DOI https://doi.org/10.32782/2220-8674-2025-15-1-27
УДК 378.091.33:004.421:004.421.2

Ю. О. Сіциліцин[1,2], Ph. D. ORCID: 0000-0002-3888-5575
М. В. Шлофаст[1], здобувач
[1] Таврійський державний агротехнологічний університет імені Дмитра Моторного
[2] Мелітопольський державний педагогічний університет імені Богдана Хмельницького
e-mail: yurii.sitsilitsin@tsatu.edu.ua

ПОРІВНЯЛЬНИЙ АНАЛІЗ ЕФЕКТИВНОСТІ ОБРОБЛЕННЯ ВЕЛИКИХ ТАБЛИЧНИХ ДАНИХ: EXCEL/VBA ПРОТИ PYTHON

Анотація. У статті проведено порівняльний аналіз ефективності оброблення великих табличних даних із застосуванням Excel/VBA та Python. Дослідження базується на експериментальному моделюванні умов оброблення 500 000 запитів, що імітують реальні сценарії роботи інформаційних систем. Розглянуто ключові параметри: час виконання, продуктивність, споживання пам'яті, масштабованість та легкість підтримки коду. Експериментальні результати демонструють, що рішення на Python дозволяє обробляти дані за 13.55 секунд із споживанням 120–150 МБ оперативної пам'яті, тоді як Excel/VBA потребує 188 секунд і використовує 250–300 МБ. Отримані дані свідчать про значні переваги сучасних технологій, що сприяють оптимізації систем оброблення інформації та підвищенню ефективності робочих процесів у реальному часі. Результати дослідження можуть бути використані для розроблення інноваційних інформаційних платформ, які відповідають високим вимогам продуктивності та масштабованості, забезпечуючи стабільну роботу аналітичних центрів і підприємств у сучасному конкурентному середовищі.

Ключові слова: Excel, VBA, Python, оброблення даних, продуктивність, пам'ять, maintainability.

Постановка проблеми. Сучасні інформаційні системи працюють із щораз вищими обсягами табличних даних, що охоплюють численні сфери: від фінансової звітності та клієнтської бази до наукових досліджень і статистичних аналізів. При цьому традиційні засоби, зокрема Excel із вбудованою мовою VBA, незважаючи на їх популярність і зручність для невеликих обсягів даних, стикаються з критичними обмеженнями під час роботи з мільйонними наборами записів. Основні проблеми включають повільну швидкість виконання обчислень, високий рівень споживання оперативної пам'яті та складність підтримки коду, що суттєво обмежує їх застосування в сучасних масштабованих системах.

З іншого боку, Python набув значної популярності завдяки своїй гнучкості та широкій екосистемі бібліотек (pandas, numpy), що забезпечують високу продуктивність під час оброблення великих даних. Ці засоби дозволяють не лише зменшити час обчислень, але й підвищити зручність адаптації та підтримки програмних рішень завдяки модульній архітектурі та зрозумілому синтаксису.

Проте існує недостатньо комплексних досліджень, які систематично порівнюють традиційний підхід Excel/VBA та сучасний підхід на базі Python за ключовими критеріями: час виконання скриптів, споживання пам'яті та легкість підтримки коду. Аналіз останніх публікацій показує, що більшість досліджень зосереджені на окремих аспектах оптимізації або розробленні специфічних алгоритмів для однієї з технологій, але комплексне порівняння залишається невирішеним питанням.

Таким чином, актуальність дослідження полягає у визначенні переваг та обмежень кожного з підходів у контексті оброблення великих табличних даних, що дозволить сформувати прак-

тичні рекомендації для розробників інформаційних систем. Визначення оптимального рішення має стратегічне значення для підвищення продуктивності, економії ресурсів та спрощення підтримки програмних продуктів у сучасних умовах високих обчислювальних навантажень.

