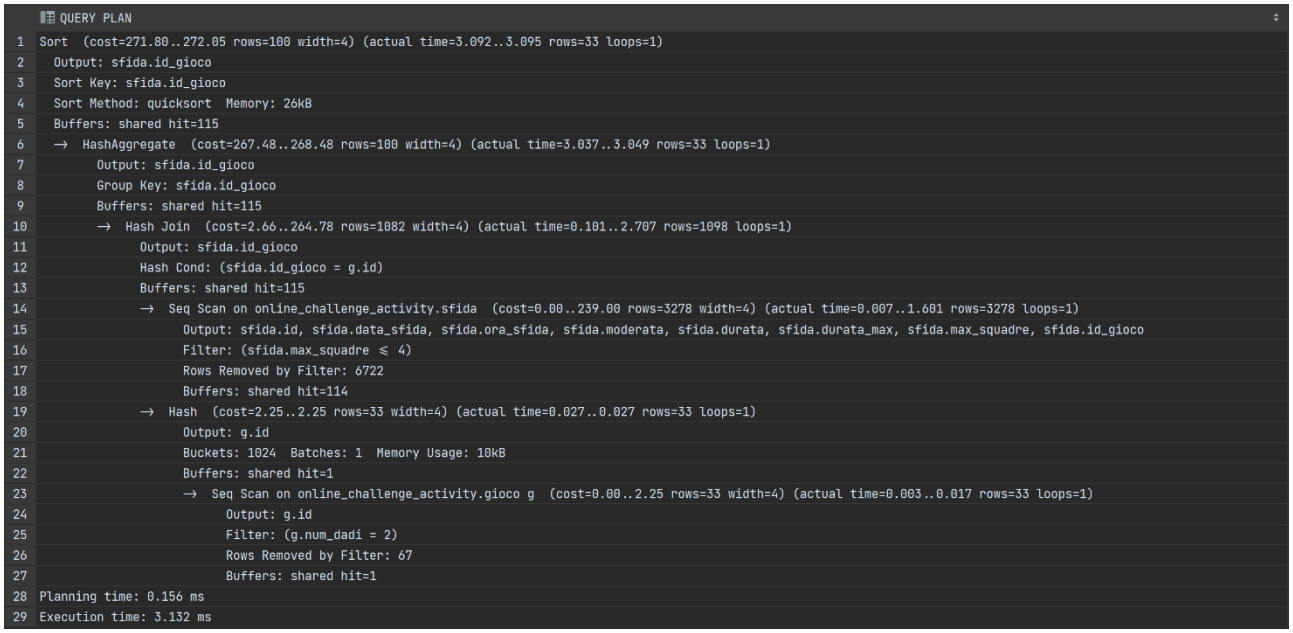
Basi di Dati A.A. 2020-2021 – Lorenzo Billi (3930391)

Progetto “Online Challenge Activity”

