Оглавление

Описание БД «Фильмотека»

Данная база данных будет содержать в себе ссылки на фильмы, расположенные на каком-либо подключенном носителе, а также подробную информацию о каждом из них:

- Информация о фильме:
  1. Название фильма на английском языке
  2. Локализованное название фильма
  3. Год выпуска
  4. Страна-производитель (выбор из списка)
  5. Слоган
  6. Режиссер (один или несколько людей)
  7. Сценарист (один или несколько людей)
  8. Продюсер (один или несколько людей)
  9. Оператор (один или несколько людей)
  10. Композитор (один или несколько людей)
  11. Художник (один или несколько людей)
  12. Монтажер (один или несколько людей)
  13. Актеры (один или несколько людей)
  14. Жанр (выбор из списка)
  15. Бюджет фильма с указанием валюты (выбор валюты из списка)
  16. Возрастное ограничение (выбор из списка)
  17. Рейтинг MPAA (выбор из списка)
  18. Длительность фильма
  19. Краткая аннотация к фильму
  20. Количество зрителей
- Информация о файле:
  1. Размер файла
  2. Формат видеофайла (выбор из списка)
  3. Используемый кодек (выбор из списка)
  4. Разрешение видеокадра
  5. Соотношение сторон (выбор из списка)
  6. Ссылка на физический файл
- Информация о человеке:
  1. ФИО
  2. Какую должность в каком фильме занимает
  3. Дата рождения
  4. Страна

Данные, помеченные как «выбор из списка», могут быть инициализированы только набором заранее внесенных в базу значений, так как эти набор этих значений меняется относительно редко.

Данные, помеченные как «один или несколько людей», могут иметь достаточно большой разброс значений, который невозможно определить заранее, и, всвязи с этим, для исключения дублирования данных при различном вводе, будет выполняться автозаполнение.

Использоваться база данных будет использоваться для поиска фильма по одному или нескольким критериям, как по точному их значению, так и по диапазону значений, а также для получения после поиска доступа к фильму на носителе в том или ином виде.

Стоит отметить, что в базу данных каждому фильму может соответствовать только 0 или 1 физический файл.

Также база данных будет позволять получать информацию о людях, которые принимали ту или иную роль в его создании, а также о физических данных файла, соответствующего определенному фильму.

Все данные будут вноситься в базу извне (в том числе и информация о файле).

Схема таблиц

Схема таблиц и связей между ними представлена в конце документа.

Скрипт создания таблиц

```
use "film collection"
create table "countries"
(
      "country id"            int                  primary key identity,
      "name"                  varchar(80)          not null
)
create table "people"
(
      "people id"             int                  primary key identity,
      "first name"            varchar(80)          not null,
      "last name"             varchar(80)          not null,
      "birth date"            date                 null,
      "country id"            int                  foreign key references "countries" ("country id") null
)
create table "positions"
(
      "position id"           int                  primary key identity,
      "name"                  varchar(80)          unique not null,
      "description"           varchar(400)         null
)
create table "genres"
(
      "genre id"              int                  primary key identity,
      "name"                  varchar(80)          unique not null,
      "description"           varchar(400)         null
)
create table "file formats"
(
      "format id"             int                  primary key identity,
      "name"                  varchar(80)          unique not null,
      "description"           varchar(400)         null
)
create table "currencies"
(
      "currency id"           int                  primary key identity,
      "name"                  varchar(80)          unique not null,
      "symbol"                nchar(5)             null
)
```

```
create table "age restrictions"
(
      "restriction id"        int                    primary key identity,
      "value"                 varchar(80)            unique not null,
      "description"           varchar(400)           null
)
create table "mpaa rating"
(
      "rating id"             int                    primary key identity,
      "name"                  varchar(5)             unique not null,
      "description"           varchar(400)           null
)
create table "films info"
(
      "film id"               int                    primary key,
      "english name"          varchar(80)            not null,
      "localized name"        nvarchar(80)           not null,
      "year"                  date                   null,
      "slogan"                varchar(400)           null,
      "duration"              time                   not null,
      "number of viewers"     bigint                 null,
      "annotation"            varchar(8000)          null,
      "budget value"          bigint                 null,
      "currency id"           int                    foreign key references "currencies" ("currency id") null,
      "age restriction id"    int                    foreign key references "age restrictions" ("restriction id") not null,
      "mpaa rating id"        int                    foreign key references "mpaa rating" ("rating id") null
)
create table "films genre"
(
      "film id"               int                    foreign key references "films info" ("film id") not null,
      "genre id"              int                    foreign key references "genres" ("genre id") not null,
      primary key("film id", "genre id")
)
create table "people in film"
(
      "film id"               int                    foreign key references "films info" ("film id") not null,
      "people id"             int                    foreign key references "people" ("people id") not null,
      "position id"           int                    foreign key references "positions" ("position id") not null,
      primary key("film id", "people id", "position id")
)
create table "codecs"
(
      "codec id"              int                    primary key identity,
      "name"                  varchar(80)            unique not null,
      "description"           varchar(400)           null
)
```

```
create table "aspect ratios"
(
      "ratio id"              int                    primary key identity,
      "name"                  varchar(80)            unique not null
)
create table "files info"
(
      "file id"                    int              primary key,
      "film id"                    int              foreign key references "films info" ("film id") not null,
      "link"                       varchar(800)     not null unique,
      "file size"                  bigint           not null,
      "frame resolution width"     int              not null,
      "frame resolution height"    int              not null,
      "format id"                  int              foreign key references "file formats" ("format id") not null,
      "codec id"                   int              foreign key references "codecs" ("codec id") not null,
      "ratio id"                   int              foreign key references "aspect ratios" ("ratio id") not null
)
```

