УДК 004.65 **doi: 10.26906/SUNZ.2024.2.127**

С. В. Мінухін[1, 2], М. О. Башкіров[1]

[1] Харківський національний університет радіоелектроніки, Харків, Україна
[2] Харківський національний економічний університет імені С. Кузнеця, Харків, Україна

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ МЕТОДІВ ГЕНЕРАЦІЇ ТЕСТОВИХ ДАНИХ В РЕЛЯЦІЙНИХ БАЗАХ ДАНИХ

Анотація. У статті проведено порівняльний аналіз трьох основних методів генерації тестових даних для реляційних баз даних: циклічного, на основі Common Table Expressions (CTE) та з використанням тимчасових таблиць. Експериментальне тестування здійснено в середовищі MS SQL Server на прикладі таблиць бази даних торговельної компанії. Проведено порівняння за ключовими метриками ефективності генерації - часу генерації, використання системних ресурсів та масштабованістю об'ємів таблиць тестової бази даних. Результати дослідження обґрунтували ефективність методу рекурсивних CTE для різних обсягів і структур даних. Подано рекомендації щодо вибору методів генерації тестових наборів відповідно до вимог проектів БД, зокрема при використанні сервісів роботи з реляційними БД на хмарних платформах.

Ключові слова: база даних, Microsoft SQL Server, тестування, генерація даних, T-SQL, оптимізація, рекурсивні запити, тимчасові таблиці, генерація з циклами.

Вступ

Генерація тестових даних, зокрема, великих об'ємів, є важливим та актуальним завданням у сучасному проектуванні та розробленні баз даних. Зі зростанням потужності обчислювальних систем та обсягів даних, з якими вони працюють, потреба в репрезентативних тестових наборах стрімко зростає.

Генерування великих масивів тестових даних дозволяє всебічно перевірити продуктивність та масштабованість системи бази даних, оцінити її стійкість до високих навантажень, а також порівняти ефективність різних методів оброблення та зберігання [1]. Без наявності репрезентативних тестових наборів, які можуть бути отримані оперативним шляхом та за існуючими моделями БД, складно забезпечити стабільну та безперебійну роботу системи в реальних умовах експлуатації великих обсягів даних. Однією з ключових проблем при цьому є вибір оптимальної стратегії (методів) генерування необхідних обсягів тестових даних, яка забезпечувала б прийнятну швидкість (час) та ресурсоефективність задіяних ресурсів. Натепер існує декілька методів автоматизованої генерації тестових даних для баз даних із використанням мови T-SQL, зокрема - це методи на основі рекурсивних запитів, тимчасових таблиць, процедур та інш. [2]. Проте комплексний порівняльний аналіз їх ефективності та оптимальності застосування в різних сценаріях на сьогодні є недостатнім.

Метою даної роботи є дослідження та порівняльний аналіз трьох основних методів генерації тестових даних: на основі циклів, Common Table Expressions (CTE) та тимчасових таблиць в середовищі СУБД MS SQL Server. Експериментальне тестування дозволить кількісно оцінити їх ефективність за ключовими метриками продуктивності та ресурсоємності для різних сценаріїв та обсягів баз даних.

На основі отриманих результатів буде зроблено обґрунтовані висновки щодо ефективності застосування кожного з методів генерації тестових даних для конкретних задач тестування та навантаження баз даних. Будуть сформульовані практичні рекомендації щодо ефективного використання проаналізованих методів на прикладах таблиць БД з різними об'ємами даних.

Аналіз проблеми генерації даних та існуючих методів

Генерація тестових даних великих обсягів для баз даних є складним завданням, що визначається наступним.

1. Важливо забезпечити репрезентативність та різноманітність даних [2]. Тестові дані мають бути максимально наближені до реальних, щоб адекватно імітувати робочі навантаження [3]. При цьому важливо уникати шаблонності, оскільки це може знизити об'єктивність результатів тестування.

2. Важливо забезпечити унікальність кожного запису та значень ключових полів. Дублювання даних може призвести до помилок та неадекватного тестування системи.

3. Важливим аспектом є швидкість (час) та масштабованість генерації. Процес має бути ефективним навіть для дуже великих обсягів даних.

Існує декілька методів використання готових генераторів, генерація через API бібліотек, експорт та модифікація реальних даних, використання скриптів СУБД [3]. Останній підхід з використанням скриптів СУБД є найбільш гнучким та ефективним для генерації великих обсягів тестових даних. У даній статті розглядаються три основні методи: генерація циклами [4], генерація за допомогою CTE [5] та генерація тимчасовими таблицями [5]. Вони дозволяють гнучко налаштовувати логіку, мають просту реалізацію та добре масштабуються.

Отже, саме поєднання простоти, гнучкості та масштабованості визначило цикли, CTE та тимчасові таблиці оптимальним вибором методів для дослідження завдання генерації тестових даних у даному дослідженні. Результати порівняльного аналізу дозволять виявити переваги та недоліки кожного підходу для різних сценаріїв їх використання. Це дозволить обрати оптимальний метод генерації даних для конкретних умов використання бази даних.

Проектування бази даних

Проектування оптимальної схеми БД − важливий етап розробки для забезпечення ефективного зберігання та швидкого доступу до даних. Для формування тестових завдань була обрана модель БД, яка наведена у роботі [6]. База даних має забезпечувати зберігання інформації про продукти, категорії, бренди, клієнтів, замовлення та працівників для автоматизації операційної діяльності та управління торговельною компанією. Для реалізації цих вимог сутності та зв’язки між ними будуть перетворені у структуру реляційної бази даних. Зокрема, кожна бізнес-сутність представляється у окрему таблицю з унікальним первинним ключем. Атрибути цих сутностей стають колонками в таблицях. Зв'язки між сутностями реалізуються за допомогою зовнішніх ключів.

