Σχεδιασμός Βάσεων Δεδομένων
Διδάσκων: Ιωάννης Κωτίδης

Εαρινό εξάμηνο 2024-2025

Πρώτη Εργασία

Ανάθεση: **12-04-2025**
Παράδοση: **30-04-2025 Ώρα (23:55)**

Οδηγίες

- *Η εργασία είναι **ατομική** και **υποχρεωτική.***
- *Η υποβολή της εργασίας πρέπει να γίνει στο eclass.*
- *Το παραδοτέο σας θα πρέπει να είναι ένα αρχείο PDF με όνομα AM.pdf (όπου ΑΜ είναι ο αριθμός μητρώου σας. π.χ. "3220001.pdf").*
- *Πιθανή αντιγραφή θα τιμωρείται με μηδενισμό όλων των εμπλεκομένων.*

Βάση Δεδομένων Κινηματογραφικών Ταινιών

Στόχος της εργασίας είναι η πρακτική εφαρμογή των γνώσεων που αποκομίσατε από τις διαλέξεις του μαθήματος σχετικά με την δημιουργία ευρετηρίων και την βελτιστοποίηση των επερωτήσεων SQL. Για την πρακτική σας εξάσκηση θα χρησιμοποιήσετε μια βάση δεδομένων και το DBMS MICROSOFT SQL SERVER. Η βάση δεδομένων περιέχει πληροφορίες για κινηματογραφικές ταινίες.
Αρχικά θα δημιουργήσετε την βάση δεδομένων και θα φορτώσετε τα δεδομένα στους πίνακες, ακολουθώντας τις παρακάτω οδηγίες. Στη συνέχεια θα απαντήσετε στα ζητούμενα της εργασίας.

1. Οδηγίες για την δημιουργία της βάσης.

Για να δημιουργήσετε την βάση δεδομένων και να φορτώσετε τις εγγραφές ακολουθείστε **ΠΡΟΣΕΚΤΙΚΑ** τα παρακάτω βήματα:

Βήμα 1: Από το περιβάλλον του Microsoft Sql Server Management Studio δημιουργείστε μια βάση δεδομένων με όνομα **MOVIE**.

Βήμα 2: Εκτελέστε το SQL script **"CreateMovieSchema.sql"** που δημιουργεί το λογικό σχήμα της βάσης. Πριν εκτελέσετε το script βεβαιωθείτε ότι η τρέχουσα βάση δεδομένων είναι η βάση **MOVIE** που δημιουργήσατε στο βήμα 1.

Βήμα 3: Εκτελέστε το SQL script **"LoadMovieData.sql"** το οποίο θα φορτώσει δεδομένα στους πίνακες της βάσης. Το συγκεκριμένο script περιέχει εντολές της μορφής:

BULK INSERT actors ! Πίνακας στον οποίο θα φορτωθούν τα δεδομένα

FROM 'C:\ movieData\actors.txt' ! Αρχείο που περιέχει τα δεδομένα

WITH (FIRSTROW =2, FIELDTERMINATOR= '|', ROWTERMINATOR = '\n');

Παράμετροι:

FIRSTROW=2 : Η πρώτη γραμμή του αρχείου περιέχει τα ονόματα των πεδίων και αγνοείται.

FIELDTERMINATOR ='|' : Ο χαρακτήρας '|' δηλώνει το τέλος κάθε πεδίου της εγγραφής.

ROWTERMINATOR='\n' : Ο χαρακτήρας αλλαγής γραμμής δηλώνει το τέλος κάθε εγγραφής του αρχείου.

ΠΡΟΣΟΧΗ: Αν τοποθετήσετε τα δεδομένα σε φάκελο διαφορετικό από τον **'C:\movieData '** θα πρέπει να τροποποιήσετε ανάλογα το path. Για παράδειγμα αν τοποθετήσετε τα δεδομένα στον φάκελο **'C:\DATA'** η παραπάνω εντολή πρέπει να αλλάξει ως εξής:

BULK INSERT actors

FROM 'C:\DATA\actors.txt'

WITH (FIRSTROW =2, FIELDTERMINATOR= '|', ROWTERMINATOR = '\n');

Τα SQL scripts **"CreateMovieSchema.sql"** και **"LoadMovieData.sql"** καθώς επίσης και τα αρχεία με τα δεδομένα που θα φορτωθούν στους πίνακες της βάσης θα τα βρείτε στο αρχείο "**movieData.zip".**

ΣΗΜΕΙΩΣΗ: Τα περιεχόμενα των παραπάνω scripts παρατίθενται στο παράρτημα που ακολουθεί στο τέλος της εργασίας.

