Σχεδιασμός Βάσεων δεδομένων

PROJECT A

Αποστολόπουλος Θεμιστοκλής p3180013

## Ζήτημα 1

Έχουμε τα παρακάτω queries:

```
select title from movies where pyear between 1990 and 2000

select pyear, title from movies where pyear between 1990 and 2000

select title, pyear  from movies where pyear between 1990 and 2000
order by pyear, title
```

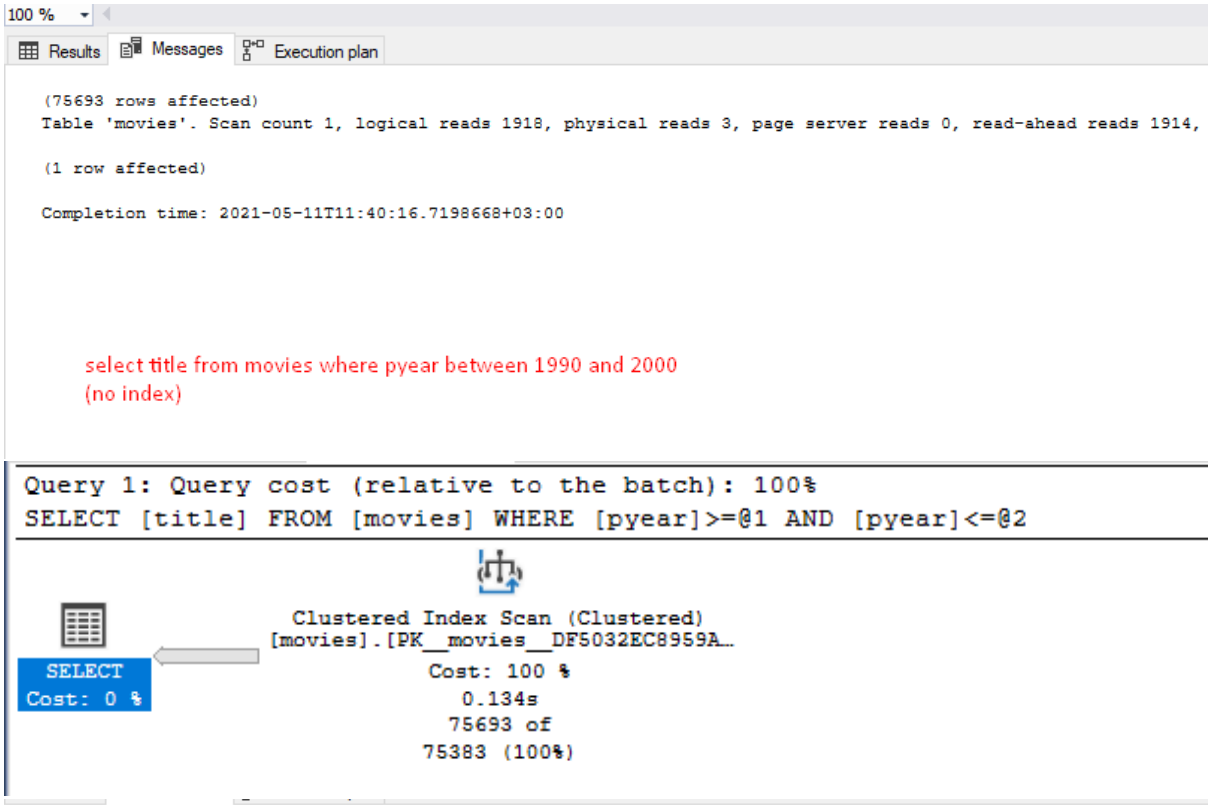
Ο πίνακας movies έχει primary key το mid οπότε και clustered index σε αύτο το γνώρισμα που δεν μας βοηθάει ιδιαίτερα σε αυτά τα queries.Σε όλα τα queries η «αναζήτηση» γίνεται στο όρισμα pyear και επιλογή στο title ή και pyear οπότε θεωρώ ορθή την δημιουργία non clustered index στα ορίσματα pyear,title(pyear πρώτο γιατί το χρησιμοποιώ για την αναζήτηση)

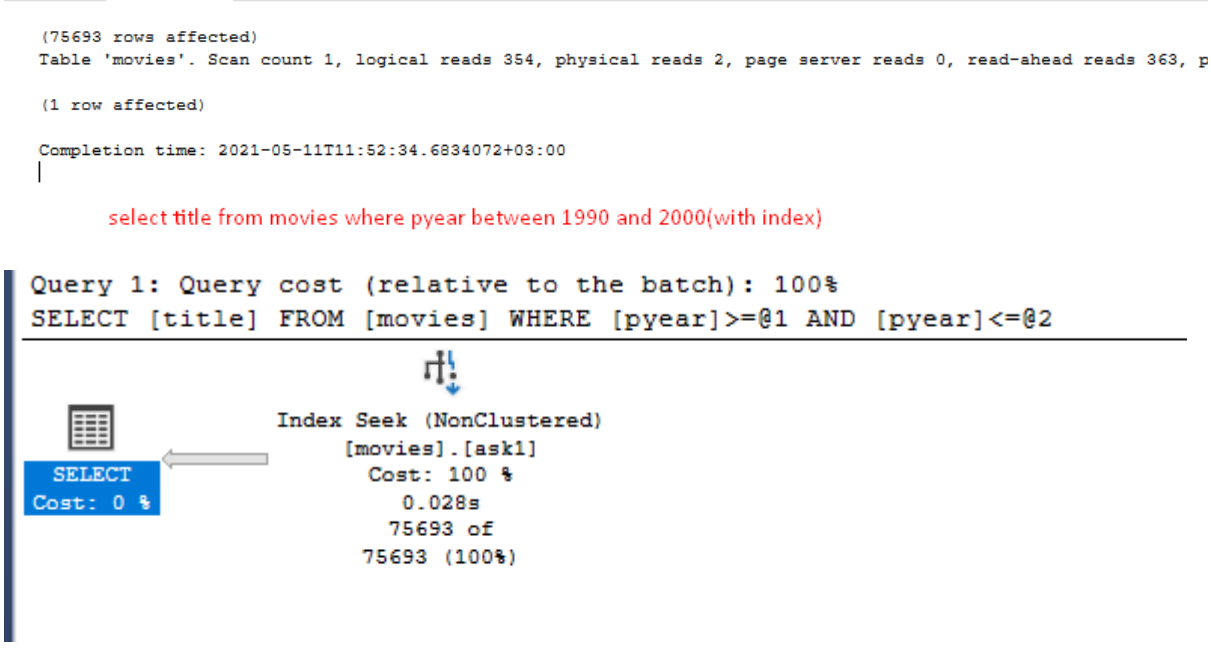
Η εντολή δημιουργίας του index είναι η ακόλουθη:

```
CREATE  INDEX ask1 ON movies(pyear, title)
```

Πριν την δημιουργία του index έτρεξα τα queries για να δώ στατιστικά σε reads καθώς και execution plan(πριν την εκτέλεση κάθε query καθάριζα buffers).Έχω τις εξής παρατηρήσεις(πρώτα 2 screenshots για κάθε query εμφανίζουν τα στατιστικά και το execution plan πριν την εκτέλεση με index ενώ τα επόμενα 2 με την εκτέλεση του query ενώ υπάρχει το index που εφτίαξα :

Για query 1(`select title from movies where pyear between 1990 and 2000`):



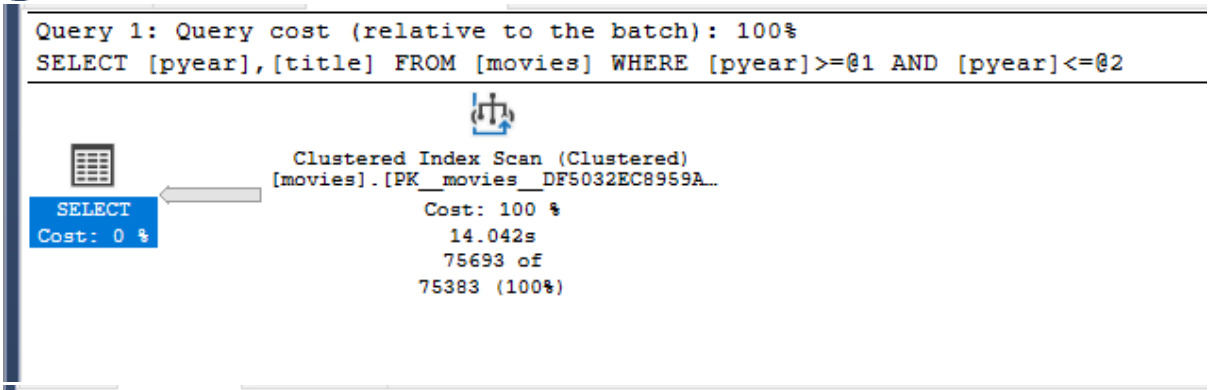




Παρατηρώ μείωση στα logical reads καθώς πλέον με το ευρετήριο δεν κάνω index scan αλλά ένα index seek αρκεί για να πραγματοποιηθεί η αναζήτηση «εύρους» στο pyear.

