Блокировки

Блокировка — это механизм, с помощью которого компонент Database Engine синхронизирует одновременный доступ нескольких пользователей к одному фрагменту данных. Прежде чем транзакция сможет распоряжаться текущим состоянием фрагмента данных, например, для чтения или изменения данных, она должна защититься от изменений этих данных другой транзакцией. Для этого транзакция запрашивает блокировку фрагмента данных. Существует несколько режимов блокировки, например общая или монопольная. Режим блокировки определяет уровень подчинения данных транзакции. Ни одна транзакция не может получить блокировку, которая противоречит другой блокировке этих данных, предоставленной другой транзакции. Если транзакция запрашивает режим блокировки, противоречащий предоставленной ранее блокировке тех же данных, экземпляр компонента Database Engine приостанавливает ее работу до тех пор, пока первая блокировка не освободится.

При изменении фрагмента данных транзакция удерживает блокировку, защищая изменения до конца транзакции. Продолжительность блокировки, полученной для защиты операций чтения, зависит от уровня изоляции транзакции. Все блокировки, удерживаемые транзакцией, освобождаются после ее завершения (при фиксации или откате).
Приложения обычно не запрашивают блокировку напрямую. За управление блокировками отвечает внутренний компонент Database Engine, называемый диспетчером блокировок. Когда экземпляр компонента Database Engine обрабатывает инструкцию Transact-SQL, обработчик запросов компонента Database Engine определяет, к каким ресурсам требуется доступ. Обработчик запросов определяет, какие типы блокировок требуются для защиты каждого ресурса, в зависимости от типа доступа и уровня изоляции транзакции. Затем обработчик запросов запрашивает соответствующую блокировку у диспетчера блокировок. Диспетчер блокировок предоставляет блокировку, если она не противоречит блокировкам, удерживаемым другими транзакциями. Использование блокировки как механизма управления транзакциями может разрешить проблемы параллелизма. Блокировка позволяет запускать все транзакции в полной изоляции друг от друга, что позволяет одновременно выполнять несколько транзакций. Уровень, на котором транзакция готова к принятию противоречивых данных, называется уровнем изоляции. Чем выше уровень изоляции, тем ниже вероятность непоследовательности данных, однако при этом появляется такой недостаток, как сокращение параллелизма.

Каждая блокировка обладает тремя свойствами: гранулярностью (или размером блокировки), режимом (или типом блокировки) и продолжительностью.

Гранулярность блокировок и иерархии блокировок

SQL Server поддерживает мультигранулярную блокировку, позволяющую транзакции блокировать различные типы ресурсов. Чтобы уменьшить издержки применения блокировок, компонент Database Engine автоматически блокирует ресурсы на соответствующем задаче уровне. Блокировка при меньшей гранулярности, например на уровне строк, увеличивает параллелизм, но в то же время увеличивает и накладные расходы на обработку, поскольку при большом количестве блокируемых строк требуется больше блокировок. Блокировки на большем уровне гранулярности, например на уровне таблиц, обходится дорого в отношении параллелизма, поскольку блокировка целой таблицы ограничивает доступ ко всем частям таблицы других транзакций. Однако накладные расходы в этом случае ниже, поскольку меньше количество поддерживаемых блокировок.

Компонент Database Engine часто получает блокировки на нескольких уровнях гранулярности одновременно, чтобы полностью защитить ресурс. Такая группа блокировок на нескольких уровнях гранулярности называется иерархией блокировки. Например, чтобы полностью защитить операцию чтения индекса, экземпляру компоненту Database Engine может потребоваться получить разделяемые блокировки на строки и намеренные разделяемые блокировки на страницы и таблицу.

Следующая таблица содержит перечень ресурсов, которые могут блокироваться компонентом Database Engine.

Ресурс	Описание
RID	Идентификатор строки, используемый для блокировки одной строки в куче
KEY	Блокировка строки в индексе, используемая для защиты диапазонов значений ключа в сериализуемых транзакциях.
PAGE	8-килобайтовая (КБ) страница в базе данных, например страница данных или индекса.
EXTENT	Упорядоченная группа из восьми страниц, например страниц данных или индекса.
HOBT	Куча или сбалансированное дерево. Блокировка, защищающая индекс или кучу страниц данных в таблице, не имеющей кластеризованного индекса.
TABLE	Таблица полностью, включая все данные и индексы.
FILE	Файл базы данных.
APPLICATION	Определяемый приложением ресурс.
METADATA	Блокировки метаданных.
ALLOCATION_UNIT	Единица размещения.
DATABASE	База данных, полностью.

