МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

КУРСОВИЙ ПРОЕКТ

з дисципліни “Бази даних”

спеціальність 121 – Програмна інженерія

на тему: Моніторингова система захворюваності на коронавірус у світі

Студент

групи КП-73 Булаєвський І. О _____

(ПІБ) (підпис)

Викладач

к.т.н, доцент кафедри

СПіСКС **Петрашенко А.В.** _____

(підпис)

Захищено з оцінкою _____

Київ – 2020

Анотація

Метою розробки даного курсового проекту є набуття практичних навичок розробки сучасного програмного забезпечення, що взаємодіє з постреляційними базами даних, а також здобуття навичок оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проектної документації. У результаті виконання курсового проекту було опановано навик розробляти програмне забезпечення для постреляційних баз даних, володіння основами використання СУБД, а також інструментальними засобами аналізу великих обсягів даних.

Темою даного курсового проекту є створення ПЗ для моніторингової системи захворюваності на коронавірус у світі. У документі викладена актуальність та проблематика аналізу великого обсягу даних, аналіз використаного інструментарію (опис мови програмування, використаних бібліотек та СКБД), описана структура бази даних, опис розробленого програмного забезпечення (загальний, опис модулів та основних алгоритмів роботи), аналіз функціонування засобів масштабування, та опис результатів проведеного аналізу.

Результатами даного проекту стали діаграми та графіки, що зображають результати аналізу захворюваності на коронавірус у світі та в Україна зокрема. З ними можна ознайомитися в додатку А.

Шардування та репліціювання було налаштовано за допомогою docker-compose, налаштування якого можна знати в додатку Б.

Зміст

Вступ

Станом та поточний рік все більше і більше компаній приходять до того, що дані, якими вони оперують та збирають в процесі своєї роботи потребують агрегації та аналізу. Використання класичних методів аналізу, що базуються на звичайних алгоритмах математичної статистики та ймовірності не проносять очікуваної користі. З плином часу все більшу і більшу актуальність набирають програмні засоби аналізу даних, які інколи базуються на алгоритмах машинного навчання, що дозволяє зробити аналіз даних інтелектульним (*en. data minning*) Цю науку, яка поєднує в собі програмування, мат. статистику, теорію ймовірності та машине навчання називаються наукою про дані (*en. Data Science*).

Сучасний Data science спеціаліст (дослідник даних) має оперувати великими обсягами даних (від одного гігабайта до декількох терабайт) та вміти виокремити з них приховані залежності, на основі яких зробити прогноз про те, як будуть надалі відбуватия ті чи інші явища. Дослідники даних використовують свої дані та аналітичні здібності для пошуку та інтерпретації великих джерел даних; керують великими обсягами даних безвідносно до апаратного та програмного забезпечення і обмежень пропускної здатності; об'єднують джерела даних; забезпечують цілісність наборів даних; створюють візуалізації для кращого розуміння даних; з використанням даних будують математичні моделі; надають тлумачення даних та висновки.

Класичним набором інструментів, яким користується Data science спеціаліст є мова програмування Python 3 і набір математичних бібліотек до неї (pandas, scikit-learn, numpy, mathplotlib) нереляційні бази даних (на кшталт MongoDB, DynamoDB, Cassandra) та закони математичного аналізу, теорії ймовірності та мат. статистики).

Метою створення даного проекту був аналіз даних про перебіг епідемії коронавірусу в різних країнах світу, з метою дослідження закономірностей, згідно з якими росте кількість хворих. Критерієм актуальності було обрано той факт, що у багатьох країнах епідемія ще навіть не досягнула свого піку. Оскільки станом на даний момент в різних країнах триває різну кількість часу – цей факт дає можливість давати оцінку розвитку епідемії в країнах, де вона почала відносно недавно. Крім цього була дана кількісна оцінка росту кількості хворих, виздоровівших та летальних випадків у різних країнах.

Дані про перебіг епідемії в різних країнах було запозичено з відкритої інтернет-енциклопедії Вікіпедія, а дані про кількість населення в різних країнах із статистичного сайту worldometers.com/coronavirus.

Аналіз інструментарію для виконання курсового проект

Аналіз СКБД

В процесі виконання цього курсового проекту перед нами встала потреба кешувати та зберігати дані про вакансії між запусками аналізатора. Кожного разу читати дані із CSV - файлів є дуже дорогою операцією, тож було прийняте рішення використати СКБД. В якості СКБД були розглянуті варіанти: PostgreSQL, MongoDB. З порівняльною характеристикою цих СКБД можна ознайомитися в таблиці 1.

таблиця 1. Порівняльна характеристика СКБД

Критерій порівняння	Назва СКБД		
	MongoDB	PostgreSQL	CassandraDB
Має відкритий вихідний код	так	так	так
Схема даних	динамічна	статична і динамічна	статична і динамічна
Підтримка ієрархічних даних	так	так (з 2012)	ні
Реляційні дані	ні	так	так
Транзакції	ні	так	так
Атомарністі операцій	всередині документа	по всій БД	всередині партіції
Мова запитів	JSON	SQL	CQL

Найлегший спосіб масштабування	горизонтальний	вертикальний	горизонтальний
Підтримка шардингів	так	так (важка конфігурація)	так (може зберігати партіції на різних машинах)
Приклад використання	Великі дані (мільярди записів) з великою кількістю паралельних оновлень, де цілісність і узгодженість даних не потрібно.	Транзакційні і операційні програми, вигода яких в нормалізованому формі, об'єднаннях, обмеження даних і підтримки транзакцій.	Багато запитів на запис/читання у одиницю часу, до даних можна задіяти партіціюваннння за ключем, дані мають лише первинні індекси
Наявність бібліотек для мови програмування Python 3	так	так	так
Підтримка реплікації	так, автоматичне переобрання головного процесу	За принципом master-slave	так, через партіціювання

Засіб збреження та відновлення даних	mongodump	pg_dump	не має окремого доданка, виконується засобами CQL
Форма збереження даних	документи BSON	таблиця	таблиця

За результатами порівнянні цих СКБД було прийнято рішення зупинитися на NoSQL рішеннях. Оскільки вони чудово поєднує в собі переваги неструктурованих баз даних та простоту використання горизонтального масштабування. Крім цього класичним прикладом використання NoSQL СКБД є системи збору та аналізу даних, до яких можна застосувати індексування за первинними та вторинними ключами.

Ця база даних є об’єктно орієнтованою та дозволяє зберігати великі масиви неструктурованих даних. На відміну від SQL баз даних ми можемо зберігати дані у “сирому” об’єктному вигляді, який використовується програмою та є більш близьким за структурою до моделі даних, яку буде використовувати ПЗ написане з використанням мови програмування Python. Це пришвидшить збір, збереження та отримання даних програмним забезпеченням. Оскільки MongoDB є представником NoSQL баз даних, вона не потребує жорсткої схеми даних, що дозволяє пришвидшити процес розробки та зробити його більш гнучким. Окрім цього дана СУБД підтримує горизонтальне масштабування за допомогою шардингу з метою зменшення навантаження на кожен окремий вузол шляхом розподілення навантаження між ними всіма.

Нижче наведено перелік основних переваг:

- Підтримка ієрархічних даних
- Динамічна схема

- Швидкість запису у колекцію
- Швидкість читання із колекції
- Простота масштабування та відновлення даних

Обґрунтування вибору мови програмування

Мовою програмування для ПЗ було обрано Python 3.8. Оскільки це строго динамічно типізована мова програмування. Це дозволяє пришвидшити швидкість розробки ПЗ за рахунок уникнення постійного оголошення типів даних, якими оперує програма, та водночас не дає зробити критичних помилок, прихованих за автоматичним приведенням типів, на відміну від мови JavaScript.

Ключовою особливістю Python є широкий інструментарій засобів розробки систем збору та аналізу даних. Де-факто ця мова є стандартом у світі математичних розрахунків та обробки даних у реальному часі. Тож під час виконання курсового проекту було відносно легко знайти необхідну документацію та приклади роботи із цією мовою.

Обґрунтування вибору бібліотек і фреймворків

Використані бібліотеки:

- pandas — бібліотека для обробки та аналізу даних. Pandas є open-source бібліотекою, ліцензована BSD (Berkeley Source Distribution), забезпечує швидкі, гнучкі та експресивні структури даних, призначені для того, щоб працювати з «реляційними» або «помітними» даними як простими, так і інтуїтивно зрозумілими. Він прагне бути основним будівельним блоком високого рівня для здійснення практичного, реального аналізу даних у Python. Можливості бібліотеки: інструменти для обміну даними між структурами в пам'яті і файлами різних форматів, засоби поєднання даних і способи обробки відсутньої інформації, Переформатування наборів даних, в тому числі створення зведених таблиць, зріз даних за значеннями індексу, розширені можливості індексування, вибірка з великих наборів даних, вставка і видалення стовпців даних, можливості угруповання дозволяють виконувати трьохетапні операції типу «поділ, зміна, об'єднання», злиття і об'єднання наборів даних;
- scikit-learn — безкоштовна бібліотека алгоритмів машинного навчання програмного забезпечення для мови програмування Python, має прості та ефективні інструменти для аналізу даних та аналізу даних. Як і pandas має комерційне використання - ліцензія BSD. Вона має різні алгоритми класифікації, регресії та кластеризації, включаючи векторні машини підтримки, випадкові ліси, градієнтні підсилювачі, k-засоби та DBSCAN, і призначені для взаємодії з чисельними та науковими бібліотеками Python NumPy SciPy та matplotlib;

- matplotlib — це бібліотека Python 2D, яка представляє числові дані у різноманітних форматах та інтерактивних середовищах на різних платформах. Також ця бібліотека - це математичне розширення NumPy. Він надає об'єктно-орієнтований API для вбудови ділянок у додатки, що використовують набір інструментів для загального графічного інтерфейсу, таких як Tkinter, wxPython, Qt або GTK+. Matplotlib може використовуватися в скриптах Python, оболонках Python та IPython, серверах веб-додатків та чотирьох графічних наборів інструментів для користувацького інтерфейсу. Бібліотека sciPy використовує matplotlib.
- numpy — математична бібліотека мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами. А також інструменти для інтеграції C/C++ і Fortran коду, корисну лінійну алгебру, перетворення Фур'є і можливості випадкових чисел. Крім очевидних наукових застосувань, NumPy також може бути використаний як ефективний багатовимірний контейнер загальних даних. Можна визначити довільні типи даних. Це дає змогу без проблем і швидкої інтеграції NumPy з різноманітними базами даних. NumPy має ліцензію за ліцензією BSD, що дозволяє повторно використовувати кілька обмежень.

