KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI
www.inf.upol.cz

**Jan Laštovička**
https://www.inf.upol.cz/lide/jan-lastovicka
jan.lastovicka@upol.cz
17. listopadu 12, 771 46 Olomouc

**Databáze ◇ poznámky k přednášce**

# 10. Dokumentová databáze

verze z 28. listopadu 2024

## 1 Dotaz nad více kolekcemi

Jako motivaci si vezměme kolekci $C_1$:

```
[
  {
    name: 'David Lynch',
    movie: [ { title: 'The Elephant Man' }, { title: 'Inland Empire' } ]
  },
  {
    name: 'Federico Fellini',
    movie: [ { title: 'La dolce vita' } ]
  },
  { name: 'Jan Švankmajer', movie: [] }
]
```

danou: „`name` je režisér“ a `movie` je dáno: „Režisér `name` natočil film `movie.title`.“ a kolekci $C_2$:

```
[
  {
    actorName: 'Anthony Hopkins',
    movieTitle: 'The Elephant Man'
  },
  {
    actorName: 'John Hurt',
    movieTitle: 'The Elephant Man'
  },
  {
    actorName: 'Laura Dern',
    movieTitle: 'Inland Empire'
  },
  {
    actorName: 'Marcello Mastroianni',
    movieTitle: 'La dolce vita'
  }
]
```

danou: „Herec `actorName` hrál ve filmu `movieTitle`.“

Úkolem je přidat do $C_1$ kolekční atribut `actorRole` daný: „Herec `actorRole.actorName` hrál ve filmu `actorRole.movieTitle` natočeném režisérem `name`.“ Chceme tedy získat kolekci $C_3$:

```
[
  {
    name: 'David Lynch',
    movie: [ { title: 'The Elephant Man' }, { title: 'Inland Empire' } ],
    actorRole: [
      {
        actorName: 'Anthony Hopkins',
        movieTitle: 'The Elephant Man'
      },
      {
        actorName: 'John Hurt',
        movieTitle: 'The Elephant Man'
      },
      {
        actorName: 'Laura Dern',
        movieTitle: 'Inland Empire'
      }
    ]
  },
  {
    name: 'Federico Fellini',
    movie: [ { title: 'La dolce vita' } ],
    actorRole: [
      {
        actorName: 'Marcello Mastroianni',
        movieTitle: 'La dolce vita'
      }
    ]
  },
  { name: 'Jan Švankmajer', movie: [], actorRole: [] }
]
```

Vezměme dokument $t$ nad $S$, kolekční atribut $y \in \mathrm{Y}(S)$ a atomický atribut $x \in \mathrm{Y}(S(y))$. Pak množinu hodnot z $D_x$ určenou vztahem

$$\{t'(x) \mid t' \in t(y)\}$$

označíme $t(y\,.\,x)$. Například pro dokument $t$:

```
{
  name: 'David Lynch',
  movie: [ { title: 'The Elephant Man' }, { title: 'Inland Empire' } ]
}
```

je $t(\texttt{movie.title})$ rovno $\{\texttt{"The Elephant Man"}, \texttt{"Inland Empire"}\}$.

Vezměme atomický atribut $x$ a konečnou množinu $X$ prvků z domény atributu $x$, pak

$$x \in X$$

je podmínka nad $\{x\}$, kterou dokument $t$ splňuje, jestliže $t(x) \in X$. Například dokument:

```
{
  actorName: 'John Hurt',
  movieTitle: 'The Elephant Man'
}
```

splňuje podmínku $\texttt{movieTitle} \in \{\texttt{"The Elephant Man"}, \texttt{"Inland Empire"}\}$.

Vezměme kolekce $C_1$ nad $S_1$ a $C_2$ nad $S_2$, kolekční atributy $y_1 \in \mathrm{Y}(S_1)$ a $y_2 \notin \mathrm{Y}(S_1)$ a atomické atributy $x_1 \in \mathrm{Y}(S_1(y_1))$ a $x_2 \in \mathrm{Y}(S_2)$. Pak kolekce

$$\{t \cup \{\langle y_2, \sigma_{x_2 \in t(y_1.x_1)}(C_2)\rangle\} \mid t \in C_1\}$$

je **vyhledání atributu** $y_1.x_1$ kolekce $C_1$ v atributu $x_2$ kolekce $C_2$ do $y_2$ a značíme $\lambda_{(y_1.x_1=x_2)\to y_2}(C_1, C_2)$. Například pro v motivaci uvedené kolekce $C_1, C_2, C_3$ platí, že $C_3 = \lambda_{(\texttt{movie.title=movieTitle})\to\texttt{actorRole}}(C_1, C_2)$.