Виходячи з аналізу предметної області, було виділено такі ключові сутності для моделювання структури бази даних: Products (Продукти); Categories (Категорії); Brands (Бренди); Customers (Клієнти); Orders (Замовлення); OrderDetails (Деталі замовлення); Employees (Працівники).

Ці сутності найбільш повно описують основні об'єкти та процеси, задіяні у діяльності торгівельної компанії. Було створено такі зв’язки між сутностями:

– Продукти − Категорії (багато-до-одного): продукти класифікуються за категоріями;

– Продукти − Бренди (багато-до-одного): бренди випускають продукти;

– Замовлення − Клієнти (багато-до-одного): клієнти роблять замовлення;

– Замовлення − Працівники (багато-до-одного): працівники обробляють замовлення;

– Замовлення − Деталі замовлень (один-до-багатьох): замовлення складаються з деталей замовлень;

– Деталі замовлень − Продукти: продукти входять в деталі замовлень. (Багато-до-одного).

На основі визначених сутностей та зв’язків побудована фізична модель даних. Завдяки фізичної моделі можна реалізувати БД в конкретній СУБД Microsoft SQL Server. Перевірка працездатності відбуватиметься шляхом генерування та заповнення таблиць тестовими даними. Фізична модель бази даних представлена на рис. 1.

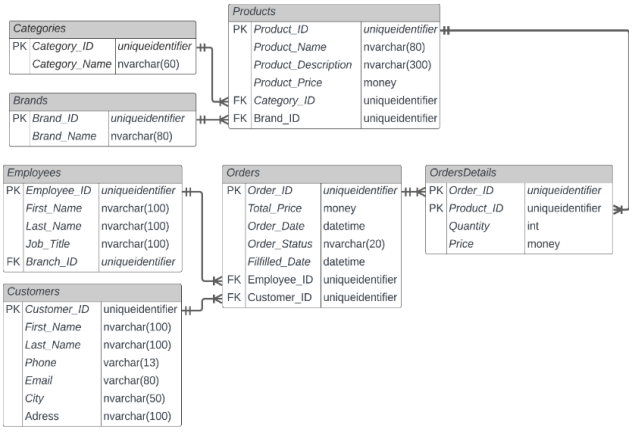

Рис. 1. Фізична модель бази даних

Отже, такий підхід до проектування БД забезпечить створення гнучкої та оптимізованої структури бази даних, що задовольнить вимоги до швидкодії та масштабованості системи.

Ця модель використана для проведення дослідження щодо генерації даних.

Підготовка даних

Для об'єктивного порівняльного аналізу методів генерації тестових даних було сформовано репрезентативну вибірку таблиць з бази даних торгівельної компанії. Обрані таблиці Products, Orders та OrderDetails як ключові для відображення бізнес-процесів компанії з складною структурою баз даних та різними типами зв'язків.

Визначено діапазон обсягів генерованих даних від 1 до 10 млн рядків для кожної таблиці. Сформовано вибірки з довідкових таблиць (Categories, Brands тощо) для забезпечення реалістичності тестових даних. Перевірено працездатність схеми БД шляхом попередньої генерації та заповнення її таблиць.

Основними критеріями відбору таблиць є такі:

– ключова роль у бізнес-процесах компанії (оформлення замовлень, облік товарів);

– складна структура даних з багатьма атрибутами та зв'язками для оцінювання ефективності методів генерації у реалістичних сценаріях;

– наявність різних типів зв'язків (один-до-багатьох, багато-до-одного, багато-до-багатьох).

Комбінація таблиць Orders, OrderDetails і Products найповніше репрезентує ключові сутності та бізнес-процеси компанії.

Тестування цих таблиць дозволить порівняти ефективність методів генерації великих обсягів реалістичних даних.

Аналіз методів генерації даних

У цій роботі проаналізовано наступні три методи генерації даних: генерація циклами, генерація за допомогою CTE (common table expressions) та генерація тимчасовими таблицями.

Метою аналізу є визначення особливостей кожного методу, а також його сильних і слабких сторін. Це дозволить обґрунтувати, за яких умов той чи інший метод буде найбільш ефективним.

Методи генерації даних порівнювалися за такими критеріями:

– швидкість генерації;

– споживання ресурсів (пам'яті);

– здатність масштабуватися при збільшенні об’ємів даних.

Для порівняння методів генерації даних обрано такі метрики:

– швидкість генерації: час, необхідний для генерації рядків таблиць бази даних. Важлива для оцінки ефективності методу для великих баз даних;

– використання ресурсів: об’єм пам'яті, що займає генерація даних. Важлива для аналізу продуктивності та ефективності використання ресурсів;

– масштабованість: здатність методу працювати зі зростанням обсягів даних, вимірюється кількістю згенерованих записів. Методи з хорошою

масштабованістю ефективно обробляють збільшення даних без значного зниження швидкості або збільшення використання ресурсів.

Важливо зазначити, що вибір оптимального методу може залежати від конкретних вимог до даних, а також від обмежень обчислювальної системи.

На цьому етапи розглядаються три основні методи генерації тестових даних:

– генерація за допомогою циклів;
– генерація за допомогою CTE;
– генерація за допомогою тимчасових таблиць.

Генерація за допомогою циклів. Є одним з базових підходів до створення тестових даних за допомогою SQL. Суть цього методу полягає у використанні стандартних засобів T-SQL для організації циклічної обробки та поступової генерації даних з подальшою їх вставкою в цільову таблицю бази даних.

