

**ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ**



**ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS**

Ακαδημαϊκό έτος: 2020-2021

Μάθημα: Σχεδιασμός Βάσεων Δεδομένων

Project Part A (Πρώτη Εργασία)

Όνομ/πώνυμο: Βασίλειος-Ηλίας Δρούζας

ΑΜ: 3180051

Σημείωση: Αρχικά ενεργοποιούμε τα στατιστικά με την εντολή

SET STATISTICS IO ON

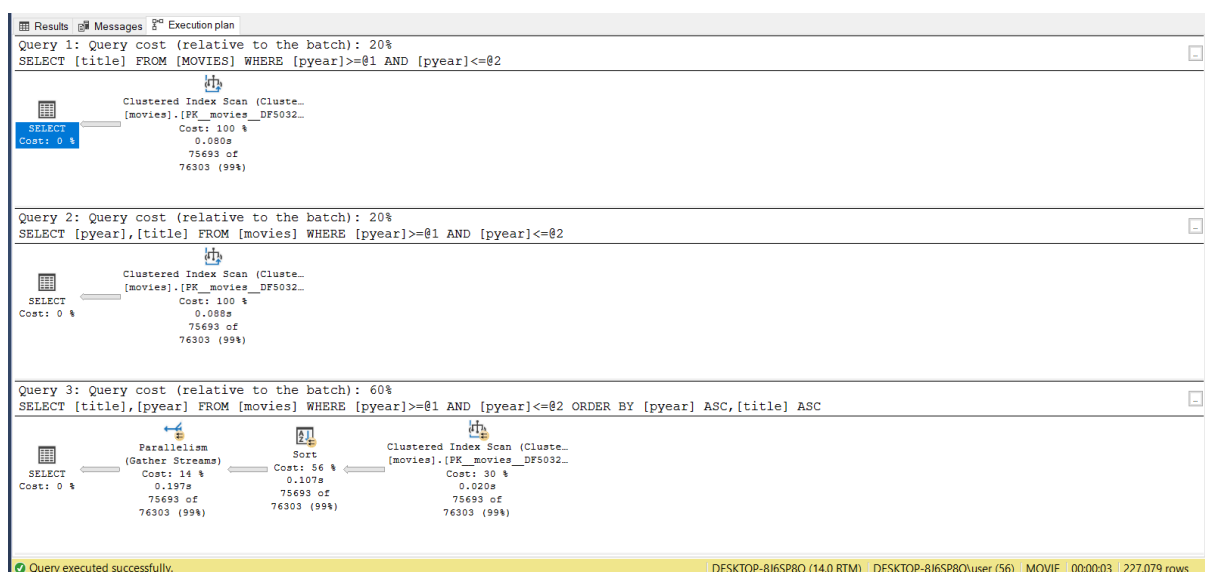
και πριν τρέξουμε κάθε query, καθαρίζουμε τους buffers

```
checkpoint  
dbcc dropcleanbuffers
```

Επίσης, κατά την διαδικασία βελτιστοποίησης, δίνω έμφαση στην ελαχιστοποίηση των **logical reads**, καθώς είναι αυτά που πιο εύκολα μειώνονται αν αλλάξουμε κατάλληλα το query ή προσθέσουμε τα κατάλληλα indexes. Τα physical reads και τα read-ahead reads είναι κάπως πιο δύσκολο να ελεγχθούν, εάν καταφέρουμε να τα μειώσουμε όμως θα είναι σίγουρα ευπρόσδεκτο.

Ζήτημα 1^ο

1.Πριν την προσθήκη ευρετηρίων έχουμε τα execution plans ως εξής:



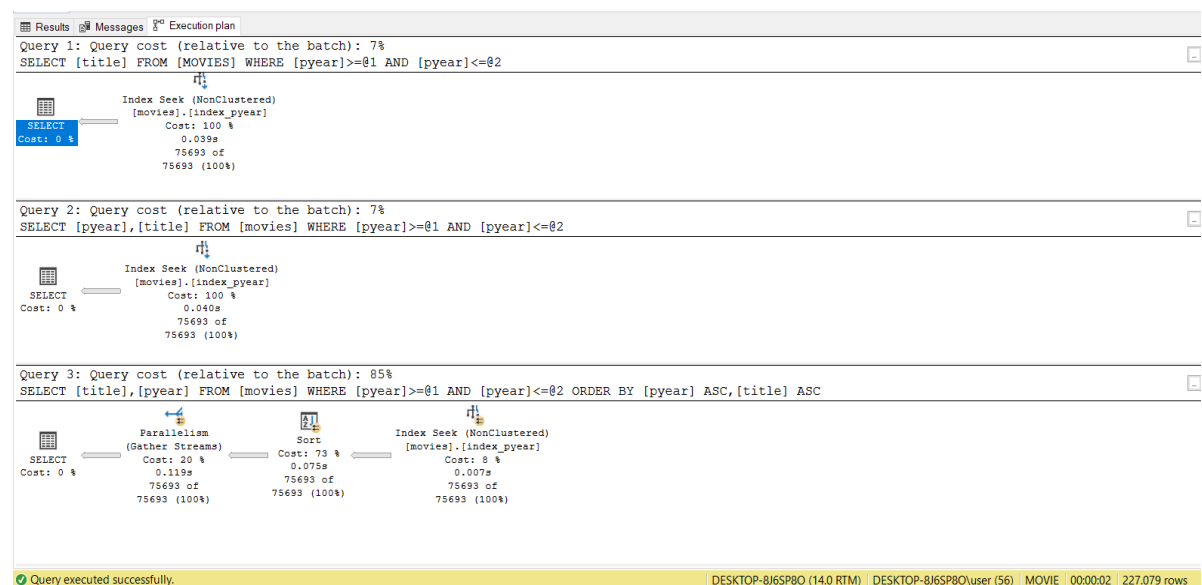
Έχουμε λοιπόν Clustered Index Scan και στα τρία ερωτήματα.

