Zaawansowane Systemy Baz Danych – Etap 4 „Neo”

Michał Ankiersztajn 311171

Spis treści

[1. Zbiór Danych 1](#_Toc183942222)

[2. Konfiguracja i import 1](#_Toc183942223)

[3. Drugi zbiór danych 3](#_Toc183942224)

[4. Indeksy 4](#_Toc183942225)

[5. Dane przestrzenne 8](#_Toc183942226)

[6. Procedura 11](#_Toc183942227)

[7. Analiza końcowego zboru danych 12](#_Toc183942228)

[8. APOC 13](#_Toc183942229)

[9. Dodatkowe wnioski 15](#_Toc183942230)

[10. Bibliografia 16](#_Toc183942231)

## Zbiór Danych

Wybieram ten sam zbiór danych co w MongoDB, czyli informacje na temat okazjonalnych cen gier wraz z metadanymi tych gier ze strony <https://apidocs.cheapshark.com> dodatkowo powiążę te dane z [Steam Store Games (Clean dataset)](https://www.kaggle.com/datasets/nikdavis/steam-store-games) ponieważ posiada on dużo danych na temat gier razem z steam appid, które posiadam również w danych z cheapshark.

Jest to moja pierwsza styczność z bazą grafową, więc wolę pracować na czymś, co już ‘znam’. Może to nie być najoptymalniejszy wybór pod względem bazy grafowej, jednak będzie on bardzo dobry dla porównania baz JSONowych i grafowych.

## Konfiguracja i import

Szczegóły na temat konfiguracji znajdują się w pliku **docker-compose.yml**

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

**Przetworzenie danych:**

Początkowe dane nie pozwalały na stworzenie 5 krawędzi i 5 relacji, więc je dodatkowo podzieliłem. Skrypty załączone wspomogłem się przy ich pisaniu ChatemGPT: <https://chatgpt.com/share/674acd27-e344-8011-a5ea-efa8412a4de6>

Importowałem zarówno CSV:

Obraz zawierający tekst, Czcionka, linia, zrzut ekranu

Opis wygenerowany automatycznie

Jak i poprzez JSONa:

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Finalna schema (wyeksportowana jako **records.json**), zawarłem w tym punkcie już dodanie kolejnego zbioru danych.

Obraz zawierający tekst, oprogramowanie, diagram, Ikona komputerowa

Opis wygenerowany automatycznie

## Drugi zbiór danych

Drugi zbiór danych jest w postaci CSV i został dodany razem z poprzednim krokiem.

Skorzystam ze zbioru, który został zaproponowany przez Macieja Michalskiego, pasuje on do mojego ponieważ posiadam już dane na temat Steama, jak i gier, a ten dataset ma rozszerzenie danych na temat gier względem tego, co znajduje się w cheapsharku.

**Zapytania:**

1. Proste wyszukiwanie deali po nazwie gry:

Obraz zawierający tekst, linia, numer, Czcionka

Opis wygenerowany automatycznie

1. Wyświetlanie szczegółów danego deala:

Obraz zawierający tekst, zrzut ekranu, linia, Czcionka

Opis wygenerowany automatycznie

1. Wyświetlenie gier, które są poniżej danej ceny i mają wyższą ocenę niż podana (za pomocą UNION):

Obraz zawierający tekst, linia, oprogramowanie, Czcionka

Opis wygenerowany automatycznie

1. Wyświetlenie wszystkich dostępnych szczegółów na temat gry:

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

1. Zapytanie wyświetlające podobne gry (za pomocą MERGE):

Obraz zawierający tekst, zrzut ekranu, linia

Opis wygenerowany automatycznie

Zauważyłem ogromny minus neo4j – interfejs graficzny potrafi mocno spowolnić, gdy nie zamyka się komórek z wynikami zapytań.

## Indeksy

W Neo indeksy są zaimplementowane jako kopie sprecyzowanych danych prymitywnych, takich jak node, relationship, czy properties. Dane przechowywane w indeksie dostarczają ścieżkę do danych w bazie danych i pozwalają na szybsze filtrowanie i przeszukiwanie danych.

Skorzystam z TEXT INDEXu dla game.title, w tym celu zmienię składnię z =~ „.\*nazwa.\*” na CONTAINS „nazwa”, aby mieć pewność, że indeksy te zostaną wykorzystane w zapytaniach.

Dalej stworzę indeksy dla metacriticScore i steamRatingPercent, bo są one wykorzystywane przy wyszukiwaniu po gier po ocenie, a także salePrice i isOnSale, które są kluczowe dla aplikacji typu porównywarka cen.

Stworzę też dla dealID i gameID ponieważ są one wykorzystywane w zapytaniach i nie rzadko w tego typu systemie.

