Курс «Базы данных»

Тема. Транзакции. Часть 2

Барабанщиков
Игорь Витальевич

План лекции

1. Проблемы выполнения параллельных транзакций и способы их решения.
2. Блокировки и взаимоблокировки.

Многопользовательский режим работы СУБД

- Если с базой данных одновременно работают несколько пользователей, **СУБД должна гарантировать, что пользователи не будут мешать друг другу**.
- В идеальном случае каждый пользователь должен работать с БД так, как **если бы он имел к ней монопольный доступ, и не должен беспокоиться о действиях других пользователей**.
- Средства обработки транзакций в SQL позволяют реляционным СУБД изолировать пользователей друг от друга именно таким образом.
- **Если транзакции не будут корректно обработаны, то могут возникнуть проблемы**.
- Такие проблемы можно разбить на четыре основные категории.

Проблемы параллелизма

Параллелизм (concurrency) – это *одновременная* обработка транзакций.

Проблемы, которые возникают при этом:

- **Потерянное обновление** (lost update)
- **Неповторяющееся чтение** (non-repeatable read)
- **Преждевременное чтение** (dirty read)
- **Фантомная вставка** (phantom insert)

Грязное чтение

«Грязное» чтение (dirty read) – чтение данных, добавленных или изменённых **незафиксированной** транзакцией, **которая впоследствии откатится**.

balance = 500

Транзакция 1	Транзакция 2

SELECT balance FROM users
  WHERE id=1;
 	
UPDATE users
  SET balance = balance – 300
  WHERE id = 1;
 	
ROLLBACK;
	SELECT balance FROM users
	  WHERE id = 1;

Неповторяющееся чтение (non-repeatable read)

Неповторяющееся чтение (non-repeatable read) – при повторном чтении в рамках одной транзакции, **ранее прочитанные данные оказываются изменёнными**.

Транзакция 1	Транзакция 2

`SELECT balance FROM users WHERE id=1;`
 	
`UPDATE users`
`SET balance = balance + 25`
`WHERE id = 1;`
 	
`COMMIT;`
	`SELECT balance FROM users`
	`WHERE id = 1`
	
	`SELECT balance FROM users`
	`WHERE id = 1`

Потерянное обновление

Потерянное обновление (lost update) – при одновременном изменении одного блока данных разными транзакциями, **одно из изменений теряется.**

```
balance = 500
```

Транзакция 1	Транзакция 2

`UPDATE users`
`SET balance = balance – 300`
`WHERE id = 1`
	`UPDATE users`
	`SET balance = balance + 300`
	`WHERE id = 1`

Фантомная вставка (phantom insert)

Фантомная вставка (phantom insert) – одна транзакция в ходе своего выполнения несколько раз выбирает множество строк по одним и тем же критериям.

Другая транзакция в интервалах между этими выборками добавляет и удаляет строки, попадающие в критерии выборки первой транзакции, и успешно заканчивается.

В результате одни и те же выборки первой транзакции дают разные множества строк.

Транзакция 1	Транзакция 2

`SELECT SUM(balance) FROM users`
`  WHERE age BETWEEN 18 AND 25;`
 	
`SELECT SUM(balance) FROM users`
`  WHERE age BETWEEN 18 AND 25;`
	`INSERT INTO users (age, balance)`
	`  VALUES(20, 100);`
	`COMMIT;`

Параллельное выполнение транзакций

- Когда 2 транзакции, A и B, выполняются параллельно, **СУБД гарантирует, что результаты их выполнения будут точно такими же**, как и в случае, если:

  - либо вначале выполняется транзакция A, а затем

    транзакция B;

  - либо вначале выполняется транзакция B, а затем

    транзакция A.

