



## Sprawozdanie z projektu nr 3

### Zaawansowane systemy baz danych

#### Wybór zbioru danych:

Do projektu wybrałem zbiór kilkudziesięciu tysięcy gier z oficjalnej platformy gier Steam: <https://www.kaggle.com/datasets/deepann/80000-steam-games-dataset>. Zestaw składa się z dokumentów jednego typu – gier, które posiadają m.in. odnośniki do grafiki, datę wydania, szczegółowe informacje o grze, tagi, kategorie oraz inne dodatkowe informacje.

Zbiór wydaje się wystarczająco rozbudowany, by spełnić wymagania projektu – posiada kilkanaście pól, w tym również zagnieżdżone. Co ważne zbiór stanowi jedną kolekcję – games, aczkolwiek dostrzegłem w nim potencjał, ponieważ zawiera w sobie dwa szczególne pola: *developer* oraz *publisher*. Moim zdaniem są to idealni kandydaci do stworzenia referencji i powiązania z innymi kolekcjami.

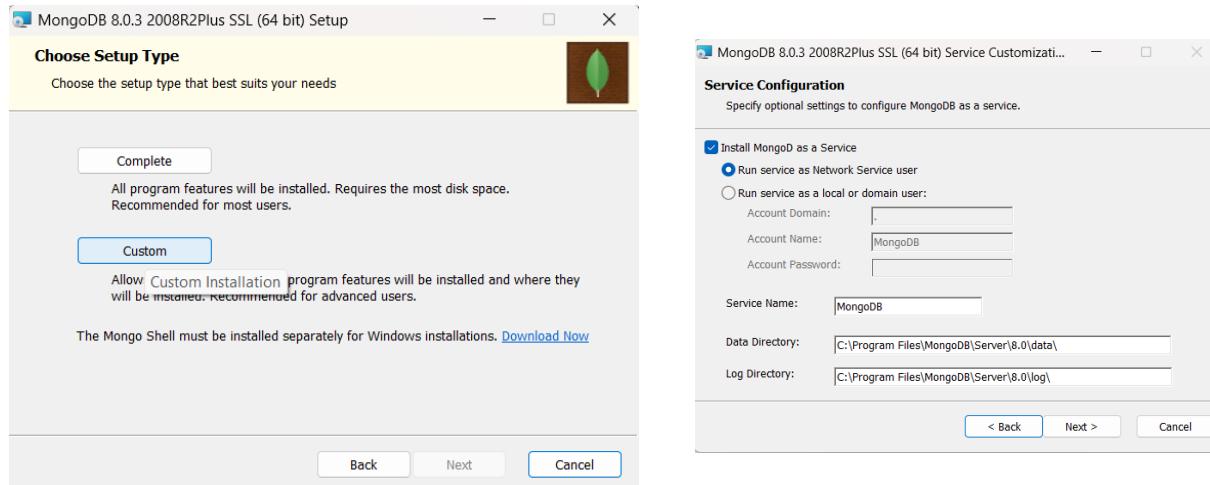
Za pomocą skryptu `parseDocumentsScript.py` obsłużę dane wejściowe z pliku `final_data_new.json` tworząc na wyjściu trzy pliki: `games.json` oraz wygenerowane sztucznie (tzn. wykorzystując bibliotekę faker do generacji mock'ów) dwie kolekcje w formie json'ów do plików: `publishers.json` oraz `developers.json`. Publishers oraz developers będą relatywnie prostymi kolekcjami zawierającymi kilka pól.

Bazowy zbiór danych z pobranego pliku `final_data_new.json` zawiera 45 000 dokumentów, co powinno być wystarczające do porównania działania wydajności indeksów. Ogromną zaletą zbioru jest fakt formatu danych w postaci JSONa – z użyciem mongoimport wczytanie danych z pliku jest banalnie proste i wymaga zaledwie jednego polecenia (o czym dalej w sekcji Import danych).

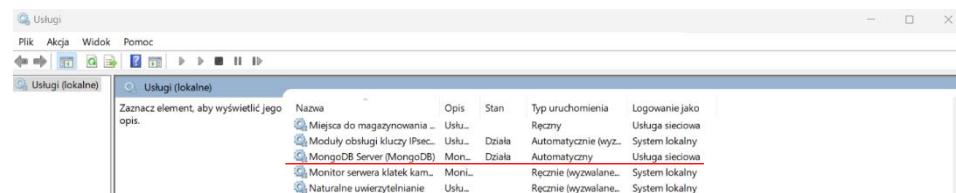
## Instalacja i konfiguracja:

Instalację serwera wykonałem krok po kroku zgodnie z oficjalną dokumentacją od MongoDB: <https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-windows/#std-label-install-mdb-community-windows>

W momencie wykonywania projektu najnowsza stabilna wersja to 8.0.3.

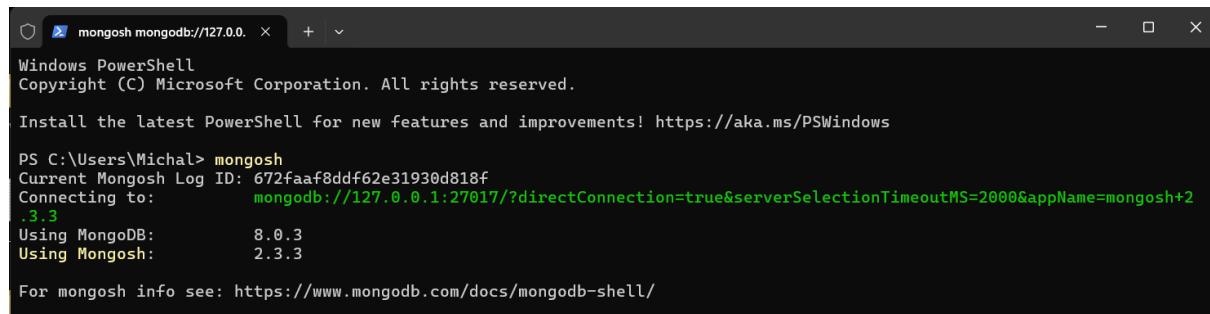


Zrzut ekranu nr 1, 2: instalacja MongoDB; konfiguracja MongoDB



Zrzut ekranu nr 3: aktywna usługa serwera MongoDB

Poza serwerem zainstalowałem również powłokę **mongosh** do komunikowania się z bazą (<https://www.mongodb.com/try/download/shell>):

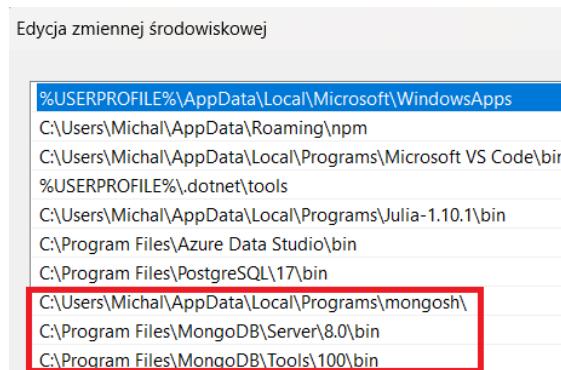


```
PS C:\Users\Michał> mongosh
Current Mongosh Log ID: 672faaf8ddf62e31930d818f
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.3
Using MongoDB:     8.0.3
Using Mongosh:    2.3.3
For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/
```

Zrzut ekranu nr 4: poprawne uruchomienie mongosh

Ostatnim narzędziem, które potrzebujemy do realizacji jest pakiet narzędzi, a dokładniej importer danych **mongoimport** – proste narzędzie konsolowe umożliwiające wczytywanie danych do kolekcji z plików JSON (<https://www.mongodb.com/docs/database-tools/installation/installation/>).

Instalacja przebiegła analogicznie do powyższych narzędzi. Ostatnim krokiem było dodanie ścieżek do mongo, mongosh oraz mongoimport do zmiennej środowiskowej \$PATH, aby wygodnie uruchamiać narzędzia.