Θα χρησιμοποιήσουμε το ευρετήριο:

-

```
CREATE INDEX index_pyear ON movies(pyear) INCLUDE (title)
```

Μετά την προσθήκη του ευρετηρίου στο **pyear**, τα execution plans αλλάζουν ως εξής:



Και η σύγκριση των reads:

Χωρίς ευρετήριο:

Query	Logical Reads	Physical Reads	Read-ahead Reads
#1	1918	2	1917
#2	1918	0	0
#3	2012	0	0

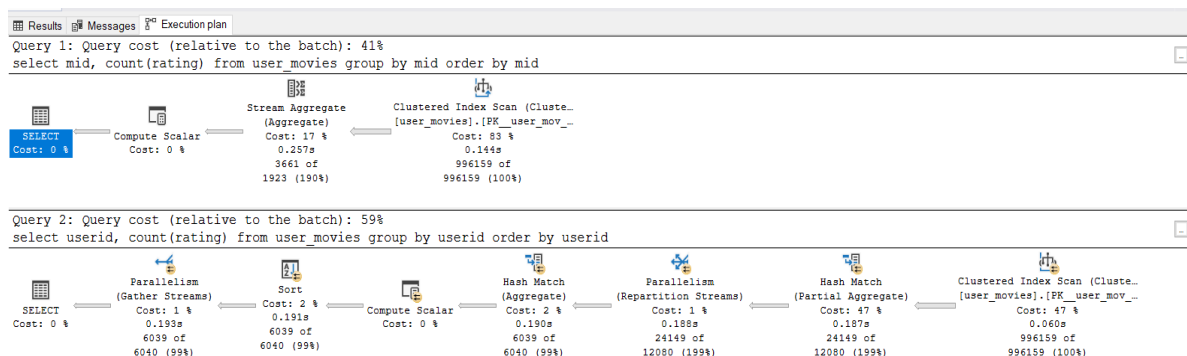
Με ευρετήριο:

Query	Logical Reads	Physical Reads	Read-ahead Reads
#1	351	2	354
#2	351	0	0
#3	370	0	0

Παρατηρούμε ότι μειώθηκε δραματικά ο αριθμός των logical reads, όπως και ο αριθμός των read-ahead reads για το 1^ο query. Από τα πλάνα εκτέλεσης είδαμε ότι πλέον ο SQL server εφαρμόζει Index Seek αντί για Clustered Index Scan, οπότε η εκτέλεση έχει βελτιωθεί.

2.

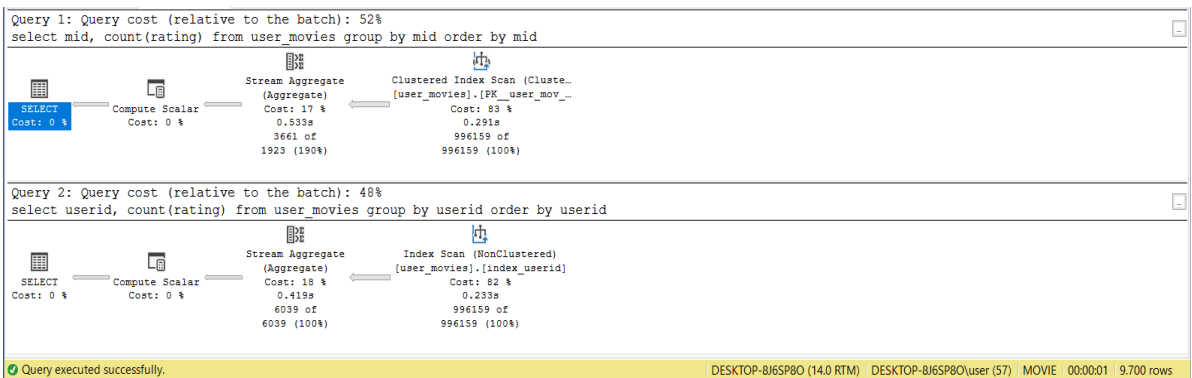
Πριν τη προσθήκη του ευρετηρίου έχουμε:



Έχουμε το περιορισμό ότι μπορούμε να δημιουργήσουμε ένα μόνο ευρετήριο, οπότε μια καλή τακτική που θα μπορούσαμε να ακολουθήσουμε είναι να χρησιμοποιήσουμε ευρετήριο στη στήλη που περιέχει τις περισσότερες εγγραφές. (εδώ έχουμε 3661 για το mid και 6039 για το userid, επομένως θα δημιουργήσουμε το ευρετήριο) :

```
CREATE INDEX index_userid ON user_movies(userid) INCLUDE (mid,rating)
```

Πλέον τα πλάνα εκτέλεσης αλλάζουν ως εξής:



Η σύγκριση των reads:

Χωρίς ευρετήριο:

Query	Logical Reads	Physical Reads	Read-ahead Reads
#1	2601	2	2603
#2	2733	0	0

Με ευρετήριο:

Query	Logical Reads	Physical Reads	Read-ahead Reads
#1	2601	2	2603
#2	2230	1	2244

Παρατηρούμε ότι ο αριθμός των logical reads δεν μειώθηκε για το πρώτο query, έχουμε όμως βελτίωση για το δεύτερο. Το σημαντικό εδώ είναι ότι αθροιστικά παρότι ο χρόνος επεξεργασίας αυξήθηκε λίγο για το πρώτο query (0.489s αντί 0.401s χωρίς το ευρετήριο), βελτιώθηκε σημαντικά (0.484s αντί 1.241s χωρίς ευρετήριο) για το δεύτερο. Παράλληλα στο πλάνο εκτέλεσης βλέπουμε ότι απαλείφθηκαν οι τελεστές hash και parallelism που έπαιρναν σημαντικό μέρος του χρόνου.

Συνολικά, αν θέλουμε να διατηρούμε καλή απόδοση και για τα δύο queries, χρησιμοποιώντας ευρετήριο στο userid μπορούμε να κρατήσουμε καλό χρόνο και για τα δύο.

Ζήτημα 2ο

1. Το query της εκφώνησης επιστρέφει 19.276 στήλες.

