ВикипедиЯ

Уровень изолированности транзакций

Материал из Википедии — свободной энциклопедии
Текущая версия страницы пока не проверялась опытными участниками и может значительно отличаться от версии, проверенной 5 июля 2023 года; проверки требуют 17 правок.

Уровень изолированности транзакций — условное значение, определяющее, в какой мере в результате выполнения логически параллельных транзакций в СУБД допускается получение несогласованных данных. Шкала уровней изолированности транзакций содержит ряд значений, проранжированных от наинизшего до наивысшего; более высокий уровень изолированности соответствует лучшей согласованности данных, но его использование может снижать количество физически параллельно выполняемых транзакций. И наоборот, более низкий уровень изолированности позволяет выполнять больше параллельных транзакций, но снижает точность данных. Таким образом, выбирая используемый уровень изолированности транзакций, разработчик информационной системы в определённой мере делает выбор между скоростью работы и обеспечением гарантированной согласованности получаемых из системы данных.

Содержание

Проблемы параллельного доступа с использованием транзакций

При параллельном выполнении транзакций возможны следующие аномалии (проблемы):

- потерянное обновление (англ. *lost update*) — при одновременном изменении одного блока данных разными транзакциями теряются все изменения, кроме последнего;
- «грязное» чтение (англ. *dirty read*) — чтение данных, добавленных или изменённых транзакцией, которая впоследствии не подтвердится (откатится);
- неповторяющееся чтение (англ. *non-repeatable read*) — при повторном чтении в рамках одной транзакции ранее прочитанные данные оказываются изменёнными;
- фантомное чтение (англ. *phantom reads*) — одна транзакция в ходе своего выполнения несколько раз выбирает множество строк по одним и тем же

критериям. Другая транзакция в интервалах между этими выборками добавляет строки или изменяет столбцы некоторых строк, используемых в критериях выборки первой транзакции, и успешно заканчивается. В результате получится, что одни и те же выборки в первой транзакции дают разные множества строк.

Рассмотрим ситуации, в которых возможно возникновение данных проблем.

Потерянное обновление

Ситуация, когда при одновременном изменении одного блока данных разными транзакциями одно из изменений теряется.

Предположим, имеются две транзакции, выполняемые одновременно:

Транзакция 1	Транзакция 2
`UPDATE tbl1 SET f2=f2+20 WHERE f1=1;`	`UPDATE tbl1 SET f2=f2+25 WHERE f1=1;`

В обеих транзакциях изменяется значение поля f2, по их завершении значение поля должно быть увеличено на 45. В действительности может возникнуть следующая последовательность действий:

1. Обе транзакции одновременно читают текущее состояние поля. Точная физическая одновременность здесь не обязательна, достаточно, чтобы вторая по порядку операция чтения выполнилась до того, как другая транзакция запишет свой результат.
2. Обе транзакции вычисляют новое значение поля, прибавляя, соответственно, 20 и 25 к ранее прочитанному значению.
3. Транзакции пытаются записать результат вычислений обратно в поле f2. Поскольку физически одновременно две записи выполнить невозможно, в реальности одна из операций записи будет выполнена раньше, другая позже. При этом вторая операция записи перезапишет результат первой.

В результате значение поля f2 по завершении обеих транзакций может увеличиться не на 45, а на 20 или 25, то есть одна из изменяющих данные транзакций «пропадёт».

«Грязное» чтение

Чтение данных, добавленных или изменённых транзакцией, которая впоследствии не подтвердится (откатится).

Предположим, имеются две транзакции, открытые различными приложениями, в которых выполнены следующие SQL-операторы:

Транзакция 1	Транзакция 2
`UPDATE tbl1 SET f2=f2+1 WHERE f1=1;`	
	`SELECT f2 FROM tbl1 WHERE f1=1;`
`ROLLBACK;`	

В транзакции 1 изменяется значение поля f2, а затем в транзакции 2 выбирается значение этого поля. После этого происходит откат транзакции 1. В результате значение, полученное второй транзакцией, будет отличаться от значения, хранимого в базе данных.

Неповторяющееся чтение

Ситуация, когда при повторном чтении в рамках одной транзакции ранее прочитанные данные оказываются изменёнными.

Предположим, имеются две транзакции, открытые различными приложениями, в которых выполнены следующие SQL-операторы:

Транзакция 1	Транзакция 2
	`SELECT f2 FROM tbl1 WHERE f1=1;`
`UPDATE tbl1 SET f2=f2+3 WHERE f1=1;`	
`COMMIT;`	
	`SELECT f2 FROM tbl1 WHERE f1=1;`

В транзакции 2 выбирается значение поля f2, затем в транзакции 1 изменяется значение поля f2. При повторной попытке выбора значения из поля f2 в транзакции 2 будет получен другой результат. Эта ситуация особенно неприемлема, когда данные считываются с целью их частичного изменения и обратной записи в базу данных.

Чтение «фантомов»

Ситуация, когда при повторном чтении в рамках одной транзакции одна и та же выборка дает разные множества строк.