Zrzut ekranu nr 5: modyfikacja zmiennej środowiskowej \$PATH

### Import danych:

Dane zostały dostosowane przy użyciu skryptu *parseDocumentsScript.py*. Na wejściu podałem zbiór danych z Kaggle'a w postaci pliku json, a na wyjściu wygenerowałem trzy pliki: games – z dodanymi referencjami oraz wygenerowane mock'i developers i publishers (generowane, gdy dokumenty wejściowe posiadały pola developer bądź publisher). Tym sposobem zasilam bazę trzema kolekcjami:

- games – 45 000 dokumentów
- developers - 2011 dokumentów
- publishers – 18 000 dokumentów

```
PS C:\Users\Michał\Desktop\SEM_II\Zaawansowane_systemy_baz_danych\Projekt_III> mongoimport .\final_data_new.json -d zsbd
  --jsonArray
  -c inputCollection
2024-11-27T18:32:50.137+0100      connected to: mongodb://localhost/
2024-11-27T18:32:51.768+0100      Failed: invalid JSON input. Position: 135. Character: N
2024-11-27T18:32:51.768+0100      45000 documents(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\Michał\Desktop\SEM_II\Zaawansowane_systemy_baz_danych\Projekt_III>
PS C:\Users\Michał\Desktop\SEM_II\Zaawansowane_systemy_baz_danych\Projekt_III>
PS C:\Users\Michał\Desktop\SEM_II\Zaawansowane_systemy_baz_danych\Projekt_III> mongoimport .\games.json -d zsbd -c games
  --jsonArray
2024-11-27T18:41:06.045+0100      connected to: mongodb://localhost/
2024-11-27T18:41:07.672+0100      Failed: invalid JSON input. Position: 168. Character: N
2024-11-27T18:41:07.672+0100      45000 documents(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\Michał\Desktop\SEM_II\Zaawansowane_systemy_baz_danych\Projekt_III>
PS C:\Users\Michał\Desktop\SEM_II\Zaawansowane_systemy_baz_danych\Projekt_III> mongoimport .\publishers.json -d zsbd -c
publishers --jsonArray
2024-11-27T18:41:32.529+0100      connected to: mongodb://localhost/
2024-11-27T18:41:32.565+0100      2011 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\Michał\Desktop\SEM_II\Zaawansowane_systemy_baz_danych\Projekt_III> mongoimport .\developers.json -d zsbd -c
developers --jsonArray
2024-11-27T18:41:55.105+0100      connected to: mongodb://localhost/
2024-11-27T18:41:55.316+0100      Failed: invalid JSON input. Position: 77. Character: N
2024-11-27T18:41:55.346+0100      18000 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\Michał\Desktop\SEM_II\Zaawansowane_systemy_baz_danych\Projekt_III> |
```

Zrzut ekranu nr 6: import danych do bazy zsbd (pomimo ostrzeżeń dane są poprawnie załadowane)

Import z plików JSON okazał się banalny, wystarczy wykorzystać narzędzie mongoimport wskazując na bazę oraz kolekcję docelową wraz z flagą --jsonArray.

```
zsbd> show collections
developers
games
publishers
```

Zrzut ekranu nr 7: poprawnie wczytane kolekcje

Następnie przeszedłem do kroku migracji identyfikatorów. Należało pozbyć się pomocniczych wiązań referencyjnych publisherId w publishers oraz developerId w developers, a w kolekcji games podmienić referencje na \_id, które są autoinkrementującymi się ObjectId. W tym celu wykonałem poniższe polecenia (przy migracji wspomogłem się ChatemGPT i poprawiłem jego błędy):

```
zsbd> const publisherMap = {}

zsbd> const developerMap = {}

zsbd> db.publishers.find({}).forEach(publisher => {
...   publisherMap[publisher.publisherId] = publisher._id;
... });

zsbd> db.developers.find({}).forEach(developer => {
...   developerMap[developer.developerId] = developer._id;
... });
```

Zrzut ekranu nr 8: zapisanie wygenerowanych, rzeczywistych \_id z kolekcji

```
zsbd> db.games.find({}).forEach(game => {
...   if (game.publisherId && publisherMap[game.publisherId]) {
...     db.games.updateOne(
...       { _id: game._id },
...       { $set: { publisherId: publisherMap[game.publisherId] } }
...     );
...   }
...
...   if (game.developerId && developerMap[game.developerId]) {
...     db.games.updateOne(
...       { _id: game._id },
...       { $set: { developerId: developerMap[game.developerId] } }
...     );
...   }
...});
```

Zrzut ekranu nr 9: podmiana referencji w kolekcji games na rzeczywiste identyfikatory

```
zsbd> db.publishers.updateMany({}, { $unset: { publisherId: "" } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2011,
  modifiedCount: 2011,
  upsertedCount: 0
}
zsbd> db.developers.updateMany({}, { $unset: { developerId: "" } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 18000,
  modifiedCount: 18000,
  upsertedCount: 0
}
```

Zrzut ekranu nr 10: usunięcie pomocniczych identyfikatorów

Weryfikacja poprzez demonstrację przykładowych dokumentów:

```
zsbd> db.games.findOne()
{
  _id: ObjectId('62096d13a61317cb22cdade8'),
  img_url: 'https://steamcdn-a.akamaihd.net/steam/apps/730/header.jpg?t=1592263625',
  date: 'Aug 21, 2012',
  developer: 'Valve, Hidden Path Entertainment',
  publisher: 'Valve',
  fullDesc: 'Counter-Strike: Global Offensive (CS: GO) expands upon the team-based action gameplay that it played online PC action game in the world almost immediately after its release in August 1999,' said Doug Lombardi at Valve to gamers on the PC as well as the next gen consoles and the Mac.'
  ,
  price: 'free',
  url_info:
    url: 'https://store.steampowered.com/app/730/CounterStrike_Global_Offensive/?snr=1_7_7_230_150_1',
    id: '730',
    type: 'app',
    url_name: 'CounterStrike Global Offensive'
  ,
  name: 'Counter-Strike: Global Offensive',
  categories: [
    'Steam Achievements Full',
    'controlled supportSteam',
    'Trading Cards Steam',
    'Workshop In-App Purchases Valve',
    'Content EnabledStats Remote',
    'Play on',
    'Phone Remote Play',
    'on Tablet Remote',
    'Play on'
  ],
  publisherId: ObjectId('62096d13a61317cb22cdade8'),
  developerId: ObjectId('6747641873588b0176d4af31')
```

Zrzut ekranu nr 11: dokument z kolekcji games (ze zrzutu ekranu wycięto pola tags oraz opis systemowy, aby oszczędzić miejsce)

```
zsbd> db.publishers.find({_id: ObjectId('67476474fd11elc9aeb5929f')})
[ {
    _id: ObjectId('67476474fd11elc9aeb5929f'),
    name: 'Valve',
    dateOfFounding: '1982-07-02',
    ceo: 'Hector Robinson',
    country: 'Netherlands Antilles'
}
]
zsbd> db.developers.find({_id: ObjectId('6747641874388b0176d4af31')})
[ {
    _id: ObjectId('6747641874388b0176d4af31'),
    name: 'Valve, Hidden Path Entertainment',
    dateOfFounding: '2003-10-13',
    lead_developer: 'Cody Schroeder',
    studio_size: 65,
    headquarters: 'North Michaelland'
}
]
```

Zrzut ekranu nr 12: poprawnie znalezione dokumentu z kolekcji developers i publishers

W tym momencie zorientowałem się, że należy poprawić kolejną kwestię w zbiorze. Ceny (pole price) są typu string, co uniemożliwia filtrowanie z pomocą operatorów \$gt, \$lt. Przekonwertowałem cenę do float poprzez skrypt:

```
zsbd> db.games.find().forEach(function(game) {
...     if (game.price && typeof game.price === "string") {
...         let numericPrice = parseFloat(game.price);
...
...         if (!isNaN(numericPrice)) {
...             db.games.updateOne(
...                 { _id: game._id },
...                 { $set: { price: numericPrice } }
...             );
...         } else {
...             db.games.updateOne(
...                 { _id: game._id },
...                 { $set: { price: 0 } }
...             );
...         }
...     }
...});
```

Zrzut ekranu nr 13: konwersja ceny do typu float

Nastecną kwestią była konwersja dat wydania gier, ponieważ zdziwiło mnie niedziałające sortowanie po dacie (niestety niektóre daty było zapisane w bardzo, bardzo dziwnym formacie 😊).

```
zsbd> db.games.find().forEach(function(game) {
...     if (game.date && typeof game.date === "string") {
...         let parsedDate = new Date(game.date);
...
...         if (!isNaN(parsedDate.getTime())) {
...             db.games.updateOne(
...                 { _id: game._id },
...                 { $set: { date: parsedDate } }
...             );
...         } else {
...             print("Invalid date format for _id:", game._id, "date:", game.date);
...         }
...     }
... })
Invalid date format for _id: ObjectId('67476414a61317cb22ce1d48') date: Q1 2021
Invalid date format for _id: ObjectId('67476414a61317cb22ce1d66') date: Q1 2021
Invalid date format for _id: ObjectId('67476415a61317cb22ce34dc') date: 30th October 2020
Invalid date format for _id: ObjectId('67476415a61317cb22ce3749') date: As soon as the unicorns allow it
```

