**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине «Разработка приложений баз данных для информационных систем»

на тему «Разработка серверной части информационной системы

в СУБД *MS SQL Server*»

Вариант № 13

Выполнил: студент гр. ИТП-31

Куцый Д.А.

Принял: доцент

Асенчик О.Д.

Гомель 2022

**Цель работы:** разработать серверную часть клиент-серверной информационной системы, основанной на базе данных в заданной предметной области средствами СУБД *MS SQL Server*.

**Задание:**

1. Разработать логическую модель реляционной базы данных, моделирующую предметную область согласно своему варианту задания. Структура БД должна быть нормализована – таблицы должны удовлетворять требованиям третьей нормальной формы.
2. Создать базу данных и таблицы в СУБД *MS SQL Server* и заполнить их тестовым набором данных, для этого написать *Transact SQL* скрипт:

2.1. Создания базы данных и ее таблиц.

При создании таблиц должны быть назначены первичные и внешние ключи и установлены необходимые ограничения целостности данных. Наименования таблиц и полей давать в соответствии с соглашением об именовании этих объектов.

2.2. Заполнения не менее чем трех таблиц БД данными (см. пример внутри этого электронного курса). При выборе таблиц для заполнения тестовыми наборами руководствоваться следующим:

* выбранные таблицы должны содержать основную информацию, касающиеся предметной области приложения;
* таблицы должны быть связаны непосредственно;
* следует воздерживаться от выбора таблиц, характеризующих кадровую подсистему приложения (данные о сотрудниках и их должностях);
* не менее, чем одна таблица должна находиться в схеме базы данных на стороне отношения «многие».
* При этом заполнение осуществлять в следующем порядке: сначала генерируются данные для таблиц на стороне отношения «один» (таблицы- «справочники»), потом – в таблицы на стороне отношения «многие» («оперативные таблицы»).
* БД заполнить записями в количестве, необходимом для отладки и демонстрации возможностей приложения. Таблицы на стороне отношения один должны содержать не менее 500 записей, таблице на стороне отношения многие должны содержать не менее 20000 записей.

1. Создать с использованием средств *Transact SQL* представления, позволяющие отображать данные в удобном для пользователя виде, и заменяющие часто используемые запросы на выборку из двух и более связанных таблиц.
2. Написать не менее трех хранимых процедур с параметрами для вставки и (или) обновление данных в таблицы базы данных.

Условие задания представлено на рисунке 1.

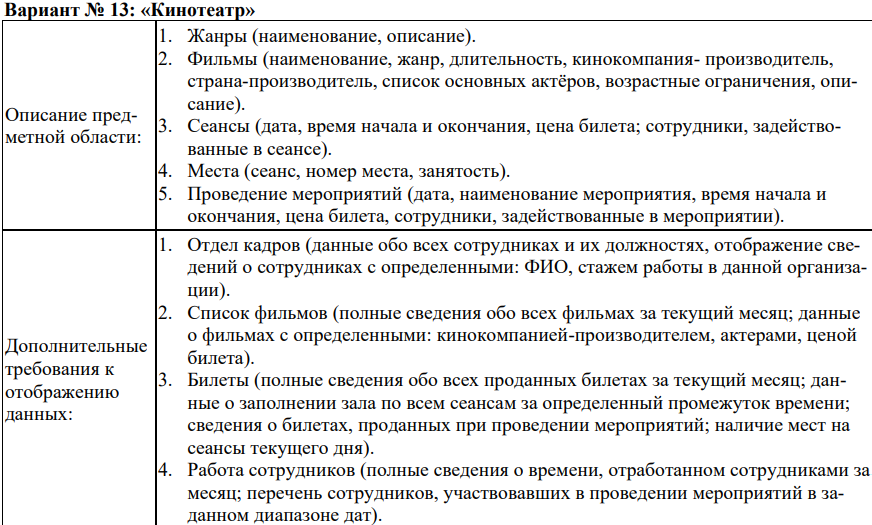


Рисунок 1 – Условие задания

**Ход работы**

**Этапы выполнения работы:**

На рисунке 2 представлена схема созданной базы данных.

