Атомарность и модели транзакций

Содержание

- Реализация атомарности для операции WRITE
- Классификация выполняемых действий
- Модели транзакций

Атомарная запись

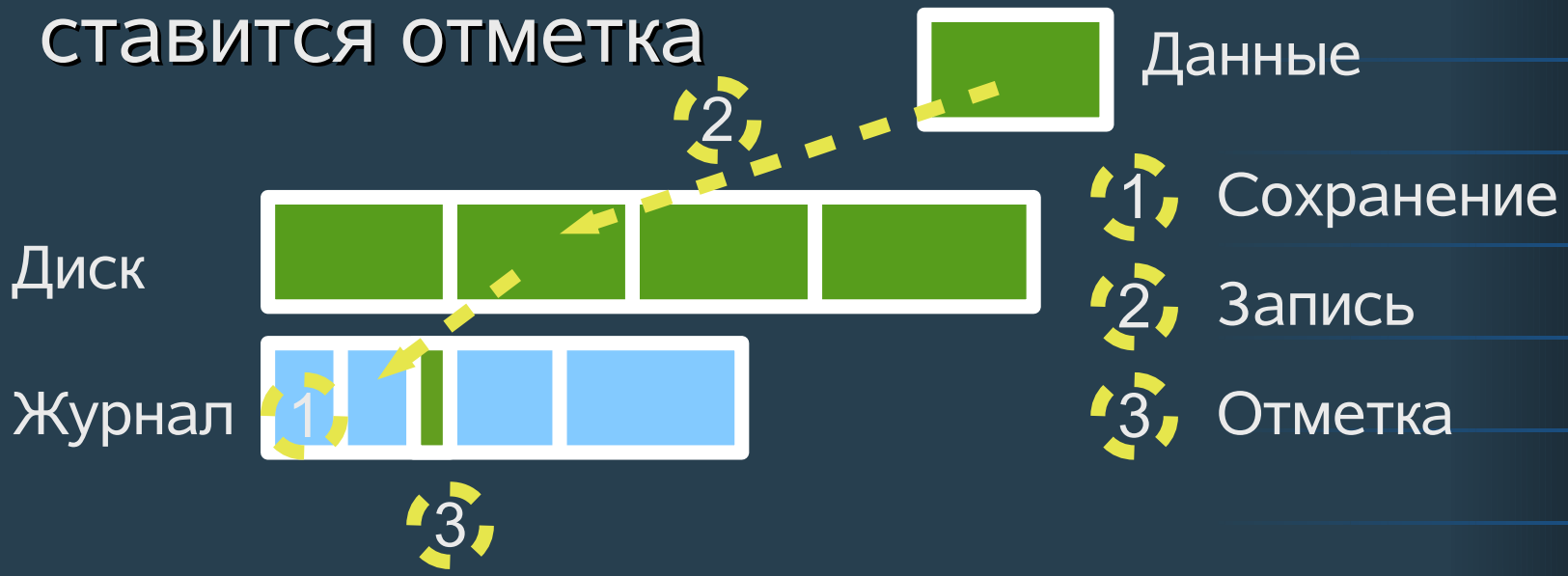
- Обычная запись
- Запись с контрольным чтением
- Дублированная запись (зеркалирование)
- Журнализированная запись

Дублированная запись

- Последовательная запись для предотвращения потери данных

Журналированная запись

- Информация для восстановления записывается в журнал
- В случае нормального завершения ставится отметка



- Откатываются блоки без отметки

Классификация действий

Insert into .... (read, write, write)

select .... from ... (read, read ..)

DRILL HOLE

Update account set ..... (read, write)

If (success)

COMMIT

Завершение работы

else

ROLLBACK

Откатываем изменения

Классификация действий

- Незащищенные действия
  - Отсутствуют все свойства транзакций, кроме непротиворечивости
  - Не являются атомарными
  - Эффект от выполнения не является зависимым
  - Являются обратимыми
- Примеры:
  - Операция WRITE
  - Операция DRAW LINE

Классификация действий

- Защищенные действия
  - Имеют ACID свойства
  - Не объявляют своих результатов до завершения
  - Изменения согласовано управляются
  - Могут быть откатаны в случае ошибки
  - Не могут быть откатанны в одностороннем порядке после успешного завершения
- Примеры
  - Операторы DML в языке SQL

Классификация действий

- Реальные(настоящие) действия
  - Происходят в реальном мире
  - Имеют CI свойства
  - Необратимы в основном
- Примеры
  - Пуск ракеты
  - Сверление дырок

Правила использования

- Незащищенные действия включаются в состав защищенных более высокого уровня
- При реализации атомарных действий необходимо знать о механизмах отката нижнего уровня
- Реальные действия откладываются до момента, когда откат невозможен
- Защищенные действия – основные строительные блоки приложения

Модели транзакций

Плоские транзакции (Flat)

- BEGIN TRANSACTION
- COMMIT
- ROLLBACK

- Только один уровень управления
  - COMMIT или ROLLBACK
- Нельзя выполнить
  - Частичный ROLLBACK
  - COMMIT за несколько шагов

Ограничения плоских транзакций

- Примеры приложений
  - Резервирование билетов
  - Массовое обновление
  - Распределенные изменения
- Причины появления других моделей
  - Сложная логика приложений
  - Распределенная и параллельная обработка данных

Резервирование билетов

- Купить билет МОСКВА – С.ПЕТЕРБУРГ
- Купить билет С.ПЕТЕРБУРГ – КАЛИНГРАД
- Вернуть билет(!)
- Купить билет МОСКВА - ХЕЛЬСИНКИ

Массовое обновление

- 1 000 000 000 записей
- Обновление процентов

  UPDATE ACCOUNTS set ACCOUNT=ACCOUNT*1.055
- Сбой при обновлении 999 999 999 записи
- Откат всей полезной работы и потеря времени на сам откат

