Projeto Otimização de Consultas Banco de Dados PostgreSQL

Realizado pesquisa para encontrar as maiores tabelas:

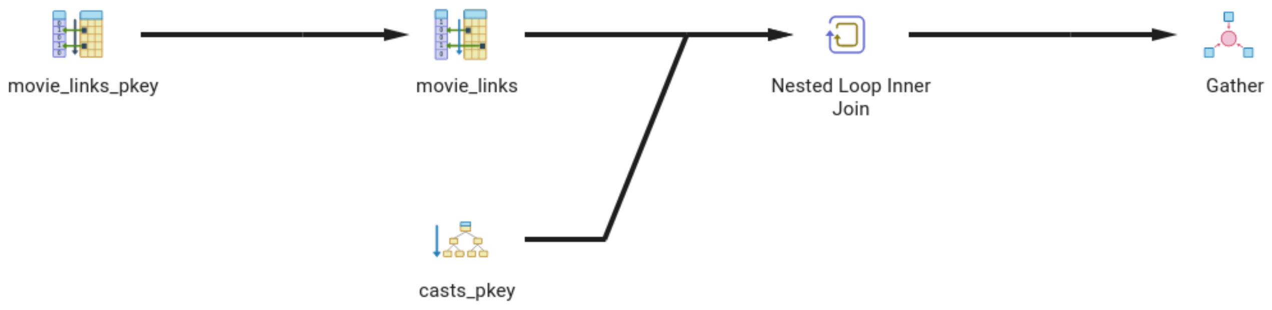
```
SELECT table_name, pg_size_pretty(pg_total_relation_size(quote_ident(table_name)))
FROM information_schema.tables WHERE table_schema = 'public' AND table_catalog =
'omdb';
```

Os resultados obtidos são:

```
 people             | 20 MB
 categories         | 1328 kB
 category_names     | 3568 kB
 image_ids          | 3768 kB
 image_licenses     | 5784 kB
 job_names          | 216 kB
 movie_abstracts_de | 17 MB
 movie_abstracts_en | 912 kB
 movie_abstracts_es | 344 kB
 access_log         | 8192 bytes
 jobs               | 96 kB
 movie_countries    | 6032 kB
 movie_languages    | 8496 kB
 movie_abstracts_fr | 72 kB
 movie_categories   | 13 MB
 people_links       | 19 MB
 trailers           | 1024 kB
 people_aliases     | 1320 kB
 casts              | 130 MB
 movies             | 22 MB
 casts_view         | 0 bytes
 movie_aliases_iso  | 28 MB
 movie_keywords     | 27 MB
 movie_links        | 32 MB
 movie_references   | 168 kB
(25 rows)
```

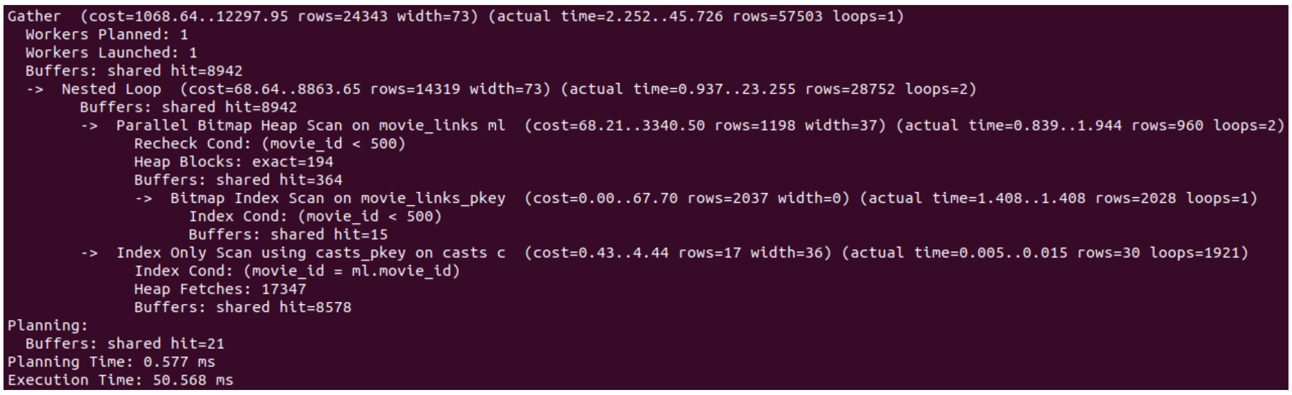
Pelo resultado da consulta foram selecionado as duas maiores tabelas sendo a casts com 130 MB e movie_links com 32 MB.

Consulta 01:6
select * from movie_links as ml, casts as c where ml.movie_id=c.movie_id and ml.movie_id < 500;



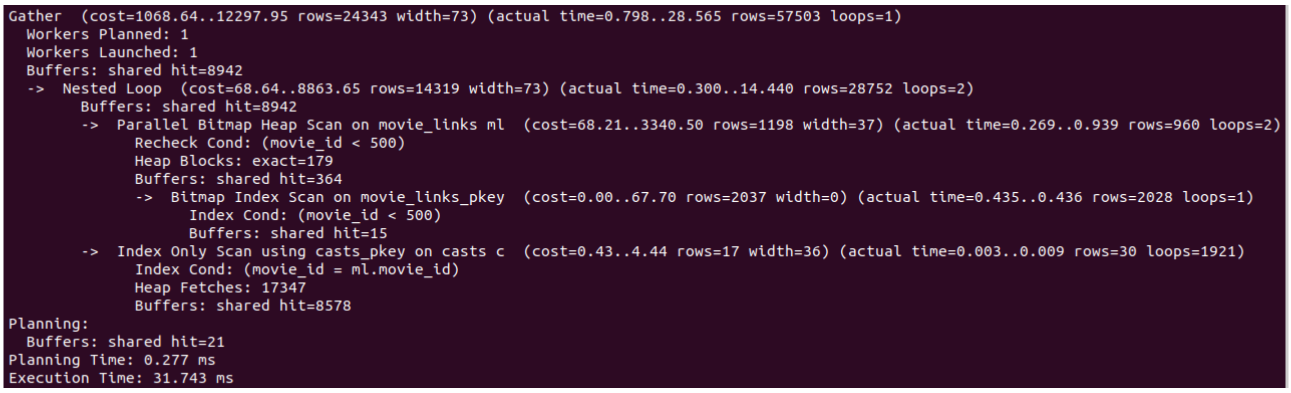
explain (buffers, costs, timing, analyse) select * from movie_links as ml, casts as c where ml.movie_id=c.movie_id and ml.movie_id < 500;

Interação 1:



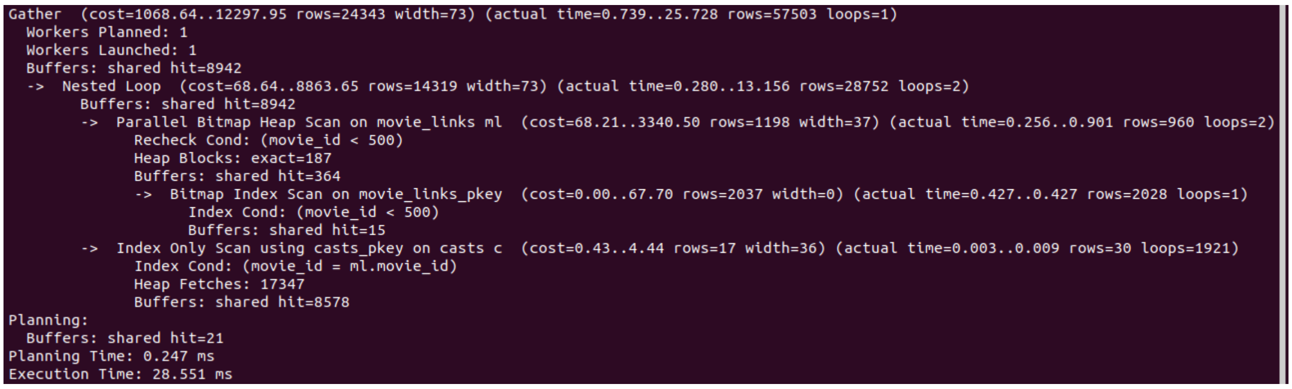
```
Gather  (cost=1068.64..12297.95 rows=24343 width=73) (actual time=2.252..45.726 rows=57503 loops=1)
  Workers Planned: 1
  Workers Launched: 1
  Buffers: shared hit=8942
  ->  Nested Loop  (cost=68.64..8863.65 rows=14319 width=73) (actual time=0.937..23.255 rows=28752 loops=2)
        Buffers: shared hit=8942
        ->  Parallel Bitmap Heap Scan on movie_links ml  (cost=68.21..3340.50 rows=1198 width=37) (actual time=0.839..1.944 rows=960 loops=2)
              Recheck Cond: (movie_id < 500)
              Heap Blocks: exact=194
              Buffers: shared hit=364
              ->  Bitmap Index Scan on movie_links_pkey  (cost=0.00..67.70 rows=2037 width=0) (actual time=1.408..1.408 rows=2028 loops=1)
                    Index Cond: (movie_id < 500)
                    Buffers: shared hit=15
        ->  Index Only Scan using casts_pkey on casts c  (cost=0.43..4.44 rows=17 width=36) (actual time=0.005..0.015 rows=30 loops=1921)
              Index Cond: (movie_id = ml.movie_id)
              Heap Fetches: 17347
              Buffers: shared hit=8578
Planning:
  Buffers: shared hit=21
Planning Time: 0.577 ms
Execution Time: 50.568 ms
```

