24
Repairing Corrupted Data

You can detect and correct data block corruption.

> **Note:**
>
> If you are not familiar with the `DBMS_REPAIR` package, then it is recommended that you work with an Oracle Support Services analyst when performing any of the repair procedures included in this package.

- Options for Repairing Data Block Corruption
  Oracle Database provides different methods for detecting and correcting data block corruption.
- About the DBMS_REPAIR Package
  The `DBMS_REPAIR` package contains data corruption repair procedures that enable you to detect and repair corrupt blocks in tables and indexes.
- Using the DBMS_REPAIR Package
  You can use the `DBMS_REPAIR` package to address data block corruption:
- DBMS_REPAIR Examples
  Examples illustrate how to use the `DBMS_REPAIR` package.

24.1 Options for Repairing Data Block Corruption

Oracle Database provides different methods for detecting and correcting data block corruption.

One method of correction is to drop and re-create an object after the corruption is detected. However, this is not always possible or desirable. If data block corruption is limited to a subset of rows, then another option is to rebuild the table by selecting all data except for the corrupt rows.

Another way to manage data block corruption is to use the `DBMS_REPAIR` package. You can use `DBMS_REPAIR` to detect and repair corrupt blocks in tables and indexes. You can continue to use objects while you attempt to rebuild or repair them.

You can also use the Recovery Manager (RMAN) command `RECOVER BLOCK` to recover a corrupt data block or set of data blocks.

> **Note:**
>
> Any corruption that involves the loss of data requires analysis and understanding of how that data fits into the overall database system. Depending on the nature of the repair, you might lose data, and logical inconsistencies can be introduced. You must determine whether the repair approach provided by this package is the appropriate tool for each specific corruption problem.

> **See Also:**
>
> *Oracle Database Backup and Recovery Reference* for more information about the `RECOVER BLOCK` RMAN command

24.2 About the DBMS_REPAIR Package

The `DBMS_REPAIR` package contains data corruption repair procedures that enable you to detect and repair corrupt blocks in tables and indexes.

- DBMS_REPAIR Procedures
  Procedures in the `DBMS_REPAIR` package enable you to detect and repair corrupt blocks.
- Limitations and Restrictions for DBMS_REPAIR Procedures
  Some limitations and restrictions apply to `DBMS_REPAIR` procedures.

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for more information on the syntax, restrictions, and exceptions for the `DBMS_REPAIR` procedures

24.2.1 DBMS_REPAIR Procedures

Procedures in the `DBMS_REPAIR` package enable you to detect and repair corrupt blocks.

The following table lists the procedures included in the `DBMS_REPAIR` package.

Procedure Name	Description
`ADMIN_TABLES`	Provides administrative functions (create, drop, purge) for repair or orphan key tables. **Note:** These tables are always created in the `SYS` schema.
`CHECK_OBJECT`	Detects and reports corruptions in a table or index
`DUMP_ORPHAN_KEYS`	Reports on index entries that point to rows in corrupt data blocks
`FIX_CORRUPT_BLOCKS`	Marks blocks as software corrupt that have been previously identified as corrupt by the `CHECK_OBJECT` procedure
`REBUILD_FREELISTS`	Rebuilds the free lists of the object
`SEGMENT_FIX_STATUS`	Provides the capability to fix the corrupted state of a bitmap entry when segment space management is `AUTO`
`SKIP_CORRUPT_BLOCKS`	When used, ignores blocks marked corrupt during table and index scans. If not used, you get error `ORA-01578` when encountering blocks marked corrupt.

These procedures are further described, with examples of their use, in "DBMS_REPAIR Examples".

24.2.2 Limitations and Restrictions for DBMS_REPAIR Procedures

Some limitations and restrictions apply to `DBMS_REPAIR` procedures.

