УДК 004.03

С.М. Кравчук
Є.А. Паламарчук

ТРАНЗАКЦІЇ В СУЧАСНИХ СИСТЕМАХ

Вінницький національний технічний університет

Анотація
У даній роботі розглянуто поняття транзакції. особливості та форми її використання у сучасних системах.
Ключові слова: транзакція, атомарна операція, транзакційна система, база даних, модель транзакцій.

Abstract
In this paper the concept of transaction, features and forms of its usage in modern systems has been considered.
Keywords: transaction, atomic operation, OLTP-system, database, transaction model.

Вступ

У загальному випадку, термін транзакція можна трактувати наступним чином - це певна група операцій, які виконуються послідовно. Транзакція має два кінцевих стани: перший - всі операції транзакції успішно виконані та зміни збережені, другий - під час виконання однієї з операцій транзакції виникла помилка та внесені зміни відміняються. При роботі з базами даних, операціями транзакції є інструкції (наприклад SQL-запити) [1].

Метою статті є розгляд сучасних підходів до реалізації транзакцій в базах даних.

Основна частина

Транзакції обробляються так званими транзакційними системами. Транзакційна система (OLTP – Online transaction processing) – обробка транзакцій в реальному часі. Дана система створює журнал транзакцій, за допомогою якого і можливе зберігання станів системи чи бази даних. Структура даного журналу може бути різною, однак головною рисою є деталізований запис про внесені зміни.

Для транзакцій існує набір властивостей, забезпечуючи які, гарантується цілісність системи незважаючи на помилки, проблеми мережі чи збій живлення. Даний набір відомий під акронімом ACID, який в 1983 році запропонували Андреас Ройтер та Тео Хердер [2], на основі роботи Джима Ґрея про концепцію транзакцій [3]. У загальному випадку кожну властивість можна трактувати як:

1. Атомарність (Atomicity) - транзакція, яка, зазвичай, містить декілька інструкцій має бути виконана, як одна операція та завершувалась збереженням усіх змін або залишала базу даних без змін. Відповідно якщо одна інструкція в транзакції завершилась помилкою, то всі інші зміни не зберігаються;
2. Узгодженість (Consistency) - гарантія того, що транкзакція переводить базу даних лише з одного допустимого стану в інший та зберігати лише допустимі дані, які відповідають вимогам, каскадам, тригерам. Це запобігає пошкодження бази даних неправильною транзакцією, але не гарантує, що внесені зміни коректні;
3. Ізольованість (Isolation) - будь-які зміни всередині однієї транзакції повинні бути невидимими для інших транзакцій, які працюють паралельно, до моменту її завершення;
4. Довговічність (Durability) - гарантія того, що зміни успішної транзакції будуть збережені, навіть у випадку відмови системи внаслідок внесених змін. Відповідно неправильні зміни можна усунути виконавши додаткову транзакцію.

Ці чотири властивості є основними гарантіями парадигми транзакцій, яка вплинула на багато аспектів розвитку систем баз даних.

Ізольованість транзакцій, являє собою синхронізацію, відповідно її реалізація дороговартісна на ресурси, тому в реальних базах даних існують режими роботи, які в різній степені ізолюють транзакції. Кожна СУБД має свій перелік рівнів ізольованості транзакцій, в таблиці 1 наведено чотири основних рівні та відповідні проблеми, які вони вирішують [4].

Таблиця 1 – Рівні ізоляції транзакцій

Проблеми	Рівні ізоляції			
	Read uncommitted	Read committed	Repeatable read	Serializable
Втрачене оновлення	неможливо	неможливо	неможливо	неможливо
"Брудне" читання	можливо	неможливо	неможливо	неможливо
Неповторюване читання	можливо	можливо	неможливо	неможливо
Читання фантомів	можливо	можливо	можливо	неможливо

Деякі СУБД також вводять додаткові рівні, наприклад SQL Server має рівень Snapshot, який за переліком покриття проблем відповідає Serializable. Їх відмінність полягає в тому що рівень Snapshot базується на оптимістичному рівні паралелізму, що дозволяє ефективніше за рахунок меншого рівня узгодженості. Рівень Serializable навпаки базується на песимістичному рівні паралелізму, що гарантує узгодженість [5].

Зазвичай використання транзакцій тісно пов'язано зі реляційними базами даних. БД даного класу, які ми маємо на сьогодні, були спроектовані та оптимізовані для OLTP. Наприклад популярні реляційні СУБД: PostreSQL, MySQL, SQL Server, Oracle Database та інші. Вони включають функції, які необхідні для підтримки надійності та безпеки для даних. До них входять вищезгадані транзакції з ACID, привілеї доступу до даних, методи множинного доступу, глибина деталізації блокування даних, багатоетапні транзакції, реплікація даних, реєстрація кроків транзакцій, відновлення бази даних до певного пункту в часі тощо. Однак реляційні бази мають недолік, який полягає в тому, що при горизонтальному масштабуванні виникають проблеми, це пов'язано з структурою даних.

