УДК 004.657 **doi: 10.26906/SUNZ.20204.1.073**

Т. В. Філімончук, В. О. Мартовицький, Д. В. Гонтарєва

Харківський національний університет радіоелектроніки, Харків, Україна

ТРАНЗАКЦІЇ І БЛОКУВАННЯ, РІВНІ ІЗОЛЬОВАНОСТІ ТРАНЗАКЦІЙ

Анотація. Предметом дослідження є ефективність використання рівнів ізольованості транзакцій. Метою даної статті є визначення доцільності використання розповсюджених рівнів ізольованості транзакцій, які впливають на блокування, для запобігання виникнення неузгодженості даних (втрачене оновлення, «брудне» читання, неповторюване читання, фантомне читання) при паралельному виконані транзакцій. Були отримані наступні **результати**. Одночасно може бути встановлений тільки один параметр рівня ізольованості транзакції, який продовжує діяти для поточного з'єднання до тих пір, поки не буде явно змінений. Коли для транзакції змінюється рівень ізоляції, ресурси, які зчитуються після зміни, захищаються відповідно до правил нового рівня. Ресурси, які зчитуються до зміни, залишаються захищеними відповідно до правил попереднього рівня. Системи управління базами даних, які забезпечують транзактивність, не завжди підтримують всі розглянуті чотири рівні ізольованості транзакцій, а також можуть вводити додаткові рівні. **Висновки**. Використовуючи високий рівень ізольованості (впорядкованість), можна захистити одну транзакцію від впливу іншої, але за рахунок істотного збитку для продуктивності бази даних. На цьому рівні результати паралельного виконання транзакцій для бази даних у більшості випадків можна вважати такими, що збігаються з послідовним виконанням тих же транзакцій (по черзі в будь-якому порядку). З іншого боку, низький рівень ізоляції (читання незафіксованих даних) транзакції породжує проблеми з неузгодженістю даних, забезпечуючи при цьому більш високу продуктивність.

Ключові слова: неузгодженість даних, транзакції, версіонування, блокування, рівні ізольованості транзакцій, читання незафіксованих даних, читання фіксованих даних, повторюваність читання, впорядкованість.

Вступ

Постановка проблеми у загальному вигляді. На сьогоднішній день комп'ютерні системи використовуються у все більш різноманітних областях та додатках. Вони стають поширенішими і виконують різноманітні завдання майже в усіх сферах нашого життя. Для того, щоб задовольнити всі наші потреби, комп'ютерні системи повинні зберігати великий обсяг інформації. Необхідно, щоб ця інформація зберігалася і була структурована таким чином, аби приносити користь і для самої системи, і, так само, для людей, які проектують та працюють з системою. Така структурована інформація називається даними і зазвичай її постійно зберігають в базі даних.

При роботі з базами даних (БД) постійно виникає необхідність підтримки даних в актуальному стані. Для цього необхідно здійснювати оновлення таблиць, що існують. Система управління базами даних (СУБД) послідовно здійснює обробку записів таблиць, які необхідно оновити. Але завжди присутній інтервал часу, коли частина записів вже містить нові значення, а частина – ще не оновлена. Також в процесі оновлення може виникнути ситуація, коли стан цілісності бази даних порушено, внаслідок того, що виник збій. Мережні, паралельні та розподілені бази даних обслуговують безліч користувачів, що працюють одночасно. При паралельному виконанні транзакцій можливі наступні неузгодженості даних:

- втрачене оновлення – при одночасній зміні одного блоку даних різними транзакціями втрачаються всі зміни, крім останньої;
- «брудне» читання – читання даних, які додані або змінені транзакцією, що згодом не підтвердиться (відкотиться);
- неповторюване читання – при повторному читанні в рамках однієї транзакції дані, які було раніше прочитано, виявляються зміненими;
- фантомне читання – одна транзакція в ході свого виконання кілька разів обирає безліч рядків за одними й тими ж критеріям. Інша транзакція в інтервалах між цими вибірками додає рядки або змінює стовпці деяких рядків, які використовуються в умовах вибірки першої транзакції, і успішно закінчується. В результаті вийде, що одні й ті ж вибірки в першій транзакції дають різні безлічі рядків.

Для того, щоб уникнути проблем, які виникають під час одночасного виконання транзакцій, використовуються блокування.

Мета статті є визначення оптимального ступеню забезпечення внутрішніми механізмами системи управління базою даних захисту від усіх або деяких видів неузгодженості даних, які виникають при паралельному виконані транзакцій.

