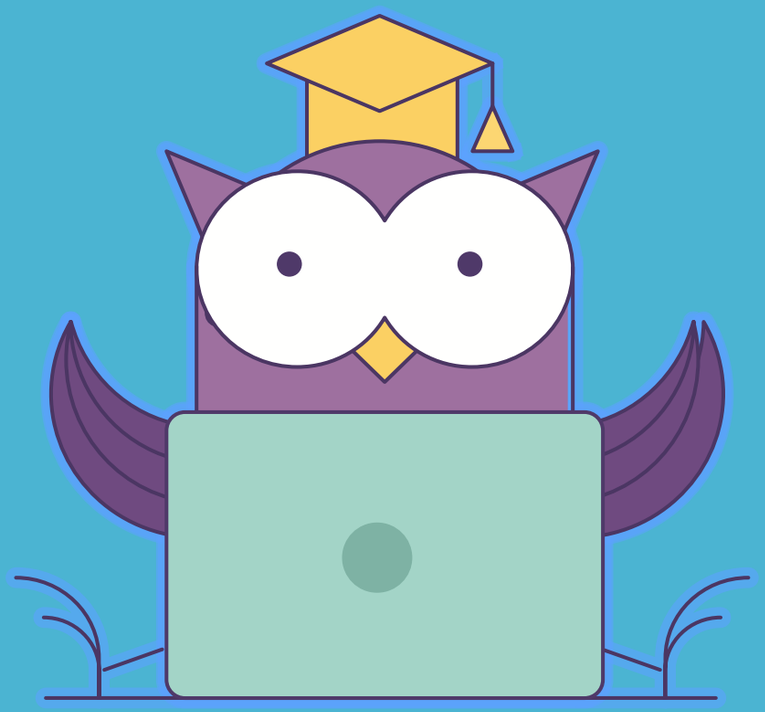
ОНЛАЙН-ОБРАЗОВАНИЕ

Транзакции

Иван Ремень

Как меня слышно и видно?



> Напишите в чат

+ если все хорошо

– если есть проблемы со звуком или с видео

Цель занятия

- Понять, что такое транзакции
- Узнать об уровнях изоляции

Реальность

- ПО или аппартное обеспечение могут отказать в любое время
- В любой момент может произойти фатальный сбой ПО
- Может случиться разрыв сети
- Клиенты работают с базой параллельно, могут перезаписывать друг друга
- Клиенты могут читать частично обновленные состояния

Транзакции

Транзакция — способ группировки приложением нескольких операций записи и чтения в одну логическую единицу.
По сути, все операции записи и чтения в ней выполняются как одна: вся транзакция или целиком выполняется успешно (с фиксацией изменений), или целиком завершается неудачно (с прерыванием и откатом).
Появились в середине 1970-х, с тех пор не сильно менялись.
В NoSQL от них часто отказываются.

- atomicity
- consistency
- isolation
- durability

Есть еще понятие BASE (Basically Available, Soft state, Eventual consistancy)
Во многом это маркетинговые термины.

Атомарность ознает "все или ничего". Гарантирует, что либо все операции транзакции будут успешно применены, либо не будет применена ниодна из них.
Благорадя этому появляется возможность повторять прерванные транзакции, не опасаясь что часть операций уже была выполнена.

Определяются инварианты системы:

- Дебит должен сходится с кредитом.
- Цена проданных билетов должна соответствовать выручке за сеанс.

В реальность означает только то, что constaints будут выполнятся.
По сути, поддержание согласованности - задача приложения, а не базы.

Isolation

Изоляция - свойство, которое позволяет выполнять параллельные транзакции как последовательные.
Существует несколько уровней изоляции, которые дают различные гарантии.
В полном смысле изоляцию обеспечивает только уровень SERIALIZABLE.

