

Vypsane testy z FITwiki:

- [23. 12. 2023 Valenta](#) ✓
- [5. 12. 2023 Trofimova](#) ✓
- [14. 12. 2021 Valenta 14:30](#) ✓
- [30. 11. 2021 Valenta](#) ✓
- [17. 12. 2018 Valenta 16:15](#) ✓
- [11. 12. 2017 Valenta 14:30](#) ✓

## Verze A

### Část 1 - SQL optimalizace

Byly zadány dvě relace  $R[A]$  a  $S[B,D]$ . Atribut A může nabývat 10 různých hodnot. Pak byly zadné  $N_r = 1000$   $B_r = 2$  a  $N_s = 1000$   $B_s = 50$ .

#### Příklad 1 (2 body)

Dopočítejte statistiky  $P_r$  a  $P_s$ :

**Řešení:**

$$P_r = N_r/B_r = 1000/2 = 500$$

$$P_s = N_s/B_s = 1000/50 = 20$$

#### Příklad 2 (5 bodů)

Nakreslete strom plánu vyhodnocení pro následující dotaz (byl zadáný v algebře i SQL)

$(R(A=5)*S)[B,D]$

`SELECT B,D FROM (SELECT * FROM R WHERE A=5) CROSS JOIN S`

Byla tam ještě poznámka, že máme brát v potaz uzávorkování výrazu v algebře

**Řešení**



### Příklad 3 (5 bodů)

Uvažujte nested loop join a určete kolik paměťových stránek bude potřeba k vyhodnocení plánu. Byli tam ještě zadané další statistiky, že blokový faktor B a D = 100.

### Příklad 4 (3 body)

Nakreslete strom plánu vyhodnocení pro výraz jako v 2, ale efektivnější.

## Část 2 - MongoDB (7 bodů) ✓

```

db.actors.save({ _id: "trojan", name: { first: "Ivan", last: "Trojan" },
year: 1964, movies: [ "samotari", "medvidek", "karamazovi" ] });
db.actors.save({ _id: "machacek", name: { first: "Jiri", last: "Machacek"
}, year: 1966, movies: [ "medvidek", "vratnelahve", "samotari" ] });
db.actors.save({ _id: "schneiderova", name: { first: "Jitka", last:
"Schneiderova" }, year: 1973, movies: [ "samotari" ] });
db.actors.save({ _id: "sverak", name: { first: "Zdenek", last: "Sverak"
}, year: 1936, movies: [ "vratnelahve" ] });
db.actors.save({ _id: "geislerova", name: { first: "Anna", last:
"Geislerova" }, year: 1976 });
db.actors.save({ _id: "vilhelmoval", name: { first: "Tatiana", last:
"Vilhelmoval" }, year: 1978, movies: [ "medvidek" ] });
db.actors.save({ _id: "menzel", name: { last: "Menzel", first: "Jiri" },
year: 1938, movies: "medvidek" });

db.movies.save({ _id: "samotari", title: { cs: "Samotari", en: "Loners"

```

```

}, year: 2000, rating: 84, actors: [ "trojan", "machacek", "schneiderova"
], genres: [ "comedy", "drama" ], country: [ "CZ", "SI" ], length: 103
});
db.movies.save({ _id: "medvidek", title: "Medvidek", year: 2007,
director: { first: "Jan", last: "Hrebejk" }, rating: 53, actors: [
"trojan", "machacek", "vilhelмова", "issova", "menzel" ], genres: [
"comedy", "drama" ], country: [ "CZ" ], length: 100 });
db.movies.save({ _id: "vratnelahve", title: { cs: "Vratne lahve", en:
"Empties" }, year: 2006, director: { first: "Jan", last: "Sverak" },
rating: 76, actors: [ "sverak", "machacek", "schneiderova" ], genres:
"comedy", country: "CZ", length: 99 });
db.movies.save({ _id: "zelary", title: "Zelary", year: 2003, director: {
last: "Trojan", first: "Ondrej" }, rating: 81, actors: [ ], genres: [
"romance", "drama" ], country: [ "CZ", "SK", "AT" ], length: 142 });
db.movies.save({ _id: "stesti", title: "Stesti", year: 2005, director: {
last: "Slama", first: "Bohdan" }, rating: 72, length: 100, awards: [ {
type: "Czech Lion", year: 2005 } ] });
db.movies.save({ _id: "kolja", title: "Kolja", year: 1996, rating: 86,
length: 105, awards: [ { type: "Czech Lion", year: 1996 }, { type:
"Noname Awards", category: "A", year: 2005 } ] });

```

## Příklad 1 (2 body) ✓

Vyberte herce narozené před rokem 1967 s křestním jménem Jiri.

### Řešení

Řešení podle Morčína:

```

db.actors.find({
  year: {
    "$lt": 1967
  },
  "name.first": "Jiri"
})

```

## Příklad 2 (2 body) ✓

Vypište herce co hráli zároveň ve filmech samotari a medvidek, a vypište jen jejich id a křestní jméno.

### Řešení

Řešení podle Morčína:

```
db.actors.find({
  movies: {
    "$all": ["samotari", "medvidek"]
  },
},
{_id:1, "name.first":1})
```

### Příklad 3 (3 body) ✓

Herci, kteří hrají ve filmech medvídek nebo samotari. Seřadte výstup vzestupně dle příjmení herce.

### Řešení

Řešení podle Morčina:

```
db.actors.find({
  movies: {
    $in: ["samotari", "medvidek"]
  }
}).sort({"name.last": 1})
```

### Část 3 - Neo4j ✓

Graf letišť, jako ze cvičení:

```
CREATE
(sf {name:'San Francisco', code:'sf'}),
(la {name:'Los Angeles', code:'la'}),
(da {name:'Dallas', code:'da'}),
(ch {name:'Chicago', code:'ch'}),
(ny {name:'New York', code:'ny'}),
(sf)-[:DIRECT {price:50}]->(la),
(la)-[:DIRECT {price:50}]->(sf),
(sf)-[:DIRECT {price:250}]->(ch),
(ch)-[:DIRECT {price:250}]->(sf),
(da)-[:DIRECT {price:300}]->(sf),
(sf)-[:DIRECT {price:300}]->(da),
(ch)-[:DIRECT {price:100}]->(da),
(da)-[:DIRECT {price:100}]->(ch),
(ch)-[:DIRECT {price:250}]->(ny),
(ny)-[:DIRECT {price:250}]->(ch),
(ny)-[:DIRECT {price:225}]->(da),
(da)-[:DIRECT {price:225}]->(ny),
```

```
(da)-[:DIRECT {price:200}]->(la),  
(la)-[:DIRECT {price:200}]->(da);
```

## Příklad 1 (2 body) ✓

Vypište všechna města jejichž kód je větší než „ch“.

### Řešení

Řešení podle Morčina:

```
MATCH (s)  
WHERE s.code > "ch"  
RETURN s
```

## Příklad 2 (3 body) ✓

Vypište všechna města kam se dá dostat z Dallasu přímým spojením. Byla tam ještě nějaká podmínka na řazení myslím...

