LABORATORIUM NIERELACYJNE BAZY DANYCH

Data wykonania ćwiczenia:	06.05.2023
Rok studiów:	3
Semestr:	6
Grupa studencka:	2
Grupa laboratoryjna:	2B

Ćwiczenie nr.	8

Temat: Sharding w MONGODB. Obleczenia rozproszone. MapReduce na serwerach

Osoby wykonujące ćwiczenia:

1. Igor Gawłowicz
2. Patryk Pawełek

Katedra Informatyki i Automatyki

Sharding w MongoDB. Obliczenia rozproszone. MapReduce na serwerach

Do pracy laboratoryjnej uczniowie powinni zostać podzieleni na trzyosobowe grupy. Jeden uczeń (1) tworzy i konfiguruje serwer konfiguracji, serwer routingu i dodaje bazę danych, drugi (2) i trzeci (3) przygotowują swój komputer do użycia jako shardu. Po wykonaniu zadania uczniowie zamieniają się rolami.

Konfiguracja serwera konfiguracji

Najpierw musimy utworzyć serwer używany w shardingu. Utworzymy trzy instancje serwera konfiguracji na różnych portach. W tym celu otwieramy trzy oddzielne konsole i uruchamiamy następujące polecenia na PC 1:

1. Serwer konfiguracji 1:

```
mongod --configsvr --replSet configReplSet --dbpath
C:\Szkola\SEMESTR6\NBD\lab8\data\config1 --port 27019
```

2. Serwer konfiguracji 2:

```
mongod --configsvr --replSet configReplSet --dbpath
C:\Szkola\SEMESTR6\NBD\lab8\data\config2 --port 27020
```

3. Serwer konfiguracji 3 (na PC 2):

```
mongod --configsvr --replSet configReplSet --dbpath
C:\Szkola\SEMESTR6\NBD\lab8\data\config3 --port 27021
```

Inicjalizacja repliki serwera konfiguracji

Następnie połączymy się z naszym pierwszym serwerem konfiguracji za pomocą `mongosh` i zainicjujemy replikę:

```
mongosh --port 27019
```

W konsoli `mongosh` wpisujemy:

```
rs.initiate({
  _id: "configReplSet",
  configsvr: true,
  members: [
    { _id: 0, host: "PC1_IP:27019" },
    { _id: 1, host: "PC1_IP:27020" },
    { _id: 2, host: "PC2_IP:27021" },
```

```
  ],
});
```

Konfiguracja shardów

PC 2 - Konfiguracja shardów

Na PC 2 uruchamiamy shardy na różnych portach:

1. Shard 1, replika 1:

```
mongod --shardsvr --replSet shardReplSet1 --dbpath
C:\Szkola\SEMESTR6\NBD\lab8\data\shard1 --port 27022
```

2. Shard 1, replika 2:

```
mongod --shardsvr --replSet shardReplSet1 --dbpath
C:\Szkola\SEMESTR6\NBD\lab8\data\shard2 --port 27023
```

3. Shard 2, replika 1:

```
mongod --shardsvr --replSet shardReplSet2 --dbpath
C:\Szkola\SEMESTR6\NBD\lab8\data\shard3 --port 27024
```

4. Shard 2, replika 2:

```
mongod --shardsvr --replSet shardReplSet2 --dbpath
C:\Szkola\SEMESTR6\NBD\lab8\data\shard4 --port 27025
```

Inicjalizacja replik shardów

Na PC 2 inicjujemy replikę dla Shard 1:

```
mongosh --port 27022
```

W konsoli `mongosh` wpisujemy:

```
rs.initiate({
  _id: "shardReplSet1",
  members: [
```

```
    { _id: 0, host: "PC2_IP:27022" },
    { _id: 1, host: "PC2_IP:27023" },
  ],
});
```

Na PC 2 inicjujemy replikę dla Shard 2:

```
mongosh --port 27024
```

W konsoli `mongosh` wpisujemy:

```
rs.initiate({
  _id: "shardReplSet2",
  members: [
    { _id: 0, host: "PC2_IP:27024" },
    { _id: 1, host: "PC2_IP:27025" },
  ],
});
```

