Przetwarzanie masywnych danych

Sprawozdanie

MSD – Schemat i transformacja danych

Maciej Spychała

5 kwietnia 2018

1 Schemat

Schemat początkowy: dwa pliki tekstowe z polami odzielonymi za pomocą ciągu znaków `<SEP>`. Plik z odtworzeniami nie zawiera track_id, przez co w przypadku gdy jeden utwór ma wiele wykonań tracimy informację na temat jakiego wykonania użytkownik słuchał.

```
triplets_sample_20p.txt
#user_id                                        song_id                time
b80344d063b5ccb3212f76538f3d9e43d87dca9e<SEP>SOBBMDR12A8C13253B<SEP>1203083335
b80344d063b5ccb3212f76538f3d9e43d87dca9e<SEP>SOBXALG12A8C13C108<SEP>984663773

unique_tracks.txt
#track_id              song_id                artist               title
TRMMMYQ128F932D901<SEP>SOQMMHC12AB0180CB8<SEP>Faster Pussy cat<SEP>Silent Night
TRMMMKD128F425225D<SEP>SOVFVAK12A8C1350D9<SEP>Karkkiautomaatti<SEP>Tanssi vaan
```

Schemat po trasformacji: jako, że technologią, której użyłem do tego projektu jest `bash` nie czułem potrzeby dużej modyfikacji oryginalnego schematu. Zamieniłem separatory na spacje a spacje w tekstach na znak _ (na początku użyłem jako separatora znaku `tab`, jednakże niektóre programy zwracały wynik jako kolumny oddzielone znakiem spacji, a program `join` wymusza jednakowy separator pomiędzy plikami). Jako, że dane przechowujemy jako plik tekstowy a nie zbiór rekordów w tablicy, mogą mieć one zadaną kolejność, dlatego oba pliki posortowałem według wartości pola `song_id` aby wyeliminować potrzebę dodatkowej pracy przed wywoływaniem komend takich jak `join` oraz `uniq`. Datę, zamiast jako `Unix time` przechowuję w formacie `ROK/MIESIĄC`.

```
#user_id                                  song_id            time
6e4d60a9f2d014958a3de6cf8007ca49050581cb SOAAADD12AB018A9DD 2001/05
6e4d60a9f2d014958a3de6cf8007ca49050581cb SOAAADD12AB018A9DD 2003/08

#track_id          song_id            artist      title
TRBGKMB128F4257851 SOAAABI12A8C13615F Herbie_Mann Afro_Jazziac
TRMTUKT12903CEE7C3 SOAAABT12AC46860F0 Bergen_Big_Band Herre_Gud_Ditt_Dyre
```

2 Proces transformacji

Cały proces transformacji plików z danymi dokonuje poniższy skrypt.

```
sed 's/<SEP>/ /g' ../triplets_sample_20p.txt |\
    awk '{ $3 = strftime("%Y/%m", $3) } { print }' |\
    sort -k2 > play.txt
sed 's/ /_/g' ../unique_tracks.txt | sed 's/<SEP>/ /g' |\
    sort -k2 > tracks.txt
```

3 Czas przetwarzania i rozmiar danych

```
$ time ./get_data.sh
./get_data.sh  120.82s user 48.31s system 130% cpu 2:09.56 total

$ du -h *.txt
1.8G    play.txt
69M     tracks.txt
# all 1.9G
```

4 Zapytania i czas ich wykonania

Ranking popularności piosenek

```
$ cat popular_tracks.sh
cut -d' ' -f2 play.txt | uniq -c |\
    join -j2 - tracks.txt -o 1.1,1.2,2.3,2.4 | sort -nr

$ time ./popular_tracks.sh | head -n 5
145267 SOBONKR12A58A7A7E0 Dwight_Yoakam You're_The_One
129778 SOAUWYT12A81C206F1 Bjork Undo
105162 SOSXLTC12AF72A7F54 Kings_Of_Leon Revelry
84981 SOFRQTD12A81C233C0 Harmonia Sehr_kosmisch
77632 SOEGIYH12A6D4FC0E3 Barry_Tuckwell/Academy_of_St_Martin...

./popular_tracks.sh  7.74s user 1.18s system 164% cpu 5.418 total
```

