COESIONE ITALIA 21-27 PIEMONTE

Cofinanziato dall'Unione europea

REGIONE PIEMONTE

Fondamenti di basi di dati
Docente: Alessandro Fiori
Titolo argomento:
Esercizi su modellazione in MongoDB

Intervento realizzato da ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

Esercizio 1

Dobbiamo progettare un database MongoDB per memorizzare i seguenti dati sui libri.

Ogni libro è identificato dal codice ISBN ed è caratterizzato da titolo, sottotitolo, numero di pagine, lingua ("IT", "EN", "FR", ecc.) e data di pubblicazione.

Il sistema deve memorizzare i prezzi dei libri (ad esempio, 12,34 dollari), per ogni paese e per ogni formato (ad esempio, "ebook", "paperback"). Si noti che i prezzi dei libri vengono ricercati in base alla loro valuta (ad esempio, tutti i prezzi dei libri in dollari o sterline) e separatamente anche in base al loro importo (ad esempio, superiore a 10,0).

Ogni libro appartiene a una o più categorie (ad esempio, "fantasy", "classico"). Le categorie sono organizzate in una struttura gerarchica (ad esempio, la categoria "bambini" include "narrativa", che include "fantasy", "classico", ecc.).

- Scrivete un documento di esempio per ogni collezione del database.
- Indicate esplicitamente i modelli di progettazione utilizzati.

Soluzione

```
LIBRO
{
   _id: <string>, // ISBN
   titolo: <string>,
   sottotitolo: <string>,
   pagine: <number>,
   lingua: <string>,
   pubblicazione: <date>,
   prezzi: [ // pattern attributi
      {
         valore: <number>,
         valuta: <string>,
         paese: <string>,
         formato: <string>
      }
   ],
   categorie: [<string>], // ["fantasy", "classico"]
   ancestor: [<string>] // pattern ad albero
}
```

Pattern

Modello di attributo per tracciare il prezzo in base al formato del libro e al suo paese.

Modello ad albero per tracciare tutti gli antenati nella gerarchia delle categorie.

Esercizio 2

Dobbiamo progettare un database MongoDB per la gestione di un sito web simile ad AirBnb in cui vengono raccolte le recensioni degli utenti sugli affitti degli immobili.

Ogni proprietà è caratterizzata da un nome, dal prezzo per notte e dalla sua posizione (codice postale, città e paese). Il prezzo per notte è caratterizzato dall'importo e dalla valuta.

Ogni recensione è caratterizzata dal timestamp, dal testo della recensione, dal voto e dalle informazioni sull'autore e si riferisce a una sola proprietà. Quando si accede a una recensione, vengono visualizzati solo il nome e l'e-mail dell'autore della recensione.

Quando si accede a una proprietà, vengono visualizzate solo le 10 recensioni più recenti, ognuna delle quali presenta solo il testo, il voto e il timestamp. Ogni pagina della proprietà riporta il numero di recensioni totali ricevute e il voto medio.

- Scrivete un documento di esempio per ogni collezione del database.
- Indicate esplicitamente i modelli di progettazione utilizzati.

Soluzione

```
PROPRIETA
{
   _id: ObjectId(),
   nome: <string>,
   prezzo: {
      importo: <number>,
      valuta: <string>
   },
   posizione: {
      citta: <string>,
      cap: <string>,
      paese: <string>,
      loc: {
         type: <string>
         coord: [<number>]
      }
   },
   recensioni: [ // pattern sottoinsieme + extended reference
      {
         id: ObjectId(),
         testo: <string>,
         timestamp: <datetime>,
         voto: <number>
      }
      /*
      [
         {
            id: ObjectiD('dsjkfhsdkjhfdsk')
            timestamp: "2024-06-25T00:01:00",
            testo: "Lorem ipsum ....",
            voto: 5
         },
         {
            id: ObjectiD('uydgsyag')
            timestamp: "2024-06-25T10:03:00",
            testo: "Lorem ipsum ....",
            voto: 6.5
         }
      ]
      */
   ],
   numRecensioni: <number>, // pattern calcolato
   votoTotale: <number> // pattern calcolato
}


RECENSIONE
{
   _id: ObjectId(),
   timestamp: <datetime>,
   testo: <string>,
   voto: <number>,
   autore: {
      nome: <string>,
      email: <string>
```

```
    },
    proprieta: { // extended reference
        id: ObjectId(),
        nome: <string>
    }

}
```

Pattern

Modello del sottoinsieme per tenere traccia solo delle recensioni più recenti di ogni proprietà.

