Звіт
до лабораторної роботи 2
з дисципліни “Сучасні технології баз даних”
виконав студент 3 курсу факультету
комп’ютерних наук та кібернетики
групи МІ-31
Бідзіля Святослав Олексійович

Постановка задачі

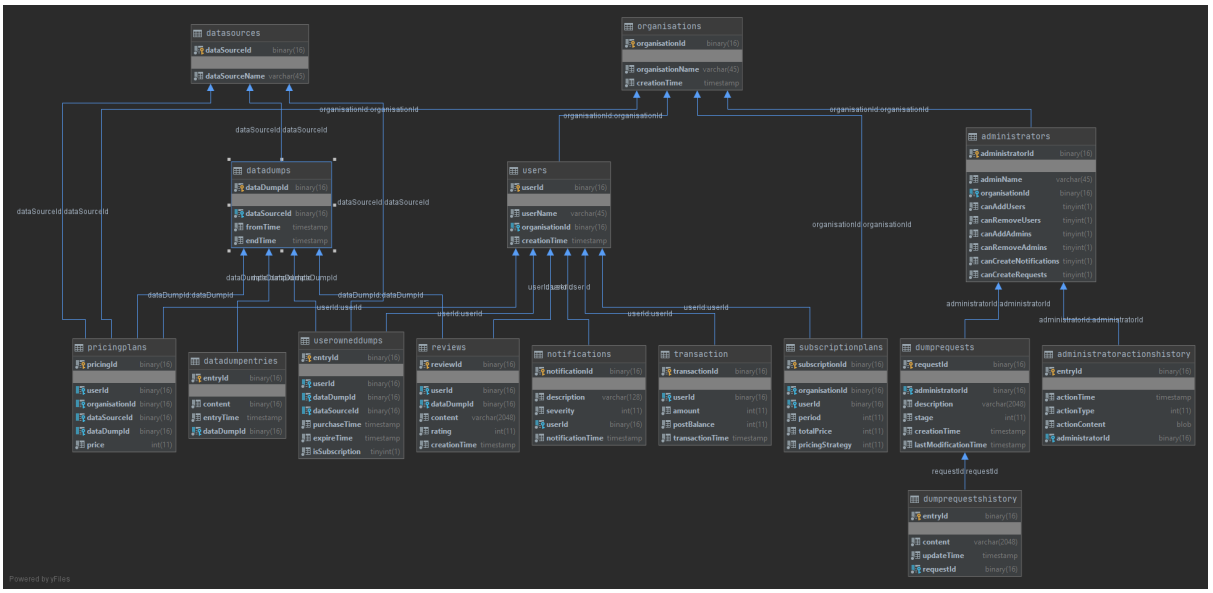
NoSQL

NoSQL бази все частіше використовуються для розв'язання нішевих задач. У випадках простих чи пет-проектів вони можуть повністю замінити реляційну базу.

Задача:
1) ознайомитись з видами NoSQL баз
2) враховуючи вашу предметну область, чи буде ефективніше винести частину даних в іншу базу? Навести аргументи
3) Винести частину даних у обрану вами NoSQL базу, чи розширити вашу предметну область(чи вимоги до уявного застосунку) такими вимогами, які б робили використання NoSQL доцільним. Реалізувати. (обрання типу бази залишаю за вами)
4) Реалізуйте теж саме в SQL базі.
5) Тестування швидкодії. Порівняти швидкодію відповідних SQL та NoSQL фрагментів баз. NoSQL має відображатись у декілька SQL таблиць. Перевірка швидкодії - процесс доволі творчий, але має бути логічно обгрунтований.