Zrzut ekranu nr 14: konwersja dat na tym IsoDate

Z racji, iż tylko 4 rekordy posiadają błędne daty naprawiłem je manualnie, ponieważ skrypt byłby nadmierny (w przypadku większej liczby błędnych rekordów należałoby objąć inną strategię i próbować naprawiać rekordy automatycznie).

```
zsbd> db.games.updateOne(
...   { _id: ObjectId("67476414a61317cb22ce1d48") },
...   { $set: { date: new Date("2021-01-01") } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
zsbd>

zsbd> db.games.updateOne(
...   { _id: ObjectId("67476414a61317cb22ce1d66") },
...   { $set: { date: new Date("2021-01-01") } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}

zsbd> db.games.updateOne(
...   { _id: ObjectId("67476415a61317cb22ce34dc") },
...   { $set: { date: new Date("2020-10-30") } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
zsbd>

zsbd> db.games.updateOne(
...   { _id: ObjectId("67476415a61317cb22ce3749") },
...   { $set: { date: null } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Zrzut ekranu nr 15: manualne reperowanie danych

### Zapytanie z wykorzystaniem dokumentów zagnieżdzonych:

```
zsbd> db.games.find({ "popu_tags": { $all: [ "Shooter", "Multiplayer" ] }, "price": { $gte: 100 }, "requirements.minimum.windows.os": { $regex: /Windows\S+7/ } }, { "full_desc": 0 }).sort({ date: -1 }).limit(1)
[
  {
    _id: ObjectId("67476414a61317cb22ce1e91"),
    img_url: 'https://cdn.cloudflare.steamstatic.com/steam/apps/1402090/header.jpg?t=1602500154',
    date: ISODate('2020-09-17T22:00:00Z'),
    developer: 'Pieslice Productions',
    full_desc: { sort: 'game' },
    requirements: {
      minimum: {
        windows: {
          processor: ' Dual Core or better',
          memory: ' 2 ',
          graphics: '',
          os: ' Windows 7'
        }
      },
      recommended: {
        windows: {
          processor: ' Intel Core i5 or AMD equivalent',
          memory: ' 4 ',
          graphics: '',
          os: ' Windows 10'
        }
      }
    },
    popu_tags: [
      'Shoot',
      'Bullet',
      'Down',
      'Difficult',
      'Graphics',
      'Arcade',
      'Linear',
      'Colorful',
      'Attack',
      'Local',
      'Multiplayer',
      'Singleplayer'
    ],
    price: 799,
    url_info: {
      url: 'https://store.steampowered.com/app/1402090/Crisis_Wing/?snr=1_7_7_230_150_1329',
      id: '1402090',
      type: 'app',
      url_name: 'Crisis Wing'
    },
    name: 'Crisis Wing',
    categories: [
      'Single-playerShared/Split Screen',
      'Co-opFull controller',
      'supportSteam is',
      'learning about',
      'this game'
    ],
    developerId: ObjectId('6747641874388b0176d4e600')
  }
]
```

Zrzut ekranu nr 16: nietrywialne zapytanie zwracające nam najnowszą grę, która ma tag'i „Shooter” oraz „Multiplayer”, droższa niż 100 euro, o minimalnym wymaganiu systemu Windows 7, z projekcją wyciąłem opis, który jest zbyt długi do prezentacji

### Zapytanie z wykorzystaniem referencji:

W tym celu należy skorzystać z wbudowanego w MongoDB mechanizmu Aggregation Framework, który pozwala nam na budowanie potoków (*ang. pipelines*) w ramach, których możemy łączyć, filtrować czy grupować dane.

```
zsdb> db.games.aggregate([{$lookup: { from: "publishers", localField: "publisherId", foreignField: "_id", as: "publisherDetails"}}, {$unwind: "$publisherDetails"}, {$group: {_id: ObjectId("67476d13a61317cb22cdade8"), date: ISODate("2012-08-20T22:00:00.000Z"), requirements: {}, minimum: {}, windows: { processor: "Intel® Core™ 2 Duo E6600 or AMD Phenom™ X3 8750 processor or better", memory: "2 GB RAM", graphics: "Video card must be 256 MB or more and should be a ", os: "Windows® 7/Vista/XP"}, linux: { processor: "64-bit Dual core from Intel or AMD at 2.8 GHz", memory: "4 GB RAM", graphics: "nVidia GeForce 8600/9600GT, ATI/AMD Radeon HD2600/3600 (Graphic Drivers: nVidia 310, AMD 12.11), OpenGL 2.1", os: "Ubuntu 12.04"}, recommended: {}}, popu_tags: [{ Shooter, 'Multiplayer', 'Competitive', 'Action', 'Team', 'Base', 'Sports', 'Tactical', 'First', 'Person', 'OnLine', 'Strategy', 'Military', 'Difficult', 'Trading', 'Realistic', 'Fast', 'Paced', 'Moddable' }], name: 'Counter-Strike: Global Offensive', publisherDetails: { _id: ObjectId("67476d4f4d1e1c9ae65929f"), name: 'Valve', dateFounded: '1998-07-02', ceo: 'Hector Robinson', country: 'Netherlands Antilles' } }])
```

**Zrzut ekranu nr 17:** zapytanie zwraca nam grę z podwiązanym za pomocą referencji publisherem, cały rezultat jest rzutowany tak by zwrócić okrojone, tylko te dane, które nas interesują (takie dane można by prezentować w dedykowanej zakładce w sklepie)

Kod dostosowujący i obrabiający dane znajduje się w katalogu */preparingData*, a zapytania w pliku *queries.js*.

### Logika biznesowa:

Przygotowałem wybrane funkcje wraz z walidacją, które zapewnią obsługę logiki biznesowej. Zrealizowałem mechanizmy:

- Dodawanie gry – **addGame(game)**
- Modyfikacja gry – **modifyGame(gameId, updatedGame)**
- Usuwanie gry – **deleteGame(gameId)**
- Wyszukiwanie gier po nazwie – **findGamesByName()**
- Zaawansowane wyszukiwanie, które uwzględnia zawieranie podanych kategorii, cenę nie droższą niż podana oraz datę wydania późniejszą niż wskazana przez użytkownika – **advancedFindGames({categories = [], maxPrice, releaseDate})**

Poniżej zaprezentuję pełne testy powyższych funkcjonalności. Kod z zaimplementowaną logiką biznesową znajduje się w załączonym pliku *businessLogic.js*. Dane do testów są syntetyczne i wygenerowane przez ChatGPT. Testy funkcjonalności znajdują się w skryptach w katalogu */businessLogicTests*.

- **Dodawanie gry:**

Scenariusz pozytywny (poprawne dodanie gry):

```
zsbd> db.games.findOne({name: "Galaxy Raiders: Infinite Void"})
null
zsbd> var addedGame = addGame(newGame)

zsbd> addedGame
ObjectID('6748b111482597ff940d8191')
zsbd> db.games.findOne({_id: ObjectId('6748b111482597ff940d8191')})
{
  _id: ObjectId('6748b111482597ff940d8191'),
  name: 'Galaxy Raiders: Infinite Void',
  date: ISODate('2023-07-15T00:00:00.000Z'),
  developer: 'Starforge Studios',
  publisher: 'Void Entertainment',
  full_desc: {
    sort: 'game',
    desc: 'A sci-fi action RPG where players explore uncharted galaxies, battle alien forces, and uncover ancient secrets.'
  },
  requirements: {
    minimum: {
      windows: {
        processor: 'Intel Core i3',
        memory: '4 GB RAM',
        graphics: 'Intel HD Graphics 620',
        os: 'Windows 8/10'
      }
    },
    recommended: {
      windows: {
        processor: 'Intel Core i7',
        memory: '8 GB RAM',
        graphics: 'NVIDIA GeForce GTX 1060',
        os: 'Windows 10'
      }
    }
  },
  popu_tags: [ 'Sci-fi', 'RPG', 'Space Exploration', 'Action' ],
  price: 29.99,
  url_info: {
    url: 'https://store.fakergames.com/app/1234/Galaxy_Raiders_Infinite_Void/',
    id: '1234',
    type: 'app',
    url_name: 'Galaxy Raiders Infinite Void'
  },
  categories: [ 'Sci-fi', 'RPG', 'Action', 'Space Exploration' ],
  publisherId: ObjectId('67476474fd11e1c9ae5b5929f'),
  developerId: ObjectId('6747641874388bb176d4af31')
```

Zrzut ekranu nr 18: rezultat testu

Scenariusz negatywny (nie zdefiniowano wymaganego pola – nazwa gry):

