks can be reactive or proactive. When an error occurs in the database, “reactive checks” are automatically executed. You can also initiate “proactive checks”, for example, by executing the `VALIDATE DATABASE` command.

In Enterprise Manager, select Availability > Perform Recovery, or click the Perform Recovery button, if you find your database in a “down” or “mounted” state.

Data Failure: Examples

- Not accessible components, for example:
  - Missing data files at the OS level
  - Incorrect access permissions
  - Offline tablespace, and so on

- Physical corruptions, such as block checksum failures or invalid block header field values
- Logical corruptions, such as inconsistent dictionary, corrupt row piece, corrupt index entry, or corrupt transaction
- Inconsistencies, such as control file is older or newer than the data files and online redo logs
- I/O failures, such as a limit on the number of open files exceeded, channels inaccessible, network or I/O error

ORACLE

Data Failure: Examples

The Data Recovery Advisor can analyze failures and suggest repair options for issues, as outlined in the slide.

Data Recovery Advisor
RMAN Command-Line Interface

RMAN Command	Action
LIST FAILURE	Lists previously executed failure assessment
ADVISE FAILURE	Displays recommended repair option
REPAIR FAILURE	Repairs and closes failures (after ADVISE in the same RMAN session)
CHANGE FAILURE	Changes or closes one or more failures

Data Recovery Advisor: RMAN Command-Line Interface

If you suspect or know that a database failure has occurred, then use the LIST FAILURE command to obtain information about these failures. You can list all or a subset of failures and restrict output in various ways. Failures are uniquely identified by failure numbers. Note that these numbers are not consecutive, so gaps between failure numbers have no significance.

The ADVISE FAILURE command displays a recommended repair option for the specified failures. It prints a summary of the input failure and implicitly closes all open failures that are already fixed. The default behavior when no option is used is to advise on all the CRITICAL and HIGH priority failures that are recorded in ADR.

The REPAIR FAILURE command is used after an ADVISE FAILURE command **within the same RMAN session**. By default, the command uses the single, recommended repair option of the last ADVISE FAILURE execution in the current session. If none exists, the REPAIR FAILURE command initiates an implicit ADVISE FAILURE command. After completing the repair, the command closes the failure.

The CHANGE FAILURE command changes the failure priority or closes one or more failures. You can change a failure priority only for HIGH or LOW priorities. Open failures are closed implicitly when a failure is repaired. However, you can also explicitly close a failure.

Listing Data Failures

The RMAN `LIST FAILURE` command lists previously executed failure assessment.

- Including newly diagnosed failures
- Removing closed failures (by default)

Syntax:

```
LIST FAILURE
[ ALL | CRITICAL | HIGH | LOW | CLOSED |
   failnum[,failnum,…] ]
[ EXCLUDE FAILURE failnum[,failnum,…] ]
[ DETAIL ]
```

ORACLE

Listing Data Failures

The RMAN `LIST FAILURE` command lists failures. If the target instance uses a recovery catalog, it can be in `STARTED` mode, otherwise it must be in `MOUNTED` mode. The `LIST FAILURE` command does not initiate checks to diagnose new failures; rather, it lists the results of previously executed assessments. Repeatedly executing the `LIST FAILURE` command revalidates all existing failures. If the database diagnoses new ones (between command executions), they are displayed. If a user manually fixes failures, or if transient failures disappear, then the Data Recovery Advisor removes these failures from the `LIST FAILURE` output. The following is a description of the syntax:

- `failnum`: Number of the failure to display repair options for
- `ALL`: List failures of all priorities.
- `CRITICAL`: List failures of `CRITICAL` priority and `OPEN` status. These failures require immediate attention, because they make the whole database unavailable (for example, a missing control file).
- `HIGH`: List failures of `HIGH` priority and `OPEN` status. These failures make a database partly unavailable or unrecoverable; so they should be repaired quickly (for example, missing archived redo logs).
- `LOW`: List failures of `LOW` priority and `OPEN` status. Failures of a low priority can wait until more important failures are fixed.

Listing Data Failures (continued)

- `CLOSED`: List only closed failures.
- `EXCLUDE FAILURE`: Exclude the specified list of failure numbers from the list.
- `DETAIL`: List failures by expanding the consolidated failure. For example, if there are multiple block corruptions in a file, the `DETAIL` option lists each one of them.

See the *Oracle Database Backup and Recovery Reference* for details of the command syntax.

Advising on Repair

The RMAN ADVISE FAILURE command:

- Displays a summary of input failure list
- Includes a warning, if new failures appeared in ADR
- Displays a manual checklist
- Lists a single recommended repair option
- Generates a repair script (for automatic or manual repair)

```
. . .
Repair script:
   /u01/app/oracle/diag/rdbms/orcl/orcl/hm/reco_2979
   128860.hm
RMAN>
```

ORACLE

Advising on Repair

The RMAN ADVISE FAILURE command displays a recommended repair option for the specified failures. The ADVISE FAILURE command prints a summary of the input failure. The command implicitly closes all open failures that are already fixed.

The default behavior (when no option is used) is to advise on all the CRITICAL and HIGH priority failures that are recorded in Automatic Diagnostic Repository (ADR). If a new failure has been recorded in ADR since the last LIST FAILURE command, this command includes a WARNING before advising on all CRITICAL and HIGH failures.

Two general repair options are implemented: no-data-loss and data-loss repairs.

When the Data Recovery Advisor generates an automated repair option, it generates a script that shows you how RMAN plans to repair the failure. If you do not want the Data Recovery Advisor to automatically repair the failure, then you can use this script as a starting point for your manual repair. The operating system (OS) location of the script is printed at the end of the command output. You can examine this script, customize it (if needed), and also execute it manually if, for example, your audit trail requirements recommend such an action.

Syntax

```
ADVISE FAILURE
[ ALL | CRITICAL | HIGH | LOW | failnum[,failnum,…] ]
[ EXCLUDE FAILURE failnum [,failnum,…] ]
```

Executing Repairs

The RMAN REPAIR FAILURE command:

- Follows the ADVISE FAILURE command
- Repairs the specified failure
- Closes the repaired failure

Syntax:

```
REPAIR FAILURE
  [USING ADVISE OPTION integer]
  [ { {NOPROMPT | PREVIEW}}...]
```

ORACLE

Executing Repairs

This command should be used after an ADVISE FAILURE command in the same RMAN session. By default (with no option), the command uses the single, recommended repair option of the last ADVISE FAILURE execution in the current session. If none exists, the REPAIR FAILURE command initiates an implicit ADVISE FAILURE command.

