53

DBMS_COMPRESSION

The DBMS_COMPRESSION package provides an interface to facilitate choosing the correct compression level for an application.

This chapter contains the following topics:

- Overview
- Security Model
- Constants
- Data Structures
- Summary of DBMS_COMPRESSION Subprograms

See Also:

- *Oracle Database Administrator's Guide*
- *Oracle Database Concepts*
- *Oracle Database SQL Language Reference*
- *Oracle Database Data Warehousing Guide*
- *Oracle Database VLDB and Partitioning Guide*
- *Oracle Database Reference*

DBMS_COMPRESSION Overview

The DBMS_COMPRESSION package gathers compression-related information within a database environment. This includes tools for estimating compressibility of a table for both partitioned and non-partitioned tables, and gathering row-level compression information on previously compressed tables. This gives the user with adequate information to make compression-related decision.

DBMS_COMPRESSION Security Model

The DBMS_COMPRESSSION package is defined with AUTHID CURRENT USER, so it executes with the privileges of the current user.

DBMS_COMPRESSION Constants

The DBMS_COMPRESSION package uses constants that can be used for specifying parameter values.

These constants are shown in the following table:

Table 53-1 DBMS_COMPRESSION Constants - Compression Types

Constant	Type	Value	Description
`COMP_NOCOMPRESS`	`NUMBER`	`1`	No compression
`COMP_ADVANCED`	`NUMBER`	`2`	Advanced row compression
`COMP_QUERY_HIGH`	`NUMBER`	`4`	High for query warehouse compression (Hybrid Columnar Compression)
`COMP_QUERY_LOW`	`NUMBER`	`8`	Low for query warehouse compression (Hybrid Columnar Compression)
`COMP_ARCHIVE_HIGH`	`NUMBER`	`16`	High archive compression (Hybrid Columnar Compression)
`COMP_ARCHIVE_LOW`	`NUMBER`	`32`	Low archive compression (Hybrid Columnar Compression)
`COMP_BLOCK`	`NUMBER`	`64`	Compressed block
`COMP_LOB_HIGH`	`NUMBER`	`128`	High compression level for LOB operations
`COMP_LOB_MEDIUM`	`NUMBER`	`256`	Medium compression level for LOB operations
`COMP_LOB_LOW`	`NUMBER`	`512`	Low compression level for LOB operations
`COMP_INDEX_ADVANCED_HIGH`	`NUMBER`	`1024`	High compression level for indexes
`COMP_INDEX_ADVANCED_LOW`	`NUMBER`	`2048`	Low compression level for indexes
`COMP_RATIO_LOB_MINROWS`	`NUMBER`	`1000`	Minimum required number of LOBs in the object for which LOB compression ratio is to be estimated
`COMP_BASIC`	`NUMBER`	`4096`	Basic table compression
`COMP_RATIO_LOB_MAXROWS`	`NUMBER`	`5000`	Maximum number of LOBs used to compute the LOB compression ratio
`COMP_INMEMORY_NOCOMPRESS`	`NUMBER`	`8192`	In-Memory with no compression
`COMP_INMEMORY_DML`	`NUMBER`	`16384`	In-Memory compression level for DML
`COMP_INMEMORY_QUERY_LOW`	`NUMBER`	`32768`	In-Memory compression level optimized for query performance
`COMP_INMEMORY_QUERY_HIGH`	`NUMBER`	`65536`	In-Memory compression level optimized on query performance as well as space saving
`COMP_INMEMORY_CAPACITY_LOW`	`NUMBER`	`131072`	In-Memory low compression level optimizing for capacity
`COMP_INMEMORY_CAPACITY_HIGH`	`NUMBER`	`262144`	In-Memory high compression level optimizing for capacity
`COMP_RATIO_MINROWS`	`NUMBER`	`1000000`	Minimum required number of rows in the object for which HCC ratio is to be estimated

Table 53-1 (Cont.) DBMS_COMPRESSION Constants - Compression Types

Constant	Type	Value	Description
COMP_RATIO_ALLROWS	NUMBER	-1	To indicate the use of all the rows in the object to estimate HCC ratio
OBJTYPE_TABLE	PLS_INTEGER	1	Identifies the object whose compression ratio is estimated as of type table
OBJTYPE_INDEX	PLS_INTEGER	2	Identifies the object whose compression ratio is estimated as of type index

Note:

Hybrid columnar compression is a feature of certain Oracle storage systems. See *Oracle Database Concepts* for more information.

