```
*** Create a table based on dba_tables

SQL> CREATE TABLE compression_test AS SELECT * FROM dba_tables;

Table created.

SQL> INSERT INTO compression_test SELECT * FROM compression_test;

1728 rows created.

SQL> INSERT INTO compression_test SELECT * FROM compression_test;

3456 rows created.

SQL> INSERT INTO compression_test SELECT * FROM compression_test;

6912 rows created.

SQL> INSERT INTO compression_test SELECT * FROM compression_test;

13824 rows created.

SQL> INSERT INTO compression_test SELECT * FROM compression_test;

27648 rows created.

SQL> commit;

Commit complete.


*** Create a standard, nocompress index

SQL> CREATE INDEX standard_index ON compression_test(owner, table_name)
PCTFREE 0;

Index created.

SQL> exec dbms_stats.gather_table_stats(ownname=> 'BOWIE', tabname=>
'COMPRESSION_TEST', estimate_percent=> null, cascade=> true, method_opt=>
'FOR ALL COLUMNS SIZE 1');

PL/SQL procedure successfully completed.

SQL> ANALYZE INDEX standard_index VALIDATE STRUCTURE;

Index analyzed.

SQL> SELECT HEIGHT, BR_BLKS, LF_BLKS FROM INDEX_STATS;

    HEIGHT    BR_BLKS    LF_BLKS
---------- ---------- ----------
         3          3        237


*** Notice how the index requires 237 leaf blocks ...


*** Following is a block dump of one of these leaf blocks (embedded
comments start with <== )
```

```
Leaf block dump
===============
header address 194146396=0xb92705c
kdxcolev 0
KDXCOLEV Flags = - - -
kdxcolok 0
kdxcoopc 0x80: opcode=0: iot flags=--- is converted=Y
kdxconco 3
kdxcosdc 0
kdxconro 230
<== We can store 230 index entries in this particular leaf block
kdxcofbo 496=0x1f0
kdxcofeo 500=0x1f4
kdxcoavs 4
kdxlespl 0
kdxlende 0
kdxlenxt 75512983=0x4803c97
kdxleprv 75512981=0x4803c95
kdxledsz 0
kdxlebksz 8036
row#0[8005] flag: ------, lock: 0, len=31
col 0; len 3; (3):  53 59 53
col 1; len 17; (17):  57 52 49 24 5f 41 44 56 5f 53 51 4c 57 5f 53 55 4d
```

<== Notice how the owner column and table_name columns are repeated a number of times within the leaf block

```
col 2; len 6; (6):  04 80 40 ec 00 1c
row#1[7974] flag: ------, lock: 0, len=31
col 0; len 3; (3):  53 59 53
col 1; len 17; (17):  57 52 49 24 5f 41 44 56 5f 53 51 4c 57 5f 53 55 4d
col 2; len 6; (6):  04 80 41 21 00 0d
row#2[7943] flag: ------, lock: 0, len=31
col 0; len 3; (3):  53 59 53
col 1; len 17; (17):  57 52 49 24 5f 41 44 56 5f 53 51 4c 57 5f 53 55 4d
col 2; len 6; (6):  04 80 41 55 00 1c
row#3[7912] flag: ------, lock: 0, len=31
col 0; len 3; (3):  53 59 53
col 1; len 17; (17):  57 52 49 24 5f 41 44 56 5f 53 51 4c 57 5f 53 55 4d
col 2; len 6; (6):  04 80 41 8a 00 0d
row#4[7881] flag: ------, lock: 0, len=31
col 0; len 3; (3):  53 59 53
col 1; len 17; (17):  57 52 49 24 5f 41 44 56 5f 53 51 4c 57 5f 53 55 4d
col 2; len 6; (6):  04 80 41 be 00 1c
row#5[7850] flag: ------, lock: 0, len=31
col 0; len 3; (3):  53 59 53
col 1; len 17; (17):  57 52 49 24 5f 41 44 56 5f 53 51 4c 57 5f 53 55 4d
col 2; len 6; (6):  04 80 41 f3 00 0d
row#6[7819] flag: ------, lock: 0, len=31
col 0; len 3; (3):  53 59 53
col 1; len 17; (17):  57 52 49 24 5f 41 44 56 5f 53 51 4c 57 5f 53 55 4d
col 2; len 6; (6):  04 80 42 27 00 1c
row#7[7788] flag: ------, lock: 0, len=31
col 0; len 3; (3):  53 59 53
col 1; len 17; (17):  57 52 49 24 5f 41 44 56 5f 53 51 4c 57 5f 53 55 4d
col 2; len 6; (6):  04 80 42 5c 00 0d
row#8[7754] flag: ------, lock: 0, len=34
col 0; len 3; (3):  53 59 53
col 1; len 20; (20):  57 52 49 24 5f 41 44 56 5f 53 51 4c 57 5f 54 41 42 4c
45 53
```

