Съдържание:

1. Описание на предметната област……………………………………………………………………..2
2. ЕR-модел на базата от данни…………………………………………………………………………....2
3. Релационен модел на базата от данни……………………………………………………….......4
4. Реализация на релационния модел на базата от данни………………………………….8
5. Създаване на заявки…………………………………………………………………………………………36

5.1. Заявки към повече от една релации…………………………………………………….36

5.2. Заявки, включващи обобщаващи функции (като SUM, COUNT, AVG, MIN, MAX)……………………………………………………………………………………………………..40

5.3. Заявки, включващи GROUP BY, HAVING...................................................43

5.4. Заявки, включващи външни съединения и/или подзаявки………………..48

5.5. Заявки, които изискват използването на изрази – CASE изрази;

изрази за дата/час, функции за работа със символни низове и/или

изрази за конвертиране на типа данни……………………………………………………….57

1. Създаване на необходимите:…………………………………………………………………………..66

6.1. Изгледи………………………………………………………………………………………………….66

6.2. Съхранени процедури и/или потребителски дефинирани функции…69

6.3. Тригери………………………………………………………………………………………………….78

7. Бъдеща работа………………………………………………………………………………………………….82

1. Описание на предметната област

* Определяне на областта на приложение, за която базата от данни може да бъде полезна: Моделът включва обекти като филми, актьорите, участващи в тях, режисьори, жанрове и филмови критици. Поддържа се информация за годината и точната дата, когато филмите са излезли на големия екран, времетраенето им, държавите, където се излъчват, езика на аудиото, бокс офиса(спечелените пари от продажбата на кино билети), имената и пола на актьорите, имената на режисьорите, ролите, точното наименование на жанровете, имената на критиците и рейтинга (от 1 до 10), с който са оценили филмите, общия брой получени рейтинги от интернет потребители и средностатистическите рейтинг точки (от 1 до 100), давани от известни критици. Базата от данни е предназначена за филмови любители, които искат да се информират

по-подробно за детайлите около един филм, филмовата история на актьори и режисьори, отношението филми – жанрове, рейтингите на професионални критици и киномани и др.

* Определяне на функционалността на приложението за бази от данни:

Базата от данни за филми съхранява информация за сравняване на филмите по различни критерии : доколко успешен е един филм в интернет пространството и сред критиците, дали неговите приходи отговарят на получения рейтинг, заслужава ли си да бъде видян на база средностатистически оценки, как се развиват актьорите и

дали участват в силни или слаби филми, подобряват или влошават

качеството на филмите си режисьорите, кои филми са спечелили най-много от продажба на билети в кината и запазват ли преднината си с течение на времето, изменя ли се рейтингът на по-старите филми, какви жанрове предпочитат актьорите и режисьорите, кои продукции

са получили най-много оценки и др.

1. ЕR-модел на базата от данни

* Моделиране на данните, съхранявани в базата данни – определяне на обектите, атрибутите, връзките, ограниченията.

Обекти : Movies, Actors, Directors, Genres, Reviewers

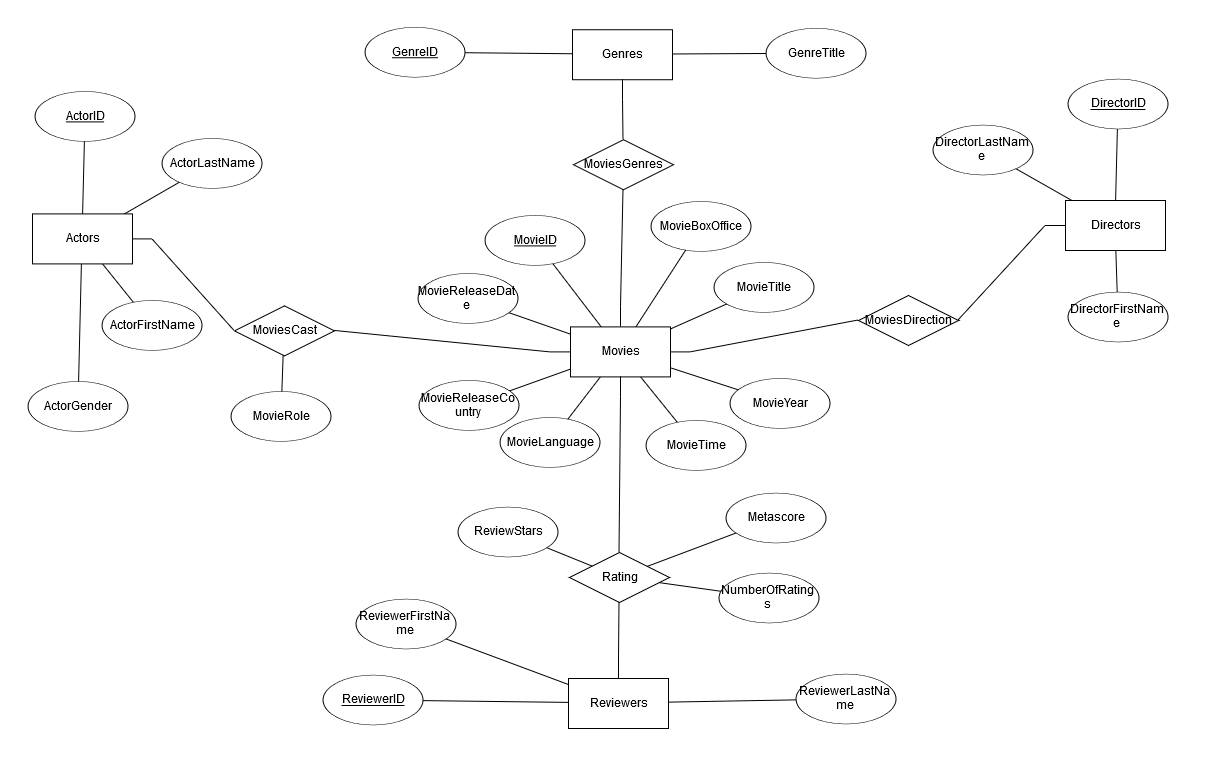
Връзки: MoviesCast, MoviesDirection, MoviesGenres, Rating

Атрибути: MovieID, MovieTitle, MovieYear, MovieTime, MovieLanguage,

MovieReleaseDate, MovieReleaseCountry, MovieBoxOffice за обекта Movies; ActorID, ActorFirstName, ActorLastName, ActorGender за обекта Actors, MovieRole за връзката MoviesCast; DirectorID, DirectorFirstName, DirectorLastName за обекта Directors; GenreID, GenreTitle за обекта Genres; ReviewerID, ReviewerFirstName, ReviewerLastName за обекта Reviewers; ReviewStars, NumberOfRatings, Metascore за връзката Rating

Ограничения: MovieID – първичен ключ за обекта Movies; ActorID – първичен ключ за обекта Actors; DirectorID – първичен ключ за обекта Directors; GenreID – първичен ключ за обекта Genres; ReviewerID – първичен ключ за обекта Reviewers

* Създаване на диаграма на обектите и връзките

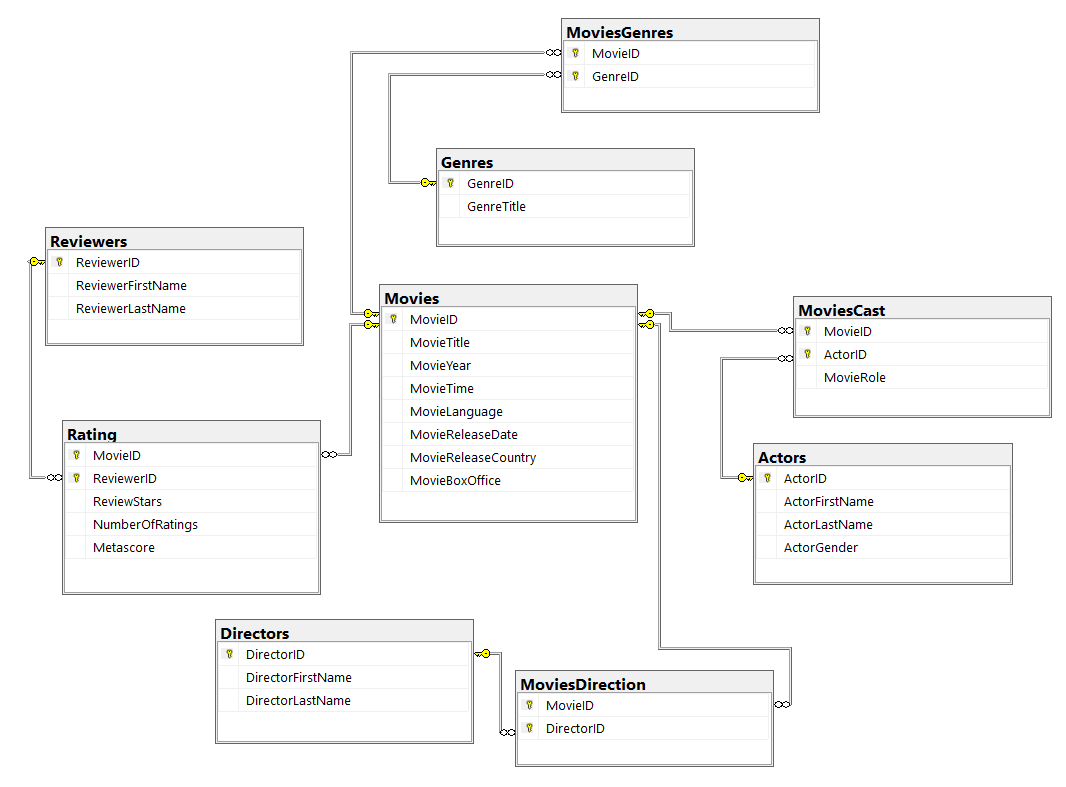


* Описване на обектите, атрибутите, връзките, ограниченията, включени в ЕR-диаграмата.

Обектът Movies се свързва с обекта Actors чрез връзката MoviesCast. Връзката е многостранна, защото в един филм могат да участват повече от един актьори и обратно един актьор може да участва в повече от един филми. Обектът Movies притежава атрибутите MovieID, MovieTitle, MovieYear, MovieTime, MovieLanguage, MovieReleaseDate, MovieReleaseCountry, MovieBoxOffice с ограничение за уникалност първичния ключ MovieID. Обектът Actors съдържа атрибутите ActorID, ActorFirstName, ActorLastName, ActorGender с ограничение за уникалност първичния ключ ActorID. Връзката MoviesCast има един атрибут MovieRole. Обектът Movies се свързва и с още три обекта : Directors чрез връзката MoviesDirection, Genres чрез връзката MoviesGenres и Reviewers чрез връзката Rating. Връзките отново са многостранни, тъй като един филм може да има няколко режисьори (например братя Уашовски за матрицата и Ethan и Joel Coen за Inside of Llewyn Davis), а един режисьор може да режисира повече от един филми, един филм да се съотнесе към няколко жанра, а един жанр да дефинира много филми, един филм да бъде оценен от няколко критици, а един критик да оцени много филми. Обектът Directors притежава атрибутите DirectorID – първичния ключ и ограничение за уникалност, DirectorFirstName и DirectorLastName. Връзката MoviesDirection няма атрибути. Обектът Genres съдържа атрибутите GenreID, който е и неговият първичен ключ и ограничение за уникалност, и GenreTitle. Връзката MoviesGenres няма атрибути. Обектът Reviewers се определя от атрибутите ReviewerID, ReviewerFirstName и ReviewerLastName, като ReviewerID изпълнява ролята на първичен ключ и ограничение за уникалност. Връзката Rating има три атрибута : ReviewStars, NumberOfRatings и Metascore.

1. Релационен модел на базата от данни

* проектиране, нормализиране и завършване на схемата на релационната база от данни
* определяне на схемата на всяка релация – име на релация, нейните атрибути и тип на данни за всеки атрибут
* описване на предназначението за всяка включена релация
* диаграма на релационните връзки между таблиците



Релациите и атрибутите, които съдържат :

Movies (MovieID, MovieTitle, MovieYear, MovieTime, MovieLanguage, MovieReleaseDate, MovieReleaseCountry, MovieBoxOffice)

Actors (ActorID, ActorFirstName, ActorLastName, ActorGender)

MoviesCast (MovieID, ActorID, MovieRole)

Directors (DirectorID, DirectorFirstName, DirectorLastName)

MoviesDirection (MovieID, DirectorID)

Genres (GenreID, GenreTitle)

MoviesGenres (MovieID, GenreID)

Reviewers (ReviewerID, ReviewerFirstName, ReviewerLastName)

Rating (MovieID, ReviewerID, ReviewStars, NumberOfRatings, Metascore)

Таблици, показващи типа на данните за всеки атрибут :

Movies

MovieID PK, integer

MovieTitle varchar (60)

MovieYear integer

MovieTime integer

MovieLanguage varchar (50)

MovieReleaseDate date

MovieReleaseCountry varchar (5)

MovieBoxOffice money

Actors

ActorID PK, integer

ActorFirstName varchar (20)

ActorLastName varchar (20)

ActorGender varchar (1)

MoviesCast

MovieID PK, FK, integer

ActorID PK, FK, integer

Directors

DirectorID PK, integer

DirectorFirstName varchar (30)

DirectorLastName varchar (30)

MoviesDirection

MovieID PK, FK, integer

DirectorID PK, FK, integer

Genres

GenreID PK, integer

GenreTitle varchar (20)

MoviesGenres

MovieID PK, FK, integer

GenreID PK, FK, integer

Reviewers

ReviewerID PK, integer

ReviewerFirstName varchar (20)

ReviewerLastName varchar (20)

Rating

MovieID PK, FK, integer

ReviewerID PK, FK, integer

ReviewStars float

NumberOfRatings integer

Metascore integer

Връзките в ER-модела са преобразувани в релациите MoviesCast, MoviesDirection, MoviesGenres и Rating, а обектите в релациите Movies, Actors, Directors, Genres и Reviewers. Релацията Movies e най-важната в създадената база от данни, защото благодарение на нея базата може да се нормализира. Първичният ключ MovieID от Movies участва в релациите MoviesCast, MoviesDirection, MoviesGenres и Rating като primary и foreign key, заради което може да се осъществи връзка между Movies и MoviesCast, , Movies и MoviesDirection, , Movies и MoviesGenres, , Movies и Rating, Reviewers и Rating, както между MoviesCast, MoviesDirection, MoviesGenres и Rating. АctorID играе ролята на primary и foreign key в релацията MoviesCast и primary key в Actors и свързва Actors с MoviesCast. DirectorID е primary и foreign key в MoviesDirection и primary key в Directors и свързва Directors с MoviesDirection. GenreID e първичен ключ в релацията Genres и primary и foreign key в MoviesGenres, което осъществява връзка между двете релации. ReviewerID като primary key в Reviewers и primary и foreign key в Rating по същия начин позволява свързването на релациите Reviewers и Rating.

1. Реализация на релационния модел на базата от данни.