`DBMS_REPAIR` procedures have the following limitations:

- Tables with LOB data types, nested tables, and varrays are supported, but the out-of-line columns are ignored.
- Clusters are supported in the `SKIP_CORRUPT_BLOCKS` and `REBUILD_FREELISTS` procedures, but not in the `CHECK_OBJECT` procedure.
- Index-organized tables and LOB indexes are not supported.
- Global temporary tables are not supported.
- The `DUMP_ORPHAN_KEYS` procedure does not operate on bitmap indexes or function-based indexes.
- The `DUMP_ORPHAN_KEYS` procedure processes keys that are no more than 3,950 bytes long.

24.3 Using the DBMS_REPAIR Package

You can use the `DBMS_REPAIR` package to address data block corruption:

- Task 1: Detect and Report Corruptions
  The first task is the detection and reporting of corruptions. Reporting not only indicates what is wrong with a block, but also identifies the associated repair directive.
- Task 2: Evaluate the Costs and Benefits of Using DBMS_REPAIR
  Before using `DBMS_REPAIR` you must weigh the benefits of its use in relation to the liabilities. You should also examine other options available for addressing corrupt objects.
- Task 3: Make Objects Usable
  `DBMS_REPAIR` makes the object usable by ignoring corruptions during table and index scans.
- Task 4: Repair Corruptions and Rebuild Lost Data
  After making an object usable, perform the following repair activities.

24.3.1 Task 1: Detect and Report Corruptions

The first task is the detection and reporting of corruptions. Reporting not only indicates what is wrong with a block, but also identifies the associated repair directive.

- About Detecting and Reporting Corruptions
  There are several ways to detect corruptions.
- DBMS_REPAIR: Using the CHECK_OBJECT and ADMIN_TABLES Procedures
  The `CHECK_OBJECT` procedure checks and reports block corruptions for a specified object. The `ADMIN_TABLES` procedure creates a repair table that facilitates correcting corruptions.
- DB_VERIFY: Performing an Offline Database Check
  Use `DB_VERIFY` as an offline diagnostic utility when you encounter data corruption.
- ANALYZE: Reporting Corruption
  The `ANALYZE TABLE...VALIDATE STRUCTURE` statement validates the structure of the analyzed object. If the database encounters corruption in the structure of the object, then an error message is returned. In this case, drop and re-create the object.

- DB_BLOCK_CHECKING Initialization Parameter
  You can enable database block checking by setting the `DB_BLOCK_CHECKING` initialization parameter to `TRUE`.

24.3.1.1 About Detecting and Reporting Corruptions

There are several ways to detect corruptions.

Table 24-1 describes the different detection methodologies.

Table 24-1 Comparison of Corruption Detection Methods

Detection Method	Description
`DBMS_REPAIR` PL/SQL package	Performs block checking for a specified table, partition, or index. It populates a repair table with results.
`DB_VERIFY` utility	Performs block checking on an offline database
`ANALYZE TABLE` SQL statement	Used with the `VALIDATE STRUCTURE` option, the `ANALYZE TABLE` statement verifies the integrity of the structure of an index, table, or cluster; checks or verifies that tables and indexes are synchronized.
`DB_BLOCK_CHECKING` initialization parameter	When `DB_BLOCK_CHECKING=TRUE`, corrupt blocks are identified before they are marked corrupt. Checks are performed when changes are made to a block.

24.3.1.2 DBMS_REPAIR: Using the CHECK_OBJECT and ADMIN_TABLES Procedures

The `CHECK_OBJECT` procedure checks and reports block corruptions for a specified object. The `ADMIN_TABLES` procedure creates a repair table that facilitates correcting corruptions.

The `CHECK_OBJECT` procedure is similar to the `ANALYZE...VALIDATE STRUCTURE` statement for indexes and tables, block checking is performed for index and data blocks.

Not only does `CHECK_OBJECT` report corruptions, but it also identifies any fixes that would occur if `FIX_CORRUPT_BLOCKS` is subsequently run on the object. This information is made available by populating a repair table, which must first be created by the `ADMIN_TABLES` procedure.

