Progetto di Basi di Dati 2
NBAstats

Di Gregorio Emanuele, Falchetta Dario

Anno Accademico 2023-2024

Abstract

Il progetto consiste nella creazione di un database di tipo non-relazionale, in particolare basato su MongoDB, contenente informazioni relative alla stagione 2022-2023 dell'NBA, il campionato di basket americano, oltre poi alla creazione di un'interfaccia web che si appoggia sulla libreria Flask di Python per l'interazione con il database.

Indice

1 Introduzione

Il database è costruito a partire da due dataset reperibili su Kaggle, NBA dataset e 2022-2023 NBA Player Stats:

- dal primo sono stati estratti i dati relativi alle partire che sono state giocate durante il campionato 2022-2023 e le informazioni delle squadre che vi hanno preso parte;
- dal secondo sono state considerate le statistiche dei singoli giocatori che hanno partecipato a questo campionato.

2 Creazione database

Il database prevede la presenza di tre collections, costruite a partire dai dati descritti, ovvero una contenente le informazioni generiche delle squadre, una per i giocatori e le loro statistiche relative alla stagione 22-23 ed una contenente i dati delle singole partite. Inoltre, per approfittare dell'aspetto non strutturato di MongoDB, i riferimenti tra una collection e l'altra sono implementati mediante *referenced relationship*. In particolare:

- il documento relativo ad ogni squadra presenta un campo chiamato *players*, un array contenente gli ObjectID di ogni giocatore che durante la stagione ha giocato in quella squadra, in modo da creare un collegamento Teams-Players;
- il documento relativo ad ogni partita presenta due campi aggiuntivi, chiamati *team_id_home* e *team_id_away* che indicano l'ObjectID rispettivamente della squadra che ha giocato in casa e quella che ha giocato fuori casa per quella partita, in modo da creare un collegamento Games-Teams.

Abbiamo scelto questa strada piuttosto quella di una *embedded relationship* in quanto le query che abbiamo effettuato sulla base di dati giustificavano l'esistenza stand-alone di questi documenti, in modo da permettere, ad esempio, l'accesso alle informazioni di un giocatore, senza dover necessariamente passare per il documento relativo alla sua squadra. Inoltre, siccome sarebbe teoricamente possibile aggiungere molte altre informazioni per i giocatori, se la dimensione dell'embedded document dovesse diventare troppo elevata, questo potrebbe influenzare le performance di lettura e di scrittura.

Di seguito saranno descritti gli steps di pre-processing effettuati sui dati prima della creazione dei vari documenti, dopodiché sarà descritto il modo in cui viene implementato l'embedding in modo totalmente automatico.

2.1 Creazione collezione *Players*

Il codice Python usato per la creazione della collezione *Players* è il più semplice tra i tre: il file csv viene letto in un dataframe della libreria *Pandas*, dopodiché alcune colonne non interessanti vengono rimosse, per poi inserire tutte le righe del dataframe, ciascuna delle quali rappresenta un giocatore, in un documento separato del database.

2.2 Creazione collezione *Teams*

Le operazioni effettuate sulla collezione *Teams* sono le stesse della collezione *Players*. In aggiunta, prima dell'inserimento, è stata aggiunta una variabile che indicasse se una data squadra appartenesse alla *Eastern conference* o alla *Western conference*, un'ulteriore suddivisione che viene fatta tra le squadre del campionato a seconda della loro provenienza geografica rispetto agli Stati Uniti.

2.3 Creazione collezione *Games*

Anche per questa collezione, le operazioni di base sono le stesse delle precedenti. Uno step importante che viene condotto è la rimozione di informazioni inutili: il dataset originale presenta due variabili chiamate *wl_home* e *wl_away* che assumono valore "W" o "L" a seconda della squadra che ha vinto, se quella in casa o se quella fuori casa. Dato che si avrà per forza che una delle due assumerà valore "W" e l'altra "L", abbiamo deciso di rimuovere una delle due. In particolare, è stato creato un nuovo campo chiamato *winner* che assume valori "home" o "away" a seconda di chi ha vinto la partita. Fatto ciò, i precedenti campi sono stati rimossi

2.4 Embedding di *Players* in *Teams*

L'obiettivo è creare un campo array chiamato *Players* per ogni squadra, contenente gli ObjectID dei giocatori appartenenti ad essa. Per fare ciò:

- dal database si ottengono le abbreviazioni, costituite da tre lettere, dei nomi di tutte le squadre, in quanto questo è il formato in cui questa in-

formazione è disponibile nella collezione dei giocatori. Tali abbreviazioni vengono aggiunte ad una lista;

- iterando sulla lista di abbreviazioni, a turno, si cercano tutti i giocatori appartenenti alla squadra con tale abbreviaizione e si aggiunge il loro ObjectID ad una lista. Dopodiché, tramite una query di update, viene creato il campo *Players* a partire da tale lista.