Για query 2(`select pyear, title from movies where pyear between 1990 and 2000`)

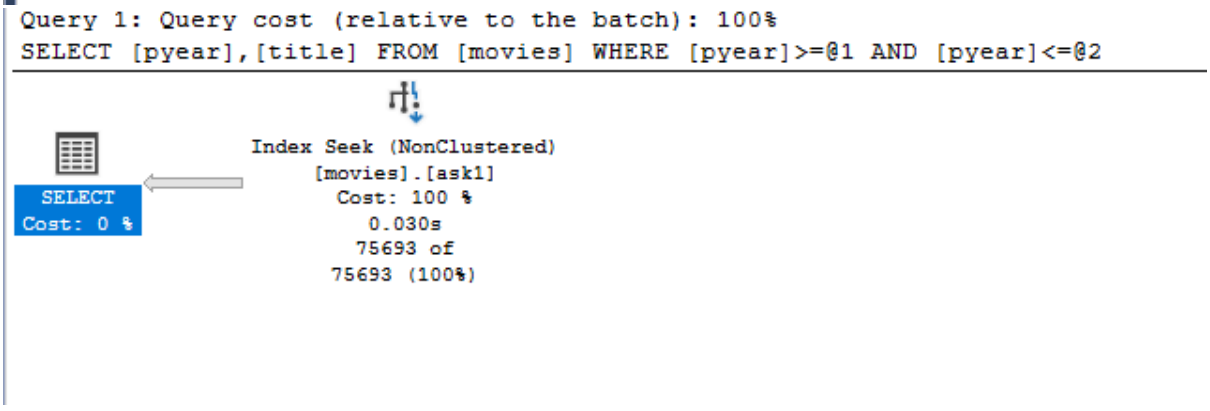
```
(75693 rows affected)
Table 'movies'. Scan count 1, logical reads 1918, physical reads 1092, page server reads 0, read

(1 row affected)

Completion time: 2021-05-11T11:41:23.0870047+03:00
```

select pyear, title from movies where pyear between 1990 and 2000
(no index)




```
(75693 rows affected)
Table 'movies'. Scan count 1, logical reads 354, physical reads 2, page server reads 0, read-ahead reads 363, pag

(1 row affected)

Completion time: 2021-05-11T11:53:15.7124598+03:00
```

select pyear, title from movies where pyear between 1990 and 2000
(with index)

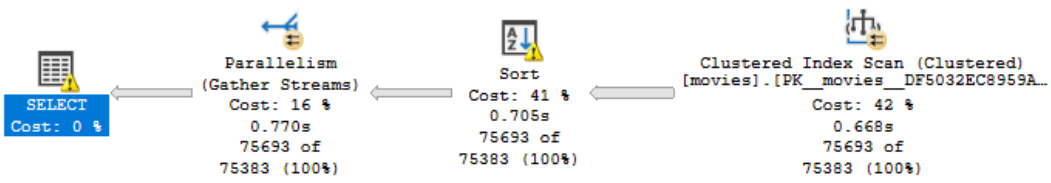



Παρατηρώ πάλι ότι το index scan γίνεται index seek λόγω του index και την μείωση στα logical reads του query μετά την δημιουργία του index.

Για query 3 (`select title, pyear from movies where pyear between 1990 and 2000 order by pyear, title`)

```
(75693 rows affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 644, page serve
Table 'movies'. Scan count 9, logical reads 2012, physical reads 778, page server reads 0, read-ahead reads 1914, page se

(1 row affected)

Completion time: 2021-05-11T11:45:51.2650830+03:00
```

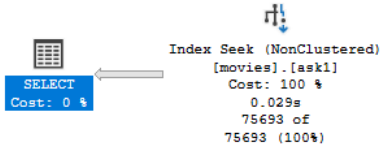
select title, pyear from movies where pyear between 1990 and 2000
order by pyear, title(no index)




```
(75693 rows affected)
Table 'movies'. Scan count 1, logical reads 354, physical reads 2, page server reads 0, read-ahead reads 363, pa

(1 row affected)

Completion time: 2021-05-11T11:54:01.0579753+03:00
```

select title, pyear from movies where pyear between 1990 and 2000
order by pyear, title(with index)

```
Query 1: Query cost (relative to the batch): 100%
SELECT [title],[pyear] FROM [movies] WHERE [pyear]>=@1 AND [pyear]<=@2 ORDER BY [pyear] ASC,[title] ASC
```




Το query περιεχεί και order by στο όρισμα pyear και title αν υπάρχει ισοπαλία στο pyear.Στην εκτέλεση του query στην αρχή, χωρίς index θα γίνει το index scan, θα παρθούν τα αποτελέσματα για pyear between 1990 and 2000 αλλά μετά γίνεται και sorting σε αυτά.Με την ύπαρξη του index (pyear ήδη ταξινομημένο στο ευρετήριο) δεν χρειάζεται ο τελεστής sort, με αποτέλεσμα μείωση των reads(μαζί με το index seek φυσικά σε σχέση με index scan).

1.2

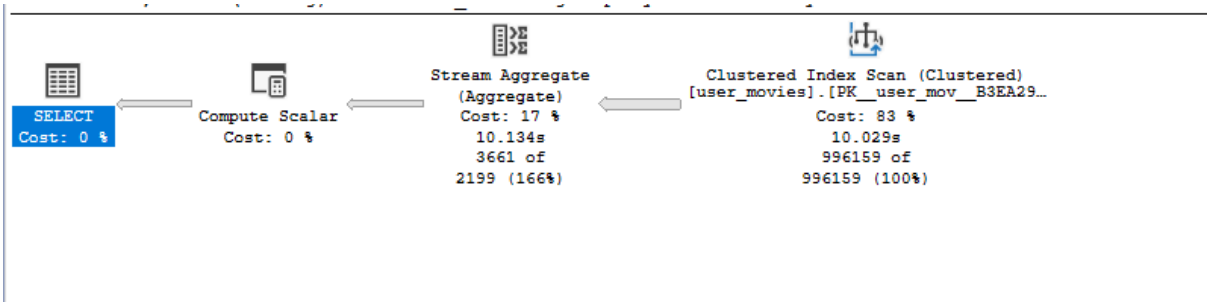
Έχουμε τα ακόλουθα queries:

```
select mid, count(rating)
 from user_movies group by mid order by mid

select userid, count(rating)
 from user_movies group by userid order by userid
```

Παρακάτω δείχνω τα reads και το execution plan κάθε query πριν δημιουργήσω κάποιο ευρετήριο και καθαρίζοντας τους buffers πριν το «τρέξιμο» (πρώτα 2 screenshots αντιστοιχούν στο ερώτημα 1, τα επόμενα 2 για το δεύτερο.)