After you run the `CHECK_OBJECT` procedure, a simple query on the repair table shows the corruptions and repair directives for the object. With this information, you can assess how best to address the reported problems.

24.3.1.3 DB_VERIFY: Performing an Offline Database Check

Use `DB_VERIFY` as an offline diagnostic utility when you encounter data corruption.

> **See Also:**
>
> *Oracle Database Utilities* for more information about `DB_VERIFY`

24.3.1.4 ANALYZE: Reporting Corruption

The `ANALYZE TABLE...VALIDATE STRUCTURE` statement validates the structure of the analyzed object. If the database encounters corruption in the structure of the object, then an error message is returned. In this case, drop and re-create the object.

You can use the `CASCADE` clause of the `ANALYZE TABLE` statement to check the structure of the table and all of its indexes in one operation. Because this operation can consume significant resources, there is a FAST option that performs a lightweight check. See "Validating Tables, Indexes, Clusters, and Materialized Views" for details.

See Also:

- *Oracle Database SQL Language Reference* for more information about the `ANALYZE` statement

24.3.1.5 DB_BLOCK_CHECKING Initialization Parameter

You can enable database block checking by setting the `DB_BLOCK_CHECKING` initialization parameter to `TRUE`.

This checks data and index blocks for internal consistency whenever they are modified. `DB_BLOCK_CHECKING` is a dynamic parameter, modifiable by the `ALTER SYSTEM SET` statement. Block checking is always enabled for the system tablespace.

Caution:

Before enabling block checking with this parameter, Oracle recommends that you detect and repair any logical corruptions in the database. Otherwise, a block that contain logical corruption will be marked as "soft corrupt" after block checking is enabled and the block is modified by a DML statement. This will result in `ORA-1578` errors and the block will be unreadable.

See Also:

Oracle Database Reference for more information about the `DB_BLOCK_CHECKING` initialization parameter.

Oracle Database Backup and Recovery User's Guide for more information about detecting and repairing logical corruptions.

24.3.2 Task 2: Evaluate the Costs and Benefits of Using DBMS_REPAIR

Before using DBMS_REPAIR you must weigh the benefits of its use in relation to the liabilities. You should also examine other options available for addressing corrupt objects.

Begin by answering the following questions:

- What is the extent of the corruption?

  To determine if there are corruptions and repair actions, execute the CHECK_OBJECT procedure and query the repair table.

- What other options are available for addressing block corruptions? Consider the following:
  - If the data is available from another source, then drop, re-create, and repopulate the object.
  - Issue the CREATE TABLE...AS SELECT statement from the corrupt table to create a new one.
  - Ignore the corruption by excluding corrupt rows from SELECT statements.
  - Perform media recovery.

- What logical corruptions or side effects are introduced when you use DBMS_REPAIR to make an object usable? Can these be addressed? What is the effort required to do so?

  You might not have access to rows in blocks marked corrupt. However, a block can be marked corrupt even if there are rows that you can validly access.

  It is also possible that referential integrity constraints are broken when blocks are marked corrupt. If this occurs, then disable and reenable the constraint; any inconsistencies are reported. After fixing all problems, you should be able to reenable the constraint.

  Logical corruption can occur when there are triggers defined on the table. For example, if rows are reinserted, should insert triggers be fired or not? You can address these issues only if you understand triggers and their use in your installation.

  If indexes and tables are not synchronized, then execute the DUMP_ORPHAN_KEYS procedure to obtain information from the keys that might be useful in rebuilding corrupted data. Then issue the ALTER INDEX...REBUILD ONLINE statement to synchronize the table with its indexes.

- If repair involves loss of data, can this data be retrieved?

  You can retrieve data from the index when a data block is marked corrupt. The DUMP_ORPHAN_KEYS procedure can help you retrieve this information.

24.3.3 Task 3: Make Objects Usable

DBMS_REPAIR makes the object usable by ignoring corruptions during table and index scans.