Структура бази даних

База даних складається з двох колекцій: countries і cases. Countries використовується для збереження даних про населення різних країн. Cases використовується для збереження детальних відомостей про нові випадки захворювання поденно.

таблиця 2. Опис властивостей документа у колекції countries

Назва властивості	Тип	Опис
_id	ObjectId	Ідентифікатор запису
name	Str	Назва країни
population	Int	Населення

таблиця 3. Опис властивостей документа у колекції cases

Назва властивості	Тип	Опис
_id	ObjectId	Ідентифікатор запису
country	Str	Назва країни
date	Date	Дата дня
deaths	Int	Загальна кількість смертей в країні станом на цей день
recoveries	Int	Загальна кількість одужавших в країні станом на цей день
active	Int	Загальна кількість хворих в країні станом на цей день
country_id	ObjectId	Ідентифікатор країни в таблиці countries

Аналіз функціонування засобів масштабування

В якості засобів масштабування було обрано реплікацію та шардинг.

Реплікація - це процес синхронізації даних на декількох серверах. Даний механізм зменшує витрати ресурсів і збільшує доступність даних, копії яких зберігаються на різних серверах. Реплікація захищає базу даних (далі - БД) від втрати єдиної сервера і дозволяє зберегти дані в разі технічної несправності на одному з серверів. У MongoDB реплікація досягається шляхом використання набору копій (replica set). Це група примірників mongod, який зберігають однакові набори даних. У копії один вузол - це ключовий вузол, який отримує всі операції запису. Всі інші вузли - вторинні, приймають операції з першого, таким чином, зберігаючи такі ж записи, як і первинний вузол. Набір копій може мати тільки один первинний вузол.

Шардінг - це процес зберігання документів на декількох серверах і це спосіб, яким MongoDB справляється з великими даними. З ростом кількості даних, один сервер не може зберігати всі даних, ні записувати їх, ні давати до них доступ. Шардінг вирішує проблему шляхом горизонтального масштабування. Завдяки даному механізму ми можемо підключати додаткові сервери для зберігання, записи і читання даних.

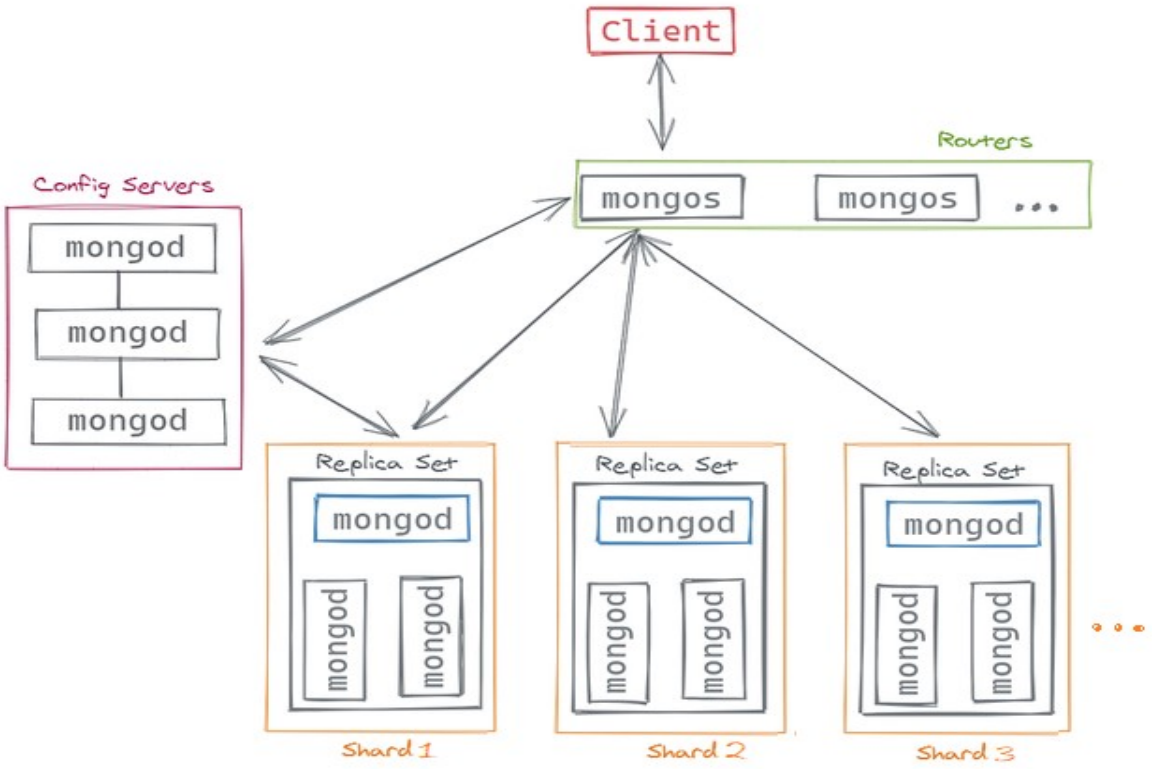

рис 1.Схема використаної моделі шардингу MongoDB у даному ПЗ

В якості ключа для розбивання даних на партіції було використано поле *date* оскільки воно зберігає дані із більш-менш лінійним розподілом по дням. Тож в якості критерія до партиціювання було використано хеш - індекс за властивістю *date*. Із результатами розбиття даних на партиції можна ознайомитися на рис 2.

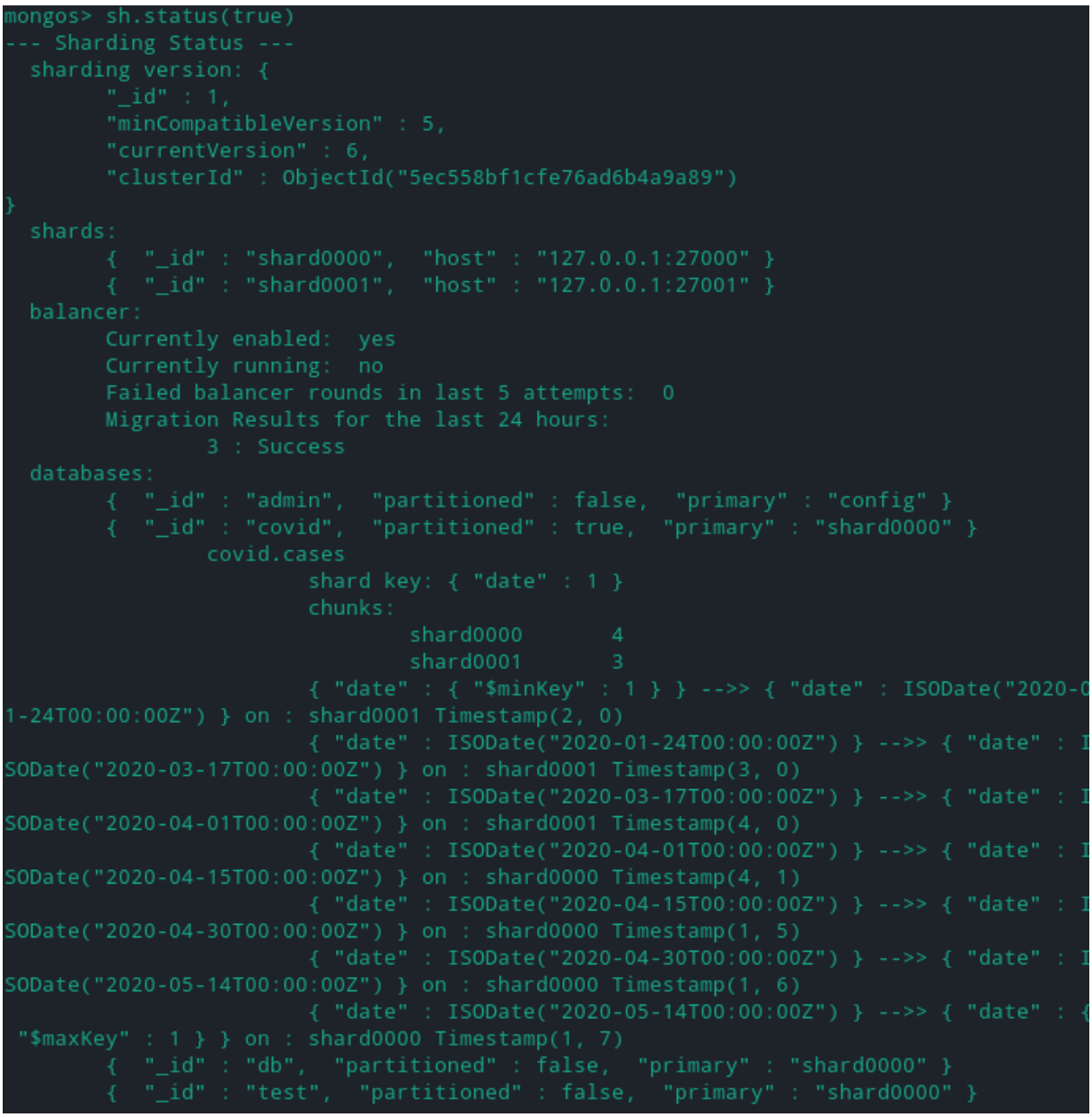

рис 2. Розподіл даних між шардами в рамках однієї колекції

Загальний алгоритм додавання нового кластеру шардингу, який складається з кількох серверів реплікації