Interação 2:



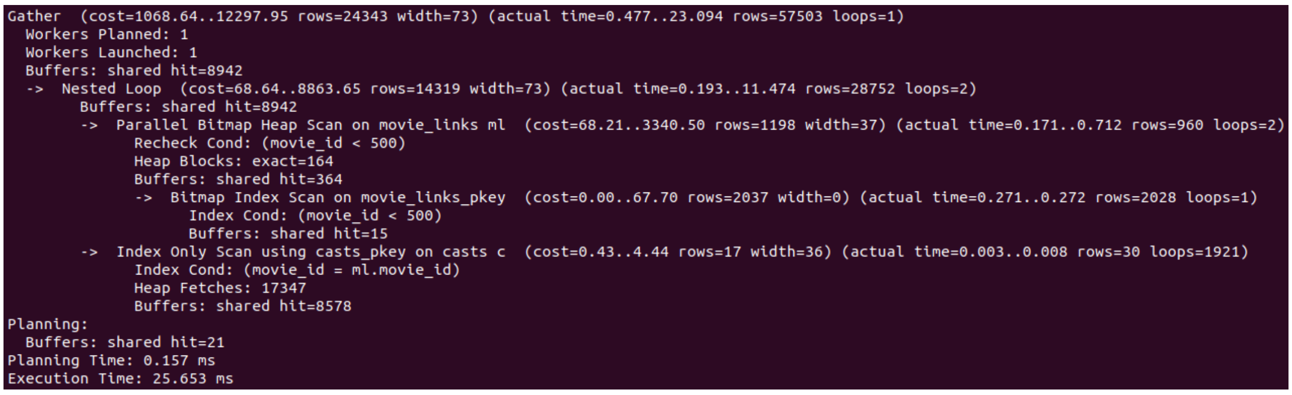
```
Gather  (cost=1068.64..12297.95 rows=24343 width=73) (actual time=0.798..28.565 rows=57503 loops=1)
  Workers Planned: 1
  Workers Launched: 1
  Buffers: shared hit=8942
  ->  Nested Loop  (cost=68.64..8863.65 rows=14319 width=73) (actual time=0.300..14.440 rows=28752 loops=2)
        Buffers: shared hit=8942
        ->  Parallel Bitmap Heap Scan on movie_links ml  (cost=68.21..3340.50 rows=1198 width=37) (actual time=0.269..0.939 rows=960 loops=2)
              Recheck Cond: (movie_id < 500)
              Heap Blocks: exact=179
              Buffers: shared hit=364
              ->  Bitmap Index Scan on movie_links_pkey  (cost=0.00..67.70 rows=2037 width=0) (actual time=0.435..0.436 rows=2028 loops=1)
                    Index Cond: (movie_id < 500)
                    Buffers: shared hit=15
        ->  Index Only Scan using casts_pkey on casts c  (cost=0.43..4.44 rows=17 width=36) (actual time=0.003..0.009 rows=30 loops=1921)
              Index Cond: (movie_id = ml.movie_id)
              Heap Fetches: 17347
              Buffers: shared hit=8578
Planning:
  Buffers: shared hit=21
Planning Time: 0.277 ms
Execution Time: 31.743 ms
```

Interação 3:



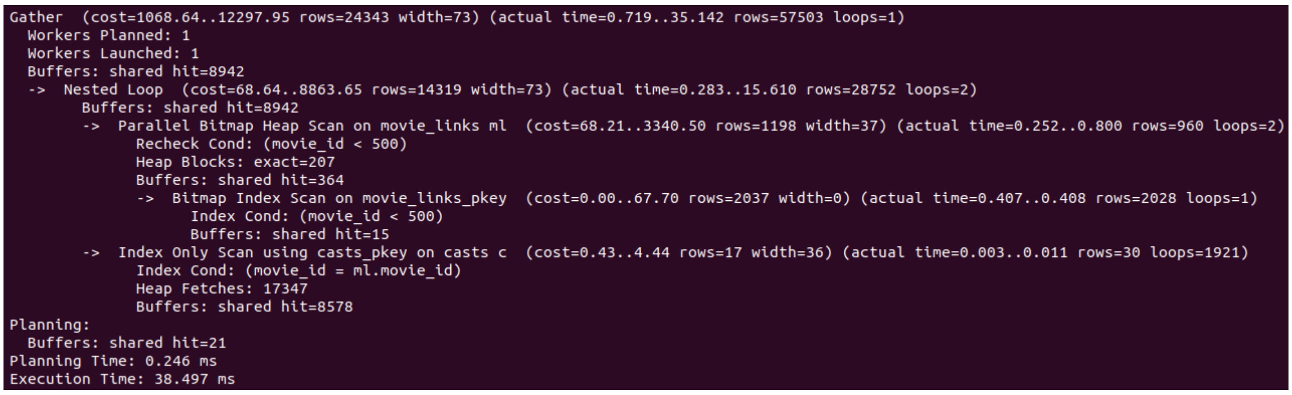
```
Gather  (cost=1068.64..12297.95 rows=24343 width=73) (actual time=0.739..25.728 rows=57503 loops=1)
  Workers Planned: 1
  Workers Launched: 1
  Buffers: shared hit=8942
  ->  Nested Loop  (cost=68.64..8863.65 rows=14319 width=73) (actual time=0.280..13.156 rows=28752 loops=2)
        Buffers: shared hit=8942
        ->  Parallel Bitmap Heap Scan on movie_links ml  (cost=68.21..3340.50 rows=1198 width=37) (actual time=0.256..0.901 rows=960 loops=2)
              Recheck Cond: (movie_id < 500)
              Heap Blocks: exact=187
              Buffers: shared hit=364
              ->  Bitmap Index Scan on movie_links_pkey  (cost=0.00..67.70 rows=2037 width=0) (actual time=0.427..0.427 rows=2028 loops=1)
                    Index Cond: (movie_id < 500)
                    Buffers: shared hit=15
        ->  Index Only Scan using casts_pkey on casts c  (cost=0.43..4.44 rows=17 width=36) (actual time=0.003..0.009 rows=30 loops=1921)
              Index Cond: (movie_id = ml.movie_id)
              Heap Fetches: 17347
              Buffers: shared hit=8578
Planning:
  Buffers: shared hit=21
Planning Time: 0.247 ms
Execution Time: 28.551 ms
```

Interação 4:



```
Gather  (cost=1068.64..12297.95 rows=24343 width=73) (actual time=0.477..23.094 rows=57503 loops=1)
  Workers Planned: 1
  Workers Launched: 1
  Buffers: shared hit=8942
  ->  Nested Loop  (cost=68.64..8863.65 rows=14319 width=73) (actual time=0.193..11.474 rows=28752 loops=2)
        Buffers: shared hit=8942
        ->  Parallel Bitmap Heap Scan on movie_links ml  (cost=68.21..3340.50 rows=1198 width=37) (actual time=0.171..0.712 rows=960 loops=2)
              Recheck Cond: (movie_id < 500)
              Heap Blocks: exact=164
              Buffers: shared hit=364
              ->  Bitmap Index Scan on movie_links_pkey  (cost=0.00..67.70 rows=2037 width=0) (actual time=0.271..0.272 rows=2028 loops=1)
                    Index Cond: (movie_id < 500)
                    Buffers: shared hit=15
        ->  Index Only Scan using casts_pkey on casts c  (cost=0.43..4.44 rows=17 width=36) (actual time=0.003..0.008 rows=30 loops=1921)
              Index Cond: (movie_id = ml.movie_id)
              Heap Fetches: 17347
              Buffers: shared hit=8578
Planning:
  Buffers: shared hit=21
Planning Time: 0.157 ms
Execution Time: 25.653 ms
```

