Изоляция транзакций

Мельников В.М., 2019

Содержание

- Теория сериализации
- Статическое и динамическое управление выполнением транзакций
- Конфликтные ситуации
- Протокол блокировок и уровни изоляции
- Уровни изоляции в SQL
- Предикатные блокировки
- Гранулированные блокировки
- Пессимистичные и оптимистичные методы
- Метод Field-By-Value
- Многоверсионность и метод временных меток
- Проблема взаимоблокировок

Теория сериализации

- Изолированность – создание видимости, что в системе одновременно выполняется только одна транзакция
- Скрытие проблем совместного выполнения
- Упорядочивание транзакций в процессе выполнения

Блокировка транзакций

- Блокировки – механизм разграничения доступа к объектам
- При обращении к объекту транзакция блокирует объект (получает в пользование, владеет)
- Если объект уже заблокирован, то транзакция приостанавливается и попадает в очередь ожидания
- Гранулированность блокировок
- Автоматическое выполнение блокировок в СУБД

Статическое планирование

- I(Ti) – множество читаемых данных
- O(Ti) – множество модифицируемых данных
- Конфликт с Tj, если
  - O(Ti) & (I(Tj) U O(Tj)) □□Ø или
  - O(Tj) & (I(Ti) U O(Ti)) □□Ø
- Недостаток
  - До выполнения нельзя описать множество используемых данных (зависит от результатов работы прикладного алгоритма)

Динамическое планирование

- Транзакции: Ti
- Объекты: Oj
- Во время выполнения:

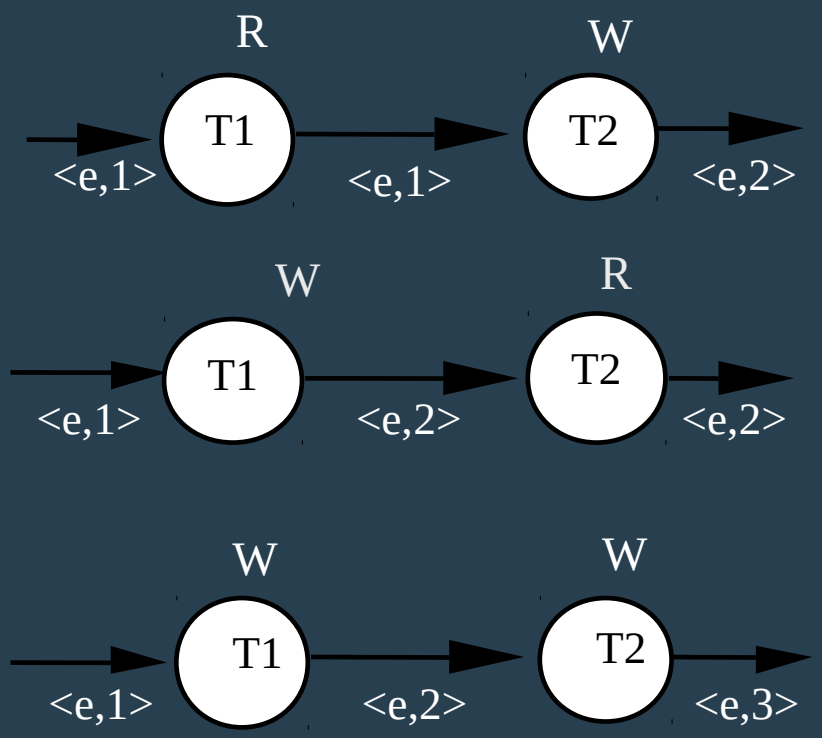
  Ti связывается с Oj (владеет объектом)
- Современный широко используемый подход
- Недостаток: возможны взаимоблокировки

Зависимости между транзакциями

- 2 операции: READ + WRITE
- 4 варианта совместного выполнения 2-х разных транзакций
- T1 READ A; T2 READ A – нет проблем
- Версии объекта <0,1>,<O,2>...<O,n>
- Если транзакция выполняет READ <O,i>, то она зависит от версии объекта
- Если транзакция выполняет WRITE <O,i>, то версия объекта зависит от транзакции

История выполнения транзакции

Потоки данных



Зависимости при изменении данных:

- READ - WRITE
  - T1 READ e
  - T2 WRITE e
- WRITE - READ
  - T1 WRITE e
  - T2 READ e
- WRITE - WRITE
  - T1 WRITE e
  - T2 WRITE e