Скрипт базового наполнения таблиц

```
use "film collection"

insert into "file formats"
values      ('mkv', 'matroshka file format'),
            ('mp4', 'MPEG-4 part 14'),
            ('avi', 'Audio Video Interleave');

insert into "codecs" ("name")
values      ('DivX Low Motion'),
            ('DivX Fast Motion'),
            ('MPEG-1'),
            ('MPEG-2'),
            ('MPEG-4'),
            ('Motion JPEG'),
            ('xvid'),
            ('x264'),
            ('nuv'),
            ('raw'),
            ('copy'),
            ('frameno'),
            ('avc');

insert into "aspect ratios"
values      ('4:3'),
            ('1,375:1'),
            ('1,43:1'),
            ('3:2'),
            ('14:9'),
            ('16:10'),
            ('1,66:1'),
            ('1,85:1'),
            ('16:9'),
            ('2,2:1'),
            ('2,35:1'),
            ('2,39:1'),
            ('2,4:1'),
            ('2,55:1'),
            ('2,6:1'),
            ('2,75:1');

insert into "currencies"
values      ('dollar', '$'),
            ('euro', '^'),
            ('rouble', null);

insert into "genres" ("name")
values      ('Action'),
            ('Adventure'),
            ('Comedy'),
            ('Crime'),
            ('Drama'),
            ('Epic'),
            ('Horror'),
            ('Sci-fi'),
            ('War'),
            ('Western');

insert into "age restrictions" ("value")
values       ('0+'),
            ('6+'),
            ('12+'),
            ('16+'),
            ('18+');
```

```sql
insert into "mpaa rating"
values      ('G', 'General Audiences'),
            ('PG', 'Parental Guidance Suggested'),
            ('PG-13', 'Parents Strongly Cautioned'),
            ('R', 'Restricted'),
            ('NC-17', 'No One 17 & Under Admitted');

insert into "countries"
values      ('Russia'),
            ('USSR'),
            ('USA'),
            ('RSFSR'),
            ('Austria'),
            ('Ukraine'),
            ('New Zealand'),
            ('Great Britan');

insert into "people"
values      ('Леонид', 'Гайдай', '30.01.1923', 2),
            ('Александр', 'Демьяненко', '30.05.1937', 2),
            ('Наталья', 'Селезнёва', '19.06.1945', 2),
            ('Алексей', 'Смирнов', '28.02.1920', 4),
            ('Юрий', 'Никулин', '18.12.1921', 1),
            ('Евгений', 'Моргунов', '27.04.1927', 2),
            ('Георгий', 'Вицин', '18.04.1917', 1),
            ('Михаил', 'Пуговкин', '13.07.1923', 2),
            ('Виктор', 'Павлов', '5.10.1940', 2),
            ('Владимир', 'Басов', '28.07.1923', 2),
            ('Валентина', 'Янковская', null, null),
            ('Артур', 'Бергер', '27.05.1892', 5),
            ('Константин', 'Бровин', null, null),
            ('Морис', 'Слободской', '30.11.1913', 1),
            ('Петр', 'Феллер', null, null),
            ('Александр', 'Зацепин', '10.03.1926', 2),
            ('Яков', 'Костюковский', '23.08.1921', 6),
            ('Питер', 'Джексон', '31.10.1961', 7),
            ('Фрэнсис', 'Уолш', '10.01.1959', 7),
            ('Элайджа', 'Вуд', '28.01.1981', 3),
            ('Иэн', 'МакКеллен', '25.05.1939', 8),
            ('Вигго', 'Мортенсен', '20.10.1958', 3);

insert into "positions" ("name")
values      ('director'),
            ('stage manager'),
            ('screenwriter'),
            ('producer'),
            ('operator'),
            ('composer'),
            ('painter'),
            ('editior'),
            ('actor');
```