Interação 5:

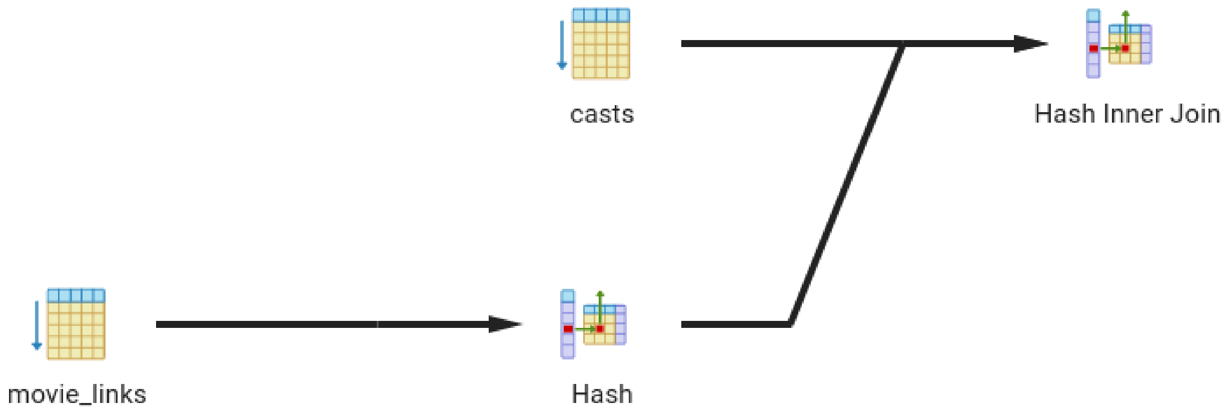


```
Gather  (cost=1068.64..12297.95 rows=24343 width=73) (actual time=0.719..35.142 rows=57503 loops=1)
  Workers Planned: 1
  Workers Launched: 1
  Buffers: shared hit=8942
  ->  Nested Loop  (cost=68.64..8863.65 rows=14319 width=73) (actual time=0.283..15.610 rows=28752 loops=2)
        Buffers: shared hit=8942
        ->  Parallel Bitmap Heap Scan on movie_links ml  (cost=68.21..3340.50 rows=1198 width=37) (actual time=0.252..0.800 rows=960 loops=2)
              Recheck Cond: (movie_id < 500)
              Heap Blocks: exact=207
              Buffers: shared hit=364
              ->  Bitmap Index Scan on movie_links_pkey  (cost=0.00..67.70 rows=2037 width=0) (actual time=0.407..0.408 rows=2028 loops=1)
                    Index Cond: (movie_id < 500)
                    Buffers: shared hit=15
        ->  Index Only Scan using casts_pkey on casts c  (cost=0.43..4.44 rows=17 width=36) (actual time=0.003..0.011 rows=30 loops=1921)
              Index Cond: (movie_id = ml.movie_id)
              Heap Fetches: 17347
              Buffers: shared hit=8578
Planning:
  Buffers: shared hit=21
Planning Time: 0.246 ms
Execution Time: 38.497 ms
```

Médias da interações:
Planning Time: 0.3008
Execution Time: 35.0024
Desvio padrão das amostras das interações:
Planning Time: 0.1608
Execution Time: 9.9266

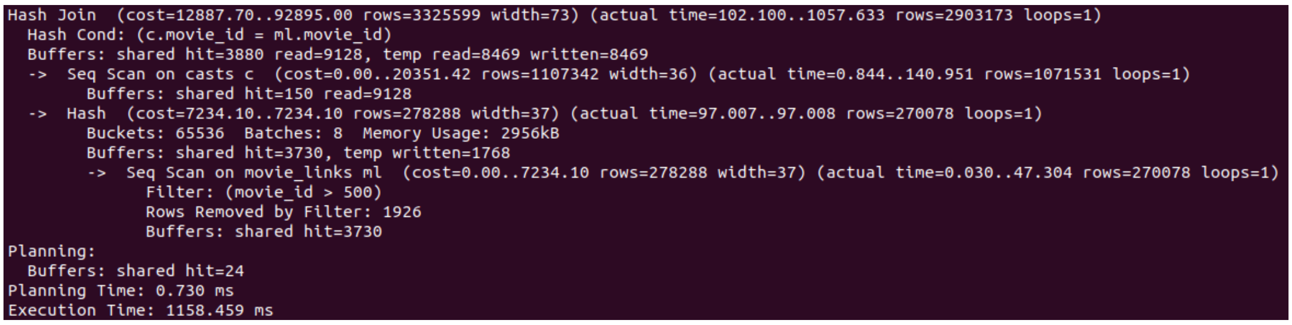
Consulta 2:

select * from casts as c, movie_links as ml where ml.movie_id=c.movie_id and ml.movie_id > 500;



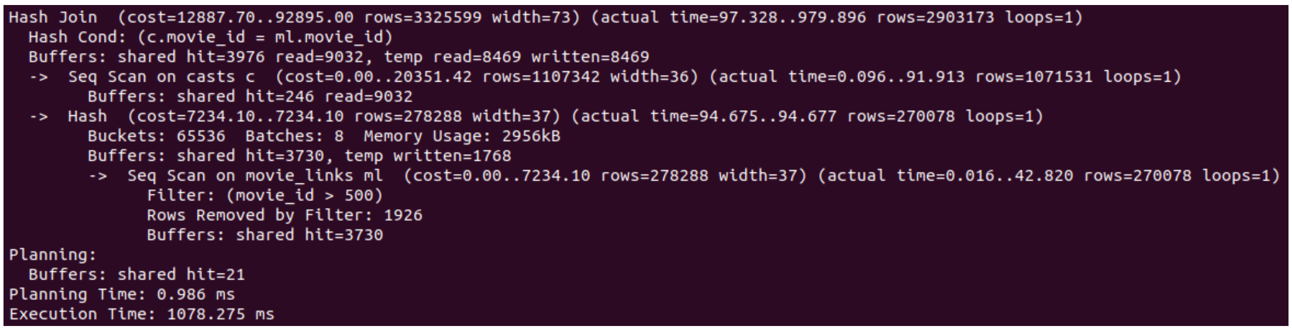
explain (buffers, costs, timing, analyze) select * from casts as c, movie_links as ml where ml.movie_id=c.movie_id and ml.movie_id < 500;

Interação 1:



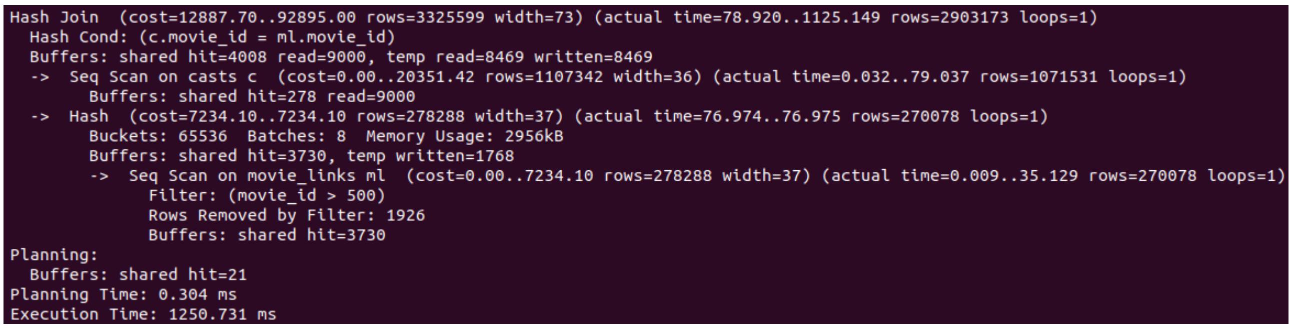
```
Hash Join  (cost=12887.70..92895.00 rows=3325599 width=73) (actual time=102.100..1057.633 rows=2903173 loops=1)
  Hash Cond: (c.movie_id = ml.movie_id)
  Buffers: shared hit=3880 read=9128, temp read=8469 written=8469
  ->  Seq Scan on casts c  (cost=0.00..20351.42 rows=1107342 width=36) (actual time=0.844..140.951 rows=1071531 loops=1)
        Buffers: shared hit=150 read=9128
  ->  Hash  (cost=7234.10..7234.10 rows=278288 width=37) (actual time=97.007..97.008 rows=270078 loops=1)
        Buckets: 65536  Batches: 8  Memory Usage: 2956kB
        Buffers: shared hit=3730, temp written=1768
        ->  Seq Scan on movie_links ml  (cost=0.00..7234.10 rows=278288 width=37) (actual time=0.030..47.304 rows=270078 loops=1)
              Filter: (movie_id > 500)
              Rows Removed by Filter: 1926
              Buffers: shared hit=3730
Planning:
  Buffers: shared hit=24
Planning Time: 0.730 ms
Execution Time: 1158.459 ms
```

