PROJECT 1

Άσκηση 1:

Φτιάξαμε δύο ευρετήρια συνολικά. Ένα στον πίνακα user_movie που ξεκινάει από το rating, και περιλαμβάνει και το `userid`. Με αυτο τον τρόπο ο server κάνει index seek και πηγαίνει απευθείας στα σημεία που το rating = 5 και δεν χρειάζεται να διαπερνά όλες τις εγγραφές.

Το δεύτερο είναι στον πίνακα user και ξεκινάει από το user id και περιλαμβάνει και το age . Ετσι οταν γίνει το group by η προσπέλαση της ηλικίας είναι πιο γρήγορη.

Οι εντολές δημιουργίας των ευρετηρίων είναι οι εξής:

1) CREATE NONCLUSTERED INDEX usermovie_ratings
   ON user_movies (rating,userid);

2) CREATE NONCLUSTERED INDEX userid_age
   ON users (userid, age);

Μέσα από τα statistics μπορούμε να δούμε οτι χωρίς τα indexes χρειάζεται 78 ms cpu time και elapsed time 108ms

```
SELECT age, count(user_movies.userid)
    FROM users INNER JOIN user_movies ON users.us
  WHERE rating = 5
 Group by age;
```

```
150 %
Results  Messages

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 2 ms.

(7 rows affected)
Table 'user_movies'. Scan count 1, logical reads 2601, physical rea
Table 'users'. Scan count 1, logical reads 29, physical reads 1, pa

 SQL Server Execution Times:
   CPU time = 78 ms,  elapsed time = 108 ms.

Completion time: 2025-04-18T18:06:33.0956308+03:00
```

ενώ με τα indexes cpu time 46ms και elapsed time 34ms.

```
CREATE NONCLUSTERED INDEX usermovie_ratings
ON user_movies (rating,userid);

CREATE NONCLUSTERED INDEX userid_age
ON users (userid, age);

SELECT age, count(user_movies.userid)
    FROM users INNER JOIN user_movies ON users
  WHERE rating = 5
 Group by age;
```

150 %

Results | Messages

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 6 ms.

(7 rows affected)
Table 'users'. Scan count 1, logical reads 13, physical reads 0
Table 'user_movies'. Scan count 1, logical reads 510, physical

 SQL Server Execution Times:
   CPU time = 46 ms,  elapsed time = 34 ms.

Completion time: 2025-04-18T18:06:52.5900370+03:00
```

Άσκηση 2:

Φτιάξαμε ένα ευρετήριο με τα χαρακτηριστικά pyear και title.Αυτό το ευρετήριο καλύπτει και τα τρία ερωτήματα.Συγκεκριμένα σε όλα τα ερωτήματα έχουμε where pyear between 1990 and 2000 που εφαρμόζει πάνω στην πρώτη στήλη του ευρετηρίου .Επίσης το order by pyear,title στο τελευταίο ερώτημα καλύπτεται ακριβώς από την διάταξη του συγκεκριμένου ευρετηρίου οπότε δεν χρειάζεται περαιτέρω ταξινόμηση ή επεξεργασία.
Η εντολή δημιουργίας του ευρετηρίου είναι:

1) CREATE NONCLUSTERED INDEX pyear_tittle
   ON movies(pyear,title)

Μέσα από τα statistics μπορούμε να παρατηρήσουμε ότι χωρίς το ευρετήριο χρειάζεται cpu time = 205 ms και elapsed time = 678 ms.

```
select title from movies where pyear

select pyear, title from movies wher

select title, pyear from movies wher
order by pyear, title
```

```
esults  Messages
 SQL Server Execution Times:
   CPU time = 63 ms,  elapsed time = 636 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

(75693 rows affected)
Table 'Worktable'. Scan count 0, logical reads 0, p
Table 'movies'. Scan count 13, logical reads 2012,

 SQL Server Execution Times:
   CPU time = 205 ms,  elapsed time = 678 ms.

Completion time: 2025-04-19T11:53:36.2937695+03:00
```

Ενώ με το ευρετήριο που δημιουργήσαμε χρειάζεται cpu time = 16 και elapsed time = 593 ms.

```
CREATE NONCLUSTERED INDEX pyear_tittle
ON movies(pyear,title)

select title from movies where pyear betwe

select pyear, title from movies where pyea

select title, pyear from movies where pyea
```

```
150 %
Results  Messages
   SQL Server parse and compile time:
      CPU time = 0 ms, elapsed time = 1 ms.
   SQL Server parse and compile time:
      CPU time = 0 ms, elapsed time = 0 ms.

   (75693 rows affected)
   Table 'movies'. Scan count 1, logical reads 353, physical

    SQL Server Execution Times:
      CPU time = 16 ms,  elapsed time = 593 ms.