Виконання

За основу виконання була взята БД з лабораторної роботи номер 1 (https://github.com/anakib1/KNU_2023_BD_lab1). За посиланням наявний детальний опис таблиць, тут прикріплена лише схема:



Розширення задачі

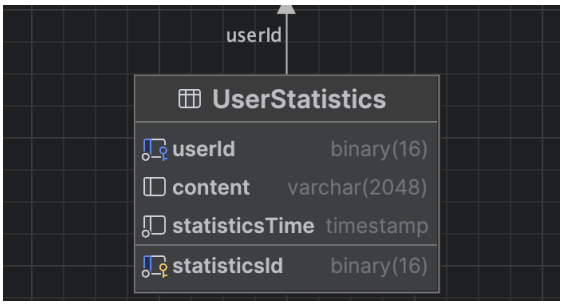
Розглянемо необхідність додатково підтримувати такі дані для адміністративної панелі застосунку:

1. Активні наразі користувачі - такі, які залогінилися, і ще не вийшли з програми
2. Статистика активних користувачів - скільки разів він залогінився, виходив, скільки чанків даних він завантажив. Тут під чанком даних розуміється сутність dataDumpEntry.
3. Активні наразі dataDumpEntry - ті, які недавно були завантажені будь-яким з користувачів. Ідея за цими даними така, всі ентріс можуть знаходитися на віддаленому та відносно повільному сервері для перманентного зберігання. Але активні дані переносятся до більш швидкого сервера для покращення швидкості доступу.

Зміни в SQL

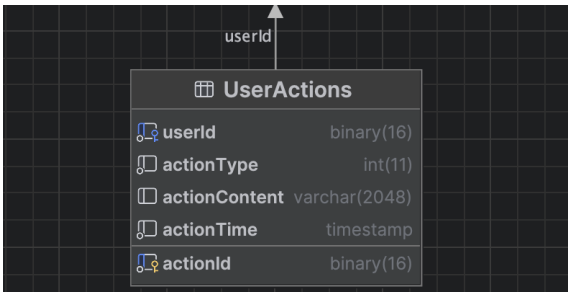
Для розширення потрібно додати такі SQL таблиці:

UserStatistics



Всі дані статистики з метою можливих легких розширень були винесені в `content` та зберігаються як json, або будь-яке інше повідомлення, що підтримує серіалізацію.

UserActions



Аналогічно, actionType - це тип дії (логін, завантаження, логаут)
actionContent - будь-які дані цієї дії.

За допомогою таблиці userActions можна легко реалізувати обчислення активних користувачів - це всі користувачі, для яких час останнього логіна був пізніше останнього логаута.
Для покращення швидкодії SQl, в обох таблицях було додано індекси за (userId, time) - для швидкого пошуку останньої дії/статистики для користувача.

Використання NoSQL db

За NoSQL db було вибрано redis (https://redis.io/) через минулий досвід роботи з цією системою БД

В redis було створено 3 ключі:
actions, statistics, activeUsers в форматі:

activeUsers - множина всіх активних користувачів наразі.

ID (Total: 700)	Key	Value
1	52863E9D-E8C2-47...	PerformanceUser33
2	DD2D9985-59D7-4...	PerformanceUser13
3	0BC0A6F9-08AC-41...	PerformanceUser52
4	1A848A5E-DF5A-4C...	PerformanceUser7
5	5F9BC259-DAD9-4...	PerformanceUser76
6	DF1EEE8B-889E-40...	PerformanceUser82
7	E031ADB8-6D65-4E...	PerformanceUser11
8	A6E4794F-7BF1-44...	PerformanceUser27
9	E843E2EE-29EB-42...	PerformanceUser75

statistics:{userId} - відсортована за часом множина статистик заданого користувача

