PPPD - Lab. 05

Zadanie punktowane, lab 05, grupa A, 2022/2023, autor: Piotr Wolszakiewicz

Uwaga: w rozwiązaniu zadania nie można używać list.

Temat: Baza danych ze zmienną liczbą plików

Zadanie inspirowane algorytmem Consistent hashing.

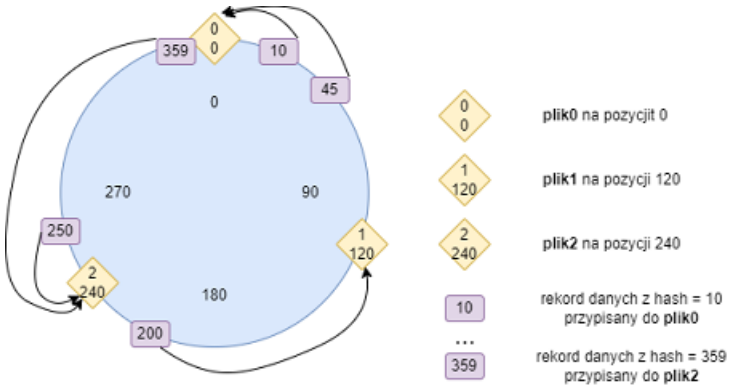
Treść zadania

Zadanie polega na zaimplementowaniu systemu do przechowywania danych w wielu plikach. Liczba plików w trakcie działania programu może się zmieniać, co nie powinno zmieniać ilości już zapisanych danych. Podczas usuwania/dodawania pliku, będziemy potrzebowali odpowiednio poprzenosić dane. Przypisanie rekordu danych do pliku odbywa się w następujący sposób:

- Wszystkie dostępne pliki rozmieszczamy na kole 0° - 360°. W naszym przypadku będziemy operowali maksymalnie trzema plikami (oznaczanymi dalej jako plik0, plik1, plik2). Pliki rozmieszczamy w sposób statyczny, mianowicie plik0 umieszczamy w 0°, plik1 - 120°, plik2 - 240°.
- Następnie wyliczamy wartość `hash` rekordu danych (jako reszta z dzielenia identyfikatora wiersza przez 360).
- Potem znajdujemy plik, który poprzedza na kole wyliczony hash (tzn. jest położony najbliżej wyliczonego `hash-a` patrząc w kierunku przeciwnym do ruchu wskazówek zegara) i do niego dopisujemy nasz rekord danych w nowej linii (na końcu pliku).