1. Запустити на виконання один або декілька процессів mongod, які стануть вузлами шардингу за допомоги команди у таблиці 3. Також необхідно запустити я мінімум один конфігураційний сервер.

таблиця 3. Скрипт запуску необхідних вузлів шардингу

mongod --dbpath data/shard1 --port 27000 --fork --syslog

```
mongod --dbpath data/shard2 --port 27001 --fork --syslog
mongod --configsvr --dbpath data/config --port 27002 --fork --syslog
mongos --configdb localhost:27002 --port 27100 --fork --syslog
```

2. Дочекатися їх ініцілізації, під’єднатися до процесу mongos сервера-роутера та виконати команду наведену у таблиці 4. Ця команда спочатку прив“яже два сервера-шарди, створить індекс date, по якому дані будуть розподілятися між шардами, а після цього розшардує колекцію по цьому індексу.

таблиця 4. Скрипт створення нових вузлів шардингу

```
sh.addShard("127.0.0.1:27000")
sh.addShard("127.0.0.1:27001")
use covid
db.cases.ensureIndex({date: 1})
use admin
db.runCommand({shardCollection: "covid.cases", key {date: 1}})
```

Загальний алгоритм додавання нового вузла реплікації серверів конфігурації

Запустити на виконання один або декілька процессів mongod, які стануть вузлами реплікації.

1. Дочекатися їх ініцілізації, під’єднатися до процесу mongod одного з них та виконати команду наведену у таблиці 6.

таблиця 5. Скрипт додавання нового серверу конфігурації до реплікації

```
rs.add( { host: "<hostnameNew>:<portNew>", priority: 0, votes: 0 } )
```

Тестування масштабування

В процесі перевірки здатності системи масштабуватися, було проведено наступні тести:

1) Тестування відмовостійкості шляхом зупинки декількох процесів із репліки сету одного із шардингу
2) Тестування відмовостійкості системи шляхом зупинки декількох процесів із репліки сети серверів конфігурації
3) Перевірка того, яку частину даних ми втратимо, в разі відмови однієї реплікації серверів шардингу

1. Тестування відмовостійкості шляхом зупинки декількох процесів із репліки сету одного із шардингу

Нами було зупинено процес shard1a в результаті чого, ми очікували, що система перенаправить усі запити, що йшли до нього до його реплікації shard1b, та обере його головним, в разі якщо він не був таким. Вивід процесу mongos серверу маршрутизації наведено на рис. 3.

```
2019-06-02T19:46:14.097+0000 I NETWORK  [conn30] end connection 172.27.0.5:48430 (14 connections now open)
2019-06-02T19:46:14.100+0000 I NETWORK  [conn24] end connection 172.27.0.5:48412 (13 connections now open)
2019-06-02T19:46:14.101+0000 I NETWORK  [conn26] end connection 172.27.0.5:48418 (12 connections now open)
2019-06-02T19:46:14.418+0000 I NETWORK  [conn22] end connection 172.27.0.5:48364 (11 connections now open)
2019-06-02T19:46:14.551+0000 I CONNPOOL [Replication] dropping unhealthy pooled connection to shard01a:27018
2019-06-02T19:46:14.551+0000 I CONNPOOL [Replication] after drop, pool was empty, going to spawn some connections
2019-06-02T19:46:14.551+0000 I ASIO     [Replication] Connecting to shard01a:27018
2019-06-02T19:46:17.564+0000 I NETWORK  [conn29] end connection 172.27.0.5:48428 (10 connections now open)
2019-06-02T19:46:23.163+0000 I REPL     [replexec-3] Member shard01a:27018 is now in state RS_DOWN
2019-06-02T19:46:23.163+0000 I REPL     [replexec-3] can't see a majority of the set, relinquishing primary
2019-06-02T19:46:23.164+0000 I REPL     [replexec-3] Stepping down from primary in response to heartbeat
2019-06-02T19:46:23.165+0000 I REPL     [replexec-3] transition to SECONDARY from PRIMARY
2019-06-02T19:46:23.165+0000 I NETWORK  [replexec-3] Skip closing connection for connection # 28
2019-06-02T19:46:23.166+0000 I NETWORK  [conn13] end connection 172.27.0.3:52250 (9 connections now open)
2019-06-02T19:46:23.166+0000 I NETWORK  [conn16] end connection 172.27.0.8:36900 (6 connections now open)
2019-06-02T19:46:23.167+0000 I NETWORK  [conn20] end connection 172.27.0.7:54636 (5 connections now open)
2019-06-02T19:46:23.166+0000 I NETWORK  [conn14] end connection 172.27.0.6:34292 (8 connections now open)
2019-06-02T19:46:23.166+0000 I NETWORK  [conn21] end connection 172.27.0.10:48316 (7 connections now open)
2019-06-02T19:46:23.167+0000 I NETWORK  [conn17] end connection 172.27.0.2:34976 (3 connections now open)
2019-06-02T19:46:23.168+0000 I NETWORK  [conn18] end connection 172.27.0.9:50076 (2 connections now open)
2019-06-02T19:46:23.167+0000 I NETWORK  [conn19] end connection 172.27.0.4:43840 (4 connections now open)
2019-06-02T19:46:23.168+0000 I NETWORK  [conn15] end connection 172.27.0.11:33882 (1 connection now open)
2019-06-02T19:46:24.552+0000 I REPL_HB  [replexec-4] Error in heartbeat (requestId: 404) to shard01a:27018, response status:
NetworkInterfaceExceededTimeLimit: Couldn't get a connection within the time limit
2019-06-02T19:46:25.612+0000 I NETWORK  [listener] connection accepted from 172.27.0.7:54858 #32 (2 connections now open)
2019-06-02T19:46:25.614+0000 I NETWORK  [conn32] received client metadata from 172.27.0.7:54858 conn32: { driver: { name: "MongoDB Internal Client", version:
"4.0.9" }, os: { type: "Linux", name: "Ubuntu", architecture: "x86_64", version: "16.04" } }
2019-06-02T19:46:26.656+0000 I NETWORK  [listener] connection accepted from 172.27.0.10:48522 #33 (3 connections now open)
2019-06-02T19:46:26.657+0000 I NETWORK  [conn33] received client metadata from 172.27.0.10:48522 conn33: { driver: { name: "MongoDB Internal Client", version:
"4.0.9" }, os: { type: "Linux", name: "Ubuntu", architecture: "x86_64", version: "16.04" } }
2019-06-02T19:46:34.254+0000 I REPL     [replexec-4] Not starting an election, since we are not electable due to: Not standing for election because I cannot
see a majority (mask 0x1)
2019-06-02T19:46:34.552+0000 I ASIO     [Replication] Connecting to shard01a:27018
```

