УДК 519.852.6
https://doi.org/10.36906/AP-2022/42

Мамедли Р.Э.
канд. физ.-мат. наук
Нижневартовский государственный университет
г. Нижневартовск, Россия

УПРАВЛЕНИЯ С МЕТОДАМИ БЛОКИРОВКИ В ТРАНЗАКЦИЯХ

Аннотация. В работе рассматриваются различные методы управления блокировками в транзакциях. Обоснована необходимость настройки производительности SQL и настройки производительности СУБД. Приведены примеры управления с методами блокировки в транзакциях для принятия важных решений по повышению эффективности обработки запросов.

Ключевые слова: транзакция; настройка производительности базы данных; методы блокировки; степень детализации блокировки; двоичная блокировка; разделяемая блокировка; исключительная блокировка; двухфазная блокировка; тупики.

Mammadli R.E.
Ph.D.
Nizhnevartovsk State University
Nizhnevartovsk, Russia

CONTROLS WITH BLOCKING METHODS IN TRANSACTIONS

Abstract. The paper discusses various methods for managing locks in transactions. The necessity of SQL performance tuning and DBMS performance tuning is substantiated. Examples of management with blocking methods in transactions are given for making important decisions to improve the efficiency of query processing.

Keywords: transaction; database performance tuning; locking methods; lock granularity; binary lock; shared lock; exclusive lock; two-phase lock; deadlocks.

Настройка производительности базы данных является критически важной темой, однако она обычно получает минимальное освещение в учебной программе по базам данных. Большинство баз данных, используемых в университете, имеют только несколько записей на таблицу. В результате основное внимание часто уделяется тому, чтобы запросы SQL выполняли намеченную задачу без учёта эффективности процесса запроса. Фактически, даже самая эффективная среда запросов не даёт видимых улучшений производительности по сравнению с наименее эффективной средой запросов, когда запрашивается только 20 или 30 строк (записей) таблицы. Отсутствие внимания к эффективности запросов может привести к

CC BY

недопустимо медленным результатам в реальном мире, когда запросы выполняются для десятков миллионов записей.

Методы блокировки являются одним из наиболее распространённых методов, используемых в управлении параллелизмом, поскольку они облегчают изоляцию элементов данных, используемых при одновременном выполнении транзакций. Блокировка гарантирует исключительное использование элемента данных для текущей транзакции. Другими словами, транзакция T2 не имеет доступа к элементу данных, который в данный момент используется транзакцией T1. Транзакция получает блокировку до доступа к данным; блокировка снимается, когда транзакция завершается, так что другая транзакция может заблокировать элемент данных для его исключительного использования. Эта серия блокирующих действий предполагает, что параллельные транзакции могут пытаться манипулировать одним и тем же элементом данных в одно и то же время. Использование блокировок, основанных на предположении о вероятности конфликта между транзакциями, обычно называют пессимистической блокировкой.

Согласованность данных не может быть гарантирована во время транзакции; база данных может находиться во временном несовместимом состоянии при выполнении нескольких обновлений. Следовательно, блокировки необходимы для предотвращения чтения несогласованными данными другой транзакции.

Современные СУБД автоматически инициируют и применяют процедуры блокировки. Вся информация о блокировке обрабатывается менеджером блокировок, который отвечает за назначение и контроль блокировок, используемых транзакциями.

Степень детализации блокировки показывает уровень использования блокировки. Блокировка может осуществляться на следующих уровнях: база данных, таблица, страница, строка или даже поле (атрибут).

При блокировке уровня базы данных вся база данных блокируется, что предотвращает использование любых таблиц в базе данных транзакцией T2 во время выполнения транзакции T1. Этот уровень блокировки хорош для пакетных процессов, но не подходит для многопользовательских СУБД.

При блокировке на уровне таблицы вся таблица блокируется, предотвращая доступ к любой строке транзакции T2, пока транзакция T1 использует таблицу. Если транзакция требует доступа к нескольким таблицам, каждая таблица может быть заблокирована. Однако две транзакции могут обращаться к одной и той же базе данных, если они обращаются к разным таблицам. Блокировки на уровне таблицы, хотя и менее строгие, чем блокировки на уровне базы данных, вызывают пробки, когда многие транзакции ожидают доступа к одной и той же таблице. Такое условие особенно утомительно, если блокировка вызывает задержку, когда разные транзакции требуют доступа к разным частям одной и той же таблицы, то есть когда транзакции не будут мешать друг другу. Следовательно, блокировки на уровне таблицы не подходят для многопользовательских СУБД.