Аналіз останніх досліджень і публікацій. Транзакція – це група послідовних операцій з базою даних, яка являє собою логічну одиницю роботи з даними [1]. Транзакція може бути виконана або цілком і успішно, дотримуючись цілісності даних і незалежно від інших транзакцій, що виконуються паралельно, або не виконана взагалі.

Проблеми, які виникають при виконані транзакцій запису та читання даних згадуються у декількох джерелах [1, 3, 8, 9] та викликають неузгодженість даних. Для того, щоб підтримувати узгодженість та ізоляцію даних, які є одними з головних характеристик транзакції, використовуються блокування, які були розглянуті в [3].

Читання за замовчуванням має на увазі вибірку даних з використанням спільного блокування. Запис же вимагає монопольного блокування, так як він призводить до зміни вмісту таблиці. Для того, щоб впливати на блокування, використовуються рівні ізоляції [2].

Виділяють чотири основні рівні ізольованості транзакцій, які наведено в [4].

Перший рівень – це мінімальний рівень ізоляції, який гарантує тільки фізичну цілісність при записі даних. В [3] його ще називають незавершеним (чорновим) читанням.

Наступний рівень називають читанням фіксованих даних, тому що він запобігає читанню «брудних» даних. На даний час більшість промислових СУБД, зокрема, Microsoft SQL Server, PostgreSQL та Oracle, за замовчуванням використовують саме цей рівень [6-8].

Третій рівень – це рівень ізоляції, при якому ніяка інша транзакція не може змінювати дані, які читаються поточною транзакцією, поки та не закінчена. При повторному запуску інструкції поточною транзакцією будуть вилучені нові рядки, що призведе до «фантомного читання», приклад якого наведено в [5].

На найвищому рівні ізольованості транзакції ізолюються одна від одної і виконуються так, ніби інших транзакцій не існує.

Версіонування та блокування

На даний час використовують наступні підходи до реалізації ізольованості: версіонування та блокування.

Версіонування – це збереження кількох версій рядків, які паралельно змінюють. При кожній зміні рядка СУБД створює нову версію цього рядка, з якою продовжує працювати змінюючи дані транзакція, в той час як будь-який іншій транзакції процесу-читання повертається остання зафіксована версія.

Перевага такого підходу полягає в тому, що він забезпечує більшу швидкість, так як запобігає блокуванню. Однак він вимагає більшої витрати оперативної пам'яті, яка витрачається на зберігання версій рядків. Крім того, при паралельній зміні даних декількома транзакціями може виникнути ситуація, коли кілька паралельних транзакцій виконують неузгоджені зміни одних і тих самих даних (оскільки блокування відсутнє, ніщо не завадить це зробити). Тоді та транзакція, яка зафіксована першою, збереже свої зміни в основній БД, а інші паралельні транзакції виявиться неможливо зафіксувати (так як це призведе до втрати поновлення першої транзакції). Єдине, що може в такій ситуації СУБД – це відкотити інші транзакції і видати повідомлення про помилку «Запис вже змінено».

Блокування даних – це підхід, який полягає в тому, що транзакція процесу письменника блокує дані, які було змінено, для транзакцій процесу читання. Таким чином, перешкоджається «брудне» читання, а дані, які було заблоковано транзакцією процесу читання, звільняюся відразу після завершення операції над ними [3].

Об'єктом блокування може бути БД цілком, окрема таблиця, фрагмент таблиці, запис (рядок) або осередок.

По області дії блокування поділяються на рядкове, гранулярне та предикатне [1].

Рядкове блокування – діє тільки на один рядок таблиці БД, не обмежуючи маніпуляції над іншими рядками таблиці;

Гранулярне блокування – діє на всю таблицю або всю сторінку та всі рядки. Блокування, що обмежує маніпуляції зі сторінкою даних в таблиці іноді називається сторінковим.

Предикатне блокування діє на область, яка обмежена предикатами. Зазвичай це блокування по діапазону ключів. При такому блокуванні для ключа або індексу вказується значення або діапазон значень, на які поширюється блокування.

За суворістю блокування поділяються на спільне та монопольне [10].

Спільне блокування – накладається транзакцією на об'єкт у разі, якщо операція, що була виконана, безпечна. Тобто не змінює ніяких даних і не має побічних ефектів. При цьому, всі транзакції можуть виконувати операцію того ж типу над об'єктом, якщо на нього накладено спільне блокування, зазвичай таке блокування використовується для операцій читання.

