Dokumentowe bazy danych – MongoDB

Ćwiczenie/zadanie

Imiona i nazwiska autorów:

- Szymon Migas
- Dawid Żak

Odtwórz z backupu bazę north0

```
mongorestore --nsInclude='north0.*' ./dump/
```

```
use north0
```

Ponieważ pracowaliśmy na klastrze zdalnym, poniżej komenda, której użyliśmy:

```
./bin/mongorestore --uri "mongodb+srv://smigas:<####>@cluster-1.0pcsnxc.mongodb.net/north0" --nsInclude='north0.*' ./north0
```

Zadanie 1 - operacje wyszukiwania danych, przetwarzanie dokumentów

a)

stwórz kolekcję `OrdersInfo` zawierającą następujące dane o zamówieniach

- pojedynczy dokument opisuje jedno zamówienie

```
[
  {
    "_id": ...

    OrderID": ... numer zamówienia

    "Customer": {  ... podstawowe informacje o kliencie skladającym
      "CustomerID": ... identyfikator klienta
      "CompanyName": ... nazwa klienta
      "City": ... miasto
      "Country": ... kraj
    },

    "Employee": {  ... podstawowe informacje o pracowniku obsługującym zamówienie
      "EmployeeID": ... idntyfikator pracownika
      "FirstName": ... imie
      "LastName": ... nazwisko
      "Title": ... stanowisko

    },

    "Dates": {
       "OrderDate": ... data złożenia zamówienia
       "RequiredDate": data wymaganej realizacji
    }

    "Orderdetails": [  ... pozycje/szczegóły zamówienia – tablica takich pozycji
      {
        "UnitPrice": ... cena
        "Quantity": ... liczba sprzedanych jednostek towaru
        "Discount": ... zniżka
        "Value": ... wartośc pozycji zamówienia
        "product": { ... podstawowe informacje o produkcie
          "ProductID": ... identyfikator produktu
          "ProductName": ... nazwa produktu
          "QuantityPerUnit": ... opis/opakowannie
          "CategoryID": ... identyfikator kategorii do której należy produkt
          "CategoryName" ... nazwę tej kategorii
        },
      },
      ...
    ],

    "Freight": ... opłata za przesyłkę
    "OrderTotal"  ... sumaryczna wartosc sprzedanych produktów

    "Shipment" : {  ... informacja o wysyłce
        "Shipper": { ... podstawowe inf o przewoźniku
           "ShipperID":
            "CompanyName":
        }
        ... inf o odbiorcy przesyłki
        "ShipName": ...
        "ShipAddress": ...
        "ShipCity": ...
        "ShipCountry": ...
    }
  }
]
```

b)

stwórz kolekcję `CustomerInfo` zawierającą następujące dane kazdym klencie

- pojedynczy dokument opisuje jednego klienta

```
[
  {
    "_id": ...

    "CustomerID": ... identyfikator klienta
    "CompanyName": ... nazwa klienta
    "City": ... miasto
    "Country": ... kraj

        "Orders": [ ... tablica zamówień klienta o strukturze takiej jak w punkcie a) (oczywiście bez informacji o kliencie)

        ]


]
```

c)

Napisz polecenie/zapytanie: Dla każdego klienta pokaż wartość zakupionych przez niego produktów z kategorii 'Confections' w 1997r

- Spróbuj napisać to zapytanie wykorzystując
  - oryginalne kolekcje (`customers, orders, orderdertails, products, categories`)
  - kolekcję `OrderInfo`
  - kolekcję `CustomerInfo`
- porównaj zapytania/polecenia/wyniki

```
[
  {
    "_id":

    "CustomerID": ... identyfikator klienta
    "CompanyName": ... nazwa klienta
        "ConfectionsSale97": ... wartość zakupionych przez niego produktów z kategorii 'Confections'  w 1997r

  }
]
```

d)

Napisz polecenie/zapytanie: Dla każdego klienta poaje wartość sprzedaży z podziałem na lata i miesiące
Spróbuj napisać to zapytanie wykorzystując
- oryginalne kolekcje (`customers, orders, orderdertails, products, categories`)
- kolekcję `OrderInfo`
- kolekcję `CustomerInfo`

- porównaj zapytania/polecenia/wyniki

```
[
  {
    "_id":

    "CustomerID": ... identyfikator klienta
    "CompanyName": ... nazwa klienta

        "Sale": [ ... tablica zawierająca inf o sprzedazy
            {
            "Year":  ....
            "Month": ....
            "Total": ...
            }
            ...
        ]
  }
]
```

e)

Załóżmy że pojawia się nowe zamówienie dla klienta 'ALFKI', zawierające dwa produkty 'Chai' oraz "Ikura"

- pozostałe pola w zamówieniu (ceny, liczby sztuk prod, inf o przewoźniku itp. możesz uzupełnić wg własnego uznania)
  Napisz polecenie które dodaje takie zamówienie do bazy
- aktualizując oryginalne kolekcje `orders` , `orderdetails`
- aktualizując kolekcję `OrderInfo`
- aktualizując kolekcję `CustomerInfo`

Napisz polecenie

- aktualizując oryginalną kolekcję orderdetails`
- aktualizując kolekcję `OrderInfo`
- aktualizując kolekcję `CustomerInfo`

f)

Napisz polecenie które modyfikuje zamówienie dodane w pkt e) zwiększając zniżkę o 5% (dla każdej pozycji tego zamówienia)

Napisz polecenie

- aktualizując oryginalną kolekcję `orderdetails`
- aktualizując kolekcję `OrderInfo`
- aktualizując kolekcję `CustomerInfo`

UWAGA:
W raporcie należy zamieścić kod poleceń oraz uzyskany rezultat, np wynik polecenia `db.kolekcka.fimd().limit(2)` lub jego fragment

Zadanie 1 - rozwiązanie

Podpunkt a)

Za pomocą agregacji `aggregate` oraz operatora `$lookup` połączyliśmy potrzebne kolekcje, aby uzyskać odpowiednią strukturę dla kolekcji `ordersInfo`:

```
db.orders.aggregate([
  {
    $lookup: {
      from: "customers",
      localField: "CustomerID",
      foreignField: "CustomerID",
      as: "customerData"
    }
  },
  {
    $unwind: "$customerData"
  },
  {
    $lookup: {
      from: "employees",
      localField: "EmployeeID",
      foreignField: "EmployeeID",
      as: "employeeData"
    }
  },
  {
    $unwind: "$employeeData"
  },
  {
    $lookup: {
      from: "shippers",
      localField: "ShipVia",
      foreignField: "ShipperID",
      as: "shipperData"
    }
  },
  {
    $unwind: "$shipperData"
  },
  {
    $lookup: {
      from: "orderdetails",
      let: { orderID: "$OrderID" },
      pipeline: [
        {
          $match: {
            $expr: { $eq: ["$OrderID", "$$orderID"] }
          }
        },
        {
          $lookup: {
            from: "products",
            localField: "ProductID",
            foreignField: "ProductID",
            as: "productData"
          }
        },
        {
          $unwind: "$productData"
        },
        {
          $lookup: {
            from: "categories",
            localField: "productData.CategoryID",
```

```
            foreignField: "CategoryID",
            as: "categoryData"
          }
        },
        {
          $unwind: "$categoryData"
        },
        {
          $project: {
            _id: 0,
            UnitPrice: 1,
            Quantity: 1,
            Discount: 1,
            Value: {
              $multiply: [
                { $subtract: [1, "$Discount"] },
                 {$multiply: ["$UnitPrice", "$Quantity"] }] },
            product: {
              ProductID: "$ProductID",
              ProductName: "$productData.ProductName",
              QuantityPerUnit: "$productData.QuantityPerUnit",
              CategoryID: "$productData.CategoryID",
              CategoryName: "$categoryData.CategoryName"
            }
          }
        }
      ],
      as: "Orderdetails"
    }
  },
  {
    $project: {
      _id: 1,
      OrderID: 1,
      Customer: {
        CustomerID: "$customerData.CustomerID",
        CompanyName: "$customerData.CompanyName",
        City: "$customerData.City",
        Country: "$customerData.Country"
      },
      Employee: {
        EmployeeID: "$employeeData.EmployeeID",
        FirstName: "$employeeData.FirstName",
        LastName: "$employeeData.LastName",
        Title: "$employeeData.Title"
      },
      Dates: {
        OrderDate: "$OrderDate",
        RequiredDate: "$RequiredDate"
      },
      Orderdetails: 1,
      Freight: "$Freight",
      OrderTotal: {
        $reduce: {
          input: "$Orderdetails",
          initialValue: 0,
          in: { $add: ["$$value", "$$this.Value"] }
        }
      },
      Shipment: {
```

```
        Shipper: {
          ShipperID: "$shipperData.ShipperID",
          CompanyName: "$shipperData.CompanyName"
        },
        ShipName: "$ShipName",
        ShipAddress: "$ShipAddress",
        ShipCity: "$ShipCity",
        ShipCountry: "$ShipCountry"
      }
    }
  },
  {
    $out: "ordersInfo"
  }
]);
```

