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

8. Dokumenty

verze z 6. listopadu 2023

1 Dokumentový model databáze

Data reprezentujeme pomocí *stromové datové struktury*. Přesněji data chápeme jako *strom*, kde uzly jsou trojího typu: atomické, pole a dokumenty. Uzlům atomického typu je přiřazena hodnota a jsou často ztotožňovány se svou hodnotou. Pouze uzly typu pole nebo dokument mohou mít následníky a ti jsou lineárně uspořádáni. Následníci uzlu typu dokument mají jednoznačně přiřazené *názvy*. Následníci uzlu typu pole se nazývají *prvky* pole. Počet prvků v poli nazýváme *délkou* pole. Nezáporné celé číslo i menší jak délka pole určuje i plus první prvek pole a nazývá se *indexem* prvku. Následníci uzlu typu *dokument* se nazývají *položky* dokumentu. Strom, kde kořen je uzel typu pole, nazýváme *polem* a typu dokument nazýváme *dokument*.

Stromy můžeme vytvořit pomocí následujících tří pravidel.

1. Libovolná atomická hodnota (číslo, řetězec, logická hodnota, ...) je považována za strom s jediným uzlem který má danou hodnotu.
2. Pokud $t_1, \ldots, t_n$ jsou stromy, tak i uzel typu pole, který má za následníky kořeny stromů $t_1, \ldots, t_n$, určuje strom.
3. Pokud $t_1, \ldots, t_n$ jsou stromy a $s_1, \ldots, s_n$ po dvou různé názvy, tak i uzel typu dokument, který má za následníky kořeny stromů $t_1, \ldots, t_n$ označené postupně jmény $s_1, \ldots, s_n$, určuje strom.

Uzel typu pole označíme `[]` a uzel typu dokument `{}`. Například uvažujme dokument D:

- `{}`
  - `_id`: 1
  - `title`: `The Godfather`
  - `year`: 1972
  - `actors`: `[]`
    - `Gary Oldman`
    - `Winona Ryder`

* `Anthony Hopkins`

Například máme, že hodnota položky `_id` dokumentu $\mathcal{D}$ je číslo jedna, řetězec `"Gary Oldman"` je prvkem pole položky `actors` dokumentu $\mathcal{D}$, položka `actors` dokumentu $\mathcal{D}$ je pole délky tři a v poli je na indexu 1 řetězec `"Winona Ryder"`.

Položka dokumentu `_id` se nazývá *identifikátor* dokumentu. Názvy položek dokumentu píšeme anglicky malými písmeny. Slova spojujeme tak, že první písmeno následujícího slova změníme na velké. Tedy používáme velbloudí notaci. Například: `movieTitle`.

Kolekce je proměnná, jejíž hodnota je množina dokumentů. Každý dokument v kolekci musí mít identifikátor. Identifikátory dokumentů v kolekci musí být po dvou různé. Kolekce tvoří *databázi*.

2 JSON

JSON (JavaScript Object Notation)[1] je textový zápis stromových datových struktur.

Hodnoty v JSON dělíme na *atomické* a *složené*. Atomické hodnoty jsou:

1. čísla,
2. řetězce,
3. logické hodnoty,
4. `null` (prázdné místo).

Používají se čísla s plovoucí desetinou čárkou zapsané podle standardu IEEE 754[2], které můžou být zapsány za použití cifer. Můžeme tedy zapsat celá čísla (`1939`), desetinná čísla (`0.25`) a čísla za použití vědecké notace (`1e4` = 1000).

Řetězce vkládáme mezi dvojité horní uvozovky. Například: `""`, `"Dracula"`, `"Francis Ford Coppola"`. Znak pro nový řádek, dvojité uvozovky a zpětné lomítko vkládáme pomocí únikového znaku zpětné lomítko. Například řetězec `"\n\"\\"` je tvořen třemi znaky: znakem pro novým řádkem, znakem dvojité uvozovky a znakem zpětného lomítka.

Logické hodnoty jsou dvě: `true` (pravda) a `false` (nepravda).

Složené hodnoty dělíme na objekty a pole.

Objekt má tvar

`{` ***pairs*** `}`

kde ***pairs*** jsou čárkou oddělené páry (*položky*). *Pár* má tvar

[1] `https://www.json.org/json-cz.html`
[2] `https://en.wikipedia.org/wiki/IEEE_754`

name: *value*

kde ***name*** je řetězec (*název*) a ***value*** je libovolná hodnota. Názvy položek objektu musí být po dvou různé.

Příklady objektů:

- `{}` (prázdný objekt),
- `{ "title": "Dracula", "year": 1992 }`.

