HW4 Query 3

```
SELECT DISTINCT
    m.title
    , md.director
FROM
    movies m
    , moviesdirectors md
    , moviesgenres mg
    , movieslanguages ml
    , movieslanguages ml2
WHERE
    m.movieid = md.movieid
    and m.movieid = mg.movieid
    and m.movieid = ml.movieid
    and m.movieid = ml2.movieid
    and m.imdbrating >= 8
    and mg.genre in ('Action', 'Sci-Fi', 'Adventure', 'Drama')
    and ml.language = 'English'
    and ml2.language <> 'English'
ORDER BY
    title
    , director ;
```

Before

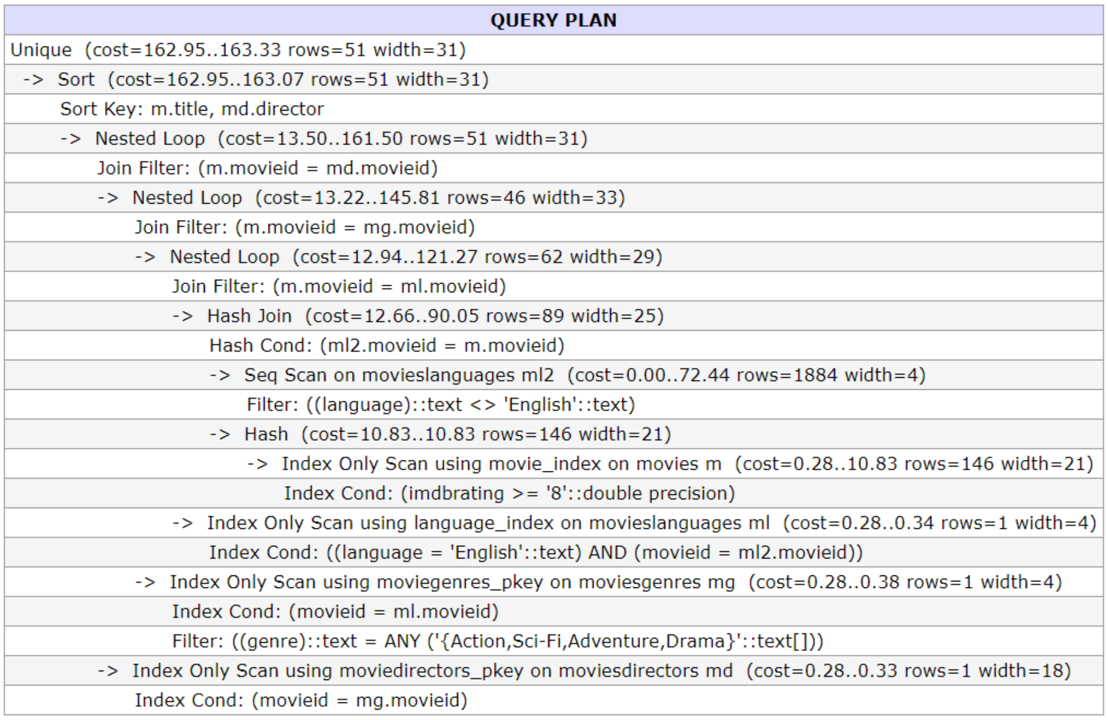
```
explain SELECT DISTINCT
    m.title
    , md.director
FROM
    movies m
    , moviesdirectors md
    , moviesgenres mg
    , movieslanguages ml
    , movieslanguages ml2
WHERE
    m.movieid = md.movieid
    and m.movieid = mg.movieid
    and m.movieid = ml.movieid
    and m.movieid = ml2.movieid
    and m.imdbrating >= 8
    and mg.genre in ('Action', 'Sci-Fi', 'Adventure', 'Drama')
    and ml.language = 'English'
    and ml2.language <> 'English'
ORDER BY
    title
    , director
```

```
QUERY PLAN
Unique  (cost=272.72..273.11 rows=51 width=31)
  ->  Sort  (cost=272.72..272.85 rows=51 width=31)
        Sort Key: m.title, md.director
        ->  Nested Loop  (cost=123.28..271.28 rows=51 width=31)
              Join Filter: (m.movieid = md.movieid)
              ->  Nested Loop  (cost=123.00..255.58 rows=46 width=33)
                    Join Filter: (m.movieid = mg.movieid)
                    ->  Nested Loop  (cost=122.72..231.05 rows=62 width=29)
                          Join Filter: (m.movieid = ml.movieid)
                          ->  Hash Join  (cost=122.44..199.83 rows=89 width=25)
                                Hash Cond: (ml2.movieid = m.movieid)
                                ->  Seq Scan on movieslanguages ml2  (cost=0.00..72.44 rows=1884 width=4)
                                      Filter: ((language)::text <> 'English'::text)
                                ->  Hash  (cost=120.61..120.61 rows=146 width=21)
                                      ->  Seq Scan on movies m  (cost=0.00..120.61 rows=146 width=21)
                                            Filter: (imdbrating >= '8'::double precision)
                          ->  Index Only Scan using movielanguages_pkey on movieslanguages ml  (cost=0.28..0.34 rows=1 width=4)
                                Index Cond: ((movieid = ml2.movieid) AND (language = 'English'::text))
                    ->  Index Only Scan using moviegenres_pkey on moviesgenres mg  (cost=0.28..0.38 rows=1 width=4)
                          Index Cond: (movieid = ml.movieid)
                          Filter: ((genre)::text = ANY ('{Action,Sci-Fi,Adventure,Drama}'::text[]))
              ->  Index Only Scan using moviedirectors_pkey on moviesdirectors md  (cost=0.28..0.33 rows=1 width=18)
                    Index Cond: (movieid = mg.movieid)
```