- Corruption Repair: Using the FIX_CORRUPT_BLOCKS and SKIP_CORRUPT_BLOCKS Procedures
  You can make a corrupt object usable by establishing an environment that skips corruptions that remain outside the scope of DBMS_REPAIR capabilities.
- Implications When Skipping Corrupt Blocks
  When skipping corrupt blocks, a query can return different results in some situations.

24.3.3.1 Corruption Repair: Using the FIX_CORRUPT_BLOCKS and SKIP_CORRUPT_BLOCKS Procedures

You can make a corrupt object usable by establishing an environment that skips corruptions that remain outside the scope of `DBMS_REPAIR` capabilities.

If corruptions involve a loss of data, such as a bad row in a data block, then all such blocks are marked corrupt by the `FIX_CORRUPT_BLOCKS` procedure. Then you can run the `SKIP_CORRUPT_BLOCKS` procedure, which skips blocks that are marked as corrupt. When the `SKIP_FLAG` parameter in the procedure is set, table and index scans skip all blocks marked corrupt. This applies to both media and software corrupt blocks.

24.3.3.2 Implications When Skipping Corrupt Blocks

When skipping corrupt blocks, a query can return different results in some situations.

If an index and table are not synchronized, then a `SET TRANSACTION READ ONLY` transaction can be inconsistent in situations where one query probes only the index, and a subsequent query probes both the index and the table. If the table block is marked corrupt, then the two queries return different results, thereby breaking the rules of a read-only transaction. One way to approach this is not to skip corruptions in a `SET TRANSACTION READ ONLY` transaction.

A similar issue occurs when selecting rows that are chained. A query of the same row may or may not access the corruption, producing different results.

24.3.4 Task 4: Repair Corruptions and Rebuild Lost Data

After making an object usable, perform the following repair activities.

- Recover Data Using the DUMP_ORPHAN_KEYS Procedures
  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 that stores the key and rowid of the corruption.
- Fix Segment Bitmaps Using the SEGMENT_FIX_STATUS Procedure
  Use the `SEGMENT_FIX_STATUS` procedure if free space in segments is being managed by using bitmaps (`SEGMENT SPACE MANAGEMENT AUTO`).

24.3.4.1 Recover Data Using the DUMP_ORPHAN_KEYS Procedures

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 that stores the key and rowid of the corruption.

After the index entry information has been retrieved, you can rebuild the index using the `ALTER INDEX...REBUILD ONLINE` statement.

24.3.4.2 Fix Segment Bitmaps Using the SEGMENT_FIX_STATUS Procedure

Use the `SEGMENT_FIX_STATUS` procedure if free space in segments is being managed by using bitmaps (`SEGMENT SPACE MANAGEMENT AUTO`).

This procedure recalculates the state of a bitmap entry based on the current contents of the corresponding block. Alternatively, you can specify that a bitmap entry be set to a specific value. Usually the state is recalculated correctly and there is no need to force a setting.

24.4 DBMS_REPAIR Examples

Examples illustrate how to use the `DBMS_REPAIR` package.

- Examples: Building a Repair Table or Orphan Key Table
  A repair table provides information about the corruptions. An orphan key table provides information about index entries that point to corrupt rows.
- Example: Detecting Corruption
  An example illustrates detecting corruption with the `CHECK_OBJECT` procedure.
- Example: Fixing Corrupt Blocks
  An example illustrates fixing corrupt blocks with the `FIX_CORRUPT_BLOCKS` procedure.
- Example: Finding Index Entries Pointing to Corrupt Data Blocks
  An example illustrates finding index entries pointing to corrupt data blocks using the `DUMP_ORPHAN_KEYS` procedure.
- Example: Skipping Corrupt Blocks
  An example illustrates skipping corrupt blocks using the `SKIP_CORRUPT_BLOCKS` procedure.

