Projekt zsbd 2 - Mongodb

Maxymilian Kowalski 307411

Zbiór danych

Zbiór danych użyty w projekcie składa się z tweetów e-commerce z Indonezji, zawierających szereg informacji, takich jak ID tweeta, ID użytkownika, treść, data i czas publikacji, liczba retweetów, polubień, a także odniesienia do innych użytkowników i multimediów.

Kolekcje, dokumenty zagnieżdżone, referencje

W ramach projektu zainstalowałem i skonfigurowałem MongoDB, a następnie zaimportowałem dane tweetów z różnych platform e-commerce w Indonezji do oddzielnych kolekcji. Po zaimportowaniu danych, przeprowadziłem proces ich przetwarzania, tworząc dodatkowe kolekcje, takie jak `users` i `conversations`. W kolekcji `users` zgromadziłem informacje o użytkownikach tweetujących, co umożliwiło mi efektywniejsze zarządzanie danymi użytkowników i ich tweetami. Przykładowo, za pomocą poniższego kodu agregacji, wyodrębniłem unikalne dane użytkowników z tweetów i zapisałem je do nowej kolekcji:

```
// Agregacja i grupowanie danych użytkownika z każdej kolekcji tweetów, unikanie duplikatów
var collections = ["Tokopedia", "Lazada", "Bukalapak", "Blibli", "Shopee"];
collections.forEach(function (collection) {
    db[collection].aggregate([
        {
            $group: {
                _id: "$user_id",
                username: { $first: "$username" },
                name: { $first: "$name" },
            },
        },
        {
            $merge: {
                into: "users",
                whenMatched: "keepExisting",
                whenNotMatched: "insert",
            },
        },
    ]);
});
```

W kolekcji `conversations` również zastosowałem referencje, aby przechowywać tweety w ramach jednej konwersacji. Referencje pozwoliły mi na łatwiejszy dostęp do powiązanych tweetów.

```
// Tworzenie kolekcji konwersacji
db.createCollection("conversations");

// Wypełnianie kolekcji konwersacji tweetami grupowanymi według conversation_id
collections.forEach(function (collection) {
    db[collection].aggregate([
        {
            $group: {
                _id: "$conversation_id",
                tweets: { $push: "$$ROOT" },
            },
        },
        {
            $merge: {
                into: "conversations",
                whenMatched: "replace",
                whenNotMatched: "insert",
            },
        },
    ]);
});

// Dodawanie referencji do konwersacji w tweetach
collections.forEach(function (collection) {
    db[collection].find().forEach(function (tweet) {
        var conversation = db.conversations.findOne({
            _id: tweet.conversation_id,
        });
        if (conversation) {
            db[collection].updateOne(
                { _id: tweet._id },
                { $set: { conversation_ref: conversation._id } }
            );
        }
    });
});
```

Różnice Między Dokumentami Zagnieżdżonymi a Referencjami

Dokumenty Zagnieżdżone

Zagnieżdżanie dokumentów pozwala na przechowywanie wszystkich powiązanych informacji w jednym dokumencie. To redukuje liczbę zapytań do bazy danych, ponieważ wszystkie potrzebne informacje są dostępne w jednym miejscu. Jednakże, może to prowadzić do redundancji danych i zwiększać rozmiar dokumentów.

Referencje

Referencje pozwalają na utrzymanie danych w oddzielnych dokumentach i kolekcjach, redukując redundancję. Wymagają wykonania dodatkowych zapytań do bazy danych, aby pobrać powiązane dane. Są bardziej elastyczne w przypadku aktualizacji danych, ponieważ aktualizacja odbywa się w jednym miejscu.

```
// Aby uzyskać tweety wraz z informacjami o użytkownikach:
db.Tokopedia.aggregate([
    {
        $lookup: {
            from: "users",
            localField: "user_ref",
            foreignField: "_id",
            as: "user_info",
        },
    },
    { $unwind: "$user_info" },
    { $limit: 5 }, // Ograniczenie do 5 wyników dla czytelności
]).pretty();
// To zapytanie wykorzystuje $lookup do połączenia tweetów z informacjami o użytkownikach, korzystając z referencji user_ref.
```