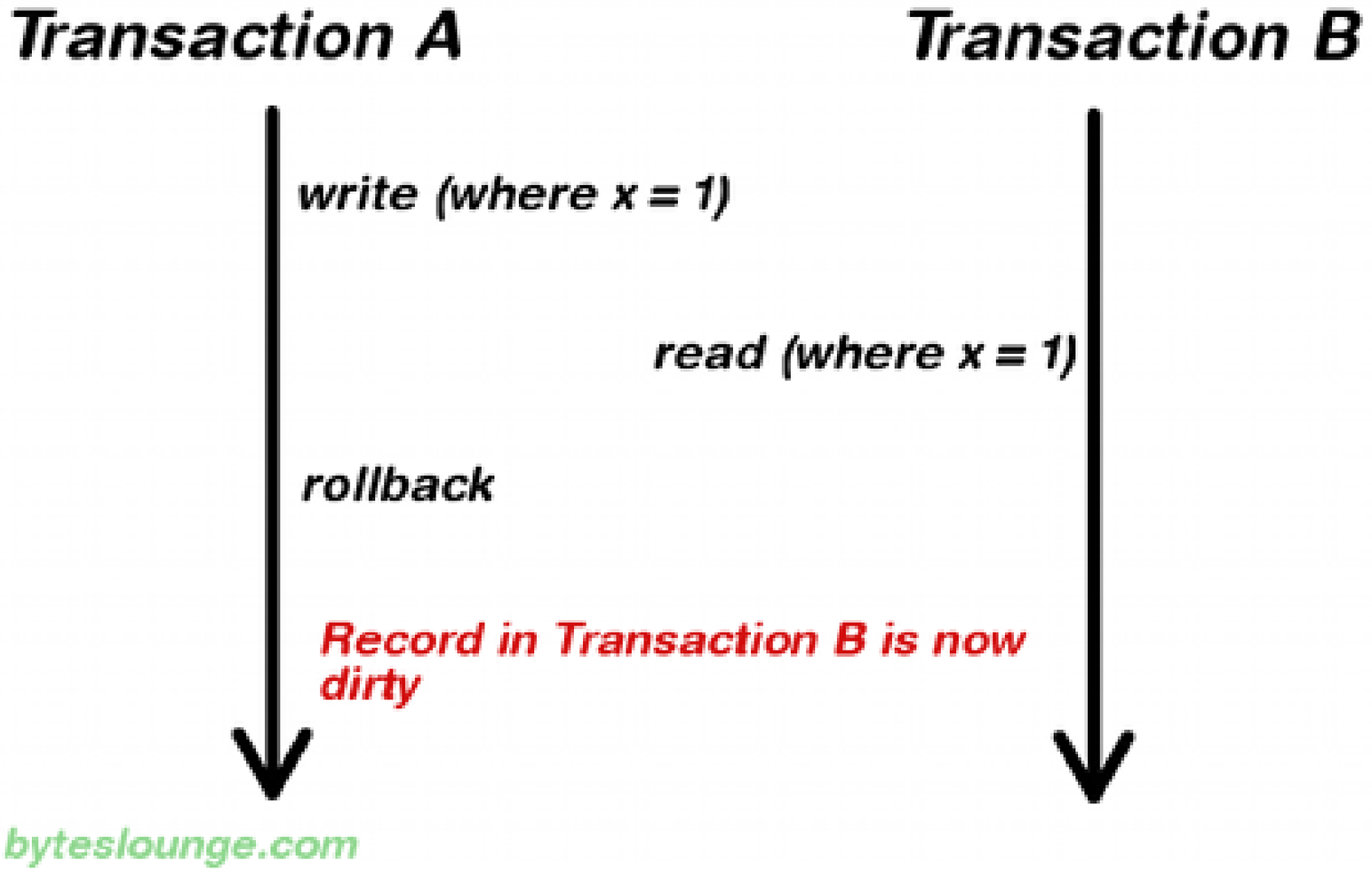
Durability

Под этим свойством подразумивают только одно - данные при отдаче ответа зафиксированны в энергонезависимой памяти (fsync).
Полностью это свойство обеспечить невозможно. Диски сбоят. Но можно повышать (репликация) гарантии.

Атомрные операции

Многие NoSQL базы, хоть и не поддерживают транзакции, обладают возможностью атомарных операций.
Для memcached это:

- inc
- dec


Transaction A
Transaction B
write (where x = 1)
read (where x = 1)
rollback
Record in Transaction B is now dirty
byteslounge.com