create index movie_index on movies(imdbrating, movieid, title);

create index language_index on movieslanguages(language, movieid);

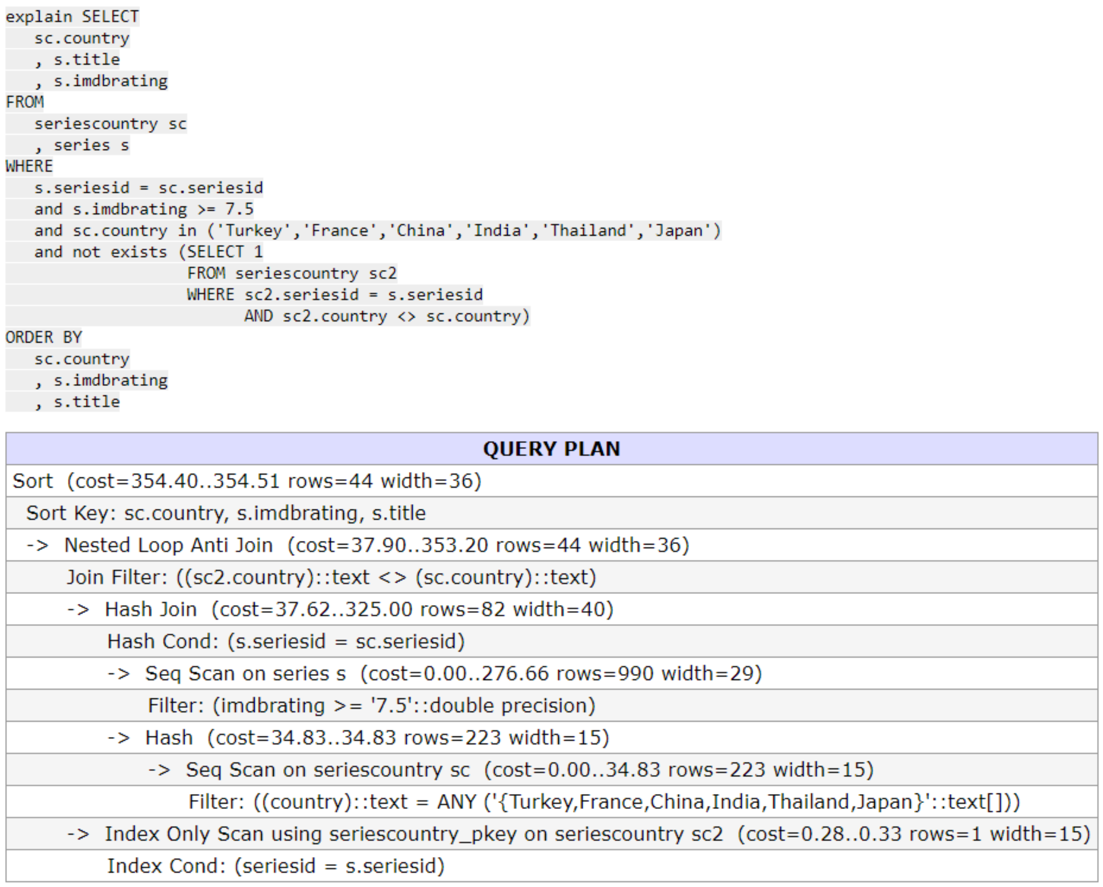
After



```
QUERY PLAN
Unique  (cost=162.95..163.33 rows=51 width=31)
  ->  Sort  (cost=162.95..163.07 rows=51 width=31)
        Sort Key: m.title, md.director
        ->  Nested Loop  (cost=13.50..161.50 rows=51 width=31)
              Join Filter: (m.movieid = md.movieid)
              ->  Nested Loop  (cost=13.22..145.81 rows=46 width=33)
                    Join Filter: (m.movieid = mg.movieid)
                    ->  Nested Loop  (cost=12.94..121.27 rows=62 width=29)
                          Join Filter: (m.movieid = ml.movieid)
                          ->  Hash Join  (cost=12.66..90.05 rows=89 width=25)
                                Hash Cond: (ml2.movieid = m.movieid)
                                ->  Seq Scan on movieslanguages ml2  (cost=0.00..72.44 rows=1884 width=4)
                                      Filter: ((language)::text <> 'English'::text)
                                ->  Hash  (cost=10.83..10.83 rows=146 width=21)
                                      ->  Index Only Scan using movie_index on movies m  (cost=0.28..10.83 rows=146 width=21)
                                            Index Cond: (imdbrating >= '8'::double precision)
                          ->  Index Only Scan using language_index on movieslanguages ml  (cost=0.28..0.34 rows=1 width=4)
                                Index Cond: ((language = 'English'::text) AND (movieid = ml2.movieid))
                    ->  Index Only Scan using moviegenres_pkey on moviesgenres mg  (cost=0.28..0.38 rows=1 width=4)
                          Index Cond: (movieid = ml.movieid)
                          Filter: ((genre)::text = ANY ('{Action,Sci-Fi,Adventure,Drama}'::text[]))
              ->  Index Only Scan using moviedirectors_pkey on moviesdirectors md  (cost=0.28..0.33 rows=1 width=18)
                    Index Cond: (movieid = mg.movieid)
```