DBMS_COMPRESSION Data Structures

The `DBMS_COMPRESSION` package defines a `RECORD` type and a `TABLE` type.

RECORD TYPES

COMPREC Record Type

TABLE TYPES

COMPRECLIST Table Type

COMPREC Record Type

The COMPREC record type is a record for calculating an individual index compression ratio on a table.

Syntax

```
TYPE COMPREC IS RECORD(
  ownname           varchar2(255),
  objname           varchar2(255),
  blkcnt_cmp        PLS_INTEGER,
  blkcnt_uncmp      PLS_INTEGER,
  row_cmp           PLS_INTEGER,
  row_uncmp         PLS_INTEGER,
  cmp_ratio         NUMBER,
  objtype           PLS_INTEGER);
```

Fields

Table 53-2 COMPREC Attributes

Field	Description
`ownname`	Schema of the object owner
`objname`	Name of the object
`blkcnt_cmp`	Number of blocks used by the compressed sample of the object
`blkcnt_uncmp`	Number of blocks used by the uncompressed sample of the object
`row_cmp`	Number of rows in a block in compressed sample of the object
`row_uncmp`	Number of rows in a block in uncompressed sample of the object
`cmp_ratio`	Compression ratio, `blkcnt_uncmp` divided by `blkcnt_cmp`
`objtype`	Type of the object

COMPRECLIST Table Type

COMPRECLIST is a table type of the COMPREC Record Type.

Syntax

```
TYPE compreclist IS TABLE OF comprec;
```

Related Topics

- COMPREC Record Type
  The COMPREC record type is a record for calculating an individual index compression ratio on a table.

Summary of DBMS_COMPRESSION Subprograms

The `DBMS_COMPRESSION` package uses the `GET_COMPRESSION_RATIO` Procedure and `GET_COMPRESSION_TYPE` Function subprograms.

Table 53-3 DBMS_COMPRESSION Package Subprograms

Subprogram	Description
GET_COMPRESSION_RATIO Procedure	Analyzes the compression ratio of a table, and gives information about compressibility of a table
GET_COMPRESSION_TYPE Function	Returns the compression type for a specified row

GET_COMPRESSION_RATIO Procedure

Use this procedure to estimate the storage space that you can save by enabling the compression feature for an existing SecureFile LOB. It analyzes the compression ratio of a table or an index and gives information about compressibility of the object. You can provide various parameters to selectively analyze different compression types.

In Oracle Database 23ai, this procedure has been enhanced to estimate the compression ratio faster for LOBs while using less space. Now you can also estimate the compression ratio for BasicFile LOBs. This helps you decide upfront whether you want to compress BasicFile LOBs, before migrating BasicFile LOBs to SecureFile LOBs. You can also estimate the compression ratio at the LOB byte level and the time taken, in hours, to compress the LOB data in the table.

The compression ratio is estimated for the number of rows in the LOB column that you specify. For example, let's consider that the compression ratio is 2.33. It indicates that after you enable the compression feature, you can save around half of the space for the sampled rows in the LOB column.

Disclaimer: The compression ratio is an approximate value, which is calculated based on the sampled rows in the LOB column. The actual space that you save when you enable compression for the complete table may be different.

Syntax

The syntax to get the compression ratio differs for objects, LOBs, IOTs, and indexes on a table.