Wynik polecenia: `db.ordersInfo.find.limit(1)`:

```
[
  {
    "_id": {"$oid": "63a060b9bb3b972d6f4e2005"},
    "Customer": {
      "CustomerID": "DUMON",
      "CompanyName": "Du monde entier",
      "City": "Nantes",
      "Country": "France"
    },
    "Dates": {
      "OrderDate": {"$date": "1996-09-20T00:00:00.000Z"},
      "RequiredDate": {"$date": "1996-10-04T00:00:00.000Z"}
    },
    "Employee": {
      "EmployeeID": 1,
      "FirstName": "Nancy",
      "LastName": "Davolio",
      "Title": "Sales Representative"
    },
    "Freight": 24.69,
    "OrderID": 10311,
    "OrderTotal": 268.79999999999995,
    "Orderdetails": [
      {
        "UnitPrice": 28.8,
        "Quantity": 7,
        "Discount": 0,
        "Value": 201.6,
        "product": {
          "ProductID": 69,
          "ProductName": "Gudbrandsdalsost",
          "QuantityPerUnit": "10 kg pkg.",
          "CategoryID": 4,
          "CategoryName": "Dairy Products"
        }
      },
      {
        "UnitPrice": 11.2,
        "Quantity": 6,
        "Discount": 0,
        "Value": 67.19999999999999,
        "product": {
          "ProductID": 42,
          "ProductName": "Singaporean Hokkien Fried Mee",
          "QuantityPerUnit": "32 - 1 kg pkgs.",
          "CategoryID": 5,
          "CategoryName": "Grains/Cereals"
        }
      }
    ],
    "Shipment": {
      "Shipper": {
        "ShipperID": 3,
        "CompanyName": "Federal Shipping"
      },
      "ShipName": "Du monde entier",
      "ShipAddress": "67, rue des Cinquante Otages",
      "ShipCity": "Nantes",
      "ShipCountry": "France"
```

```
    }
  }
]
```

Podpunkt b)

Następnie tak jak w podpunkcie A, stworzyliśmy odpowiednie zapytanie agregacyjne, aby wypełnić kolekcję `customerInfo` danymi.

```
db.customers.aggregate([
  {
    $lookup: {
      from: "ordersInfo",
      let: { customerID: "$CustomerID" },
      pipeline: [
        {
          $match: {
            $expr: { $eq: ["$Customer.CustomerID", "$$customerID"] }
          }
        },
        {
          $project: {
            _id: 0,
            OrderID: 1,
            Employee: 1,
            Dates: 1,
            Orderdetails: 1,
            Freight: 1,
            OrderTotal: 1,
            Shipment: 1
          }
        }
      ],
      as: "Orders"
    }
  },
  {
    $project: {
      _id: 1,
      CustomerID: 1,
      CompanyName: 1,
      City: 1,
      Country: 1,
      Orders: 1
    }
  },
  {
    $out: "customerInfo"
  }
]);
```