Създаване на базата данни :

CREATE DATABASE Movies

GO

USE Movies

GO

Създаване на първата таблица Movies :

CREATE TABLE Movies(

MovieID int IDENTITY NOT NULL,

MovieTitle VARCHAR(60) NOT NULL,

MovieYear int NULL,

MovieTime int NOT NULL,

MovieLanguage VARCHAR(50) NOT NULL,

MovieReleaseDate date NOT NULL,

MovieReleaseCountry VARCHAR(5) NOT NULL,

MovieBoxOffice money NOT NULL,

CONSTRAINT UC\_Movies UNIQUE(MovieBoxOffice),

CONSTRAINT PK\_Movies PRIMARY KEY CLUSTERED(MovieID ASC)

)

GO

SET IDENTITY\_INSERT Movies ON

INSERT INTO Movies(MovieID, MovieTitle, MovieYear, MovieTime, MovieLanguage,

MovieReleaseDate, MovieReleaseCountry, MovieBoxOffice)

VALUES(1, 'Inception', 2010, 148, 'English', '2010-07-23', 'USA', 829895144)

INSERT INTO Movies(MovieID, MovieTitle, MovieYear, MovieTime, MovieLanguage,

MovieReleaseDate, MovieReleaseCountry, MovieBoxOffice)

VALUES(2, 'The Bourne Supremacy', 2004, 108, 'English', '2004-12-03', 'USA', 288702062)

INSERT INTO Movies(MovieID, MovieTitle, MovieYear, MovieTime, MovieLanguage,

MovieReleaseDate, MovieReleaseCountry, MovieBoxOffice)

VALUES(3, 'Batman Begins', 2005, 140, 'English', '2005-06-17', 'USA', 373413297)

INSERT INTO Movies(MovieID, MovieTitle, MovieYear, MovieTime, MovieLanguage,

MovieReleaseDate, MovieReleaseCountry, MovieBoxOffice)

VALUES(4, 'V for Vendetta', 2005, 132, 'English', '2006-04-21', 'USA', 132511035)

INSERT INTO Movies(MovieID, MovieTitle, MovieYear, MovieTime, MovieLanguage,

MovieReleaseDate, MovieReleaseCountry, MovieBoxOffice)

VALUES(5, 'Interstellar', 2014, 169, 'English', '2014-11-07', 'USA', 677471339)

....................................................................

INSERT INTO Movies(MovieID, MovieTitle, MovieYear, MovieTime, MovieLanguage,

MovieReleaseDate, MovieReleaseCountry, MovieBoxOffice)

VALUES(250, 'Eiga Koe no katachi', 2016, 130, 'Japanese', '2016-09-17', 'JP', 30490447)

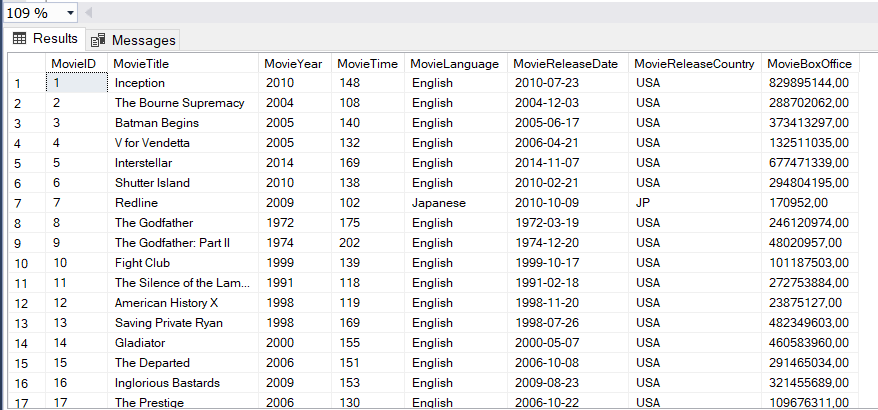
SET IDENTITY\_INSERT Movies OFF

GO

Резултат :

SELECT \*

FROM Movies



Създаваме таблицата с командата CREATE TABLE Movies и задаваме подходящ тип данни за всяка колона, като не допускаме NULL стойности (NOT NULL) с изключение на MovieYear – (NULL). Задаваме ограниченията CONSTRAINT UC\_Movies UNIQUE(MovieBoxOffice) за MovieBoxOffice, което означава, че всеки филм трябва да има различна печалба и CONSTRAINT PK\_Movies PRIMARY KEY CLUSTERED(MovieID ASC) – за стойностите на първичния ключ MovieID във възходящ ред. Със SET IDENTITY\_INSERT Movies ON/OFF позволявамe/отменяме въвеждане на стойности за първичния ключ MovieID, дефиниран от int IDENTITY NOT NULL. Колоните MovieID, MovieYear и MovieTime са от целочислен тип, MovieTitle, MovieLanguage и MovieReleaseCountry - от стрингов, MovieReleaseDate – date , a MovieBoxOffice – money. Таблицата съдържа общо 250 реда. С командата INSERT въвеждаме данни, с DELETE трием данни, а с UPDATE oбновяваме данните във вече създадена таблица.

Примерни команди:

DELETE

FROM Movies

WHERE MovieID = 1

Изтрива всички записи от таблицата Movies, където MovieID = 1

DELETE

FROM Movies

WHERE MovieTitle = 'Avatar'

Изтрива всички записи от таблицата Movies, където MovieTitle = 'Avatar'

DELETE

FROM Movies

WHERE MovieYear = 2000

Изтрива всички записи от таблицата Movies, където MovieYear = 2000

UPDATE Movies

SET MovieTime = 190

WHERE MovieID = 66

Променя MovieTime = 190, където MovieID = 66

UPDATE Movies

SET MovieReleaseCountry = 'USA'

WHERE MovieID = 31

Променя MovieReleaseCountry = 'USA' , където MovieID = 31

Aко искаме да изтрием цялата таблица, използваме командата DROP TABLE Movies

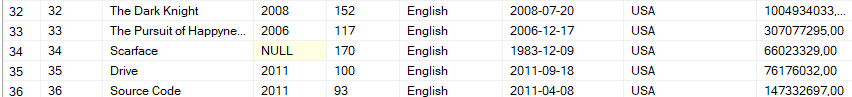
Ако искаме да изтрием колона от таблицата, използваме командите :

ALTER TABLE Movies

DROP COLUMN MovieLanguage

Изтриваме колоната MovieLanguage

NULL стойност – примерен резултат



Създаване на втората таблица Actors :

CREATE TABLE Actors(

ActorID int IDENTITY NOT NULL,

ActorFirstName VARCHAR(20) NOT NULL,

ActorLastName VARCHAR(20) NOT NULL,

ActorGender VARCHAR(1) NULL ,

CONSTRAINT PK\_Actors PRIMARY KEY CLUSTERED(ActorID ASC)

)

GO

SET IDENTITY\_INSERT Actors ON

INSERT INTO Actors(ActorID, ActorFirstName, ActorLastName, ActorGender)

VALUES(1, 'Leonardo', 'DiCaprio', 'M')

INSERT INTO Actors(ActorID, ActorFirstName, ActorLastName, ActorGender)

VALUES(2, 'Joseph', 'Gordon-Levitt', 'M')

INSERT INTO Actors(ActorID, ActorFirstName, ActorLastName, ActorGender)

VALUES(3, 'Ellen', 'Page', 'F')

INSERT INTO Actors(ActorID, ActorFirstName, ActorLastName, ActorGender)

VALUES(4, 'Matt', 'Damon', 'M')

INSERT INTO Actors(ActorID, ActorFirstName, ActorLastName, ActorGender)

VALUES(5, 'Franka', 'Potente', 'F')

....................................................................

INSERT INTO Actors(ActorID, ActorFirstName, ActorLastName, ActorGender)

VALUES(494, 'Aoi', 'Yûki', 'F')

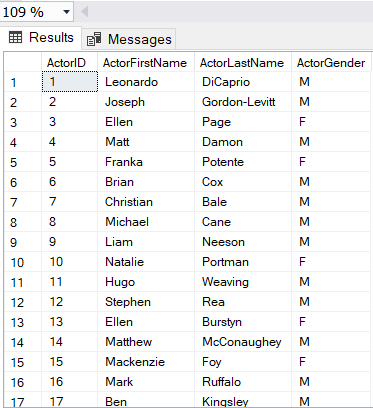
GO

SET IDENTITY\_INSERT Actors OFF

Резултат :

SELECT \*

FROM Actors



Създаваме таблицата с командата CREATE TABLE Actors и задаваме подходящ тип данни за всяка колона, като не допускаме NULL стойности (NOT NULL) с изключение на ActorGender – (NULL). Задаваме ограничениeто CONSTRAINT PK\_Actors PRIMARY KEY CLUSTERED(ActorID ASC) – за стойностите на първичния ключ ActorID във възходящ ред. Със SET IDENTITY\_INSERT Actors ON/OFF позволявамe/отменяме въвеждане на стойности за първичния ключ ActorID, дефиниран от int IDENTITY NOT NULL. Колоната ActorID е от целочислен тип, а ActorFirstName, ActorLastName и ActorGender - от стрингов. Таблицата съдържа общо 494 реда. С командата INSERT въвеждаме данни, с DELETE трием данни, а с UPDATE oбновяваме данните във вече създадена таблица.

Примерни команди:

DELETE

FROM Actors

WHERE ActorID = 20

Изтрива всички записи от таблицата Actors, където ActorID = 20

DELETE

FROM Actors

WHERE ActorFirstName = 'Joaquin'

Изтрива всички записи от таблицата Actors, където ActorFirstName = 'Joaquin'

DELETE

FROM Actors

WHERE ActorGender = 'M'

Изтрива всички записи от таблицата Actors, където актьорите са от мъжки пол

UPDATE Actors

SET ActorFirstName = 'Timothy'

WHERE ActorID = 141

Променя ActorFirstName = 'Timothy', където ActorID = 141

UPDATE Actors

SET ActorFirstName = 'Taku'

WHERE ActorID = 293

Променя ActorFirstName = 'Taku', където ActorID = 293

Aко искаме да изтрием цялата таблица, използваме командата DROP TABLE Actors

Ако искаме да изтрием колона от таблицата, използваме командите :

ALTER TABLE Actors

DROP COLUMN ActorLastName

Изтриваме колоната ActorLastName

NULL стойност – примерен резултат

Екранна снимка (27).png

Създаване на третата таблица MoviesCast :

CREATE TABLE MoviesCast(

MovieID int NOT NULL,

ActorID int NOT NULL,

MovieRole VARCHAR(50) NULL,

CONSTRAINT PK\_MoviesCast PRIMARY KEY CLUSTERED(MovieID ASC, ActorID ASC)

)

GO

INSERT INTO MoviesCast(MovieID, ActorID, MovieRole)

VALUES(1, 1, 'Cobb')

INSERT INTO MoviesCast(MovieID, ActorID, MovieRole)

VALUES(1, 2, 'Arthur')

INSERT INTO MoviesCast(MovieID, ActorID, MovieRole)

VALUES(1, 3, 'Ariadne')

INSERT INTO MoviesCast(MovieID, ActorID, MovieRole)

VALUES(1, 54, 'Eames')

INSERT INTO MoviesCast(MovieID, ActorID, MovieRole)

VALUES(1, 8, 'Miles')

....................................................................

INSERT INTO MoviesCast(MovieID, ActorID, MovieRole)

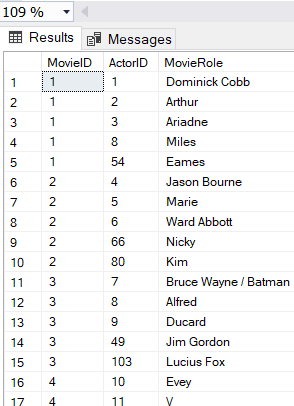
VALUES(250, 494, 'Yuzuru Nishimiya')

GO

Резултат :

SELECT \*

FROM MoviesCast



Създаваме таблицата с командата CREATE TABLE MoviesCast и задаваме подходящ тип данни за всяка колона, като не допускаме NULL стойности (NOT NULL) с изключение на MovieRole – (NULL). Задаваме ограничениeта CONSTRAINT PK\_MoviesCast PRIMARY KEY CLUSTERED(MovieID ASC, ActorID ASC) – за стойностите на първичния ключ, дефиниран от колоните MovieID и ActorID във възходящ ред. Колоните MovieID и ActorID са от целочислен тип, а MovieRole - от стрингов. Таблицата съдържа общо 831 реда. С командата INSERT въвеждаме данни, с DELETE трием данни, а с UPDATE oбновяваме данните във вече създадена таблица.

Примерни команди:

DELETE

FROM MoviesCast

WHERE MovieID = 5

Изтрива всички записи от таблицата MoviesCast, където MovieID = 5

DELETE

FROM MoviesCast

WHERE ActorID = 9

Изтрива всички записи от таблицата MoviesCast, където ActorID = 9

DELETE

FROM MoviesCast

WHERE MovieRole = 'Joker'

Изтрива всички записи от таблицата MoviesCast, където MovieRole = 'Joker'

UPDATE MoviesCast

SET MovieRole = 'Dominick Cobb'

WHERE MovieID = 1 AND ActorID = 1

Променя MovieRole = 'Dominick Cobb', където MovieID = 1 и ActorID = 1

UPDATE MoviesCast

SET MovieRole = 'Jason Bourne'

WHERE MovieID = 27 AND ActorID = 4

Променя MovieRole = 'Jason Bourne', където MovieID = 27 и ActorID = 4

Aко искаме да изтрием цялата таблица, използваме командата DROP TABLE MoviesCast

Ако искаме колона от таблицата, използваме командите :

ALTER TABLE MoviesCast

DROP COLUMN MovieRole

Изтриваме колоната MovieRole

NULL стойност – примерен резултат



Създаване на четвъртата таблица Directors :

CREATE TABLE Directors(

DirectorID int IDENTITY NOT NULL,

DirectorFirstName VARCHAR(30) NULL DEFAULT 'Unknown FirstName',

DirectorLastName VARCHAR(30) NOT NULL,

CONSTRAINT PK\_Directors PRIMARY KEY CLUSTERED (DirectorID ASC)

)

GO

SET IDENTITY\_INSERT Directors ON

INSERT INTO Directors(DirectorID, DirectorFirstName, DirectorLastName)

VALUES(1, 'Christopher', 'Nolan')

INSERT INTO Directors(DirectorID, DirectorFirstName, DirectorLastName)

VALUES(2, 'Paul', 'Greengrass')

INSERT INTO Directors(DirectorID, DirectorFirstName, DirectorLastName)

VALUES(3, 'James', 'McTeigue')

INSERT INTO Directors(DirectorID, DirectorFirstName, DirectorLastName)

VALUES(4, 'Martin', 'Scorsese')

INSERT INTO Directors(DirectorID, DirectorFirstName, DirectorLastName)

VALUES(5, 'Takeshi', 'Koike')

....................................................................

