Advanced query:

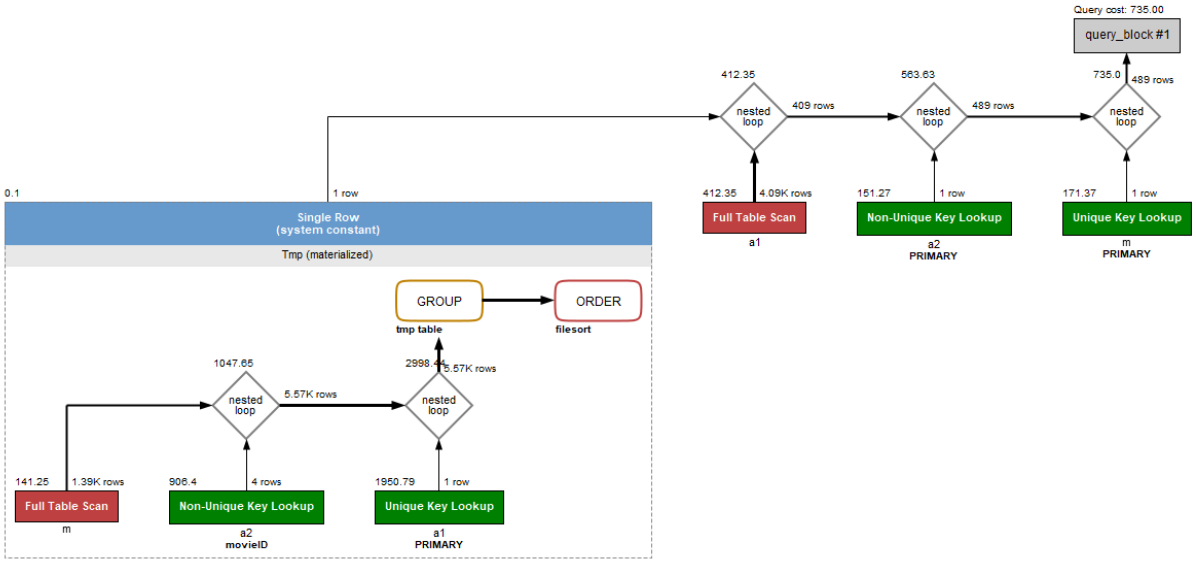
Find the average score for all movies from rottentomatoes with actors born in which year was the highest, and the number of movies is not less than 5. List actorID, actorName and movieName.

```
select a1.actorID, a1.name as actorName, m.name as movieName
from Actor a1 natural join Act a2 join Movie m on (a2.movieid = m.movieid) natural join(
   select birthyear, avg(ratingfromtomato) avg_rating, count(m.movieid) num_movie
      from Actor a1 natural join Act a2 join Movie m on (a2.movieid = m.movieid)
      group by birthyear
   having num_movie >= 5
   order by avg_rating desc
   limit 1) Tmp
where avg_rating is not null
```

Result:

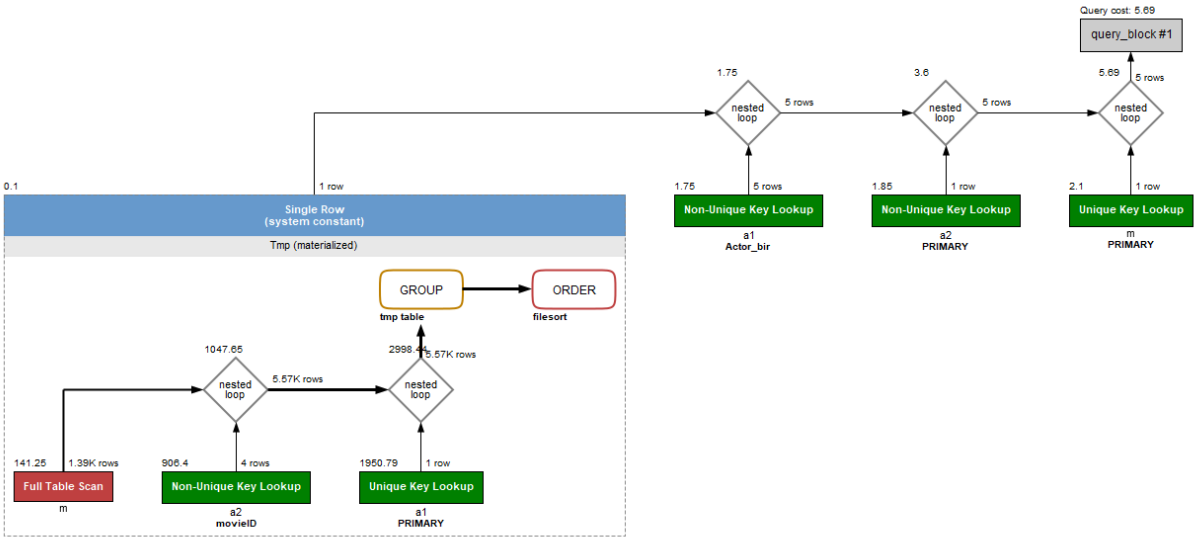
actorID	actorName	movieName
3582	Quvenzhané Wallis	Kahlil Gibran's The Prophet
3757	Jaeden Martell	The Confirmation
3789	Jack Dylan Grazer	Beautiful Boy
3844	Brock Brazda	Panic Room
3962	Anna Cathcart	To All the Boys: P.S. I Still Love You

