ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

NOME CORSO

Fintech Software Developer

Unità Formativa (UF): Basi di dati SQL

Docente: Durando Giulio

Titolo argomento: Pipeline di Aggregazione

Operazioni di aggregazione

Le operazioni di aggregazione elaborano più documenti e restituiscono risultati calcolati. È possibile utilizzare le operazioni di aggregazione per:

- Raggruppare i valori di più documenti insieme.
- Eseguire operazioni sui dati raggruppati per restituire un unico risultato.
- Analizzare le modifiche dei dati nel tempo.

Per eseguire operazioni di aggregazione è possibile utilizzare:

- Pipeline di aggregazione (cioè un elenco ordinato di operatori di aggregazione), che sono il metodo preferito per eseguire le aggregazioni.
- Metodi di aggregazione a scopo singolo, semplici ma privi delle funzionalità di una pipeline di aggregazione.

Funzionamento di una pipeline di aggregazione MongoDB?

Il diagramma di figura sotto illustra una tipica pipeline di aggregazione MongoDB :

Una pipeline di aggregazione è costituita da una o più fasi che elaborano i documenti:

- Fase **$match**: filtra i documenti con cui dobbiamo lavorare, quelli che soddisfano le nostre esigenze
- Fase **$group** – fa il lavoro di aggregazione
- Fase **$sort**: ordina i documenti risultanti nel modo richiesto (crescente o decrescente)

L'input della pipeline può essere una singola collection, altre possono essere unite in un secondo momento lungo la pipeline.

La pipeline esegue quindi trasformazioni successive sui dati fino al raggiungimento del nostro obiettivo.

In questo modo, possiamo scomporre una query complessa in fasi più semplici, in ognuna delle quali completiamo un'operazione diversa sui dati. Quindi, entro la fine della pipeline di query, avremo ottenuto tutto ciò che volevamo.

Questo approccio ci consente di verificare se la nostra query funziona correttamente in ogni fase, esaminando sia l'input che l'output. L'output di ogni fase sarà l'input del successivo

Il comando che permettere di aggregare dei documenti è **db.collection.aggregate()**

db.collection.aggregate()

Nel metodo **db.collection.aggregate()**, le fasi della pipeline vengono visualizzate in una matrice. I documenti passano attraverso le fasi in sequenza. Tutte le fasi tranne **$out**, **$merge** e **$geoNear** possono essere visualizzate più volte in una pipeline.

Per creare una sequenza di aggregazione si deve utilizzare la seguente sintassi in MongoDB Shell:

```
db.<collection>.aggregate([
  {   <$fase1>  },
  {   <$fase2>  }
  ...
])
```

Operatori normalmente usati in una pipeline

$match	Filtra il flusso di documenti per consentire solo ai documenti corrispondenti di passare senza modifiche alla fase successiva della pipeline. **$match** utilizza query MongoDB standard. Per ogni documento di input, genera un documento (una corrispondenza) o zero documenti (nessuna corrispondenza).
$group	Raggruppa i documenti di input in base a un'espressione identificativa specificata e applica le espressioni dell'accumulatore, se specificate, a ciascun gruppo. Esaurisce tutti i documenti di input e genera un documento per ogni gruppo distinto. I documenti di output contengono solo il campo dell'identificatore e, se specificato, i campi accumulati.
$sort	Riordina il flusso di documenti in base a una chiave di ordinamento specificata. Cambia solo l'ordine; i documenti rimangono invariati. Per ogni documento di input, emette un documento.
$count	Passa un documento alla fase successiva che contiene un conteggio del numero di documenti selezionati nella fase.
$sum	Calcola e restituisce la somma collettiva dei valori numerici. ignora i valori non numerici.

$match()

$match filtra i documenti per passare solo i documenti che soddisfano le condizioni specificate alla fase successiva della pipeline. La sintassi della fase in cui si usa il comando $match è la seguente

```
{ $match: { <query> } }
```

Esempio

```
$match:{borough:"Manhattan"}
```

In questo caso l'uso dell'operatore $match selezionerà per la fase successiva della pipeline di aggregazione solo i ristoranti in cui il campo "borough" vale "Manhattan"

$group()

$group raggruppa i documenti in gruppi in base a una "chiave di gruppo". L'output è un documento per ogni chiave di gruppo univoca.