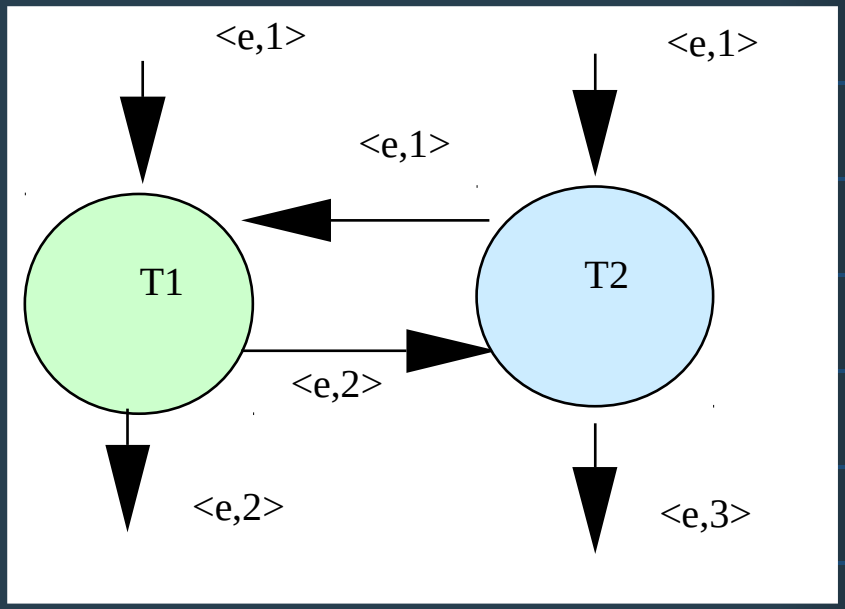
Конфликтные ситуации между транзакциями

- Потерянные изменения (lost update)

  T2 READ <c,1>

  T1 WRITE <c,2>

  T2 WRITE <c,3>



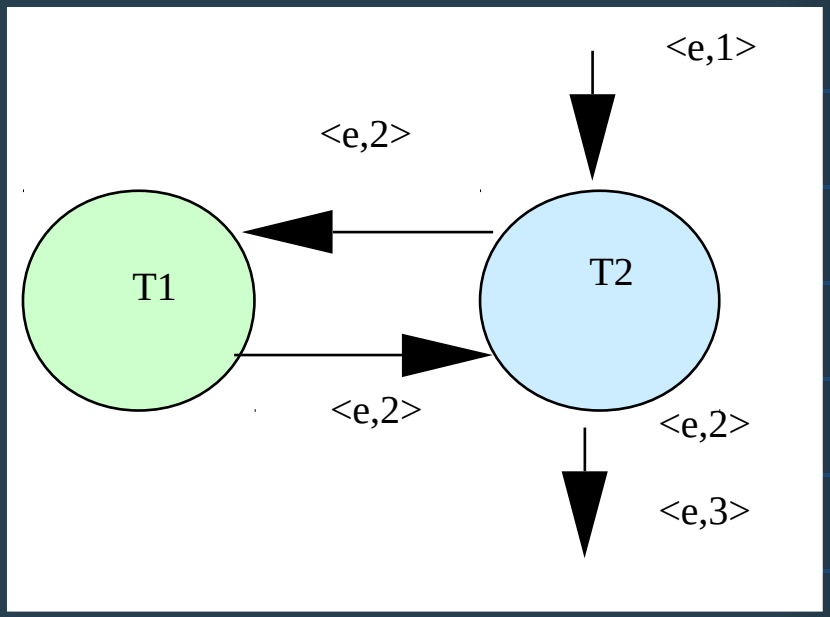
Конфликтные ситуации между транзакциями

- Грязное чтение (dirty read)

