Project 1

Σχεδιασμός Βάσεων Δεδομένων

Καρράς Κωνσταντίνος 3180076

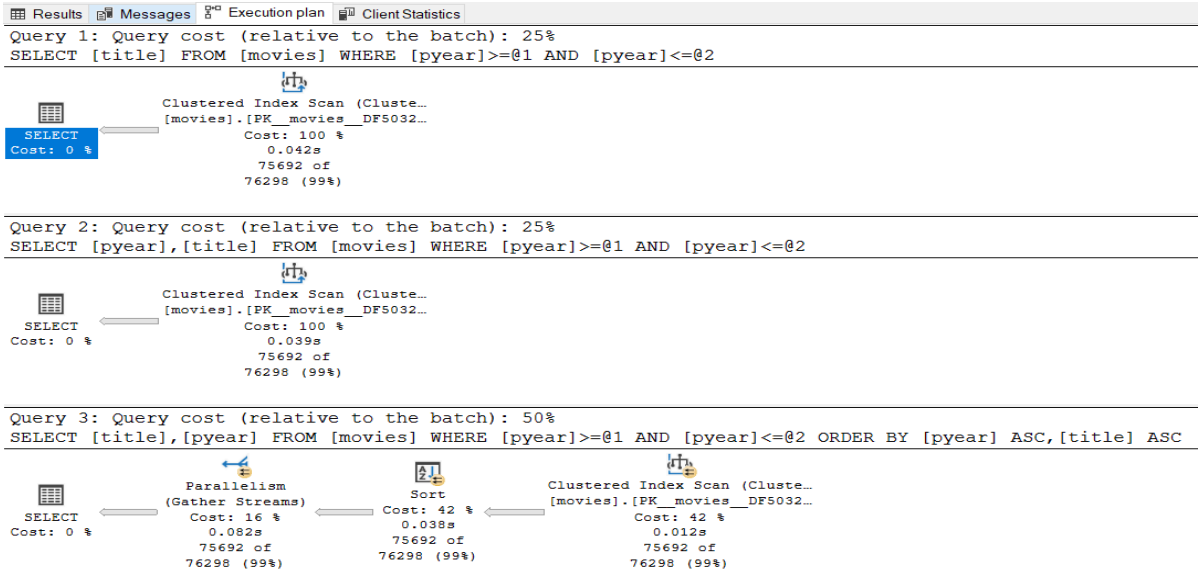
❖ Ζήτημα Πρώτο

- **1.**
  - *Ευρετήριο*

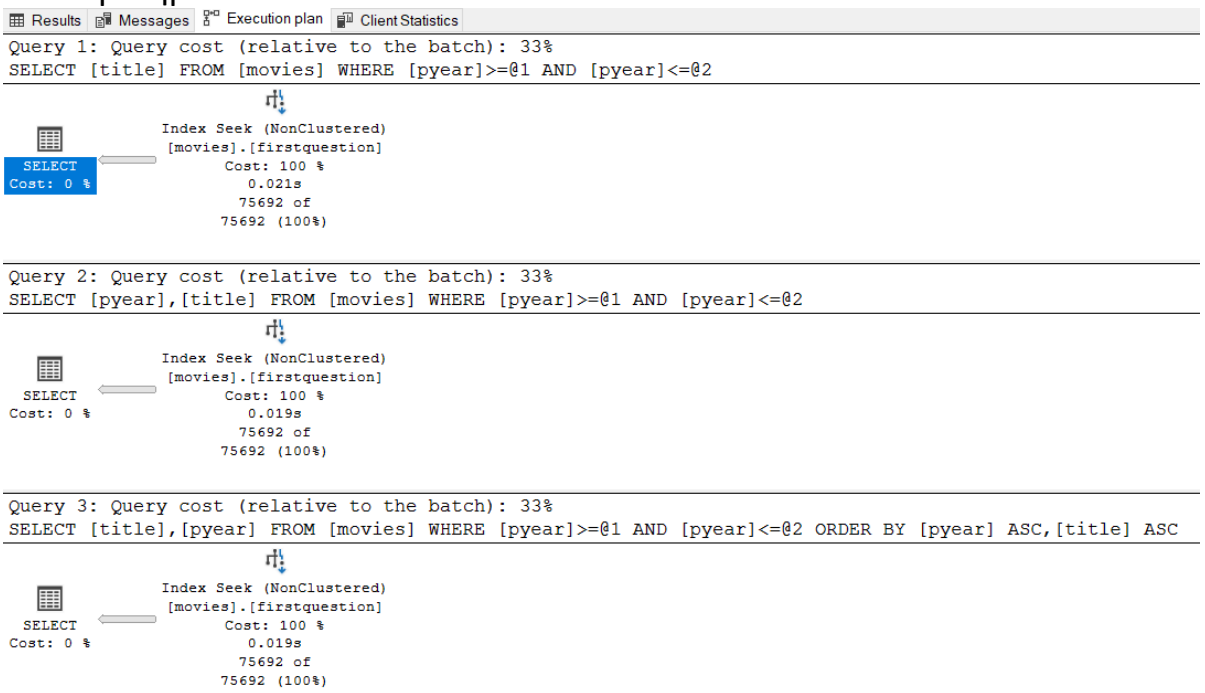
    ```
    create nonclustered index firstquestion on movies(pyear, title);
    ```