24.4.1 Examples: Building a Repair Table or Orphan Key Table

A repair table provides information about the corruptions. An orphan key table provides information about index entries that point to corrupt rows.

- About Repair Tables or Orphan Key Tables
  The `ADMIN_TABLES` procedure is used to create, purge, or drop a repair table or an orphan key table.
- Example: Creating a Repair Table
  An example illustrates creating a repair table using the `ADMIN_TABLES` procedure.
- Example: Creating an Orphan Key Table
  An example illustrates creating an orphan key table using the `ADMIN_TABLES` procedure.

24.4.1.1 About Repair Tables or Orphan Key Tables

The `ADMIN_TABLES` procedure is used to create, purge, or drop a repair table or an orphan key table.

A repair table provides information about the corruptions that were found by the `CHECK_OBJECT` procedure and how these will be addressed if the `FIX_CORRUPT_BLOCKS` procedure is run. Further, it is used to drive the execution of the `FIX_CORRUPT_BLOCKS` procedure.

An orphan key table is used when the `DUMP_ORPHAN_KEYS` procedure is executed and it discovers index entries that point to corrupt rows. The `DUMP_ORPHAN_KEYS` procedure populates the orphan key table by logging its activity and providing the index information in a usable manner.

24.4.1.2 Example: Creating a Repair Table

An example illustrates creating a repair table using the `ADMIN_TABLES` procedure.

The following example creates a repair table for the `users` tablespace.

```
BEGIN
  DBMS_REPAIR.ADMIN_TABLES (
     TABLE_NAME => 'REPAIR_TABLE',
     TABLE_TYPE => dbms_repair.repair_table,
     ACTION     => dbms_repair.create_action,
     TABLESPACE => 'USERS');
END;
/
```

For each repair or orphan key table, a view is also created that eliminates any rows that pertain to objects that no longer exist. The name of the view corresponds to the name of the repair or orphan key table and is prefixed by `DBA_` (for example, `DBA_REPAIR_TABLE` or `DBA_ORPHAN_KEY_TABLE`).

The following query describes the repair table that was created for the `users` tablespace.

```
DESC REPAIR_TABLE

 Name                          Null?    Type
 ----------------------------- -------- --------------
 OBJECT_ID                     NOT NULL NUMBER
 TABLESPACE_ID                 NOT NULL NUMBER
 RELATIVE_FILE_ID              NOT NULL NUMBER
 BLOCK_ID                      NOT NULL NUMBER
 CORRUPT_TYPE                  NOT NULL NUMBER
 SCHEMA_NAME                   NOT NULL VARCHAR2(128)
 OBJECT_NAME                   NOT NULL VARCHAR2(128)
 BASEOBJECT_NAME                        VARCHAR2(128)
 PARTITION_NAME                         VARCHAR2(128)
 CORRUPT_DESCRIPTION                    VARCHAR2(2000)
 REPAIR_DESCRIPTION                     VARCHAR2(200)
 MARKED_CORRUPT                NOT NULL VARCHAR2(10)
 CHECK_TIMESTAMP               NOT NULL DATE
 FIX_TIMESTAMP                          DATE
 REFORMAT_TIMESTAMP                     DATE
```

24.4.1.3 Example: Creating an Orphan Key Table

An example illustrates creating an orphan key table using the `ADMIN_TABLES` procedure.

This example illustrates the creation of an orphan key table for the `users` tablespace.

```
BEGIN
  DBMS_REPAIR.ADMIN_TABLES (
     TABLE_NAME => 'ORPHAN_KEY_TABLE',
     TABLE_TYPE => dbms_repair.orphan_table,
     ACTION     => dbms_repair.create_action,
     TABLESPACE => 'USERS');
END;
/
```