рис 3. Вивід процесу shard1b після того, вимкнення shard1a

Як видно із виводу, сервер *shard1b* не зміг отримати відповідь від своєї реплікації *shard1a* на запит перевірки “серцебиття” (*en. heartbeat*). В результаті чого він почав надсилати цей запит повторно кожну секунду та взяв на себе відповідальність бути лідером у реплікації із сервером *shard1a.* При цьому усі дані, які зберігалися у цьому кластері шардингу ще доступні на запис та читання. Після відновлення процесу *shard1a*, нами отримано наступний вивід процесу *shard1b*, наведений на рис 4. З нього стає зрозуміло що сервер *shard1b* зміг отримати відповідь від *shard1a* та запустив процес синхронізації та обрання нового лідера реплікації.

```
2019-06-02T19:40:05.443+0000 I ASIO     [Replication] Connecting to shard01a:27018
2019-06-02T19:40:05.445+0000 I ASIO     [Replication] Failed to connect to shard01a:27018 - HostUnreachable: Error
connecting to shard01a:27018 (172.27.0.5:27018) :: caused by :: Connection refused
2019-06-02T19:40:05.445+0000 I CONNPOOL [Replication] Dropping all pooled connections to shard01a:27018 due to
HostUnreachable: Error connecting to shard01a:27018 (172.27.0.5:27018) :: caused by :: Connection refused
2019-06-02T19:40:05.445+0000 I REPL_HB  [replexec-1] Error in heartbeat (requestId: 175) to shard01a:27018,
response status: HostUnreachable: Error connecting to shard01a:27018 (172.27.0.5:27018) :: caused by :: Connection
refused
2019-06-02T19:40:05.464+0000 W NETWORK  [ReplicaSetMonitor-TaskExecutor] Unable to reach primary for set shard01
2019-06-02T19:40:05.481+0000 I REPL     [rsBackgroundSync] waiting for 2 pings from other members before syncing
2019-06-02T19:40:05.866+0000 I NETWORK  [listener] connection accepted from 172.27.0.5:48408 #23 (11 connections
now open)
2019-06-02T19:40:05.869+0000 I NETWORK  [conn23] end connection 172.27.0.5:48408 (10 connections now open)
2019-06-02T19:40:05.875+0000 I NETWORK  [listener] connection accepted from 172.27.0.5:48412 #24 (11 connections
now open)
2019-06-02T19:40:05.875+0000 I NETWORK  [conn24] received client metadata from 172.27.0.5:48412 conn24: { driver:
{ name: "NetworkInterfaceTL", version: "4.0.9" }, os: { type: "Linux", name: "Ubuntu", architecture: "x86_64",
version: "16.04" } }
2019-06-02T19:40:05.945+0000 I ASIO     [Replication] Connecting to shard01a:27018
2019-06-02T19:40:05.949+0000 I REPL     [replexec-1] Member shard01a:27018 is now in state SECONDARY
2019-06-02T19:40:06.225+0000 I REPL     [replexec-0] Starting an election, since we've seen no PRIMARY in the past
10000ms
2019-06-02T19:40:06.225+0000 I REPL     [replexec-0] conducting a dry run election to see if we could be elected.
current term: 2
2019-06-02T19:40:06.226+0000 I REPL     [replexec-1] VoteRequester(term 2 dry run) received a yes vote from
shard01a:27018; response message: { term: 2, voteGranted: true, reason: "", ok: 1.0, operationTime:
Timestamp(1559503702, 3), $gleStats: { lastOpTime: Timestamp(0, 0), electionId:
ObjectId('000000000000000000000000') }, lastCommittedOpTime: Timestamp(0, 0), $configServerState: { opTime: { ts:
Timestamp(1559504403, 1), t: 1 } }, $clusterTime: { clusterTime: Timestamp(1559504405, 1), signature: { hash:
BinData(0, 0000000000000000000000000000000000000000), keyId: 0 } } }
2019-06-02T19:40:06.226+0000 I REPL     [replexec-1] dry election run succeeded, running for election in term 3
2019-06-02T19:40:06.232+0000 I REPL     [replexec-2] VoteRequester(term 3) received a yes vote from
shard01a:27018; response message: { term: 3, voteGranted: true, reason: "", ok: 1.0, operationTime:
Timestamp(1559503702, 3), $gleStats: { lastOpTime: Timestamp(0, 0), electionId:
ObjectId('000000000000000000000000') }, lastCommittedOpTime: Timestamp(0, 0), $configServerState: { opTime: { ts:
Timestamp(1559504403, 1), t: 1 } }, $clusterTime: { clusterTime: Timestamp(1559504405, 1), signature: { hash:
BinData(0, 0000000000000000000000000000000000000000), keyId: 0 } } }
2019-06-02T19:40:06.232+0000 I REPL     [replexec-2] election succeeded, assuming primary role in term 3
2019-06-02T19:40:06.232+0000 I REPL     [replexec-2] transition to PRIMARY from SECONDARY
2019-06-02T19:40:06.232+0000 I REPL     [replexec-2] Resetting sync source to empty, which was :27017
2019-06-02T19:40:06.232+0000 I REPL     [replexec-2] Entering primary catch-up mode.
2019-06-02T19:40:06.233+0000 I REPL     [replexec-2] Caught up to the latest optime known via heartbeats after
becoming primary.
2019-06-02T19:40:06.233+0000 I REPL     [replexec-2] Exited primary catch-up mode.
2019-06-02T19:40:06.233+0000 I REPL     [replexec-2] Stopping replication producer
2019-06-02T19:40:07.516+0000 I REPL     [rsSync-0] transition to primary complete; database writes are now
permitted
2019-06-02T19:40:07.881+0000 I NETWORK  [listener] connection accepted from 172.27.0.5:48418 #26 (12 connections
now open)
2019-06-02T19:40:07.881+0000 I NETWORK  [conn26] received client metadata from 172.27.0.5:48418 conn26: { driver:
{ name: "NetworkInterfaceTL", version: "4.0.9" }, os: { type: "Linux", name: "Ubuntu", architecture: "x86_64",
version: "16.04" } }
2019-06-02T19:40:07.882+0000 I SH_REFR  [ConfigServerCatalogCacheLoader-0] Refresh for database config took 0 ms
and found { _id: "config", primary: "config", partitioned: true }
2019-06-02T19:40:07.883+0000 I ASIO     [ShardRegistry] Connecting to config03:27017
2019-06-02T19:40:07.890+0000 I SHARDING [conn26] setting this node's cached database version for config to {}
2019-06-02T19:40:07.959+0000 I NETWORK  [listener] connection accepted from 172.27.0.6:34598 #28 (13 connections
now open)
```

рис 4. Вивід процесу shard1b після відновлення процесу shard1a

2. Тестування відмовостійкості системи шляхом зупинки декількох процесів із репліки сети серверів конфігурації

Сервери конфігурація, які знаходять у реплікації поводять себе так само, як і при тестуванні реплікації кластерів шардів. Після невдачі отримати відповідь на запит перевірки серцебиття (рис 5) від серверу конфігурації (*config1*), кожен з процесів серверу конфігурації, що ще онлайн робить запит до вимкненого сервера кілька раз на секунду. Крім цього відбувається обрання нового лідера реплікації. Після поновлення *config1* відбувається обрання нового лідера реплікації та система переходить до звичайного режиму роботи (рис 6)

```
2019-06-02T19:59:58.789+0000 I NETWORK  [conn85] end connection 172.27.0.3:34694 (21 connections now open)
2019-06-02T19:59:58.799+0000 I NETWORK  [conn87] end connection 172.27.0.3:34700 (20 connections now open)
2019-06-02T20:00:20.241+0000 I NETWORK  [conn13] end connection 172.27.0.7:51964 (19 connections now open)
2019-06-02T20:00:20.244+0000 I NETWORK  [conn8] end connection 172.27.0.7:51932 (18 connections now open)
2019-06-02T20:00:21.963+0000 I CONNPOOL [Replication] dropping unhealthy pooled connection to config03:27017
2019-06-02T20:00:21.963+0000 I CONNPOOL [Replication] after drop, pool was empty, going to spawn some connections
2019-06-02T20:00:21.963+0000 I ASIO     [Replication] Connecting to config03:27017
2019-06-02T20:00:21.971+0000 I ASIO     [Replication] Failed to connect to config03:27017 - HostUnreachable: Error connecting to config03:27017 :: caused by :: Could not find
address for config03:27017: SocketException: Host not found (authoritative)
2019-06-02T20:00:21.971+0000 I CONNPOOL [Replication] Dropping all pooled connections to config03:27017 due to HostUnreachable: Error connecting to config03:27017 :: caused by ::
Could not find address for config03:27017: SocketException: Host not found (authoritative)
2019-06-02T20:00:21.972+0000 I REPL_HB  [replexec-42] Error in heartbeat (requestId: 3811) to config03:27017, response status: HostUnreachable: Error connecting to config03:27017
:: caused by :: Could not find address for config03:27017: SocketException: Host not found (authoritative)
2019-06-02T20:00:21.972+0000 I ASIO     [Replication] Connecting to config03:27017
2019-06-02T20:00:21.977+0000 I ASIO     [Replication] Failed to connect to config03:27017 - HostUnreachable: Error connecting to config03:27017 :: caused by :: Could not find
address for config03:27017: SocketException: Host not found (authoritative)
2019-06-02T20:00:21.977+0000 I CONNPOOL [Replication] Dropping all pooled connections to config03:27017 due to HostUnreachable: Error connecting to config03:27017 :: caused by ::
Could not find address for config03:27017: SocketException: Host not found (authoritative)
2019-06-02T20:00:21.978+0000 I REPL_HB  [replexec-39] Error in heartbeat (requestId: 3813) to config03:27017, response status: HostUnreachable: Error connecting to config03:27017
:: caused by :: Could not find address for config03:27017: SocketException: Host not found (authoritative)
2019-06-02T20:00:21.978+0000 I ASIO     [Replication] Connecting to config03:27017
```