### Řešení

Řešení podle Morčina:

```
MATCH (s {name: "Dallas"})-[:DIRECT]->(d)  
RETURN DISTINCT d.name
```

## Příklad 3 (3 body) ✓

Vypsát čárkou oddělený seznam kódů měst kam se dá dostat z New Yorku s maximálně 4 přestupy.

### Řešení

```
MATCH path=(s{name: 'New York'})-[:DIRECT *..5]->(d)  
RETURN [x in nodes(path) | x.code]
```

## Část 4 - XQuery

Zde bylo zadáno XML něco ve stylu:

```
<body>  
  <div id="book132">
```

```

    <h1>Název knihy 1</h1>
    <div>
      
    </div>
  </div>
  <div id="book221">
    <h1>Název knihy 2</h1>
    <div>
      
    </div>
  </div>
  <div id="book114">
    <h1>Název knihy 3</h1>
    <div>
      <div></div>
    </div>
  </div>
</body>

```

## Příklad 1 (5 bodů)

Vypište ID divů, které obsahují obrázek, který končí na jpg.

## Příklad 2 (5 bodů)

Vypište abecedně seřazené knihy podle jejího názvu (element h1) a počet obrázků, které kniha obsahuje (v libovolně vnořené úrovni). Výpis proveďte v následující struktuře.

```

<element>
  <book>název knihy</book>
  <images>počet obrázků</images>
</element>

```

```

let $books := //div[boolean(../h1)]

for $book in $books
  let $name := $book/h1/text()
  order by $name
  return
    <element>
      <book>{$name}</book>
      <images>{count($book//img)}</images>
    </element>

```

# Verze B

## Část 1 (7 bodů)

Mějme relace  $R(A, B)$  a  $S(X, Y, Z)$ . Předpokládejme, že  $n_R = 2\,000$  (počet záznamů),  $b_R = 10$  (blokovací faktor),  $n_S = 1\,000$  a  $b_S = 10$ . Atribut  $A$  nabývá  $v_{R.A} = 50$  různých hodnot (každá z nich je stejně pravděpodobná). Navíc máme k dispozici clusterovaný index pro atribut  $A$  v relaci  $R$ . Jeho hloubka je  $i_{R.A} = 2$  a listová úroveň obsahuje  $p_{R.A} = 50$  bloků.

### Příklad 1 (1 bod)

Dopočítejte statistiky  $P_r$  a  $P_s$

### Příklad 2 (2 body)

Vytvořte prováděcí plán ve formě stromu pro vyhodnocení dotazu (byl zadáný v algebře i SQL)

$(R(A = 13) \times S)$

`select from (select from R where A=13) cross join S;`

Byla tam ještě poznámka, že máme brát v potaz uzávorkování výrazu v algebře

### Příklad 3 (3 body)

Uvažujte plán vyhodnocení, ve kterém se spojení realizuje pomocí algoritmu Nested Loops. Pro mezivýsledek každé dílčí operace určete počty záznamů, blokovací faktor a počet bloků. Nakonec určete celkovou cenu vyhodnocení celého plánu. K dispozici máme systémovou paměť o velikosti  $M = 22$  bloků. Postup komentujte.

### Příklad 4 (1 bod)

Navrhnete (opět formou syntaktického stromu jako výše), jak by bylo možné získat rychlejší plán vyhodnocení. Pro tento plán již nemusíte nic dalšího počítat.

## Část 2 - MongoDB (7 bodů) ✓

Byla zadána DB jako byla ve cvičeních tj

```
db.actors.save({ _id: "trojan", name: { first: "Ivan", last: "Trojan" },
year: 1964, movies: [ "samotari", "medvidek", "karamazovi" ] });
db.actors.save({ _id: "machacek", name: { first: "Jiri", last: "Machacek" },
year: 1966, movies: [ "medvidek", "vratnelahve", "samotari" ] });
```

```

db.actors.save({ _id: "schneiderova", name: { first: "Jitka", last:
"Schneiderova" }, year: 1973, movies: [ "samotari" ] });
db.actors.save({ _id: "sverak", name: { first: "Zdenek", last: "Sverak"
}, year: 1936, movies: [ "vratnelahve" ] });
db.actors.save({ _id: "geislerova", name: { first: "Anna", last:
"Geislerova" }, year: 1976 });
db.actors.save({ _id: "vilhelmovva", name: { first: "Tatiana", last:
"Vilhelmovva" }, year: 1978, movies: [ "medvidek" ] });
db.actors.save({ _id: "menzel", name: { last: "Menzel", first: "Jiri" },
year: 1938, movies: "medvidek" });

db.movies.save({ _id: "samotari", title: { cs: "Samotari", en: "Loners"
}, year: 2000, rating: 84, actors: [ "trojan", "machacek", "schneiderova"
], genres: [ "comedy", "drama" ], country: [ "CZ", "SI" ], length: 103
});
db.movies.save({ _id: "medvidek", title: "Medvidek", year: 2007,
director: { first: "Jan", last: "Hrebejk" }, rating: 53, actors: [
"trojan", "machacek", "vilhelmovva", "issova", "menzel" ], genres: [
"comedy", "drama" ], country: [ "CZ" ], length: 100 });
db.movies.save({ _id: "vratnelahve", title: { cs: "Vratne lahve", en:
"Empties" }, year: 2006, director: { first: "Jan", last: "Sverak" },
rating: 76, actors: [ "sverak", "machacek", "schneiderova" ], genres:
"comedy", country: "CZ", length: 99 });
db.movies.save({ _id: "zelary", title: "Zelary", year: 2003, director: {
last: "Trojan", first: "Ondrej" }, rating: 81, actors: [ ], genres: [
"romance", "drama" ], country: [ "CZ", "SK", "AT" ], length: 142 });
db.movies.save({ _id: "stesti", title: "Stesti", year: 2005, director: {
last: "Slama", first: "Bohdan" }, rating: 72, length: 100, awards: [ {
type: "Czech Lion", year: 2005 } ] });
db.movies.save({ _id: "kolja", title: "Kolja", year: 1996, rating: 86,
length: 105, awards: [ { type: "Czech Lion", year: 1996 }, { type:
"Noname Awards", category: "A", year: 2005 } ] });

```

## Příklad 1 (2 body) ✓

Herci, narození dříve než 1970 s křestním jménem Jiří.

### Řešení

Řešení podle Morčína:

```

db.actors.find({
  year: {
    "$lt": 1970
  },

```



```
"name.first": "Jiri"  
})
```

## Příklad 2 (2 body) ✓

Filmy, které natočil Jan Hřebejk (director). Vypište pouze ID a názvy filmů.