The orphan key table is described in the following query:

```
DESC ORPHAN_KEY_TABLE

 Name                          Null?    Type
 ----------------------------- -------- -----------------
 SCHEMA_NAME                   NOT NULL VARCHAR2(128)
 INDEX_NAME                    NOT NULL VARCHAR2(128)
 IPART_NAME                             VARCHAR2(128)
 INDEX_ID                      NOT NULL NUMBER
 TABLE_NAME                    NOT NULL VARCHAR2(128)
 PART_NAME                              VARCHAR2(128)
 TABLE_ID                      NOT NULL NUMBER
 KEYROWID                      NOT NULL ROWID
 KEY                           NOT NULL ROWID
 DUMP_TIMESTAMP                NOT NULL DATE
```

24.4.2 Example: Detecting Corruption

An example illustrates detecting corruption with the `CHECK_OBJECT` procedure.

The `CHECK_OBJECT` procedure checks the specified object, and populates the repair table with information about corruptions and repair directives. You can optionally specify a range, partition name, or subpartition name when you want to check a portion of an object.

Validation consists of checking all blocks in the object that have not previously been marked corrupt. For each block, the transaction and data layer portions are checked for self consistency. During `CHECK_OBJECT`, if a block is encountered that has a corrupt buffer cache header, then that block is skipped.

The following is an example of executing the `CHECK_OBJECT` procedure for the `scott.dept` table.

```
SET SERVEROUTPUT ON
DECLARE num_corrupt INT;
BEGIN
 num_corrupt := 0;
 DBMS_REPAIR.CHECK_OBJECT (
     SCHEMA_NAME => 'SCOTT',
     OBJECT_NAME => 'DEPT',
     REPAIR_TABLE_NAME => 'REPAIR_TABLE',
     CORRUPT_COUNT =>  num_corrupt);
 DBMS_OUTPUT.PUT_LINE('number corrupt: ' || TO_CHAR (num_corrupt));
END;
/
```

SQL*Plus outputs the following line, indicating one corruption:

```
number corrupt: 1
```

Querying the repair table produces information describing the corruption and suggesting a repair action.

```
SELECT OBJECT_NAME, BLOCK_ID, CORRUPT_TYPE, MARKED_CORRUPT,
       CORRUPT_DESCRIPTION, REPAIR_DESCRIPTION
     FROM REPAIR_TABLE;

OBJECT_NAME                     BLOCK_ID CORRUPT_TYPE MARKED_COR
------------------------------ ---------- ------------ ----------
CORRUPT_DESCRIPTION
--------------------------------------------------------------------------------
REPAIR_DESCRIPTION
--------------------------------------------------------------------------------
```

```
DEPT                                    3           1 FALSE
kdbchk: row locked by non-existent transaction
        table=0   slot=0
        lockid=32   ktbbhitc=1
mark block software corrupt
```

The corrupted block has not yet been marked corrupt, so this is the time to extract any meaningful data. After the block is marked corrupt, the entire block must be skipped.

24.4.3 Example: Fixing Corrupt Blocks

An example illustrates fixing corrupt blocks with the `FIX_CORRUPT_BLOCKS` procedure.

Use the `FIX_CORRUPT_BLOCKS` procedure to fix the corrupt blocks in specified objects based on information in the repair table that was generated by the `CHECK_OBJECT` procedure. Before changing a block, the block is checked to ensure that the block is still corrupt. Corrupt blocks are repaired by marking the block software corrupt. When a repair is performed, the associated row in the repair table is updated with a timestamp.

This example fixes the corrupt block in table `scott.dept` that was reported by the `CHECK_OBJECT` procedure.

```
SET SERVEROUTPUT ON
DECLARE num_fix INT;
BEGIN
 num_fix := 0;
 DBMS_REPAIR.FIX_CORRUPT_BLOCKS (
     SCHEMA_NAME => 'SCOTT',
     OBJECT_NAME=> 'DEPT',
     OBJECT_TYPE => dbms_repair.table_object,
     REPAIR_TABLE_NAME => 'REPAIR_TABLE',
     FIX_COUNT=> num_fix);
 DBMS_OUTPUT.PUT_LINE('num fix: ' || TO_CHAR(num_fix));
END;
/
```