  T2 WRITE <c,2>

  T1 READ <c,2>

  T2 WRITE <c,3>



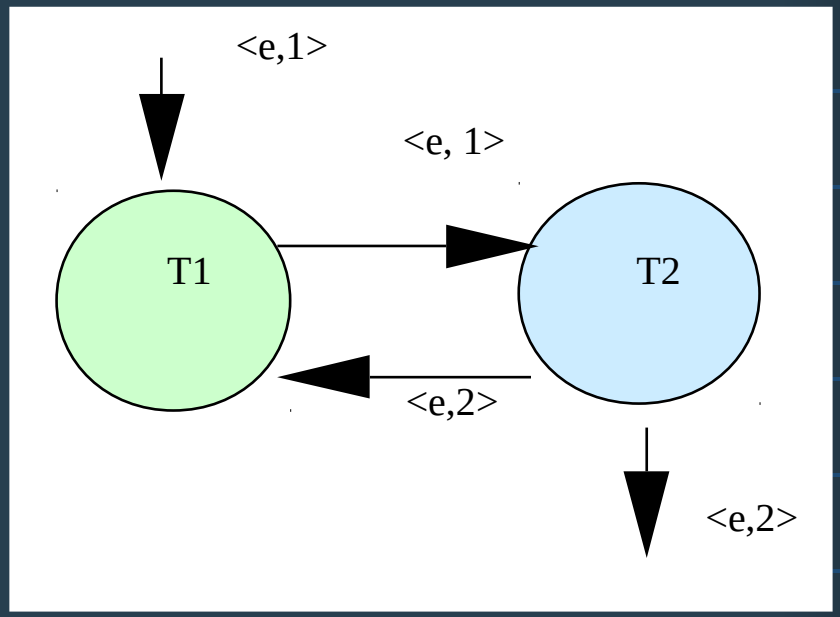
Конфликтные ситуации между транзакциями

- Неповторяемое чтение (unrepeatable read)