```
zsbd> var addedGame = addGame(gameWithNoName)
Error: Game name is required.
```

Zrzut ekranu nr 19: rezultat testu

Scenariusz negatywny (gra o podanym tytule już istnieje):

```
zsbd> var addedGame = addGame(newGame);
Error: Game: Galaxy Raiders: Infinite Void already exists.
```

Zrzut ekranu nr 20: rezultat testu

Scenariusz negatywny (podobno referencję do nieistniejącego publishera):

```
zsbd> var addedGame = addGame(gameWithNotExistingPublisher);
Error: Publisher with id:aaaaaaaaaaaaaaaaaaaaaaa does not exist
```

Zrzut ekranu nr 21: rezultat testu

- **Modyfikacja gry:**

Scenariusz pozytywny (modyfikacja istniejącej gry):

```
zsbd> var existingGame = db.games.findOne({name: "Galaxy Raiders: Infinite Void"});  
zsbd> var validGame = {  
...   name: "UPDATED: Galaxy Raiders: Infinite Void",  
...   publisherId: ObjectId("67476474fd1le1c9ae5929f"),  
...   developerId: ObjectId("6747641874388b0176d4af31")  
... };  
zsbd> modifyGame(existingGame._id, validGame);  
zsbd> db.games.findOne({name: "Galaxy Raiders: Infinite Void"})  
null  
zsbd> db.games.findOne({name: "UPDATED: Galaxy Raiders: Infinite Void"})  
{  
  _id: ObjectId("6748b111482597ff940d8191"),  
  name: 'UPDATED: Galaxy Raiders: Infinite Void',  
  date: ISODate('2023-07-15T00:00:00Z'),  
  developer: 'StarForge Studios',  
  publisher: 'Void Entertainment',  
  full_desc: {  
    sort: 'game',  
    desc: 'A sci-fi action RPG where players explore uncharted galaxies, battle alien forces, and uncover ancient secrets.'  
  },  
  requirements: {  
    minimum: {  
      windows: {  
        processor: 'Intel Core i3',  
        memory: '4 GB RAM',  
        graphics: 'Intel HD Graphics 620',  
        os: 'Windows 8/10'  
      }  
    },  
    recommended: {  
      windows: {  
        processor: 'Intel Core i7',  
        memory: '8 GB RAM',  
        graphics: 'NVIDIA GeForce GTX 1060',  
        os: 'Windows 10'  
      }  
    }  
  },  
  popu_tags: [ 'Sci-fi', 'RPG', 'Space Exploration', 'Action' ],  
  price: 29.99,  
  url_info: {  
    url: 'https://store.fakegames.com/app/1234/Galaxy_Raiders_Infinite_Void/',  
    id: '1234',  
    type: 'app',  
    url_name: 'Galaxy Raiders Infinite Void'  
  },  
  categories: [ 'Sci-fi', 'RPG', 'Action', 'Space Exploration' ],  
  publisherId: ObjectId("67476474fd1le1c9ae5929f"),  
  developerId: ObjectId("6747641874388b0176d4af31")  
}
```

Zrzut ekranu nr 22: rezultat testu

Scenariusz negatywny (gra o podanym id nie istnieje):

```
zsbd> var nonExistingGameId = ObjectId("aaaaaaaaaaaaaaaaaaaaaaaaaaa");  
zsbd> var validUpdateGame = {  
...   name: "Non-Existing Game"  
... };  
zsbd> modifyGame(nonExistingGameId, validUpdateGame);  
Error: Game with id: aaaaaaaaaaaaaaaaaaaaa does not exist.
```

Zrzut ekranu nr 23: rezultat testu

Scenariusz negatywny (próba modyfikacji nazwy na taką, która jest już zajęta):

```
zsbd> var existingGame = db.games.findOne({name: "UPDATED: Galaxy Raiders: Infinite Void"});  
zsbd> var game = db.games.findOne();  
zsbd> var gameWithTakenName = {  
...   name: game.name  
... };  
zsbd> modifyGame(existingGame._id, gameWithTakenName);  
Error: Given game name: Counter-Strike: Global Offensive is already taken.
```

Zrzut ekranu nr 24: rezultat testu

Scenariusz negatywny (podano publisherId nieistniejącego wydawcy):

```
zsbd> var existingGame = db.games.findOne({name: "UPDATED: Galaxy Raiders: Infinite Void"});  
zsbd> var invalidPublisherUpdate = { publisherId: ObjectId("aaaaaaaaaaaaaaaaaaaaaaa") };  
zsbd> modifyGame(existingGame._id, invalidPublisherUpdate);  
Error: Publisher with id: aaaaaaaaaaaaaaaaaaaaa does not exist.
```

Zrzut ekranu nr 25: rezultat testu

- **Usuwanie gry:**

Scenariusz pozytywny (usunięcie istniejącej gry):

```
zsbd> var newGame = { name: "Test game to delete", date: ISODate("2023-07-15T00:00:00.000Z"), category: "Space Exploration" };
zsbd> addGame(newGame);
ObjectID('6748bf59482597ff940d8194')
zsbd> var game = db.games.findOne({name: "Test game to delete"})
zsbd> var result = deleteGame(game._id);
zsbd> result
Game with id: 6748bf59482597ff940d8194 successfully deleted.
zsbd> db.games.findOne({name: "Test game to delete"})
null
```

Zrzut ekranu nr 26: rezultat testu

Scenariusz negatywny (gra o podanym id nie istnieje):

```
zsbd> var notExistingGameId = ObjectId("aaaaaaaaaaaaaaaaaaaaaaaa");
zsbd> deleteGame(notExistingGameId);
Error: Game with id: aaaaaaaaaaaaaaaaaaaaa does not exist.
```

Zrzut ekranu nr 27: rezultat testu

- **Wyszukiwanie gier po nazwie:**

Wyszukiwanie wykorzystuje regex i zwraca gry, które zawierają podany fragment w nazwie. Dane są ograniczone do id, daty, ceny i nazwy

Scenariusz pozytywny:

```
zsbd> var games = findGamesByName("Ski jump")
zsbd> games
[
  {
    _id: ObjectId('67476414a61317cb22cdf2ed'),
    date: ISODate('2019-12-05T23:00:00.000Z'),
    price: 199,
    name: 'Ski Jumping Pro VR'
  },
  {
    _id: ObjectId('67476414a61317cb22cdf851'),
    date: ISODate('2020-01-23T23:00:00.000Z'),
    price: 599,
    name: 'K-Point Ski Jumping'
  },
  {
    _id: ObjectId('67476414a61317cb22ce1598'),
    date: ISODate('2017-06-07T22:00:00.000Z'),
    price: 499,
    name: 'Ski Jump VR'
  },
  {
    _id: ObjectId('67476415a61317cb22ce5493'),
    date: ISODate('2020-04-02T22:00:00.000Z'),
    price: 999,
    name: 'Ultimate Ski Jumping 2020'
  }
]
```

Zrzut ekranu nr 28: rezultat wyszukania gry zawierającej w nazwie „Ski jump”

Scenariusz negatywny (nie podano ciągu znaków do wyszukiwania):

```
zsbd> var games = findGamesByName(1)
Error: Invalid or missing game name parameter.
```

Zrzut ekranu nr 29: rezultat testu

- Zaawansowane wyszukiwanie:

```
zsbd> var games = advancedFindGames({maxPrice: 100, releaseDate: new Date("2021-01-01")});  
zsbd> games  
[  
  {  
    date: ISODate('2021-01-19T23:00:00.000Z'),  
    price: 0,  
    name: 'Imagine Earth',  
    categories: [  
      'Single-playerSteam Achievements Steam',  
      'Trading Cards Steam',  
      'Workshop Steam Cloud Steam'  
    ]  
  },  
  {  
    date: ISODate('2021-07-30T22:00:00.000Z'),  
    price: 0,  
    name: 'Death World',  
    categories: [ 'Single-playerPartial Controller', 'Support Profile Features' ]  
  },  
  {  
    name: 'UPDATED: Galaxy Raiders: Infinite Void',  
    date: ISODate('2023-07-15T00:00:00.000Z'),  
    price: 29.99,  
    categories: [ 'Sci-fi', 'RPG', 'Action', 'Space Exploration' ]  
  }  
]
```

Zrzut ekranu nr 30: wyszukiwanie gier nie droższych niż 100 euro, wydanych po 1 stycznia 2021