- Syntax to get the compression ratio for an object (table or index, default is table).

```
DBMS_COMPRESSION.GET_COMPRESSION_RATIO (
   scratchtbsname        IN     VARCHAR2,
   ownname               IN     VARCHAR2,
   objname               IN     VARCHAR2,
   subobjname            IN     VARCHAR2,
   comptype              IN     NUMBER,
   blkcnt_cmp            OUT    PLS_INTEGER,
   blkcnt_uncmp          OUT    PLS_INTEGER,
   row_cmp               OUT    PLS_INTEGER,
   row_uncmp             OUT    PLS_INTEGER,
   cmp_ratio             OUT    NUMBER,
   comptype_str          OUT    VARCHAR2,
   block_compr_ratio     OUT    PLS_INTEGER,
   byte_comp_ratio       OUT    NUMBER,
   subset_numrows        IN     NUMBER  DEFAULT COMP_RATIO_MINROWS,
   objtype               IN     PLS_INTEGER DEFAULT OBJTYPE_TABLE);
```

- Syntax to get compression ratio for BasicFile and SecureFile LOBs:

```
DBMS_COMPRESSION.GET_COMPRESSION_RATIO (
   scratchtbsname        IN     VARCHAR2,
   tabowner              IN     VARCHAR2,
   tabname               IN     VARCHAR2,
   lobname               IN     VARCHAR2,
   partname              IN     VARCHAR2,
   comptype              IN     NUMBER,
   blkcnt_cmp            OUT    PLS_INTEGER,
   blkcnt_uncmp          OUT    PLS_INTEGER,
   lobcnt                OUT    PLS_INTEGER,
   cmp_ratio             OUT    NUMBER,
   comptype_str          OUT    VARCHAR2,
   byte_comp_ratio       OUT    NUMBER,
   total_time            OUT    NUMBER
   subset_numrows        IN     NUMBER DEFAULT COMP_RATIO_LOB_MAXROWS);
```

- Syntax to get the compression ratio for all indexes on a table. The compression ratios are returned as a collection.

```
DBMS_COMPRESSION.GET_COMPRESSION_RATIO (
  scratchtbsname         IN     VARCHAR2,
  ownname                IN     VARCHAR2,
  tabname                IN     VARCHAR2,
  comptype               IN     NUMBER,
  index_cr               OUT    DBMS_COMPRESSION.COMPRECLIST,
  comptype_str           OUT    VARCHAR2,
  subset_numrows         IN     NUMBER DEFAULT COMP_RATIO_INDEX_MINROWS);
```

- Syntax to get the compression ratio for IOTs.

```
DBMS_COMPRESSION.GET_COMPRESSION_RATIO (
     scratchtbsname        IN     VARCHAR2,
     ownname               IN     VARCHAR2,
     objname               IN     VARCHAR2,
     subobjname            IN     VARCHAR2,
     comptype              IN     NUMBER,
     iotcomp_cr            OUT    DBMS_COMPRESSION.COMPRECLIST,
     comptype_str          OUT    VARCHAR2,
     subset_numrows        IN     NUMBER DEFAULT COMP_RATIO_INDEX_MINROWS);
```