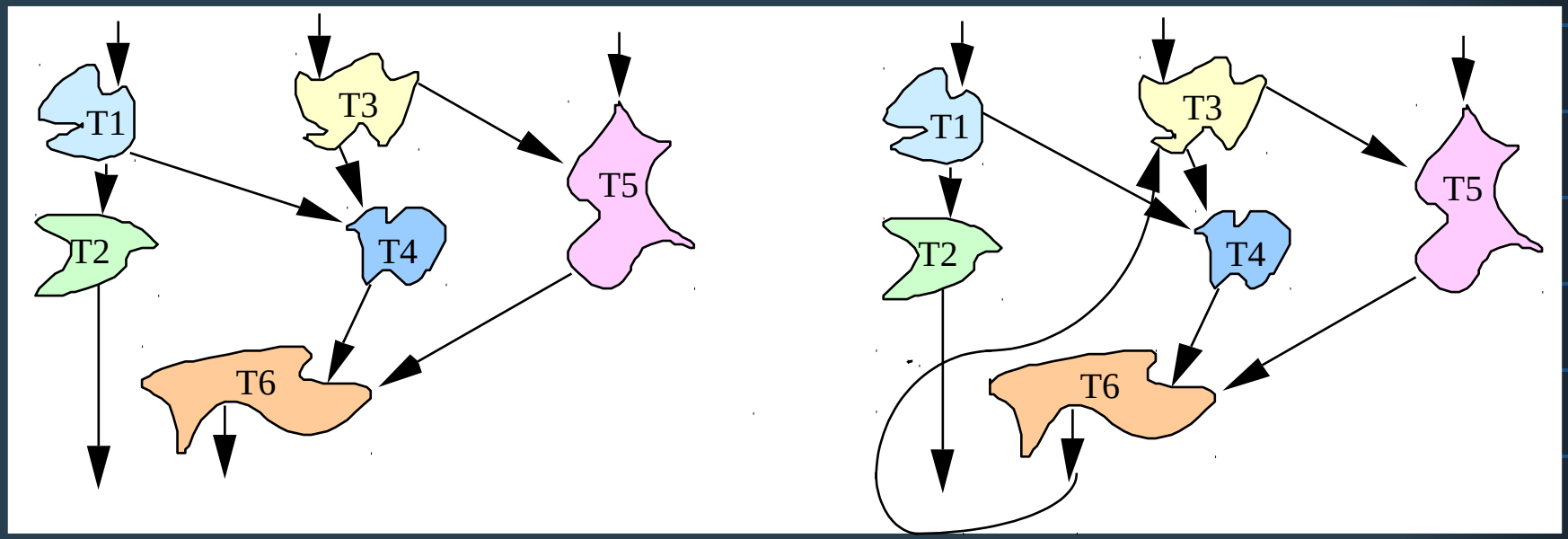
  T1 READ <c,1>

  T2 WRITE <c,2>

  T1 READ <c,2>



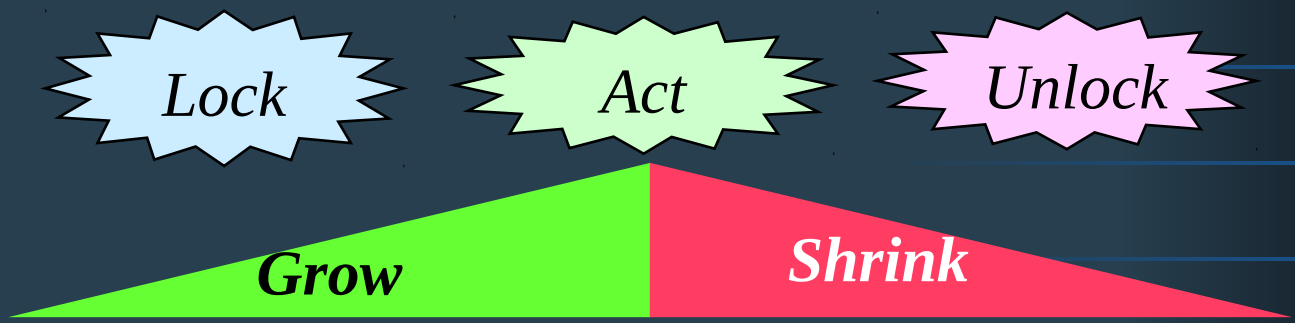
Теория сериализации



- Наличие циклов в потоках данных между транзакциями приводит к конфликтным ситуациям
- При отсутствии циклов возможно упорядоченное выполнение транзакций без конфликтов

Протокол блокировок

- 3 оператора для работы с блокировками
  - LOCK READ (SHARED LOCK)
  - LOCK WRITE(EXCLUSIVE LOCK)
  - UNLOCK
- Правильно сформированная транзакция
- Двухфазная транзакция



Совместимость блокировок

		Текущий режим		
		нет	SLOCK	XLOCK
Требуемый режим	SLOCK	+	+	-
	XLOCK	+	-	-

История выполнения транзакций

Допустимая последовательная

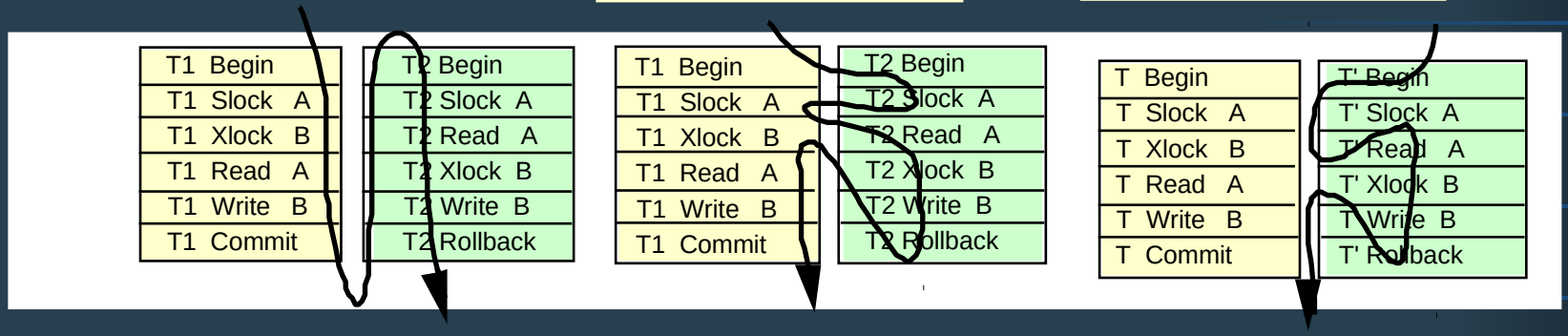
T1	SLOCK	A
T1	XLOCK	B
T1	READ	A
T1	WRITE	B
T1	UNLOCK	A
T1	UNLOCK	B
T2	SLOCK	A
T2	READ	A
T2	XLOCK	B
T2	WRITE	B
T2	WRITE	B
T2	UNLOCK	A
T2	UNLOCK	B