Interação 2:



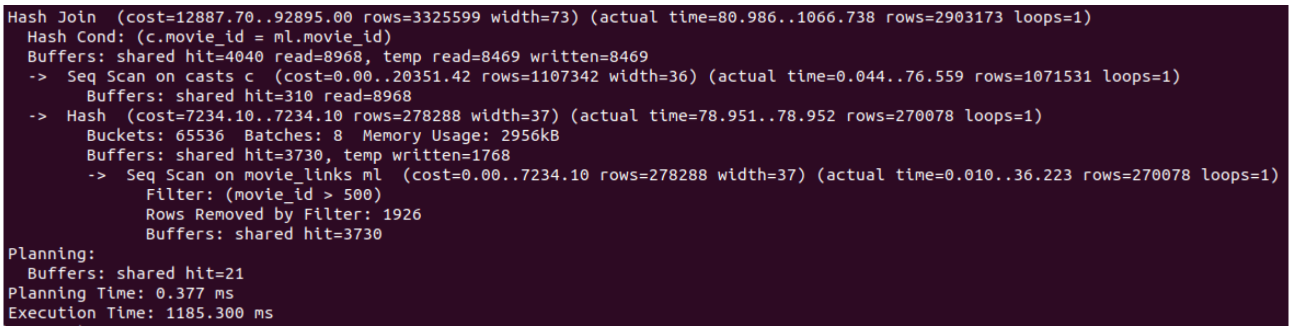
```
Hash Join  (cost=12887.70..92895.00 rows=3325599 width=73) (actual time=97.328..979.896 rows=2903173 loops=1)
  Hash Cond: (c.movie_id = ml.movie_id)
  Buffers: shared hit=3976 read=9032, temp read=8469 written=8469
  ->  Seq Scan on casts c  (cost=0.00..20351.42 rows=1107342 width=36) (actual time=0.096..91.913 rows=1071531 loops=1)
        Buffers: shared hit=246 read=9032
  ->  Hash  (cost=7234.10..7234.10 rows=278288 width=37) (actual time=94.675..94.677 rows=270078 loops=1)
        Buckets: 65536  Batches: 8  Memory Usage: 2956kB
        Buffers: shared hit=3730, temp written=1768
        ->  Seq Scan on movie_links ml  (cost=0.00..7234.10 rows=278288 width=37) (actual time=0.016..42.820 rows=270078 loops=1)
              Filter: (movie_id > 500)
              Rows Removed by Filter: 1926
              Buffers: shared hit=3730
Planning:
  Buffers: shared hit=21
Planning Time: 0.986 ms
Execution Time: 1078.275 ms
```

Interação 3:



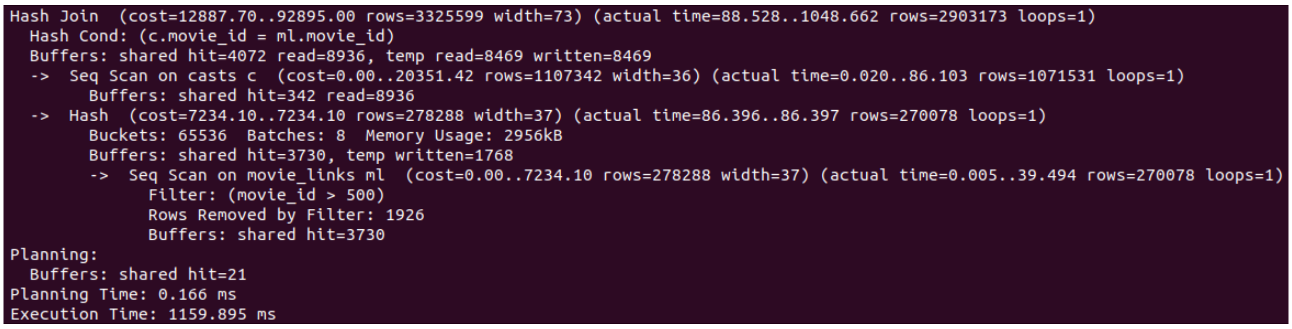
```
Hash Join  (cost=12887.70..92895.00 rows=3325599 width=73) (actual time=78.920..1125.149 rows=2903173 loops=1)
  Hash Cond: (c.movie_id = ml.movie_id)
  Buffers: shared hit=4008 read=9000, temp read=8469 written=8469
  ->  Seq Scan on casts c  (cost=0.00..20351.42 rows=1107342 width=36) (actual time=0.032..79.037 rows=1071531 loops=1)
        Buffers: shared hit=278 read=9000
  ->  Hash  (cost=7234.10..7234.10 rows=278288 width=37) (actual time=76.974..76.975 rows=270078 loops=1)
        Buckets: 65536  Batches: 8  Memory Usage: 2956kB
        Buffers: shared hit=3730, temp written=1768
        ->  Seq Scan on movie_links ml  (cost=0.00..7234.10 rows=278288 width=37) (actual time=0.009..35.129 rows=270078 loops=1)
              Filter: (movie_id > 500)
              Rows Removed by Filter: 1926
              Buffers: shared hit=3730
Planning:
  Buffers: shared hit=21
Planning Time: 0.304 ms
Execution Time: 1250.731 ms
```

Interação 4:



```
Hash Join  (cost=12887.70..92895.00 rows=3325599 width=73) (actual time=80.986..1066.738 rows=2903173 loops=1)
  Hash Cond: (c.movie_id = ml.movie_id)
  Buffers: shared hit=4040 read=8968, temp read=8469 written=8469
  ->  Seq Scan on casts c  (cost=0.00..20351.42 rows=1107342 width=36) (actual time=0.044..76.559 rows=1071531 loops=1)
        Buffers: shared hit=310 read=8968
  ->  Hash  (cost=7234.10..7234.10 rows=278288 width=37) (actual time=78.951..78.952 rows=270078 loops=1)
        Buckets: 65536  Batches: 8  Memory Usage: 2956kB
        Buffers: shared hit=3730, temp written=1768
        ->  Seq Scan on movie_links ml  (cost=0.00..7234.10 rows=278288 width=37) (actual time=0.010..36.223 rows=270078 loops=1)
              Filter: (movie_id > 500)
              Rows Removed by Filter: 1926
              Buffers: shared hit=3730
Planning:
  Buffers: shared hit=21
Planning Time: 0.377 ms
Execution Time: 1185.300 ms
```

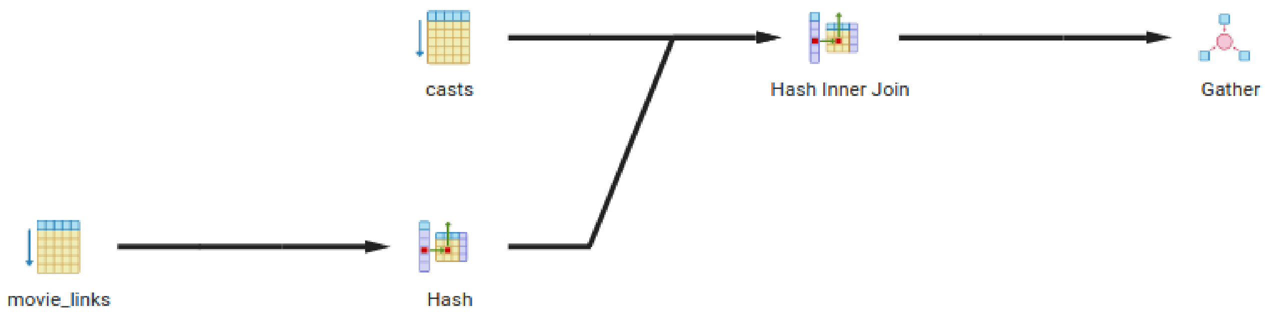
Interação 5:



```
Hash Join  (cost=12887.70..92895.00 rows=3325599 width=73) (actual time=88.528..1048.662 rows=2903173 loops=1)
  Hash Cond: (c.movie_id = ml.movie_id)
  Buffers: shared hit=4072 read=8936, temp read=8469 written=8469
  ->  Seq Scan on casts c  (cost=0.00..20351.42 rows=1107342 width=36) (actual time=0.020..86.103 rows=1071531 loops=1)
        Buffers: shared hit=342 read=8936
  ->  Hash  (cost=7234.10..7234.10 rows=278288 width=37) (actual time=86.396..86.397 rows=270078 loops=1)
        Buckets: 65536  Batches: 8  Memory Usage: 2956kB
        Buffers: shared hit=3730, temp written=1768
        ->  Seq Scan on movie_links ml  (cost=0.00..7234.10 rows=278288 width=37) (actual time=0.005..39.494 rows=270078 loops=1)
              Filter: (movie_id > 500)
              Rows Removed by Filter: 1926
              Buffers: shared hit=3730
Planning:
  Buffers: shared hit=21
Planning Time: 0.166 ms
Execution Time: 1159.895 ms
```