Dodatkowo podstawowo baza Neo4J korzysta z Token Lookup Indexów, które są wykorzystywane do odszukiwania relacji pomiędzy węzłami.

Najpierw sprawdzam czas bez indeksów:

Obraz zawierający tekst, Czcionka, linia, zrzut ekranu

Opis wygenerowany automatycznie

Tworzę indeks tekstowy na tytuł gry:



Zapytania korzystające z indeksów:

Obraz zawierający tekst, Czcionka, numer, linia

Opis wygenerowany automatycznie

Bez zmian, prawdopodobnie w bazie nie ma wystarczająco dużo danych na temat gier (jest ich parę tyś.), aby zauważyć większą zmianę powinno być ich dużo więcej.

Tworzę indeksy na pricing i rating:

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Zapytania korzystające z indeksów:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznie

Ponieważ dane są wielokrotnie wykorzystywane dodatkowo z klauzulą UNION to indeksy mają znaczenie i przyspieszają działanie z 10 ms do 2ms, czyli aż o 500%! Zdecydowanie warto indeksować tego typu dane.

Indeksy na ID:

Obraz zawierający tekst, zrzut ekranu, linia, Czcionka

Opis wygenerowany automatycznie

Zapytania korzystające z indeksów:

Obraz zawierający tekst, numer, Czcionka, linia

Opis wygenerowany automatycznie

Zapytanie z gameID nie przyspieszyło, a wręcz zwolniło – prawdopodobnie korzystając z {} Neo nie wykorzystuje indeksów, ponieważ drugie zapytanie korzystające z dealID przez WHERE przyspieszyło o 200% z 3ms do 1ms.

Nie jestem pewny, czy to tak działa, a więc decyduję się to sprawdzić za pomocą PROFILE:

Obraz zawierający tekst, zrzut ekranu, Czcionka, diagram

Opis wygenerowany automatycznie

Node z game jest znajdywany od razu, czyli indeks powinien działać, sprawdźmy co się stanie, gdy usunę indeks:

Obraz zawierający tekst, zrzut ekranu, Czcionka, diagram

Opis wygenerowany automatycznie

Tym razem przeszukane zostało 1970 wierszy, czyli indeks pomógł i to znacznie, bo ograniczyliśmy liczbę wyszukiwani do 1, różnica czasu musiała wynikać z wolniejszego połączenia z bazą danych lub czasu procesów.

Narzędzie PROFILE jest niesamowicie wygodne i przydatne względem tego co jest w MS SQL, MongoDB, czy PostgreSQL. Prawdopodobnie wynika to z możliwości wglądu w planner query i graficznej reprezentacji.

Dodatkowo możemy optymalizować za pomocą:

1. Klauzul WHERE i LIMIT
2. Tworząc optymalne krawędzie, jeśli często potrzebujemy wziąć coś co jest dalej połączone w grafie, a nie jest bezpośrednią relacją między 2 węzłami to warto zastanowić się nad takim połączeniem, aby zoptymalizować liczbę przeszukań
3. Build-in cache – w Neo istnieje wbudowana pamięć podręczna wykorzystywana przy często wykonywanych zapytaniach.
4. Pluginów, jak np. apoc, którego wykorzystałem do ładowania JSONa, można też do załadowania csv z czego finalnie nie skorzystałem.

## Dane przestrzenne

Dla mojego zestawu danych nie ma danych przestrzennych i bardzo ciężko je wprowadzić, więc korzystam z chat GPT do wygenerowania takich danych:

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Będzie to mały dataset miast. Zacznijmy od sprofilowania najkrótszej ścieżki wyszukiwania miast, które znajdują się w pewnej odległości od pewnego punktu:

Obraz zawierający tekst, zrzut ekranu, linia, Czcionka

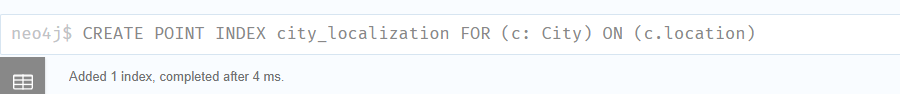
Opis wygenerowany automatycznie

Planner:

Obraz zawierający tekst, elektronika, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Po dodaniu indeksu:



Obraz zawierający tekst, elektronika, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Niestety, w tym przypadku korzyść z indeksu jest zerowa i można powiedzieć, że tylko zaśmieca nam bazę. Sprawdźmy, czy sytuacja jest taka sama w przypadku agregacji:

Sprawdzimy średnią odległość, latitude i longitude w bazie (nie ma to za bardzo sensu funkcjonalnie, ale powinno dobrze przetestować agregację):

Obraz zawierający tekst, Czcionka, linia, zrzut ekranu

Opis wygenerowany automatycznie

Po usunięciu indeksów odpalam zapytanie i patrzę w planer:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