Ranking użytkowników ze względu na największą liczbę odsłuchanych unikalnych utworów

```
$ cat users_unique_tracks.sh
cut -d' ' -f -2 play.txt | sort -u | cut -d' ' -f1 | uniq -c | sort -nr

$ time ./users_unique_tracks.sh | head -n 5
1040 ec6dfcf19485cb011e0b22637075037aae34cf26
641 119b7c88d58d0c6eb051365c103da5caf817bea6
637 b7c24f770be6b802805ac0e2106624a517643c17
592 4e73d9e058d2b1f2dba9c1fe4a8f416f9f58364f
```

```
586 d7d2d888ae04d16e994d6964214a1de81392ee04

./users_unique_tracks.sh  36.67s user 6.19s system 128% cpu 33.423 total
```

Artysta z największą liczbą odsłuchań

```
$ cat best_artist.sh
cut -d' ' -f2 play.txt | uniq -c | join -j2 -o 1.1,2.3 - tracks.txt |\
    awk '{ a[$2] += $1 } END { for (i in a) print a[i], i }' | sort -nr

$ time ./best_artist.sh | head -n 5
230846 Kings_Of_Leon
224939 Coldplay
156776 Florence_+_The_Machine
146580 Dwight_Yoakam
143865 Bjork

./best_artist.sh  7.96s user 1.17s system 168% cpu 5.422 total
```

Sumaryczna liczba odsłuchań w podziale nna poszczególne miesiące

```
$ cat monthly.sh
cut -d' ' -f3 play.txt |\
    awk '{ a[$1] += 1 } END { for (i in a) print a[i], i }' | sort -nr

$ time ./monthly.sh | head -n 5
236824 2004/05
236474 2003/12
236267 2007/05
236141 2008/10
236139 2009/07

./monthly.sh  12.77s user 0.78s system 166% cpu 8.143 total
```

Wszyscy użytkownicy, którzy odsłuchali wszystkie trzy najbardziej popularne piosenki zespołu Queen

```
$ cat users_who_listen_top.sh
if [ "$#" -ne 1 ]; then
    echo "usage: $0 ARTIST_NAME"
fi

grep -Ff <(grep " $1 " tracks.txt | cut -d' ' -f2 | uniq) play.txt > /tmp/artist_plays
tracks_id=$(cut -d' ' -f2 /tmp/artist_plays | uniq -c |\
    sort -nr | head -n 3 | awk '{print $2}')
grep -Ff <(printf "%s\n" "${tracks_id[@]}") /tmp/artist_plays |\
    cut -d' ' -f -2 | sort -u | cut -d' ' -f1 | uniq -c |\
    grep ' 3 ' | awk '{print $2}'

$ time ./users_who_listen_top.sh Queen | head -n 5
00832bf55ed890afeb2b163024fbcfaf58803098
01cb7e60ba11f9b96e9899dd8da74c277145066e
```

```
0ac20218b5168c10b8075f1f8d4aff2746a2da39
1084d826f98b307256723cc5e9a3590600b87399
11abd6aaa54a50ed5575e8af9a485db752b97b42

./users_who_listen_top.sh Queen  1.45s user 0.30s system 101% cpu 1.727 total
```

5 Porównanie oryginalnego i nowego schematu

Dzięki zamiany znaku separacji pól mogłem z łatwością korzystać z UNIXowych programów. Posortowanie plików względem `sort_id` pozwoliło znacznie przyspieszyć większość zapytań. Zamiana `UNIX time` na format `ROK/MIESIĄC` pozwoliło na szybsze grupowanie po miesiącach, jednakże straciliśmy informację o dokładnym czasie odtworzenia piosenki przez użytkownika.

6 Podsumowanie

Wybrane przeze mnie rozwiązanie posiada kilka zalet:

- taką analizę możemy dokonać na każdym UNIXowym systemie, bez żadnych specjalistycznych programów
- łatwość dzielenia się skryptami między ludźmi
- możliwość korzystania z systemu kotrolii wersji
- szybkość działania
- brak potrzeby importu danych do wybranego przez nas programu

Jednakże nie jest to rozwiązanie idealne. Korzystanie z SZBD na pewno pozwoliłoby na łatwiejsze formułowanie trudniejszych zapytań oraz szybszy development samych zapytań. Korzystanie z bashowych skryptów jest bardzo wrażliwe na zmianę danych wejściowych. Dodanie jednej kolumny do któregokolwiek z plików skutkowałoby potrzebą zmiany prawdopodobnie wszystkich zapytań, w przypadku BD zapewne żadne zapytanie nie wymagałoby zmiany.