Αλλάζουμε το query:

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

UNION ALL

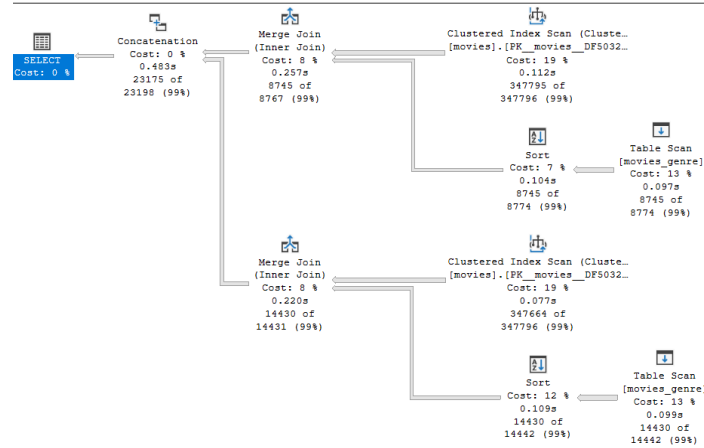
select title
from movies, movies_genre
where movies.mid=movies_genre.mid and genre ='Action'
```

Αντί του τελεστή UNION, μπορούμε να χρησιμοποιήσουμε τον τελεστή UNION ALL για καλύτερη απόδοση. Ο τελεστής UNION ALL επιστρέφει και τις διπλότυπες εγγραφές, επομένως δεν χάνει επιπλέον χρόνο για τον έλεγχο διπλότυπων.

Το νέο query λοιπόν επιστρέφει 23.175 στήλες.

Αρχικό Πλάνο εκτέλεσης

Query 1: Query cost (relative to the batch): 100%
 select title from movies, movies_genre where movies.mid=movies_genre.mid and genre ='Adventure' UNION ALL select title from movies, movies_genre whe...
 Missing Index (Impact 13.9058): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[movies_genre] ([genre]) INCLUDE ([mid])



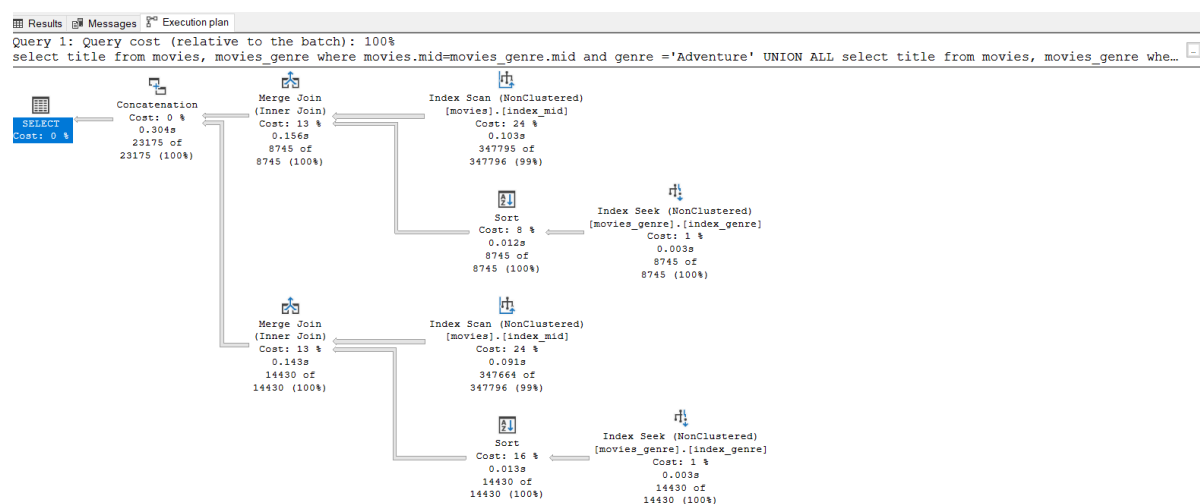
Query executed successfully. DESKTOP-8J6SP8O (14.0 RTM) DESKTOP-8J6SP8O\user (53) MOVIE 00:00:00 23.175 rows

Χρησιμοποιούμε τα ευρετήρια:

CREATE INDEX index_genre ON Movies_Genre(genre) INCLUDE (mid)

CREATE INDEX index_mid ON Movies(mid) INCLUDE (title)

Πλάνο εκτέλεσης μετά την προσθήκη ευρετηρίων



Query executed successfully. DESKTOP-8J6SP8O (14.0 RTM) DESKTOP-8J6SP8O\user (53) MOVIE 00:00:00 23.175 rows

Η σύγκριση των reads:

Χωρίς ευρετήριο:

Table	Logical Reads	Physical Reads	Read-ahead Reads
Movies_Genre	2246	0	1123
Movies	3836	2	1917

Με ευρετήριο:

Table	Logical Reads	Physical Reads	Read-ahead Reads
Movies_Genre	91	1	84
Movies	2798	3	1409

Βλέπουμε ότι τα Table Scan που είχαμε αρχικά για το genre, τα αντικαταστήσαμε με Index Seek και χρησιμοποιείται ύστερα index scan αντί Clustered Index Scan στο mid. Επίσης ο αριθμός των logical reads μειώθηκε και για τους δύο πίνακες, και μάλιστα εντυπωσιακά για τον Movies_Genre.

2.

Query #1:

```
SELECT DISTINCT title, Movies.mid
FROM Movies, Actors, Roles
WHERE Actors.aid=Roles.aid
AND Roles.mid=Movies.mid
GROUP BY title, Movies.mid
HAVING SUM(case when Actors.gender='F' then 1 else 0 end)=0
INTERSECT
SELECT title, Movies.mid
FROM Movies, Actors, Roles
WHERE Actors.aid=Roles.aid
AND Roles.mid=Movies.mid
GROUP BY title, Movies.mid
HAVING SUM(case when Actors.gender='M' then 1 else 0 end)>0
```

Query #2: (ουσιαστικά είναι μια παραλλαγή του query 1)