INSERT INTO Directors(DirectorID, DirectorFirstName, DirectorLastName)

VALUES(33, DEFAULT, 'Mangold')

....................................................................

INSERT INTO Directors(DirectorID, DirectorFirstName, DirectorLastName)

VALUES(166, 'Naoko', 'Yamada')

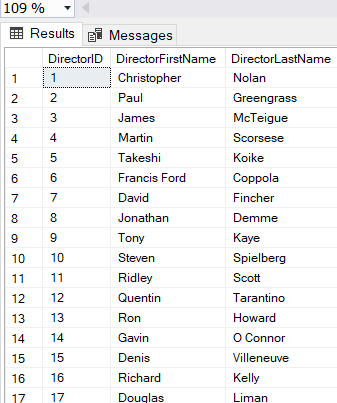
SET IDENTITY\_INSERT Directors OFF

GO

Резултат :

SELECT \*

FROM Directors



Създаваме таблицата с командата CREATE TABLE Directors и задаваме подходящ тип данни за всяка колона, като не допускаме NULL стойности (NOT NULL) с изключение на DirectorFirstName – (NULL), за която командата DEFAULT задава стойността 'Unknown FirstName' при отсъствие на въведени данни. Задаваме ограничението CONSTRAINT PK\_Directors PRIMARY KEY CLUSTERED(DirectorID ASC) – за стойностите на първичния ключ DirectorID във възходящ ред. Със SET IDENTITY\_INSERT Directors ON/OFF позволявамe/отменяме въвеждане на стойности за първичния ключ DirectorID, дефиниран от int IDENTITY NOT NULL. Колоната DirectorID е от целочислен тип, а DirectorFirstName, DirectorLastName - от стрингов. Таблицата съдържа общо 166 реда. С командата INSERT въвеждаме данни, с DELETE трием данни, а с UPDATE oбновяваме данните във вече създадена таблица.

Примерни команди:

DELETE

FROM Directors

WHERE DirectorID = 52

Изтрива всички записи от таблицата Directors, където DirectorID = 52

DELETE

FROM Directors

WHERE DirectorFirstName = 'Stanley'

Изтрива всички записи от таблицата Directors, където DirectorFirstName = 'Stanley'

DELETE

FROM Directors

WHERE DirectorLastName = 'Nolan'

Изтрива всички записи от таблицата Directors, където DirectorLastName = 'Nolan'

UPDATE Directors

SET DirectorFirstName = 'Douglas'

WHERE DirectorID = 17

Променя DirectorFirstName = 'Douglas', където DirectorID = 17

UPDATE Directors

SET DirectorFirstName = 'Ronald'

WHERE DirectorID = 13

Променя DirectorFirstName = 'Ronald', където DirectorID = 13

Aко искаме да изтрием цялата таблица, използваме командата DROP TABLE Directors

Ако искаме да изтрием колона от таблицата, използваме командите :

ALTER TABLE Directors

DROP COLUMN DirectorLastName

Изтриваме колоната DirectorLastName

NULL стойност – примерен резултат



DEFAULT стойност – примерен резултат



Създаване на петата таблица MoviesDirection :

CREATE TABLE MoviesDirection(

MovieID int NOT NULL,

DirectorID int NOT NULL,

CONSTRAINT PK\_MoviesDirection PRIMARY KEY CLUSTERED(MovieID ASC, DirectorID ASC)

)

GO

INSERT INTO MoviesDirection(MovieID, DirectorID)

VALUES(1,1)

INSERT INTO MoviesDirection(MovieID, DirectorID)

VALUES(2,2)

INSERT INTO MoviesDirection(MovieID, DirectorID)

VALUES(3,1)

INSERT INTO MoviesDirection(MovieID, DirectorID)

VALUES(4,3)

INSERT INTO MoviesDirection(MovieID, DirectorID)

VALUES(5,1)

....................................................................

INSERT INTO MoviesDirection(MovieID, DirectorID)

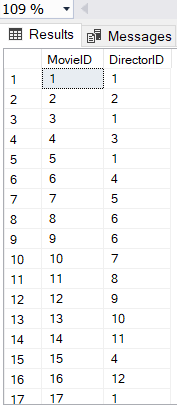
VALUES(250,166)

GO

Резултат :

SELECT \*

FROM MoviesDirection



Създаваме таблицата с командата CREATE TABLE MoviesDirection и задаваме подходящ тип данни за всяка колона, като не допускаме NULL стойности (NOT NULL). Задаваме ограничениeта CONSTRAINT PK\_MoviesDirection PRIMARY KEY CLUSTERED(MovieID ASC, DirectorID ASC) – за стойностите на първичния ключ, дефиниран от колоните MovieID и DirectorID във възходящ ред. Колоните MovieID и DirectorID са от целочислен тип. Таблицата съдържа общо 262 реда. С командата INSERT въвеждаме данни, с DELETE трием данни, а с UPDATE oбновяваме данните във вече създадена таблица.

Примерни команди:

DELETE

FROM MoviesDirection

WHERE MovieID = 10

Изтрива всички записи от таблицата MoviesDirection, където MovieID = 10

DELETE

FROM MoviesDirection

WHERE DirectorID = 65

Изтрива всички записи от таблицата MoviesDirection, където DirectorID = 65

UPDATE MoviesDirection

SET DirectorID = 30

WHERE MovieID = 7

Променя DirectorID = 30, където MovieID = 7

UPDATE MoviesDirection

SET MovieID = 11

WHERE DirectorID = 4

Променя MovieID = 11, където DirectorID = 4

Aко искаме да изтрием цялата таблица, използваме командата DROP TABLE MoviesDirection

Ако искаме колона от таблицата, използваме командите :

ALTER TABLE MoviesDirection

DROP COLUMN DirectorID

Изтриваме колоната DirectorID

Създаване на шестата таблица Genres :

CREATE TABLE Genres(

GenreID int IDENTITY NOT NULL,

GenreTitle VARCHAR(20) NOT NULL,

CONSTRAINT UC\_Genres UNIQUE (GenreTitle),

CONSTRAINT PK\_Genres PRIMARY KEY CLUSTERED(GenreID ASC)

)

GO

SET IDENTITY\_INSERT Genres ON

INSERT INTO Genres(GenreID, GenreTitle)

VALUES(1,'Action')

INSERT INTO Genres(GenreID, GenreTitle)

VALUES(2,'Adventure')

INSERT INTO Genres(GenreID, GenreTitle)

VALUES(3,'Animation')

INSERT INTO Genres(GenreID, GenreTitle)

VALUES(4,'Biography')

INSERT INTO Genres(GenreID, GenreTitle)

VALUES(5,'Comedy')

....................................................................

INSERT INTO Genres(GenreID, GenreTitle)

VALUES(16,'Western')

SET IDENTITY\_INSERT Genres OFF

Резултат :

SELECT \*

FROM Genres



Създаваме таблицата с командата CREATE TABLE Genres и задаваме подходящ тип данни за всяка колона, като не допускаме NULL стойности (NOT NULL) . Задаваме ограниченията CONSTRAINT UC\_Genres UNIQUE(GenreTitle) за GenreTitle, което означава, че всеки жанр има уникално наименование, и CONSTRAINT PK\_Genres PRIMARY KEY CLUSTERED(GenreID ASC) – за стойностите на първичния ключ GenreID във възходящ ред. Колоната GenreID е от целочислен тип, а GenreTitle - от стрингов. Таблицата съдържа общо 16 реда. С командата INSERT въвеждаме данни, с DELETE трием данни, а с UPDATE oбновяваме данните във вече създадена таблица.

Примерни команди:

DELETE

FROM Genres

WHERE GenreID = 8

Изтрива всички записи от таблицата Genres, където GenreID = 8

DELETE

FROM Genres

WHERE GenreTitle = 'Thriller'

Изтрива всички записи от таблицата Genres, където GenreTitle = 'Thriller'

UPDATE Genres

SET GenreTitle = 'Science fiction'

WHERE GenreID = 13

Променя GenreTitle = 'Science fiction', където GenreID = 13

UPDATE Genres

SET GenreTitle = 'History'

WHERE GenreID = 15

Променя GenreTitle = 'History'**,** къдетоGenreID= 15

Aко искаме да изтрием цялата таблица, използваме командата DROP TABLE Genres

Ако искаме да изтрием колона от таблицата, използваме командите :

ALTER TABLE Genres

DROP COLUMN GenreTitle

Изтриваме колоната GenreTitle

Създаване на седмата таблица MoviesGenres :

CREATE TABLE MoviesGenres(

MovieID int NOT NULL,

GenreID int NOT NULL,

CONSTRAINT PK\_MoviesGenres PRIMARY KEY CLUSTERED(MovieID ASC, GenreID ASC)

)

GO

INSERT INTO MoviesGenres(MovieID, GenreID)

VALUES(1,1)

INSERT INTO MoviesGenres(MovieID, GenreID)

VALUES(2,1)

INSERT INTO MoviesGenres(MovieID, GenreID)

VALUES(3,1)

INSERT INTO MoviesGenres(MovieID, GenreID)

VALUES(4,1)

INSERT INTO MoviesGenres(MovieID, GenreID)

VALUES(5,2)

....................................................................

INSERT INTO MoviesGenres(MovieID, GenreID)

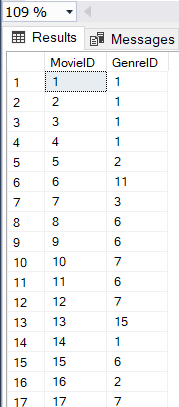
VALUES(250,3)

GO

Резултат :

SELECT \*

FROM MoviesGenres



Създаваме таблицата с командата CREATE TABLE MoviesGenres и задаваме подходящ тип данни за всяка колона, като не допускаме NULL стойности (NOT NULL). Задаваме ограниченията CONSTRAINT PK\_MoviesGenres PRIMARY KEY CLUSTERED(MovieID ASC, GenreID ASC) – за стойностите на първичния ключ, дефиниран от колоните MovieID и GenreID във възходящ ред. Колоните MovieID и GenreID са от целочислен тип. Таблицата съдържа общо 250 реда. С командата INSERT въвеждаме данни, с DELETE трием данни, а с UPDATE oбновяваме данните във вече създадена таблица.

Примерни команди:

DELETE

FROM MoviesGenres

WHERE MovieID = 200

Изтрива всички записи от таблицата MoviesGenres, където MovieID = 200

DELETE

FROM MoviesGenres

WHERE GenreID = 12

Изтрива всички записи от таблицата MoviesGenres, където GenreID = 12

UPDATE MoviesGenres

SET GenreID = 7

WHERE MovieID = 63

Променя GenreID = 7, където MovieID = 63

Aко искаме да изтрием цялата таблица, използваме командата DROP TABLE MoviesGenres

Ако искаме колона от таблицата, използваме командите :

ALTER TABLE MoviesGenres

DROP COLUMN GenreID

Изтриваме колоната GenreID

Създаване на осмата таблица Reviewers :

CREATE TABLE Reviewers(

ReviewerID int IDENTITY NOT NULL,

ReviewerFirstName VARCHAR(20) NOT NULL,

ReviewerLastName VARCHAR(20) NULL DEFAULT 'Unknown LastName',

CONSTRAINT PK\_Reviewers PRIMARY KEY CLUSTERED (ReviewerID ASC)

)

GO

SET IDENTITY\_INSERT Reviewers ON

INSERT INTO Reviewers (ReviewerID, ReviewerFirstName, ReviewerLastName)

VALUES (1, 'Guy', 'Gilbert')

INSERT INTO Reviewers (ReviewerID, ReviewerFirstName, ReviewerLastName)

VALUES (2, 'Kevin', DEFAULT)

INSERT INTO Reviewers (ReviewerID, ReviewerFirstName, ReviewerLastName)

VALUES (3, 'Roberto', 'Tamburello')

INSERT INTO Reviewers (ReviewerID, ReviewerFirstName, ReviewerLastName)

VALUES (4, 'Rob', 'Walters')

INSERT INTO Reviewers (ReviewerID, ReviewerFirstName, ReviewerLastName)

VALUES (5, 'Thierry', 'Harrison')

....................................................................

INSERT INTO Reviewers (ReviewerID, ReviewerFirstName, ReviewerLastName)

VALUES (150, 'Stephanie', 'Conroy')

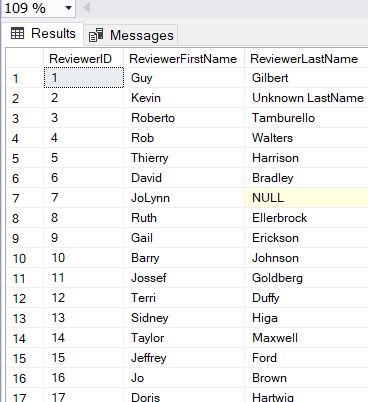
SET IDENTITY\_INSERT Reviewers OFF

GO

Резултат :

SELECT \*

FROM Reviewers



Създаваме таблицата с командата CREATE TABLE Reviewers и задаваме подходящ тип данни за всяка колона, като не допускаме NULL стойности (NOT NULL) с изключение на ReviewerLastName – (NULL), за която командата DEFAULT задава стойността 'Unknown LastName' при отсъствие на въведени данни. Задаваме ограничението CONSTRAINT PK\_Reviewers PRIMARY KEY CLUSTERED(ReviewerID ASC) – за стойностите на първичния ключ ReviewerID във възходящ ред. Със SET IDENTITY\_INSERT Reviewers ON/OFF позволявамe/отменяме въвеждане на стойности за първичния ключ ReviewerID, дефиниран от int IDENTITY NOT NULL. Колоната ReviewerID е от целочислен тип, а ReviewerFirstName, ReviewerLastName - от стрингов. Таблицата съдържа общо 150 реда. С командата INSERT въвеждаме данни, с DELETE трием данни, а с UPDATE oбновяваме данните във вече създадена таблица.

