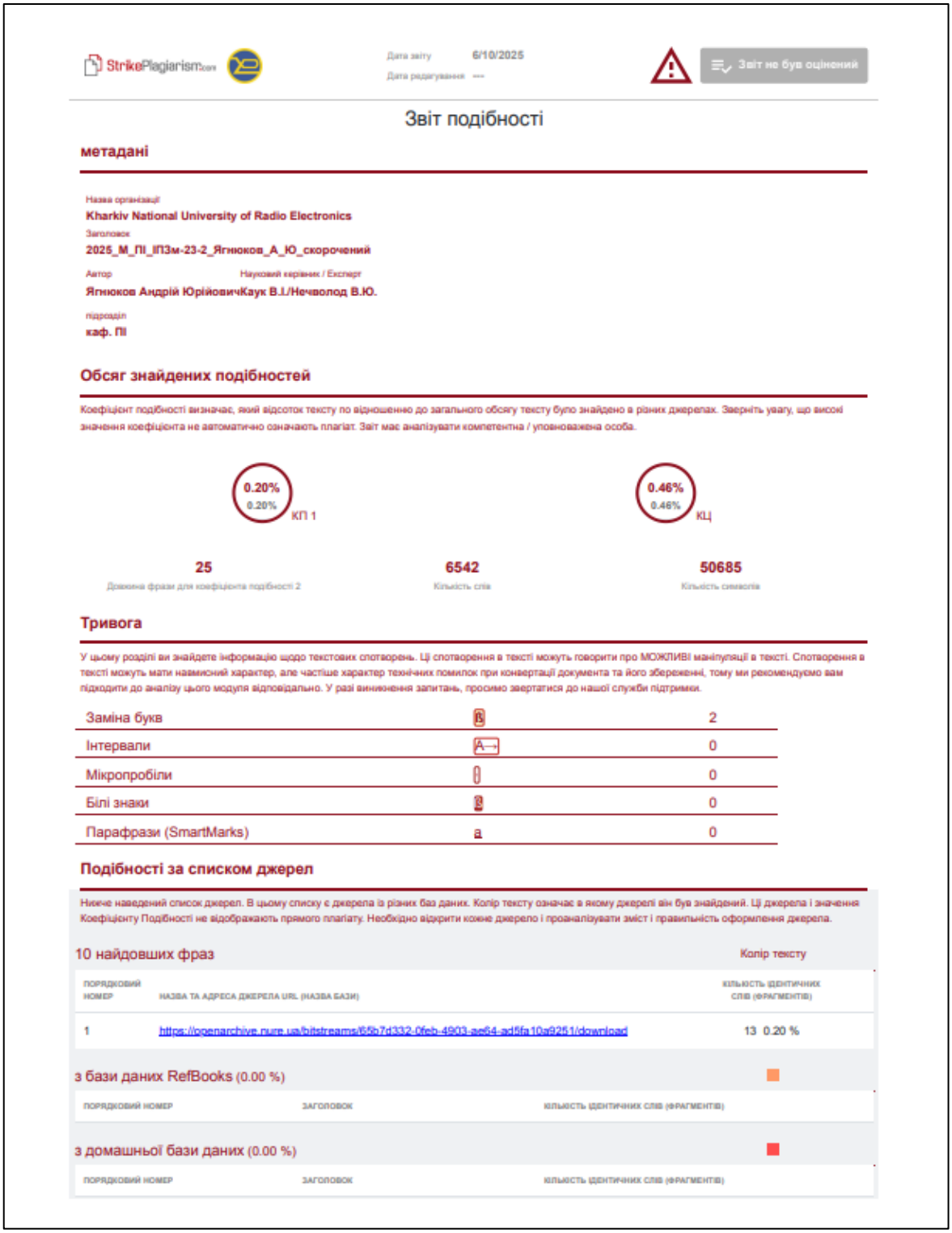
ДОДАТОК А

РЕЗУЛЬТАТ ПРОХОДЖЕННЯ СИСТЕМИ ПЕРЕВІРКИ ДОБРОЧЕСНОСТІ



StrikePlagiarism.com

Дата звіту 6/10/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_M_ПІ_ІП3м-23-2_Ягнюков_А_Ю_скорочений
Автор
Ягнюков Андрій Юрійович
Науковий керівник / Експерт
Каук В.І./Нечволод В.Ю.
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

