

Изоляция транзакций

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
- 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

- $I(T_i)$ – множество читаемых данных
- $O(T_i)$ – множество модифицируемых данных
- Конфликт с T_j , если
 - $O(T_i) \cap (I(T_j) \cup O(T_j)) \neq \emptyset$ или
 - $O(T_j) \cap (I(T_i) \cup O(T_i)) \neq \emptyset$
- Недостаток
 - До выполнения нельзя описать множество используемых данных (зависит от результатов работы прикладного алгоритма)

Динамическое планирование

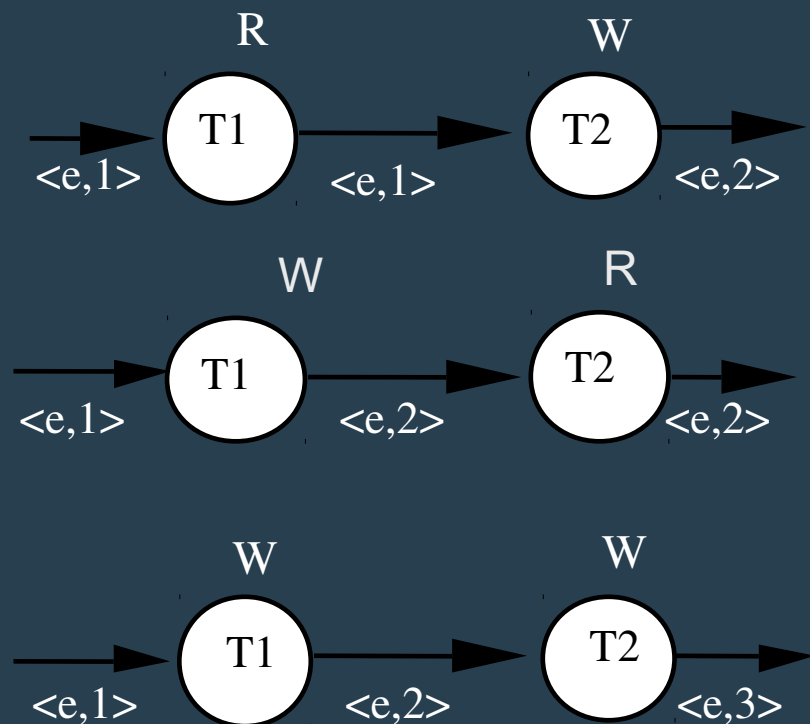
- Транзакции: T_i
- Объекты: O_j
- Во время выполнения:
 T_i связывается с O_j (владеет объектом)
- Современный широко используемый подход
- Недостаток: возможны взаимоблокировки

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 $\langle O, 1 \rangle, \langle O, 2 \rangle \dots \langle O, n \rangle$
- Если транзакция выполняет READ $\langle O, i \rangle$, то она зависит от версии объекта
- Если транзакция выполняет WRITE $\langle O, i \rangle$, 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