рис 5. Спроби отримати відповідь від вимкненого сервера конфігурації config1

```
2019-06-02T20:08:25.508+0000 I ASIO     [Replication] Connecting to config03:27017
2019-06-02T20:08:25.515+0000 I ASIO     [Replication] Failed to connect to config03:27017 - HostUnreachable: Error
connecting to config03:27017 :: caused by :: Could not find address for config03:27017: SocketException: Host not
found (authoritative)
2019-06-02T20:08:25.515+0000 I CONNPOOL [Replication] Dropping all pooled connections to config03:27017 due to
HostUnreachable: Error connecting to config03:27017 :: caused by :: Could not find address for config03:27017:
SocketException: Host not found (authoritative)
2019-06-02T20:08:25.516+0000 I REPL_HB  [replexec-29] Error in heartbeat (requestId: 17392) to config03:27017,
response status: HostUnreachable: Error connecting to config03:27017 :: caused by :: Could not find address for
config03:27017: SocketException: Host not found (authoritative)
2019-06-02T20:08:27.414+0000 I NETWORK  [listener] connection accepted from 172.27.0.7:38784 #83 (18 connections
now open)
2019-06-02T20:08:27.415+0000 I NETWORK  [conn83] end connection 172.27.0.7:38784 (17 connections now open)
2019-06-02T20:08:27.422+0000 I NETWORK  [listener] connection accepted from 172.27.0.7:38788 #84 (18 connections
now open)
2019-06-02T20:08:27.422+0000 I NETWORK  [conn84] received client metadata from 172.27.0.7:38788 conn84: { driver:
{ name: "NetworkInterfaceTL", version: "4.0.9" }, os: { type: "Linux", name: "Ubuntu", architecture: "x86_64",
version: "16.04" } }
2019-06-02T20:08:27.516+0000 I ASIO     [Replication] Connecting to config03:27017
2019-06-02T20:08:27.520+0000 I REPL     [replexec-29] Member config03:27017 is now in state SECONDARY
2019-06-02T20:08:28.428+0000 I NETWORK  [listener] connection accepted from 172.27.0.7:38792 #86 (19 connections
now open)
2019-06-02T20:08:28.428+0000 I NETWORK  [conn86] received client metadata from 172.27.0.7:38792 conn86: { driver:
{ name: "NetworkInterfaceTL", version: "4.0.9" }, os: { type: "Linux", name: "Ubuntu", architecture: "x86_64",
version: "16.04" } }
2019-06-02T20:08:28.442+0000 I NETWORK  [listener] connection accepted from 172.27.0.7:38794 #87 (20 connections
now open)
2019-06-02T20:08:28.443+0000 I NETWORK  [conn87] received client metadata from 172.27.0.7:38794 conn87: { driver:
{ name: "NetworkInterfaceTL", version: "4.0.9" }, os: { type: "Linux", name: "Ubuntu", architecture: "x86_64",
version: "16.04" } }
```

рис 6. Створення нового пулу з’єднань із config1 після його поновлення та обрання нового лідера репліки

Перевірка того, яку частину даних ми втратимо, в разі відмови однієї реплікації серверів шардингу

Нами було зупинено процеси *shard1a* та *shard2a*. В результаті чого частина даних, яка зберігалися на цих серверах, повинна бути втрачена, а після поновлення хоча б одного процесу shard1X вони знову повинні бути доступними. Внаслідок розірвання зв’зку між серверами конфігурації та кластерами шардингу, ми не могли запустити запит на виконання до бд, що стосувався даних у колекції, яка була розбита на партиції. Після поновлення процесів *shard1a* та *shard2a* цілісність даних було відновлено і ми знову змогли виконати запит до колекції.

Опис результатів аналізу предметної галузі

В результаті виконання курсового проекту було проаналізовано перебіг епідемії в різних країнах. Було отримано наступні дані:

1. Згідно з рисунком А1

   Загалом країни за характером епідемії поділяються на 3 кластери: країни із спокійним перебігом епідемії, країни із епідемією середньої важкості, та країни, в яких епідемія набула екстремальних масштабів.

2. Згідно із рисунком А2

За допомогою автоматичного будування графіків реалізовано можливість порівняння перебігу епідемії в різних країнах беручи до уваги різні складові захворюваності.

3. Згідно із рисунком А3

Знайдено криву залежності загальної кількості летальних випадків від дати для України.

4. Згідно із рисунком А4

Знайдено криву залежності загальної кількості активних випадків захворювання від дати для України.

5. Згідно із рисунком А5

Знайдено криву залежності загальної кількості видужавших від дати для України.

6. Згідно із рисунком А6

Знайдено криву залежності загальної кількості усіх випадків захворюваня від дати для України.

7. Згідно із рисунком А7

   Знайдено криву залежності нової кількості летальних випадків від дати для України.

8. Згідно із рисунком А8

Знайдено криву залежності нової кількості видужавших від дати для України.

9. Згідно із рисунком А9

Знайдено криву залежності загальної кількості усіх випадків захворбвання від дати для України.

Висновки

В процесі виконання даного курсового проекту було отримано практичні навички обробки великих масивів даних за допомогою мови програмування Python 3 та СКБД MongoDB.

Було проаналізовано сучасні методи та інструменти для роботи із великими даними, знайдено гарно працюючу комбінацію із мови програмування Python 3 та бібліотек до нього: Pandas, Scikit-learn, Mathplotlib, Numpy. Було складено порівняльну таблицю із трьох баз даних, які найчастіше використовують для роботи із часовими рядами. На основі цих даних було обрано в якості СКБД NoSQL рішення MongoDB.

Для забезпечення горизонтального масштабування було використано засоби MongoDB, такі як: репліка сет та шардинг. Практичним шляхом було знайдено спосіб партиціювання даних часових рядів на основі хеш-функції від мітки часу. Це дозволило розподілити дані, які зберігаються в рамках однієї колекції між багатьма серверами кластерів шардингу. За допомогою реплікації було досягнуто відмовостійкість системи, в результаті чого, під час тестування ми впевнилися, що система продовжує функціонування після виходу із ладу кількох реплік.

На основі зібраних даних було проаналізовано ринок кіноіндустрії у світі за останні 20 років. Знайдено кореляції та залежності між різними показниками (бюджетом та касовими зборами на різним етапах після виходу фільму в прокат). Дані аналізу приведені у Додатку Б. Текстовий опис надано в відповідному розділу цього курсового проекту.

В ході виконання даного курсового проекту було досягнуто поставленої мети: було набуто практичних навичок розробки сучасного програмного забезпечення, що взаємодіє з NoSQL базами даних, а також були здобуті навички оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проектної документації. У результаті

виконання курсового проекту я навчився писати програмне забезпечення для NoSQL баз даних, володіти основами використання СУБД, а також інструментальними засобами аналізу великих обсягів даних, а саме відкритими бібліотеками мови Python.

Література

1. Hashed sharding [Електронний ресурс]. Режим доступу до ресурсу: https://docs.mongodb.com/manual/core/hashed-sharding/
2. Aggregation [Електронний ресурс]. Режим доступу до ресурсу: https://docs.mongodb.com/manual/aggregation/
3. 5 причин, почему машинное обучение станет технологией 2018 [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://blogs.oracle.com/russia/machine-learning-2018;
4. Сложности накопления данных для интеллектуального анализа [Електронний ресурс]. – 2012. – Режим доступу до ресурсу: https://habr.com/post/154787/;
5. Почему Python так хорош в научных вычислениях [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://habr.com/post/349482/;
6. Python [Електронный ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Python;
7. PostgreSQL лучше чем MongoDB[Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://habr.com/post/272735/;
8. 27 шпаргалок п машинному обучению [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://proglib.io/p/ds-cheatsheets/;
9. Pandas [Електронний ресурс]. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Pandas;
10. Scikit-learn [Електронний ресурс]. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Scikit-learn;
11. Matplotlib [Електронний ресурс]. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Matplotlib;
12. NumPy [Електронний ресурс]. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/NumPy;

13. 15.Горизонтальное масштабировнаие: когда и как? [Електронний ресурс]. – Режим доступу до ресурсу: https://habr.com/company/oleg-bunin/blog/319526

Додаток А

```
['Sierra Leone', 'Morocco', 'Myanmar', 'Latvia', 'Sri Lanka', 'North Macedonia', 'Mala
wi', 'Ukraine', 'Jamaica', 'Libya', 'Saint Kitts and Nevis', 'Pakistan', 'Kuwait', 'Ka
zakhstan', 'New Zealand', 'Burkina Faso', 'Colombia', 'Bulgaria', 'Slovakia', 'Burundi
', 'Peru', 'Eswatini', 'Saudi Arabia', 'Luxembourg', 'Dominica', 'Saint Lucia', 'Armen
ia', 'Germany', 'Montserrat', 'Ghana', 'Iceland', 'Vietnam', 'Sint Maarten', 'Curaçao'
, 'Vatican City', 'Ethiopia', 'Nigeria', 'Uganda', 'Switzerland', 'Aruba', 'Belgium',
'Republic of Ireland', 'Montenegro', 'Finland', 'Cameroon', 'Lebanon', 'Poland', 'Réun
ion', 'Barbados', 'Argentina', 'Taiwan', 'Bangladesh', 'Portugal', 'Russia', 'French P
olynesia', 'Nicaragua', 'Belarus', 'Norway', 'Jordan', 'Costa Rica', 'Bosnia and Herze
govina', 'Kenya', 'Algeria', 'South Africa', 'Croatia', 'Bhutan', 'Mayotte', 'Serbia',
 'Egypt', 'Hong Kong', 'Indonesia', 'Uzbekistan', 'Australia', 'India', 'Mauritania',
'Panama', 'Syria', 'Afghanistan', 'Malaysia', 'Yemen', 'Guyana', 'Togo', 'Anguilla',
Greece', 'Cyprus', 'Greenland', 'Tunisia', 'Kyrgyzstan', 'Mali', 'Ivory Coast', 'Turke
y', 'Senegal', 'Estonia', 'Brunei', 'Macau', 'Denmark', 'Ecuador', 'Malta', 'Bahrain',
 'Benin', 'Azerbaijan', 'Moldova', 'Oman', 'New Caledonia', 'Thailand', 'Lithuania',
Venezuela', 'Singapore', 'Israel', 'El Salvador', 'Albania', 'Canada', 'Romania', 'Slo
venia', 'Trinidad and Tobago', 'Mexico', 'Honduras', 'Papua New Guinea', 'Hungary', 'I
raq', 'Uruguay', 'San Marino', 'Tanzania', 'Iran', 'Angola', 'Qatar', 'Suriname', 'Fre
nch Guiana', 'Fiji', 'Tajikistan', 'Japan', 'Mauritius', 'Paraguay', 'Chile']

Cluster number: 1
['Brazil', 'Italy', 'Spain', 'France', 'United Kingdom']

Cluster number: 2
['United States']
```