Монопольне блокування – накладається транзакцією на об'єкт у разі, якщо операція, що була виконана, змінює дані. Тільки одна транзакція може виконувати подібну операцію над об'єктом, якщо на нього накладено монопольне блокування. Блокування не може бути накладенео на об'єкт, якщо на нього вже накладено спільне блокування.

За логікою реалізації блокування поділяються на оптимістичне та песимістичне [3].

Оптимістичне блокування – не обмежує модифікацію даних, які було оброблено сторонніми сесіями, однак перед початком модифікації запитує значення деякого виділеного атрибуту кожного з рядків даних (зазвичай використовується найменування VERSION та цілочисельний тип з ініціальним значенням 0). Перед записом модифікацій в базу даних перевіряється значення виділеного атрибуту, і якщо воно змінилося, то транзакція відкочується або застосовуються різні схеми дозволу колізій. Якщо значення виділеного атрибуту не змінилося – проводиться фіксація модифікацій з одночасною зміною значення виділеного атрибуту для сигналізації іншим сесіям про те, що дані змінилися.

Песимістичне блокування накладається перед модифікацією даних, яка є передбаченою, на всі рядки, які така модифікація імовірно зачіпає. Весь час дії такого блокування виключена модифікація даних зі сторонніх сесій, дані з блокованих рядків доступні відповідно до рівня ізольованості транзакції. По завершенню модифікації, що є передбаченою, гарантується несуперечливий запис результатів.

Рівні ізольованості транзакцій

Під «рівнем ізоляції транзакцій» розуміється ступінь, що забезпечується внутрішніми механізмами СУБД захисту від усіх або деяких видів перерахованих вище неузгодженостей даних, що виникають при паралельному виконанні транзакцій. Стандарт SQL-92 [2] визначає шкалу з чотирьох рівнів ізоляції: read uncommitted, read committed, repeatable read, serializable. Перший з них є найслабшим, останній – найсильнішим, кожний наступний включає в себе всі попередні.

Перший рівень (read uncommitted) – це мінімальний рівень ізоляції, гарантує тільки фізичну цілісність при записі даних. Цей рівень, має найгіршу узгодженість даних, але найвищу швидкість виконання транзакцій. Назва рівня говорить сама за себе – кожна транзакція бачить незафіксовані зміни іншої транзакції (феномен «брудного» читання) [4].

Типовий спосіб реалізації даного рівня ізоляції – це блокування даних на час виконання команди зміни, яке гарантує, що команди зміни одних і тих же рядків, які запущено паралельно, фактично виконуються послідовно, і жодна з змін не загубиться. Транзакції, що виконують тільки читання, при даному рівні ізоляції ніколи не блокуються. Процеси-читачі можуть зчитувати дані незавершеної транзакції процесу письменника.

Другий рівень (read committed) – це рівень ізоляції, який вказує, що інструкції не можуть зчитувати дані, які були змінені іншими транзакціями, але ще не були зафіксовані. Наведений рівень ще називають читання фіксованих даних, тому що він запобігає читанню «брудних» даних. Дані можуть бути замінені іншими транзакціями між окремими інструкціями в поточній транзакції, результатом чого буде повторюване читання або фантомні дані.

Рівень ізоляції (repeatable read) – це рівень, при якому транзакція, що читає, не бачить зміни даних, які були нею раніше прочитані. При цьому ніяка інша транзакція не може змінювати дані, які читаються поточної транзакцією, поки та не закінчена [5].

Спільне блокування застосовуються до всіх даних, які зчитуються будь-якою інструкцією транзакції, і зберігаються до її завершення. Це забороняє іншим транзакціям змінювати рядки, які зчитуються поточною транзакцією. Інші транзакції можуть вставляти нові рядки, що відповідають умовам пошуку інструкцій, які містяться в поточної транзакції. При повторному запуску інструкції поточною транзакцією будуть вилучені нові рядки, що призведе до «фантомного читання».

Рівень впорядкованості (serializable) – найвищий рівень ізольованості, тому що транзакції повністю ізолюються одна від одної, кожна виконується так, як ніби паралельних транзакцій не існує. Тільки на цьому рівні паралельні транзакції не схильні до ефекту «фантомного читання» [5].