```
zsbd> var games = advancedFindGames({categories: ["Play on", "Phone Remote Play"], maxPrice: 200, releaseDate: "2020-06-06"});  
zsbd> games  
[  
  {  
    date: ISODate('2020-06-10T22:00:00.000Z'),  
    price: 149,  
    name: 'Journey',  
    categories: [  
      'Single-playerOnline Co-opSteam',  
      'Achievements Full controller',  
      'supportSteam Cloud Remote',  
      'Play on',  
      'Phone Remote Play',  
      'on Tablet Remote',  
      'Play on'  
    ]  
  },  
  {  
    date: ISODate('2020-07-08T22:00:00.000Z'),  
    price: 0,  
    name: 'Soda Dungeon 2',  
    categories: [  
      'Single-playersteam Achievements In-App',  
      'Purchases Steam Cloud Remote',  
      'Play on',  
      'Phone Remote Play',  
      'on Tablet'  
    ]  
  },  
  {  
    date: ISODate('2020-08-05T22:00:00.000Z'),  
    price: 0,  
    name: 'Soccering',  
    categories: [  
      'Single-playerShared/Split Screen',  
      'PvPShared/Split Screen',  
      'Co-opSteam Achievements Full',  
      'controller supportRemote',  
      'Play on',  
      'Phone Remote Play',  
      'on Tablet Remote',  
      'Play on',  
      'TVRemote Play',  
      'Together Profile Features',  
      'Limited Requires agreement',  
      'to a',  
      '3rd-party EULASoccering'  
    ]  
  }]
```

Zrzut ekranu nr 31: wyszukiwanie gier nie droższych niż 200 euro, wydanych po 6 czerwcu 2021, o kategoriach „Play on” i „Phone Remote Play”

Bardzo analogicznie funkcje można przygotować do obsługi kolekcji publishers oraz developers:

- Dodawanie wydawcy – addPublisher()
- Modyfikacja wydawcy – modifyPublisher()
- Usuwanie wydawcy – deletePublisher()
- Wyszukiwanie wydawcy po nazwie – findPublisherByName()
- Dodawanie developera – addDeveloper ()
- Modyfikacja developera – modifyDeveloper ()
- Usuwanie developera – deleteDeveloper()
- Wyszukiwanie developera po nazwie – findDeveloperByName()

### Indeksy:

Zachowanie indeksów zademonstruję na trzech przykładach:

1. Nałożenie indeksu prostego na pole date oraz testy pobierania gier wydanych przed podaną datą (zapytania z filtrowaniem po dacie).
2. Zbadanie wydajności indeksu prostego na polu name w porównaniu do indeksu dedykowanego do przeszukiwań tekstowych typu text.
3. Sprawdzenie wydajności indeksu kompozytowego na name w połączeniu z date.

Eksperymenty opieram na wbudowaną funkcję **explain()**, która po podaniu parametru „**executionStats**” daje nam bardzo ciekawe informacje – możemy obserwować, który plan zapytania został wybrany (skanowanie kolekcji/skanowanie indeksu), obejrzeć ile czasu trwał dany etap przetwarzania i ile rekordów zostało odfiltrowanych na danym etapie oraz inne statystki analityczne. Kod testów zamieściłem w pliku *indexesTests.js*.

Domyślnie mongo tworzy indeksy na polach `_id` w kolekcjach, zostawiam te indeksy, ponieważ jest to częsty sposób wyszukiwania i jest to słuszne podejście.

```
zsbd> db.games.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
```

Zrzut ekranu nr 32: domyślnie utworzony indeks przez MongoDB na `_id`

### Testy indeksu prostego na polu date:

```
zsbd> db.games.find({ "date": { $lt: new ISODate("2020-01-01T00:00:00Z") } }).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'zsbd.games',
    parsedQuery: { date: { '$lt': ISODate('2020-01-01T00:00:00.000Z') } },
    indexFilterSet: false,
    queryHash: '302FAA8E',
    planCacheKey: '9A93560E',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { date: { '$lt': ISODate('2020-01-01T00:00:00.000Z') } },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 37990,
    executionTimeMillis: 33,
    totalKeysExamined: 0,
    totalDocsExamined: 45000,
    executionStages: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { date: { '$lt': ISODate('2020-01-01T00:00:00.000Z') } },
      nReturned: 37990,
      executionTimeMillisEstimate: 32,
      works: 45001,
      advanced: 37990,
      needTime: 7010,
      needYield: 0,
      saveState: 2,
      restoreState: 2,
      isEOF: 1,
      direction: 'forward',
      docsExamined: 45000
    }
  },
  command: {
    find: 'games',
    filter: { date: { '$lt': ISODate('2020-01-01T00:00:00.000Z') } },
    '$db': 'zsbd'
  },
  serverInfo: {
    host: 'ZDANUKM',
    port: 27017,
    version: '8.0.3',
    gitVersion: '89d97f2744a2b9851ddfb51bdf22f687562d9b06'
  }
}
```

Zrzut ekranu nr 33: zapytanie przed wprowadzeniem indeksu na dacie, przetwarzanie zajęło 33ms i wybrana została strategia skanowania kolekcji (nie znaleziono indeksów), dopasowano 37990 rekordów (znaczna część całej kolekcji)

Następnie dodałem indeks prosty na polu date:

```
zsbd> db.games.createIndex({ "date": 1 })
date_1
```

Zrzut ekranu nr 34: założenie indeksu na polu date

```
zsbd> db.games.find({ "date": { $lt: new ISODate("2020-01-01T00:00:00Z") } }).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'zsbd.games',
    parsedQuery: { date: { '$lt': ISODate('2020-01-01T00:00:00.000Z') } },
    indexFilterSet: false,
    queryHash: '302FAA8E',
    planCacheKey: '36D88A09',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { date: 1 },
        indexName: 'date_1',
        isMultiKey: false,
        multiKeyPaths: { date: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: {
          date: [
            '[new Date(-9223372036854775808), new Date(1577836800000))'
          ]
        }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 37990,
    executionTimeMillis: 95,
    totalKeysExamined: 37990,
    totalDocsExamined: 37990,
    executionStages: {
      isCached: false,
      stage: 'FETCH',
      nReturned: 37990,
      executionTimeMillisEstimate: 73,
      works: 37991,
      advanced: 37990,
      needTime: 0,
      needYield: 0,
      saveState: 5,
      restoreState: 5,
      isEOF: 1,
      docsExamined: 37990,
      alreadyHasObj: 0,
      inputStage: {
        stage: 'IXSCAN',
        nReturned: 37990,
        executionTimeMillisEstimate: 20,
        works: 37991,
        advanced: 37990,
        needTime: 0,
        needYield: 0,
        saveState: 5,
        restoreState: 5,
        isEOF: 1,
      }
    }
  }
}
```

Zrzut ekranu nr 35: zapytanie z wykorzystaniem indeksu, czas realizacji: 95ms

Jak widzimy w tym zapytaniu indeks nie pomógł, a wręcz przeciwnie spowolnił wyszukiwanie. MongoDB wybrało plan ze skanowaniem indeksu, indeks musiał zeskanować niemal całą kolekcję (37990/45000), a zatem był zbędny w tym konkretnym przypadku i zapytanie było znacznie wolniejsze niż bez indeksu.

Spróbowuję wykonać zapytanie jeszcze raz, ale zawęzić wyniki do maksymalnie 2016 roku zamiast 2020 (czyli operować na węższym zakresie).