Original:



0.047 sec / 0.000 sec
-> Nested loop inner join (cost=735.00 rows=490) (actual time=1.053..1.218 rows=5 loops=1)
  -> Nested loop inner join (cost=563.63 rows=490) (actual time=1.048..1.204 rows=5 loops=1)
    -> Filter: (a1.birthYear = '2003') (cost=412.35 rows=409) (actual time=1.037..1.181 rows=5 loops=1)
      -> Table scan on a1 (cost=412.35 rows=4091) (actual time=0.030..0.973 rows=4091 loops=1)
    -> Index lookup on a2 using PRIMARY (actorID=a1.actorID) (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loops=5)
  -> Single-row index lookup on m using PRIMARY (movieID=a2.movieID) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=5)

Index: Actor.birthyear

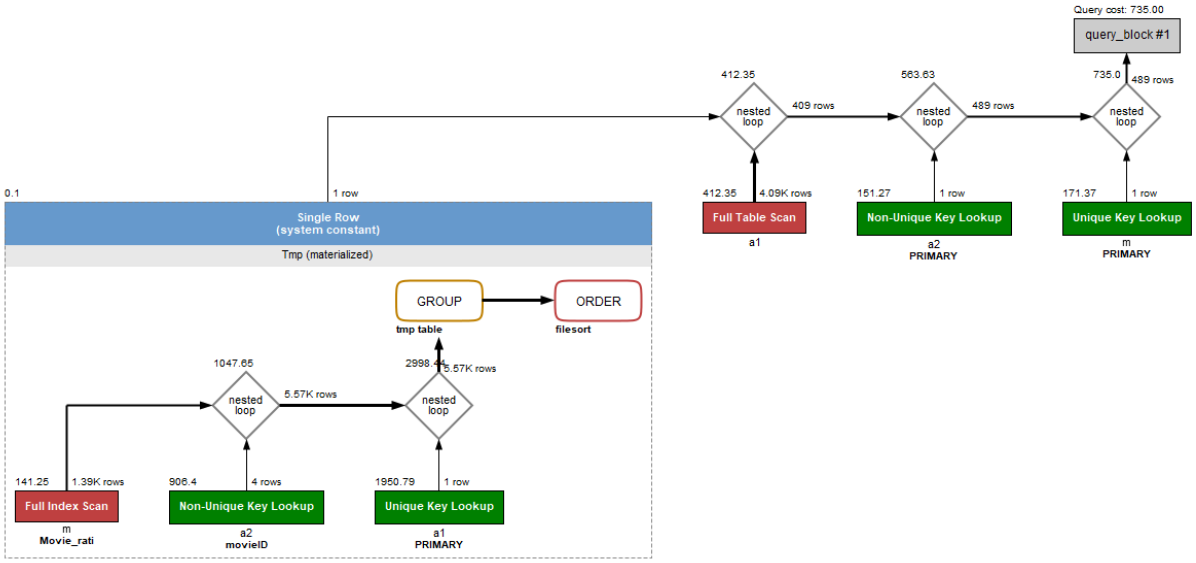


Query cost: 5.69
0.047 sec / 0.000 sec

-> Nested loop inner join (cost=5.69 rows=6) (actual time=0.030..0.052 rows=5 loops=1)
   -> Nested loop inner join (cost=3.60 rows=6) (actual time=0.024..0.037 rows=5 loops=1)
      -> Index lookup on a1 using Actor_bir (birthYear='2003') (cost=1.75 rows=5) (actual time=0.015..0.016 rows=5 loops=1)
      -> Index lookup on a2 using PRIMARY (actorID=a1.actorID) (cost=0.27 rows=1) (actual time=0.004..0.004 rows=1 loops=5)
   -> Single-row index lookup on m using PRIMARY (movieID=a2.movieID) (cost=0.27 rows=1) (actual time=0.003..0.003 rows=1 loops=5)

Why choose: the select query has used Actor.BirthYear in group by function, indexing can always shorten the time to group by, so we choose this column.
effect: This index helps this query reduce the time in Actor table to get the actors who birthyear is 2003.