Скрипт заполнения информации о фильмах и файлах

```
use "film collection"

insert into "films info"
values      (1, 'Operacia Y', 'Операция «Ы» и другие приключения Шурика', '1965', null,
'01:35', 69600000, null, null, 3, 1, null),
            (2,   'The Lord of the Rings: The Fellowship of the Ring', 'Властелин колец:
Братство кольца', '2001', 'Power can be held in the smallest of things', '02:58',
100000000, null, 93000000, 1, 4, 3),
            (3, 'The Lord of the Rings: The Two Towers', 'Властелин колец: Две крепости',
'2002', 'Приключение продолжается', '02:59', 150000000, null, 94000000, 1, 3, 3),
            (4,   'The Lord of the Rings: The Return of the King', 'Властелин колец:
Возвращение Короля', '2003', 'There can be no triumph without loss. No victory without
suffering. No freedom without sacrifice', '03:21', 1650000000, null, 94000000, 1, 3, 3);

insert into "files info"
values      (1, 1, '/films/Operacia Y (1965).avi', 1564293120, 704, 528, 3, 7, 1),
            (2, 2, '/films/The Lord of the Rings: The Fellowship of the Ring (2001).mkv',
14416671502, 1280, 534, 1, 8, 12),
            (3, 3, '/films/The Lord of the Rings: The Two Towers (2002).mkv',
14903363544, 1280, 536, 1, 8, 12),
            (4, 4, '/films/The Lord of the Rings: The Return of the King (2003).mkv',
17658135843, 1280, 532, 1, 8, 12);

insert into "people in film"
values      (1, 1, 2),
            (1, 2, 9),
            (1, 3, 9),
            (1, 4, 9),
            (1, 5, 9),
            (1, 6, 9),
            (1, 7, 9),
            (1, 8, 9),
            (1, 9, 9),
            (1, 10, 9),
            (1, 15, 4),
            (1, 1, 3),
            (1, 14, 3),
            (1, 13, 5),
            (1, 16, 6),
            (1, 12, 7),
            (1, 11, 8),
            (1, 17, 3),
            (2, 18, 2),
            (2, 19, 3),
            (2, 20, 9),
            (2, 21, 9),
            (2, 22, 9),
            (3, 18, 2),
            (3, 19, 3),
            (3, 20, 9),
            (3, 21, 9),
            (3, 22, 9),
            (4, 18, 2),
            (4, 19, 3),
            (4, 20, 9),
            (4, 21, 9),
            (4, 22, 9);

insert into "films genre"
values      (1, 2),
            (1, 3),
            (1, 4),
            (2, 2),
            (2, 9),
            (3, 2),
            (3, 9),
            (4, 2),
            (4, 9);
```