Примерни команди:

DELETE

FROM Reviewers

WHERE ReviewerID = 45

Изтрива всички записи от таблицата Reviewers, където ReviewerID = 45

DELETE

FROM Reviewers

WHERE ReviewerFirstName = 'Kevin'

Изтрива всички записи от таблицата Reviewers, където ReviewerFirstName = 'Kevin'

DELETE

FROM Reviewers

WHERE ReviewerLastName = 'Berndt'

Изтрива всички записи от таблицата Reviewers, където ReviewerLastName = 'Berndt'

UPDATE Reviewers

SET ReviewerLastName = 'Ackerman'

WHERE ReviewerID = 14

Променя ReviewerLastName = 'Ackerman', където ReviewerID = 14

UPDATE Reviewers

SET ReviewerFirstName = 'Robert'

WHERE ReviewerID = 67

Променя ReviewerFirstName = 'Robert', където ReviewerID = 67

Aко искаме да изтрием цялата таблица, използваме командата DROP TABLE Reviewers

Ако искаме да изтрием колона от таблицата, използваме командите :

ALTER TABLE Reviewers

DROP COLUMN ReviewerFirstName

Изтриваме колоната ReviewerFirstName

Създаване на деветата таблица Rating :

CREATE TABLE Rating(

MovieID int NOT NULL,

ReviewerID int NOT NULL,

ReviewStars float NULL ,

NumberOfRatings int NULL,

Metascore int NOT NULL,

CONSTRAINT CHK\_Rating CHECK (ReviewStars BETWEEN 1 AND 10 AND Metascore BETWEEN 0 AND 100),

CONSTRAINT PK\_Rating PRIMARY KEY CLUSTERED (MovieID ASC, ReviewerID ASC)

)

GO

INSERT INTO Rating(MovieID, ReviewerID, ReviewStars, NumberOfRatings, Metascore)

VALUES (1,1, 8.8, 1904938, 74)

INSERT INTO Rating(MovieID, ReviewerID, ReviewStars, NumberOfRatings, Metascore)

VALUES (2,2, 7.7, 414924, 73)

INSERT INTO Rating(MovieID, ReviewerID, ReviewStars, NumberOfRatings, Metascore)

VALUES (3,3, 8.2, 1231839, 70)

INSERT INTO Rating(MovieID, ReviewerID, ReviewStars, NumberOfRatings, Metascore)

VALUES (4,4, 8.2, 975495, 62)

INSERT INTO Rating(MovieID, ReviewerID, ReviewStars, NumberOfRatings, Metascore)

VALUES (5,5, 8.6, 1359593, 74)

....................................................................

INSERT INTO Rating(MovieID, ReviewerID, ReviewStars, NumberOfRatings, Metascore)

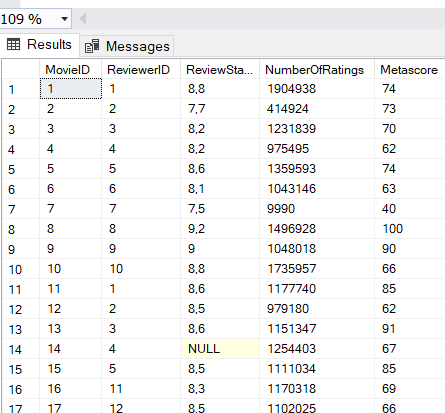
VALUES (250,100, 8.2, 31755, 78)

GO

Резултат :

SELECT \*

FROM Rating



Създаваме таблицата с командата CREATE TABLE Rating и задаваме подходящ тип данни за всяка колона, като не допускаме NULL стойности (NOT NULL) с изключение на ReviewStars и NumberOfRatings – (NULL). Задаваме ограниченията CONSTRAINT CHK\_Rating CHECK (ReviewStars BETWEEN 1 AND 10 AND Metascore BETWEEN 0 AND 100), което проверява дали сме въвели данни между 1 и 10 за колоната ReviewStars и между 0 и 100 за колоната Metascore и CONSTRAINT PK\_Rating PRIMARY KEY CLUSTERED(MovieID ASC, ReviewerID ASC) – за стойностите на първичния ключ, дефиниран от колоните MovieID и ReviewerID във възходящ ред. Колоните MovieID, ReviewerID, NumberOfRatings и Metascroe са от целочислен тип, а ReviewStars – от реален. Таблицата съдържа общо 250 реда. С командата INSERT въвеждаме данни, с DELETE трием данни, а с UPDATE oбновяваме данните във вече създадена таблица.

Примерни команди:

DELETE

FROM Rating

WHERE MovieID = 25

Изтрива всички записи от таблицата Rating, където MovieID = 25

DELETE

FROM Rating

WHERE ReviewerID = 99

Изтрива всички записи от таблицата Rating, където ReviewerID = 99

UPDATE Rating

SET Metascore = 40

WHERE MovieID = 7

Променя Metascore = 40, където MovieID = 7

UPDATE Rating

SET ReviewerID = 130

WHERE MovieID = 220

Променя ReviewerID = 130, където MovieID = 220

UPDATE Rating

SET ReviewStars = 7.4

WHERE ReviewerID = 105

Aко искаме да изтрием цялата таблица, използваме командата DROP TABLE Rating

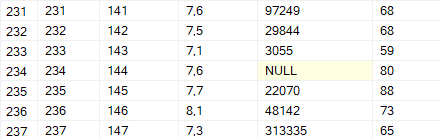
Ако искаме колона от таблицата, използваме командите :

ALTER TABLE Rating

DROP COLUMN Metascore

Изтриваме колоната Metascore

NULL стойност за NumberOfRatings – примерен резултат



След като попълним достатъчно данни, прилагаме FOREIGN KEY и CHECK ограничения за следните таблици : MoviesCast, MoviesDirection, MoviesGenres и Rating с помощта на командата ALTER TABLE :

ALTER TABLE MoviesCast

WITH CHECK ADD CONSTRAINT FK\_MoviesCast\_Movies FOREIGN KEY(MovieID)

REFERENCES Movies(MovieID)

GO

ALTER TABLE MoviesCast

CHECK CONSTRAINT FK\_MoviesCast\_Movies

GO

ALTER TABLE MoviesCast

WITH CHECK ADD CONSTRAINT FK\_MoviesCast\_Actors FOREIGN KEY(ActorID)

REFERENCES Actors(ActorID)

GO

ALTER TABLE MoviesCast

CHECK CONSTRAINT FK\_MoviesCast\_Actors

GO

ALTER TABLE MoviesDirection

WITH CHECK ADD CONSTRAINT FK\_MoviesDirection\_Movies FOREIGN KEY(MovieID)

REFERENCES Movies(MovieID)

GO

ALTER TABLE MoviesDirection

CHECK CONSTRAINT FK\_MoviesDirection\_Movies

GO

ALTER TABLE MoviesDirection

WITH CHECK ADD CONSTRAINT FK\_MoviesDirection\_Directors FOREIGN KEY(DirectorID)

REFERENCES Directors(DirectorID)

GO

ALTER TABLE MoviesDirection

CHECK CONSTRAINT FK\_MoviesDirection\_Directors

GO

ALTER TABLE MoviesGenres

WITH CHECK ADD CONSTRAINT FK\_MoviesGenres\_Movies FOREIGN KEY(MovieID)

REFERENCES Movies(MovieID)

GO

ALTER TABLE MoviesGenres

CHECK CONSTRAINT FK\_MoviesGenres\_Movies

GO

ALTER TABLE MoviesGenres

WITH CHECK ADD CONSTRAINT FK\_MoviesGenres\_Genres FOREIGN KEY(GenreID)

REFERENCES Genres(GenreID)

GO

ALTER TABLE MoviesGenres

CHECK CONSTRAINT FK\_MoviesGenres\_Genres

GO

ALTER TABLE Rating

WITH CHECK ADD CONSTRAINT FK\_Rating\_Movies FOREIGN KEY(MovieID)

REFERENCES Movies(MovieID)

GO

ALTER TABLE Rating

CHECK CONSTRAINT FK\_Rating\_Movies

GO

ALTER TABLE Rating

WITH CHECK ADD CONSTRAINT FK\_Rating\_Reviewers FOREIGN KEY(ReviewerID)

REFERENCES Reviewers(ReviewerID)

GO

ALTER TABLE Rating

CHECK CONSTRAINT FK\_Rating\_Reviewers

GO

Базата вече е завършена и не може да се изтриват таблици с командата DROP TABLE. Ако е нужно да коригираме данни и да създадем таблиците наново, първо трябва да изтрием дефинираните ограничения със следните команди :

ALTER TABLE MoviesCast

DROP CONSTRAINT FK\_MoviesCast\_Actors

ALTER TABLE MoviesCast

DROP CONSTRAINT FK\_MoviesCast\_Movies

ALTER TABLE MoviesDirection

DROP CONSTRAINT FK\_MoviesDirection\_Directors

ALTER TABLE MoviesDirection

DROP CONSTRAINT FK\_MoviesDirection\_Movies

ALTER TABLE MoviesGenres

DROP CONSTRAINT FK\_MoviesGenres\_Genres

ALTER TABLE MoviesGenres

DROP CONSTRAINT FK\_MoviesGenres\_Movies

ALTER TABLE Rating

DROP CONSTRAINT FK\_Rating\_Reviewers

ALTER TABLE Rating

DROP CONSTRAINT FK\_Rating\_Movies

Като сме готови с корекциите, изпълняваме първо посочените ALTER TABLE команди отново.

1. Създаване на заявки

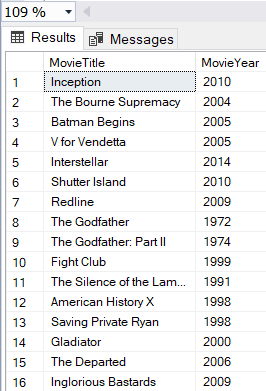
* Заявки към повече от една релации

/\* 1) Заявка, която намира заглавието на филма и годината, когато е излязъл по кината. Заявката обхваща цялата таблица Movies. \*/

SELECT MovieTitle, MovieYear

FROM Movies

Резултат :



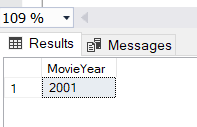
/\* 2) Заявка, която намира годината, когато филмът Donnie Darko е излязъл по кината. \*/

SELECT MovieYear

FROM Movies

WHERE MovieTitle = 'Donnie Darko'

Резултат :



/\* 3) Заявка, която намира заглавията на филмите, които са излезли през 2019 година. Подредени са по азбучен ред. \*/

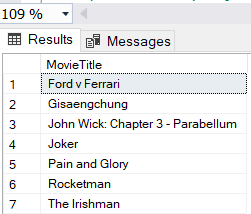
SELECT MovieTitle

FROM Movies

WHERE MovieYear = 2019

ORDER BY MovieTitle

Резултат :



/\* 4) Заявка, която намира заглавията на филмите, които са излезли преди 1980 година. \*/

SELECT MovieTitle

FROM Movies

WHERE MovieYear < 1980

Резултат :



/\* 5) Заявка, която обединява всички данни за фамилията на критиците от таблицата Reviewers с всички заглавия на филмите. \*/

SELECT Reviewers.ReviewerLastName

FROM Reviewers

UNION

(SELECT Movies.MovieTitle

FROM Movies)

Резултат :



/\* 6) Заявка, която намира името и фамилията на критиците с конкатенация от таблицата Reviewers, оценили филми с повече от 9 звезди, при условие че името и фамилията им нямат стойност NULL. \*/

SELECT Reviewers.ReviewerFirstName + ' ' + Reviewers.ReviewerLastName AS ReviewerFullName

FROM Reviewers, Rating

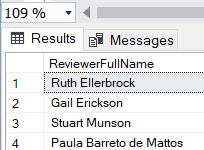
WHERE Rating.ReviewerID = Reviewers.ReviewerID

AND Rating.ReviewStars >= 9

AND Reviewers.ReviewerFirstName IS NOT NULL

AND Reviewers.ReviewerLastName IS NOT NULL

Резултат :



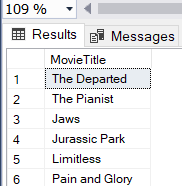
/\* 7) Заявка, която намира заглавията на филмите с ID-та : 15, 70, 120, 190, 210, 235. \*/

SELECT MovieTitle

FROM Movies

WHERE MovieID IN (15, 70, 120, 190, 210, 235)

Резултат :



/\* 8) Заявка, която намира заглавията на филмите, съдържащи думите Stars Wars. Резултатите се подреждат спрямо годината на поява на филмите на големия екран във възходящ ред. \*/

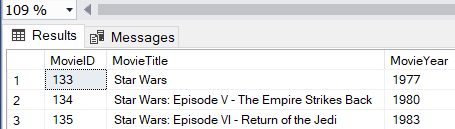
SELECT MovieID, MovieTitle, MovieYear

FROM Movies

WHERE MovieTitle LIKE '%Star%Wars%'

ORDER BY MovieYear ASC

Резултат :



/\* 9) Заявка, която намира ID-то на актьора, чието малко име е Benedict, а фамилията – Cumberbatch. \*/

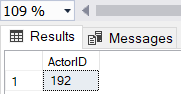
SELECT ActorID

FROM Actors

WHERE ActorFirstName = 'Benedict'

AND ActorLastName = 'Cumberbatch'

Резултат :



* Заявки, включващи обобщаващи функции (като SUM, COUNT, AVG, MIN, MAX)

/\* 1) Заявка, която намира заглавието на най-късия филм, годината, когато е излязъл по кината, името на режисьора му, актьорите, които участват в него и ролите, които са изиграли. \*/

SELECT MovieTitle, MovieYear, DirectorFirstName, DirectorLastName,

ActorFirstName, ActorLastName, MovieRole

FROM Movies

JOIN MoviesDirection

ON MoviesDirection.MovieID = Movies.MovieID

JOIN MoviesCast

ON MoviesCast.MovieID = MoviesDirection.MovieID

JOIN Directors

ON Directors.DirectorID = MoviesDirection.DirectorID

JOIN Actors

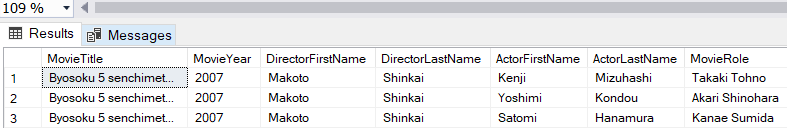
ON Actors.ActorID = MoviesCast.ActorID

WHERE MovieTime =

(SELECT MIN(MovieTime) AS MinMovieTime

FROM Movies)

Резултат :



/\* 2) Заявка, която намира сумата от бокс офиса, спечелен от филмите с режисьор Christopher Nolan. \*/

SELECT SUM(MovieBoxOffice) AS BoxOfficeSum

FROM Movies

JOIN MoviesDirection

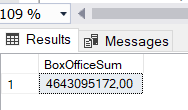
ON MoviesDirection.MovieID = Movies.MovieID

JOIN Directors

ON Directors.DirectorID = MoviesDirection.DirectorID

WHERE DirectorFirstName = 'Christopher' AND DirectorLastName = 'Nolan'

Резултат :

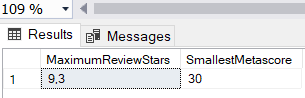


/\* 3) Заявка, която намира най-високия рейтинг и минималния Metascore. \*/

SELECT MAX(ReviewStars) AS MaximumReviewStars, MIN (Metascore) AS SmallestMetascore

FROM Rating

Резултат :



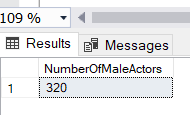
/\* 4) Заявка, която намира броя на актьорите от мъжки пол, при условие че ActorGender няма стойност NULL. \*/

SELECT COUNT(ActorGender) AS NumberOfMaleActors

FROM Actors

WHERE ActorGender = 'M' AND ActorGender IS NOT NULL

Резултат :



/\* 5) Заявка, която намира средния рейтинг на филмите, в които участва актрисата Jodie Foster. \*/