0.20% 0.20% КП 1

0.46% 0.46% КЦ

25	6542	50685
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		2
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)	a	0

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://openarchive.nure.ua/bitstreams/65b7d332-0feb-4903-ae64-ad5fa10a9251/download	13 0.20 %

з бази даних RefBooks (0.00 %)

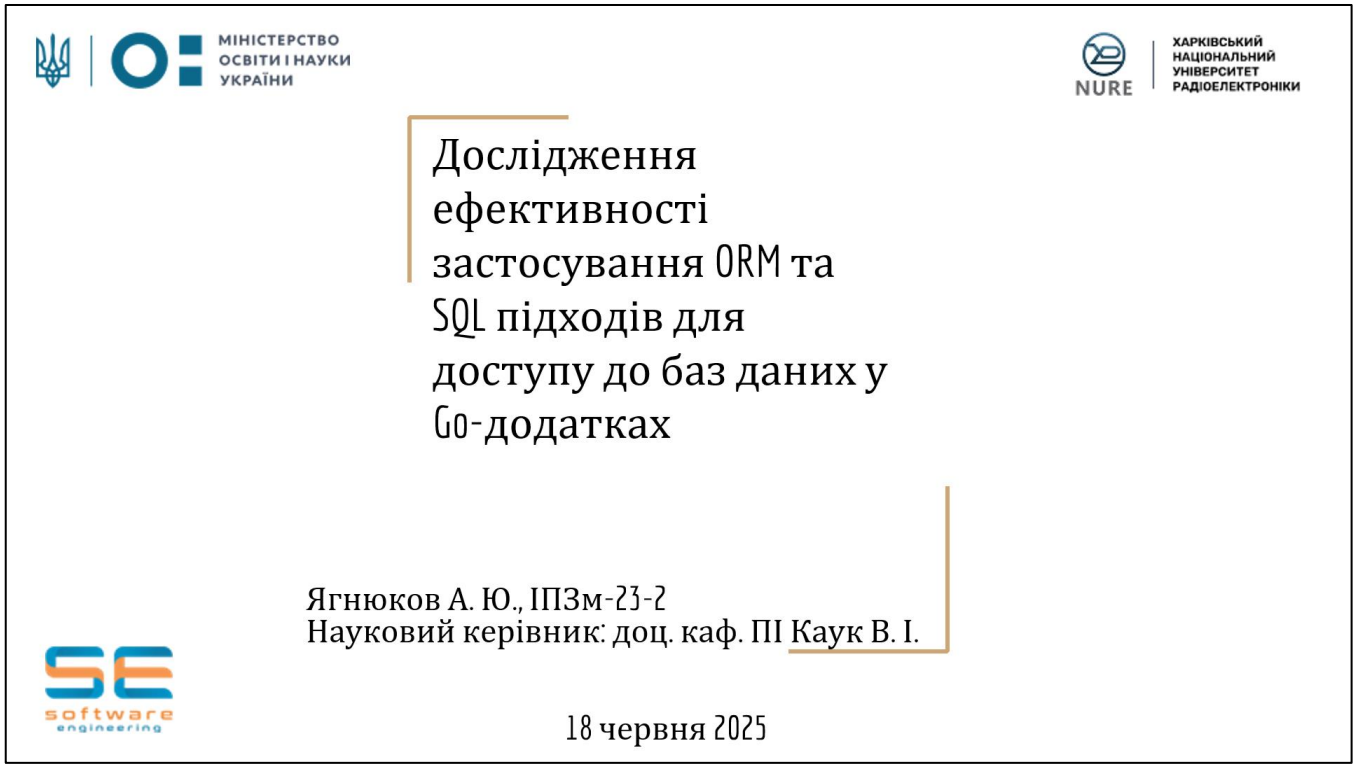
ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