2. Περιγραφή των πινάκων της βάσης

Ακολουθεί η περιγραφή των πινάκων και των δεδομένων της βάσης.

ACTORS: Πίνακας με στοιχεία ηθοποιών. Αριθμός Εγγραφών=817062	
aid	Κωδικός ηθοποιού
firstName	Όνομα ηθοποιού
lastName	Επώνυμο ηθοποιού
gender	Φύλο (F=female, M=Male)

DIRECTORS: Πίνακας με στοιχεία σκηνοθετών. Αριθμός εγγραφών=86880	
did	Κωδικός σκηνοθέτη
firstName	Όνομα σκηνοθέτη
lastName	Επώνυμο σκηνοθέτη

MOVIES: πίνακας με τα στοιχεία των ταινιών. Αριθμός εγγραφών=347796.	
mid	Κωδικός ταινίας
title	Τίτλος ταινίας
pyear	Έτος κυκλοφορίας
mrank	Κατάταξη [1.0 - 9.9]

MOVIE_DIRECTORS: Πίνακας που συνδέει τις ταινίες με τους σκηνοθέτες. Αριθμός εγγραφών=319117	
mid	Κωδικός ταινίας
did	Κωδικός σκηνοθέτη

MOVIES_GENRE: Πίνακας την κατηγορία/κατηγορίες κάθε ταινίας. Αριθμός εγγραφών=387390	
mid	Κωδικός ταινίας
genre	Κατηγορία

ROLES: Πίνακας που συνδέει τις ταινίες με τους ηθοποιούς. Αριθμός εγγραφών=1093499	
mid	Κωδικός ταινίας
aid	Κωδικός ηθοποιού
a_role	Ο ρόλος του ηθοποιού στην συγκεκριμένη ταινία

USERS: Πίνακας με τα στοιχεία των χρηστών. Αριθμός εγγραφών=6039. **ΤΑ ΣΤΟΙΧΕΙΑ ΤΩΝ ΧΡΗΣΤΩΝ ΔΕΝ ΕΙΝΑΙ ΠΡΑΓΜΑΤΙΚΑ.**	
userid	Κωδικός χρήστη
uname	Ονοματεπώνυμο χρήστη
gender	Φύλο (F=female, M=Male)
age	Ηλικία [18-56]

USER_MOVIES: Πίνακας με στοιχεία αξιολόγησης των ταινιών. Αριθμός εγγραφών=996159.	
mid	Κωδικός ταινίας
userid	Κωδικός χρήστη
rating	Βαθμός αξιολόγησης [1-5]

3. Ζητούμενα εργασίας

Ακολουθούν τα ζητούμενα της εργασίας. Για την απάντηση των ζητημάτων **δεν επιτρέπεται καμία απολύτως τροποποίηση του σχήματος** εκτός φυσικά από την δημιουργία των ζητούμενων ευρετηρίων. Επίσης **απαγορεύεται** η δημιουργία και η χρήση όψεων (views).

Σε κάθε ζήτημα δεν αρκεί μόνο να παραθέσετε τα επερωτήματα σε γλώσσα SQL ή/και τις εντολές δημιουργίας των ευρετηρίων που ζητούνται. Σε κάθε περίπτωση πρέπει να τεκμηριώσετε τις απαντήσεις σας και να παραθέσετε στοιχεία που επιβεβαιώνουν τους ισχυρισμούς σας. Για παράδειγμα:

- Σε περιπτώσεις που ζητείται να αποδείξετε ότι ένα ευρετήριο επιταχύνει ένα ερώτημα, εκτελέστε το επερώτημα δίχως το ευρετήριο και εξετάστε το πλάνο εκτέλεσης. Αφού δημιουργήσετε το ευρετήριο εκτελέστε εκ νέου το επερώτημα και επανεξετάστε το πλάνο εκτέλεσης. Συγκρίνοντας τα δύο πλάνα μπορείτε να καταλήξετε σε συμπεράσματα σχετικά με την καταλληλότητα του ευρετηρίου. Εξηγείστε συνοπτικά πως το ευρετήριο επιταχύνει το επερώτημα.

