УДК 658.012

Б. І. Мороз, доктор технічних наук,
декан факультету інформаційних
та транспортних систем і технологій
Університету митної справи та фінансів
Л. В. Кабак, кандидат технічних наук, доцент
кафедри інформаційних систем та технологій
Університету митної справи та фінансів
О. Н. Молотков, кандидат технічних наук, доцент
кафедри інформаційних систем та технологій
Університету митної справи та фінансів
Ю. В. Ульяновська, кандидат технічних наук,
доцент кафедри інформаційних систем
та технологій Університету митної справи
та фінансів
В. М. Пономарьов, старший викладач кафедри
інформаційних систем та технологій
Університету митної справи та фінансів

МЕТОД КЕРУВАННЯ ТРАНЗАКЦІЯМИ, ЯКІ КОНКУРУЮТЬ В OLTP-СИСТЕМАХ

Досліджено основні напрями розвитку моделей керування транзакціями, що працюють у суміші. Запропоновано метод керування транзакціями, які працюють у суміші, за ознаками старіння інформації. Метод доцільно використовувати для керування транзакціями в OLTP-системах одночасно із синхронними алгоритмами за умови, що на кожен об'єкт, який бере участь у транзакціях, будуть виписані операції, на які слід застосовувати асинхронну фіксацію в разі взаємоблокувань.

Ключові слова: *транзакція; серіалізація; старіння інформації; OLTP-системи; транзакції, які взаємоблокуються; управління транзакціями.*

The researches are realized, the basic directions of development of models for transaction management working in the mixture. We propose a method for transaction management, working in a mixture on the basis of the information aging. The developed method should be used for transaction management in OLTP systems. The proposed method can be used together with synchronous algorithms provided that each object that takes part in transactions are transactions issued to which to apply in the case of asynchronous fixation deadlocks.

Key words: *transaction; serialization; aging information; OLTP system; blocking transaction; transaction controlling.*

Постановка проблеми. Нині як у Департаменті митної справи України, так і в багатьох організаціях обробляється великий обсяг даних, на основі яких можна виконувати різноманітні аналітичні й управлінські завдання. Проблеми збереження й обробки аналітичної інформації стають усе актуальнішими і привертають увагу фахівців і фірм, які працюють у галузі інформаційних технологій, що привело до формування повноцінного ринку технологій бізнес-аналізу.

Постановка завдання. Збирання та зберігання інформації, а також виконання завдань інформаційно-пошукового запиту ефективно реалізуються засобами систем керування базами даних (СКБД). У підсистемах OLTP (Online Transaction Processing) реалізується транзакційна обробка даних. Організація керування транзакціями, які працюють у суміші, з метою уникнення блокувань – досить складне завдання, для виконання якого існує багато механізмів. Проте універсального рішення нині практично немає. Майже всі готові системи OLTP накладають істотні обмеження на структуру й способи накопичення і зміни даних у таблицях бази даних. Це пов'язано з різноманіттям цілей використання СКБД в обробці даних.

Аналіз останніх досліджень і публікацій. Із завантаженням даних з OLTP-системи в сховище даних (далі – СД) відбувається дублювання даних. Проте в ході цього завантаження дані фільтруються, оскільки не всі з них мають значення для проведення процедур аналізу. В СД зберігається узагальнена інформація, якої в OLTP-системі немає. Надмірність інформації можна звести до нуля, використовуючи віртуальне СД. У такій системі дані з OLTP-системи не копіюються в єдине сховище. Вони витягуються, перетворюються й інтегруються безпосередньо під час аналітичних запитів у режимі реального часу. Фактично такі запити безпосередньо передаються до OLTP-системи.

OLTP-системи не орієнтовані на зберігання даних за тривалий період часу, в міру необхідності дані вивантажуються в архівні, тому не завжди є фізична можливість отримання повного набору даних у СД.

Швидкодія інформаційних систем з використанням OLTP-серверів залежить від ефективності керування транзакціями. Розглянемо наявні методи керування транзакціями, які використовуються в OLTP-системах.