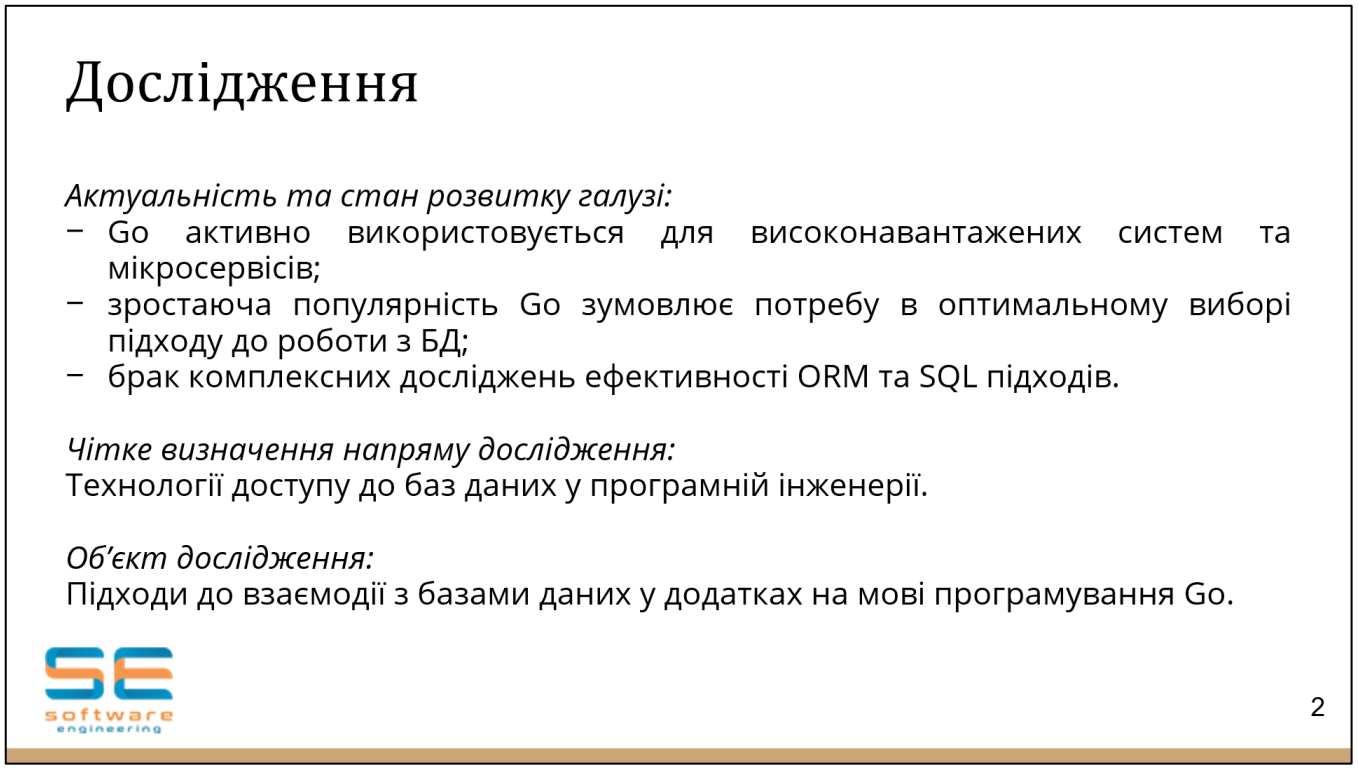
з домашньої бази даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

ДОДАТОК Б

СЛАЙДИ ПРЕЗЕНТАЦІЇ





Огляд літератури

Перелік основних джерел та теорій:

- Procaccianti G., Lago P., Diesveld W. «Energy Efficiency of ORM Approaches: an Empirical Evaluation»: емпіричне дослідження енергоефективності ORM-підходів;
- Colley D., Stanier C., Asaduzzaman M. «The Impact of Object-Relational Mapping Frameworks on Relational Query Performance»: вплив ORM-фреймворків на продуктивність;
- Chen T., Shang W., Jiang Z., Hassan A., Nasser M. N., Flora P. «Finding and Evaluating the Performance Impact of Redundant Data Access for Applications that are Developed Using Object-Relational Mapping Frameworks»: аналіз проблем надлишкового доступу до даних в ORM;
- Zmaranda D., Pop-Fele L.-L., Gyorodi C., Gyorodi R., Pecherle G. «Performance Comparison of CRUD Methods using NET Object Relational Mappers: A Case Study»: порівняння продуктивності CRUD-операцій.