# 2 Roura

Jako motivaci si vezměme kolekční proměnnou `director` danou: „Režisér `_id` se jmenuje `name`.“ s kolekčním atributem `movie` daným: „Režisér `_id` natočil film `movie.title`.“. Proměnná má hodnotou:

```
[
  {
    _id: 1,
    name: 'David Lynch',
    movie: [ { title: 'The Elephant Man' }, { title: 'Inland Empire' } ]
  },
  {
    _id: 2,
    name: 'Federico Fellini',
    movie: [ { title: 'La dolce vita' } ]
  },
  { _id: 3, name: 'Jan Švankmajer', movie: [] }
]
```

a kolekční proměnnou `role` danou: „Číslo `_id` je role herce `actorName` ve filmu `movieTitle`.“ s hodnotou:

```
[
  {
    _id: 1,
    actorName: 'Anthony Hopkins',
    movieTitle: 'The Elephant Man'
  },
  { _id: 2, actorName: 'John Hurt', movieTitle: 'The Elephant Man' },
  { _id: 3, actorName: 'Laura Dern', movieTitle: 'Inland Empire' },
  {
    _id: 4,
    actorName: 'Marcello Mastroianni',
    movieTitle: 'La dolce vita'
  }
]
```

**Roura** je dána posloupností fází $O_1, \ldots, O_n$ a schémat $S_1, \ldots, S_{n+1}$. Každá fáze $O_i$ je zobrazení, které kolekci nad $S_i$ přiřazuje kolekci nad $S_{i+1}$. Roura se **aplikuje** na kolekci $C_1$ nad $S_1$. Vzniká posloupnost kolekcí $C_2, \ldots, C_{n+1}$ taková, že $C_{i+1} = O_i(C_i)$ pro každé $1 \leq i \leq n$. Tedy $C_2 = O_1(C_1)$, $C_3 = O_2(C_2)$, $\ldots$, $C_{n+1} = O_n(C_n)$. Výsledkem aplikace roury je kolekce $C_{n+1}$ nad $S_{n+1}$. Platí, že

$$C_{n+1} = O_n(O_{n-1}(\ldots O_2(O_1(C_1))\ldots)).$$

Fázi $O_i$ můžeme definovat jako

1. projekci: $O_i(C_i) = \pi_S(C_i)$,
2. restrikci: $O_i(C_i) = \sigma_\theta(C_i)$,
3. nebo vyhledání: $O_i(C_i) = \lambda_{(y_1.x_1=x_2)\to y_2}(C_i, C_{collection})$.

Například uvažujme rouru s fázemi $\sigma_{\texttt{movie.(title="Inland Empire")}}(C_1)$, $\pi_{\{\texttt{name}\}}(C_2)$, kterou aplikujeme na $C_1$:

```
[
  {
    name: 'David Lynch',
    movie: [ { title: 'The Elephant Man' }, { title: 'Inland Empire' } ]
  },
  {
    name: 'Federico Fellini',
    movie: [ { title: 'La dolce vita' } ]
  },
  { name: 'Jan Švankmajer', movie: [] }
]
```

výsledkem aplikace je kolekce $\pi_{\{\texttt{name}\}}(\sigma_{\texttt{movie.(title="Inland Empire"}}(C_1))$:

```
[ { name: 'David Lynch' } ]
```

dána: „Režisér `name` natočil film Inland Empire.“

Fázi typu projekce $O_i(C_i) = \pi_S(C_i)$ zapíšeme řetězcem:

```
{ $project: S }
```

fázi typu restrikce $O_i(C_i) = \sigma_\theta(C_i)$ zapíšeme řetězcem:

```
{ $match: θ }
```

a fázi typu vyhledání $O_i(C_i) = \lambda_{(y_1.x_1=x_2)\to y_2}(C_i, C_{collection})$ zapíšeme řetězcem:

```
{
  $lookup: {
    from: "collection",
    localField: "y1.x1",
    foreignField: "x2",
    as: "y2"
  }
}
```