Ερώτημα	Πίνακας	Φυσικές Αναγνώσεις	Λογικές Αναγνώσεις	Read-Ahead Αναγνώσεις
Χωρίς ευρετήριο	1ο ερώτημα movies	2	1918	1917
	2ο ερώτημα movies	2	1918	1917
	3ο ερώτημα movies	2	2012	1917
Με ευρετήριο	1ο ερώτημα movies	3	354	362
	2ο ερώτημα movies	3	354	362
	3ο ερώτημα movies	3	354	362

- *Execution plans*
  - Χωρίς Ευρετήριο



  - Με Ευρετήριο



Είναι προφανές ότι η ταχύτητα των queries πλέον διπλασιάζεται στα 2 πρώτα ερωτήματα, ενώ στο τρίτο ερώτημα σχεδόν οχταπλασιάζεται (ή ο χρόνος εκτέλεσης είναι ο μισός και το 1/8 αντίστοιχα)! Πέρα, όμως, και από τους χρόνους εκτέλεσης, η ταχύτητα φαίνεται ότι αυξάνεται και από το πλήθος των σελίδων όπου διαβάζονται. Το συγκεκριμένο ευρετήριο

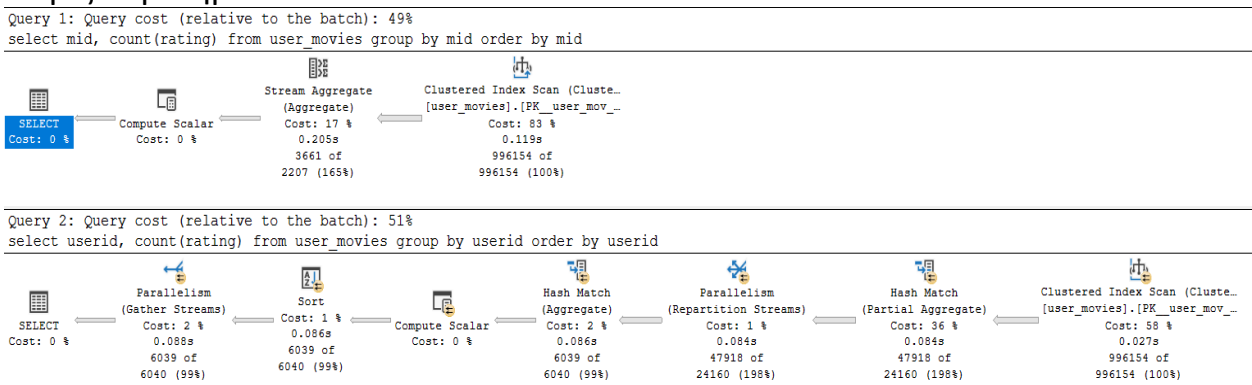
επιλέχθηκε, διότι το γνώρισμα pyear χρησιμοποιείται στο where clause και των τριών ερωτημάτων και προστέθηκε και το title γιατί σε όλα τα ερωτήματα θέλουμε και το title (στο select). Επομένως, το B+ δέντρο θα πρέπει να είναι ένα ευρετήριο <u>όχι μόνο</u> με το pyear στα φύλα, αλλά και με το title.