Položka objektu může opět být objektem:

```
{
  "title": "Dracula",
  "year": 1992,
  "director": {
    "name": "Francis Ford Coppola",
    "born": 1939
  }
}
```

Pole má tvar:

[items]

kde ***items*** jsou libovolné hodnoty (*prvky*) oddělené čárkou. Například: `[]` (prázdné pole), `[1972, 1974, 1990]`, `[true, false]`. Prvek pole opět může být polem: `[[1, 2], [3, 4]]`. Můžeme mít i pole objektů:

```
[
  {
    "title": "The Godfather",
    "year": 1972
  },
  {
    "title": "The Godfather: Part II",
    "year": 1974
  },
  {
    "title": "The Godfather: Part III",
    "year": 1990
  }
]
```

Položkou objektu může být pole:

```
{
  "title": "The Godfather",
  "year": 1972,
  "actors": [
    "Gary Oldman",
    "Winona Ryder",
    "Anthony Hopkins",
    "Keanu Reeves"
  ]
}
```

3 BSON

BSON (Binary JSON)[3] je binární zápis stromově uspořádaných dat. Formát vychází z JSON a přidává další typy hodnot (datum, binární data, ...).

Hodnoty v BSON lze reprezentovat v JSON tak, že hodnoty přidaných typů zapíšeme jako objekt s jediným prvkem jehož název je znak dolaru (`$`) následovaný názvem typu. Prvek je pak většinou řetězec obsahující textovou reprezentaci původní hodnoty. Například datum:

```
{ "$date": "2022-10-26T11:03:14.774Z" }
```

4 MongoDB

MongoDB[4] je databázový systém založený na dokumentech. Existuje volně dostupná komunitní verze.[5] Dokumentaci nelezete zde: `https://www.mongodb.com/docs/manual/`.

Server se spouští z příkazové řádky operačního systému příkazem:

```
% ./mongodb-macos-x86_64-7.0.2/bin/mongod --dbpath path
```

kde ***path*** je cesta k adresáři, ve kterém server uchovává data. Před prvním spuštěním vytvořte prázdný adresář a zadejte cestu k němu.

Pro práci se systémem lze použít desktopovou aplikaci Compass[6], příkazový řádek Shell [7] nebo rozšíření pro VS Code [8]. Aplikace Compass v sobě Shell zahrnuje.

[3] `https://bsonspec.org`
[4] `https://www.mongodb.com`
[5] `https://www.mongodb.com/try/download/community`
[6] `https://www.mongodb.com/products/compass`
[7] `https://www.mongodb.com/products/shell`
[8] `https://www.mongodb.com/products/vs-code`

Příkazy pro systém se zapisují v programovacím jazyce JavaScript.Znalost jazyka není vyžadována. V průběhu se seznámíme s potřebnými rysy. V mnohém je JavaScript podobný Pythonu.

Spusťte Shell buď příkazem operačního systému:

```
% ./mongosh-2.0.2-darwin-x64/bin/mongosh
```

nebo otevřením spodní částí (`>_MONGOSH`) aplikace Compass. Při otevření aplikace se nejprve připojte k systému. Pro připojení k lokálnímu systému stačí kliknout na tlačítko `Connect`.

Shell vás vyzve k zadání příkazu:

```
test>
```

Před znakem `>` se nachází název aktuálně používané databáze.

Skripty pro MongoDB jsou programy v JavaScriptu, které mají příponu `mongodb`.

5 Základy

Výraz jazyka je příkazem, který se vykoná tak, že se vyhodnotí výraz a vytiskne jeho hodnota.

Databázi lze změnit výrazem:

use(*dbname*)

kde *dbname* je řetězec udávající název databáze. Například:

```
test> use("moviedb")
switched to db moviedb
moviedb>
```

Z pohledu systému existují databáze všech možných jmen.

JavaScript zjednodušuje zadávání objektů tak, že pokud je název položky identifikátor JavaScriptu, pak nemusíme jméno uzavírat do uvozovek. Pokud je název tvořen písmeny anglické abecedy, podtržítky a znaky dolaru, pak se jedná o identifikátor. Například: `{ title: "Dracula", year: 1992 }` je objekt `{ "title": "Dracula", "year": 1992 }`. Dále JavaScript umožňuje uzavírat řetězce mezi jednoduché uvozovky: `'Dracula'`.

Výraz

db.*collection*.insertOne(*document*)

přidá dokument *document* do kolekce *collection*. Kolekci není potřeba vytvářet, všechny kolekce již existují a jsou prázdné (neobsahují žádný dokument).

Například:

```
moviedb> db.movies.insertOne({ _id: 1, title: "Dracula", year: 1992 })
{ acknowledged: true, insertedId: 1 }
```

Vidíme, že hodnotou výrazu je objekt.