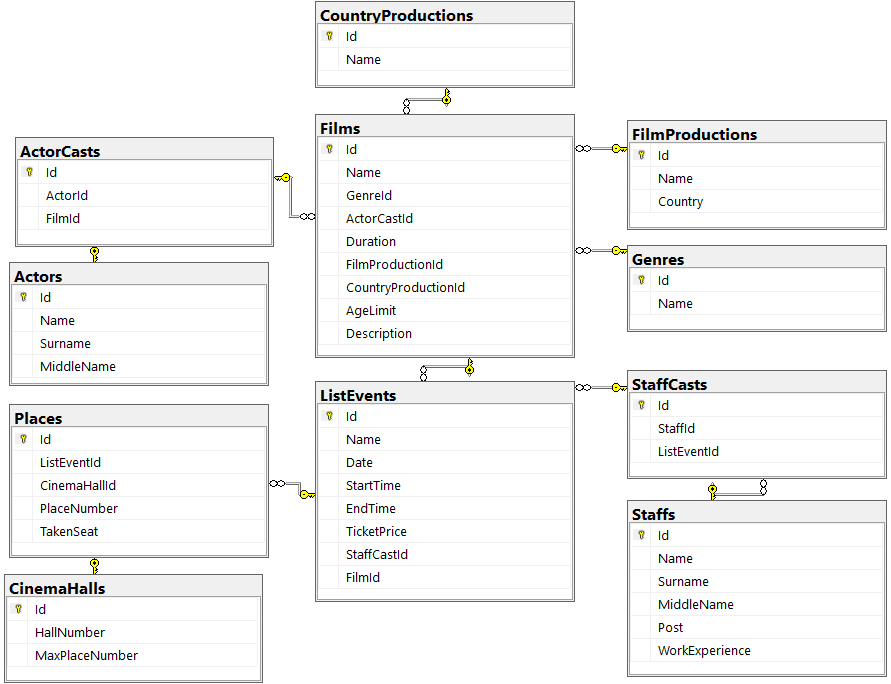


Рисунок 2 – Схема базы данных

На рисунке 3 представлен результат заполнения таблицы *Actors* случайными значениями.

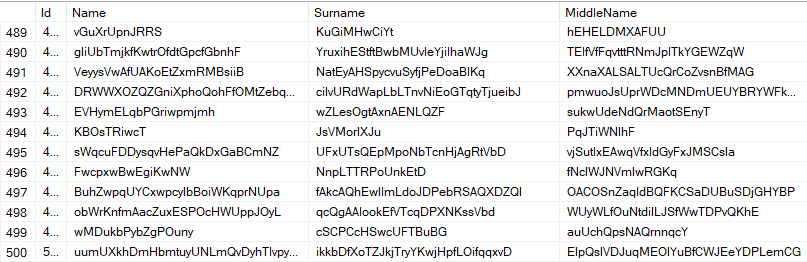


Рисунок 3 – Результат заполнения таблицы *Actors*

На рисунке 4 представлен результат заполнения таблицы *ActorCasts* случайными значениями.

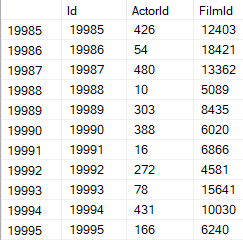


Рисунок 4 – Результат заполнения таблицы *ActorCasts*

На рисунке 5 представлен результат заполнения таблицы *FilmProductions* случайными значениями.



Рисунок 5 – Результат заполнения таблицы *FilmProductions*

На рисунке 6 представлен результат заполнения таблицы *CountryProductions* случайными значениями.

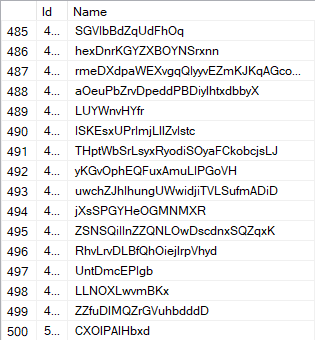


Рисунок 6 – Результат заполнения таблицы *CountryProductions*

На рисунке 7 представлен результат заполнения таблицы *Genres* случайными значениями.