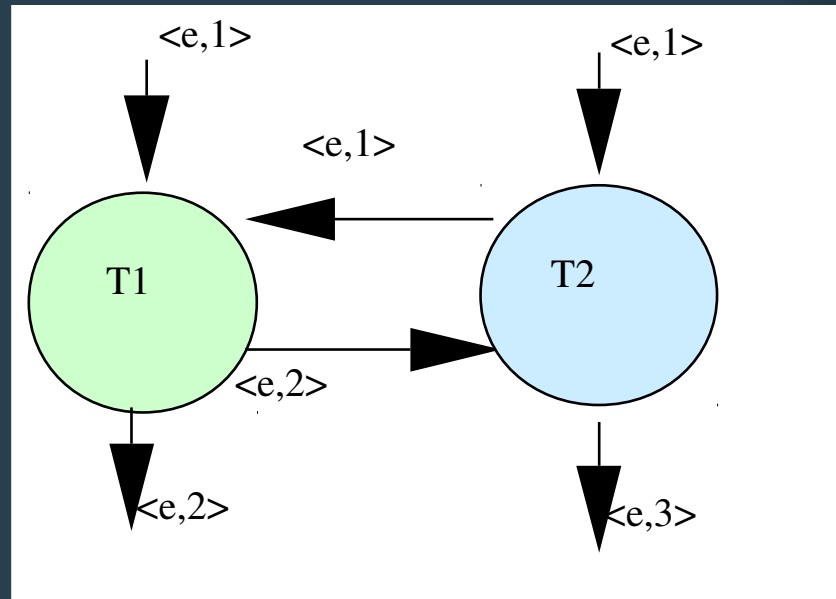
Конфликтные ситуации между транзакциями

- Потерянные изменения (lost update)

T2 READ $\langle c, 1 \rangle$

T1 WRITE $\langle c, 2 \rangle$

T2 WRITE $\langle c, 3 \rangle$



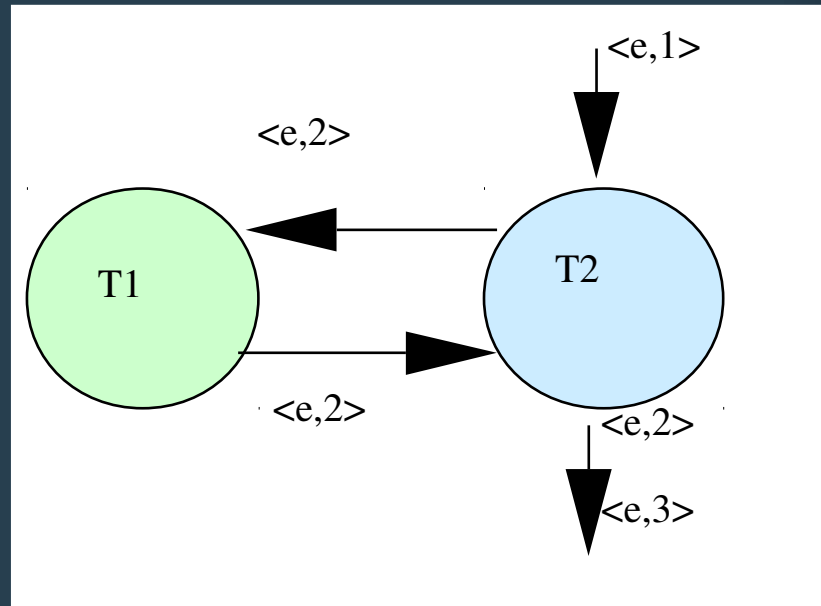
Конфликтные ситуации между транзакциями

- Грязное чтение (dirty read)

T2 WRITE $\langle c, 2 \rangle$

T1 READ $\langle c, 2 \rangle$

T2 WRITE $\langle c, 3 \rangle$



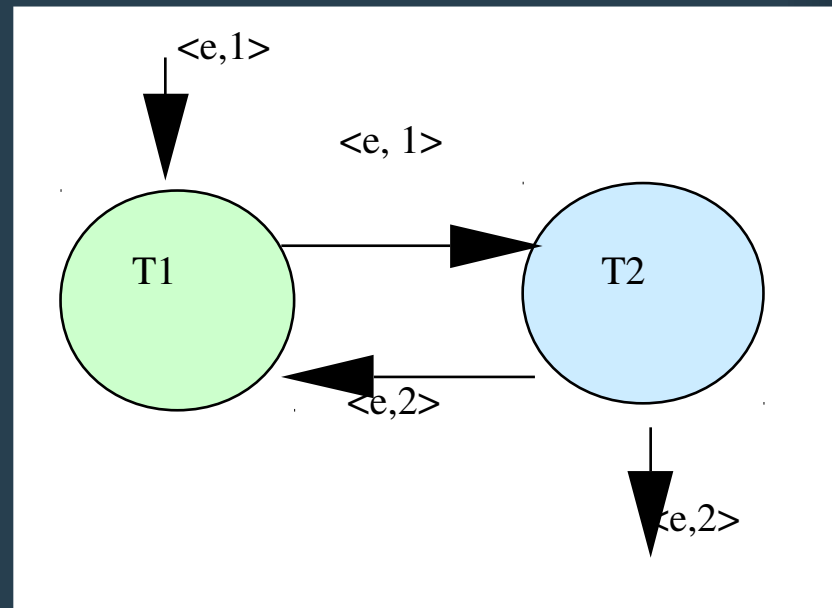
Конфликтные ситуации между транзакциями

- Неповторяемое чтение (unrepeatable read)