```
zsbd> db.games.find({ "date": { $lt: new ISODate("2016-01-01T00:00:00Z") } }).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'zsbd.games',
    parsedQuery: { date: { '$lt': ISODate('2016-01-01T00:00:00.000Z') } },
    indexFilterSet: false,
    queryHash: '302FAA8E',
    planCacheKey: '36D88A09',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { date: 1 },
        indexName: 'date_1',
        isMultiKey: false,
        multiKeyPaths: { date: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: {
          date: [
            '[new Date(-9223372036854775808), new Date(1451606400000))'
          ]
        }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 10097,
    executionTimeMillis: 15,
    totalKeysExamined: 10097,
    totalDocsExamined: 10097,
    executionStages: {
      isCached: false,
      stage: 'FETCH',
      nReturned: 10097,
      executionTimeMillisEstimate: 10,
      works: 10098,
      advanced: 10097,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      docsExamined: 10097,
      alreadyHasObj: 0,
      inputStage: {
        stage: 'IXSCAN',
        nReturned: 10097,
        executionTimeMillisEstimate: 0,
        works: 10098,
        advanced: 10097,
        needTime: 0,
        needYield: 0,
        saveState: 0,
        restoreState: 0,
        isEOF: 1
      }
    }
  }
}
```

Zrzut ekranu nr 36: zapytanie z wykorzystaniem indeksu, tym razem razem tylko 15ms

Zobaczmy również wyszukiwanie z datą do 2016 bez indeksu:

```
zsbd> db.games.find({ "date": { $lt: new ISODate("2016-01-01T00:00:00Z") } }).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'zsbd.games',
    parsedQuery: { date: { '$lt': ISODate('2016-01-01T00:00:00.000Z') } },
    indexFilterSet: false,
    queryHash: '302FAA8E',
    planCacheKey: '9A93560E',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { date: { '$lt': ISODate('2016-01-01T00:00:00.000Z') } },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 10097,
    executionTimeMillis: 33,
    totalKeysExamined: 0,
    totalDocsExamined: 45000,
    executionStages: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { date: { '$lt': ISODate('2016-01-01T00:00:00.000Z') } },
      nReturned: 10097,
      executionTimeMillisEstimate: 31,
      works: 45001,
      advanced: 10097,
      needTime: 34903,
      needYield: 0,
      saveState: 1,
      restoreState: 1,
      isEOF: 1,
      direction: 'forward',
      docsExamined: 45000
    }
  }
}
```

**Zrzut ekranu nr 37:** zapytanie bez indeksu, wykorzystane skanowanie kolekcji, tutaj jest wolniej: 33ms

W tym przypadku indeks dał nam 2 razy szybciej wynik. Zatem przy odpowiednim użyciu indeks przyspieszy nam zapytania, powinniśmy rozważyć jak dużą część zbioru/od której strony będzie wykorzystany indeks. Decyzję możemy podjąć po przeprowadzeniu testów wydajnościowych i analizie.

### Testy indeksu prostego na polu name:

Wyszukiwanie gry, która ma w nazwie fragment „Counter-Str” bez indeksu:

```
zsbd> db.games.find({ name: { $regex: "Counter-Str", $options: "i" } }).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'zsbd.games',
    parsedQuery: { name: { '$regex': 'Counter-Str', '$options': 'i' } },
    indexFilterSet: false,
    queryHash: '684877C5',
    planCacheKey: 'E9672561',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { name: { '$regex': 'Counter-Str', '$options': 'i' } },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 7,
    executionTimeMillis: 35,
    totalKeysExamined: 0,
    totalDocsExamined: 45002,
    executionStages: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { name: { '$regex': 'Counter-Str', '$options': 'i' } },
      nReturned: 7,
      executionTimeMillisEstimate: 21,
      works: 45003,
      advanced: 7,
      needTime: 44995,
      needYield: 0,
      saveState: 2,
      restoreState: 2,
      isEOF: 1,
      direction: 'forward',
      docsExamined: 45002
    }
  }
},
```

Zrzut ekranu nr 38: zapytanie bez indeksu, przetwarzanie zajęło 35ms

Następnie dodałem indeks prosty na pole name:

```
zsbd> db.games.createIndex({ "name": 1 })
name_1
```

Zrzut ekranu nr 39: założenie indeksu na polu name

```
zsbd> db.games.find({ name: { $regex: "Counter-Str", $options: "i" } }).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'zsbd.games',
    parsedQuery: { name: { '$regex': 'Counter-Str', '$options': 'i' } },
    indexFilterSet: false,
    queryHash: '684877C5',
    planCacheKey: 'ABF3263F',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        filter: { name: { '$regex': 'Counter-Str', '$options': 'i' } },
        keyPattern: { name: 1 },
        indexName: 'name_1',
        isMultiKey: false,
        multiKeyPaths: { name: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { name: [ '[ "", {} ]', '[ /Counter-Str/i, /Counter-Str/i ]' ] }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 7,
    executionTimeMillis: 39,
    totalKeysExamined: 45002,
    totalDocsExamined: 7,
    executionStages: {
      isCached: false,
      stage: 'FETCH',
      nReturned: 7,
      executionTimeMillisEstimate: 32,
      works: 45003,
      advanced: 7,
      needTime: 44995,
      needYield: 0,
      saveState: 2,
      restoreState: 2,
      isEOF: 1,
      docsExamined: 7,
      alreadyHasObj: 0,
      inputStage: {
        stage: 'IXSCAN',
        filter: { name: { '$regex': 'Counter-Str', '$options': 'i' } },
        nReturned: 7,
        executionTimeMillisEstimate: 32,
        works: 45003,
        advanced: 7,
        needTime: 44995,
      }
    }
  }
}, rejectedPlans: []
},
executionStats: {
  executionSuccess: true,
  nReturned: 7,
  executionTimeMillis: 39,
  totalKeysExamined: 45002,
  totalDocsExamined: 7,
  executionStages: {
    isCached: false,
    stage: 'FETCH',
    nReturned: 7,
    executionTimeMillisEstimate: 32,
    works: 45003,
    advanced: 7,
    needTime: 44995,
  }
}
}
```

Zrzut ekranu nr 40: zapytanie z wykorzystaniem indeksu zajęło 39ms

Zapytanie z użyciem indeksu okazało się wolniejsze.

Teraz podmienię indeks na indeks do wyszukiwania tekstowych typu *text*:

```
zsbd> db.games.getIndex()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1' }
]
zsbd> db.games.dropIndex("name_1")
{ nIndexesWas: 2, ok: 1 }
zsbd> db.games.createIndex({ "name": "text" })
name_text
```

Zrzut ekranu nr 41: założenie indeksu typu *text* na pole *name*

**Testy indeksu typu text:**

```
zsbd> db.games.find({ $text: { $search: "counter-str" } }).explain("executionStats")
```

```
    executionStats: {
        executionSuccess: true,
        nReturned: 17,
        executionTimeMillis: 0,
        totalKeysExamined: 17,
        totalDocsExamined: 17,
        executionStages: {
            isCached: false,
            stage: 'TEXT_MATCH',
            nReturned: 17,
            executionTimeMillisEstimate: 0,
            works: 19,
            advanced: 17,
            needTime: 1,
            needYield: 0,
            saveState: 0,
            restoreState: 0,
            isEOF: 1,
            indexPrefix: {},
            indexName: 'name_text',
            parsedTextQuery: {
                terms: [ 'counter', 'str' ],
                negatedTerms: [],
                phrases: [],
                negatedPhrases: []
            },
            textIndexVersion: 3,
            docsRejected: 0,
            inputStage: {
                stage: 'FETCH',
                nReturned: 17,
                executionTimeMillisEstimate: 0,
            }
        }
    }
```

Zrzut ekranu nr 42: rezultat z wykorzystaniem tego dedykowanego wyszukiwania i indeksu jest natychmiastowe i zdaje się być rewelacyjnym rozwiązaniem

### Testy indeksu kompozytowego (name, date):

W przypadku tych testów pozwolę sobie już pominąć zrzuty ekranu i wstawię wyłącznie wyniki pomiarów. Pomiary bez indeksu:

```
zsbd> db.games.find({ date: { $gte: ISODate("2019-01-01") }, name: { $regex: "craft", $options: "i" } }).explain("executionStats")
Zrzut ekranu nr 43: czas wyszukiwania to 28ms
```

```
zsbd> db.games.find({ date: { $gte: ISODate("2019-01-01") } }).sort({ date: 1, name: 1 }).explain("executionStats")
Zrzut ekranu nr 44: czas wyszukiwania to 80ms
```

```
zsbd> db.games.createIndex({ date: 1, name: 1 });
date_1_name_1
```

Zrzut ekranu nr 45: założenie indeksu kompozytowego na (name, date)

Wyniki po założeniu indeksu:

```
zsbd> db.games.find({ date: { $gte: ISODate("2019-01-01") }, name: { $regex: "craft", $options: "i" } }).explain("executionStats")
Zrzut ekranu nr 46: czas wyszukiwania to 12 ms
```