Огляд літератури

Основні теоретичні положення:

- ORM прискорює розробку, але створює додаткові витрати ресурсів;
- SQL забезпечує повний контроль та стабільну продуктивність;
- зниження продуктивності ORM спричиняють надмірні звернення до даних.

Зазначення прогалин у наявних дослідженнях:

- недостатність метрик для повної оцінки підходів;
- відсутність складних запитів;
- брак рекомендацій щодо вибору та використання підходів.

Постановка задачі

Чітке формулювання проблеми:

Пошук ефективного підходу доступу до баз даних, що забезпечує оптимальну продуктивність Go-додатків.

Очікувані результати:

- побудова порівняльної моделі ORM-бібліотек на основі багатокритеріального аналізу;
- визначення ефективного підходу доступу до баз даних за результатами експериментального дослідження;
- розробка Go-додатку з однаковою функціональністю, який демонструє оптимальний підхід доступу до БД;
- обґрунтовані практичні рекомендації щодо вибору підходу для Go-програмних рішень.

Методологія

Використані методи дослідження:

- аналіз наукових публікацій: вивчення сучасних підходів до оцінювання ефективності ORM та SQL;
- багатокритеріальний аналіз: оцінювання ORM-бібліотек за низкою критеріїв для обґрунтованого відбору найперспективніших варіантів;
- експериментальне досліджування: проведення серії експериментів для порівняння ефективності та продуктивності підходів;
- систематизація даних: формування рекомендацій на основі отриманих результатів.

Зміст проведеного дослідження

Вхідні дані:

- 4 ORM-бібліотеки: GORM, XORM, Ent та SQLBoiler;
- офіційна документація бібліотек;
- статистичні дані з GitHub.

Зміст проведеного дослідження

Критерії:

- документація та навчальні матеріали;
- кількість підтримуваних СУБД;
- популярність на GitHub;
- легкість інтеграції та використання;
- підтримка спільноти та оновлення.

Послідовність:

1. збір та систематизація даних по кожній альтернативі;
2. визначення Парето-оптимальних рішень;
3. нормалізація критеріїв за шкалами;
4. визначення вагових коефіцієнтів;
5. розрахунок інтегральних оцінок.

Результати дослідження

Для GORM:

$$\frac{4}{22} \times 1 + \frac{5}{22} \times 0 + \frac{3}{22} \times 1 + \frac{6}{22} \times 1 + \frac{4}{22} \times 1 = 0{,}773$$

Для Ent:

$$\frac{4}{22} \times 1 + \frac{5}{22} \times 1 + \frac{3}{22} \times 0 + \frac{6}{22} \times 1 + \frac{4}{22} \times 1 = 0{,}864$$

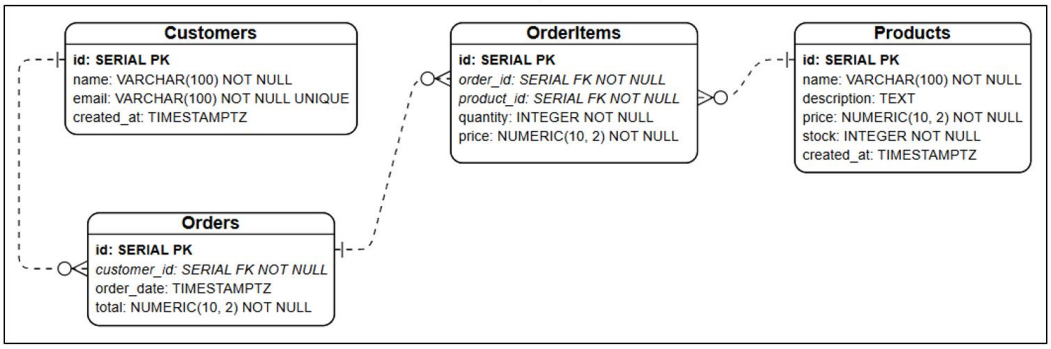
Архітектура додатку

Архітектура:

- Go-додаток з трьома модулями для предметної області електронної комерції:
  - модуль GORM;
  - модуль Ent;
  - модуль database/sql;
- БД (PostgreSQL) – зберігання клієнтів, продуктів, замовлень, позицій замовлень.