   Completion time: 2025-04-19T12:01:05.1218999+03:00
```

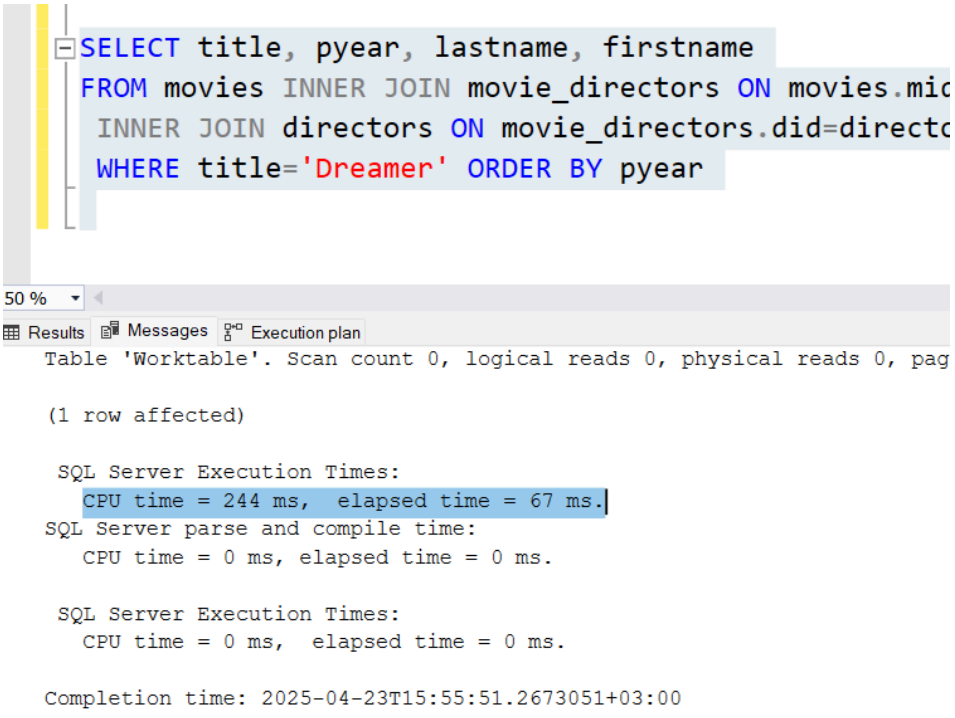
Άσκηση 3:

Όσον αφορά το πρώτο επερώτημα φτιάξαμε ένα ευρετήριο με τα χαρακτηριστικά tittle και pyear.

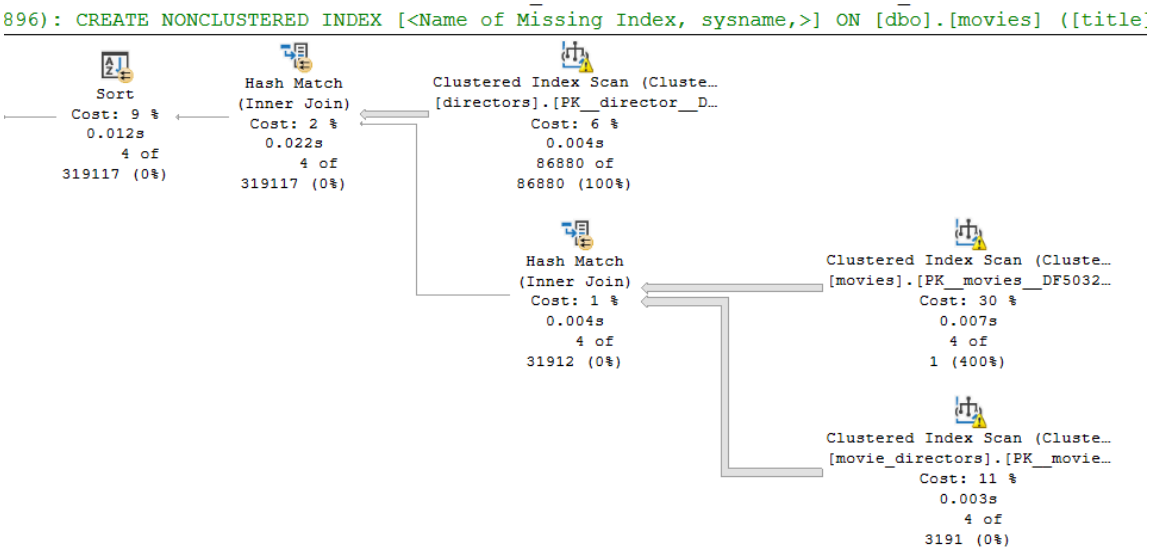
Η εντολή δημιουργίας του ευρετηρίου είναι:

CREATE NONCLUSTERED INDEX title_pyear
ON movies (title, pyear);

Αρχικά μέσα από τα statistics παρατηρούμε οτι CPU time = 244 ms, ενώ elapsed time = 67 ms.



Επισης το αρχικό execution plan χρησιμοποιεί index scan στον πίνακα movies και το cost ειναι 30%.



Μετά την δημιουργία του ευρετηρίου μεσα απο τα statistics παρατηρούμε οτι CPU time = 15 ms και elapsed time = 43 ms .

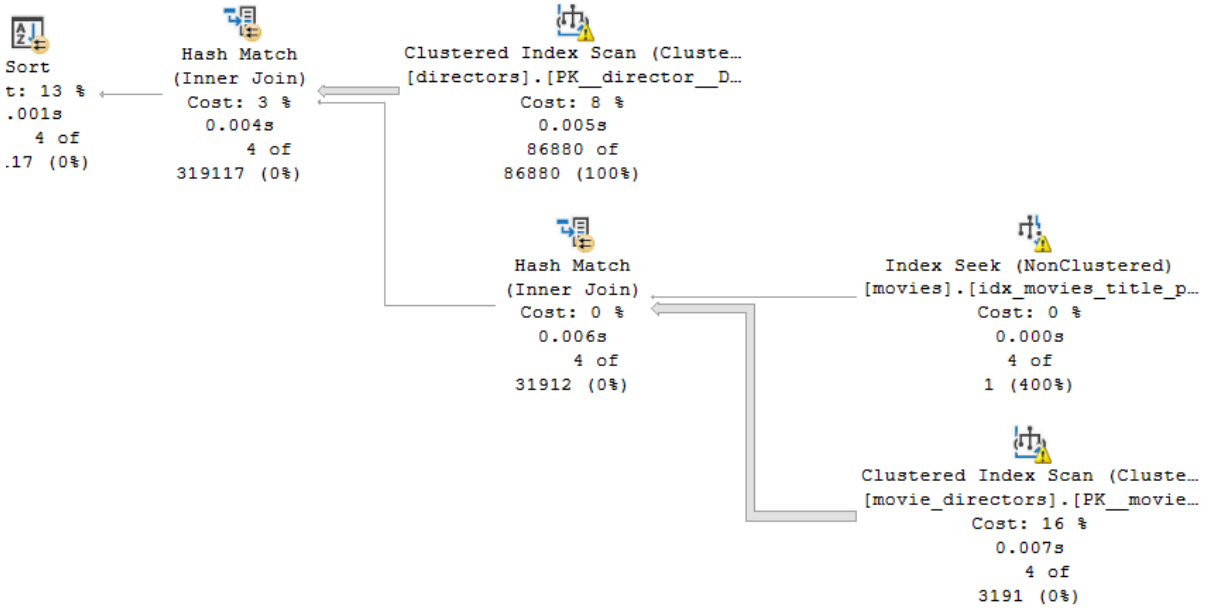
```
FROM movies
INNER JOIN movie_directors ON movies.mid = mov
INNER JOIN directors ON movie_directors.did =
WHERE title = 'Dreamer'
ORDER BY pyear;
```

150 %

Results | Messages | Execution plan