- **2.**
  - *<u>Ευρετήριο</u>*

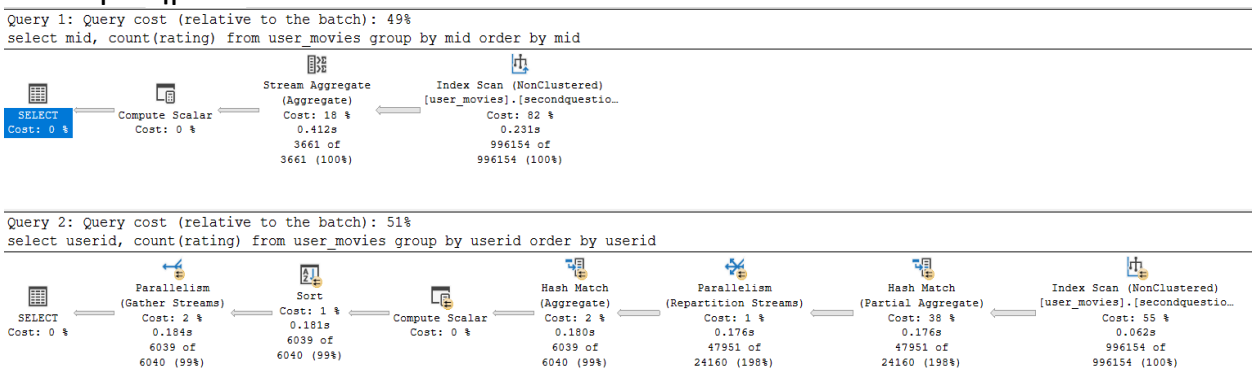
```
create nonclustered index secondquestion on user_movies(mid, rating);
```

Ερώτημα	Πίνακας	Φυσικές Αναγνώσεις	Λογικές Αναγνώσεις	Read-Ahead Αναγνώσεις
Χωρίς ευρετήριο	1ο ερώτημα user_movies	2	2601	2603
	2ο ερώτημα user_movies	2	2733	2603
Με ευρετήριο	1ο ερώτημα user_movies	2	2232	2277
	2ο ερώτημα user_movies	3	2346	2277

  - *<u>Execution plans</u>*
    - Χωρίς Ευρετήριο



➢ Με Ευρετήριο



Το ευρετήριο δημιουργείται πάνω στις μεταβλητές mid και rating. Χρησιμοποιείται το mid, διότι είναι κλειδί (μαζί με το userid) στον πίνακα user_movies. Ωστόσο, στο ευρετήριο δεν χρησιμοποιώ το userid. Βάζοντας, λοιπόν, το mid, το διάβασμα των σελίδων για το 2ο επερώτημα δεν επηρεάζεται γιατί δεν χρησιμοποιεί καν το mid, ενώ στο 1ο επερώτημα μειώνεται ο αριθμός σελίδων (χρησιμοποιείται και στο group by και στο order by). Το rating, από τη άλλη πλευρά, αφού υπάρχει το mid στο ευρετήριο και γίνεται ο διαχωρισμός πρώτα ως προς το mid δεν θα επηρεάσει εξίσου τον αριθμό των σελίδων για το 1ο επερώτημα. Στο 2ο , όμως, επερώτημα το rating θα συμβάλλει στη μείωση, γιατί θα υπάρχουν στο ευρετήριο οι βαθμολογίες (ratings) και άρα θα γίνει γρηγορότερα το count.