Piani di esecuzione

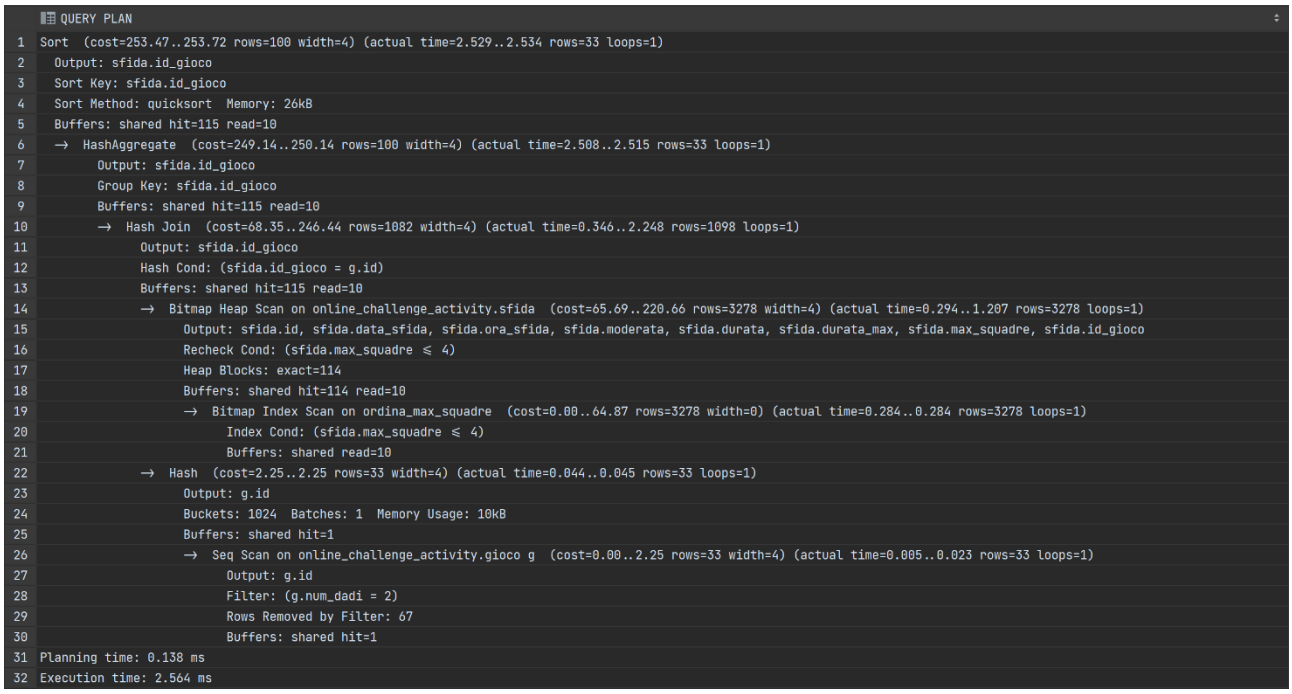
Piano di esecuzione per carico di lavoro 1 – senza indici



```
QUERY PLAN
Sort  (cost=271.80..272.05 rows=100 width=4) (actual time=3.092..3.095 rows=33 loops=1)
  Output: sfida.id_gioco
  Sort Key: sfida.id_gioco
  Sort Method: quicksort  Memory: 26kB
  Buffers: shared hit=115
  →  HashAggregate  (cost=267.48..268.48 rows=100 width=4) (actual time=3.037..3.049 rows=33 loops=1)
        Output: sfida.id_gioco
        Group Key: sfida.id_gioco
        Buffers: shared hit=115
        →  Hash Join  (cost=2.66..264.78 rows=1082 width=4) (actual time=0.101..2.707 rows=1098 loops=1)
              Output: sfida.id_gioco
              Hash Cond: (sfida.id_gioco = g.id)
              Buffers: shared hit=115
              →  Seq Scan on online_challenge_activity.sfida  (cost=0.00..239.00 rows=3278 width=4) (actual time=0.007..1.601 rows=3278 loops=1)
                    Output: sfida.id, sfida.data_sfida, sfida.ora_sfida, sfida.moderata, sfida.durata, sfida.durata_max, sfida.max_squadre, sfida.id_gioco
                    Filter: (sfida.max_squadre ⩽ 4)
                    Rows Removed by Filter: 6722
                    Buffers: shared hit=114
              →  Hash  (cost=2.25..2.25 rows=33 width=4) (actual time=0.027..0.027 rows=33 loops=1)
                    Output: g.id
                    Buckets: 1024  Batches: 1  Memory Usage: 10kB
                    Buffers: shared hit=1
                    →  Seq Scan on online_challenge_activity.gioco g  (cost=0.00..2.25 rows=33 width=4) (actual time=0.003..0.017 rows=33 loops=1)
                          Output: g.id
                          Filter: (g.num_dadi = 2)
                          Rows Removed by Filter: 67
                          Buffers: shared hit=1
Planning time: 0.156 ms
Execution time: 3.132 ms
```

Il carico di lavoro 1 prevede un JOIN, che viene eseguito dal sistema come HASH JOIN. La selezione avviene tramite scansione sequenziale di default. Il tempo di esecuzione totale è di circa 3 millisecondi.