```
Table 'movies'. Scan count 5, logical reads 6, physical reads 0,
Table 'directors'. Scan count 13, logical reads 1048, physical r
Table 'Worktable'. Scan count 0, logical reads 0, physical reads

(1 row affected)

 SQL Server Execution Times:
   CPU time = 15 ms,  elapsed time = 43 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

Completion time: 2025-04-23T15:56:11.9511687+03:00
```

Επίσης βλέπουμε ότι στον Πίνακα movies πλέον χρησιμοποιείται index seek και το cost μειώθηκε στο 0%



Στο δεύτερο επερώτημα φτιάξαμε ένα index όπου περιέχει το lastname και το firstname των directors ωστε να κάνει index seek και οχι scan για να κερδίσει χρόνο στην εντολή where.Επίσης κάνει include to did για να βοηθήσει στο inner join.Μέσα από τα statistics παρατηρούμε ότι αρχικά CPU time = 15 ms και elapsed time = 106 ms.

```
SELECT lastname, firstname, title, pyear
FROM movies INNER JOIN movie_directors ON movies.mid=m
 INNER JOIN directors ON movie_directors.did=directors
 WHERE lastname='Nolan' ORDER BY lastname, firstname
```

```
50 %
Results  Messages  Execution plan
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page se
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page se

(1 row affected)

 SQL Server Execution Times:
   CPU time = 15 ms,  elapsed time = 106 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

Completion time: 2025-04-23T16:49:04.1715570+03:00
```

Η εντολή δημιουργίας του ευρετηρίου είναι:
CREATE NONCLUSTERED INDEX lastname_firstname_did
ON directors (lastname, firstname)
INCLUDE (did);

Μετά το ευρετήριο ,από τα statistics παρατηρούμε ότι αρχικά CPU time = 0 ms και elapsed time = 98 ms. Οπότε το ευρετήριο βοήθησε στην βελτίωση του execution time.

```
CREATE NONCLUSTERED INDEX lastname_firs
ON directors (lastname, firstname)
INCLUDE (did);


