MARVEL GRAPH DATABASE

FABRIZIO COMINETTI, DAVIDE ABETE, RUBEN AGAZZI

INDICE

ABSTRACT

Il mondo Marvel è composto da numerosi fumetti, film e serie tv, in crescita numerica costante. Quest'ultimo è inoltre caratterizzato dalla presenza di un quantitativo enorme di personaggi. Il progetto realizzato si pone dunque l'obiettivo di creare una base di dati contenente le relazioni fra i vari prodotti Marvel. All'interno del database sono presenti relazioni come ad esempio fra personaggi e fumetti, personaggi e personaggi, e numerose altre. Per la realizzazione di questo progetto è stato scelto un database non relazionale di tipologia a grafo, costruito tramite l'utilizzo di Neo4j.

1 INTRODUZIONE

Il progetto ha avuto origine dall'idea di realizzare un ampio database relativo al mondo Marvel, una base di dati che permettesse di navigare al suo interno tra i vari prodotti Marvel, fra cui personaggi, film, serie tv e fumetti. L'obiettivo del progetto è di ottenere una sorta di "rete sociale" di super-eroi o personaggi Marvel, i quali sono collegati ai rispettivi fumetti, film, serie tv in cui sono presenti, oppure ancora presentano i rispettivi collegamenti interpersonali fra personaggi ed altri personaggi. Per questi motivi la scelta in fase di costruzione del progetto è ricaduta su un modello a grafo. I dati sono stati ottenuti mediante l'utilizzo di API e web scraping, per arrivare alla costruzione del database tramite l'utilizzo della piattaforma open-source Neo4j. I database a grafo sono organizzati in un insieme di nodi e di relazioni tra i nodi (archi), i quali possono avere proprietà. I nodi possono rappresentare entità, le proprietà sono dati relativi ai nodi. Tra le proprietà principali dei modelli a grafo troviamo la caratteristica 'schema free', infatti in un modello a grafo non esiste uno schema fisso e posso modellare nodi e relazioni, con le relative proprietà, senza dover seguire uno schema rigido. Una criticità dei modelli a grafo è che non scalano in modo efficace, perciò hanno una gestione dei dati inferiore rispetto ad altri modelli NoSQL. Le applicazioni di questa tipologia sono numerose, come ad esempio social networks, sistemi di raccomandazione, applicazioni geografiche, logistics networks, financial transaction graphs, master data management, bioinformatics, authorization and access control, e molte altre.

2 FONTI DATI

Per l'ottenimento dei dati sono stati utilizzati due metodi diversi: l'utilizzo di una Web API e Web Scraping. Le API, Application Programming Interface, sono protocolli utilizzati come interfaccia di comunicazione tra componenti software. Esse consistono in insiemi di routine, strutture dati o variabili che permettono al programmatore di richiamare le funzionalità di un'applicazione di terze parti. [2] JSON (JavaScript Object Notation) è il formato con cui tipicamente le Web API rispondono. Questo formato è basato su due strutture: una collezione di coppie nomi/valore (objects) e una lista ordinata di valori (array). Spesso inoltre i file JSON hanno una struttura nidificata. Lo scraping è il processo di estrarre dati da un sistema senza l'utilizzo di Web APIs. Lo scraping indica anche il processo di estrarre dati da pagine web.

2.1 Web API

Per l'ottenimento dei dati tramite Web API è stato utilizzato il servizio fornito dalla Marvel, che mette a disposizione una sua web API gratuita per ottenere numerosi dati, tra cui quelli relativi a personaggi e fumetti. L'ottenimento dei dati è avvenuto tramite esecuzione di un jupyter notebook ed il linguaggio di programmazione python. Di seguito riassumiamo i principali passi di esecuzione del notebook:

1. Ottenimento dei dati dei personaggi in formato JSON tramite chiamata al relativo endpoint della Web API a gruppi di 100 personaggi, in quanto corrisponde al limite imposto dagli sviluppatori del servizio.

2. Salvataggio su file CSV dei dati relativi ai personaggi ottenuti, in modo da eseguire successive manipolazioni facilmente tramite la libreria *pandas*, prima di inserire i dati nel database.

3. Ottenimento dei dati dei fumetti in formato JSON tramite chiamata al relativo endpoint della web API a gruppi di 100 fumetti, in quanto corrisponde al limite imposto dagli sviluppatori del servizio.

