**Блокировки. Совместимость блокировок. Протокол доступа к данным. Решение проблем параллелизма при помощи блокировок.**

***Блокировки***

Основная идея блокировок заключается в том, что если для выполнения некоторой транзакции необходимо, чтобы некоторый объект не изменялся без ведома этой транзакции, то этот объект должен быть заблокирован, т.е. доступ к этому объекту со стороны других транзакций ограничивается на время выполнения транзакции, вызвавшей блокировку.

Различают два типа блокировок:

* ***Монопольные блокировки*** (***X-блокировки***, ***X-locks*** - eXclusive locks) - блокировки без взаимного доступа (блокировка записи).
* ***Разделяемые блокировки*** (***S-блокировки***, ***S-locks*** - Shared locks) - блокировки с взаимным доступом (блокировка чтения).

Если транзакция A блокирует объект при помощи X-блокировки, то всякий доступ к этому объекту со стороны других транзакций отвергается.

Если транзакция A блокирует объект при помощи S-блокировки, то

* запросы со стороны других транзакций на X-блокировку этого объекта будут отвергнуты,
* запросы со стороны других транзакций на S-блокировку этого объекта будут приняты.

Доступ к объектам базы данных на чтение и запись должен осуществляться в соответствии со следующим ***протоколом доступа к данным***:

1. Прежде чем прочитать объект, транзакция должна наложить на этот объект S-блокировку.
2. Прежде чем обновить объект, транзакция должна наложить на этот объект X-блокировку. Если транзакция уже заблокировала объект S-блокировкой (для чтения), то перед обновлением объекта S-блокировка должна быть заменена X-блокировкой.
3. Если блокировка объекта транзакцией B отвергается оттого, что объект уже заблокирован транзакцией A, то транзакция B переходит в ***состояние ожидания***. Транзакция B будет находиться в состоянии ожидания до тех пор, пока транзакция A не снимет блокировку объекта.
4. X-блокировки, наложенные транзакцией A, сохраняются до конца транзакции A.

***Решение проблем параллелизма при помощи блокировок***

Рассмотрим, как будут себя вести транзакции, вступающие в конфликт при доступе к данным, если они подчиняются протоколу доступа к данным.

***Проблема потери результатов обновления***

Две транзакции по очереди записывают некоторые данные в одну и ту же строку и фиксируют изменения.

Результат. Обе транзакции ожидают друг друга и не могут продолжаться. Возникла ситуация *тупика*.

***Проблема незафиксированной зависимости (чтение "грязных" данных, неаккуратное считывание)***

Транзакция B изменяет данные в строке. После этого транзакция A читает измененные данные и работает с ними. Транзакция B откатывается и восстанавливает старые данные. Результат. Транзакция A притормозилась до окончания (отката) транзакции B. После этого транзакция A продолжила работу в обычном режиме и работала с правильными данными. Конфликт разрешен за счет некоторого увеличения времени работы транзакции A (потрачено время на ожидание снятия блокировки транзакцией B).

***Проблема несовместимого анализа***

***Неповторяемое считывание***

Транзакция A дважды читает одну и ту же строку. Между этими чтениями вклинивается транзакция B, которая изменяет значения в строке.

Результат. Транзакция B притормозилась до окончания транзакции A. В результате транзакция A дважды читает одни и те же данные правильно. После окончания транзакции A, транзакция B продолжила работу в обычном режиме.

***Фиктивные элементы (фантомы)***

Транзакция A дважды выполняет выборку строк с одним и тем же условием. Между выборками вклинивается транзакция B, которая добавляет новую строку, удовлетворяющую условию отбора.

Результат. Блокировка на уровне строк не решила проблему появления фиктивных элементов.

***Собственно несовместимый анализ***

Длинная транзакция выполняет некоторый анализ по всей таблице, например, подсчитывает общую сумму денег на счетах клиентов банка для главного бухгалтера. Пусть на всех счетах находятся одинаковые суммы, например, по $100. Короткая транзакция в этот момент выполняет перевод $50 с одного счета на другой так, что общая сумма по всем счетам не меняется.

Результат. Обе транзакции ожидают друг друга и не могут продолжаться. Возникла ситуация *тупика*.

***Разрешение тупиковых ситуаций***

Итак, при использовании протокола доступа к данным с использованием блокировок часть проблем разрешилось (не все), но возникла новая проблема - тупики:

* Проблема потери результатов обновления - *возник тупик*.
* Проблема незафиксированной зависимости (чтение "грязных" данных, неаккуратное считывание) -*проблема разрешилась*.
* Неповторяемое считывание - *проблема разрешилась*.
* Появление фиктивных элементов - *проблема не разрешилась*.
* Проблема несовместимого анализа - *возник тупик*.

Методом разрешения тупиковой ситуации является откат одной из транзакций (транзакции-жертвы) так, чтобы другие транзакции продолжили свою работу. После разрешения тупика, транзакцию, выбранную в качестве жертвы можно повторить заново.

Можно представить два принципиальных подхода к обнаружению тупиковой ситуации и выбору транзакции-жертвы:

1. СУБД не следит за возникновением тупиков. Транзакции сами принимают решение, быть ли им жертвой.

За возникновением тупиковой ситуации следит сама СУБД, она же принимает решение, какой транзакцией пожертвовать.