Блокування діапазону встановлюється в діапазоні значень ключа, що відповідає умовам пошуку будь-якої інструкції, яка була виконана під час транзакції. Оновлення та вставка рядків, які відповідають інструкціям поточної транзакції, блокується для інших транзакцій. Це гарантує, що якщо будь-яка інструкція транзакції виконується повторно, вона буде зчитувати той же самий набір рядків. Блокування діапазону зберігається до завершення транзакції. Це найсуворіший рівень ізоляції, оскільки він блокує цілі діапазони ключів та зберігає блокування до завершення транзакції. Через низький паралелізм цей рівень рекомендується використовувати тільки при необхідності.

У таблиці нижче показано, від яких побічних ефектів кожен з чотирьох рівнів ізоляції страждає (від проблем паралелізму, які було перераховано раніше). Значення "Так" в таблиці означає, що певна проблема можлива при даному рівні ізоляції, а значення "Ні" – що проблема на даному рівні ізоляції неможлива.

Таблиця 1 – **Поведінка при різних рівнях ізольованості**

Рівень ізоляції	**Фантомне читання**	**Неповторюване читання**	**«Брудне» читання**	**Втрачене оновлення**
Serializable	Ні	Ні	Ні	Ні
Repeatable read	Так	Ні	Ні	Ні
Read committed	Так	Так	Ні	Ні
Read umcommitted	Так	Так	Так	Ні

Одночасно може бути встановлено тільки один параметр рівня ізоляції, який продовжує діяти для поточного з'єднання до тих пір, поки не буде явно змінений. Всі операції зчитування, які було виконано в рамках транзакції, функціонують відповідно до правил рівня ізоляції.

Рівні ізоляції транзакції визначають тип блокування, який застосовується до операцій зчитування. Спільні блокування, що застосовуються для read committed або repeatable read, як правило, є блокуваннями рядків, але при цьому, якщо в процесі зчитування йде звернення до великої кількості рядків, блокування рядків може бути розширене до блокування сторінок або таблиць. Якщо рядок був змінений транзакцією після зчитування, для захисту такого рядка транзакція застосовує монопольне блокування, яке зберігається до завершення транзакції. Наприклад, якщо транзакція repeatable read має спільне блокування рядків і при цьому змінює його, таке блокування перетворюється в монопольне. У будь-який момент транзакції можна переключитися з одного рівня ізоляції на інший.

Коли для транзакції змінюється рівень ізоляції, ресурси, які зчитуються після зміни, захищаються відповідно до правил нового рівня. Ресурси, які зчитуються до зміни, залишаються захищеними відповідно до правил попереднього рівня. Наприклад, якщо для транзакції рівень ізоляції змінюється з read committed на serializable, то спільні блокування, які отримано після зміни, будуть утримуватися до завершення транзакції.

СУБД, які забезпечують транзакційність, не завжди підтримують всі чотири рівні, а можуть вводити додаткові.

Можливі також різні нюанси в забезпеченні ізоляції.

Так, Oracle [7] в принципі не підтримує нульовий рівень, так як його реалізація транзакцій виключає «брудні читання», і формально не дозволяє встановлювати рівень repeatable read. Тобто Oracle підтримує тільки рівні read committed (за замовчуванням) та serializable. При цьому на рівні окремих команд він, фактично, гарантує повторюване читання (якщо команда SELECT в першій транзакції обирає з бази набір рядків, і в цей час паралельна друга транзакція змінює якісь з цих рядків, то результую-

чий набір, який отримано першою транзакцією, буде містити незмінені рядки, як ніби другий транзакції не було). Також Oracle підтримує так звані read-only транзакції, які відповідають serializable, але при цьому не можуть самі змінювати дані.

Microsoft SQL Server підтримує всі чотири стандартних рівня ізоляції транзакцій та додатково – рівень snapshot [6], на якому транзакція бачить той стан даних, який було зафіксовано до її запуску, а також зміни, які було внесено нею самою. Тобто поводиться так, начебто отримала при запуску моментальний знімок даних БД і працює з ним. Відмінність від serializable полягає в тому, що не використовуються блокування, але в результаті фіксація змін може перестати працювати, якщо паралельна транзакція змінила ті ж самі дані раніше; в цьому випадку друга транзакція при спробі виконати commit викличе повідомлення про помилку і буде скасована.

Висновки

В цій статті було описано процес управління ступенем узгодженості даних за допомогою вибору різних рівнів ізоляції і те, як цей вибір впливає на паралельний доступ. Було розглянуто чотири режими, в яких використовуються блокування.

