**Задание върху бази от данни**

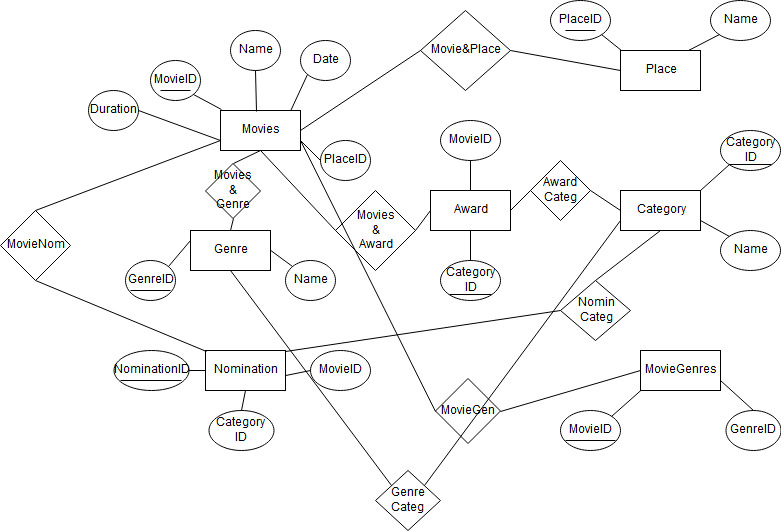
**Изработил: Доброслав**

**1. Описание на предметната област**

• определяне на областта на приложение, за която базата от данни може да бъде полезна; - Областта на приложение е във филмовата индустрия.

• определяне на функционалността на приложението за бази от данни; - Базата от данни съдържа филми и информация за тях. Всяка таблица е с различно предназначение, някои от тях са свързани помежду си.

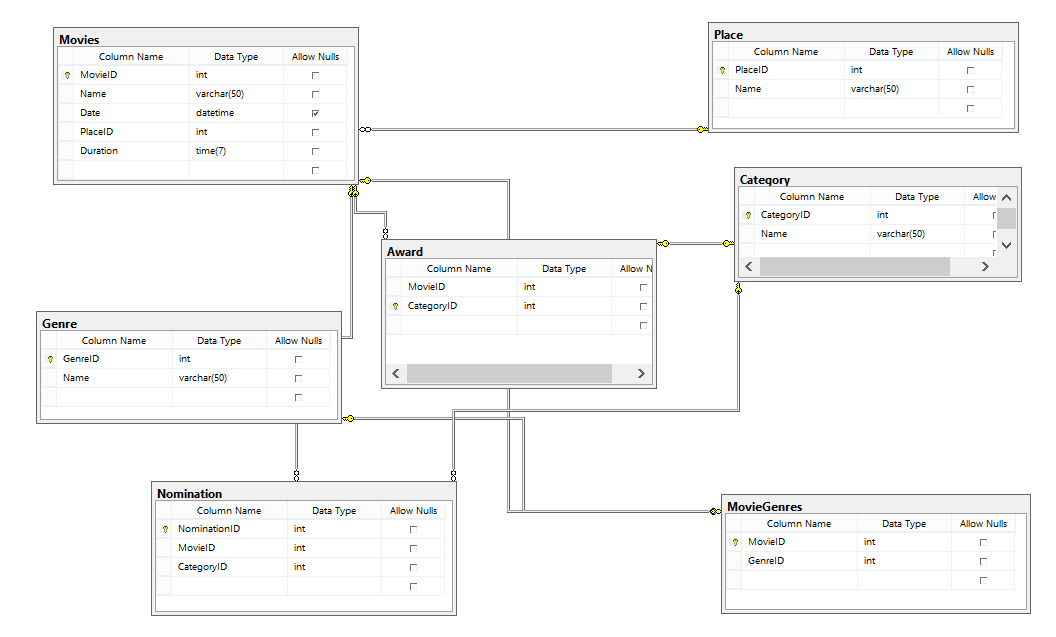
2. ER-модел на базата от данни



• описване на обектите, атрибутите, връзките, ограниченията, включени в ER-диаграмата; - В диаграмата са включени 7 таблици; Movies, съдържаща името на филмите, времетраенето и датата на премиерата; Genre, съдържаща жанровете на филмите; MovieGenres, съдържаща номера на филма и номера на жанра в който попада даден филм; Category, категории, в които се избират най-добрият актьор, актриса и тн. от всички филми; Award и Place, където са подредени филмите по печеливши позиции.