Konfiguracja serwera routingu (mongos)

Na PC 1 uruchamiamy serwer `mongos`, który będzie pełnił rolę routera:

```
mongos --configdb configReplSet/PC1_IP:27019,PC1_IP:27020,PC2_IP:27021 --port
27017
```

Dodawanie shardów do klastra

Łączymy się z serwerem `mongos` na PC 1:

```
mongosh --port 27017
```

Dodajemy shardy do klastra:

```
sh.addShard("shardReplSet1/PC2_IP:27022,PC2_IP:27023");
sh.addShard("shardReplSet2/PC2_IP:27024,PC2_IP:27025");
```

Włączanie shardingu dla bazy danych

Włączamy sharding dla bazy danych:

```
sh.enableSharding("myapp");
```

Shardowanie kolekcji

Shardujemy kolekcję w bazie danych, definiując klucz shardujący:

```
sh.shardCollection("myapp.shardCollection", { shardKeyField: 1 });
```

Weryfikacja i monitorowanie

Sprawdzamy status shardingu:

```
sh.status();
```

Sprawdzamy status replik:

```
rs.status();
```

Powiedz serwerowi nazwę kolekcji do shardingu:

```
db.runCommand({ shardCollection: "pdb.phones", key: { phoneNumber: 1 } });
```

Uzasadnienie

- `phoneNumber` jest unikalnym identyfikatorem dla każdego dokumentu.
- Rozkład wartości tego pola jest zazwyczaj równomierny, co pomaga w równomiernym rozłożeniu danych między shardami.
- Unikalne i równomiernie rozłożone klucze minimalizują hot spots i zwiększają wydajność klastrów.

Utwórz map i reduce funkcje w oparciu o wiedzę uzyskane w toku poprzednich prac laboratoryjnych

```
var mapFunction = function () {
  emit(this.type, 1);
};

var reduceFunction = function (key, values) {
  return Array.sum(values);
};

db.phones.mapReduce(mapFunction, reduceFunction, {
```

```
  out: "phone_type_counts",
});
```

Uruchom komendę mapReduce w kolekcji „telefony". Wyprowadź wynik pracy do konsoli i zapisz go jako oddzielną kolekcję

```
[
  { _id: "home", value: 2 },
  { _id: "mobile", value: 3 },
  { _id: "work", value: 2 },
];
```

Pytania kontrolne

1. **Czym jest sharding w kontekście MongoDB i jakie są jego główne założenia?** Sharding w MongoDB to metoda rozpraszania danych w wielu serwerach w celu rozłożenia obciążenia i zwiększenia wydajności. Główne założenia shardingu to podział dużych kolekcji danych na mniejsze części (shardy), które są rozproszone w wielu węzłach, aby umożliwić skalowanie poziome oraz zapewnić wysoką dostępność i wydajność.

2. **Jakie są korzyści z zastosowania sharding w MongoDB?** Korzyści z zastosowania shardingu w MongoDB to:

    - Skalowalność: umożliwia rozdzielenie danych na wiele serwerów.
    - Wydajność: rozkład obciążenia na wiele maszyn zmniejsza opóźnienia w dostępie do danych.
    - Wysoka dostępność: dane są replikowane, co zapewnia redundancję i odporność na awarie.
    - Elastyczność: łatwe dodawanie nowych shardów w miarę wzrostu danych.

3. **Jak MongoDB realizuje sharding danych w środowisku rozproszonym?** MongoDB realizuje sharding poprzez podział danych na "chunks" (kawałki) na podstawie klucza shardującego. Każdy chunk jest przypisany do konkretnego sharda. Routery (mongos) zarządzają zapytaniami, kierując je do odpowiednich shardów, a serwery konfiguracji (config servers) przechowują metadane dotyczące struktury shardingu i lokalizacji chunków.

4. **Jakie są główne komponenty sharding w MongoDB?** Główne komponenty shardingu w MongoDB to:

    - **Shardy:** Przechowują rzeczywiste dane. Każdy shard jest autonomicznym MongoDB instance.
    - **Mongos:** Routery, które kierują zapytania klienta do odpowiednich shardów.
    - **Config servers:** Przechowują metadane dotyczące shardingu i konfiguracji całego klastra.