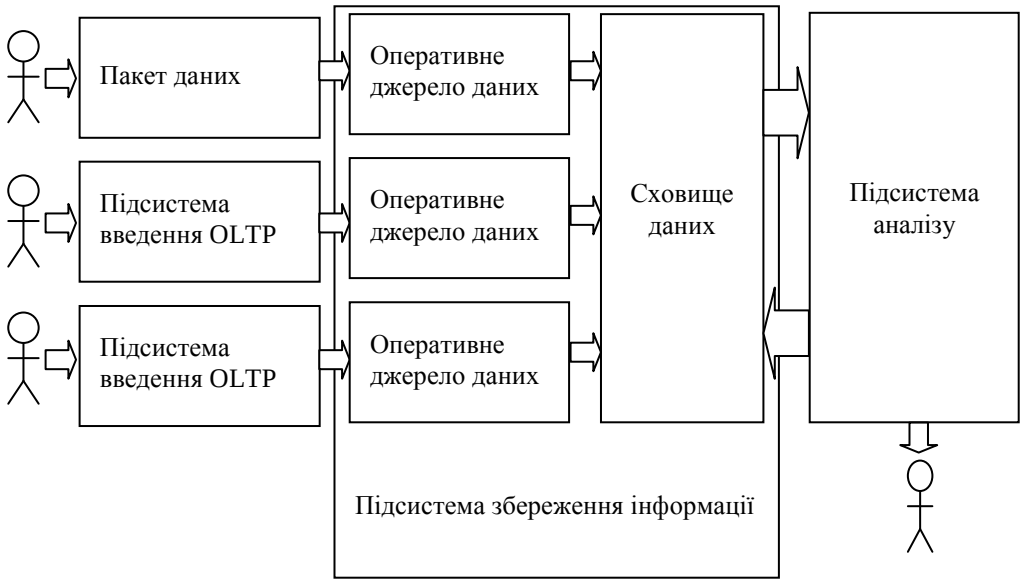

Рис. 1. Структура розподіленої інформаційної системи з використанням OLTP-систем

Найважливішим фундаментальним поняттям у сучасних системах керування базами даних (далі – СКБД) є транзакція. Транзакція – це набір неподільних атомарних операцій над базою даних. Виділяються чотири основні властивості транзакцій: атомарність (atomicity), узгодженість (consistency), ізоляція (isolation) і довговічність (durability). Ми позначатимемо ці властивості абревіатурою за першими літерами назв відповідних властивостей ACID. Властивість атомарності означає, що або результати всіх операторів, які входять у транзакцію, відображаються в базі даних, або впливу цих операторів зовсім немає. Властивість узгодженості означає, що транзакції переводять один узгоджений стан бази даних в інший без обов'язкової підтримки узгодженості в усіх проміжних точках. Властивість ізольованості означає, що транзакції, які виконуються, "не бачать одна одну". Це означає: якщо навіть запустити безліч конкуруючих транзакцій, будь-яке оновлення певної транзакції буде приховано від інших до тих пір, поки ця транзакція не завершиться. Властивість довговічності означає, що коли транзакція виконана, її оновлення зберігаються, навіть якщо в наступний момент відбудеться збій у системі [1].

Відповідно до вищеописаних властивостей транзакцій можна поділити на групи методи забезпечення цих властивостей. Перша група складається з методів забезпечення ізоляції паралельних транзакцій, друга – з методів забезпечення атомарності й довготривалості транзакцій. Дотепер розроблено досить велику кількість алгоритмів, методів і підходів у рамках кожної з цих груп.

Однією з основних вимог до сучасних СКБД є підтримка мультирежиму транзакцій, який означає можливість одночасної обробки в СКБД декількох транзакцій з доступом до одних і тих самих даних в один і той самий час. Як відомо, в такій системі для коректної обробки транзакцій без виникнення конфліктних ситуацій необхідні методи контролю виконання паралельних транзакцій. Без використання таких методів у СКБД можуть виникати такі ситуації, як втрата результатів оновлення, "брудне" читання, читання, яке не повторюється, та фантоми [1]. Якщо з базою даних одночасно працюють кілька користувачів, СКБД має не тільки коректно виконати індивідуальні транзакції та відновлювати узгоджений стан бази даних, але й забезпечити коректну паралельну роботу всіх користувачів над одними й тими самими даними.

Серіалізація транзакцій – це механізм виконання транзакцій за деяким планом, завдяки якому транзакції працюють одночасно, не заважаючи одна одній. Забезпечення такого механізму є основною функцією компонента СКБД, відповідального за керування транзакціями.

Цей механізм працює за такими правилами.

1. У процесі виконання транзакції програма "бачить" лише узгоджені стани бази даних; користувач ніколи не може отримати доступ до незафіксованих змін у даних, досягнутих у результаті дії іншої програми.