### Řešení

Řešení podle Morčina:

```
db.movies.find({  
  "director.first": "Jan",  
  "director.last": "Hrebejk"  
},  
{  
  _id: 1,  
  title: 1  
})
```

## Příklad 3 (3 body) ✓

Herci, kteří hrají ve filmech medvídek a samotari. Seřadte výstup vzestupně dle příjmení herce.

### Řešení

Řešení podle Morčina:

```
db.actors.find({  
  movies: {  
    $all: ["samotari", "medvidek"]  
  }  
}).sort({"name.last": 1})
```

## Část 3 - Neo4j (8 bodů) ✓

Graf letišť, jako ze cvičení:

```
CREATE  
(sf {name:'San Francisco', code:'sf'}),  
(la {name:'Los Angeles', code:'la'}),  
(da {name:'Dallas', code:'da'}),  
(ch {name:'Chicago', code:'ch'}),
```

```
(ny {name: 'New York', code: 'ny'}),
(sf)-[:DIRECT {price:50}]->(la),
(la)-[:DIRECT {price:50}]->(sf),
(sf)-[:DIRECT {price:250}]->(ch),
(ch)-[:DIRECT {price:250}]->(sf),
(da)-[:DIRECT {price:300}]->(sf),
(sf)-[:DIRECT {price:300}]->(da),
(ch)-[:DIRECT {price:100}]->(da),
(da)-[:DIRECT {price:100}]->(ch),
(ch)-[:DIRECT {price:250}]->(ny),
(ny)-[:DIRECT {price:250}]->(ch),
(ny)-[:DIRECT {price:225}]->(da),
(da)-[:DIRECT {price:225}]->(ny),
(da)-[:DIRECT {price:200}]->(la),
(la)-[:DIRECT {price:200}]->(da);
```

## Příklad 1 (2 body) ✓

Vypište trojice <cíl, cena, start> pro přímé lety, výstup seřadíte abecedně dle startu a ceny.

### Řešení

Řešení podle Morčina:

```
MATCH (s)-[r:DIRECT]->(d)
RETURN d.name AS cil, r.price AS cena, s.name AS start
ORDER BY start, cena
```

## Příklad 2 (3 body) ✓

Vypište letiště dosažitelná z Los Angeles pomocí právě jednoho přestupu.

### Řešení

Řešení podle Morčina:

```
MATCH (s {name: 'Los Angeles'})-[:DIRECT *2]->(d)
RETURN DISTINCT d
```

Pozn. to **DISTINCT** tam asi nemusí nutně být.

## Příklad 3 (3 body) ✓

Vypište cesty (kódy letišť oddělené čárkou) začínající v Los Angeles do maximální délky 4.

## Řešení

Řešení podle Morčina:

```
MATCH path=(s {name:'Los Angeles'})-[:DIRECT *..4]->(d)
RETURN [x in nodes(path) | x.code]
```

Takhle jsme to měli na cviku, nevím co přesně myslí tím "kódy letišť oddělené čárkou".  
Kód výše má výstup v tomto formátu:

```
["la", "sf", "la"]
```

Jinak pokud čárkou myslí "-" a fakt to chtějí v jednom stringu, tak by to mělo být takhle:

```
MATCH path = (s {name: 'Los Angeles'})-[:DIRECT*..4]->(d)
RETURN REDUCE(result = '', x IN nodes(path) | result + CASE WHEN result = '' THEN '' ELSE '-' END + x.code) AS cesta
```

Tohle ale myslím že je až moc fancy aby to po nás chtěli :D

## Část 4 (10 bodů)

Zde bylo zadané XML něco ve stylu:

```
<html><body>
  <div id="kniha1">
    <h1>Mravenečkova dobrodružství</h1>
    <div>
      
    </div>
  </div>
  <div id="kniha2">
    <h1>1984</h1>
    
  </div>
  <div id="kniha3">
    <h1>Cesta kolem světa</h1>
    <div>
      Autor: <strong>Jules Verne</strong>
      <div></div>
    </div>
    
  </div>
```

```
<div id="kniha4">
    <h1>Pravidla českého pravopisu</h1>
</div>
</body></html>
```

## Příklad 1 (4 body)

Napište dotaz, který vybere id všech elementů div, které obsahují (jakkoliv vnořeně) obrázek s příponou jpg.

## Příklad 2 (4 bodů)

Napište dotaz, který setřídí elementy div s nějakým id podle obsahu elementu h1 a na výstup vypíše jméno knihy (z h1) a počet obrázků (img) ke knize přiřazených. Příklad:

```
<zaznam><kniha>Cesta kolem světa</kniha><obr>2</obr></zaznam>
```

## Verze C

Mějme relace  $R(X, Y)$  a  $S(Y, Z)$ , kde  $S.Y$  je cizím klíčem do  $R.Y$ . Předpokládejme, že  $n_R = 10\,000$  (počet záznamů),  $b_R = 100$  (blokovací faktor),  $n_S = 100\,000$  a  $b_S = 40$ . Atribut  $Z$  nabývá v  $S.Z = 250$  různých hodnot (každá z nich je stejně pravděpodobná).

## Část 1 - SQL optimalizace (7 bodů)

### Příklad 1 (1 bod)

Dopočítejte statistiky  $P_r$  a  $P_s$

### Příklad 2 (2 body)

Vytvořte prováděcí plán ve formě stromu pro vyhodnocení dotazu (byl zadáný v algebře i SQL)

**$(S(Z = 56) * R)[X]$**

**select X from (select \* from S where Z=56) natural join R;**

Byla tam ještě poznámka, že máme brát v potaz uzávorkování výrazu v algebře

### Příklad 3 (3 body)

Uvažujte plán vyhodnocení, ve kterém se spojení realizuje pomocí algoritmu Nested Loops. Pro mezivýsledek každé dílčí operace určete počty záznamů, blokovací faktor a

počet bloků. Nakonec určete celkovou cenu vyhodnocení celého plánu. K dispozici máme systémovou paměť o velikosti  $M = 52$  bloků. Blokovací faktor pro atribut  $X$  je  $b\pi = 10$ . Postup komentujte.

## Příklad 4 (1 bod)

Navrhněte (opět formou syntaktického stromu jako výše), jak by bylo možné získat rychlejší plán vyhodnocení. Pro tento plán již nemusíte nic dalšího počítat.