Рис 1. Розподіл країн по кластерам за характером перебігу епідемії

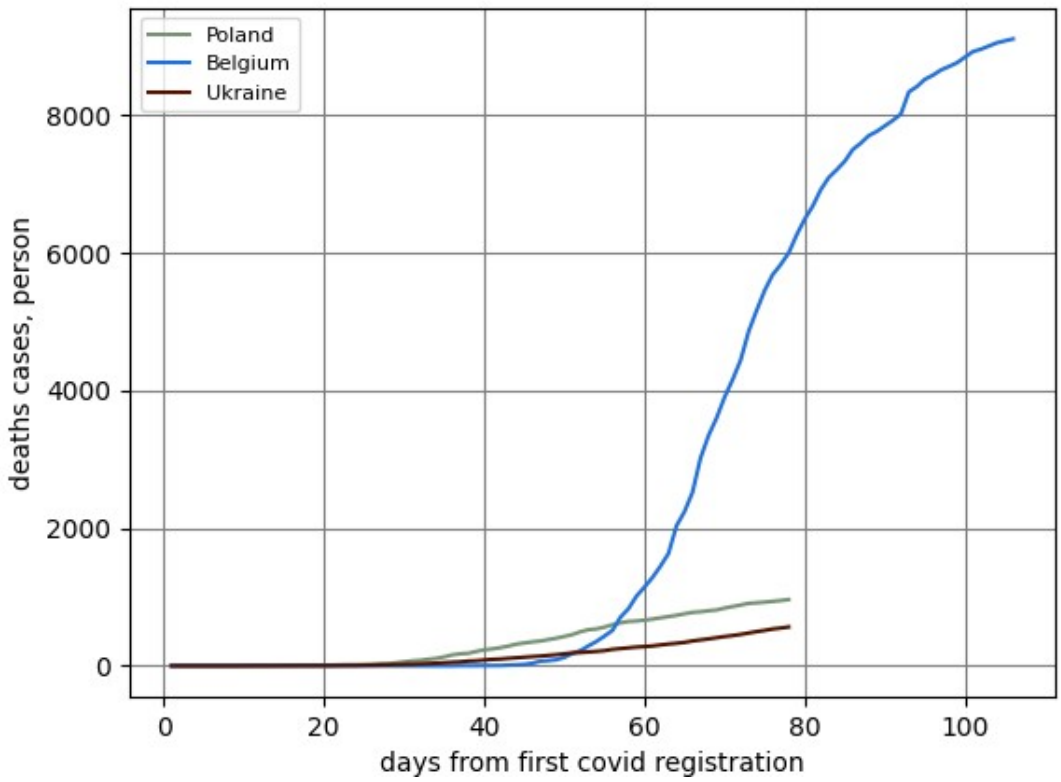

Рис 2. Графічне порівняннякількості летальноих випадків і країнах

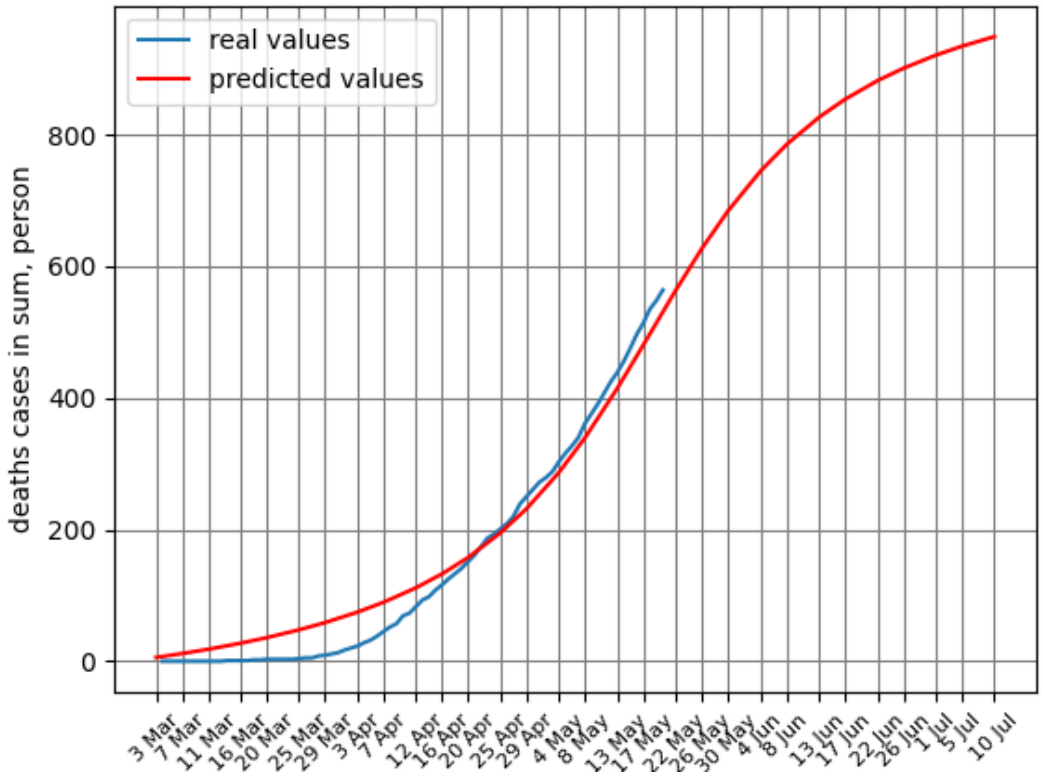

Рис 3. Прогноз кількості летальних випадків в Україні

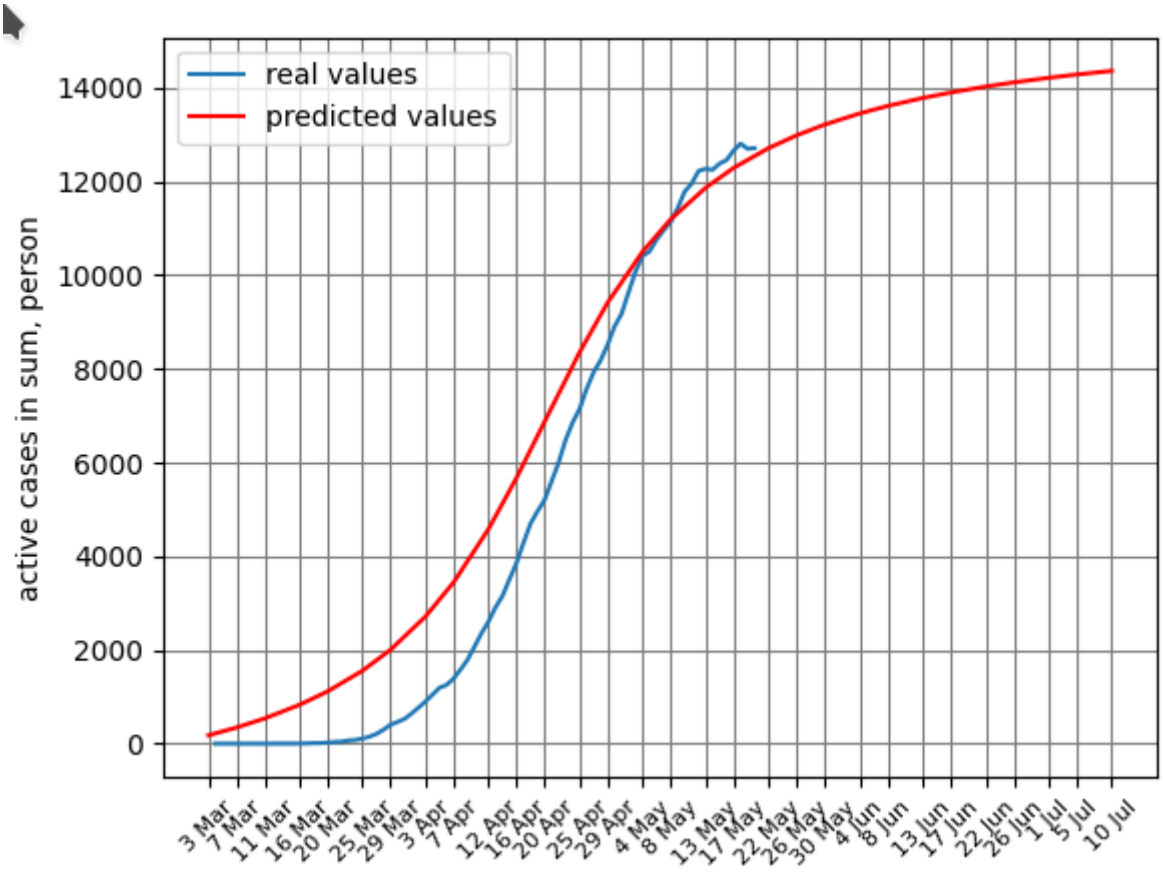

Рис 4. Прогноз кількості активних випадків в Україні

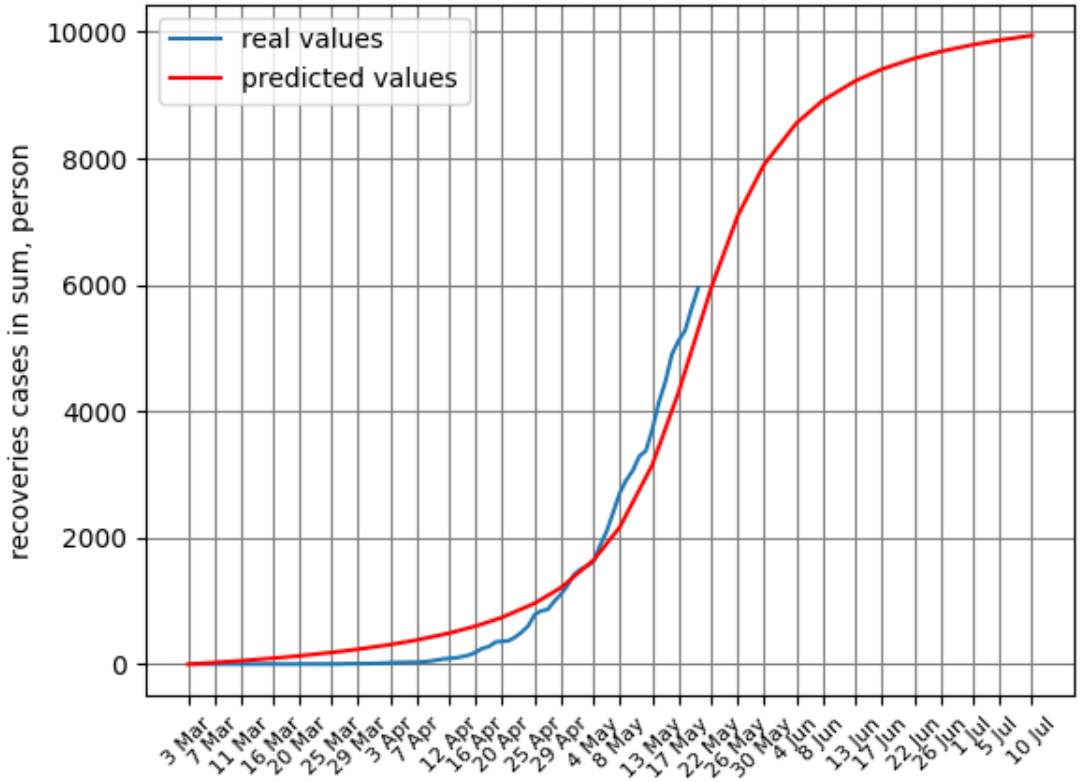

Рис 5. Прогноз кількості одужавших в Україні

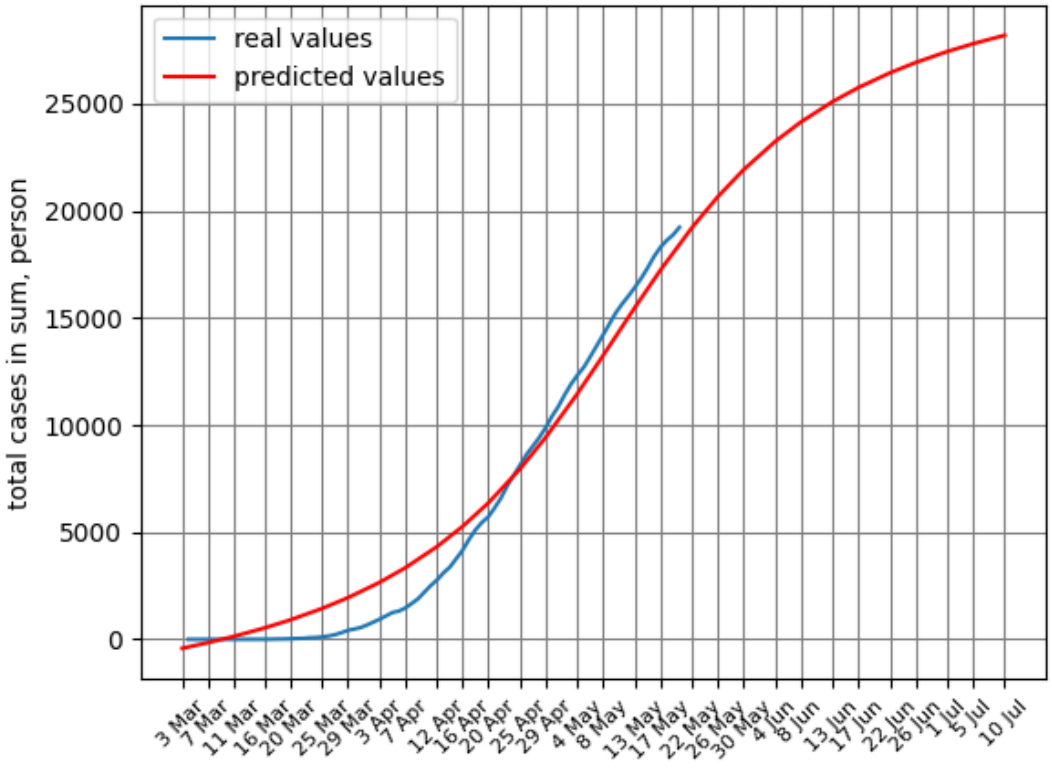

Рис 6. Прогноз загальної кількості випадків захворювання в Україні

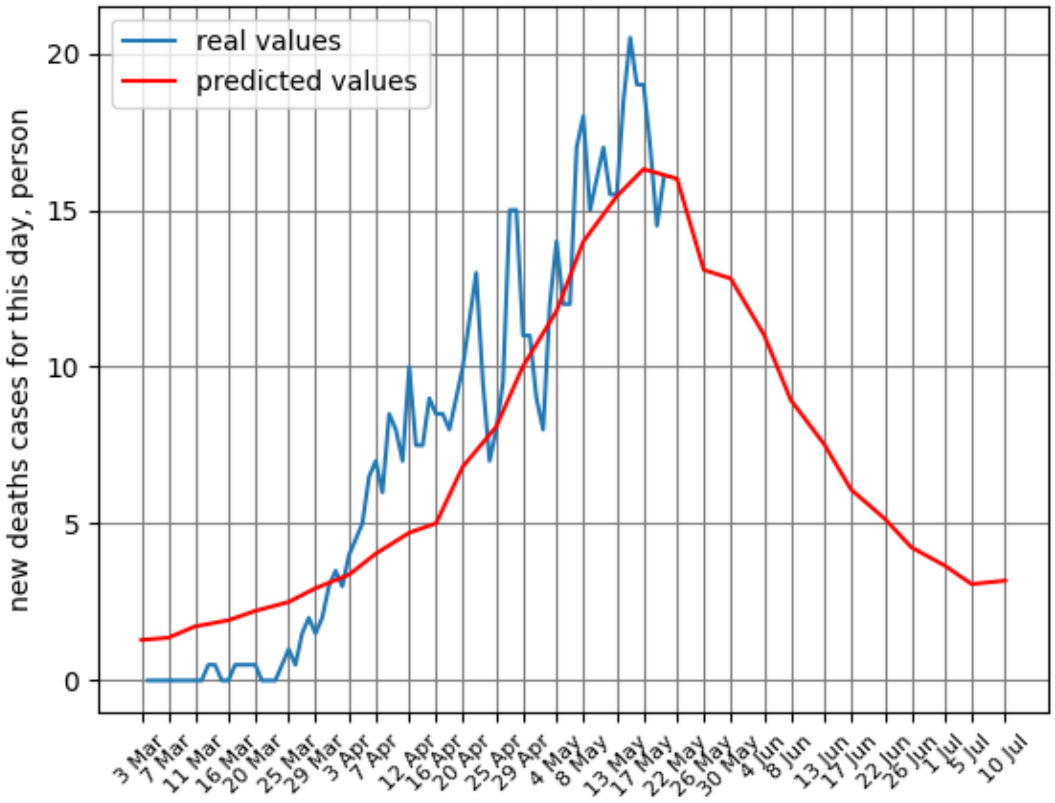

Рис 7. Прогноз кількості нових летальних випадків для України

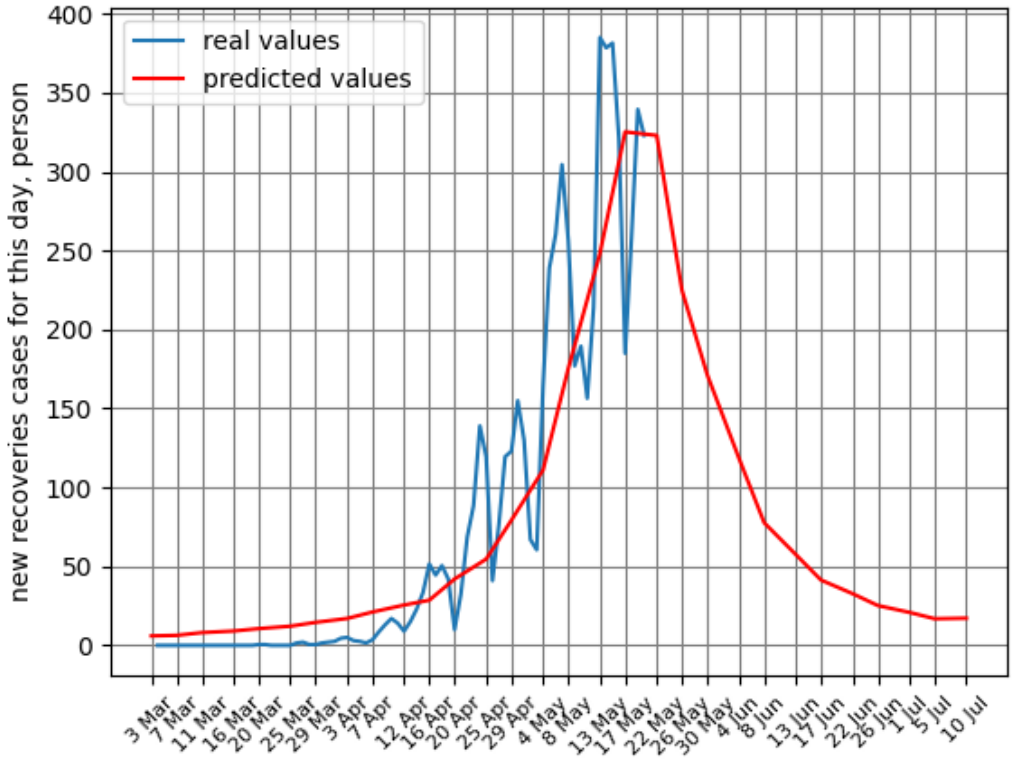

Рис 8. Прогноз щоденної кількості одужавших в Україні

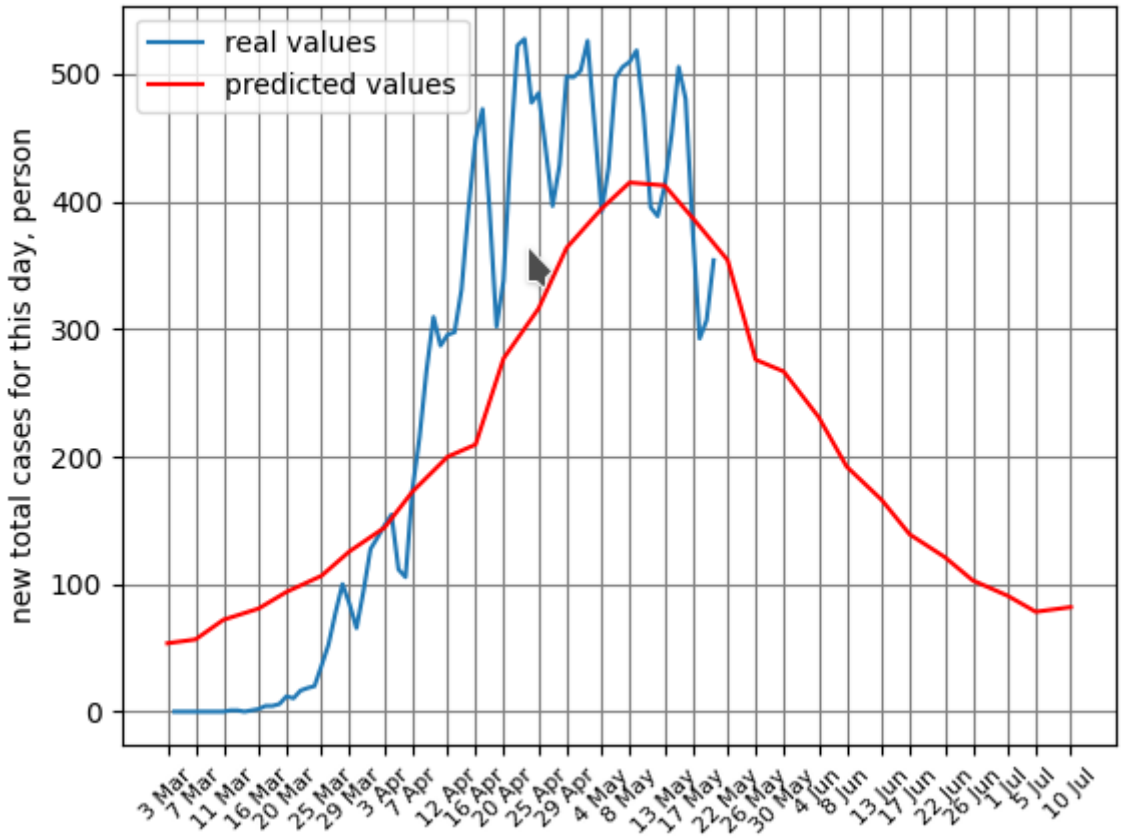

Рис 9. Прогноз щоденного збільшення усіх випадків захворювання в Україні

Додаток Б

Таблиця 1. Вміст конфігураційного файлу docker-compose

docker-compose.yml

```
version: '2'
services:

## Router
  router01:
    image: mongo:4.0
    container_name: rydell-router-01
    command: mongos --port 27017 --configdb
rs-config-server/configsvr01:27017,configsvr02:27017,configsvr03:27017 --bind_ip_all
    ports:
      - 27117:27017
    volumes:
      - ./scripts:/scripts
  router02:
    image: mongo:4.0
    container_name: rydell-router-02
    command: mongos --port 27017 --configdb
rs-config-server/configsvr01:27017,configsvr02:27017,configsvr03:27017 --bind_ip_all