SELECT AVG(ReviewStars) AS AverageReviewStars

FROM Rating

JOIN MoviesCast

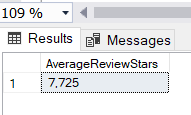
ON MoviesCast.MovieID = Rating.MovieID

JOIN Actors

ON Actors.ActorID = MoviesCast.ActorID

WHERE ActorFirstName = 'Jodie' AND ActorLastName = 'Foster'

Резултат :



/\* 6) Заявка, която намира имената на критика на най-слабо оценения филм и оценката, която е дал със заглавието на филма. \*/

SELECT Reviewers.ReviewerFirstName + ' '

+ Reviewers.ReviewerLastName AS ReviewerFullName,

Movies.MovieTitle, Rating.ReviewStars

FROM Reviewers, Movies, Rating

WHERE Rating.ReviewStars = (

SELECT MIN(Rating.ReviewStars)

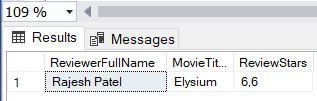
FROM Rating

)

AND Rating.ReviewerID = Reviewers.ReviewerID

AND Rating.MovieID = Movies.MovieID

Резултат :



* Заявки, включващи GROUP BY, HAVING

/\* 1) Заявка, която намира наименованието на жанровете и броя на филмите с определен жанр на режисьора Martin Scorsese. \*/

SELECT GenreTitle, COUNT(GenreTitle) AS NoMovieGenres

FROM Genres

JOIN MoviesGenres

ON MoviesGenres.GenreID = Genres.GenreID

JOIN MoviesDirection

ON MoviesDirection.MovieID = MoviesGenres.MovieID

JOIN Directors

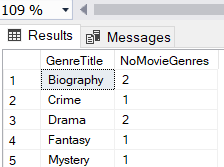
ON Directors.DirectorID = MoviesDirection.DirectorID

WHERE DirectorFirstName = 'Martin' AND DirectorLastName = 'Scorsese'

GROUP BY GenreTitle

ORDER BY GenreTitle

Резултат :



/\* 2) Заявка, която намира наименованието на жанровете, средното времетраене на филмите за съответстващия жанр както и броя на филмите, които се отнасят към определен жанр. \*/

SELECT GenreTitle, AVG(MovieTime) AS AverageMovieTime, COUNT(GenreTitle) AS CountGenreTitle

FROM Movies

JOIN MoviesGenres ON

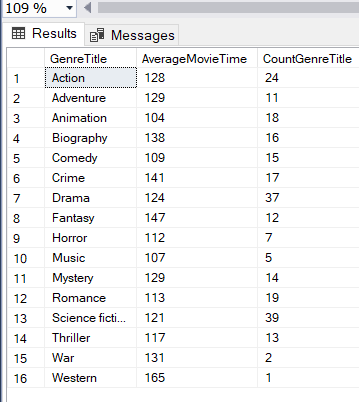
MoviesGenres.MovieID = Movies.MovieID

JOIN Genres ON

Genres.GenreID = MoviesGenres.GenreID

GROUP BY GenreTitle

Резултат :



/\* 3) Заявка, която намира името и фамилията на режисьорите, които са получили средно повече от 8.3 рейтинг звезди и при условие че са режисирали повече от един филм, се извежда броя на направените от тях филми. Резултатите са показани в низходящ ред спрямо получените средно рейтинг звезди. \*/

SELECT DirectorFirstName, DirectorLastName, AVG(ReviewStars) AS AverageReviewStars, COUNT(MoviesDirection.MovieID) AS CountOfMovies

FROM Rating

JOIN MoviesDirection

ON MoviesDirection.MovieID = Rating.MovieID

JOIN Directors

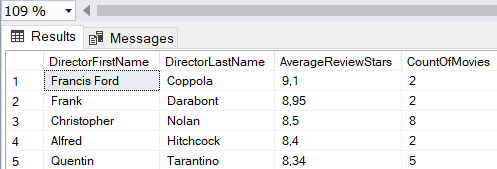
ON Directors.DirectorID = MoviesDirection.DirectorID

GROUP BY DirectorFirstName, DirectorLastName

HAVING AVG (ReviewStars) > 8.3 AND COUNT(MoviesDirection.MovieID) > 1

ORDER BY AverageReviewStars DESC

Резултат :



/\* 4) Заявка, която намира заглавията на филмите, точната дата, когато са излезли на големия екран, държавата, в която е била премиерата им, за които са въведени данни за повече от пет актьори. Резултатите се подреждат в низходящ ред спрямо броя на въведените актьори. \*/

SELECT MovieTitle, MovieReleaseDate, MovieReleaseCountry, COUNT(MoviesCast.ActorID) AS NumberOfActors

FROM Movies

JOIN MoviesCast

ON MoviesCast.MovieID = Movies.MovieID

JOIN Actors

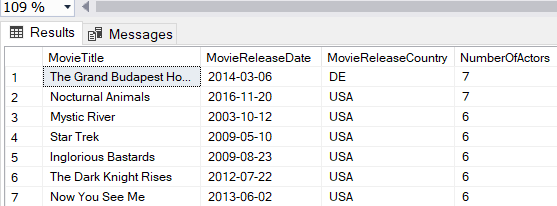
ON Actors.ActorID = MoviesCast.ActorID

GROUP BY MovieTitle, MovieReleaseDate, MovieReleaseCountry

HAVING COUNT(MoviesCast.ActorID) > 5

ORDER BY COUNT(MoviesCast.ActorID) DESC, MovieReleaseDate

Резултат :



/\* 5) Заявка, която намира първите три филма с максимален Metascore, които са оценени от един критик заедно с името и фамилията на критиците. Резултатите се подреждат в низходящ ред спрямо получения брой мета точки. \*/

SELECT TOP(3) MAX(Metascore) AS MaxMetascore, COUNT(Rating.ReviewerID) AS ReviewerIDCount, ReviewerFirstName, ReviewerLastName

FROM Rating

JOIN Reviewers

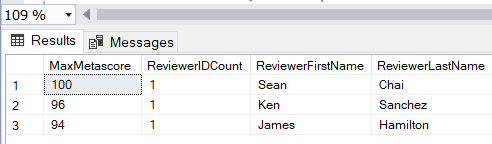
ON Reviewers.ReviewerID = Rating.ReviewerID

GROUP BY ReviewerFirstName, ReviewerLastName

HAVING COUNT(Rating.ReviewerID) = 1

ORDER BY MAX(Metascore) DESC

Резултат :



/\* 6) Заявка, която намира наименованието на жанровете както и броя на филмите, които се отнасят към определен жанр, при условие че има повече от 15 филма, които да спадат към някой от въведените жанрове.\*/

SELECT GenreTitle, COUNT(MoviesGenres.GenreID) AS NumberOfGenres

FROM Genres

JOIN MoviesGenres

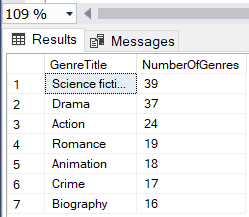
ON MoviesGenres.GenreID = Genres.GenreID

GROUP BY GenreTitle

HAVING COUNT(MoviesGenres.GenreID) > 15

ORDER BY COUNT(MoviesGenres.GenreID) DESC

Резултат :



/\* 7) Заявка, която намира минималния брой рейтинги, дадени от интернет потребители и броя на филмите, които е направил определен режисьор заедно с името и фамилията му, при условие че тези филми са повече от четири. Резултатите се подреждат във възходящ ред спрямо броя на филмите и по азбучен ред спрямо малкото име на режисьора. \*/

SELECT MIN(NumberOfRatings) AS MinNumberOfRatings, COUNT(MoviesDirection.DirectorID) AS CountDirectorID, DirectorFirstName, DirectorLastName

FROM Rating

JOIN MoviesDirection

ON MoviesDirection.MovieID = Rating.MovieID

JOIN Directors

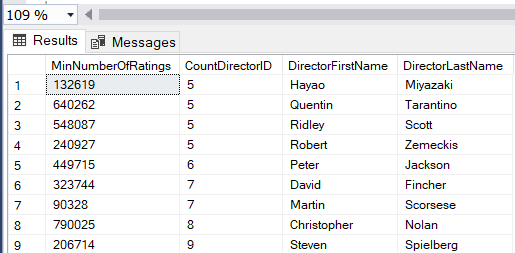
ON Directors.DirectorID = MoviesDirection.DirectorID

GROUP BY DirectorFirstName, DirectorLastName

HAVING COUNT(MoviesDirection.DirectorID) > 4

ORDER BY COUNT(MoviesDirection.DirectorID) ASC, DirectorFirstName ASC

Резултат :



/\* 8) Заявка, която намира заглавията на филмите, в които участват един или повече актьори с роли в повече от осем филма. \*/

SELECT MovieTitle

FROM Movies

WHERE MovieID IN(

SELECT MovieID

FROM MoviesCast

WHERE ActorID IN(

SELECT ActorID

FROM Actors

WHERE ActorID IN(

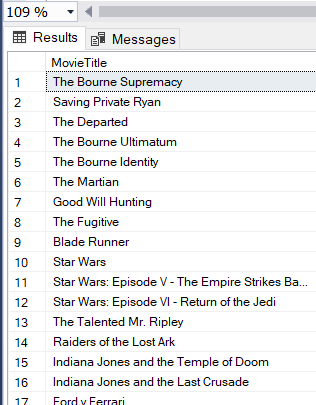
SELECT ActorID

FROM MoviesCast

GROUP BY ActorID

HAVING COUNT(ActorID) > 8)))

Резултат :



* Заявки, включващи външни съединения и/или подзаявки.

/\* 1) Заявка, която намира имената на актьорите както и ролята, която са изиграли във филма The Lord of the Rings : The Return of the King. \*/

SELECT ActorFirstName, ActorLastName, MovieRole

FROM Actors

JOIN MoviesCast

ON Actors.ActorID = MoviesCast.ActorID

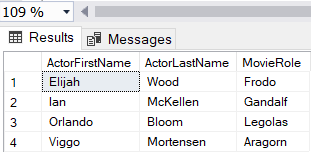
JOIN Movies

ON MoviesCast.MovieID = Movies.MovieID

AND Movies.MovieTitle = 'The Lord of the Rings: The Return of the King'

ORDER BY ActorFirstName ASC

Резултат :



/\* 2) Заявка, която намира имената на режисьора, направил филма Before Sunrise заедно с неговото заглавие. \*/

SELECT DirectorFirstName, DirectorLastName, MovieTitle

FROM Directors

JOIN MoviesDirection

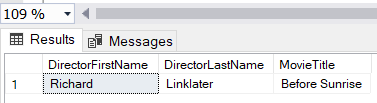
ON MoviesDirection.DirectorID = Directors.DirectorID

JOIN Movies

ON Movies.MovieID = MoviesDirection.MovieID

AND MovieTitle = 'Before Sunrise'

Резултат :



/\* 3) Заявка, която намира имената на режисьора, направил филм, за който се отнася ролята Forrest Gump. \*/

SELECT DirectorFirstName, DirectorLastName, MovieTitle

FROM Directors

JOIN MoviesDirection

ON Directors.DirectorID = MoviesDirection.DirectorID

JOIN Movies

ON MoviesDirection.MovieID = Movies.MovieID

JOIN MoviesCast

ON MoviesCast.MovieID = Movies.MovieID

WHERE MovieRole = 'Forrest Gump'

По друг начин :

SELECT DirectorFirstName, DirectorLastName, MovieTitle

FROM Directors, MoviesDirection, Movies, MoviesCast

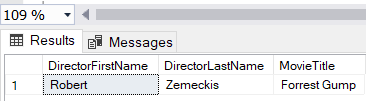
WHERE Directors.DirectorID=MoviesDirection.DirectorID

AND MoviesDirection.MovieID=Movies.MovieID

AND Movies.MovieID=MoviesCast.MovieID

AND MoviesCast.MovieRole= 'Forrest Gump'

Резултат :



/\* 4) Заявка, която намира имената на актьорите заедно със заглавията на филмите, в които са участвали, и годината, когато филмите им са излезли на големия екран. Актьорите не трябва да са участвали във филм през периода 1960 – 2020 година. \*/

SELECT ActorFirstName, ActorLastName, MovieTitle, MovieYear

FROM Actors

JOIN MoviesCast

ON Actors.ActorID = MoviesCast.ActorID

JOIN Movies

ON MoviesCast.MovieID = Movies.MovieID

WHERE MovieYear NOT BETWEEN 1960 AND 2020

ORDER BY MovieYear, ActorFirstName

По друг начин :

SELECT a.ActorFirstName, a.ActorLastName, c.MovieTitle, c.MovieYear

FROM Actors a, MoviesCast b, Movies c

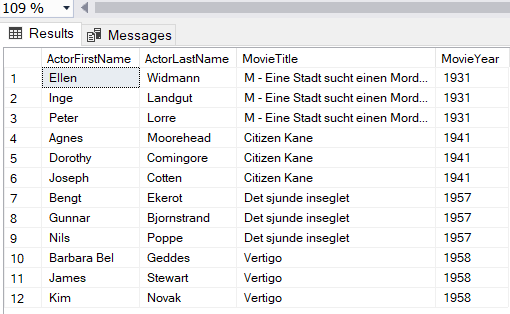
WHERE a.ActorID = b.ActorID

AND b.MovieID = c.MovieID

AND c.MovieYear NOT BETWEEN 1960 AND 2020

ORDER BY MovieYear, ActorFirstName

Резултат :



/\* 5) Заявка, която намира заглавията на филмите от 1980 година и техния жанр. \*/

SELECT MovieTitle, MovieYear, GenreTitle

FROM Movies

JOIN MoviesGenres

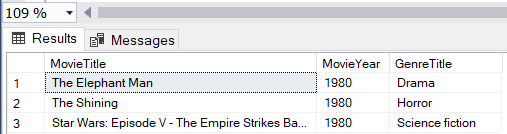
ON MoviesGenres.MovieID = Movies.MovieID

JOIN Genres

ON Genres.GenreID = MoviesGenres.GenreID

WHERE MovieYear = 1980

Резултат :



/\* 6) Заявка, която намира заглавията на филмите, годината, когато са излъчени за пръв път по кината, техния жанр и имената на режисьорите им, при услови че бокс офисът им е по-голям от 1100000000. Резултатите се подреждат в низходящ ред спрямо бокс офиса. \*/

SELECT MovieTitle, MovieYear, GenreTitle, DirectorFirstName, DirectorLastName

FROM Movies

JOIN MoviesGenres

ON MoviesGenres.MovieID = Movies.MovieID

JOIN Genres

ON Genres.GenreID = MoviesGenres.GenreID

JOIN MoviesDirection

ON MoviesDirection.MovieID = MoviesGenres.MovieID

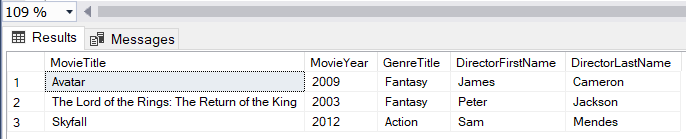
JOIN Directors

ON Directors.DirectorID = MoviesDirection.DirectorID

WHERE MovieBoxOffice > 1100000000

ORDER BY MovieBoxOffice DESC

Резултат :