Основою реалізації є використання циклу WHILE, який дозволяє багаторазово повторювати групу операторів, поки задана умова залишається істинною. На кожній ітерації циклу формуються значення для атрибутів чергового запису (рядка таблиці), після чого виконується оператор INSERT для додавання запису в таблицю.

Перевагою цього підходу є простота та гнучкість реалізації логіки генерації даних за допомогою стандартних конструкцій мови T-SQL. Розробник може детально налаштувати алгоритм формування значень атрибутів, використання довідникових даних, логіку вставки записів тощо.

Однак генерація циклами має і певні недоліки. Зокрема, через необхідність постійного звернення до БД для вставки чергової порції даних цей метод може демонструвати гіршу продуктивність в порівнянні з іншими методами. Крім того, існує ризик перевантаження транзакційної системи БД при роботі з великими обсягами даних.

Для оптимізації методу генерації циклами слід застосовувати пакетні вставки даних, коли записи акумулюються в окремій структурі, а потім вставляються групою у таблицю. Це зменшує кількість окремих транзакцій. Також важливо використовувати ефективні способи генерації випадкових значень та посилань на довідникові дані.

Отже, генерація за допомогою циклів є універсальним підходом, який дозволяє реалізувати будь-яку логіку обробки даних на T-SQL. Проте цей метод може вимагати оптимізації для досягнення достатньої продуктивності при створенні великих обсягів тестових даних. **Алгоритм методу реалізовано за такими кроками:**

– створюються допоміжні об'єкти для зберігання довідкових даних, таких як списки можливих категорій, брендів тощо. Це можуть бути тимчасові таблиці, табличні змінні або звичайні змінні;
– ініціалізація циклу. Встановлюється лічильник записів та задається максимальна кількість записів для генерування;
– на кожній ітерації генеруються випадкові значення для атрибутів запису (ціна, опис тощо). Обираються випадкові посилання на довідкові дані (категорія, бренд) з підготовлених таблиць. Формується INSERT-запит для вставки запису у цільову таблицю;
– видалення тимчасових об'єктів. Після завершення циклу видаляються допоміжні таблиці та змінні.

Генерація за допомогою CTE. Є ефективним підходом до генерації тестових даних, який дозволяє істотно прискорити цей процес у порівнянні з традиційними циклами. Суть методу полягає у визначенні CTE − спеціального об'єкту, який конструює тимчасовий набір результатів для подальшого використання в запиті [7]. На відміну від фізичних таблиць, CTE існує тільки під час виконання запиту і не зберігається постійно в базі даних. CTE дозволяє реалізувати складну логіку генерації даних в рамках єдиного запиту. Наприклад, за допомогою рекурсивних CTE можна ефективно моделювати ієрархічні дані. Інша перевага − швидкість, оскільки дані генеруються "на льоту" [8] без звернень до БД. Після генерації дані з CTE однією операцією INSERT вставляються у цільову таблицю. Це значно ефективніше традиційних підходів, де кожен новий рядок вимагає окремої транзакції. Однак CTE має деякі обмеження: наприклад, складність реалізації логіки, неможливість повторного використання CTE, відсутність підтримки деяких оптимізацій тощо.

Загалом, генерація за допомогою CTE є інструментом, що дозволяє істотно прискорити створення тестових даних різної структури та обсягів. Проте він вимагає ретельного підходу, з врахуванням його специфіки та особливостей конкретної СУБД.

Алгоритм методу CTE реалізовано за такими кроками:

– створюється CTE, яка буде містити генеровані дані для таблиці. Вона оголошується на початку запиту за допомогою ключового слова WITH [8];
– всередині CTE створюється рекурсивний підзапит, який ітеративно генерує дані. На кожній ітерації відбувається генерація випадкових значень для основних атрибутів таблиці;
– для генерації значень зовнішніх ключів, які посилаються на інші таблиці, використовуються вкладені підзапити. Вони вибирають випадкові існуючі значення з відповідних таблиць;
– генерація даних відбувається в обмеженій кількості ітерацій, необхідній для отримання бажаної кількості рядків;
– після завершення генерації, дані з CTE ефективно вставляються в цільову таблицю Products за допомогою однієї операції INSERT.

Генерація за допомогою тимчасових таблиць. Цей підхід базується на використанні тимчасових таблиць в SQL для проміжного зберігання даних під час генерації. На відміну від звичайних таблиць, тимчасові існують тільки протягом поточного сеансу і автоматично видаляються після завершення роботи [9].

Даний підхід дозволяє оптимізувати процес генерації за рахунок підготовки та зберігання проміжних даних. Однак він також вимагає додаткових ресурсів для тимчасових таблиць та уваги до їх видалення. Для його ефективного застосування слід

використовувати оптимальний розмір пакетів даних, індексацію тимчасових таблиць, зберігання лише необхідних стовпців. Це дозволить отримати певний баланс продуктивності та економії ресурсів при генерації тестових даних.

Отже, використання тимчасових таблиць − метод, який дає змогу оптимізувати процес створення великих наборів тестових даних в SQL за рахунок ефективного використання проміжних результатів. **Алгоритм методу реалізовано за такими кроками:**

– створюються тимчасові таблиці для вибірок категорій і брендів;

– генерується ще одна тимчасова таблиця з випадковими даними для продуктів;

– за допомогою INSERT відбувається вставка даних у цільову таблицю на основі тимчасових;

– після завершення видаляються усі тимчасові таблиці.

Проаналізовані методи генерації даних наведені в табл. 1.