Index: Movie.ratingfromtomato

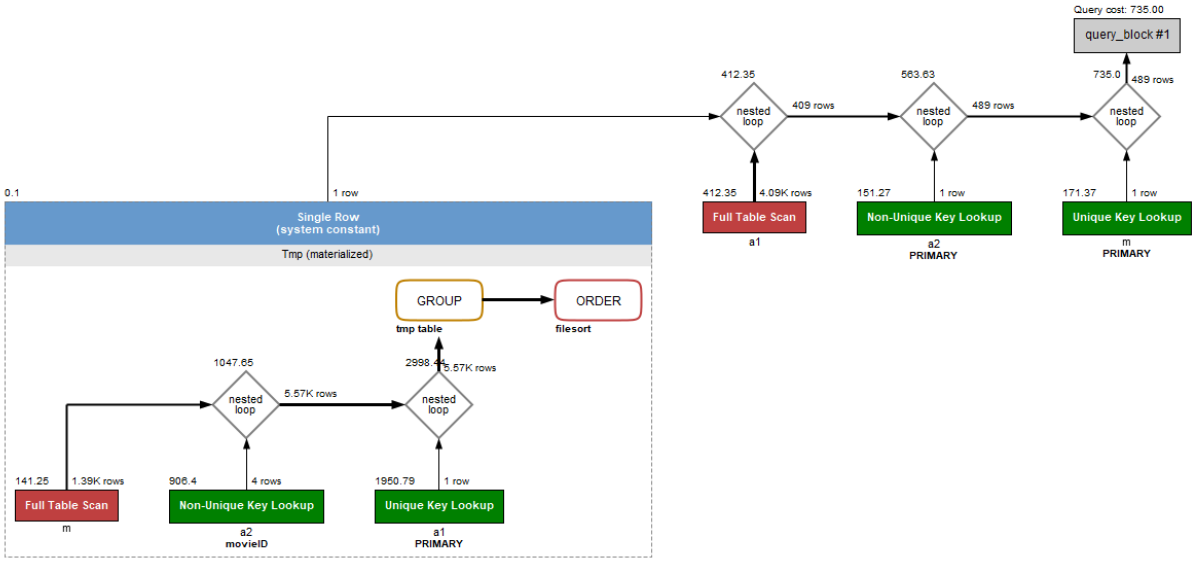


0.046 sec / 0.000 sec

-> Nested loop inner join (cost=735.00 rows=490) (actual time=1.055..1.212 rows=5 loops=1)
   -> Nested loop inner join (cost=563.63 rows=490) (actual time=1.050..1.199 rows=5 loops=1)
      -> Filter: (a1.birthYear = '2003') (cost=412.35 rows=409) (actual time=1.042..1.179 rows=5 loops=1)
         -> Table scan on a1 (cost=412.35 rows=4091) (actual time=0.026..0.948 rows=4091 loops=1)
      -> Index lookup on a2 using PRIMARY (actorID=a1.actorID) (cost=0.25 rows=1) (actual time=0.003..0.004 rows=1 loops=5)
   -> Single-row index lookup on m using PRIMARY (movieID=a2.movieID) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=5)

Why choose: Movie. Ratingfromtomato is used in aggregate function.
Why no effect: Index are helpful when we need to find just a few rows, but we still need all rows in max function, so this index does not help.

Index: Actor.name



0.047 sec / 0.000 sec

-> Nested loop inner join (cost=735.00 rows=490) (actual time=1.251..1.806 rows=5 loops=1)
  -> Nested loop inner join (cost=563.63 rows=490) (actual time=1.198..1.598 rows=5 loops=1)
    -> Filter: (a1.birthYear = '2003') (cost=412.35 rows=409) (actual time=1.057..1.215 rows=5 loops=1)
      -> Table scan on a1 (cost=412.35 rows=4091) (actual time=0.031..0.990 rows=4091 loops=1)
    -> Index lookup on a2 using PRIMARY (actorID=a1.actorID) (cost=0.25 rows=1) (actual time=0.075..0.076 rows=1 loops=5)
  -> Single-row index lookup on m using PRIMARY (movieID=a2.movieID) (cost=0.25 rows=1) (actual time=0.041..0.041 rows=1 loops=5)

Why choose: Actor.name needs to be looked up in the last step.
Why no effect: we still need to go through all rows to extract the name of actors, so this index does not help.

Therefore, we decide to use default index for this query.

Advanced query:

