NIERELACYJNE ROZWIĄZANIA BAZODANOWE

Wykład 4

Agenda

- Przegląd zaawansowanych funkcji NoSQL
- Przykłady funkcji dla MongoDB w języku Python
- Zaawansowane instrukcje warunkowe

Przegląd zaawansowanych funkcji NoSQL

Bazy dokumentowe

Przykładowy zbiór danych


```
cities = [
    {
        "city": "Warszawa",
        "population": 1790658,
        "voivodeship": "Mazowieckie",
        "area_km2": 517.24,
        "latitude": 52.2297,
        "longitude": 21.0122,
        "points_of_interest": ["Stare Miasto", "Pałac Kultury i Nauki", "Łazienki"]
    },
    {
        "city": "Kraków",
        "population": 779115,
        "voivodeship": "Małopolskie",
        "area_km2": 327.0,
        "latitude": 50.0647,
        "longitude": 19.945,
        "points_of_interest": ["Zamek wawelski"]
    }
]
```

Rodzaje zaawansowanych funkcji

1. Akumulatory (agregacje)
2. Tablicowe
3. Wyrażenia warunkowe
4. Funkcje daty
5. Operatory zbiorów
6. Funkcje łańcucha znaków

Przegląd funkcji

- 1. Agregujące funkcje matematyczne i statystyczne
  - Służą do obliczania wartości w grupach.

- Opis funkcji
  - $sum Oblicza sumę wartości liczbowych.
  - $avg Oblicza średnią wartości liczbowych.
  - $min Znajduje wartość minimalną.
  - $max Znajduje wartość maksymalną.
  - $count Podaje liczbę dokumentów (odpowiednik COUNT(*) w SQL).
  - $stdDevPop Oblicza odchylenie standardowe populacji.
  - $stdDevSamp Oblicza odchylenie standardowe próbki.

Przegląd funkcji

- 2. Operatory tablicowe
  - Służą do przetwarzania tablic.

- Opis funkcji
  - $push Zbiera wszystkie wartości w tablicy.
  - $addToSet Zbiera unikalne wartości w tablicy (bez duplikatów).
  - $first Zwraca pierwszą wartość w grupie.
  - $last Zwraca ostatnią wartość w grupie.
  - $arrayElemAt Zwraca element o określonym indeksie w tablicy.
  - $size Zwraca rozmiar tablicy.
  - $reverseArray Odwraca kolejność tablicy.
  - $reduce Stosuje wyrażenie sekwencyjnie, aby zredukować tablicę do pojedynczej wartości.

Przegląd funkcji

- 3. Wyrażenia warunkowe
  - Dla if/else-like w zapytaniach.

- Opis funkcji
  - $cond Ocenia warunek i zwraca wartość na podstawie prawda/fałsz.
  - $ifNull Zwraca wartość domyślną, jeśli pole jest nullem lub nie istnieje.
  - $switch Ocenia wiele warunków i wybiera jedną wartość do zwrócenia.

Przegląd funkcji

- 4. Funkcje daty
  - Służą do przetwarzania i manipulowania polami daty/czasu.

- Opis funkcji
  - $dateToString Konwertuje obiekt daty na ciąg znaków o niestandardowym formacie.
  - $year Wyodrębnia rok z daty.
  - $month Wyodrębnia miesiąc z daty.
  - $dayOfMonth Wyodrębnia dzień miesiąca z daty.
  - $hour Wyodrębnia godzinę z daty.
  - $minute Wyodrębnia minutę z daty.
  - $second Wyodrębnia sekundę z daty.
  - $millisecond Wyodrębnia milisekundę z daty.
  - $isoDayOfWeek Wyodrębnia dzień tygodnia ISO 8601 (1 = poniedziałek, 7 = niedziela).

Przegląd funkcji

- 5. Operatory zbiorów
  - Służą do porównywania tablic.

- Opis funkcji
  - $setEquals Zwraca wartość true, jeśli dwa zbiory są równe.
  - $setIntersection Zwraca wspólne elementy tablic.
  - $setUnion Zwraca sumę tablic (wszystkie unikalne elementy).
  - $setDifference Zwraca elementy z pierwszej tablicy, ale nie z drugiej.
  - $isSubset Zwraca wartość true, jeśli pierwsza tablica jest podzbiorem drugiej.

Przegląd funkcji

- 6. Funkcje łańcucha znaków (string)
  - Do przetwarzania i manipulowania danymi łańcuchowymi.