With USING ADVISE OPTION *integer*, you specify your desired repair option by its option number (from the ADVISE FAILURE command); this is not the failure number.

By default, you are asked to confirm the command execution because you may be requesting substantial changes that take time to complete. During the execution of a repair, the output of the command indicates what phase of the repair is being executed.

After completing the repair, the command closes the failure.

You cannot run multiple concurrent repair sessions. However, concurrent REPAIR … PREVIEW sessions are allowed.

- PREVIEW means: Do not execute the repairs; instead, display the previously generated RMAN script with all repair actions and comments.
- NOPROMPT means: Do not ask for confirmation.

Classifying (and Closing) Failures

The RMAN CHANGE FAILURE command:

- Changes the failure priority (except for CRITICAL)
- Closes one or more failures

Example:

```
RMAN> change failure 5 priority low;
List of Database Failures
=========================
Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
5          HIGH     OPEN      20-DEC-06     one or more
    datafiles are missing
Do you really want to change the above failures (enter YES or
    NO)? yes
changed 1 failures to LOW priority
```

ORACLE

Classifying (and Closing) Failures

The CHANGE FAILURE command is used to change the failure priority or close one or more failures.

Syntax

```
CHANGE FAILURE
{ ALL | CRITICAL | HIGH | LOW | failnum[,failnum,…] }
[ EXCLUDE FAILURE failnum[,failnum,…] ]
{ PRIORITY {CRITICAL | HIGH | LOW} |
CLOSE }
```

– change status of the failure(s) to closed

[NOPROMPT] – do not ask user for a confirmation

A failure priority can be changed only from HIGH to LOW and from LOW to HIGH. It is an error to change the priority level of CRITICAL. (One reason why you may want to change a failure from HIGH to LOW is to avoid seeing it on the default output list of the LIST FAILURE command. For example, if a block corruption has HIGH priority, you may want to temporarily change it to LOW if the block is in a little-used tablespace.)

Open failures are closed implicitly when a failure is repaired. However, you can also explicitly close a failure. This involves a reevaluation of all other open failures, because some of them may become irrelevant as the result of the closure of the failure.

By default, the command asks the user to confirm a requested change.

Data Recovery Advisor Views

Querying V$ views:

- V$IR_FAILURE: List of all failures, including closed ones (result of the LIST FAILURE command)
- V$IR_MANUAL_CHECKLIST: List of manual advice (result of the ADVISE FAILURE command)
- V$IR_REPAIR: List of repairs (result of the ADVISE FAILURE command)
- V$IR_FAILURE_SET: Cross-reference of failure and advice identifiers

Data Recovery Advisor Views

See the Oracle Database Reference for details of the dynamic data dictionary views that the Data Recovery Advisor uses.

Best Practice: Proactive Checks

Invoking proactive health check of the database and its components:

- Health Monitor or RMAN `VALIDATE DATABASE` command
- Checking for logical and physical corruption
- Findings logged in the ADR

Best Practice: Proactive Checks

For very important databases, you may want to execute additional proactive checks (possibly daily during low peak interval periods). You can schedule periodic health checks through the Health Monitor or by using the RMAN `VALIDATE` command. In general, when a reactive check detects failure(s) in a database component, you may want to execute a more complete check of the affected component.

The RMAN `VALIDATE DATABASE` command is used to invoke health checks for the database and its components. It extends the existing `VALIDATE BACKUPSET` command. Any problem detected during validation is displayed to you. Problems initiate the execution of a failure assessment. If a failure is detected, it is logged into ADR as a finding. You can use the `LIST FAILURE` command to view all failures recorded in the repository.

The `VALIDATE` command supports validation of individual backup sets and data blocks. In a physical corruption, the database does not recognize the block at all. In a logical corruption, the contents of the block are logically inconsistent. By default, the `VALIDATE` command checks for physical corruption only. You can specify `CHECK LOGICAL` to check for logical corruption as well.

Block corruptions can be divided into interblock corruption and intrablock corruption. In intrablock corruption, the corruption occurs within the block itself and can be either physical or logical corruption. In interblock corruption, the corruption occurs between blocks and can be only logical corruption. The `VALIDATE` command checks for intrablock corruptions only.

What Is Block Corruption?

- Whenever a block is read or written, a consistency check is performed.
  - Block version
  - DBA (data block address) value in cache as compared to the DBA value in the block buffer
  - Block-checksum, if enabled
- A corrupt block is identified as being one of the following:
  - Media corrupt
  - Logically (or software) corrupt

What Is Block Corruption?

A corrupted data block is a block that is not in a recognized Oracle format, or whose contents are not internally consistent. Typically, corruptions are caused by faulty hardware or operating system problems. The Oracle database identifies corrupt blocks as either “logically corrupt” or “media corrupt.” If it is logically corrupt, there is an Oracle internal error. Logically corrupt blocks are marked corrupt by the Oracle database after it detects the inconsistency. If it is media corrupt, the block format is not correct; the information in the block does not make any sense after being read from disk.

As you just learned, a number of data failures and corruptions can be repaired with the Data Recovery Advisor. Now you learn about a manual way to diagnose and repair corruptions.

You can repair a media corrupt block by recovering the block or dropping the database object that contains the corrupt block, or both. If media corruption is due to faulty hardware, the problem will not be completely resolved until the hardware fault is corrected.

Block Corruption Symptoms: ORA-01578

The error ORA-01578: "ORACLE data block corrupted (file # %s, block # %s)":

- Is generated when a corrupted data block is found
- Always returns the relative file number and block number
- Is returned to the session that issued the query being performed when the corruption was discovered
- Appears in the alert.log file

Block Corruption Symptoms: ORA-01578

Usually, the ORA-01578 error is the result of a hardware problem. If the ORA-01578 error is always returned with the same arguments, it is most likely a media corrupt block.

If the arguments change each time, there may be a hardware problem, and you should have the memory and page space checked, and the I/O subsystem checked for bad controllers.

Note: ORA-01578 returns the relative file number, but the accompanying ORA-01110 error displays the absolute file number.