• определяне на схемата на всяка релация - име на релация, нейните атрибути и тип на данните за всеки атрибут;

* **Табл. Movies:** MovieID (тип: int), Name (тип: varchar(50)), Date (тип: datetime), PlaceID (тип: int), Duration (тип: time(7));
* **Табл. Place:** PlaceID (тип: int), Name (тип: varchar(50));
* **Табл. Genre:** GenreID (тип: int), Name (тип: varchar(50));
* **Табл. Award:** MovieID (тип: int), CategoryID (тип: int);
* **Табл. Category:** CategoryID (тип: int), Name (тип: varchar(50));
* **Табл. Nomination:** NominationID (тип: int), MovieID (тип: int), CategoryID (тип: int);
* **Табл. MovieGenres:** MovieID (тип: int), GenreID (тип: int).

**3. Релационен модел на базата от данни**

**4. Реализация на релационния модел на базата от данни**

**Създаване на Movies таблица:**

CREATE TABLE [dbo].[Movies](

[MovieID] [int] NOT NULL,

[Name] [varchar](50) NOT NULL,

[Date] [datetime] NULL,

[PlaceID] [int] NOT NULL,

[Duration] [time](7) NOT NULL,

CONSTRAINT [PK\_Movies] PRIMARY KEY CLUSTERED

([MovieID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS =

ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY],

CONSTRAINT [MovieNameUnique] UNIQUE NONCLUSTERED

([Name] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS =

ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

GO

**Създаване на Genre таблица:**

CREATE TABLE [dbo].[Genre](

[GenreID] [int] NOT NULL,

[Name] [varchar](50) NOT NULL,

CONSTRAINT [PK\_Genre] PRIMARY KEY CLUSTERED

([GenreID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS =

ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY],

CONSTRAINT [GenreNameUnique] UNIQUE NONCLUSTERED

([Name] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS =

ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

GO

**Създаване на MovieGenres таблица:**

CREATE TABLE [dbo].[MovieGenres](

[MovieID] [int] NOT NULL,

[GenreID] [int] NOT NULL,

CONSTRAINT [PK\_MovieGenres] PRIMARY KEY CLUSTERED

([MovieID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS =

ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY],

CONSTRAINT [IX\_MovieGenres] UNIQUE NONCLUSTERED

([MovieID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS =

ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

GO

*Check ограничения:*

ALTER TABLE [dbo].[MovieGenres] WITH CHECK ADD CONSTRAINT [FK\_MovieGenres\_Genre] FOREIGN

KEY([GenreID])

REFERENCES [dbo].[Genre] ([GenreID])

ON UPDATE CASCADE

GO

ALTER TABLE [dbo].[MovieGenres] CHECK CONSTRAINT [FK\_MovieGenres\_Genre]

GO

ALTER TABLE [dbo].[MovieGenres] WITH CHECK ADD CONSTRAINT [FK\_MovieGenres\_Movies] FOREIGN

KEY([MovieID])

REFERENCES [dbo].[Movies] ([MovieID])

ON UPDATE CASCADE

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[MovieGenres] CHECK CONSTRAINT [FK\_MovieGenres\_Movies]

GO

**Създаване на Category таблица:**

CREATE TABLE [dbo].[Category](

[CategoryID] [int] NOT NULL,

[Name] [varchar](50) NOT NULL,

CONSTRAINT [PK\_Category] PRIMARY KEY CLUSTERED

([CategoryID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS =

ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY],

CONSTRAINT [NameCategoryUnique] UNIQUE NONCLUSTERED

([Name] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS =

ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

GO

**Създаване на Award таблица:**

CREATE TABLE [dbo].[Award](

[MovieID] [int] NOT NULL,

[CategoryID] [int] NOT NULL,

CONSTRAINT [PK\_Award] PRIMARY KEY CLUSTERED

([CategoryID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS =

ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

GO

*Check ограничения:*

ALTER TABLE [dbo].[Award] WITH CHECK ADD CONSTRAINT [FK\_Award\_Category] FOREIGN KEY

([CategoryID])

REFERENCES [dbo].[Category] ([CategoryID])

GO

ALTER TABLE [dbo].[Award] CHECK CONSTRAINT [FK\_Award\_Category]

GO

ALTER TABLE [dbo].[Award] WITH CHECK ADD CONSTRAINT [FK\_Award\_Movies] FOREIGN KEY

([MovieID])

REFERENCES [dbo].[Movies] ([MovieID]) ALTER TABLE [dbo].[Award] CHECK CONSTRAINT [FK\_Award\_Movies]

