

# Bazy danych NoSQL

laboratorium

**Kamil Staśko**

NIESTACJONARNE, GRUPA: L1

Opracowanie laboratorium (Mongo, MariaDB)



Uniwersytet Pedagogiczny  
im. Komisji Edukacji Narodowej  
w Krakowie

# Wstęp

Laboratorium zostało przeprowadzone z wykorzystaniem:

- Środowisko: VMware® Workstation 14 Player
- Wersja środowiska: 14.1.8 build-14921873
- System operacyjny: Debian GNU/Linux 10 (buster)
- Procesor: Intel(R) Core(TM) i7-6700HQ CPU @2.60GHz 2.60GHz
- Architektura: 64-bit
- Pamięć RAM: 2GB

# Rozdział 1

## Komunikacja MariaDB - Mongo

### 1.1 Kody plików

- [lab1.yml](#) - dla docker-compose - ustawienie środowiska, obrazy (mongo, mariadb, python)

```
1 version: '3'
2 networks:
3     cluster:
4         external:
5             name: cluster
```

Rys. 1.1: Stworzenie klastra

```
1 services:
2     mongo:
3         container_name: "mongo"
4         networks:
5             cluster:
6                 ipv4_address: 10.100.100.2
7                 aliases:
8                     - mongo
9         image: mongo
```

Rys. 1.2: Konfiguracja Mongo

```

1  mariadb:
2      container_name: "mariadb"
3      networks:
4          cluster:
5              ipv4_address: 10.100.100.3
6          aliases:
7              - mariadb
8      image: mariadb
9      ports:
10         - "3306:3306"
11     volumes:
12         - /root/ZADANIE:/DATA
13     restart: always
14     environment:
15         MYSQL_ROOT_PASSWORD: root
16         MYSQL_USER: root
17         MYSQL_PASSWORD: root
18         MYSQL_DATABASE: LAB_KAMIL
19     healthcheck:
20         test: ["CMD", "mysql", "--user=root", "-proot", "-e",
21             ↪ show databases"]
22         interval: 2s
23         timeout: 1s
24         retries: 20

```

Rys. 1.3: Konfiguracja MariaDB

```

1  python:
2      container_name: "python"
3      networks:
4          cluster:
5              ipv4_address: 10.100.100.4
6          aliases:
7              - python
8      image: deb_py
9      volumes:
10         - /root/ZADANIE:/PY3
11     command: tail -f /etc/passwd > /dev/null

```

Rys. 1.4: Konfiguracja python3

- [lab1.sql](#) - plik odtwarzający tabele w bazie maria (10000 rekordów)

```

1  USE LAB_KAMIL;
2
3  CREATE TABLE TABLE_KAMIL(
4      id INT NOT NULL AUTO_INCREMENT,
5      val0 INT NOT NULL,
6      val1 INT NOT NULL,
7      val2 INT NOT NULL,
8      val3 INT NOT NULL,
9      val4 INT NOT NULL,
10     val5 INT NOT NULL,
11     val6 INT NOT NULL,
12     val7 INT NOT NULL,
13     val8 INT NOT NULL,
14     val9 INT NOT NULL,
15     PRIMARY KEY (id)
16 );

```

Rys. 1.5: Stworzenie tabeli

```

1 drop procedure if exists myLoop;
2 DELIMITER //
3 CREATE PROCEDURE myLoop()
4 BEGIN
5 DECLARE i INT DEFAULT 1;
6 DECLARE j INT DEFAULT 1;
7 WHILE (i <= 10000) DO
8     INSERT INTO TABLE_KAMIL
9         (val0, val1, val2, val3, val4, val5, val6, val7, val8,
10          ↪ val9)
11     VALUES
12         (10*j, 10*j+1, 10*j+2, 10*j+3, 10*j+4, 10*j+5, 10*j+6,
13          ↪ 10*j+7, 10*j+8, 10*j+9);
14     SET i = i+1;
15     SET j = j+1;
16 END WHILE;
17 END;
18 //
19 CALL myLoop();

```

Rys. 1.6: Procedura wprowadzenia danych

- `lab1.py` - plik przepisujący tabele z mariadb do kolekcji mongo

```

1 #!/usr/bin/python
2 import time
3 import mysql.connector
4 from pymongo import MongoClient
5
6 conn = mysql.connector.connect(user='root', password='root',
7     ↪ host='10.100.100.3', database='LAB_KAMIL')
8 cur = conn.cursor()
9 cur.execute('SELECT id, val0, val1, val2, val3, val4, val5, val6
10     ↪ , val7, val8, val9 FROM TABLE_KAMIL')