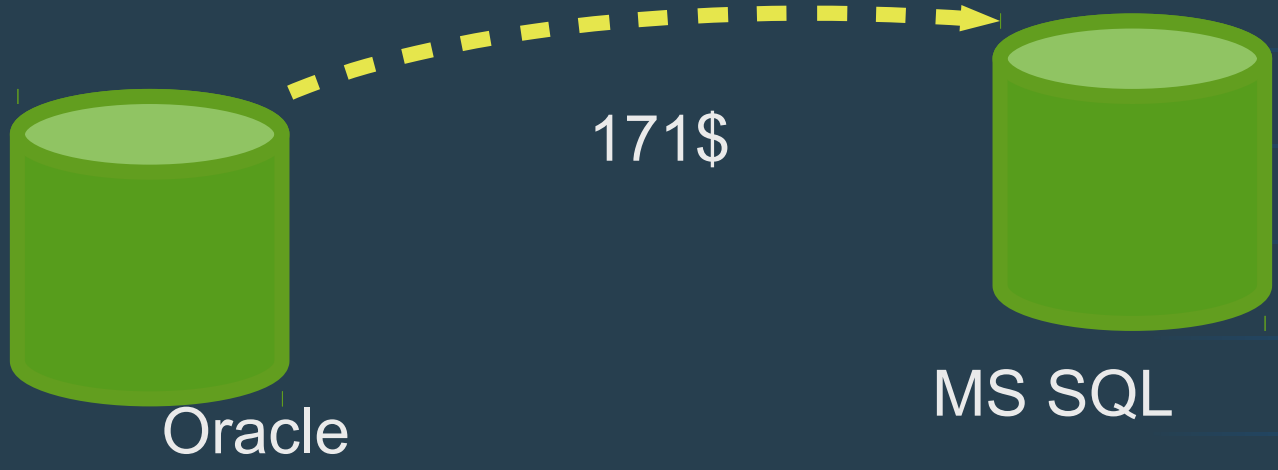
Распределенное обновление

- Перевод денег между банками

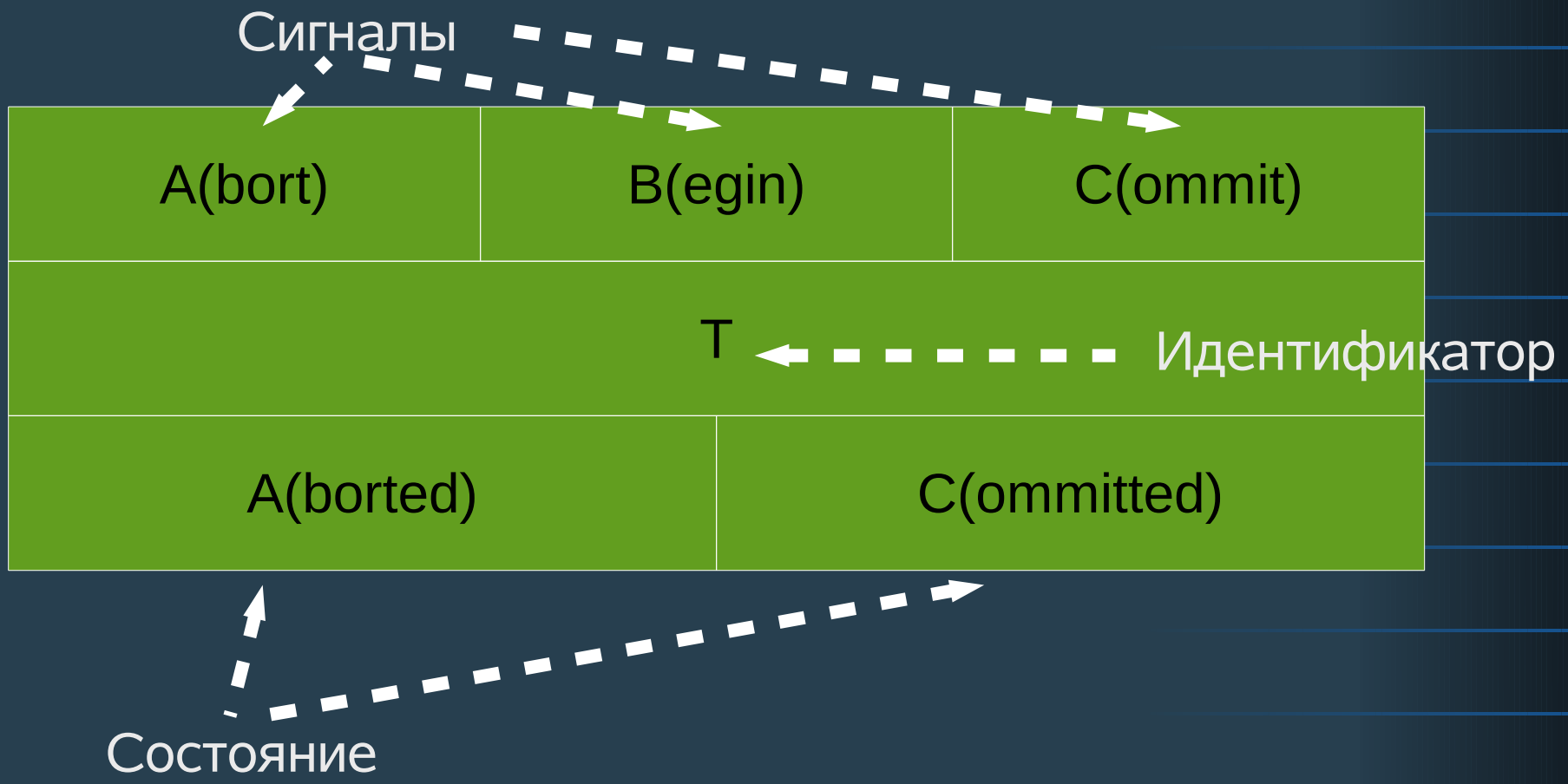
  БД1: update ACCOUNTS set ACCOUNT=ACCOUNT-171 where ID=1657;

  БД2: update ACCOUNTS set ACCOUNT=ACCOUNT+171 where ID=6571;