Wynik zapytania `db.customerInfo.find.limit(1)`:

```
[
  {
    "_id": {"$oid": "63a05cdfbb3b972d6f4e09aa"},
    "City": "Portland",
    "CompanyName": "Lonesome Pine Restaurant",
    "Country": "USA",
    "CustomerID": "LONEP",
    "Orders": [
      {
        "OrderID": 10544,
        "Orderdetails": [
          {
            "UnitPrice": 14,
            "Quantity": 7,
            "Discount": 0,
            "Value": 98,
            "product": {
              "ProductID": 67,
              "ProductName": "Laughing Lumberjack Lager",
              "QuantityPerUnit": "24 - 12 oz bottles",
              "CategoryID": 1,
              "CategoryName": "Beverages"
            }
          },
          {
            "UnitPrice": 45.6,
            "Quantity": 7,
            "Discount": 0,
            "Value": 319.2,
            "product": {
              "ProductID": 28,
              "ProductName": "Rössle Sauerkraut",
              "QuantityPerUnit": "25 - 825 g cans",
              "CategoryID": 7,
              "CategoryName": "Produce"
            }
          }
        ],
        "Employee": {
          "EmployeeID": 4,
          "FirstName": "Margaret",
          "LastName": "Peacock",
          "Title": "Sales Representative"
        },
        "Dates": {
          "OrderDate": {"$date": "1997-05-21T00:00:00.000Z"},
          "RequiredDate": {"$date": "1997-06-18T00:00:00.000Z"}
        },
        "Freight": 24.91,
        "OrderTotal": 417.2,
        "Shipment": {
          "Shipper": {
            "ShipperID": 1,
            "CompanyName": "Speedy Express"
          },
          "ShipName": "Lonesome Pine Restaurant",
          "ShipAddress": "89 Chiaroscuro Rd.",
          "ShipCity": "Portland",
          "ShipCountry": "USA"
```

```
      }
    },
    {
      "OrderID": 10665,
      "Orderdetails": [
        {
          "UnitPrice": 53,
          "Quantity": 20,
          "Discount": 0,
          "Value": 1060,
          "product": {
            "ProductID": 51,
            "ProductName": "Manjimup Dried Apples",
            "QuantityPerUnit": "50 - 300 g pkgs.",
            "CategoryID": 7,
            "CategoryName": "Produce"
          }
        },
        {
          "UnitPrice": 55,
          "Quantity": 1,
          "Discount": 0,
          "Value": 55,
          "product": {
            "ProductID": 59,
            "ProductName": "Raclette Courdavault",
            "QuantityPerUnit": "5 kg pkg.",
            "CategoryID": 4,
            "CategoryName": "Dairy Products"
          }
        },
        {
          "UnitPrice": 18,
          "Quantity": 10,
          "Discount": 0,
          "Value": 180,
          "product": {
            "ProductID": 76,
            "ProductName": "Lakkalikööri",
            "QuantityPerUnit": "500 ml",
            "CategoryID": 1,
            "CategoryName": "Beverages"
          }
        }
      ],
      "Employee": {
        "EmployeeID": 1,
        "FirstName": "Nancy",
        "LastName": "Davolio",
        "Title": "Sales Representative"
      },
      "Dates": {
        "OrderDate": {"$date": "1997-09-11T00:00:00.000Z"},
        "RequiredDate": {"$date": "1997-10-09T00:00:00.000Z"}
      },
      "Freight": 26.31,
      "OrderTotal": 1295,
      "Shipment": {
        "Shipper": {
          "ShipperID": 2,
```

```
          "CompanyName": "United Package"
        },
        "ShipName": "Lonesome Pine Restaurant",
        "ShipAddress": "89 Chiaroscuro Rd.",
        "ShipCity": "Portland",
        "ShipCountry": "USA"
      }
    },
    {
      "OrderID": 10662,
      "Orderdetails": [
        {
          "UnitPrice": 12.5,
          "Quantity": 10,
          "Discount": 0,
          "Value": 125,
          "product": {
            "ProductID": 68,
            "ProductName": "Scottish Longbreads",
            "QuantityPerUnit": "10 boxes x 8 pieces",
            "CategoryID": 3,
            "CategoryName": "Confections"
          }
        }
      ],
      "Employee": {
        "EmployeeID": 3,
        "FirstName": "Janet",
        "LastName": "Leverling",
        "Title": "Sales Representative"
      },
      "Dates": {
        "OrderDate": {"$date": "1997-09-09T00:00:00.000Z"},
        "RequiredDate": {"$date": "1997-10-07T00:00:00.000Z"}
      },
      "Freight": 1.28,
      "OrderTotal": 125,
      "Shipment": {
        "Shipper": {
          "ShipperID": 2,
          "CompanyName": "United Package"
        },
        "ShipName": "Lonesome Pine Restaurant",
        "ShipAddress": "89 Chiaroscuro Rd.",
        "ShipCity": "Portland",
        "ShipCountry": "USA"
      }
    },
    {
      "OrderID": 10307,
      "Orderdetails": [
        {
          "UnitPrice": 10,
          "Quantity": 3,
          "Discount": 0,
          "Value": 30,
          "product": {
            "ProductID": 68,
            "ProductName": "Scottish Longbreads",
            "QuantityPerUnit": "10 boxes x 8 pieces",
```

```
            "CategoryID": 3,
            "CategoryName": "Confections"
          }
        },
        {
          "UnitPrice": 39.4,
          "Quantity": 10,
          "Discount": 0,
          "Value": 394,
          "product": {
            "ProductID": 62,
            "ProductName": "Tarte au sucre",
            "QuantityPerUnit": "48 pies",
            "CategoryID": 3,
            "CategoryName": "Confections"
          }
        }
      ],
      "Employee": {
        "EmployeeID": 2,
        "FirstName": "Andrew",
        "LastName": "Fuller",
        "Title": "Vice President, Sales"
      },
      "Dates": {
        "OrderDate": {"$date": "1996-09-17T00:00:00.000Z"},
        "RequiredDate": {"$date": "1996-10-15T00:00:00.000Z"}
      },
      "Freight": 0.56,
      "OrderTotal": 424,
      "Shipment": {
        "Shipper": {
          "ShipperID": 2,
          "CompanyName": "United Package"
        },
        "ShipName": "Lonesome Pine Restaurant",
        "ShipAddress": "89 Chiaroscuro Rd.",
        "ShipCity": "Portland",
        "ShipCountry": "USA"
      }
    },
    {
      "OrderID": 10317,
      "Orderdetails": [
        {
          "UnitPrice": 14.4,
          "Quantity": 20,
          "Discount": 0,
          "Value": 288,
          "product": {
            "ProductID": 1,
            "ProductName": "Chai",
            "QuantityPerUnit": "10 boxes x 20 bags",
            "CategoryID": 1,
            "CategoryName": "Beverages"
          }
        }
      ],
      "Employee": {
        "EmployeeID": 6,
```

```
      "FirstName": "Michael",
      "LastName": "Suyama",
      "Title": "Sales Representative"
    },
    "Dates": {
      "OrderDate": {"$date": "1996-09-30T00:00:00.000Z"},
      "RequiredDate": {"$date": "1996-10-28T00:00:00.000Z"}
    },
    "Freight": 12.69,
    "OrderTotal": 288,
    "Shipment": {
      "Shipper": {
        "ShipperID": 1,
        "CompanyName": "Speedy Express"
      },
      "ShipName": "Lonesome Pine Restaurant",
      "ShipAddress": "89 Chiaroscuro Rd.",
      "ShipCity": "Portland",
      "ShipCountry": "USA"
    }
  },
  {
    "OrderID": 10883,
    "Orderdetails": [
      {
        "UnitPrice": 4.5,
        "Quantity": 8,
        "Discount": 0,
        "Value": 36,
        "product": {
          "ProductID": 24,
          "ProductName": "Guaraná Fantástica",
          "QuantityPerUnit": "12 - 355 ml cans",
          "CategoryID": 1,
          "CategoryName": "Beverages"
        }
      }
    ],
    "Employee": {
      "EmployeeID": 8,
      "FirstName": "Laura",
      "LastName": "Callahan",
      "Title": "Inside Sales Coordinator"
    },
    "Dates": {
      "OrderDate": {"$date": "1998-02-12T00:00:00.000Z"},
      "RequiredDate": {"$date": "1998-03-12T00:00:00.000Z"}
    },
    "Freight": 0.53,
    "OrderTotal": 36,
    "Shipment": {
      "Shipper": {
        "ShipperID": 3,
        "CompanyName": "Federal Shipping"
      },
      "ShipName": "Lonesome Pine Restaurant",
      "ShipAddress": "89 Chiaroscuro Rd.",
      "ShipCity": "Portland",
      "ShipCountry": "USA"
    }
```

```
    },
    {
      "OrderID": 11018,
      "Orderdetails": [
        {
          "UnitPrice": 38,
          "Quantity": 20,
          "Discount": 0,
          "Value": 760,
          "product": {
            "ProductID": 12,
            "ProductName": "Queso Manchego La Pastora",
            "QuantityPerUnit": "10 - 500 g pkgs.",
            "CategoryID": 4,
            "CategoryName": "Dairy Products"
          }
        },
        {
          "UnitPrice": 62.5,
          "Quantity": 10,
          "Discount": 0,
          "Value": 625,
          "product": {
            "ProductID": 18,
            "ProductName": "Carnarvon Tigers",
            "QuantityPerUnit": "16 kg pkg.",
            "CategoryID": 8,
            "CategoryName": "Seafood"
          }
        },
        {
          "UnitPrice": 38,
          "Quantity": 5,
          "Discount": 0,
          "Value": 190,
          "product": {
            "ProductID": 56,
            "ProductName": "Gnocchi di nonna Alice",
            "QuantityPerUnit": "24 - 250 g pkgs.",
            "CategoryID": 5,
            "CategoryName": "Grains/Cereals"
          }
        }
      ],
      "Employee": {
        "EmployeeID": 4,
        "FirstName": "Margaret",
        "LastName": "Peacock",
        "Title": "Sales Representative"
      },
      "Dates": {
        "OrderDate": {"$date": "1998-04-13T00:00:00.000Z"},
        "RequiredDate": {"$date": "1998-05-11T00:00:00.000Z"}
      },
      "Freight": 11.65,
      "OrderTotal": 1575,
      "Shipment": {
        "Shipper": {
          "ShipperID": 2,
          "CompanyName": "United Package"
```

```
        },
        "ShipName": "Lonesome Pine Restaurant",
        "ShipAddress": "89 Chiaroscuro Rd.",
        "ShipCity": "Portland",
        "ShipCountry": "USA"
      }
    },
    {
      "OrderID": 10867,
      "Orderdetails": [
        {
          "UnitPrice": 32.8,
          "Quantity": 3,
          "Discount": 0,
          "Value": 98.39999999999999,
          "product": {
            "ProductID": 53,
            "ProductName": "Perth Pasties",
            "QuantityPerUnit": "48 pieces",
            "CategoryID": 6,
            "CategoryName": "Meat/Poultry"
          }
        }
      ],
      "Employee": {
        "EmployeeID": 6,
        "FirstName": "Michael",
        "LastName": "Suyama",
        "Title": "Sales Representative"
      },
      "Dates": {
        "OrderDate": {"$date": "1998-02-03T00:00:00.000Z"},
        "RequiredDate": {"$date": "1998-03-17T00:00:00.000Z"}
      },
      "Freight": 1.93,
      "OrderTotal": 98.39999999999999,
      "Shipment": {
        "Shipper": {
          "ShipperID": 1,
          "CompanyName": "Speedy Express"
        },
        "ShipName": "Lonesome Pine Restaurant",
        "ShipAddress": "89 Chiaroscuro Rd.",
        "ShipCity": "Portland",
        "ShipCountry": "USA"
      }
    }
  ]
}
]
```


Komentarz

Kluczowe dla wygody uzupełniania kolekcji dokumentami było użycie operatora `$out`, który zbiera zebrane podczas agregacji dokumenty i zapisuje je do podanej kolekcji.

Podpunkt c)

Zapytanie agregacyjne używając jedynie oryginalnych kolekcji w bazie:

```
db.customers.aggregate([
  {
    $lookup: {
      from: "orders",
      localField: "CustomerID",
      foreignField: "CustomerID",
      as: "orders"
    }
  },
  { $unwind: { path: "$orders" } },
  {
    $match: {
      "orders.OrderDate": {
        $gte: new Date("1997-01-01"),
        $lt: new Date("1998-01-01")
      }
    }
  },
  {
    $lookup: {
      from: "orderdetails",
      localField: "orders.OrderID",
      foreignField: "OrderID",
      as: "details"
    }
  },
  { $unwind: { path: "$details"} },
  {
    $lookup: {
      from: "products",
      localField: "details.ProductID",
      foreignField: "ProductID",
      as: "product"
    }
  },
  { $unwind: { path: "$product"} },
  {
    $lookup: {
      from: "categories",
      localField: "product.CategoryID",
      foreignField: "CategoryID",
      as: "category"
    }
  },
  { $unwind: { path: "$category"} },
  {
    $match: {
      "category.CategoryName": "Confections"
    }
  },
  {
    $addFields: {
      value: {
        $multiply: [
          "$details.UnitPrice",
          "$details.Quantity",
          { $subtract: [1, "$details.Discount"] }
        ],
      }
```

```
    }
  },
  {
    $group: {
      _id: "$CustomerID",
      CustomerID: { $first: "$CustomerID" },
      CompanyName: { $first: "$CompanyName" },
      ConfectionsSale97: { $sum: "$value" }
    }
  },
  {
    $addFields: {
      ConfectionsSale97: { $round: ["$ConfectionsSale97", 2] }
    }
  },
  { $sort: { CustomerID: 1 } },

]);
```

Wynik:

```
[
  {
    "_id": "ANTON",
    "CompanyName": "Antonio Moreno Taquería",
    "ConfectionsSale97": 958.93,
    "CustomerID": "ANTON"
  }
]
```