How to Handle Corruption

- Check the alert log and operating system log file.
- Use available diagnostic tools to find out the type of corruption.
- Determine whether the error persists by running checks multiple times.
- Recover data from the corrupted object if necessary.
- Resolve any hardware issues:
  - Memory boards
  - Disk controllers
  - Disks
- Recover or restore data from the corrupt object if necessary.

ORACLE

How to Handle Corruption

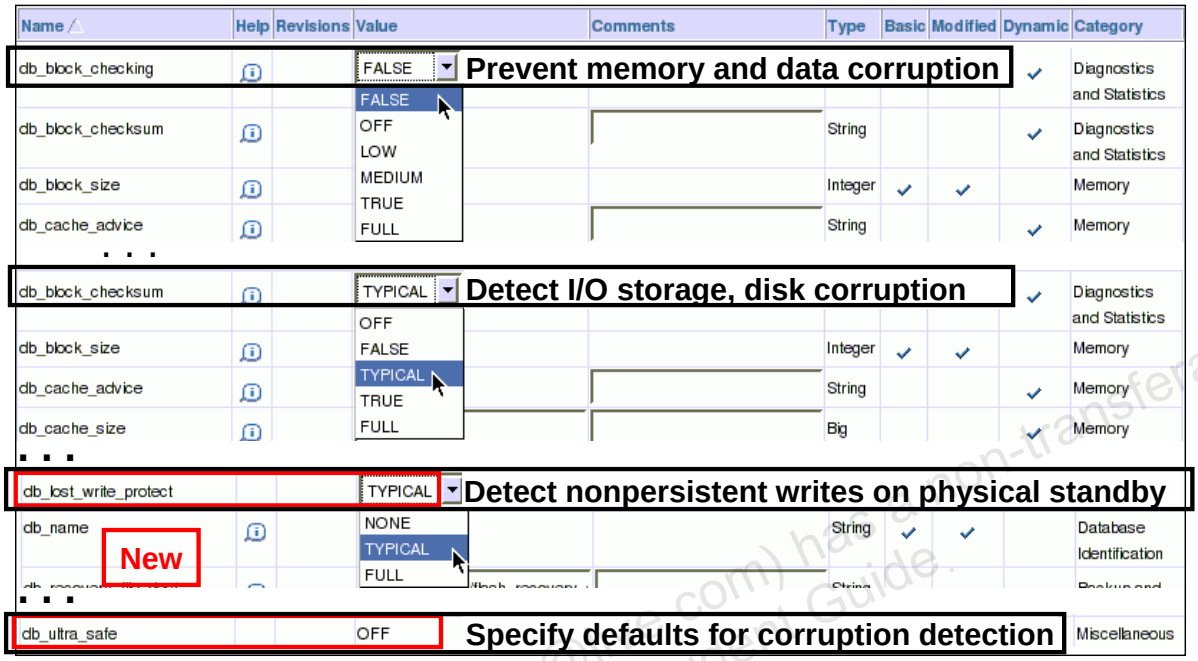
Always try to find out whether the error is permanent. Run the ANALYZE command multiple times or, if possible, perform a shutdown and a startup and try again to perform the operation that failed earlier. Find out whether there are more corruptions. If you encounter one, there may be other corrupted blocks, as well.

Hardware failures should be addressed immediately. When you encounter hardware problems, the vendor should be contacted and the machine should be checked and fixed before continuing. A full hardware diagnostics session should be run.

Many types of hardware failures are possible:

- Faulty I/O hardware or firmware
- Operating system I/O or caching problem
- Memory or paging problems
- Disk repair utilities

Setting Parameters to Detect Corruption



EM > Server > Initialization Parameters

Setting Parameters to Detect Corruption

You can use the DB_ULTRA_SAFE parameter for easy manageability. It affects the default values of the following parameters:

- DB_BLOCK_CHECKING, which initiates checking of database blocks. This check can often prevent memory and data corruption. (Default: FALSE, recommended: FULL)
- DB_BLOCK_CHECKSUM, which initiates the calculation and storage of a checksum in the cache header of every data block when writing it to disk. Checksums assist in detecting corruption caused by underlying disks, storage systems, or I/O systems. (Default: TYPICAL, recommended: TYPICAL)
- DB_LOST_WRITE_PROTECT, which initiates checking for "lost writes." Data block lost writes occur on a physical standby database, when the I/O subsystem signals the completion of a block write, which has not yet been completely written in persistent storage. Of course, the write operation has been completed on the primary database. (Default: TYPICAL, recommended: TYPICAL)

If you set any of these parameters explicitly, your values remain in effect. The DB_ULTRA_SAFE parameter (which is new in Oracle Database 11*g*) changes only the **default** values for these parameters.

Setting Parameters to Detect Corruption

DB_ULTRA_SAFE	OFF	DATA_ONLY	DATA_AND_INDEX
DB_BLOCK_CHECKING	OFF or FALSE	MEDIUM	FULL or TRUE
DB_BLOCK_CHECKSUM	TYPICAL	FULL	FULL
DB_LOST_WRITE_PROTECT	TYPICAL	TYPICAL	TYPICAL

Setting Parameters to Detect Corruption (continued)

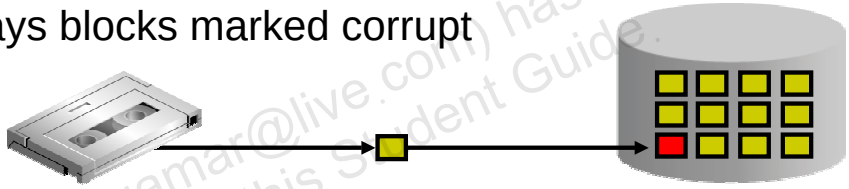
Depending on your system's tolerance for block corruption, you can intensify the checking for block corruption. Enabling the DB_ULTRA_SAFE parameter (default: OFF) results in increased system overhead because of these more intensive checks. The amount of overhead is related to the number of blocks changed per second, so it cannot be easily quantified. For a "high-update" application, you can expect a significant increase in CPU, likely in the ten-to-twenty percent range, but possibly higher. This overhead can be alleviated by allocating additional CPUs.