  БД1: COMMIT, БД2: COMMIT

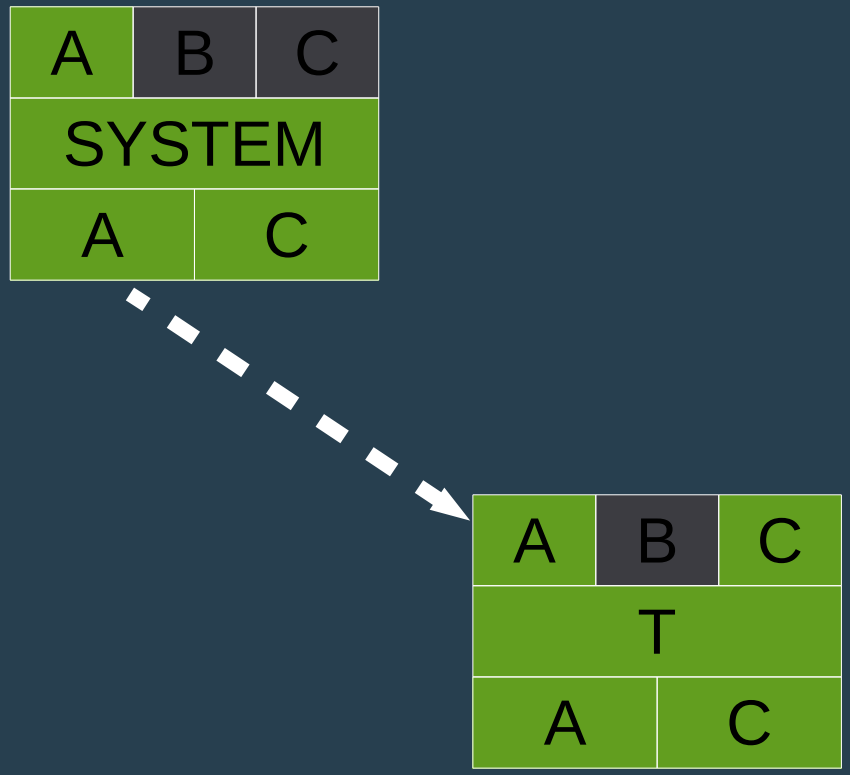


Представление транзакции



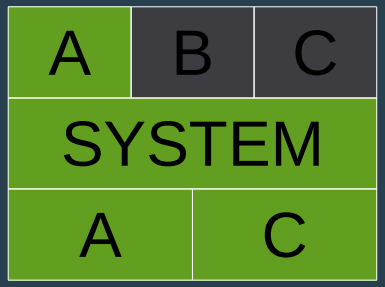
Плоские транзакции

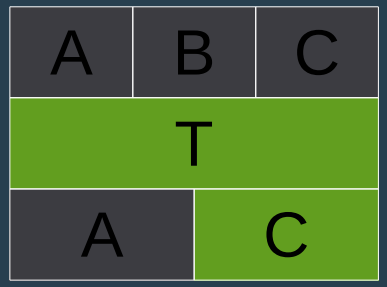
BEGIN TRANSACTION T



Плоские транзакции

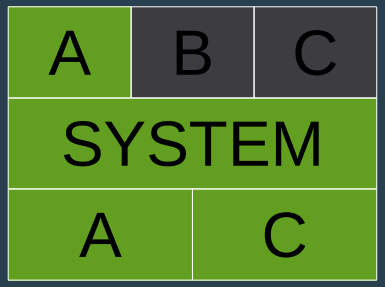
COMMIT T

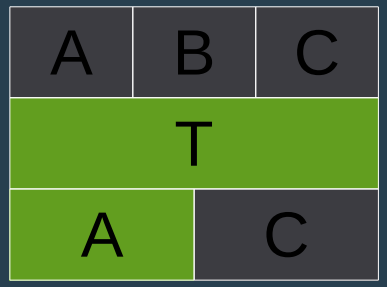




Плоские транзакции

ROLLBACK T





Плоские транзакции с контрольными точками

- Добавляются операторы
  - SAVEPOINT T,<SP>
  - ROLLBACK T,<SP>

История выполнения транзакции

BEGIN TRANSACTION: 1

action-a

action-b

SAVEPOINT: 2

action-c

SAVEPOINT: 3

action-d

action-e

action-f

SAVEPOINT: 4

action-g

ROLLBACK: 2

action-h

action-i

SAVEPOINT: 5

action-j

SAVEPOINT: 6

action-k

action-l

SAVEPOINT: 7

action-m

action-n

ROLLBACK: 7

action-o

action-p

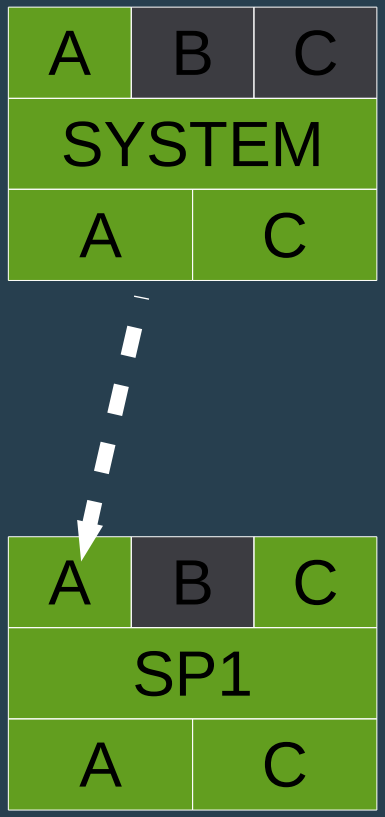
SAVEPOINT: 8

action-q

COMMIT

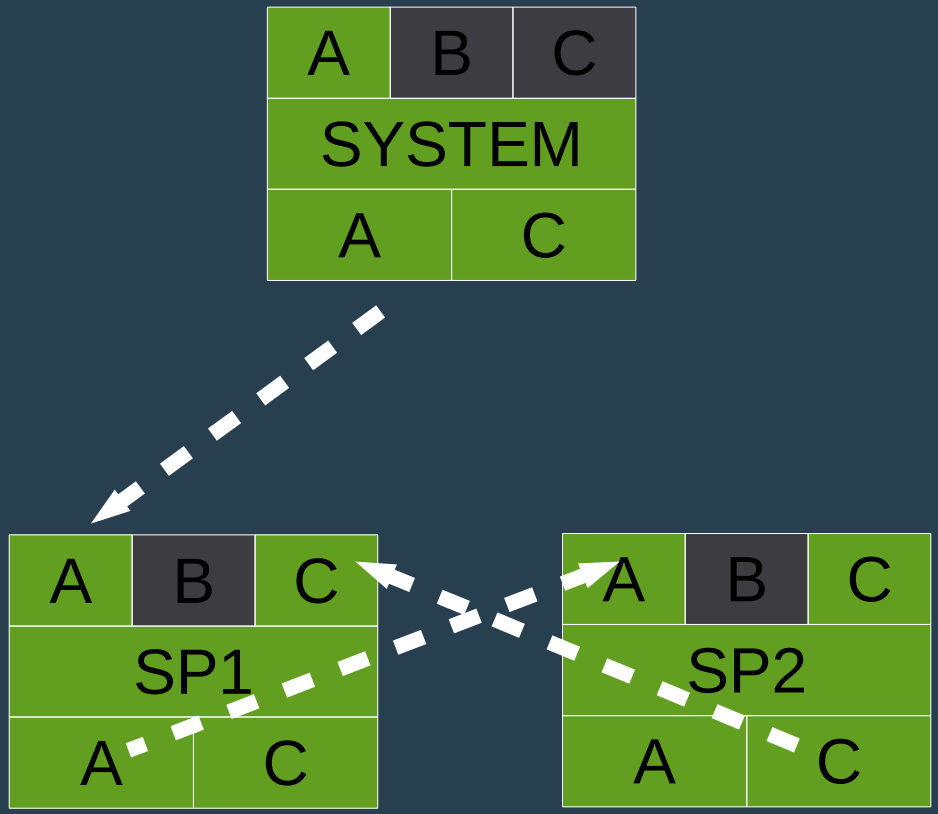
Плоские транзакции с контрольными точками