```
SELECT DISTINCT title, Movies.mid
FROM Movies, Actors, Roles
WHERE Actors.aid=Roles.aid
AND Roles.mid=Movies.mid
GROUP BY title, Movies.mid
HAVING SUM(case when Actors.gender='F' then 1 else 0 end)=0
AND SUM(case when Actors.gender='M' then 1 else 0 end)>=1
```

Query #3:

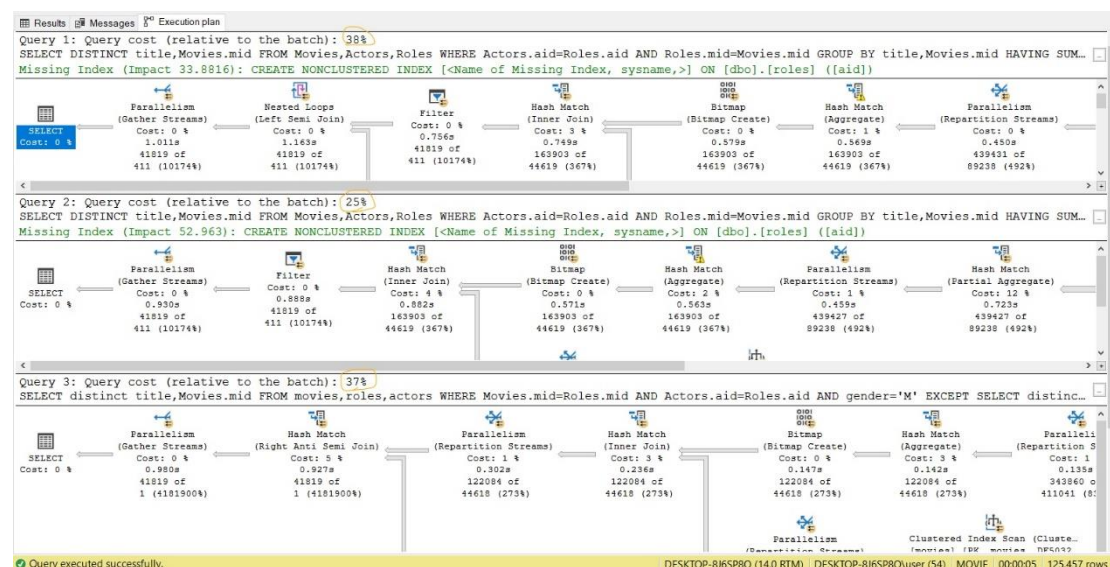
```

SELECT distinct title,Movies.mid
FROM movies,roles,actors
WHERE Movies.mid=Roles.mid
AND Actors.aid=Roles.aid
AND gender='M'
EXCEPT
SELECT distinct title,Movies.mid
FROM movies,roles,actors
WHERE Movies.mid=Roles.mid
AND Actors.aid=Roles.aid
AND gender='F'

```

Και τα τρία queries επιστρέφουν τον ίδιο αριθμό στηλών (41.819).

Τρέχοντας τα τρία queries σε batch, το ποσοστό κόστους για τα Queries 1,2,3 είναι 38%,25% και 37% αντίστοιχα, όπως φαίνεται και στην εικόνα:



Επομένως θα επιλέξουμε το Query #2, μιας και είναι το πιο γρήγορο.

Το query αυτό επιστρέφει τα αποτελέσματα:

Results

Messages

Execution plan

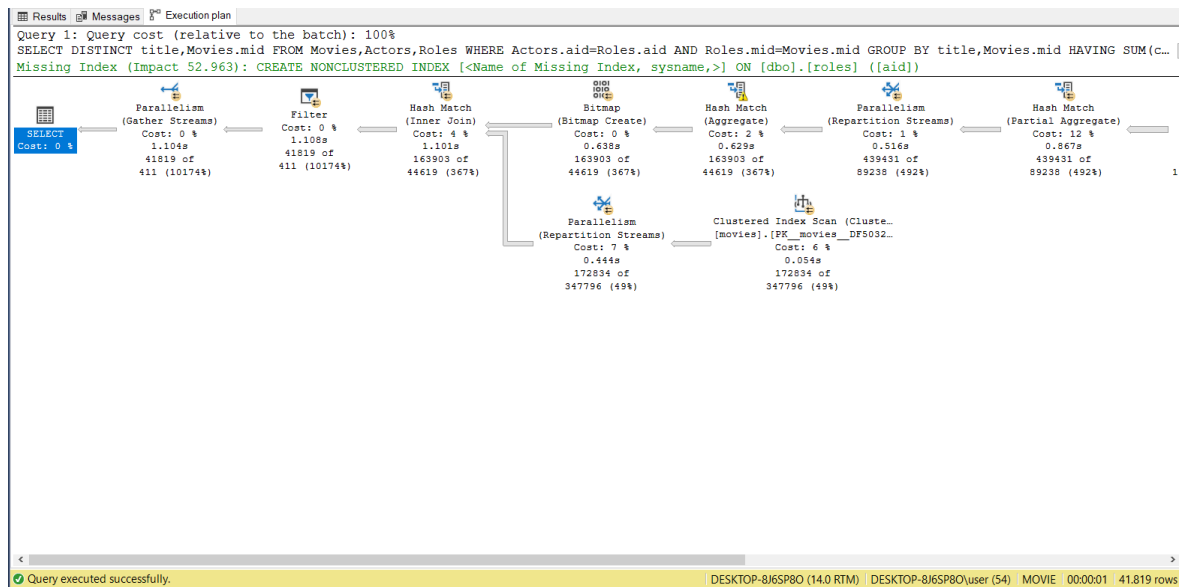
	title	mid
1	Along the Oregon Trail	12355
2	Alpamyss idyot v zhkolu	12374
3	Alpha Caper, The	12389
4	Alpha City	12391
5	Alpha Incident, The	12392
6	Alpine Climbers	12423
7	Alster-Tanz	12507
8	Altalena	12542
9	Alte Liebe, alte Snde	12576
10	Altman on His Own T...	12656
11	Altra donna, L' (2002/)	12675
12	Altri uomini	12690
13	Altimenti ci arrabbia...	12692
14	Aluohan shen shou	12723
15	Alyas 1 2 3	12790
16	Alyas Batman en Ro...	12793
17	Am I Cursed?	12858

Query executed successfully.

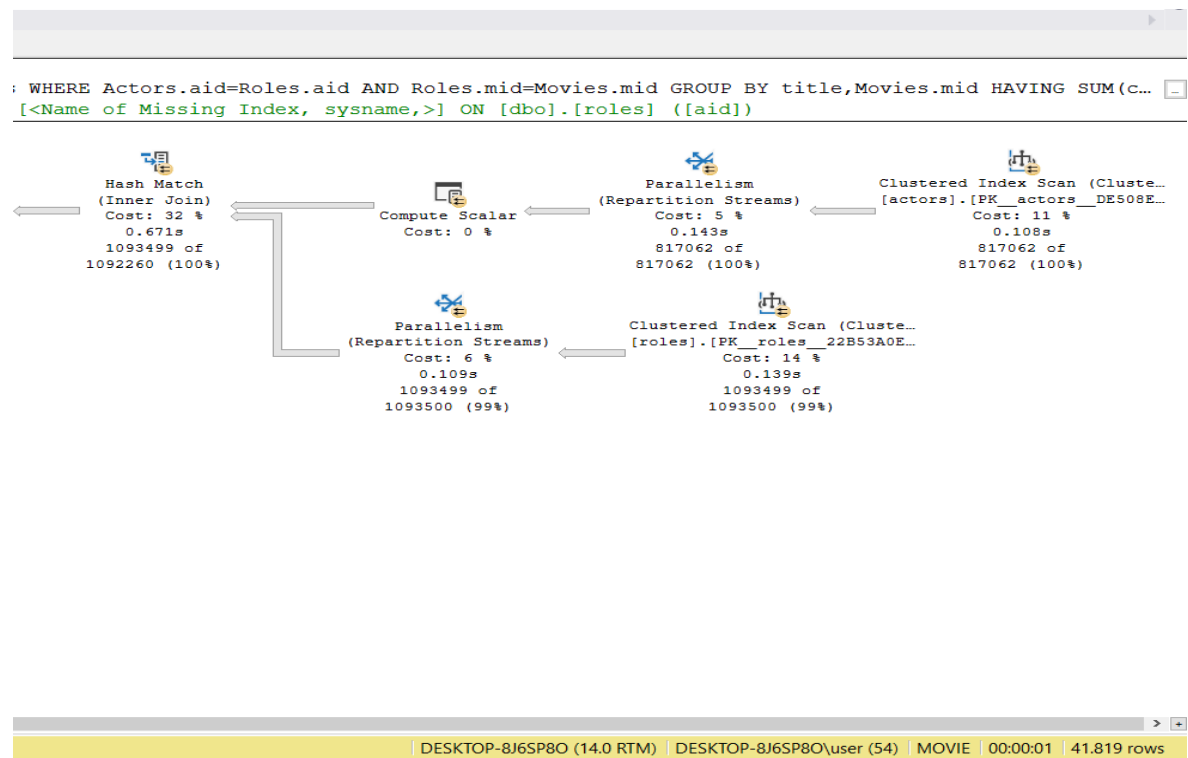
DESKTOP-8J6SP80 (14.0 RTM) | DESKTOP-8J6SP80\user (52) | MOVIE | 00:00:02 | 41.819 rows

Αρχικό Πλάνο Εκτέλεσης

(Ακολουθούν 2 εικόνες,για καλύτερη ανάλυση,καθώς το execution plan ήταν μεγάλο)



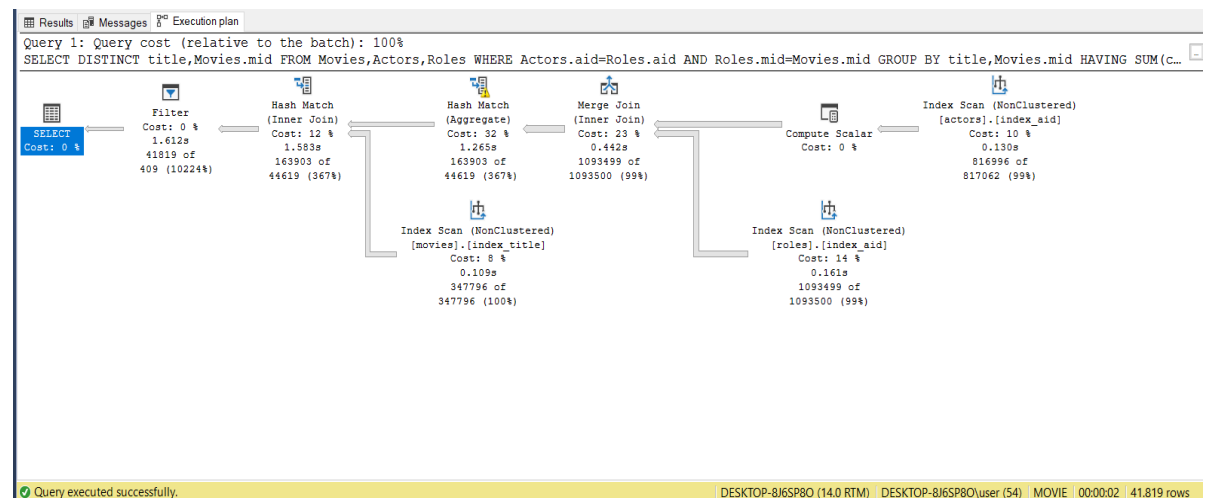
και η συνέχεια:



Προσθέτουμε τα ευρετήρια:

```
CREATE NONCLUSTERED INDEX index_aid ON Roles(aid) INCLUDE(mid)
CREATE INDEX index_title ON Movies(title) INCLUDE(mid)
CREATE INDEX index_aid ON Actors(aid) INCLUDE (gender)
```

Το πλάνο εκτέλεσης αλλάζει ως εξής:



Η σύγκριση των reads:

Χωρίς ευρετήριο:

Table	Logical Reads
Actors	2012
Roles	4489
Movies	3530

Με ευρετήριο:

Table	Logical Reads
Actors	1405
Roles	1901
Movies	1117

Στο πλάνο εκτέλεσης καταφέραμε να μετατρέψουμε τα Clustered Index Scan σε Index Seek, ενώ απαλείψαμε και τελεστές Parallelism και Bitmap. Ο χρόνος επεξεργασίας μειώθηκε με τη χρήση ευρετηρίου στα 5.302s (από 7.054s). Παράλληλα τα logical reads μειώθηκαν σε όλους τους πίνακες.

Ζήτημα 3^ο

(στην άσκηση αυτή τα ζητούμενα 1 και 2 λύθηκαν μαζί για κάθε query)

Query #1:

```
--1. Δείξε τις ταινίες οι οποίες έχουν άριστη βαθμολογία (>=8.5) σε φθίνουσα σειρά
-- με θέμα 'Action' και λιγότερους από 6 ηθοποιούς.

