Apache Ignite

Kamil Dółkowski, Wojciech Ignaczak

Maj 2025

Spis treści

1 Wstępne informacje o bazie Apache Ignite

Apache Ignite to rozproszona baza danych dla aplikacji o wysokiej wydajności z szybkością pamięci operacyjnej (RAM). Jest to baza danych **in-memory**. Dane w niej są przechowywane w postaci par **klucz-wartość**.

1.1 Zastosowania Apache Ignite

- Cache do bazy danych (In-Memory Cache)
- Baza danych w pamięci (In-Memory Database)
- Warstwa pośrednia między aplikacją a bazą danych (In-Memory Data Grid)

2 Instalacja i konfiguracja maszyny wirtualnej

2.1 Parametry maszyny

Nazwa maszyny wirtualnej: **ApacheIgnite**
System operacyjny: **ubuntu-24.04.2**
RAM: **8000 MB**
Wątki: **3**
Rozmiar dysku wirtualnego: **20 GB**

2.2 Logowanie

Username: **vboxuser**
Hasło: **changeme**

3 Instalacja i konfiguracja bazy Apache Ignite

Bazę Apache Ignite zainstalowano pomyślnie używając pakietu DEB posługując się instrukcją instalacji zamieszczonej na oficjalnej stronie Apache Ignite.

3.1 Polecenia do instalacji

```
sudo apt update
sudo apt install gnupg ca-certificates --no-install-recommends -y

sudo bash -c 'cat <<EOF > /etc/apt/sources.list.d/ignite.list
deb http://apache.org/dist/ignite/deb/ apache-ignite main
EOF'
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 0EE62FB37A00258D
sudo apt update
sudo apt install apache-ignite --no-install-recommends
```

4 Uruchomienie klastra Apache Ignite

4.1 Jeśli pobrano i zainstalowano przez ZIP Archive

```
./ignite.sh
```

4.2 Jeśli pobrano i zainstalowano przez DEB

```
sudo service apache-ignite@default-config.xml start
sudo systemctl daemon-reload
sudo service apache-ignite@default-config.xml start
```

5 Przykład użycia bazy Apache Ignite za pomocą pythona

5.1 Struktura bazy danych

Przykład stanowi **baza danych przychodni**, w której składowane są dane o **pacjentach, ich wizytach, wydanych receptach, skierowaniach i lekarzach**.

Dane są zapisywane w postaci par **PESEL - PACJENT_INFO**, gdzie kluczem jest numer PESEL pacjenta, a wartością są wszystkie dane o pacjencie zapisane w formacie JSON o strukturze pokazanej poniżej.

```
PESEL -> PACJENT_INFO

PACJENT_INFO = {
    "name": <string>,
    "name2": <string>,
    "surname": <string>,
    "address": <string>,
    "phone": <string>,
    "visits": [
        {"timestamp": <string>, "doctor": <string>, "description": <string>,
        "recommendations": <string>},
        ...
    ],
    "prescriptions": [
        {"date": <string>, "doctor": <string>, "medicines": <string>},
        ...
    ],
    "referrals": [
        {"date": <string>, "doctor": <string>, "test": <string>},
        ...
    ]
```

}

5.2 Przykłady operacji na bazie danych

5.2.1 Łączenie z bazą

Aby zacząć pracować z bazą Apache Ignite należy zaimportować klasę *Client* z biblioteki ***pyignite***. Następnie należy utworzyć instancję ***Client()*** i na niej wykonać funkcję ***connect(ip, port)***, gdzie *ip* to adres ip (u nas loopback - 127.0.0.1) i *port* to numer portu (domyślnie Apache Ignite działa na porcie 10800).

Przykład:

```
from pyignite import Client

client = Client()
client.connect('127.0.0.1', 10800)
```

5.2.2 Tworzenie cache'a

Kolejnym ważnym krokiem jest utworzenie ***cache'a***. Jest to podstawowy sposób przechowywania danych w pamięci RAM klastra. Pełni on podobną rolę co tabele w relacyjnych bazach danych.

Przykład:

```
patients_cache = client.get_or_create_cache('patients')
```

5.2.3 Dodawanie danych

Dodawanie danych odbywa się przez funkcję ***put(key, value)***, gdzie *key* to klucz, a *value* to wartość. Pełni ona podobną rolę co polecenie INSERT z języka SQL.

Przykład:

```
pesel="01234567890"
data = {
        "name": 'Adam',
        "surname": 'Kowalski',
        "visits": [
            {"time": '12.04.2025 8:00', "doctor": 'Jan Nowak'}
        ]
}

json_data = json.dumps(data)
```

```
patients_cache.put(pesel, json_data)
```

5.2.4 Pobieranie danych

Pobieranie danych odbywa się przez funkcję ***get(key)***, gdzie *key* to klucz. Pełni ona podobną rolę co polecenie SELECT z języka SQL.