- When the DB_ULTRA_SAFE parameter is set to DATA_ONLY, the DB_BLOCK_CHECKING parameter is set to MEDIUM. This checks that data in a block are logically self-consistent. Basic block header checks are performed after block contents change in memory (for example, after UPDATE or INSERT commands, on-disk reads, or interinstance block transfers in Oracle RAC). This level of checks includes semantic block checking for all non-index-organized table blocks.
- When the DB_ULTRA_SAFE parameter is set to DATA_AND_INDEX, the DB_BLOCK_CHECKING parameter is set to FULL. In addition to the preceding checks, semantic checks are executed for index blocks (that is, blocks of subordinate objects that can actually be dropped and reconstructed when faced with corruption).
- When the DB_ULTRA_SAFE parameter is set to DATA_ONLY or DATA_AND_INDEX, the DB_BLOCK_CHECKSUM parameter is set to FULL and the DB_LOST_WRITE_PROTECT parameter is set to TYPICAL.

Block Media Recovery

Block media recovery:

- Lowers the mean time to recover (MTTR)
- Increases availability during media recovery
  - The data file remains online during recovery
  - Only blocks being recovered are inaccessible
- Is invoked using the RMAN **`RECOVER...BLOCK`** command
  - Restores blocks using flashback logs and full or level 0 backups
  - Media recovery is performed using redo logs
- The **`V$DATABASE_BLOCK_CORRUPTION`** view displays blocks marked corrupt



ORACLE

Block Media Recovery

In most cases, the database marks a block as media corrupt and then writes it to disk when the corruption is first encountered. No subsequent read of the block will be successful until the block is recovered. You can only perform block recovery on blocks that are marked corrupt or fail a corruption check. Block media recovery is performed using the RMAN `RECOVER...BLOCK` command. By default, RMAN searches the flashback logs for good copies of the blocks, and then searches for the blocks in full or level 0 incremental backups. When RMAN finds good copies, it restores them and performs media recovery on the blocks. Block media recovery can only use redo logs for media recovery, not incremental backups.

The `V$DATABASE_BLOCK_CORRUPTION` view displays blocks marked corrupt by database components such as RMAN commands, ANALYZE, `dbv`, SQL queries, and so on. The following types of corruption result in rows added to this view:

- **Physical/Media corruption:** The database does not recognize the block: the checksum is invalid, the block contains all zeros, or the block header is fractured. Physical corruption checking is enabled by default.
- **Logical corruption:** The block has a valid checksum, the header and footer match, but the contents are inconsistent. Block media recovery cannot repair logical block corruption. Logical corruption checking is disabled by default. You can turn it on by specifying the `CHECK LOGICAL` option of the `BACKUP`, `RESTORE`, `RECOVER`, and `VALIDATE` commands.

Prerequisites for Block Media Recovery

- The target database must be in ARCHIVELOG mode.
- The backups of the data files containing the corrupt blocks must be full or level 0 backups.
  - Proxy copies must be restored to a non-default location before they can be used.
- RMAN can use only archived redo logs for the recovery.
- The corrupted data block can be restored from Flashback Logs if available.

Prerequisites for Block Media Recovery

The following prerequisites apply to the `RECOVER ... BLOCK` command:

- The target database must run in ARCHIVELOG mode and be open or mounted with a current control file.
- The backups of the data files containing the corrupt blocks must be full or level 0 backups and not proxy copies. If only proxy copy backups exist, then you can restore them to a non-default location on disk, in which case RMAN considers them data file copies and searches them for blocks during block media recovery.
- RMAN can use only archived redo logs for the recovery. RMAN cannot use level 1 incremental backups. Block media recovery cannot survive a missing or inaccessible archived redo log, although it can sometimes survive missing redo records.
- Flashback Database must be enabled on the target database for RMAN to search the flashback logs for good copies of corrupt blocks. If flashback logging is enabled and contains older, uncorrupted versions of the corrupt blocks, then RMAN can use these blocks, possibly speeding up the recovery.

The RECOVER . . . BLOCK Command

The RMAN **RECOVER . . . BLOCK** command:

- Identifies the backups containing the blocks to recover
- Reads the backups and accumulates requested blocks into in-memory buffers
- Manages the block media recovery session by reading the archive logs from backup if necessary

```
RECOVER DATAFILE 6 BLOCK 3; Recover a single block

RECOVER                     Recover multiple blocks
DATAFILE 2 BLOCK 43         in multiple data files
DATAFILE 2 BLOCK 79
DATAFILE 6 BLOCK 183;

RECOVER CORRUPTION LIST;    Recover all blocks logged in
                            V$DATABASE_BLOCK_CORRUPTION
```

ORACLE

Recovering Individual Blocks

Before block recovery can take place, you must identify the corrupt blocks. Typically, block corruption is reported in the following locations:

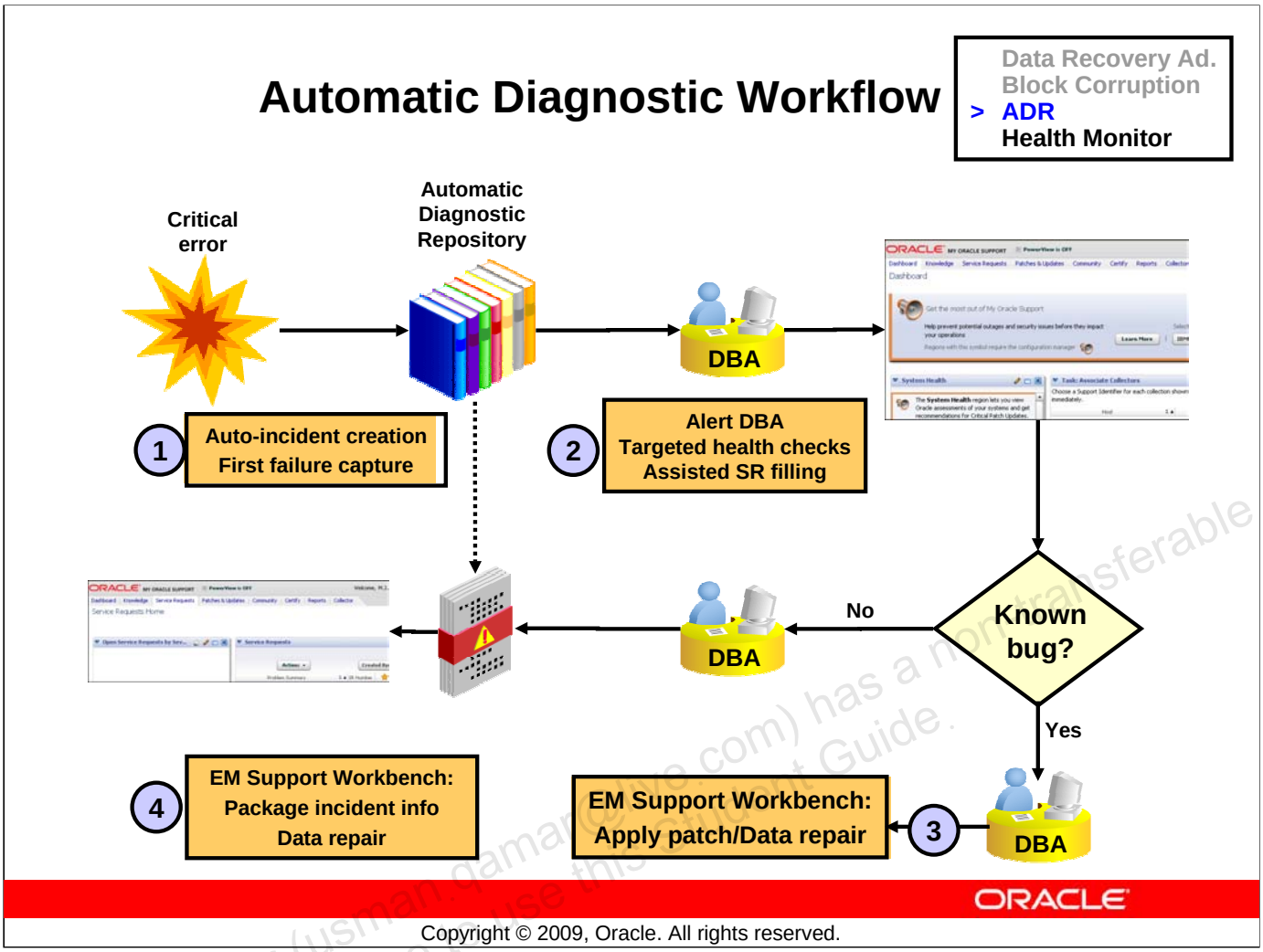
- Results of the `LIST FAILURE`, `VALIDATE`, or `BACKUP ... VALIDATE` command
- The `V$DATABASE_BLOCK_CORRUPTION` view
- Error messages in standard output
- The alert log and user trace files (identified in the `V$DIAG_INFO` view)
- Results of the SQL `ANALYZE TABLE` and `ANALYZE INDEX` commands
- Results of the DBVERIFY utility

For example, you may discover the following messages in a user trace file:

```
ORA-01578: ORACLE data block corrupted (file # 7, block # 3)
ORA-01110: data file 7: '/oracle/oradata/orcl/tools01.dbf'
ORA-01578: ORACLE data block corrupted (file # 2, block # 235)
ORA-01110: data file 2: '/oracle/oradata/orcl/undotbs01.dbf'
```

One the blocks have been identified, run the `RECOVER ... BLOCK` command at the RMAN prompt, specifying the file and block numbers for the corrupted blocks.

```
RECOVER
DATAFILE 7 BLOCK 3
DATAFILE 2 BLOCK 235;
```