Архітектура додатку

ER-діаграма



Опис програмного забезпечення

Опис процесу розробки:

- розробка велася на основі результатів теоретичного дослідження: вибрані найефективніші ORM-бібліотеки;
- етапи розробки:
  - проєктування ER-моделі та схеми бази даних;
  - налаштування PostgreSQL у Docker-контейнері;
  - розробка спільних структур даних;
  - реалізація модулів з GORM, Ent та database/sql;
  - налаштування міграцій та наповнення тестовими даними.

Опис програмного забезпечення

Метрики:

- середня затримка (мс)

$$AvgLatency = \frac{1}{N}\sum_{i=1}^{N}\frac{t_i}{10^6};$$

- 95-й перцентиль затримки (мс)

$$P95Latency = \frac{t_{[0,95N]}}{10^6};$$

- пропускна здатність (оп/сек)

$$Throughput = \frac{N}{T_{total}};$$

- середнє споживання оперативної пам'яті (Мб)

$$AvgRAM = \frac{\Delta M}{N \times 1024^2};$$

- час пауз GC (мс)

$$GCPause = \frac{\Delta P}{10^6}.$$

Опис програмного забезпечення

Таблиця результатів виконання операції на створення нового продукту

Метрики / Альтернативи	GORM	Ent	database/sql
Середня затримка (мс)	2,5343	2,0303	2,4251
95-й перцентиль затримки (мс)	4,6652	2,6053	4,4141
Пропускна здатність (оп/сек)	394,5868	492,5464	412,3479
Середнє споживання ОЗП (Мб)	0,0068	0,0034	0,0006
Час пауз GC (мс)	2,2064	1,0053	0,0000

Таблиця результатів виконання операції на отримання інформації клієнта за ідентифікатором

Метрики / Альтернативи	GORM	Ent	database/sql
Середня затримка (мс)	0,3139	0,6479	0,6169
95-й перцентиль затримки (мс)	0,6498	0,9998	0,9981
Пропускна здатність (оп/сек)	3186,2296	1543,3845	1620,9478
Середнє споживання ОЗП (Мб)	0,0044	0,0040	0,0011
Час пауз GC (мс)	0,0000	0,5543	0,1128

Опис програмного забезпечення

Таблиця результатів виконання операції на оновлення ціни продукту за ідентифікатором

Метрики / Альтернативи	GORM	Ent	database/sql
Середня затримка (мс)	2,2002	3,1357	1,9903
95-й перцентиль затримки (мс)	3,1580	3,7838	2,5135
Пропускна здатність (оп/сек)	454,5036	318,9031	502,4480
Середнє споживання ОЗП (Мб)	0,0062	0,0052	0,0003
Час пауз GC (мс)	3,0859	0,1206	0,0000

Таблиця результатів виконання операції на видалення продукту за назвою

Метрики / Альтернативи	GORM	Ent	database/sql
Середня затримка (мс)	3,6118	3,3223	3,3006
95-й перцентиль затримки (мс)	4,3545	4,0859	4,0921
Пропускна здатність (оп/сек)	276,8694	300,9072	302,9795
Середнє споживання ОЗП (Мб)	0,0052	0,0019	0,0003
Час пауз GC (мс)	3,6996	0,8759	0,0000

Опис програмного забезпечення

Таблиця результатів виконання операції на створення замовлення з продуктами для конкретного клієнта за ідентифікатором (транзакція)

Метрики / Альтернативи	GORM	Ent	database/sql
Середня затримка (мс)	2,5915	3,2192	3,1404
95-й перцентиль затримки (мс)	3,2037	3,9317	3,7949
Пропускна здатність (оп/сек)	385,8746	310,6328	318,4332
Середнє споживання ОЗП (Мб)	0,0188	0,0073	0,0019
Час пауз GC (мс)	3,4119	0,2680	0,0000