Przypisanie wiersza do pliku

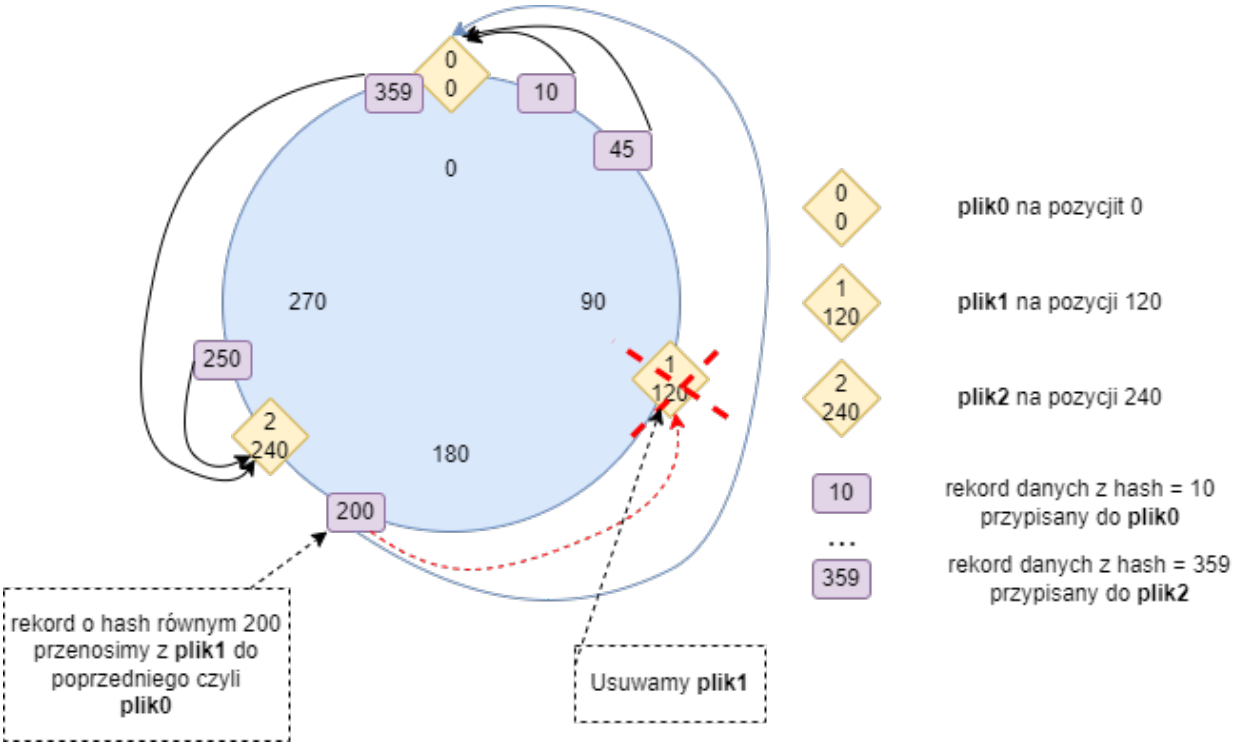
W poniższym przykładzie mamy przypadek z trzema plikami. Mamy również dane, dla których wyliczyliśmy wartości hash jako: 10, 45, 200, 250, 359. Wiersze:
10, 45 - lądują w `file0`
200 - ląduje w `file1`
250, 359 - lądują w `file2`



Rysunek 1: Consistent hashing illustration

Operacja usunięcia pliku

Podczas usuwania pliku przenosimy zapisane w nim rekordy danych do pierwszego istniejącego pliku poprzedzającego go na kole.
W przykładzie usuwamy `file1` i jego rekord danych z hash wyliczonym jako `200` dopisujemy na koniec pliku poprzedzającego - `file0`.



Rysunek 2: Consistent hashing illustration

Wymagania

Twoim zadaniem jest napisanie programu, który działa na zasadzie stałej interakcji z użytkownikiem. Mianowicie oczekuje na wprowadzenie akcji, którą użytkownik chce wykonać. Potem podejmuje działanie odpowiadające wybranej akcji, wyświetla wynik i czeka na kolejną akcję. Program kończy działanie po wybraniu akcji odpowiadającej za wyjście. Akcje mogą być wykonywane w dowolnej kolejności. Program powinien być odporny na sytuacje, gdzie akcja wymaga, aby przed nią była wykonana inna akcja. Przy starcie programu i każdorazowo po wybraniu nieprawidłowej akcji program powinien wyświetlić listę dostępnych akcji.
W funkcji main powinny być przechowywane zmienne: `file0_exists`, `file1_exists` i `file2_exists` jako wartości `bool`, które przechowują informację czy dany `plik` istnieje. W niej również powinna znajdować się obsługa wczytania akcji, wypisania możliwych akcji (np. poprzez wywołanie funkcji), a po każdej akcji 2 i 3 wydrukowanie stanu plików (patrz funkcja `print_files_state` i `get_file_name`)

Możliwe akcje to:

1. Wygeneruj wiersz danych
   W ramach tej akcji napisz funkcję `generate_row()`, która zwraca krotkę w postaci `row_id`, `login`, `level`, gdzie:

- `row_id` - losowa wartość z przedziału 0 - 100000
- `login` - losowo wygenerowany ciąg znaków `a-z` (liter od `a` do `z` jest 26) o długości 5. Litery obok siebie nie mogą się powtarzać, czyli `aabcd` - jest nieprawidłowe, ale `abaca` - jest prawidłowym loginem.