HW5 Query 6

```
SELECT
    sc.country
    , s.title
    , s.imdbrating
FROM
    seriescountry sc
    , series s
WHERE
    s.seriesid = sc.seriesid
    and s.imdbrating >= 7.5
    and sc.country in
('Turkey','France','China','India','Thailand','Japan')
    and not exists (SELECT 1
                    FROM seriescountry sc2
                   WHERE sc2.seriesid = s.seriesid
                          AND sc2.country <> sc.country)
ORDER BY
    sc.country
    , s.imdbrating
    , s.title;
```

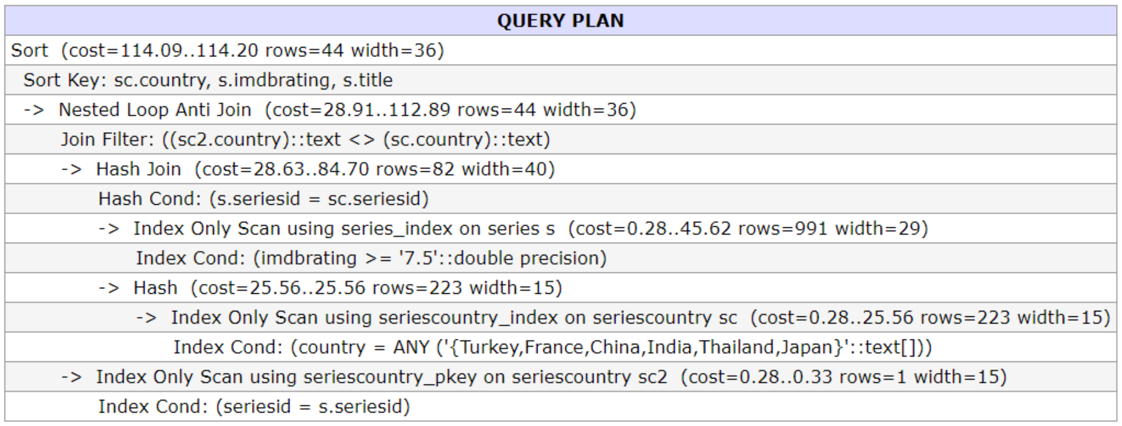


```
explain SELECT
   sc.country
   , s.title
   , s.imdbrating
FROM
   seriescountry sc
   , series s
WHERE
   s.seriesid = sc.seriesid
   and s.imdbrating >= 7.5
   and sc.country in ('Turkey','France','China','India','Thailand','Japan')
   and not exists (SELECT 1
                   FROM seriescountry sc2
                   WHERE sc2.seriesid = s.seriesid
                         AND sc2.country <> sc.country)
ORDER BY
   sc.country
   , s.imdbrating
   , s.title
```

```
QUERY PLAN
Sort  (cost=354.40..354.51 rows=44 width=36)
  Sort Key: sc.country, s.imdbrating, s.title
  ->  Nested Loop Anti Join  (cost=37.90..353.20 rows=44 width=36)
        Join Filter: ((sc2.country)::text <> (sc.country)::text)
        ->  Hash Join  (cost=37.62..325.00 rows=82 width=40)
              Hash Cond: (s.seriesid = sc.seriesid)
              ->  Seq Scan on series s  (cost=0.00..276.66 rows=990 width=29)
                    Filter: (imdbrating >= '7.5'::double precision)
              ->  Hash  (cost=34.83..34.83 rows=223 width=15)
                    ->  Seq Scan on seriescountry sc  (cost=0.00..34.83 rows=223 width=15)
                          Filter: ((country)::text = ANY ('{Turkey,France,China,India,Thailand,Japan}'::text[]))
        ->  Index Only Scan using seriescountry_pkey on seriescountry sc2  (cost=0.28..0.33 rows=1 width=15)
              Index Cond: (seriesid = s.seriesid)
```

