# 5η Εργασία: Βελτιστοποίηση Ερωτημάτων και Εναύσματα

## Ονοματεπώνυμο

### Ερώτημα Α

Να δοθεί screenshot του πλάνου εκτέλεσης του ερωτήματος:

A screen shot of a computer

Description automatically generated

### Ερώτημα Β

1. Να φτιάξετε **2 ευρετήρια** τα οποία επιταχύνουν την εκτέλεση του ερωτήματος.

CREATE INDEX title\_index ON movie(title);

CREATE INDEX movie\_id\_index ON movie\_cast(movie\_id);

1. Να δοθεί το αντίστοιχο πλάνο εκτέλεσης του ερωτήματος αφού έχουν φτιαχτεί τα ευρετήρια.

A computer screen shot of a diagram

Description automatically generated

1. Ποιές είναι οι αλλαγές που έχουν γίνει στο πλάνο εκτέλεσης;

Πλέον σαρώνουμε τα ευρετήρια title\_index και movie\_id\_index και κάνουμε σύζευξη των αποτελεσμάτων τους, αντί να χρησιμοποιήσουμε μόνο τα primary keys ως ευρετήρια. Ως αποτέλεσμα αυτού παρατηρούμε ότι δεν χρησιμοποιούμε Hash Match αλλά Key Lookup και Nested Loops

### Ερώτημα C

Χρησιμοποιώντας τις εντολές GROUP BY και HAVING στον πίνακα Ratings:

1. Ποιό είναι το ερώτημα το οποίο επιστρέφει τον μέσο όρο βαθμολογίας ανά ταινία για ταινίες με *μέση βαθμολογία πάνω από 4;*

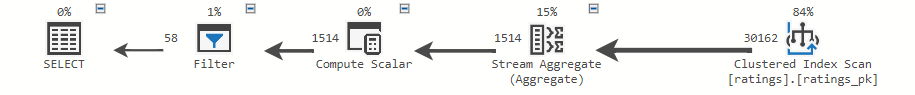
SELECT movie\_id, AVG(rating) as average\_rating

FROM ratings

GROUP BY movie\_id

HAVING AVG(rating) > 4;

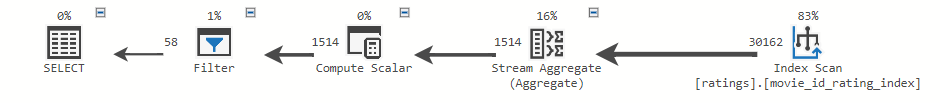
1. Να τρέξετε το ερώτημα και να επιστρέψετε το αντίστοιχο πλάνο εκτέλεσης.



1. Να δημιουργηθεί ένα *ευρετήριο επικάλυψης* το οποίο μπορεί να χρησιμοποιηθεί για την απάντηση στο συγκεκριμένο ερώτημα.

CREATE INDEX movie\_id\_rating\_index ON ratings(movie\_id, rating);

1. Καταγράψτε το πλάνο εκτέλεσης με το πλάνο επικάλυψης,



1. Τι παρατηρείτε σε αυτό ως προς το διάβασμα του πίνακα Ratings και γιατί συμβαίνει αυτό;

Στο δεύτερο πλάνο χρησιμοποιούμε το ευρετήριο επικάλυψης που κατασκευάσαμε στο ερώτημα ( c ) (και μόνο αυτό, αφού είναι ευρετήριο επικάλυψης), προκειμένου να βελτιώσουμε την ταχύτητα αναζήτησης, αφού μέσω αυτού δεν θα χρειαστεί να αναζητήσουμε τις πληροφορίες διαβάζοντας κάθε γραμμή του πίνακα ratings.

### Ερώτημα D

1. Να δοθεί η εντολή Alter Table για την δημιουργία του γνωρίσματος AVG\_Rating στον πίνακα Movie.

ALTER TABLE Movie

ADD AVG\_Rating float();

1. Χρησιμοποιώντας την εντολή UPDATE, αρχικοποιήστε το συγκεκριμένο γνώρισμα με την μέση βαθμολογία ανά ταινία σε περίπτωση που αυτή υπάρχει και Null διαφορετικά.

UPDATE movie

SET AVG\_Rating = (

SELECT AVG(rating)

FROM ratings

WHERE movie.id = ratings.movie\_id

);

**Σχόλιο**: δεν χρειάζεται να το θέσουμε ρητά σε NULL διαφορετικά.

1. Γράψτε ένα *έναυσμα* (trigger) το οποίο για κάθε εισαγωγή στον πίνακα Ratings ενημερώνει το γνώρισμα AVG\_Rating του πίνακα Movie με την μέση βαθμολογία μόνο των ταινιών των οποίων η βαθμολογία άλλαξε.

CREATE TRIGGER update\_AVG\_Rating

ON ratings

AFTER INSERT

AS

BEGIN

DECLARE @movie\_id INT;

SELECT @movie\_id = movie\_id FROM inserted;

UPDATE movie

SET AVG\_Rating = (

SELECT AVG(rating)

FROM ratings

WHERE ratings.movie\_id = @movie\_id

)

WHERE id = @movie\_id;

END;

1. Ποιό από τα ευρετήρια που φτιάξατε βοηθάει στην εκτέλεση του συγκεκριμένου trigger.

Προφανώς, το movie\_id\_rating\_index.

### Ερώτημα Ε

Περιγράψτε κάποιον εναλλακτικό τρόπο με τον οποίο θα μπορούσατε να υλοποιήσετε το trigger σας, χωρίς να χρειαστεί να τρέχετε κάποια group by συνάρτηση στα υπάρχοντα δεδομένα.

Προσθέτουμε στον πίνακα movie το γνώρισμα ratings\_count, δηλαδή πόσες εγγραφές στον πίνακα ratings αναφέρονται στην ταινία (έχουν ίδιο id). Τότε, το νέο AVG\_Rating υπολογίζεται ως (έστω new\_rating το rating της καινούριας εγγραφής που γίνεται insert στον πίνακα ratings):

AVG\_Rating = (ratings\_count\*AVG\_Rating + new\_rating) / (++ratings\_count)

(Σημείωση, αφού η SQL δεν έχει pre-increment ++ operator όπως της c++ που χρησιμοποιήσαμε στον ψευδοκώδικα, γιαυτό εκεί γράφουμε ratings\_count + 1 και μετά στο τέλος κάνουμε ratings\_count = ratings\_count + 1)