/\* 7) Заявка, която намира заглавията на филмите, годината, когато са излъчени за пръв път, точната дата на първата им премиера по кината, времетраенето им и имената на режисьорите им. Резултатите се отнасят до филми, излъчени за пръв път от 1 януари до 2010 до 31 декември 2010 включително и се подреждат в низходящ ред спрямо датата на първата им премиера. \*/

SELECT m.MovieTitle, m.MovieYear, m.MovieReleaseDate, m.MovieTime, d.DirectorFirstName, d.DirectorLastName

FROM Movies AS m

JOIN MoviesDirection AS md

ON m.MovieID = md.MovieID

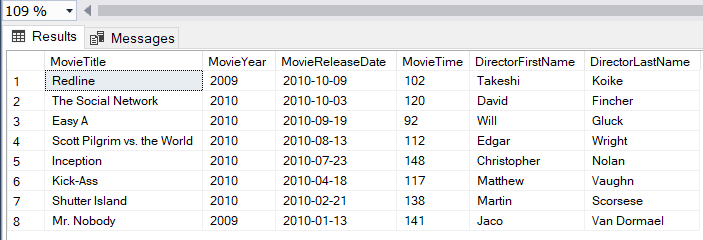
JOIN Directors AS d

ON md.DirectorID = d.DirectorID

WHERE m.MovieReleaseDate > '01/01/2010' AND m.MovieReleaseDate < '12/31/2010'

ORDER BY MovieReleaseDate DESC

Резултат :



/\* 8) Заявка, която намира цялата информация за актьори, които участват във филма The Godfather. \*/

SELECT \*

FROM Actors

WHERE ActorID IN(

SELECT ActorID

FROM MoviesCast

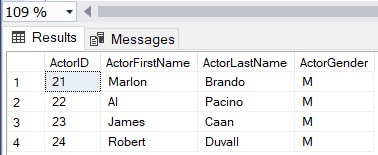
WHERE MovieID IN(

SELECT MovieID

FROM Movies

WHERE MovieTitle = 'The Godfather'))

Резултат :



/\* 9) Заявка, която намира имената на режисьора, който е режисирал филма Gladiator, при условие че съществуват данни за роля в този филм. \*/

SELECT DirectorFirstName, DirectorLastName

FROM Directors

WHERE DirectorID IN(

SELECT DirectorID

FROM MoviesDirection

WHERE MovieID IN(

SELECT MovieID

FROM MoviesCast WHERE MovieRole = ANY(

SELECT MovieRole

FROM MoviesCast

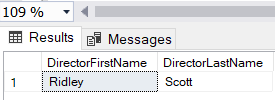
WHERE MovieID IN(

SELECT MovieID

FROM Movies

WHERE MovieTitle = 'Gladiator'))))

Резултат :



/\* 10) Заявка, която намира имената на актьорите, които участват във филми, които са излезли по кината за пръв път в страни различни от САЩ. \*/

SELECT ActorFirstName, ActorLastName

FROM Actors

WHERE ActorID IN(

SELECT ActorID

FROM MoviesCast

WHERE MovieID IN(

SELECT MovieID

FROM Movies

WHERE MovieReleaseCountry <> 'USA'))

Резултат :



/\* 11) Заявка, която намира заглавията на филмите, годината, когато са направени, датата, когато е била първата им премиера, имената на режисьорите им и имената на актьорите, участващи в тях, при условие че фамилията на техния критик има стойност NULL. \*/

SELECT MovieTitle, MovieYear, MovieReleaseDate,

DirectorFirstName, DirectorLastName, ActorFirstName, ActorLastName

FROM Movies a, MoviesDirection b, Directors c,

Rating d, Reviewers e, Actors f, MoviesCast g

WHERE a.MovieID = b.MovieID

AND b.DirectorID = c.DirectorID

AND a.MovieID = d.MovieID

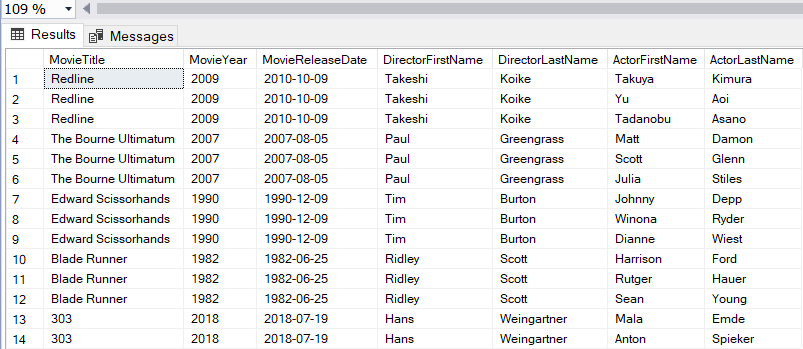
AND d.ReviewerID = e.ReviewerID

AND a.MovieID = g.MovieID

AND g.ActorID = f.ActorID

AND e.ReviewerLastName IS NULL

Резултат :



/\* 12) Заявка, която намира заглавието на всички филми с режисьор David Fincher. \*/

SELECT MovieTitle

FROM Movies

WHERE MovieID IN (

SELECT MovieID

FROM MoviesDirection

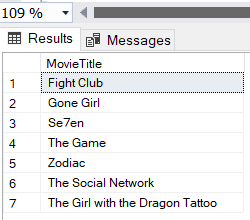
WHERE DirectorID IN (

SELECT DirectorID

FROM Directors

WHERE DirectorFirstName = 'David' AND DirectorLastName = 'Fincher'))

Резултат :



/\* 13) Заявка, която намира всички години, за които има филми, оценени с повече от 8.6 рейтинг звезди. Резултатите се подреждат спрямо годината, когато филмът е направен, във върходящ ред. \*/

SELECT DISTINCT MovieYear

FROM Movies

WHERE MovieID IN(

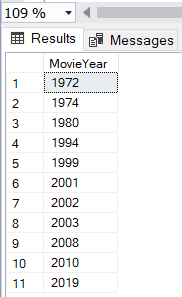
SELECT MovieID

FROM Rating

WHERE ReviewStars > 8.6)

ORDER BY MovieYear

Резултат :



/\* 14) Заявка, която намира малкото име на критиците, за които се отнасят рейтинги със стойност NULL. \*/

SELECT DISTINCT ReviewerFirstName

FROM Reviewers

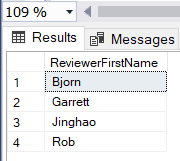
WHERE ReviewerID IN(

SELECT ReviewerID

FROM Rating

WHERE ReviewStars IS NULL)

Резултат :



/\* 15) Заявка, която намира имената на критиците, оценили филма Interstellar. \*/

SELECT DISTINCT Reviewers.ReviewerFirstName, Reviewers.ReviewerLastName

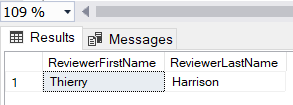
FROM Reviewers, Rating, Movies

WHERE Reviewers.ReviewerID = Rating.ReviewerID

AND Movies.MovieID = Rating.MovieID

AND Movies.MovieTitle = 'Interstellar'

Резултат :



* Заявки, които изискват използването на CASE изрази

/\* 1) Заявка, която дава оценка на рейтинга спрямо точкови критерий. Резултатите се подреждат въз основа на получените рейтинг точки в низходящ ред. \*/

SELECT ReviewStars,

CASE

WHEN ReviewStars BETWEEN 6.6 AND 7.2 THEN 'Very Low'

WHEN ReviewStars BETWEEN 7.3 AND 7.6 THEN 'Low'

WHEN ReviewStars BETWEEN 7.7 AND 8.0 THEN 'Medium'

WHEN ReviewStars BETWEEN 8.1 AND 8.4 THEN 'High'

WHEN ReviewStars BETWEEN 8.5 AND 8.8 THEN 'Very High'

WHEN ReviewStars BETWEEN 8.9 AND 9.1 THEN 'Excellent'

WHEN ReviewStars >= 9.2 THEN 'Perfect'

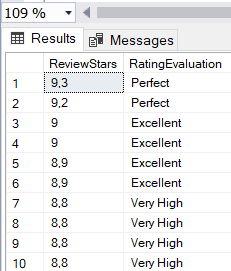
ELSE 'Undefined'

END AS RatingEvaluation

FROM Rating

ORDER BY ReviewStars DESC

Резултат :



/\* 2) Заявка, която дава оценка на метаскора спрямо точкови критерий. Резултатите се подреждат въз основа на получените мета точки във възходящ ред. \*/

SELECT Metascore,

CASE

WHEN Metascore BETWEEN 20 AND 30 THEN 'Awful'

WHEN Metascore BETWEEN 31 AND 40 THEN 'Bad'

WHEN Metascore BETWEEN 41 AND 50 THEN 'Cheap'

WHEN Metascore BETWEEN 51 AND 60 THEN 'Ok'

WHEN Metascore BETWEEN 61 AND 70 THEN 'Good'

WHEN Metascore BETWEEN 71 AND 80 THEN 'Very Good'

WHEN Metascore BETWEEN 81 AND 90 THEN 'Superb'

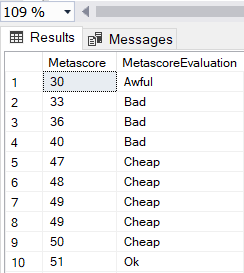
ELSE 'Immaculate'

END AS MetascoreEvaluation

FROM Rating

ORDER BY Metascore ASC

Резултат :



* Заявки, които изискват използването на изрази за дата/час

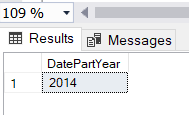
/\* 1) Заявка, която взема годината от MovieReleaseDate за MovieID = 23. \*/

SELECT DATEPART(year, MovieReleaseDate) AS DatePartYear

FROM Movies

WHERE MovieID = 23

Резултат :



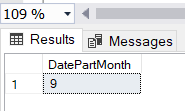
/\* 2) Заявка, която взема месеца от MovieReleaseDate за MovieID = 101. \*/

SELECT DATEPART(month, MovieReleaseDate) AS DatePartMonth

FROM Movies

WHERE MovieID = 101

Резултат :



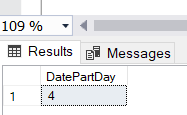
/\* 3) Заявка, която взема деня от MovieReleaseDate за MovieID = 179.\*/

SELECT DATENAME(day, MovieReleaseDate) AS DatePartDay

FROM Movies

WHERE MovieID = 179

Резултат :



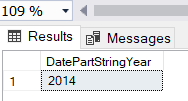
/\* 4) Заявка, която взема годината от MovieReleaseDate за MovieID = 23. \*/

SELECT DATENAME(year, MovieReleaseDate) AS DatePartStringYear

FROM Movies

WHERE MovieID = 23

Резултат :



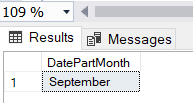
/\* 5) Заявка, която дава наименованието на месеца от MovieReleaseDate за MovieID = 101. \*/

SELECT DATENAME(month, MovieReleaseDate) AS DatePartMonth

FROM Movies

WHERE MovieID = 101

Резултат :



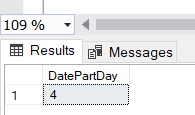
/\* 6) Заявка, която взема деня от MovieReleaseDate за MovieID = 179.\*/

SELECT DATENAME(day, MovieReleaseDate) AS DatePartDay

FROM Movies

WHERE MovieID = 179

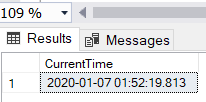
Резултат :



/\* 7) Заявка, която показва точното време към момента на изпълнението й.\*/

SELECT GETDATE() AS CurrentTime

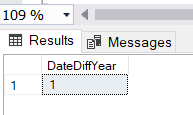
Резултат :



/\* 8) Заявка, която показва разликата в години между две посочени години.\*/

SELECT DATEDIFF(year, '01/01/2019', '01/01/2020') AS DateDiffYear

Резултат :



* Заявки, които изискват използването на функции за работа със символни низове

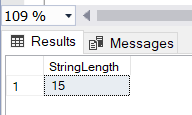
/\* 1) Заявка, която намира дължината на заглавието на филма с ID = 32. \*/

SELECT LEN(MovieTitle) AS StringLength

FROM Movies

WHERE MovieID = 32

Резултат :



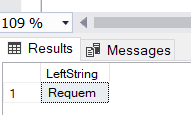
/\* 2) Заявка, която намира първите шест символа от ляво на дясно на заглавието на филма с ID = 65. \*/

SELECT LEFT(MovieTitle, 6) AS LeftString

FROM Movies

WHERE MovieID = 65

Резултат :



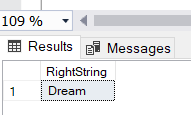
/\* 3) Заявка, която намира първите пет символа от дясно на ляво на заглавието на филма с ID = 65. \*/

SELECT RIGHT(MovieTitle, 5) AS RightString

FROM Movies

WHERE MovieID = 65

Резултат :



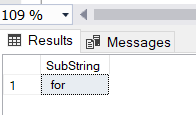
/\* 4) Заявка, която намира символите след 7 от ляво на дясно и след 4 от дясно наляво на заглавието на филма с ID = 65. \*/

SELECT SUBSTRING(MovieTitle, 7,4) AS [SubString]

FROM Movies

WHERE MovieID = 65

Резултат :



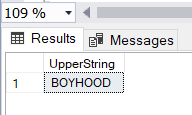
/\* 5) Заявка, която показва заглавието на филма с ID = 207 само с главни букви. \*/

SELECT UPPER(MovieTitle) AS UpperString

FROM Movies

WHERE MovieID = 207

Резултат :



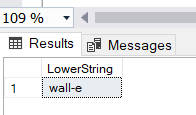
/\* 6) Заявка, която показва заглавието на филма с ID = 242 само малки букви. \*/

SELECT LOWER(MovieTitle) AS LowerString

FROM Movies

WHERE MovieID = 242

Резултат :

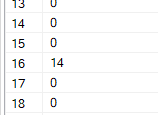


/\* 7) Заявка, която показва индекса в стринга – заглавие на филм, където за пръв път се среща star. \*/

SELECT PATINDEX('%star%', MovieTitle)

FROM Movies

Резултат :



/\* 8) Заявка, която обръща на обратно заглавието на филма с

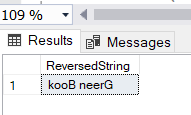
ID = 196. \*/

SELECT REVERSE(MovieTitle) AS ReversedString

FROM Movies

WHERE MovieID = 196

Резултат :



* Заявки, които изискват функции за работа с изрази за конвертиране на данните.