Modello extended reference per le proprietà, per visualizzare le informazioni rilevanti delle recensioni senza collegamenti.

Pattern calcolato per memorizzare il numero complessivo di recensioni e il loro voto medio.

Esercizio 3

Ci viene richiesto di progettare il database per la gestione di una richiesta di assicurazione.

Ogni cliente è caratterizzato da nome, cognome, data di nascita, modalità di contatto e indirizzo di residenza. Le modalità di contatto possono essere, ad esempio, il numero di telefono, l'e-mail, un nome utente Skype, ecc. L'indirizzo è caratterizzato dal nome della via, dal numero civico, dalla città, dalla provincia e dalla regione.

Ogni polizza assicurativa è firmata da un cliente ed è caratterizzata dal tipo di polizza, dalla data di firma e dall'elenco delle voci incluse. Gli elementi inclusi possono essere modificati dal cliente nel tempo, aggiungendone di nuovi o rimuovendone di indesiderati.

L'applicazione deve recuperare in modo efficiente i dati assicurativi attualmente attivi. Tuttavia, le versioni precedenti delle polizze devono essere disponibili su richiesta.

Oltre ai dati assicurativi, è necessario mostrare solo il nome, il cognome e le modalità di contatto del cliente.

- Scrivete un documento di esempio per ogni collezione del database.
- Indicate esplicitamente i modelli di progettazione utilizzati.

Soluzione

```
CLIENTE
{
    _id: ObjectId(),
    nome: <string>,
    cognome: <string>,
    data_nascita: <date>,
    contatto: { // polimorfico
        email: <string>,
        tel: <string>,
        skype: <string>
    },
    contatto2: [ // pattern attributi
        { type: <string>,
            valore: <string>
        }
    ],
    indirizzo: {
        via: <string>,
        numero: <string, number>,
        citta: <string>,
        provincia: <string>,
        regione: <string>
    }
}

POLIZZA_ATTUALE
```

```
{
   _id: ObjectId(),
   tipo: <string>,
   data: <date>,
   voci: [<string>],
   versione: <number>,
   cliente:{ // extended reference
      id: ObjectId(),
      nome: <string>,
      cognome: <string>,
      contatto: { // polimorfico
         email: <string>,
         tel: <string>,
         skype: <string>
      }
   }
}

POLIZZE_VECCHIE // pattern document versioning
{
   _id: ObjectId(),
   tipo: <string>,
   data: <date>,
   voci: [<string>],
   versione: <number>,
   cliente:{ // extended reference
      id: ObjectId(),
      nome: <string>,
      cognome: <string>,
      contatto: { // polimorfico
         email: <string>,
         tel: <string>,
         skype: <string>
      }
   }
}
```

Pattern

Versioning dei documenti per l'aggiornamento delle polizze assicurative.
Modello polimorfico per tracciare solo i metodi di contatto disponibili.
Extended reference per le informazioni sul cliente.

Esercizio 4

Progettare un database MongoDB per gestire un magazzino per la consegna di pacchi secondo i seguenti requisiti.

I clienti del servizio di consegna pacchi sono cittadini identificati dal loro codice fiscale.
Possono essere mittenti o destinatari dei pacchi consegnati.
Sono caratterizzati da nome, cognome, indirizzo e-mail, numero di telefono e da diversi indirizzi, uno per ogni tipo, ad esempio un indirizzo di fatturazione, un indirizzo di casa, un indirizzo di lavoro, ecc.
Ogni indirizzo è composto da nome della via, numero civico, codice postale, città, provincia e Paese.

I pacchi sono caratterizzati da un codice a barre unico e dalle loro dimensioni fisiche (in particolare: larghezza, altezza, profondità e peso).
Tutte le larghezze, le altezze e le profondità sono sempre espresse in metri.
Tutti i pesi sono sempre espressi in chilogrammi.