Таблиця 1 − **Основні методи перерозподілу мережних ресурсів**

Методи	Особливості	Переваги	Недоліки
Генерація за допомогою циклів	Використовуються цикли T-SQL (WHILE) [10]; поетапна генерація та INSERT даних в цільову таблицю.	Простота реалізації; можливість детального налаштування [10].	Велике навантаження на транзакції; повільніший за інші методи.
Генерація за допомогою CTE	Використовується конструкція Common Table Expression [11]; генерування та INSERT даних відбувається в одному запиті.	Висока швидкість обробки; Мінімум накладних витрат.	Складніший підхід; Обмежені можливості налаштування [11].
Генерація з використанням тимчасових таблиць	Спочатку формуються тимчасові таблиці [12]; потім виконується INSERT даних в цільову таблицю.	Висока швидкість обробки; гнучке налаштування логіки.	Більш складна реалізація [12]; потреба в очищенні тимчасових таблиць; обмежена кількість генерації випадкових даних.

На наступних етапах дослідження проведено експериментальне тестування проаналізованих методів генерації даних на основі обраних метрик. Це дозволить зробити обґрунтовані висновки щодо оптимальності кожного методу для різних сценаріїв та обсягів даних.

Критерії оцінювання ефективності методів генерації даних

Для проведення дослідження було визначено ключові метрики для оцінки різних методів генерування тестових даних – швидкість (час) генерації, використання системних ресурсів та масштабованість записів в кожній таблиці БД. Для системного аналізу ефективності методів генерування тестових даних розглядалися такі критерії: час генерації, обсяг згенерованих даних та масштабованість. Час генерації є ключовим показником продуктивності методу, оскільки він відображає здатність оперативно формувати великі обсяги даних, що є необхідним для тестування методів. Обсяг згенерованих даних вказує на використання дискового простору бази даних. Оптимальне використання дискового простору є критичним для великих обсягів даних і тестування навантаження. Цей показник допомагає виявити проблемні точки (вузькі місця) в методах генерування даних. Масштабованість відображає здатність методів підтримувати прийнятну продуктивність при збільшенні обсягів даних. Це є важливим аспектом надійної роботи методів, що генерують великі набори даних.

Отже, детальний аналіз часу виконання, обсягу згенерованих даних та масштабованості різних методів є складовими всебічної оцінки їх ефективності. Поєднання цих критеріїв із раніше визначеними метриками дозволяє комплексно оцінити переваги, недоліки та обмеження кожного методу генерації тестових даних для конкретних сценаріїв використання реляційних баз даних.

Експериментальні дослідження методів генерації даних

Експериментальне тестування є важливим етапом дослідження, оскільки саме на його основі можна об'єктивно порівняти ефективність різних методів генерації тестових даних та визначити оптимальні підходи для конкретних сценаріїв.

Метою цього етапу є емпірична перевірка ефективності трьох проаналізованих раніше методів генерування даних на основі визначених метрик. Для цього було розроблено експериментальну методику, що включає:

– генерацію наборів даних різного обсягу (від 1 до 10 млн рядків, окрім метода тимчасових таблиць, так як в цьому методі використовується обмежена системна змінна sys.all_objects для генерації випадкових записів, і цей метод обмежується 5 млн рядків) для заданих таблиць Products, Orders, OrderDetails;

– використання кожного з трьох методів генерації окремо для кожного набору даних таблиць;

– вимірювання часу виконання та обсягу займаного місця в БД;

– порівняння отриманих результатів та їх аналіз.

Вибір трьох таблиць (Products, Orders, OrderDetails) для тестування обумовлений їх ключовою роллю у бізнес-процесах компанії, а також складною структурою та зв’язками, що дозволяє оцінити методи в реалістичних умовах. Обраний діапазон обсягів даних від 1 до 10 млн рядків дає змогу дослідити поведінку алгоритмів як на невеликих, так і на значних масивах даних. Результати експериментів разом з теоретичним оглядом та аналізом дозволять комплексно оцінити їх ефективність та сформулювати конкретні рекомендації щодо їх подальшого використання.

Аналіз результатів для ключової таблиці Products, яка містить інформацію про всі товари компанії, графік часу генерації представлено на рис. 2. Аналізуючи час генерації даних, можна зробити висновок, що метод з використанням рекурсивних CTE має певну перевагу.

Наприклад, для генерації 5 млн рядків йому знадобилося 31 сек., а для генерації 10 млн – 1 хв. 58 сек. Це можна пояснити тим, що рекурсивні CTE дозволяють значно оптимізувати процес обробки даних, уникаючи непотрібних операцій читання/запису в БД на кожній ітерації.