```

Rys. 1.7: Połączenie z MariaDB

```

1 client = MongoClient('10.100.100.2:27017')
2 baza = client['LAB_KAMIL']
3 kolekcja = baza['KOLEKCJA_KAMIL']

```

Rys. 1.8: Połączenie z Mongo

```

1 start_time = time.time()
2
3 for (id, val0, val1, val2, val3, val4, val5, val6, val7, val8,
   ↪ val9) in cur:
4     kolekcja.insert_one({'_id': id, 'val0': val0, 'val1':
   ↪ val1, 'val2': val2, 'val3': val3, 'val4': val4, '
   ↪ val5': val5, 'val6': val6, 'val7': val7, 'val8':
   ↪ val8, 'val9': val9})
5
6 end_time = time.time() - start_time
7
8 print(end_time)

```

Rys. 1.9: Przepisanie danych oraz obliczenie czasu procesu

- `lab1.sh` - skrypt bash automatyzujący cały proces

```

1 #!/bin/sh
2
3 cd /root/
4 mkdir ZADANIE
5 cd ./ZADANIE

```

Rys. 1.10: Stworzenie katalogu z danymi

```

1  echo "version: '3'
2  networks:
3      cluster:
4          external:
5              name: cluster
6  services:
7      mongo:
8          container_name: \"mongo\"
9          networks:
10             cluster:
11                 ipv4_address: 10.100.100.2
12                 aliases:
13                     - mongo
14             image: mongo
15      mariadb:
16          container_name: \"mariadb\"
17          networks:
18             cluster:
19                 ipv4_address: 10.100.100.3
20                 aliases:
21                     - mariadb
22             image: mariadb
23             ports:
24                 - \"3306:3306\"
25             volumes:
26                 - /root/ZADANIE:/DATA
27             restart: always
28             environment:
29                 MYSQL_ROOT_PASSWORD: root
30                 MYSQL_USER: root
31                 MYSQL_PASSWORD: root
32                 MYSQL_DATABASE: LAB_KAMIL
33             healthcheck:
34                 test: [\"CMD\", \"mysql\", \"--user=root\", \"-proot\",
35                     ↪ \"-e show databases\"]
36                 interval: 2s

```



```
36         timeout: 1s
37         retries: 20
38     python:
39         container_name: \"python\"
40     networks:
41         cluster:
42             ipv4_address: 10.100.100.4
43         aliases:
44             - python
45     image: deb_py
46     volumes:
47         - /root/ZADANIE:/PY3
48     command: tail -f /etc/passwd > /dev/null
49 " > lab1.yml
```

Rys. 1.11: Stworzenie pliku lab1.yml

```

1  echo "USE LAB_KAMIL;
2
3  CREATE TABLE TABLE_KAMIL(
4      id INT NOT NULL AUTO_INCREMENT,
5      val0 INT NOT NULL,
6      val1 INT NOT NULL,
7      val2 INT NOT NULL,
8      val3 INT NOT NULL,
9      val4 INT NOT NULL,
10     val5 INT NOT NULL,
11     val6 INT NOT NULL,
12     val7 INT NOT NULL,
13     val8 INT NOT NULL,
14     val9 INT NOT NULL,
15     PRIMARY KEY (id)
16 );
17
18 drop procedure if exists myLoop;
19 DELIMITER //
20 CREATE PROCEDURE myLoop()
21 BEGIN
22 DECLARE i INT DEFAULT 1;
23 DECLARE j INT DEFAULT 1;
24 WHILE (i <= 10000) DO
25     INSERT INTO TABLE_KAMIL
26         (val0, val1, val2, val3, val4, val5, val6, val7, val8,
27          ↪ val9)
28     VALUES
29         (10*j, 10*j+1, 10*j+2, 10*j+3, 10*j+4, 10*j+5, 10*j+6,
30          ↪ 10*j+7, 10*j+8, 10*j+9);
31     SET i = i+1;
32     SET j = j+1;
33 END WHILE;
34 END;
35 //