2. Якщо дві транзакції A і B виконуються паралельно, то СКБД вважає, що результат буде таким самим, якби транзакція A виконувалась першою, а за нею – виконувалась інша транзакція B, тобто послідовно.

Основна реалізаційна проблема полягає у виборі серіалізаційного набору транзакцій, який би не надто обмежував їх паралельність; найпростішим може бути вибір послідовного виконання транзакцій. Між транзакціями можуть виникати такі види конфліктів:

- W-W – транзакція 2 намагається змінити об'єкт, змінений незакінченою транзакцією 1;
- R-W – транзакція 2 намагається змінити об'єкт 1, прочитаний незакінченою транзакцією 1;
- W-R – транзакція 2 намагається прочитати об'єкт, змінений незавершеною транзакцією 1.

Виходячи з теорії, що кожен користувач і кожна транзакція повинні мати властивість ізольованості, вони мають виконуватись так, ніби тільки один користувач працював з базою даних. Засоби СКБД дозволяють ізолювати користувачів один від одного, проте в цьому випадку виникають проблеми з уповільненням роботи. У реалізації метод керування паралельними транзакціями визначає поведінку планувальника транзакцій. Основне завдання планувальника полягає у своєчасному виявленні та розв'язанні конфліктів між виконуваними транзакціями. Після виявлення конфлікту СКБД має вибрати метод його усунення.

Методи розв'язання конфліктів блокування відбирають одну з конфліктуючих транзакцій і виконують її після закінчення всіх інших транзакцій.

Найвідоміші методи, які широко застосовуються, розглядаються у книгах Бернштейна і Гудмена [1].

Бернштейн і Гудмен зазначають, що найбільш ранній відомий їм алгоритм роботи планувальника для версійних СКБД базується на тимчасових мітках. Цей планувальник обробляє операції так, щоб сумарний результат виконання операцій був еквівалентний послідовному виконанню транзакцій. При цьому їх порядок задається порядком тимчасових міток, які отримують транзакції під час старту. Тимчасові мітки також використовуються для ідентифікації версій даних під час читання і модифікації. Кожна версія отримує тимчасову мітку тієї транзакції, яка її записала. Планувальник не тільки стежить за порядком виконання дій транзакцій, але й відповідає за трансформацію операцій над даними в операції над версіями, тобто кожна операція виду "прочитати елемент даних *x*" має бути перетворена планувальником на операцію: "прочитати версію в елементі даних *x*". Такий алгоритм роботи прийнято називати MVTO (Multi Version Transactions Optimization).

Також у сучасних СКБД використовується багатоверсійний варіант двофазного протоколу синхронізації MV2PL (Multiversion Two-Phase Locking Protocol). У MV2PL планувальник стежить за тим, щоб у кожний момент часу існувало не більше однієї незавершеної версії. Залежно від того, чи дозволяється транзакціям читати незавершені версії даних, розрізняють два варіанти цього алгоритму. Цей алгоритм нічого не говорить про кількість версій одного й того самого елемента, які можуть одночасно існувати в базі даних. Це створює проблеми зі зберіганням версій. По-перше, вони можуть займати занадто багато місця (легко уявити ситуацію, коли обсяг старих версій стає в кілька разів більший, ніж обсяг усієї поточної бази даних). По-друге, виникають труднощі з розміщенням цих "старих" даних. Ураховуючи, що кількість версій заздалегідь не відома, складно створити ефективну структуру їх зберігання, яка б не призвела до помітних витрат. І, нарешті, така система надто складна в реалізації.

Ймовірно, саме через ці проблеми на практиці найчастіше використовується протокол 2V2PL, уперше запропонований у праці Байєра [2]. У цьому протоколі можливе одночасне існування двох версій елемента даних: однієї поточної версії даних і не більше однієї незавершеної. Така організація версій вигідна насамперед для транзакцій, що виконують операцію читання. У 2V2PL використовуються три типи блокувань. Кожне блокування утримується до кінця транзакції.

Також існують багатоверсійні протоколи ROMV (Multiversion Protocol for Read-Only Transactions) для транзакцій, які не змінюють дані.

У роботі багатьох додатків переважають транзакції, що не змінюють дані (read-only transactions). Такі додатки зчитують і аналізують великі обсяги даних. За наявності

хоча б невеликої кількості паралельно виконуваних транзакцій, які роблять зміни, компонент, що відповідає за керування паралельними транзакціями, повинен мати узгодженість прочитаних даних. У разі використання алгоритмів планування без підтримки версій такі довготривалі транзакції можуть призвести до надзвичайного падіння продуктивності системи. Наприклад, у використанні 2PL існує дуже велика ймовірність блокування транзакцій, які роблять оновлення даних. Результатом роботи цих транзакцій буде дуже великий час відгуку.