Zset | statistics:0019B86C-8767-4623-A562-E655CAB336FF | TTL | -1

Add New Line

ID (Total: 100)	Score	Member	Keyword Search
1	1701257453359	{"loginCnt": 1, "logoutCnt": 0, "downloadCnt": 199}	
2	1701257453077	{"loginCnt": 1, "logoutCnt": 0, "downloadCnt": 197}	
3	1701257452800	{"loginCnt": 1, "logoutCnt": 0, "downloadCnt": 195}	
4	1701257452512	{"loginCnt": 1, "logoutCnt": 0, "downloadCnt": 193}	
5	1701257452234	{"loginCnt": 1, "logoutCnt": 0, "downloadCnt": 191}	
6	1701257451953	{"loginCnt": 1, "logoutCnt": 0, "downloadCnt": 189}	
7	1701257451672	{"loginCnt": 1, "logoutCnt": 0, "downloadCnt": 187}	
8	1701257451392	{"loginCnt": 1, "logoutCnt": 0, "downloadCnt": 185}	
9	1701257451116	{"loginCnt": 1, "logoutCnt": 0, "downloadCnt": 183}	
10	1701257450834	{"loginCnt": 1, "logoutCnt": 0, "downloadCnt": 181}	
11	1701257450545	{"loginCnt": 1, "logoutCnt": 0, "downloadCnt": 179}	
12	1701257450258	{"loginCnt": 1, "logoutCnt": 0, "downloadCnt": 177}	
13	1701257449951	{"loginCnt": 1, "logoutCnt": 0, "downloadCnt": 175}	
14	1701257449637	{"loginCnt": 1, "logoutCnt": 0, "downloadCnt": 173}	

actions:{userId} - аналогічно дії відсортовані за часом для кожного користувача

Zset | actions:001FCDE4-D043-4011-807F-53B20685DBE0 | TTL | -1

Add New Line

ID (Total: 93)	Score	Member	Keyword Search
1	1701257453672	{"content" : "Performance user logout", "type" : "LOGOUT"}	
2	1701257453368	{"content" : "Downloaded dataset #176", "type" : "DOWNLOAD_DATASE...	
3	1701257453085	{"content" : "Downloaded dataset #844", "type" : "DOWNLOAD_DATASE...	
4	1701257452808	{"content" : "Downloaded dataset #679", "type" : "DOWNLOAD_DATASE...	
5	1701257452520	{"content" : "Downloaded dataset #54", "type" : "DOWNLOAD_DATASET"}	
6	1701257452242	{"content" : "Downloaded dataset #443", "type" : "DOWNLOAD_DATASE...	
7	1701257451961	{"content" : "Downloaded dataset #633", "type" : "DOWNLOAD_DATASE...	
8	1701257451681	{"content" : "Downloaded dataset #812", "type" : "DOWNLOAD_DATASE...	
9	1701257451401	{"content" : "Downloaded dataset #895", "type" : "DOWNLOAD_DATASE...	
10	1701257451124	{"content" : "Downloaded dataset #645", "type" : "DOWNLOAD_DATASE...	
11	1701257450842	{"content" : "Downloaded dataset #531", "type" : "DOWNLOAD_DATASE...	
12	1701257450553	{"content" : "Downloaded dataset #926", "type" : "DOWNLOAD_DATASE...	
13	1701257450266	{"content" : "Downloaded dataset #516", "type" : "DOWNLOAD_DATASE...	
14	1701257449961	{"content" : "Downloaded dataset #965", "type" : "DOWNLOAD_DATASE...	
15	1701257449645	{"content" : "Downloaded dataset #865", "type" : "DOWNLOAD_DATASE...	
16	1701257449363	{"content" : "Downloaded dataset #924", "type" : "DOWNLOAD_DATASE...	

Реалізація клієнту

Для реалізації взаємодії було реалізовано клієнт мовою пайтон, який має такі методи:

createNewUser(user) - додає нового користувача то БД
loginExistingUser(user) - Логінить нового користувача (додає action)
logoutExistingUser(user) - логаутить користувача (додає action)
createNewDataset(data) - створює новий DataDumpEntry
downloadDatasetForUser(user, dataset) - додає дію - завантаження датасета
updateStats() - оновлює статистику для усіх користувачів, які зараз залогінені

Всю реалізацію можна знайти в репозиторії:
https://github.com/anakib1/KNU_2023_BD_lab2_client

