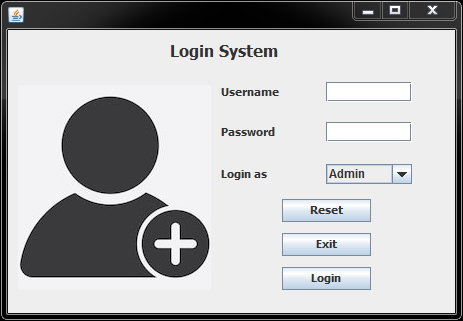
**Aplicatie de feedback si clasificare a filmelor**

Aplicatia va folosi o baza de date in care sunt stocate detalii despre diverse filme si actorii participanti. Fiecare utilizator va putea accesa aceasta baza de date, avand posibilitatea de a evalua filmul (pe baza unui usermane si a unui userID) prin acordarea unei note (cuprinsa intre 1 si 10) si de a lasa un comentariu cu opinia sa.

 Totodata, aplicatia de fata foloseste mediul de dezvoltare de programe Java intitulat Eclipse prin care se doreste conectarea la baza de date (elaborata in SQL Server Management Studio 2012) propriu-zisa si executarea unor cereri fara ca utilizatorul sa aiba acces direct la baza de date. Conectarea cu baza de date se realizeaza printr-o interfata ce contine 3 campuri: Username, Password, Login as (conectarea la aplicatie se va face in calitate de admin sau client). Diferenta dintre admin si client este faptul ca numai adminul are permisiunea de a modifica inregistrari din tabele, astfel incat in interfata pentru client vor lipsi butoanele care realizeaza aceste schimbari.

Valorile pentru aceste campuri, astfel incat sa permita conectarea aplicatiei la baza de date, se gasesc in tabela User\_parola. Pe langa aceste campuri mai avem 3 butoane: Login (prin apasarea caruia se incearca conectarea la baza de date), Exit (prin care se iese din interfata) si Reset (care reseteaza valorile primelor 2 campuri: Username, Password – campului Login as ii este asociat un buton de tip JcomboBox care nu poate fi resetat, fiind posibila alegerea a doua valori – Admin si Client).

Avem urmatoarele tabele:

* Actor (actor\_id, actor\_nume, actor\_prenume, actor\_sex)
* Regizor (regizor\_id, regizor\_nume, regizor\_prenume)
* Film\_actori (actor\_id, film\_id, rol)
* Film (film\_id, film\_titlu, film\_an, film\_durata, film\_limba, film\_tara, regizor\_id, descriere)
* Review (review\_id, film\_id, comentariu, user\_id, review\_nota, ora\_comentariu)
* User (user\_id, username)

Relatii intre tabele:

Film – Regizor

Film – Review

Review – User

Film\_actori – Actor

Film\_actori – Film

Toate relatiile de mai sus sunt one to many.

Chei tabele:

Actor : actor\_id (cheie primara)

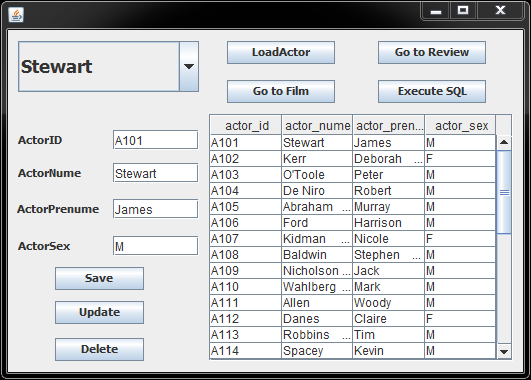
Regizor : regizor\_id (cheie primara)

Film\_actori : actor\_id (cheie primara) , film\_id (cheie primara)