Багатоверсійні алгоритми дозволяють уникнути подібних проблем завдяки тому, що транзакція, яка вносить зміни до бази даних, не конфліктує з транзакціями читання. Але водночас багатоверсійний алгоритми зазвичай складний у реалізації, запити до версійної СКБД призводять до помітних витрат. ROMV-планувальник розділяє всі транзакції під час їх створення на дві групи: запити (queries) і транзакції, що змінюють дані (update transactions). Відповідно, транзакції різних типів обробляються по-різному. Такий протокол виявляється простішим у реалізації та дозволяє отримати більше вигод, що надаються суто версійним протоколам.

Багатоверсійний алгоритм планування узагальнює широко відому моноверсійну техніку SGT (Serialization Graph Testing). Планувальники, які базуються на SGT, працюють за таким принципом. Планувальник підтримує граф конфліктів, у якому вершини й дуги додаються динамічно залежно від операцій, які отримує на вхід планувальник. При цьому конфліктуючими називаються будь-які дві операції над одним і тим самим елементом даних, якщо хоча б одна з них є операцією модифікації. Інакше кажучи, для конфліктуючих операцій бажано мати порядок їх виконання. Таким чином, планування конфліктуючих операцій накладає обмеження на порядок серіалізації транзакцій. Ці обмеження і висловлює граф конфліктів (рис. 2). Далі розглянемо, як відбувається планування чергової операції *pi(x)* за допомогою SGT-планувальника.

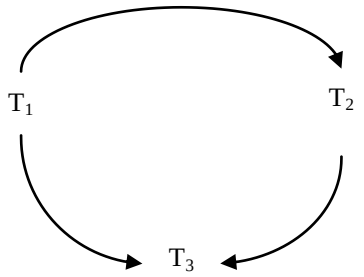

Рис. 2. Граф серіалізації для плану *S*, складений за правилами SGT

Якщо це перша операція транзакції *ti*, що надійшла планувальнику, то створюється новий вузол у графі серіалізації.

У граф додаються дуги виду *(tj, ti)* для кожної запланованої раніше операції *qj(x) (i ≠ j)*, яка конфліктує з *pi(x)*. Тепер можливі два варіанти:

Результуючий граф містить цикли. У цьому випадку транзакція *ti* відкладається. Отриманий граф ациклічний. У цьому випадку дія додається до списку запланованих.

Цей граф не має циклів. Однак для *S* не існує еквівалентного моноверсійного послідовного розкладу [3; 4].

Мета статті. Розглянуті методи керування транзакціями не враховують ціну та цінність інформації під час вилучення транзакції з циклу. У праці [5] наведено методи для врахування старіння та ціни інформації для керування транзакціями. Мета статті – розробка методів, які враховують критерії ціни та цінності інформації для керування транзакціями.

Виклад основного матеріалу. Ефективно організувати управління транзакціями можна за допомогою керованих дисциплін обслуговування. Ці дисципліни мають переваги через статичні дисципліни обслуговування.

Характеристики, використовувані в наявних статичних і динамічних дисциплінах, дають усереднені та ймовірнісні оцінки параметрів функціонування системи, які не завжди доцільно застосовувати як критерії, оскільки в певних ситуаціях можуть бути неприпустимі втрати транзакцій у процесі обробки або затримки видачі результатів. Лінійність самих критеріїв систем обслуговування, за якими проводиться оптимізація, не дозволяє враховувати ситуацію в поточний момент часу, внаслідок чого неможливо визначити зміну характеристик окремих транзакцій у часі.

Раціональна обробка вступників у систему транзакцій має враховувати такі характеристики транзакцій, як цінність і старіння, дозволяючи обробляти транзакцію без втрат, із найбільшою цінністю у дотриманні критерію старіння інформації. Для цього необхідно застосувати керовану дисципліну обслуговування потоків інформації в інформаційно-обчислювальних комплексах. Організована з її допомогою обробка потоків транзакцій дозволяє враховувати цінність і старіння інформації. Для розгляду вищезазначеної дисципліни обслуговування висунемо основні положення про структурні й функціональні властивості обчислювальної системи.

