Ondřej Schejbal - schejond

PODOBNOSTNÍ DOTAZOVÁNÍ – THRESHOLD TOP-K OPERÁTOR

Popis projektu

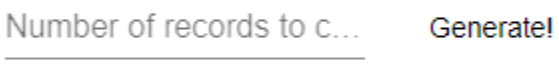
Cílem projektu je zhotovit webovou aplikaci, která bude umožňovat najít v databázi nejlepších ***k*** objektů, za pomocí Threshold algoritmu, v závislosti na uživatelem vybraných atributech daného objektu a vybrané agregační funkci

Způsob řešení

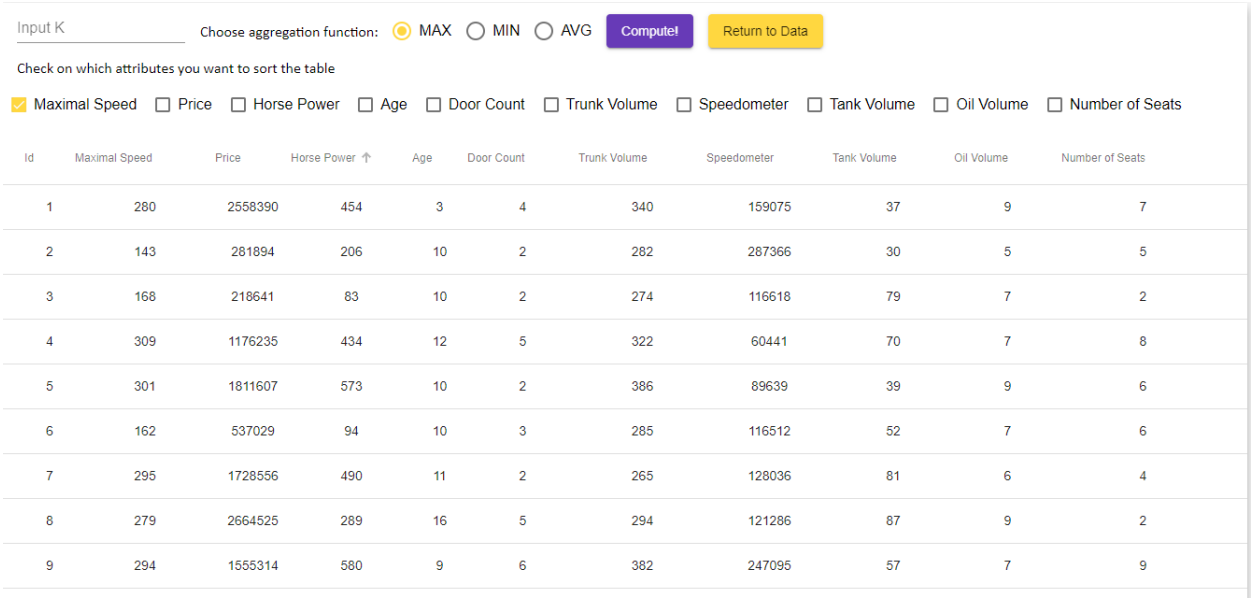
Pro řešení jsem zvolil následující rozhraní pro uživatele.

Vždy se budou operace provádět nad objektem Car(Auto), který má 10 různých atributů (maximální rychlost, cena, koňská síla, věk, obsah nádrže, obsah kufru, obsah olejové nádrže, počet sedaček, počet dveří a tachometr ujetých kilometrů). Všechny objekty i jednotlivé atributy jsou generovány automaticky a pseudonáhodně, aby nagenerované hodnoty byly relevantní.

Nejprve se uživateli zobrazí úvodní obrazovka, ve které může uživatel určit, jaké množství dat daného objektu si přeje vygenerovat, hned vedle input fieldu je tlačítko, kterým se data vygenerují a přenesou uživatele na hlavní obrazovku.



Úvodní obrazovka



Id	Maximal Speed	Price	Horse Power	Age	Door Count	Trunk Volume	Speedometer	Tank Volume	Oil Volume	Number of Seats
1	280	2558390	454	3	4	340	159075	37	9	7
2	143	281894	206	10	2	282	287366	30	5	5
3	168	218641	83	10	2	274	116618	79	7	2
4	309	1176235	434	12	5	322	60441	70	7	8
5	301	1811607	573	10	2	386	89639	39	9	6
6	162	537029	94	10	3	285	116512	52	7	6
7	295	1728556	490	11	2	265	128036	81	6	4
8	279	2664525	289	16	5	294	121286	87	9	2
9	294	1555314	580	9	6	382	247095	57	7	9

Hlavní obrazovka

Velkou část hlavní obrazovky tvoří tabulka s nagenerovanými daty. Tabulka umožňuje, pro lepší práci s daty, data seřadit vzestupně resp. sestupně podle libovolného sloupce.