Допустимая непоследовательная

T2	SLOCK	A
T1	SLOCK	A
T2	READ	A
T2	XLOCK	B
T2	WRITE	B
T2	WRITE	B
T2	UNLOCK	A
T2	UNLOCK	B
T1	XLOCK	B
T1	READ	A
T1	WRITE	B
T1	UNLOCK	A
T1	UNLOCK	B

Недопустимая непоследовательная

T1	SLOCK	A
T1	XLOCK	B
T2	SLOCK	A
T2	READ	A
T2	***XLOCK***	***B***
T2	WRITE	B
T2	WRITE	B
T2	UNLOCK	A
T2	UNLOCK	B
T1	READ	A
T1	WRITE	B
T1	UNLOCK	A
T1	UNLOCK	B



Теория сериализации

- Если транзакция является правильно сформированной и двухфазной, то все истории выполнения транзакции являются допустимыми и последовательными (допускает изоляцию)
- Если транзакций не является двухфазной или правильно сформированной, то существует допустимая, но не последовательная история выполнения транзакции

Уровни изоляции

0□: Транзакции устанавливают short xlocks (правильно сформированная относительно write, не двухфазная)

1□: Транзакции устанавливают long xlocks (правильно сформированная относительно write, двухфазная)

2□: Транзакция устанавливает short slocks (правильно сформированная, недвухфазная)

3□: Транзакция устанавливает long slocks (правильно сформированная, двухфазная)

Сравнение уровней изоляции

	0	1	2	3
Название	Хаос	Browse, Read uncomitted	Cursor stability, Read committed	Serializible, Repeatable read
Защита	Позволяют другим работать	+ Нет потерянных изменений	+ Нет грязного чтения	+ Нет неповторяемого чтения
Согласование изменений	Изменения видны сразу	Изменения видны после завершения транзакции		
Протокол блокировок	Short XLOCK	Long XLOCK	Long XLOCK + Short SLOCK	Long XLOCK + Long SLOCK
Структура транзакции	Правильно сф. для WRITE	Правильно сф. для WRITE + 2Ф	Правильно сформированная	Правильно сф. + 2Ф
Совместный доступ	Отличный			Плохой

Уровни изоляции в SQL

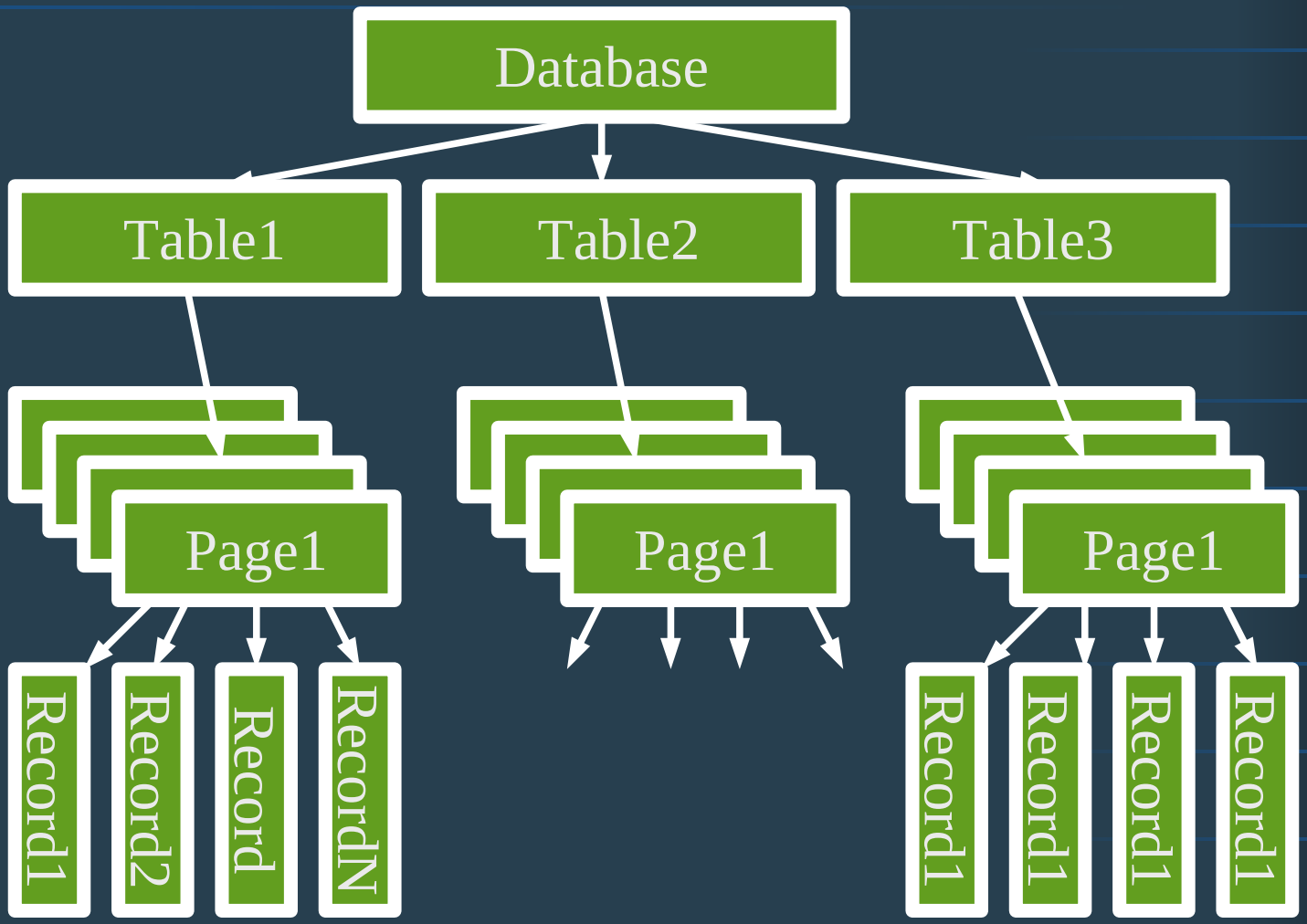
- SET ISOLATION LEVEL TO
  [READ UNCOMMITED |
  READ COMMITED |
  REPEATABLE READ |
  SERIALIZABLE]
- OPEN CURSOR C – “cursor stability”