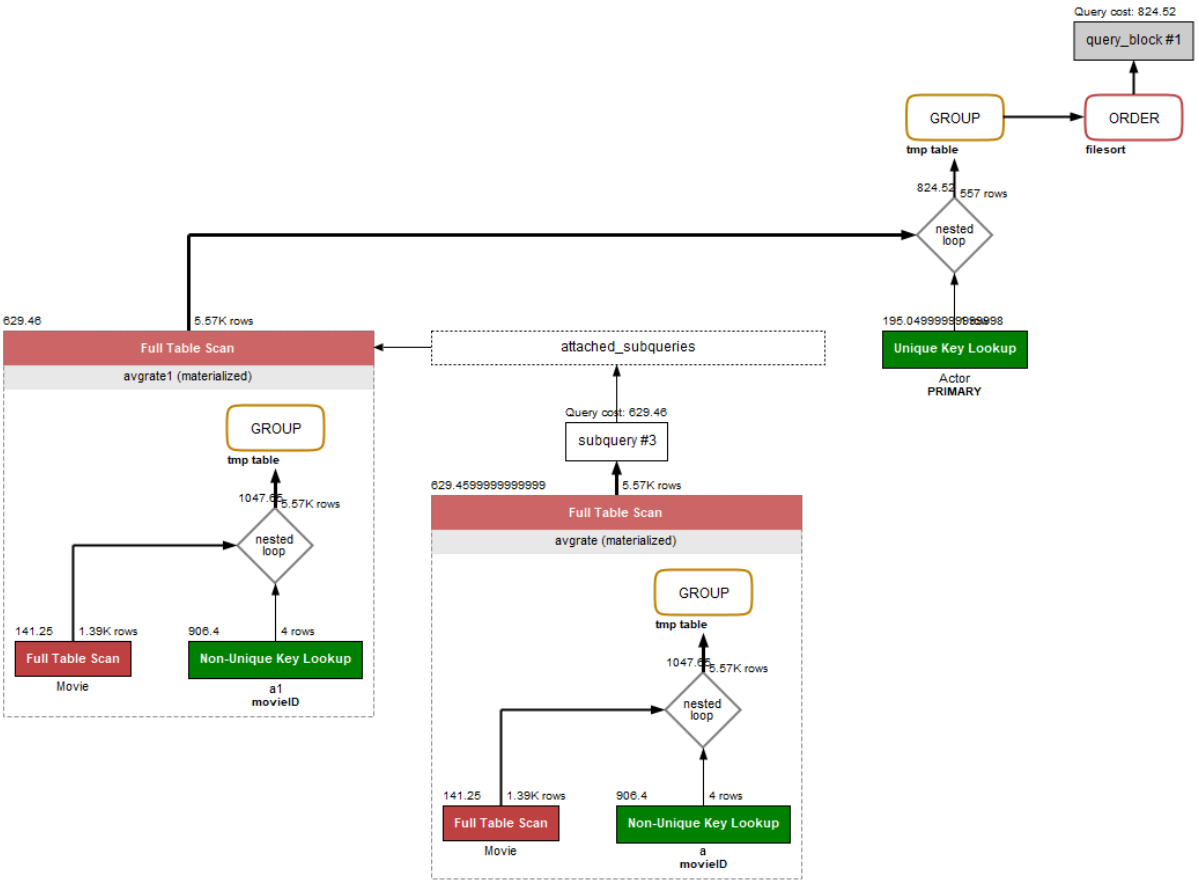
List actorID，name of actors (not only one), whose movie get the highest Average Rotten Tomatoes rating. Sort the results in a descending order by the name

```
 select actorID, name
from (select a1.actorID, AVG(ratingFromTomato) as avgrating1
        from Act a1 natural join Movie
        group by a1.actorID) as avgrate1
Natural Join Actor
where avgrating1 = (select max(avgrating) as maxrating
                        from (select a.actorID, AVG(ratingFromTomato) as avgrating
                              from Act a natural join Movie
                              group by a.actorID) as avgrate)
group by actorID
ORDER BY name
```

Results:

actorID	name
1196	Abdel Ahmed Ghili
721	Ahn Nae-sang
2733	Alice Pol
3470	Allie Anderson
2895	Andy Pandini
2267	Anna Acton
3718	Bame
4055	Bartosz Graczyk
3716	Bhanu
3561	Bradley Graham
1565	Byron McKim
3246	Candice-May Davies
3457	Carlin Joseph
825	Carlos Bernard
2099	Charles-André Bourassa