Například restrikci $\sigma_{\texttt{movie.(title="Inland Empire")}}(C_i)$ zapíšeme řetězcem:

```
{
  $match: {
    movie: {
      $elemMatch: {
        title: "Inland Empire"
      }
    }
  }
}
```

projekci $\pi_{\{\texttt{name}\}}(C_i)$ zapíšeme řetězcem:

```
{
  $project: { _id: 0, name: 1 }
}
```

a vyhledávání $\lambda_{\texttt{(movie.title=movieTitle)}\to\texttt{actorRole}}(C_i, C_{\texttt{role}})$ řetězcem:

```
{
  $lookup: {
```

```
      from: "role",
      localField: "movie.title",
      foreignField: "movieTitle",
      as: "actorRole"
  }
}
```

Hodnota kolekčního výrazu:

```
db.collection.aggregate([O_1, ..., O_n])
```

je rovna výsledku aplikace roury $O_1, \ldots, O_n$ na hodnotu kolekční proměnné *collection*.

Například:

```
> db.director.aggregate([{
   $match: {
     movie: {
       $elemMatch: {
         title: "Inland Empire"
       }
     }
   }
 },{
   $project: { _id: 0, name: 1 }
 }])

[ { name: 'David Lynch' } ]
```

je kolekce dána: „Řetězec `name` je jméno režiséra, který natočil film Inland Empire.“
Dále:

```
> db.director.aggregate([{
   $lookup: {
       from: "role",
       localField: "movie.title",
       foreignField: "movieTitle",
       as: "actorRole"
   }
 }])

[
  {
    _id: 1,
    name: 'David Lynch',
    movie: [ { title: 'The Elephant Man' }, { title: 'Inland Empire' } ],
```

```
    actorRole: [
      {
        _id: 1,
        actorName: 'Anthony Hopkins',
        movieTitle: 'The Elephant Man'
      },
      {
        _id: 2,
        actorName: 'John Hurt',
        movieTitle: 'The Elephant Man'
      },
      { _id: 3, actorName: 'Laura Dern', movieTitle: 'Inland Empire' }
    ]
  },
  {
    _id: 2,
    name: 'Federico Fellini',
    movie: [ { title: 'La dolce vita' } ],
    actorRole: [
      {
        _id: 4,
        actorName: 'Marcello Mastroianni',
        movieTitle: 'La dolce vita'
      }
    ]
  },
  { _id: 3, name: 'Jan Švankmajer', movie: [], actorRole: [] }
]
```

je kolekce dán: „Režisér `_id` se jmenuje `name`.“, kde `movie` je dáno: „Režisér `_id` natočil film `movie.title`.“ a `actorRole` je dáno: „Číslo `actorRole._id` je role herce `actorRole.actorName` ve filmu `actorRole.movieTitle` natočeném režisérem `_id`.“

V závěrečném příkladu je:

```
> db.director.aggregate([{
   $lookup: {
       from: "role",
       localField: "movie.title",
       foreignField: "movieTitle",
       as: "actorRole"
   }
 }, {
   $match: {
     actorRole: {
       $elemMatch: {
```

```
        actorName: "Anthony Hopkins"
      }
    }
  }
}, {
  $project: { _id: 0, name: 1 }
}])
```

```
[ { name: 'David Lynch' } ]
```

kolekce dána: „Řetězec `name` je jméno režiséra, v jehož filmu hraje Anthony Hopkins.“

# 3 Deklarace schématu

Celočíselnou doménu zapíšeme řetězcem:

```
{ bsonType: "int"}
```

a doménu řetězců zapíšeme řetězcem:

```
{ bsonType: "string"}
```

Schéma $\{x_1, \ldots, x_m, y_1.S_1, \ldots, y_n.S_n\}$, kde $x_1, \ldots, x_m$ jsou atomické a $y_1, \ldots, y_n$ kolekční atributy, zapíšeme řetězcem:

```
{
  bsonType: "object",
  required: [ x1, ..., xm, y1, ..., yn ],
  additionalProperties: false,
  properties: {
    x1: Dx1,
    ⋮
    xm: Dxm,
    y1: {
      bsonType: "array",
      items: S1
    },
    ⋮
    yn: {
      bsonType: "array",
      items: Sn
```

```
    }
  }
}
```