Patrz pomocnicze funkcje `join_letter` do sklejania liter i `number_to_letter` do zamiany liczb 0-25 na odpowiadającą im literę.

- `level` - jest losowo wygenerowaną wartością z `[beginner, regular, senior, expert]`, przy czym beginner - wypada z prawdopodobieństwem 50%, regular - 30%, senior - 15%, expert - 5%.

2. Zapisz wygenerowany uprzednio wiersz do odpowiedniego `pliku`.
   Zapis do pliku `file0`, oznacza zapis do pliku `file0.txt`. W ramach tej akcji są do napisania dwie funkcje:

- `get_file_for_row(row_id, file0_exists, file1_exists, file2_exists)`, gdzie: `row_id` - to identyfikator wiersza. Na podstawie hash-a wyliczonego z `row_id`, funkcja zwraca id `pliku` do którego przynależy dany `row_id` (0, 1 bądź 2). Hash wyliczamy jako resztę z dzielenia row_id z 360.
- `save_row_in_file(file_name, row_id, login, level)`, gdzie `file_name` to nazwa pliku (np. `file0.txt`), a reszta to informacje uzyskane w akcji nr 1. Jeden wiersz danych (row_id, login i level) zapisujemy w oddzielnej linii oddzielone spacją.

3. Usuń `plik` o podanym id (0-2).
   W ramach tej akcji należy:

- wczytać i zwalidować numer pliku do usunięcia (0-2) - `file_id`
- napisać funkcję `can_delete_file` która przyjmuje wszystkie potrzebne parametry do sprawdzenia czy dany `file_id` może być usunięty, a zwraca wartość `bool`. `Pliku` nie można usunąć, jeśli pozostał tylko jeden, bądź `plik` o podanym id już nie istnieje. W takim przypadku odpowiedni komunikat powinien zostać wypisany.
- napisać funkcję `remove_file(file_id, file0_exists, file1_exists, file2_exists)` gdzie: `file_id` - identyfikator `pliku` {0,1,2} Funkcja ta powinna odczytać dane z usuwanego pliku i przepisać je do odpowiedniego istniejącego. Na końcu powinna usunąć plik o podanym `file_id`. Patrz pomocnicza dostarczona funkcja `remove_file_from_disk(file_name)`. Wskazówka: `remove_file` jest wywoływana po walidacji `can_delete_file` - możesz przyjąć, że co najmniej dwa pliki muszą istnieć.

4. Wyjdź z programu

Ustawienia startowe

- Proszę ustawić `seed` dla funkcji `random` wartością 2022
- W funkcji `main` przechowuj informacje czy pliki istnieją `file0_exists`, `file1_exists`, `file2_exists`. Na start wszystkie 3 pliki są dostępne.

Pomocnicze funkcje

```
import os
import os.path

def remove_file_from_disk(file_name):
    if os.path.exists(file_name):
        os.remove(file_name)

def number_to_letter(number):
    return chr(number + ord('a'))

def join_letter(base, letter):
    return base + letter

def get_file_name(file_id):
    """Zwraca nazwę pliku na podstawie identyfikatora pliku"""
```

```
    return f"file{file_id}.txt"

def print_files_state(files_count=3):
    """Wypisuje stan plików"""
    for file_id in range(files_count):
        file_name = get_file_name(file_id)
        if not os.path.exists(file_name):
            continue
        content = ''
        with open(file_name, "r") as read_file:
            print(f'{file_name}: ')
            content = read_file.read()

        if content.strip() !='':
            print(content.strip())
```

```
# Przykłady użycia
initial ='a'
initial = join_letter(initial, 'b')
print(initial)     # ab

print(number_to_letter(0))     # a
print(number_to_letter(25))    # z
```

Punktacja

Komunikacja z użytkownikiem (tzn. wszystkie wczytywania danych i wypisywanie informacji) powinny znajdować się w funkcji `main`. Za poszczególne elementy można uzyskać następującą liczbę punktów:

- Prawidłowo stworzony rdzeń programu: wczytanie akcji, drukowanie dostępnych akcji, zakończenie działania programu, wydrukowanie stanu plików - `2pkt`
- Poprawnie zaimplementowana funkcja `generate_row`, parametry wejściowe i logika losowania - `2pkt`
- Poprawnie zaimplementowana i wywołana funkcja `get_file_for_row` - `2pkt`
- Poprawnie zaimplementowana i wywołana funkcja `save_row_in_file` - `1pkt`
- wczytanie identyfikatora pliku do usunięcia i poprawnie zaimplementowanie funkcji `can_delete_file` - `1pkt`
- Poprawnie zaimplementowana i wywołana w main funkcja `remove_file` - `2pkt`
- Każdy z wymienionych etapów wymaga, aby były stworzone odpowiednie funkcje do tego etapu i aby były one prawidłowo wywołane w `main`

Uwaga

- Jeśli rozwiązanie nie spełnia postawionych wymagań (korzysta z list), zadanie jest oceniane na 0 punktów.
- Jeśli program się nie kompiluje (interpretuje), ocena jest zmniejszana o połowę.
- Jeśli kod programu jest niskiej jakości (nieestetycznie formatowanie, mylące nazwy zmiennych itp.), ocena jest zmniejszana o `2pkt`.

Przykłady interakcji użytkownika z programem

```
Możliwe akcje to:
1 - Wygeneruj wiersz danych
2 - Zapisz wygenerowany uprzednio wiersz do odpowiedniego pliku.
```

```
3 - Usuń plik o podanym id (0-2)
4 - Wyjdź z programu
Podaj nr akcji do wykonania: 1
Wygenerowano wiersz: row_id:69681, login:jorjs, level:beginner
Podaj nr akcji do wykonania: 2
Wiersz dopisany do file1.txt
file1.txt:
69681 jorjs beginner
Podaj nr akcji do wykonania: 1
Wygenerowano wiersz: row_id:90643, login:wnvuk, level:senior
Podaj nr akcji do wykonania: 2
Wiersz dopisany do file2.txt
file1.txt:
69681 jorjs beginner
file2.txt:
90643 wnvuk senior
Podaj nr akcji do wykonania: 1
Wygenerowano wiersz: row_id:99745, login:njuor, level:senior
Podaj nr akcji do wykonania: 2
Wiersz dopisany do file0.txt
file0.txt:
99745 njuor senior
file1.txt:
69681 jorjs beginner
file2.txt:
90643 wnvuk senior
Podaj nr akcji do wykonania: 2
Najpierw trzeba wygenerować wiersz, aby go zapisać
Podaj nr akcji do wykonania: 1
Wygenerowano wiersz: row_id:74311, login:bxiby, level:senior
Podaj nr akcji do wykonania: 2
Wiersz dopisany do file1.txt
file0.txt:
99745 njuor senior
file1.txt:
69681 jorjs beginner
74311 bxiby senior
file2.txt:
90643 wnvuk senior
Podaj nr akcji do wykonania: 3
Podaj numer pliku do usunięcia: 0
Plik 0 został usunięty
file1.txt:
69681 jorjs beginner
74311 bxiby senior
file2.txt:
90643 wnvuk senior
99745 njuor senior
Podaj nr akcji do wykonania: 3
Podaj numer pliku do usunięcia: 0
Nie można usunąć pliku 0
Podaj nr akcji do wykonania: 3
Podaj numer pliku do usunięcia: 1
Plik 1 został usunięty
```

```
file2.txt:
90643 wnvuk senior
99745 njuor senior
69681 jorjs beginner
74311 bxiby senior
Podaj nr akcji do wykonania: 1
Wygenerowano wiersz: row_id:48226, login:jmcni, level:beginner
Podaj nr akcji do wykonania: 2
Wiersz dopisany do file2.txt
file2.txt:
90643 wnvuk senior
99745 njuor senior
69681 jorjs beginner
74311 bxiby senior
48226 jmcni beginner
Podaj nr akcji do wykonania: 4
```