Предположим, имеется две транзакции, открытые различными приложениями, в которых выполнены следующие SQL-операторы:

Транзакция 1	Транзакция 2
	`SELECT SUM(f2) FROM tbl1;`
`INSERT INTO tbl1 (f1,f2) VALUES (15,20);`	
`COMMIT;`	
	`SELECT SUM(f2) FROM tbl1;`

В транзакции 2 выполняется SQL-оператор, использующий все значения поля f2. Затем в транзакции 1 выполняется вставка новой строки, приводящая к тому, что повторное выполнение SQL-оператора в транзакции 2 выдаст другой результат. Такая ситуация называется чтением фантома (фантомным чтением). От неповторяющегося чтения оно отличается тем, что результат повторного обращения к данным изменился не из-за изменения/удаления самих этих данных, а из-за появления новых (фантомных) данных.

Уровни изоляции

Под «**уровнем изоляции транзакций**» понимается степень обеспечиваемой внутренними механизмами СУБД (то есть не требующей специального программирования) защиты от всех или некоторых вышеперечисленных видов несогласованности данных, возникающих при параллельном выполнении транзакций. Стандарт SQL-92 определяет шкалу из четырёх уровней изоляции: Read uncommitted, Read committed, Repeatable read, Serializable. Первый из них является самым слабым, последний — самым сильным, каждый последующий включает в себя все предыдущие.

Read uncommitted (чтение незафиксированных данных)

Низший (первый) уровень изоляции[1]. Если несколько параллельных транзакций пытаются изменять одну и ту же строку таблицы, то в окончательном варианте строка будет иметь значение, определённое всем набором успешно выполненных транзакций. При этом возможно считывание не

только логически несогласованных данных, но и данных, изменения которых ещё не зафиксированы.

Типичный способ реализации данного уровня изоляции — блокировка данных на время выполнения команды изменения, что гарантирует, что команды изменения одних и тех же строк, запущенные параллельно, фактически выполняются последовательно, и ни одно из изменений не потеряется. Транзакции, выполняющие только чтение, при данном уровне изоляции никогда не блокируются.

Read committed (чтение фиксированных данных)

Большинство промышленных СУБД, в частности, Microsoft SQL Server, PostgreSQL и Oracle Database, по умолчанию используют именно этот уровень. На этом уровне обеспечивается защита от чернового, «грязного» чтения, тем не менее, в процессе работы одной транзакции другая может быть успешно завершена и сделанные ею изменения зафиксированы. В итоге первая транзакция будет работать с другим набором данных.

Реализация чтения зафиксированных данных может основываться на одном из двух подходов: блокировании или версионности.

Блокирование читаемых и изменяемых данных.
: Заключается в том, что пишущая транзакция блокирует изменяемые данные для читающих транзакций, работающих на уровне read committed или более высоком, до своего завершения, препятствуя, таким образом, «грязному» чтению, а данные, блокируемые читающей транзакцией, освобождаются сразу после завершения команды SELECT (таким образом, на данном уровне изоляции может возникать ситуация «неповторяющегося чтения»).

Сохранение нескольких версий параллельно изменяемых строк.
: При каждом изменении строки СУБД создаёт новую версию этой строки, с которой продолжает работать изменившая данные транзакция, в то время как любой другой «читающей» транзакции возвращается последняя зафиксированная версия. Преимущество такого подхода в том, что он обеспечивает бóльшую скорость, так как предотвращает блокировки. Однако он требует, по сравнению с первым, существенно бóльшего расхода оперативной памяти, которая тратится на хранение версий строк. Кроме того, при параллельном изменении данных несколькими транзакциями может создаться ситуация, когда несколько параллельных транзакций произведут несогласованные изменения одних и тех же данных (поскольку блокировки отсутствуют, ничто не помешает это сделать). Тогда та транзакция, которая зафиксируется первой, сохранит свои изменения в основной БД, а остальные параллельные транзакции окажется невозможно зафиксировать (так как это приведёт к потере обновления первой транзакции). Единственное, что может в такой ситуации СУБД — это откатить остальные транзакции и выдать сообщение об ошибке «Запись уже изменена».

Конкретный способ реализации выбирается разработчиками СУБД, а в ряде случаев может настраиваться. Так, по умолчанию MS SQL использует блокировки, но (в версии 2005 и выше), при установке параметра READ_COMMITTED_SNAPSHOT базы данных, переходит на стратегию версионности, Oracle исходно работает только по версионной схеме. В Informix можно предотвратить конфликты между читающими и пишущими транзакциями, установив параметр конфигурации USELASTCOMMITTED (начиная с версии 11.1), при этом читающая транзакция будет получать последние подтвержденные данные[2]

Repeatable read (повторяющееся чтение)

Уровень, при котором читающая транзакция «не видит» изменения данных, которые были ею ранее прочитаны. При этом никакая другая транзакция не может изменять данные,

читаемые текущей транзакцией, пока та не окончена.