<== At this point, we finally have a different value for the table_name column

```
col 2; len 6; (6):  04 80 36 a4 00 13
row#9[7720] flag: ------, lock: 0, len=34
col 0; len 3; (3):  53 59 53
col 1; len 20; (20):  57 52 49 24 5f 41 44 56 5f 53 51 4c 57 5f 54 41 42 4c
45 53
col 2; len 6; (6):  04 80 36 d9 00 0f
row#10[7686] flag: ------, lock: 0, len=34
col 0; len 3; (3):  53 59 53
col 1; len 20; (20):  57 52 49 24 5f 41 44 56 5f 53 51 4c 57 5f 54 41 42 4c
45 53
col 2; len 6; (6):  04 80 37 8d 00 1e
row#11[7652] flag: ------, lock: 0, len=34
col 0; len 3; (3):  53 59 53
col 1; len 20; (20):  57 52 49 24 5f 41 44 56 5f 53 51 4c 57 5f 54 41 42 4c
45 53
col 2; len 6; (6):  04 80 37 c2 00 0f
row#12[7618] flag: ------, lock: 0, len=34
col 0; len 3; (3):  53 59 53
col 1; len 20; (20):  57 52 49 24 5f 41 44 56 5f 53 51 4c 57 5f 54 41 42 4c
45 53
col 2; len 6; (6):  04 80 37 f6 00 1e
row#13[7584] flag: ------, lock: 0, len=34
col 0; len 3; (3):  53 59 53
col 1; len 20; (20):  57 52 49 24 5f 41 44 56 5f 53 51 4c 57 5f 54 41 42 4c
45 53
col 2; len 6; (6):  04 80 38 2b 00 0f
row#14[7550] flag: ------, lock: 0, len=34
col 0; len 3; (3):  53 59 53
col 1; len 20; (20):  57 52 49 24 5f 41 44 56 5f 53 51 4c 57 5f 54 41 42 4c
45 53
col 2; len 6; (6):  04 80 38 5f 00 1e
row#15[7516] flag: ------, lock: 0, len=34
col 0; len 3; (3):  53 59 53
col 1; len 20; (20):  57 52 49 24 5f 41 44 56 5f 53 51 4c 57 5f 54 41 42 4c
45 53
col 2; len 6; (6):  04 80 38 94 00 0f


*** Example 1: Compression of 2 columns

*** We now create the index again, but this time with the COMPRESS option.
As the index is non-unique, both columns are compressed


SQL> CREATE INDEX compressed_index ON compression_test(owner, table_name)
PCTFREE 0 COMPRESS;

Index created.


SQL> SELECT HEIGHT, BR_BLKS, LF_BLKS FROM INDEX_STATS;

    HEIGHT    BR_BLKS    LF_BLKS
---------- ---------- ----------
         2          1         83
```

*** The number of leaf blocks has reduced significantly, enough so in fact to also reduce the overall height of the index ...