4. Salvataggio su file CSV dei dati relativi ai fumetti ottenuti, in modo da eseguire successive manipolazioni facilmente tramite la libreria *pandas*, prima di inserire i dati nel database.

2.2 Web Scraping

Per la parte riguardante il Web Scraping come sorgente dati è stata utilizzata la Marvel Cinematic Universe Wiki. [1] Lo scraping è stato effettuato, anche in questo caso, tramite esecuzione di un jupyter notebook ed il linguaggio di programmazione python. I principali passi eseguiti dal notebook per completare lo scraping sono i seguenti:

1. Ottenimento, dalla pagina relativa a tutti i personaggi della wiki, contenente i nomi dei personaggi con i relativi link alla pagina personale.

2. Per ogni personaggio, aprendo la pagina personale, acquisizione sempre tramite scraping delle informazioni rilevanti, come ad esempio: biografia, lista di film in cui è presente il personaggio, lista di serie tv in cui è presente il personaggio e relazioni con altri personaggi.

3. Per ogni film trovato nelle pagine dei personaggi, selezionare la pagina relativa al film e ottenere tramite scraping le informazioni del film come ad esempio la trama, i registi, scrittori, compositori, incasso, data di uscita e durata del film.

4. Per ogni serie, trovata nelle pagine dei personaggi, aprire la pagina relativa alla serie ed ottenere tramite scraping le informazioni della serie in questione, come ad esempio la trama, i registi, i produttori e i compositori.

5. Salvataggio temporaneo all'interno di file CSV dei film, serie tv, personaggi ed altre informazioni ottenute per essere processati in seguito.

3 DATA EXPLORATION

La fase di data exploration è stata realizzata sempre utilizzando python, tramite l'ausilio di alcune librerie grafiche per la creazione di visualizzazioni.

3.1 Data Cleaning

Le principali problematiche nella qualità dei dati sono state riscontrate nelle biografie dei personaggi e nelle relazioni tra i vari personaggi ottenute tramite web scraping.

3.1.1 Data Cleaning Biografia

Per quanto riguarda la qualità dei dati delle biografie le principali problematiche sono elencate di seguito:

1. Il notebook salva i vari paragrafi delle biografie come una lista di stringhe, quindi prima di tutto la biografia viene trasformata in una stringa, concatenando le varie stringhe presenti nella lista e inserendo un carattere newLine fra un elemento e l'altro.

2. Inoltre, il notebook salvava alcuni paragrafi della biografia più volte, per questo motivo prima di concatenare la stringa viene effettuato un controllo per vedere se la porzione di stringa è già stata aggiunta alla stringa finale

3. Infine, viene fatto un escaping dei caratteri speciali, come ad esempio il carattere "o", in modo da non avere problemi nell'inserimento nel database dei dati.

3.1.2 Data Cleaning Relazioni

Per quanto riguarda la qualità dei dati delle relazioni interpersonali ottenute tramite web scraping, la principale problematica consiste nell'assenza in alcune relazioni del nome del personaggio interessato,

ad esempio ci sono relazioni che hanno come nome personaggio "Mother", "Father" e così via. Per risolvere questo problema il notebook esegue le seguenti operazioni:

1. Per ogni personaggio vengono recuperate le relative relazioni.
2. Per ogni relazione ottenuta viene effettuato un controllo sul nome del personaggio della relazione, se il nome non è presente all'interno della lista di personaggi la relazione viene scartata.