SELECT lastname, firstname, title, pyea
FROM movies INNER JOIN movie_directors
 INNER JOIN directors ON movie_director
 WHERE lastname='Nolan' ORDER BY lastna
```

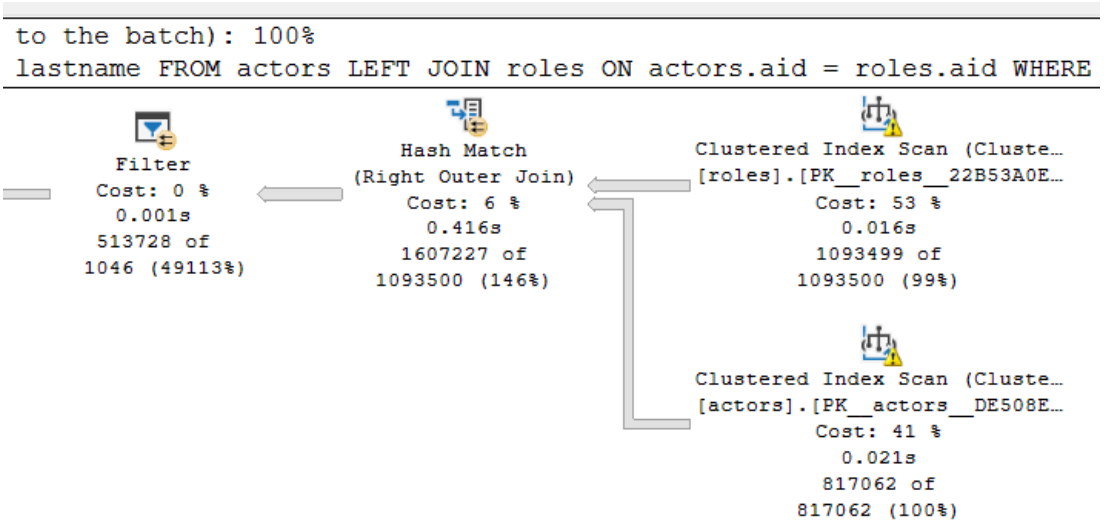
```
.50 %
Results  Messages  Execution plan
Table 'Worktable'. Scan count 0, logical reads 0, physi
Table 'Worktable'. Scan count 0, logical reads 0, physi

(1 row affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 98 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

Completion time: 2025-04-23T17:16:52.4556063+03:00
```

Άσκηση 4:
ΕΡΩΤΗΜΑ Α:
Στο πρώτο επερώτημα φτιάξαμε το εξής ευρετήριο

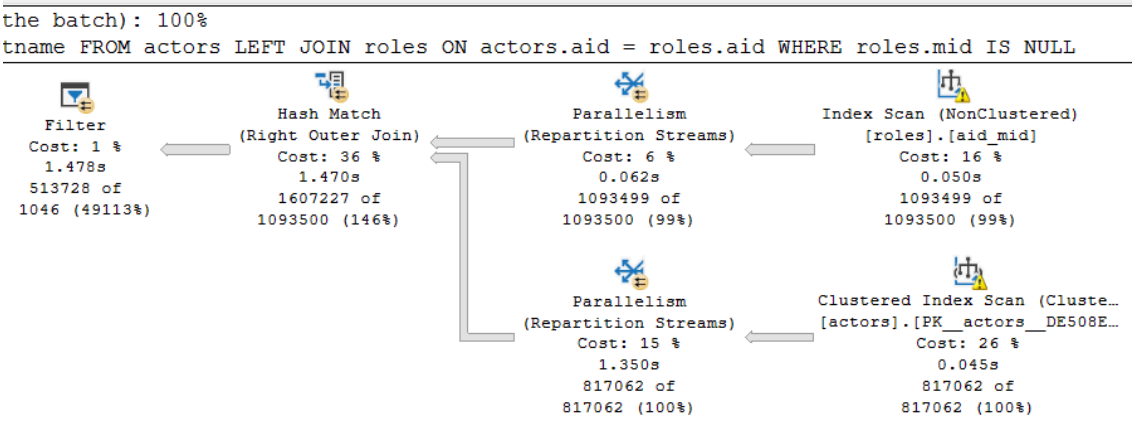
CREATE INDEX aid_mid
ON roles (mid,aid);

Το συγκεκριμένο ευρετήριο βοηθάει στην πιο γρήγορη εύρεση του aid για την υλοποίηση του left join αλλα και στην ταχύτερη αντιστοίχηση του με το mid.

Συγκεκριμένα μέσα από το execution plan μπορούμε να δούμε οτι αρχικά το index scan στον πίνακα roles ήταν clustered και είχε κόστος53%



Μετά το ευρετήριο παρατηρούμε ότι το index scan εγινε clustered και κόστος έγινε 16%



Επίσης από τα στατιστικά παρατηρούμε ότι το execution time έμεινε σχεδόν ίδιο ομως τα logical read πριν ηταν 4489 για τον πινακα roles

```
SELECT actors.aid, firstname, lastname
FROM actors LEFT JOIN roles ON actors.aid = roles.aid
WHERE roles.mid IS NULL;
```

150 %

Results | Messages | Execution plan

```
   CPU time = 0 ms, elapsed time = 8 ms.