Una chiave di gruppo è spesso un campo o un gruppo di campi. La chiave di gruppo può anche essere il risultato di un'espressione. Si deve utilizzare il campo ***_id*** nella fase ***$group*** della pipeline per impostare la chiave del gruppo

Nell'esempio seguente viene usato l'operatore **$group** per raggruppare i ristoranti per quartiere.

```
db.restaurants.aggregate(
   [
      {$group:
          { _id:"$borough" }
       }
    ]
)
```

Il campo per cui si vuole fare il raggruppamento va sempre messo tra virgolette e preceduto dal simbolo **$**. Il risultato di questa aggregate è il seguente

```
[
  { _id: 'Queens' },
  { _id: 'Missing' },
  { _id: 'Brooklyn' },
  { _id: 'Staten Island' },
  { _id: 'Manhattan' },
  { _id: 'Bronx' }
]
```

$sort()

$sort() ordina tutti i documenti di input e li restituisce alla pipeline in ordine crescente o decrescente. La sintassi della fase in cui si usa il comando $sort è la seguente

```
{ $sort: { <field1>: <sort order>, <field2>: <sort order> ... } }
```

Se si esegue l'ordinamento su più campi, l'ordinamento viene valutato da sinistra a destra. Ad esempio, nel modulo precedente, i documenti vengono prima ordinati per <field1>. Quindi i documenti con gli stessi <field1>valori vengono ulteriormente ordinati per <field2>. È possibile ordinare su un massimo di 32 chiavi

Il parametro sort order deve valere **1** se si vuole l'ordinamento crescente, **-1** se si vuole l'ordinamento decrescente.

Il comando seguente usa la fase **$sort** per ordinare sul campo borough:

```
db.restaurants.aggregate(
   [
     { $sort : { borough : 1 } }
   ]
)
```

In questo caso, l'ordinamento potrebbe non essere coerente, poiché il campo borough contiene valori duplicati sia per Manhattan che per Brooklyn. I documenti vengono restituiti in ordine alfabetico da borough, ma l'ordine di quei documenti con valori duplicati per borough potrebbe non essere lo stesso in più esecuzioni dello stesso ordinamento come si può vedere sotto

$count

Passa un documento alla fase successiva che contiene un conteggio del numero di documenti selezionati nella fase

Nell'esempio seguente viene usato l'operatore **$group** per raggruppare i ristoranti per quartiere. L'operatore **$count {}** permette di effettuare il conteggio dei ristoranti per quartiere.

```
db.restaurants.aggregate(
                    [
                      {$group:
                         { _id:"$borough",
                           numero: {$count:{} }
                         }
                      }
                    ]
                   )
```

L'output di questa aggregate è il seguente:

```
{ _id: 'Brooklyn', numero: 6086 },
{ _id: 'Staten Island', numero: 969 },
{ _id: 'Queens', numero: 5656 },
{ _id: 'Manhattan', numero: 10259 },
{ _id: 'Bronx', numero: 2338 },
{ _id: 'Missing', numero: 51 }
```

$sum

Calcola e restituisce la somma collettiva dei valori numerici. $sum ignora i valori non numerici.

Gli operatori della pipeline di aggregazione creano espressioni da utilizzare nelle fasi della pipeline di aggregazione. Vediamo l'elenco degli operatori di pipeline di aggregazione.