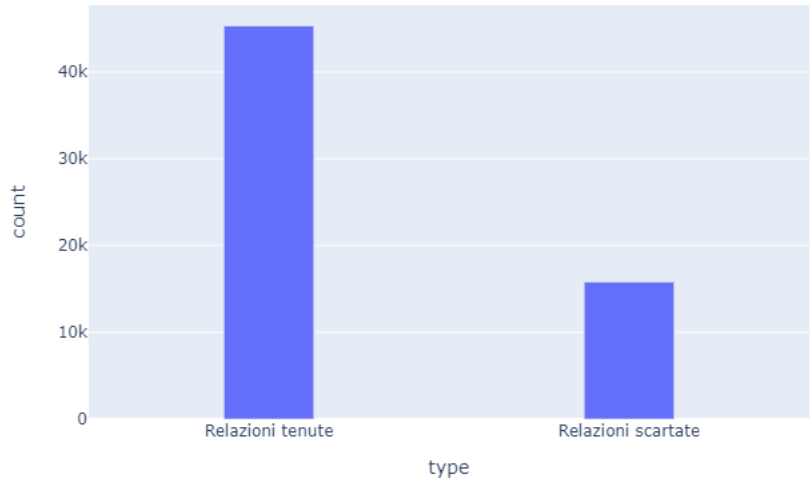

Figura 1: Grafico relativo alle relazioni tenute rispetto a quelle scartate

3.2 Data Quality

Sui personaggi è stata fatta un analisi che consiste nel confronto fra i dati ottenuti dalla web API e quelli ottenuti tramite web scraping. In particolare si può riscontrare una corrispondenza di 323 personaggi della web API con quelli ottenuti tramite web scraping, al netto dei 1559 personaggi della web API.

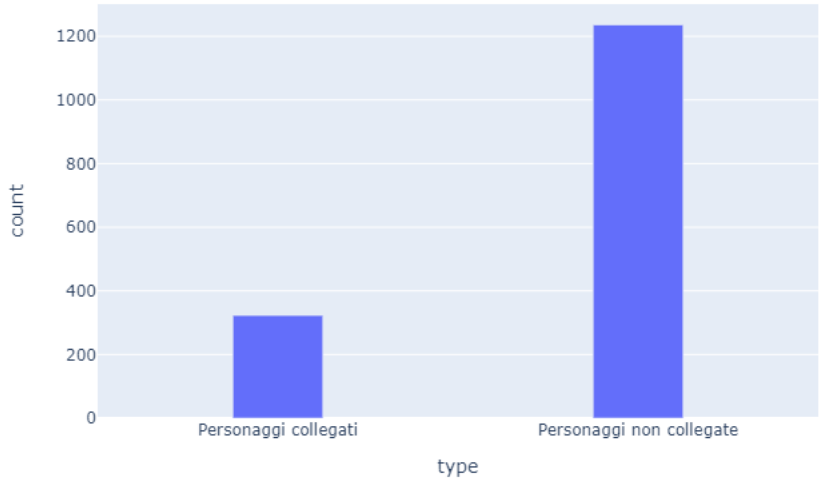

Figura 2: Grafico relativo al numero di personaggi che sono collegabili con i dati ottenuti tramite web scraping.

3.2.1 Completezza

Sono state effettuate altre analisi riguardanti la completezza dei dati, in particolare riguardo alla completezza a livello di tabella dei vari dati ottenuti. Per calcolare la completezza abbiamo utilizzato la seguente formula:

$$\frac{\mathrm{NumeroMissingValuesTabella}}{\mathrm{NumeroColonne} * \mathrm{NumeroRighe}}$$

PERSONAGGI WEB API In seguito all'analisi di completezza riguardante i dati relativi ai personaggi ottenuti tramite consultazione della web API, abbiamo riscontrato che il dataset relativo ha una completezza pari al $92,6\%$

PERSONAGGI WEB SCRAPING In seguito all'analisi di completezza riguardante i dati relativi ai personaggi ottenuti tramite Web Scraping, abbiamo riscontrato che il dataset relativo ha una completezza pari al $95,6\%$

FUMETTI WEB API In seguito all'analisi di completezza riguardante i dati relativi ai fumetti ottenuti tramite Web API, abbiamo riscontrato che il dataset relativo ha una completezza pari al $83,9\%$

SERIE TV WEB SCRAPING In seguito all'analisi di completezza riguardante i dati relativi alle serie TV ottenuti tramite Web API, abbiamo riscontrato che il dataset relativo ha una completezza pari al $96,7\%$

FILM WEB SCRAPING In seguito all'analisi di completezza riguardante i dati relativi ai Film ottenuti tramite Web API, abbiamo riscontrato che il dataset relativo ha una completezza pari al $89,1\%$

3.3 Ridondanza

Un'ulteriore analisi di qualità effettuata consiste nell'analisi di ridondanza dei valori a livello di tabella, in particolare è stato contato, per ogni tabella, il numero di righe con attributo identificativo, come ad esempio nome o titolo, uguali. In particolare abbiamo ottenuto per ogni "tabella":

- Personaggi(Web API): è stato trovato un solo personaggio duplicato.
- Personaggi(Web Scraping): sono stati trovati o personaggi duplicati.
- Fumetti(Web API): sono stati trovati 2216 fumetti duplicati.
- Film(Web Scraping): sono stati trovati o film duplicati.
- Serie TV(Web Scraping): sono state trovate o serie duplicate.