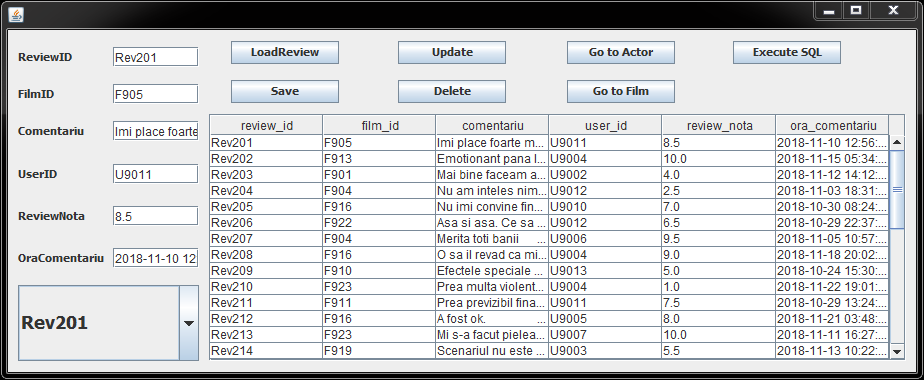
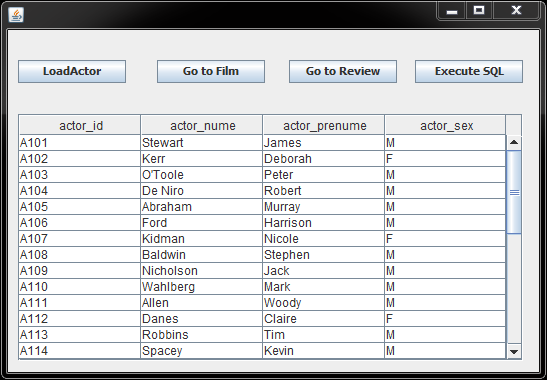
Film : film\_id (cheie primara), regizor\_id (cheie straina)

Review : review\_id (cheie primara), user\_id (cheie straina), film\_id (cheie straina)

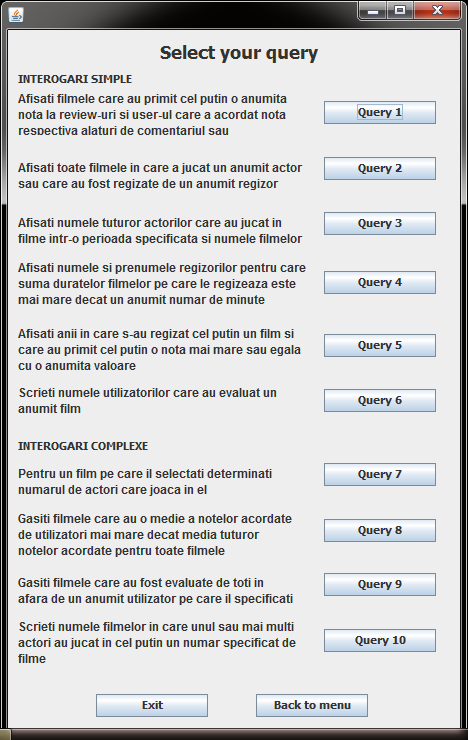
User : user\_id (cheie primara)

 Pentru fiecare dintre tabelele Actor, Film si Review am creat cate o clasa (de tip JFrame) care sa imi permita crearea unei interfete usor de folosit. In figura din dreapta avem interfata pentru clasa ModifActor (permite modificari in cadrul tabelei cu date despre actori). Avem astfel:

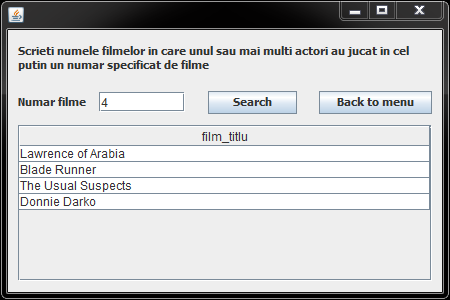
1. Un buton LoadActor care incarca in tabelul din dreapta jos inregistrarile stocate de baze de date la momentul respective
2. Un buton Save care imi permite inserarea unei inregistrari in baza de date prin completarea campurilor
3. Un buton Update care seteaza valoarea unor atribute pentru o inregistrare deja existent
4. Un buton Delete care elimina o inregistrare
5. Un buton de tip JComboBox care permite cautarea unei inregistrari dupa numele actorului in cazul de fata
6. Butoanele “Go to Film” si “Go to Review” redirectioneaza utilizatorul catre interfetele ce ofera informatii despre filme si review-uri
7. Un buton Execute SQL care va afisa o interfata ce permite cautarea unor inregistrari dupa anumite criterii introduse de utilizator intr-un camp sau mai multe



In stanga este o interfata pentru afisarea detaliilor despre actori pentru un utilizator de tip client, consecinta fiind ca nu avem butoanele Save, Delete sau Update.



De asemenea, pentru executarea cererilor SQL fara ca utilizatorul sa aiba acces direct la baza de date (indifferent daca utilizatorul este admin sau clinet) am implementat o clasa numita ExecuteSQLQuery care creaza o interfata simpla. Butoanele Reset si Exit au aceeasi functionalitate ca la LoginCheck. In stanga este descris ce fel de rezultat returneaza executarea cererii corespunzatoare butonului Query din dreapta. Rezultatul cererii este afisat in tabelul (de tip JTable) de jos.

Constrangeri de integritate

1. PRIMARY KEY

Actor : actor\_id

Regizor : regizor\_id

Film\_actori : actor\_id , film\_id

Film : film\_id

Review : review\_id

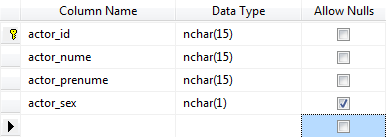
User : user\_id

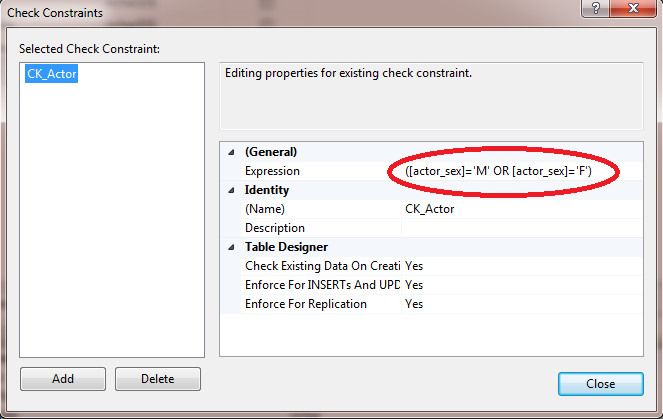
1. FOREIGN KEY

Film : regizor\_id (din tabela Regizor)

Review : user\_id (din tabela User), film\_id (din tabela Film)

1. NOT NULL



In toate tabelele, cel mult doua atribute au selectate optiunea Allow Nulls.

1. CHECK

In tabela Actor, atributul actor\_sex poate fi numai ‘M’ sau ‘F’.

**Interogari simple:**

1) Afisati filmele care au primit cel putin o anumita nota la review-uri si user-ul care a acordat nota respectiva alaturi de comentariul sau

SELECT f.film\_titlu, r.user\_id, r.comentariu

FROM Film f INNER JOIN Review r

ON f.film\_id = r.film\_id

WHERE r.review\_nota = ?

2) Afisati toate filmele in care a jucat un anumit actor sau care au fost regizate de un anumit regizor

SELECT f.film\_titlu

FROM Film\_actori fa INNER JOIN Film f

ON f.film\_id = fa.film\_id

INNER JOIN Actor a

ON a.actor\_id = fa.actor\_id

WHERE a.actor\_nume = ? AND a.actor\_prenume = ?

UNION

SELECT f.film\_titlu

FROM Film f INNER JOIN Regizor r

ON f.regizor\_id = r.regizor\_id

WHERE r.regizor\_nume = ?