```
zsbd> db.games.find({ date: { $gte: ISODate("2019-01-01") } }).sort({ date: 1, name: 1 }).explain("executionStats")
Zrzut ekranu nr 47: czas wyszukiwania to 11 ms
```

Na bazie moich eksperymentów stwierdzam, że odpowiednio założone indeksy w MongoDB znaczco przyspieszają wyszukiwanie danych. Mechanika indeksu - tworzy struktury umożliwiające szybki dostęp do dokumentów na podstawie wartości kluczy, co zdaje się działać podobnie jak w relacyjnych BD. W odróżnieniu od indeksów w bazach relacyjnych, MongoDB musi uwzględnić specyfikę dokumentowej struktury danych, w tym indeksowanie wbudowanych pól czy dynamicznie zmieniających się schematów. Bardzo ciekawym okazał się typ indeksu *text*, który zdaje się być rewelacyjnym rozwiązaniem do przeszukiwań tekstowych. Poza przetestowanymi przeze mnie typami indeksów MongoDB daje również kilka innych specyficznych indeksów np. dotyczących danych geolokacyjnych (<https://www.mongodb.com/docs/manual/core/index-types/>).

### Mechanizm transakcji:

MongoDB posiada dwa bazowe rodzaje transakcji:

- Jednodokumentowe:

SZBD gwarantuje nam atomową operację na jednym dokumencie tzn. cały dokument zostanie przetworzony (dodany/zaktualizowany/usunięty) albo operacja zostanie wycofana.

- Wielodokumentowe:

Transakcja na wielu dokumentach jednocześnie, MongoDB traktuje całą sekwencję atomowo jako jednostka pracy (*ang. unit of work*) – albo wszystkie zmiany zostaną zaaplikowane albo wszystkie zrollbackowane.

**Przykład:** Dodanie publishera, a następnie nowej gry z powiązanym nowo utworzonym publisherem.

Zadanie okazało się problematyczne, ponieważ moja konfiguracja okazała się błędna – tzn., aby transakcje działały instancja MongoDB musi być uruchomiona jako „replica set” albo być zamontowana w klastrze. Bazowa konfiguracja nie zakłada tego. Po długiej walce udało mi się dokopać do bardzo cennego wpisu na blogu dragonflydb (<https://www.dragonflydb.io/faq/mongodb-windows-replica-set>), który pomógł mi rozwiązać problem.

Zmodyfikowałem plik konfiguracyjny `mongod.cfg` uzupełniając go o wskazanie, że serwer ma być uruchomiony jako replica set. Następnie uruchomiłem demona mongod wskazując na plik konfiguracyjny. Oczywiście potrzebne były uprawnienia administratora, by zmodyfikować wpis konfiguracyjny i uruchomić proces.

PS C:\Users\Michał> `mongod --config 'C:\Program Files\MongoDB\Server\8.0\bin\mongod.cfg'`  
Zrzut ekranu nr 48: uruchomienie serwera skonfigurowanego pod replica set

```
test> use zsbdb
switched to db zsbdb
zsbdb> rs.initiate()
{
  info2: 'no configuration specified. Using a default configuration for the set',
  me: '127.0.0.1:27017',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1732828235, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAA='),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1732828235, i: 1 })
}
```

Zrzut ekranu nr 49: poprawnie uruchomiona replica

```
rs0 [direct: secondary] zsbdb> var session = db.getMongo().startSession();
rs0 [direct: primary] zsbdb>
rs0 [direct: primary] zsbdb> session.startTransaction();
rs0 [direct: primary] zsbdb>
rs0 [direct: primary] zsbdb> try {
...   var publishersCollection = session.getDatabase("zsbdb").publishers;
...   var gamesCollection = session.getDatabase("zsbdb").games;
...
...   var newPublisher = {
...     name: 'Test publisher',
...     dateOfFounding: ISODate("1932-11-06"),
...     ceo: 'Anthony Golden',
...     country: 'Uganda'
...   };
...   var publisherInsertResult = publishersCollection.insertOne(newPublisher);
...   var publisherId = publisherInsertResult.insertedId;
...
...   var newGame = {
...     name: "Test game in transaction",
...     date: ISODate("2023-07-15T00:00:00Z"),
...     full_desc: {
...       sort: "game",
...       desc: "A game!"
...     },
...     categories: ["Sci-fi", "RPG", "Action", "Space Exploration"],
...     publisherId: publisherId
...   };
...   gamesCollection.insertOne(newGame);
...
...   session.commitTransaction();
...   console.log("Transaction committed successfully.");
... } catch (error) {
...   session.abortTransaction();
...   console.error("Transaction aborted due to:", error);
...   throw error;
... } finally {
...   session.endSession();
... }
Transaction committed successfully.
```

Zrzut ekranu nr 50: poprawnie wykonana transakcja

```
rs0 [direct: primary] zsbdb> db.games.findOne({name: 'Test game in transaction'})
{
  _id: ObjectId('6748dc5cd3302705b00d8191'),
  name: 'Test game in transaction',
  date: ISODate('2023-07-15T00:00:00.000Z'),
  full_desc: { sort: 'game', desc: 'A game!' },
  categories: [ 'Sci-fi', 'RPG', 'Action', 'Space Exploration' ],
  publisherId: ObjectId('6748dc5cd3302705b00d8190')
}
rs0 [direct: primary] zsbdb> db.publishers.findOne({_id: ObjectId('6748dc5cd3302705b00d8190')})
{
  _id: ObjectId('6748dc5cd3302705b00d8190'),
  name: 'Test publisher',
  dateOfFounding: ISODate('1932-11-06T00:00:00.000Z'),
  ceo: 'Anthony Golden',
  country: 'Uganda'
}
```

Zrzut ekranu nr 51, 52: weryfikacja rezultatu transakcji

## Zaawansowane funkcjonalności MongoDB:

Zaprezentuję trzy zaawansowane funkcjonalności MongoDB na konkretnych przykładach pokazując ich użyteczność oraz praktyczne wykorzystanie. Funkcje i prezentacja wyników w pliku *advancedMongoFunctions.js*.

### 1. Map/Reduce

To zaawansowana funkcja, która przetwarza dane w dwóch krokach.

Etap Map: funkcja mapująca napisana w JS stosowana jest do każdego dokumentu podanej kolekcji, tworząc pary key-value na podstawie określonych kryteriów.

Etap Reduce: funkcja redukująca łączy wartości o tym samym kluczu tworząc zagregowane wyniki końcowe (np. min/max, średnia etc.)

**Przykład 1:** wyliczenie częstotliwości występowania tag'ów w grach (pozwala nam pokazać jakiego typu gry są najpopularniejsze – jest ich najwięcej):

```
zsbdb> db.games.mapReduce(
...   function () {
...     if (this.popu_tags && Array.isArray(this.popu_tags)) {
...       this.popu_tags.forEach(tag => emit(tag, 1));
...     }
...   },
...   function (key, values) {
...     return Array.sum(values);
...   },
...   {
...     out: "tagCounts"
...   }
... );
{ result: 'tagCounts', ok: 1 }
zsbdb> db.tagCounts.find().sort({ value: -1 }).limit(5);
[
  { _id: 'Indie', value: 23014 },
  { _id: 'Action', value: 18264 },
  { _id: 'Adventure', value: 15792 },
  { _id: 'Casual', value: 13566 },
  { _id: 'Strategy', value: 10283 }
]
```

Zrzut ekranu nr 53: rezultat wyliczenia najpopularniejszych tag'ów

**Przykład 2:** zliczanie, ile gier wychodziło rok do roku:

```
zsbd> db.games.mapReduce(
...     function () {
...         if (this.date) {
...             const year = this.date.getFullYear();
...             emit(year, 1);
...         }
...     },
...     function (key, values) {
...         return Array.sum(values);
...     },
...     {
...         out: "games_by_year"
...     }
... );
{ result: 'games_by_year', ok: 1 }
zsbd> db.games_by_year.find().sort({ _id: 1 });
[
    { _id: 1970, value: 1 },
    { _id: 1981, value: 1 },
    { _id: 1983, value: 1 },
    { _id: 1984, value: 2 },
    { _id: 1985, value: 1 },
    { _id: 1986, value: 3 },
    { _id: 1987, value: 6 },
    { _id: 1988, value: 7 },
    { _id: 1989, value: 11 },
    { _id: 1990, value: 17 },
    { _id: 1991, value: 12 },
    { _id: 1992, value: 12 },
    { _id: 1993, value: 26 },
    { _id: 1994, value: 35 },
    { _id: 1995, value: 35 },
    { _id: 1996, value: 46 },
    { _id: 1997, value: 46 },
    { _id: 1998, value: 56 },
    { _id: 1999, value: 55 },
    { _id: 2000, value: 49 }
]
Type "it" for more
```

**Zrzut ekranu nr 54:** rezultat zliczania gier w kolejnych latach

W funkcji map/reduce widzę potencjał zastosowania w wyliczaniu statystyk mogących służyć do analizy np. badanie charakterystyki rynku/wydawców gier – największa/najdroższa gra w danym roku, średnia cena gier z gatunku X etc.

## 2. Capped Collections

Funkcjonalność pozwalająca na tworzenie kolekcji o ograniczonym rozmiarze. Działają podobnie do buforów. Kiedy kolekcja zapełni całą zaalokowaną przestrzeń, zacznie nadpisywać najstarsze wpisy. Twórcy MongoDB zalecają użycie tej funkcjonalności do składowania logów serwerowych. Chociaż sugerują, że TTL (Time To Live) indexes stanowią lepszą alternatywę, która jest bardziej elastyczna i wydajniejsza: (<https://www.mongodb.com/docs/manual/core/capped-collections/>)

**Przykład:** przygotuję kolekcję na logi, gdzie będziemy składować informacje o operacjach wykonanych na grach. Zasymuluję to przez dodanie prostych wpisów ręcznie, a rozmiar ograniczę do 3 (tylko w celu demonstracji nadpisywania, w rzeczywistości rozmiar powinien być znacznie większy i rozsądnie dopasowany tak by składować operacje powiedzmy z ostatnich 3 miesięcy).