Скрипты вставки/удаления/обновления

```
use "film collection"
go
insert into "people"
values  ('Вигго', 'Мортенсен', '20.10.1958', 3);

insert into "people"
values      ('Вигго', 'Мортенсен', '20.10.1958', 3);

insert into "people"
values      ('Вигго', 'Мортенсен', '20.10.1958', 4);

insert into "people"
values  ('Вигго', 'Мортенсен', '20.10.1958', 4);

insert into "people"
values  ('Вигго', 'Мортенсен', '20.10.1958', 5);

update "people"
set "country id" = 3
where "first name" = 'Вигго' and "last name" = 'Мортенсен' and "birth date" = '20.10.1958' and "country id" = 4

delete from "people"
where  "first  name"  =  'Вигго'  and  "last  name"  =  'Мортенсен'  and  "birth  date"  =  '20.10.1958'  and  "country  id"  =  5
```

Скрипт создания хранимых процедур

```
use "film collection"
go
```

1. Позволяет получить список всех фильмов, в которых принимал участие указанный человек

```
create procedure "films by people"
      @firstname varchar(80),
      @lastname  varchar(80)
as
      select "films info"."english name", year("films info"."year") as "year"
      from "films info"
      where "film id" in
            (select "people in film"."film id"
            from "people in film"
            where "people in film"."people id" in
                  (select "people"."people id"
                  from "people"
                  where "people"."first name" = @firstname and "people"."last name" = @lastname
                  )
            )
go
```

2. Позволяет получить диапазон дат выпуска фильмов, хранящихся в базе

```
create procedure "year range"
as
      select min(year("films info"."year")) as "min year", max(year("films info"."year")) as "max year"
      from "films info"
go
```

3. Позволяет получить размер фильма по ссылке на него

```
create procedure "get file size by link"
      @link   varchar(800),
      @size     bigint         output
as
      select @size = "files info"."file size"
      from "files info"
            where @link = "files info"."link"
```

Скрипты выполнения процедур

```
use "film collection"

exec "films by people" @firstname = 'Элайджа', @lastname = 'Вуд';
go

exec "year range"
go

declare @filesize int;
declare @filelink varchar(800);
set @filesize = 0;
set @filelink = '/films/Operacia Y (1965).avi';
exec "get file size by link" @link = @filelink, @size = @filesize output
      select @filelink as "link", @filesize as "size"
```

Скрипты создания триггеров

```
use "film collection"
go
```

1) Удаление дубликатор при добавлении/обновлении. Внутри используется курсор

```
create trigger "exclude duplication on insert/update"
on "people"
for insert, update
as
	declare @firstname	varchar(80)
	declare @lastname varchar(80)
	declare @birthdate	date
	declare @countryid	int
	declare @peopleid	int
	declare @count int

	declare insertedData cursor local forward_only static
		for select "people id", "first name", "last name", "birth date", "country id" from inserted;

	open insertedData

	fetch next from insertedData into @peopleid, @firstname, @lastname, @birthdate, @countryid

	while @@fetch_status = 0
	begin
		select @count = count(*) from "people"
		where "first name" Collate Cyrillic_General_CS_AS = @firstname and
				"last name" Collate Cyrillic_General_CS_AS = @lastname and
				"birth date" = @birthdate and
				"country id" = @countryid

		if (@count = 2)
		begin
			delete from "people"
			where "people id" = @peopleid
		end

		fetch next from insertedData into @peopleid, @firstname, @lastname, @birthdate, @countryid
	end

	close insertedData
	deallocate insertedData
	go
```

2) Приводит ФИО к нормальному виду про добавлении. Внутри используется курсор

```
create trigger "format names on create/update"
on "people"
for insert, update
as
	declare @id int
	declare @firstname varchar(80)
	declare @lastname  varchar(80)
	declare @length   int
	declare @newfirstname varchar(80)
	declare @newlastname  varchar(80)

	declare insertedData cursor local forward_only static
		for select "people id", "first name", "last name" from inserted;

	open insertedData

	fetch next from insertedData into @id, @firstname, @lastname

	while @@fetch_status = 0
	begin
		--does not deleted by another trigger
		if exists
		(
			select count(*)
			from "people"
			where "people id" = @id
		)
		begin
			--if bad names, delete
			if ((len(@firstname) = 0) or (len(@lastname) = 0))
			begin
				delete from "people"
				where "people id" = @id
			end
			else
			begin
				set @length = len(@firstname)
				set @newfirstname = upper(substring(@firstname, 1, 1)) + lower(substring(@firstname, 2, @length - 1))
				set @length = len(@lastname)
				set @newlastname = upper(substring(@lastname, 1, 1)) + lower(substring(@lastname, 2, @length - 1))

				--if new names equals old, do nothing
				if ((@newfirstname Collate Cyrillic_General_CS_AS != @firstname) or (@newlastname Collate
Cyrillic_General_CS_AS != @lastname))
				begin
					update "people"
```

```
                        set "first name" = @newfirstname, "last name" = @newlastname
                        where "people id" = @id
                  end
            end
      end

      fetch next from insertedData INTO @id, @firstname, @lastname
end

close insertedData
deallocate insertedData
go
```