Незавершене (чорнове) читання – це мінімальний рівень ізоляції який гарантує тільки фізичну цілісність при записі даних. Цей рівень, має найгіршу узгодженість даних, але найвищу швидкість виконання транзакцій.

Читання фіксованих даних – це рівень ізоляції, який вказує, що інструкції не можуть зчитувати дані, які були змінені іншими транзакціями, але ще не були зафіксовані.

Третій рівень використовується за замовчуванням в MySQL. Він відрізняється від другого рівня тим, що знову додані дані вже будуть доступні в транзакції, але не будуть доступні до підтвердження ззовні.

Рівень впорядкованості – найсуворіший рівень ізоляції, оскільки він блокує цілі діапазони ключів та зберігає блокування до завершення транзакції. Через низький паралелізм цей рівень рекомендується використовувати тільки при необхідності.

Реальні бази даних потребують застосування компромісу між паралельним доступом і серіалізованими режимами операцій. Ключовий момент тут полягає в тому, що, використовуючи високий рівень ізоляції, можна захистити одну транзакцію від впливу іншої, але за рахунок істотного збитку для продуктивності бази даних. З іншого боку, низький рівень ізоляції транзакції породжує проблеми з даними, описані раніше в цій статті, забезпечуючи при цьому більш високу продуктивність.

СПИСОК ЛІТЕРАТУРИ

1. Фиайли К. SQL. Москва: ДМК Пресс, 2013, 456 с.
2. Connolly T., Begg C. Database Systems A Practical Approach to Design, Implementation, and Management: Global Edition. Boston: Pearson Education, 2014, 1440 p.
3. Elmasri R., Navathe S.B. Fundamentals of Database Systems. Boston: Addison Wesley, 2016, 1272 p.
4. Кен Хендерсон, Профессиональное руководство по SQL Server. Структура и реализация. Москва: Вильямс, 2006, 1056 с.
5. Coronel C., Morris S. Database system: Design, Implementation, & Management: 13th Edition. Boston: Cengage, 2019, 802 p.
6. Davidson L., Moss J. Pro SQL Server Relational Database Design and Implementation: Fifth Edition. New York: Apress, 2012, 791 p.
7. Pavlovic Z., Veselica M. Oracle Database 12c Security Cookbook. Birmingham: Packt publishing, 2016, 358p.
8. Korotkevitch D. Expert SQL Server Transactions and Locking. Birmingham: Packt publishing, 2018, 320 p.
9. Nechausov A., Mamusuĉ I., Kuchuk N. Synthesis of the air pollution level control system on the basis of hyperconvergent infrastructures. *Сучасні інформаційні системи*. 2017. Т. 1, № 2. С. 21 – 26. DOI: https://doi.org/10.20998/2522-9052.2017.2.04
10. Гайдаржи В., Ізварін І. Бази даних в інформаційних системах. Київ: Університет «Україна», 2018, 418 с.

Received (Надійшла) 19.10.2020
Accepted for publication (Прийнята до друку) 11.11.2020

Transactions and locks, transaction isolation levels

T. Filimonchuk, V. Martovytskyi, D. Hontarieva

Abstract. The subject of the study is the effectiveness of the use of transaction isolation levels. The purpose of this publication is to determine the feasibility of using common levels of transaction isolation that affect blocking, to prevent data inconsistencies (lost updates, "dirty" reading, non-repetitive reading, phantom reading) in parallel transactions. The **results** were obtained. Only one transaction isolation level parameter can be set at a time, which remains valid for the current connection until it is explicitly changed. When the isolation level for a transaction changes, the resources read after the change are protected according to the rules of the new level. Resources that are read before the change remain protected according to the rules of the previous level. Database management systems that provide transactivity do not always support all four levels of transaction isolation and may introduce additional levels. **Conclusions.** Using a high level of isolation (order), you can protect one transaction from the impact of another, but at the expense of significant damage to database performance. At this level, the results of parallel transactions for the database in most cases can be considered to coincide with the sequential execution of the same transactions (alternately in any order). On the other hand, the low level of isolation (reading of unrecorded data) of the transaction creates problems with data inconsistency, while providing higher performance.

Keywords: data inconsistency, transactions, versioning, blocking, levels of transaction isolation, reading of unfixed data, reading of fixed data, repeatability of reading, orderliness.