Przykład:

```
pesel="01234567890"
json_data = patients_cache.get(pesel)
```

5.2.5 Usuwanie danych

Usuwanie danych odbywa się przez funkcję ***remove_key(key)***, gdzie *key* to klucz. Funkcja ta usuwa rekord (parę klucz-wartość) z pamięci podręcznej na podstawie klucza. Pełni ona podobną rolę co polecenie DELETE z języka SQL.

Przykład:

```
pesel="01234567890"
patients_cache.remove_key(pesel)
```

5.2.6 Edytowanie danych

Apache Ignite **nie ma dedykowanej funkcji** do edycji danych. Wynika to z jego charakterystyki. Aby edytować, zaktualizować dane trzeba je **nadpisać** poprzez metodę ***put(key, value)***. Następstwem tego jest **trudność edycji** tylko części danych, jeśli mają złożoną postać, np. JSON. Wymaga to od nas najpierw pobrania całej zawartości danego rekordu, następnie jej edytowanie, a na końcu ponowne dodanie do tego samego klucza. Wymaga to większego nakładu pracy niż wykonanie polecenia UPDATE w relacyjnych bazach danych.

Przykład:

```
pesel = "01234567890"
json_data = patients_cache.get(pesel)

data = json.loads(json_data)
data['surname'] = 'Nowak' # edycja nazwiska

updated_json = json.dumps(data)
patients_cache.put(pesel, updated_json)
```

5.2.7 Wypisywanie całej zawartości cache'a

Do wypisania całej zawartości cache'a można użyć funkcji ***scan()***. Zwraca ona wszystkie pary klucz-wartość.

Przykład:

```
for key, value in patients_cache.scan():
    print(f"PESEL: {key}, Dane: {value}")
```

5.2.8 Usuwanie cache'a

Aby usunąć cache, należy użyć funkcji ***destroy()***. Ma podobne działanie jak DROP TABLE w relacyjnych bazach danych.

Przykład:

```
patients_cache.destroy()
```

5.3 Opis działania programu

Załączony program ***patients.py*** pozwala w sposób dynamiczny zarządzać bazą danych pacjentów.

5.3.1 Funkcja *print_choices()*

Jest to prosta funkcja, która służy do wyświetlania możliwości użytkownika działania na bazie danych.

5.3.2 Funkcja *add_patient(patients_cache)*

Funkcja służąca do dodawania pacjentów do bazy. Użytkownik musi podać: imię, drugie imię (opcjonalne), nazwisko, pesel, adres, numer telefonu.

5.3.3 Funkcja *patient_info(patients_cache)*

Funkcja służąca do wyświetlenia danych o pacjencie, jego dane, wizyty, recepty, skierowania, na podstawie klucza jakim jest pesel.

5.3.4 Funkcja *add_visit(patients_cache)*

Funkcja dodająca wizytę danego pacjenta na podstawie numeru pesel. Trzeba podać datę wizyty(dzień.miesiąc.rok), godzinę(godzina:minuta), lekarza, opis z wizyty, zalecenia.

5.3.5 Funkcja *add_prescription(patients_cache)*

Funkcja dodająca receptę danego pacjenta na podstawie numeru pesel. Trzeba podać datę (dzień.miesiąc.rok), lekarza oraz przepisane leki.

5.3.6 Funkcja *add_referral(patients_cache)*

Funkcja dodająca skierowanie danego pacjenta na podstawie numeru pesel. Trzeba podać datę (dzień.miesiąc.rok), lekarza oraz badanie.

5.3.7 Funkcja *all_patients_info(patients_cache)*

Funkcja służąca do wyświetlenia danych o wszystkich pacjentach, ich dane, wizyty, recepty, skierowania.

5.3.8 Funkcja *destroy_cache(patients_cache)*

Jest to prosta funkcja służąca do usunięcia cache'a *patients_cache*.

5.3.9 Funkcja *delete_patient(patients_cache)*

Funkcja usuwająca pacjenta z bazy za pomocą klucza, czyli numeru pesel.

5.3.10 Funkcja *main()*

Główna funkcja, która:

- nawiązuje połączenie z serwerem Apache Ignite
- tworzy lub pobiera cache *patients_cache*
- dodaje przykładowe dane
- pętla służąca do interakcji z użytkownikiem
  - dodawanie danych
  - przeglądanie
  - usuwanie pacjentów
  - usuwanie całego cache'a
- kończy pracę i zamyka połączenie

6 Przemyślenia końcowe

Apache Ignite to wydajne i elastyczne rozwiązanie do przechowywania danych w pamięci operacyjnej, które może pełnić funkcję cache'a, bazy danych in-memory lub warstwy pośredniej. Integracja z Pythonem poprzez bibliotekę ***pyignite*** umożliwia szybkie tworzenie aplikacji obsługujących dane w strukturze klucz-wartość. Wyzwania mogą pojawić się przy modyfikacji złożonych danych (np. JSON), jednak prostota interfejsu rekompensuje te trudności.