BEGIN TRANSACTION T



Плоские транзакции с контрольными точками

SAVEPOINT 2



Плоские транзакции с контрольными точками

- Достоинства
  - Возможность частичного отката
  - Каждая элементарная транзакция является плоской
- Недостатки
  - Потеря всей работы при сбое на операции массового обновления

Плоские транзакции с сохраняемыми контрольными точками

- При выполнении SAVEPOINT информация о контрольной точке сохраняется во внешнюю память
- Откатывается только последний незавершенный интервал работы
- При перезапуске после сбоя устанавливается состояние последней сохраненной точки
- Phoenix transaction

Плоские транзакции с сохраняемыми контрольными точками

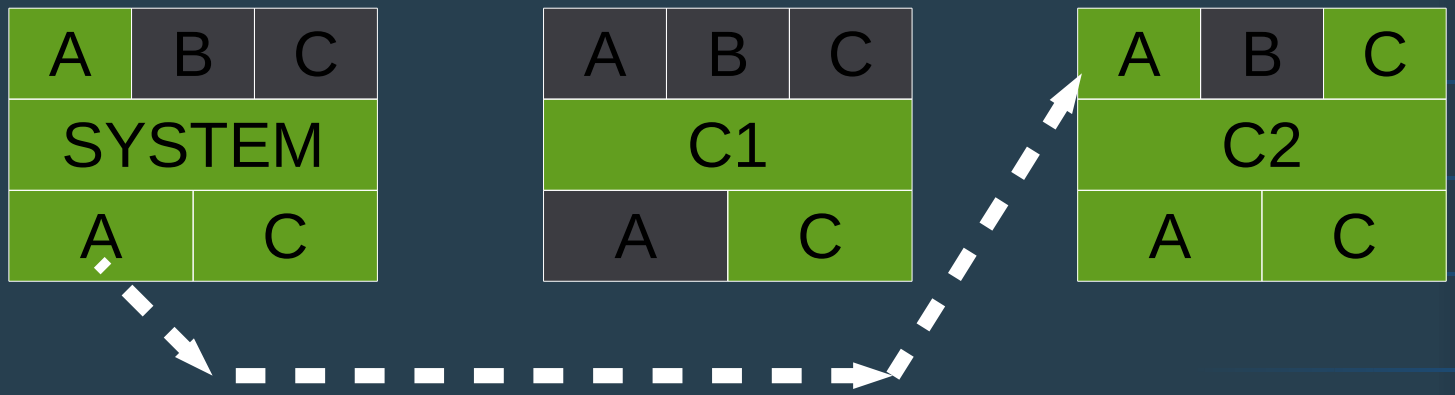
- Достоинства
  - Возможность продолжать работу
- Недостатки
  - Запись во внешнюю память
  - Необходимо API для получения текущего состояния
  - Необходимо дополнительная логика синхронизации приложения и БД

Цепочные транзакции

- Компромис между гибкостью ROLLBACK и количеством потерянной работы
- Новый оператор
  - CHAIN ~ COMMIT + BEGIN TRANSACTION
- Наследование контекста и видимость результата только в пределах транзакции
- Ни одна транзакция не может выполниться между COMMIT и BEGIN

Цепочные транзакции

CHAIN

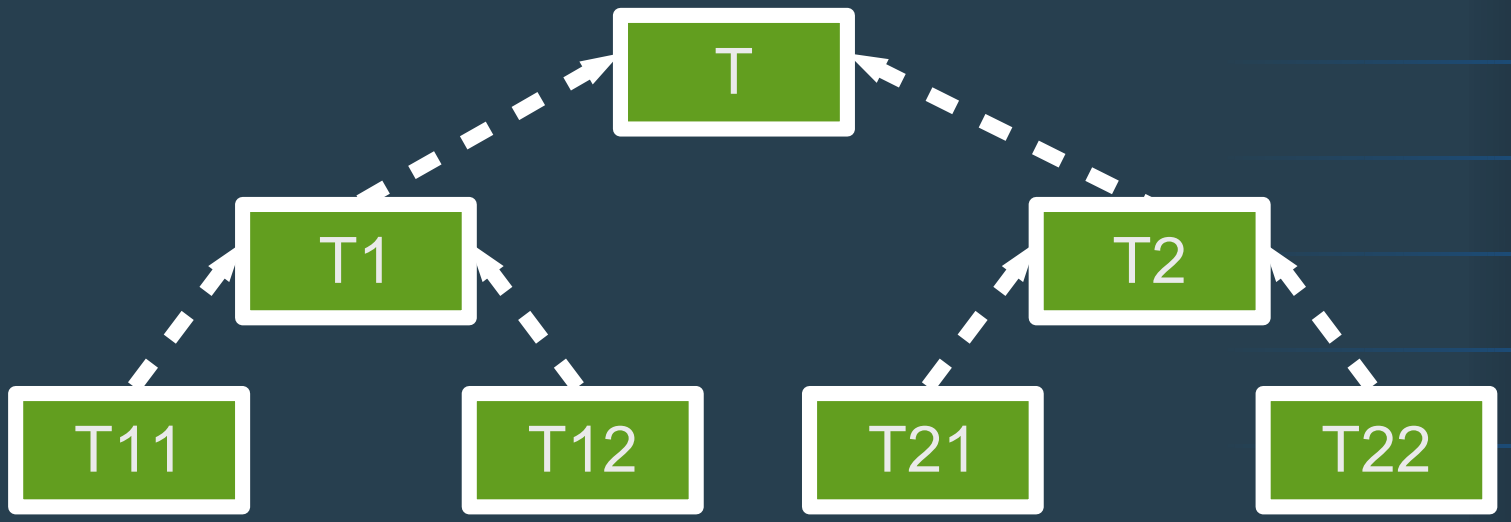


Цепочные транзакции и транзакции с контрольными точками

- Цепочные транзакции
  - Есть возможность продолжить работу
  - Нужна синхронизация приложения
  - Откатить все нельзя, дополнительное восстановление за счет логики приложения
- Транзакции с контрольными точками
  - Всегда откатывается все