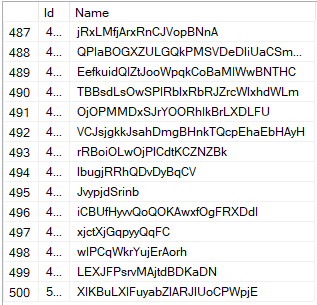


Рисунок 7 – Результат заполнения таблицы *Genres*

На рисунке 8 представлен результат заполнения таблицы *Films* случайными значениями.

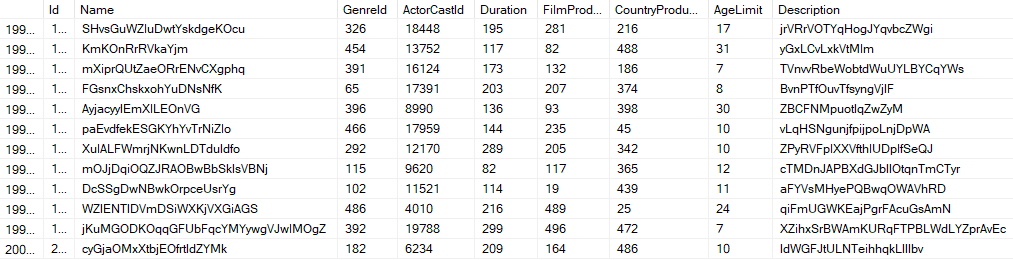


Рисунок 8 – Результат заполнения таблицы *Films*

В приложении А представлен:

1. код, генерирующий случайные значения в таблицы;
2. код, создающий хранимых процедуры;
3. код, создающий представления.

Ссылка на *github*: [kutsydanil/MSSQL\_Lab- (github.com)](https://github.com/kutsydanil/MSSQL_Lab-).

**Вывод:** в ходе выполнения лабораторной работы была разработана серверная часть клиент-серверной информационной системы, основанной на базе данных кинотеатра средствами СУБД *MS SQL Server*.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Листинг программы**

**Листинг хранимых процедур:**

/\*Procedure\*/

CREATE PROC ActorCreate

@ActorName NVARCHAR(60), @Surname NVARCHAR(60), @MiddleName NVARCHAR(60)

AS

INSERT INTO Actors(Name , Surname , MiddleName) VALUES (@ActorName, @Surname, @MiddleName)

GO

CREATE PROC ActorUpdate

@ActorId INT, @ActorName NVARCHAR(60), @Surname NVARCHAR(60), @MiddleName NVARCHAR(60)

AS

UPDATE Actors SET Name = @ActorName, Surname = @Surname , MiddleName = @MiddleName

WHERE Actors.Id = @ActorId

GO

CREATE PROC GenreCreate

@Name NVARCHAR(100)

AS

INSERT INTO Genres(Name) VALUES (@Name)

GO

CREATE PROC GenreUpdate

@Name NVARCHAR(100) , @GenreId INT

AS

UPDATE Genres SET Genres.Name = @Name WHERE Genres.Id = @GenreId

GO

CREATE PROC FilmProductionCreate

@ProductionName NVARCHAR(100), @ProductionCountry NVARCHAR(150)

AS

INSERT INTO FilmProductions(Name, Country) VALUES (@ProductionName , @ProductionCountry)

GO

CREATE PROC FilmProductionCreateUpdate

@ProductionId INT, @ProductionName NVARCHAR(100), @ProductionCountry NVARCHAR(150)