## Část 2 - MongoDB (7 bodů) ✓

Byla zadána DB jako byla ve cvičeních tj

```
db.actors.save({ _id: "trojan", name: { first: "Ivan", last: "Trojan" },
year: 1964, movies: [ "samotari", "medvidek", "karamazovi" ] });
db.actors.save({ _id: "machacek", name: { first: "Jiri", last: "Machacek"
}, year: 1966, movies: [ "medvidek", "vratnelahve", "samotari" ] });
db.actors.save({ _id: "schneiderova", name: { first: "Jitka", last:
"Schneiderova" }, year: 1973, movies: [ "samotari" ] });
db.actors.save({ _id: "sverak", name: { first: "Zdenek", last: "Sverak"
}, year: 1936, movies: [ "vratnelahve" ] });
db.actors.save({ _id: "geislerova", name: { first: "Anna", last:
"Geislerova" }, year: 1976 });
db.actors.save({ _id: "vilhelмова", name: { first: "Tatiana", last:
"Vilhelмова" }, year: 1978, movies: [ "medvidek" ] });
db.actors.save({ _id: "menzel", name: { last: "Menzel", first: "Jiri" },
year: 1938, movies: "medvidek" });

db.movies.save({ _id: "samotari", title: { cs: "Samotari", en: "Loners"
}, year: 2000, rating: 84, actors: [ "trojan", "machacek", "schneiderova"
], genres: [ "comedy", "drama" ], country: [ "CZ", "SI" ], length: 103
});
db.movies.save({ _id: "medvidek", title: "Medvidek", year: 2007,
director: { first: "Jan", last: "Hrebejk" }, rating: 53, actors: [
"trojan", "machacek", "vilhelмова", "issova", "menzel" ], genres: [
"comedy", "drama" ], country: [ "CZ" ], length: 100 });
db.movies.save({ _id: "vratnelahve", title: { cs: "Vratne lahve", en:
"Empties" }, year: 2006, director: { first: "Jan", last: "Sverak" },
rating: 76, actors: [ "sverak", "machacek", "schneiderova" ], genres:
"comedy", country: "CZ", length: 99 });
db.movies.save({ _id: "zelary", title: "Zelary", year: 2003, director: {
last: "Trojan", first: "Ondrej" }, rating: 81, actors: [ ], genres: [
"romance", "drama" ], country: [ "CZ", "SK", "AT" ], length: 142 });
db.movies.save({ _id: "stesti", title: "Stesti", year: 2005, director: {
last: "Slama", first: "Bohdan" }, rating: 72, length: 100, awards: [ {
type: "Czech Lion", year: 2005 } ] });
```

```
db.movies.save({ _id: "kolja", title: "Kolja", year: 1996, rating: 86,
length: 105, awards: [ { type: "Czech Lion", year: 1996 }, { type:
"Noname Awards", category: "A", year: 2005 } ] });
```

## Příklad 1 (2 body) ✓

Herci se jménem Anna nebo Jiří.

### Řešení

Řešení podle Morčína:

```
db.actors.find({
  "$or": [
    {
      "name.first": "Anna"
    },
    {
      "name.first": "Jiri"
    }
  ]
})
```

## Příklad 2 (2 body) ✓

Filmy, které mají hodnocení horší než 90 a patří do žánru romance. Vypište pouze názvy filmů (title) a id.

### Řešení

Řešení podle Morčína:

```
db.movies.find({
  "rating": {
    "$lt": 90
  },
  "genres": {
    "$in": ["romance"]
  }
})
```

## Příklad 3 (3 body) ✓

Herci, kteří hrají ve filmech medvídek a samotáři. Seřadte výstup vzestupně dle příjmení herce.

## Řešení

Řešení podle Morčina:

```
db.actors.find({
  movies: {
    $all: ["samotari", "medvidek"]
  }
}).sort({"name.last": 1})
```

## Část 3 (8 bodů)

Graf letišť, jako ze cvičení:

```
CREATE
(sf {name:'San Francisco', code:'sf'}),
(la {name:'Los Angeles', code:'la'}),
(da {name:'Dallas', code:'da'}),
(ch {name:'Chicago', code:'ch'}),
(ny {name:'New York', code:'ny'}),
(sf)-[:DIRECT {price:50}]->(la),
(la)-[:DIRECT {price:50}]->(sf),
(sf)-[:DIRECT {price:250}]->(ch),
(ch)-[:DIRECT {price:250}]->(sf),
(da)-[:DIRECT {price:300}]->(sf),
(sf)-[:DIRECT {price:300}]->(da),
(ch)-[:DIRECT {price:100}]->(da),
(da)-[:DIRECT {price:100}]->(ch),
(ch)-[:DIRECT {price:250}]->(ny),
(ny)-[:DIRECT {price:250}]->(ch),
(ny)-[:DIRECT {price:225}]->(da),
(da)-[:DIRECT {price:225}]->(ny),
(da)-[:DIRECT {price:200}]->(la),
(la)-[:DIRECT {price:200}]->(da);
```

### Příklad 1 (2 body) ✓

Vypište trojice <start, cena, cíl> pro přímé lety dražší než 200, výstup seřadíte abecedně dle startu a ceny.

## Řešení

Řešení podle Morčina:

```
MATCH (s)-[r:DIRECT]->(d)
WHERE r.price > 200
RETURN s.name AS start, r.price AS cena, d.name AS cil
ORDER BY start, cena
```

## Příklad 2 (3 body) ✓

Vypište letiště dosažitelná z Los Angeles pomocí právě dvou přestupů.

Řešení

Řešení podle Morčina:

```
MATCH (s {name: "Los Angeles"})-[:DIRECT *3]->(d)
RETURN DISTINCT d
```

## Příklad 3 (3 body) ✓

Vypište cesty (kódy letišť oddělené čárkou) začínající v Dallasu do maximální délky 5.

Řešení

Řešení podle Morčina:

```
MATCH path=(s {name: "Dallas"})-[:DIRECT *..5]->(d)
RETURN REDUCE (res = "", x IN nodes(path) | res + CASE WHEN res = "" THEN
"" ELSE "-" END + x.code) AS cesta
```

## Část 4 (10 bodů)

Zde bylo zadané XML s pacienty něco ve stylu:

```
<pacienti>
  <pacient jmeno="Pepa" prijmeni="Vomacka">
    <navsteva datum="2019-01-10" diagnoza="Myxomatoza"/>
    <navsteva datum="2020-01-13" diagnoza="Simuluje"/>
    <navsteva datum="2021-02-20" diagnoza="Simuluje"/>
  </patient>
  <pacient jmeno="Jan" prijmeni="Skocdopole">
    <navsteva datum="2019-02-20" diagnoza="Triska v levem uchu"/>
    <navsteva datum="2020-01-13" diagnoza="Simuluje"/>
    <navsteva datum="2020-12-20" diagnoza="Zlomeny nos"/>
  </patient>
</pacienti>
```



```
</pacient>
</pacienti>
```

## Příklad 1 (4 body)

Napište dotaz, který vybere jméno pacienta s největším počtem návštěv kde diagnóza je „Simuluje“ (Výsledkem bude „Pepa“).

## Příklad 2 (4 bodů)

Napište dotaz, který pro každé datum (tedy hodnoty atributu datum v elementu navsteva) vypíše seznam pacientů (jméno) kteří byli na návštěvě Příklad:

```
<zaznam>
  <datum>2020-01-13</datum>
  <pacient jmeno="Pepa"/>
  <pacient jmeno="Jan"/>
</zaznam>
```

## Verze D

### Část 1 (7 bodů)

Mějme relace  $R(A, B, C)$  a  $S(D, E)$ . Předpokládejme, že  $n_R = 1\,000$  (počet záznamů),  $b_R = 5$  (blokovací faktor),  $n_S = 1\,000$  a  $b_S = 20$ . Atribut  $A$  nabývá v  $R.A = 10$  různých hodnot (každá z nich je stejně pravděpodobná). Primární soubor relace  $R$  je setříděný podle  $A$ .