SQL-Only

Для SQL only клієнту реалізовано клас DataSquidClient:

```python
class DataSquidClient:
    def __init__(self, adaptor:DatabaseAdaptor):
        self.adaptor = adaptor
        self.users = []
        self.datasets = []
    def createNewUser(self, user:User):
        self.users.append(user)
        self.adaptor.writeNewUser(self.users[-1])
    def loginExistingUser(self, userId):
        self.adaptor.writeNewAction(Action(userId, ActionType.LOGIN, 'Performance user login'))
    def logoutExistingUser(self, userId):
        self.adaptor.writeNewAction(Action(userId, ActionType.LOGOUT, 'Performance user logout'))
    def createNewDataset(self):
        self.datasets.append(f'PerformanceDataset{len(self.datasets)}')
        self.adaptor.writeNewDataset(self.datasets[-1])

    def downloadDatasetForUser(self, userId, datasetId):
        self.adaptor.writeNewAction(Action(userId, ActionType.DOWNLOAD_DATASET, f'Downloaded dataset #{datase

    def updateStats(self):
        currentTime = int(time_ns()/1_000_000)
        users = self.adaptor.queryActiveUsers()
        for userId in users:
            latestStats = self.adaptor.queryLatestUserStatistics(userId)
            if latestStats == None:
                latestStats = Stats(userId)
                latestStats.timestamp = 0
            actions = self.adaptor.queryUserActions(userId, latestStats.timestamp, currentTime)
            latestStats.mergeWith(actions)
            self.adaptor.writeNewStats(latestStats)
```

Він викликає дії лише DatabaseAdaptor (реалізація наявна також в репозиторії, там немає нічого важливого для тестування, лише форматування та виклик SQL запитів до mariaDB)

SQL+Redis

Для SQL+Redis клієнту було реалізовано клас DataSquidClientEx - він вносить додатково до SQL інформацію і в редіс ДБ

```python
class DataSquidClientEx:
    def __init__(self, adaptor:DatabaseAdaptor, redis:RedisAdaptor):
        self.adaptor = adaptor
        self.redis = redis
        self.redis.flush()
        self.users = {}
        self.datasets = []

    def createNewUser(self, user:User):
        self.users[user.userId] = user
        self.adaptor.writeNewUser(user)


    def loginExistingUser(self, userId):
        action = Action(userId, ActionType.LOGIN, 'Performance user login')
        self.adaptor.writeNewAction(action)
        self.redis.addUserToActiveSet(self.users[userId])
        self.redis.addActionEntry(action)

    def logoutExistingUser(self, userId):
        action = Action(userId, ActionType.LOGOUT, 'Performance user logout')
        self.adaptor.writeNewAction(action)
        self.redis.removeUserFromActiveSet(self.users[userId])
        self.redis.addActionEntry(action)

    def createNewDataset(self):
        self.datasets.append(f'PerformanceDataset{len(self.datasets)}')
        self.adaptor.writeNewDataset(self.datasets[-1])

    def downloadDatasetForUser(self, userId, datasetId):
        action = Action(userId, ActionType.DOWNLOAD_DATASET, f'Downloaded dataset #{datasetId}')
        self.adaptor.writeNewAction(action)
        self.redis.addActionEntry(action)

    def updateStats(self):
        currentTime = int(time_ns() / 1_000_000)
        users = self.redis.queryAllActiveUsers()
        for userId in users:
            latestStats = self.redis.queryLatestUserStats(userId)
            if latestStats == None:
                latestStats = Stats(userId)
                latestStats.timestamp = 0
            actions = self.redis.queryUserActions(userId, latestStats.timestamp, currentTime)
            latestStats.mergeWith(actions)
            self.adaptor.writeNewStats(latestStats)
            self.redis.addStatEntry(latestStats)
```

Він додатково до DatabaseAdaptor використвоує ще й RedisAdaptor - аналогічно реалізований адаптор, який є просто обгорткою навколо офіційного редіс-клієнту для нашої задачі. Цей клієнт дулблює створення всіх сутностей в редіс, а також робить запити по знаходженню не в SQL, а в редіс, що працює швидше.