Zapytanie korzystając z nowo utworzonej przez nas kolekcji `orderInfo`

```
db.ordersInfo.aggregate([

  {
    $match: {
      "Dates.OrderDate": {
        $gte: new Date("1997-01-01"),
        $lt: new Date("1998-01-01")
      }
    }
  },


  { $unwind: "$Orderdetails" },


  {
    $match: {
      "Orderdetails.product.CategoryName": "Confections"
    }
  },


  {
    $group: {
      _id: "$Customer.CustomerID",
      CustomerID: { $first: "$Customer.CustomerID" },
      CompanyName: { $first: "$Customer.CompanyName" },
      ConfectionsSale97: { $sum: "$Orderdetails.Value" }
    }
  },

  {
    $addFields: {
      ConfectionsSale97: { $round: ["$ConfectionsSale97", 2] }
    }
  },


  { $sort: { CustomerID: 1 } }
]);
```

Wynik:

```
[
  {
    "_id": "ANTON",
    "CompanyName": "Antonio Moreno Taquería",
    "ConfectionsSale97": 958.93,
    "CustomerID": "ANTON"
  }
]
```

Oraz zapytanie tworzone przy użyciu kolekcji `customerInfo`

```
db.customerInfo.aggregate([

  { $unwind: "$Orders" },


  {
    $match: {
      "Orders.Dates.OrderDate": {
        $gte: new Date("1997-01-01"),
        $lt: new Date("1998-01-01")
      }
    }
  },


  { $unwind: "$Orders.Orderdetails" },


  {
    $match: {
      "Orders.Orderdetails.product.CategoryName": "Confections"
    }
  },


  {
    $group: {
      _id: "$CustomerID",
      CustomerID: { $first: "$CustomerID" },
      CompanyName: { $first: "$CompanyName" },
      ConfectionsSale97: { $sum: "$Orders.Orderdetails.Value" }
    }
  },

    {
    $addFields: {
      ConfectionsSale97: { $round: ["$ConfectionsSale97", 2] }
    }
  },

  { $sort: { CustomerID: 1 } }
]);
```

Wynik:

```
[
  {
    "_id": "ANTON",
    "CompanyName": "Antonio Moreno Taquería",
    "ConfectionsSale97": 958.93,
    "CustomerID": "ANTON"
  }
]
```

Podpunkt d)

Następnie stworzyliśmy trzy zapytania, dzięki którym uzyskamy informację o łącznej kwocie zamówień klientów z podziałem na lata i pieniądze

Zapytanie z użyciem oryginalnych kolekcji:

```
db.customers.aggregate([

  {
    $lookup: {
      from: "orders",
      localField: "CustomerID",
      foreignField: "CustomerID",
      as: "orders"
    }
  },

  { $unwind: { path: "$orders", preserveNullAndEmptyArrays: false } },


  {
    $lookup: {
      from: "orderdetails",
      localField: "orders.OrderID",
      foreignField: "OrderID",
      as: "details"
    }
  },

  { $unwind: { path: "$details", preserveNullAndEmptyArrays: false } },


  {
    $addFields: {
      year: { $year: "$orders.OrderDate" },
      month: { $month: "$orders.OrderDate" },
      saleValue: {
        $multiply: [
          "$details.UnitPrice",
          "$details.Quantity",
          { $subtract: [1, "$details.Discount"] }
        ]
      }
    }
  },


  {
    $group: {
      _id: {
        customerID: "$CustomerID",
        year: "$year",
        month: "$month"
      },
      Total: { $sum: "$saleValue" },
      CompanyName: { $first: "$CompanyName" }
    }
  },


  {
    $sort: {
      "_id.year": 1,
      "_id.month": 1
    }
```

```
  },


  {
    $group: {
      _id: "$_id.customerID",
      CustomerID: { $first: "$_id.customerID" },
      CompanyName: { $first: "$CompanyName" },
      Sale: {
        $push: {
          Year: "$_id.year",
          Month: "$_id.month",
          Total: { $round: ["$Total", 2] }
        }
      }
    }
  },


  { $sort: { CustomerID: 1 } }
]);
```

Wynik:

```
[
  {
    "_id": "ALFKI",
    "CompanyName": "Alfreds Futterkiste",
    "CustomerID": "ALFKI",
    "Sale": [
      {
        "Year": 1997,
        "Month": 8,
        "Total": 814.5
      },
      {
        "Year": 1997,
        "Month": 10,
        "Total": 1208
      },
      {
        "Year": 1998,
        "Month": 1,
        "Total": 845.8
      },
      {
        "Year": 1998,
        "Month": 3,
        "Total": 471.2
      },
      {
        "Year": 1998,
        "Month": 4,
        "Total": 933.5
      },
      {
        "Year": 2025,
        "Month": 5,
        "Total": 401.1
      }
    ]
  }
]
```


Zapytanie wykorzystujące tabelę `orderInfo`:

```
db.ordersInfo.aggregate([

  { $unwind: "$Orderdetails" },


  {
    $addFields: {
      year: { $year: "$Dates.OrderDate" },
      month: { $month: "$Dates.OrderDate" }
    }
  },


  {
    $group: {
      _id: {
        customerID: "$Customer.CustomerID",
        year: "$year",
        month: "$month"
      },
      Total: { $sum: "$Orderdetails.Value" },
      CompanyName: { $first: "$Customer.CompanyName" }
    }
  },


  {
    $sort: {
      "_id.year": 1,
      "_id.month": 1
    }
  },


  {
    $group: {
      _id: "$_id.customerID",
      CustomerID: { $first: "$_id.customerID" },
      CompanyName: { $first: "$CompanyName" },
      Sale: {
        $push: {
          Year: "$_id.year",
          Month: "$_id.month",
          Total: { $round: ["$Total", 2] }
        }
      }
    }
  },


  { $sort: { CustomerID: 1 } }
]);
```

Wynik:

```
[
  {
    "_id": "ALFKI",
    "CompanyName": "Alfreds Futterkiste",
    "CustomerID": "ALFKI",
    "Sale": [
      {
        "Year": 1997,
        "Month": 8,
        "Total": 814.5
      },
      {
        "Year": 1997,
        "Month": 10,
        "Total": 1208
      },
      {
        "Year": 1998,
        "Month": 1,
        "Total": 845.8
      },
      {
        "Year": 1998,
        "Month": 3,
        "Total": 471.2
      },
      {
        "Year": 1998,
        "Month": 4,
        "Total": 933.5
      },
      {
        "Year": 2025,
        "Month": 5,
        "Total": 252.2
      }
    ]
  }
]
```


Zapytanie wykorzystujące tabelę `customerInfo`:

```
db.customerInfo.aggregate([

  { $unwind: "$Orders" },


  { $unwind: "$Orders.Orderdetails" },


  {
    $addFields: {
      year: { $year: "$Orders.Dates.OrderDate" },
      month: { $month: "$Orders.Dates.OrderDate" }
    }
  },


  {
    $group: {
      _id: {
        customerID: "$CustomerID",
        year: "$year",
        month: "$month"
      },
      Total: { $sum: "$Orders.Orderdetails.Value" },
      CompanyName: { $first: "$CompanyName" }
    }
  },


  {
    $sort: {
      "_id.year": 1,
      "_id.month": 1
    }
  },


  {
    $group: {
      _id: "$_id.customerID",
      CustomerID: { $first: "$_id.customerID" },
      CompanyName: { $first: "$CompanyName" },
      Sale: {
        $push: {
          Year: "$_id.year",
          Month: "$_id.month",
          Total: { $round: ["$Total", 2] }
        }
      }
    }
  },


  { $sort: { CustomerID: 1 } }
]);
```

Wynik:

```
[
  {
    "_id": "ALFKI",
    "CompanyName": "Alfreds Futterkiste",
    "CustomerID": "ALFKI",
    "Sale": [
      {
        "Year": 1997,
        "Month": 8,
        "Total": 814.5
      },
      {
        "Year": 1997,
        "Month": 10,
        "Total": 1208
      },
      {
        "Year": 1998,
        "Month": 1,
        "Total": 845.8
      },
      {
        "Year": 1998,
        "Month": 3,
        "Total": 471.2
      },
      {
        "Year": 1998,
        "Month": 4,
        "Total": 933.5
      },
      {
        "Year": 2025,
        "Month": 5,
        "Total": 401.1
      }
    ]
  }
]
```


Podpunkt e)