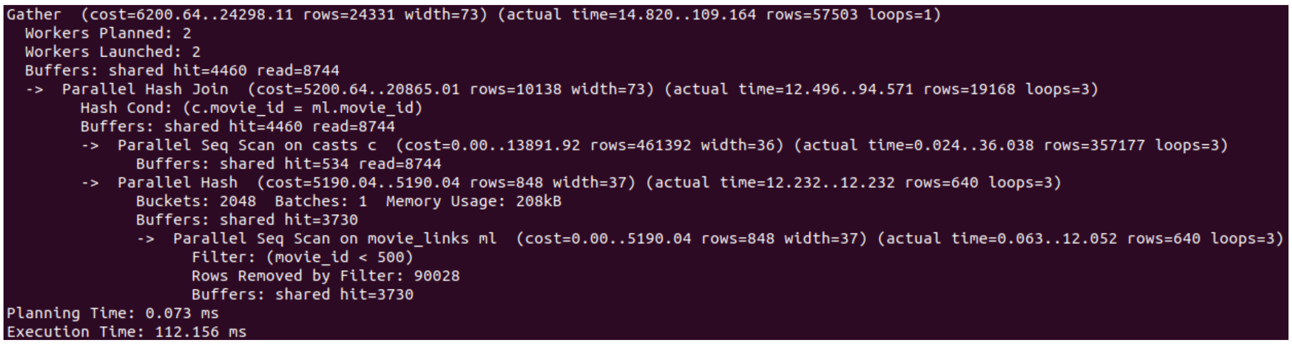
Médias da interações:
Planning Time: 0.5126
Execution Time: 1166.532
Desvio padrão das amostras das interações:
Planning Time: 0.3366
Execution Time: 61.9278

Removendo Índices para realizar as consultas.
Consulta 1:
select * from casts as c, movie_links as ml where ml.movie_id=c.movie_id and ml.movie_id < 500;



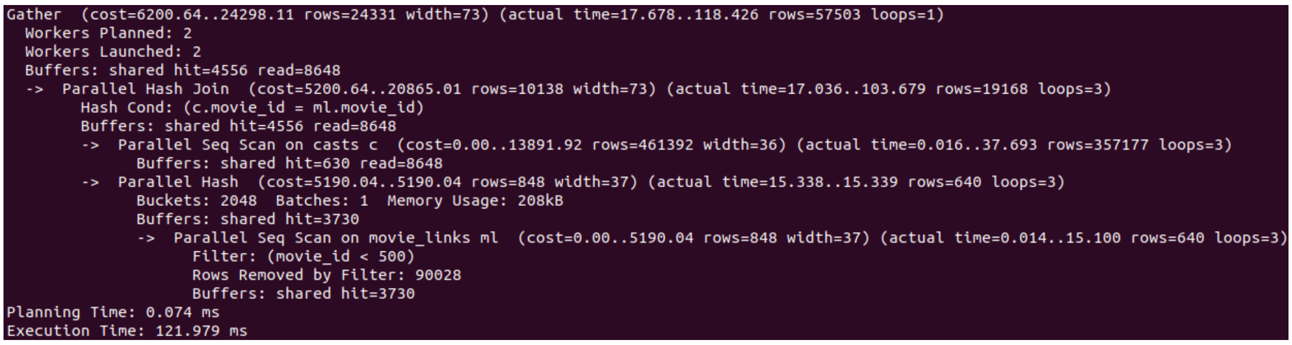
explain (buffers, costs, timing, analyze) select * from casts as c, movie_links as ml where ml.movie_id=c.movie_id and ml.movie_id < 500;

Interação 1:



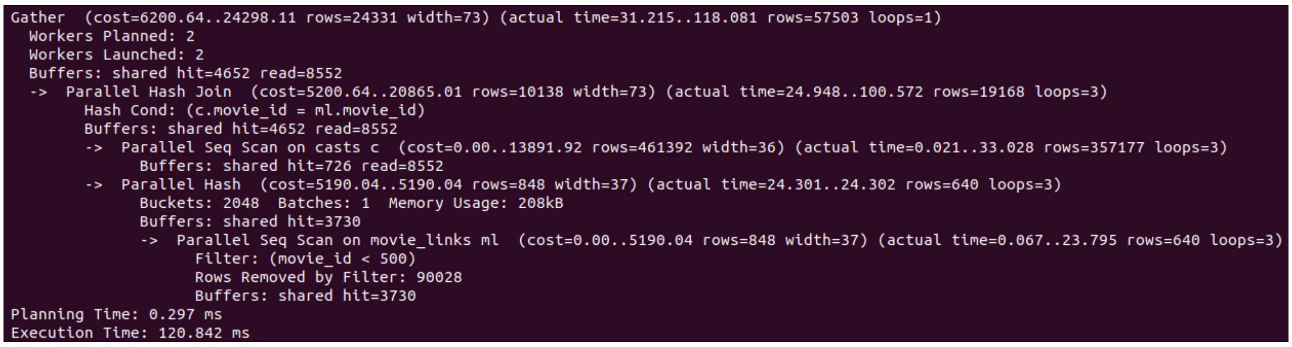
```
Gather  (cost=6200.64..24298.11 rows=24331 width=73) (actual time=14.820..109.164 rows=57503 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  Buffers: shared hit=4460 read=8744
  ->  Parallel Hash Join  (cost=5200.64..20865.01 rows=10138 width=73) (actual time=12.496..94.571 rows=19168 loops=3)
        Hash Cond: (c.movie_id = ml.movie_id)
        Buffers: shared hit=4460 read=8744
        ->  Parallel Seq Scan on casts c  (cost=0.00..13891.92 rows=461392 width=36) (actual time=0.024..36.038 rows=357177 loops=3)
              Buffers: shared hit=534 read=8744
        ->  Parallel Hash  (cost=5190.04..5190.04 rows=848 width=37) (actual time=12.232..12.232 rows=640 loops=3)
              Buckets: 2048  Batches: 1  Memory Usage: 208kB
              Buffers: shared hit=3730
              ->  Parallel Seq Scan on movie_links ml  (cost=0.00..5190.04 rows=848 width=37) (actual time=0.063..12.052 rows=640 loops=3)
                    Filter: (movie_id < 500)
                    Rows Removed by Filter: 90028
                    Buffers: shared hit=3730
Planning Time: 0.073 ms
Execution Time: 112.156 ms
```

Interação 2:



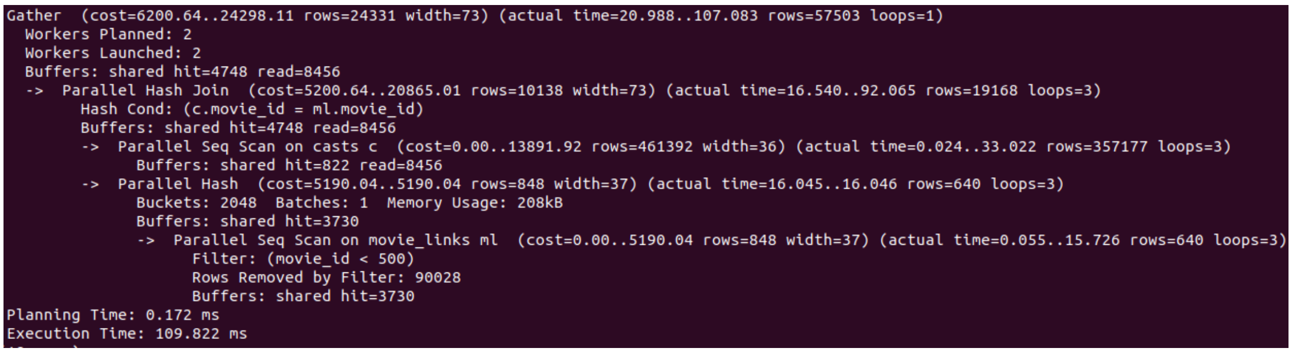
```
Gather  (cost=6200.64..24298.11 rows=24331 width=73) (actual time=17.678..118.426 rows=57503 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  Buffers: shared hit=4556 read=8648
  ->  Parallel Hash Join  (cost=5200.64..20865.01 rows=10138 width=73) (actual time=17.036..103.679 rows=19168 loops=3)
        Hash Cond: (c.movie_id = ml.movie_id)
        Buffers: shared hit=4556 read=8648
        ->  Parallel Seq Scan on casts c  (cost=0.00..13891.92 rows=461392 width=36) (actual time=0.016..37.693 rows=357177 loops=3)
              Buffers: shared hit=630 read=8648
        ->  Parallel Hash  (cost=5190.04..5190.04 rows=848 width=37) (actual time=15.338..15.339 rows=640 loops=3)
              Buckets: 2048  Batches: 1  Memory Usage: 208kB
              Buffers: shared hit=3730
              ->  Parallel Seq Scan on movie_links ml  (cost=0.00..5190.04 rows=848 width=37) (actual time=0.014..15.100 rows=640 loops=3)
                    Filter: (movie_id < 500)
                    Rows Removed by Filter: 90028
                    Buffers: shared hit=3730
Planning Time: 0.074 ms
Execution Time: 121.979 ms
```