Origin

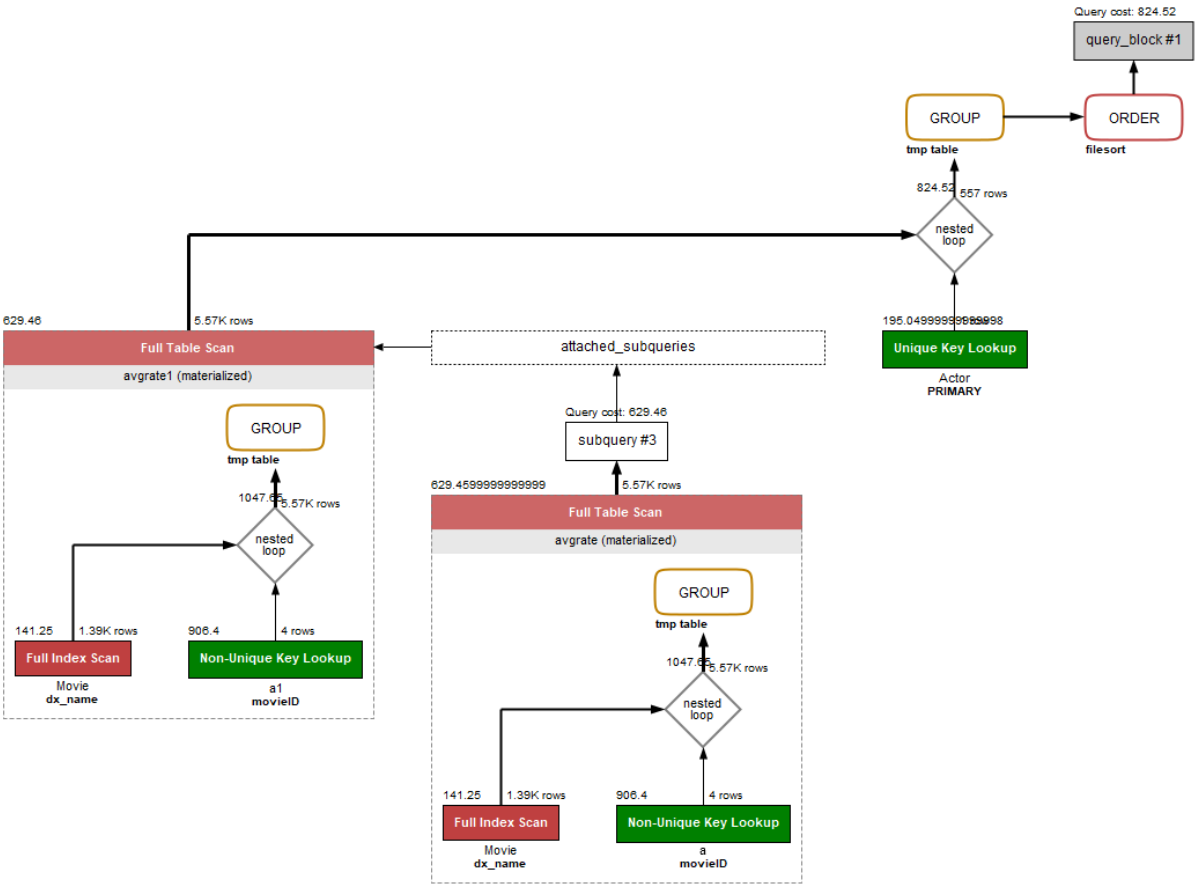


0.031 sec / 0.000 sec

-> Sort: <temporary>.name (actual time=15.536..15.542 rows=116 loops=1)
   -> Table scan on <temporary> (actual time=0.001..0.009 rows=116 loops=1)
      -> Temporary table with deduplication (actual time=15.469..15.485 rows=116 loops=1)
         -> Nested loop inner join (actual time=14.527..15.406 rows=116 loops=1)
            -> Filter: (avgrate1.avgrating1 = (select #3)) (actual time=14.508..15.218 rows=116 loops=1)
               -> Table scan on avgrate1 (actual time=0.001..0.204 rows=4084 loops=1)
                  -> Materialize (actual time=7.067..7.512 rows=4084 loops=1)
                     -> Table scan on <temporary> (actual time=0.001..0.176 rows=4084 loops=1)
                        -> Aggregate using temporary table (actual time=6.141..6.565 rows=4084 loops=1)
                           -> Nested loop inner join (cost=1047.65 rows=5574) (actual time=0.077..4.222 rows=4888 loops=1)
                              -> Table scan on Movie (cost=141.25 rows=1390) (actual time=0.060..0.379 rows=1387 loops=1)
                              -> Index lookup on a1 using movieID (movieID=Movie.movieID) (cost=0.25 rows=4) (actual time=0.002..0.002 rows=4 loops=1387)
               -> Select #3 (subquery in condition; run only once)

-> Aggregate: max(avgrate.avgrating) (actual time=7.421..7.421 rows=1 loops=1)
    -> Table scan on avgrate (actual time=0.001..0.181 rows=4084 loops=1)
        -> Materialize (actual time=6.737..7.170 rows=4084 loops=1)
            -> Table scan on <temporary> (actual time=0.001..0.171 rows=4084 loops=1)
                -> Aggregate using temporary table (actual time=5.856..6.271 rows=4084 loops=1)
                    -> Nested loop inner join (cost=1047.65 rows=5574) (actual time=0.058..4.058 rows=4888 loops=1)
                        -> Table scan on Movie (cost=141.25 rows=1390) (actual time=0.045..0.361 rows=1387 loops=1)
                        -> Index lookup on a using movieID (movieID=Movie.movieID) (cost=0.25 rows=4) (actual time=0.002..0.002 rows=4 loops=1387)
    -> Single-row index lookup on Actor using PRIMARY (actorID=avgrate1.actorID) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=116)