Operatori di espressioni aritmetiche

$abs	Restituisce il valore assoluto di un numero
$add	Somma due o più numeri o una data. Se uno degli argomenti è una data, l'altro argomento viene usato come millisecondi da aggiungere alla data.
$ceil	Restituisce il numero intero più piccolo maggiore o uguale al numero specificato.
$divide	Divide uno o più numeri per un altro e restituisce il risultato.
$exp	L'operatore viene utilizzato per elevare il numero di Eulero all'esponente specificato e restituisce il risultato.
$floor	Restituisce l'intero più grande minore o uguale al numero specificato
$ln	Calcola il logaritmo naturale di un numero e restituisce il risultato come double
$log	Calcola il log di un numero per la base specificata e restituisce il risultato come double
$log10	Calcola il logaritmo in base 10 di un numero e restituisce il risultato come double.
$mod	Divide un numero con un altro e restituisce il resto.
$multiply	Fornisce il prodotto di due o più numeri.
$pow	Eleva il numero all'esponente dato e restituisce il risultato

$round	Arrotonda un numero a un intero o a un numero con un numero di cifre decimali specificate.
$sqrt	Restituisce su un double la radice quadrata di un numero positivo
$subtract	Sottrae due o più numeri per restituire la differenza
$trunc	Tronca il numero dalla posizione decimale specificata.

Operatori sugli array

$arrayElemAt	Restituisce l'elemento all'indice di matrice specificato.
$arrayToObject	Converte una matrice in un unico documento.
$concatArrays	unisce più array e restituisce l'array concatenato.
$filter	Seleziona un sottoinsieme di una matrice per restituire il risultato in base alla condizione specificata
$in	Restituisce un valore booleano che indica se il valore specificato è nella matrice (true) o no (false).
$indexOfArray	Cerca nell'array l'occorrenza di un valore specificato e restituisce l'indice dell'array della prima occorrenza.
$isArray	Determina se l'operando è un array e restituisce un valore booleano se l'operando è un array.
$map	Attribuisce un valore a ogni elemento in una matrice e restituisce una matrice con il risultato applicato.
$objectToArray	Converte un documento in una matrice
$range	Restituisce una matrice i cui elementi sono una sequenza di numeri
$reduce	applica un'espressione a ogni elemento in una matrice e li combina in un unico valore.
$reverseArray	Restituisce un array con gli elementi in ordine inverso
$size	Conta e restituisce il numero totale di elementi in una matrice
$slice	• risulta in un sottoinsieme di una matrice
$zip	traspone un array in modo che il primo elemento dell'array di output sia un array contenente il primo elemento del primo array di input.

Esempio di pipeline di aggregazione

Il seguente esempio di pipeline di aggregazione contiene due fasi e restituisce il numero di ristoranti del quartiere di Manhattan suddivisi per tipo di cucina:

```
db.restaurants.aggregate
(
    [
        // Fase 1: Filtra i documenti relativi ai ristoranti del quartiere di Manhattan
        {
                $match:{borough:"Manhattan"}
        },
        // Fase 2: Raggruppa i rimanenti documenti per tipo cucina e calcola la quantità totale
        {
                $group:{_id:"$cuisine", numero: { $sum: 1 }}
        }
    ]
```

)

Il risultato è il seguente

```
{ _id: 'Jewish/Kosher', numero: 73 },
{ _id: 'Middle Eastern', numero: 54 },
{ _id: 'Other', numero: 371 },
{ _id: 'Chinese/Cuban', numero: 4 },
{ _id: 'Hawaiian', numero: 2 },
{ _id: 'Juice, Smoothies, Fruit Salads', numero: 125 },
{ _id: 'Pizza', numero: 339 },
{ _id: 'Café/Coffee/Tea', numero: 680 },
{ _id: 'Fruits/Vegetables', numero: 1 },
{ _id: 'Southwestern', numero: 4 },
{ _id: 'Greek', numero: 35 },
{ _id: 'Thai', numero: 154 },
{ _id: 'Russian', numero: 14 },
{ _id: 'Continental', numero: 32 },
{ _id: 'Steak', numero: 61 },
{ _id: 'Indian', numero: 152 },
{ _id: 'Ethiopian', numero: 14 },
{ _id: 'African', numero: 17 },
{ _id: 'Egyptian', numero: 5 },
{ _id: 'Californian', numero: 1 }
```

La fase **$match**: Filtra i documenti dei ristoranti del quartiere di Manhattan
La fase **$group**: Raggruppa i documenti rimanenti per tipo di cucina.
$sum:1 calcola la quantità totale dei ristoranti per tipo di cucina. Il totale viene archiviato nel campo numero restituito dalla pipeline di aggregazione.