/\* 1) Заявка, която конвертира типа на данните за MovieTime от int във float. \*/

SELECT CAST(MovieTime AS float) AS MovieTimeFloat

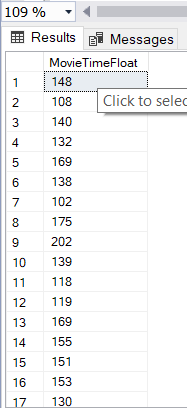
FROM Movies

* По друг начин :

SELECT CONVERT(float, MovieTime) AS MovieTimeFloat

FROM Movies

Резултат :



/\* 2) Заявка, която връща рейтинг оценката като стринг

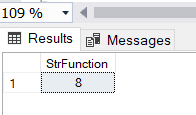
за MovieID = 43. \*/

SELECT STR(ReviewStars) AS StrFunction

FROM Rating

WHERE MovieID = 43

Резултат :



/\* 3) Заявка, която конвертира типа на данните за MovieReleaseDate от date в smalldatetime. \*/

SELECT CONVERT(smalldatetime, MovieReleaseDate) AS Smalldate

FROM Movies

WHERE MovieID = 52

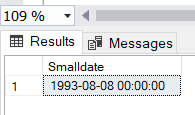
* По друг начин :

SELECT CAST(MovieReleaseDate AS smalldatetime) AS Smalldate

FROM Movies

WHERE MovieID = 52

Резултат :



1. Създаване на необходимите :

* Изгледи

/\* 1) Създаваме изглед с командата CREATE VIEW V\_SpielbergMovies, който съдържа пълната информация за филмите на Steven Spielberg. С командата ALTER VIEW V\_SpielbergMovies ограничаваме информацията за годините от 1990 до 2010. После с командата UPDATE VIEW V\_SpielbergMovies променяме MovieTime = 142 за MovieID = 156 \*/

CREATE VIEW V\_SpielbergMovies AS

SELECT Movies.MovieID, MovieTitle, MovieYear, MovieTime, MovieLanguage, MovieReleaseDate, MovieReleaseCountry, MovieBoxOffice

FROM Movies

INNER JOIN MoviesDirection

ON MoviesDirection.MovieID = Movies.MovieID

INNER JOIN Directors

ON Directors.DirectorID = MoviesDirection.DirectorID

WHERE DirectorFirstName = 'Steven' AND DirectorLastName =

'Spielberg'

WITH CHECK OPTION

ALTER VIEW V\_SpielbergMovies AS

SELECT Movies.MovieID, MovieTitle, MovieYear, MovieTime, MovieLanguage, MovieReleaseDate, MovieReleaseCountry, MovieBoxOffice

FROM Movies

INNER JOIN MoviesDirection

ON MoviesDirection.MovieID = Movies.MovieID

INNER JOIN Directors

ON Directors.DirectorID = MoviesDirection.DirectorID

WHERE DirectorLastName = 'Spielberg' AND MovieYear BETWEEN 1990 AND 2010

WITH CHECK OPTION

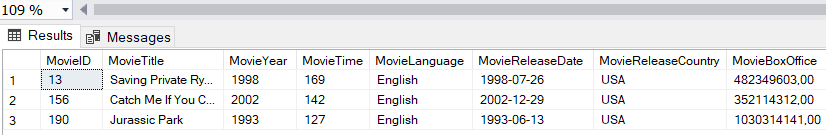
UPDATE V\_SpielbergMovies

SET MovieTime = 142

WHERE MovieID = 156

Крайният резултат :

SELECT \* FROM V\_SpielbergMovies



/\* 2) Създаваме изглед с командата CREATE VIEW V\_JapaneseMovies, който съдържа информация за имената на актьорите, ролите им, оценката на определен критик, броя дадени рейтинги от интернет потребители, мета точките и заглавията на филмите на японски език. С командата ALTER VIEW V\_JapaneseMovies премахваме заглавията на филмите и слагаме наименованията на техните жанрове. После с командата UPDATE VIEW V\_JapaneseMovies премахваме информацията, че някои роли са озвучени от актьорите и поставяме само наименованието на ролята. Със SELECT заявка филтрираме данните за ReviewStars > 8.1 и Metascore >= 80. С командата DROP VIEW V\_JapaneseMovies можем да изтрием изгледа. \*/

CREATE VIEW V\_JapaneseMovies AS

SELECT ActorFirstName, ActorLastName, MovieRole, ReviewStars, NumberOfRatings, Metascore, MovieTitle

FROM Actors

INNER JOIN MoviesCast

ON MoviesCast.ActorID = Actors.ActorID

INNER JOIN Rating

ON Rating.MovieID = MoviesCast.MovieID

INNER JOIN Movies

ON Movies.MovieID = Rating.MovieID

WHERE MovieLanguage = 'Japanese'

WITH CHECK OPTION

ALTER VIEW V\_JapaneseMovies AS

SELECT ActorFirstName, ActorLastName, MovieRole, ReviewStars, Metascore, MovieTitle, GenreTitle

FROM Actors

INNER JOIN MoviesCast

ON MoviesCast.ActorID = Actors.ActorID

INNER JOIN Rating

ON Rating.MovieID = MoviesCast.MovieID

INNER JOIN Movies

ON Movies.MovieID = Rating.MovieID

INNER JOIN MoviesGenres

ON MoviesGenres.MovieID = Movies.MovieID

INNER JOIN Genres

ON Genres.GenreID = MoviesGenres.GenreID

WHERE MovieLanguage = 'Japanese'

WITH CHECK OPTION

UPDATE V\_JapaneseMovies

SET MovieRole = 'Hauru'

WHERE ActorFirstName = 'Taku'

UPDATE V\_JapaneseMovies

SET MovieRole = 'Sonoshee'

WHERE ActorFirstName = 'Yu' AND ActorLastName = 'Aoi'

UPDATE V\_JapaneseMovies

SET MovieRole = 'Frisbee'

WHERE ActorFirstName = 'Tadanobu' AND ActorLastName = 'Asano'

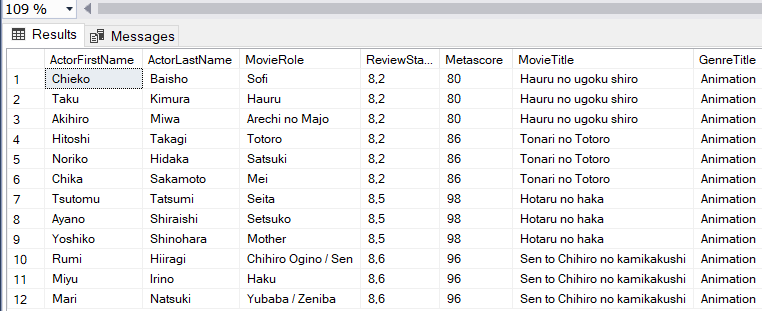
SELECT \*

FROM V\_JapaneseMovies

WHERE ReviewStars > 8.1 AND Metascore >= 80

ORDER BY ReviewStars ASC

Крайният резултат :



/\* 3) Създаваме изглед с командата CREATE VIEW V\_MysteryMovies, който съдържа информация за заглавията на филмите с жанр Mystery, годината им на създаване, езика на който се говори в тях и имената на критиците им с конкатенация . С командата ALTER VIEW V\_MysteryMovies добавяме имената на режисьорите с конкатенация преди ReviewerFullName. Със помощта на SELECT заявка филтрираме данните за тези филми, чийто основен език не е английски. Подреждаме филмите спрямо годината им на излизане в низходящ ред. \*/

CREATE VIEW V\_MysteryMovies AS

SELECT MovieTitle, MovieYear, MovieLanguage, ReviewerFirstName + ' ' +ReviewerLastName AS ReviewerFullName

FROM Movies

INNER JOIN Rating

ON Rating.MovieID = Movies.MovieID

INNER JOIN Reviewers

ON Reviewers.ReviewerID = Rating.ReviewerID

INNER JOIN MoviesGenres

ON MoviesGenres.MovieID = Movies.MovieID

INNER JOIN Genres

ON Genres.GenreID = MoviesGenres.GenreID

WHERE GenreTitle = 'Mystery'

ALTER VIEW V\_MysteryMovies AS

SELECT MovieTitle, MovieYear, MovieLanguage, DirectorFirstName + ' ' +DirectorLastName AS DirectorFullName, ReviewerFirstName + ' ' +ReviewerLastName AS ReviewerFullName

FROM Movies

INNER JOIN MoviesDirection

ON MoviesDirection.MovieID = Movies.MovieID

INNER JOIN Directors

ON Directors.DirectorID = MoviesDirection.DirectorID

INNER JOIN Rating

ON Rating.MovieID = Movies.MovieID

INNER JOIN Reviewers

ON Reviewers.ReviewerID = Rating.ReviewerID

INNER JOIN MoviesGenres

ON MoviesGenres.MovieID = Movies.MovieID

INNER JOIN Genres

ON Genres.GenreID = MoviesGenres.GenreID

WHERE GenreTitle = 'Mystery'

WITH CHECK OPTION

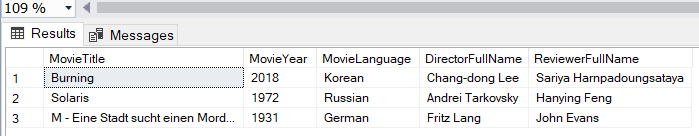
SELECT \*

FROM V\_MysteryMovies

WHERE MovieLanguage NOT IN('English')

ORDER BY MovieYear DESC

Крайният резултат :



* Съхранени процедури

/\* 1) Процедура, която връща пълната информация за филм до 10 символа. Процедурата се отнася до таблицата Movies, а нейният параметър @MovieTitle е обвързан с MovieTitle. Изпълняваме процедурата с командата EXEC за филма K-PAX. \*/

CREATE PROCEDURE MoviesShortString @MovieTitle VARCHAR(10)

AS

SELECT \*

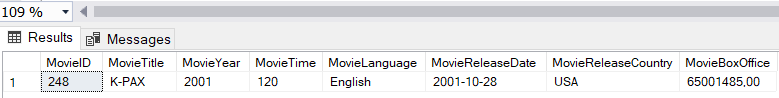
FROM Movies

WHERE MovieTitle = @MovieTitle

GO

Резултат :

EXEC MoviesShortString @MovieTitle = 'K-PAX'



/\* 2) Процедура (без параметри), която връща заглавието на филмите, имената на режисьорите им с конкатенация, рейтинга, даден от критиците и мета точките, при условие че бокс офисът им е между 100000 и 10000000. С командата ALTER PROCEDURE SmallBoxOffice добавяме ново условие – рейтингът на тези филми да бъде по-голям от 7.5. Подреждаме резултатите в низходящ ред спрямо получения рейтинг. Можем да изтрием процедурата с командата DROP PROCEDURE SmallBoxOffice. \*/

CREATE PROCEDURE SmallBoxOffice AS

SELECT MovieTitle, DirectorFirstName + ' ' +DirectorLastName AS

DirectorFullName, ReviewStars, Metascore

FROM Movies

JOIN MoviesDirection

ON MoviesDirection.MovieID = Movies.MovieID

JOIN Directors

ON Directors.DirectorID = MoviesDirection.DirectorID

JOIN Rating

ON Rating.MovieID = Movies.MovieID

WHERE MovieBoxOffice BETWEEN 100000 AND 10000000

ORDER BY ReviewStars DESC

ALTER PROCEDURE SmallBoxOffice AS

SELECT MovieTitle, DirectorFirstName + ' ' +DirectorLastName AS

DirectorFullName, ReviewStars, Metascore

FROM Movies

JOIN MoviesDirection

ON MoviesDirection.MovieID = Movies.MovieID

JOIN Directors

ON Directors.DirectorID = MoviesDirection.DirectorID

JOIN Rating

ON Rating.MovieID = Movies.MovieID

WHERE MovieBoxOffice BETWEEN 100000 AND 10000000

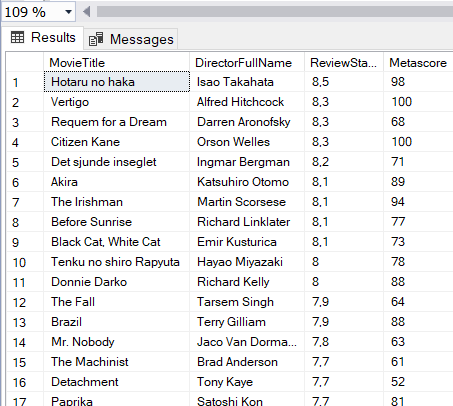
AND ReviewStars > 7.5

ORDER BY ReviewStars DESC

DROP PROCEDURE SmallBoxOffice

Резултат :

EXEC SmallBoxOffice



/\* 3) Процедура (без параметри), която връща ID-то на филмите, тяхното заглавие и годината им на създаване за тези филми, за които фамилията на критика им има стойност NULL. \*/

CREATE PROCEDURE UnknownReviewerLastName AS

SELECT MovieID, MovieTitle, MovieYear

FROM Movies

WHERE MovieID IN(

SELECT MovieID

FROM Rating

WHERE ReviewerID IN(

SELECT ReviewerID

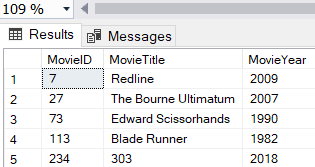
FROM Reviewers

WHERE ReviewerLastName IS NULL))

EXEC UnknownReviewerLastName

RETURN @@ROWCOUNT

Резултат :



/\* 4) Процедура (с параметър), която връща заглавието на филмите на Robert Zemeckis, основния им език, датата когато са излезли по кината и имената на актьорите, участващи в тях с конкатенация. Изпълняваме процедурата за актьора с ID = 197. По втория начин ID = 197 се декларира още в процедурата при параметъра @ActorID и се изпълнява чрез DEFAULT. \*/

CREATE PROCEDURE ActorsProcedure

@ActorID int

AS

SELECT MovieTitle, MovieLanguage, MovieReleaseDate, ActorFirstName + ' ' + ' ' +ActorLastNameAS ActorFullName

FROM Movies

JOIN MoviesDirection

ON MoviesDirection.MovieID = Movies.MovieID

JOIN Directors

ON MoviesDirection.DirectorID = Directors.DirectorID

JOIN MoviesCast

ON MoviesCast.MovieID = Movies.MovieID

JOIN Actors

ON Actors.ActorID = MoviesCast.ActorID

WHERE DirectorLastName = 'Zemeckis'

AND Actors.ActorID = @ActorID

ORDER BY MovieReleaseDate

EXEC ActorsProcedure 197

* По друг начин

ALTER PROCEDURE ActorsProcedure

@ActorID int = 197

AS

SELECT MovieTitle, MovieLanguage, MovieReleaseDate, ActorFirstName + ' ' + ' ' +ActorLastName