### Příklad 1 (1 bod)

Dopočítejte statistiky  $P_r$  a  $P_s$

### Příklad 2 (2 body)

Vytvořte prováděcí plán ve formě stromu pro vyhodnocení dotazu (byl zadáný v algebře i SQL)

$(R(A = 3) \times S)[B, D]$

**select B,D from (select \* from R where A=3) cross join S;**

Byla tam ještě poznámka, že máme brát v potaz uzavorkování výrazu v algebře

### Příklad 3 (3 body)

Uvažujte plán vyhodnocení, ve kterém se spojení realizuje pomocí algoritmu Nested Loops. Pro mezivýsledek každé dílčí operace určete počty záznamů, blokovací faktor a počet bloků. Nakonec určete celkovou cenu vyhodnocení celého plánu. K dispozici máme systémovou paměť o velikosti  $M = 12$  bloků. Blokovací faktor pro dvojici atributů B a D je  $b_{\pi} = 100$ . Postup komentujte.

## Příklad 4 (1 bod)

Navrhnete (opět formou syntaktického stromu jako výše), jak by bylo možné získat rychlejší plán vyhodnocení. Pro tento plán již nemusíte nic dalšího počítat.

## Část 2 - MongoDB (7 bodů) ✓

Byla zadána DB jako byla ve cvičeních tj

```
db.actors.save({ _id: "trojan", name: { first: "Ivan", last: "Trojan" },
year: 1964, movies: [ "samotari", "medvidek", "karamazovi" ] });
db.actors.save({ _id: "machacek", name: { first: "Jiri", last: "Machacek"
}, year: 1966, movies: [ "medvidek", "vratnelahve", "samotari" ] });
db.actors.save({ _id: "schneiderova", name: { first: "Jitka", last:
"Schneiderova" }, year: 1973, movies: [ "samotari" ] });
db.actors.save({ _id: "sverak", name: { first: "Zdenek", last: "Sverak"
}, year: 1936, movies: [ "vratnelahve" ] });
db.actors.save({ _id: "geislerova", name: { first: "Anna", last:
"Geislerova" }, year: 1976 });
db.actors.save({ _id: "vilhelmoval", name: { first: "Tatiana", last:
"Vilhelmoval", year: 1978, movies: [ "medvidek" ] });
db.actors.save({ _id: "menzel", name: { last: "Menzel", first: "Jiri" },
year: 1938, movies: "medvidek" });

db.movies.save({ _id: "samotari", title: { cs: "Samotari", en: "Loners"
}, year: 2000, rating: 84, actors: [ "trojan", "machacek", "schneiderova"
], genres: [ "comedy", "drama" ], country: [ "CZ", "SI" ], length: 103
});
db.movies.save({ _id: "medvidek", title: "Medvidek", year: 2007,
director: { first: "Jan", last: "Hrebejk" }, rating: 53, actors: [
"trojan", "machacek", "vilhelmoval", "issoval", "menzel" ], genres: [
"comedy", "drama" ], country: [ "CZ" ], length: 100 });
db.movies.save({ _id: "vratnelahve", title: { cs: "Vratne lahve", en:
"Empties" }, year: 2006, director: { first: "Jan", last: "Sverak" },
rating: 76, actors: [ "sverak", "machacek", "schneiderova" ], genres:
"comedy", country: "CZ", length: 99 });
db.movies.save({ _id: "zelary", title: "Zelary", year: 2003, director: {
last: "Trojan", first: "Ondrej" }, rating: 81, actors: [ ], genres: [
"romance", "drama" ], country: [ "CZ", "SK", "AT" ], length: 142 });
```

```
db.movies.save({ _id: "stesti", title: "Stesti", year: 2005, director: {
  last: "Slama", first: "Bohdan" }, rating: 72, length: 100, awards: [ {
  type: "Czech Lion", year: 2005 } ] });
db.movies.save({ _id: "kolja", title: "Kolja", year: 1996, rating: 86,
  length: 105, awards: [ { type: "Czech Lion", year: 1996 }, { type:
  "Noname Awards", category: "A", year: 2005 } ] });
```

## Příklad 1 (2 body) ✓

Herci narození v roce 1964 nebo 1966

### Řešení

Řešení podle Morčína:

```
db.actors.find({
  $or: [
    {
      year: 1964
    },
    {
      year: 1966
    }
  ]
})
```

## Příklad 2 (2 body) ✓

Filmy, které mají hodnocení lepší než 70 a patří do žánru drama. Vypište pouze názvy filmů (title) a id.

### Řešení

Řešení podle Morčína:

```
db.movies.find({
  rating: {
    "$gt": 70
  },
  genres: {
    "$in": [
      "drama"
    ]
  }
})
```

```
{
  _id: 1,
  title: 1
})
```

### Příklad 3 (3 body) ✓

Herci, kteří hrají ve filmech **samotáři** a **vratnelahve**. Seřadte výstup vzestupně dle příjmení herce.

### Řešení

Řešení podle Morčina:

```
db.actors.find({
  movies: {
    $all: [
      "samotari",
      "vratnelahve"
    ]
  }
}).sort({"name.last": 1})
```

### Část 3 (8 bodů)

Graf letišť, jako ze cvičení:

```
CREATE
(sf {name:'San Francisco', code:'sf'}),
(la {name:'Los Angeles', code:'la'}),
(da {name:'Dallas', code:'da'}),
(ch {name:'Chicago', code:'ch'}),
(ny {name:'New York', code:'ny'}),
(sf)-[:DIRECT {price:50}]->(la),
(la)-[:DIRECT {price:50}]->(sf),
(sf)-[:DIRECT {price:250}]->(ch),
(ch)-[:DIRECT {price:250}]->(sf),
(da)-[:DIRECT {price:300}]->(sf),
(sf)-[:DIRECT {price:300}]->(da),
(ch)-[:DIRECT {price:100}]->(da),
(da)-[:DIRECT {price:100}]->(ch),
(ch)-[:DIRECT {price:250}]->(ny),
(ny)-[:DIRECT {price:250}]->(ch),
(ny)-[:DIRECT {price:225}]->(da),
(da)-[:DIRECT {price:225}]->(ny),
```

```
(da)-[:DIRECT {price:200}]->(la),  
(la)-[:DIRECT {price:200}]->(da);
```

## Příklad 1 (2 body) ✓

Vypište trojice <start, cena, cíl> pro přímé lety levnější než 200, výstup seřadíte abecedně dle startu a ceny.