- Данная концепция называется **сериализацией** транзакций.
- **На практике это означает, что каждый пользователь может работать с БД так, как**

Сериализация транзакций

Процедура согласованного выполнения параллельных транзакций:

- В ходе выполнения транзакции пользователь видит только **согласованные данные**.
- **Пользователь не должен видеть несогласованных промежуточных данных**.
- Когда в БД две транзакции выполняются параллельно, то СУБД гарантированно поддерживает **принцип независимого**

Уровни изоляции транзакций

- Стандарт ANSI/ISO для SQL устанавливает различные **4 уровня изоляции** транзакций.
- Уровень изоляции **определяет может ли читающая транзакция считывать («видеть») результаты работы других одновременно выполняемых завершенных и\или незавершенных пишущих транзакций.**
- По мере увеличения уровня изоляции **увеличивается число проблем, от которых СУБД защищает пользователя.**

Уровни изоляции транзакций

- ***READ UNCOMMITTED*** - *нефиксированное чтение*. Здесь возможно получение разных результатов для одинаковых запросов без учета *фиксации транзакции*.
- ***READ COMMITTED*** - *фиксированное чтение*. Этот уровень позволяет получать разные результаты для одинаковых запросов, но только после *фиксации транзакции*, повлекшей изменение данных;
- ***REPEATABLE READ*** - *повторяющееся чтение*. На этом уровне разрешено выполнение операторов INSERT, приводящих к конфликтной ситуации "*фантомная вставка*".
- ***SERIALIZABLE*** - *последовательное выполнение* (используется по умолчанию). Этот уровень гарантирует предотвращение всех описанных выше конфликтных ситуаций, но при нем наблюдается самая

Уровни изоляции транзакций

Уровень изоляции	Потерянные изменения	Неаккуратное считывание	Неповторяемое считывание	Фиктивные элементы
READ UNCOMMITED	Нет	Да	Да	Да
READ COMMITED	Нет	Нет	Да	Да
REPEATABLE READ	Нет	Нет	Нет	Да
SERIALIZABLE	Нет	Нет	Нет	Нет

Блокировка

- Наиболее распространенный механизм разграничения пишущих транзакций – **использование блокировок**.

- **Блокировка** – **это временное ограничение доступа к данным**, участвующим в транзакции, со стороны других

Типы блокировок

Различают следующие типы блокировок:

- **По степени доступности данных**:
  - разделяемые
  - исключающие
- **По множеству блокируемых данных**:
  - строчные
  - страничные
  - табличные
- **По способу установки**:
  - автоматические
  - явные

Режимы блокировки

- **S (Shared) Совместный режим блокировки** — нежесткая или *разделяемая блокировка* :
  - Разделяемый захват объекта
  - Для выполнения операции чтения объекта (SELECT)
  - Объекты не изменяются при выполнении транзакции и доступны другим транзакциям только для чтения

- **X (eXclusive) Монопольный режим блокировки** — жесткая или эксклюзивная блокировка :
  - Монопольный захват объекта.
  - Для выполнения операций занесения, удаления и модификации (INSERT, DELETE, UPDATE).
  - Объекты недоступны для других транзакций до

Матрица совместимости блокировок

Транзакция A наложила блокировку	**Транзакция B пытается наложить блокировку:**	
	S-блокировку	X-блокировку
S-блокировку	Да	НЕТ (Конфликт R-W)
X-блокировку	НЕТ (Конфликт W-R)	НЕТ (Конфликт W-W)

Блокировки в Oracle

- При выполнении транзакции СУБД Oracle **автоматически блокирует необходимые данные.**
- Блокировка устанавливается на самом **минимальном уровне – на уровне строки**.
- Это позволяет одновременно **выполнять много параллельных**