3) Afisati numele tuturor actorilor care au jucat in filme intr-o perioada specificata si numele filmelor

SELECT a.actor\_nume, a.actor\_prenume, f.film\_titlu, f.film\_an

FROM Film\_actori fa INNER JOIN Film f

ON f.film\_id = fa.film\_id

INNER JOIN Actor a

ON a.actor\_id = fa.actor\_id

WHERE f.film\_an BETWEEN ? AND ?

4) Afisati numele si prenumele regizorilor pentru care suma duratelor filmelor pe care le regizeaza este mai mare decat un anumit numar de minute

SELECT r.regizor\_nume, r.regizor\_prenume, SUM(f.film\_durata) AS Durata\_Filme

FROM Film f INNER JOIN Regizor r

ON f.regizor\_id = r.regizor\_id

GROUP BY r.regizor\_nume, r.regizor\_prenume

HAVING SUM(f.film\_durata) > ?

ORDER BY SUM(f.film\_durata) DESC

5) Afisati anii in care s-au regizat cel putin un film si care au primit cel putin o nota mai mare decat o anumita valoare

SELECT DISTINCT f.film\_an

FROM Film f INNER JOIN Review r

ON f.film\_id = r.film\_id

GROUP BY f.film\_an, r.review\_nota

HAVING COUNT(\*)>=1 AND r.review\_nota >=?

6) Scrieti numele utilizatorilor care au evaluat un anumit film

SELECT DISTINCT u.username

FROM Review r INNER JOIN [User] u

ON r.user\_id = u.user\_id

INNER JOIN Film f

ON f.film\_id = r.film\_id

WHERE f.film\_titlu = ?

**Interogari complexe:**

1) Pentru un anumit film determinati numarul de actori care joaca in el

SELECT f.film\_titlu, f.film\_id, COUNT(\*) AS NrActori

FROM Film\_actori fa INNER JOIN Film f

ON f.film\_id = fa.film\_id

INNER JOIN Actor a

ON a.actor\_id = fa.actor\_id

WHERE f.film\_id IN (SELECT fa.film\_id FROM Film\_actori fa WHERE a.actor\_id

IN (SELECT a.actor\_id FROM Actor a WHERE EXISTS (SELECT a.actor\_id FROM Actor a WHERE a.actor\_id = fa.actor\_id))) AND f.film\_titlu = ?

GROUP BY f.film\_titlu, f.film\_id

2) Gasiti filmele care au o medie a notelor acordate de utilizatori mai mare decat media tuturor notelor acordate pentru toate filmele

SELECT f.film\_titlu, AVG(r.review\_nota) AS MedieReview,

(SELECT AVG(r.review\_nota) FROM Review r) AS MedieTotal

FROM Film f INNER JOIN Review r

ON f.film\_id = r.film\_id

GROUP BY f.film\_titlu

HAVING AVG(r.review\_nota) > (SELECT AVG(r.review\_nota) FROM Review r)

3) Gasiti filmele care au fost evaluate de toti in afara de un anumit utilizator pe care il specificati

SELECT f.film\_titlu

FROM Film f

WHERE f.film\_id IN

(SELECT r.film\_id

FROM Review r

WHERE r.user\_id NOT IN

(SELECT u.user\_id

FROM [User] u

WHERE u.username = ?))

4) Scrieti numele filmelor in care unul sau mai multi actori au jucat in cel putin un anumit numar de filme

SELECT f.film\_titlu

FROM Film f

WHERE f.film\_id IN

(SELECT fa.film\_id

FROM Film\_actori fa

WHERE fa.actor\_id IN

(SELECT a.actor\_id

FROM Actor a

WHERE a.actor\_id IN

(SELECT fa1.actor\_id

FROM Film\_actori fa1 GROUP BY fa1.actor\_id

HAVING COUNT(fa1.actor\_id)>?)))