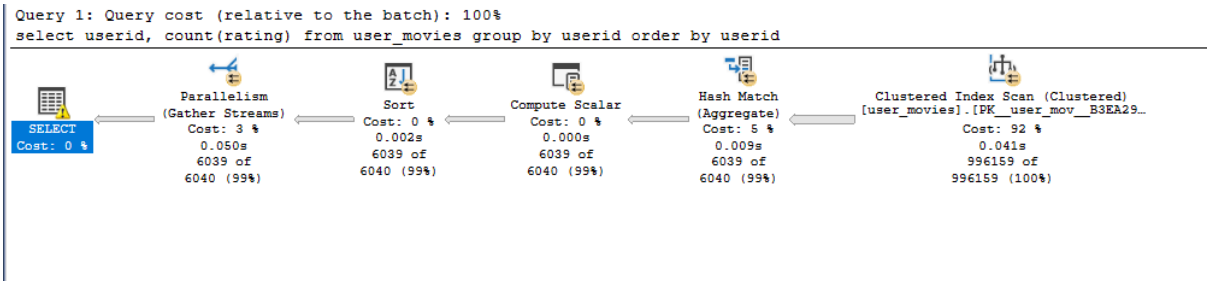
```
(3661 rows affected)
Table 'user_movies'. Scan count 1, logical reads 2601, physical reads 1213, page server reads 0, read-ahead reads 2597, page serv

(1 row affected)

Completion time: 2021-05-11T14:42:35.6150353+03:00
```

select mid, count(rating)
from user_movies group by mid order by mid(no index)




```
(6039 rows affected)
Table 'user_movies'. Scan count 9, logical reads 2733, physical reads 2, page server reads 0, read-ahe
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead re

(1 row affected)

Completion time: 2021-05-11T14:47:16.9664861+03:00
```

select userid, count(rating)
from user_movies group by userid order by userid(no index)

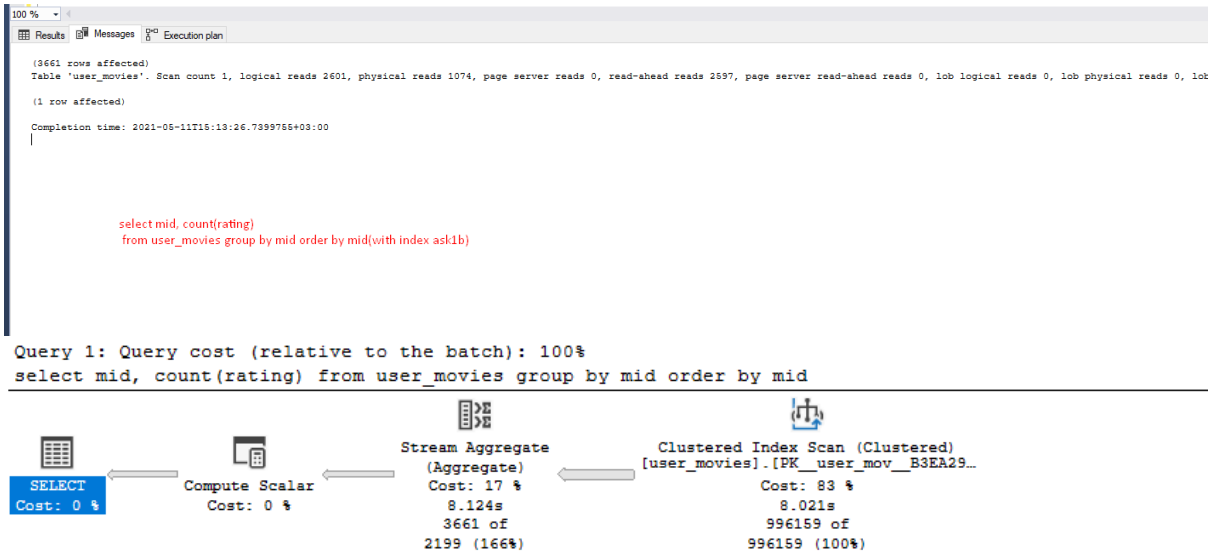


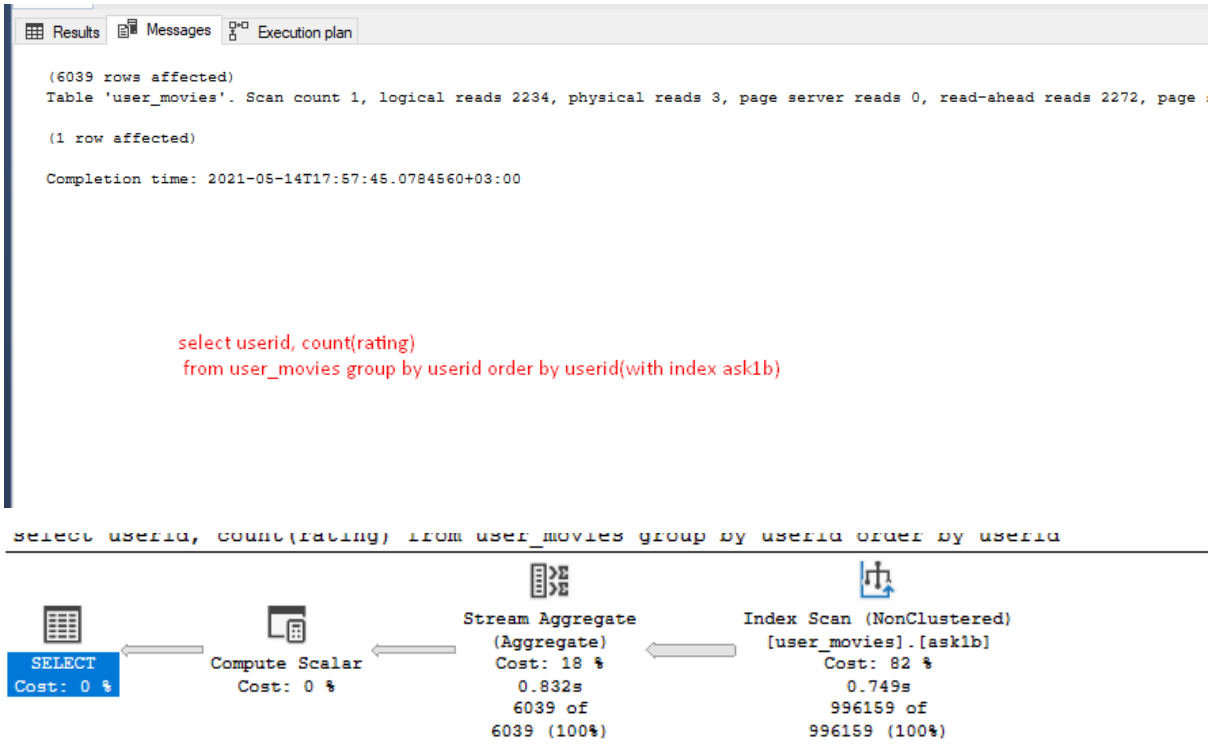

..

Παρατηρώ ότι το 2ο query που γίνεται το group by και order by με το userid χρειάζεται παραπάνω reads από το 1ο και στο πλάνο εκτέλεσης εμφανίζονται διάφοροι «τελεστές» που καθιστερούν το query, όπως sort , hash match. Ο πίνακας user_movies έχει primary key με 2 attributes(mid, userid) όμως επειδή το userid εμφανίζεται ως δεύτερο στην σχέση αποφασίζω να δώσω προτεραιότητα σε αυτό καθώς με το clustered index το mid είναι καλυμένο.Αφού η συχνότητα των ερωτημάτων είναι ίδια πιστεύω ότι η απαλοιφή των sort, hashmatch στο 2ο query είναι καλύτερη επιλογή από την βελτίωση του 1ου query.

Θα δημιουργήσω το ακόλουθο index: `CREATE INDEX ask1b ON user_movies(userid, rating);`

Ξανατρέχω τα queries και έχω:






Με το συγκεκριμένο ευρετήριο πέτυχα τον σκοπό μου: έγινε απαλοιφή του sort, hash match κτλπ και το ερώτημα χρειάζεται λιγότερα reads.