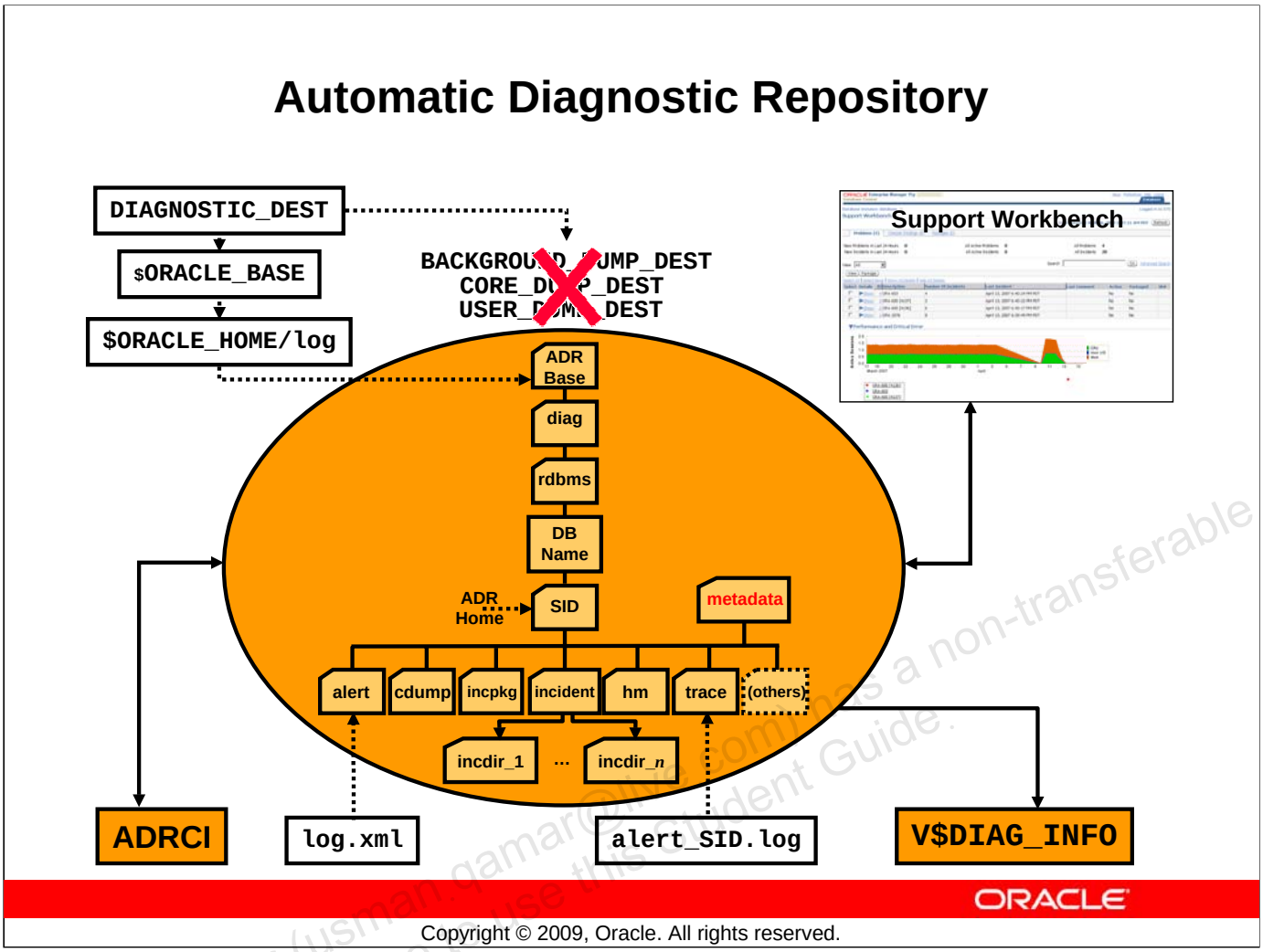


Automatic Diagnostic Workflow

An always-on, in-memory tracing facility enables database components to capture diagnostic data upon first failure for critical errors. A special repository, called Automatic Diagnostic Repository, is automatically maintained to hold diagnostic information about critical error events. This information can be used to create incident packages to be sent to Oracle Support Services for investigation.

Here is a typical workflow for a diagnostic session:

1. Incident causes an alert to be raised in Enterprise Manager (EM).
2. The DBA can view the alert via the EM Alert page.
3. The DBA can drill down to incident and problem details.
4. The DBA or Oracle Support Services can decide or ask for that information to be packaged and sent to Oracle Support Services via My Oracle Support. The DBA can add files to the data to be packaged automatically.



Automatic Diagnostic Repository (ADR)

The ADR is a file-based repository for database diagnostic data such as traces, incident dumps and packages, the alert log, Health Monitor reports, core dumps, and more. It has a unified directory structure across multiple instances and multiple products stored outside of any database. It is, therefore, available for problem diagnosis when the database is down.

Beginning with Oracle Database 11*g* R1, the database, Automatic Storage Management (ASM), Cluster Ready Services (CRS), and other Oracle products or components store all diagnostic data in the ADR. Each instance of each product stores diagnostic data underneath its own ADR home directory. For example, in a Real Application Clusters environment with shared storage and ASM, each database instance and each ASM instance have a home directory within the ADR. ADR's unified directory structure, consistent diagnostic data formats across products and instances, and a unified set of tools enable customers and Oracle Support to correlate and analyze diagnostic data across multiple instances.

The ADR root directory is known as the ADR base. Its location is set by the `DIAGNOSTIC_DEST` initialization parameter. If this parameter is omitted or left null, the database sets `DIAGNOSTIC_DEST` upon startup as follows: If environment variable `ORACLE_BASE` is set, `DIAGNOSTIC_DEST` is set to `$ORACLE_BASE`. If environment variable `ORACLE_BASE` is not set, `DIAGNOSTIC_DEST` is set to `$ORACLE_HOME/log`.

The ADR Command-Line Tool (ADRCI)

- ADRCI provides interaction with ADR from an operating system prompt.
- Using ADRCI, you can view diagnostic data within the Automatic Diagnostic Repository.

```
$ adrci
ADRCI: Release 11.1.0.5.0 - On Sat Jul 7 08:01:40 2007
Copyright (c) 1982, 2007, Oracle.  All rights reserved.

ADR base = "/u01/app/oracle"

ADRCI> show incident
ADR Home = /u01/app/oracle/product/11.1.0/db_1/log/diag/rdbms/orcl/orcl:
*************************************************************************
INCIDENT_ID PROBLEM_KEY                          CREATE_TIME
----------- ------------------------------------ ----------------------
1681        ORA-600_dbgris01:1,_addr=0xa9876541  17-JAN-07 09.17.44.843125…
1682        ORA-600_dbgris01:12,_addr=0xa9876542 18-JAN-07 09.18.59.434775…
2 incident info records fetched
```

ORACLE

The ADR Command-Line Tool (ADRCI)

ADRCI is a command-line tool that is part of the database fault diagnosability infrastructure. ADRCI enables you to:

- View diagnostic data within the Automatic Diagnostic Repository (ADR)
- Package incident and problem information into a zip file for transmission to Oracle Support

ADRCI can be used in interactive mode or within scripts. In addition, ADRCI can execute scripts of ADRCI commands in the same way that SQL*Plus executes scripts of SQL and PL/SQL commands. There is no need to log in to ADRCI, because the data in the ADR is not intended to be secure. ADR data is secured only by operating system permissions on the ADR directories.

The easiest way to package and otherwise manage diagnostic data is with the Support Workbench of Enterprise Manager (which assists with the resolution of database errors, as well as ASM errors).

ADRCI provides a command-line alternative to most of the functionality of Support Workbench, and adds capabilities such as listing and querying trace files. The example in the slide shows you an ADRCI session where you are listing all open incidents stored in ADR.

Note: For more information about ADRCI and the Support Workbench, refer to the *Oracle Database Utilities* guide.