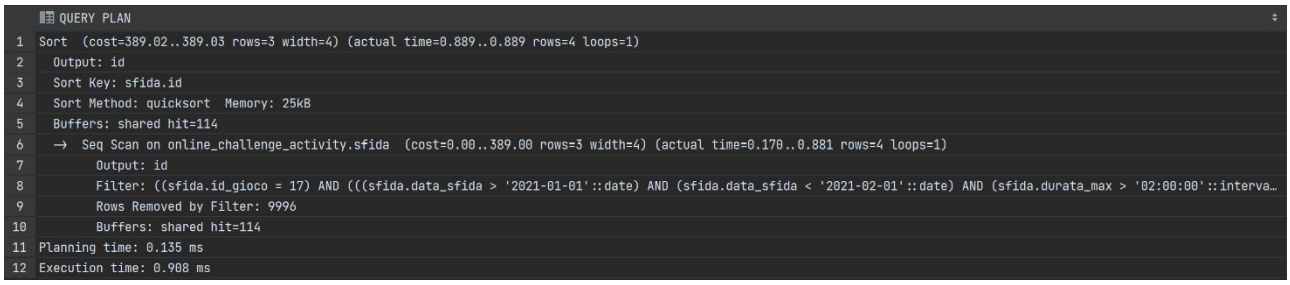
Piano di esecuzione per carico di lavoro 1 – con indici



```
QUERY PLAN
Sort  (cost=253.47..253.72 rows=100 width=4) (actual time=2.529..2.534 rows=33 loops=1)
  Output: sfida.id_gioco
  Sort Key: sfida.id_gioco
  Sort Method: quicksort  Memory: 26kB
  Buffers: shared hit=115 read=10
  →  HashAggregate  (cost=249.14..250.14 rows=100 width=4) (actual time=2.508..2.515 rows=33 loops=1)
        Output: sfida.id_gioco
        Group Key: sfida.id_gioco
        Buffers: shared hit=115 read=10
        →  Hash Join  (cost=68.35..246.44 rows=1082 width=4) (actual time=0.346..2.248 rows=1098 loops=1)
              Output: sfida.id_gioco
              Hash Cond: (sfida.id_gioco = g.id)
              Buffers: shared hit=115 read=10
              →  Bitmap Heap Scan on online_challenge_activity.sfida  (cost=65.69..220.66 rows=3278 width=4) (actual time=0.294..1.207 rows=3278 loops=1)
                    Output: sfida.id, sfida.data_sfida, sfida.ora_sfida, sfida.moderata, sfida.durata, sfida.durata_max, sfida.max_squadre, sfida.id_gioco
                    Recheck Cond: (sfida.max_squadre ⩽ 4)
                    Heap Blocks: exact=114
                    Buffers: shared hit=114 read=10
                    →  Bitmap Index Scan on ordina_max_squadre  (cost=0.00..64.87 rows=3278 width=0) (actual time=0.284..0.284 rows=3278 loops=1)
                          Index Cond: (sfida.max_squadre ⩽ 4)
                          Buffers: shared read=10
              →  Hash  (cost=2.25..2.25 rows=33 width=4) (actual time=0.044..0.045 rows=33 loops=1)
                    Output: g.id
                    Buckets: 1024  Batches: 1  Memory Usage: 10kB
                    Buffers: shared hit=1
                    →  Seq Scan on online_challenge_activity.gioco g  (cost=0.00..2.25 rows=33 width=4) (actual time=0.005..0.023 rows=33 loops=1)
                          Output: g.id
                          Filter: (g.num_dadi = 2)
                          Rows Removed by Filter: 67
                          Buffers: shared hit=1
Planning time: 0.138 ms
Execution time: 2.564 ms
```

Con l'introduzione degli indici, il JOIN viene sempre eseguito come HASH JOIN, con un costo medio leggermente superiore a prima, in quanto il file è stato clusterizzato. La scansione sequenziale tuttavia è ora più efficiente rispetto a prima, proprio grazie alla clusterizzazione. Questo accorcia il tempo di esecuzione di questo carico di lavoro a poco più di 2,5 millisecondi.