Таблиця результатів виконання операції на отримання статистики по замовленнях та витратах клієнта за ідентифікатором (агрегація)

Метрики / Альтернативи	GORM	Ent	database/sql
Середня затримка (мс)	1,7731	2,2285	2,2091
95-й перцентиль затримки (мс)	2,4585	2,8527	2,8251
Пропускна здатність (оп/сек)	563,9747	448,7343	452,6629
Середнє споживання ОЗП (Мб)	0,0046	0,0008	0,0008
Час пауз GC (мс)	1,4302	0,8388	0,0000

Опис програмного забезпечення

Таблиця результатів виконання операції на отримання інформації про продажі топ продуктів (складний JOIN)

Метрики / Альтернативи	GORM	Ent	database/sql
Середня затримка (мс)	6,0444	6,1471	6,4545
95-й перцентиль затримки (мс)	6,8305	6,7861	7,3262
Пропускна здатність (оп/сек)	165,4425	162,6774	154,9314
Середнє споживання ОЗП (Мб)	0,0086	0,0013	0,0013
Час пауз GC (мс)	7,7937	0,5720	0,0000

Публікація результатів

1 Міжнародна науково-практична конференція «СУЧАСНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ MIT@AIS-2025»

Підсумки

Результати теоретичного та експериментального досліджень доводять різну ефективність SQL та ORM підходів, акцентуючи увагу на їхніх перевагах і недоліках у складних сценаріях, що не було висвітлено у попередніх роботах.

Реалізований Go-додаток поєднує модулі для GORM, Ent та database/sql, що забезпечує комплексне й об'єктивне порівняння підходів за розширеним набором метрик і заповнює прогалину щодо недостатності даних для повної оцінки.

Сформульовані практичні рекомендації дозволяють розробникам робити обґрунтований вибір ефективного інструменту для досягнення оптимальної продуктивності Go-додатків, виходячи з домінуючих типів операцій:

- database/sql для критичних за ресурсами операцій запису;
- Ent для ефективного створення даних та як збалансований підхід;
- GORM для складних операцій читання.

Дякую за увагу!

ДОДАТОК В

АПРОБАЦІЯ РЕЗУЛЬТАТІВ РОБОТИ

1 Міжнародна науково-практична конференція «СУЧАСНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ MIT@AIS-2025»

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ЗАСТОСУВАННЯ ORM ТА SQL ПІДХОДІВ ДЛЯ ДОСТУПУ ДО БАЗ ДАНИХ У GO-ДОДАТКАХ

А.Ю. Ягнюков[1*], В.І. Каук[1]

[1] *Харківський національний університет радіоелектроніки, пр. Науки, 14, м. Харків, 61166, Україна*

Ключові слова: бази даних; ефективність; Go; ORM; SQL

АНОТАЦІЯ

У тезах проаналізовано ефективність трьох методів доступу до реляційної бази даних у Go-додатках: виконання SQL-запитів із застосуванням пакету database/sql та дві ORM-бібліотеки (GORM, Ent). Для дослідження розроблено тестові додатки, що реалізують стандартні CRUD-операції та комплексні запити (транзакції, агрегації, приєднання таблиць) до PostgreSQL.
Експериментальне дослідження включало вимірювання двох ключових показників для кожного підходу: середній час виконання операцій та обсяг використаної оперативної пам'яті. Результати систематизовано у зведеній таблиці.
Експерименти показали, що доступ через database/sql забезпечує найменший час виконання операцій серед усіх підходів. Серед ORM-бібліотек Ent у багатьох випадках демонструє результати, близькі до database/sql, тоді як GORM зазвичай має довший час виконання та помітно більше споживає пам’яті.

ПЕРЕДУМОВА

У розробці програмного забезпечення реляційні бази даних та SQL залишаються основними технологіями управління даними. ORM-бібліотеки пропонують спрощені механізми взаємодії з базами даних, однак зазвичай супроводжуються додатковими накладними витратами обчислювальних ресурсів та пам'яті порівняно з використанням SQL-запитів [1].
У контексті розробки Go-додатків виникає питання знаходження оптимального балансу між ефективністю системи та програмування при організації доступу до бази даних.

