Janusz Erazmus
Wrocław 20.09.2018r.

Zadanie rekrutacyjne dla Mok Yok IT

Adres repozytorium z rozwiązaniem zadania:
https://github.com/HunT37/client_status

Zadanie 1.

W ramach zadania pierwszego wykonałem prosty program w Javie, który wczytuje plik .json, a dokładniej strukturę „records” do obiektu typu JSONArray. Następnie wszystkie rekordy zawierające datę po 1 lipca 2017 wczytałem do ArrayListy obiektów JSONObject w celu skorzystania z metody Collections.sort, której argumenty to lista objektów do posortowania oraz zdefiniowany dla niej komparator. Komparator porównuje najpierw id klientów(klient_id a następnie jeśli id klienta jest takie same, to stosuje sortowanie po czasie kontaktu (kontakt_ts). Na koniec posortowaną listę zapisuję do pliku csv. Kod źródłowy został umieszczony na githubie pod adresem wspomnianym powyżej.

Zadanie 2.

Rozwiązanie: plik zadanie2.sql z repozytorium.

Wyjaśnienie:

Na potrzeby testów rozwiązania utworzyłem lokalnie tabelę client_status w swoim schemacie. Tabela to zaimportowane dane z pliku statuses.json. Tabelę statusów łączę z zapytaniem zwracający klientów, z którymi kontaktowano się więcej niż 3 razy, aby zostawić tylko takie rekordy. Następne złączenie zwraca czas ostatniej próby kontaktu z klientem dla każdego unikalnego klienta. Ostatnie złączenie podaje status ostatniego kontaktu dla wybranej wcześniej daty(timestamp). Wyniki sortuję rosnąco po identyfikatorze klienta.

Przykładowy wynik:

klient_id	liczba_kontaktów	ostatni_kontakt	ostatni_status
2	10	2017-09-14 02:17:14	poczta_glosowa
3	5	2017-09-15 03:38:10	poczta_glosowa
10	5	2017-07-08 12:12:29	zainteresowany
14	5	2017-05-20 06:26:07	zainteresowany
25	5	2017-07-19 04:46:27	zainteresowany
33	14	2017-09-17 02:44:45	zainteresowany
34	31	2017-09-20 09:19:44	poczta_glosowa
35	38	2017-09-17 10:23:08	zainteresowany
36	36	2017-09-18 10:35:12	nie_ma_w_domu
37	20	2017-09-21 02:26:02	nie_ma_w_domu
38	5	2017-08-18 10:41:46	poczta_glosowa

Zadanie 3.

Rozwiązanie: plik zadanie3.sql z repozytorium.

Wyjaśnienie:

Zapytanie stanowiące rozwiązanie tego zadania można podzielić na dwie części. Pierwsza, zawarta w klauzuli FROM zawiera tabelę informację o wszystkich próbach kontaktu z klientem. Jeśli w danym dniu z klientem kontaktowano się wielokrotnie, brana jest pod uwagę tylko ostatnia próba, zgodnie ze specyfikacja zadania. W głównym selekcie dane te są grupowanie po dacie kontaktu i sumowane są ilości sukcesów, utrat oraz informacje o braku kontaktu w danym dniu.
Druga część zadania zawarta jest klauzuli LEFT JOIN (ze względu na to, że danego dnia nie musiała wystąpić zmiana decyzji przez klienta). Najpierw dla każdego klienta wybieram dane na temat ostatniego kontaktu z nim. Następnie łączę to z informacją na temat przedostatniego kontaktu z klientem w sposób taki, że wybieram najpóźniejszą datę kontaktu, która jest wcześniejsza niż najpóźniejsza, czyli poprzedzającą ostatnią. Pomijam tutaj wyniki zawierające brak kontaktu ze strony klienta, ponieważ nie informują one o zmianie przez niego zdania. Na koniec grupuję wynik po dacie, w której ostatnio wystąpiła zmiana zdania przez klienta i sumuję wyniki. Do głównego selekta łączę lewostronnie wyniki.

Przykładowy wynik:

data	sukcesy	utraty	do_ponowienia	zainteresowani_utraty	niezainteresowani_sukcesy
2017-06-10	1	0	2	0	0
2017-06-11	0	3	3	1	0
2017-06-12	5	0	0	0	0
2017-06-13	1	2	2	0	0
2017-06-14	1	1	2	0	1
2017-06-15	2	0	3	0	0
2017-06-16	2	0	1	0	0
2017-06-17	2	1	2	0	0
2017-06-18	3	0	0	0	0
2017-06-19	1	2	1	0	0
2017-06-20	0	0	4	0	0

Zadanie 4.

Ostatniego zadania bonusowego niestety nie udało mi się wykonać, ponieważ brakło mi czasu na zapoznanie się ze środowiskiem TalendStudio.

Uwagi

Pierwsze zadanie udało się wykonać w miarę szybko i gładko, z pomocą informacji w internecie na temat przetwarzania formatu JSON oraz sortowania kolekcji, w tym przypadku ArrayList. Zadania dotyczące sqla pisałem w oparciu o spreparowane przeze mnie dodatkowo dane kontaktów, korzystając z istniejących już ID klientów i ich statusów(losowo wygenerowanych) oraz losowych dat z godzinami, lecz z przedziału tych zawartych oryginalnie w pliku statuses.json. Były mi one potrzebne by dokładnie przetestować działanie zadania trzeciego, które pochłonęło najwięcej czasu. Rozwiązania testowałem na lokalnym serwerze MySQL.