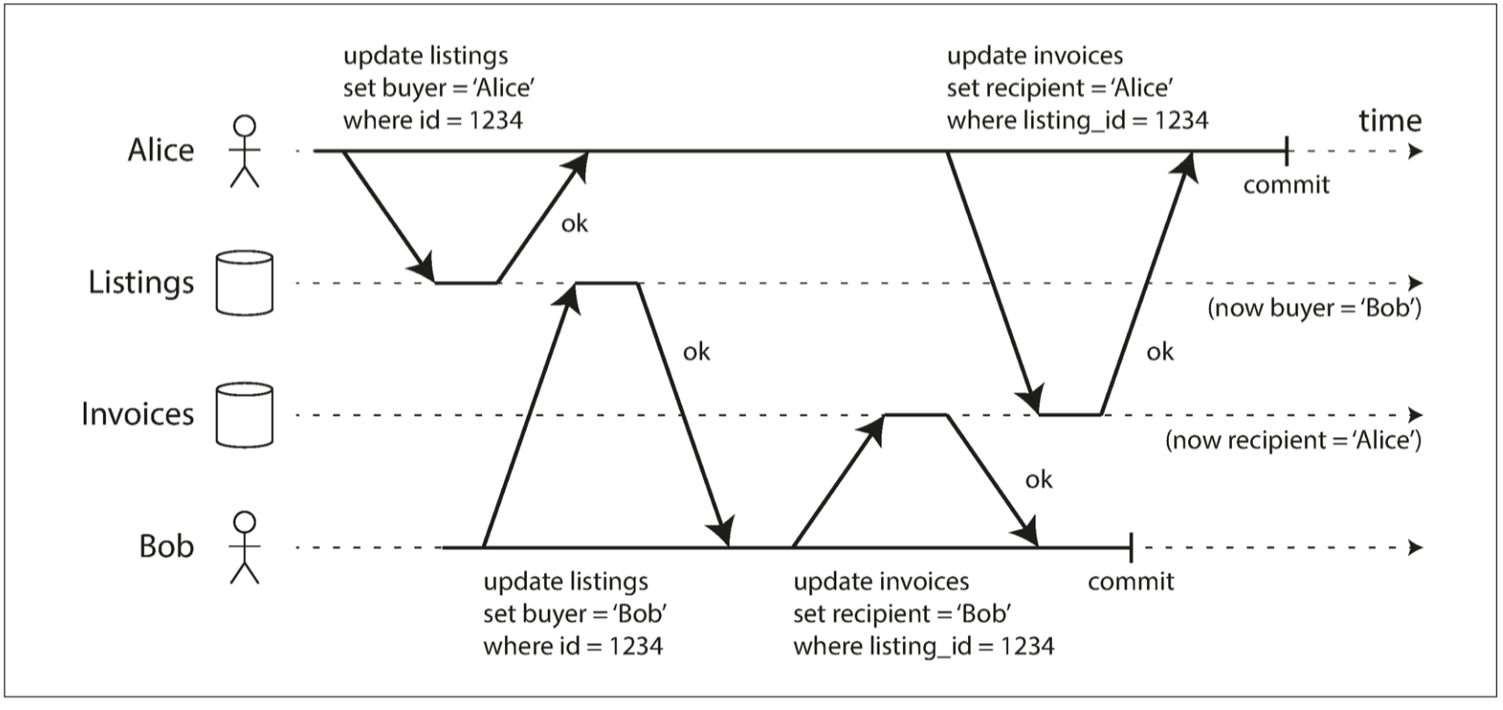