По умолчанию блокировки укрупняются с уровня отдельных строк до целых страниц даже до уровня таблицы из соображений повышения производительности. Хотя в общем случае укрупнение считается хорошей штукой, оно может создавать проблемы, например, когда один SPID блокирует всю таблицу, препятствуя другому SPID работать ней. Можно настроить параметры блокировки на уровне строк и страниц. Такие блокировки по умолчанию разрешены для индексов.

Режимы блокировки

SQL Server блокирует ресурсы с помощью различных режимов блокировки, которые определяют доступ одновременных транзакций к ресурсам. В следующей таблице показаны режимы блокировки ресурсов, применяемые компонентом Database Engine.

Режим блокировки	Описание
Совмещаемая блокировка (S)	Совмещаемая Используется для операций считывания, которые не меняют и не обновляют данные, такие как инструкция SELECT.
Блокировка обновления (U)	Применяется к тем ресурсам, которые могут быть обновлены. Предотвращает возникновение распространенной формы взаимоблокировки, возникающей тогда, когда несколько сеансов считывают, блокируют и затем, возможно, обновляют ресурс.
Монопольная блокировка (X)	Используется для операций модификации данных, таких как инструкции INSERT, UPDATE или DELETE. Гарантирует, что несколько обновлений не будет выполнено одновременно для одного ресурса.
Блокировка с намерением	Используется для создания иерархии блокировок. Типы намеренной блокировки: с намерением совмещаемого доступа (IS), с намерением монопольного доступа (IX), а также совмещаемая с намерением монопольного доступа (SIX).
Блокировка схемы	Используется во время выполнения операции, зависящей от схемы таблицы. Типы блокировки схем: блокировка изменения схемы (Sch-S) и блокировка стабильности схемы (Sch-M).
Блокировка массового обновления (BU)	Используется, если выполняется массовое копирование данных в таблицу и указана подсказка TABLOCK.
Диапазон ключей	Защищает диапазон строк, считываемый запросом при использовании уровня изоляции сериализуемой транзакции. Запрещает другим транзакциям вставлять строки, что помогает запросам сериализуемой транзакции уточнять, были ли запросы запущены повторно.

Совмещаемые блокировки. Совмещаемые (S) блокировки позволяют одновременным транзакциям считывать (SELECT) ресурс под контролем пессимистичного параллелизма. Пока для ресурса существуют совмещаемые (S) блокировки, другие транзакции не могут изменять данные. Совмещаемые блокировки (S) ресурса снимаются по завершении операции считывания, если только уровень изоляции транзакции не установлен на повторяющееся чтение или более высокий уровень, а также если совмещаемые блокировки (S) не продлены на все время транзакции с помощью указания блокировки.

Блокировки обновления. Блокировки обновления (U) предотвращают возникновение распространенной формы взаимоблокировки. В сериализуемой транзакции или транзакции операцией чтения с возможностью повторения транзакция считывает данные, запрашивает совмещаемую (S) блокировку на ресурс (страницу или строку), затем выполняет изменение данных, что требует преобразование блокировки в монопольную (X). Если две транзакции запрашивают совмещаемую блокировку на ресурс и затем пытаются одновременно обновить

данные, то одна из транзакций пытается преобразовать блокировку в монопольную (X). Преобразование совмещаемой блокировки в монопольную потребует некоторого времени, поскольку монопольная блокировка для одной транзакции несовместима с совмещаемой блокировкой для другой транзакции. Начнется ожидание блокировки. Вторая транзакция попытается получить монопольную (X) блокировку для обновления. Поскольку обе транзакции выполняют преобразование в монопольную (X) блокировку и при этом каждая из транзакций ожидает, пока вторая снимет совмещаемую блокировку, то в результате возникает взаимоблокировка.
Чтобы избежать этой потенциальной взаимоблокировки, применяются блокировки обновления (U). Блокировку обновления (U) может устанавливать для ресурса одновременно только одна транзакция. Если транзакция изменяет ресурс, то блокировка обновления (U) преобразуется в монопольную (X) блокировку.