```

```
35 CALL myLoop();
36 " > lab1.sql
```

Rys. 1.12: Stworzenie pliku lab1.sql

```
1 echo "#!/usr/bin/python
2 import time
3 import mysql.connector
4 from pymongo import MongoClient
5
6 conn = mysql.connector.connect(user='root', password='root',
    ↪ host='10.100.100.3', database='LAB_KAMIL')
7 cur = conn.cursor()
8 cur.execute('SELECT id, val0, val1, val2, val3, val4, val5, val6
    ↪ , val7, val8, val9 FROM TABLE_KAMIL')
9
10 client = MongoClient('10.100.100.2:27017')
11 baza = client['LAB_KAMIL']
12 kolekcja = baza['KOLEKCJA_KAMIL']
13
14 start_time = time.time()
15
16 for (id, val0, val1, val2, val3, val4, val5, val6, val7, val8,
    ↪ val9) in cur:
17     kolekcja.insert_one({'_id': id, 'val0': val0, 'val1':
        ↪ val1, 'val2': val2, 'val3': val3, 'val4': val4, '
        ↪ val5': val5, 'val6': val6, 'val7': val7, 'val8':
        ↪ val8, 'val9': val9})
18
19 end_time = time.time() - start_time
20
21 print(end_time)
22 " > lab1.py
```

Rys. 1.13: Stworzenie pliku lab1.py

```

1  docker network create --subnet=10.100.100.0/24 cluster
2  wait $!
3  docker-compose -f lab1.yml up -d
4  wait $!
5
6  value=$(docker inspect --format='{{.State.Health.Status}}'
    ↪ mariadb)
7  while [ "$value" != "healthy" ]
8  do
9      value=$(docker inspect --format='{{.State.Health.Status}}'
    ↪ mariadb)
10 done
11
12 docker exec -it mariadb bash -c "mysql -u root -proot -e 'source
    ↪ /DATA/lab1.sql'"
13 wait $!
14
15 docker exec -it python bash -c "virtualenv --no-site-packages --
    ↪ python=python3 PY3"
16 wait $!

```

Rys. 1.14: Przygotowanie środowiska

```

1  docker exec -it python bash -c "cd PY3/; source bin/activate;
    ↪ pip install pymongo; pip install mysql-connector; python
    ↪ lab1.py > result.txt"
2  wait $!
3
4  docker-compose -f lab1.yml down
5  wait $!
6  docker network rm cluster
7  wait $!
8
9  result=$(cat result.txt)
10 echo ""
11 echo ""
12 echo "KONIEC PROGRAMU"
13 echo ""
14 echo "CZAS KOPIOWANIA [10000 REKORDÓW] MARIADB->MONGO WYNIÓŚŁ:
    ↪ $result s"
15 echo ""

```

Rys. 1.15: Przygotowanie python3 oraz wykonanie testu

## 1.2 Wykonanie testów

Pomiar czasu pobrania rekordów z MariaDB oraz wpisania ich do kolekcji w Mongo.

Tabela 1.1: Testy komunikacji MariaDB - Mongo

LICZBA REKORDÓW	CZAS [s]
10	0.0170135498046875
100	0.07358860969543457
1000	0.6900181770324707
10000	9.058728456497192
100000	65.4701337814331

# Rozdział 2

## Komunikacja Mongo - Mongo

### 2.1 Kody plików

- [lab2.yml](#) - dla docker-compose - ustawienie środowiska, obrazy (mongo1, mongo2, python)

```
1 version: '3'
2 networks:
3     cluster:
4         external:
5             name: cluster
```

Rys. 2.1: Stworzenie klastra

```
1 services:
2     mongo1:
3         container_name: "mongo1"
4         networks:
5             cluster:
6                 ipv4_address: 10.100.100.2
7                 aliases:
8                     - mongo1
9         image: mongo
```

Rys. 2.2: Konfiguracja Mongo1

```

1  mongo2:
2      container_name: "mongo2"
3      networks:
4          cluster:
5              ipv4_address: 10.100.100.3
6      aliases:
7          - mongo2
8      image: mongo

```

Rys. 2.3: Konfiguracja Mongo2

```

1  python:
2      container_name: "python"
3      networks:
4          cluster:
5              ipv4_address: 10.100.100.4
6      aliases:
7          - python
8      image: deb_py
9      volumes:
10         - /root/ZADANIE:/PY3
11     command: tail -f /etc/passwd > /dev/null

```

Rys. 2.4: Konfiguracja python3

- [lab2create.py](#) - plik przygotowujący kolekcję danych w mongo1

```

1  #!/usr/bin/python
2  from pymongo import MongoClient
3
4  client = MongoClient('10.100.100.2:27017')
5  baza = client['LAB_KAMIL']
6  kolekcja = baza['KOLEKCJA_KAMIL']