Вложенные транзакции

- Обобщение контрольных точек
- Организация транзакций в виде дерева



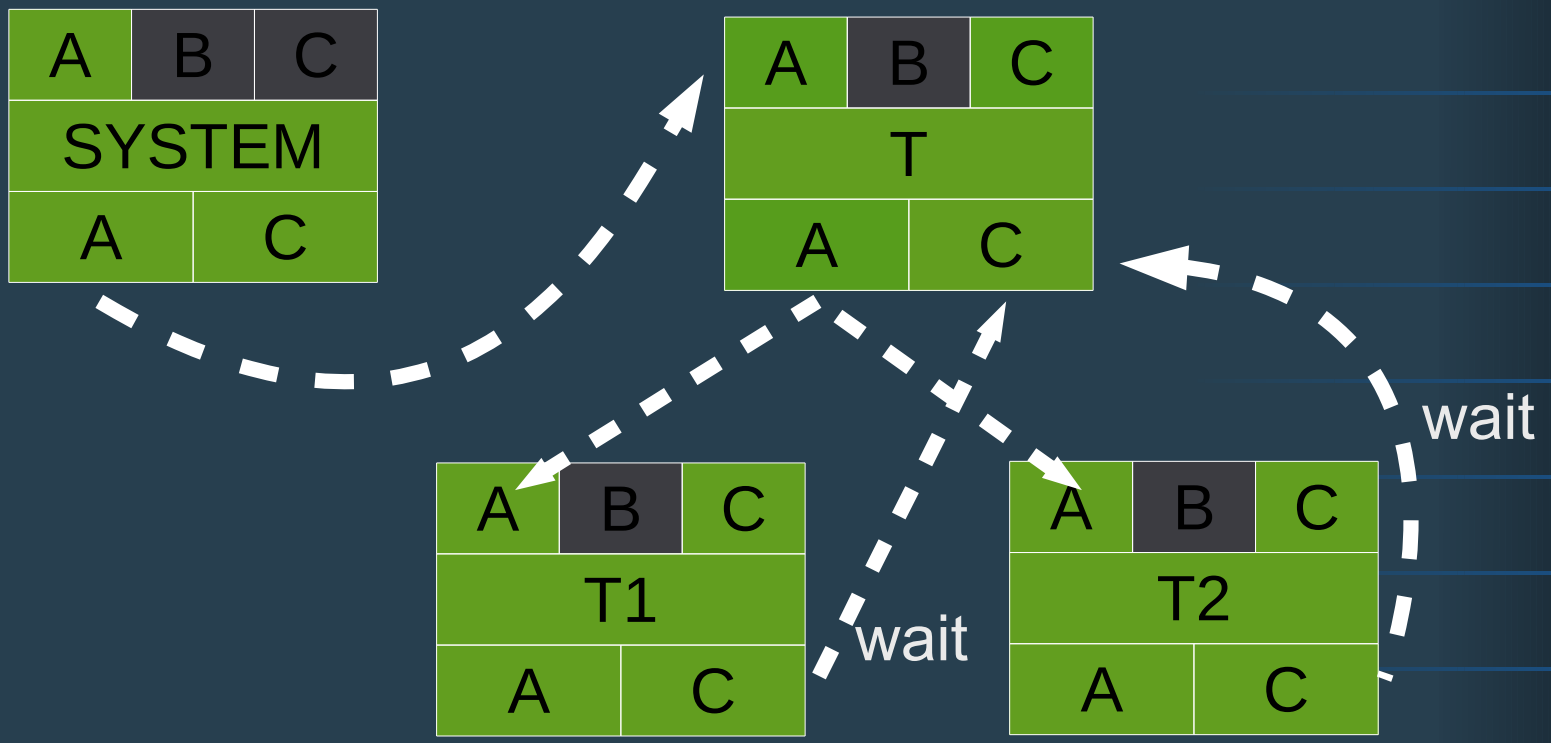
- Нижний уровень – плоские транзакции

Вложенные транзакции

- Вложенные транзакции – дерево транзакций, в котором каждое поддерево вложенная или плоская транзакция
- Листья дерева – плоские транзакции
- Транзакция в корне дерева – основная головная транзакция, остальные - субтранзакции
- При выполнении субтранзакцией COMMIT выполнение откладывается до COMMIT транзакции-предка
- Выполнение ROLLBACK вызывает откат всех субтранзакций
- Изменения, сделанные в транзакции, видны всем субтранзакциям
- Субтранзакции одной транзакции выполняются параллельно и не видят изменений друг друга

Вложенные транзакции

BEGIN TRANSACTION T, <PARENT-T>



Вложенные транзакции

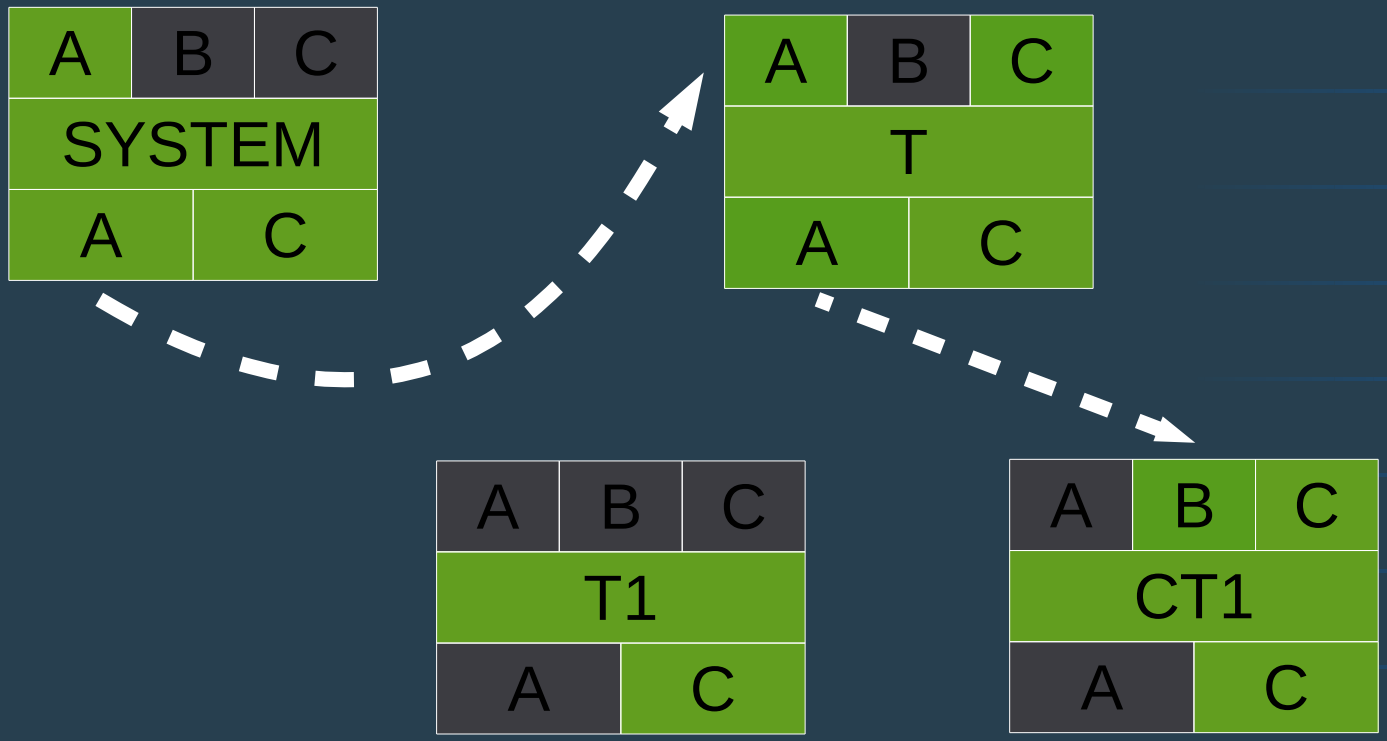
- Достоинства
  - Поддержка приложений со сложной структурой
  - Поддержка параллельной обработки данных с единым механизмом согласования
- Если обработка выполняется последовательно, то можно эмулировать с помощью контрольных точек