Například schéma {`_id`, `actorName`, `movieTitle`} zapíšeme:

```
{
  bsonType: "object",
  required: [ "_id", "actorName", "movieTitle" ],
  additionalProperties: false,
  properties: {
    _id: { bsonType: "int" },
    actorName: { bsonType: "string" },
    movieTitle: { bsonType: "string" }
  }
}
```

a schéma {`_id`, `name`, `movie.{title}`} zapíšeme:

```
{
  bsonType: "object",
  required: [ "_id", "name", "movie" ],
  additionalProperties: false,
  properties: {
    _id: { bsonType: "int" },
    name: { bsonType: "string" },
    movie: {
      bsonType: "array",
      items: {
        bsonType: "object",
        required: [ "title" ],
        properties: {
          title: { bsonType: "string"}
        }
      }
    }
  }
}
```

Před vložením prvního dokumentu do kolekční proměnné *`collection`* můžeme deklarovat její typ $S$ příkazem:

```
db.createCollection("collection", {
  validator: {
    $jsonSchema: S
```

```
  }
})
```

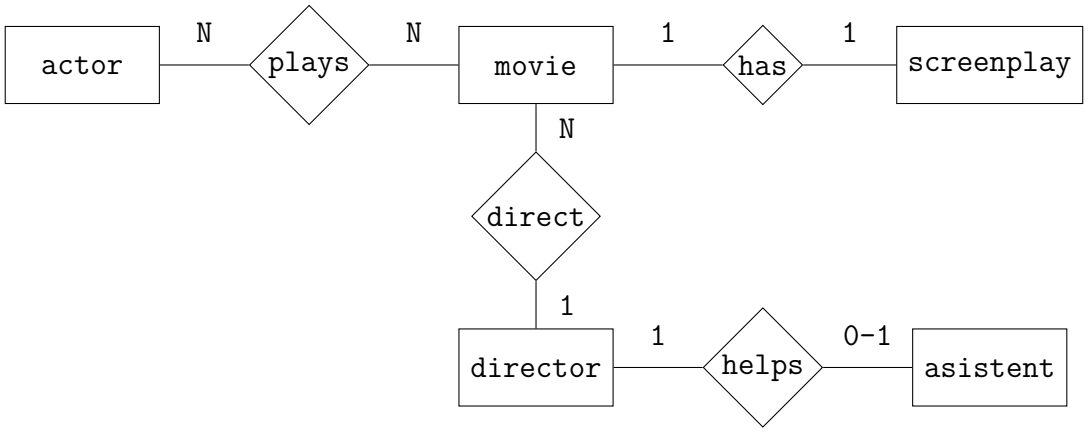
Systém pak zaručí, že hodnota relační proměnné ***collection*** bude vždy kolekce nad $S$. Například:

```
db.createCollection("role", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: [ "_id", "actorName", "movieTitle" ],
      additionalProperties: false,
      properties: {
        _id: { bsonType: "int" },
        actorName: { bsonType: "string" },
        movieTitle: { bsonType: "string" }
      }
    }
  }
})
```

deklaruje kolekční proměnnou `role` nad $\{$`_id`, `actorName`, `movieTitle`$\}$.

# 4 Návrh dokumentové databáze

Vyjdeme z entitně vztahového modelu databáze. Například vezmeme model z páté přednášky:




Podle postupu z téže přednášky jej převedeme na relační model. Převodem výše uvedeného diagramu obdržíme základní relace:

1. Základní relace `movie`:

- Schéma: {movie_id, title, screenplay_id, scene_count, director_id}
- Charakteristická vlastnost: „Film movie_id s názvem title z roku year byl natočený režisérem director_id podle scénáře screenplay_id obsahující scene_count scén.“
- Primární klíč: {movie_id}
- Alternativní klíče: {screenplay_id}
- Cizí klíče: {director_id} na director

2. Základní relace actor:

   - Schéma: {actor_id, name, born}
   - Charakteristická vlastnost: „Herce actor_id se jmenuje name a narodil se roku born.“
   - Primární klíč: {actor_id}

3. Základní relace director:

   - Schéma: {director_id, name, oscar_count}
   - Charakteristická vlastnost: „Režisér director_id se jmenuje name a získal oscar_count Oskarů.“
   - Primární klíč: {director_id}

4. Základní relace asistent:

   - Schéma: {asistent_id, name, director_id}
   - Charakteristická vlastnost: „Asistent asistent_id se jmenuje name a je k ruce režisérovi director_id.“
   - Primární klíč: {asistent_id}
   - Cizí klíče: {director_id} na director

5. Základní relace movie_cast:

   - Schéma: {actory_id, movie_id}
   - Charakteristická vlastnost: „Herec actor_id hrál ve filmu movie_id.“
   - Primární klíč: {actory_id, movie_id}
   - Cizí klíče: {actor_id} na actor, {movie_id} na movie

Pro každou základní relaci vytvoříme kolekční proměnnou nad schématem, kde přejmenujeme atribut v primárním klíči na _id, pokud je jediný, nebo do schématu přidáme _id. V dalším nás budou zajímat charakteristické vlastnosti a cizí klíče takto obdržených kolekčních proměnných.