Index: Movie.RatingFromTomato



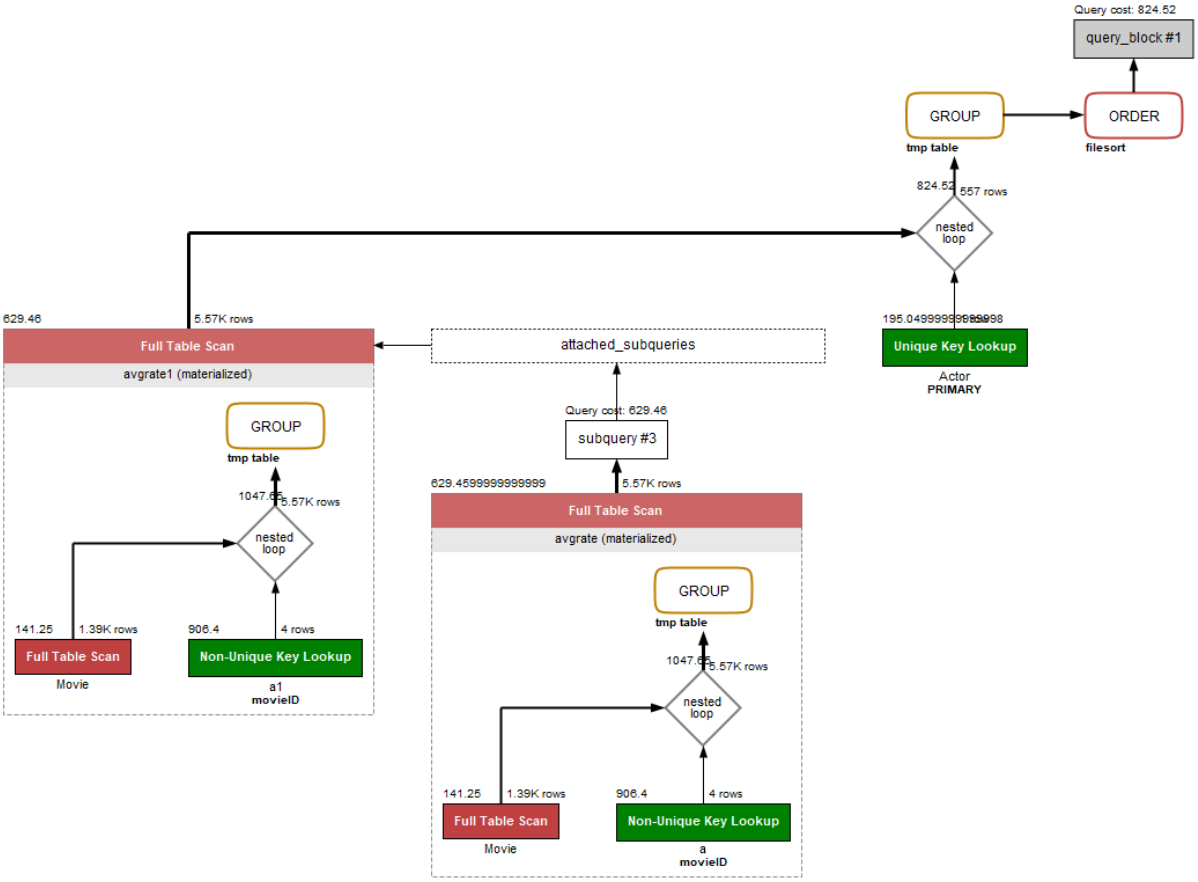
0.031 sec / 0.000 sec

-> Sort: <temporary>.name (actual time=17.985..17.991 rows=116 loops=1)
  -> Table scan on <temporary> (actual time=0.001..0.008 rows=116 loops=1)
    -> Temporary table with deduplication (actual time=17.901..17.920 rows=116 loops=1)
      -> Nested loop inner join (actual time=17.530..17.771 rows=116 loops=1)
        -> Filter: (avgrate1.avgrating1 = (select #3)) (actual time=17.506..17.537 rows=116 loops=1)
          -> Table scan on avgrate1 (actual time=0.001..0.194 rows=4084 loops=1)
            -> Materialize (actual time=8.307..8.748 rows=4084 loops=1)
              -> Table scan on <temporary> (actual time=0.002..0.181 rows=4084 loops=1)
                -> Aggregate using temporary table (actual time=7.371..7.798 rows=4084 loops=1)
                  -> Nested loop inner join (cost=1047.65 rows=5574) (actual time=0.108..4.947 rows=4888 loops=1)
                    -> Index scan on Movie using dx_name (cost=141.25 rows=1390) (actual time=0.052..0.383 rows=1387 loops=1)
                    -> Index lookup on a1 using movieID (movieID=Movie.movieID) (cost=0.25 rows=4) (actual time=0.002..0.003 rows=4 loops=1387)
        -> Select #3 (subquery in condition; run only once)

```
-> Aggregate: max(avgrate.avgrating)  (actual time=8.503..8.504 rows=1 loops=1)
    -> Table scan on avgrate  (actual time=0.001..0.180 rows=4084 loops=1)
        -> Materialize  (actual time=7.811..8.245 rows=4084 loops=1)
            -> Table scan on <temporary>  (actual time=0.001..0.189 rows=4084 loops=1)
                -> Aggregate using temporary table  (actual time=6.840..7.286 rows=4084
loops=1)
                    -> Nested loop inner join  (cost=1047.65 rows=5574) (actual time=0.061..4.770
rows=4888 loops=1)
                        -> Index scan on Movie using dx_name  (cost=141.25 rows=1390) (actual
time=0.047..0.341 rows=1387 loops=1)
                        -> Index lookup on a using movieID (movieID=Movie.movieID)  (cost=0.25
rows=4) (actual time=0.002..0.003 rows=4 loops=1387)
    -> Single-row index lookup on Actor using PRIMARY (actorID=avgrate1.actorID)  (cost=0.25
rows=1) (actual time=0.002..0.002 rows=1 loops=116)
```