При блокировке на уровне страницы СУБД блокирует всю страницу диска. Страница диска, или страница, является эквивалентом дискового блока, который может быть описан как

CC BY

непосредственно адресуемый раздел диска. Страница имеет фиксированный размер, например, 4КБ, 8КБ или 16КБ. Например, если необходимо записать только 73 байта на страницу 4КБ, всю страницу необходимо прочитать с диска, обновить в памяти и записать обратно на диск. Таблица может занимать несколько страниц, а страница может содержать несколько строк одной или нескольких таблиц. В настоящее время блокировки на уровне страниц являются наиболее часто используемым методом блокировки для многопользовательских СУБД.

Блокировка на уровне строк гораздо менее ограничительная. СУБД позволяет одновременным транзакциям получать доступ к разным строкам одной и той же таблицы, даже если строки расположены на одной странице. Хотя подход блокировки на уровне строк улучшает доступность данных, его управление требует больших накладных расходов, поскольку существует блокировка для каждой строки в таблице базы данных, участвующей в конфликтующей транзакции. Современные СУБД автоматически повышают блокировку с уровня строки до уровня страницы, когда сеанс приложения запрашивает несколько блокировок на одной странице.

Блокировка на уровне поля позволяет одновременным транзакциям получать доступ к одной и той же строке, если они требуют использования разных полей (атрибутов) в этой строке. Хотя блокировка на уровне поля однозначно обеспечивает наиболее гибкий многопользовательский доступ к данным, она редко реализуется в СУБД, поскольку требует чрезвычайно высокого уровня издержек компьютера и потому что блокировка на уровне строк гораздо более полезна на практике.

Независимо от уровня блокировки, СУБД может использовать разные типы или режимы блокировки: двоичная или разделяемая/исключительная.

Двоичная блокировка имеет только два состояния: заблокировано (1) или разблокировано (0). Если объект, такой как база данных, таблица, страница или строка, заблокирован транзакцией, никакая другая транзакция не может использовать этот объект. Если объект разблокирован, любая транзакция может заблокировать объект для его использования. Каждая операция с базой данных требует, чтобы уязвимый объект был заблокирован. Как правило, транзакция должна разблокировать объект после его завершения. Поэтому каждая транзакция требует операции блокировки и разблокировки для каждого элемента данных, к которому осуществляется доступ. Такие операции автоматически управляются и планируются СУБД; пользователь не блокирует и не разблокирует элементы данных.

Исключительная блокировка существует, когда доступ зарезервирован специально для транзакции, которая заблокировала объект. Исключительная блокировка должна использоваться, когда существует вероятность конфликта. Общая блокировка существует, когда параллельным транзакциям предоставляется доступ на чтение на основе общей блокировки. Общая блокировка не вызывает конфликта, если все параллельные транзакции доступны только для чтения.

CC BY

Общая блокировка выдаётся, когда транзакция хочет прочитать данные из базы данных, и для этого элемента данных не установлена исключительная блокировка. Исключительная блокировка выдаётся, когда транзакция хочет обновить (записать) элемент данных, и в настоящий момент никакие блокировки для этого элемента данных не удерживаются какой-либо другой транзакцией. Используя концепцию общей/исключительной блокировки, блокировка может иметь три состояния: разблокирована, общая (чтение) и исключительная (запись).

Две транзакции конфликтуют только тогда, когда хотя бы одна транзакция является записью. Поскольку две транзакции чтения могут быть безопасно выполнены одновременно, общие блокировки позволяют нескольким транзакциям чтения одновременно считывать один и тот же элемент данных. Например, если транзакция T1 имеет общую блокировку элемента X данных, и транзакция T2 хочет прочитать элемент X данных, T2 также может получить общую блокировку элемента X данных.

Если транзакция T2 обновляет элемент данных X, T2 требует исключительную блокировку для элемента данных X. Исключительная блокировка предоставляется, если никакие другие блокировки не удерживаются для элемента данных (это условие известно, как правило взаимоисключающего доступа: только одна транзакция за раз может владеть исключительной блокировкой объекта.) Следовательно, если совместно используемая (или исключительная) блокировка для элемента данных X уже удержана транзакцией T1, исключительная блокировка не может быть предоставлена транзакции T2, и T2 должен ждать начать, пока T1 не совершит. Другими словами, общая блокировка всегда будет блокировать исключительную (запись) блокировку; следовательно, снижение параллелизма транзакций.

Хотя использование общих блокировок делает доступ к данным более эффективным, схема общей/эксклюзивной блокировки увеличивает накладные расходы менеджера блокировок по нескольким причинам:

• Тип блокировки должен быть известен до того, как она может быть предоставлена.