Проблема фантомных записей

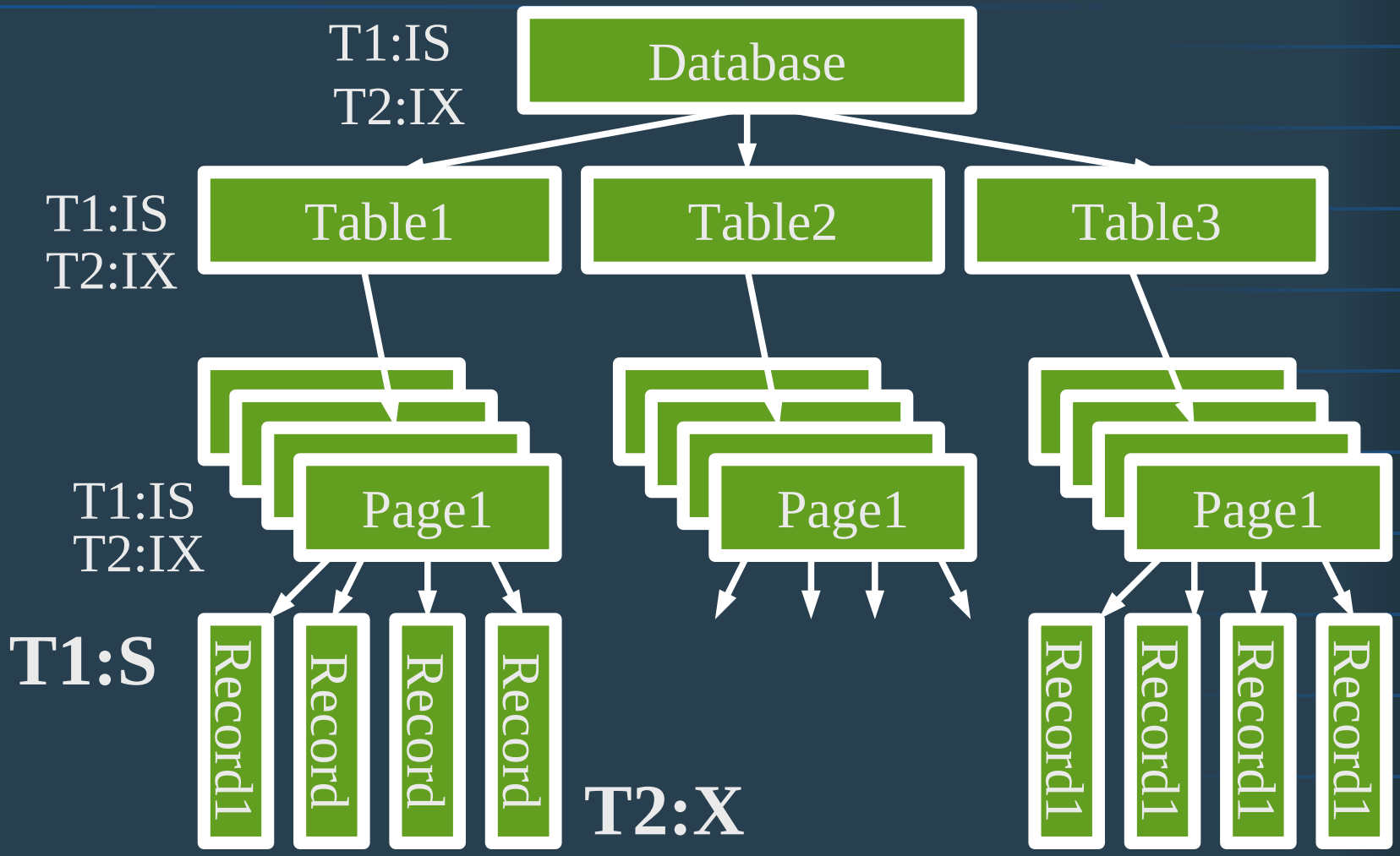
- T1: Select phone from ... where NAME=’Иванов’ (5 записей)
- T2: Insert into ... values(77, ‘Иванов’,....)
- T1: Select phone from ... where NAME=’Иванов’ (6 записей !)