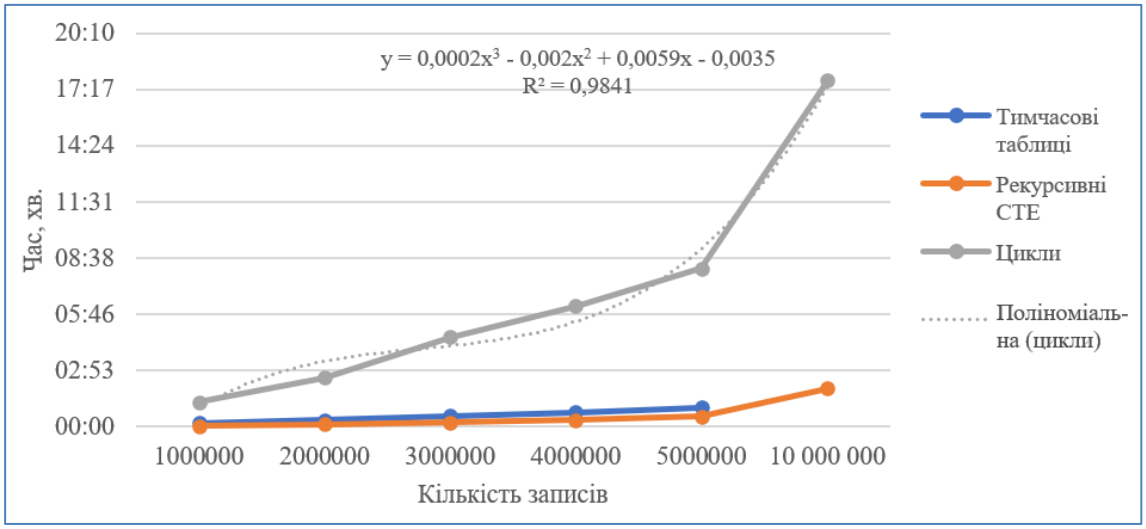

Рис. 2. Графік залежності часу генерації від кількості записів для таблиці Products

Натомість метод з циклами вимагає постійного звернення до БД для запису порції даних, що істотно уповільнює процес. Саме тому на генерацію того ж обсягу даних було витрачено 17 хв. 46 сек.

Метод тимчасових таблиць значно повільніший, а також обмежений у часі генерації ніж рекурсивні CTE. Це тому, що тимчасові таблиці також вимагають додаткових операцій читання з БД для збереження проміжних даних. За обсягом займаного місця перевага у методі тимчасових таблиць – 673 МБ після генерації 5 млн рядків (рис. 3). Це пояснюється тим, що вони зберігають лише необхідні для генерації стовпці на відміну від повноцінних таблиць.

Натомість методи з CTE та циклами формують повні табличні дані, що й призводить до більшого обсягу для 5 млн записів – 975 МБ, а для 10 млн – до 1991 МБ у методі CTE та 2000 МБ у методі використання циклів.

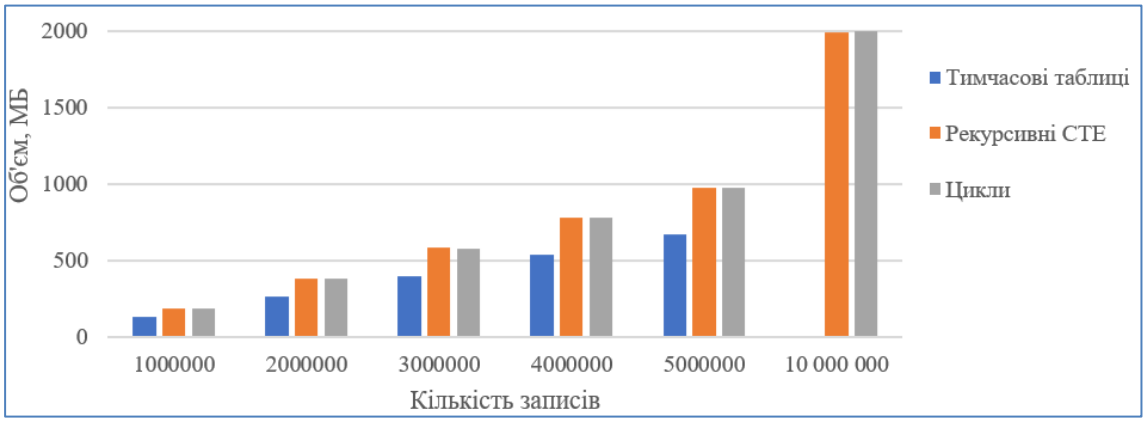

Рис. 3. Графік залежності об'єму згенерованих даних від кількості записів у таблиці Products

Отже, при генерації даних у таблиці Products рекурсивні CTE демонструють найкращу швидкодію завдяки ефективній обробці даних без зайвих операцій з БД, а шо до обсягу даних, тимчасові таблиці мають невелику перевагу. Далі наведено аналіз генерації даних для таблиці Orders. Результати часу генерації для таблиці представлено на рис. 4.

Треба звернути увагу на те, що для таблиці Orders спостерігається подібна тенденція щодо швидкості генерації даних, що і для таблиці Products. Зокрема, рекурсивні CTE показують найкращу швидкодію – 32 сек. для 5 млн рядків та 1 хв. 59 сек. – для 10 млн.

Для таблиці Orders різниця між CTE та тимчасовими таблицями відрізняється практично на 150%.

Що стосується обсягу згенерованих даних (рис. 5), тимчасові таблиці показують більшу економію місця: різниця у порівнянні з методом CTE складає майже 230 МБ на користь тимчасових таблиць для 5 млн записів. Отже, хоча за швидкістю

рекурсивні CTE мають перевагу, суттєва економія місця при використанні тимчасових таблиць заслуговує на увагу, але ж метод з тимчасовими таблицями має певні обмеження щодо об'ємів генерованих даних.

