```
*** Start by creating a simple table

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

Table created.


*** Next, create a Reverse Key Index on the id column (Note: a non-unique index is being used)

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

Index created.


*** Let's insert a whole bunch of rows and collect statistics

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

1000000 rows created.

SQL> COMMIT;

Commit complete.

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

PL/SQL procedure successfully completed.


*** Let's attempt a very simple, innocent looking range scan predicate

*** But first, let's start a 10053 trace to see what execution plans the CBO considers


SQL> ALTER SESSION SET EVENTS '10053 trace name context forever, level 1';

Session altered.


SQL> SELECT * FROM reverse_details WHERE id BETWEEN 42 AND 43;

        ID NAME
---------- --------------------
        42 David Bowie
        43 David Bowie


Execution Plan
----------------------------------------------------------
Plan hash value: 3030292439

-------------------------------------------------------------------------------------
| Id  | Operation         | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                 |     3 |    48 |   680   (3)| 00:00:09 |
|*  1 |  TABLE ACCESS FULL| REVERSE_DETAILS |     3 |    48 |   680   (3)| 00:00:09 |
-------------------------------------------------------------------------------------


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

   1 - filter("ID"<=43 AND "ID">=42)


*** No good, Oracle performed a Full Table Scan even though we were only after 2 rows ...


*** A partial dump of the 10053 dump reveals the following ...



BASE STATISTICAL INFORMATION
***********************
Table Stats::
  Table: REVERSE_DETAILS  Alias: REVERSE_DETAILS
    #Rows: 1000000  #Blks:  3033  AvgRowLen:  16.00
Index Stats::
  Index: REVERSE_INDEX  Col#: 1
    LVLS: 2  #LB: 2966  #DK: 1000000  LB/K: 1.00  DB/K: 1.00  CLUF: 999994.00
***************************************
```

```
SINGLE TABLE ACCESS PATH
  Column (#1): ID(NUMBER)
    AvgLen: 5.00 NDV: 1000000 Nulls: 0 Density: 1.0000e-006 Min: 1 Max: 1000000
  Table: REVERSE_DETAILS  Alias: REVERSE_DETAILS
    Card: Original: 1000000  Rounded: 3  Computed: 3.00  Non Adjusted: 3.00
  Access Path: TableScan
    Cost:  679.86  Resp: 679.86  Degree: 0
      Cost_io: 665.00  Cost_cpu: 221601538
      Resp_io: 665.00  Resp_cpu: 221601538
  Best:: AccessPath: TableScan
         Cost: 679.86  Degree: 1  Resp: 679.86  Card: 3.00  Bytes: 0


*** The CBO is fully aware of the reverse key index as shown in the Index stats
*** But in the Single Table Access Path does not even consider the Reverse Key Index as a valid
option ...


*** You can try to hint the thing as much as you want but the CBO does not consider Index Range
Scans with Range Predicates.

*** You might be able to generate an Full Index Scan out of Oracle ...


SQL> SELECT /*+ INDEX(rd) */ * FROM reverse_details rd WHERE id BETWEEN 42 AND 43;

        ID NAME
---------- --------------------
        42 David Bowie
        43 David Bowie


Execution Plan
----------------------------------------------------------
Plan hash value: 3845841859

-----------------------------------------------------------------------------------------------
| Id  | Operation                    | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                 |     3 |    48 |     3033(1)| 00:00:37 |
|   1 |  TABLE ACCESS BY INDEX ROWID| REVERSE_DETAILS |     3 |    48 |     3033(1)| 00:00:37 |
|*  2 |   INDEX FULL SCAN            | REVERSE_INDEX   |    43 |       |     2990(1)| 00:00:36 |
-----------------------------------------------------------------------------------------------


*** Equality conditions are not a problem ...


SQL> SELECT * FROM reverse_details WHERE id = 42;

        ID NAME
---------- --------------------
        42 David Bowie


Execution Plan
----------------------------------------------------------
Plan hash value: 1002750038

-----------------------------------------------------------------------------------------------
| Id  | Operation                    | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                 |     1 |    16 |        3(0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| REVERSE_DETAILS |     1 |    16 |        3(0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | REVERSE_INDEX   |     1 |       |        2(0)| 00:00:01 |
-----------------------------------------------------------------------------------------------


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

   2 - access("ID"=42)



*** If a range predicate can be rewritten as an IN condition
*** Oracle can convert the predicate to separate OR equality conditions and can use the Reverse Key
Index


SQL> SELECT * FROM reverse_details WHERE id IN (42, 43);
```

```
        ID NAME
---------- --------------------
        42 David Bowie
        43 David Bowie


Execution Plan
----------------------------------------------------------
Plan hash value: 1015170962

------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                 |     2 |    32 |      6 (0)| 00:00:01 |
|   1 |  INLIST ITERATOR             |                 |       |       |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID| REVERSE_DETAILS |     2 |    32 |      6 (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN          | REVERSE_INDEX   |     2 |       |      4 (0)| 00:00:01 |
------------------------------------------------------------------------------------------------


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

   3 - access("ID"=42 OR "ID"=43)



*** If a range scan is "really" an equality condition, then again, not a problem


SQL> SELECT * FROM reverse_details WHERE id BETWEEN 42 AND 42;

        ID NAME
---------- --------------------
        42 David Bowie


Execution Plan
----------------------------------------------------------
Plan hash value: 1002750038

-----------------------------------------------------------------------------------------------
| Id  | Operation                   | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                 |     1 |    16 |      3(0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| REVERSE_DETAILS |     1 |    16 |      3(0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | REVERSE_INDEX   |     1 |       |      2(0)| 00:00:01 |
-----------------------------------------------------------------------------------------------


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

   2 - access("ID"=42)



*** Another example

SQL> SELECT * FROM reverse_details WHERE id >= 42 AND id <= 42;

        ID NAME
---------- --------------------
        42 David Bowie


Execution Plan
----------------------------------------------------------
Plan hash value: 1002750038

-----------------------------------------------------------------------------------------------
| Id  | Operation                   | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                 |     1 |    16 |      3(0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| REVERSE_DETAILS |     1 |    16 |      3(0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | REVERSE_INDEX   |     1 |       |      2(0)| 00:00:01 |
-----------------------------------------------------------------------------------------------


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