Parameters

Table 53-4 GET_COMPRESSION_RATIO Procedure Parameters

Parameter	Description
`scratchtbsname`	Temporary scratch tablespace that can be used for analysis
`ownname` / `tabowner`	Schema of the table to analyze
`tabname`	Name of the table to analyze
`objname`	Name of the object
`subobjname`	Name of the partition or sub-partition of the object
`comptype`	Compression types for which analysis should be performed When the object is an index, only the following compression types are valid: `COMP_INDEX_ADVANCED_HIGH` (value `1024`) and `COMP_INDEX_ADVANCED_LOW` (value `2048`). **Note:** The following compression types cannot be specified in this parameter for any type of object: `COMP_BLOCK` (value `64`) and `COMP_BASIC` (value `4096`).
`blkcnt_cmp`	Number of blocks used by compressed sample of the table
`blkcnt_uncmp`	Number of blocks used by uncompressed sample of the table
`row_cmp`	Number of rows in a block in compressed sample of the table
`row_uncmp`	Number of rows in a block in uncompressed sample of the table
`cmp_ratio`	Compression ratio, `blkcnt_uncmp` divided by `blkcnt_cmp`. It provides the ratio of **blocks** occupied by the uncompressed data to the blocks occupied by the compressed data.
`comptype_str`	String describing the compression type
`subset_numrows`	Number of rows sampled to estimate compression ratio.
`objtype`	Type of the object, either `OBJTYPE_TABLE` or `OBJTYPE_INDEX`
`lobname`	Name of the LOB column
`partname`	In case of partitioned tables, the related partition name
`lobcnt`	Number of lobs actually sampled to estimate compression ratio

Table 53-4 (Cont.) GET_COMPRESSION_RATIO Procedure Parameters

Parameter	Description
`byte_comp_ratio`	Provides the ratio of bytes of uncompressed data to the bytes of compressed data for LOBs.
`index_cr`	List of indexes and their estimated compression ratios
`iotcomp_cr`	Compression ratio for the IOT The first object contains the compression ratio for the whole IOT. The second object contains the compression ratio only for the top index section of the IOT (excludes the overflow segment).
`total_time`	Provides an estimate of the time taken, in hours, to compress the LOB data in the table.

Example: Estimate the compression ratio for inline and out-of-line LOBs

The following example shows how to estimate the compression ratio for LOBs.

```
SET SERVEROUTPUT ON
DECLARE
    bcmp                    INTEGER;
    buncmp                  INTEGER;
    lobcmp                  INTEGER;
    cr                      NUMBER;
    byte_cr                 NUMBER;
    cstr                    VARCHAR2(2000);
    total_time              NUMBER;
    l_segment_name          VARCHAR2(30);
    l_segment_size_blocks   NUMBER;
    l_segment_size_bytes    NUMBER;
    l_used_blocks           NUMBER;
    l_used_bytes            NUMBER;
    l_expired_blocks        NUMBER;
    l_expired_bytes         NUMBER;
    l_unexpired_blocks      NUMBER;
    l_unexpired_bytes       NUMBER;
BEGIN
   DBMS_COMPRESSION.GET_COMPRESSION_RATIO (
     scratchtbsname        => 'LOBTBSP',
     tabowner              => 'CMPADV',
     tabname               => p_tablename,
     lobname               => 'C',
     partname              => NULL,
     comptype              => 256,
     blkcnt_cmp            => bcmp,
     blkcnt_uncmp          => buncmp,
     lobcnt                => lobcmp,
     cmp_ratio             => cr,
     comptype_str          => cstr,
     subset_numrows        => 1000,
     byte_comp_ratio       => byte_cr,
     total_time            => total_time
);
DBMS_OUTPUT.put_line('Estimated ratio of blocks used by the uncompressed data
to the compressed data :  ' || cr);
DBMS_OUTPUT.put_line('Estimated ratio of bytes used by the uncompressed data
to the compressed data  :  ' || byte_cr);
END;
/
```

To understand the output of this procedure, let's consider `tab_inline`, an inline table, and `tab_outofline`, an out-of-line table as shown in the following example.

```
CREATE TABLE tab_inline
(
    a NUMBER,
    c CLOB
)
LOB(c) STORE AS SECUREFILE  (ENABLE STORAGE IN ROW CACHE LOGGING);


CREATE TABLE tab_outofline
(
    a NUMBER,
    c CLOB
)
LOB(c) STORE AS SECUREFILE  (DISABLE STORAGE IN ROW CACHE LOGGING);
```

Data is stored in different ways in `tab_inline` and `tab_outofline`. In the `tab_inline` table, if the LOB is less than 4K, then data is stored in the table segment; otherwise, it is stored in the LOB segment. For the `tab_outofline` table, data of all sizes is stored in the LOB segment.