*** If we look at a partial dump of a leaf block now ...

```
Leaf block dump
===============
header address 170599004=0xa2b225c
kdxcolev 0
KDXCOLEV Flags = - - -
kdxcolok 0
kdxcoopc 0xa0: opcode=0: iot flags=-C- is converted=Y
kdxconco 3
kdxcosdc 0
kdxconro 669
kdxcofbo 1466=0x5ba
kdxcofeo 1474=0x5c2
kdxcoavs 8
kdxlespl 0
kdxlende 0
kdxlenxt 75511565=0x480370d
kdxleprv 75511563=0x480370b
kdxledsz 0
kdxlebksz 8036
kdxlepnro 22
kdxlepnco 2
prefix row#0[8006] flag: -P----, lock: 0, len=30
```

<== we notice a new Prefix section in the block. Each unique combination of compressed columns is listed here

```
col 0; len 8; (8):  41 46 50 31 31 35 35 34
```

<== first unique combination of the owner and

```
col 1; len 18; (18):  46 4d 5f 50 45 52 53 4f 4e 5f 4f 52 47 5f 49 4e 43 54
```

<== table_name column (prefix row#0)

```
prc 2
prefix row#1[7963] flag: -P----, lock: 0, len=25
col 0; len 8; (8):  41 46 50 31 31 35 35 34
<== second unique combination of the owner and
col 1; len 13; (13):  46 4d 5f 50 45 52 5f 41 43 54 55 41 4c
```

<== table_name column (prefix row#1)

```
prc 32
prefix row#2[7649] flag: -P----, lock: 0, len=26
col 0; len 8; (8):  41 46 50 31 31 35 35 34
```

<== and so on ...

```
col 1; len 14; (14):  46 4d 5f 53 54 41 54 5f 50 45 52 46 4f 52
prc 32
prefix row#3[7336] flag: -P----, lock: 0, len=25
col 0; len 8; (8):  41 46 50 31 31 35 35 34
col 1; len 13; (13):  46 4d 5f 54 45 4d 50 5f 53 45 43 55 52
```

```
prc 32
prefix row#4[7028] flag: -P----, lock: 0, len=20
col 0; len 8; (8):  41 46 50 31 31 35 35 34
col 1; len 8; (8):  46 55 4e 43 54 49 4f 4e
prc 32

.
.
.

prefix row#21[1717] flag: -P----, lock: 0, len=19
```

<== in fact, there are a total of 22 unique combinations of owner and table_name in this particular leaf block

```
col 0; len 8; (8):  41 46 50 31 31 35 35 34
col 1; len 7; (7):  4f 52 47 55 4e 49 54
prc 27
row#0[7997] flag: ------, lock: 0, len=9
```

<== we now notice that the first actual index row entry consists of just

```
col 0; len 6; (6):  04 80 42 53 00 0b
```

<== the rowid column and

```
psno 0
```

<== a positional reference to the first prefix row entry #0

```
row#1[7988] flag: ------, lock: 0, len=9
col 0; len 6; (6):  04 80 42 87 00 19
psno 0
row#2[7954] flag: ------, lock: 0, len=9
col 0; len 6; (6):  04 80 36 9b 00 0f
psno 1
row#3[7945] flag: ------, lock: 0, len=9
col 0; len 6; (6):  04 80 36 d0 00 0d
psno 1
```

<== the fourth index row entry now references the second prefix row entry #1

```
row#4[7936] flag: ------, lock: 0, len=9
col 0; len 6; (6):  04 80 37 04 00 1a

.
.
.

row#666[1492] flag: ------, lock: 0, len=9
col 0; len 6; (6):  04 80 41 13 00 12
psno 21
row#667[1483] flag: ------, lock: 0, len=9
col 0; len 6; (6):  04 80 41 48 00 01
psno 21
row#668[1474] flag: ------, lock: 0, len=9
col 0; len 6; (6):  04 80 41 7c 00 12
psno 21
```

<== and the last few index row entry references the last prefix row entry #21

----- end of leaf block dump -----
End dump data blocks tsn: 21 file#: 18 minblk 14092 maxblk 14092

*** Therefore, we now only store all these repeated values once and once only and the specific index row entries simply reference back to the corresponding prefix row entry

*** As a result, we can store 669 index entries in the leaf block rather than the 230 odd we stored previously