    volumes:
      - ./scripts:/scripts
    ports:
      - 27118:27017
    links:
      - router01

## Config Servers
  configsvr01:
    image: mongo:4.0
    container_name: rydell-mongo-config-01
    command: mongod --port 27017 --configsvr --replSet rs-config-server
    volumes:
      - ./scripts:/scripts
    ports:
      - 27119:27017
    links:
      - shard01-a
      - shard02-a
      - shard03-a
  configsvr02:
    image: mongo:4.0
    container_name: rydell-mongo-config-02
```

```
    command: mongod --port 27017 --configsvr --replSet rs-config-server
    volumes:
      - ./scripts:/scripts
    ports:
      - 27120:27017
    links:
      - configsvr01
  configsvr03:
    image: mongo:4.0
    container_name: rydell-mongo-config-03
    command: mongod --port 27017 --configsvr --replSet rs-config-server
    volumes:
      - ./scripts:/scripts
    ports:
      - 27121:27017
    links:
      - configsvr02

## Shards
  ## Shards 01
  shard01-x:
    image: mongo:4.0
    container_name: rydell-shard-01-node-x
    command: mongod --port 27017 --shardsvr --replSet rs-shard-01
    volumes:
      - ./scripts:/scripts
    ports:
      - 27122:27017
    links:
      - shard01-a
      - shard01-b
  shard01-a:
    image: mongo:4.0
    container_name: rydell-shard-01-node-a
    command: mongod --port 27017 --shardsvr --replSet rs-shard-01
    volumes:
      - ./scripts:/scripts
    ports:
      - 27123:27017
  shard01-b:
    image: mongo:4.0