Аналіз останніх досліджень. Оброблення великих табличних даних є актуальною проблемою, що привертає увагу як дослідників, так і практиків у галузі інформаційних технологій. У літературі дослідження зосереджені на аналізі продуктивності різних технологій, серед яких традиційний Excel/VBA та сучасні платформи, зокрема Python з його бібліотеками (pandas, numpy). Excel разом із VBA протягом десятиліть залишається основним інструментом для аналізу та візуалізації даних у бізнес-середовищі. Автори підручника з аналізу даних [1] демонструють, що під час оброблення невеликих та середніх наборів даних Excel забезпечує зручний інтерфейс та простоту використання завдяки інтегрованим можливостям. Однак низка робіт підкреслює, що в разі збільшення обсягів даних виникають значні обмеження. Наприклад, низька швидкість виконання циклічних обчислень та обмежені можливості оптимізації алгоритмів у VBA обмежують застосування цієї технології у випадках, коли оброблення даних перевищує сотні тисяч записів. Крім того, розширення функціональності VBA часто призводить до зростання складності коду, що ускладнює його підтримку та модифікацію. У другій половині останнього десятиліття Python став популярною альтернативою завдяки відкритості коду, активній спільноті та розвиненій екосистемі бібліотек для аналізу даних. Дослідження [2] Lee et al. відзначають, що використання бібліотек pandas та numpy дозволяє значно скоротити час оброблення даних завдяки векторизованим операціям та оптимізованим алгоритмам. Багато сучасних робіт підтверджують, що Python демонструє кращі показники в обробленні великих обсягів даних, забезпечуючи більш ефективне управління пам'яттю та масштабованість обчислень. Наприклад, порівняльні експерименти, проведені у [3] Zumstein, свідчать про те, що Python здатен обробляти дані, що перевищують обсяги, з якими ефективно працює Excel/VBA, без суттєвого зниження продуктивності. Існує низка публікацій, присвячених порівнянню продуктивності різних підходів до оброблення даних. Зокрема, дослідження орієнтовані на вимірювання часу виконання скриптів та споживання пам'яті під час роботи з великими наборами даних. Такі методики включають профілювання з використанням спеціалізованих інструментів, як-от memory_profiler для Python, що дозволяють точно оцінити ресурси, які споживаються у виконанні алгоритмів. У деяких працях підкреслюється, що модульна архітектура Python-коду сприяє легшій підтримці та адаптації програмних рішень, що є важливим аспектом для довгострокового впровадження в комерційних системах. Незважаючи на переваги Python, окремі дослідження вказують на необхідність урахування специфіки завдань. Наприклад, для невеликих обчислювальних задач Excel/VBA може бути більш доцільним через простоту використання та відсутність потреби в додаткових знаннях програмування [4]. Проте зростання обсягів даних та щораз вищі вимоги до швидкодії роблять Python більш привабливим варіантом для сучасних інформаційних систем. Отже, аналіз досліджень свідчить про те, що вибір технології має базуватися на конкретних завданнях і обсягах даних, а також ураховувати перспективи подальшого масштабування та інтеграції з іншими системами. Загалом, огляд літератури показує, що, хоча Excel/VBA залишається популярним інструментом для рутинних обчислень і аналізу даних, його обмеження стають очевидними під час роботи з великими наборами даних. Python завдяки своїй масштабованості та потужним бібліотекам демонструє значні переваги в ефективності оброблення, що підтверджується як експериментальними дослідженнями, так і теоретичними розрахунками. Подальші дослідження спрямовані на оптимізацію алгоритмів, розроблення гібридних рішень та інтеграцію сучасних технологій у традиційні інформаційні системи.

Формулювання мети статті. Метою статті є проведення порівняльного аналізу ефективності оброблення великих табличних даних за допомогою Excel/VBA та Python.