Уявімо обчислювальну систему у вигляді графа, вершинами якого є множина складових елементів системи $X = \{x_1, x_2, \ldots, x_n\}$, а ребрами – зв'язки між елементами в процесі функціонування. Кожен елемент множини генерує транзакції, які потребують виконання відповідного інформаційного процесу, що складається з певної сукупності операцій: оновлення, вставка, видалення, зберігання, подання. За допомогою транзакцій елементи множини взаємодіють між собою. За заданим критерієм відбувається класифікація транзакцій на типи. Вся номенклатура операцій інформаційного процесу має виконуватись у рамках вимог, які сформульовано в загальному вигляді:

– величина старіння транзакції *i*-го типу має бути мінімальною;

– старіння транзакції *i*-го типу на момент часу закінчення обробки не має перевищувати допустимої для даного типу величини старіння;

– втрата цінності транзакції *i*-го типу на момент часу закінчення обробки не має перевищувати допустимого значення для даного типу;

– цінність транзакції *i*-го типу має бути наближена до максимального значення, визначеного для даного типу.

Транзакції, що генеруються елементами множини *X*, характеризуватимемо інтенсивністю $\Lambda^{[X_5^{\xi}]}$, складовими частинами котрої виступають інтенсивності потоків транзакцій різних типів.

Для виконання операцій обробки потоків транзакцій певної інтенсивності в рамках вищезазначених вимог окремі елементи множини *X* мають володіти певною потрібною продуктивністю, яка прямо залежить від капітальних та експлуатаційних витрат. Під час функціонування системи інтенсивність транзакцій у вузлах їх генерації може

змінюватись, причому характер змін може бути як регулярним, так і випадковим. Це призводить до зміни в часі значень потрібної продуктивності як окремих елементів, так і всієї системи в цілому. У такій ситуації типовою практикою є вибір елементів системи з великою продуктивністю, необхідною в середньому для функціонування, але яка могла б забезпечити виконання операцій інформаційного процесу в моменти максимальних значень $\lambda_i^{[X_\xi]}$.

Фактичну продуктивність слід обирати нижчого рівня, а необхідні операції інформаційного процесу за пікових завантажень системи здійснюватимуться під час виконання певних маневрів з розподілу ресурсів системи в часі. Самі пікові значення інтенсивностей "згладжуватимуться", знижуючи свої значення. Очевидно, що ефективність функціонування окремого вузла системи в будь-який момент часу максимальна, якщо величина його потрібної продуктивності наближена до величини фактичної продуктивності. Тоді в певний момент часу для отримання максимальної ефективності функціонування всієї системи необхідно мінімізувати за всіма елементами системи різницю між фактичною продуктивністю, встановлюючи правила обробки інформації з урахуванням цінності та старіння. Стратегію управління обробкою транзакцій рекомендується вибирати на певних відрізках часу функціонування системи, варіюючи при цьому розподілом ресурсів як між потоками транзакцій $\lambda_i^{[X_\xi]}$ у межах одного елемента множини *X*, так і між потоками $\lambda_i^{[X_\xi]}$ у межах усієї системи. На ефективність управління певною мірою впливає значення фактичної продуктивності елементів, яку слід обрати так, щоб із застосуванням окремих процедур і засобів управління вона дозволяла виконувати необхідні операції в рамках заданих вимог. З іншого боку, фактична продуктивність не має бути надмірною за тих же умов, оскільки це призводить до збільшення капітальних і експлуатаційних витрат. Для випадку, коли потоки транзакцій регулярні й мають незначні коливання, це завдання легко виконати.

Змінювати величину фактичної продуктивності системи в процесі її функціонування для полегшення управління можна, але це пов'язано з перебудовою відносин елементів множини *X*, заміною технічних засобів та іншими операціями, що в більшості випадків зробити важко. Тому доцільно вибирати значення фактичної продуктивності обчислювальної системи на етапі проектування, враховуючи прогнозні оцінки розвитку системи, і маневрувати цим параметром тільки в крайньому разі.

Для забезпечення обґрунтованого вибору фактичної продуктивності на етапі створення системи та управління нею в подальшому розроблено метод раціональної організації обробки транзакцій. Метод включає моделі, що дозволяють у конкретних режимах і умовах функціонування системи визначати різні параметри, а також набір спеціальних моделей, алгоритмів і процедур управління обробкою інформаційних потоків у рамках заданих вимог залежно від певного набору вхідних параметрів. Цей метод використовується як інструмент для вибору і дослідження основних характеристик системи, а також дозволяє знаходити в процесі функціонування необхідні рішення щодо вибору стратегії управління в разі зміни умов функціонування. Розглянуту систему можна зарахувати до класу адаптивних або саморегулівних.