Le informazioni sul destinatario e sul mittente necessarie per la consegna di ogni pacco devono essere sempre disponibili quando si accede ai dati di un pacco.

Le informazioni sul destinatario e sul mittente necessarie per la consegna di un pacco sono: nome e cognome, via, numero civico, codice postale, città, provincia e paese.

Ad esempio, i dati del destinatario possono essere: Mario Rossi, corso Duca degli Abruzzi, 24, 10129, Torino, Torino, Italia.

Il magazzino pacchi è suddiviso in diverse aree.

Ogni area è identificata da un codice univoco, ad esempio "area_51" ed è composta da diverse linee.

Ogni linea è identificata da un codice univoco, ad esempio "linea_12", e ospita diversi rack.

Ogni rack è identificato da un codice univoco, ad esempio "rack_33", ed è composto da scaffali.

Ogni pacco viene collocato su uno specifico scaffale del magazzino, identificato da un codice univoco, ad esempio "scaffale_99".

Il database deve tracciare la posizione di ogni pacco all'interno del magazzino.

Dato un pacco, il database deve essere progettato per fornire in modo efficiente la sua posizione completa, dallo scaffale, fino all'area, passando per lo scaffale e la linea.

Dato un cliente, il database deve essere progettato per fornire in modo efficiente tutti i suoi pacchi come mittente e tutti i suoi pacchi come destinatario.

- Scrivete un documento di esempio per ogni collezione del database.
- Indicate esplicitamente i modelli di progettazione utilizzati.

Soluzione

```
CLIENTE
{
   _id: <string>, // Codice fiscale
   nome: <string>,
   cognome: <string>,
   email: <string>,
   telefono: <string>,
   indirizzi: { // polimorfismo
      casa: {
         via: <string>,
         numero: <string>,
         cap: <string>,
         citta: <string>,
         provincia: <string>,
         paese: <string>
      },
      lavoro: {
         via: <string>,
         numero: <string>,
         cap: <string>,
         citta: <string>,
         provincia: <string>,
         paese: <string>
      },
      indirizzi2: [ // pattern attributi
         {
            tipo: <string>,
            via: <string>,
            numero: <string>,
            cap: <string>,
```

```
            citta: <string>,
            provincia: <string>,
            paese: <string>
        }
    ]
  },
  pacchi_mittente: [<string>], // outliers
  pacchi_dest: [<string>],
  has_extra_mittente: <bool>,
  has_extra_dest: <bool>
}
```

PACCO

```
{
  _id: <string>, // barcode
  dim: {
    larghezza: <number>,
    altezza: <number>,
    profondita: <number>,
    peso: <number>
  },
  mittente: { //extended reference
    id: <string>,
    nome: <string>,
    cognome: <string>,
    indirizzo: {
      via: <string>,
      numero: <string>,
      cap: <string>,
      citta: <string>,
      provincia: <string>,
      paese: <string>
    }
  },
  destinatario: { //extended reference
    id: <string>,
    nome: <string>,
    cognome: <string>,
    indirizzo: {
      via: <string>,
      numero: <string>,
      cap: <string>,
      citta: <string>,
      provincia: <string>,
      paese: <string>
    }
  },
  posizione: [<string>] // pattern ad albero

}
```

Pattern

Modello ad albero per la posizione.
L'elenco completo degli antenati del modello ad albero è obbligatorio.
L'antenato genitore del modello ad albero è facoltativo.

Modello extended reference per le informazioni sull'indirizzo del destinatario e del mittente.
Gli _id del destinatario e del mittente sono necessari per cercare tutti i pacchi di un determinato cliente.

Nessuna collezione per le aree, dal momento che non vengono tracciati dati se non il loro codice.

Esercizio 5

Progettate un database MongoDB per gestire le esposizioni museali in base ai seguenti requisiti.

I musei sono caratterizzati da nome, indirizzo, numero di telefono e sito web (se disponibile). L'indirizzo è composto da coordinate geografiche, nome e numero della via, codice postale e città.

