```
*** Create a new table called david_bowie that we'll be searching for in a moment

SQL> CREATE table david_bowie (id NUMBER);

Table created.


*** Create a table from dba_objects which includes a row for the newly created david_bowie table

SQL> CREATE TABLE reverse_stuff AS SELECT * FROM dba_objects;

Table created.


*** Start by creating a normal, non-reverse key index

SQL> CREATE INDEX reverse_object_name_i ON reverse_stuff(object_name);

Index created.


*** Let's collect stats

SQL> EXEC dbms_stats.gather_table_stats(ownname=>'BOWIE', tabname=>'REVERSE_STUFF',
estimate_percent=> null, cascade=> TRUE, method_opt=> 'FOR ALL COLUMNS SIZE 1');

PL/SQL procedure successfully completed.



*** Let's see if our normal little index will be used for the following query with a LIKE and a
leading wildcard ...

SQL> set feedback 1


SQL> SELECT * FROM reverse_stuff WHERE object_name LIKE '%BOWIE';

1 row selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 518781941

--------------------------------------------------------------------------------------
| Id  | Operation         | Name          | Rows  | Bytes | Cost (%CPU)| Time  |
--------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |               |   722 | 62814 |    44   (3)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| REVERSE_STUFF |   722 | 62814 |    44   (3)| 00:00:01 |
--------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("OBJECT_NAME" LIKE '%BOWIE')


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        191  consistent gets
          0  physical reads
          0  redo size
       1205  bytes sent via SQL*Net to client
        396  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

*** We note the index is not used (or considered) because of the leading wildcard ...
*** But it did correctly return the one row of interest as expected

*** Partial extract from a 10053 trace

SINGLE TABLE ACCESS PATH
  Column (#2): OBJECT_NAME(VARCHAR2)
    AvgLen: 19.00 NDV: 10862 Nulls: 0 Density: 9.2064e-005
```

```
  Table: REVERSE_STUFF  Alias: REVERSE_STUFF
    Card: Original: 14437  Rounded: 722  Computed: 721.85  Non Adjusted: 721.85
  Access Path: TableScan
    Cost:  43.52  Resp: 43.52  Degree: 0
      Cost_io: 43.00  Cost_cpu: 5388539
      Resp_io: 43.00  Resp_cpu: 5388539
  Best:: AccessPath: TableScan
         Cost: 43.52  Degree: 1  Resp: 43.52  Card: 721.85  Bytes: 0


*** The 10053 trace shows that the index was not considered by the CBO



*** Unfortunately, the suggestion to use a Reverse Key Index makes matters a lot worse ...

*** Let's rebuild the index as a reverse key index

SQL> ALTER INDEX reverse_object_name_i REBUILD ONLINE REVERSE COMPUTE STATISTICS;

Index altered.


*** Let's now programmatically reverse the required text and see what happens ...

SQL> SELECT * FROM reverse_stuff WHERE object_name LIKE 'EIWOB%';

no rows selected


Execution Plan
----------------------------------------------------------
Plan hash value: 518781941

--------------------------------------------------------------------------------------
| Id  | Operation         | Name          | Rows  | Bytes | Cost (%CPU)|    Time  |
--------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |               |     1 |    87 |    44   (3)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| REVERSE_STUFF |     1 |    87 |    44   (3)| 00:00:01 |
--------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("OBJECT_NAME" LIKE 'EIWOB%')


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        190  consistent gets
          0  physical reads
          0  redo size
        995  bytes sent via SQL*Net to client
        385  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed


*** Worse because it still performs a full table scan

*** Partial extract from a 10053 trace

SINGLE TABLE ACCESS PATH
  Table: REVERSE_STUFF  Alias: REVERSE_STUFF
    Card: Original: 14437  Rounded: 14437  Computed: 14437.00  Non Adjusted: 14437.00
  Access Path: TableScan
    Cost:  43.36  Resp: 43.36  Degree: 0
      Cost_io: 43.00  Cost_cpu: 3785999
      Resp_io: 43.00  Resp_cpu: 3785999
  Best:: AccessPath: TableScan
         Cost: 43.36  Degree: 1  Resp: 43.36  Card: 14437.00  Bytes: 0


*** worse still as it still doesn't even consider the index

*** and much much worse, no longer returns the correct results (now returns no rows !!)
```

*** as of course Oracle has no idea that the searched data has been programmatically reversed and is naturally searching for an object_name beginning with 'EIWOB' ...

*** Another Oracle index related myth that can be easily disclaimed with a few very simple tests ...

*** However, you could achieve success with the use of a Function-Based Index using the REVERSE function

```
SQL> CREATE INDEX reverse_stuff_func_idx ON reverse_stuff(reverse(object_name));

Index created.

SQL> EXEC dbms_stats.gather_table_stats(ownname=>'BOWIE', tabname=>'REVERSE_STUFF',
estimate_percent=> null, cascade=> TRUE, method_opt=> 'FOR ALL COLUMNS SIZE 1');

PL/SQL procedure successfully completed.
```

*** Let's now perform this troublesome select statement by using the REVERSE function to reverse the search column
*** and programmatically reverse the required search string which places the wildcard at the end

```
SQL> SELECT * FROM reverse_stuff WHERE reverse(object_name) LIKE 'EIWOB%';

1 row selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 3662976059

-------------------------------------------------------------------------------------------------------
--
| Id  | Operation                   | Name                   | Rows  | Bytes | Cost (%CPU)| Time
|
-------------------------------------------------------------------------------------------------------
--
|   0 | SELECT STATEMENT            |                        |     2 |   196 |   3   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| REVERSE_STUFF          |     2 |   196 |   3   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | REVERSE_STUFF_FUNC_IDX |     2 |       |   2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------
--


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access(REVERSE("OBJECT_NAME") LIKE 'EIWOB%')
       filter(REVERSE("OBJECT_NAME") LIKE 'EIWOB%')


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
          4  consistent gets
          0  physical reads
          0  redo size
       1203  bytes sent via SQL*Net to client
        396  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

*** The index is used and the correct output is generated

*** You can tidy the SELECT statement up a bit and make it more readable by using the REVERSE function on the search string as well

```
SQL> SELECT * FROM reverse_stuff WHERE reverse(object_name) LIKE REVERSE('%BOWIE');

1 row selected.


Execution Plan
```

```
-------------------------------------------------------------
Plan hash value: 3662976059

-------------------------------------------------------------------------------------------------------
--
| Id  | Operation                    | Name                    | Rows  | Bytes | Cost (%CPU)| Time
|
-------------------------------------------------------------------------------------------------------
--
|   0 | SELECT STATEMENT             |                         |     2 |   196 |   3   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| REVERSE_STUFF           |     2 |   196 |   3   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | REVERSE_STUFF_FUNC_IDX |     2 |       |   2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------
--


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access(REVERSE("OBJECT_NAME") LIKE 'EIWOB%')
       filter(REVERSE("OBJECT_NAME") LIKE 'EIWOB%')


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
          4  consistent gets
          0  physical reads
          0  redo size
       1203  bytes sent via SQL*Net to client
        396  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```