Монопольные блокировки. Монопольная (X) блокировка запрещает транзакциям одновременный доступ к ресурсу. Если ресурс удерживается монопольной (X) блокировкой, то другие транзакции не могут изменять данные. Операции считывания будут допускаться только при наличии указания NOLOCK или уровня изоляции незафиксированной операции чтения. Изменяющие данные инструкции, такие как INSERT, UPDATE или DELETE, соединяют как операции изменения, так и операции считывания. Чтобы выполнить необходимые операции изменения данных, инструкция сначала получает данные с помощью операций считывания. Поэтому, как правило, инструкции изменения данных запрашивают как совмещаемые, так и монопольные блокировки. Например, инструкция UPDATE может изменять строки в одной таблице, основанной на соединении данных из другой таблицы. В этом случае инструкция UPDATE кроме монопольной блокировки обновляемых строк запрашивает также совмещаемые блокировки для строк, считываемых в соединенной таблице.

Блокировки с намерением. В компоненте Компонент Database Engine блокировки с намерением применяются для защиты размещения совмещаемой (S) или монопольной (X) блокировки ресурса на более низком уровне иерархии. Блокировки с намерением называются так потому, что их получают до блокировок более низкого уровня, то есть они обозначают намерение поместить блокировку на более низком уровне.

Блокировка с намерением выполняет две функции:

- предотвращает изменение ресурса более высокого уровня другими транзакциям таким образом, что это сделает недействительной блокировку более низкого уровня;
- повышает эффективность компонента Компонент Database Engine при распознавании конфликтов блокировок на более высоком уровне гранулярности.

Например, в таблице требуется блокировка с намерением совмещаемого доступа до того, как для страниц или строк этой таблицы будет запрошена совмещаемая (S) блокировка. Если задать блокировку с намерением на уровне таблицы, то другим транзакциям будет запрещено получать монопольную (X) блокировку для таблицы, содержащей эту страницу.

Блокировка с намерением повышает производительность, поскольку компонент Компонент Database Engine проверяет наличие таких блокировок только на уровне таблицы, чтобы определить, может ли транзакция безопасно получить для этой таблицы совмещаемую блокировку. Благодаря этому нет необходимости проверять блокировки в каждой строке и на каждой странице, чтобы убедиться, что транзакция может заблокировать всю таблицу.

Блокировки схем. В компоненте Компонент Database Engine блокировка изменения схемы (Sch-M) применяется с операциями языка DDL для таблиц, например при добавлении столбца или очистке таблицы. Пока удерживается блокировка изменения схемы (Sch-M),

одновременный доступ к таблице запрещен. Это означает, что любые операции вне блокировки изменения схемы (Sch-M) будут запрещены до снятия блокировки.

- Блокировка изменения схемы (Sch-M) применяется с некоторыми операциями языка обработки данных, например усечением таблиц, чтобы предотвратить одновременный доступ к таблице.
- Блокировка стабильности схемы (Sch-S) применяется компонентом Компонент Database Engine при компиляции и выполнении запросов. Блокировка стабильности схемы (Sch-S) не влияет на блокировки транзакций, включая монопольные (X) блокировки. Поэтому другие транзакции (даже транзакции с монопольной блокировкой (X) для таблицы) могут продолжать работу во время компиляции запроса. Однако одновременные операции DDL и DML, которые запрашивают блокировки изменения схемы (Sch-M), не могут выполняться над таблицей.

Блокировки массового обновления. Блокировка массового обновления (BU) позволяет поддерживать несколько одновременных потоков массовой загрузки данных в одну и ту же таблицу и при этом запрещать доступ к таблице любым другим процессам, отличным от массовой загрузки данных. Компонент Database Engine использует блокировки массового обновления (BU), если выполняются два следующих условия. Используется инструкция Transact-SQL BULK INSERT, функция OPENROWSET(BULK) или одна из таких команд массовой вставки API, как .NET SqlBulkCopy, OLEDB Fast Load APIs или ODBC Bulk Copy APIs, для массового копирования данных в таблицу. Выделено указание TABLOCK или установлен параметр таблицы table lock on bulk load с помощью хранимой процедуры sp_tableoption.