Для випадку регулярних і квазірегулярних вхідних інформаційних потоків обробку транзакцій в одноканальній обчислювальній системі з очікуванням пропонується здійснювати відповідно до керованої дисципліною обслуговування потоків інформа-

ційну систему, у яку надходить потік інформації, складовими якого виступають потоки транзакцій різних типів. Стан загального потоку інформації в будь-який момент часу T характеризується вектором $\Lambda_t = \{\Lambda_{1t}, \Lambda_{2t}, ..., \Lambda_{it}, ..., \Lambda_{ht}\}$, як Λ_{it} виступають інтенсивності складових потоків транзакцій.

Тип транзакції визначається, виходячи з характеристик цінності та старіння інформації, що міститься в конкретній транзакції. Уявімо процес обслуговування потоків інформації у вигляді часової діаграми, яка дозволяє графічно простежити динаміку системи. На діаграмі (рис. 3) кожен потік забезпечений двома горизонтальними осями: верхня відповідає черзі очікування, нижня – диспетчеру обслуговування. Транзакції складових потоків з'являються на ділянці часу Δt через інтервали часу $t_{ц.г.i.}$ і вибудовуються залежно від типу в окремі черги. Величина $t_{ц.г.i.}$ – час циклу генерації транзакції i-го типу. Попередньо для кожної черги на підставі якісних характеристик інформації визначається пороговий час обробки $T_{порог.i.}$.

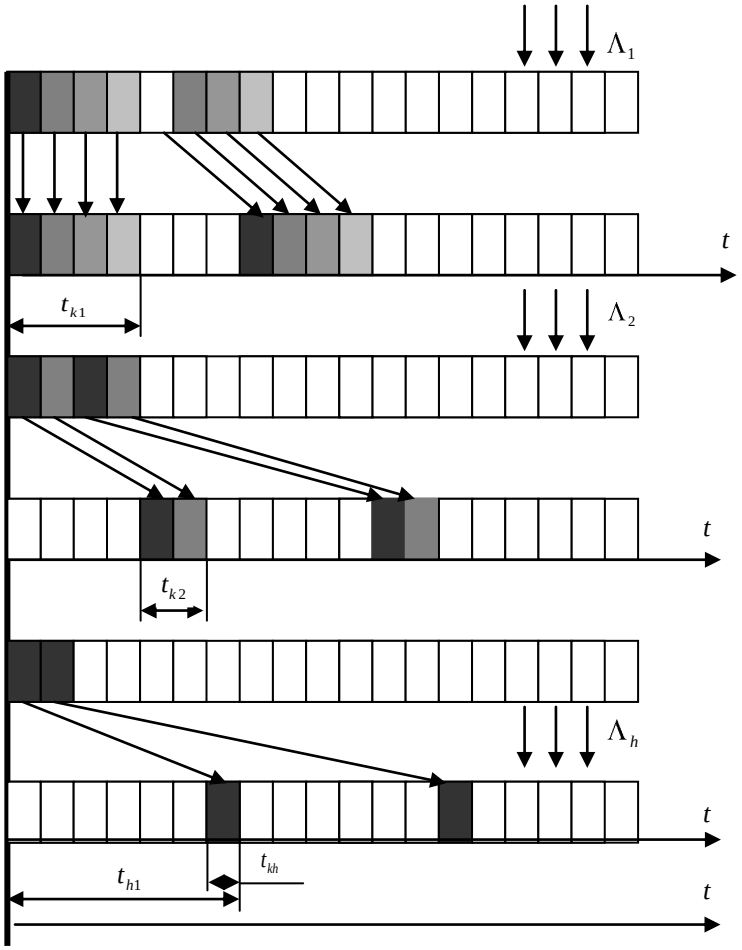

Рис. 3. Діаграма обслуговування регулярних вхідних потоків інформації

Система обробляє транзакції, послідовно надаючи кожній *i*-й черзі квант часу t_{ki}, за який можна обробити одну або декілька транзакцій. Кількість таких транзакцій визначається як t_{ki} / b_i та для кожної черги може бути різним. Величина b_i визначає відведений для *i*-ї черги час обробки:

$$T_t = \{t_{k1}, t_{k2}, ..., t_{ki}, ..., t_{kn}\}.$$

Час, проведений транзакцією в системі, складається з часу очікування і часу обслуговування, де $B_i(t)$ – функція розподілу часу обробки транзакції:

$$b_i = \int_0^\infty t dB_i(t).$$

Дана модель обробки транзакції може бути регульованою. Зі зміною інтенсивності потоків транзакції як керуючий вектор використовується вектор квантів часу.