Let's consider that you have inserted 1000 LOBs of 3K each in both the tables, and then calculate the compression ratios. You can use the `dbms_space.space_usage` procedure to calculate the space used by the data that is stored in the LOB segments.

Sample output of compression ratio for inline LOBs.

```
Estimated block compression ratio : 1
Estimated byte compression ratio          : 57.6
Space used(in bytes)                      : 0
space used(in blocks)                     : 0
```

Sample output of compression ratio for out-of-line LOBs.

```
Estimated block compression ratio : 1
Estimated byte compression ratio          : 56.1
Space used(in bytes)                      : 8 MB
space used(in blocks)                     : 1000
```

In this example, even though the estimated byte and block compression ratios are almost the same for inline and out-of-line LOBs, the space that is used is different. In the case of `tab_inline`, LOB segment is not used so the space used is 0. In both cases, the data is approximately 3KB, which is small. Therefore, the data before and after compression uses the same number of blocks (that is 1 block), so the block compression ratio is 1. However, the byte level compression ratio, `byte_comp_ratio`, which compares the actual number of bytes used by the LOBs before and after compression is 57.6 or 56.1.

Example: Estimate the compression ratio for indexes on a table with low compression type

The following example shows how to estimate the compression ratio for advanced index compression (low):

```
SET SERVEROUTPUT ON
DECLARE
  l_blkcnt_cmp       PLS_INTEGER;
  l_blkcnt_uncmp   PLS_INTEGER
  l_row_cmp       PLS_INTEGER;
  l_row_uncmp       PLS_INTEGER;
  l_cmp_ratio       NUMBER;
  l_comptype_str   VARCHAR2(32767);
BEGIN
   DBMS_COMPRESSION.GET_COMPRESSION_RATIO (
    scratchtbsname =>       'USERS' ,
    ownname       =>       'TEST' ,
    objname       =>       'SALES_IDX' ,
    subobjname     =>        NULL ,
    comptype       =>        DBMS_COMPRESSION.COMP_INDEX_ADVANCED_LOW,
    blkcnt_cmp     =>        l_blkcnt_cmp,
    blkcnt_uncmp   =>        l_blkcnt_uncmp,
    row_cmp       =>        l_row_cmp,
    row_uncmp      =>        l_row_uncmp,
    cmp_ratio      =>        l_cmp_ratio,
    comptype_str   =>        l_comptype_str,
    subset_numrows =>        DBMS_COMPRESSION.comp_ratio_minrows,
    objtype       =>        DBMS_COMPRESSION.objtype_index
   );
DBMS_OUTPUT.put_line( 'Number of blocks used by the compressed sample of the
object   :  ' || l_blkcnt_cmp);
DBMS_OUTPUT.put_line( 'Number of blocks used by the uncompressed sample of
the object :  ' || l_blkcnt_uncmp);
DBMS_OUTPUT.put_line( 'Number of rows in a block in compressed sample of the
object   :  ' || l_row_cmp);
DBMS_OUTPUT.put_line( 'Number of rows in a block in uncompressed sample of
the object :  ' || l_row_uncmp);
DBMS_OUTPUT.put_line( 'Estimated Compression Ratio of
Sample                          :  ' || l_cmp_ratio);
DBMS_OUTPUT.put_line( 'Compression Type                              :  ' ||
l_comptype_str);
END;
/
```

Output of compression advisor estimate for advanced index compression (Low):

```
Number of blocks used by the compressed sample of the object     : 243
Number of blocks used by the uncompressed sample of the object   : 539
Number of rows in a block in compressed sample of the object     : 499
Number of rows in a block in uncompressed sample of the object   : 145
Estimated Compression Ratio of Sample                  : 2.2
Compression Type                             : “Compress Advanced Low”
```