- UNREPEATABLE READ!

Гранулированные блокировки



Гранулированные блокировки



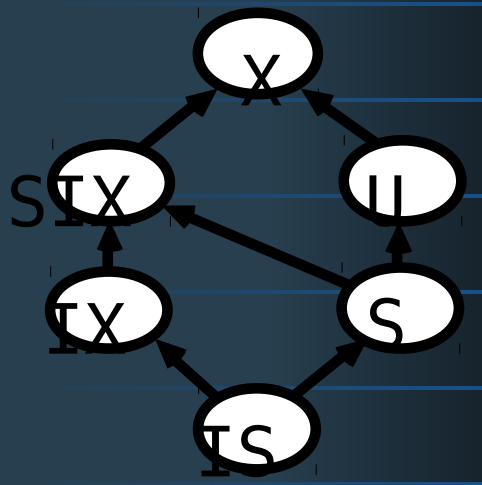
Блокировка Intent

		Текущий режим				
		IS	IX	SIX	S	X
Требуемый режим	IS	+	+	+	-	-
	IX	+	+	-	-	-
	SIX	+	-	-	-	-
	S	-	-	-	+	-
	X	+	-	-	-	-

Порядок блокирования

- От корня дерева к узлам
- Если S блокировка, то вышестоящая как минимум IS (IX, S, SIX, U, X)
- Если X блокировка, то вышестоящая как минимум IX (SIX, U, X)
- Блокировка обновления Update
- Порог эскалации

  блокировок = N



Гранулированные блокировки

- Достоинства:
  - Решение проблемы фантомных записей
  - Меньше ресурсов при массовых обновлениях

Предикатные блокировки

- T1: Select PHONE from ... where NAME=’Иванов’

  <T1, SLOCK, STUDENTS.NAME=’Иванов’>
- T2: Insert into ... values(77, ‘Иванов’,....)

  <T2, XLOCK, STUDENTS.NAME=’Иванов’ and STUDENTS.ID=77>
- Проверка совместимости предикатов
- Транзакция приостанавливается, если возможно существование записей, делающих оба предиката истинными

Предикатные блокировки

- Достоинства
  - Решение проблемы фантомных записей
- Недостатки
  - Сложное выделение предикатов
  - Сложность задачи определения совместимости предикатов (NP-полная задача)
  - Пессимистичное поведение (записей может и не быть в БД, а транзакция будет ждать). Пример, “в БД нет синеглазых шатенов”

Точные блокировки

- Проверка конфликтов на момент извлечения или модификации данных

- Select ... from ... where eye=’blue’
- <T1, READ, eye=blue>

- Insert into ... values(5773,’Иванов’,’grey’)
- <T2, WRITE, ‘id=5773, name=Иванов, eye=grey>