AS

UPDATE FilmProductions SET Name = @ProductionName , Country = @ProductionCountry

WHERE FilmProductions.Id = @ProductionId

GO

**Листинг представлений:**

/\*VIEW\*/

GO

CREATE VIEW VwActorActorCasts

AS

SELECT ActorCasts.Id, Actors.Name, Actors.MiddleName, Actors.Surname FROM Actors

INNER JOIN ActorCasts ON ActorCasts.ActorId = Actors.Id

Go

CREATE VIEW VwActorActorCastsFilms

AS

SELECT Actors.Name, Actors.MiddleName, Actors.Surname, Films.Name as FilmName , ActorCasts.Id AS CastsId

FROM ActorCasts INNER JOIN Actors ON Actors.Id = ActorCasts.ActorId

INNER JOIN Films ON Films.ActorCastId = ActorCasts.FilmId

GO

CREATE VIEW VwGenreFilms

AS

SELECT Films.Name AS FilmName, Genres.Name FROM Films INNER JOIN Genres ON Genres.Id = Films.GenreId

GO

CREATE VIEW VwGenreProductionFilms

AS

SELECT Films.Name AS FilmName, Films.Duration , Genres.Name AS GenreName, Films.AgeLimit, FilmProductions.Name AS ProductionName,

CountryProductions.Name AS ContryProduction , Films.Description

FROM Films INNER JOIN Genres ON Genres.Id = Films.GenreId

INNER JOIN FilmProductions ON FilmProductions.Id = Films.FilmProductionId

INNER JOIN CountryProductions ON CountryProductions.Id = Films.CountryProductionId

GO

CREATE VIEW VwListEventFilms

AS

SELECT ListEvents.Date, ListEvents.TicketPrice, Films.Name AS FilmName, Films.Duration , Genres.Name AS GenreName, Films.AgeLimit, FilmProductions.Name AS ProductionName,

CountryProductions.Name AS ContryProduction , Films.Description

FROM Films INNER JOIN Genres ON Genres.Id = Films.GenreId

INNER JOIN FilmProductions ON FilmProductions.Id = Films.FilmProductionId

INNER JOIN CountryProductions ON CountryProductions.Id = Films.CountryProductionId

INNER JOIN ListEvents ON ListEvents.FilmId = Films.Id

GO

CREATE VIEW VwEventListCinemaHallPlaces

AS

SELECT ListEvents.Name AS EventName, ListEvents.Date , CinemaHalls.HallNumber, Places.PlaceNumber , Places.TakenSeat

FROM Places INNER JOIN CinemaHalls ON CinemaHalls.Id = Places.CinemaHallId

INNER JOIN ListEvents ON ListEvents.Id = Places.ListEventId

GO

**Листинг генератора значений:**

use Cinema

GO

DECLARE @CharList CHAR(52) = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz',

@Position INT,

@index INT,

@OneLink INT,

@ManyLink INT,

@NameLimit INT,

@MaxLimit INT,

@MinLimit INT,

@RowCount INT,

@AgeLimit INT,

@FilmDuration INT,

@FilmProductionName NVARCHAR(70),

@CountryFilmProductionName NVARCHAR(70),

@CountryName NVARCHAR(70),

@GenreName NVARCHAR(50),

@Name NVARCHAR(50),

@SurName NVARCHAR(50),

@MiddleName NVARCHAR(50),

@FilmName NVARCHAR(50),

@FilmDescription NVARCHAR(400),

@GenreId INT,

@ActorCastId INT,

@ActorId INT,

@FilmProductionId INT,

@CountryProductionId INT,

@FilmId INT

SET @OneLink = 500

SET @ManyLink = 20000

SET @index = 1

SET @MaxLimit = 35

SET @MinLimit = 10

SET @RowCount = 1

/\*Генерация значений в таблицу - Жанры/Страна-производитель/Компания-производитель/Актеры\*/

SET @RowCount = 1

WHILE @RowCount <= @OneLink

BEGIN

SET @index = 1

/\*Жанры\*/

SET @GenreName = ''

SET @NameLimit = @MinLimit + RAND()\*(@MaxLimit - @MinLimit)

WHILE @index <= @NameLimit

BEGIN

SET @Position = RAND()\*52

SET @GenreName = @GenreName + SUBSTRING(@CharList , @Position , 1)