SELECT Movies.title,COUNT(Actors.aid) as Number_of_Actors,mrank
FROM Movies,Roles,Movies_genre,Actors
WHERE Movies.mid=Movies_Genre.mid
AND Movies.mid=Roles.mid
AND Actors.aid=Roles.aid
AND Movies_Genre.genre='Action'
AND Movies.mrank>=8.5
GROUP BY title,Movies.mrank
HAVING COUNT(Actors.aid)<6
ORDER BY mrank DESC
```

Το παραπάνω query επιστρέφει τους τίτλους των ταινιών, τον αριθμό των ηθοποιών και τη βαθμολογία της ταινίας:

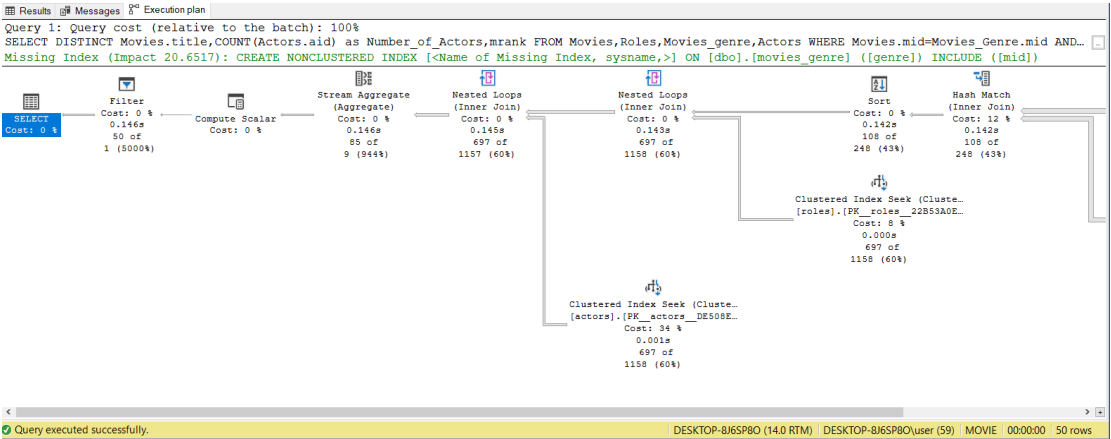
	title	Number_of_Actors	mrank
1	Cleaning, The	3	9.9
2	Suspension of Disbelief	3	9.8
3	Taste of Paradise	5	9.8
4	Marc Maron, agente S. 077	4	9.6
5	Iancu Jianu, zapciul	1	9.5
6	Warpath	3	9.5
7	Cerebral Print File #0604	3	9.4
8	Dark Mist, The	2	9.4
9	Okkadu	2	9.4
10	Zhuang dao zheng	1	9.4
11	Apradh	1	9.2
12	Black Gulch	4	9.2
13	Kaliman	2	9.2
14	Revenge of Cerebral Print	2	9.2
15	Speed Merchants, The	1	9.2
16	Anderson Unbound	3	9.1
17	Nan quan bei hui dou jin hu	2	9.1
18	Pratiya	1	9.1
19	24	2	9.0
20	A La Cart	2	9.0
21	Chanoc y el hijo del Sant...	2	9.0
22	Evening in Paris, An	2	9.0
23	Hibotan bakuto: hanafud...	3	9.0
24	Majing Zetto tai Ankoku D...	4	9.0
25	Sabogou 009	1	9.0
26	Wilhelm Cucenitorul	1	9.0
27	Cut and Run	5	8.9

Query executed successfully. DESKTOP-8J6SP8O (14.0 RTM) DESKTOP-8J6SP8O(user: 55) MOVIE 00:00:02 50 rows

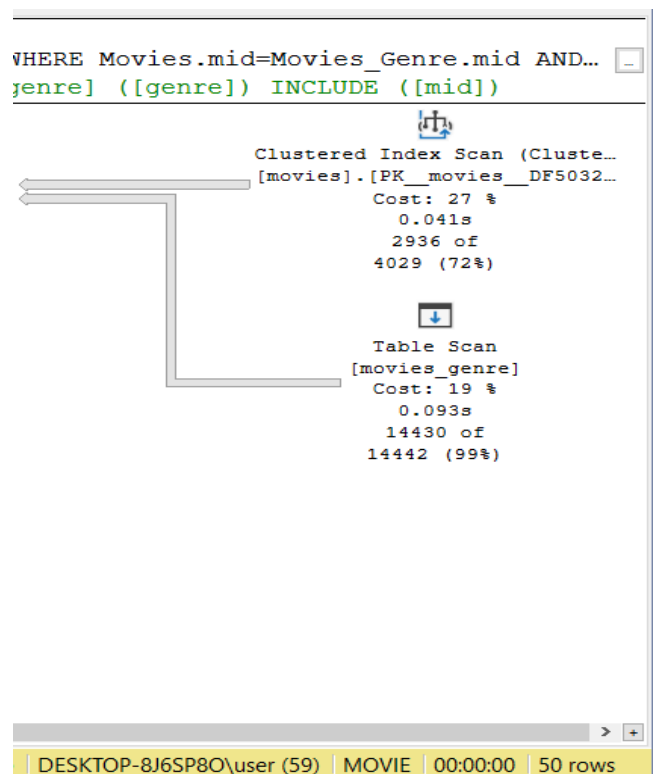
Όπως βλέπουμε και στην εικόνα, επιστρέφει 50 στήλες.

Αρχικό Πλάνο Εκτέλεσης

(ακολουθούν 2 εικόνες,όπως και πριν)



και η συνέχεια:

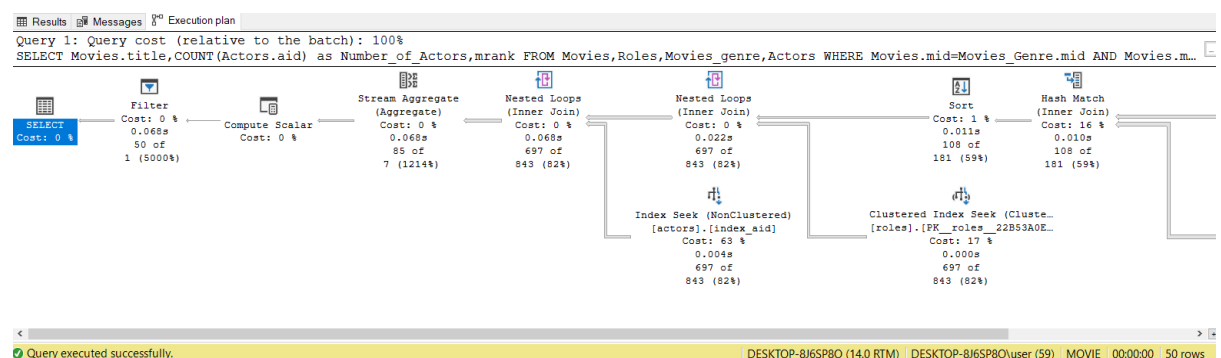


Προσθέτουμε τα ευρετήρια:

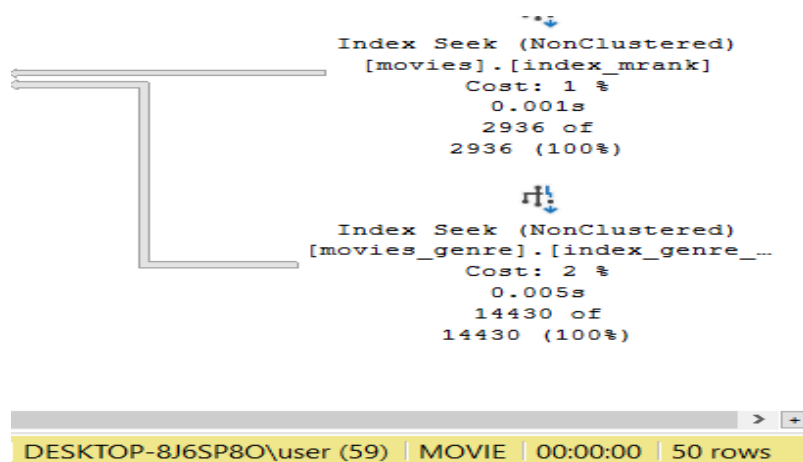
```

