```
*** Create a table and non-reverse index with monotonically increasing IDs


SQL> CREATE TABLE reverse_details (id NUMBER, name VARCHAR2(20));

Table created.

SQL> CREATE INDEX normal_index ON reverse_details(id);

Index created.

SQL> INSERT INTO reverse_details SELECT rownum, 'David Bowie' FROM dual
CONNECT BY LEVEL <= 1000000;

1000000 rows created.

SQL> COMMIT;

Commit complete.


*** Analyze index

SQL> ANALYZE INDEX normal_index VALIDATE STRUCTURE;

Index analyzed.

SQL> SELECT blocks, lf_blks, pct_used FROM index_stats;

    BLOCKS    LF_BLKS   PCT_USED
---------- ---------- ----------
      2048       1999        100


*** PCT_USED is 100% as each insert ID value is the maximum, causing
efficient 90-10 block splits


*** Repeat same thing but this time with a Reverse Key index


SQL> TRUNCATE TABLE reverse_details;

Table truncated.

SQL> CREATE INDEX reverse_index ON reverse_details(id) REVERSE;

Index created.

SQL> INSERT INTO reverse_details SELECT rownum, 'David Bowie' FROM dual
CONNECT BY LEVEL <= 1000000;

1000000 rows created.

SQL> COMMIT;

Commit complete.

SQL> ANALYZE INDEX normal_index VALIDATE STRUCTURE;

Index analyzed.
```

```
SQL> SELECT blocks, lf_blks, pct_used FROM index_stats;

    BLOCKS    LF_BLKS   PCT_USED
---------- ---------- ----------
      3072       2966         68


*** Blocks have increased by approximately 50% and pct_used is now only 68%

*** As IDs are now reversed and inserted "randomly" within the index, block
splits are now 50-50 resulting in a far less compact index structure


-------------------------------------------------



*** However, reverse indexes may have advantages if there are many
deletions that result in many sparsely populated blocks

*** Create and populate a table and non-reverse index similar to before


SQL> CREATE TABLE reverse_details_del1 (id NUMBER, name VARCHAR2(20));

Table created.

SQL> CREATE INDEX del1_normal_index ON reverse_details_del1(id);

Index created.

SQL> INSERT INTO reverse_details_del1 SELECT rownum, 'David Bowie' FROM
dual CONNECT BY LEVEL <= 1000000;

1000000 rows created.

SQL> COMMIT;

Commit complete.


*** But this time delete a whole bunch of rows that still leaves some data
in the individual leaf blocks

*** Only 1 row in 250 is not deleted


SQL> DELETE reverse_details_del1 WHERE MOD(id,250) <> 0;

996000 rows deleted.

SQL> COMMIT;

Commit complete.



*** Now insert a whole bunch of new rows with IDs greater than previous
values
```

*** This will make the index continue with 90-10 splits but it will not be able to reuse any of the deleted space from the existing leaf blocks

```
SQL> INSERT INTO reverse_details_del1 SELECT rownum+1000000, 'David Bowie'
FROM dual CONNECT BY LEVEL <= 1000000;

1000000 rows created.

SQL> COMMIT;

Commit complete.

SQL> ANALYZE INDEX del1_normal_index VALIDATE STRUCTURE;

Index analyzed.

SQL> SELECT lf_blks, pct_used FROM index_stats;

   LF_BLKS   PCT_USED
---------- ----------
      4126         53
```

*** At the end of this process, PCT_USED is only 53%. If this were to continue, this figure will only get worse ...

*** Repeat with a Reverse Key Index

```
SQL> CREATE TABLE reverse_details_del2 (id NUMBER, name VARCHAR2(20));

Table created.

SQL> CREATE INDEX del2_reverse_index ON reverse_details_del2(id) REVERSE;

Index created.

SQL> INSERT INTO reverse_details_del2 SELECT rownum, 'David Bowie' FROM
dual CONNECT BY LEVEL <= 1000000;

1000000 rows created.

SQL> COMMIT;

Commit complete.

SQL> DELETE reverse_details_del2 WHERE MOD(id,250) <> 0;

996000 rows deleted.

SQL> COMMIT;

Commit complete.
```

*** Note when we insert more values, these are distributed within the existing index structure and deleted index space can be reused

```
SQL> INSERT INTO reverse_details_del2 SELECT rownum+1000000, 'David Bowie'
FROM dual CONNECT BY LEVEL <= 1000000;

1000000 rows created.

SQL> COMMIT;

Commit complete.

SQL> ANALYZE INDEX del2_reverse_index VALIDATE STRUCTURE;

Index analyzed.

SQL> SELECT lf_blks, pct_used FROM index_stats;

   LF_BLKS   PCT_USED
---------- ----------
      3354         64
```

*** The reverse index now uses significantly fewer blocks than the other non-reverse index and the PCT_USED is better at 64%.

*** This difference will only improve over time in comparison to the other non-reverse index.