❖ Ζήτημα Δεύτερο

- **1.**
  - *Αποδοτικότερο ερώτημα*

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

  - *Επεξήγηση*

    Το επερώτημα της εκφώνησης δεν είναι τόσο αποδοτικό διότι πραγματοποιεί το ίδιο join στους πίνακες movies και movies_genre στο γνώρισμα mid. Αυτό έχει ως αποτέλεσμα να διαβαστούν 2 φορές οι ίδιες λογικές σελίδες. Το φυσικό διάβασμα δεν γλιτώνεται

σε καμία περίπτωση. Το παραπάνω ερώτημα κάνει ακριβώς το ίδιο διάβασμα φυσικών σελίδων, όμως οι λογικές σελίδες διαβάζονται μόνο μία φορά γιατί το join γίνεται μία φορά και όχι δύο. Τέλος, επειδή χρησιμοποιείται το union στο επερώτημα της εκφώνησης πάλι δεν γίνεται να γλιτώσω το distinct.

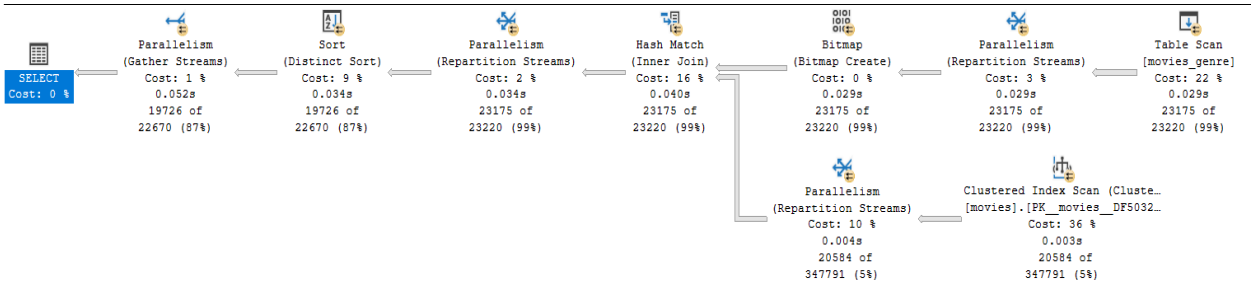
Ερώτημα	Πίνακας	Φυσικές Αναγνώσεις	Λογικές Αναγνώσεις	Read-Ahead Αναγνώσεις
Εκφώνησης	movies	2	4024	1917
	movies_genre	0	2246	1123
Δικό μου	movies	2	2012	1917
	movies_genre	0	1123	1123

- *Ευρετήριο*

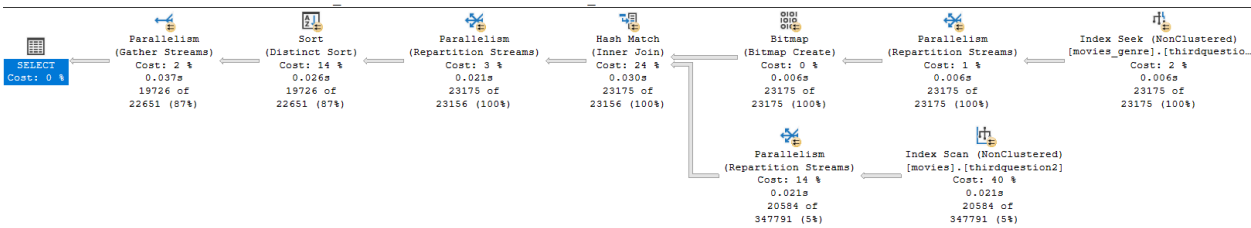
```
create nonclustered index thirdquestion on movies_genre(genre, mid);
create nonclustered index thirdquestion2 on movies(title, mid);
```