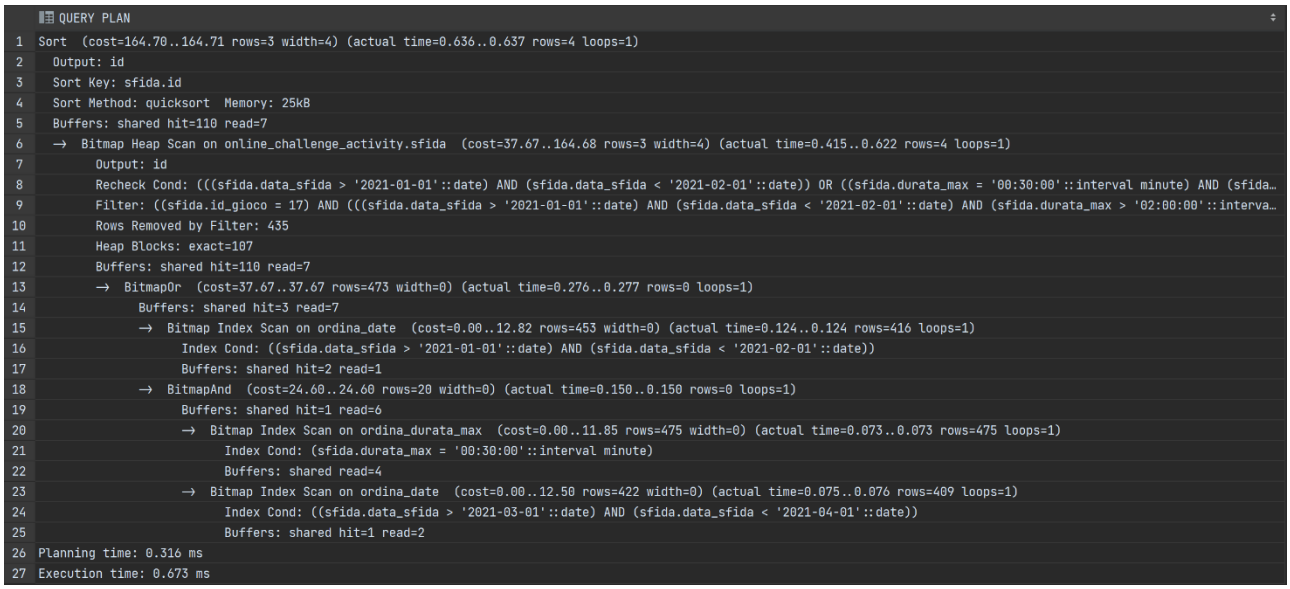
Piano di esecuzione per carico di lavoro 2 – senza indici



```
QUERY PLAN
Sort  (cost=389.02..389.03 rows=3 width=4) (actual time=0.889..0.889 rows=4 loops=1)
  Output: id
  Sort Key: sfida.id
  Sort Method: quicksort  Memory: 25kB
  Buffers: shared hit=114
  →  Seq Scan on online_challenge_activity.sfida  (cost=0.00..389.00 rows=3 width=4) (actual time=0.170..0.881 rows=4 loops=1)
        Output: id
        Filter: ((sfida.id_gioco = 17) AND (((sfida.data_sfida > '2021-01-01'::date) AND (sfida.data_sfida < '2021-02-01'::date) AND (sfida.durata_max > '02:00:00'::interva…
        Rows Removed by Filter: 9996
        Buffers: shared hit=114
Planning time: 0.135 ms
Execution time: 0.908 ms
```

Anche in questo caso il sistema esegue una scansione sequenziale, in quanto nel secondo carico di lavoro si sta operando molto sui range di valori. Tempo di esecuzione di poco inferiore al millisecondo.