Основна частина. Розглянемо більш детально параметри тестових даних, інструменти профілювання продуктивності та споживання пам'яті, а також критерії оцінювання. Для експериментів будемо генерувати набори даних від 500 000 до 1 000 000 записів, що відповідає реальним умовам оброблення великих таблиць. Дані мають містити поля «Дисципліна», «ПІБ», «Група» та «Семестр». Дані створимо із застосуванням фіксованих списків дисциплін, імен, прізвищ та груп, що забезпечує однорідний та рівномірний розподіл значень, а використання генератора випадкових чисел дозволяє імітувати повторюваність типових реальних сценаріїв. Для досягнення реплікації результатів важливо фіксувати насіння генератора, що забезпечує можливість відтворення експериментів. Таким чином, структура даних спрямована на тестування алгоритмів агрегації та сортування в умовах рівномірного розподілу, що відповідає вимогам сучасних інформаційних систем. Для Python застосуємо різні інструменти профілювання продуктивності та споживання пам'яті. Наприклад, для вимірювання часу виконання окремих блоків коду використовуватимемо бібліотеку timeit, що дозволяє з високою точністю оцінити швидкодію, а вбудований профайлер cProfile допоможе визначити, які частини коду споживають найбільше ресурсів. Детальний моніторинг використання пам'яті здійснюватиметься за допомогою memory_profiler, який дозволяє відстежувати споживання пам'яті на рівні рядка, а бібліотека psutil надасть інформацію про системні ресурси, зокрема CPU та оперативну пам'ять, у режимі реального часу. Для Excel/VBA застосуємо вбудовану функцію Timer для вимірювання часу виконання макросів, а також скористаємося зовнішніми засобами, як-от Windows Performance Monitor, для відстеження використання системних ресурсів. Крім того, логування часу виконання через запис даних у текстовий файл дозволить більш детально аналізувати продуктивність і споживання пам'яті. Критерії оцінювання розглядаються як комплексна характеристика ефективності алгоритмів оброблення даних. Загальний час оброблення є критичним показником, оскільки навіть незначні зменшення можуть суттєво підвищити продуктивність систем у роботі з великими наборами записів [1]. Аналіз споживання пам'яті дозволяє оцінити оптимізацію використання ресурсів, що має вирішальне значення в масштабуванні обчислень, як зазначено в роботах [2] та [3]. Легкість підтримки коду та його структурна модульність є важливими для забезпечення довгострокової адаптивності та можливості розширення функціональності, що також підтверджується сучасними публікаціями [5]. Окрім цього, масштабованість алгоритмів і здатність до відтворення експериментальних результатів забезпечують стабільність системи в різних умовах навантаження. Таким чином, вибір даних критеріїв обґрунтовано їх впливом на загальну ефективність і стійкість інформаційних систем, що знаходять підтвердження у відповідних наукових дослідженнях. У результаті аналізу було виділено такі критерії:

1. Час виконання
2. Споживання пам'яті
3. Продуктивність алгоритмів
4. Легкість підтримки коду
5. Масштабованість
6. Відтворюваність результатів

Створимо математичну модель, яка буде описувати оптимізацію оброблення даних для обох підходів.

Для Excel/VBA можна розглядати модель часу оброблення даних у вигляді:

$$T_{VBA} = T_{read} + n \cdot T_{loop} + T_{write}, \quad (1)$$

де T_{read} – час на читання даних, n – кількість записів, T_{loop} – середній час оброблення одного запису (циклічна оброблення), а T_{write} – час на запис результатів.

У Python із застосуванням бібліотеки pandas оптимізацію можна описати такою моделлю:

$$T_{Python} = T_{read} + T_{vectorized} \cdot (n) + T_{write} \quad , (2)$$

де $T_{vectorized}(n)$ – приблизно дорівнює $C\,n$ C n з дуже малою константою C завдяки векторизованим операціям, що виконуються на рівні С-коду.

Для оцінювання переваг використання одного підходу над іншим вводять коефіцієнт прискорення:

$$S = \frac{T_{VBA}}{T_{Python}}, \quad (3)$$

що дозволяє кількісно оцінити зниження часу оброблення.

Щодо споживання пам'яті, можна використовувати модель:

$$M = n \cdot m_{record} + M_{overhead}, \quad (4)$$

де m_{record} – пам'ять, яку займає один запис, а $M_{overhead}$ – системні витрати пам'яті. Завдяки оптимізованим структурам даних у pandas значення m_{record} може бути значно меншим, що сприяє ефективнішому використанню оперативної пам'яті.