    container_name: rydell-shard-01-node-b
    command: mongod --port 27017 --shardsvr --replSet rs-shard-01
    volumes:
      - ./scripts:/scripts
    ports:
      - 27124:27017
```

```
## Shards 02
shard02-x:
  image: mongo:4.0
  container_name: rydell-shard-02-node-x
  command: mongod --port 27017 --shardsvr --replSet rs-shard-02
  volumes:
    - ./scripts:/scripts
  ports:
    - 27125:27017
  links:
    - shard02-a
    - shard02-b
shard02-a:
  image: mongo:4.0
  container_name: rydell-shard-02-node-a
  command: mongod --port 27017 --shardsvr --replSet rs-shard-02
  volumes:
    - ./scripts:/scripts
  ports:
    - 27126:27017
shard02-b:
  image: mongo:4.0
  container_name: rydell-shard-02-node-b
  command: mongod --port 27017 --shardsvr --replSet rs-shard-02
  volumes:
    - ./scripts:/scripts
  ports:
    - 27127:27017

## Shards 03
shard03-x:
  image: mongo:4.0
  container_name: rydell-shard-03-node-x
  command: mongod --port 27017 --shardsvr --replSet rs-shard-03
  volumes:
    - ./scripts:/scripts
  ports:
    - 27128:27017
  links:
    - shard03-a
    - shard03-b
shard03-a:
  image: mongo:4.0
  container_name: rydell-shard-03-node-a
  command: mongod --port 27017 --shardsvr --replSet rs-shard-03
  volumes:
```

```
    - ./scripts:/scripts
  ports:
    - 27129:27017
shard03-b:
  image: mongo:4.0
  container_name: rydell-shard-03-node-b
  command: mongod --port 27017 --shardsvr --replSet rs-shard-03
  volumes:
    - ./scripts:/scripts
  ports:
    - 27130:27017
```