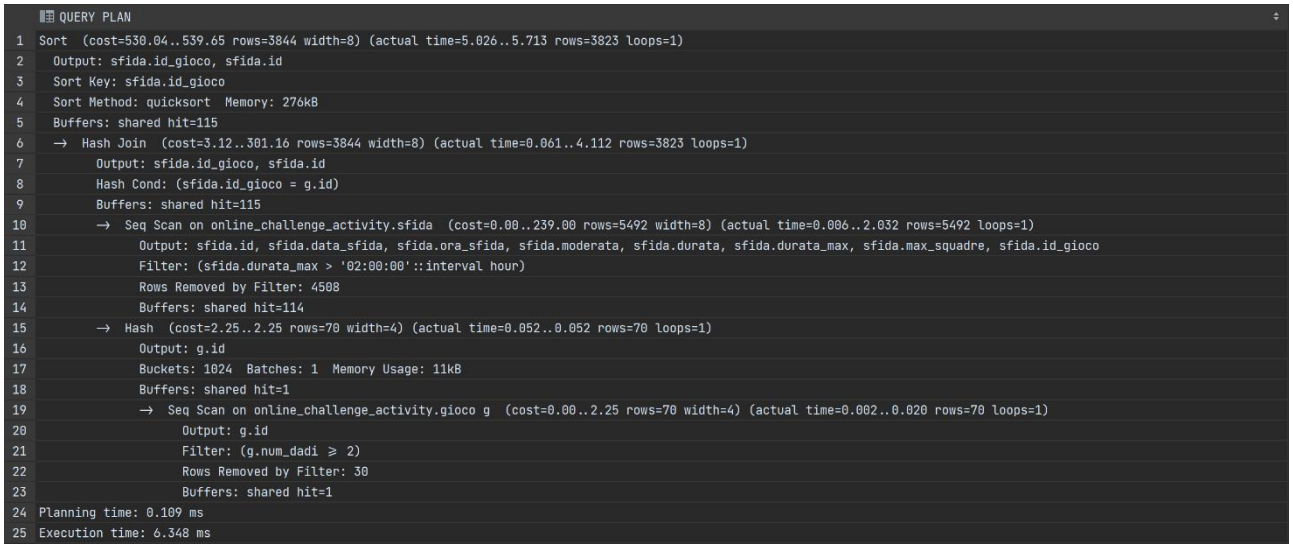
Piano di esecuzione per carico di lavoro 2 – con indici



```
QUERY PLAN
Sort  (cost=164.70..164.71 rows=3 width=4) (actual time=0.636..0.637 rows=4 loops=1)
  Output: id
  Sort Key: sfida.id
  Sort Method: quicksort  Memory: 25kB
  Buffers: shared hit=110 read=7
  →  Bitmap Heap Scan on online_challenge_activity.sfida  (cost=37.67..164.68 rows=3 width=4) (actual time=0.415..0.622 rows=4 loops=1)
        Output: id
        Recheck Cond: (((sfida.data_sfida > '2021-01-01'::date) AND (sfida.data_sfida < '2021-02-01'::date)) OR ((sfida.durata_max = '00:30:00'::interval minute) AND (sfida…
        Filter: ((sfida.id_gioco = 17) AND (((sfida.data_sfida > '2021-01-01'::date) AND (sfida.data_sfida < '2021-02-01'::date) AND (sfida.durata_max > '02:00:00'::interva…
        Rows Removed by Filter: 435
        Heap Blocks: exact=107
        Buffers: shared hit=110 read=7
        →  BitmapOr  (cost=37.67..37.67 rows=473 width=0) (actual time=0.276..0.277 rows=0 loops=1)
              Buffers: shared hit=3 read=7
              →  Bitmap Index Scan on ordina_date  (cost=0.00..12.82 rows=453 width=0) (actual time=0.124..0.124 rows=416 loops=1)
                    Index Cond: ((sfida.data_sfida > '2021-01-01'::date) AND (sfida.data_sfida < '2021-02-01'::date))
                    Buffers: shared hit=2 read=1
              →  BitmapAnd  (cost=24.60..24.60 rows=20 width=0) (actual time=0.150..0.150 rows=0 loops=1)
                    Buffers: shared hit=1 read=6
                    →  Bitmap Index Scan on ordina_durata_max  (cost=0.00..11.85 rows=475 width=0) (actual time=0.073..0.073 rows=475 loops=1)
                          Index Cond: (sfida.durata_max = '00:30:00'::interval minute)
                          Buffers: shared read=4
                    →  Bitmap Index Scan on ordina_date  (cost=0.00..12.50 rows=422 width=0) (actual time=0.075..0.076 rows=409 loops=1)
                          Index Cond: ((sfida.data_sfida > '2021-03-01'::date) AND (sfida.data_sfida < '2021-04-01'::date))
                          Buffers: shared hit=1 read=2
Planning time: 0.316 ms
Execution time: 0.673 ms
```