4 DATA INTEGRATION

Per il processo di Data Integration è stato scelto di integrazione fra le due basi di dati: sono state collegate, dove possibile, le due sorgenti dati tramite gli schemi dei personaggi, i quali fungono da punto di incontro fra le due sorgenti dati. È stata realizzata un'operazione di record linkage fra i dati dei personaggi.

4.1 Schema Transformation

Durante la fase di schema transformation è stato effettuato un processo di 'Reverse Engineering' per capire cosa rappresentassero i dati della web API. Tale operazione non è stata fatta per lo scraping, in quanto abbiamo scelto noi quali dati prendere e dunque lo schema era già chiaro a priori. L'analisi dello schema relativo ai personaggi è così composto:

1. Id: identificativo univoco assegnato dall'API ad ogni personaggio.
2. description: breve descrizione del personaggio.
3. modified: ultima modifica apportata al personaggio.

4. thumbnail: URL che porta ad un immagine del personaggio.
5. comics: lista di fumetti in cui il personaggio è presente, con relativo id fumetto.
6. series: lista di serie di fumetti in cui il personaggio è presente.
7. stories: lista di storie in cui il personaggio è presente.
8. events: lista di eventi in cui ha partecipato il personaggio.

L'analisi dello schema relativo ai fumetti ottenuti tramite web API ha permesso di identificare i seguenti attributi:

1. Id: identificativo univoco relativo al fumetto.
2. title: titolo del fumetto.
3. description: breve descrizione del fumetto.
4. modified: ultima modifica dei dati relativi al fumetto.
5. isbn, upc, diamondCode, ean, issn: identificativi editoriali relativi al fumetto.
6. forma: formato di stampa del fumetto.
7. pageCount: numero di pagine del fumetto.
8. series: lista delle serie a cui appartiene il fumetto.
9. thumbnail: URL relativa ad un immagine della copertina del fumetto.
10. images: URL relativi a varie immagini del fumetto.
11. creators: lista dei creatori del fumetto.
12. characters: lista di personaggi appartenenti al fumetto.
13. stories: lista di storie appartenenti al fumetto.
14. events: lista di eventi significativi presenti nel fumetto.

4.2 Corrispondences Investigation

Durante la fase di *Corrispondences Investigation*, si è notato che si possono unire le due basi di dati sulla base dei personaggi, più specificatamente è possibile stabilire un collegamento fra i personaggi sulla base del loro nome. Dopo varie analisi si è notato che, sia nel caso della base di dati ottenuta tramite consultazione della web API sia di quella ottenuta tramite web scraping, un personaggio è identificato tramite un nome privo di parentesi e un'eventuale variante del personaggio è identificata da un nome all'interno della parentesi. Alla luce di queste considerazioni una regola di confronto fra i due schemi consiste nel vedere se due nomi di personaggi sono uguali dopo aver rimosso parentesi ed eventuali testi all'interno delle parentesi.

4.3 Data Preparation

In seguito alle analisi effettuate, primo luogo si aggiunge una nuova colonna al dataset del web scraping e della web API, contenente il nome del personaggio senza alcuni caratteri particolari come ad esempio "\n" e senza qualsiasi porzione di testo racchiusa fra delle parentesi. Queste operazioni sono state effettuate tramite l'utilizzo della libreria python "re", che fa utilizzo delle regular expression per pulire le stringhe. Le parentesi sono tolte per poter fare il confronto dei personaggi senza tenere conto in questo caso della singola variante del personaggio (ad esempio, stesso personaggio di universi differenti). In output si avranno dunque i due schemi relativi ai personaggi con un campo nome processato come detto precedentemente.

4.4 Schemas Integration

La fase di schemas integration avviene durante l'inserimento dei dati nel database: viene creato all'interno del database un nodo di tipo *character* se non è già presente un nodo dello stesso tipo con lo stesso nome. Il nome utilizzato per la creazione è quello creato al passo precedente, ovvero senza caratteri speciali e parentesi. Al nodo character, sia nel caso in cui sia già stato creato o che fosse già presente, viene collegato un nodo *Character Variant* identificato tramite nome completo, ovvero il nome prima della rimozione delle parentesi, contenente tutte le informazioni presenti nel record del personaggio in esame. Ovviamente anche il nodo *Character Variant* viene creato solo qualora non esista già, mentre in caso contrario vengono aggiunte solo le informazioni mancanti presenti nella base di dati da cui proviene il personaggio in esame. In questo modo possiamo tener conto sia del singolo personaggio base sia delle sue varianti. In questo modo quando due nodi *character variant* hanno lo stesso nome, le informazioni

presenti su entrambe le fonti dati saranno aggiunte allo stesso modo, unendo le informazioni.