Pro výše uvedené základní relace dostáváme:

1. Kolekční proměnná movie:

- Schéma: {`_id`, `title`, `screenplayId`, `sceneCount`, `directorId`}
- Charakteristická vlastnost: „Film `_id` s názvem `title` z roku `year` byl natočený režisérem `directorId` podle scénáře `screenplayId` obsahující `sceneCount` scén.“
- Cizí klíče: {`directorId`} na `director`

2. Kolekční proměnná `actor`:
   - Schéma: {`_id`, `name`, `born`}
   - Charakteristická vlastnost: „Herce `_id` se jmenuje `name` a narodil se roku `born`.“
3. Kolekční proměnná `director`:
   - Schéma: {`_id`, `name`, `oscarCount`}
   - Charakteristická vlastnost: „Režisér `_id` se jmenuje `name` a získal `oscarCount` Oskarů.“
4. Kolekční proměnná `asistent`:
   - Schéma: {`_id`, `name`, `directorId`}
   - Charakteristická vlastnost: „Asistent `_id` se jmenuje `name` a je k ruce režisérovi `directorId`.“
   - Cizí klíče: {`directorId`} na `director`
5. Kolekční proměnná `movieCast`:
   - Schéma: {`_id`, `actoryId`, `movieId`}
   - Charakteristická vlastnost: „Herec `actorId` hrál ve filmu `movieId` roli `_id`.“
   - Cizí klíče: {`actorId`} na `actor`, {`movieId`} na `movie`

Dokud je to možné, opakujeme následující.

1. Vezmeme kolekční proměnnou ***`collection1`*** nad $S_1$, která má cizí klíč $\{y_1\}$ na kolekční proměnnou ***`collection2`*** nad $S_2$.
2. Kolekční proměnnou ***`collection1`*** odstraníme.
3. Změníme schéma kolekční proměnné ***`collection2`*** na $S_2 \cup \{\langle y_2, S_1 - \{y_1\}\rangle\}$, kde $y_2$ je nový kolekční atribut.
4. Přeformulujeme charakteristickou vlastnost kolekční proměnné ***`collection1`*** na charakteristickou vlastnost kolekčního atributu $y_2$.

Například z kolekční proměnné `movieCast` můžeme učinit kolekční atribut kolekční proměnné `actor`:

- Schéma: {`_id`, `name`, `born`, `movieCast`.{`_id`, `movieId`}}
- Charakteristická vlastnost: „Herce `_id` se jmenuje `name` a narodil se roku `born`.“
- Charakteristická vlastnost `movieCast`: „Herec `_id` hrál ve filmu `movieCast.movieId` roli `movieCast._id`.“

Podobně můžeme přesunout kolekční proměnné `movie` a `asistent` do kolekční proměnné `director`:

- Schéma:

  {`_id`, `name`, `oscarCount`,
  `movie`.{`_id`, `title`, `screenplayId`, `sceneCount`},
  `asistent`.{`_id`, `name`}}

- Charakteristická vlastnost: „Režisér `_id` se jmenuje `name` a získal `oscarCount` Oskarů.“
- Charakteristická vlastnost `movie`: „Film `movie._id` s názvem `movie.title` z roku `movie.year` byl natočený režisérem `_id` podle scénáře `movie.screenplayId` obsahující `movie.sceneCount` scén.“
- Charakteristická vlastnost `asistent`: „Asistent `asistent._id` se jmenuje `asistent.name` a je k ruce režisérovi `_id`.“