(513728 rows affected)
Table 'actors'. Scan count 13, logical reads 3488, physical reads 2, page ser
Table 'roles'. Scan count 13, logical reads 4489, physical reads 1, page serv
```

ενώ μετά το ευρετηριο μειώθηκαν σε 1953

```
CREATE INDEX aid_mid ON roles(aid,mid);

SELECT actors.aid, firstname, lastname
FROM actors LEFT JOIN roles ON actors.aid
WHERE roles.mid IS NULL;
```

150 %

Results | Messages | Execution plan

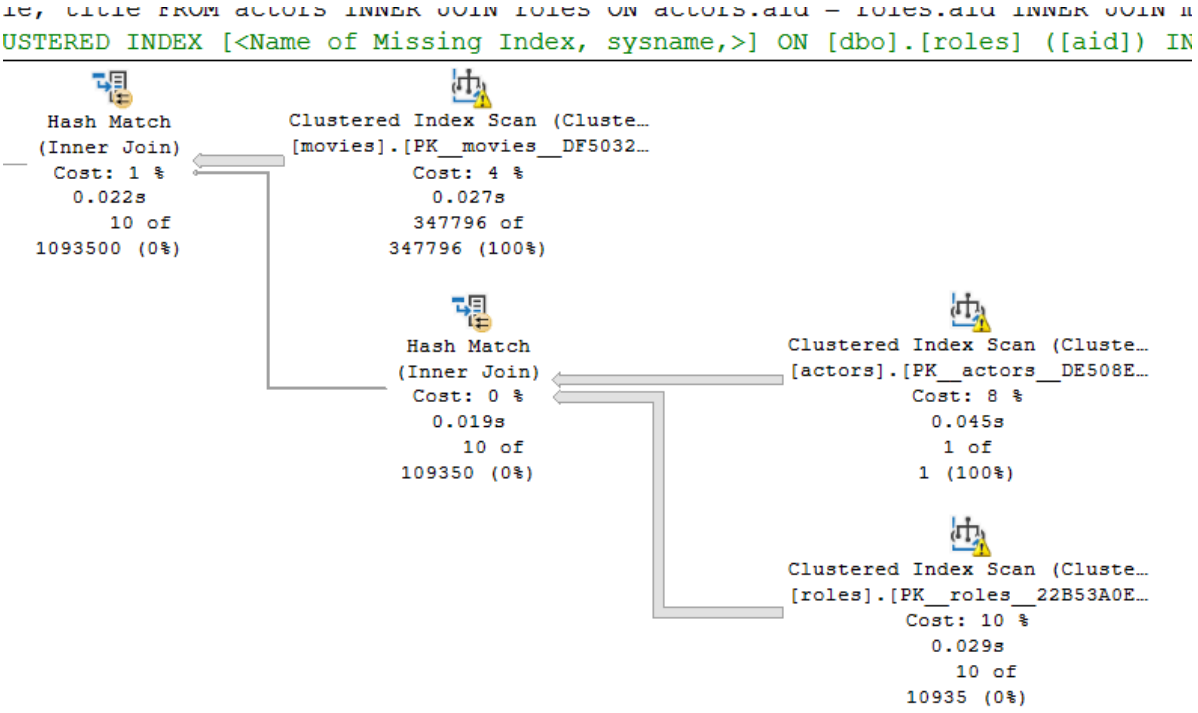
```
Table 'roles'. Scan count 13, logical reads 1948, physica
Table 'actors'. Scan count 13, logical reads 3488, physic
Table 'Workfile'. Scan count 0, logical reads 0, physical
```

Στο δεύτερο επερώτημα δημιουργήσαμε τα παρακάτω ευρετήρια:
CREATE INDEX lastname_firstname_aid
ON actors(lastname, firstname, aid); στον πίνακα actors
και το,

CREATE INDEX mid_title
ON movies(mid, title); στον πίνακα movies