### Řešení

Řešení podle Morčina:

```
MATCH (s)-[r:DIRECT]->(d)  
WHERE r.price < 200  
RETURN s.name as start, r.price as cena, d.name as cil  
ORDER BY start, cena
```

## Příklad 2 (3 body) ✓

Vypište letiště dosažitelná ze New Yorku pomocí právě dvou přestupů.

### Řešení

Řešení podle Morčina:

```
MATCH (s {name: "New York"})-[:DIRECT *3]->(d)  
RETURN DISTINCT d
```

## Příklad 3 (3 body) ✓

Vypište cesty (kódy letišť oddělené čárkou) začínající v Los Angeles do maximální délky 3.

### Řešení

Řešení podle Morčina:

```
MATCH path=(s {name: "Los Angeles"})-[:DIRECT *..3]->(d)  
RETURN REDUCE (res = "", x IN nodes(path) | res + CASE WHEN res = "" THEN  
"" ELSE "-" END + x.code) AS cesta
```

## Část 4 (10 bodů)

Zde bylo zadáno XML s pacienty něco ve stylu:

```

<objednavky>
  <objednavka datum="2020-01-13">
    <zakaznik jmeno="Pepa Vomacka"/>
    <polozka nazev="kleste" cena="150"/>
    <polozka nazev="stipacky" cena="200"/>
    <polozka nazev="vrtacka" cena="1800"/>
  </objednavka>
  <objednavka datum="2020-04-10">
    <zakaznik jmeno="Jan Skocdopole"/>
    <polozka nazev="sbijacka" cena="40"/>
    <polozka nazev="stipacky" cena="200"/>
    <polozka nazev="stipacky" cena="200"/>
  </objednavka>
</objednavky>

```

## Příklad 1 (4 body)

Napište dotaz, který najde všechny objednávky, na nichž se vyskytují štípačky minimálně 2x a jejich celková cena ( v rámci objednávky) je minimálně 250.

## Příklad 2 (4 bodů)

Napište dotaz, který pro každého zákazníka spočítá součet cen všech položek na všech jeho objednávkách Příklad:

```

<soucet>
  <jmeno>Pepa Vomacka</jmeno>
  <cena>2150</cena>
</soucet>

```

# Verze E

## Část 1 (15 bodů)

Rovnaké ako [1. úloha minulý rok](#), rozdiel akurát v zadaných hodnotách Br a Bs.

Ve verzi D zadaný příklad následovně: Mějme relace  $R[A, B]$  a  $S[X, Y, Z]$ .

Předpokládejme, že  $N_r = 2000$  (počet záznamů),  $B_r = 10$  (blokový faktor),  $N_s = 1000$  a  $B_s = 10$ . Atribut A nabývá  $VR.A = 50$  různých hodnot (každá z nich je stejně pravděpodobná). Navíc máme k dispozici clusterovaný index pro atribut A v relaci R. Jeho hloubka je  $iR.A = 2$  a listová úroveň obsahuje pR.A 50 bloků. Uvažujte plán vyhodnocení, ve kterém spojení realizuje pomocí algoritmu Nested Loops. Pro mezivýsledek každé dílčí operace určete počty záznamů, blok. faktor a počet bloků.

Nakonec určete elkovou cenu vyhodnocení celého plánu. K dispozici máme systémovou paměť o velikosti  $M = 22$  bloků. Postup komentujte.

Dopočtěte statistiky: Pr, Ps

## Řešení

$Pr = Nr/Br = 2000/10 = 200$

$Ps = Ns/Bs = 1000/10 = 100$

## Část 2 - MongoDB (7 bodů) ✓

Byla zadána DB jako byla ve cvičeních tj

```
db.actors.save({ _id: "trojan", name: { first: "Ivan", last: "Trojan" },
year: 1964, movies: [ "samotari", "medvidek", "karamazovi" ] });
db.actors.save({ _id: "machacek", name: { first: "Jiri", last: "Machacek"
}, year: 1966, movies: [ "medvidek", "vratnelahve", "samotari" ] });
db.actors.save({ _id: "schneiderova", name: { first: "Jitka", last:
"Schneiderova" }, year: 1973, movies: [ "samotari" ] });
db.actors.save({ _id: "sverak", name: { first: "Zdenek", last: "Sverak"
}, year: 1936, movies: [ "vratnelahve" ] });
db.actors.save({ _id: "geislerova", name: { first: "Anna", last:
"Geislerova" }, year: 1976 });
db.actors.save({ _id: "vilhelmoval", name: { first: "Tatiana", last:
"Vilhelmoval" }, year: 1978, movies: [ "medvidek" ] });
db.actors.save({ _id: "menzel", name: { last: "Menzel", first: "Jiri" },
year: 1938, movies: "medvidek" });

db.movies.save({ _id: "samotari", title: { cs: "Samotari", en: "Loners"
}, year: 2000, rating: 84, actors: [ "trojan", "machacek", "schneiderova"
], genres: [ "comedy", "drama" ], country: [ "CZ", "SI" ], length: 103
});
db.movies.save({ _id: "medvidek", title: "Medvidek", year: 2007,
director: { first: "Jan", last: "Hrebejk" }, rating: 53, actors: [
"trojan", "machacek", "vilhelmoval", "issoval", "menzel" ], genres: [
"comedy", "drama" ], country: [ "CZ" ], length: 100 });
db.movies.save({ _id: "vratnelahve", title: { cs: "Vratne lahve", en:
"Empties" }, year: 2006, director: { first: "Jan", last: "Sverak" },
rating: 76, actors: [ "sverak", "machacek", "schneiderova" ], genres:
"comedy", country: "CZ", length: 99 });
db.movies.save({ _id: "zelary", title: "Zelary", year: 2003, director: {
last: "Trojan", first: "Ondrej" }, rating: 81, actors: [ ], genres: [
"romance", "drama" ], country: [ "CZ", "SK", "AT" ], length: 142 });
db.movies.save({ _id: "stesti", title: "Stesti", year: 2005, director: {
last: "Slama", first: "Bohdan" }, rating: 72, length: 100, awards: [ {
```

```
type: "Czech Lion", year: 2005 } ] });  
db.movies.save({ _id: "kolja", title: "Kolja", year: 1996, rating: 86,  
length: 105, awards: [ { type: "Czech Lion", year: 1996 }, { type:  
"Noname Awards", category: "A", year: 2005 } ] });
```

Upravit

## Příklad 1 (2 body) ✓

Vyberte herce s křesním jménem Jiri nebo Anna

### Řešení

Řešení podle Morčína:

```
db.actors.find({  
  "$or": [  
    {  
      "name.first": "Anna"  
    },  
    {  
      "name.first": "Jiri"  
    }  
  ]  
})
```

## Příklad 2 (2 body) ✓

Vypište herce co hráli zároveň ve filmech samotari a medvidek, a vypište jen jejich id a křestní jméno

### Řešení

Řešení podle Morčína:

```
db.actors.find({  
  movies: {  
    $all: [  
      "samotari",  
      "medvidek"  
    ]  
  }  
},  
{  
  _id: 1,
```



```
"name.first": 1
})
```

### Příklad 3 (3 body) ✓

Vypište filmy, které mají žánr romance a hodnocení menší než 90.