Многоуровневые транзакции

- Согласование в несколько этапов
  - PRECOMMIT
  - COMMIT
- PRECOMMIT – раннее согласование с возможностью отката
- COMMIT – окончательная фиксация изменений
- Реализуется с использованием компенсационных транзакций на каждом уровне вложенности

Многоуровневые транзакции

PRECOMMIT T1



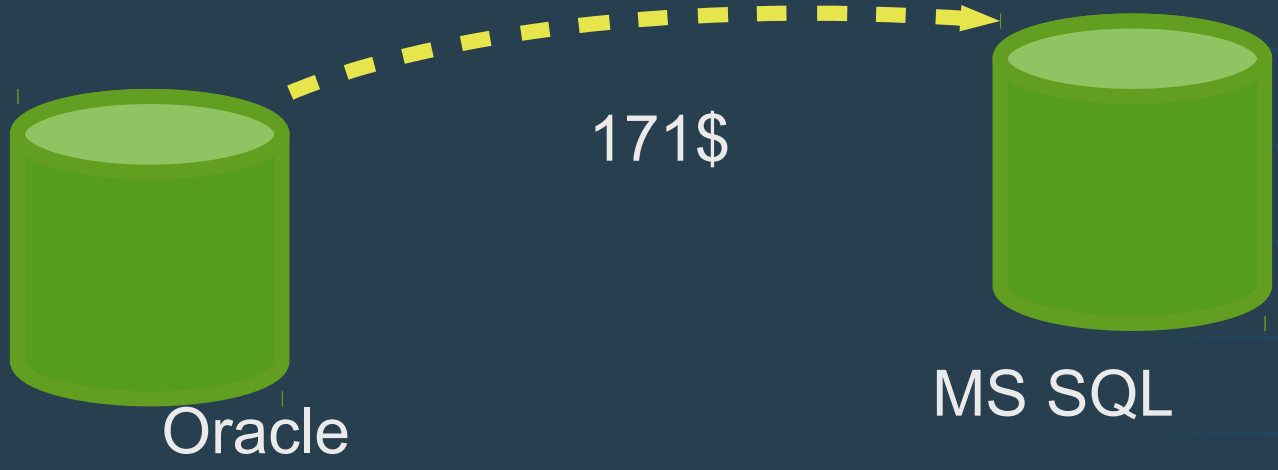
Распределенное обновление

- Перевод денег между банками

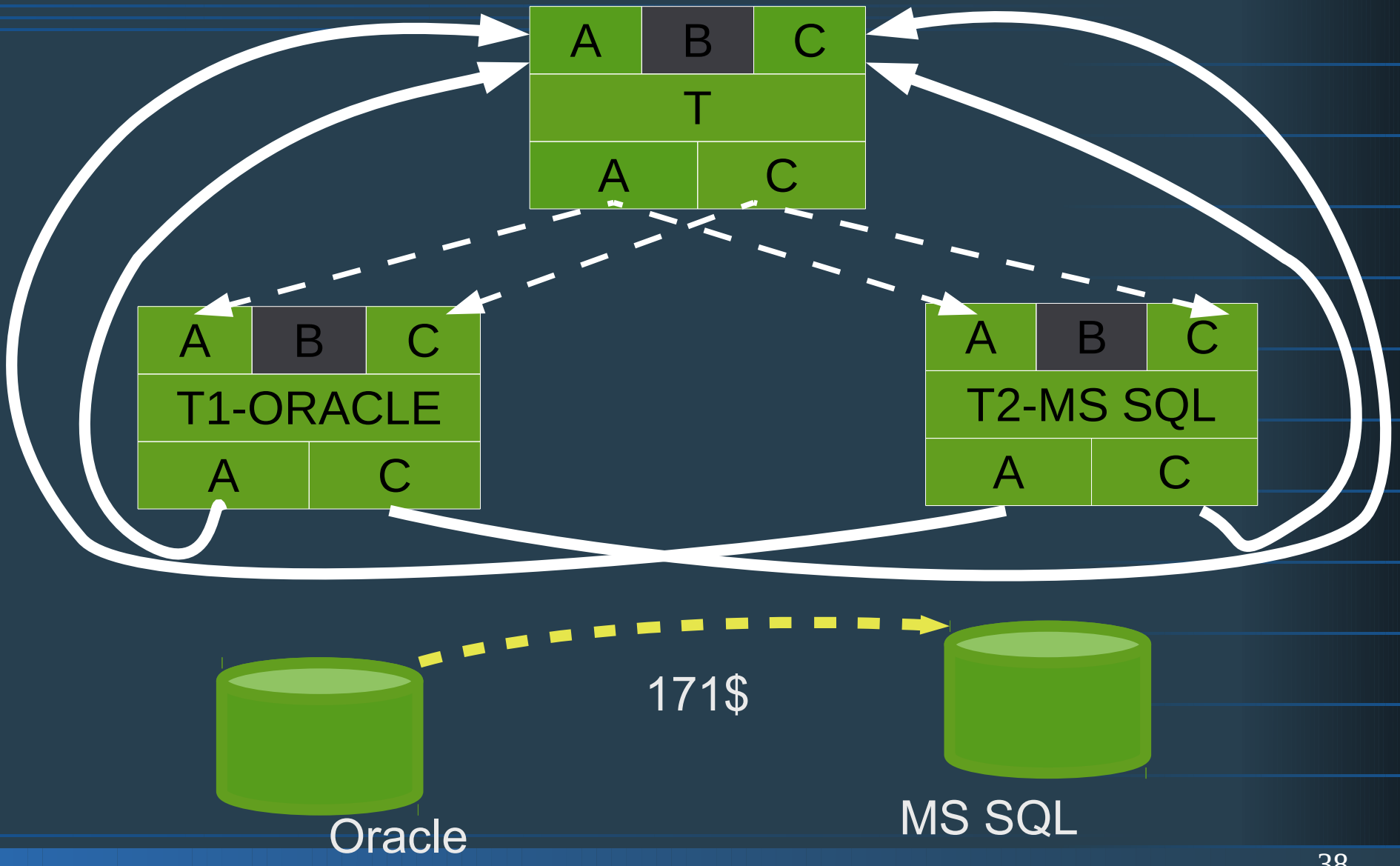
  БД1: update ACCOUNTS set ACCOUNT=ACCOUNT-171 where ID=1657;