Ζήτημα 2ο

1.

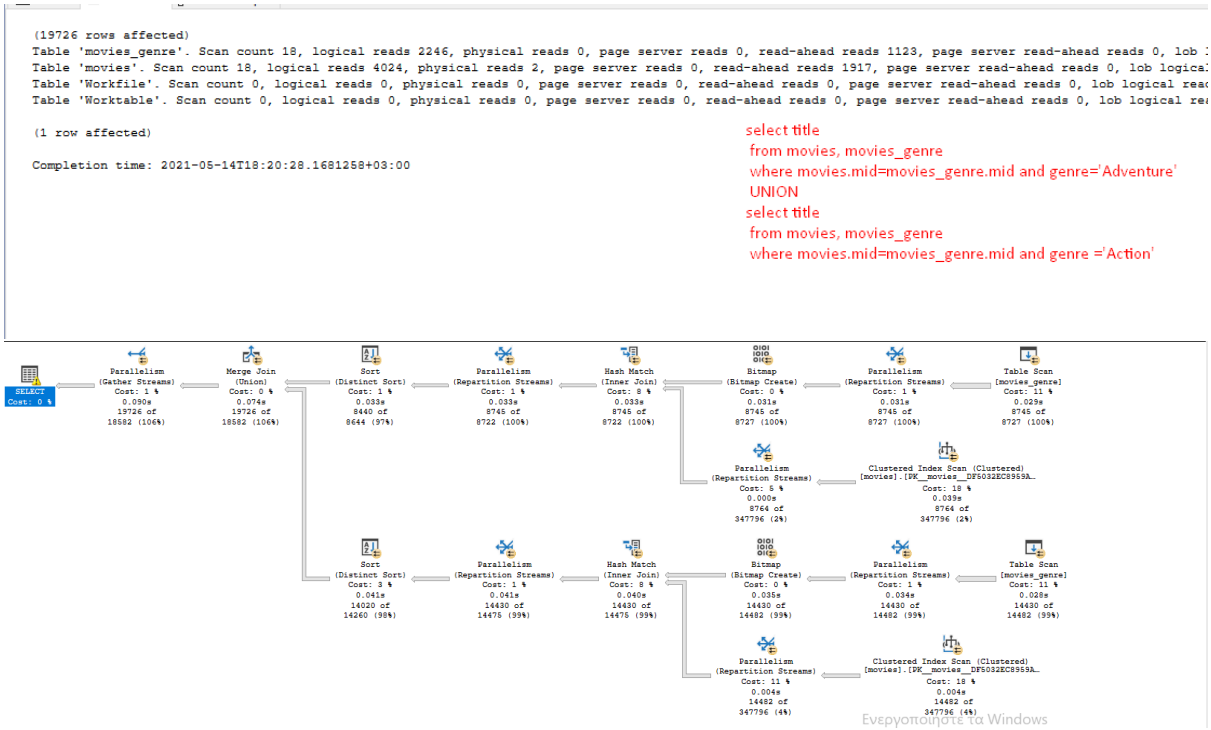
Το ερώτημα

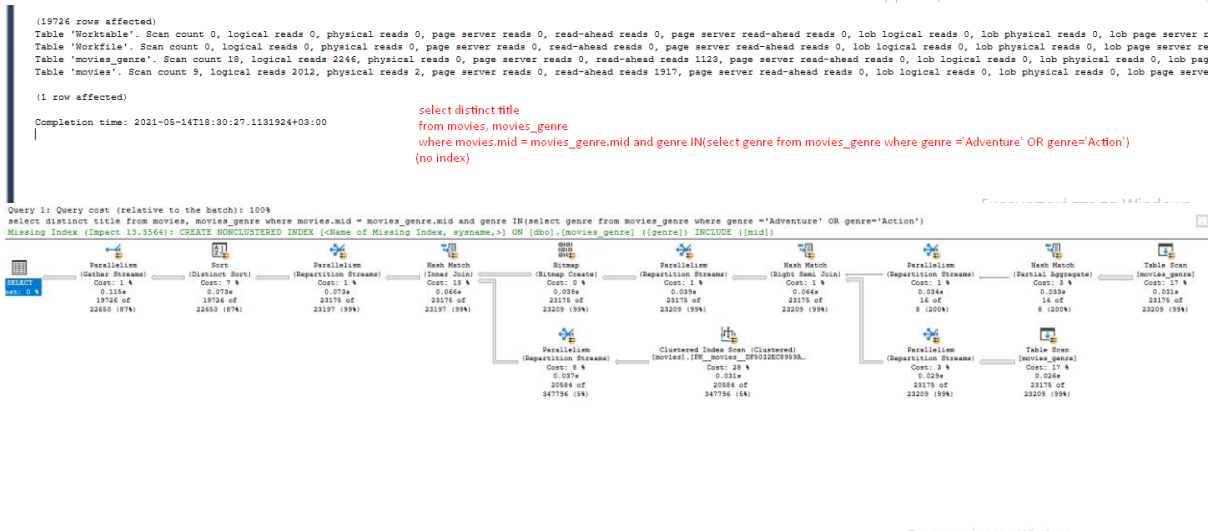
```
select title
 from movies, movies_genre
 where movies.mid=movies_genre.mid and genre='Adventure'
 UNION
select title
 from movies, movies_genre
        where movies.mid=movies_genre.mid and genre ='Action'
```

μου γυρνάει ίδια αποτελέσματα με το:

```
select distinct title
 from movies, movies_genre
        where movies.mid = movies_genre.mid and genre IN(select genre from
       movies_genre where genre ='Adventure' OR genre='Action')
```

.Τρέχω τα ερωτήματα για να δώ αν το query μου είναι πιο «γρήγορο» από το original query(καθαρίζοντας buffers πριν την εκτέλεση του κάθε query).Παραθέτω screenshots με τα αποτελέσματα και το πλάνο εκτέλεσης κάθε ερωτήματος(2 πρώτα screenshots αντιστοιχούν στο original query, τα επόμενα στο δικό μου).



Παρατήρω ότι έχω λιγότερο sum logical reads για το ερώτημα μου.

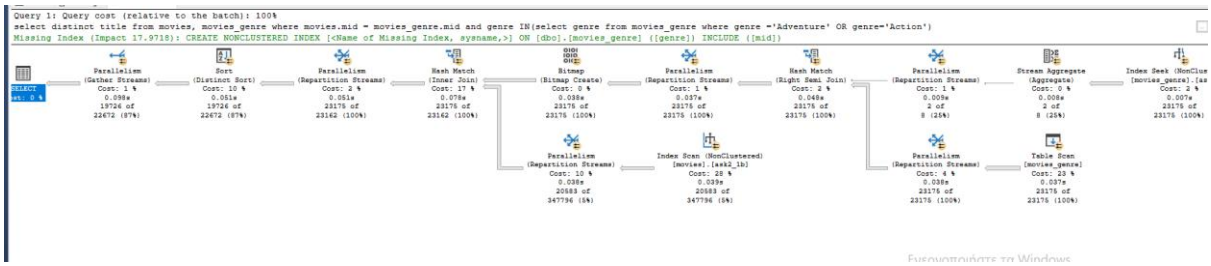
Τα ευρετήρια που αποφάσισα να δημιουργήσω για να κάνω πιο γρήγορο το ερώτημα μου είναι : `create index ask2_1 on movies_genre(genre);`

Και `create index ask2_1b on movies(title);`

Τρέχοντας τώρα το query μου ξανά έχω:

```
(19726 rows affected)
Table 'movies_genre'. Scan count 26, logical reads 1207, physical reads 2, page server reads 0, read-ahead reads 1196, page server
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead
Table 'movies'. Scan count 9, logical reads 1484, physical reads 3, page server reads 0, read-ahead reads 1443, page server read-ah

(1 row affected)

Completion time: 2021-05-14T18:44:33.6572446+03:00
```

```
select distinct title
 from movies, movies_genre
 where movies.mid = movies_genre.mid and genre IN(select genre from movies_genre where genre ='Adventure' OR genre='Action')
(with indexes)
```



Έχει μειωθεί το άθροισμα των logical reads από τους πίνακες movies, movies_genre .