CREATE INDEX index_genre_mid ON Movies_Genre(genre,mid)
CREATE INDEX index_mrank on Movies(mrank) INCLUDE(title,mid)
CREATE INDEX index_aid ON Roles(aid) INCLUDE (mid)
  
```

Πλάνο εκτέλεσης μετά την προσθήκη ευρετηρίων



και η συνέχεια



Η σύγκριση των reads:

Χωρίς ευρετήριο:

Table	Logical Reads	Physical Reads	Read-ahead Reads
Actors	3252	1	2368
Roles	826	1	752
Movies_Genre	1123	0	1123
Movies	1918	2	1917

Με ευρετήριο:

Table	Logical Reads	Physical Reads	Read-ahead Reads
Actors	3252	1	2368
Roles	779	1	752
Movies_Genre	53	1	50
Movies	17	1	14

Τα indexes που βάλαμε στα genre,mrank μειώνουν τα logical reads, ιδιαίτερα στους πίνακες Movies_Genre,Movies. Επίσης θετικό είναι ότι στην θέση των table scan, clustered index scan βλέπουμε (με τη χρήση των ευρετηρίων) index seek.

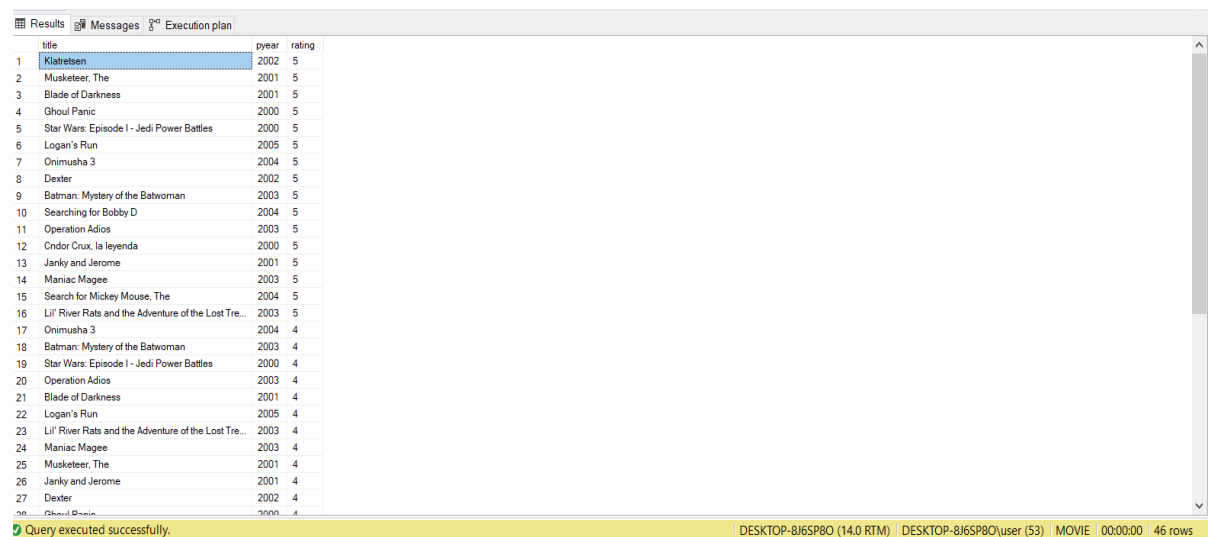
Query #2:

```
--2. Δείξε μας τις adventure ταινίες του 21ου αιώνα σε φθίνουσα σειρά
-- ανάλογα με τη βαθμολογία (να είναι τουλάχιστον 3) που έχουν λάβει από τους
-- χρήστες , οι οποίοι να είναι ηλικιακά 'νέοι' (18-35 ετών)

-- Ένας λόγος που επιλέγουμε να φιλτράρουμε το κοινό είναι επειδή οι ταινίες
-- αυτές έχουν αυξημένη απήχηση στο κοινό αυτό,άρα και οι βαθμολογίες αυτές
--θα είναι πιο αντιπροσωπευτικές.