#### Řešení

Řešení podle Morčina:

```
db.movies.find({
  rating: {
    "$lt": 90
  },
  genres: {
    "$in": [
      "romance"
    ]
  }
})
```

### Část 3 (8 bodů)

Byl zadáný celkem jednoduchý graf jen s jedním typem uzlů a hran a to: Města a přímé cesty mezi nimi. Hrany byly ohodnocené cenou spoje. Uzel Města obsahoval vlastnosti name a code.

```
(sf {name:'San Francisco', code:'sf'}),
(la {name:'Los Angeles', code:'la'}),
(da {name:'Dallas', code:'da'}),
(ch {name:'Chicago', code:'ch'}),
(ny {name:'New York', code:'ny'}),
(sf)-[:DIRECT {price:50}]->(la),
(la)-[:DIRECT {price:50}]->(sf),
(sf)-[:DIRECT {price:250}]->(ch),
(ch)-[:DIRECT {price:250}]->(sf),
(da)-[:DIRECT {price:300}]->(sf),
(sf)-[:DIRECT {price:300}]->(da),
(ch)-[:DIRECT {price:100}]->(da),
(da)-[:DIRECT {price:100}]->(ch),
(ch)-[:DIRECT {price:250}]->(ny),
(ny)-[:DIRECT {price:250}]->(ch),
(ny)-[:DIRECT {price:225}]->(da),
(da)-[:DIRECT {price:225}]->(ny),
```

```
(da)-[:DIRECT {price:200}]->(la),  
(la)-[:DIRECT {price:200}]->(da);
```

## Příklad 1 (2 body) ✓

Vypište všechny trojice <start, cíl, cena> pro přímé cesty.

### Řešení

Řešení podle Morčína:

```
MATCH (s {name: "Los Angeles"})-[:DIRECT]->(d)  
RETURN s.name, d.name, r.price
```

## Příklad 2 (3 body) ✓

Vypište všechna města kam se dá dostat z Los Angeles s maximálně 1 přestupem

### Řešení

Řešení podle Morčína:

```
MATCH (s {name: "Los Angeles"})-[:DIRECT *..2]->(d)  
RETURN DISTINCT d
```

## Příklad 3 (3 body) ✓

Vypsát čárkou oddělený seznam kódů měst kam se dá dostat z Los Angeles s maximálně 4 přestupy.

### Řešení

Řešení podle Morčína:

```
MATCH path=(s {name: "Los Angeles"})-[:DIRECT *..5]->(d)  
RETURN REDUCE (res = "", x IN nodes(path) | res + CASE WHEN res = "" THEN  
"" ELSE "-" END + x.code) AS cesta
```

## Část 4 (10 bodů)

Zde bylo zadáno XML něco ve stylu:

```
<body>  
  <div id="book132">
```

```

    <h1>Název knihy 1</h1>
    <div>
      
    </div>
  </div>
  <div id="book221">
    <h1>Název knihy 2</h1>
    <div>
      
    </div>
  </div>
  <div id="book114">
    <h1>Název knihy 3</h1>
    <div>
      <div></div>
    </div>
  </div>
</body>

```

## Příklad 1 (5 bodů)

Vypište ID divů, které obsahují obrázek, který končí na jpg.

```
//div[ends-with(./img/@src, "jpg")]/@id
```

## Příklad 2 (5 bodů)

Vypište abecedně seřazené knihy podle jejího názvu (element h1) a počet obrázků, které kniha obsahuje (v libovolně vnořené úrovni). Výpis proveďte v následující struktuře

```

<element>
  <book>název knihy</book>
  <images>počet obrázků</images>
</element>

```

## Řešení

```

let $books := //div[boolean(./h1)]

for $book in $books
  let $name := $book/h1/text()
  order by $name
  return
    <element>

```

```
<book>{$name}</book>
<images>{count($book//img)}</images>
</element>
```

## Verze F (mix více testů)

Ostatní části byly namixovné z ostatních verzí testu.

### Část 1 - SQL optimalizace

Mějme relace Student(*sid*, jmeno, vek), Hra(*hid*, nazev, zanr) a Hraje(*sid*, *hid*, *datum*), kde Hraje.sid je cizím klíčem do Student.sid a Hraje.hid je cizím klíčem do Hra.hid.

Předpokládejme, že nStudent = 2 000 (počet záznamů), bStudent = 20 (blokovací faktor), nHra = 1 000 a bHra = 5. Atribut zanr nabývá vHra.zanr = 10 různých hodnot (každá z nich je stejně pravděpodobná). Primární soubor relace Hra je seříděný podle zanru.

#### Příklad 1 (1 bod)

Spočtete další potřebné charakteristiky: pStudent a pHra

#### Příklad 2 (2 body)

Vytvořte prováděcí plán ve formě stromu pro vyhodnocení dotazu zapsaného v relační algebře: (Hra(Zanr='FPS') x Student). Jedná se o přepis SQL dotazu:

```
select * from (select * from Hra where zanr='FPS') cross join Student;
```

Respektuje pořadí vyhodnocení operací dané uzávorkováním výrazu v relační algebře.

#### Příklad 3 (3 body)

Uvažujte plán vyhodnocení, ve kterém se spojení realizuje pomocí algoritmu Nested Loops. Pro mezivýsledek každé dílčí operace určete počty záznamů, blokovací faktor a počet bloků. Nakonec určete celkovou cenu vyhodnocení celého plánu. K dispozici máme systémovou paměť o velikosti M = 12 bloků. Postup komentujte.

#### Příklad 4 (1 bod)

Navrhněte (opět formou syntaktického stromu jako výše), jak by bylo možné získat rychlejší ohodnocení. Pro tento plán již nemusíte nic dalšího počítat.

### Část 2 - MongoDB (7 bodů) ✓

## Struktura mongo DB

```
[
  {
    "address": {
      "building": "1007",
      "coord": [ -73.856077, 40.848447 ],
      "street": "Morris Park Ave",
      "zipcode": "10462"
    },
    "borough": "Bronx",
    "cuisine": "Bakery",
    "grades": [
      { "date": { "day": 2022-11-09 }, "grade": "A", "score": 2 },
      { "date": { "day": 2022-11-09 }, "grade": "A", "score": 6 },
      { "date": { "day": 2022-11-09 }, "grade": "A", "score": 10 },
      { "date": { "day": 2022-11-09 }, "grade": "A", "score": 9 },
      { "date": { "day": 2022-11-09 }, "grade": "B", "score": 14 }
    ],
    "name": "Morris Park Bake Shop",
    "restaurant_id": "30075445"
  },
  {
    "address": {
      "building": "23-A",
      "coord": [ -73.851066, 40.739583 ],
      "street": "Harris Lord Kekw",
      "zipcode": "10465"
    },
    "borough": "Manhattan",
    "cuisine": "Sushi",
    "grades": [
      { "date": { "day": 2020-11-23 }, "grade": "C", "score": 3 },
      { "date": { "day": 2021-02-21 }, "grade": "B", "score": 6 },
      { "date": { "day": 2021-09-19 }, "grade": "C", "score": 4 },
      { "date": { "day": 2022-04-17 }, "grade": "B", "score": 7 },
      { "date": { "day": 2022-11-09 }, "grade": "A", "score": 13 }
    ],
    "name": "Heaven Sushi",
    "restaurant_id": "30075487"
  }
]
```

### Příklad 1 (2 body) ✓

Restaurace z Manhattanu, hodnocené po roce 2021.