До найважливіших характеристик обробки належить функція часу очікування:

$$W_i(t) = f(\overline{T}_k, \overline{\Lambda}, \overline{B}),$$

де $\overline{T}_k$ – вектор квантів часу;

$\overline{\Lambda}$ – вектор інтенсивностей інформаційних потоків;

$\overline{B}$ – вектор розподілу часів обробки транзакцій.

Величина $W_i(t)$ – це час, який витратять на очікування початку обробки транзакції *i*-ї черги, що надійшли в момент часу *t* ($t = 0$ відповідає початку процесу обробки). Значення $W(t)$ розраховується для кожного елемента, що надійшов в систему транзакції. Головний критерій обробки: $W_i(t) < T_{порог\ i}$, тобто обробка кожної транзакції ведеться тільки у встановлених для нього кордонах часового інтервалу. Ця умова досягається шляхом перерозподілу ресурсів обчислювальної системи на "перевантажені" черги: збільшення часу квантування завантажених черг шляхом зменшення квантів часу недовантажених.

Очевидно, що $T_{порог\ i}$ має призначатися з урахуванням цінності та старіння, тому інформацію слід певним чином класифікувати. На різних рівнях управління необхідно застосовувати типові системи класифікації для різних видів інформації. Питання класифікації є темою окремого дослідження і в рамках цієї статті не розглядається, а йдеться лише про те, що інформація класифікована й для кожного потоку інформації обрано величину $T_{порог\ i}$.

Розглянуті методи зручно застосовувати для організації раціональної обробки інформації в системах, вхідні інформаційні потоки яких детерміновані. Однією зі сфер застосування виступають системи, в яких поточний стан керованого об'єкта (процесу) характеризується даними, що надходять у сервер БД через певні проміжки часу.

Основна ідея методу полягає у створенні шарового адресного простору (ШАП) бази даних, який дозволяє використовувати одне й те саме подання даних як у зовнішній, так і в оперативній пам'яті. Це й дозволяє реалізувати операцію переходу за покажчиком дуже ефективно, оскільки немає витрат, пов'язаних з трансляцією адрес.

Фактично ШАП реалізує віртуальну пам'ять СКБД на основі віртуальної пам'яті операційної системи. Однак застосування версійного механізму спільно із ШАП може призводити до втрати ефективності операції переходу за покажчиками.

Якщо транзакції блокують одна одну, слід застосувати метод, який дозволяє знімати сеанси користувачів, транзакції котрих блокують одна одну. За допомогою інформації з подання, до якого має доступ адміністратор бази даних, ми можемо отримати інформацію про час початку транзакції, користувача, в сеансі якого розпочалася ця транзакція, програмний додаток, який згенерував цю транзакцію, тощо. Якщо тайм-аут t_{ws} перевищує заданий ліміт можна визначити, яка з транзакцій має найнижчий пріоритет, і зняти сеанс користувача, що створив цю транзакцію. Для розв'язання цієї задачі на основі наявної інформації, отриманої з представлень бази даних, використовуватимемо нестаціонарну оптимізаційну модель:

$$M[Y(t_{ws}, OSuser_n, PR, Mk, Modul_n)] \rightarrow extr, \quad (1)$$

де t_{ws} – граничний час очікування;

$OSuser_n$ – користувач, у сеансі якого збуджується транзакція;

PR – програмний додаток, що збудив транзакцію;

Mk – комп'ютер, який використовує транзакцію;

$Modul_n$ – клас чи модуль, який у програмному додатку створив транзакцію;

M – символ математичного сподівання;

Y – функціонал, що описує роботу системи.

Для розв'язання цієї задачі пропонуємо такий метод.

Метод вилучення з циклу транзакції, яка блокує інші, працює так.

1. Спочатку знаходимо сесії, транзакції яких взаємно заблокували одна одну та перебувають у режимі очікування, t (задається адміністратором БД):

select waiting_session from dba_waiters.