```
   2 - access("ID"=42)
```

```
*** LIKE predicates are also Range Predicates that cause Reverse Key Indexes to be ignored by the
CBO

*** Just creating another table with a character based reverse key index


SQL> CREATE TABLE reverse_stuff AS SELECT * FROM dba_objects;

Table created.

SQL> CREATE INDEX reverse_object_name_i ON reverse_stuff(object_name) REVERSE;

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.

SQL> ALTER SESSION SET EVENTS '10053 trace name context forever, level 1';

Session altered.

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


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 'REVERSE%')


*** Again, the 10053 trace shows how the reverse key index is ignored by the CBO


*** Portion of 10053 trace file

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: 1  Computed: 1.33  Non Adjusted: 1.33
  Access Path: TableScan
    Cost:  43.50  Resp: 43.50  Degree: 0
      Cost_io: 43.00  Cost_cpu: 5229919
      Resp_io: 43.00  Resp_cpu: 5229919
  Best:: AccessPath: TableScan
         Cost: 43.50  Degree: 1  Resp: 43.50  Card: 1.33  Bytes: 0



*** Note, index access path not even considered by the CBO



*** However, again Oracle can pick up on when a LIKE is really equivalent to an equality predicate


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


Execution Plan
----------------------------------------------------------
Plan hash value: 3482594567

-----------------------------------------------------------------------------------------------
```

```
| Id  | Operation                   | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                      |     1 |    87 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| REVERSE_STUFF        |     1 |    87 |     2   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | REVERSE_OBJECT_NAME_I |     1 |       |     1   (0)| 00:00:01 |
------------------------------------------------------------------------------------------------


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

   2 - access("OBJECT_NAME" LIKE 'BOWIE')


*** portion of 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: 1  Computed: 1.33  Non Adjusted: 1.33
  Access Path: TableScan
    Cost:  43.43  Resp: 43.43  Degree: 0
      Cost_io: 43.00  Cost_cpu: 4508069
      Resp_io: 43.00  Resp_cpu: 4508069
  Access Path: index (AllEqRange)
    Index: REVERSE_OBJECT_NAME_I
    resc_io: 2.00  resc_cpu: 16273
    ix_sel: 9.2064e-005  ix_sel_with_filters: 9.2064e-005
    Cost: 2.00  Resp: 2.00  Degree: 1
  Best:: AccessPath: IndexRange  Index: REVERSE_OBJECT_NAME_I
         Cost: 2.00  Degree: 1  Resp: 2.00  Card: 1.33  Bytes: 0


*** The index is considered in this example ....


*** Note in all these examples, a Non-Unique index has been used and in each example, Oracle has
been using an Index range Scan ...


*** Note also that a Unique Index can also use an Index Range scan if it has more than one column
and the leading column known but not all other columns are used.


SQL> CREATE TABLE reverse_test AS SELECT * FROM dba_tables;

Table created.

SQL> CREATE UNIQUE INDEX reverse_test_uk ON reverse_test(table_name, owner) REVERSE;

Index created.


SQL> SELECT * FROM reverse_test WHERE table_name = 'TEST2';


Execution Plan
----------------------------------------------------------
Plan hash value: 1917598036

-----------------------------------------------------------------------------------------------
| Id  | Operation                   | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                 |     1 |   212 |        3(0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| REVERSE_TEST    |     1 |   212 |        3(0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | REVERSE_TEST_UK |     1 |       |        2(0)| 00:00:01 |
-----------------------------------------------------------------------------------------------


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

   2 - access("TABLE_NAME"='TEST2')


*** Oracle will ignore Reverse Key Index for Range *Predicates*

*** But can use *Index Range Scans* if the index is Non-Unique or not all columns (but at least the
leading column) of a Unique Index is used
```