The V$DIAG_INFO View

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

NAME                 VALUE
-------------------  ------------------------------------------------
Diag Enabled         TRUE
ADR Base             /u01/app/oracle
ADR Home             /u01/app/oracle/diag/rdbms/orcl/orcl
Diag Trace           /u01/app/oracle/diag/rdbms/orcl/orcl/trace
Diag Alert           /u01/app/oracle/diag/rdbms/orcl/orcl/alert
Diag Incident        /u01/app/oracle/diag/rdbms/orcl/orcl/incident
Diag Cdump           /u01/app/oracle/diag/rdbms/orcl/orcl/cdump
Health Monitor       /u01/app/oracle/diag/rdbms/orcl/orcl/hm
Default Trace File   /u01/app/oracle/diag/.../trace/orcl_ora_11424.trc
Active Problem Count 3
Active Incident Count 8
```


The V$DIAG_INFO View

The V$DIAG_INFO view lists all important ADR locations:

- ADR Base: Path of the ADR base
- ADR Home: Path of the ADR home for the current database instance
- Diag Trace: Location of the text alert log and background/foreground process trace files
- Diag Alert: Location of an XML version of the alert log
- Diag Incident: Incident logs are written here.
- Diag Cdump: Diagnostic core files are written to this directory.
- Health Monitor: Location of logs from Health Monitor runs
- Default Trace File: Path to the trace file for your session. SQL trace files are written here.

Location for Diagnostic Traces

Diag Data	Previous Location	ADR Location
Foreground process traces	USER_DUMP_DEST	*ADR_HOME*/trace
Background process traces	BACKGROUND_DUMP_DEST	*ADR_HOME*/trace
Alert log data	BACKGROUND_DUMP_DEST	*ADR_HOME*/alert *ADR_HOME*/trace
Core dumps	CORE_DUMP_DEST	*ADR_HOME*/cdump
Incident dumps	USER|BACKGROUND_DUMP_DEST	*ADR_HOME*/incident/incdir_*n*

Location for Diagnostic Traces

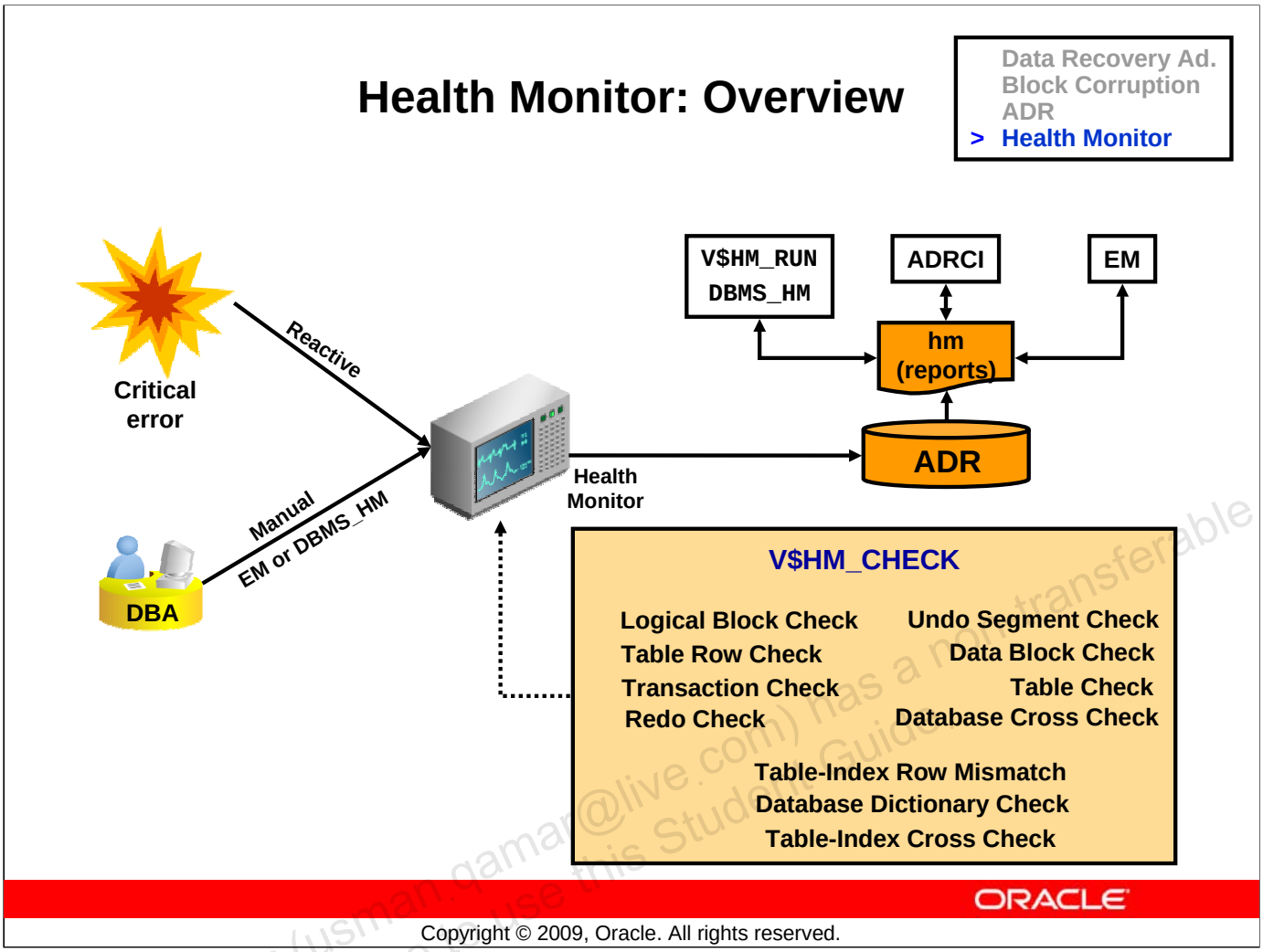
The table in the slide compares the different classes of trace data and dumps that reside both in Oracle Database 10*g* and Oracle Database 11*g*.

With Oracle Database 11*g*, there is no distinction between foreground and background traces files. Both types of files go into the *ADR_HOME*/`trace` directory.

All nonincident traces are stored inside the `trace` subdirectory. This is the main difference compared with previous releases where critical error information is dumped into the corresponding process trace files instead of incident dumps. Incident dumps are placed in files separated from the normal process trace files starting with Oracle Database 11*g*.

The main difference between a trace and a dump is that a trace is more of a continuous output such as when SQL tracing is turned on, and a dump is a one-time output in response to an event such as an incident. Also, a core is a binary memory dump that is port specific.

Note: In the slide, *ADR_HOME* represents the path `/u01/app/oracle/diag/rdbms/orcl/orcl`, assuming an instance name of `orcl`. However, there is no official environment variable called *ADR_HOME*.



Health Monitor: Overview

The Oracle database includes a framework called Health Monitor for running diagnostic checks on various components of the database. Health Monitor checks examine various components of the database, including files, memory, transaction integrity, metadata, and process usage. These checks generate reports of their findings as well as recommendations for resolving problems. The fault diagnosability infrastructure can run Health Monitor checks automatically in response to critical errors or the DBA, can manually run Health Monitor health checks by using either the `DBMS_HM` `PL/SQL` package or the Enterprise Manager interface.

For a complete description of all possible checks that Health Monitor can run, look at `V$HM_CHECK`. These health checks fall into one of two categories:

- **DB-online:** These checks can be run while the database is open (that is, in `OPEN` mode).
- **DB-offline:** In addition to being "runnable" while the database is open, these checks can also be run when the instance is available and the database itself is closed (`NOMOUNT` mode).

After a checker has run, it generates a report containing information about the checker's findings, including the priorities (low, high, or critical), descriptions of the findings and their consequences, and basic statistics about the execution. Health Monitor generates reports in XML and stores the reports in ADR. You can view these reports using either `V$HM_RUN`, `DBMS_HM`, `ADRCI`, or Enterprise Manager.

Running Health Checks Manually: PL/SQL Example