Gli oggetti esposti nei musei sono identificati da un numero progressivo e caratterizzati da un titolo, una descrizione e un elenco di nomi di autori.
Gli oggetti sono classificati come reperti archeologici, dipinti o sculture.
Il database deve registrare tutte le caratteristiche principali di ogni oggetto, come le dimensioni (larghezza, altezza, peso, ecc.). Ogni caratteristica ha almeno un nome e un valore, ed eventualmente un'unità di misura. Ad esempio, il materiale principale è una caratteristica di un reperto archeologico, le dimensioni geometriche sono caratteristiche di un dipinto.
Per ogni oggetto deve essere registrato il museo a cui appartiene, con il nome del museo frequentemente consultato insieme all'oggetto stesso.

Ogni museo ospita diverse mostre.
La mostra è caratterizzata da un titolo, da una descrizione e dall'elenco dei nomi dei curatori.
È necessario registrare tutti gli oggetti associati a ciascuna mostra, che possono essere nell'ordine delle centinaia.
Un oggetto può far parte di diverse mostre.
Inoltre, ogni mostra può essere ospitata da più musei in periodi diversi.
È necessario registrare le date di inizio e fine di ogni mostra in ogni museo.

Dato un oggetto, il database deve essere progettato per fornire in modo efficiente il nome del museo che lo possiede.
Data una mostra, il database deve essere progettato per fornire in modo efficiente il nome del museo e le coordinate geografiche in cui è stata ospitata.
Inoltre, data una mostra, deve essere restituito in modo efficiente l'elenco degli oggetti inclusi nella mostra e il loro numero.

- Scrivete un documento di esempio per ogni collezione del database.
- Indicate esplicitamente i modelli di progettazione utilizzati.

Soluzione

```
MUSEO
 {
   _id: ObjectId(),
   nome: <string>,
   indirizzo: {
      via: <string>,
      numero: <string>,
      cap: <string>,
      ciita: <string>,
      geo: {
         type: <string>,
         coord: [<number>]
      }
   },
   tel: <string>,
```

```
  sito_web: <string> // polimorfico
}

OGGETTI
{
  _id: <number>,
  titolo: <string>,
  descrizione: <string>,
  autori: [<string>],
  categoria: <string>,
  caratteristiche: [ // pattern attributi+ polimorfico
      {
        nome: <string>,
        valore: <string, number>,
        uom: <string>
      }
  ],
  museo: { // extended reference
     id: ObjectId(),
     nome: <string>
  }
}

MOSTRA // modello a bucket
{
  _id: ObjectId(),
  titolo: <string>,
  descrizione: <string>,
  curatori: [<string>],
  min_data: <date>,
  max_data: <date>,
  nedizioni: <number>,
  oggetti: [<number>],
  edizioni: [
      {
        inizio: <date>,
        fine: <date>,
        museo: { // extended reference
           id: ObjectId(),
           nome: <string>
        }
      }
  ]

}
```

Pattern

Pattern polimorfico per tracciare le informazioni del sito web nella collezione del museo.
Pattern degli attributi (con pattern polimorfo) per tracciare le caratteristiche di ogni elemento.
Pattern extended reference per tracciare le informazioni del museo associate a ciascun oggetto.
Pattern Bucket con pattern di riferimento esteso per tracciare quando una mostra è ospitata in un museo.

Esercizio 6

Progettare un database MongoDB per gestire i corsi online secondo i seguenti requisiti.

I docenti sono caratterizzati da nome, cognome, e-mail e dall'elenco delle materie che possono insegnare (ad esempio, matematica, elettronica, ecc.). Ogni insegnante può avere uno o più profili online su diverse piattaforme (ad esempio, Facebook, LinkedIn, Wikipedia, ecc.). Per ogni profilo online, se disponibile, il database tiene traccia dell'URL corrispondente del profilo (ad esempio, https://en.wikipedia.org/wiki/Ranjitsinh_Disale). Si noti che per ogni docente e per ogni piattaforma può esistere al massimo un profilo. Un docente può insegnare diversi corsi.