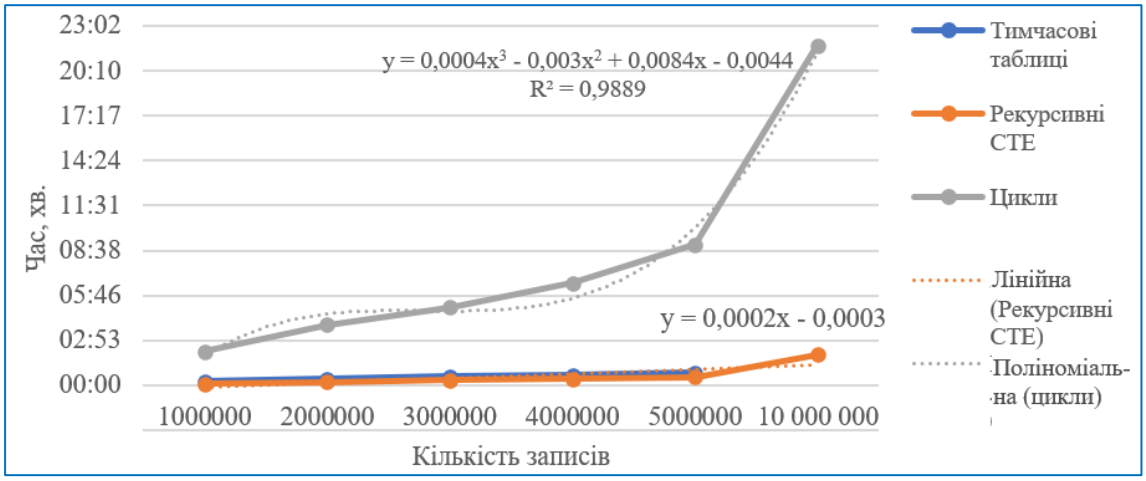

Рис. 4. Графік залежності часу генерації від кількості записів для таблиці Orders

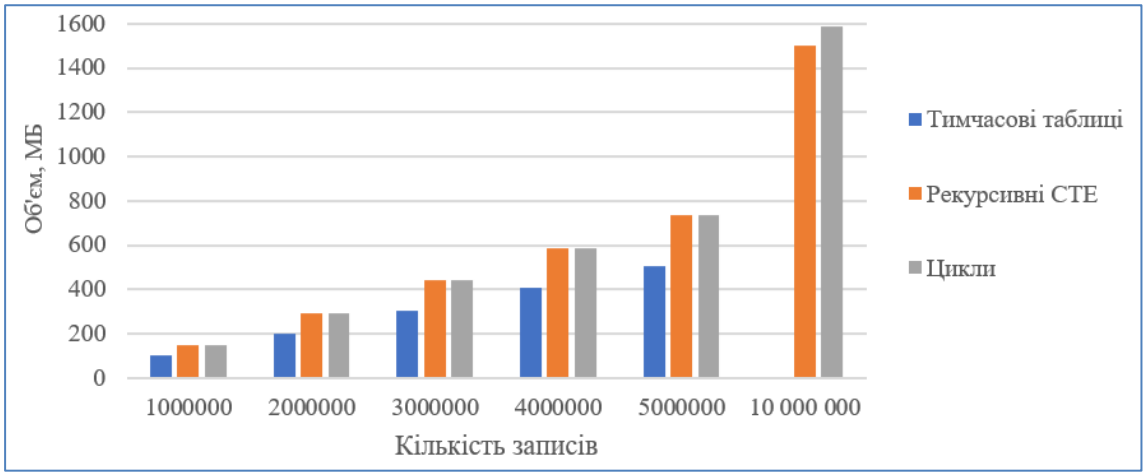

Рис. 5. Графік залежності об'єму згенерованих даних від кількості записів для таблиці Orders

Таблиця OrdersDetails. Результати її генерації − часу генерації даних для цієї таблиці − представлено на рис. 6.

Результати демонструють високу швидкість генерації даних за допомогою рекурсивних CTE. На відміну від попередніх таблиць, ця перевага є більш значною − для генерації 5 мільйонів рядків рекурсивні CTE витратили 16 сек., що у 5 разів швидше, ніж метод тимчасових таблиць, та у 16 разів швидше, ніж метод циклів.

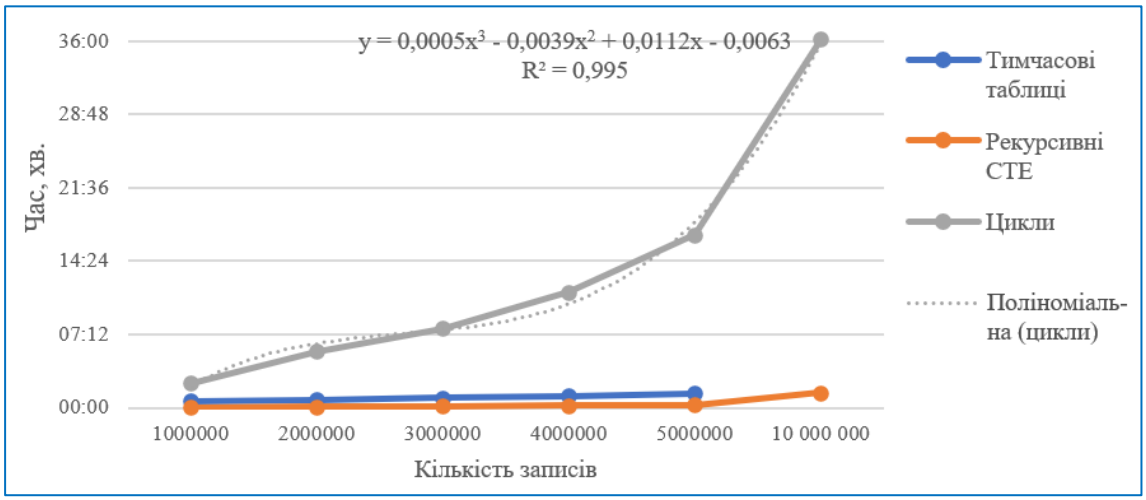

Рис. 6. Графік залежності часу генерації від кількості записів для таблиці OrdersDetails

Такі результати пояснюються особливостями оптимізації запитів в рекурсивних CTE.

Вони дозволяють ефективно генерувати складні ієрархічні дані, що й є актуальним для таблиць типу OrdersDetails.