```
SQL> exec dbms_hm.run_check('Database Dictionary Check',
                                              'mycheck',0,'TABLE_NAME=tab$');
```

```
SQL> set long 100000
SQL> select dbms_hm.get_run_report('mycheck') from dual;

DBMS_HM.GET_RUN_REPORT('mycheck')
--------------------------------------------------------------------------------
<?xml version="1.0" encoding="US-ASCII"?>
<HM-REPORT REPORT_ID="mycheck"><TITLE>HM Report: mycheck</TITLE>
  <RUN_INFO>
   <CHECK_NAME>Database Dictionary Check</CHECK_NAME>
    <RUN_ID>21</RUN_ID><RUN_NAME>mycheck</RUN_NAME>
    <RUN_MODE>MANUAL</RUN_MODE><RUN_STATUS>COMPLETED</RUN_STATUS> …
  </RUN_INFO>
  <RUN_PARAMETERS><RUN_PARAMETER>TABLE_NAME=tab$</RUN_PARAMETER> … </RUN_PARAMETERS>
  <RUN-FINDINGS><FINDING>
    <FINDING_NAME>Dictionary Inconsistency</FINDING_NAME><FINDING_ID>22</FINDING_ID>
    <FINDING_TYPE>FAILURE</FINDING_TYPE><FINDING_STATUS>OPEN</FINDING_STATUS>
    <FINDING_PRIORITY>CRITICAL</FINDING_PRIORITY> …
    <FINDING_CREATION_TIME>…</FINDING_CREATION_TIME>
    <FINDING_MESSAGE>…invalid column number 7 on Object tab$ Failed</FINDING_MESSAGE>
    <FINDING_MESSAGE>Damaged … Object SH.JFVTEST is referenced </FINDING_MESSAGE> …
```

ORACLE

Running Health Checks Manually: PL/SQL Example

You can use the DBMS_HM.RUN_CHECK procedure for running a health check. To call RUN_CHECK, supply the name of the check found in V$HM_CHECK, the name for the run (this is just a label used to retrieve reports later), and the corresponding set of input parameters for controlling its execution. You can view these parameters using the V$HM_CHECK_PARAM.

In the slide example, you want to run a Database Dictionary Check for TAB$ table, considered an important core dictionary object. You call this run MYCHECK, and you do not want to set any timeout for this check.

When executed, you execute the DBMS_HM.GET_RUN_REPORT function to get the report extracted from VHM_RUN, VHM_FINDING, and V$HM_RECOMMENDATION. The output clearly shows you that a critical error was found in TAB$. This table contains an entry for a table with an invalid number of columns. Furthermore, the report gives you the name of the damaged table in TAB$.

When you call the GET_RUN_REPORT function, it generates the XML report file in the HM directory of your ADR. In the example, the file is called HMREPORT_mycheck.hm.

Note: Refer to the *Oracle Database PL/SQL Packages and Types Reference* for more information about DBMS_HM.

Viewing HM Reports Using the ADRCI Utility

```
adrci>>show hm_run
…
------------------------------------------------------------
RUN_ID                        11081
RUN_NAME                      HM_RUN_11081
CHECK_NAME                    Database Cross Check
NAME_ID                       2
MODE                          2
START_TIME                    2007-04-13 03:20:31.161396 -07:00
RESUME_TIME
END_TIME                      2007-04-13 03:20:37.903984 -07:00
MODIFIED_TIME                 2007-04-17 01:16:37.106344 -07:00
TIMEOUT                       0
FLAGS                         0
STATUS                        5
SRC_INCIDENT_ID               0
NUM_INCIDENTS                 0
ERR_NUMBER                    0
REPORT_FILE
…
adrci>>create report hm_run HM_RUN_11081
Adrci>>show report hm_run HM_RUN_11081
…
```

ORACLE

Viewing HM Reports Using the ADRCI Utility

You can create and view Health Monitor checker reports using the ADRCI utility. To do that, ensure that operating system environment variables such as `ORACLE_HOME` are set properly, and then enter the following command at the operating system command prompt: `adrci`.

The utility starts and displays its prompt as shown in the slide. Optionally, you can change the current ADR home. Use the `SHOW HOMES` command to list all ADR homes, and the `SET HOMEPATH` command to change the current ADR home.

You can then enter the `SHOW HM_RUN` command to list all the checker runs registered in the ADR repository and visible from `V$HM_RUN`. Locate the checker run for which you want to create a report and note the checker run name using the corresponding `RUN_NAME` field. The `REPORT_FILE` field contains a file name if a report already exists for this checker run. Otherwise, you can generate the report using the `CREATE REPORT HM_RUN` command as shown in the slide. To view the report, use the `SHOW REPORT HM_RUN` command.

Quiz

The Data Recovery Advisor handles both cases: when you cannot start up the database (because some required database files are missing, inconsistent, or corrupted) and when file corruptions are discovered during run time.

1. True
2. False

Answer: 1

Quiz

After executing the `ADVISE FAILURE` command, the repair is automatically executed. So, it is no longer under your control.

1. True
2. False

Answer: 2

Quiz

The ADR resides in the database. Therefore, an instance must be mounted for incident analysis.

1. True
2. False

Answer: 2

Quiz

Which of the following checks can the Health Monitor perform?

1. Intuitive commit check
2. Memory check
3. Metadata check
4. Redo check
5. Transaction check
6. User alertness check
7. Undo segment check

Answer: 2, 3, 4, 5, 7

Summary

In this lesson, you should have learned how to:

- Detect and repair database corruption:
  - Use the new RMAN data repair commands to:
    - List failures
    - Receive a repair advice
    - Repair failures
  - Perform proactive failure checks
- Handle block corruption:
  - Verifying block integrity in real time
  - Performing block media recovery
- Set up Automatic Diagnostic Repository
- Run health checks

Practice 9 Overview: Diagnosing the Database

This practice covers the following topics:

- Discovering corruptions
- Repairing corruptions