- *Execution plans*

➢ Χωρίς Ευρετήριο



➢ Με ευρετήριο



Δημιούργησα ευρετήριο στον πίνακα movies_genre στα γνωρίσματα genre και mid. Με αυτόν το τρόπο θα μπορώ να βλέπω γρήγορα (με index seek) αν μου κάνουν ή όχι τα συγκεκριμένα genres. Αν μου κάνουν θα πρέπει να περιλαμβάνεται και το mid στο ευρετήριο, γιατί με αυτό γίνονται join οι δύο πίνακες. Το ευρετήριο στον πίνακα movies με το title και το mid δεν συμβάλλει στην βελτίωση της ταχύτητας. Ακόμη και να μη γίνει index scan, θα γίνει clustered index scan, αφού το mid είναι primary key στον πίνακα movies. Ωστόσο, αποφάσισα να το κρατήσω, διότι φαίνεται να χρησιμοποιείται από το DBMS.

- **2.**
  - *1ο Επερώτημα*

```
select distinct title from movies,
(select movies.mid, count(aid) as actors_count
from movies, roles
where movies.mid = roles.mid group by movies.mid) as all_actors,
(select movies.mid, count(roles.aid) as men_actors_count
from actors, roles, movies
where gender = 'M' and movies.mid = roles.mid and roles.aid =
actors.aid group by movies.mid) as men_actors
where men_actors.men_actors_count = all_actors.actors_count and
all_actors.mid = men_actors.mid and all_actors.mid = movies.mid
```

- *2ο Επερώτημα*

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

- *Ευρετήρια για το 1ο Επερώτημα*

```
create nonclustered index fourthquestion on actors(gender);
create nonclustered index fourthquestion2 on movies(title);
create nonclustered index fourthquestion3 on roles(aid);
```

Ερώτημα	Πίνακας	Φυσικές Αναγνώσεις	Λογικές Αναγνώσεις	Read-Ahead Αναγνώσεις
Χωρίς Ευρετήρια	actors	2	3530	3362
	roles	1	8978	4426
	movies	2	6036	1917
Με Ευρετήρια	actors	2	738	706
	roles	0	3862	1960
	movies	1	4455	1452

- *Ευρετήρια για το 2ο Επερώτημα*

```
create nonclustered index fourthquestion on actors(gender, aid);
create nonclustered index fourthquestion2 on movies(title);
create nonclustered index fourthquestion3 on roles(mid, aid);
```

Ερώτημα	Πίνακας	Φυσικές Αναγνώσεις	Λογικές Αναγνώσεις	Read-Ahead Αναγνώσεις
Χωρίς Ευρετήρια	actors	2	7060	3362
	roles	3	8981	4410
	movies	242	125457	0
Με Ευρετήρια	actors	2	1610	1111
	roles	3	3873	1943
	movies	1	1485	1452

Και στις 2 περιπτώσεις τα ευρετήρια είναι σχεδόν τα ίδια. Υπάρχει το mid στον roles (στο ευρετήριο) για να γίνεται πιο γρήγορα ο έλεγχος με το except και υπάρχει και το aid επιπλέον στον actors (στο ευρετήριο πάλι) για να γίνονται πιο γρήγορα τα joins. Από τα 2 όμως αυτά επερωτήματα, θα κρατούσα το 1^{ο}. Και αυτό, γιατί γίνεται ένα join στον πίνακα all_actors ανάμεσα στον roles και τον movies και ένα join στον men_actors ανάμεσα στον roles, τον movies και τον actors. Τέλος, για το 1^{ο} επερώτημα το join που θα γίνει μεταξύ των all_actors, men_actors και movies δεν θα επιβαρύνει ιδιαίτερα, λόγω της συνθήκης ότι θα πρέπει τα δύο counts να είναι ίσα. Ενώ, λοιπόν , και το 2^{ο} επερώτημα κάνει τα ίδια join, όπως και το 1^{ο} επερώτημα (εκτός από το τελευταίο join), ελέγχει στο where clause αν το mid περιέχεται στο σύνολο που επιστρέφουν τα άλλα δύο queries, το οποίο επιβαρύνει, αφού θα πρέπει να γίνει table scan σε αυτό που έχουν επιστρέψει τα φωλιασμένα queries. Αυτό επιβεβαιώνεται και από τη χρονική διάρκεια που παίρνουν τα 2 queries να εκτελεστούν (το 1^{ο} διαρκεί 1 περίπου δευτερόλεπτο, ενώ το 2^{ο} σχεδόν 2 δευτερόλεπτα).