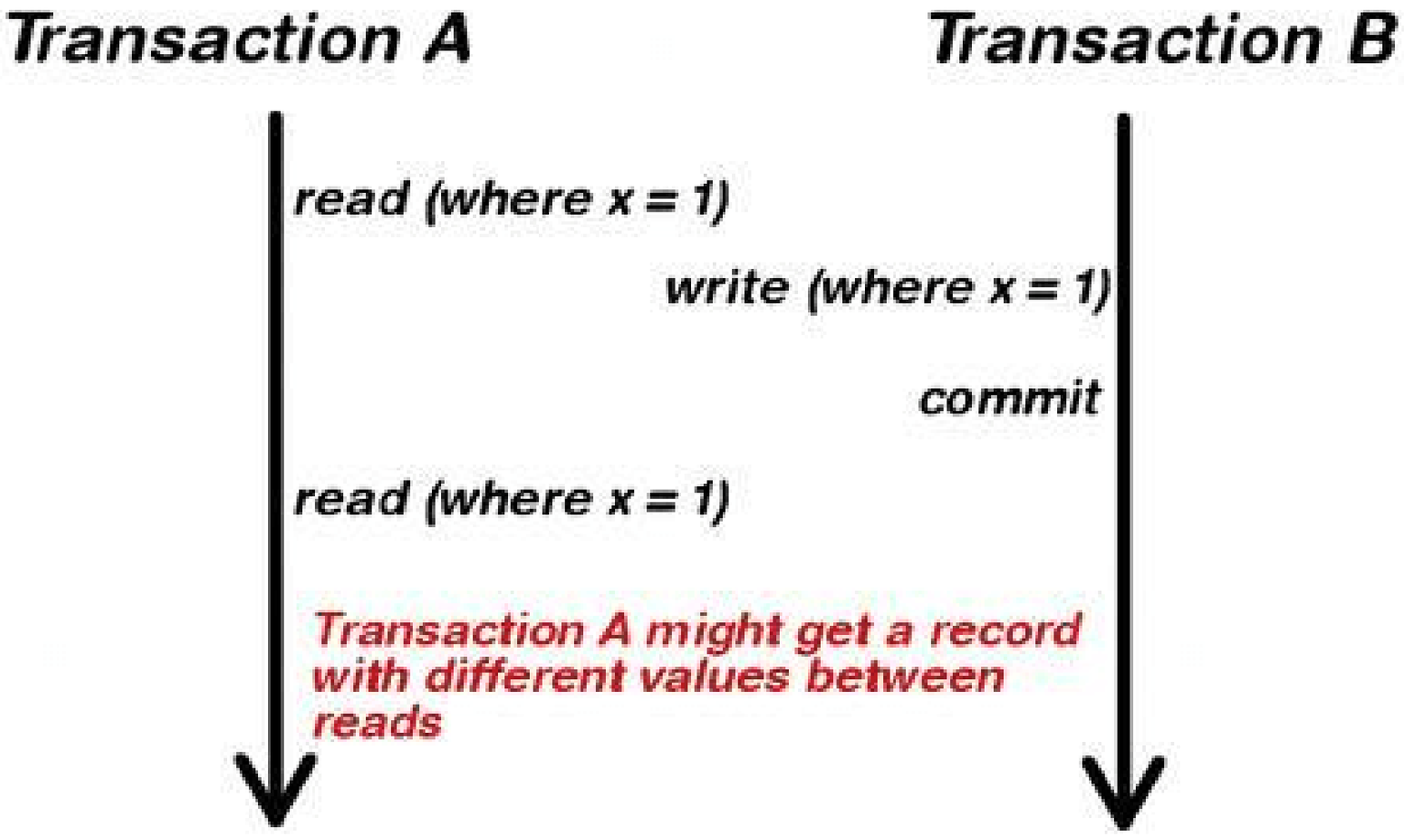