2.2

Πρωτο query :

```
select title
from movies
where mid in(select  movies.mid from movies, roles, actors
    where movies.mid=roles.mid and actors.aid=roles.aid
    except
    select distinct movies.mid
    from movies, roles, actors
          where movies.mid=roles.mid and actors.aid=roles.aid and gender = 'F');
```

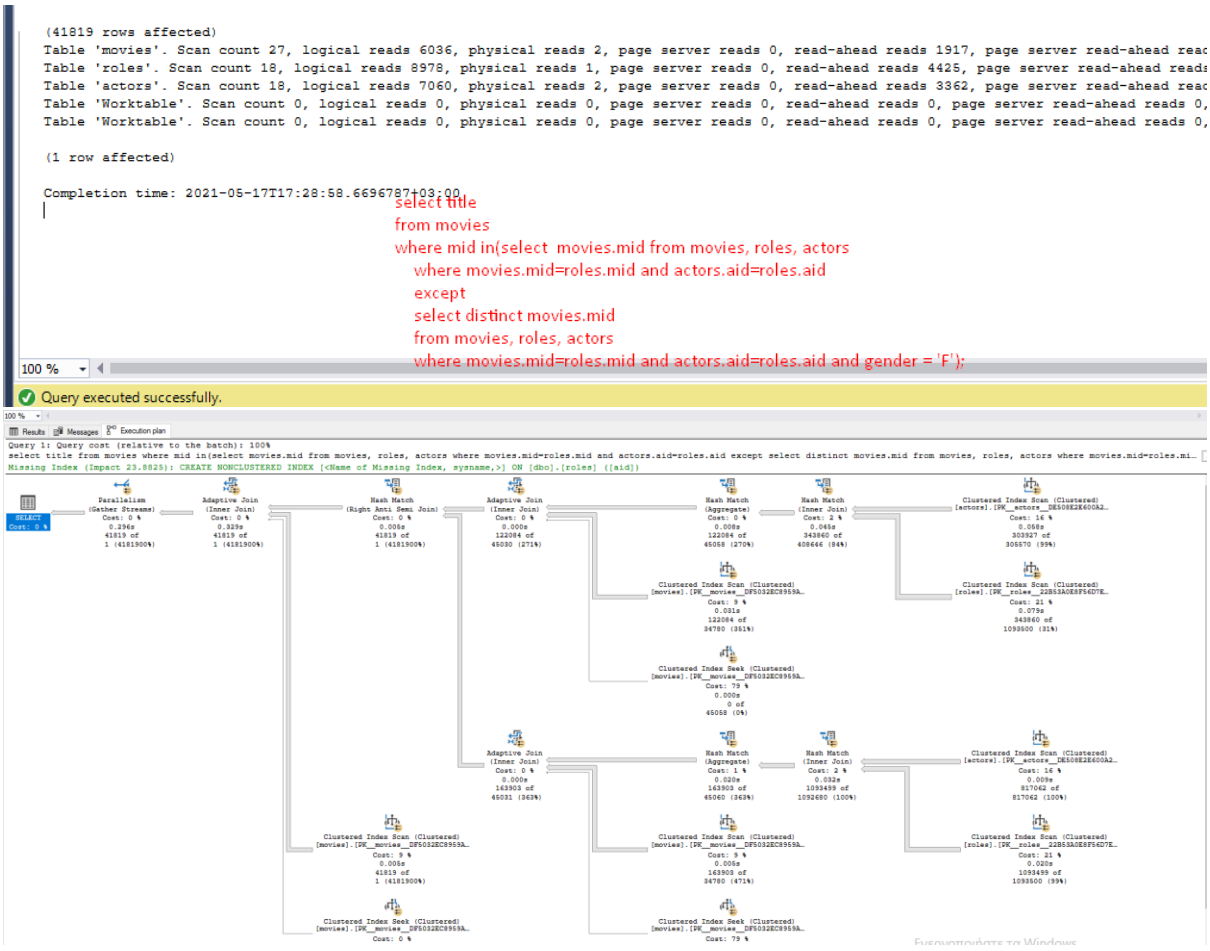
Η λογικη πίσω από αυτό το query είναι η ακόλουθη:ήθελα να πάρω τα mid στα οποία έπαιζαν μόνο άντρες ηθοποιοί οπότε βρήκα όλους τους ηθοποιούς και με το except «αφαίρεσα» τις γυναίκες.Η όλη διαδικασία γίνεται με mid και όχι με title καθώς παρατήρησα διπλότυπα στο title και με distinct title χωρίς το mid IN έχω λάθος αποτελέσματα).

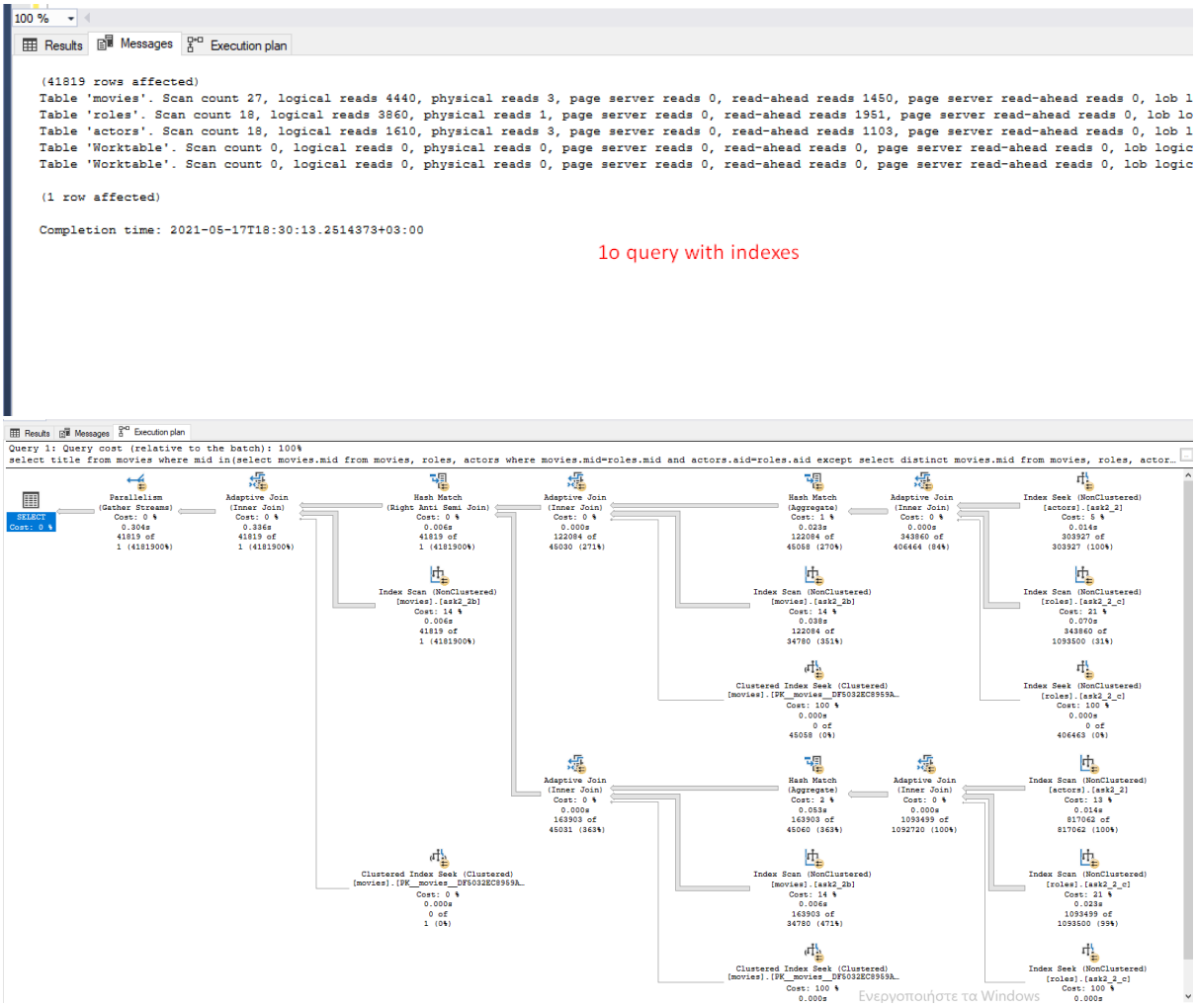
Τα ευρετήρια που επέλεξα είναι τα ακόλουθα:

```
create index ask2_2 on actors(gender)

create index ask2_2b on movies(title)

create index ask2_2c on roles(aid)
```