Натомість методи тимчасових таблиць та циклів потребують більше часу через необхідність створення та обробки проміжних результатів. Це помітно для методу циклів, який витратив майже 17 хв. на той самий обсяг даних. Щодо обсягу згенерованих даних (рис. 7), то рекурсивні CTE та тимчасові таблиці продемонстрували схожі результати для 5 млн записів. При використанні методу циклів знадобилось більше місця через генерування повних таблиць.

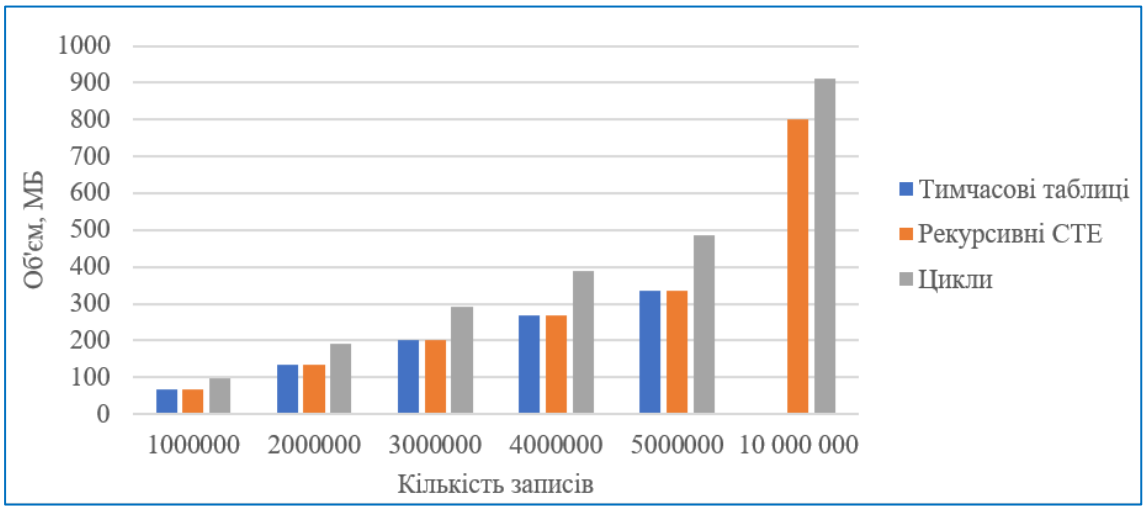

Рис. 7. Графік залежності об'єму згенерованих даних від кількості записів для таблиці OrdersDetails

Отже, для таблиці OrdersDetails найбільш ефективним є метод з рекурсивними CTE. Він перевершує інші методи як за часом генерації, так і за економією місця в БД. Це підтверджує високу ефективність методу CTE для складних структур даних.

Отже, результати експериментального тестування трьох методів генерації даних для тестових таблиць Products, Orders та OrderDetails визначили на основі метрик ефективності їх використання наступне.

1. **Швидкість (час) генерації**:

– рекурсивні CTE виявилися найефективнішим методом генерації даних для всіх трьох таблиць;

– тимчасові таблиці також демонстрували прийнятну швидкість, але в порівнянні із рекурсивними CTE були менш продуктивними, а також є обмеженими щодо обсягів генерації даних;

– циклічний метод виявився значно менш ефективним для великих обсягів даних, особливо для таблиць Orders та OrdersDetails.

2. **Використання ресурсів**:

– рекурсивні CTE економлять ресурси та прискорюють генерацію, оскільки уникають зайвих операцій читання/запису в БД на кожній ітерації;

– тимчасові таблиці потребують додаткових операцій читання/запису для збереження проміжних даних, що призводить до зменшення ефективності в порівнянні з рекурсивними CTE;

– циклічний метод вимагає постійного звертання до БД для запису порції даних, що призводить до істотного зниження продуктивності.

3. **Масштабованість**:

– рекурсивні CTE виявили високу масштабованість, зберігаючи ефективність при роботі з різними обсягами даних;

– тимчасові таблиці також забезпечують прийнятну масштабованість, але ж меншу в порівнянні з рекурсивними CTE;

– циклічний метод зменшує ефективність при роботі з великими обсягами даних.

4. **Обсяг займаного місця в БД**:

– тимчасові таблиці ефективно економлять місце в БД, зберігаючи лише необхідні для генерації стовпці;

– рекурсивні CTE та метод циклів формують повні табличні дані, що призводить до збільшення обсягу в БД;

– різниця щодо об’єму пам’яті між рекурсивними CTE та циклічним методом не є критичною, але доведена ефективність методу тимчасових таблиць щодо збереження даних.

Загалом, результати експериментального дослідження дозволяють зробити висновок, що найбільш універсальним та ефективним підходом до генерації тестових даних для заданих таблиць є використання рекурсивних CTE. Цей метод продемонстрував оптимальне поєднання швидкодії (часу генерацій), економії ресурсів, масштабованості та надійності. Переваги рекурсивних CTE полягають в ефективній оптимізації запитів, уникненні зайвих операцій з БД та високій гнучкості до структури і обсягу даних. Це робить їх найбільш придатним інструментом для генерації як простих, так і складних ієрархічних даних в широкому діапазоні їх масштабованості.

Циклічний метод виявився найменш продуктивним та ресурсоємним. Отже, його можна рекомендувати **лише для невеликих обсягів** простих даних через низьку масштабованість.

Таким чином, можна зробити висновок, що рекурсивні CTE є оптимальним універсальним

інструментом генерації тестових даних для задач тестування та навантаження реляційних БД.