Ці моделі дозволяють кількісно обґрунтувати оптимізацію оброблення даних для обох підходів та підтверджують результати, наведені в наукових дослідженнях [1; 2; 3].

Для формування тестових даних було розроблено спеціальний скрипт, що згенерував файл у форматі.xlsx із 500 000 записів (рис. 1). У створеній таблиці містяться чотири основні колонки: «Назва дисципліни», «ПІБ», «Група» та «Семестр». Така структура дозволяє імітувати умови реальної освітньої бази, де кожен рядок відповідає поєднанню певної дисципліни, конкретного студента, належності до групи та семестру.

	A	B	C	D	E	F
1	**Назва дисципліни**	**ПІБ**	**Група**	**Семестр**		
2	Англійська	Мельник Іван	Група A	2		
3	Фізика	Бондаренко Дмитро	Група B	1		
4	Історія	Мельник Олег	Група E	1		
5	Хімія	Шевченко Андрій	Група B	2		
6	Англійська	Мельник Олег	Група E	2		
7	Історія	Мельник Петро	Група D	1		
8	Історія	Мельник Дмитро	Група C	1		
9	Фізика	Марченко Іван	Група E	2		
10	Математика	Бондаренко Іван	Група C	1		
11	Англійська	Шевченко Іван	Група A	2		
12	Фізика	Шевченко Дмитро	Група A	2		
13	Англійська	Ткаченко Олег	Група E	1		
14	Біологія	Ткаченко Петро	Група E	1		
15	Математика	Шевченко Дмитро	Група C	1		
16	Хімія	Мельник Тарас	Група B	1		
17	Біологія	Бондаренко Андрій	Група A	1		
18	Історія	Бондаренко Дмитро	Група A	1		
19	Історія	Марченко Олег	Група E	2		
20	Хімія	Бондаренко Іван	Група A	2		
21	Математика	Ткаченко Тарас	Група A	1		
22	Фізика	Шевченко Дмитро	Група A	2		
23	Математика	Бондаренко Дмитро	Група E	2		
24	Математика	Мельник Дмитро	Група E	2		

Рис. 1. Згенеровані тестові дані

Для заповнення полів використовувалися випадкові комбінації імен і прізвищ із заздалегідь заданого списку, а також розподіл за групами та семестрами, що забезпечує достатній рівень

варіативності. Генерація такого обсягу даних (пів мільйона записів) дає змогу виявити особливості та недоліки різних підходів до оброблення інформації, адже значні масиви дають більш наочний ефект під час порівняння швидкодії та використання ресурсів.

Файл із 500 000 записів є досить великим для того, щоб підкреслити відмінності між алгоритмами та підходами, а також виявити потенційні «вузькі місця» в процесі зчитування, групування чи сортування даних. Завдяки цьому можна отримати об'єктивні результати про продуктивність різних систем і методів оброблення табличної інформації, а також перевірити, як вони поводяться в реалістичних сценаріях.

Далі представлено результати експериментального оцінювання ефективності оброблення великих табличних даних за допомогою двох підходів: Python та VBA. Метою експерименту було порівняти час виконання, продуктивність, споживання оперативної пам'яті, легкість підтримки коду та масштабованість вибраних технологій на основі оброблення 500 000 запитів.

Для дослідження було розроблено два окремих рішення. Одне з них реалізоване мовою Python із використанням бібліотек pandas та openpyxl, що дозволяють швидко зчитувати, обробляти та зберігати великі обсяги даних у форматі XLSX. Інше рішення – реалізація на VBA для Microsoft Excel – використовує стандартні засоби VBA, як-от об'єкти Dictionary та цикли для групування, сортування та запису даних. Вхідний набір даних містить 500 000 записів та описаний вище.

Для проведення експерименту було використано комп'ютер із такими характеристиками:

– Процесор: Intel Core i7 (8 ядер, 12 потоків);
– Оперативна пам'ять: 40 ГБ DDR4;
– Операційна система: Windows 11 Pro;
– Версія Python: 3.9 з бібліотеками pandas та openpyxl;
– Версія Excel: Microsoft Excel 365 з підтримкою VBA.