Блокировки диапазона ключа. Блокировки диапазона ключей защищают диапазон строк, неявно включенный в набор записей, считываемый инструкцией Transact-SQL при использовании уровня изоляции сериализуемых транзакций. Блокировка диапазона ключей предотвращает фантомные чтения. Кроме того, защита диапазона ключей между строк предотвращает фантомную вставку или удаление из набора записи, к которому получает доступ транзакция.

Совместимость блокировок

Совместимость блокировок определяет, могут ли несколько транзакций одновременно получить блокировку одного и того же ресурса. Если ресурс уже блокирован другой транзакцией, новая блокировка может быть предоставлена только в том случае, если режим запрошенной блокировки совместим с режимом существующей. В противном случае транзакция, запросившая новую блокировку, ожидает освобождения ресурса, пока не истечет время ожидания существующей блокировки.
Например, с монопольными блокировками не совместим ни один из режимов блокировки. Пока удерживается монопольная (X) блокировка, больше ни одна из транзакций не может получить блокировку ни одного из типов (разделяемую, обновления или монопольную) на этот ресурс, пока не будет освобождена монопольная (X) блокировка. И наоборот, если к ресурсу применяется разделяемая (S) блокировка, другие транзакции могут получать разделяемую блокировку или блокировку обновления (U) на этот элемент, даже если не завершилась первая транзакция. Тем не менее, другие транзакции не могут получить монопольную блокировку до освобождения разделяемой. Полная матрица совместимости блокировок приводится в справочниках.

Продолжительность блокировки

Продолжительность блокировок также определяется типами запросов. Когда в рамках транзакции запрос не выполняется, и не используются подсказки блокировки, блокировки для выполнения инструкций SELECT выполняются только на время чтения ресурса, но не во время запроса. Блокировки для инструкций INSERT, UPDATE и DELETE сохраняются на все время выполнения запроса. Это помогает гарантировать согласованность данных и позволяет SQL Server откатывать запросы в случае необходимости.
Когда запрос выполняется в рамках транзакции, продолжительность блокировки определяется тремя факторами:

- типом запроса;
- уровнем изоляции транзакции;
- наличием или отсутствием подсказок блокировки.

Кратковременные (locking) и обычные (blocking) блокировки — нормальное явление в реляционных базах данных, но они могут ухудшать производительность, если блокировки ресурсов сохраняются на протяжении длительного времени. Производительность также страдает, когда, заблокировав ресурс, SPID не в состоянии освободить его.
В первом случае проблема обычно разрешается через какое-то время, так как SPID, в конечном счете, освобождает блокировку, но угроза деградации производительности остается вполне реальной. Проблемы с блокировкой второго типа могут вызвать серьезное падение производительности, но, к счастью, они легко обнаруживаются при мониторинге SQL Server на предмет кратковременных и обычных блокировок.

Эскалация блокировок и их влияние на работу системы

Эскалация (lock escalation) связана с тем, что по мере увеличения, количества отдельных малых заблокированных объектов накладные расходы, связанные с их поддержкой, начинают значительно сказываться на производительности. Блокировки длятся дольше, что приводит к спорным ситуациям – чем дольше существует блокировка, тем выше вероятность обращения к заблокированному объекту со стороны другой транзакции. Очевидно, что на некотором этапе потребуется выполнить объединение (увеличение масштаба) блокировок, чем собственно и занимается диспетчер блокировок. В инструкции ALTER TABLE предусмотрена опция вида SET (LOCK_ESCALATION = { AUTO | TABLE | DISABLE }), которая указывает разрешенные методы укрупнения блокировки для таблицы.