❖ Ζήτημα Τρίτο

- ***Φυσική γλώσσα***
  1. Εμφάνισε τον τίτλο της ταινίας, τον κωδικό του σκηνοθέτη και τον μέσο όρο βαθμολογίας των ταινιών, όπου οι χρήστες που τις βαθμολόγησαν ,ηλικίας από 46 μέχρι και 56, είναι περισσότεροι από 150 και ο μέσος όρος βαθμολόγησης αυτών των ταινιών από τους συγκεκριμένους χρήστες της προαναφερθείσας ηλικιακής κατηγορίας είναι ίση με 5.

  2. Εμφάνισε τον κωδικό, το όνομα και το επίθετο των ηθοποιών, που είναι γυναίκες και έχουν πάρει μέρος σε περισσότερες από 150 ταινίες, ταξινομώντας τες σε (αύξουσα) αλφαβητική σειρά ως προς το επίθετό τους

- ***SQL***
  1. ```
     select title, did, movies.mid, average from movies, movie_directors,
     (select mid, avg(rating) as average from user_movies, users
     where user_movies.userid = users.userid and users.age between 46
     and 56 group by mid, rating having count(rating) > 150) as temp
     where temp.mid = movies.mid and movies.mid=movie_directors.mid
     group by title, did, movies.mid, temp.average
     having temp.average = 5
     ```

  2. ```
     select roles.aid, firstName, lastname from actors, roles
     where actors.aid = roles.aid and gender = 'F'
     group by roles.aid, firstName, lastname
     having (count(roles.mid) > 150)
     order by lastname
     ```

- ***Ευρετήρια για το 1ο Επερώτημα***

  ```
  create nonclustered index fifthquestion on user_movies(rating);
  create nonclustered index fifthquestion2 on users(age);
  create nonclustered index fifthquestion3 on movies(mid, title);
  create nonclustered index fifthquestion4 on movie_directors(mid);
  ```

  Αν και τα ευρετήρια στον movies και τον movie_directors δεν συμβάλλουν στην ταχύτερη εκτέλεση της επερώτησης (αντιθέτως το index στον movies φαίνεται ότι διαβάζει περισσότερες σελίδες),

αποφάσισα να τα κρατήσω, διότι στο execution plan φαίνεται ότι χρησιμοποιούνται από το DBMS.

Ερώτημα	Πίνακας	Φυσικές Αναγνώσεις	Λογικές Αναγνώσεις	Read-Ahead Αναγνώσεις
Χωρίς Ευρετήρια	users	1	82	34
	user_movies	2	2733	2603
	movie_directors	10	24	0
	movies	7	15	0
Με Ευρετήρια	users	1	10	3
	user_movies	3	2344	2245
	movie_directors	10	24	0
	movies	10	15	0

- ***Ευρετήρια για το 2ο Επερώτημα***

```
create nonclustered index fifthquestion on roles(mid, aid);
create nonclustered index fifthquestion2 on actors(gender, aid,
firstName, lastname);
```

Ερώτημα	Πίνακας	Φυσικές Αναγνώσεις	Λογικές Αναγνώσεις	Read-Ahead Αναγνώσεις
Χωρίς Ευρετήρια	roles	1	4489	4426
	actors	2	3530	3362
Με Ευρετήρια	roles	1	1935	1967
	actors	2	1202	1141