Logika biznesowa

Aby poprawnie obsłużyć logikę biznesową dodałem takie funkcje jak dodawanie tweeta, aktualizacja tweeta, autoinkrementacja tweetId oraz inne funkcje przedstawione poniżej:

```
// Logika biznesowa
// dodawanie tweetów
function addTweet(collectionName, tweetData) {
    db[collectionName].insert(tweetData);
}
// aktualizacja tweeta po ID
function updateTweet(collectionName, tweetId, updateData) {
    db[collectionName].update({ _id: tweetId }, { $set: updateData });
}

// autoinkrementacja - sekwencja
db.counters.insert({ _id: "tweetId", seq: 0 });

function getNextSequence(name) {
    var ret = db.counters.findAndModify({
        query: { _id: name },
        update: { $inc: { seq: 1 } },
        new: true,
    });

    return ret.seq;
}

// agregacja
// ilość tweetów na użytkownika
db.Tokopedia.aggregate([{ $group: { _id: "$user_id", count: { $sum: 1 } } }]);

// najpopularniejsze hashtagi
db.Tokopedia.aggregate([
    { $unwind: "$hashtags" },
    { $group: { _id: "$hashtags", count: { $sum: 1 } } },
    { $sort: { count: -1 } },
    { $limit: 5 },
]);
// { "_id" : "#mulaiajadulu", "count" : 3838 }
// { "_id" : "#tokopedia", "count" : 3790 }
// { "_id" : "#blogtokopedia", "count" : 2418 }
// { "_id" : "#tokopediahotlist", "count" : 1928 }
// { "_id" : "#temutoppers", "count" : 1786 }

db.Shopee.aggregate([
    { $unwind: "$hashtags" },
    { $group: { _id: "$hashtags", count: { $sum: 1 } } },
    { $sort: { count: -1 } },
    { $limit: 5 },
]);
// { "_id" : "#shopeeid", "count" : 6100 }
// { "_id" : "#shopeeadaronaldo", "count" : 3802 }
// { "_id" : "#shopeexgfriend", "count" : 2784 }
// { "_id" : "#shopeegfriend", "count" : 2422 }
// { "_id" : "#shopeegfriendtvshow", "count" : 1608 }
```

Indeksy i testy wydajności

Na początku dodałem indeks na tweetId w kolekcji Shopee i porównałem czas przetwarzania zapytania:

```
// indeksy
var start = new Date();
db.Shopee.find({ id: NumberLong("1182099023072780289") }).explain(
    "executionStats"
);
var end = new Date();
print("Czas wykonania zapytania: ", end - start, "ms");

// Czas wykonania zapytania przed indeksem:  119 ms

db.Shopee.createIndex({ id: 1 });

// Czas wykonania zapytania po indeksie:  14 ms
```

Aby indeks kompozytowy miał sens, należy połączyć kolekcje tweetów z różnych kanałów do jednej dużej kolekcji i dodać indeks kompozytowy na userId i np. date. Porównałem wydajność tego indeksu z kolekcją bez indeksu oraz kolekcją z indeksem prostym:

```
var start = new Date();
db.allTweets
    .find({ user_id: someUserId, date: someDate })
    .explain("executionStats");
var end = new Date();
print("Czas wykonania zapytania bez indeksów: ", end - start, "ms");
// Czas wykonania zapytania bez indeksów:  334 ms

db.allTweets.createIndex({ user_id: 1 });

var start = new Date();
db.allTweets
    .find({ user_id: someUserId, date: someDate })
    .explain("executionStats");
var end = new Date();
print("Czas wykonania zapytania z indeksem na user_id: ", end - start, "ms");
// Czas wykonania zapytania z indeksem na user_id:  317 ms

db.allTweets.dropIndex({ user_id: 1 }); // Usuń poprzedni indeks
db.allTweets.createIndex({ user_id: 1, date: 1 });

var start = new Date();
db.allTweets
    .find({ user_id: someUserId, date: someDate })
    .explain("executionStats");
var end = new Date();
print("Czas wykonania zapytania z indeksem kompozytowym: ", end - start, "ms");
// Czas wykonania zapytania z indeksem kompozytowym:  18 ms
```

Transakcje

Transakcje w MongoDB, dostępne od wersji 4.0, pozwalają na grupowanie wielu operacji na dokumentach w jedną jednostkę pracy, która jest atomowa, spójna, odizolowana i trwała (ACID). Oznacza to, że wszystkie operacje w ramach transakcji są wykonywane razem lub nie są wykonywane wcale.

```
// Rozpoczęcie sesji
var session = db.getMongo().startSession();

// Rozpoczęcie transakcji
session.startTransaction();

try {
    // Przykładowe ID tweetów
    const tweetId1 = NumberLong("2983890506");
    const tweetId2 = NumberLong("1182099023072780289");

    // Wykonanie operacji aktualizacji w ramach transakcji
    db.all_tweets.updateOne({ _id: tweetId1 }, { $inc: { likes: 1 } }, { session });
    db.all_tweets.updateOne({ _id: tweetId2 }, { $inc: { likes: 1 } }, { session });

    // Zatwierdzenie transakcji
    session.commitTransaction();
} catch (error) {
    // Wycofanie transakcji w przypadku błędu
    console.error("Transakcja nie powiodła się:", error);
    session.abortTransaction();
} finally {
    // Zamknięcie sesji
    session.endSession();
}
```

W tym scenariuszu:

- Rozpoczynamy sesję i transakcję.
- W ramach transakcji wykonujemy dwie operacje aktualizacji, każda zwiększająca liczbę polubień (`likes`) o 1 dla wybranych tweetów.
- Jeśli wszystkie operacje się powiodą, zatwierdzamy transakcję, w przeciwnym razie wycofujemy ją, co zapewnia, że zmiany są atomowe.
- Na końcu zamykamy sesję.