Následuje deklarace dvou výsledných kolekčních proměnných.

```
db.createCollection("director", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: [ "_id", "name", "oscarCount", "movie", "asistent" ],
      additionalProperties: false,
      properties: {
        _id: { bsonType: "int" },
        name: { bsonType: "string" },
        oscarCount: { bsonType: "int" },
        movie: {
          bsonType: "array",
          items: {
            bsonType: "object",
            required: [ "_id", "title", "screenplayId", "sceneCount" ],
            properties: {
              _id: { bsonType: "int" },
              title: { bsonType: "string"},
```

```
              screenplayId: { bsonType: "int" },
              sceneCount: { bsonType: "int" }
            }
          }
        },
        asistent: {
          bsonType: "array",
          items: {
            bsonType: "object",
            required: [ "_id", "name" ],
            properties: {
              _id: { bsonType: "int" },
              name: { bsonType: "string"}
            }
          }
        }
      }
    }
  }
})

db.createCollection("actor", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: [ "_id", "name", "born", "movieCast" ],
      additionalProperties: false,
      properties: {
        _id: { bsonType: "int" },
        name: { bsonType: "string" },
        born: { bsonType: "int" },
        movieCast: {
          bsonType: "array",
          items: {
            bsonType: "object",
            required: [ "_id", "movieId" ],
            properties: {
              _id: { bsonType: "int" },
              movieId: { bsonType: "int"}
            }
          }
        }
      }
    }
  }
})
```

# Otázky a úkoly na cvičení

1. Vezměme dokument $t$:

```
{
  _id: 1,
  name: 'Andrei Tarkovsky',
  movie: [
    { title: 'Solaris', year: 1972 },
    { title: 'Stalker', year: 1979 }
  ]
}
```

   Určete množiny hodnot označené $t$(`movie.title`) a $t$(`movie.year`).

2. Určete výsledek restrikce kolekce:

```
[
  { _id: 1, title: 'Gladiator', year: 1992 },
  { _id: 2, title: 'Gladiator', year: 2000 },
  { _id: 3, title: 'X-Men', year: 2000 },
  { _id: 4, title: 'The Matrix', year: 1999 }
]
```

   s charakteristickou vlastností „Film `_id` se jmenuje `title` a byl vydán v roce `year`.“ vzhledem k podmínce `year` $\in \{1992, 1999\}$. Jaká bude její charakteristická vlastnost?

3. Vezměme kolekci $C_1$:

```
[ {
  _id: 1,
  title: 'Megalopolis',
  actor_ref: [
    { id: 1},
    { id: 2 }
  ]
}, {
  _id: 2,
  title: 'Ferrari',
  actor_ref: [
    { id: 1 },
    { id: 3 }
  ]
} ]
```

s charakteristickou vlastností „Film `_id` se jmenuje `title`“ a kolekčním atributem `actor_ref` s charakteristickou vlastností „Ve filmu `_id` hrál herec `actor_ref.id`.“ a kolekci $C_2$:

```
[ { _id: 1, name: 'Adam Driver', born: 1983 },
  { _id: 2, name: 'Laurence Fishburne', born: 1961 },
  { _id: 3, name: 'Penélope Cruz', born: 1974 }
]
```

s charakteristickou vlastností „Herec `_id` se jmenuje `name` a narodil se roku `born`.“ Určete dokumenty a charakteristickou vlastnost kolekce

$$\lambda_{(\texttt{actor\_ref.id=\_id})\rightarrow\texttt{actor}}(C_1, C_2).$$

4. Vložme kolekci $C_1$ z předchozího příkladu do proměnné `movie` a kolekci $C_2$ do proměnné `actor`. Určete hodnotu a charakteristickou vlastnost následujícího výrazu.

```
db.movie.aggregate([{
   $lookup: {
       from: "actor",
       localField: "actor_ref.id",
       foreignField: "_id",
       as: "actor"
   }
 }, {
   $match: {
     actor: {
       $elemMatch: {
         born: 1961
       }
     }
   }
 }, {
   $project: { _id: 0, title: 1 }
 }])
```

   Hodnotu výrazu si ověřte v MongoDB.

5. Vezměme proměnné z předchozího příkladu. Napište kolekční výraz s charakteristickou vlastností „Ve filmu `title` hrála Penélope Cruz.“ a určete jeho hodnotu. Vámi určenou hodnotu si ověřte v MongoDB.

6. Deklarujte v MongoDB schéma pro proměnné `movie` a `actor` z předchozích příkladů.

7. Vyjděte z návrhu entitně vztahového modelu z páté přednášky, který jste vytvořili v pátém příkladu a jeho relačního modelu z následujícího příkladu. Převeďte relační model do dokumentového modelu.