5. **Jak definiuje się klucz shardujący w MongoDB i dlaczego jest to istotne?** Klucz shardujący jest polem (lub zestawem pól) w dokumencie, na podstawie którego MongoDB dzieli dane na chucks. Definiuje się go podczas tworzenia kolekcji shardowanej. Jest istotny, ponieważ wpływa na równomierność rozkładu danych między shardy i ma bezpośredni wpływ na wydajność operacji bazodanowych.

6. **W jaki sposób MongoDB zarządza danymi w klastrze shardów?** MongoDB zarządza danymi w klastrze shardów poprzez:
    - **Rozdzielanie danych:** Dzieli dane na chunks i przypisuje je do shardów.
    - **Rebalancing:** Przenosi chunks między shardami, aby zachować równomierne rozłożenie danych.
    - **Replikację:** Każdy shard jest częścią repliki, co zapewnia redundancję i wysoką dostępność.
7. **Jakie są strategie migracji danych w sharding MongoDB?** Strategie migracji danych w shardingu MongoDB obejmują:
    - **Automatyczne przemieszczanie chunks:** MongoDB automatycznie przenosi chunks, aby zbalansować obciążenie shardów.
    - **Manualny balans:** Administrator może ręcznie wywołać operacje balansowania, używając poleceń administacyjnych.
    - **Migracje z narzędziami:** Narzędzia takie jak `mongoimport`, `mongodump`, i `mongorestore` mogą być używane do migracji danych między klastrami.
8. **Jakie są wyzwania związane z konfiguracją i zarządzaniem klastrami shardów w MongoDB?** Wyzwania obejmują:
    - **Skonfigurowanie i zarządzanie:** Konfiguracja shardingu i zarządzanie klastrem wymaga dokładnej znajomości MongoDB i jego architektury.
    - **Równomierność rozłożenia danych:** Utrzymanie równomiernego rozłożenia danych między shardy może być trudne.
    - **Monitorowanie i diagnostyka:** Skuteczne monitorowanie i diagnostyka problemów mogą być skomplikowane w środowiskach rozproszonych.
    - **Optymalizacja wydajności:** Konieczność ciągłej optymalizacji zapytań i kluczy shardujących w celu zapewnienia wydajności.
9. **Jakie są metody monitorowania i diagnostyki wydajności w środowiskach z shardingiem w MongoDB?** Metody obejmują:
    - **Narzędzia MongoDB:** Takie jak `mongostat`, `mongotop`, `serverStatus`, `shardStatus`, oraz `collStats`.
    - **Monitoring zewnętrzny:** Narzędzia jak MongoDB Cloud Manager, Datadog, czy Prometheus.
    - **Logowanie i analiza logów:** Analiza logów MongoDB dla wykrywania problemów z wydajnością.
    - **Dashboards:** Korzystanie z dashboardów i alertów do monitorowania kluczowych metryk w czasie rzeczywistym.
10. **Jakie są najlepsze praktyki dotyczące projektowania, wdrażania i utrzymania sharding MongoDB?** Najlepsze praktyki obejmują:
    - **Przemyślany wybór klucza shardującego:** Klucz powinien zapewniać równomierne rozłożenie danych i minimalizować hot spots.
    - **Monitorowanie i optymalizacja:** Regularne monitorowanie stanu klastra i optymalizacja zapytań oraz konfiguracji shardingu.
    - **Regularne testy:** Testowanie wydajności i odporności klastra, w tym testy failover.
    - **Automatyzacja zadań administracyjnych:** Korzystanie z narzędzi do automatyzacji zadań, takich jak backupy i balansowanie danych.

- **Planowanie pojemności:** Planowanie z wyprzedzeniem na potrzeby skalowania i zarządzania wzrostem danych.
- **Dokumentacja i procedury:** Utrzymanie szczegółowej dokumentacji i procedur zarządzania klastrem shardów.