GO GO

**Създаване на Nomination таблица:**

CREATE TABLE [dbo].[Nomination](

[NominationID] [int] NOT NULL,

[MovieID] [int] NOT NULL,

[CategoryID] [int] NOT NULL,

CONSTRAINT [PK\_Nomination] PRIMARY KEY CLUSTERED

([NominationID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS =

ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

GO

*Check ограничения:*

ALTER TABLE [dbo].[Nomination] WITH CHECK ADD CONSTRAINT [FK\_Nomination\_Category] FOREIGN

KEY([CategoryID])

REFERENCES [dbo].[Category] ([CategoryID])

GO

ALTER TABLE [dbo].[Nomination] CHECK CONSTRAINT [FK\_Nomination\_Category]

GO

ALTER TABLE [dbo].[Nomination] WITH CHECK ADD CONSTRAINT [FK\_Nomination\_Movies] FOREIGN

KEY([MovieID])

REFERENCES [dbo].[Movies] ([MovieID])

GO

ALTER TABLE [dbo].[Nomination] CHECK CONSTRAINT [FK\_Nomination\_Movies]

GO

**Създаване на Place таблица:**

CREATE TABLE [dbo].[Place](

[PlaceID] [int] NOT NULL,

[Name] [varchar](50) NOT NULL,

CONSTRAINT [PK\_Place] PRIMARY KEY CLUSTERED

([PlaceID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS =

ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY],

CONSTRAINT [PlaceNameUnique] UNIQUE NONCLUSTERED

([Name] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS =

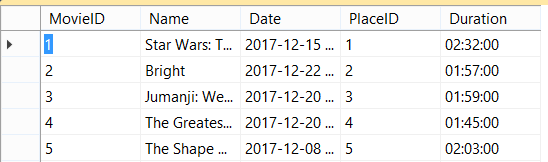
ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

GO

• попълване на таблиците с подходящи данни - изпълняване на примерни INSERT, DELETE и UPDATE команди за всяка създадена таблица;

**За таблица Movies**

INSERT:

**** INSERT INTO dbo.Movies

(MovieID, Name, Date, PlaceID, Duration )

VALUES

(1, 'Star Wars: The Last Jedi', '2017-12-15', '1', '02:32' ),

(2, 'Bright', '2017-12-22', '2', '01:57' ),

(3, 'Jumanji: Welcome to the Jungle', '2017-12-20', '3', '01:59' ),

(4, 'The Greatest Showman', '2017-12-20', '4', '01:45' ),

(5, 'The Shape of Water', '2017-12-08', '5', '02:03' );

DELETE:

DELETE FROM dbo.Movies

WHERE Name='Star Wars: The Last Jedi';

UPDATE:

UPDATE dbo.Movies SET Name = 'New Movie Name', Date= '2016-12-12' WHERE MovieID = 3;

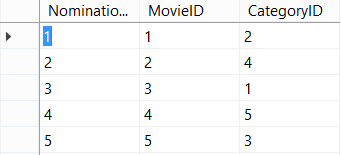
**За таблица Nomination**

INSERT:

INSERT INTO dbo.Nomination

(Nomination, MovieID, CategoryID)

VALUES

 (1, 1, 2),

(2, 2, 4),

(3, 3, 1),

(4, 4, 5),

(5, 5, 3);

DELETE:

DELETE FROM dbo.Nomination

WHERE NominationID='1';

UPDATE:

UPDATE dbo.Nomination

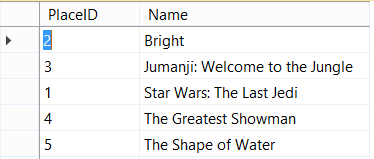
SET NominationID= '1', CategoryID= '1'

WHERE MovieID = 5;

**За таблица Place**

INSERT:

INSERT INTO dbo.Place

 (PlaceID, Name)

VALUES

(1, 'Star Wars: The Last Jedi'),

(2, 'Bright'),

(3, 'Jumanji: Welcome to the Jungle'),

(4, 'The Greatest Showman'),

(5, 'The Shape of Water');

DELETE:

DELETE FROM dbo.Place

WHERE Name='The Shape of Water';

UPDATE:

UPDATE dbo.Place

SET PlaceID= '6', Name= 'Jumanji: Bye, Jungle'

WHERE PlaceID = 3;

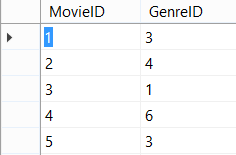
**За таблица MovieGenres**

INSERT:

INSERT INTO dbo.MovieGenres

(MovieID, GenreID)

VALUES

 (1, 3),

(2, 4),

(3, 1),

(4, 6),

(5, 3);