- Σε περιπτώσεις που πρέπει να συγκρίνετε ένα η περισσότερα επερωτήματα, εκτελέστε τα όλα μαζί σε δέσμη και εξετάστε τα πλάνα εκτέλεσης. Ο SQL server δείχνει το κόστος κάθε επερωτήματος ως ποσοστό επί του συνολικού κόστους εκτέλεσης της δέσμης.

- Ενεργοποιείστε τα στατιστικά στοιχεία I/O με την εντολή: SET STATISTICS TIME, IO ON. Με τον τρόπο αυτό μπορείτε να βλέπετε κάθε φορά που εκτελείτε ένα επερώτημα πόσες σελίδες διαβάζονται από τον δίσκο ή/και από την μνήμη (buffer) καθώς και τον χρόνο εκτέλεσης.

- Κάθε φορά πριν την εκτέλεση ενός επερωτήματος, εκτελέστε τις παρακάτω εντολές που "καθαρίζουν" τους buffers που χρησιμοποιεί ο SQL server για την αποθήκευση των δεδομένων και των πλάνων εκτέλεσης:

  ```
  CHECKPOINT
  DBCC dropcleanbuffers
  ```

  Με τον τρόπο αυτό διασφαλίζετε ότι, το επερώτημα που θα εκτελέσετε δεν θα χρησιμοποιήσει τυχόν σελίδες που υπάρχουν στην μνήμη από προηγούμενες εκτελέσεις του ιδίου ή/και άλλων επερωτημάτων. Σε αντίθετη περίπτωση μπορεί να οδηγηθείτε σε λάθος συμπεράσματα.

ΠΡΟΣΟΧΗ: Κάθε ζήτημα πρέπει να το αντιμετωπίσετε ανεξάρτητα από τα υπόλοιπα και να το υλοποιήσετε στο αρχικό στιγμιότυπο της βάσης. Για παράδειγμα αν θέλετε να εξετάσετε κατά πόσο ένα ευρετήριο κάνει πιο αποδοτικό ένα ερώτημα, βεβαιωθείτε ότι έχετε διαγράψει (drop index) τα ευρετήρια που έχετε δημιουργήσει για την βελτιστοποίηση άλλων επερωτημάτων.

Ζήτημα Πρώτο [10 μονάδες]
Το παρακάτω επερώτημα εμφανίζει το πλήθος των αξιολογήσεων με βαθμό πέντε (rating=5) ανά ηλικία χρηστών.

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

Ζητείται να δημιουργήσετε κατάλληλα ευρετήρια που επιταχύνουν την εκτέλεση του επερωτήματος. Να παραθέσετε τις εντολές δημιουργίας των ευρετηρίων, καθώς επίσης και στοιχεία που να αποδεικνύουν ότι τα ευρετήρια που δημιουργήσατε επιταχύνουν την εκτέλεση του παραπάνω επερωτήματος. Να αιτιολογήσετε εν συντομία τις επιλογές σας.

Ζήτημα Δεύτερο [15 μονάδες]
Να δημιουργήσετε **ένα μόνο** ευρετήριο που να επιταχύνει την εκτέλεση και των τριών παρακάτω επερωτημάτων. Να παραθέσετε την εντολή δημιουργίας του ευρετηρίου και να αιτιολογήσετε την επιλογή σας.

```
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
```

Ζήτημα Τρίτο [25 μονάδες]

Δίνονται τα παρακάτω δύο επερωτήματα σε γλώσσα SQL:

1.
```
SELECT  title, pyear, lastname, firstname
FROM movies INNER JOIN movie_directors ON movies.mid=movie_directors.mid
            INNER JOIN directors ON movie_directors.did=directors.did
      WHERE title='Dreamer' ORDER BY pyear
```

2.
```
SELECT  lastname, firstname, title, pyear
FROM movies INNER JOIN movie_directors ON movies.mid=movie_directors.mid
            INNER JOIN directors ON movie_directors.did=directors.did
       WHERE lastname='Nolan' ORDER BY lastname, firstname
```

Το πρώτο επερώτημα αναζητά ταινίες με βάση τον τίτλο τους και το δεύτερο με βάση το επώνυμο του σκηνοθέτη. Θεωρήστε ότι και τα δύο επερωτήματα έχουν την ίδια συχνότητα εκτέλεσης.

Ζητείται να δημιουργήσετε κατάλληλα ευρετήρια που επιταχύνουν την εκτέλεση των παραπάνω δύο επερωτημάτων. Να παραθέσετε τις εντολές δημιουργίας των ευρετηρίων, καθώς επίσης και στοιχεία που να αποδεικνύουν ότι τα ευρετήρια που δημιουργήσατε επιταχύνουν την εκτέλεση των επερωτημάτων. Να αιτιολογήσετε εν συντομία τις επιλογές σας.

Ζήτημα Τέταρτο [25 μονάδες]

Δίνονται τα παρακάτω δύο επερωτήματα σε γλώσσα SQL:

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

2. SELECT actors.aid, lastname, firstname,  a_role, title
   FROM actors INNER JOIN roles ON actors.aid = roles.aid
               INNER JOIN movies ON roles.mid=movies.mid
   WHERE lastname='Paez' and firstname='Alex'
   ORDER BY lastname, firstname, actors.aid
```

Θεωρήστε ότι και τα δύο επερωτήματα έχουν την ίδια συχνότητα εκτέλεσης.

Το πρώτο επερώτημα εμφανίζει τους ηθοποιούς που δεν έχουν συμμετάσχει σε καμία ταινία, ενώ το δεύτερο επερώτημα εμφανίζει τους ρόλους τους οποίους έχει υποδυθεί ένας συγκεκριμένος ηθοποιός (στην συγκεκριμένη περίπτωση ο «Paez Alex») στις ταινίες που έχει συμμετάσχει. Στα αποτελέσματα εμφανίζεται και ο κωδικός του ηθοποιού για την περίπτωση που υπάρχουν περισσότεροι από ένας ηθοποιοί με το ίδιο ονοματεπώνυμο.

Ζητείται:

Α. Να δημιουργήσετε τα κατάλληλα ευρετήρια που επιταχύνουν την εκτέλεση των παραπάνω δύο επερωτημάτων. Να παραθέσετε τις εντολές δημιουργίας των ευρετηρίων, καθώς επίσης και στοιχεία που να αποδεικνύουν ότι τα ευρετήρια που δημιουργήσατε επιταχύνουν την εκτέλεση των παραπάνω επερωτημάτων. Να αιτιολογήσετε εν συντομία τις επιλογές σας.

Β. Να ξαναγράψετε **το πρώτο επερώτημα** χρησιμοποιώντας εναλλακτική σύνταξη SQL που επιστρέφει ακριβώς το ίδιο αποτέλεσμα αλλά εκτελείται με αποδοτικότερο τρόπο. Δεν επιτρέπεται να τροποποιήσετε τα ευρετήρια που δημιουργήσατε στο ζητούμενο Α, ή να δημιουργήσετε νέα ευρετήρια.

Ζήτημα Πέμτο [25 μονάδες]

Δίνονται τα παρακάτω δύο επερωτήματα σε γλώσσα SQL:

```
1. SELECT AVG(rating) AS average_rating
        FROM movies_genre INNER JOIN user_movies
             ON movies_genre.mid = user_movies.mid
    WHERE genre ='Drama'