L'introduzione di un indice ordinato sulle date delle sfide riduce drasticamente il costo delle due scansioni sequenziali, e porta il tempo di esecuzione al di sotto del millisecondo (0,6).

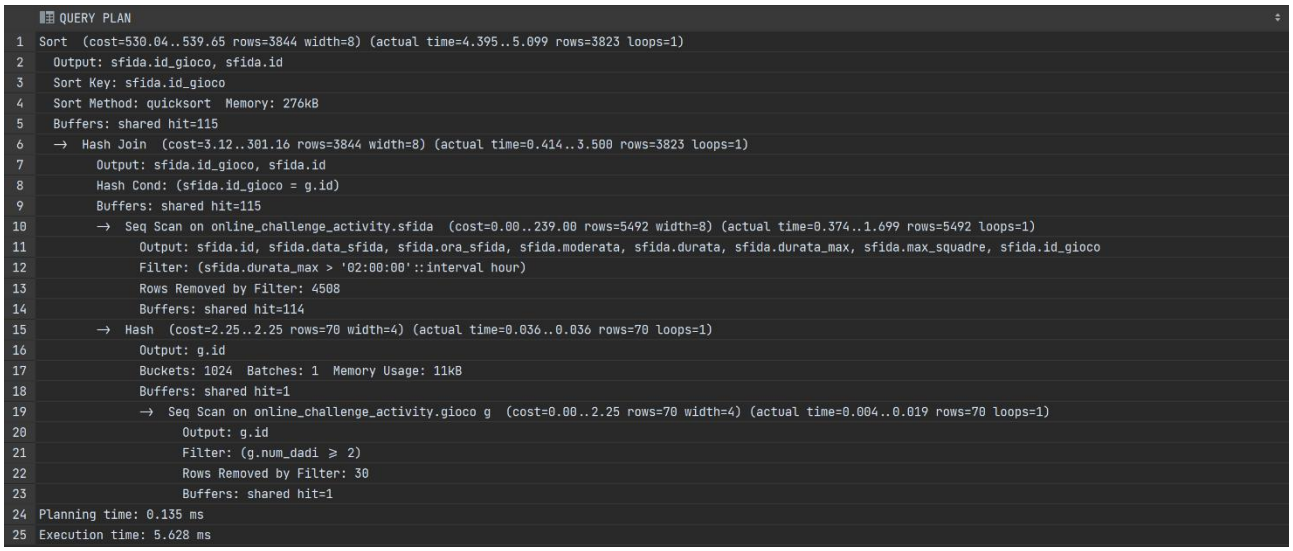
Piano di esecuzione per carico di lavoro 3 – senza indici



```
QUERY PLAN
Sort  (cost=530.04..539.65 rows=3844 width=8) (actual time=5.026..5.713 rows=3823 loops=1)
  Output: sfida.id_gioco, sfida.id
  Sort Key: sfida.id_gioco
  Sort Method: quicksort  Memory: 276kB
  Buffers: shared hit=115
  →  Hash Join  (cost=3.12..301.16 rows=3844 width=8) (actual time=0.061..4.112 rows=3823 loops=1)
        Output: sfida.id_gioco, sfida.id
        Hash Cond: (sfida.id_gioco = g.id)
        Buffers: shared hit=115
        →  Seq Scan on online_challenge_activity.sfida  (cost=0.00..239.00 rows=5492 width=8) (actual time=0.006..2.032 rows=5492 loops=1)
              Output: sfida.id, sfida.data_sfida, sfida.ora_sfida, sfida.moderata, sfida.durata, sfida.durata_max, sfida.max_squadre, sfida.id_gioco
              Filter: (sfida.durata_max > '02:00:00'::interval hour)
              Rows Removed by Filter: 4508
              Buffers: shared hit=114
        →  Hash  (cost=2.25..2.25 rows=70 width=4) (actual time=0.052..0.052 rows=70 loops=1)
              Output: g.id
              Buckets: 1024  Batches: 1  Memory Usage: 11kB
              Buffers: shared hit=1
              →  Seq Scan on online_challenge_activity.gioco g  (cost=0.00..2.25 rows=70 width=4) (actual time=0.002..0.020 rows=70 loops=1)
                    Output: g.id
                    Filter: (g.num_dadi ≥ 2)
                    Rows Removed by Filter: 30
                    Buffers: shared hit=1
Planning time: 0.109 ms
Execution time: 6.348 ms
```