Обидва рішення запускалися на одному й тому самому комп'ютері, що дозволяло порівняти їх продуктивність в однакових умовах. Для вимірювання часу виконання у Python використовувалися модулі time та timeit, а у VBA – вбудована функція Timer. Крім того, для оцінювання споживання оперативної пам'яті використовувалися зовнішні інструменти, зокрема Windows Task Manager та Resource Monitor, що дозволило зафіксувати пікові значення використання пам'яті процесами Python та Excel.

Результати вимірювання часу виконання та інших показників представлено в таблиці 1.

Таблиця 1

Результати експерименту

Показник	Python	VBA	Примітки
Час виконання	13,55 с	188 с	Загальний час оброблення 500 000 запитів
Продуктивність (запитів/с)	≈ 36 900 запитів/с	≈ 2 659 запитів/с	Розраховано як 500 000 / час виконання
Споживання пам'яті	120–150 МБ	250–300 МБ	Оцінка за типове навантаження
Легкість підтримки коду	9/10	5/10	Сучасні стандарти розроблення та читабельність коду
Масштабованість	Висока	Низька	Підходить для роботи з великими наборами даних

Як видно з таблиці, рішення на Python забезпечує значно кращі показники продуктивності, що дозволяє обробляти до 36 900 запитів за секунду порівняно з VBA, яке здатне обробити лише 2659 запитів за секунду. Крім того, Python використовує менше оперативної пам'яті, що є критичним у масштабуванні системи для роботи з ще більшими наборами даних. Щодо

підтримки коду Python також набуває перевагу завдяки сучасним стандартам програмування, високій читабельності коду та широкій екосистемі бібліотек, що дозволяє швидко адаптувати рішення до змінних умов.

Зважаючи на проведені вимірювання, зробимо висновки. Рішення на Python виконало оброблення 500 000 запитів за 13.55 секунд, тоді як VBA потребувало 188 секунд. Така різниця в часі оброблення свідчить про значну перевагу використання векторизованих операцій та оптимізованих бібліотек у Python. Більш висока швидкість обчислень дозволяє використовувати Python у системах, де критичним є час реакції. Показник запитів за секунду для Python становить близько 36 900, що значно перевищує VBA. Цей критерій має особливе значення під час оброблення даних у режимі реального часу або виконання складних аналітичних задач, де навіть невеликі затримки можуть впливати на загальну продуктивність системи. Експеримент показав, що Python споживає менше оперативної пам'яті (120–150 МБ) порівняно з VBA (250–300 МБ). Ефективне управління пам'яттю є важливим фактором за оброблення великих обсягів даних, оскільки зниження витрат пам'яті сприяє кращій масштабованості та зменшує ризик зупинки процесів через перевантаження ресурсів. З точки зору розроблення та подальшої підтримки рішення на Python отримало оцінку 9 із 10, що свідчить про його зручність для розробників, легкість модифікації та розширення функціональності. У порівнянні, код на VBA має складнішу структуру та меншу гнучкість, що відображається в оцінці 5 із 10. Python продемонстрував високу масштабованість, що дозволяє ефективно працювати з наборами даних, що перевищують 500 000 записів. Це особливо важливо для сучасних інформаційних систем, де обсяги даних постійно зростають, а вимоги до продуктивності стають дедалі суворішими.

Результати експерименту однозначно вказують на те, що використання Python є більш ефективним підходом для оброблення великих табличних даних у порівнянні з традиційним рішенням на VBA. Основна перевага Python пов'язана із застосуванням високопродуктивних бібліотек, які використовують векторизовані операції, що значно знижують час виконання алгоритмів. Крім того, сучасна екосистема Python забезпечує оптимальне управління пам'яттю та можливість легкого розширення функціональності за рахунок додаткових модулів.

З іншого боку, рішення на VBA демонструє суттєві обмеження, зокрема повільний час виконання та високе споживання ресурсів, що робить його менш придатним для сценаріїв із високими навантаженнями. Хоча VBA може бути зручним для розроблення невеликих та середніх рішень, його використання в умовах оброблення мільйонних наборів даних є неефективним.