Example: Estimate the compression ratio for LOBs with medium compression type

The following example shows how to estimate the compression ratio for advanced LOB compression (medium):

```
SET SERVEROUTPUT ON
DECLARE
  l_blkcnt_cmp      PLS_INTEGER;
  l_blkcnt_uncmp    PLS_INTEGER;
  l_row_cmp        PLS_INTEGER;
  l_lobcnt        PLS_INTEGER;
  l_cmp_ratio       NUMBER;
  l_comptype_str    VARCHAR2(32767);
BEGIN
   DBMS_COMPRESSION.GET_COMPRESSION_RATIO (
    scratchtbsname => 'USERS' ,
    tabowner        => 'TEST' ,
    tabname        => 'PARTS' ,
    lobname        =>  'PART_DESCRIPTION' ,
    partname        =>  NULL ,
    comptype        =>  DBMS_COMPRESSION.COMP_LOB_MEDIUM,
    blkcnt_cmp      => l_blkcnt_cmp,
    blkcnt_uncmp    => l_blkcnt_uncmp,
    row_cmp        => l_row_cmp,
    lobcnt        => l_lobcnt,
    cmp_ratio       => l_cmp_ratio,
    comptype_str    => l_comptype_str,
    subset_numrows => DBMS_COMPRESSION.comp_ratio_lob_maxrows
   );
DBMS_OUTPUT.put_line( 'Number of blocks used by the compressed sample of the
object    :  ' || l_blkcnt_cmp);
DBMS_OUTPUT.put_line( 'Number of blocks used by the uncompressed sample of
the object  :  ' || l_blkcnt_uncmp);
DBMS_OUTPUT.put_line( 'Number of rows in a block in compressed sample of the
object    :  ' || l_row_cmp);
DBMS_OUTPUT.put_line( 'Number of LOBS actually
sampled                      :  ' || l_lobcnt);
DBMS_OUTPUT.put_line( 'Estimated Compression Ratio of
Sample                           :  ' || l_cmp_ratio);
DBMS_OUTPUT.put_line( 'Compression
Type                                                 :  ' || l_comptype_str);
END;
/
```

Output of compression advisor estimate for advanced LOB compression (Medium):

```
Number of blocks used by the compressed sample of the object   : 199
Number of blocks used by the uncompressed sample of the object     : 389
Number of rows in a block in compressed sample of the object       : 293
Number of LOBS actually sampled                                : 55
Estimated Compression Ratio of Sample                          : 1.9
Compression Type                                         : “Compress Medium”
```