K identifikaci dokumentů se používá typ ObjectId. ObjectId je 12 bajtové číslo. První čtyři bajty jsou časové razítko udávající počet sekund od začátku roku 1970. Dalších pět bajtů je číslo náhodně vybrané při spuštění serveru. Poslední tři bajty udávají hodnotu automaticky se zvětšujícího počítadla.

ObjectId lze zadat pomocí 24 číslic v šestnáctkové soustavě. Například:

$$618162f7c76a6312121d3af3.$$

Výraz:

ObjectId(*hexadecimal*)

kde *hexadecimal* je řetězec délky 24 obsahující znaky šestnáctkové soustavy, se vyhodnotí na ObjectId se zadanou hodnotou. Například:

```
ObjectId("618162f7c76a6312121d3af3")
```

Výraz `ObjectId()` vytvoří nový identifikátor. Například:

```
moviedb> ObjectId()
ObjectId("63591f74b5c9d364805b4ac0")
```

Výraz

function(*arg1*, *arg2*, ...)

zavolá funkci *function* s argumenty *arg1*, *arg2*, ... Hodnotou výrazu je návratová hodnota funkce.

Funkci `ObjectId` nazýváme konstruktor. Její zavolání vrací hodnotu zadaného typu. Pomocí konstruktorů zadáváme hodnoty typů, které nejsou v JSON formátu. Například 64-bitové číslo zadáme konstruktorem `Long`:

```
moviedb> Long("299792458")
Long("299792458")
```

Jak vidíme, konstruktory se používají i k tisku hodnot.

Pokud zadáme dokument bez identifikátoru, databáze vytvoří ObjectId a určí jej za identifikátor. Například:

```
moviedb> db.movies.insertOne({ title: "The Godfather", year: 1972 })
{
  acknowledged: true,
  insertedId: ObjectId("63591ff5b5c9d364805b4ac1")
}
```

Hodnotou výrazu:

db.*collection*.find()

je pole všech dokumentů v kolekci.

```
moviedb> db.movies.find()
[
  { _id: 1, title: 'Dracula', year: 1992 },
  {
    _id: ObjectId("63591ff5b5c9d364805b4ac1"),
    title: 'The Godfather',
    year: 1972
  }
]
```

Položky stejného názvu mohou mít různý typ. Například `_id` je v předchozích dokumentech jednou číslo a podruhé ObjectId.

Dokumenty nemusí mít stejnou sadu názvů položek. Například:

```
moviedb> db.movies.insertOne({ title: "The Matrix" })
{
  acknowledged: true,
  insertedId: ObjectId("6359249eb5c9d364805b4ac2")
}
```

nebo:

```
moviedb> db.movies.insertOne({
...   title: "Cloud Atlas",
...   directors: [ "Lana Wachowski", "Tom Tykwer", "Lilly Wachowski" ]
... })
{
  acknowledged: true,
  insertedId: ObjectId("635924b1b5c9d364805b4ac3")
}
```

Dokumenty kolekce:

```
moviedb> db.movies.find()
[
  { _id: 1, title: 'Dracula', year: 1992 },
  {
    _id: ObjectId("63591ff5b5c9d364805b4ac1"),
    title: 'The Godfather',
    year: 1972
```

```
  },
  { _id: ObjectId("6359249eb5c9d364805b4ac2"), title: 'The Matrix' },
  {
    _id: ObjectId("635924b1b5c9d364805b4ac3"),
    title: 'Cloud Atlas',
    directors: [ 'Lana Wachowski', 'Tom Tykwer', 'Lilly Wachowski' ]
  }
]
```

Výraz:

db.*collection*.insertMany(*documents*)

přidá do kolekce *collection* dokumenty v poli *documents*.

Například:

```
moviedb> db.movies.insertMany([
...   {
...     title: "The Conversation",
...     year: 1974
...   },
...   {
...     title: "Apocalypse Now",
...     year: 1979
...
...   }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("635927b8b5c9d364805b4ac4"),
    '1': ObjectId("635927b8b5c9d364805b4ac5")
  }
}
```

Následující výraz přidá dokument, který jako prvek obsahuje pole dokumentů.

```
db.movies.insertOne({
  title: "The Fifth Element",
  year: 1997,
  actors: [
    {
      name: "Bruce Willis",
      birth: 1955,
      favorite: true
    },
```

```
    {
      name: "Gary Oldman",
      birth: 1958
    },
    {
      name: "Milla Jovovich",
      gender: "female"
    }
  ]
})
```

Výraz:

db.*collection*.drop()

zruší kolekci *collection*. Kolekce se stane prázdnou.

Například:

```
moviedb> db.movies.drop()
true
```