Why choose: Movie. Ratingfromtomato is used in aggregate function.
Why no effect: we still need to go through all rows to extract the Ratingfromtomato, so this index does not help.

Index: Actor.name



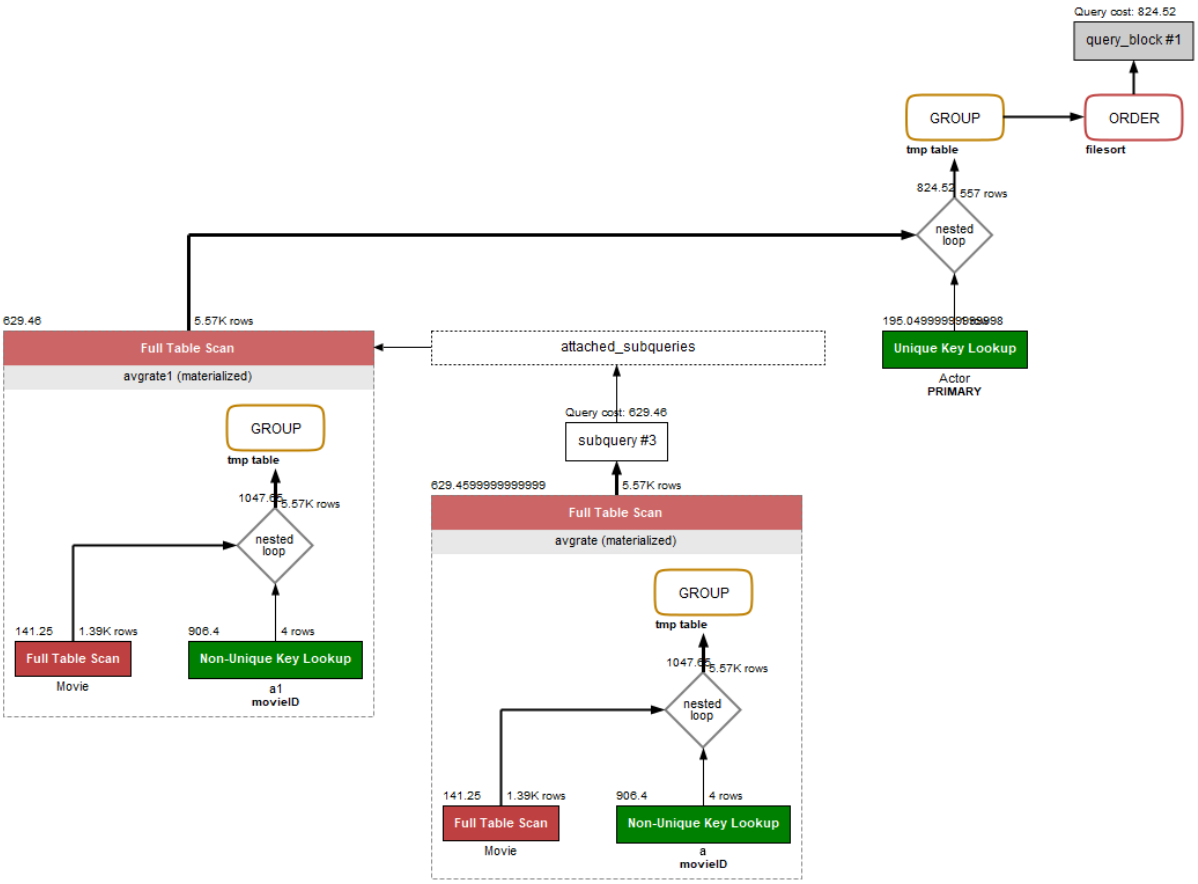
0.032 sec / 0.000 sec

-> Sort: <temporary>.name (actual time=15.597..15.603 rows=116 loops=1)
   -> Table scan on <temporary> (actual time=0.001..0.008 rows=116 loops=1)
      -> Temporary table with deduplication (actual time=15.507..15.523 rows=116 loops=1)
         -> Nested loop inner join (actual time=14.556..15.441 rows=116 loops=1)
            -> Filter: (avgrate1.avgrating1 = (select #3)) (actual time=14.538..15.245 rows=116 loops=1)
               -> Table scan on avgrate1 (actual time=0.001..0.207 rows=4084 loops=1)
                  -> Materialize (actual time=6.968..7.417 rows=4084 loops=1)
                     -> Table scan on <temporary> (actual time=0.001..0.178 rows=4084 loops=1)
                        -> Aggregate using temporary table (actual time=6.043..6.474 rows=4084 loops=1)
                           -> Nested loop inner join (cost=1047.65 rows=5574) (actual time=0.062..4.121 rows=4888 loops=1)
                              -> Table scan on Movie (cost=141.25 rows=1390) (actual time=0.050..0.380 rows=1387 loops=1)
                              -> Index lookup on a1 using movieID (movieID=Movie.movieID) (cost=0.25 rows=4) (actual time=0.002..0.002 rows=4 loops=1387)
               -> Select #3 (subquery in condition; run only once)
                  -> Aggregate: max(avgrate.avgrating) (actual time=7.554..7.554 rows=1 loops=1)

```
-> Table scan on avgrate  (actual time=0.001..0.173 rows=4084 loops=1)
    -> Materialize  (actual time=6.865..7.299 rows=4084 loops=1)
        -> Table scan on <temporary>  (actual time=0.001..0.171 rows=4084 loops=1)
            -> Aggregate using temporary table  (actual time=5.991..6.407 rows=4084
loops=1)
                -> Nested loop inner join  (cost=1047.65 rows=5574) (actual time=0.042..4.151
rows=4888 loops=1)
                    -> Table scan on Movie  (cost=141.25 rows=1390) (actual time=0.033..0.347
rows=1387 loops=1)
                    -> Index lookup on a using movieID (movieID=Movie.movieID)  (cost=0.25
rows=4) (actual time=0.002..0.002 rows=4 loops=1387)
    -> Single-row index lookup on Actor using PRIMARY (actorID=avgrate1.actorID)  (cost=0.25
rows=1) (actual time=0.001..0.001 rows=1 loops=116)
```

Why choose: we look up actorName in last step