Aby podczas dodawania nowych dokumentów do bazy w razie niepowodzenia nie utracić spójności danych skorzystaliśmy z mechanizmu transakcji:
`https://www.mongodb.com/docs/manual/core/transactions-in-applications/`

```
const session = db.getMongo().startSession();
session.startTransaction();

try {

  const ordersCollection = session.getDatabase("north0").orders;
  const orderDetailsCollection = session.getDatabase("north0").orderdetails;
  const ordersInfoCollection = session.getDatabase("north0").ordersInfo;
  const customerInfoCollection = session.getDatabase("north0").customerInfo;


  const maxOrderId = ordersCollection.find({}, { OrderID: 1 }).sort({ OrderID: -1 }).limit(1).toArray()[0].OrderID;
  const newOrderId = maxOrderId + 1;
  const customer = session.getDatabase("north0").customers.findOne({ CustomerID: "ALFKI" });
  const employee = session.getDatabase("north0").employees.findOne({ EmployeeID: 5 });
  const chaiProduct = session.getDatabase("north0").products.findOne({ ProductName: "Chai" });
  const ikuraProduct = session.getDatabase("north0").products.findOne({ ProductName: "Ikura" });
  const chaiCategory = session.getDatabase("north0").categories.findOne({ CategoryID: chaiProduct.CategoryID });
  const ikuraCategory = session.getDatabase("north0").categories.findOne({ CategoryID: ikuraProduct.CategoryID });
  const shipper = session.getDatabase("north0").shippers.findOne({ ShipperID: 1 });


  const chaiValue = chaiProduct.UnitPrice * 5 * (1 - 0.0);
  const ikuraValue = ikuraProduct.UnitPrice * 2 * (1 - 0.05);
  const orderTotal = chaiValue + ikuraValue;


  const orderDoc = {
    OrderID: newOrderId,
    CustomerID: "ALFKI",
    EmployeeID: 5,
    OrderDate: new Date(),
    RequiredDate: new Date(new Date().setDate(new Date().getDate() + 7)),
    ShippedDate: null,
    ShipVia: shipper.ShipperID,
    Freight: 10.50,
    ShipName: customer.CompanyName,
    ShipAddress: customer.Address,
    ShipCity: customer.City,
    ShipRegion: customer.Region,
    ShipPostalCode: customer.PostalCode,
    ShipCountry: customer.Country
  };
  ordersCollection.insertOne(orderDoc);


  orderDetailsCollection.insertMany([
    {
      OrderID: newOrderId,
      ProductID: chaiProduct.ProductID,
      UnitPrice: chaiProduct.UnitPrice,
      Quantity: 5,
      Discount: 0.0
    },
    {
      OrderID: newOrderId,
      ProductID: ikuraProduct.ProductID,
      UnitPrice: ikuraProduct.UnitPrice,
      Quantity: 2,
```

```
    Discount: 0.05
  }
]);


const orderInfoDoc = {
  OrderID: newOrderId,
  Customer: {
    CustomerID: customer.CustomerID,
    CompanyName: customer.CompanyName,
    City: customer.City,
    Country: customer.Country
  },
  Employee: {
    EmployeeID: employee.EmployeeID,
    FirstName: employee.FirstName,
    LastName: employee.LastName,
    Title: employee.Title
  },
  Dates: {
    OrderDate: new Date(),
    RequiredDate: new Date(new Date().setDate(new Date().getDate() + 7))
  },
  Orderdetails: [
    {
      UnitPrice: chaiProduct.UnitPrice,
      Quantity: 5,
      Discount: 0.0,
      Value: chaiValue,
      product: {
        ProductID: chaiProduct.ProductID,
        ProductName: chaiProduct.ProductName,
        QuantityPerUnit: chaiProduct.QuantityPerUnit,
        CategoryID: chaiProduct.CategoryID,
        CategoryName: chaiCategory.CategoryName
      }
    },
    {
      UnitPrice: ikuraProduct.UnitPrice,
      Quantity: 2,
      Discount: 0.05,
      Value: ikuraValue,
      product: {
        ProductID: ikuraProduct.ProductID,
        ProductName: ikuraProduct.ProductName,
        QuantityPerUnit: ikuraProduct.QuantityPerUnit,
        CategoryID: ikuraProduct.CategoryID,
        CategoryName: ikuraCategory.CategoryName
      }
    }
  ],
  Freight: 10.50,
  OrderTotal: orderTotal,
  Shipment: {
    Shipper: {
      ShipperID: shipper.ShipperID,
      CompanyName: shipper.CompanyName
    },
    ShipName: customer.CompanyName,
    ShipAddress: customer.Address,
```

```
      ShipCity: customer.City,
      ShipCountry: customer.Country
    }
  };
  ordersInfoCollection.insertOne(orderInfoDoc);


  const orderForCustomerInfo = { ...orderInfoDoc };
  delete orderForCustomerInfo.Customer;

  customerInfoCollection.updateOne(
    { CustomerID: "ALFKI" },
    { $push: { Orders: orderForCustomerInfo } }
  );


  session.commitTransaction();
} catch (error) {

  session.abortTransaction();
  throw error;
} finally {

  session.endSession();
}
```

Podpunkt f)

```
const maxOrderId = db.orders.find({ CustomerID: "ALFKI" }, { OrderID: 1 })
  .sort({ OrderID: -1 })
  .limit(1)
  .toArray()[0].OrderID;


const session = db.getMongo().startSession();
session.startTransaction();

try {

  const orderDetailsCollection = session.getDatabase("north0").orderdetails;
  orderDetailsCollection.updateMany(
    { OrderID: maxOrderId },
    { $inc: { Discount: 0.05 } }
  );




  const ordersInfoCollection = session.getDatabase("north0").ordersInfo;
  const orderInfo = ordersInfoCollection.findOne({ OrderID: maxOrderId });


  let updatedOrderdetails = orderInfo.Orderdetails.map(detail => {
    const newDiscount = detail.Discount + 0.05;

    const newValue = detail.UnitPrice * detail.Quantity * (1 - newDiscount);

    return {
      ...detail,
      Discount: newDiscount,
      Value: newValue
    };
  });


  const newOrderTotal = updatedOrderdetails.reduce((sum, detail) => sum + detail.Value, 0);


  ordersInfoCollection.updateOne(
    { OrderID: maxOrderId },
    {
      $set: {
        Orderdetails: updatedOrderdetails,
        OrderTotal: newOrderTotal
      }
    }
  );


  const customerInfoCollection = session.getDatabase("north0").customerInfo;
  customerInfoCollection.updateOne(
    {
      CustomerID: "ALFKI",
      "Orders.OrderID": maxOrderId
    },
    {
      $set: {
```

```
        "Orders.$.Orderdetails": updatedOrderdetails,
        "Orders.$.OrderTotal": newOrderTotal
      }
    }
  );


  session.commitTransaction();
  print("Transakcja zakończona sukcesem. Zniżki zostały zwiększone o 5% we wszystkich kolekcjach.");

} catch (error) {

  session.abortTransaction();
  print("Wystąpił błąd podczas aktualizacji zniżek: " + error.message);

} finally {

  session.endSession();
}


print("\nUpdated data for OrderID: " + maxOrderId);

print("\n1. OrderDetails:");
db.orderdetails.find({ OrderID: maxOrderId }).forEach(printjson);

print("\n2. OrdersInfo:");
const orderInfo = db.ordersInfo.findOne({ OrderID: maxOrderId });
if (orderInfo) {
    print(`OrderID: ${orderInfo.OrderID}, Total: ${orderInfo.OrderTotal}`);
    orderInfo.Orderdetails.forEach(d =>
        print(`  ${d.product.ProductName}: Discount=${d.Discount}, Value=${d.Value}`)
    );
}

print("\n3. CustomerInfo:");
const customer = db.customerInfo.findOne({
    CustomerID: "ALFKI",
    "Orders.OrderID": maxOrderId
});
if (customer) {
    const order = customer.Orders.find(o => o.OrderID === maxOrderId);
    print(`Order for ${customer.CompanyName}: Total=${order.OrderTotal}`);
    order.Orderdetails.forEach(d =>
        print(`  ${d.product.ProductName}: Discount=${d.Discount}, Value=${d.Value}`)
    );
}
```

Zadanie 2 - modelowanie danych

Zaproponuj strukturę bazy danych dla wybranego/przykładowego zagadnienia/problemu

Należy wybrać jedno zagadnienie/problem (A lub B lub C)

Przykład A

- Wykładowcy, przedmioty, studenci, oceny
  - Wykładowcy prowadzą zajęcia z poszczególnych przedmiotów
  - Studenci uczęszczają na zajęcia
  - Wykładowcy wystawiają oceny studentom
  - Studenci oceniają zajęcia

Przykład B

- Firmy, wycieczki, osoby
  - Firmy organizują wycieczki
  - Osoby rezerwują miejsca/wykupują bilety
  - Osoby oceniają wycieczki

Przykład C

- Własny przykład o podobnym stopniu złożoności

a) Zaproponuj różne warianty struktury bazy danych i dokumentów w poszczególnych kolekcjach oraz przeprowadzić dyskusję każdego wariantu (wskazać wady i zalety każdego z wariantów)

- zdefiniuj schemat/reguły walidacji danych
- wykorzystaj referencje
- dokumenty zagnieżdżone
- tablice

b) Kolekcje należy wypełnić przykładowymi danymi

c) W kontekście zaprezentowania wad/zalet należy zaprezentować kilka przykładów/zapytań/operacji oraz dla których dedykowany jest dany wariant

W sprawozdaniu należy zamieścić przykładowe dokumenty w formacie JSON (pkt a) i b)), oraz kod zapytań/ operacji (pkt c)), wraz z odpowiednim komentarzem opisującym strukturę dokumentów oraz polecenia ilustrujące wykonanie przykładowych operacji na danych

Do sprawozdania należy kompletny zrzut wykonanych/przygotowanych baz danych (taki zrzut można wykonać np. za pomocą poleceń `mongoexport`, `mongdump` …) oraz plik z kodem operacji/zapytań w wersji źródłowej (np. plik .js, np. plik .md), załącznik powinien mieć format zip

Zadanie 2 - rozwiązanie

Do wykonania tego zadania postanowiliśmy stworzyć strukturę bazy danych dla naszego projektu, którego celem jest stworzenie aplikacji do zarządzania wynajmem sprzędu budowlanego.