Результатом запиту буде кортеж:

$$S(WS)=<ID_n, HS_n, MH_n, MR_n, LID_n>, \quad (2)$$

де ID_n – ідентифікаційний номер транзакцій, що перебувають у режимі очікування;

HS_n – ідентифікаційний номер транзакцій, що блокує інші транзакції;

MH_n – модель блокування;

MR_n – тип блокування;

LID_n – ідентифікаційний номер блокування.

2. Визначаємось із транзакціями, які заблоковані та перебувають у стані очікування:

*select * into tr from dba_blockers.*

Результатом запиту буде кортеж:

$$S(Db)=<HS_n>, \quad (3)$$

де HS_n – список заблокованих транзакцій.

3. Визначимось, які із сеансів використовують транзакції, що перебувають у стані очікування:

select sid, SERIAL#, USERNAME, OSUSER, PROGRAM, MACHINE, module from v$session where sid in (Tr[1],TR[2],..Tr[n]);

$$\sigma_{sid \subset HSn}(VSTR)=\{(VSTR) \mid VSTR<sid> \subset DB<HSn>)\}. \quad (4)$$

Результатом запиту буде кортеж:

$$S(VSTR)=<SID_n,SER_n,OSuser_n,PR,Mk,Modul_n>. \quad (5)$$

4. З цього подання можна отримати інформацію про вузол, з якого почалась транзакція, програмний додаток, який згенерував цю транзакцію, час, коли почався сеанс. За допомогою методу, описаного в [5], знаходимо транзакцію, яка потрапила до циклу, має найменшу цінність і найстаріша. Знаходимо транзакцію з мінімальною цінністю.

$$\underset{OS,PR,Mk,Mod}{Lim} Cost(Tr(SIDn,SERn)) \to \min, \quad (6)$$

де *Cost* – функція, що вирішує, яка транзакція має найменшу цінність. Знаходимо, в якому із сеансів вона використовується.

5. Адміністратор БД, чи процедура, запущена від імені адміністратора, не може втручатись у хід транзакції. Проте для уникнення взаємоблокувань можна тимчасово зупинити сеанс:

$$alter\ system\ kill\ session\ `\ SID_n,\ SER_n\ ',$$

тобто вилучити найменш цінну транзакцію, яка блокувала інші.

Висновки з даного дослідження і перспективи подальших розвідок у даному напрямку. У результаті проведеного дослідження набув подальшого розвитку метод керування транзакціями за ознаками цінності та старіння інформації. За допомогою цього методу розроблено алгоритм, здатний підвищити пропускну спроможність OLTP-систем. Важливою особливістю є те, що розроблений метод може використовуватись одночасно із синхронними алгоритмами за умови, що на кожен об'єкт, який бере участь у транзакціях, буде виписано операції, для яких слід застосовувати асинхронну фіксацію в разі взаємоблокувань. Таким чином, у реальних системах виникає задача визначення оптимальної сфери застосування методу залежно від імовірності виникнення конфліктних ситуацій. Сфера застосування методу обмежена системами керування базами даних, які використовують мову PL\SQL, і не може бути істотно розширена за умови збереження транзакційності (дискретності відносного часу в системах) запитів до СКБД.

У подальшому доцільно розробити програмний додаток, який можна використовувати для пошуку та зняття транзакцій, що взаємоблокуються, без втручання адміністратора БД у процес пошуку та зняття транзакцій.

Список використаних джерел:

1. Bernstein P. A. Multiversion concurrency control – theory and algorithms / P. A. Bernstein, N. Goodman // ACM Transactions on Database Systems. – 1983. – Vol. 8. – № 4. – P. 465–483.

2. Beyer K. DB2 goes hybrid: Integrating XML and XQuery with relational data and SQL / K. Beyer // IBM Systems Journal. – 2006. – Vol. 45. – № 2.

3. The Implementation of an Integrated Concurrency Control and Recovery Scheme. Proc / A. Chan, S. Fox, W. Lin and oth. // SIGMOD Conference. – Orlando, 1982.

4. Chan A. Implementing Distributed Read-Only Transactions. IEEE Trans / A. Chan, R. Gray // On Software Eng. – 1985. – SE-11(2).

5. Мороз Б. І. Методи класифікації інформації для організації реплікацій в розподілених базах даних за ознаками цінності і старіння / Л. В. Кабак, Б. І. Мороз // Вісник Академії митної служби України. Серія: "Технічні науки". – 2010. – № 1 (43). – С. 86–92.