Блокировки в разделяющем режиме применяются ко всем данным, считываемым любой инструкцией транзакции, и сохраняются до её завершения. Это запрещает другим транзакциям изменять строки, которые были считаны незавершённой транзакцией. Однако другие транзакции могут вставлять новые строки, соответствующие условиям поиска инструкций, содержащихся в текущей транзакции. При повторном запуске инструкции текущей транзакцией будут извлечены новые строки, что приведёт к фантомному чтению. Учитывая то, что разделяющие блокировки сохраняются до завершения транзакции, а не снимаются в конце каждой инструкции, степень параллелизма ниже, чем при уровне изоляции READ COMMITTED. Поэтому пользоваться данным и более высокими уровнями изоляции транзакций без необходимости обычно не рекомендуется.

Serializable (упорядочиваемость)

Самый высокий уровень изолированности; транзакции полностью изолируются друг от друга. Результат выполнения нескольких параллельных транзакций должен быть таким, как если бы они выполнялись последовательно. Только на этом уровне параллельные транзакции не подвержены эффекту «фантомного чтения».

Это может достигаться за счет того, что изменяющая транзакция блокирует всю таблицу для изменяющих и читающих транзакций, а также читающая транзакция блокирует всю таблицу для изменяющих транзакций. Или менее радикальное - изменяющая транзакция блокирует строки для тех транзакций, которые захватывают этот диапазон строк, в котором находится изменяющая транзакция, а также читающая транзакция блокирует строки для тех изменяющих транзакций, которые захватывают этот диапазон строк, в котором находится читающая транзакция.

Поддержка изоляции транзакций в реальных СУБД

СУБД, обеспечивающие транзакционность, не всегда поддерживают все четыре уровня, а также могут вводить дополнительные. Возможны также различные нюансы в обеспечении изоляции.

Так, Oracle Database в принципе не поддерживает нулевой уровень, так как его реализация транзакций исключает «грязные чтения», и формально не позволяет устанавливать уровень Repeatable read, то есть поддерживает только Read committed (по умолчанию) и Serializable. При этом на уровне отдельных команд он, фактически, гарантирует повторяемость чтения (если команда SELECT в первой транзакции выбирает из базы набор строк, и в это время параллельная вторая транзакция изменяет какие-то из этих строк, то результирующий набор, полученный первой транзакцией, будет содержать неизменённые строки, как будто второй транзакции не было). Также Oracle поддерживает так называемые READ-ONLY транзакции, которые соответствуют Serializable, но при этом не могут сами изменять данные.

Microsoft SQL Server поддерживает все четыре стандартных уровня изоляции транзакций, а дополнительно — уровень SNAPSHOT, на котором транзакция видит то состояние данных, которое было зафиксировано до её запуска, а также изменения, внесённые ею самой, то есть ведёт себя так, как будто получила при запуске моментальный снимок данных БД и работает с ним. Отличие от Serialized состоит в том, что не используются блокировки, но в результате фиксация изменений может оказаться невозможной, если параллельная транзакция изменила те же самые данные раньше; в этом случае вторая транзакция при попытке выполнить COMMIT вызовет сообщение об ошибке и будет отменена.

MySQL 8.x декларирует (https://dev.mysql.com/doc/refman/8.0/en/set-transaction.html) поддержку всех четырёх стандартных уровня изоляции транзакций.

Поведение при различных уровнях изолированности

«+» — предотвращает, «-» — не предотвращает.

Уровень изоляции	Фантомное чтение	Неповторяющееся чтение	«Грязное» чтение	Потерянное обновление[3]
SERIALIZABLE	+	+	+	+
REPEATABLE READ	-	+	+	+
READ COMMITTED	-	-	+	+ блокировка / - в версионной схеме
READ UNCOMMITTED	-	-	-	+[4]

Примечания

1. Understanding Isolation Levels (http://msdn.microsoft.com/en-us/library/ms378149.aspx). Дата обращения: 14 ноября 2011. Архивировано (https://web.archive.org/web/20120518024822/http://msdn.microsoft.com/en-us/library/ms378149.aspx) 18 мая 2012 года.
2. Параметр конфигурации USELASTCOMMITTED http://publib.boulder.ibm.com/infocenter/idshelp/v115/topic/com.ibm.adref.doc/ids_adr_0186.htm
3. Understanding the Available Transaction Isolation Levels (http://technet.microsoft.com/en-us/library/cc546518.aspx). Дата обращения: 30 августа 2012. Архивировано (https://web.archive.org/web/20121014161726/http://technet.microsoft.com/en-us/library/cc546518.aspx) 14 октября 2012 года.
4. *Paul Wilton, John Colby.* Beginning SQL (https://books.google.com/books?id=9eqbXSnji84C&newbks=0&printsec=frontcover&pg=PA319&dq=Read+Uncommitted+lost+update&hl=ru). — John Wiley & Sons, 2005-03-04. — С. 319. — 522 с. — ISBN 978-0-7645-9632-2. Архивировано (https://web.archive.org/web/20210422143125/https://books.google.com/books?id=9eqbXSnji84C&newbks=0&printsec=frontcover&pg=PA319&dq=Read+Uncommitted+lost+update&hl=ru) 22 апреля 2021 года.