Задание определенного типа блокировки в запросе

Для повышения эффективности исполнения запросов и получения дополнительного контроля над блокировками в запросе можно давать подсказки (hints или хинты), указывая их непосредственно за именем таблицы, которая нуждается в том или ином типе блокировки. Существуют следующие параметры оптимизатора запросов:
SERIALIZABLE/HOLDLOCK
READUNCOMMITTED/NOLOCK
READCOMMITTED
REPEATABLEREAD
READPAST
ROWLOCK
PAGLOCK
TABLOCK
TABLOCKX
UPDLOCK

XLOCK

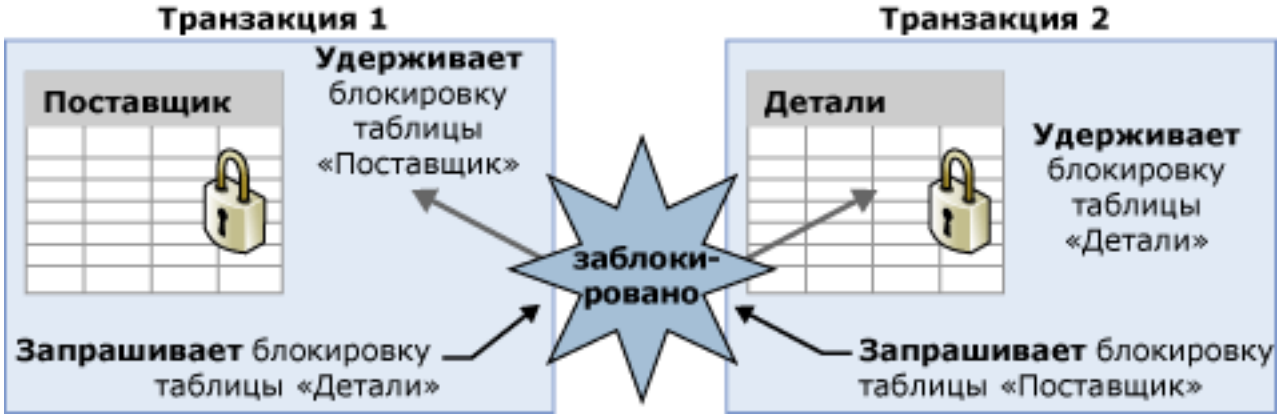
Пример задания эксклюзивной табличной блокировки для таблицы Orders (вместо блокировки на уровне ключей или строк, которую может предложить оптимизатор) может быть таким:

```
SELECT *
FROM Orders AS o WITH (TABLOCKX) JOIN [Order Details] AS od
        ON o.OrderID = od.OrderID
```

Поскольку оптимизатор запросов обычно выбирает лучший план выполнения запроса, использовать подсказки рекомендуется только опытным разработчикам и администраторам баз данных в самом крайнем случае. Сведения об активных блокировках на текущий момент времени можно получить с помощью системной хранимой процедуры sp_lock.

Взаимоблокировки транзакций

Взаимоблокировки или тупиковые ситуации (deadlocks) возникают тогда, когда одна из транзакций не может завершить свои действия, поскольку вторая транзакция заблокировала нужные ей ресурсы, а вторая в то же время ожидает освобождения ресурсов первой транзакцией. На рисунке транзакция T1 зависит от транзакции T2 для ресурса блокировки таблицы Детали. Аналогично транзакция T2 зависит от транзакции T1 для ресурса блокировки таблицы Поставщик. Так как эти зависимости из одного цикла, возникает взаимоблокировка транзакций T1 и T2.



Транзакция T1 не может завершиться до того, как завершится транзакция T2, а транзакция T2 заблокирована транзакцией T1. Такое условие также называется цикличной зависимостью: транзакция T1 зависит от транзакции T2, а транзакция T2 зависит от транзакции T1 и этим замыкает цикл. Обе транзакции находятся в состоянии взаимоблокировки и будут всегда находиться в состоянии ожидания, если взаимоблокировка не будет разрушена внешним процессом.

Взаимоблокировки часто путают с обычными блокировками. Если транзакция запрашивает блокировку на ресурс, заблокированный дугой транзакцией, то запрашивающая транзакция ожидает до тех пор, пока блокировка не освобождается. По умолчанию время ожидания транзакций SQL Server не ограничено, если только не установлен параметр LOCK_TIMEOUT.

Значение -1 (по умолчанию) указывает на отсутствие времени ожидания (то есть инструкция будет ждать всегда). Когда ожидание блокировки превышает значение времени ожидания, возвращается ошибка. Значение «0» означает, что ожидание отсутствует, а сообщение

возвращается, как только встречается блокировка. Функция @@LOCK_TIMEOUT возвращает значение времени ожидания блокировки в миллисекундах для текущего сеанса.

