```
-- Create a table and populate it with data that has an implied relationship between
-- the ID and CODE columns
-- Create multiple rows of each combination, 10000 rows of each ID and CODE
-- combination to be precise.

SQL> create table radiohead (id number, code varchar2(5), name varchar2(20));

Table created.

SQL> begin
  2  for i in 1..10000 loop
  3    insert into radiohead values(1, 'AAA', 'Description A');
  4    insert into radiohead values(2, 'BBB', 'Description B');
  5    insert into radiohead values(3, 'CCC', 'Description C');
  6    insert into radiohead values(4, 'DDD', 'Description D');
  7    insert into radiohead values(5, 'EEE', 'Description E');
  8    insert into radiohead values(6, 'FFF', 'Description F');
  9    insert into radiohead values(7, 'GGG', 'Description G');
 10    insert into radiohead values(8, 'HHH', 'Description H');
 11    insert into radiohead values(9, 'III', 'Description I');
 12    insert into radiohead values(10, 'JJJ', 'Description J');
 13  end loop;
 14  commit;
 15  end;
 16  /

PL/SQL procedure successfully completed.



-- Create a second table, but this one with only the one occurance of each row

SQL> create table ok_computer (id number, code varchar2(5), description varchar2(20));

Table created.

SQL> begin
  2    insert into ok_computer values(1, 'AAA', 'Description A');
  3    insert into ok_computer values(2, 'BBB', 'Description B');
  4    insert into ok_computer values(3, 'CCC', 'Description C');
  5    insert into ok_computer values(4, 'DDD', 'Description D');
  6    insert into ok_computer values(5, 'EEE', 'Description E');
  7    insert into ok_computer values(6, 'FFF', 'Description F');
  8    insert into ok_computer values(7, 'GGG', 'Description G');
  9    insert into ok_computer values(8, 'HHH', 'Description H');
 10    insert into ok_computer values(9, 'III', 'Description I');
 11    insert into ok_computer values(10, 'JJJ', 'Description J');
 12    commit;
 13  end;
 14  /

PL/SQL procedure successfully completed.


-- Create an index only on the second table

SQL> alter table ok_computer add primary key(id, code);

Table altered.


-- Collect stats on both tables

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

PL/SQL procedure successfully completed.

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

PL/SQL procedure successfully completed.


-- Run a query that performs a cartesian join on both tables for a specific ID and
-- CODE combination. Because the CBO doesn't recognise the relationship between the ID
-- and CODE columns it gets the rows cardinality (1000) wrong on the RADIOHEAD table
-- and references this table first in the execution plan

SQL> set autotrace traceonly
SQL> select * from radiohead r, ok_computer o where r.id = 5 and r.code = 'EEE';


100000 rows selected.


Execution Plan
```

```
-------------------------------------------------------------
Plan hash value: 3348432975

-------------------------------------------------------------------------------------------
| Id  | Operation            | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |             | 10000 |   410K|   373   (1)| 00:00:05 |
|   1 |  MERGE JOIN CARTESIAN|             | 10000 |   410K|   373   (1)| 00:00:05 |
|*  2 |   TABLE ACCESS FULL  | RADIOHEAD   |  1000 | 21000 |   100   (1)| 00:00:02 |
|   3 |   BUFFER SORT        |             |    10 |   210 |   272   (0)| 00:00:04 |
|   4 |    TABLE ACCESS FULL | OK_COMPUTER |    10 |   210 |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------


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

   2 - filter("R"."ID"=5 AND "R"."CODE"='EEE')


Statistics
-----------------------------------------------------------
          1  recursive calls
          0  db block gets
        386  consistent gets
          0  physical reads
          0  redo size
    2603137  bytes sent via SQL*Net to client
        605  bytes received via SQL*Net from client
         21  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
     100000  rows processed


-- Now create an index on both the ID and CODE columns from the RADIOHEAD table

SQL> create index radiohead_idx on radiohead(id, code);

Index created.


-- Running the same query again and this time the CBO uses the index statistics to
-- determine the correct row cardinality (10000) and now references the RADIOHEAD
-- table last in the exection plan

SQL> select * from radiohead r, ok_computer o where r.id = 5 and r.code = 'EEE';


100000 rows selected.