SET @index = @index + 1

END;

/\*Страна-производитель\*/

SET @CountryName = ''

SET @index = 1

SET @NameLimit = @MinLimit + RAND()\*(@MaxLimit - @MinLimit)

WHILE @index <= @NameLimit

BEGIN

SET @Position = RAND()\*52

SET @CountryName = @CountryName + SUBSTRING(@CharList , @Position , 1)

SET @index = @index + 1

END

/\*Компания-производитель\*/

SET @CountryFilmProductionName = ''

SET @FilmName = ''

SET @index = 1

SET @NameLimit = @MinLimit + RAND()\*(@MaxLimit - @MinLimit)

WHILE @index <= @NameLimit

BEGIN

SET @Position = RAND()\*52

SET @CountryFilmProductionName = @CountryName + SUBSTRING(@CharList , @Position , 1)

SET @Position = RAND()\*52

SET @FilmName = @FilmName + SUBSTRING(@CharList , @Position , 1)

SET @index = @index + 1

END

/\*Актеры\*/

SET @index = 1

SET @Name = ''

SET @MiddleName = ''

SET @SurName = ''

SET @NameLimit = @MinLimit + RAND()\*(@MaxLimit - @MinLimit)

WHILE @index <= @NameLimit

BEGIN

SET @Position = RAND()\*52

SET @Name = @Name + SUBSTRING(@CharList , @Position , 1)

SET @Position = RAND()\*52

SET @MiddleName = @MiddleName + SUBSTRING(@CharList , @Position , 1)

SET @Position = RAND()\*52

SET @SurName = @SurName + SUBSTRING(@CharList , @Position , 1)

SET @index = @index + 1

END

INSERT INTO Genres(Name) VALUES (@GenreName)

INSERT INTO CountryProductions(Name) VALUES(@CountryName)

INSERT INTO FilmProductions(Name, Country) VALUES(@FilmName ,@CountryName)

INSERT INTO Actors(Name, MiddleName, Surname) VALUES(@Name , @MiddleName , @SurName)

SET @RowCount +=1

END

/\*Генерация значений в таблицу - Актерские группы/Фильмы\*/

SET @RowCount = 1

WHILE @RowCount <= @ManyLink

BEGIN

/\*Актерские группы\*/

SET @ActorId = 1+RAND()\*(@OneLink-1)

SET @FilmId = 1+RAND()\*(@ManyLink-1)

INSERT INTO ActorCasts(ActorId, FilmId) VALUES(@ActorId , @FilmId)

SET @RowCount +=1

END

SET @RowCount = 1

WHILE @RowCount <= @ManyLink

BEGIN

SET @index = 1

/\*Фильмы\*/

SET @GenreId = 1+RAND()\*(@OneLink-1)

SET @ActorCastId = 1+RAND()\*(@ManyLink-1)

SET @FilmProductionId = 1+RAND()\*(@OneLink-1)

SET @CountryProductionId = 1+RAND()\*(@OneLink-1)

SET @AgeLimit = 6+RAND()\*(32-6)

SET @FilmDuration = 60+RAND()\*(300 - 60)

SET @NameLimit = @MinLimit + RAND()\*(@MaxLimit - @MinLimit)

SET @FilmDescription = ''

SET @FilmName = ''

WHILE @index <= @NameLimit

BEGIN

SET @Position = RAND()\*52

SET @FilmDescription = @FilmDescription + SUBSTRING(@CharList , @Position , 1)

SET @Position = RAND()\*52

SET @FilmName = @FilmName + SUBSTRING(@CharList , @Position , 1)

SET @index = @index + 1

END

INSERT INTO Films(Name, GenreId, ActorCastId, Duration, FilmProductionId, CountryProductionId, AgeLimit, Description)

VALUES (@FilmName, @GenreId, @ActorCastId, @FilmDuration, @FilmProductionId , @CountryProductionId, @AgeLimit, @FilmDescription)

SET @RowCount +=1

END