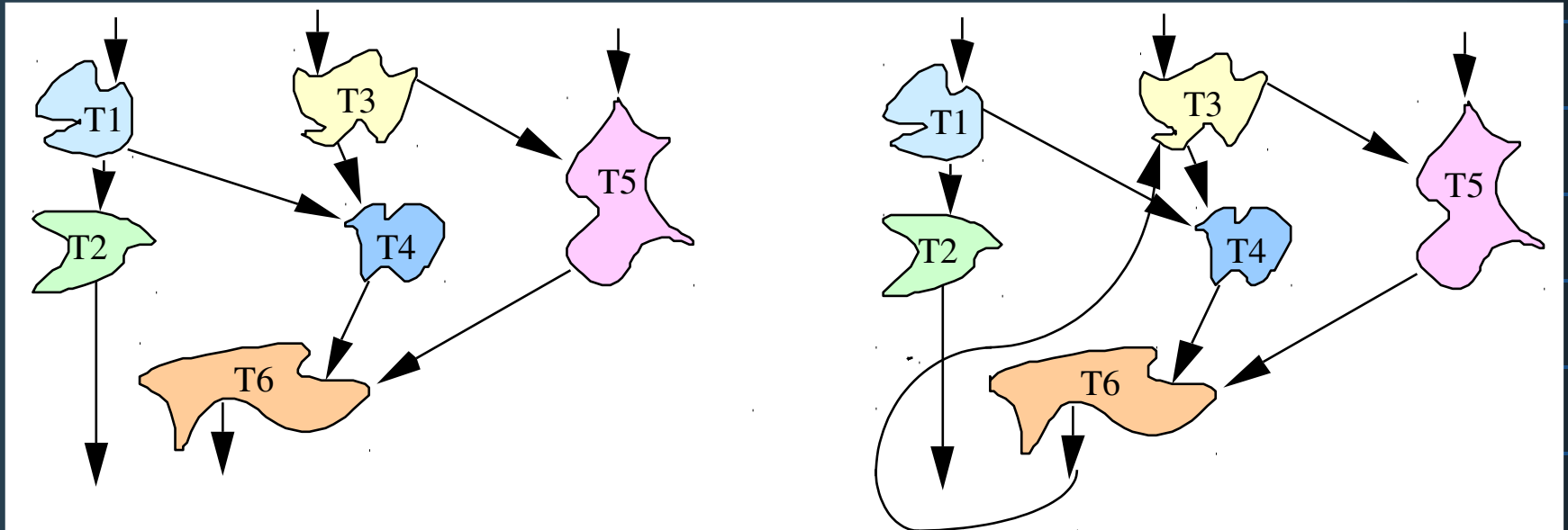
T1 READ $\langle c, 1 \rangle$

T2 WRITE $\langle c, 2 \rangle$

T1 READ $\langle c, 2 \rangle$



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

T1 Begin	T2 Begin
T1 Slock A	T2 Slock A
T1 Xlock B	T2 Read A
T1 Read A	T2 Xlock B
T1 Write B	T2 Write B
T1 Commit	T2 Rollback

T1 Begin	T2 Begin
T1 Slock A	T2 Slock A
T1 Xlock B	T2 Read A
T1 Read A	T2 Xlock B
T1 Write B	T2 Write B
T1 Commit	T2 Rollback

T Begin	T' Begin
T Slock A	T' Slock A
T Xlock B	T' Read A
T Read A	T' Xlock B
T Write B	T' Write B
T Commit	T' Rollback