Interação 3:



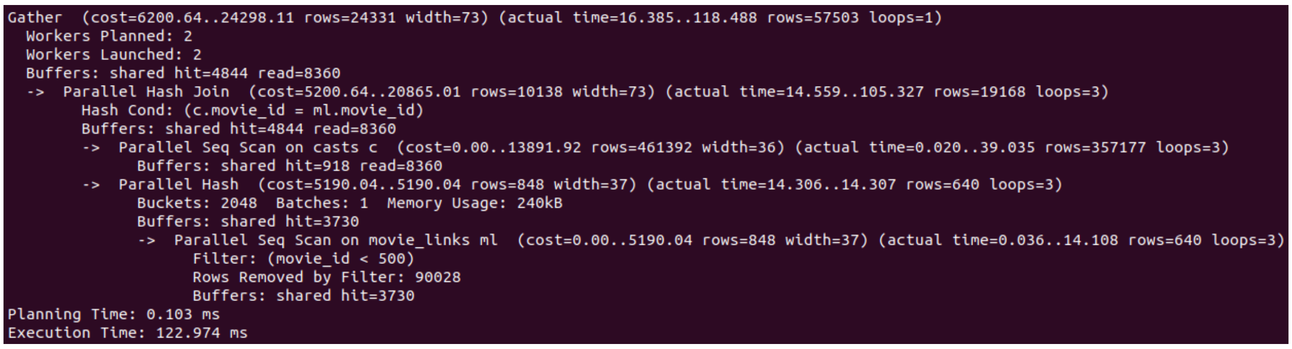
```
Gather  (cost=6200.64..24298.11 rows=24331 width=73) (actual time=31.215..118.081 rows=57503 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  Buffers: shared hit=4652 read=8552
  ->  Parallel Hash Join  (cost=5200.64..20865.01 rows=10138 width=73) (actual time=24.948..100.572 rows=19168 loops=3)
        Hash Cond: (c.movie_id = ml.movie_id)
        Buffers: shared hit=4652 read=8552
        ->  Parallel Seq Scan on casts c  (cost=0.00..13891.92 rows=461392 width=36) (actual time=0.021..33.028 rows=357177 loops=3)
              Buffers: shared hit=726 read=8552
        ->  Parallel Hash  (cost=5190.04..5190.04 rows=848 width=37) (actual time=24.301..24.302 rows=640 loops=3)
              Buckets: 2048  Batches: 1  Memory Usage: 208kB
              Buffers: shared hit=3730
              ->  Parallel Seq Scan on movie_links ml  (cost=0.00..5190.04 rows=848 width=37) (actual time=0.067..23.795 rows=640 loops=3)
                    Filter: (movie_id < 500)
                    Rows Removed by Filter: 90028
                    Buffers: shared hit=3730
Planning Time: 0.297 ms
Execution Time: 120.842 ms
```

Interação 4:



```
Gather  (cost=6200.64..24298.11 rows=24331 width=73) (actual time=20.988..107.083 rows=57503 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  Buffers: shared hit=4748 read=8456
  ->  Parallel Hash Join  (cost=5200.64..20865.01 rows=10138 width=73) (actual time=16.540..92.065 rows=19168 loops=3)
        Hash Cond: (c.movie_id = ml.movie_id)
        Buffers: shared hit=4748 read=8456
        ->  Parallel Seq Scan on casts c  (cost=0.00..13891.92 rows=461392 width=36) (actual time=0.024..33.022 rows=357177 loops=3)
              Buffers: shared hit=822 read=8456
        ->  Parallel Hash  (cost=5190.04..5190.04 rows=848 width=37) (actual time=16.045..16.046 rows=640 loops=3)
              Buckets: 2048  Batches: 1  Memory Usage: 208kB
              Buffers: shared hit=3730
              ->  Parallel Seq Scan on movie_links ml  (cost=0.00..5190.04 rows=848 width=37) (actual time=0.055..15.726 rows=640 loops=3)
                    Filter: (movie_id < 500)
                    Rows Removed by Filter: 90028
                    Buffers: shared hit=3730
Planning Time: 0.172 ms
Execution Time: 109.822 ms
```

Interação 5:



```
Gather  (cost=6200.64..24298.11 rows=24331 width=73) (actual time=16.385..118.488 rows=57503 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  Buffers: shared hit=4844 read=8360
  ->  Parallel Hash Join  (cost=5200.64..20865.01 rows=10138 width=73) (actual time=14.559..105.327 rows=19168 loops=3)
        Hash Cond: (c.movie_id = ml.movie_id)
        Buffers: shared hit=4844 read=8360
        ->  Parallel Seq Scan on casts c  (cost=0.00..13891.92 rows=461392 width=36) (actual time=0.020..39.035 rows=357177 loops=3)
              Buffers: shared hit=918 read=8360
        ->  Parallel Hash  (cost=5190.04..5190.04 rows=848 width=37) (actual time=14.306..14.307 rows=640 loops=3)
              Buckets: 2048  Batches: 1  Memory Usage: 240kB
              Buffers: shared hit=3730
              ->  Parallel Seq Scan on movie_links ml  (cost=0.00..5190.04 rows=848 width=37) (actual time=0.036..14.108 rows=640 loops=3)
                    Filter: (movie_id < 500)
                    Rows Removed by Filter: 90028
                    Buffers: shared hit=3730
Planning Time: 0.103 ms
Execution Time: 122.974 ms
```

Médias da interações:

Planning Time: 0.1438

Execution Time: 117.5546

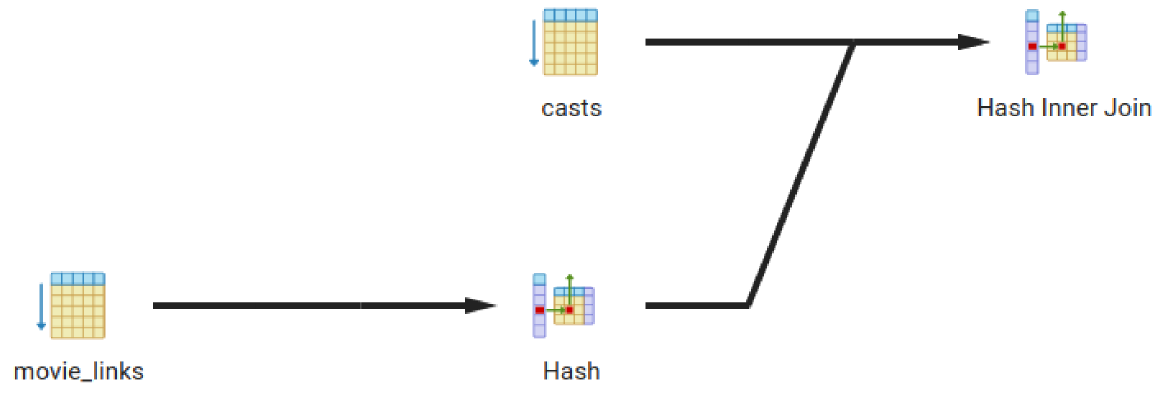
Desvio padrão das amostras das interações:

Planning Time: 0.0946

Execution Time: 6.0969

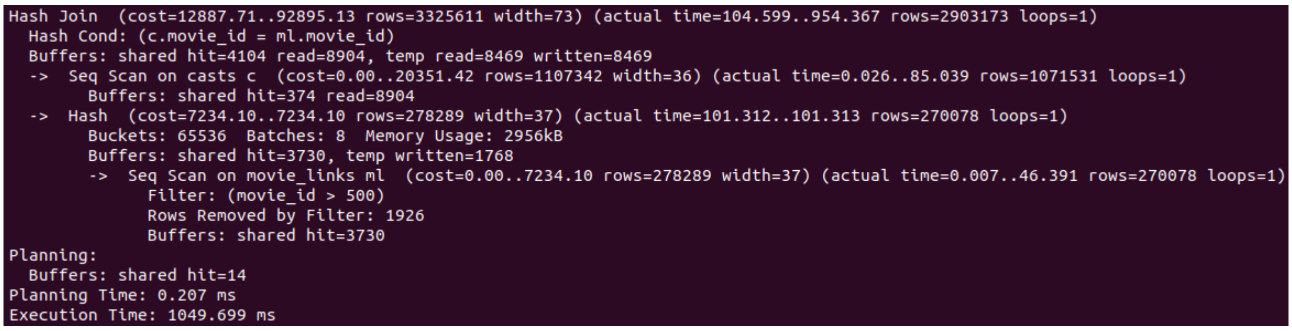
Consulta 2:

select * from casts as c, movie_links as ml where ml.movie_id=c.movie_id and ml.movie_id > 500;



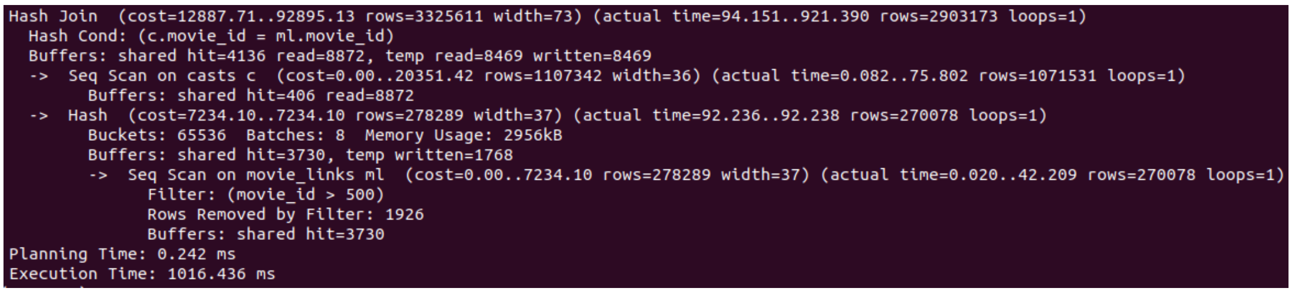
explain (buffers, costs, timing, analyze) select * from casts as c, movie_links as ml where ml.movie_id=c.movie_id and ml.movie_id > 500;

Interação 1:



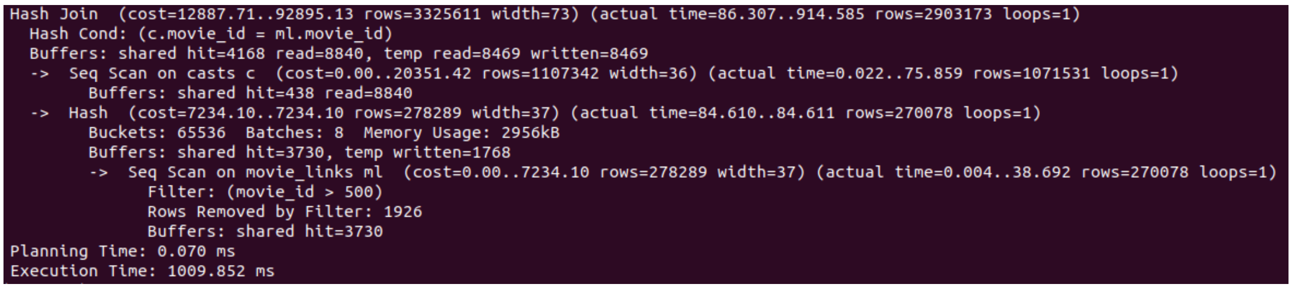
```
Hash Join  (cost=12887.71..92895.13 rows=3325611 width=73) (actual time=104.599..954.367 rows=2903173 loops=1)
  Hash Cond: (c.movie_id = ml.movie_id)
  Buffers: shared hit=4104 read=8904, temp read=8469 written=8469
  ->  Seq Scan on casts c  (cost=0.00..20351.42 rows=1107342 width=36) (actual time=0.026..85.039 rows=1071531 loops=1)
        Buffers: shared hit=374 read=8904
  ->  Hash  (cost=7234.10..7234.10 rows=278289 width=37) (actual time=101.312..101.313 rows=270078 loops=1)
        Buckets: 65536  Batches: 8  Memory Usage: 2956kB
        Buffers: shared hit=3730, temp written=1768
        ->  Seq Scan on movie_links ml  (cost=0.00..7234.10 rows=278289 width=37) (actual time=0.007..46.391 rows=270078 loops=1)
              Filter: (movie_id > 500)
              Rows Removed by Filter: 1926
              Buffers: shared hit=3730
Planning:
  Buffers: shared hit=14
Planning Time: 0.207 ms
Execution Time: 1049.699 ms
```

Interação 2:



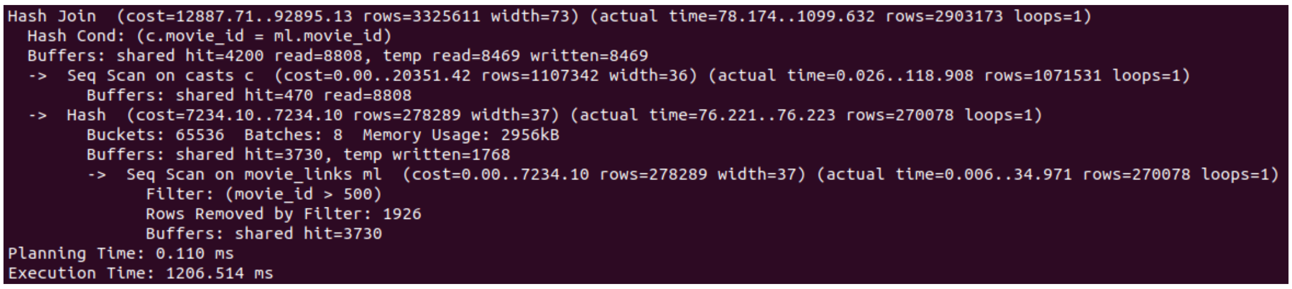
```
Hash Join  (cost=12887.71..92895.13 rows=3325611 width=73) (actual time=94.151..921.390 rows=2903173 loops=1)
  Hash Cond: (c.movie_id = ml.movie_id)
  Buffers: shared hit=4136 read=8872, temp read=8469 written=8469
  ->  Seq Scan on casts c  (cost=0.00..20351.42 rows=1107342 width=36) (actual time=0.082..75.802 rows=1071531 loops=1)
        Buffers: shared hit=406 read=8872
  ->  Hash  (cost=7234.10..7234.10 rows=278289 width=37) (actual time=92.236..92.238 rows=270078 loops=1)
        Buckets: 65536  Batches: 8  Memory Usage: 2956kB
        Buffers: shared hit=3730, temp written=1768
        ->  Seq Scan on movie_links ml  (cost=0.00..7234.10 rows=278289 width=37) (actual time=0.020..42.209 rows=270078 loops=1)
              Filter: (movie_id > 500)
              Rows Removed by Filter: 1926
              Buffers: shared hit=3730
Planning Time: 0.242 ms
Execution Time: 1016.436 ms
```

Interação 3:



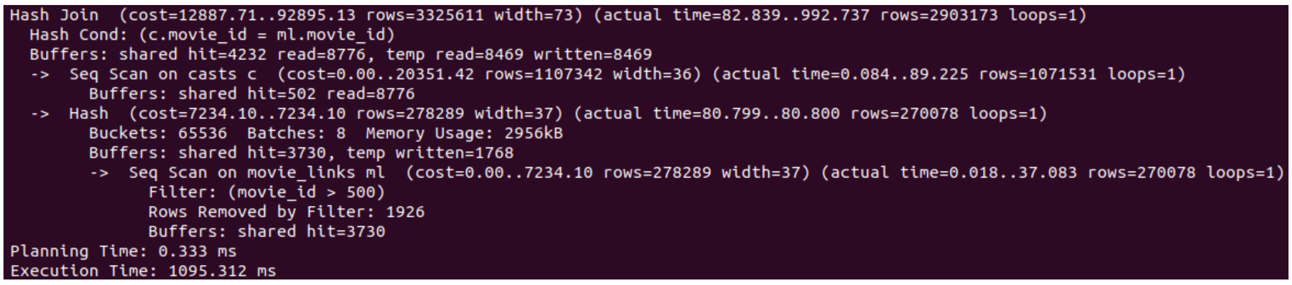
```
Hash Join  (cost=12887.71..92895.13 rows=3325611 width=73) (actual time=86.307..914.585 rows=2903173 loops=1)
  Hash Cond: (c.movie_id = ml.movie_id)
  Buffers: shared hit=4168 read=8840, temp read=8469 written=8469
  ->  Seq Scan on casts c  (cost=0.00..20351.42 rows=1107342 width=36) (actual time=0.022..75.859 rows=1071531 loops=1)
        Buffers: shared hit=438 read=8840
  ->  Hash  (cost=7234.10..7234.10 rows=278289 width=37) (actual time=84.610..84.611 rows=270078 loops=1)
        Buckets: 65536  Batches: 8  Memory Usage: 2956kB
        Buffers: shared hit=3730, temp written=1768
        ->  Seq Scan on movie_links ml  (cost=0.00..7234.10 rows=278289 width=37) (actual time=0.004..38.692 rows=270078 loops=1)
              Filter: (movie_id > 500)
              Rows Removed by Filter: 1926
              Buffers: shared hit=3730
Planning Time: 0.070 ms
Execution Time: 1009.852 ms
```

Interação 4:



```
Hash Join  (cost=12887.71..92895.13 rows=3325611 width=73) (actual time=78.174..1099.632 rows=2903173 loops=1)
  Hash Cond: (c.movie_id = ml.movie_id)
  Buffers: shared hit=4200 read=8808, temp read=8469 written=8469
  ->  Seq Scan on casts c  (cost=0.00..20351.42 rows=1107342 width=36) (actual time=0.026..118.908 rows=1071531 loops=1)
        Buffers: shared hit=470 read=8808
  ->  Hash  (cost=7234.10..7234.10 rows=278289 width=37) (actual time=76.221..76.223 rows=270078 loops=1)
        Buckets: 65536  Batches: 8  Memory Usage: 2956kB
        Buffers: shared hit=3730, temp written=1768
        ->  Seq Scan on movie_links ml  (cost=0.00..7234.10 rows=278289 width=37) (actual time=0.006..34.971 rows=270078 loops=1)
              Filter: (movie_id > 500)
              Rows Removed by Filter: 1926
              Buffers: shared hit=3730
Planning Time: 0.110 ms
Execution Time: 1206.514 ms
```

Interação 5:



```
Hash Join  (cost=12887.71..92895.13 rows=3325611 width=73) (actual time=82.839..992.737 rows=2903173 loops=1)
  Hash Cond: (c.movie_id = ml.movie_id)
  Buffers: shared hit=4232 read=8776, temp read=8469 written=8469
  ->  Seq Scan on casts c  (cost=0.00..20351.42 rows=1107342 width=36) (actual time=0.084..89.225 rows=1071531 loops=1)
        Buffers: shared hit=502 read=8776
  ->  Hash  (cost=7234.10..7234.10 rows=278289 width=37) (actual time=80.799..80.800 rows=270078 loops=1)
        Buckets: 65536  Batches: 8  Memory Usage: 2956kB
        Buffers: shared hit=3730, temp written=1768
        ->  Seq Scan on movie_links ml  (cost=0.00..7234.10 rows=278289 width=37) (actual time=0.018..37.083 rows=270078 loops=1)
              Filter: (movie_id > 500)
              Rows Removed by Filter: 1926
              Buffers: shared hit=3730
Planning Time: 0.333 ms
Execution Time: 1095.312 ms
```

O **Hash Join** é um algoritmo de junção de tabelas em banco de dados que funciona criando uma tabela hash a partir de uma das tabelas envolvidas na junção. Em seguida, o algoritmo compara os valores da outra tabela com a tabela hash, procurando por correspondências. Quando uma correspondência é encontrada, os dados das duas tabelas são combinados em uma única linha na saída da consulta. Esse algoritmo é eficiente para junções de grandes tabelas, especialmente quando há uma correspondência entre os valores de uma das tabelas e a chave da tabela hash. Enquanto que o **Hash Inner Join** é uma variação, que retorna apenas as linhas que correspondem em ambas as tabelas.

Médias da interações:
Planning Time: 0.1924

Execution Time: 1075.5626
Desvio padrão das amostras das interações:
Planning Time: 0.1051
Execution Time: 80.6552

Tabela geral das médias e desvios.

	Consulta 01		Consulta 02	
	Com index	Sem index	Com index	Sem index
Média planejamento	0.3008	0.1438	0.5126	0.1924
Desvio Planejamento	0.1608	0.0946	0.3366	0.1051
Média Execução	35.0024	117.5546	1166.532	1075.5626
Desvio Execução	9.9266	6.0969	61.9278	80.6552

De acordo com as nossas consultas feitas percebemos que o tempo de planejamento é maior do que a consulta realizada sem índice, o desvio de planejamento é maior comparando com a consulta sem índice. o desvio de execução ficou maior inclusive.

Na consulta analisando os dados da tabela acima, podemos afirmar que a média de tempo de planejamento da consulta com índice ficou maior do que aquela sem indice o desvio de planejamento também é maior, a média de execução da consulta com índice ficou maior, porém o desvio de execução está menor comparando com a consulta 1. Quando tem indices o SGBD precisa mais tempo para organizar os dados.

Vamos definir cada algoritmo que se usa em cada consulta:

explain (buffers, costs, timing, analyze) select * from movie_links as ml, casts as c where ml.movie_id=c.movie_id and ml.movie_id < 500;

Nessa **primeira** consulta com índice, o SGBD optou por usar o Bitmap Index Scan, Bitmap Heap Scan, Nested Loop, Gather. Enquanto que na sem índice usou o Seq Scan, Hash e Gather.

Para segunda consulta respectivamentes mudando agora o comparador:
explain (buffers, costs, timing, analyze) select * from casts as c, movie_links as ml where ml.movie_id=c.movie_id and ml.movie_id < 500;

Com o índice utilizou o Seq Scan e funções Hash e já sem o índice novamente usou os Seq Scan e funções hash

Algoritmos:

O **Bitmap Index Scan** é um algoritmo utilizado em bancos de dados para realizar buscas eficientes em grandes quantidades de dados. Ele funciona criando um mapa de bits (bitmap) que representa a relação entre os dados e os índices, permitindo realizar a busca de forma mais rápida e eficiente. Esse algoritmo é frequentemente utilizado em conjunto com outros algoritmos, como o Hash Join e o Nested Loop, para realizar consultas em bancos de dados de forma mais rápida e eficiente.

O **Bitmap Heap Scan** é uma estratégia de consulta de banco de dados que utiliza índices bitmap para recuperar dados de uma tabela. A estratégia consiste em criar um mapa de bits que representa as linhas da tabela que satisfazem uma determinada condição, e em seguida, acessar diretamente essas linhas na tabela. Este algoritmo é eficiente em situações em que há muitos valores nulos ou duplicados na tabela, pois permite identificar rapidamente quais linhas precisam ser retornadas para a consulta.

O **Nested Loop** é um algoritmo de junção utilizado em bancos de dados para combinar linhas de duas ou mais tabelas. Ele funciona testando cada linha de uma tabela com todas as linhas da outra tabela e retornando as combinações que satisfazem a condição de junção. Este algoritmo pode ser eficiente quando há poucas linhas na tabela menor, mas se torna ineficiente à medida que o tamanho das tabelas aumenta.

O "**Gather**" é um algoritmo de junção utilizado em bancos de dados para coletar e reunir dados de várias fontes diferentes em uma única tabela ou conjunto de resultados. É usado principalmente em consultas que requerem a combinação de dados de várias tabelas em uma única tabela de saída. O algoritmo "Gather" funciona basicamente agrupando os dados das fontes e juntando-os em uma única tabela, que é usada para produzir o resultado final da consulta.

O **Seq Scan** (Sequential Scan) é um algoritmo utilizado em banco de dados para realizar uma varredura sequencial da tabela inteira. Neste processo, o banco de dados lê todas as linhas da tabela, uma a uma, para encontrar as linhas que atendem aos critérios da consulta. É uma opção de escaneamento mais simples e menos eficiente, mas pode ser apropriado para tabelas pequenas ou para consultas que retornam uma grande porcentagem das linhas da tabela.