2.5 Embedding di *Teams* in *Games*

L'obiettivo è creare due campi chiamati *team_id_home* e *team_id_away*, contenenti gli ObjectID del squadre che hanno partecipato a ciascuna partita. Per fare ciò:

- dal database si ottengono le abbreviazioni e gli ObjectID di ogni squadra in modo da avere la chiave da cercare ed il valore da inserire;
- per ognuna di queste coppie "abbreviazione-ID", tramite una *update_many()* si vanno ad aggiornare tutti i documenti della collezione *Games* in cui l'abbreviazione della squadra in casa corrisponde a quella specificata, aggiungendo il campo *team_id_home* con valore corrispondente all'ID di quella squadra. Contemporaneamente, si fa la stessa cosa anche sulle squadre fuori casa.

3 Funzionalità da implementare

L'obiettivo del sito è quello di permettere all'utente di interfacciarsi con il database. Tale interfaccia introdurrà la possibilità di:

- ottenere la classifica del campionato per le due conferences separatamente;
- visualizzare tutte le informazioni relative ad una squadra e a tutti i giocatori ad essa appartenenti;
- elencare tutti i giocatori del database;
- ottenere una lista dei top giocatori di varie categorie (es. migliori tiratori da tre, migliori difensori ecc.);

- interagire con la collection dei giocatori per inserire nuovi giocatori, modificare le informazioni di quelli esistenti, oppure rimuoverne alcuni, rimuovendo anche il loro riferimento nella collection Teams.

4 Implementazione query

Di seguito verranno descritte in dettaglio le query che il sito permette di eseguire per restituire i risultati desiderati.

4.1 Classifica conferences

Per quanto riguarda questo progetto, questa è senza dubbio la query più elaborata che è stata utilizzata. L'obiettivo è quello di calcolare il numero di partite vinte da ogni squadra facendo una sorta di aggregazione condizionale. L'idea è la seguente:

- per ogni documento di questa collection (Games), le variabili da usare per l'aggregazione sono l'ID ed il nome della squadra che ha vinto la data partita. Per questo motivo, saranno usati l'ID ed il nome della squadra che ha giocato in casa SE il valore del campo *winner* per quella partita è uguale a "home", mentre si useranno l'ID ed il nome della squadra fuori casa in caso contrario;
- come valore derivante dall'aggregazione si usa semplicemente *$sum:1*, cioè si contano i record ottenuti che rispettano le precedenti condizioni;
- per il calcolo del numero di partite perse semplicemente si sottrae da 82, il numero di partite che ogni squadra gioca in una stagione, il numero di partite vinte.
- infine, i risultati sono ordinati in ordine decrescente di partite vinte;
- la suddivisione in conference è ottenuta mostrando solo le squadre di quella desiderata.

4.2 Ricerca squadra

La ricerca squadra permette di visualizzare tutte le informazioni sulla squadra cercata e sui suoi giocatori. Questo è implementato tramite una semplice *find*.

Tuttavia, una volta trovata la squadra, invece che visualizzare la lista di riferimenti ai giocatori, ogni riferimento è usato per una ricerca nella collection Players in modo da visualizzare le loro informazioni.

4.3 Lista giocatori

Questa query è una semplice *find* sulla collection dei giocatori che visualizza tutti i risultati ottenuti.

4.4 Top players

Le query per i top players restituiscono, attraverso l'istruzione *limit* applicata sui risultati ordinati in senso decrescente, i 5 giocatori che hanno il valore più alto per alcune delle statistiche scelte.