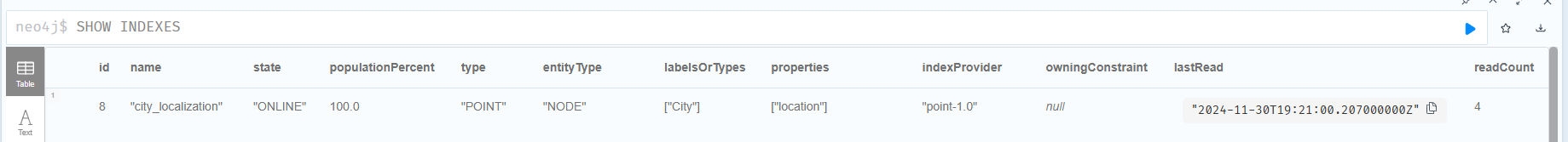
Opis wygenerowany automatycznie

Dodaję indeksy i znowu:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

Sytuacja jest identyczna jest tyle samo uderzeń do bazy, wygląda na to, że przy zapytaniu z agregacją indeks typu point nie jest wykorzystywany. Potwierdza się to przy przejrzeniu indeksów:



readCount nie zwiększa się pomimo uruchamiania zapytania z agregacją. Co oznacza, że dla tego typu zapytań indeksy geolokacji nie są wykorzystywane. Ma to sens, ponieważ i tak w tym wypadku chcemy iterować po wszystkich lokacjach miast.

Zalety takich funkcji to prostota, reużywalność i uwspólnione API dla developerów, zmieniając firmę nie zmieniają się funkcje.

Wadą jest to, że jesteśmy zmuszeni do przechowywania w ten sposób danych, aby móc z nich skorzystać, a są sposoby, aby zrobić to inaczej.

Dane geoprzestrzenne najczęściej wykorzystuje się w firmach w których znaczenie mają mapy (np. Google Maps, Apple Maps), trackowanie lokalizacji (np. Uber, Bolt), jak i rozwiązania IOT (np. włączamy ogrzewanie w domu, tylko gdy właściciel jest w promieniu 10km).

## Procedura

Do stworzenia procedur skorzystam z <https://github.com/neo4j-examples/neo4j-procedure-template/>, tak abym mógł się zapoznać z strukturą i podjąć decyzję, jakiego typu procedurę chciałbym utworzyć.

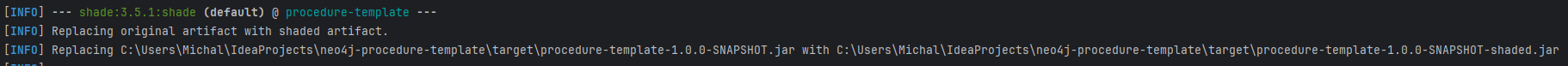
Stwierdziłem, że stworzę procedurę, która pokaże nam wszystkie krawędzie powiązane z danym węzłem, jest to dużo szybsze niż szukanie wszystkich krawędzi w schemach:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Tworzę plik jar za pomocą skryptu:





I wrzucam go do neo4j\_db > plugins, a następnie uruchamiam:

Obraz zawierający tekst, oprogramowanie, Ikona komputerowa, Strona internetowa

Opis wygenerowany automatycznie

Jak widać procedura działa poprawnie i znalazła relacje należące do Deals (zielona kropka).

Procedury w Neo różnią się od relacyjnych tym, że są pisane w zewnętrznym języku programowania. Daje nam to dużo szersze możliwości pisania bardziej skomplikowanych procedur, minus jest taki, że musimy znać używany do procedur język programowania i można korzystać z wielu języków na raz co znacznie podnosi skomplikowanie systemu plus dużo ciężej jest napisać i od razu korzystać z takiej procedury bo jest parę kroków po drodze. Dodatkowo, ponieważ korzystamy z języka programowania możemy po prostu napisać testy i mieć ‘pewność’, że wszystko jest tak jak powinno.

Bardzo łatwo dorzucić plugin z procedurami, ponieważ jest to prosty plik Jar, więc istnieje możliwość stworzenia util bibliotek, które można wykorzystywać w wielu projektach, jedną z takich bibliotek jest APOC. Z tego też względu procedury w Neo dużo częściej są ‘utilsami’, które pomagają w codziennej pracy.

## Analiza końcowego zboru danych

Mój zbiór nie ma oczywistego rozłożenia na wielu maszynach fizycznych. To co przychodzi mi do głowy to rozproszenie bazy danych przez :

* Wydzielenie Steams z SteamRequirements do oddzielnego podgrafu SteamRelated, działało by to na oddzielnej maszynie
* Wydzielenie Deals i Stores do oddzielnego podgrafu Distribution
* Dodatkowo w przypadku bardzo dużego rozrośnięcia się bazy dodałbym oddzielna maszynę dla deali z ratingiem >9, ponieważ to te najlepsze deale są najczęściej wyświetlane i szukane, a cała reszta dużo rzadziej i można by to podzielić na oddzielne maszyny, wtedy jednak dobrze, aby stores były synchronizowane na obydwu maszynach, a nie, aby znajdywały się na oddzielnej maszynie, bo będzie to często odpytywany węzeł przez obydwie maszyny.