Example: Estimate the compression ratio for IoTs

The following example shows how to estimate the compression ratio for IOTs:

```
SET SERVEROUTPUT ON
DECLARE
  bcmp       INTEGER;
  buncmp     INTEGER;
  rowcmp     INTEGER;
  rowuncmp   INTEGER;
  cr         NUMBER;
  cstr       VARCHAR2(2000);
  iotcomp_cr DBMS_COMPRESSION.COMPRECLIST;
 BEGIN
    DBMS_COMPRESSION.GET_COMPRESSION_RATIO (
    scratchtbsname        => 'USERS',
    ownname               => 'TEST',
    objname               => 'SALES',
    subobjname            =>  NULL,
    comptype              =>  DBMS_COMPRESSION.COMP_INDEX_ADVANCED_LOW,
    iotcomp_cr            =>  iotcomp_cr,
    comptype_str          =>  cstr,
    subset_numrows        =>  DBMS_COMPRESSION.COMP_RATIO_ALLROWS
    );
 --information about the index and the overflow segment
 DBMS_OUTPUT.put_line( 'Number of blocks used by the compressed sample of the
IOT table                                   : ' || iotcomp_cr(1).blkcnt_cmp);
 DBMS_OUTPUT.put_line( 'Number of blocks used by the uncompressed sample of
the IOT table                                 : ' ||
iotcomp_cr(1).blkcnt_uncmp);
 DBMS_OUTPUT.put_line( 'Average number of rows in a block in the compressed
sample of the IOT table                       : ' || iotcomp_cr(1).row_cmp);
 DBMS_OUTPUT.put_line( 'Average number of rows in a block in the uncompressed
sample of the IOT table                     : ' || iotcomp_cr(1).row_uncmp);
 DBMS_OUTPUT.put_line( 'Estimated Compression Ratio of the
sample                                                        : ' ||
iotcomp_cr(1).cmp_ratio);
 --information about the index segment
 DBMS_OUTPUT.put_line( 'Number of blocks used by the compressed sample of the
index segment of the IOT table              : ' || iotcomp_cr(2).blkcnt_cmp);
 DBMS_OUTPUT.put_line( 'Number of blocks used by the uncompressed sample of
the index segment of the IOT table            : ' ||
iotcomp_cr(2).blkcnt_uncmp);
 DBMS_OUTPUT.put_line( 'Average number of rows in a block in the compressed
sample of the index segment of the IOT table  : ' || iotcomp_cr(2).row_cmp);
 DBMS_OUTPUT.put_line( 'Average number of rows in a block in the uncompressed
sample of the index segment of the IOT table : ' || iotcomp_cr(2).row_uncmp);
 DBMS_OUTPUT.put_line( 'Estimated Compression Ratio of the
sample                                                        : ' ||
iotcomp_cr(2).cmp_ratio);
 END;
 /
```

Output of the compression ratio for IOTs:

```
Number of blocks used by the compressed sample of the IOT
table                                    : 5027
Number of blocks used by the uncompressed sample of the IOT
table                                  : 7950
Average number of rows in a block in the compressed sample of the IOT
table                        : 199
Average number of rows in a block in the uncompressed sample of the IOT
table                      : 126
Estimated Compression Ratio of the
sample                                                          : 1.58
Number of blocks used by the compressed sample of the index segment of the IOT
table               : 3238
Number of blocks used by the uncompressed sample of the index segment of the IOT
table             : 6161
Average number of rows in a block in the compressed sample of the index segment of the
IOT table   : 309
Average number of rows in a block in the uncompressed sample of the index segment of the
IOT table : 162
Estimated Compression Ratio of the
sample                                                          : 1.9
```

Usage Notes

- The procedure creates different tables in the scratch tablespace and runs analysis on these objects. It does not modify anything in the user-specified tables.
- From 23ai onwards, this feature has been enhanced to estimate the compression ratio faster for LOBs while using less space. To get a more accurate result, run the following command to switch to the old method. The older method to calculate the compression ratio takes more time to return the results and uses more space.

  ```
  alter session set "_kdlf_new_compression_adv"= FALSE;
  ```

- To understand the impact of compression, use the value of the byte compression ratio for inline LOBs and for out-of-line LOBs, use the value of the block compression ratio and space used.
- You can get more benefits when you compress large volume of data as compared to small volumes of data. If you want to compress small volumes of data, look at the byte ratio instead of the block ratio to understand the impact of compression.

GET_COMPRESSION_TYPE Function

This function returns the compression type for a specified row. If the row is chained, the function returns the compression type of the head piece only, and does not examine the intermediate or the tail piece since head pieces can be differently compressed.

Syntax

```
DBMS_COMPRESSION.GET_COMPRESSION_TYPE (
   ownname      IN    VARCHAR2,
   tabname      IN    VARCHAR2,
   row_id       IN    ROWID,
   subobjname   IN    VARCHAR2 DEFAULT NULL))
  RETURN NUMBER;
```

Parameters

Table 53-5 GET_COMPRESSION_TYPE Function Parameters

Parameter	Description
`ownname`	Schema name of the table
`tabname`	Name of table
`rowid`	Rowid of the row
`subobjname`	Name of the table partition or subpartition

Return Values

Flag to indicate the compression type (see Table 53-1).