Głównym założeniem takiego systemu będzie możliwość monitorowania dostępności sprzętu budowlanego, jego stanu oraz historii wynajmu. Użytkownicy będą mogli przeglądać dostępne maszyny oraz składać zamówienia na wynajem.

Możliwe będzie również dodawanie ocen dla poszczególnych maszyn dostępnych w systemie.

Propozycja A

Naszą pierwszą propozycją jest stworzenie bazy danych, w której będziemy przetrzymywać jak najwięcej informacji w dokumencie opisującym sprzęt budowlany została ona nazwana `equipment`. Redundancja danych w tej kolekcji pozwoli na szybsze przeszukiwanie i uzyskiwanie najczęściej używanych informacji.

Dokument w tej kolekcji będzie wyglądał następująco:

```
{
  "_id": ObjectId("..."),
  "name": "Młotowiertarka udarowa",
  "category": "Elektronarzędzia",
  "manufacturer": "Bosch",
  "model": "GBH 2-26",
  "price_per_day": 50.00,
  "availability": true,
  "location": {
    "warehouse": "Kraków Południe",
    "address": "ul. Magazynowa 1"
  },
  "rentals": [
    {
      "rental_id": ObjectId("..."),
      "client_id": ObjectId("..."),
      "rental_date": ISODate("2025-05-20T08:00:00Z"),
      "planned_return_date": ISODate("2025-05-22T17:00:00Z"),
      "actual_return_date": null,
      "notes": "Brak uszkodzeń"
    },
    {
      "rental_id": ObjectId("..."),
      "client_id": ObjectId("..."),
      "rental_date": ISODate("2025-05-25T10:00:00Z"),
      "planned_return_date": ISODate("2025-05-27T16:00:00Z"),
      "actual_return_date": ISODate("2025-05-27T15:30:00Z"),
      "notes": "Lekkie zarysowania"
    }
  ]
}
```

Dodatkowo stworzone zostały kolekcje `clients`, `rentals` oraz `reviews`, które będą przechowywać informacje o klientach, wynajmach oraz ocenach sprzętu.

Ich przykładowe dokumenty wyglądają następująco:

Kolekcja `clients`

```
{
  "_id": ObjectId("..."),
  "first_name": "Jan",
  "last_name": "Kowalski",
  "email": "jan.kowalski@example.com",
  "phone": "123-456-789"
}
```

Kolekcja `rentals`

```
{
  "_id": ObjectId("..."),
  "equipment_id": ObjectId("..."), // Referencja do kolekcji Equipment
  "client_id": ObjectId("..."), // Referencja do kolekcji Clients
  "rental_date": ISODate("2025-05-21T09:00:00Z"),
  "planned_return_date": ISODate("2025-05-23T18:00:00Z"),
  "actual_return_date": null,
  "notes": "Sprawna"
}
```

Kolekcja `reviews`

```
{
  "_id": ObjectId("..."),
  "equipment_id": ObjectId("..."),
  "client_id": ObjectId("..."),
  "rating": 4,
  "comment": "Fajne, fajne",
  "date": ISODate("2025-05-20T12:00:00Z")
}
```

Poniżej przedstawiliśmy schematy walidacji dla każdej z kolekcji:

Kolekcja `equipment`

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["name", "category", "price_per_day", "availability", "location"],
    "properties": {
      "_id": { "bsonType": "objectId" },
      "name": { "bsonType": "string" },
      "category": { "bsonType": "string" },
      "manufacturer": { "bsonType": "string" },
      "model": { "bsonType": "string" },
      "price_per_day": { "bsonType": "double" },
      "availability": { "bsonType": "bool" },
      "location": {
        "bsonType": "object",
        "required": ["warehouse"],
        "properties": {
          "warehouse": { "bsonType": "string" },
          "address": { "bsonType": "string" }
        }
      },
      "rentals": {
        "bsonType": "array",
        "items": {
          "bsonType": "object",
          "required": ["client_id", "rental_date", "planned_return_date"],
          "properties": {
            "rental_id": { "bsonType": "objectId" },
            "client_id": { "bsonType": "objectId" },
            "rental_date": { "bsonType": "date" },
            "planned_return_date": { "bsonType": "date" },
            "actual_return_date": { "bsonType": ["date", "null"] },
            "notes": { "bsonType": "string" }
          }
        }
      }
    }
  }
}
```

Kolekcja `clients`

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["first_name", "last_name", "email", "phone"],
    "properties": {
      "_id": { "bsonType": "objectId" },
      "first_name": { "bsonType": "string" },
      "last_name": { "bsonType": "string" },
      "email": {
        "bsonType": "string",
        "pattern": "^.+@.+\\..+$"
      },
      "phone": {
        "bsonType": "string",
        "pattern": "^\\d{3}-\\d{3}-\\d{3}$"
      }
    }
  }
}
```

Kolekcja `rentals`

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["equipment_id", "client_id", "rental_date", "planned_return_date"],
    "properties": {
      "_id": { "bsonType": "objectId" },
      "equipment_id": { "bsonType": "objectId" },
      "client_id": { "bsonType": "objectId" },
      "rental_date": { "bsonType": "date" },
      "planned_return_date": { "bsonType": "date" },
      "actual_return_date": { "bsonType": ["date", "null"] },
      "notes": { "bsonType": ["string", "null"] }
    }
  }
}
```

Kolekcja `reviews`

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["equipment_id", "client_id", "rating", "comment", "date"],
    "properties": {
      "_id": { "bsonType": "objectId" },
      "equipment_id": { "bsonType": "objectId" },
      "client_id": { "bsonType": "objectId" },
      "rating": {
        "bsonType": "int",
        "minimum": 1,
        "maximum": 5
      },
      "comment": {
        "bsonType": ["string", "null"]
      },
      "date": {
        "bsonType": ["date", "null"]
      }
    }
  }
}
```

Zalety i wady (A)

- Zalety
  - Wszystkie informacje o danym sprzęcie i jego wypożyczeniach znajdują się w jednym dokumencie, co może przyspieszyć odczyt informacji o konkretnym sprzęcie i jego historii wypożyczeń.
  - Wiele zapytań dotyczących historii wypożyczeń danego sprzętu można wykonać na jednym dokumencie.
- Wady
  - W przypadku dużej liczby wypożyczeń, dokument może stać się bardzo duży, co może prowadzić do problemów z wydajnością.
  - Zmiany w historii wypożyczeń mogą wymagać aktualizacji całego dokumentu, co może być kosztowne w przypadku dużych zbiorów danych oraz może prowadzić do problemów z współbieżnością.
  - Potrzebujemy dodatkowych kolekcji do przechowywania informacji o klientach i ocenach, co może prowadzić do redundancji danych.

Propozycja B

W drugiej propozycji zdecydowaliśmy się na bardziej rozdzieloną strukturę, w której informacje o sprzęcie, klientach i wypożyczeniach są przechowywane w osobnych kolekcjach. W tej wersji każda kolekcja będzie miała swoje własne dokumenty, a relacje między nimi będą realizowane za pomocą referencji.