Тестування

Протокол тестування

Тестування виконувалося на такому сценарії. Для параметрів totalUsers, loggedInUsers, totalDatasets, totalIterations, downloadsPerIter ми робимо таку послідовність дій:

1. Створємо totalUsers користувачів
2. Створюємо totalDatasets датасетів
3. Логінимо loggedInUsers з totalUsers
4. повторюємо дії 5, 6 totalIterations разів
5. для кожного користувача скачуємо випадкові downloadsPerIter датасетів
6. оновлюємо статистику
7. вилогінимо всіх користувачів

Ми заміряємо, скільки часу виконуються дії в реалізації на SQL та в гібридній реалізації з двома БД.

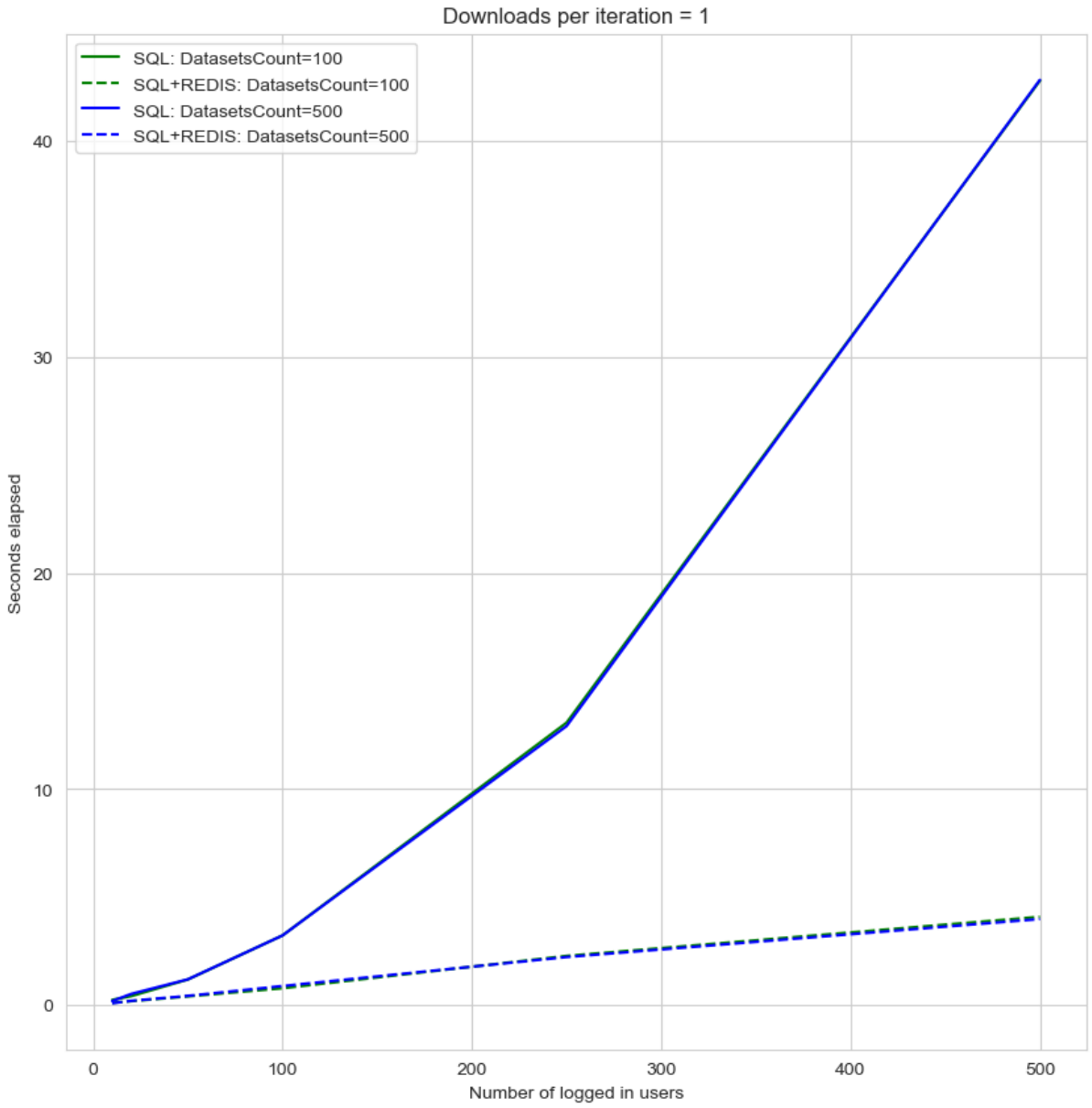
Реалізація тестування

Реалізовано також класом TestRunner на пайтоні, який дозволяє робити аналіз результатів зручніше.
Перед кожним тестом і змінені таблиці БД і весь редіс очищується.

```
def run_testing(self):
    start_time = time()

    for user in self.users[:self.logged_in_users]:
        self.client.loginExistingUser(userId=user.userId)

    for iteration in range(self.total_iterations):
        for user in self.users[:self.logged_in_users]:
            for datasetDownloads in range(self.dataset_downloads):
                chosen_dataset = random.randint(0, self.total_datasets - 1)
                self.client.downloadDatasetForUser(user.userId, chosen_dataset)

        self.client.updateStats()

    for user in self.users[:self.logged_in_users]:
        self.client.logoutExistingUser(userId=user.userId)

    end_time = time()

    return (end_time - start_time)

def initRunTesting(useEx:bool, total:int, loggedIn:int, datasets:int, iterations:int, perIterDownloads:int):
    client = DataSquidClientEx(DatabaseAdaptor(DatabaseOperator()), RedisAdaptor()) if useEx else DataSquidClient(DatabaseAdaptor(DatabaseOperator()))

    stress = Stress(
        client=client,
        total_users=total,
        logged_in_users=loggedIn,
        total_datasets=datasets,
        total_iterations=iterations,
        datasets_downloads=perIterDownloads,
    )

    return stress.run_testing()
```