На сьогоднішній день, великої популярності та поширення набули бази даних NoSQL. Відповідно до назви, вони пропонують відмінні від реляційних структури баз даних, наприклад документо-орієнтована, ключ-значення, на основі графів, пошукові БД та інші. Дані системи мають гнучку продуктивність, яка зазвичай є більшою ніж в реляційних БД та підтримують автоматизоване горизонтальне масштабування.

Більшість NoSQL-систем відмовляються від реалізації ACID і реалізують свій унікальний набір гарантій збереження цілісності системи. Ці гарантії можуть бути, як досить віддаленими, так і близькими до ACID. Як результат, було створено нову модель транзакцій, щоби відповідати властивостям NoSQL-систем [6]. Дана модель відома під акронімом BASE, який означає три поняття:

1. Доступність (Basically Available) – гарантія того, що дані будуть завжди доступні. Наприклад, це можна досягнути за допомогою створення копій, які поширені на всі вузли кластерів.

2. М'який стан (Soft State) – через відсутність безпосереднього рівня узгодженості, значення даних можуть змінюватись з часом. Забезпечення узгодженості відводиться розробнику.

3. Можлива узгодженність (Eventually Consistent) – гарантія того, що в певний момент в майбутньому дані приймуть узгоджений стан. Однак поки це не відбудеться, зчитування даних все ще можливо (навіть якщо вони можуть не відображати реальний стан).

В основі NoSQL не закладалось поняття транзакційних систем, дані системи вирішують іншу інженерну задачу - ефективне масштабування та роботу з кластерами. Однак, на прикладі NoSQL продукту – RavenDB, як документо-орієнтованої бази даних, можна побачити реалізацію різнорівневих транзакцій, які є повністю відповідають ACID. Підтримуються транзакції на рівні одного документу, для певної групи документів на одному сервері, розподілені транзакції для кластерів [7]. У більшості випадків транзакції з декількома документами несуть більші витрати на продуктивність порівняно із записом для одного.

Важливим аспектом роботи баз даних є можливість масштабування. Як було згадано, існує два типи масштабування: горизонтальне та вертикальне. Під першим розуміють збільшення обчислювальних ресурсів для інфраструктури, за рахунок додавання більшої кількості обчислювальний блоків (наприклад серверів), для вертикального масштабування – за рахунок додавання більшої потужності до існуючого обладнання (наприклад центральний процесор, оперативна пам'ять). Зазвичай, для баз даних при горизонтальному масштабуванні виконується розділення даних між мережею пов'язаних вузлів, тобто кожен вузол зберігатиме лише частину даних.

У загальному випадку можна виділити наступні переваги та недоліки для двох типів масштабування: горизонтальне – дороговартісне, однак надає гнучкість при масштабуванні потужностей, які використовують на даний момент, при роботі з оновленнями та під навантаженням, адже при відключені одного вузла, навантаження буде розподілено між іншими; вертикальне – відносно дешеве, однак має жорстке обмеження в масштабуванні потужностей, що також може викликати простої. Також характерний низький рівень гнучкості та стійкості системи, при відключені єдиного вузла, вимикається вся система.

Розбиття великих наборів даних на менші та наступний розподіл наборів даних та навантаження від запитів на ці набори даних є необхідними умовами для високої масштабованості бази даних. Для баз даних існує дві моделі розбиття даних на менші частини: секціонування (partitioning) та сегментування (sharding). За допомогою секціонування, набір даних розділяється на менші частини в межах одного вузла за допомогою певних стратегій, наприклад: за інтервалами (range partitioning), за списком значень (list partitioning), за хеш-функцією (hash partitioning), за ключами (key partitioning). Сегментування працює схожим чином, однак сформовані частини даних розподіляються між декількома вузлами, відповідно дана модель є засобом для горизонтального масштабування.

Потрібно відмітити, що ці поняття можуть дещо перекривати одне одного. На прикладі програмного забезпечення для Oracle Database під назвою Oracle RAC, яке дозволяє створити менші частини даних та розподілити їх між декількома вузлами. Однак це не є випадком сегментування, через те. що Oracle RAC базується на архітектурі спільного дискового простору (shared-disk architecture), а сегментування пов’язане зі архітектурою без спільних елементів системи (shared-nothing architecture), де кожен вузол має приватну оперативну пам’ять та дисковий простір [8].