5 DATABASE

Il database scelto per lo storage dei dati è un database a grafo. Di seguito elenchiamo i principali motivi che hanno portato alla scelta di questo modello:

1. Dati non in real time: la mole di dati riguardante l'universo Marvel non è enorme e non necessita di un inserimento di dati (near) real-time, quindi le operazioni di scrittura sono relativamente poche e le operazioni in lettura sono molte di più; proprio per questo motivo risulta più idoneo l'utilizzo di un database a grafo, in quanto lento in scrittura ma veloce in lettura.
2. Lo scopo del database è di mostrare le varie relazioni fra i prodotti Marvel e per questo motivo un database a grafo rappresenta la soluzione ideale, in quanto è basato sul concetto di relazioni e collegamenti.

5.1 Struttura Database

In questa sezione illustreremo la struttura del database e le sue principali componenti:

5.1.1 Nodi

I nodi presenti all'interno del database sono:

1. Comic: nodo che rappresenta un singolo fumetto Marvel, con al suo interno il suo id fumetto ottenuto dall'API, i vari codici di identificazione, come ad esempio ISBN, la descrizione del fumetto se è presente, il formato del fumetto e il numero di pagine.
2. Movie: nodo rappresentante un film Marvel, con al suo interno i vari dati relativi al film come ad esempio la trama del film, i registi, scrittori, produttori, compositori, incassi, durata del film, e data di rilascio del film.
3. Tv Show: nodo rappresentante una serie tv Marvel, con al suo interno i vari dati relativi al prodotto come ad esempio la trama della prima stagione, i registi, gli showrunner, i produttori e i compositori.

4. Character: nodo rappresentante la versione generica di un personaggio, contenente al suo interno il nome del personaggio.

5. Character Variant: nodo che rappresenta una variante precisa di un personaggio, contenente informazioni come ad esempio il nome, una descrizione del personaggio e una biografia del personaggio.

5.1.2 Relazioni

Le relazioni presenti fra i nodi all'interno del database sono:

1. In Film: relazione che collega un nodo «Character Variant» ad un nodo film; rappresenta appunto la partecipazione del personaggio all'interno del film.

2. In Serie: relazione che collega un nodo «Character Variant» ad un nodo Tv Show; rappresenta la partecipazione del personaggio all'interno di una serie tv.

3. In fumetto: relazione che collega un nodo «Character Variant» ad un nodo «Comic»; rappresenta la partecipazione del personaggio all'interno di un fumetto.

4. Conosce: relazione che collega un nodo «Character Variant» ad un nodo «Character Variant»; rappresenta una relazione interpersonale fra due personaggi.

5. «Variante di»: relazione che collega un nodo di tipo «character» ad un nodo di tipo «character variant»; rappresenta una variante di un personaggio base.

5.2 Statistiche database

All'interno del database sono stati creati in totale 60411 nodi distribuiti nel seguente modo:

Etichetta Nodo	Numero Di Nodi
Character	4609
Character Variant	5113
Comic	50607
Movie	43
Tv Show	19

Fra i nodi del database sono state create 117763 relazioni distribuite come di seguito:

Etichetta Relazione	Numero Di Relazioni
Conosce	19040
In Film	531
In Serie	2405
In Fumetto	90503
Variante di	5284