- Opis funkcji
  - $concat Łączy wiele łańcuchów w jeden łańcuch.
  - $substr Wyodrębnia podłańcuch.
  - $toLower Konwertuje łańcuch na małe litery.
  - $toUpper Konwertuje łańcuch na wielkie litery.
  - $trim Usuwa odstępy lub określone znaki z obu końców łańcucha.
  - $indexOfBytes Zwraca indeks bajtu podłańcucha w łańcuchu.
  - $split Dzieli łańcuch na tablicę za pomocą ogranicznika.

Przykłady funkcji dla MongoDB w języku Python

Bazy dokumentowe

Podstawowe zapytania agregujące

- Liczebność
- Minimum
- Maksimum
- Suma
- Średnia

Liczebność zbioru


```
count = collection.aggregate([
    {"$count": "total_cities"}
])
for w in count:
    print(w)
```

Minimum i maksimum


```
min_population = collection.aggregate([
    {"$group": {"_id": None, "min_population": {"$min": "$population"}}}
])
for w in min_population:
    print(w)

max_population = collection.aggregate([
    {"$group": {"_id": None, "max_population": {"$max": "$population"}}}
])
for w in max_population:
    print(w)
```

Suma


```
total_population = collection.aggregate([
    {"$group": {"_id": None, "total_population": {"$sum": "$population"}}}
])
for w in total_population:
    print(w)
```

Średnia


```
average_population = collection.aggregate([
    {"$group": {"_id": None, "average_population": {"$avg": "$population"}}}
])
for w in average_population:
    print(w)
```

Funkcje tablicowe

- # 1. $push: Ddoanie nowego elementu tablicy
- collection.update_many({}, {"$push": {"points_of_interest": "Nowa atrakcja miasta"}})
- for miasto in collection.find({}, {"city": 1, "points_of_interest": 1}):
- print(miasto)

Funkcje tablicowe

- # 2. $addToSet: Dodanie unikalnego elementu listy
- collection.update_many({}, {"$addToSet": {"points_of_interest": "Nowa unikalna atrakcja"}})

- for miasto in collection.find({}, {"city": 1, "points_of_interest": 1}):
- print(miasto)

Funkcje tablicowe


```
# 3. $size: Filtrowanie danych z tablicami czteroelementowymi
query = {"$expr": {"$gt": [{"$size": "$points_of_interest"}, 4]}}
for miasto in collection.find(query, {"city": 1, "points_of_interest": 1}):
    print(miasto)
```

Funkcje tablicowe


```
# 4. $arrayElemAt: Pobranie drugiego elementu tablicy
pipeline = [
    {"$project": {"city": 1, "second_point_of_interest": {"$arrayElemAt": ["$points_of_interest", 1]}}}
]
for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje tablicowe


```
# 5. $slice: Pobranie pierwszych dwóch elementów tablicy
pipeline = [
    {"$project": {"city": 1, "first_two_points_of_interest": {"$slice": ["$points_of_interest", 2]}}}
]
for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje tablicowe


```
# 6. $reduce: Połączenie wszystkich elementów w jeden string
pipeline = [    {        "$project": {
            "city": 1,
            "points_of_interest_string": {
                "$reduce": {
                    "input": "$points_of_interest",
                    "initialValue": "",
                    "in": {
                        "$cond": {
                            "if": {"$eq": ["$$value", ""]},
                            "then": "$$this",
                            "else": {"$concat": ["$$value", ", ", "$$this"]}
                        }                    }                }            }        }    }]
for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje warunkowe


```
pipeline = [
    {
        "$project": {
            "city": 1,
            "population": 1,
            "size_category": {
                "$cond": {
                    "if": {"$gte": ["$population", 500000]},
                    "then": "duże",
                    "else": "małe"
                }
            }
        }
    }
]
for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje warunkowe


```
pipeline = [
    {
        "$project": {
            "city": 1,
            "average_temperature": {
                "$ifNull": ["$average_temperature", 10]
            }
        }
    }
]

for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje warunkowe


```
pipeline = [
    {
        "$project": {
            "city": 1,
            "population": 1,
            "population_category": {
                "$switch": {
                    "branches": [
                        {
                            "case": {"$gte": ["$population", 1000000]},
                            "then": "Obszar metropolitalny"
                        },
                        {
                            "case": {"$gte": ["$population", 500000]},
                            "then": "Wielkie miasto"
                        },
                        {
                            "case": {"$gte": ["$population", 100000]},
                            "then": "Male miasto"
                        }
                    ],
                    "default": "Wies"
                }
            }
        }
    }
]

for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje łańcucha znaków


```
pipeline = [
    {
        "$project": {
            "city": 1,
            "full_description": {
                "$concat": ["$city", " to miasto w Polsce."]
            }
        }
    }
]

for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje łańcucha znaków