Шляхи оптимізації

Вище було наведено окремі випадки систем зі використанням транзакцій, які наділені своїми перевагами та недоліками. Однак сучасні реалії вимагають швидкодіючих транзакцій, як в реляційних БД, так і ефективну масштабованість, як в NoSQL. Відповіддю на таку інженерну задачу постають бази даних NewSQL [9]. Даний клас БД вирішує головні проблеми пов’язані зі традиційними транкзакційними системами. Також NewSQL здатні оброблювати дані з складною структурою [10, 11]. Іншими словами, NewSQL – це системи реляційних баз даних, що поєднують OLTP, підвищену продуктивність та масштабованість NoSQL. Основні категорії систем Newsql включають нові архітектури, прозоре проміжне програмне забезпечення для шардингу, двигуни SQL та Database-as-a-Service. Прикладами можуть бути: Amazon Relational Database Service, Google Spanner, MySQL Cluster with NDB.

Висновок

Таким чином, розглянуто поняття транзакції, її властивості та процеси її обробки. Розглянуто дві найбільш популярні моделі транзакцій баз даних – ACID та BASE, які забезпечують узгоджену систему та систему високої доступності відповідно.

Системи в різній степені реалізовують гарантії збереження цілісності, через те, що в залежності від потреб, реалізують відповідні моделі та шари роботи транзакцій, які у свою чергу мають різну глибину узгодженості, адже високі рівні узгодженості потребують більшу кількість ресурсів. Наприклад, транзакції на рівні документа в документо-орієнтованій базі даних, які гарантують забезпечення узгодженості в межах даного документу, що дозволить працювати зі базою швидше та зменшити навантаження. Тому рішення щодо вибору моделей транзакцій чи бази даних в цілому, повинно бути прийнято, враховуючи всі аспекти проекту.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. SQL: полное руководство 3-е изд. : Пер. с англ. / Д. Р. Грофф, П. Н. Вайнберг, Э. Дж. Оппелъ. – М.: ООО "И.Д. Вильяме", 2015. – С. 276-414. – ISBN 978-5-8459-1654-9.

2. Haerder T. Principles of transaction-oriented database recovery / T. Haerder, A. Reuter // ACM Computing Surveys. – 1983. – V. 15. – P. 287-317.

3. Gray J. The Transaction Concept: Virtues and Limitations / J. Gray. – 1981. – 34 p.

4. Нестеров С. А. Базы данных : учебник и практикум для академического бакалавриата / С. А. Нестеров. – Москва : Издательство Юрайт, 2018. – 230 с. – ISBN 978-5-534-00874-6

5. Serializable vs. Snapshot Isolation Level [Web resource]. – Access mode: https://techcommunity.microsoft.com/t5/sql-server/serializable-vs-snapshot-isolation-level/ba-p/383281

6. Pritchett D. BASE: An Acid Alternative: In partitioned databases, trading some consistency for availability can lead to dramatic improvements in scalability / D. Pritchett // Queue. – 2008. – V.6. – 48-55.

7. RavenDB Transaction Support [Web Resource]. – Access mode: https://ravendb.net/docs/article-page/5.1/csharp/client-api/faq/transaction-support

8. Sokolinsky L.B. Choosing Multiprocessor SystemArchitecture for Parallel Database Systems : Proc. 2nd Int. Workshop on Comput. Sci. and Inform. Technologies(CSIT'2000). – Ufa. – 2000.

9. Aslett M. NoSQL, NewSQL and Beyond: The answer to SPRAINed relational databases [Web Resource]. – 2011. – Access mode: http://blogs.451research.com/information_management/2011/04/15/nosql-newsql-and-beyond/

10. Aslett M. How Will The Database Incumbents Respond To NoSQL And NewSQL? [Web Resource]. – 451 Group. – 2011. – 5 p.

11. Venkatesh P. NewSQL – The New Way to Handle Big Data [Web Resource]. – 2012. – Access mode: https://www.opensourceforu.com/2012/01/newsql-handle-big-data/

Кравчук Сергій Миколайович – студент групи 1АКІТ-20м, факультет комп'ютерних систем та автоматики, кафедра автоматизації і інтелектуальний інформаційних технологій, Вінницький національний технічний університет, м. Вінниця.

Науковий керівник: **Паламарчук Євген Анатолійович** – кандидат технічних наук, доцент кафедри автоматизації та інтелектуальних інформаційних технологій, Вінницький національний технічний університет, м. Вінниця, e-mail: p@vntu.edu.ua

Serhii M. Kravchuk – student, Faculty of Computer Control Systems and Automatics, Vinnytsia National Technical University, Vinnytsia.

Supervisor: **Yevhen A. Palamarchuk** — Ph.D. of Technical Sciences, associate professor at the Automation and Intelligent Information Technologies Department, Vinnytsia National Technical University, Vinnytsia, e-mail: p@vntu.edu.ua