AS ActorFullName

FROM Movies

JOIN MoviesDirection

ON MoviesDirection.MovieID = Movies.MovieID

JOIN Directors

ON MoviesDirection.DirectorID = Directors.DirectorID

JOIN MoviesCast

ON MoviesCast.MovieID = Movies.MovieID

JOIN Actors

ON Actors.ActorID = MoviesCast.ActorID

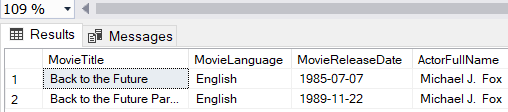
WHERE DirectorLastName = 'Zemeckis'

AND Actors.ActorID = @ActorID

ORDER BY MovieReleaseDate

EXEC ActorsProcedure DEFAULT

Резултат :



/\* 5) Процедура (с параметър), която връща всички данни за режисьорите, ако DirectorFirstName няма стойност NULL. \*/

CREATE PROCEDURE DirectorsProcedure

@DirectorFirstName VARCHAR(20) = NULL

AS

IF @DirectorFirstName IS NOT NULL

SELECT \*

FROM Directors

WHERE DirectorFirstName LIKE @DirectorFirstName + '%'

ELSE

SELECT \*

FROM Directors

RETURN @@ROWCOUNT

EXECUTE DirectorsProcedure

Резултат :



/\* 6) Процедура (с параметър), която връща ID-то, заглавието, оценката от критика, броя на получени рейтинги от интернет потребители и мета точките на филмите с рейтинг > 8.5. В примера информацията важи за филма с ID = 78 \*/

CREATE PROCEDURE TopRatedMovies

@MovieID int

AS

SELECT m.MovieID, m.MovieTitle, r.ReviewStars, r.NumberOfRatings, r.Metascore

FROM Movies AS m

JOIN Rating AS r

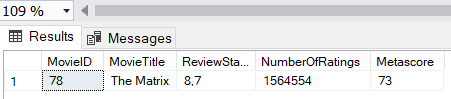
ON r.MovieID = m.MovieID

WHERE ReviewStars > 8.5

AND m.MovieID = @MovieID

EXEC TopRatedMovies 78

Резултат :



/\* 7) Процедура (с параметри), която връща мета точките на филма с ID = 115. За целта използваме два параметъра @MovieID int и @Metascore int OUTPUT. Декларираме @MovieID int, изпълняваме за MovieID = 115 и избираме @Metascore като HighMetascore. \*/

CREATE PROCEDURE BestMovies

@MovieID int, @Metascore int OUTPUT

AS

SELECT @Metascore = Metascore

FROM Rating

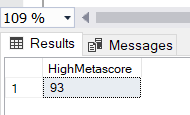
WHERE MovieID = @MovieID

DECLARE @Metascore int

EXEC BestMovies 115, @Metascore OUTPUT

SELECT @Metascore AS HighMetascore

Резултат :



* Потребителски дефинирани функции

/\* 1) Функция, която форматира дадена дата от типа date в типа smalldatetime. \*/

CREATE FUNCTION fnFormatMovieReleaseDate

(@sdtMovieReleaseDate date)

RETURNS smalldatetime

AS

BEGIN

DECLARE @sdtDateFormat smalldatetime

SET @sdtDateFormat = @sdtMovieReleaseDate

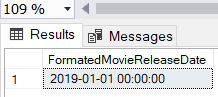
RETURN @sdtDateFormat

END

SELECT dbo.fnFormatMovieReleaseDate

('01/01/2019') AS FormatedMovieReleaseDate

Резултат :



/\* 2) Функция, която връща заглавията на филмите заедно година, в която са излезли на големия екран, основния им език и бокс офиса им, в които е участвал актьор с ID = 7. Функцията се изпълнява под формата на таблица. \*/

ALTER FUNCTION fnMoviesByActor

(@iActorID int)

RETURNS table

AS

RETURN

( SELECT m.MovieTitle, m.MovieYear, m.MovieLanguage, m.MovieBoxOffice

FROM Movies AS m

JOIN MoviesCast AS mc

ON mc.MovieID = m.MovieID

JOIN Actors AS a

ON a.ActorID = mc.ActorID

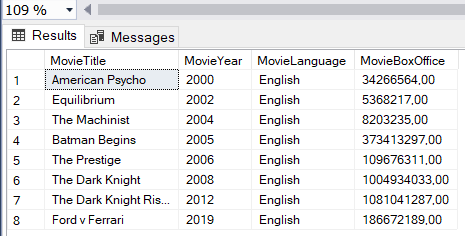
WHERE a.ActorID = @iActorID )

SELECT \*

FROM fnMoviesByActor (7)

ORDER BY MovieYear

Резултат :



/\* 3) Функция, която връща заглавието на филмите заедно с годината, в която са излезли на големия екран, основния им език и бокс офиса им, ID на актьорите, участвали в тях и имената им. Функцията се изпълнява под формата на таблица. SELECT заявката се отнася до функцията и дава информация за филм със заглавие Tonari no totoro. \*/

CREATE FUNCTION fnMoviesActors

(@sMovieTitle VARCHAR(30))

RETURNS table

AS

RETURN

( SELECT m.MovieTitle, m.MovieYear, m.MovieLanguage, m.MovieBoxOffice, a.ActorID, a.ActorFirstName, a.ActorLastName

FROM Movies AS m

JOIN MoviesCast AS mc

ON mc.MovieID = m.MovieID

JOIN Actors AS a

ON a.ActorID = mc.ActorID

WHERE m.MovieTitle = @sMovieTitle)

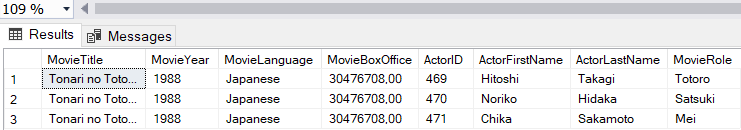
SELECT fn.\*, mc.MovieRole

FROM fnMoviesActors ('Tonari no totoro') fn

INNER JOIN MoviesCast mc

ON mc.ActorID = fn.ActorID

Резултат :



/\* 4) Функция, която връща заглавието на филмите заедно година, в която са излезли на големия екран както и основния им език. SELECT заявката се отнася до функцията и дава информация за 1998 година. \*/

CREATE FUNCTION fnMyFavouriteMovies

(@iMovieYear int)

RETURNS @tblFavouriteMovies table

(MovieTitle VARCHAR(30),

MovieYear int,

MovieLanguage VARCHAR(20))

AS

BEGIN

INSERT INTO @tblFavouriteMovies

(MovieTitle, MovieYear, MovieLanguage)

SELECT MovieTitle, MovieYear, MovieLanguage

FROM Movies

WHERE MovieYear = @iMovieYear

ORDER BY MovieYear ASC

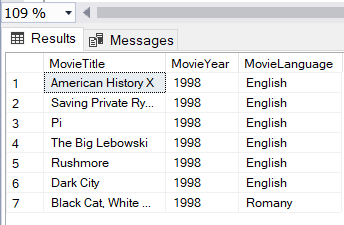
RETURN

END

SELECT \*

FROM fnMyFavouriteMovies(1998)

Резултат :



/\* 5) Функция, която задава стойност Unknown Language за MovieLanguage, ако липсва такава. \*/

CREATE FUNCTION fnMyFavouriteMovies

(@iMovieYear int)

RETURNS @tblFavouriteMovies table

(MovieTitle VARCHAR(30),

MovieYear int,

MovieLanguage VARCHAR(20))

AS

BEGIN

INSERT INTO @tblFavouriteMovies

(MovieTitle, MovieYear, MovieLanguage)

SELECT MovieTitle, MovieYear, MovieLanguage

FROM Movies

WHERE MovieYear = @iMovieYear

ORDER BY MovieYear ASC

IF NOT EXISTS (SELECT \* FROM @tblFavouriteMovies)

INSERT INTO @tblFavouriteMovies (MovieLanguage)

VALUES('Unknown Language')

RETURN

END

* Тригери

/\* 1) Тригер, който забранява изтриването на данни от таблицата Directors. По същия начин могат да се зададат такива тригери и за останалите таблици. \*/

CREATE TRIGGER No\_DeleteDirectors

ON Directors

INSTEAD OF DELETE

AS

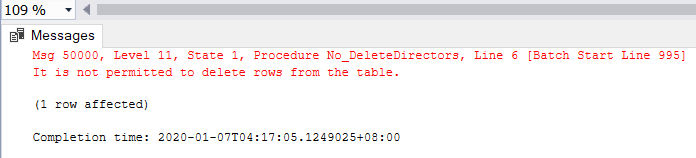
IF @@ROWCOUNT = 0 RETURN

RAISERROR('It is not permitted to delete rows from the table.',11,1)

DELETE FROM Directors

WHERE DirectorID = 1

Резултат :



/\* 2) Тригер, който забранява въвеждането на данни в таблицата MoviesDirection, ако няма валиден MovieID. \*/

CREATE TRIGGER MovieDirector

ON MoviesDirection

FOR INSERT,

UPDATE

AS

IF @@ROWCOUNT = 0 RETURN

IF EXISTS

( SELECT \* FROM inserted i WHERE

i.MovieID NOT IN

(SELECT m.MovieID FROM Movies m)

)

BEGIN

RAISERROR('MoviesDirection must have a valid MovieID.', 16, 1)

ROLLBACK TRANSACTION

END

/\* 3) Тригер, който забранява изтриването на данните за филм, който притежава рейтинг. \*/

CREATE TRIGGER RatingHasMovieID

ON Rating

FOR DELETE

AS

IF @@ROWCOUNT = 0 RETURN

IF EXISTS

( SELECT \* FROM

Deleted d

INNER JOIN Movies m ON d.MovieID = m.MovieID )

BEGIN

RAISERROR('The movie has a rating. Deletion is not completed!', 16, 1)

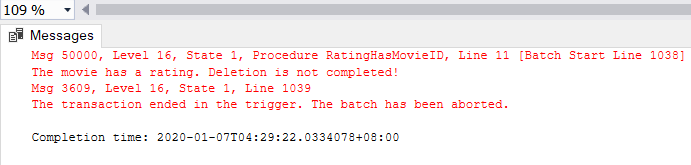
ROLLBACK TRANSACTION

END

DELETE FROM Rating

WHERE MovieID = 250

Резултат :



/\* 4) DDL Тригер, който записва информация в log таблица за изпълнените действия за дефиниране на данни, които проследява. Използваме функцията EventData().\*/

CREATE TABLE EvtLog

(

PostTime DATETIME,

LoginName NVARCHAR(100),

EventType NVARCHAR(100),

TSQLCommand NVARCHAR(2000)

)

GO

CREATE TRIGGER trPreventTblChange

ON DATABASE

FOR ALTER\_TABLE

AS

DECLARE @Data XML

SET @Data = EventData()

INSERT EvtLog (PostTime, LoginName, EventType, TSQLCommand)

VALUES

(GETDATE(),

CONVERT(NVARCHAR(100), CURRENT\_USER),

@Data.value('(/EVENT\_INSTANCE/EventType)[1]', 'nvarchar(100)'),

@Data.value('(/EVENT\_INSTANCE/TSQLCommand)[1]', 'nvarchar(2000)') ) ;

GO

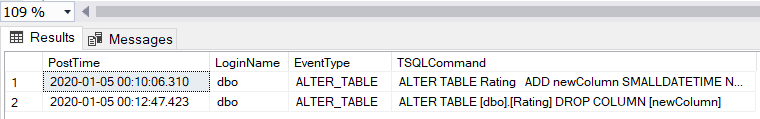
ALTER TABLE Rating

ADD newColumn SMALLDATETIME NULL

SELECT \*

FROM Evtlog

Резултат :



/\* 5) DDL тригер, използващ функцията ЕventData() и базиращ се върху таблицата index\_logs. Предоставя се информация за действието, стартирало DDL тригера. Връща се стойност от тип XML. XML схемата включва информация за времето, когато действието се е изпълнило – PostTime, идентификатора на системния процес SPID на конекцията, от която тригерът се изпълнява, типа на действието EventType, стартирало тригера и други. Създаваме NONCLUSTERED индекси за таблицата Actors. \*/

CREATE TABLE index\_logs (

log\_id INT IDENTITY PRIMARY KEY,

event\_data XML NOT NULL,

changed\_by SYSNAME NOT NULL

)

GO

CREATE TRIGGER trg\_index\_changes

ON DATABASE

FOR

CREATE\_INDEX,

ALTER\_INDEX,

DROP\_INDEX

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO index\_logs (

event\_data,

changed\_by

)

VALUES (

EVENTDATA(),

USER

)

END

GO

CREATE NONCLUSTERED INDEX nidx\_fname

ON Actors(ActorFirstName);

GO

CREATE NONCLUSTERED INDEX nidx\_lname

ON Actors(ActorLastName);

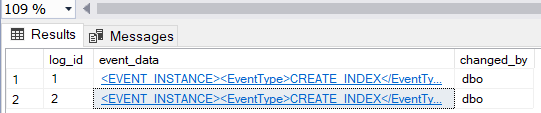
GO

SELECT

\*

FROM index\_logs

Резултат :



Като кликнем върху един от сините линкове, получаваме :

<EVENT\_INSTANCE>

<EventType>CREATE\_INDEX</EventType>

<PostTime>2020-01-05T00:29:11.573</PostTime>

<SPID>54</SPID>

<ServerName>LAPTOP-HJMIKUD0</ServerName>

<LoginName>LAPTOP-HJMIKUD0\Plamenna Petrova</LoginName>

<UserName>dbo</UserName>

<DatabaseName>Movies</DatabaseName>

<SchemaName>dbo</SchemaName>

<ObjectName>nidx\_fname</ObjectName>

<ObjectType>INDEX</ObjectType>

<TargetObjectName>Actors</TargetObjectName>

<TargetObjectType>TABLE</TargetObjectType>

<TSQLCommand>

<SetOptions ANSI\_NULLS="ON" ANSI\_NULL\_DEFAULT="ON" ANSI\_PADDING="ON" QUOTED\_IDENTIFIER="ON" ENCRYPTED="FALSE" />

<CommandText>CREATE NONCLUSTERED INDEX nidx\_fname

ON Actors(ActorFirstName)</CommandText>

</TSQLCommand>

</EVENT\_INSTANCE>

1. Бъдеща работа

* Описване на възможните посоки на развитие на разработеното приложение.

Създадената база от данни може да претърпи много промени, корекции и обновления - например ако някой актьор си смени името, рейтингът на даден филм се покачи или спадне, или за да се предотврати неволна грешка. Освен това се очаква да се попълнят данни за филми, на които тепърва им предстои премиера на големия екран и все още са в процес на разработка. Същото важи за нови лица при актьорите, режисьорите и критиците. Таблицата Genres e недовършена, тъй като съществуват още много филмови жанрове, които могат да бъдат вмъкнати в базата. Могат да се създадат и нови таблици за наградите и номинациите за филми, актьори и режисьори, да се добави подробна информация за излъчването на филмите по кината, телевизията и онлайн стрийминг платформите.

В заключение MoviesDatabase представлява всеобхватна база от данни, която снабдява с информация в различни спектри за филмите и екипа, който участва в тях и може да бъде доразвита в много посоки.