Figure 7-5. With dirty writes, conflicting writes from different transactions can be mixed up.

- Нет dirty reads
- Нет dirty writes

Реализация:

- Блокировка строк при записи
- Сохраняем после записи старые значения строк до коммита изменений.


Transaction A
Transaction B
read (where x = 1)
write (where x = 1)
commit
read (where x = 1)
Transaction A might get a record with different values between reads

REPEATABLE READS

- Нет unrepeatable reads

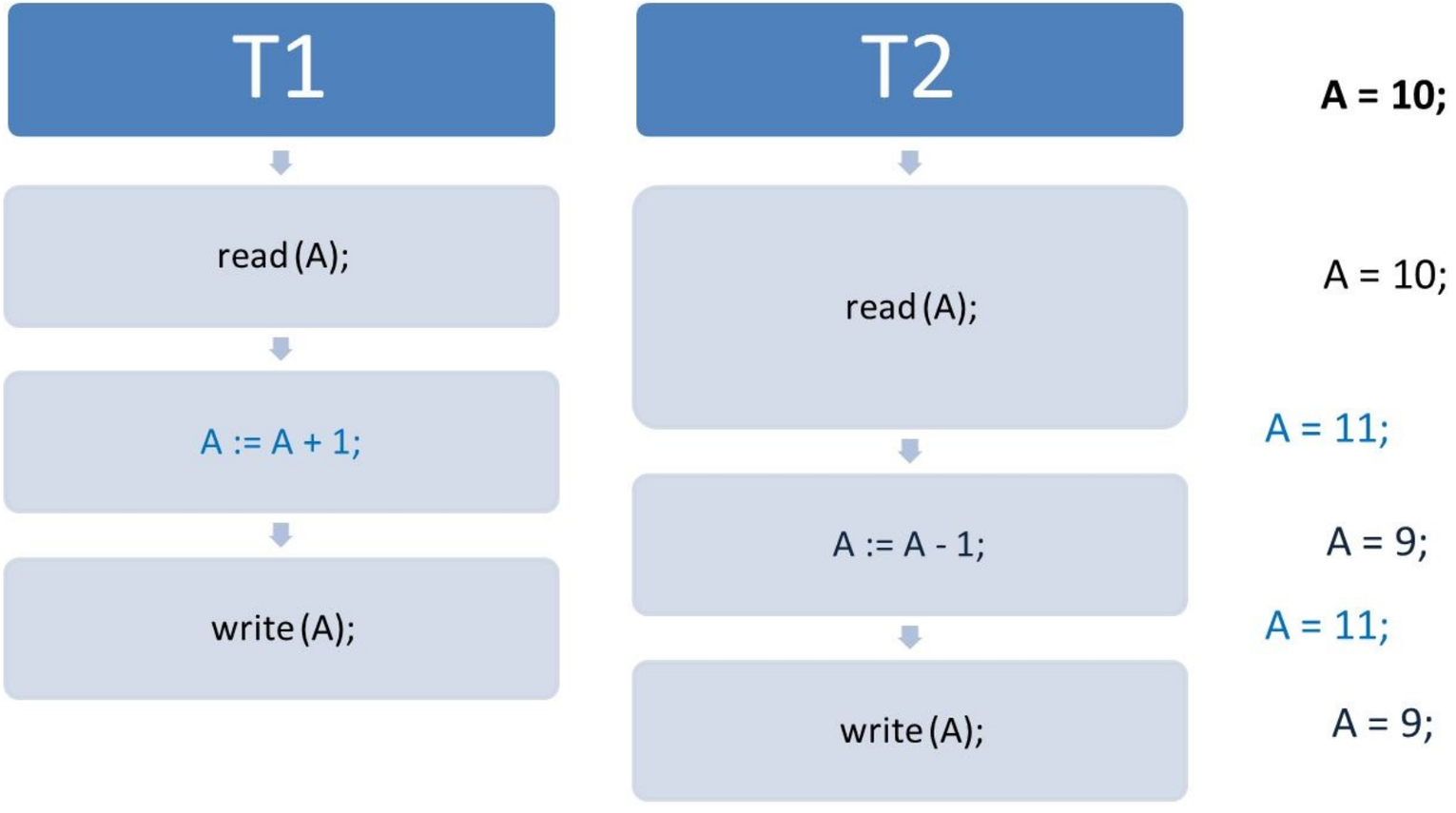
Реализация:

- Блокировки (чтение не блокирует запись, а запись не блокирует чтение)
- MVCC

В некоторых базах есть путаница между repeatable reads и serealizable.