```
zsbd> db.createCollection("gameLogs", {capped: true, size: 10000, max: 3})
{ ok: 1 }
zsbd> db.gameLogs.insertMany([{details: "Game with id: 1 was added to store"}, {details: "Game with id: 2 was added to store"}, ...])
{
  acknowledged: true,
  insertedIds: [
    '_0': ObjectId('674af72fe549d8cb860d8190'),
    '_1': ObjectId('674af72fe549d8cb860d8191'),
    '_2': ObjectId('674af72fe549d8cb860d8192')
  ]
}
zsbd> db.gameLogs.find()
[ {
  _id: ObjectId('674af72fe549d8cb860d8190'),
  details: 'Game with id: 1 was added to store'
},
{
  _id: ObjectId('674af72fe549d8cb860d8191'),
  details: 'Game with id: 2 was added to store'
},
{
  _id: ObjectId('674af72fe549d8cb860d8192'),
  details: 'Game with id: 1 was removed from store'
} ]
```

Zrzut ekranu nr 55: utworzenie capped colelction na logi oraz dodanie 3 wpisów

```
zsbd> db.gameLogs.insertOne({details: "Game with id: 3 was added to store"})
{
  acknowledged: true,
  insertedId: ObjectId('674af759e549d8cb860d8193')
}
zsbd> db.gameLogs.find()
[ {
  _id: ObjectId('674af72fe549d8cb860d8191'),
  details: 'Game with id: 2 was added to store'
},
{
  _id: ObjectId('674af72fe549d8cb860d8192'),
  details: 'Game with id: 1 was removed from store'
},
{
  _id: ObjectId('674af759e549d8cb860d8193'),
  details: 'Game with id: 3 was added to store'
} ]
```

Zrzut ekranu nr 56: dodanie kolejnego wpisu (poprawne nadpisanie przez MongoDB najstarszego log'a)

### 3. Zapytania ad-hoc

Jest to typ zapytań budowany dynamicznie na bazie podanych zmiennych bez wcześniejszego definiowania struktury zapytań. Umożliwiają elastyczne pobieranie danych na podstawie zmieniających się kryteriów.

**Przykład:** Prosty przykład poprzez zapytanie o grę droższą niż i tańszą niż podane przedziały, zwrócona zdefiniowana liczba gier

```
zsbd> var req = {query: {
  ...   afterDate: new Date("2018-01-01"),
  ...   beforeDate: new Date("2020-01-01"),
  ...   count: 5
  ... }};
```

Zrzut ekranu nr 57: przygotowanie dynamicznie obiektu, który może przychodzić z request'a

```
zsbd> var games = db.games.find({
...   date: {
...     $gte: req.query.afterDate,
...     $lt: req.query.beforeDate
...   },
...   {
...     _id: 1,
...     name: 1,
...     date: 1,
...     price: 1
...   }
... }).limit(req.query.count);
```

Zrzut ekranu nr 58: zapytanie ad-hoc

```
zsbd> games
[
  {
    _id: ObjectId('67476414a61317cb22cdde0a'),
    date: ISODate('2018-01-01T23:00:00.000Z'),
    price: 149,
    name: 'BASIC8'
  },
  {
    _id: ObjectId('67476414a61317cb22ce02de'),
    date: ISODate('2018-01-01T23:00:00.000Z'),
    price: 299,
    name: 'Bit-Boom'
  },
  {
    _id: ObjectId('67476415a61317cb22ce5c39'),
    date: ISODate('2018-01-01T23:00:00.000Z'),
    price: 99,
    name: 'Comit the Astrodian 3'
  },
  {
    _id: ObjectId('67476414a61317cb22cddeeb'),
    date: ISODate('2018-01-01T23:00:00.000Z'),
    price: 599,
    name: 'LINCH'
  },
  {
    _id: ObjectId('67476414a61317cb22ce2069'),
    date: ISODate('2018-01-01T23:00:00.000Z'),
    price: 99,
    name: 'Mesozoica'
  }
]
```

Zrzut ekranu nr 59: rezultat zapytania

## Wnioski:

Wykorzystanie bazy nierelacyjnej w mojej problematyce różni się od użycia tradycyjnej bazy relacyjnej i oferuje kilka ciekawych zalet.

- **Elastyczność schematu** – ogromna zaleta, zwłaszcza gdy dane są niepełne (powszechny problem) lub obiekty różnią się od siebie (np. różne informacje opcjonalne). W bazie relacyjnej wymagałoby to tworzenia dużych/wielu tabel, z których większość pól byłaby NULL'ami, co prowadzi do marnowania potencjału i niewydajności.
- **Skalowalność i wydajność** – pomimo całkiem dużej kolekcji (kilkadziesiąt tysięcy rekordów) wyszukiwanie danych było szybkie (kilkadziesiąt ms), a z użyciem rozsądnego indeksów bardzo szybkie i wydajne (czasy poniżej 10ms).
- **Zapytania i aggregation framework** – MongoDB oferuje zaawansowane i bardzo elastyczne możliwości wykonywania zapytań, które pozwalają na pisanie intuicyjnego kodu wprost bez kombinowania „na około”. Przetwarzanie potokowe z użyciem agregacji zdaje się być potężnym narzędziem pozwalającym na wykonywanie wszystkich potrzebnych, zaawansowanych zapytań.
- **Zaawansowane funkcje** – MongoDB oferuje wiele funkcjonalności dedykowanych specyficzny wymaganiom. Ciekawe i praktycznie wykorzystywane w rzeczywistych projektach wydają mi się: przeszukiwanie pełno tekstowe oraz funkcje związane z geolokalizacją (choć nie miałem okazji ich użyć, to dokumentacja wskazuje, że oferują duży potencjał i gotowe rozwiązania, które w innych technologiach musielibyśmy pisać od zera, potencjalnie popełniając błędy lub tworząc niewydajne rozwiązanie).
- **Zasilanie bazy:** Proces zasilania bazy jest bardzo prosty, co uważam za dużą zaletę. Dzięki prostemu poleceniu możemy łatwo wczytywać dane z plików, automatycznie generując identyfikatory i indeksując dokumenty na \_id.

Moje ogólne wrażenia po realizacji całego projektu są bardzo pozytywne i traktuję to ćwiczenie jako cenne doświadczenie oraz poszerzenie horyzontów. Projekt pozwolił mi lepiej zrozumieć możliwości MongoDB w praktycznych zastosowaniach.

## Źródła:

- <https://stackoverflow.com/questions/15171622/mongoimport-of-json-file>
- <https://stackoverflow.com/questions/31071999/date-comparison-in-mongodb>
- <https://www.mongodb.com/docs/manual/reference/operator/aggregation/lookup/>
- <https://www.mongodb.com/docs/manual/reference/operator/aggregation/project/>
- <https://www.mongodb.com/docs/drivers/node/current/fundamentals/transactions/>
- <https://stackoverflow.com/questions/51461952/mongodb-v4-0-transaction-mongoerror-transaction-numbers-are-only-allowed-on-a>
- <https://www.dragonflydb.io/faq/mongodb-windows-replica-set>
- <https://www.mongodb.com/resources/basics/databases/database-index>
- <https://www.mongodb.com/docs/manual/core/indexes/index-types/>
- <https://www.mongodb.com/resources/products/capabilities/features>
- <https://www.geeksforgeeks.org/mongodb-map-reduce/>
- <https://www.mongodb.com/docs/manual/core/capped-collections/>
- <https://www.mongodb.com/docs/manual/core/capped-collections/>
- <https://stackoverflow.com/questions/70188585/are-all-queries-in-mongodb-ad-hoc>