SQL*Plus outputs the following line:

```
num fix: 1
```

The following query confirms that the repair was done.

```
SELECT OBJECT_NAME, BLOCK_ID, MARKED_CORRUPT
     FROM REPAIR_TABLE;

OBJECT_NAME                     BLOCK_ID MARKED_COR
------------------------------ ---------- ----------
DEPT                                    3 TRUE
```

24.4.4 Example: Finding Index Entries Pointing to Corrupt Data Blocks

An example illustrates finding index entries pointing to corrupt data blocks using the `DUMP_ORPHAN_KEYS` procedure.

The `DUMP_ORPHAN_KEYS` procedure reports on index entries that point to rows in corrupt data blocks. For each index entry, a row is inserted into the specified orphan key table. The orphan key table must have been previously created.

This information can be useful for rebuilding lost rows in the table and for diagnostic purposes.

> **Note:**
>
> This should be run for every index associated with a table identified in the repair table.

In this example, `pk_dept` is an index on the `scott.dept` table. It is scanned to determine if there are any index entries pointing to rows in the corrupt data block.

```
SET SERVEROUTPUT ON
DECLARE num_orphans INT;
BEGIN
 num_orphans := 0;
 DBMS_REPAIR.DUMP_ORPHAN_KEYS (
     SCHEMA_NAME => 'SCOTT',
     OBJECT_NAME => 'PK_DEPT',
     OBJECT_TYPE => dbms_repair.index_object,
     REPAIR_TABLE_NAME => 'REPAIR_TABLE',
     ORPHAN_TABLE_NAME=> 'ORPHAN_KEY_TABLE',
     KEY_COUNT => num_orphans);
 DBMS_OUTPUT.PUT_LINE('orphan key count: ' || TO_CHAR(num_orphans));
END;
/
```

The following output indicates that there are three orphan keys:

```
orphan key count: 3
```

Index entries in the orphan key table implies that the index should be rebuilt. This guarantees that a table probe and an index probe return the same result set.

24.4.5 Example: Skipping Corrupt Blocks

An example illustrates skipping corrupt blocks using the `SKIP_CORRUPT_BLOCKS` procedure.

The `SKIP_CORRUPT_BLOCKS` procedure enables or disables the skipping of corrupt blocks during index and table scans of the specified object. When the object is a table, skipping applies to the table and its indexes. When the object is a cluster, it applies to all of the tables in the cluster, and their respective indexes.

The following example enables the skipping of software corrupt blocks for the `scott.dept` table:

```
BEGIN
  DBMS_REPAIR.SKIP_CORRUPT_BLOCKS (
     SCHEMA_NAME => 'SCOTT',
     OBJECT_NAME => 'DEPT',
     OBJECT_TYPE => dbms_repair.table_object,
     FLAGS => dbms_repair.skip_flag);
END;
/
```

Querying `scott`'s tables using the `DBA_TABLES` view shows that `SKIP_CORRUPT` is enabled for table `scott.dept`.

```
SELECT OWNER, TABLE_NAME, SKIP_CORRUPT FROM DBA_TABLES
    WHERE OWNER = 'SCOTT';

OWNER                          TABLE_NAME                     SKIP_COR
```

```
------------------------------ ------------------------------ --------
SCOTT                          ACCOUNT                        DISABLED
SCOTT                          BONUS                          DISABLED
SCOTT                          DEPT                           ENABLED
SCOTT                          DOCINDEX                       DISABLED
SCOTT                          EMP                            DISABLED
SCOTT                          RECEIPT                        DISABLED
SCOTT                          SALGRADE                       DISABLED
SCOTT                          SCOTT_EMP                      DISABLED
SCOTT                          SYS_IOT_OVER_12255             DISABLED
SCOTT                          WORK_AREA                      DISABLED

10 rows selected.
```