МЕТА

Дослідити оптимальний підхід доступу до реляційних баз даних у Go-додатках шляхом порівняльного аналізу ефективності за критеріями швидкодії та використання оперативної пам'яті при виконанні базових CRUD-операцій і комплексних запитів.

МЕТОДИ

Проведено експерименти порівняльного аналізу трьох підходів у Golang: чистого SQL (пакет database/sql) та ORM-бібліотек (GORM і Ent). Розроблено тестові застосунки для виконання стандартних CRUD-операцій та комплексних запитів (транзакції, агрегації, приєднання) до PostgreSQL. Для кожного сценарію вимірювалися середній час виконання операції і обсяг спожитої оперативної пам’яті [2]. Результати зведено в єдиній таблиці.

РЕЗУЛЬТАТИ

Дані з таблиці показують, що для всіх CRUD-операцій та комплексних запитів доступ через database/sql забезпечує найменший час виконання. ORM-бібліотеки в цілому демонструють нижчу

ефективність: у більшості випадків GORM працює повільніше та споживає більше пам’яті, тоді як результати Ent наближені до database/sql. При цьому доступ через database/sql зазвичай потребує менше ресурсів, тоді як ORM-сценарії супроводжуються додатковими накладними витратами.

Таблиця 1. Результати ефективності підходів.

Операція	Час виконання запитів (мс)			Споживання оперативної пам’яті (Мб)		
	GORM	Ent	database/sql	GORM	Ent	database/sql
Створення	13,20	13,08	12,32	0,0153	0,0158	0,0110
Отримання	9,20	9,46	8,58	0,0174	0,0177	0,0221
Оновлення	16,81	20,52	16,32	0,0233	0,0140	0,0155
Видалення	20,69	16,89	18,25	0,0289	0,0157	0,0116
Транзакція	15,92	16,83	14,49	0,0300	0,0102	0,0121
Агрегація	10,23	14,54	9,32	0,0331	0,0139	0,0187
Приєднання	21,82	19,50	9,60	0,1634	0,0529	0,0128

Зокрема, операції створення у database/sql виконуються приблизно на 6–7 % швидше, ніж у GORM і Ent (12,32 мс проти 13,20 мс та 13,08 мс). Під час отримання даних database/sql також показує перевагу близько 7–10 % (8,58 мс проти 9,20 мс та 9,46 мс). Найбільша різниця спостерігається при операціях з приєднанням таблиць: database/sql витрачає 9,60 мс, що у більш ніж два рази менше, ніж час GORM (21,82 мс), і майже вдвічі менше, ніж час Ent (19,50 мс).

За обсягом пам’яті database/sql зазвичай також ефективніший: у простих операціях витрачається приблизно в 1,5–2,5 рази менше пам’яті, ніж GORM (наприклад, при видаленні записів database/sql споживає 0,0116 Мб проти 0,0289 Мб у GORM). При виконанні складного запиту з приєднанням таблиць database/sql витрачає 0,0128 Мб, тоді як GORM – 0,1634 Мб (у понад 12 разів більше), а Ent – 0,0529 Мб. Ent у більшості операцій споживає помітно менше пам’яті, ніж GORM (приблизно в 2 рази менше для більшості сценаріїв).

ВИСНОВКИ

За результатами дослідження встановлено, що використання database/sql дає найкращі показники швидкодії та зазвичай найменше споживання оперативної пам’яті у більшості сценаріїв. ORM-бібліотеки підвищують зручність програмування, але призводять до зниження продуктивності: Ent у багатьох випадках демонструє результати, наближені до database/sql, тоді як GORM має найнижчі показники продуктивності.

ЛІТЕРАТУРА

1 D. Colley, C. Stanier, M. Asaduzzaman, The Impact of Object-Relational Mapping Frameworks on Relational Query Performance, in: 2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE), 2018, pp. 47–52. doi:10.1109/ICCECOME.2018.8659222.