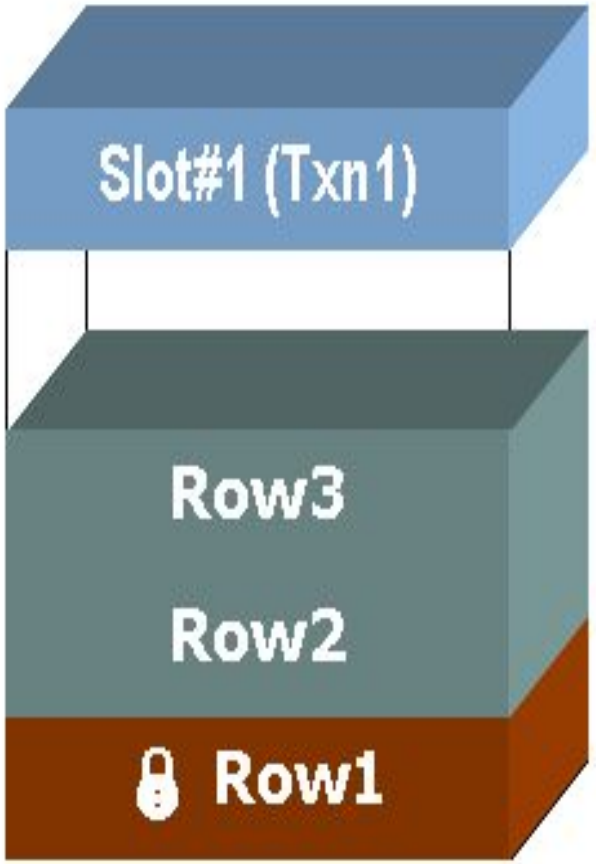
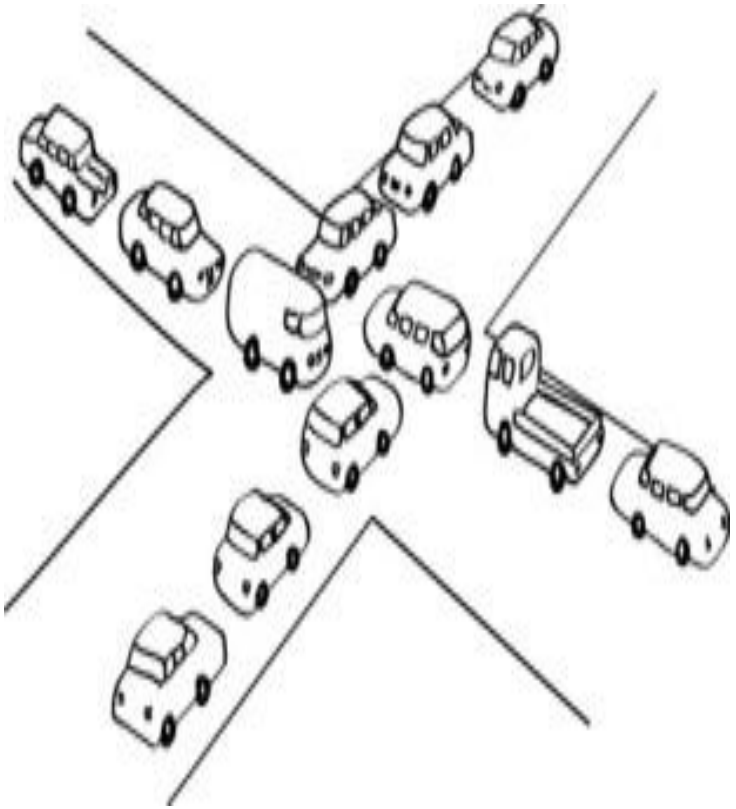