Ve vrchní části stránky se nachází input field pro zadání hodnoty ***k***, která určí výsledný počet nalezených top objektů. Dále je k dispozici na výběr mezi 3 agregačními funkcemi, které budou použity v rámci Threshold Top-k algoritmu.

Tlačítko ***Compute!*** spustí Threshold Top-k algoritmus a změní zobrazená data v tabulce na nalezené objekty. Zároveň spustí i algoritmus, který projde všechna data sekvenčně a změří dobu průběhu každého z těchto dvou algoritmů. Následně v pravém horním rohu zobrazí informace o čase, jak dlouho každý zmíněný algoritmus probíhal.

Tlačítko ***Return to Data*** slouží k návratu do původní plné tabulky s vygenerovanými daty.

Hned pod touto částí se nachází checkboxy, umožňující určit množinu atributů, které se při hledání top-k objektů mají zohlednit.

Implementace Threshold Top-k algoritmu byla napsána na základě přednášky Doc. RNDr. Tomáše Skopala, Ph.D.

Pro Threshold Top-k algoritmus jsou data hned po vygenerování předzpracována do 10 různých listů, každý seřazený dle normalizační hodnoty pro daný atribut objektu. Hodnota normalizační hodnoty se odvíjí od vygenerovaných dat. Nejnižší vygenerovaná hodnota má normalizační hodnotu 0 a nejvyšší 1.

I pro ukládání mezivýsledků Threshold Top-k algoritmu je opět použit list, který je průběžně řazen, tentokrát na základě hodnoty získané ze zvolené agregační funkce. Pro efektivnější výkon algoritmu, po každé iteraci seřazený top-k list zbavím všech přebytečných objektů, kvůli kterým

velikost listu přesahuje hodnotu zadaného ***k***. Zajistím tím také rychlé řazení tohoto listu při mezivýpočtech.

Implementace

Backend webové aplikace je napsán v Jave a jako framework je použit Spring boot. Na frontend byl použit Angular 7. Jako ORM jsem použil Ebean ORM. Projekt jsem postavil nad PostgreSQL databází.

Diskuze

Z počátku projektu jsem používal databázi na vkládání a ukládání prvků, ale vzhledem k tomu, že jsem data vždy generoval při spuštění programu znovu a přemazával tím data v databázi, tak jsem za účelem zrychlení chodu programu používání databáze nakonec zakomentoval, a data si vytvářím a ponechávám uložené lokálně, nepředpokládám totiž, že by bylo nutné data uchovávat i do dalšího běhu programu.

Během implementace jsem narazil na problém v rámci paralelního průchodu seřazených listů. Problém spočíval v tom, jak správně a efektivně poznat, že prvek, který by se měl do top-k listu objektů přidat na základě normativní hodnoty, nebyl již do tohoto seznamu přidán z jiného sloupce. Jak tento problém řešit, aniž bych si někde musel uchovávat seznam přidaných objektů, nebo procházel celý list top-k objektů. Nakonec jsem se rozhodl pro přidání privátní proměnné typu boolean do objektu Car, kde si zaznamenávám, zda jsem již objekt přidal do top-k listu, nebo ne. Díky referencování objektů, které funguje v javě se mi změna objektu v jednom listu vždy projeví a je dostupná i z ostatních listů, a tím jsem tento problém vyřešil.

V následném testování na různých velikostech dat jsem zjistil, že se výhody algoritmu v porovnání se sekvenčním průchodem projevují až u velkého množství dat, konkrétně když hledám poměrně malé množství ***k*** objektů. Při malém množství dat je rychlost algoritmu srovnatelná s rychlostí sekvenčního průchodu. Velkého rozdílu rychlostí dosáhnu při větším množství dat, hlavně díky předpočítaným a předřazeným seznamům. Například pokud vyberu pouze 1 atribut, podle kterého chci data seřadit, Threshold-top k algoritmus je shopen výsledky zobrazit takřka okamžitě zatímco sekvenční průchod, který data předpočítaná nemá, musí projít všechna data.

Jeden z hlavních nedostatků tohoto projektu jsem objevil během testování. Problém spočíval v tom, že při vygenerování velkého množství dat se poměrně dlouhou dobu čeká, než se data přenesou na frontend. Zobrazování velkého množství dat na jedné stránce, které způsobovalo až zamrznutí prohlížeče, jsem nakonec vyřešil stránkováním tabulky, ale i tak se při přenášení velkého množství dat z backendu na frontend musí chvíli počkat.

Závěr

Myslím si, že semestrální práce proběhla úspěšně a povedlo se mi sestavit webovou aplikaci, na které si uživatel může otestovat fungování algoritmu v plném rozsahu. Hlavně díky možnosti

nechat si vygenerovat libovolný počet dat a možnosti vrátit se po jednom spuštění algoritmu k původním datům, a díky tomu mít možnost opakovaného spuštění algoritmu s jinými parametry nad stejnou sadou dat.