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 uncommitted	Cursor stability, Read committed	Serializable, Repeatable read
Защита	Позволяют другим работать	нет потерянных изменений	нет грязного чтения	Нет неповторяемо го чтения
Согласование изменений	Изменения видны сразу	Изменения видны после завершения транзакции		
Протокол блокировок	Short xlock	Long xlock	Short slock	Long slock
Структура транзакции	Правильно сф. для WRITE	Правильно сф. для WRITE + 2Ф	Правильно сформирован ная	Правильно сф. + 2Ф
Совместный доступ	Отличный			Плохой

Уровни изоляции в SQL

- set isolation level to
[READ UNCOMMITTED |
READ COMMITTED |
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!**

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”

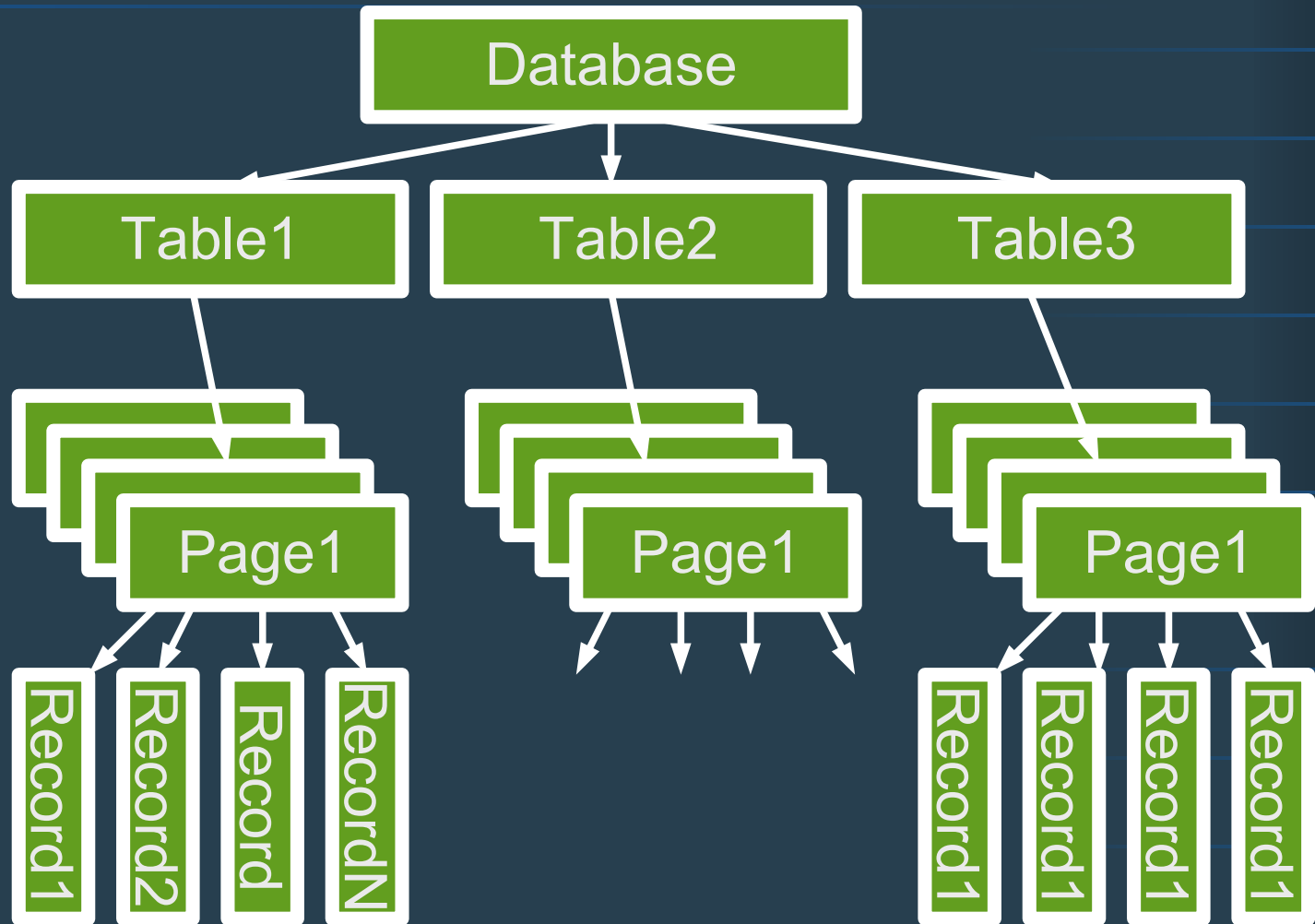
Точные блокировки

- 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>`

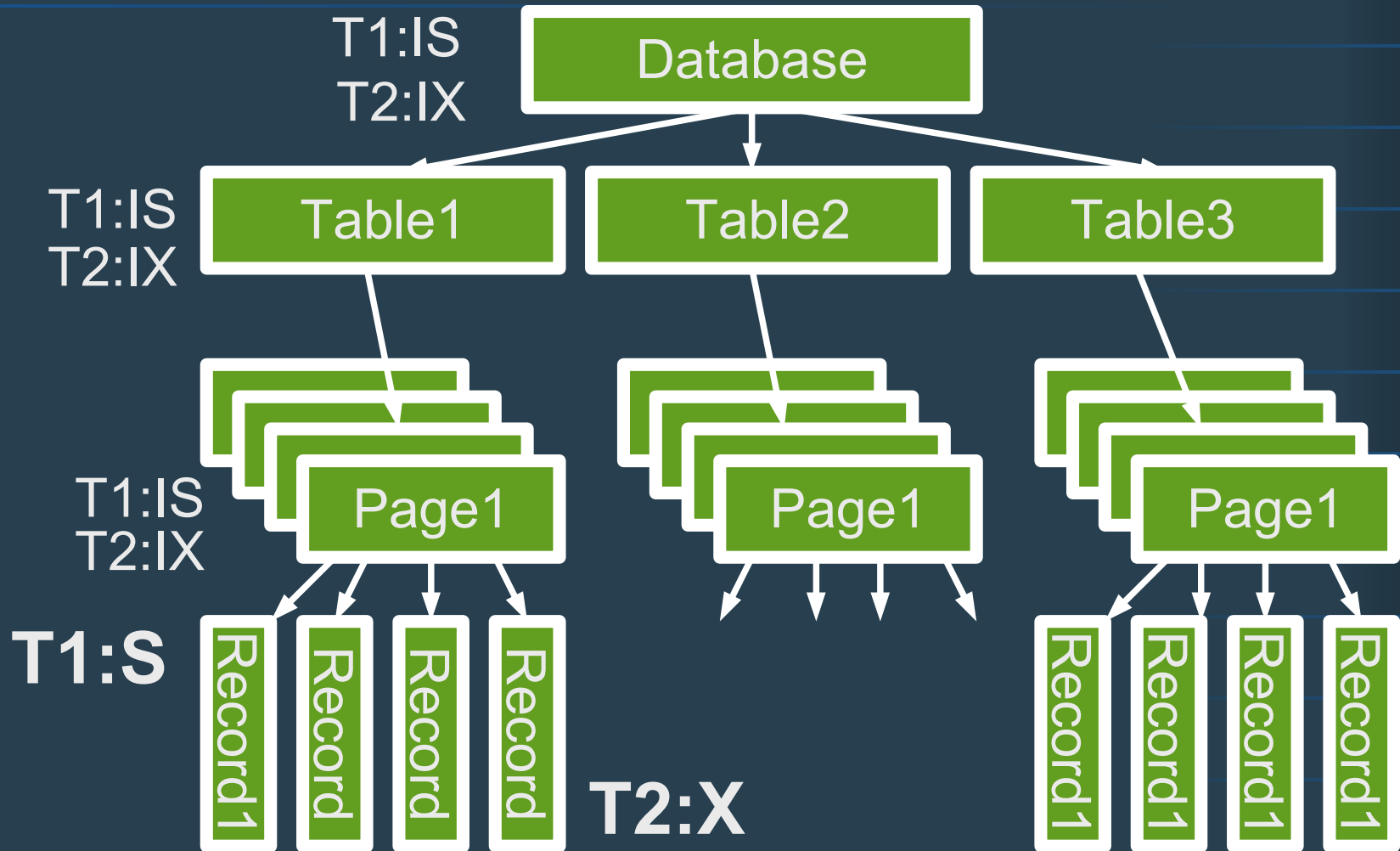
Точные блокировки

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

Гранулированные блокировки



Гранулированные блокировки

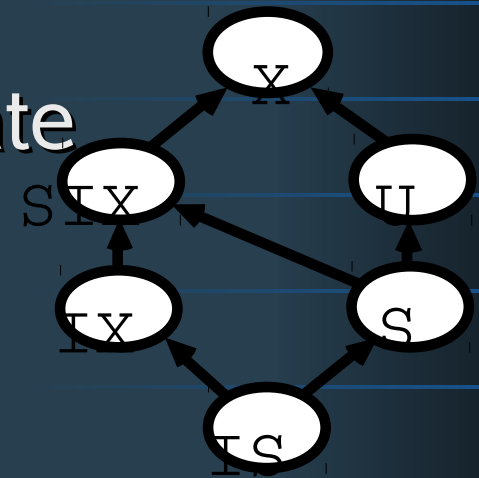


Блокировка Intent

		Текущий режим				
		IS	IX	SIX	S	X
Требуем ый 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