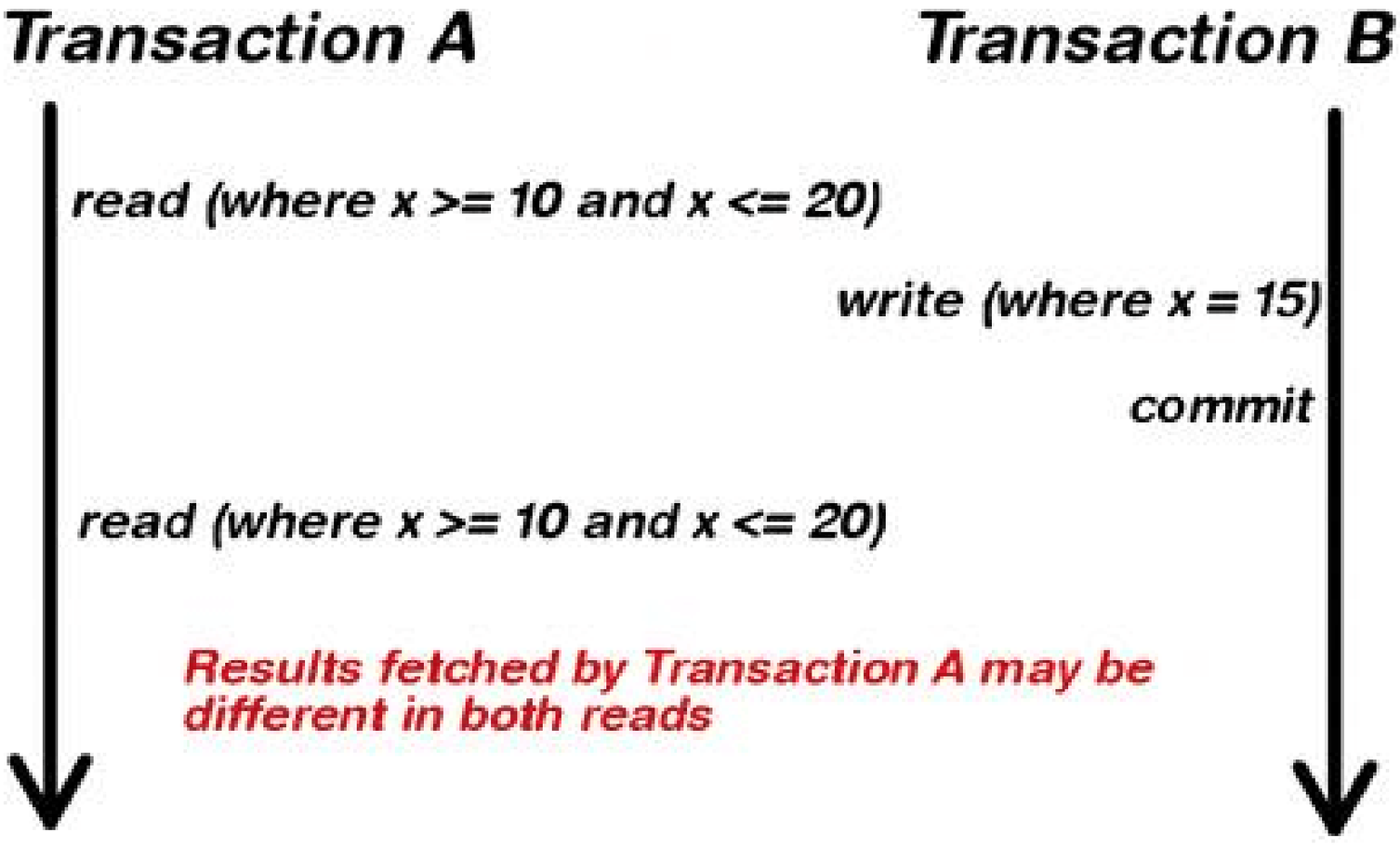
- В начале каждой транзакции база данных создает список всех остальных выполняемых на текущий момент транзакций (но еще не зафиксированных или прерванных). Все выполненные этими транзакциями изменения игнорируются, даже если впоследствии они будут зафиксированы.
- Все операции записи, выполненные прерванными транзакциями, игнорируются.
- Все операции записи, выполненные транзакциями с более поздним идентификатором транзакции (то есть начавшиеся после запуска текущей транзакции), игнорируются независимо от того, были ли они зафиксированы.
- Результаты всех остальных операций записи видны запросам приложения.

Сложность - работа с индексами. Вариант решения - не модифицировать страницы, а создавать новые влоть до корня. Еще одно решение - хранить в индексе все версии.


T1
read (A);
A := A + 1;
write (A);
T2
read (A);
A := A - 1;
write (A);
A = 10;
A = 10;
A = 11;
A = 9;
A = 11;
A = 9;

Защита от потерянных обновлений

- Атомарные операции
- Явные блокировки (SELECT * FROM tbl WHERE id = 10 FOR UPDATE)
- Автоматическое обнаружение
- CAS-операции (UPDATE wiki_pages SET content = 'new content' WHERE id = 1234 AND content = 'old content'), но не будет работать если есть REPEATABLE READ.


Transaction A
Transaction B
read (where x >= 10 and x <= 20)
write (where x = 15)
commit
read (where x >= 10 and x <= 20)
Results fetched by Transaction A may be different in both reads

Варианты защиты от фантомного чтения

- SERIALIZABLE изоляция
- Материализация фантомов

Задача

Как решить с помощью материализации фантомов следующую задачу:
Нужно блокировать переговорную комнату.

- Последовательное выполнение
- Двухфазная блокировка
- Методы оптимистичного управления конкурентным доступом

Пример - tarantool, redis.
Хорошо подходит для In-Memory баз данных. Крайне рекомендуется использовать шардинг (но кросс-шард запросы надо синхронизировать).
Есть комбинированное решение. OLAP - выносится в отдельные потоки выполнения с MVCC.

Не путайте с 2-Phase Commit
Требуется предекативная блокировка или блокировка по диапазонам.
Есть риск deadlock'ов.

Вопросы?

Спасибо за внимание!