DELETE:

DELETE FROM dbo.MovieGenres

WHERE MovieID='1';

UPDATE:

UPDATE dbo.MovieGenres

SET MovieID= '1', GenreID= '1'

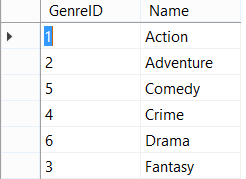
WHERE MovieID = 5;

**За таблица Genre**

INSERT:

INSERT INTO dbo.Genre

(GenreID, Name)

 VALUES

(1, 'Action'),

(2, 'Adventure'),

(3, 'Fantasy'),

(4, 'Crime'),

(5, 'Comedy'),

(6, 'Drama');

DELETE:

DELETE FROM dbo.Genre WHERE GenreID='3';

UPDATE:

UPDATE dbo.Genre

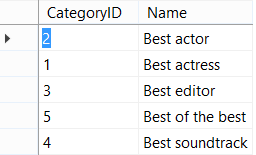
SET GenreID= '5', Name= 'Thriller'

WHERE MovieID = 6;

**За таблица Category**

INSERT:

INSERT INTO dbo.Category

 (CategoryID, Name)

VALUES

(1, 'Best actress'),

(2, 'Best actor'),

(3, 'Best editor'),

(4, 'Best soundtrack'),

(5, 'Best of the best');

DELETE:

DELETE FROM dbo.Category

WHERE Name='Best editor';

UPDATE:

UPDATE dbo.Category

SET CategoryID= '6', Name= 'The BEST'

WHERE PlaceID = 5;

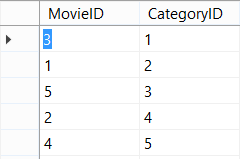
**За таблица Award**

INSERT:

INSERT INTO dbo.Award

(MovieID, CategoryID)

VALUES

 (3, 1),

(1, 2),

(5, 3),

(2, 4),

(4, 5);

DELETE:

DELETE FROM dbo.Award

WHERE CategoryID='3';

UPDATE:

UPDATE dbo.Award

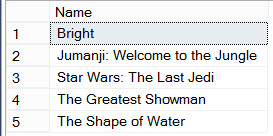
SET MovieID= '6', CategoryID= '6'

WHERE PlaceID = 1;

**5. Създаване на заявки**

• написване на поне пет различни заявки, като се използва SELECT конструкцията на SQL:

*- Заявка за подреждане на Name по азбучен ред в таблица Movies:*

SELECT Name

FROM dbo.Movies

ORDER BY Name ASC;

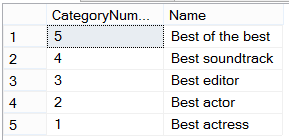
*- Заявка за извличане на данни за запис от таблица Movies:*

SELECT MovieID, Name, Date, PlaceID, Duration

FROM dbo.Movies

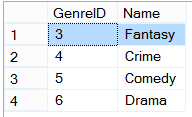
WHERE Name='Bright'

*- Заявка за извличане на данни от колона CategoryID в нова колона CategoryNumber и подреждането ѝ в низходящ ред:*

SELECT CategoryID AS CategoryNumber, Name

FROM dbo.Category

ORDER BY CategoryNumber DESC;

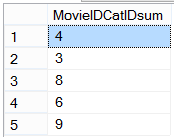
*- Заявка за извличане на данни от таблица, чиито ID на жанра е по-голямо или равно на 3:*

SELECT \* FROM Genre

WHERE GenreID >= 3

*- Заявка за конкатенация на MovieID и CategoryID от таблица Award:*

SELECT (MovieID + ' - ' + CategoryID) AS MovieIDCatIDsum

FROM dbo.Award

• заявки, включващи обобщаващи функции (като SUM, COUNT, AVG, MIN, MAX);

*- Заявка за преброяване на филмите в таблица Movies.*

*- Заявка за средна стойност в таблица Genre*

SELECT

 AVG(GenreID)AS 'Average GenreID'

FROM dbo.Genre

*- Заявка за извличане на филма с най-малко времетраене:*

SELECT MIN(Duration) as 'Minimum duration'

FROM dbo.Movies

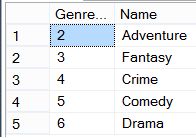
*- Заявка за извличане на филма с най-голямо времетраене:*

SELECT MAX(Duration) as 'Maximum duration'

FROM dbo.Movies

• заявки, включващи GROUP BY, HAVING;

*- Заявка с GROUP BY за извличане на данни от таблица Movies и групирането им по MovieID.*