Για να γίνει seek αντί για scan στον actors στο gender =’F’ ,στο roles.aid και στο title

Παραθέτω τα reads και τα execution plans(πρωτα 2 screenshots πριν το index)



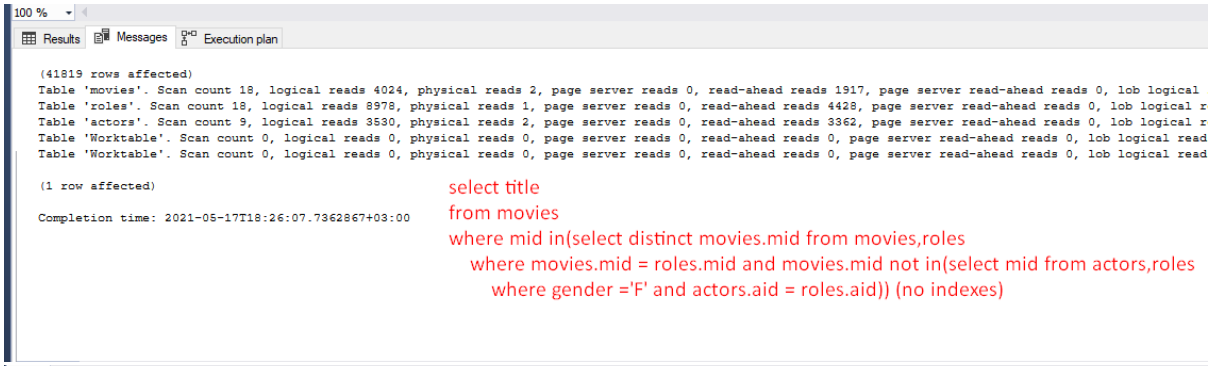



Δεύτερο query:

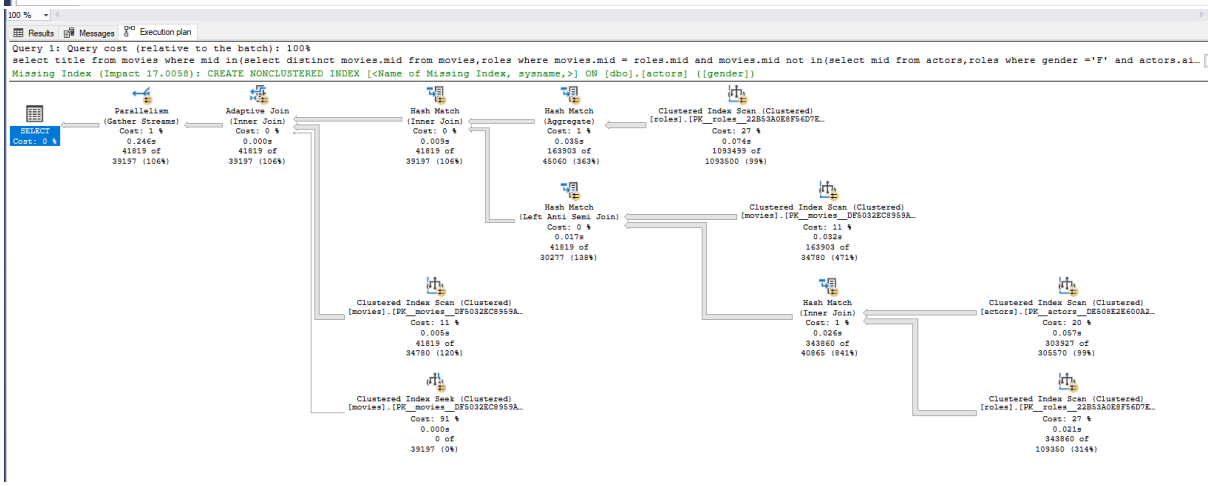
```
Select title
From movies where mid in(
select distinct movies.mid
from movies,roles
where movies.mid = roles.mid and movies.mid not in(select mid from actors,roles
            where gender ='F' and actors.aid = roles.aid))
```

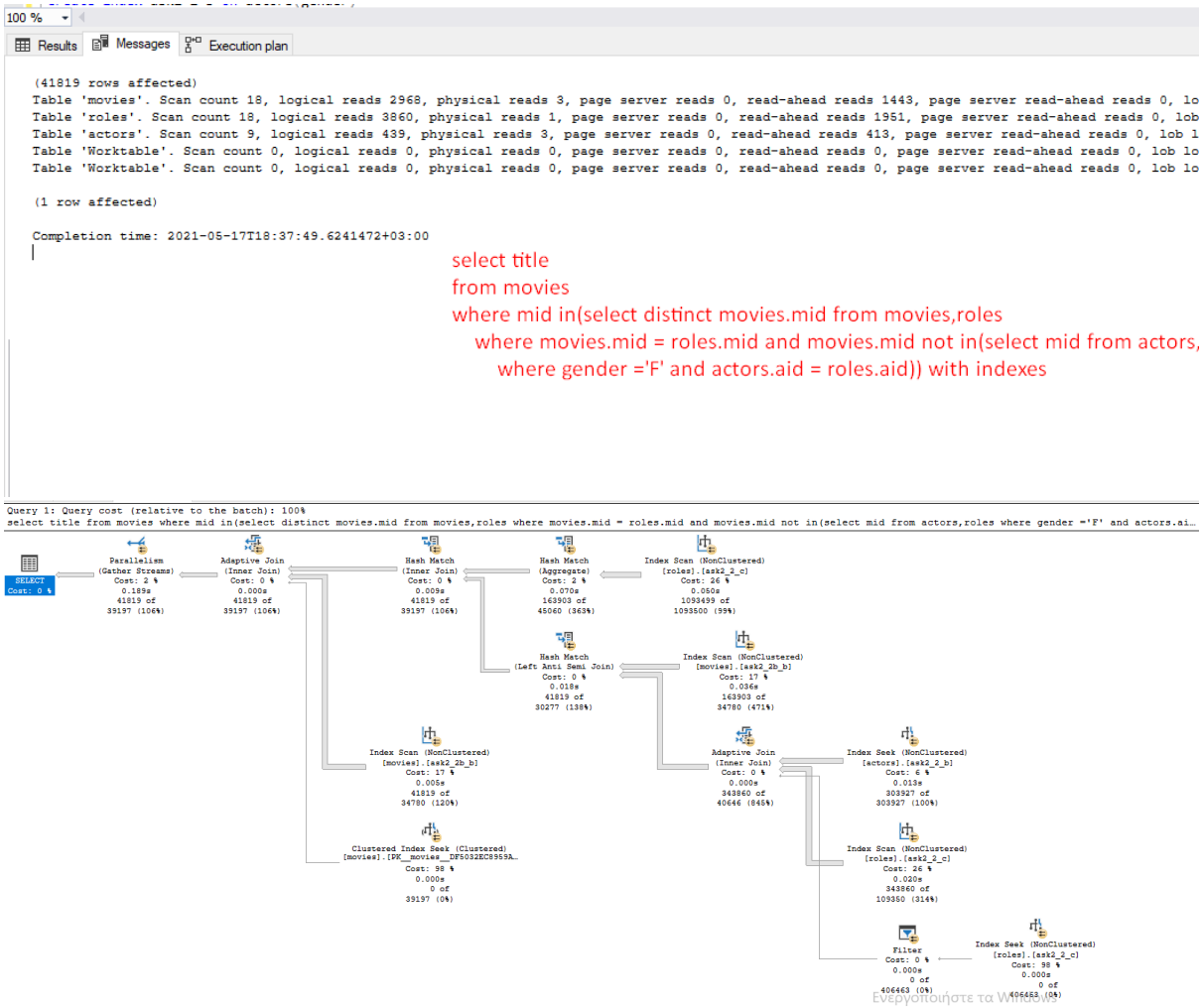
Η λογική πίσω από αυτό το query είναι πάλι παρόμοια με το πρώτο.παίρνω τις ταινίες που έχουν ηθοποιούς και βλέπω αν το mid τους βρίσκονται ανάμεσα σε αυτά των ταινιών οπού υπάρχει γυναίκα ηθοποιός.Αν ναι δεν τα θέλω στο αποτέλεσμα μου(επιτυγχάνεται με το NOT IN).έπειτα γίνεται εμφάνιση των τίτλων των ταινιών που πληρούν τα κριτήρια που προ ανέφερα.