**Podsumowując:**

* Mosty przegubowe to (Deals)-[:FOR]->(Games) oraz (Steams)-[:RATING\_FOR]->(Games)
* Węzły przegubowe to Deals, Games oraz Steams.

## APOC

Po zapoznaniu się z bibliotekom APOC skorzystałem z:

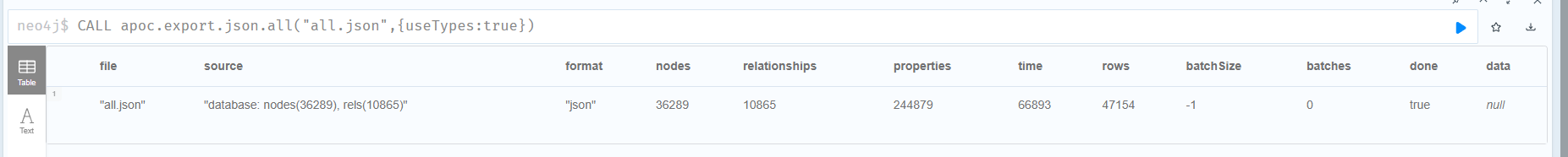
1. Importu danych JSONowych

Bardzo przydatna funkcja zwłaszcza, gdy migrujemy z JSONowej bazy danych jak MongoDB, chcemy skorzystać z danych z publicznego API lub po prostu mamy dane w takim formacie, wykorzystałem to już w rozdziale konfiguracja:

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

1. Skorzystam z eksportu do JSONa, aby można było w prostu sposób odtworzyć i załadować moją bazę nie tylko w neo4j, ale i innych rozwiązaniach baz JSONowych:



Plik załączony **all.json**

1. Formatowanie liczb, nierzadko w zależności od kraju lub aplikacji jesteśmy zmuszeni do reprezentacji liczb w inny sposób:

Obraz zawierający tekst, linia, numer, Czcionka

Opis wygenerowany automatycznie

W moim przypadku dobrze, aby steamRatingCount miał jakiś format, bo może być zarówno bardzo duży, jak i bardzo mały

1. Inwersja relacji – bardzo przydatne, gdy się pomylimy albo w trakcie zauważymy, że któryś kierunek ma większy sens:

Przed:

Obraz zawierający tekst, krąg, zrzut ekranu, design

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Po:

Obraz zawierający krąg, zrzut ekranu, design

Opis wygenerowany automatycznie

1. Teraz dla odwróconej relacji warto zmienić jej nazwę tak, aby była bardziej opisowa:

Obraz zawierający tekst, linia, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Obraz zawierający krąg

Opis wygenerowany automatycznie

## Dodatkowe wnioski

W trakcie tworzenia bazy zrozumiałem na czym polega moc baz grafowych i, że dobrany przeze mnie zbiór danych nie był idealny, ponieważ nie posiadałem przykładowo relacja sam-do-siebie, jak to może być w przypadku ludzi (np. Marian jest dzieckiem Ani, Ania jest dzieckiem … itd.), jak i dwukierunkowości np. w filmie człowiek może być aktorem, jak i producentem i da się to oznaczyć relacjami.

Korzystałem z interfejsu w localhost:7474/browser, problem polegał na tym, że w komórkach nie działa crtl+z, a jest przydatne i przy rozwinięciu komórek z dłuższym wynikiem zapytania przeglądarka się mocno zacinała.

Procedury w neo4j dają dużo więcej możliwości niż w innych bazach i zaskakująco łatwo jest je uruchomić dzięki templatkom i skryptom napisanym przez innych do generowania jarów.

## Bibliografia

<https://medium.com/@matthewghannoum/simple-graph-database-setup-with-neo4j-and-docker-compose-061253593b5a>

<https://community.neo4j.com/t/create-a-relationship-between-two-already-existing-nodes-by-one-common-property/46914>

<https://stackoverflow.com/questions/35281066/neo4j-is-it-possible-to-visualise-a-simple-overview-of-my-database>

<https://stackoverflow.com/questions/37299077/neo4j-importing-local-csv-file>

<https://neo4j.com/labs/apoc/5/installation/#docker>

<https://neo4j.com/docs/cypher-manual/current/indexes/>

<https://neo4j.com/docs/>

<https://neo4j.com/docs/cypher-manual/current/functions/spatial/>