Używanie transakcji w MongoDB w ten sposób pozwala na grupowanie wielu operacji, które albo wszystkie się powiodą, albo żadna z nich nie zostanie zastosowana, co zapewnia spójność danych.

Inne zaawansowane funkcje

Agregacje

Agregacje w MongoDB pozwalają na przetwarzanie i agregowanie danych w zaawansowany sposób. Możemy wykorzystać potężne operacje rurkowe (pipeline operations) do przetwarzania i analizy tweetów.

```
// Załóżmy, że chcesz znaleźć najpopularniejsze hashtagi w tweetach:
db.allTweets.aggregate([
    { $unwind: "$hashtags" },
    { $group: { _id: "$hashtags", count: { $sum: 1 } } },
    { $sort: { count: -1 } },
    { $limit: 10 },
]);
// { "_id" : "#lazadaid", "count" : 6962 }
// { "_id" : "#bliblisekarang", "count" : 6752 }
// { "_id" : "#shopeeid", "count" : 6100 }
// { "_id" : "#kamisganteng", "count" : 4074 }
// { "_id" : "#mulaiajadulu", "count" : 3838 }
// { "_id" : "#shopeeadaronaldo", "count" : 3802 }
// { "_id" : "#tokopedia", "count" : 3790 }
// { "_id" : "#blogbliblifriends", "count" : 3080 }
// { "_id" : "#fridayhoray", "count" : 2982 }
// { "_id" : "#shopeexgfriend", "count" : 2784 }
```

W tym przykładzie, rozkładamy dokumenty na pojedyncze hashtagi, grupujemy je i zliczamy ich wystąpienia, a następnie sortujemy od najczęściej występujących.

Map/Reduce

Map/Reduce w MongoDB to sposób na przetwarzanie dużych ilości danych i generowanie agregowanych wyników. Jest to szczególnie przydatne w przypadku złożonych operacji, które nie mogą być łatwo osiągnięte za pomocą standardowych operacji agregacji.

```
// Oblicz średnią liczbę retweetów dla każdego użytkownika:
var mapFunction = function () {
    emit(this.user_id, this.retweets_count);
};

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

db.allTweets.mapReduce(mapFunction, reduceFunction, { out: "averageRetweets" });
db.averageRetweets.find().limit(5).pretty();
// { "_id" : 515203442, "value" : 1.0418439120506495 }
// { "_id" : 113271721, "value" : 0.9988526088563039 }
// { "_id" : 72293042, "value" : 40.240273497688754 }
// { "_id" : 280834900, "value" : 2.0223053794365646 }
// { "_id" : NumberLong("2983890506"), "value" : 1.9486887928187577 }
```

Ocena zaawansowanych funkcji

Agregacje są niezwykle potężne dla analizy danych, pozwalają na szybkie i elastyczne przetwarzanie dużych ilości danych.
Map/Reduce, choć mniej wydajne niż operacje agregacji, oferuje większą elastyczność w przetwarzaniu i generowaniu skomplikowanych zestawów danych.

Podsumowanie

Wykorzystanie MongoDB, bazy dokumentowej, w analizie tweetów e-commerce z Indonezji, w porównaniu do tradycyjnych baz danych relacyjnych, oferuje kilka kluczowych różnic i zalet:

- **Elastyczność Schematu**: MongoDB pozwala na przechowywanie dokumentów o zmiennych strukturach, co jest idealne dla dynamicznie zmieniających się danych, jakimi są tweety.

- **Skalowalność**: MongoDB jest zaprojektowana do łatwego skalowania poziomego, co jest korzystne przy obsłudze dużych zbiorów danych, takich jak tweety.

- **Zapytania i Agregacje**: MongoDB oferuje zaawansowane możliwości zapytań i agregacji, które są przydatne w analizie dużych zbiorów danych, na przykład w identyfikowaniu trendów i wzorców w tweetach.

- **Zarządzanie Dużymi Obiektami**: Funkcje takie jak GridFS ułatwiają zarządzanie dużymi obiektami (np. obrazami w tweetach), co może być bardziej skomplikowane w bazach relacyjnych.

Podsumowując, MongoDB zapewnia większą elastyczność i skuteczność w analizie i zarządzaniu dużymi, zmiennymi zbiorami danych, co czyni ją dobrze przystosowaną do projektów związanych z przetwarzaniem danych społecznościowych, takich jak tweety. Jednak w sytuacjach wymagających bardziej zaawansowanego zarządzania relacjami i transakcjami, tradycyjne bazy danych relacyjne mogą być bardziej odpowiednie.