SELECT title,pyear,rating
FROM Movies,Movies_Genre,User_Movies,Users
WHERE Movies.mid=Movies_Genre.mid
AND movies.mid=User_Movies.mid
AND User_Movies.userid=Users.userid
AND rating>=3
AND genre='Adventure'
AND pyear BETWEEN 2000 AND 2021
AND age BETWEEN 18 AND 35
GROUP BY title,pyear,rating
ORDER BY rating DESC
```

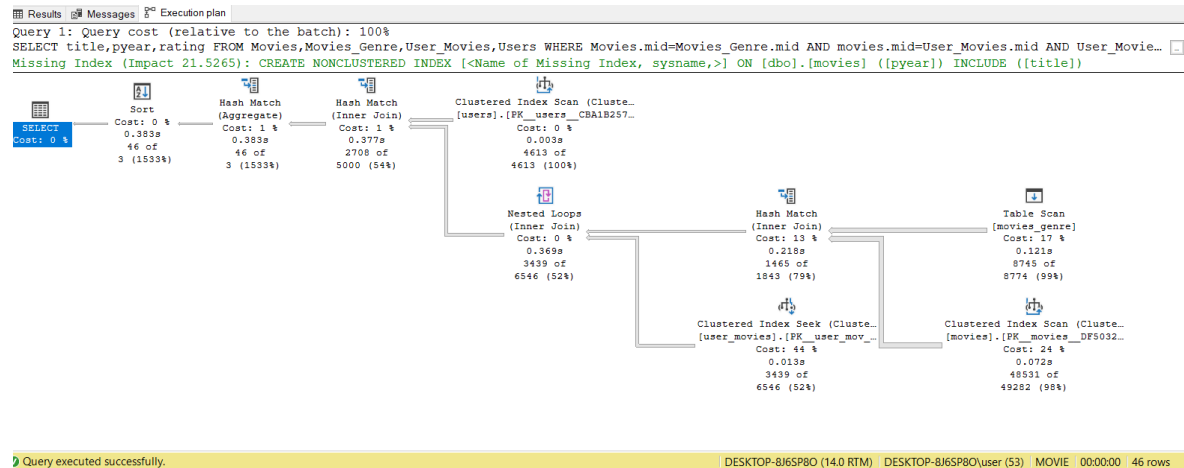
Το query αυτό μας επιστρέφει τον τίτλο,το έτος κυκλοφορίας και τον βαθμό αξιολόγησης των χρηστών. Επιστρέφει 46 στήλες, όπως βλέπουμε και στην εικόνα:



	title	pyear	rating
1	Klatschen	2002	5
2	Musketeer, The	2001	5
3	Blade of Darkness	2001	5
4	Ghoul Panic	2000	5
5	Star Wars: Episode I - Jedi Power Battles	2000	5
6	Logan's Run	2005	5
7	Onimusha 3	2004	5
8	Dexter	2002	5
9	Batman: Mystery of the Batwoman	2003	5
10	Searching for Bobby D	2004	5
11	Operation Adios	2003	5
12	Cndor Crux, la leyenda	2000	5
13	Janky and Jerome	2001	5
14	Maniac Magee	2003	5
15	Search for Mickey Mouse, The	2004	5
16	Li'l River Rats and the Adventure of the Lost Tre...	2003	5
17	Onimusha 3	2004	4
18	Batman: Mystery of the Batwoman	2003	4
19	Star Wars: Episode I - Jedi Power Battles	2000	4
20	Operation Adios	2003	4
21	Blade of Darkness	2001	4
22	Logan's Run	2005	4
23	Li'l River Rats and the Adventure of the Lost Tre...	2003	4
24	Maniac Magee	2003	4
25	Musketeer, The	2001	4
26	Janky and Jerome	2001	4
27	Dexter	2002	4
28	Ghoul Panic	2000	4

Query executed successfully. DESKTOP-8J6SP80 (14.0 RTM) | DESKTOP-8J6SP80\user (53) | MOVIE | 00:00:00 | 46 rows

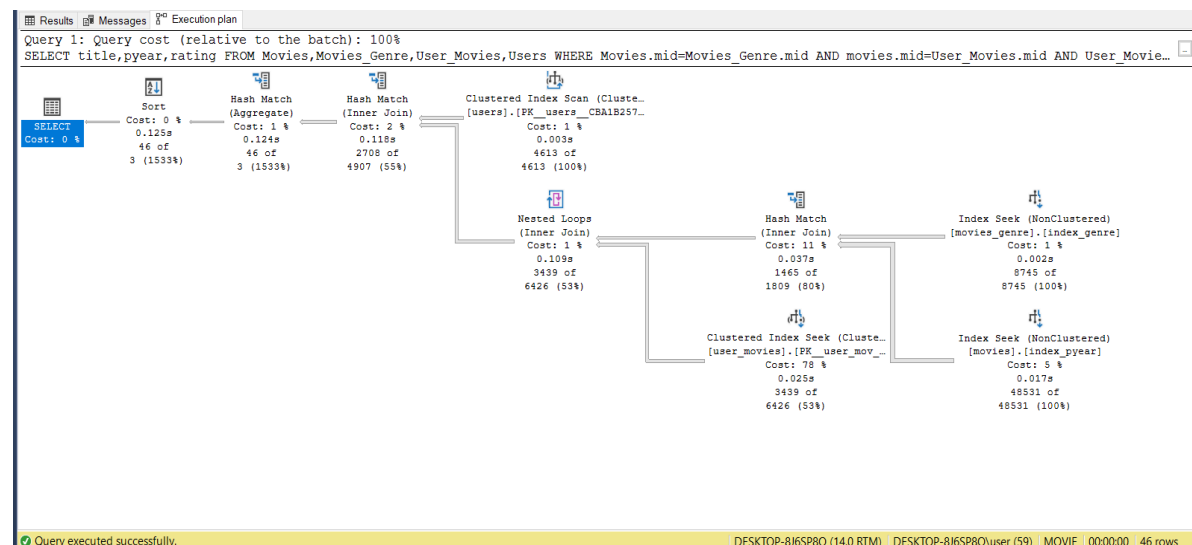
Αρχικό Πλάνο Εκτέλεσης



Προσθέτουμε τα ευρετήρια:

```
CREATE INDEX index_genre ON Movies_Genre(genre) INCLUDE (mid)
CREATE INDEX index_pyear ON Movies(pyear) INCLUDE(title)
```

Πλάνο Εκτέλεσης μετά την προσθήκη των ευρετηρίων



Η σύγκριση των reads:

Χωρίς ευρετήριο:

Table	Logical Reads
User_Movies	9954
Movies	1918
Movies_Genre	1123
Users	29

Με ευρετήριο:

Table	Logical Reads
User_Movies	8094
Movies	224
Movies_Genre	38
Users	29

Τα ευρετήρια λοιπόν στο **genre** και το **pyear** μας βοηθούν αρκετά. Στο πλάνο εκτέλεσης βλέπουμε ότι το table scan και το clustered index scan (ουσιαστικά table scan και αυτό, δεν αναγράφεται ως table scan λόγω του primary key) μετατρέπονται σε index seek . Τα logical reads μειώνονται στον πίνακα User_Movies και σημαντικά στους Movies_Genre, Movies.