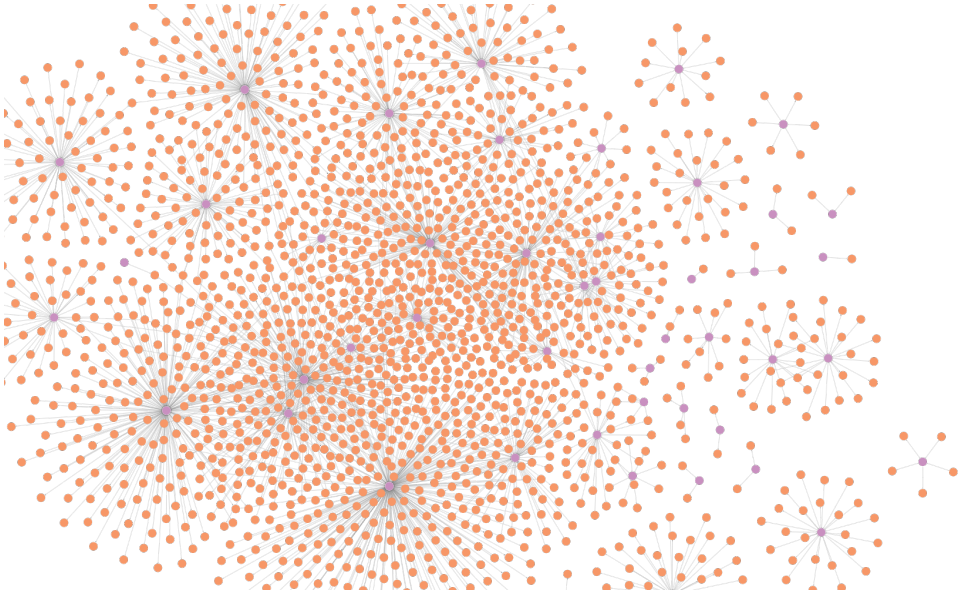

Figura 3: Visualizzazione di alcuni nodi Character variant (Viola) in relazione a dei nodi Comic (Arancione)

5.3 Alcune Query

CQL, ovvero Cypher Query Language è il linguaggio di query per interrogare database a grafo, tra cui proprio Neo4j. Cypher è un linguaggio di tipo dichiarativo e pattern-matching, ed inoltre ha una sintassi simile ad SQL. Grazie alla struttura a grafo del database è possibile ottenere facilmente tramite query relazioni fra i nodi. Ad esempio, potremmo trovare i personaggi conosciuti dal personaggio "Iron Man" e che sono presenti nel film *Avengers*. Per ottenere questo risultato si può utilizzare la seguente query CYPHER:

MATCH (i:character_variant)-[r:conosce]->(c:character_variant)-[p:in_film] ->(m:movie) WHERE i.name = "Iron Man" AND m.title = "The Avengers" RETURN i,r,c,p,m

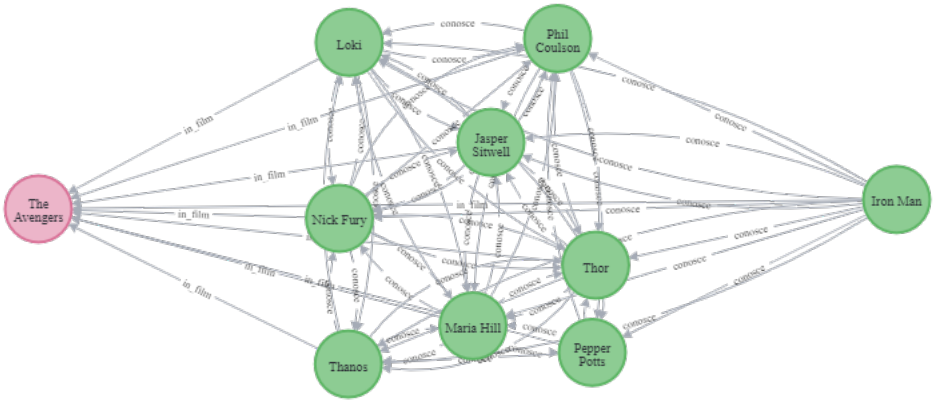

Figura 4: Output - Visualizzazione grafica della query

In questo secondo esempio invece visualizziamo alcuni fumetti in cui è presente il personaggio "Hulk". Per ottenere questo secondo risultato si può utilizzare invece la seguente query CYPHER:

MATCH (c:character_variant)-[r:in_fumetto]->(f:comic) WHERE c.name = "Hulk" RETURN c,f,r LIMIT 100