• Существуют три операции блокировки: READ_LOCK для проверки типа блокировки, WRITE_LOCK для выдачи блокировки и UNLOCK для снятия блокировки.

• Схема была усовершенствована, чтобы разрешить обновление блокировки с общего на эксклюзивный и понижение блокировки с эксклюзивного на общий.

Хотя блокировки предотвращают серьёзные несоответствия данных, они могут привести к двум основным проблемам:

• Итоговое расписание транзакций может быть не сериализуемым.

• Расписание может создать тупики. Взаимная блокировка возникает, когда две транзакции бесконечно ждут друг друга, чтобы разблокировать данные. Блокировка базы данных, которая аналогична блокировке трафика в большом городе, возникает, когда две или более транзакций ждут друг друга, чтобы разблокировать данные.

К счастью, обеими проблемами можно управлять: сериализуемость достигается с помощью протокола блокировки, известного как двухфазная блокировка, а тупиковыми

CC BY

ситуациями можно управлять с помощью методов обнаружения и предотвращения взаимных блокировок. Эти методы рассматриваются в следующих двух разделах.

Двухфазная блокировка (2ФБ) определяет, как транзакции получают и снимают блокировки. Двухфазная блокировка гарантирует сериализуемость, но не предотвращает взаимоблокировки. Две фазы:

Растущая фаза, на которой транзакция получает все необходимые блокировки без разблокировки каких-либо данных. Как только все блокировки были получены, транзакция находится в своей заблокированной точке.

Фаза сжатия, при которой транзакция снимает все блокировки и не может получить новую блокировку.

Протокол двухфазной блокировки регулируется следующими правилами:

Две транзакции не могут иметь конфликтующие блокировки.

Операция блокировки не может предшествовать операции блокировки в той же транзакции.

На данные не влияют, пока не будут получены все блокировки, то есть до тех пор, пока транзакция не окажется в своей заблокированной точке.

Взаимная блокировка возникает, когда две транзакции бесконечно ждут друг друга, чтобы разблокировать данные. Например, тупик возникает, когда две транзакции, T1 и T2, существуют в следующем режиме:

T1 = доступ к элементам данных X и Y

T2 = доступ к элементам данных Y и X

Если T1 не имеет разблокированного элемента данных Y, T2 не может начаться; если T2 не разблокировал элемент данных X, T1 не может продолжить. Следовательно, и T1 и T2 ждут, пока другой разблокирует требуемый элемент данных. Такой тупик также известен как смертельные объятия.

В СУБД одновременно может быть выполнено гораздо больше транзакций, что увеличивает вероятность возникновения тупиковых ситуаций. Взаимоблокировки возможны только тогда, когда одна из транзакций хочет получить монопольную блокировку элемента данных; между общими блокировками не может быть тупиковых условий.

Три основных метода управления тупиками:

• Предотвращение тупиковой ситуации. Транзакция, запрашивающая новую блокировку, прерывается, когда существует вероятность возникновения тупика. Если транзакция отменяется, все изменения, сделанные этой транзакцией, откатываются и все блокировки, полученные транзакцией, снимаются. Затем транзакция переносится на исполнение. Предотвращение взаимоблокировки работает, потому что оно избегает условий, которые приводят к взаимоблокировке.

• Обнаружение тупика. СУБД периодически проверяет базу данных на наличие взаимоблокировок. Если обнаружена взаимоблокировка, транзакция «жертвы» прерывается (откатывается и перезапускается), а другая транзакция продолжается.

CC BY

• Избегание от тупиковой ситуации. Транзакция должна получить все необходимые блокировки, прежде чем она может быть выполнена. Этот метод позволяет избежать отката конфликтующих транзакций, требуя, чтобы блокировки были получены последовательно. Однако назначение последовательной блокировки, необходимое для предотвращения тупика, увеличивает время отклика на действие.

Выбор используемого метода управления взаимоблокировкой зависит от среды базы данных. Если вероятность взаимоблокировки низкая, рекомендуется обнаружение взаимоблокировки. Если вероятность взаимоблокировки высока, рекомендуется предотвращение взаимоблокировки. Если время отклика не является высоким в списке приоритетов системы, может быть использовано предотвращение тупиковых ситуаций. Все существующие СУБД поддерживают обнаружение взаимоблокировок в транзакционных базах данных, в то время как некоторые СУБД используют сочетание методов предотвращения и предотвращения для других типов данных, таких как хранилища данных или данные XML.

© Мамедли Р.Э., 2022