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

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

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

Многоверсионность

- Запись R хранит историю изменений

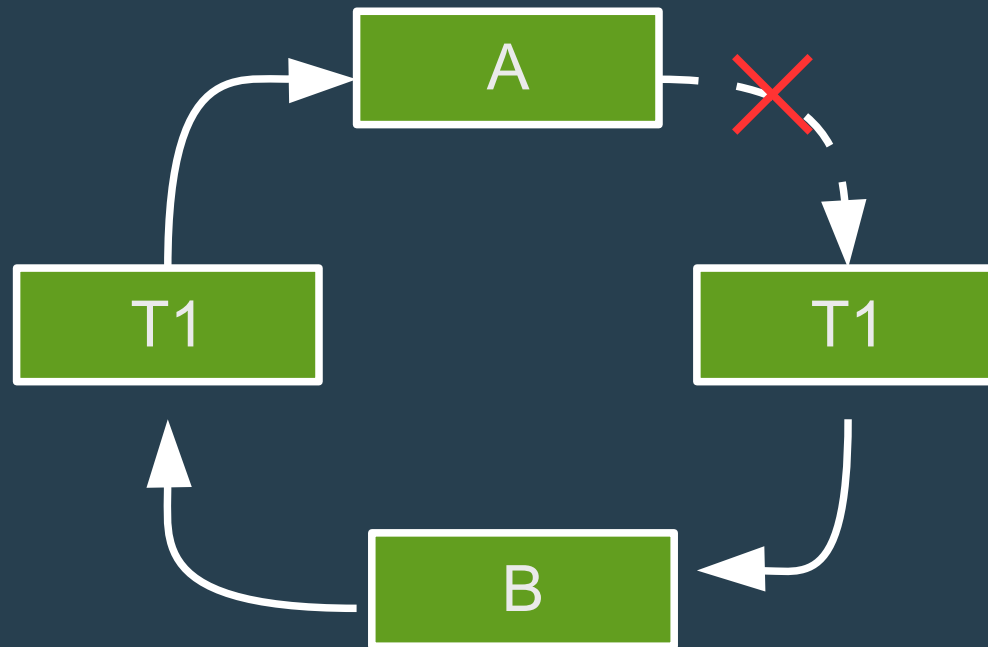
R: $[T_0..T_1): V_0$, $[T_1...T_2) V_1$, $[T_2....) V_3$.

- T_i – время выполнения COMMIT транзакцией
- Последняя версия в история – актуальная версия
- При старте транзакции запоминаем время t_i
- При чтении транзакция читает значения записей на момент t_i
- 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



www.pictofigo.com

Вопросы?