4.5 Insert, delete e update giocatori

- per quanto riguarda l'inserimento, il documento contenente le informazioni relative ad un nuovo giocatore vengono inserite nella collection *Players* attraverso una semplice *insert_ one()*. Fatto ciò, attraverso il nome della squadra in cui questo gioca, si va a cercare nella collection *Teams* il documento ad essa corrispondente e l'ID del giocatore viene inserito nel campo (lista) *Players*;

- la modifica di un giocatore esistente è implementata semplicemente attraverso una *update_ one()* che cerca il documento del giocatore attraverso il suo nome e la squadra in cui gioca e aggiunge ad esso tutti i nuovi campi specificati dall'utente;

- infine, la cancellazione funziona allo stesso modo dell'inserimento. Prima si cerca il documento del giocatore di interesse attraverso il suo nome, in modo da rimuoverlo, e poi, attraverso il nome della sua squadra, lo si rimuove anche dalla lista *Players* del documento della squadra.

5 Interfaccia web

Per quanto riguarda l'interfaccia web abbiamo deciso di affidarci a Python come linguaggio di programmazione utilizzato, utilizzando il framework Flask.

5.1 Funzionalità di Flask

- Routing: la caratteristica principale che è stata messa a disposizione da Flask è la possibilità di reindirizzare attraverso funzioni le varie pagine web all'esecuzione di determinate circonstanze;
- Gestione delle Richieste e Risposte: facilita la gestione delle richieste HTTP e la formulazione delle risposte;
- Template Rendering: supporta la generazione dinamica di HTML usando Jinja2, un potente motore di template.

All'interno del progetto il file che svolge questo ruolo principale è *__init__.py*, in cui è possibile trovare il cuore pulsante del progetto. In primo luogo troviamo la connessione al nostro DB hostato su Mongo all'URL *mongodb://localhost:27017/* dal quale andiamo a recuperare il DB e le collections di cui abbiamo bisogno, ovvero *Players*, *Games* e *Teams*.

5.2 Funzioni

Successivamente vengono dichiarate delle funzioni per la gestione del routing dell'interfaccia web.

- *def index()*: questa funzione permette il reindirizzamento alla HomePage dell'interfaccia ed è possibile accedervi tramite il Logo, il nome del sito ed il pulsante Home presenti nella navbar;
- *def topPlaer()*: questa funzione utilizza quattro query descritte precedentemente, (*top_scorer*, *top_assists*, *top_3_pointers*, *top_rebounds*) le quali restituiscono una lista di cinque elementi successivamente inseriti in una tabella creata dinamicamente e passata tramite argomento al comando *render_template* che permette la visualizzazione delle pagine html;

- *def listPlayer()*: questa funzione recupera la collection *Players* e la passa come argomento a *render_ template* che crea una tabella nella pagina html contenente tutti i giocatori passati con i le caratteristiche da noi scelte;

- def classificaEst(): questa funzione recuera la lista di squadre appartenenti alla Eastern Conference tramite una find filtrata e, successivamente, tramite le query *vittorie_ tot()*, *vittorie_ casa()* e *vittorie_ trans* vengono recuperate le informazioni non presenti nella collection Teams. Fatto ciò vengono creati dei dizionari dove inserire tutti i documenti raccolti, riuniti in una lista, ordinati per vittorie e inseriti in una tabella dinamica poi passata come argomento al *render_ template*;

- *def classificaOvest()*: questa funzione è identica alla precedente, se non per la condizione imposta sulla find;

- *def cercaSquadra()*: questa funzione indirizza alla pagina nella quale è possibile selezionare quale delle 30 squadre visualizzare tramite un menu a tendina. Il pulsante invierà il risultato della scelta alla funzione cercaSquadra2();

- *def cercaSquadra2()*: questa funzione prende in risposta tramite la request il valore del menu a tendina e visualizza le informazione della Squadra scelta, inoltre tramite il riferimento ai giocatori appartenenti a quella Squadra sono visualizzate anche le informazioni sul Roster.

- def querymix(): questa funzione permette l'indirizzamento alla pagina principale delle query, dalle quali è possibile selezionare tramite il menu a tendina quale query utilizzare. La risposta tramite uno script in javascript selezionerà quale form visualizare a schermo, in base alla query selezionata verranno visualizzati campi diversi e, con la pressione del submit finale, i dati inseriti verranno mandati alle query di modifica, inserimento o cancellazione ed agiranno direttamente sul DB.