Точные блокировки

- Достоинства
  - Решается проблема фантомных записей
  - Простая проверка на конфликты
- Недостатки
  - Высокая вероятность взамного блокирования

Пессимистичные и оптимистичные методы блокирования

- Пессимистичные – высокая вероятность обращения к одним и тем же данным разных транзакция
- Оптимистичные – низкая вероятность работы с одними и теми же данными
- Ключевые отличия:
  - Завершаются всегда/могут не завершиться
  - Блокируют все записи на все время работы транзакции/блокируют только для проверки
- Hot Spot записи в БД

Снимки значений

- Снимки значений:
  - Сохранить старое значение каждого объекта
  - Откладываем обновления до фазы 2
  - Фаза 1 COMMIT: если значение поменялось, то откат, иначе блокируем XLOCK
  - Фаза 2 COMMIT: выполняем все изменения, снимаем все блокировки

Отметки времени

- Отметки времени:
  - Сохранить время обновления каждого объекта
  - Откладываем обновления до фазы 2
  - Фаза 1 COMMIT: если время поменялось, то откат, иначе блокируем XLOCK
  - Фаза 2 COMMIT: выполняем все изменения, снимаем все блокировки

Многоверсионность

- Запись R хранит историю изменений

R: [T0..T1): V0, [T1…T2) V1, [T2….) V3.

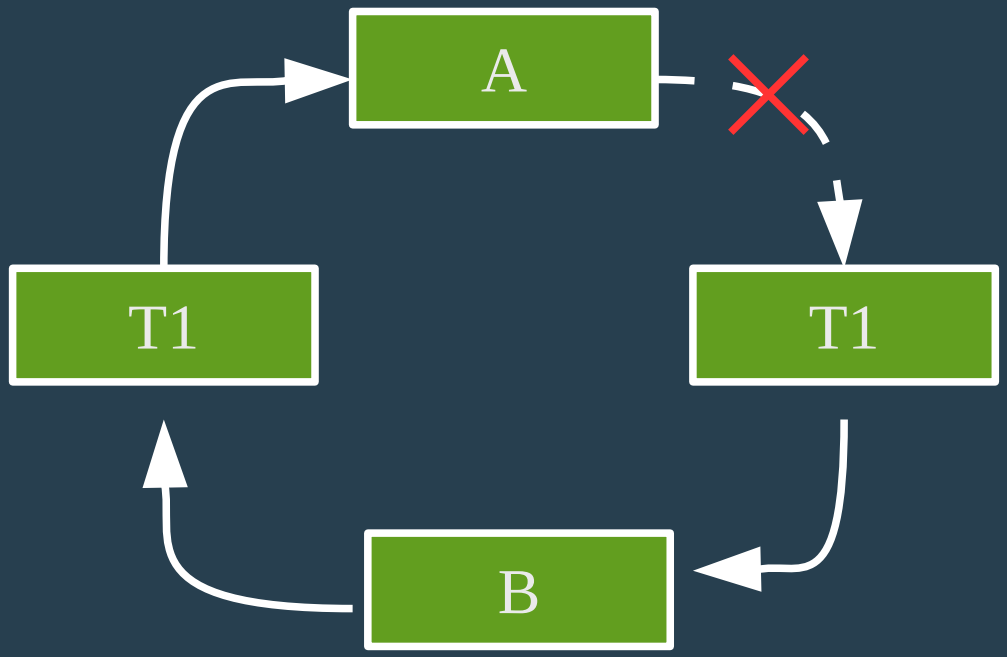
- Ti – время выполнения COMMIT транзакцией
- Последняя версия в история – актуальная версия
- При старте транзакции запоминаем время ti
- При чтении транзакция читает значения записей на момент ti
- При изменении записи создается частная версия
- При выполнении COMMIT выполняется согласование

Взаимное блокирование

- Определение момента блокирования
- Разрешение конфликтной ситуации

- T1 SLOCK A
- T1 READ A
- T2 SLOCK B
- T2 READ B
- T1 XLOCK B
- T2 XLOCK A

Определение взаимоблокировок



- Определение циклов в графе зависимостей

Разрешение конфликта

- Откатывается транзакция, которая привела к образованию цикла
- Откатывается транзакция, на повторное выполнение которой будет затрачено меньшее количество ресурсов

Вопросы?