```

Rys. 2.5: Połączenie z Mongo1

```

1 i = 1
2 j = 1
3
4 while i <= 1000:
5     kolekcja.insert_one({'_id': i, 'val0': 10*j, 'val1': 10*j
6         ↪ +1, 'val2': 10*j+2, 'val3': 10*j+3, 'val4': 10*j
7         ↪ +4, 'val5': 10*j+5, 'val6': 10*j+6, 'val7': 10*j
            ↪ +7, 'val8': 10*j+8, 'val9': 10*j+9})
6     i += 1
7     j += 1

```

Rys. 2.6: Wypełnienie danymi kolekcji Mongo1

- `lab2test.py` - plik wykonujący test podstawowych operacji (find, read, write)

```

1 #!/usr/bin/python
2 from pymongo import MongoClient
3 import random
4 import time
5
6 client1 = MongoClient('10.100.100.2:27017')
7 baza1 = client1['LAB_KAMIL']
8 kolekcja1 = baza1['KOLEKCJA_KAMIL']

```

Rys. 2.7: Połączenie z Mongo1

```

1 client2 = MongoClient('10.100.100.3:27017')
2 baza2 = client2['LAB_KAMIL']
3 kolekcja2 = baza2['KOLEKCJA_KAMIL']

```

Rys. 2.8: Połączenie z Mongo2



```
1 start_time = time.time()
2
3 while kolekcja1.count_documents({}) != kolekcja2.count_documents
  ↳ ({}):
4     val = 0
5
6     while val == 0:
7         val = random.randint(1, kolekcja1.count_documents
  ↳ ({}))
8
9         if kolekcja2.count_documents({"_id": val}) > 0:
10             val = 0
11
12         kolekcja2.insert_one(kolekcja1.find_one({'_id': val}))
13
14 end_time = time.time() - start_time
15
16 print(end_time)
```

Rys. 2.9: Wykonanie testu

## 2.2 Wykonanie testów

Pomiar czasu wykonania następującego cyklu zapytań (liczba zależna od liczby rekordów w Mongo1):

- Mongo1 - count\_document (określenie liczby rekordów)
- Mongo2 - count\_document (określenie liczby rekordów)
- wykonanie cyklu zapytań (liczba zależna od wylosowania id, które nie znajduje się w Mongo2)
  - Mongo1 - count\_document (w celu wylosowania id)
  - Mongo2 - count\_document (w celu sprawdzenia czy istnieje element o danym id)
- Mongo1 - find\_one (pobranie rekordu o podanym id)
- Mongo2 - insert\_one (wstawienie pobranego rekordu z Mongo1)

Tabela 2.1: Testy wydajności Mongo - Mongo

LICZBA REKORDÓW	CZAS [s]
10	0.08119702339172363
100	0.9195506572723389
1000	18.200843334197998
10000	658.4297559261322

# Listings

1.1	Stworzenie klastra . . . . .	3
1.2	Konfiguracja Mongo . . . . .	3
1.3	Konfiguracja MariaDB . . . . .	4
1.4	Konfiguracja python3 . . . . .	5
1.5	Stworzenie tabeli . . . . .	5
1.6	Procedura wprowadzenia danych . . . . .	6
1.7	Połączenie z MariaDB . . . . .	6
1.8	Połączenie z Mongo . . . . .	7
1.9	Przepisanie danych oraz obliczenie czasu procesu . . . . .	7
1.10	Stworzenie katalogu z danymi . . . . .	7
1.11	Stworzenie pliku lab1.yml . . . . .	8
1.12	Stworzenie pliku lab1.sql . . . . .	10
1.13	Stworzenie pliku lab1.py . . . . .	11
1.14	Przygotowanie środowiska . . . . .	12
1.15	Przygotowanie python3 oraz wykonanie testu . . . . .	13
2.1	Stworzenie klastra . . . . .	14
2.2	Konfiguracja Mongo1 . . . . .	14
2.3	Konfiguracja Mongo2 . . . . .	15
2.4	Konfiguracja python3 . . . . .	15
2.5	Połączenie z Mongo1 . . . . .	15
2.6	Wypełnienie danymi kolekcji Mongo1 . . . . .	16
2.7	Połączenie z Mongo1 . . . . .	16
2.8	Połączenie z Mongo2 . . . . .	16
2.9	Wykonanie testu . . . . .	17

# Spis tabel

1.1	Testy komunikacji MariaDB - Mongo . . . . .	13
2.1	Testy wydajności Mongo - Mongo . . . . .	18