  БД2: update ACCOUNTS set ACCOUNT=ACCOUNT+171 where ID=6571;

  БД1: COMMIT, БД2: COMMIT

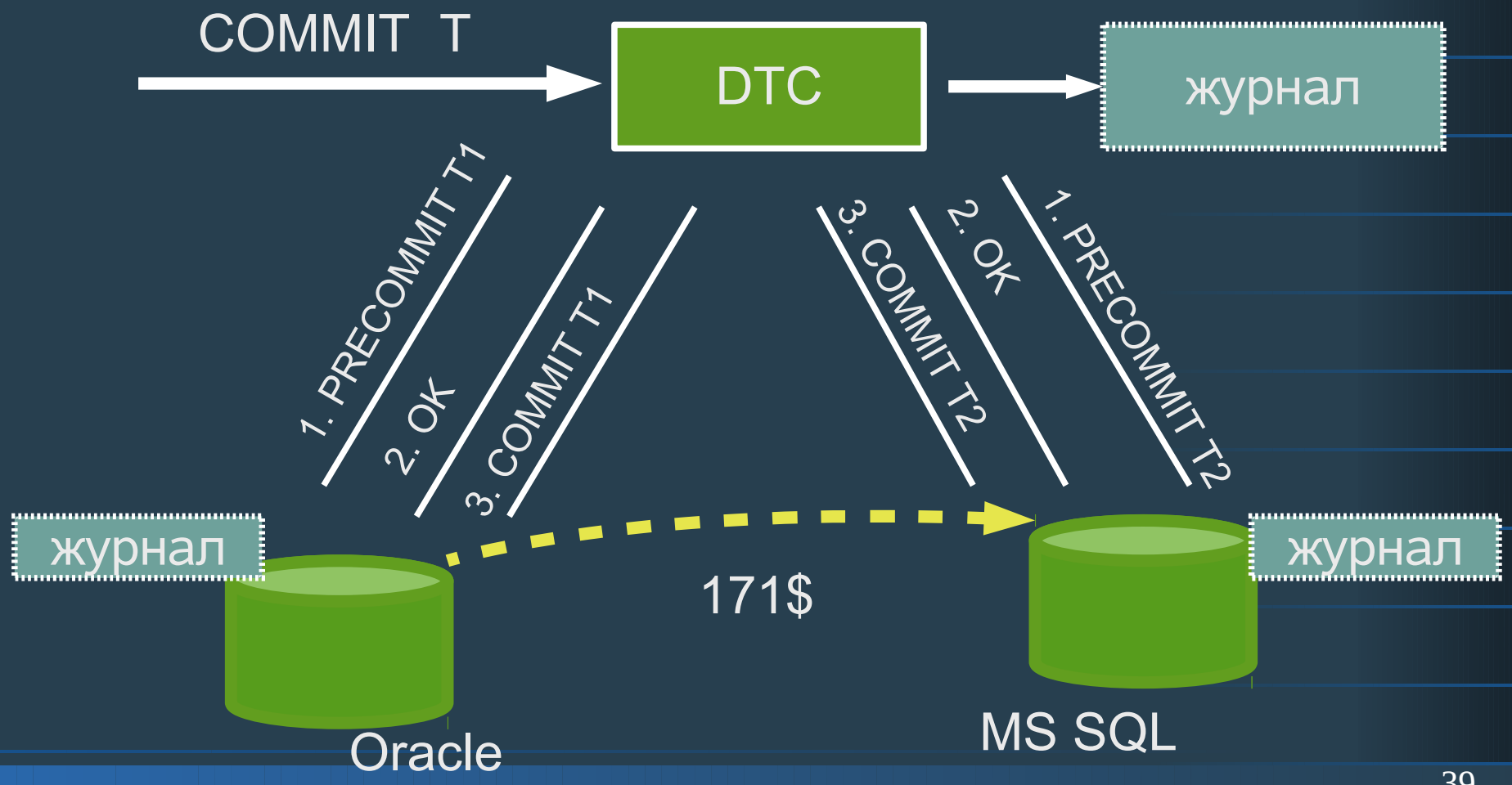


Распределенные транзакции



Двухфазный протокол согласования (2PC)

- 2PC - протокол



Двухфазный протокол согласования (2PC)