2 D. Zmaranda, L.-L. Pop-Fele, C. Gyorodi, R. Gyorodi, G. Pecherle, Performance Comparison of CRUD Methods using NET Object Relational Mappers: A Case Study, International Journal of Advanced Computer Science and Applications 11 (2020). doi:10.14569/ijacsa.2020.0110107.

ДОДАТОК Г

ЕКСПЕРТНИЙ ВИСНОВОК РЕЗУЛЬТАТІВ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА ВІДПОВІДНІСТЬ ОФОРМЛЕННЯ ВИМОГАМ ДСТУ 3008:2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-23-2 (група)

Андрій ЯГНЮКОВ
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні познаки	
	7.15 Додатки	

Експерт _____ (підпис) Вадим НЕЧВОЛОД (прізвище, ініціали)

Зауважень з оформлення немає.
10.06.2025

ДОДАТОК Д

КОД ДЛЯ ЗБОРУ ТА ОБЧИСЛЕННЯ МЕТРИК

```
1. package utils
2.
3. import (
4.    "fmt"
5.    "log"
6.    "math"
7.    "runtime"
8.    "slices"
9.    "time"
10.   )
11.
12.   type Result struct {
13.        Operation  string
14.        AvgLatency float64
15.        P95Latency float64
16.        Throughput float64
17.        AvgRAM     float64
18.        GCPause    float64
19.   }
20.
21.   type Operation interface {
22.        Name() string
23.        Execute(int) error
24.   }
25.
26.   func Run(op Operation, iterations int) Result {
27.        times := make([]time.Duration, iterations)
28.        var memStart, memEnd runtime.MemStats
29.
30.        runtime.GC()
31.        runtime.ReadMemStats(&memStart)
32.        gcPauseStart := memStart.PauseTotalNs
33.
34.        for i := 0; i < iterations; i++ {
35.             start := time.Now()
36.
37.             if err := op.Execute(i); err != nil {
38.                  log.Fatalf("failed to execute operation \"%s\":
   %v", op.Name(), err)
39.             }
40.
41.             times[i] = time.Since(start)
42.        }
43.
44.        runtime.ReadMemStats(&memEnd)
45.        gcPauseEnd := memEnd.PauseTotalNs
46.
47.        var totalTime time.Duration
48.        for _, t := range times {
```

```
49.             totalTime += t
50.         }
51.
52.         avgLatency := float64(totalTime.Nanoseconds()) /
   float64(iterations) / 1e6
53.
54.         slices.Sort(times)
55.         idx := max(int(math.Ceil(0.95*float64(iterations)))-1, 0)
56.         p95Latency := float64(times[idx].Nanoseconds()) / 1e6
57.
58.         throughput := float64(iterations) / totalTime.Seconds()
59.
60.         avgRAM := float64(memEnd.TotalAlloc-memStart.TotalAlloc) /
   float64(iterations) / (1024 * 1024)
61.
62.         gcPause := float64(gcPauseEnd-gcPauseStart) / 1e6
63.
64.         return Result{
65.             Operation:  op.Name(),
66.             AvgLatency: avgLatency,
67.             P95Latency: p95Latency,
68.             Throughput: throughput,
69.             AvgRAM:     avgRAM,
70.             GCPause:    gcPause,
71.         }
72.   }
73.
74.   func PrintResult(operations []Operation, iterations int) {
75.         fmt.Printf(
76.             "%-60s %-20s %-20s %-20s %-20s %-20s\n",
77.             "Operation", "Avg Latency (ms)", "P95 Latency (ms)",
   "Throughput (ops/s)", "Avg RAM (MB)", "GC Pause (ms)",
78.         )
79.         for _, operation := range operations {
80.             result := Run(operation, iterations)
81.             fmt.Printf(
82.                 "%-60s %-20.4f %-20.4f %-20.4f %-20.4f %-20.4f\n",
83.                 result.Operation,
84.                 result.AvgLatency,
85.                 result.P95Latency,
86.                 result.Throughput,
87.                 result.AvgRAM,
88.                 result.GCPause,
89.             )
90.         }
91.   }
```