## Řešení

Řešení podle Morčina:

```
db.restaurants.find({
  "borough": "Manhattan",
  "grades": {
    $elemMatch: {
      "date.day": {
        "$gt": "2021"
      }
    }
  }
})
```

## Příklad 2 (2 body) ✓

Restaurace z Bronxu, jejichž název končí na „Shop“, vypsát pouze názvy restaurací.

## Řešení

Řešení podle Morčina:

```
db.restaurants.find({
  borough: "Bronx",
  name: {
    $regex: "Shop$"
  }
})
```

## Příklad 3 (3 body) ✓

Skóre restaurací, které byly hodnocené jen v roce 2022 a dostaly známku B, vypsát oddělené čárkou a sestupně.

## Řešení

Řešení podle Morčina:

```
db.restaurants.find({
  "grades": {
    $elemMatch: {
      "date.day": {
        $regex: "^2022"
      },

```

```
    "grade": "B"
  }
}
}).sort({ "name": -1 });
```

## Část 2b - MongoDB (7 bodů) ✓

```
db.peaks.insertMany ([
{
  "name":"Everest",
  "height":8848,
  "location":["Nepal","China"],
  "ascents":{"first":{"year":1953}, "first_winter":{"year":1980},
"total":5656 }
},
{
  "name":"K2",
  "height":8611,
  "location":["Pakistan","China"],
  "ascents":{"first":{"year":1954}, "first_winter":{"year":1921},
"total":306 }
},
{
  "name":"Kangchenjunga",
  "height":8586,
  "location":["Nepal","India"],
  "ascents":{"first":{"year":1955}, "first_winter":{"year":1986},
"total":283 }
},
{
  "name":"Lhotse",
  "height":8516,
  "location":["Nepal","China"],
  "ascents":{"first":{"year":1956}, "first_winter":{"year":1988},
"total":461 }
},
{
  "name":"Makalu",
  "height":8485,
  "location":["China","Nepal"],
  "ascents":{"first":{"year":1955}, "first_winter":{"year":2009},
"total":361 }
}
])
```

### Příklad 1 (2 body) ✓

Vrcholy, které leží v Pakistanu a jsou nižší než 8000 metrů.

#### Řešení

Řešení podle Morčína:

```
db.peaks.find({
  height: {
    "$lt": 8000
  },
  location: {
    "$in": [
      "Pakistan"
    ]
  }
})
```

### Příklad 2 (2 body) ✓

Vrcholy, které se nacházejí buď v Indii nebo Nepálu a byly vystoupány více než 1000 krát. Vypíšte pouze název a výšku vrcholu.

#### Řešení

Řešení podle Morčína:

```
db.peaks.find({
  "ascents.total": {
    "$gt": 1000
  },
  location: {
    "$in": [
      "India",
      "Nepal"
    ]
  }
})
```

### Příklad 3 (3 body) ✓

Vrcholy, které se *nenacházejí* v Chině. Seřadte výstup abecedně podle názvu.

#### Řešení



Řešení podle Morčina:

```
db.peaks.find({
  location: {
    "$nin": [
      "China"
    ]
  }
}).sort({"name": 1})
```

## Část 3 - Neo4j (8 bodů) ✓

```
CREATE
(philip:Person {name:"Philip"})-[:IS_FRIEND_OF]->(emil:Person {name:"Emil"}),
(philip)-[:IS_FRIEND_OF]->(michael:Person {name:"Michael"}),
(philip)-[:IS_FRIEND_OF]->(andreas:Person {name:"Andreas"});

CREATE
(sushi:Cuisine {name:"Sushi"}),
(nyc:City {name:"New York"}),
(iSushi:Restaurant {name:"iSushi"})-[:SERVES]->(sushi),
(iSushi)-[:LOCATED_IN]->(nyc),
(michael)-[:LIKES]->(iSushi),
(andreas)-[:LIKES]->(iSushi),
(zam:Restaurant {name:"Zushi Zam"})-[:SERVES]->(sushi),
(zam)-[:LOCATED_IN]->(nyc),
(andreas)-[:LIKES]->(zam);
```

### Příklad 1 (2 body) ✓

Vypište názvy všech restaurací, výstup seřadíte abecedně.

### Řešení

Řešení podle Morčina:

```
MATCH (r:Restaurant)
RETURN DISTINCT r.name as restaurant
ORDER BY restaurant
```

### Příklad 2 (2 body) ✓

Vypište pratele Emila a pratele jeho pratel do maximalní délky 3.

## Řešení

Řešení podle Morčina:

```
MATCH (e:Person {name: "Emil"})<-[ :IS_FRIEND_OF *1..3]->(f:Person)
WHERE e <> f
RETURN DISTINCT f.name
```

## Příklad 3 (4 body) ✓

Najděte restaurace v Londýně, které servírují pizzu a jsou oblíbené Emilovými přáteli.  
(Ano, pizza ani Londýn v databázi nejsou)

## Řešení

Řešení podle Morčina:

```
MATCH (r:Restaurant)-[:LOCATED_IN]->(l:City {name: "London"})
MATCH (e:Person {name: "Emil"})<-[ :IS_FRIEND_OF]->(p:Person)-[:LIKES]->
(r)-[:SERVES]->(c:Cuisine {name:"Pizza"})
RETURN r.name
```

## Část 4 - XQuery

```
<contact-info>
  <contact1>
    <name first="Tamara" last="Patrick"/>
    <company>Marvel</company>
    <phone>123456789</phone>
    <address street="Linden ave." city="London"/>
    
    <comment>Met at the conference.</comment>
  </contact1>
  <contact2>
    <name first="John" last="Black"/>
    <company>DC</company>
    <phone>987654321</phone>
    <address street="Bowl blv." city="Las Vegas"/>
    
    
    <comment>Studied at high school.</comment>
  </contact2>
</contact-info>
```

## Příklad 1 (4 body)

Napište dotaz, který vybere všechny společnosti, které se vyskytují v databázi.

## Příklad 2 (4 body)

Napište dotaz, který seřídí kontakty podle obsahu elementu comment a na výstup vypíše jméno kontaktu a počet přiřazených obrázků (img).

Příklad:

```
<contact>
<name>Tamara Patrick</name>
<img>1</img>
</contact>
```