Важливим аспектом експерименту є також оцінювання легкості підтримки коду. Сучасні програмні засоби, зокрема Python, дозволяють розробляти модульний, легко масштабований та підтримуваний код, що є ключовою вимогою під час розроблення інформаційних систем, які повинні швидко адаптуватися до змін ринку і технологій. Результати експерименту підтверджують, що інвестиції в розроблення рішень на Python окуповуються через значне підвищення продуктивності та зниження витрат ресурсів.

Висновки. Проведений експеримент з оброблення 500 000 запитів показав, що Python є значно ефективнішим засобом оброблення великих обсягів даних порівняно з VBA. Зокрема, час виконання для Python становив 13.55 секунд проти 188 секунд для VBA, що дає приріст продуктивності приблизно у 14 разів. Крім того, ефективне управління пам'яттю в Python дозволило знизити споживання оперативної пам'яті до 120–150 МБ, тоді як VBA спожив близько 250–300 МБ. Ці результати свідчать про можливість використання Python для масштабованих систем, де критично важлива швидкість оброблення та оптимізація ресурсів.

У таблиці 1 наведено основні показники, що дозволяють порівняти продуктивність обох підходів. Розглянуті результати експерименту підтверджують, що для завдань з оброблення

великих табличних даних доцільніше використовувати рішення на базі Python, оскільки воно забезпечує значно вищу швидкість, менше споживання пам'яті та кращу підтримку коду.

Таким чином, результати експерименту відкривають нові можливості для оптимізації процесів оброблення даних у сучасних інформаційних системах. Подальші дослідження можуть зосередитися на розробленні гібридних рішень, які поєднують зручність традиційних засобів Excel/VBA з високою продуктивністю Python, що дозволить розширити спектр застосувань та підвищити ефективність роботи систем в умовах усе вищих обсягів даних.

Список використаних джерел

1. Khadka B. Data analysis theory and practice. Case: Python and Excel Tools. Kokkola : Centria university of applied sciences, 2019. 97 p.
2. Lee J., Chang J., Kao L., Lee C. Essentials of Excel VBA, Python, and R. Bern: Springer, 2023. 696 p.
3. Zumstein F. Python for Excel. A modern environment for automation and data analysis. Sebastopol: O'Reilly Media, 2021. 275 p.
4. Mergel D. Physics with Excel and Python. Bern: Springer, 2023. 482 p.
5. Kelvin K., Wahab W., Wulandari M. Computer Resource Utilization Analysis for Microsoft Excel and Python in Data Processing. *Engineering, MAthematics and Computer Science,* 2024. Vol. 6 No. 2. DOI: https://doi.org/10.21512/emacsjournal.v6i2.11736

Стаття надійшла до редакції 18.03.2024 р.

Y. Sitsylitsyn [1, 2], M. Shelofast [1]
[1]Dmytro Motornyi Tavria State Agrotechnological University
[2] Bogdan Khmelnitsky Melitopol State Pedagogical University

COMPARATIVE ANALYSIS OF THE EFFICIENCY OF PROCESSING LARGE TABULAR DATA: EXCEL/VBA VS PYTHON

Summary

The article presents a comparative analysis of the efficiency of processing large tabular data using Excel/VBA and Python. The study is based on an experimental simulation of processing 500,000 queries that mimic real-world information system scenarios. Key parameters examined include execution time, performance, memory consumption, scalability, and code maintainability. The experimental results demonstrate that the Python solution can process data in 13.55 seconds with a memory consumption of 120–150 MB, whereas Excel/VBA requires 188 seconds and uses 250–300 MB. The data obtained indicate significant advantages of modern technologies that contribute to the optimization of information processing systems and enhance the efficiency of operational processes in real time. The findings of this study can be used for the development of innovative information platforms that meet high performance and scalability requirements, ensuring the stable operation of analytical centers and enterprises in today's competitive environment.

***Keywords*:** Excel, VBA, Python, data processing, performance, memory, maintainability.