create index series_index on series(imdbrating, seriesid, title);

create index seriescountry_index on seriescountry(country, seriesid);

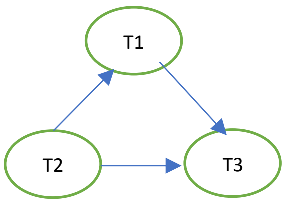


```
QUERY PLAN
Sort  (cost=114.09..114.20 rows=44 width=36)
  Sort Key: sc.country, s.imdbrating, s.title
  ->  Nested Loop Anti Join  (cost=28.91..112.89 rows=44 width=36)
        Join Filter: ((sc2.country)::text <> (sc.country)::text)
        ->  Hash Join  (cost=28.63..84.70 rows=82 width=40)
              Hash Cond: (s.seriesid = sc.seriesid)
              ->  Index Only Scan using series_index on series s  (cost=0.28..45.62 rows=991 width=29)
                    Index Cond: (imdbrating >= '7.5'::double precision)
              ->  Hash  (cost=25.56..25.56 rows=223 width=15)
                    ->  Index Only Scan using seriescountry_index on seriescountry sc  (cost=0.28..25.56 rows=223 width=15)
                          Index Cond: (country = ANY ('{Turkey,France,China,India,Thailand,Japan}'::text[]))
        ->  Index Only Scan using seriescountry_pkey on seriescountry sc2  (cost=0.28..0.33 rows=1 width=15)
              Index Cond: (seriesid = s.seriesid)
```

Q2

CONFLICT:

r1(x) --- w3(x)   r2(z) --- w3(z)   w2(z) --- r3(z), w3(z)   w2(w) --- r1(w)



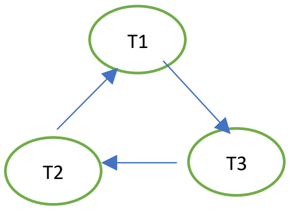
sl1(x) sl2(z) sl1(y) **r1(x) r2(z) r1(y)** xl2(w) xl2(z) **w2(w) w2(z)** ul2(z) sl3(z) **r3(z)**

--- T2 is in the shrinking phase

ul2(w) sl1(w) xl1(y) ul1(x) xl3 (x) xl3 (z) **w3(x) r1(w) w1(y) w3(z)** --- T1 is in the shrinking phase

CONFLICT:

r1(x) --- w3(x)   r3(z) --- w2(z)   w2(w) --- r1(w)



sl1(x) sl1(y) sl2(z) sl3(z) sl3(x) **r1(x) r1(y) r2(z) r3(z) r3(x)**

w3(x) w2(w) w2(z) r1(w) w1(y)

For **w3(x)** in here, we need to **ul(x)** first or we can put **r1(w) w1(y)** in front of T2 and T3. By the first way, if we do **ul(x), T1 is in the shrinking phase now, we cannot add any more lock then such as r1(w) w1(y)** and it fails. By the second case, if we put **r1(w) w1(y)** lock first and then do **ul(x),** we can find transaction order in there is wrong: we need do **w2(w)** in front of **r1(w),**

Q3

a)

Redo

103 T3 update P1

105 T1 update P4

Undo

107 abort T1 105

108 undo 105 100

109 undo 100 –

110 T1 end

b) NO force is used, because when T3 is committed, the last update on P1 should be 103, but in data pages entry it is 101.

c) Steal is used, because when T1 is not committed, the update of T1 on any page should not be recorded. But “100 T1 update P2” is showed on data pages, so it is steal.