Come nel caso del primo carico di lavoro, anche nel terzo carico di lavoro si ha a che fare con un JOIN, che viene implementato sempre tramite HASH JOIN. Tempo di esecuzione: 6,3 millisecondi.

Piano di esecuzione per carico di lavoro 3 – con indici



```
QUERY PLAN
Sort  (cost=530.04..539.65 rows=3844 width=8) (actual time=4.395..5.099 rows=3823 loops=1)
  Output: sfida.id_gioco, sfida.id
  Sort Key: sfida.id_gioco
  Sort Method: quicksort  Memory: 276kB
  Buffers: shared hit=115
  →  Hash Join  (cost=3.12..301.16 rows=3844 width=8) (actual time=0.414..3.500 rows=3823 loops=1)
        Output: sfida.id_gioco, sfida.id
        Hash Cond: (sfida.id_gioco = g.id)
        Buffers: shared hit=115
        →  Seq Scan on online_challenge_activity.sfida  (cost=0.00..239.00 rows=5492 width=8) (actual time=0.374..1.699 rows=5492 loops=1)
              Output: sfida.id, sfida.data_sfida, sfida.ora_sfida, sfida.moderata, sfida.durata, sfida.durata_max, sfida.max_squadre, sfida.id_gioco
              Filter: (sfida.durata_max > '02:00:00'::interval hour)
              Rows Removed by Filter: 4508
              Buffers: shared hit=114
        →  Hash  (cost=2.25..2.25 rows=70 width=4) (actual time=0.036..0.036 rows=70 loops=1)
              Output: g.id
              Buckets: 1024  Batches: 1  Memory Usage: 11kB
              Buffers: shared hit=1
              →  Seq Scan on online_challenge_activity.gioco g  (cost=0.00..2.25 rows=70 width=4) (actual time=0.004..0.019 rows=70 loops=1)
                    Output: g.id
                    Filter: (g.num_dadi ⩾ 2)
                    Rows Removed by Filter: 30
                    Buffers: shared hit=1
Planning time: 0.135 ms
Execution time: 5.628 ms
```

L’introduzione degli indici non ha in questo caso apportato significativi benefici in termini di costi di accesso al disco. E’ però interessante notare come il tempo di esecuzione sia comunque diminuito (5,6 millisecondi), tuttavia tale miglioramento è più da attribuire al buffer (che probabilmente ha ancora in memoria le tuple della precedente interrogazione senza indici) che agli indici in sé.