Το ευρετήριο lastname_firstname_aid επιταχύνει το φίλτρο where lastname='Paez' and firstname='Alex' και επιπλέον βοηθάει στην πιο γρήγορη κατάταξη του order by Το ευρετήριο idx_movies_mid_title βοηθά στη επιτάχυνση του inner join με τον πίνακα movies

Αρχικά μέσα από το execution plan παρατηρούμε πως χρησιμοποιείται κυρίως index clustered index scan που καθυστερεί την αναζήτηση.



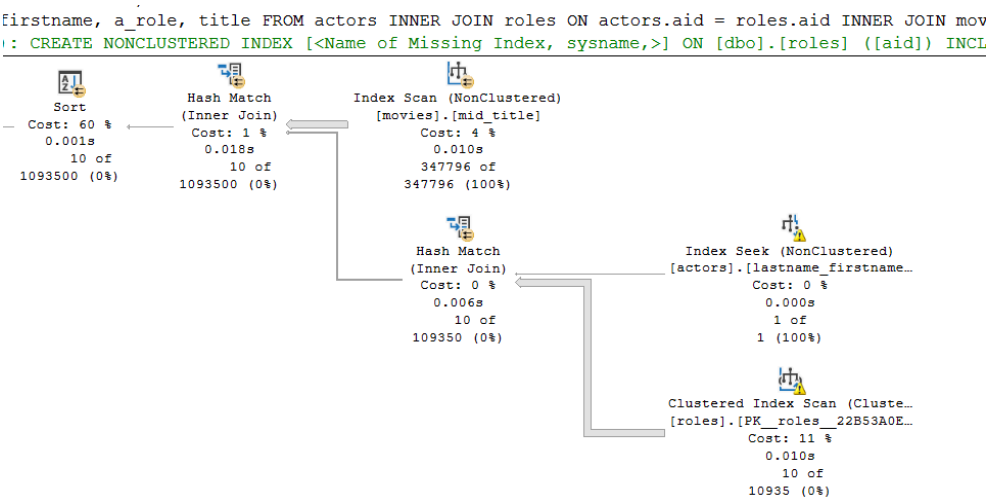
Επίσης μέσα από τα στατιστικά παρατηρούμε οτι CPU time = 78 ms και elapsed time = 156 ms

Table 'roles'. Scan count 13, logical reads 4489,
Table 'actors'. Scan count 13, logical reads 3530,
Table 'movies'. Scan count 13, logical reads 2012

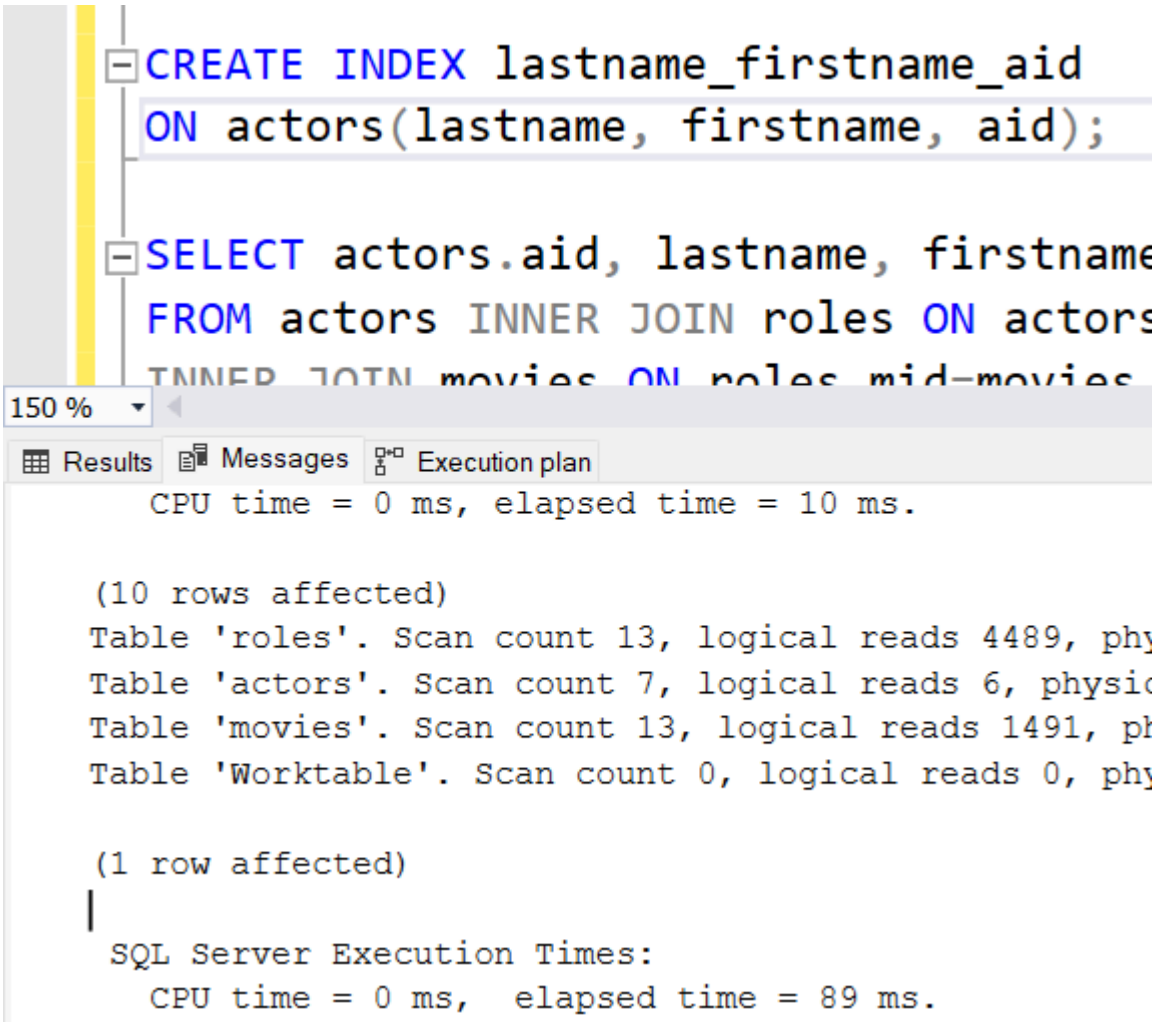
```
SELECT actors.aid, lastname, firstnai
FROM actors INNER JOIN roles ON actoi
 INNER JOIN movies ON roles.mid=movi
WHERE lastname='Paez' and firstname=
ORDER BY lastname, firstname, actors
```