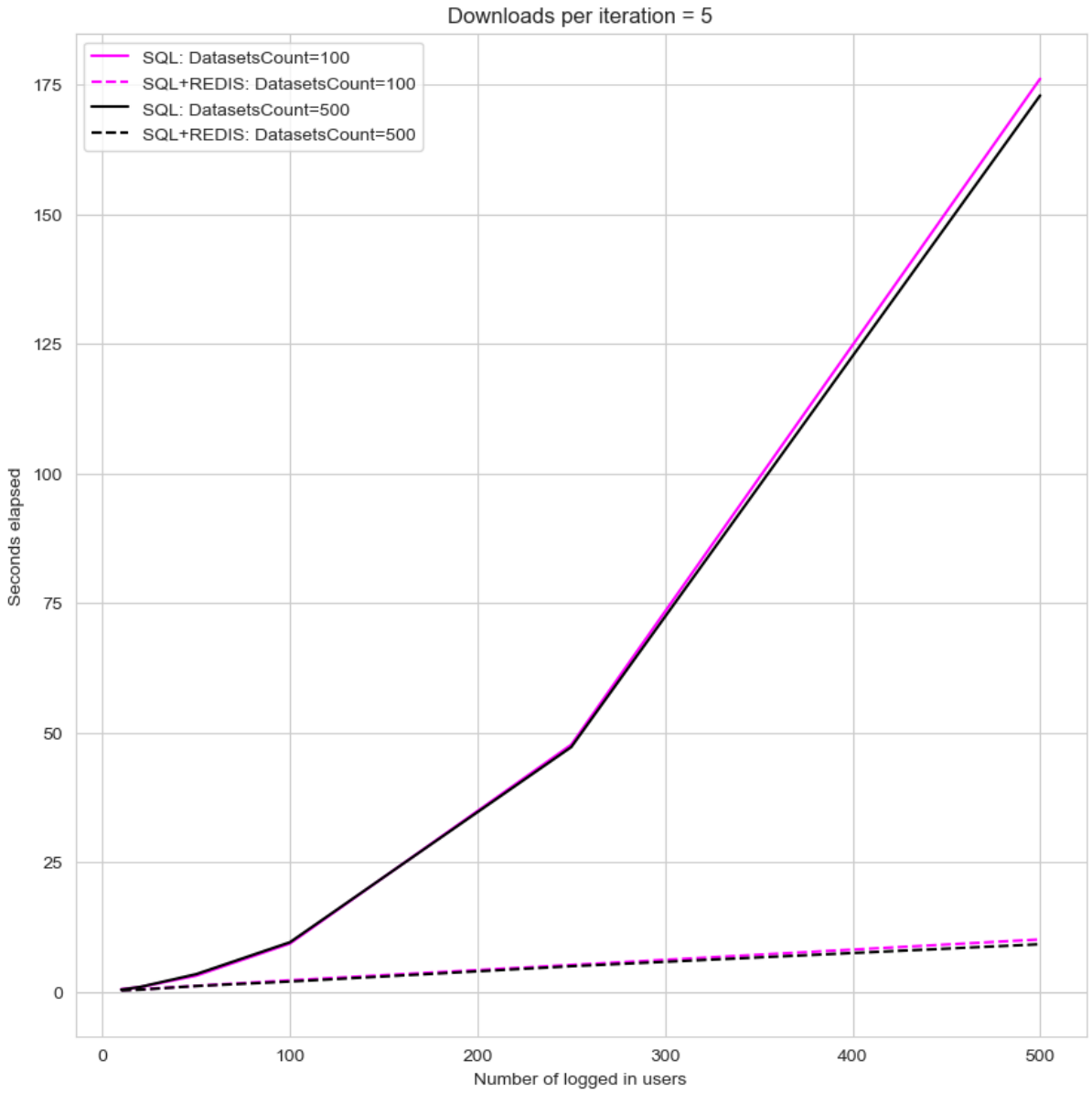
Результати тестування

Наводжу результати тестування для обраних параметрів тестування:



Як ми бачимо, швидкість реалізації з використанням redis в декілька разів перевищує швидкість з використанням лише SQL.

Ще сильніше це помітно, якщо ми виконуємо 5 загрузок даних для кожного користувача:



Дані в форматі таблиці:

	10(SQL)	10(SQL+)	20(SQL)	20(SQL+)	50(SQL)	50(SQL+)	100(SQL)	100(SQL+)	250(SQL)	250(SQL+)	500(SQL)	500(SQL+)
100D1R	0.241038	0.106109	0.411759	0.183149	1.186145	0.401592	3.225867	0.778823	13.078661	2.276618	42.805386	4.087094
100D5R	0.463111	0.271552	0.865073	0.439426	3.079803	1.110753	9.277195	2.190448	47.648808	5.154669	176.134717	10.068613
500D1R	0.186803	0.085333	0.516274	0.195917	1.192929	0.426709	3.226323	0.882650	12.919350	2.223650	42.845380	3.983056
500D5R	0.386469	0.205240	0.889613	0.367480	3.372416	1.083434	9.528657	1.942385	47.189740	4.900755	172.916681	9.141543

Висновки

В цій лабораторній роботі ми порівняли реалізацію розширення БД за допомогою SQL таблиць та за бази даних Redis.
До плюсів використання Redis можна віднести - високу швидкодію, легкість реалізації.

До мінусів - відсутність перманентного зберігання, необхідність дублювати дані в основній базі. Мінус може бути вирішеним за допомогою бекапів, які уповільнюють базу, але додають надійності.

З результатів ми бачимо, що, реалізація з redis працює майже в 17 разів швидше на найбільших тестах і набагато краще масштабується, не додаючи додаткових проблем до реалізації, що в конкретному випадку (підтримка активної множини користувачів) дуже покращує можливості застосунку.

Що ще можна додати в майбутньому 🙂:

1. порівняння в інших сценаріях
2. використання batch operations
3. використання асинхронних операцій в обох адаптерах