Indexes επέλεξα ίδια με το 1$^{ο}$ query για τους λόγους που ανέφερα και πιο πάνω.




select title
from movies
where mid in(select distinct movies.mid from movies,roles
where movies.mid = roles.mid and movies.mid not in(select mid from actors,roles
where gender ='F' and actors.aid = roles.aid)) (no indexes)






Αθροιστικά τα reads του 2ου query είναι λιγότερα από το 1ο οπότε και θα το προτιμήσω.

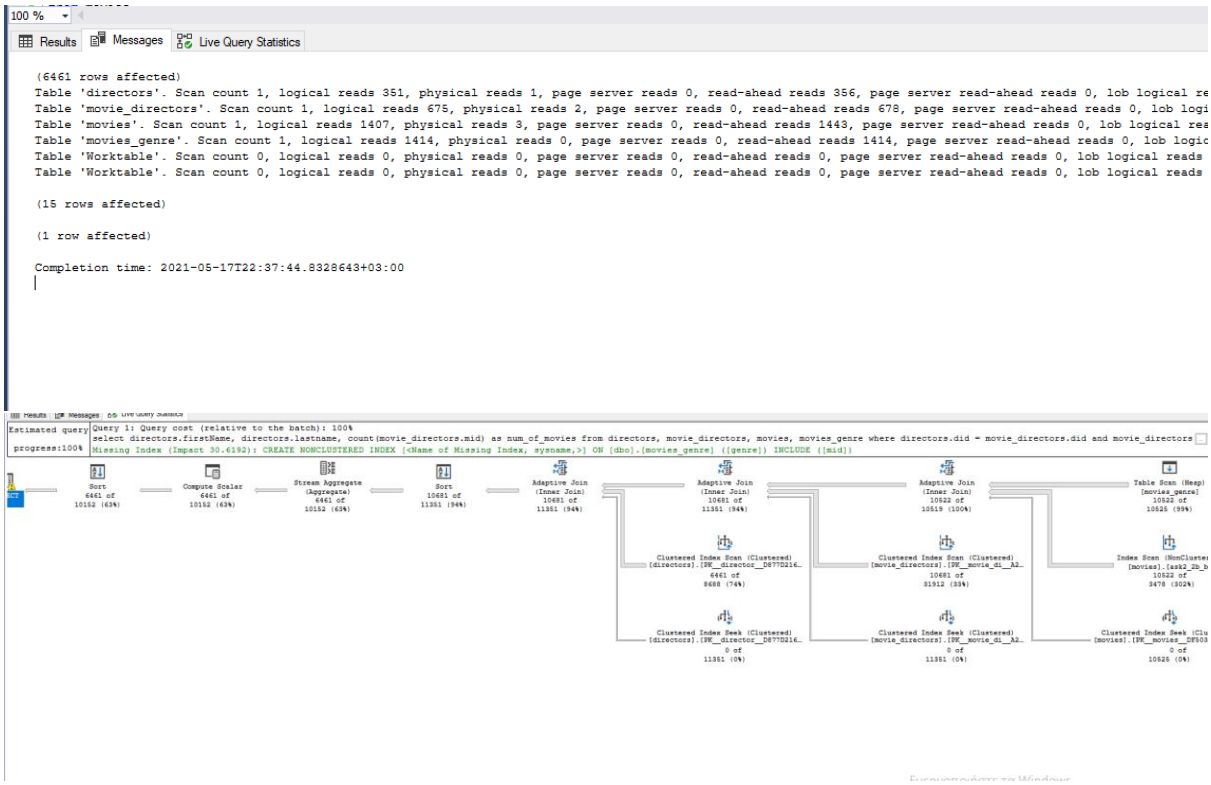
## Ζήτημα 3

«Συνολικός αριθμός ταινιών που έχει σκηνοθετήσει κάποιος σκηνοθέτης ,όπου η ταινία είναι τύπου ‘Thriller’ ”

```
select directors.firstName, directors.lastname, count(movie_directors.mid) as
num_of_movies
from directors, movie_directors, movies, movies_genre
where directors.did = movie_directors.did and movie_directors.mid = movies.mid and
movies.mid = movies_genre.mid and movies_genre.genre = 'Thriller'
```

```
group by directors.firstName, directors.lastname
        order by num_of_movies DESC
```

Τα reads και το πλάνο πριν την δημιουργία indexes:

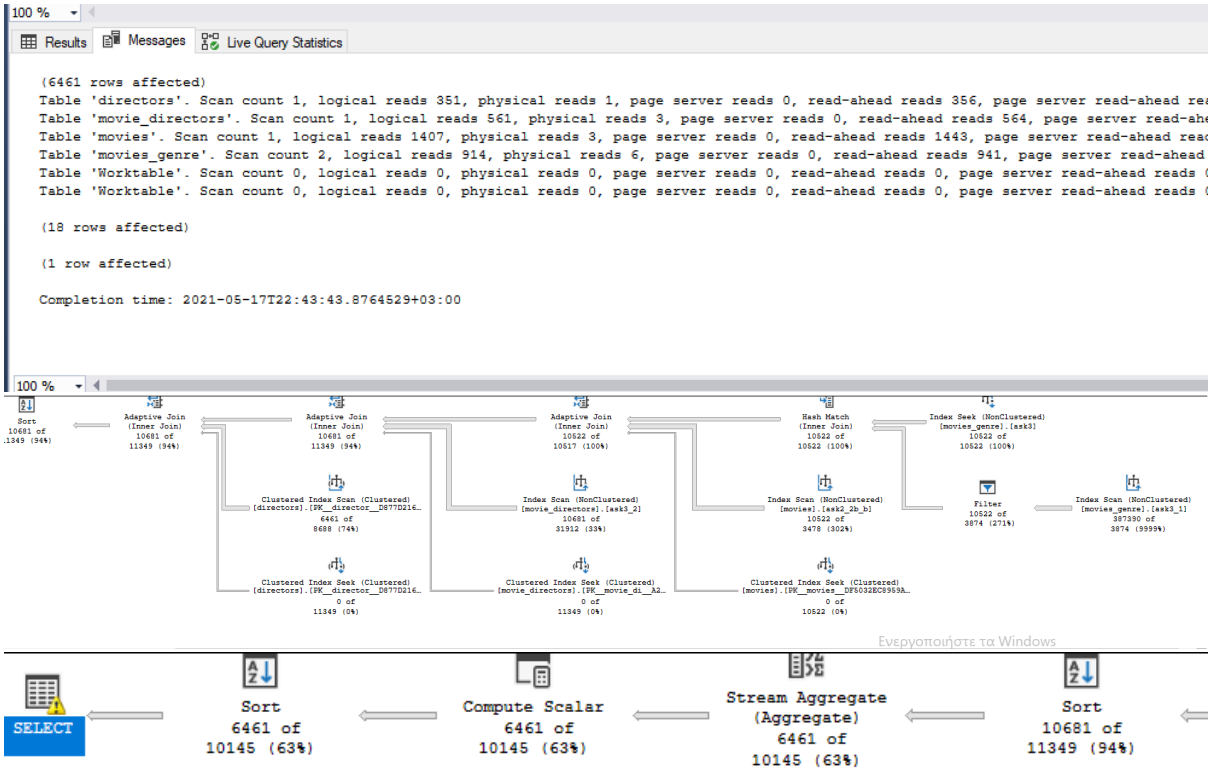



Δημιουργώ τα ευρετήρια:

```
create index ask3 on movies_genre(genre)

create index ask3_1 on movies_genre(mid)

create index ask3_2 on movie_directors(did)

create index ask3_3 on directors(lastname)
```