Dokument w kolekcji `equipment` będzie wyglądał następująco:

```
{
  "_id": ObjectId("..."),
  "name": "Młotowiertarka udarowa",
  "category": "Elektronarzędzia",
  "manufacturer": "Bosch",
  "model": "GBH 2-26",
  "price_per_day": 50.00,
  "availability": true,
  "location": {
    "warehouse": "Kraków Południe",
    "address": "ul. Magazynowa 1"
  }
}
```

Dokument w kolekcji `clients`:

```
{
  "_id": ObjectId("..."),
  "first_name": "Jan",
  "last_name": "Kowalski",
  "email": "jan.kowalski@example.com",
  "phone": "123-456-789"
}
```

Dokument w kolekcji `rentals`:

```
{
  "_id": ObjectId("..."),
  "equipment_id": ObjectId("..."),
  "client_id": ObjectId("..."),
  "rental_date": ISODate("2025-05-21T09:00:00Z"),
  "planned_return_date": ISODate("2025-05-23T18:00:00Z"),
  "actual_return_date": null,
  "notes": "Sprawna"
}
```

Dokument w kolekcji `reviews`:

```
{
  "_id": ObjectId("..."),
  "equipment_id": ObjectId("..."),
  "client_id": ObjectId("..."),
  "rating": 4,
  "comment": "Fajne, fajne",
  "date": ISODate("2025-05-20T12:00:00Z")
}
```

Schematy walidacji dla każdej z kolekcji:
Kolekcja `equipment`

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["name", "category", "price_per_day", "availability", "location"],
    "properties": {
      "_id": { "bsonType": "objectId" },
      "name": { "bsonType": "string" },
      "category": { "bsonType": "string" },
      "manufacturer": { "bsonType": "string" },
      "model": { "bsonType": "string" },
      "price_per_day": { "bsonType": "double" },
      "availability": { "bsonType": "bool" },
      "location": {
        "bsonType": "object",
        "required": ["warehouse"],
        "properties": {
          "warehouse": { "bsonType": "string" },
          "address": { "bsonType": "string" }
        }
      }
    }
  }
}
```

Schematy walidacji dla pozostałych kolekcji są takie same jak w propozycji A.

Zalety i wady (B)

- Zalety
  - Mniejsze dokumenty, co może poprawić wydajność operacji zapisu i aktualizacji.
  - Możliwość łatwego dodawania nowych pól do dokumentów bez wpływu na inne kolekcje.
  - Łatwiejsze zarządzanie historią wypożyczeń, ponieważ każda transakcja jest przechowywana w osobnym dokumencie.
- Wady
  - Wymaga więcej zapytań do bazy danych, aby uzyskać pełne informacje o sprzęcie i jego historii wypożyczeń.
  - Złożoność zapytań wzrasta, ponieważ trzeba łączyć dane z różnych kolekcji.

Wypełnianie przykładowymi danymi

Poniżej znajdują się wyniki poleceń `db.<kolekcja>.find()` dla każdej z kolekcji.

Tabela `equipment` w przykładzie A:

```
[
  {
    "_id": {"$oid": "60b0b0b0b0b0b0b0b0b0b001"},
    "availability": true,
    "category": "Elektronarzędzia",
    "location": {
      "warehouse": "Kraków Południe",
      "address": "ul. Magazynowa 1"
    },
    "manufacturer": "Bosch",
    "model": "GBH 2-26",
    "name": "Młotowiertarka udarowa",
    "price_per_day": 50.99,
    "rentals": [
      {
        "rental_id": {"$oid": "60c0c0c0c0c0c0c0c0c001"},
        "client_id": {"$oid": "60a0a0a0a0a0a0a0a0a001"},
        "rental_date": {"$date": "2025-05-20T08:00:00.000Z"},
        "planned_return_date": {"$date": "2025-05-22T17:00:00.000Z"},
        "actual_return_date": null,
        "notes": "Brak uszkodzeń"
      },
      {
        "rental_id": {"$oid": "60c0c0c0c0c0c0c0c0c002"},
        "client_id": {"$oid": "60a0a0a0a0a0a0a0a0a002"},
        "rental_date": {"$date": "2025-05-25T10:00:00.000Z"},
        "planned_return_date": {"$date": "2025-05-27T16:00:00.000Z"},
        "actual_return_date": {"$date": "2025-05-27T15:30:00.000Z"},
        "notes": "Lekkie zarysowania"
      }
    ]
  },
  {
    "_id": {"$oid": "60b0b0b0b0b0b0b0b0b002"},
    "availability": true,
    "category": "Maszyny Budowlane",
    "location": {
      "warehouse": "Warszawa Centrum",
      "address": "ul. Centralna 5"
    },
    "manufacturer": "Belle",
    "model": "B150",
    "name": "Betoniarka",
    "price_per_day": 80.99,
    "rentals": [
      {
        "rental_id": {"$oid": "60c0c0c0c0c0c0c0c0c003"},
        "client_id": {"$oid": "60a0a0a0a0a0a0a0a0a001"},
        "rental_date": {"$date": "2025-06-01T09:00:00.000Z"},
        "planned_return_date": {"$date": "2025-06-03T18:00:00.000Z"},
        "actual_return_date": null,
        "notes": "Sprawna"
      }
    ]
  }
]
```


Kolekcja `equipment` w przykładzie B:

```
[
  {
    "_id": {"$oid": "60b0b0b0b0b0b0b0b0b0b001"},
    "availability": true,
    "category": "Elektronarzędzia",
    "location": {
      "warehouse": "Kraków Południe",
      "address": "ul. Magazynowa 1"
    },
    "manufacturer": "Bosch",
    "model": "GBH 2-26",
    "name": "Młotowiertarka udarowa",
    "price_per_day": 50.99
  },
  {
    "_id": {"$oid": "60b0b0b0b0b0b0b0b0b0b002"},
    "availability": true,
    "category": "Maszyny Budowlane",
    "location": {
      "warehouse": "Warszawa Centrum",
      "address": "ul. Centralna 5"
    },
    "manufacturer": "Belle",
    "model": "B150",
    "name": "Betoniarka",
    "price_per_day": 80.99
  }
]
```


Teraz reszta kolekcji wygląda tak samo dla obu przykładów, więc nie będziemy ich powtarzać.

Kolekcja `clients`:


```
[
  {
    "_id": {"$oid": "60a0a0a0a0a0a0a0a0a001"},
    "email": "jan.kowalski@example.com",
    "first_name": "Jan",
    "last_name": "Kowalski",
    "phone": "123-456-789"
  },
  {
    "_id": {"$oid": "60a0a0a0a0a0a0a0a0a002"},
    "email": "anna.nowak@example.com",
    "first_name": "Anna",
    "last_name": "Nowak",
    "phone": "987-654-321"
  }
]
```


Kolekcja `rentals`:

```
[
  {
    "_id": {"$oid": "60c0c0c0c0c0c0c0c0c0c001"},
    "actual_return_date": {"$date": "2025-05-27T14:30:00.000Z"},
    "client_id": {"$oid": "60a0a0a0a0a0a0a0a0a0a001"},
    "equipment_id": {"$oid": "60b0b0b0b0b0b0b0b0b0b001"},
    "notes": "Brak uszkodzeń",
    "planned_return_date": {"$date": "2025-05-22T17:00:00.000Z"},
    "rental_date": {"$date": "2025-05-20T08:00:00.000Z"}
  },
  {
    "_id": {"$oid": "60c0c0c0c0c0c0c0c0c0c002"},
    "actual_return_date": {"$date": "2025-05-27T15:30:00.000Z"},
    "client_id": {"$oid": "60a0a0a0a0a0a0a0a0a0a002"},
    "equipment_id": {"$oid": "60b0b0b0b0b0b0b0b0b0b001"},
    "notes": "Lekkie zarysowania",
    "planned_return_date": {"$date": "2025-05-27T16:00:00.000Z"},
    "rental_date": {"$date": "2025-05-25T10:00:00.000Z"}
  },
  {
    "_id": {"$oid": "60c0c0c0c0c0c0c0c0c0c003"},
    "actual_return_date": {"$date": "2025-05-27T16:30:00.000Z"},
    "client_id": {"$oid": "60a0a0a0a0a0a0a0a0a0a001"},
    "equipment_id": {"$oid": "60b0b0b0b0b0b0b0b0b0b002"},
    "notes": "Sprawna",
    "planned_return_date": {"$date": "2025-06-03T18:00:00.000Z"},
    "rental_date": {"$date": "2025-06-01T09:00:00.000Z"}
  }
]
```


Kolekcja `reviews`:


```
[
  {
    "_id": {"$oid": "60d0d0d0d0d0d0d0d0d0d001"},
    "client_id": {"$oid": "60a0a0a0a0a0a0a0a0a0a001"},
    "comment": "Fajne, fajne",
    "date": {"$date": "2025-05-20T12:00:00.000Z"},
    "equipment_id": {"$oid": "60b0b0b0b0b0b0b0b0b0b001"},
    "rating": 4
  },
  {
    "_id": {"$oid": "60d0d0d0d0d0d0d0d0d0d002"},
    "client_id": {"$oid": "60a0a0a0a0a0a0a0a0a0a001"},
    "comment": "Super sprzęt!",
    "date": {"$date": "2025-06-04T10:00:00.000Z"},
    "equipment_id": {"$oid": "60b0b0b0b0b0b0b0b0b0b002"},
    "rating": 5
  }
]
```

Przykłady zapytań

Dodanie nowego wypożyczenia

Aby dodać nowe wypożyczenie w przykładzie A, należy zaktualizować dokument w kolekcji `equipment` oraz dodać nowy dokument do kolekcji `rentals`. W przykładzie B wystarczy dodać nowy dokument do kolekcji `rentals`, ponieważ nie ma potrzeby aktualizacji dokumentu w kolekcji `equipment`.

Zapytanie do dodania nowego wypożyczenia w przykładzie A:

```
const equipmentId = ...;// ID sprzętu, do którego dodajemy wypożyczenie
const clientId = ...;// ID klienta, który wypożycza sprzęt
const newRentalId = new ObjectId();
const rentalDate = new ISODate();
const plannedReturnDate = new ISODate("2025-06-01T12:00:00Z");

db.equipment.updateOne(
  { _id: equipmentId },
  {
    $push: {
      rentals: {
        rental_id: newRentalId, // Używamy tego samego ID
        client_id: clientId,
        rental_date: rentalDate,
        planned_return_date: plannedReturnDate,
        actual_return_date: null,
        notes: "Note"
      }
    }
  }
);

db.rentals.insertOne({
  _id: newRentalId, // Używamy tego samego ID
  equipment_id: equipmentId,
  client_id: clientId,
  rental_date: rentalDate,
  planned_return_date: plannedReturnDate,
  actual_return_date: null,
  notes: "Note"
});
```

W przykładzie B zapytanie do dodania nowego wypożyczenia wygląda następująco:

```
const equipmentId = ...;// ID sprzętu, do którego dodajemy wypożyczenie
const clientId = ...;// ID klienta, który wypożycza sprzęt
const newRentalId = new ObjectId();
const rentalDate = new ISODate();
const plannedReturnDate = new ISODate("2025-06-01T12:00:00Z");
db.rentals.insertOne({
  _id: newRentalId,
  equipment_id: equipmentId,
  client_id: clientId,
  rental_date: rentalDate,
  planned_return_date: plannedReturnDate,
  actual_return_date: null,
  notes: "Note"
});
```

Wyciąganie danych o sprzęcie i jego wypożyczeniach

Aby wyciągnąć dane o sprzęcie i jego wypożyczeniach w przykładzie A, wystarczy użyć następującego zapytania:

```
const equipmentId = ObjectId("...");
db.equipment.findOne({ _id: equipmentId });
```

Do uzyskania podobnego wyniku w przykładzie B, trzeba wykonać zapytanie aggregacyjne, aby połączyć dane z kolekcji `equipment` i `rentals`:

```
// Załóżmy, że znamy _id sprzętu
const equipmentId = ObjectId("...");
db.equipment.aggregate([
  { $match: { _id: equipmentId } },
  {
    $lookup: {
      from: "rentals",
      localField: "_id",
      foreignField: "equipment_id",
      as: "rental_history"
    }
  }
]);
```

Wynik polecenia (Przykład B) dla sprzętu o ID `60b0b0b0b0b0b0b0b0b0b001` wygląda następująco:

```
[
  {
    "_id": {"$oid": "60b0b0b0b0b0b0b0b0b0b001"},
    "availability": true,
    "category": "Elektronarzędzia",
    "location": {
      "warehouse": "Kraków Południe",
      "address": "ul. Magazynowa 1"
    },
    "manufacturer": "Bosch",
    "model": "GBH 2-26",
    "name": "Młotowiertarka udarowa",
    "price_per_day": 50.99,
    "rental_history": [
      {
        "_id": {"$oid": "60c0c0c0c0c0c0c0c0c001"},
        "equipment_id": {"$oid": "60b0b0b0b0b0b0b0b0b001"},
        "client_id": {"$oid": "60a0a0a0a0a0a0a0a001"},
        "rental_date": {"$date": "2025-05-20T08:00:00.000Z"},
        "planned_return_date": {"$date": "2025-05-22T17:00:00.000Z"},
        "actual_return_date": {"$date": "2025-05-27T14:30:00.000Z"},
        "notes": "Brak uszkodzeń"
      },
      {
        "_id": {"$oid": "60c0c0c0c0c0c0c0c0c002"},
        "equipment_id": {"$oid": "60b0b0b0b0b0b0b0b0b001"},
        "client_id": {"$oid": "60a0a0a0a0a0a0a0a002"},
        "rental_date": {"$date": "2025-05-25T10:00:00.000Z"},
        "planned_return_date": {"$date": "2025-05-27T16:00:00.000Z"},
        "actual_return_date": {"$date": "2025-05-27T15:30:00.000Z"},
        "notes": "Lekkie zarysowania"
      }
    ]
  }
]
```


W tym przypadku zdecydowanie lepiej sprawdzi się przykład A, ponieważ wszystkie dane o wypożyczeniach są już w dokumencie sprzętu, więc nie trzeba wykonywać dodatkowego zapytania do kolekcji `rentals`.

Aktualizacja stanu wypożyczenia

W przykładzie A:

```
const rentalId = ObjectId("60b0b0b0b0b0b0b0b0b001");
const doc = db.rentals.findOne({ _id: rentalId });
const equipmentId = doc.equipment_id;

db.equipment.updateOne(
  { _id: equipmentId, "rentals.rental_id": rentalId },
  { $set: { "rentals.$.actual_return_date": new ISODate(), "rentals.$.notes": "Sprzęt zwrócony" } }
);

db.rentals.updateOne(
  { _id: rentalId },
  { $set: { actual_return_date: new ISODate(), notes: "Sprzęt zwrócony" } }
```

Dla przykładu B:

```
const rentalIdToUpdate = ...;
db.rentals.updateOne(
  { _id: rentalIdToUpdate },
  { $set: { notes: "Sprzęt zwrócony" } }
);
```

Wynik polecenia (B):


```
[
  {
    "_id": {"$oid": "60c0c0c0c0c0c0c0c0c0c001"},
    "actual_return_date": {"$date": "2025-05-27T14:30:00.000Z"},
    "client_id": {"$oid": "60a0a0a0a0a0a0a0a0a0a001"},
    "equipment_id": {"$oid": "60b0b0b0b0b0b0b0b0b0b001"},
    "notes": "Sprzęt zwrócony",
    "planned_return_date": {"$date": "2025-05-22T17:00:00.000Z"},
    "rental_date": {"$date": "2025-05-20T08:00:00.000Z"}
  }
]
```


Wynik polecenia w kolekcji `equipment` (A):

```
[
  {
    "_id": {"$oid": "60b0b0b0b0b0b0b0b0b0b001"},
    "availability": true,
    "category": "Elektronarzędzia",
    "location": {
      "warehouse": "Kraków Południe",
      "address": "ul. Magazynowa 1"
    },
    "manufacturer": "Bosch",
    "model": "GBH 2-26",
    "name": "Młotowiertarka udarowa",
    "price_per_day": 50.99,
    "rentals": [
      {
        "rental_id": {"$oid": "60c0c0c0c0c0c0c0c0c001"},
        "client_id": {"$oid": "60a0a0a0a0a0a0a0a0a001"},
        "rental_date": {"$date": "2025-05-20T08:00:00.000Z"},
        "planned_return_date": {"$date": "2025-05-22T17:00:00.000Z"},
        "actual_return_date": {"$date": "2025-05-20T23:04:39.911Z"},
        "notes": "Sprzęt zwrócony"
      },
      {
        "rental_id": {"$oid": "60c0c0c0c0c0c0c0c0c002"},
        "client_id": {"$oid": "60a0a0a0a0a0a0a0a0a002"},
        "rental_date": {"$date": "2025-05-25T10:00:00.000Z"},
        "planned_return_date": {"$date": "2025-05-27T16:00:00.000Z"},
        "actual_return_date": {"$date": "2025-05-27T15:30:00.000Z"},
        "notes": "Lekkie zarysowania"
      }
    ]
  },
]
```


W tym przypadku lepszy okazał się przykład B, ponieważ nie trzeba było aktualizować dokumentu w kolekcji `equipment`, a jedynie zaktualizować dokument w kolekcji `rentals`. Dzięki temu operacja była szybsza i bardziej efektywna.

Wnioski

Końcowo mimo wszystko do naszego projektu lepiej pasuje przykład B, ponieważ mamy mniejsze dokumenty, co może poprawić wydajność operacji zapisu i aktualizacji. Możemy również łatwiej zarządzać historią wypożyczeń, ponieważ każda transakcja jest przechowywana w osobnym dokumencie. Dodatkowo możemy łatwo dodawać nowe pola do dokumentów bez wpływu na inne kolekcje.

Punktacja:

zadanie	pkt

1	1
2	1
razem	2