SELECT GenreID, Name

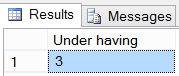
FROM dbo.Genre

WHERE GenreID > 1

GROUP BY GenreID, Name

*- Заявка с HAVING за намиране броя на жанровете с ID под средното за колона GenreID (таблица Genre).*

SELECT

 AVG(GenreID)AS 'Under having'

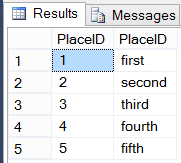
FROM dbo.Genre

HAVING AVG(GenreID) > 1

• заявки, които изискват използването на изрази - CASE израз

SELECT PlaceID,

CASE PlaceID

 WHEN 1 THEN 'first'

WHEN 2 THEN 'second'

WHEN 3 THEN 'third'

WHEN 4 THEN 'fourth'

WHEN 5 THEN 'fifth'

ELSE 'unknown'

END AS PlaceID FROM Movies

• примери за изрази за дата/час:

- DATEADD(day, 30, '1/1/2017')

- DATEDIFF(year, '31/12/2016', '1/1/2018')

- DATEPART(day, '30')

- DATENAME(month, '01/11/2017')

• заявки, включващи външни съединения и/или подзаявки;

SELECT MovieID, GenreID, Duration

FROM dbo.Movies

INNER JOIN dbo.Genre

ON dbo.Movies.MovieID = dbo.Genre.GenreID;

GO

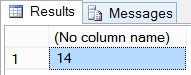
*- Заявка за прочитане на броя символи в даден стринг:*

DECLARE @testvar varchar(40)

SELECT

@testvar = 'Hello, DB. :-)'

SELECT LEN(@testvar);

Резултат: 

*- Заявка за извличане на X брой символи (в случая 10) от лявата страна на стринг:*

DECLARE @testvar varchar(40)

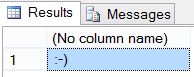
SELECT

@testvar = 'Hello, DB. :-)'

SELECT LEFT(@testvar, 10);

Резултат: 

*- Заявка за извличане на X брой символи (в случая 10) в обратен ред на стринг:*

Резултат: 

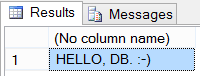
*- Заявка за големи букви на даден стринг:*

DECLARE @testvar varchar(40)

SELECT

@testvar = 'Hello, DB. :-)'

SELECT UPPER(@testvar)

Резултат: 

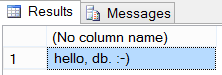
*- Заявка за малки букви на даден стринг:*

DECLARE @testvar varchar(40)

SELECT

@testvar = 'Hello, DB. :-)'

SELECT LOWER(@testvar)

Резултат: 

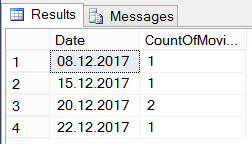
*- Конвертиране и групиране на данни:*

SELECT CONVERT(char(10), Date, 104) AS Date,

COUNT(\*) AS CountOfMovies

FROM Movies

GROUP BY CONVERT(char(10), Date, 104)

Резултат: 

**6. Създаване на необходимите:**

- Изглед:

CREATE VIEW NewDBView

AS

SELECT Name, Date, PlaceID

FROM Movies

Изтриване на изглед:

DROP VIEW dbo.NewView

* Потребителски дефинирана функция:

CREATE FUNCTION MoviesFunction3

(@Duration varchar(15))

RETURNS varchar(15)

AS

BEGIN

Declare @UpdDuration varchar(15)

IF @Duration > 1

SET @UpdDuration = @Duration

ELSE SET @UpdDuration = '(' + LEFT(@UpdDuration, 3) + ') ' +

SUBSTRING(@UpdDuration, 4, 3) +

'-' + RIGHT (@Duration, 4)

RETURN @UpdDuration

END

* Тригер:

CREATE TRIGGER DisabledDelete

ON Movies

INSTEAD OF DELETE

AS

IF @@ROWCOUNT > 0

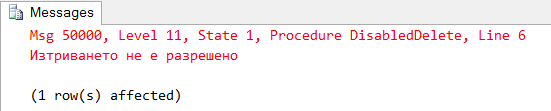
RETURN

RAISERROR ('Изтриването не е разрешено', 11, 1)

Тестване на тригера:

DELETE FROM Movies

WHERE MovieID = 3

Резултат: 

7. Бъдеща работа

• описване на възможните посоки на бъдещо развитие на разработеното приложение. - Поддръжката и разрастването на базата от данни с актуална информация може да я направи богата филмова библиотека.