3) Форматирование ссылок на файлы в Linux/Unix стиль. Внутри используется курсор

```
create trigger "unix/linux links format"
on "files info"
for insert, update
as
      declare @id int

      declare insertedData cursor local forward_only static
            for select "file id" from inserted;

      open insertedData

      fetch next from insertedData into @id

      while @@fetch_status = 0
      begin
            update "files info"
            set "link" = replace ("link", '\', '/')
            where "file id" = @id

            fetch next from insertedData INTO @id
      end

      close insertedData
      deallocate insertedData
      go
```

Скрипты создания функций

```
use "film collection"
go
```

1) Получить количество фильмов, в создании которых участвовал человек

```
create function "get count of films by person" (@firstname varchar(80), @lastname varchar(80), @birthdate date, @countryid int)
returns int
as
begin
	declare @personid int
	declare @count int
	set @personid = 0
	select @personid = "people"."people id"
	from "people"
	where @firstname = "people"."first name" and
		@lastname = "people"."last name" and
		@birthdate = "people"."birth date" and
		@countryid = "people"."country id"
	select @count = count(distinct "people in film"."film id")
	from "people in film"
	where "people id" = @personid
	return @count
end
go
```

2) Получить год самого старого фильма в базе

```
create function "get min year" ()
returns int
as
begin
	declare @min int
	select @min = year(min("films info"."year"))
	from "films info"
	return @min
end
go
```

3) Получить количество людей, участвующих в создании фильма

```
create function "get count of people by film id"(@id int)
returns int
as
begin
	declare @count int
	select @count = count(distinct "people in film"."people id")
	from "people in film"
	where "film id" = @id
	return @count
end
```

Скрипты вызова функций

```
use "film collection"
go

select "dbo"."get count of films by person" ('Питер', 'Джексон', '1961-10-31', 7)
select "dbo"."get count of films by person" ('Леонид', 'Гайдай'  ,'1923-01-30', 2)
select "dbo"."get min year"()
select "dbo"."get count of people by film id"(1)
```

Скрипт создания индексов

```
use "film collection"
go

create nonclustered index films
on "films info" ("localized name" asc)
with (pad_index = on)
go

create nonclustered index people
on "people" ("first name" asc, "last name" asc, "birth date" asc, "country id" asc)
go

create unique index filmsID
on "files info" ("film id" asc)
go
```

Скрипт создания представлений

```
use "film collection"
go

create view "films & files"
as
select "localized name", "link"
from "films info", "files info"
where "films info"."film id" = "files info"."film id"
go

create view "people & countries"
as
select "first name", "last name", "name"
from "people", "countries"
where "people"."country id" = "countries"."country id"
go

create view "films & restrictions"
as
```

```
select "localized name", "value"
from "films info", "age restrictions"
where "films info"."age restriction id" = "age restrictions"."restriction id"
go
```

Скрипт работы с временными объектами

```
use "film collection"
go

declare @temptable1 table ("localized name"    nvarchar(80) not null, "link" varchar(800) not null unique)

select "localized name", "link"
into #temptable1
from "films info", "files info"
where "films info"."film id" = "files info"."film id"

select * from #temptable1

drop table #temptable1
go


create procedure "#year range temp"
as
      select min(year("films info"."year")) as "min year", max(year("films info"."year")) as "max year"
      from "films info"
go

exec "#year range temp"

drop procedure "#year range temp"
go


declare @temptable1 table ("year" int)
insert into @temptable1
select year("films info"."year") from "films info"
select * from @temptable1
go
```