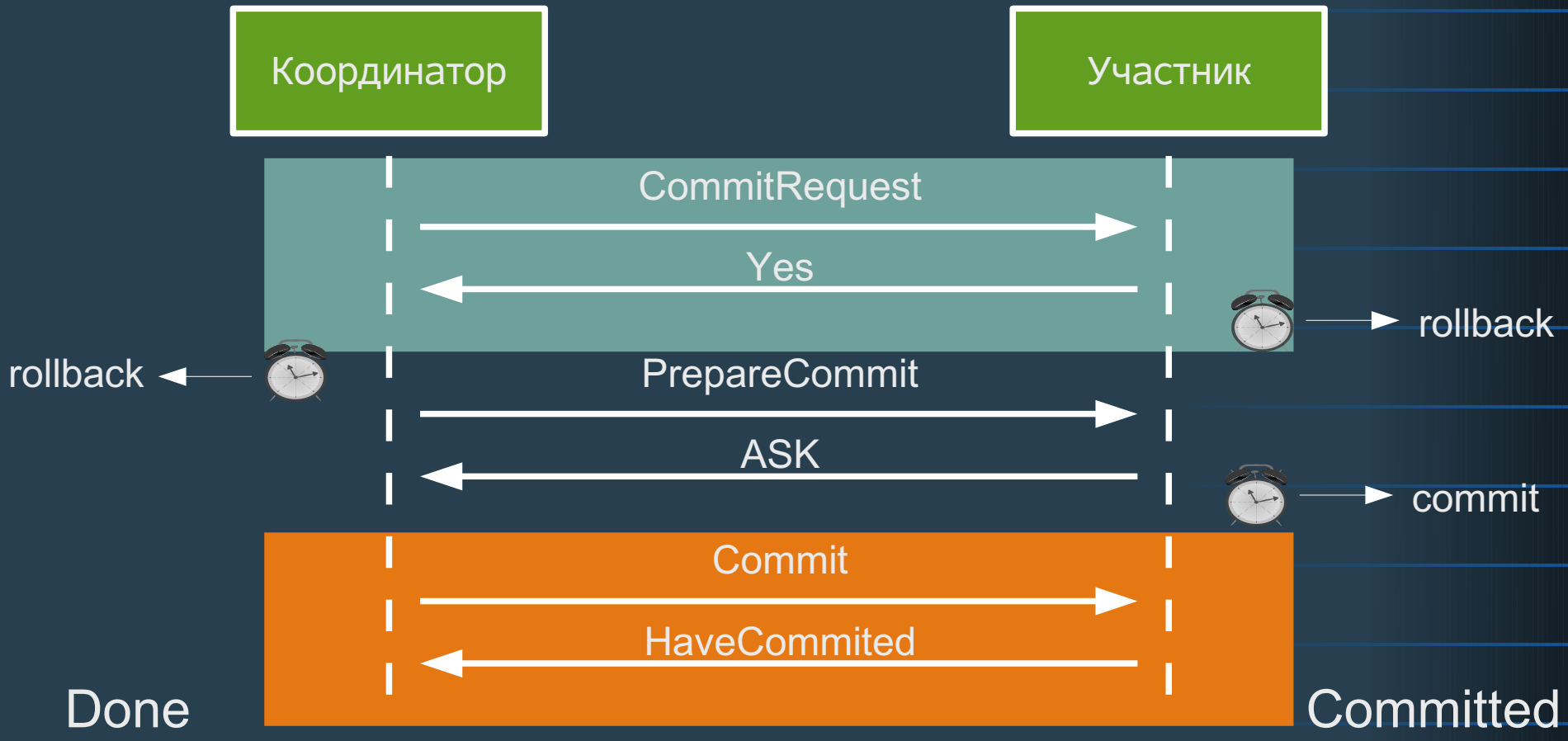
- Недостатки
  - Блокирующий протокол
  - Узел может ждать и блокировать параллельную работу транзакций даже если другие узлы выдали сбой
  - При сбое координатора после PRECOMMIT ожидание может быть очень долгим

Трехфазный протокол согласования (3PC)

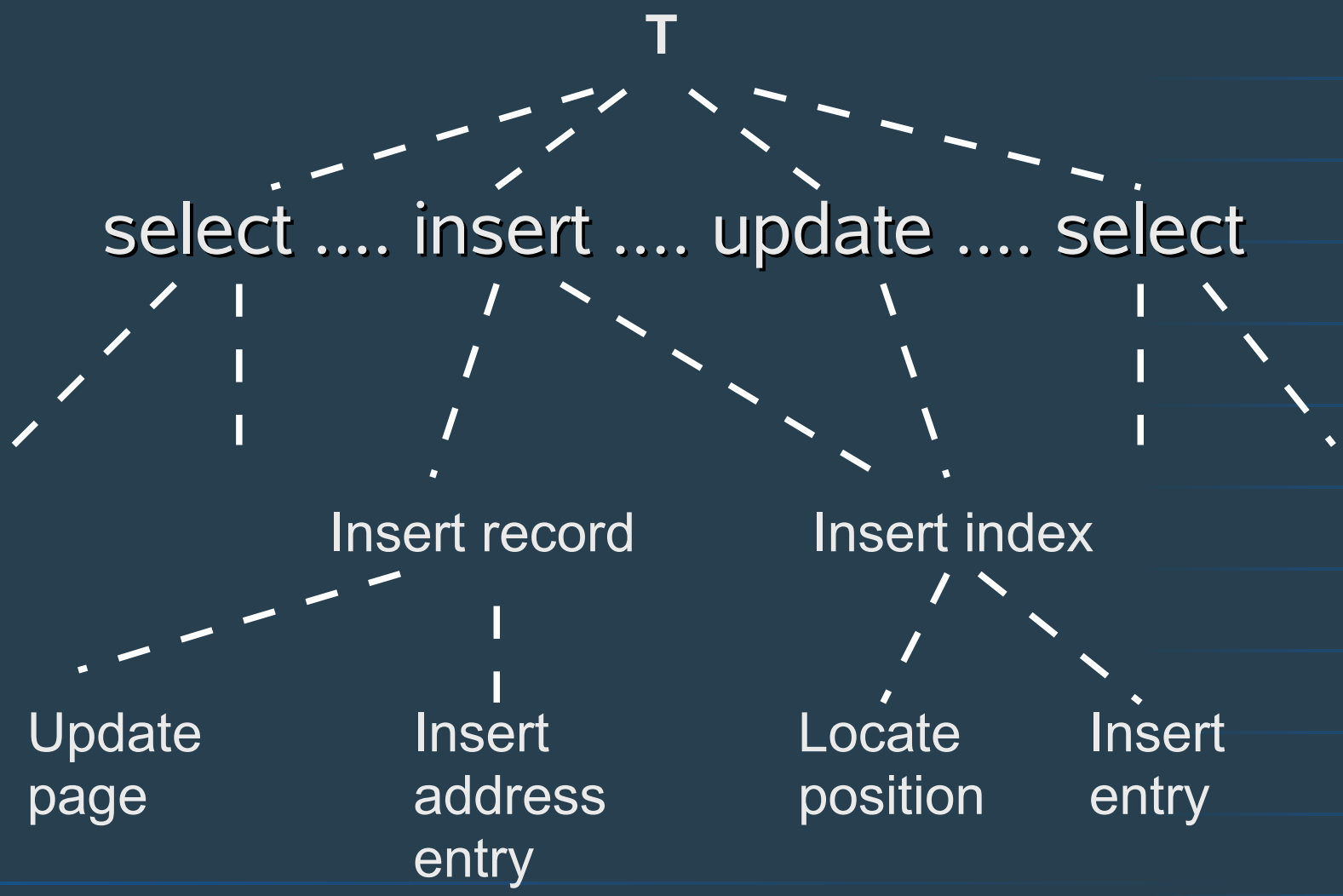
- COMMIT_REQUEST
- PREPARE_COMMIT
- COMMIT

- После начала согласования транзакции ожидание любого узла не превысит гарантированного времени
- Ограничение
  - Только один узел может выдать сбой при согласовании изменений
  - Невозможно восстановление в случае сбоя сети

Трехфазный протокол согласования (3PC)



Использование 2PC



Вопросы?