I corsi sono caratterizzati da un nome, un programma, un elenco di parole chiave e il docente. Ogni corso ha diverse edizioni. Per ogni edizione sono noti la data di inizio, la data di fine e il numero di studenti iscritti.

Dato un corso, il database deve essere progettato per fornire in modo efficiente il nome, il cognome e l'e-mail del docente.

Inoltre, dato un corso, devono essere restituiti in modo efficiente il numero di edizioni e il numero medio di studenti iscritti per ogni edizione.

I docenti sono tipicamente recuperati per materia (ad esempio, tutti quelli che insegnano matematica) e per piattaforma di profilo online (ad esempio, tutti quelli che hanno una pagina wikipedia).

- Scrivete un documento di esempio per ogni collezione del database.
- Indicate esplicitamente i modelli di progettazione utilizzati.

Soluzione

```
Teacher
{
_id: ObjectId(),
name: <string>,
surname: <string>,
email: <string>,
profiles: {
      facebook: <url>,
      linkedin: <url>,
      ….
}
subjects: [ <string>]
}

Course
{
_id: ObjectId(),
name: <string>,
syllabus:<string>,
keywords: [ <string> ],
teacher: {
      _id: ObjectId(),
name: <string>,
surname: <string>,
email: <string>,
}
editions: [
{start: <date>,
end: <date>,
n_students: <number>
```

]
n_editions: <number>,
tot_students: <number>
}

Pattern

Modello polimorfico per tracciare le informazioni del profilo online nella collezione Teacher.
Pattern extended reference per tenere traccia delle informazioni sul docente associate a ogni corso.
Modello a bucket per tenere traccia di quando viene fornito un corso.
Modello calcolato per gli studenti medi di ogni edizione

Esericizio 7

Progettare un database MongoDB per memorizzare le recensioni di hotel da un sito web secondo i seguenti requisiti.

I dati da visualizzare sul sito web delle recensioni per ogni hotel includono il nome dell'hotel, il numero di stelle e l'elenco dei servizi offerti, ad esempio wifi gratuito, baby parking, animali domestici ammessi, ecc. Per ogni hotel devono essere sempre visualizzate le informazioni sulla struttura e le 10 migliori recensioni. Le informazioni sulla struttura sono costituite dall'indirizzo, dalla città e dal Paese. Inoltre, l'indirizzo del sito web ufficiale può essere incluso nelle informazioni sulla struttura.

Ogni recensione è composta da un timestamp, un punteggio (ad esempio, 4,5), il nickname dell'autore, il numero di "mi piace" e una descrizione testuale. Ogni recensione si riferisce a un hotel specifico.

Dato un hotel, il database deve essere progettato per fornire in modo efficiente tutti i dati che descrivono l'hotel, le sue 10 recensioni migliori (quelle con il maggior numero di "mi piace"), il numero totale di recensioni e il loro punteggio medio.
Invece, data una recensione, il database deve fornire in modo efficiente il nome dell'hotel, il numero di stelle e la città.

- Scrivete un documento di esempio per ogni collezione del database.
- Indicate esplicitamente i modelli di progettazione utilizzati.

Soluzione

Hotel
{
_id: ObjectId(),
name: <string>,
stars: <number>,
services: [<string>],
venue: {
    address: <url>,
    city: <string>,
    country: <string>,
    website: <url>
},
top_reviews: [
{_id: ObjectId(),
timestamp: <date>,
score: <number>,
nickname: <string>,
likes: <number>,

description: <string> }
],
tot_reviews: <number>,
avg_score: <number>
}

Review
{_id: ObjectId(),
timestamp: <date>,
score: <number>,
nickname: <string>,
likes: <number>,
description: <string>,
hotel: {
_id: ObjectId(),
name: <string>,
stars: <number>,
city: <string>
 }
}

Pattern

Pattern polimorfico per tracciare le informazioni sulla sede nella collezione di hotel (a causa delle informazioni facoltative sul sito web).
Pattern di sottoinsieme per tracciare le 10 migliori recensioni per ogni hotel.
Pattern calcolato per il punteggio medio e il numero totale di recensioni di ogni hotel.
Extended reference per la collezione di recensioni per mostrare le informazioni sull'hotel.