Запрашивающая транзакция блокируется, но не устанавливается в состояние взаимоблокировки, потому что запрашивающая транзакция ничего не сделала, чтобы заблокировать транзакцию, владеющую блокировкой. Наконец, владеющая транзакция завершится и освободит блокировку, и затем запрашивающая транзакция получит блокировку и продолжится.

Как SQL Server обнаруживает взаимоблокировки?

Обнаружение взаимоблокировки выполняется потоком диспетчера блокировок, который периодически производит поиск по всем задачам в экземпляре компонента Database Engine. Следующие пункты описывают процесс поиска:

- Значение интервала поиска по умолчанию составляет 5 секунд.
- Если диспетчер блокировок находит взаимоблокировки, интервал обнаружения взаимоблокировок снижается с 5 секунд до 100 миллисекунд в зависимости от частоты взаимоблокировок.
- Если поток диспетчера блокировки прекращает поиск взаимоблокировок, компонент Database Engine увеличивает интервал до 5 секунд.
- Если взаимоблокировка была только что найдена, предполагается, что следующие потоки, которые должны ожидать блокировки, входят в цикл взаимоблокировки. Первая пара элементов, ожидающих блокировки, после того как взаимоблокировка была обнаружена, запускает поиск взаимоблокировок вместо того, чтобы ожидать следующий интервал обнаружения взаимоблокировки. Например, если текущее значение интервала равно 5 секунд и была обнаружена взаимоблокировка, следующий ожидающий блокировки элемент немедленно приводит в действие детектор взаимоблокировок. Если этот ожидающий блокировки элемент является частью взаимоблокировки, она будет обнаружена немедленно, а не во время следующего поиска взаимоблокировок.
- Компонент Database Engine обычно выполняет только периодическое обнаружение взаимоблокировок. Так как число взаимоблокировок, произошедших в системе, обычно мало, периодическое обнаружение взаимоблокировок помогает сократить издержки от взаимоблокировок в системе.
- Если монитор блокировок запускает поиск взаимоблокировок для определенного потока, он идентифицирует ресурс, ожидаемый потоком. После этого монитор блокировок находит владельцев определенного ресурса и рекурсивно продолжает поиск взаимоблокировок для этих потоков до тех пор, пока не найдет цикл. Цикл, определенный таким способом, формирует взаимоблокировку.
- После обнаружения взаимоблокировки компонент Database Engine завершает взаимоблокировку, выбрав один из потоков в качестве жертвы взаимоблокировки. Компонент Database Engine прерывает выполняемый в данный момент пакет потока, производит откат транзакции жертвы взаимоблокировки и возвращает приложению ошибку 1205. Откат транзакции жертвы взаимоблокировки снимает все блокировки, удерживаемые транзакцией. Это позволяет транзакциям потоков разблокироваться, и продолжить выполнение. Ошибка 1205 жертвы взаимоблокировки записывает в журнал ошибок сведения обо всех потоках и ресурсах, затронутых взаимоблокировкой.

Как выбирается жертва взаимоблокировки?

По умолчанию в качестве жертвы взаимоблокировки выбирается сеанс, выполняющий ту транзакцию, откат которой потребует меньше всего затрат. В качестве альтернативы

пользователь может указать приоритет сеансов, используя инструкцию SET DEADLOCK_PRIORITY. DEADLOCK_PRIORITY может принимать значения LOW, NORMAL или HIGH или в качестве альтернативы может принять любое целочисленное значение на отрезке [-10..10]. По умолчанию DEADLOCK_PRIORITY устанавливается на значение NORMAL. Если у двух сеансов имеются различные приоритеты, то в качестве жертвы взаимоблокировки будет выбран сеанс с более низким приоритетом. Если у обоих сеансов установлен одинаковый приоритет, то в качестве жертвы взаимоблокировки будет выбран сеанс, откат которого потребует наименьших затрат. Если сеансы, вовлеченные в цикл взаимоблокировки, имеют один и тот же приоритет и одинаковую стоимость, то жертва взаимоблокировки выбирается случайным образом.