Execution Plan
-----------------------------------------------------------
Plan hash value: 3583729066

-------------------------------------------------------------------------------------------
| Id  | Operation            | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |             |   100K|  4101K|   989   (2)| 00:00:12 |
|   1 |  MERGE JOIN CARTESIAN|             |   100K|  4101K|   989   (2)| 00:00:12 |
|   2 |   TABLE ACCESS FULL  | OK_COMPUTER |    10 |   210 |     2   (0)| 00:00:01 |
|   3 |   BUFFER SORT        |             | 10000 |   205K|   987   (2)| 00:00:12 |
|*  4 |    TABLE ACCESS FULL | RADIOHEAD   | 10000 |   205K|    99   (2)| 00:00:02 |
-------------------------------------------------------------------------------------------


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

   4 - filter("R"."ID"=5 AND "R"."CODE"='EEE')


Statistics
-----------------------------------------------------------
          1  recursive calls
          0  db block gets
        375  consistent gets
          0  physical reads
          0  redo size
    1203277  bytes sent via SQL*Net to client
        605  bytes received via SQL*Net from client
         21  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
     100000  rows processed
```

```
-- Now make the index on the RADIOHEAD table "invisible" ...

SQL> alter index radiohead_idx invisible;

Index altered.

SQL>  SELECT index_name, visibility FROM user_indexes WHERE index_name =
'RADIOHEAD_IDX';

INDEX_NAME                     VISIBILITY
------------------------------ ---------------
RADIOHEAD_IDX                  INVISIBLE


-- However, when we run the same query again, the execution plan remains the same.
-- The CBO is still using the index statistics to get the correct cardinality
-- estimates, even though the index itself is invisible

SQL> select * from radiohead r, ok_computer o where r.id = 5 and r.code = 'EEE';


100000 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 3583729066

-------------------------------------------------------------------------------------------
| Id  | Operation            | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |             |   100K|  4101K|   989   (2)| 00:00:12 |
|   1 |  MERGE JOIN CARTESIAN|             |   100K|  4101K|   989   (2)| 00:00:12 |
|   2 |   TABLE ACCESS FULL  | OK_COMPUTER |    10 |   210 |     2   (0)| 00:00:01 |
|   3 |   BUFFER SORT        |             | 10000 |   205K|   987   (2)| 00:00:12 |
|*  4 |    TABLE ACCESS FULL | RADIOHEAD   | 10000 |   205K|    99   (2)| 00:00:02 |
-------------------------------------------------------------------------------------------


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

   4 - filter("R"."ID"=5 AND "R"."CODE"='EEE')


Statistics
----------------------------------------------------------
        181  recursive calls
          0  db block gets
        401  consistent gets
          0  physical reads
          0  redo size
    1203277  bytes sent via SQL*Net to client
        605  bytes received via SQL*Net from client
         21  SQL*Net roundtrips to/from client
          7  sorts (memory)
          0  sorts (disk)
     100000  rows processed


-- If we now actually drop the index ...

SQL> drop index radiohead_idx;

Index dropped.


-- The query now reverts back to the original execution plan with the wrong
-- cardinality estimates for the RADIOHEAD table ...

SQL> select * from radiohead r, ok_computer o where r.id = 5 and r.code = 'EEE';


100000 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 3348432975

-------------------------------------------------------------------------------------------
| Id  | Operation            | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |             | 10000 |   410K|   373   (1)| 00:00:05 |
|   1 |  MERGE JOIN CARTESIAN|             | 10000 |   410K|   373   (1)| 00:00:05 |
|*  2 |   TABLE ACCESS FULL  | RADIOHEAD   |  1000 | 21000 |   100   (1)| 00:00:02 |
```

```
|   3 |   BUFFER SORT        |              |    10 |   210 |   272   (0)| 00:00:04 |
|   4 |    TABLE ACCESS FULL | OK_COMPUTER |    10 |   210 |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------


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

   2 - filter("R"."ID"=5 AND "R"."CODE"='EEE')


Statistics
----------------------------------------------------------
       1004  recursive calls
          0  db block gets
        542  consistent gets
          0  physical reads
          0  redo size
    2603137  bytes sent via SQL*Net to client
        605  bytes received via SQL*Net from client
         21  SQL*Net roundtrips to/from client
         14  sorts (memory)
          0  sorts (disk)
     100000  rows processed
```