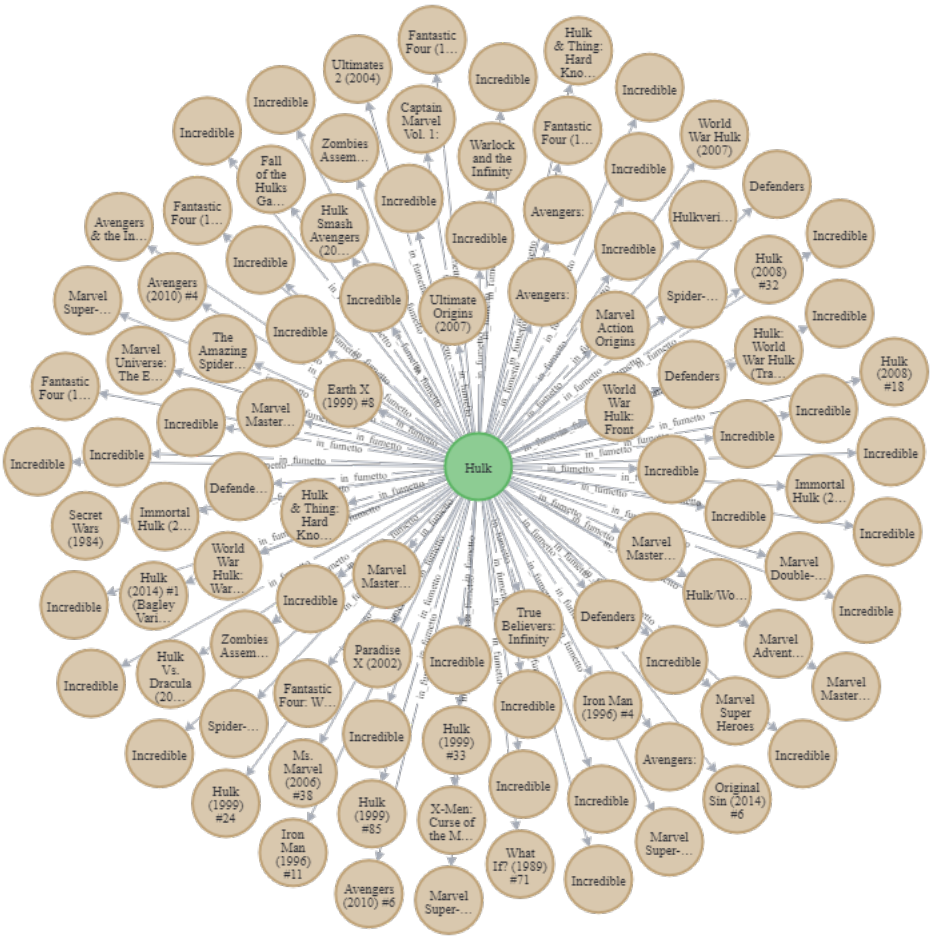

Figura 5: Output - Visualizzazione grafica della query

6 CONCLUSIONE E SVILUPPI FUTURI

In conclusione, abbiamo realizzato un database a grafo partendo dalle due fonti dati, API e web scraping, e successivamente integrato, su cui abbiamo poi svolto una verifica della qualità. Il database finale è completo delle varie relazioni tra personaggi, film e fumetti del mondo Marvel, inoltre ogni nodo contiene diverse informazioni in merito alla sua natura. Durante lo sviluppo del progetto, abbiamo riscontrato alcune limitazioni o problematiche: abbiamo constatato una lentezza in fase di scrittura del database a grafo, ciò però non risulta un problema limitante in caso di inserimento di pochi dati alla volta, in quanto le operazioni di scrittura sono poche rispetto a quelle di lettura. Inoltre, non tantissimi personaggi hanno un match tra le due fonti di dati, una soluzione potrebbe consistere nell'aggiungere una terza fonte di dati per riuscire ad integrare meglio i dati. Il database realizzato potrebbe essere utilizzato per implementare sistemi di raccomandazione, oppure per fornire una piattaforma interattiva per appassionati, che consenta loro di scoprire ed esplorare varie relazioni all'interno di questo mondo. Infine, il database potrebbe essere aggiornato ed ampliato con regolarità, in corrispondenza della produzione di nuovi film, serie tv e fumetti da parte della Marvel Entertainment.

RIFERIMENTI BIBLIOGRAFICI

[1] *Marvel Cinematic Universe Wiki*. URL: https://marvelcinematicuniverse.fandom.com/wiki/Marvel_Cinematic_Universe_Wiki.

[2] Alessandro Rezzani. *Big Data Analytics. Il manuale del data scientist*. Apogeo Education, 2017. ISBN: 8891621854.