Why no effect: we still need to go through all rows to extract the name of actors, so this index does not help.

Index: Act.actorID



0.032 sec / 0.000 sec

-> Sort: <temporary>.name (actual time=18.146..18.152 rows=116 loops=1)
  -> Table scan on <temporary> (actual time=0.001..0.011 rows=116 loops=1)
    -> Temporary table with deduplication (actual time=18.051..18.075 rows=116 loops=1)
      -> Nested loop inner join (actual time=17.033..17.970 rows=116 loops=1)
        -> Filter: (avgrate1.avgrating1 = (select #3)) (actual time=17.007..17.741 rows=116 loops=1)
          -> Table scan on avgrate1 (actual time=0.001..0.212 rows=4084 loops=1)
            -> Materialize (actual time=8.108..8.562 rows=4084 loops=1)
              -> Table scan on <temporary> (actual time=0.001..0.188 rows=4084 loops=1)
                -> Aggregate using temporary table (actual time=7.182..7.656 rows=4084 loops=1)
                  -> Nested loop inner join (cost=1047.65 rows=5574) (actual time=0.202..4.739 rows=4888 loops=1)
                    -> Table scan on Movie (cost=141.25 rows=1390) (actual time=0.176..0.569 rows=1387 loops=1)
                    -> Index lookup on a1 using movieID (movieID=Movie.movieID) (cost=0.25 rows=4) (actual time=0.002..0.003 rows=4 loops=1387)
          -> Select #3 (subquery in condition; run only once)
            -> Aggregate: max(avgrate.avgrating) (actual time=8.867..8.867 rows=1 loops=1)
              -> Table scan on avgrate (actual time=0.001..0.259 rows=4084 loops=1)

```
                    -> Materialize  (actual time=8.089..8.593 rows=4084 loops=1)
                      -> Table scan on <temporary>  (actual time=0.001..0.197 rows=4084 loops=1)
                        -> Aggregate using temporary table  (actual time=7.132..7.585 rows=4084
loops=1)
                          -> Nested loop inner join  (cost=1047.65 rows=5574) (actual time=0.067..4.660
rows=4888 loops=1)
                            -> Table scan on Movie  (cost=141.25 rows=1390) (actual time=0.050..0.436
rows=1387 loops=1)
                            -> Index lookup on a using movieID (movieID=Movie.movieID)  (cost=0.25
rows=4) (actual time=0.002..0.003 rows=4 loops=1387)
          -> Single-row index lookup on Actor using PRIMARY (actorID=avgrate1.actorID)  (cost=0.25
rows=1) (actual time=0.002..0.002 rows=1 loops=116)
```

Why no effect: it shows the query still use the default index to query.

Therefore, we decide to use default index for this query.