(θεωρώ οτί σε κανονική πρακτική ότι δεν χρειάζονται τόσα ευρετήρια, ειδικά αν το query είναι σπάνιο, για αυτό δεν πρόσθεσα και το directors(firstName) σαν index)

Με τα οποία έχω:



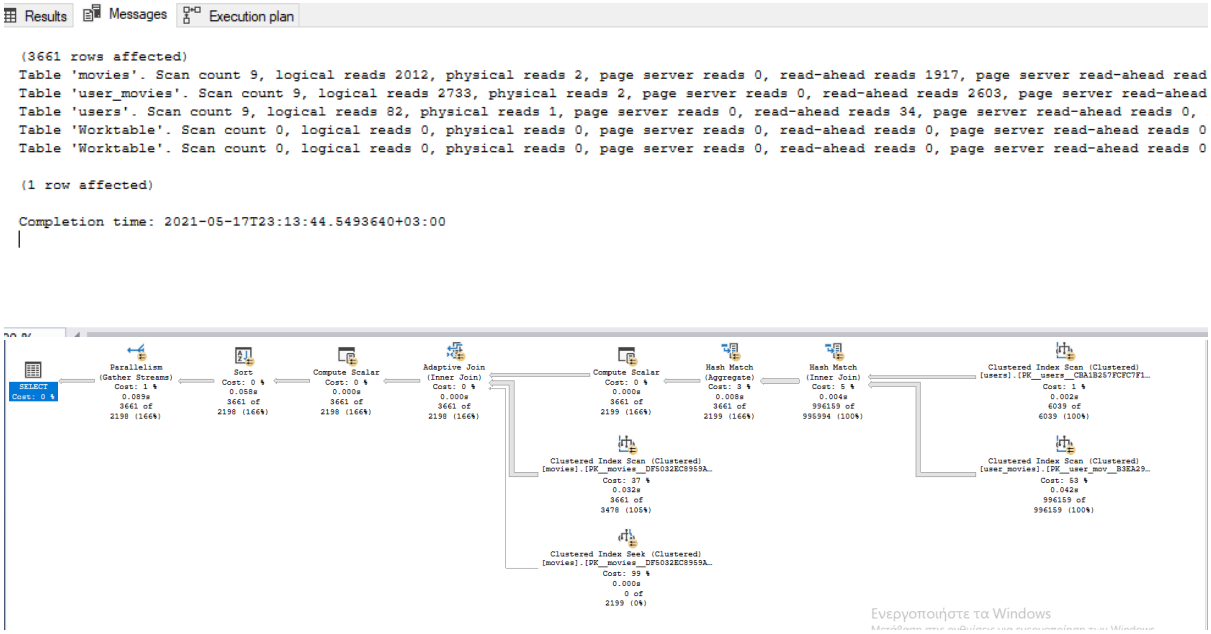

Έχω μείωση στα reads λόγω των ευρετηρίων.

Query 2:

Βρες τον μέσο όρο rating των ταινιών που τους έχει γίνει review και τον συνολικό αριθμό reviews κάθε ταινίας

```
select movies.title, movies.mid, sum(user_movies.rating)/count(user_movies.mid) as
sum_of_ratings, count(user_movies.mid) as total_reviews
from movies, users, user_movies
where users.userid = user_movies.userid and user_movies.mid = movies.mid
group by movies.title, movies.mid
       order by sum_of_ratings desc
```

Χωρίς indexes έχω :



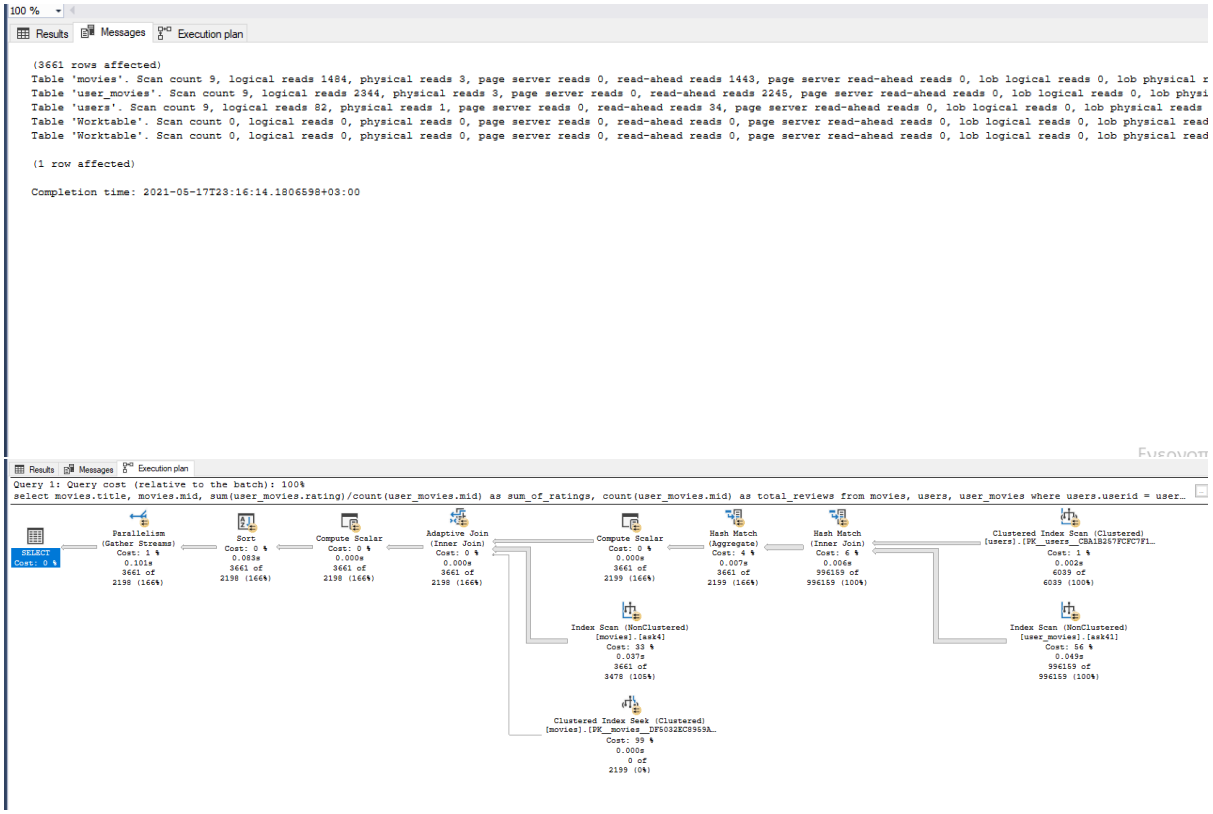




Δημιουργώ τα indexes :

```
create index ask4 on movies(title)
create index ask41 on user_movies(rating)
create index ask42 on user_movies(userid)
```

Με τα οποία έχω:




Με την χρήση των indexes έχω μείωση στα συνολικά reads του ερωτήματος(ειδικά στον πίνακα movies και user_movies)