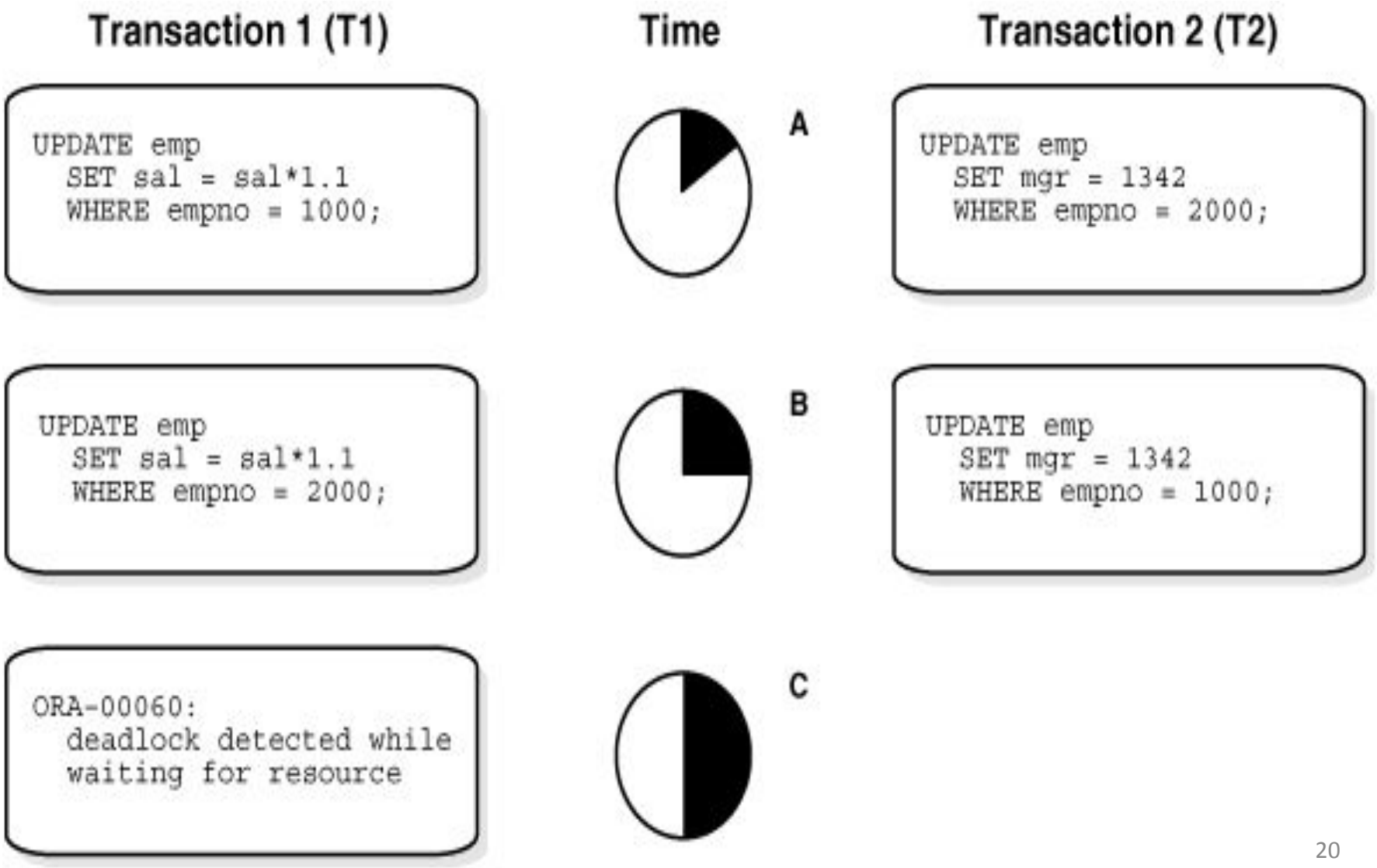
Рисунок 3

Взаимоблокировки

Взаимная блокировка (*deadlock*) — это ситуация в СУБД, при которой **двое или более сеансов находятся в состоянии бесконечного ожидания ресурсов**, захваченных самими этими же сеансами



Пример. Взаимоблокировка



Итоги

- В различных СУБД механизм транзакций реализован по-разному.
- **Для наиболее эффективного использования конкретной СУБД необходимо понимать то, как в ней реализован механизм транзакций.**
- СУБД Oracle имеет эффективный механизм транзакций, который допускает одновременную работу большого числа пользователей.