```
pipeline = [
    {
        "$project": {
            "city": 1,
            "city_lowercase": {"$toLower": "$city"},
            "city_uppercase": {"$toUpper": "$city"}
        }
    }
]

for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje łańcucha znaków


```
pipeline = [
    {
        "$project": {
            "city": 1,
            "shortcut": {"$substr": ["$city", 0, 2]}
        }
    }
]

for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje łańcucha znaków


```
pipeline = [
    {
        "$project": {
            "city": 1,
            "position_of_capital": {"$indexOfBytes": ["$city", "saw"]}
        }
    }
]

for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje łańcucha znaków


```
pipeline = [
    {
        "$project": {
            "city": 1,
            "words_splitted": {"$split": ["$voivodeship", " "]}
        }
    }
]

for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje łańcucha znaków


```
pipeline = [
    {
        "$project": {
            "city": 1,
            "trimmed_description": {"$trim": {"input": "$voivodeship"}}
        }
    }
]

for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje daty

Konieczne dodanie nowego miasta:

```
import dateutil.parser
import datetime
dateStr = "2024-12-24T12:34:56.00000Z"
dateutil.parser.parse(dateStr)
ISODate = datetime.datetime.fromtimestamp(ts, None)

collection.insert_one({
  "city": "IT city",
  "population": 453,
  "established_date": ISODate,
  "description": "AiB is the virtual city of Poland."
})
```

Funkcje daty


```
pipeline = [
    {
        "$project": {
            "city": 1,
            "year": {"$year": "$established_date"},
            "month": {"$month": "$established_date"},
            "day": {"$dayOfMonth": "$established_date"},
            "hour": {"$hour": "$established_date"},
            "minute": {"$minute": "$established_date"},
            "second": {"$second": "$established_date"}
        }
    }
]

for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje daty


```
pipeline = [
    {
        "$project": {
            "city": 1,
            "formatted_date": {
                "$dateToString": {
                    "format": "%Y-%m-%d",
                    "date": "$established_date"
                }
            }
        }
    }
]

for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje daty


```
pipeline = [
    {
        "$project": {
            "city": 1,
            "iso_week_year": {"$isoWeekYear": "$established_date"},
            "iso_day_of_week": {"$isoDayOfWeek": "$established_date"}
        }
    }
]

for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje daty


```
pipeline = [
    {
        "$project": {
            "city": 1,
            "years_since_established": {
                "$dateDiff": {
                    "startDate": "$established_date",
                    "endDate": "$established_date",
                    "unit": "year"
                }
            }
        }
    }
]

for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje daty


```
pipeline = [
    {
        "$project": {
            "city": 1,
            "date_plus_100_years": {"$add": ["$established_date", 100 * 365 * 24 * 60 * 60 * 1000]},
            "date_minus_100_years": {"$subtract": ["$established_date", 100 * 365 * 24 * 60 * 60 *
1000]}
        }
    }
]

for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje operatorów zbiorów i tablic

Niezbędna aktualizacja kolekcji:

```
collection.update_many(
    {},
    {"$set": {"historical_sites": ["Stare miasto","Old Town"]}}
)
collection.delete_many({'city':'IT City'})
```

Funkcje operatorów zbiorów i tablic


```
pipeline = [
    {
        "$project": {
            "city": 1,
            "all_attractions": {
                "$setUnion": ["$points_of_interest", "$historical_sites"]
            }
        }
    }
]

for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje operatorów zbiorów i tablic


```
pipeline = [
    {
        "$project": {
            "city": 1,
            "common_attractions": {
                "$setIntersection": ["$points_of_interest", "$historical_sites"]
            }
        }
    }
]

for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Funkcje operatorów zbiorów i tablic


```
pipeline = [
    {
        "$project": {
            "city": 1,
            "unique_points_of_interest": {
                "$setDifference": ["$points_of_interest", "$historical_sites"]
            }
        }
    }
]

for miasto in collection.aggregate(pipeline):
    print(miasto)
```

Zaawansowane instrukcje warunkowe

Zapytanie LIKE

- query = {"city": {"$regex": "^L", "$options": "i"}}
- miasta=collection.find(query)
- for miasto in miasta:
- print(miasto)

- $options – i – nie jest case sensitive
- $regex - ^L – zaczyna się na L

Zapytanie LIKE – ostatnia litera nazwy

- query = {"city": {"$regex": "n$", "$options": "i"}}
- n$ - ostatnia litera ma być n

Pytanie

Dodanie unikalnego elementu tablicy jest możliwe dzięki instrukcji:

- addToSet
- addToTable
- addToArray
- addToCollection