```
.50 %
Results  Messages  Execution plan
      CPU time = 0 ms, elapsed time = 13 ms.

(10 rows affected)
Table 'roles'. Scan count 13, logical reads 4489, p
Table 'actors'. Scan count 13, logical reads 3530,
Table 'movies'. Scan count 13, logical reads 2012,
Table 'Worktable'. Scan count 0, logical reads 0, p

(1 row affected)

 SQL Server Execution Times:
   CPU time = 78 ms,  elapsed time = 156 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
```

Μετά την προσθήκη τον ευρετηρίων βλέπουμε ότι πλέον στον πίνακα actors έχουμε nonclustered index seek και το cost απο 8% μειώνεται σε 0% ενω επιπλεον το index scan στο movies έγινε nonclustered.



Επιπλέον τα logial reads και τα scan count έχουν διαμορφωθεί ως εξής:
Table 'roles'. Scan count 13, logical reads 4489.
Table 'actors'. Scan count 7, logical reads 6,
Table 'movies'. Scan count 13,logical reads 1491
Δηλαδή στον πίνακα actors τα scan counts μειώθηκαν κατα 6 και τα logical reads απο 3530 έγιναν 6
Ενώ στον πίνακα movies τα logical reads απο 2012 μειώθηκαν σε 1491
Τέλος τα CPU time = 78 ms και elapsed time = 156 ms μειώθηκαν σε CPU time = 0 ms, elapsed time = 89 ms αντίστοιχα



ΕΡΩΤΗΜΑ Β:
Στο πρώτο ερώτημα ουσιαστικά ψάχνουμε τους ηθοποιούς που δεν έχουν αναλάβει κανένα ρόλο. Έτσι μπορούμε να αντικαταστήσουμε το αρχικό κώδικα με

```
SELECT actors.aid, firstname, lastname
FROM actors
WHERE NOT EXISTS (
   SELECT 1
   FROM roles
   WHERE roles.aid = actors.aid);
```

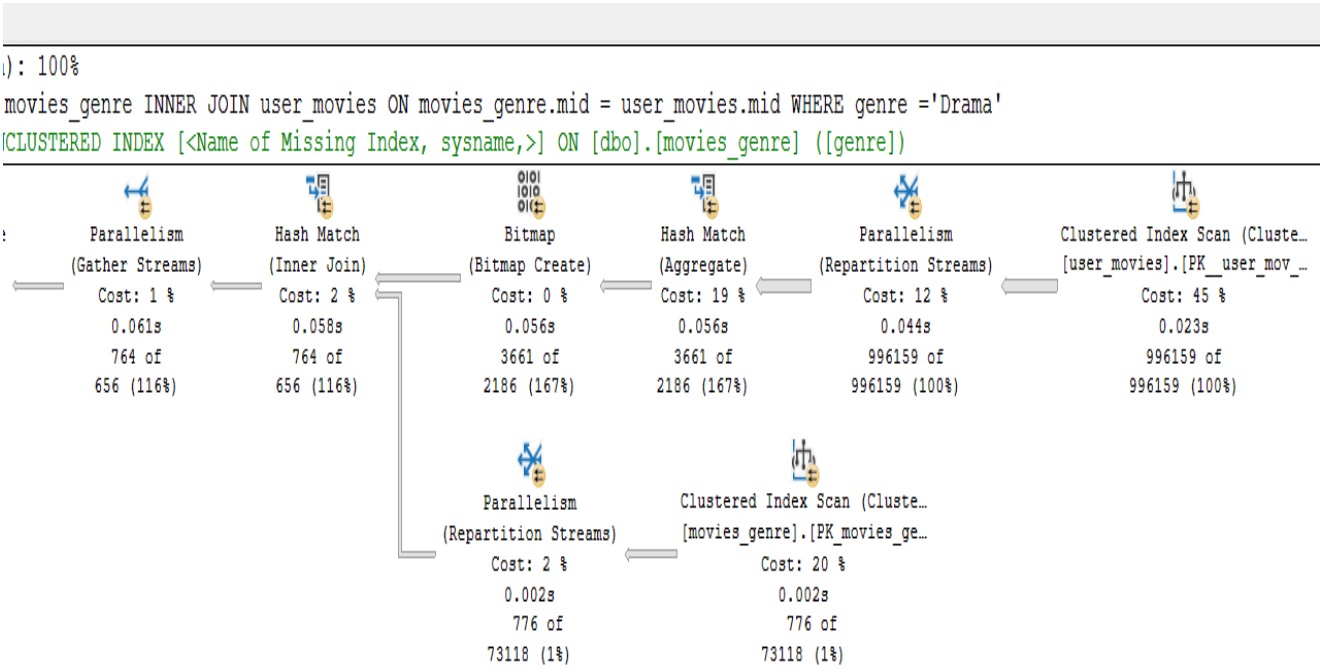
Αν εκτελέσουμε τον κώδικα παρατηρούμε ότι τα αποτελέσματα παραμένουν ίδια αλλά πλεον το ερώτημα υλοποιείται πιο αποδοτικά μετά την τροποποιήση. Συγκεκριμένα αρχικά χρειαζόμασταν cpu time = 221 ms και elapsed time = 2620 ms. Ενώ πλέον οι χρόνοι μειωθηκαν σε cpu time =172 ms και elapsed time = 2185 ms αντίστοιχα.Τα logical reads παραμένουν ίδια.

Άσκηση 5:
Primary key θα ορίσουμε το ζευγάρι (`mid`, `genre`), αφού μία ταινία μπορεί να έχει πολλά genres.

Η εντολή είναι η εξής:

```
ALTER TABLE movies_genre

ADD CONSTRAINT PK_movies_genre PRIMARY KEY (mid, genre);
```

Αρχικά το ερώτημα χρειάζεται CPU time = 250 ms και elapsed time = 78 ms και έχει αυτό το execution plan:

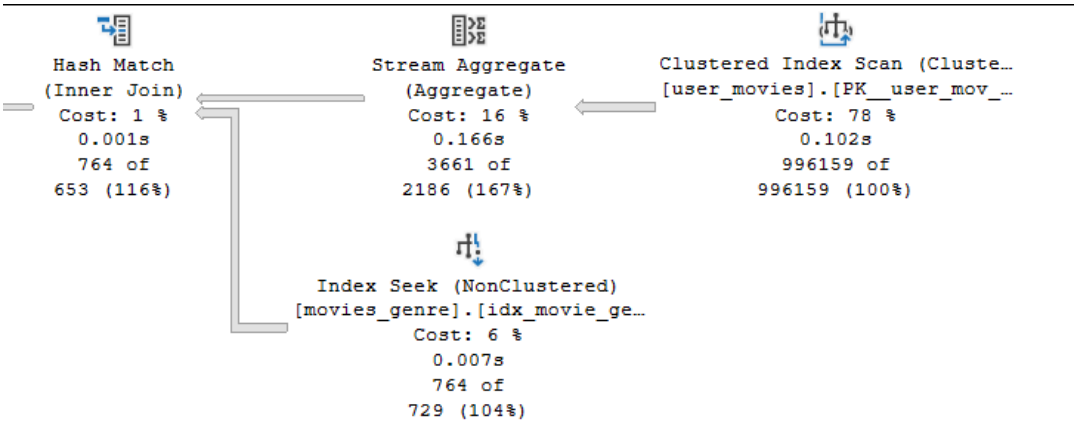


Στην συνέχεια δημιουργούμε το εξής ευρετήριο:

CREATE INDEX idx_movie_genre

ON movies_genre(genre);

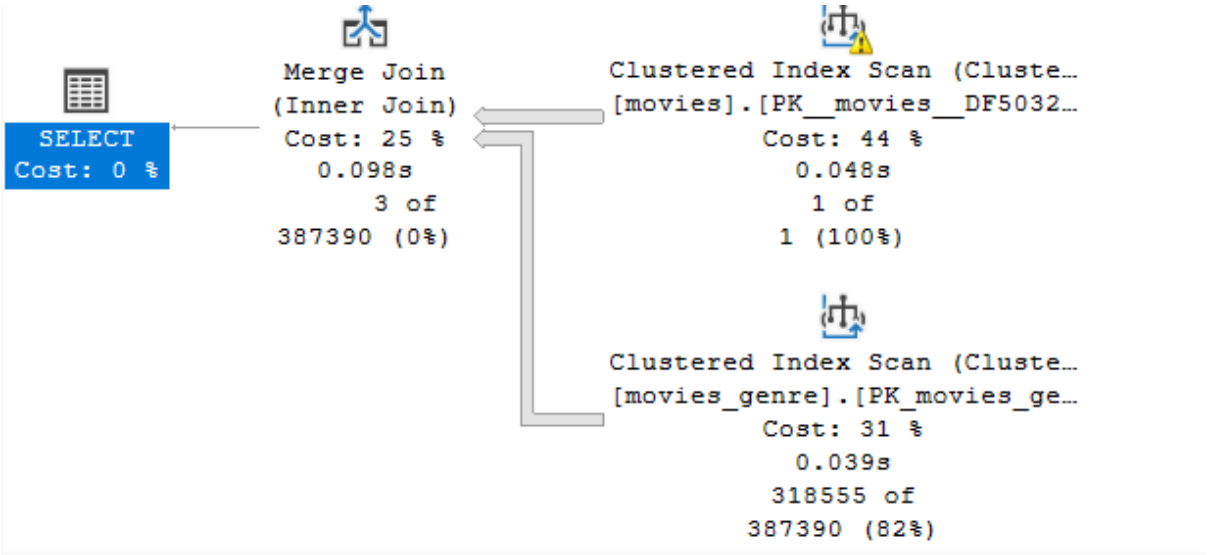
Πλεον το ερώτημα χρειάζεται,CPU time = 141 ms και elapsed time = 167 ms που όπως βλέπουμε ο χρόνος είναι σχετικά μειωμένος .Επίσης το execution plan άλλαξε.Πλέον στον πίνακα movies_genre δεν κάνουμε index scan αλλά Index seek και το κόστος μειώνεται απο 20% σε 6% και το execution plan έγινε έτσι.



Στην συνέχεια στο δεύτερο επερώτημα παρατηρούμε ότι χρειαζόμαστε αρχικά CPU time = 78 ms και elapsed time = 125 ms.

Έπειτα δημιουργούμε το ευρετήριο:
CREATE INDEX movies_title
ON movies (title);
Μετά την δημιουργία του ευρετηρίου οι χρόνοι μειώνονται σε:
CPU time = 31 ms, elapsed time = 72 ms.

Επίσης παρατηρούμε ότι από το execution plan οτι αρχικά στον πίνακα movie χρησιμοποιείται index scan με κόστος 44%



πλεον στο execution plan, στον πίνακα movies χρησιμοποιείται index seek και το κόστος έχει μειωθεί σε 0%