2.  SELECT movies.*, genre
         FROM movies INNER JOIN movies_genre
             ON movies.mid = movies_genre.mid
     WHERE title = 'Spider-Man 3';
```

Το πρώτο επερώτημα υπολογίζει και εμφανίζει τον μέσο όρο των αξιολογήσεων των χρηστών των ταινιών μιας συγκεκριμένης κατηγορίας (στην συγκεκριμένη περίπτωση «Drama»). Το δεύτερο επερώτημα ανακτά και εμφανίζει τα στοιχεία των ταινιών με βάση τον τίτλο τους, μαζί με τις κατηγορίες στις οποίες ανήκουν.

Δεδομένου ότι:

- η συχνότητα εκτέλεσης του πρώτου επερωτήματος είναι 20% και του δευτέρου 80%
- ο πίνακας movies_genre δεν έχει πρωτεύον κλειδί

Ζητείται:

Να δημιουργήσετε το πρωτεύον κλειδί (primary key) του πίνακα movies_genre καθώς επίσης και κατάλληλα ευρετήρια ώστε να υποστηρίζεται η αποδοτική εκτέλεση των παραπάνω επερωτημάτων και να ελαχιστοποιηθεί ο συνολικός φόρτος του (workload) του συστήματος. Να παραθέσετε την εντολή δημιουργίας του πρωτεύοντος κλειδιού και τις εντολές δημιουργίας των ευρετηρίων.
Να παραθέσετε στοιχεία που να αποδεικνύουν ότι τα ευρετήρια που δημιουργήσατε επιταχύνουν την εκτέλεση των παραπάνω επερωτημάτων και να αιτιολογήσετε εν συντομία τις επιλογές σας.

ΠΑΡΑΡΤΗΜΑ Α

CreateMovieSchema.sql

```
create table actors
(aid int primary key,
 firstName varchar(100),
 lastname varchar(100),
 gender char(1)
 );

 create table directors
(did int primary key,
 firstName varchar(100),
 lastname varchar(100)
 );

create table movies
 (mid int primary key,
  title varchar(200),
  pyear int,
  mrank decimal(2,1)
  );

 create table movie_directors
 ( did int foreign key references directors(did),
   mid int foreign key references movies(mid)
   primary key(mid, did),
 );

 create table roles
 ( aid int foreign key references actors(aid),
   mid int foreign key references movies(mid),
   a_role varchar(100),
   primary key (mid,aid)
  );

  create table movies_genre
  (mid int not null foreign key references movies(mid),
   genre varchar(100) not null,

   );

create table users
(userid int primary key,
 uname varchar(50),
 gender char(1),
 age int
 );

 create table user_movies
 (userid int foreign key references users(userid),
  mid int foreign key references movies(mid),
  rating int,
  primary key (mid,userid)
  );
```

loadMovieData.sql

```
BULK INSERT actors
 FROM 'C:\movieData\actors.txt'
WITH (FIRSTROW =2, FIELDTERMINATOR= '|', ROWTERMINATOR = '\n');

BULK INSERT directors
 FROM 'C:\movieData\directors.txt'
WITH ( FIRSTROW =2, FIELDTERMINATOR= '|', ROWTERMINATOR = '\n');

BULK INSERT movies
 FROM 'C:\movieData\movies.txt'
WITH ( FIRSTROW =2, FIELDTERMINATOR= '|', ROWTERMINATOR = '\n');

BULK INSERT movie_directors
 FROM 'C:\movieData\movie_directors.txt'
WITH ( FIRSTROW =2, FIELDTERMINATOR= '|', ROWTERMINATOR = '\n');

BULK INSERT movies_genre
 FROM 'C:\movieData\movies_genre.txt'
WITH ( FIRSTROW =2, FIELDTERMINATOR= '|', ROWTERMINATOR = '\n');

BULK INSERT roles
 FROM 'C:\movieData\roles.txt'
WITH ( FIRSTROW =2, FIELDTERMINATOR= '|', ROWTERMINATOR = '\n');

BULK INSERT users
 FROM 'C:\movieData\users.txt'
WITH (FIRSTROW =2, FIELDTERMINATOR= '|', ROWTERMINATOR = '\n');

BULK INSERT user_movies
 FROM 'C:\movieData\user_movies.txt'
WITH (FIRSTROW =2, FIELDTERMINATOR= '|', ROWTERMINATOR = '\n');
```