Висновки

У даній роботі досліджено актуальну проблему генерації тестових даних великих обсягів для завдань тестування та оптимізації роботи реляційних баз даних.

За результатами експериментального порівняльного тестування методів на основі ключових метрик ефективності (часу генерації, витрат ресурсів, масштабованості) обґрунтовано, що найбільш ефективним є метод рекурсивних CTE.

Цей метод продемонстрував оптимальне поєднання швидкодії, економії ресурсів та надійності для широкого діапазону обсягів і структур даних. Водночас, за певних умов, доцільно поєднувати переваги різних методів: зокрема, тимчасові таблиці можуть застосовуватися для економії місця при середніх обсягах даних, а циклічний метод є обмежено придатним, зокрема для невеликих простих наборів даних.

Таким чином, проведене дослідження надає комплексне уявлення про можливості та обмеження різних методів генерації даних у реальних умовах їх застосування.

Результати даного дослідження надають рекомендації щодо їх використання для генерації тестових даних на хмарних платформах – наприклад, для тестування сервісів Azure SQL Database та Azure WebApp + SQL Database хмарної платформи Azure для прийнятті рішень щодо вибору оптимального рівня моделей DTU та vCore при моделюванні процесів генерації даних та створенні запитів різної складності у реляційних БД [13].

СПИСОК ЛІТЕРАТУРИ

1. Synthetic Data Generation [Електронний ресурс] // AI Multiple. – Режим доступу: https://research.aimultiple.com/synthetic-data-generation/. – Дата звернення: 02.03.2024.
2. Unit Testing: Generate Test Data Sets SQL Server Database Applications [Електронний ресурс] // MSSQLTips. – Режим доступу: https://www.mssqltips.com/sqlservertip/7766/unit-testing-generate-test-data-sets-sql-server-database-applications/.
3. Houkjær K., Torp K., Wind R. Simple and Realistic Data Generation // Very Large Data Base Endowment Inc. (VLDB Endowment). 2006. С. 1243-1246. – Режим доступу: https://www.vldb.org/conf/2006/p1243-houkjar.pdf /.
4. WHILE (Transact-SQL) [Електронний ресурс] // Microsoft Learn. – Режим доступу: https://learn.microsoft.com/ru-ru/sql/t-sql/language-elements/while-transact-sql?view=sql-server-ver16.
5. Common Table Expressions (CTEs) vs Temporary Tables in BigQuery [Електронний ресурс] // Medium. – Режим доступу: https://medium.com/@sahaabhik9/common-table-expressions-ctes-vs-temporary-tables-in-bigquery-f6057e688f01.
6. Сергій Мінухін, Мирослав Башкіров. Моделювання роботи з базами даних торгівельних компаній на хмарних платформах // Інформаційні системи та технології: матеріали 12-ої Міжнародної науково-технічної конференції. Частина 2. Молодіжна секція, Харків, (28 листопада 2023 – 01 грудня 2023 р.) / наук. ред. В.В. Безкоровайний, L. Petryshyn, З.В. Дудар, Ю.В. Міщеряков.: ХНУРЕ, 2023. – С.45–47.
7. Common Table Expressions (CTE) [Електронний ресурс] // dbschema.com. 2023. – Режим доступу: https://dbschema.com/2023/07/02/sqlserver/common-table-expressions/.
8. Stuparu D., Petrescu M. Common Table Expression: Different database systems approach. Journal of Computer-Mediated Communication. 2009. Vol. 6, No. 3. – Режим доступу: https://www.researchgate.net/publication/259441876_Common_Table_Expression_Different_database_systems_approach.
9. Which are more performant: CTE or Temporary Tables? [Електронний ресурс] // Stack Overflow. – Режим доступу: https://stackoverflow.com/questions/690465/which-are-more-performant-cte-or-temporary-tables.
10. T-SQL WHILE Loop – GOTO Loop Time Series Data [Електронний ресурс] // MSSQLTips. – Режим доступу: https://www.mssqltips.com/sqlservertip/7139/t-sql-while-loop-goto-loop-time-series-data/.
11. WITH common_table_expression (Transact-SQL) [Електронний ресурс] // Microsoft Learn. – Режим доступу: https://learn.microsoft.com/en-us/sql/t-sql/queries/with-common-table-expression-transact-sql?view=sql-server-ver16.
12. SQL Temp Table: How to Create a Temporary SQL Table [Електронний ресурс] // freeCodeCamp.org. – Режим доступу: https://www.freecodecamp.org/news/sql-temp-table-how-to-create-a-temporary-sql-table/.
13. Minukhin S. Performance study of the DTU model for relational databases on the Azure platform // Сучасний стан наукових досліджень та технологій в промисловості. – 2022. – №. 1 (19). – С. 27-39. https://doi.org/10.30837/ITSSI.2022.19.027.

Received (Надійшла) 12.02.2024
Accepted for publication (Прийнята до друку) 17.04.2024

to research the effectiveness of test data generation methods in relational databases

S. Minukhin, M. Bashkirov

Abstract. The article analyzes three main methods of generating test data for relational databases: cyclic, based on Common Table Expressions (CTE) and using temporary tables. Experimental testing was carried out in the MS SQL Server environment on the example of database tables of a trading company. A comparison was made by key metrics of generation efficiency - generation time, system resource utilization, and scalability of the test database tables. The results of the study substantiated the effectiveness of the recursive CTE method for different volumes and data structures. Recommendations for choosing test suite generation methods in accordance with the requirements of database projects, in particular when using services for working with relational databases on cloud platforms, are given.

Keywords: Database, Microsoft SQL Server, Testing, Database generation, T-SQL, Optimization, Recursive queries, Temporary Tables, Generation with Loops.