Транзакции в РСУБД

Определение

- **Транзакция** – это последовательность операторов манипулирования данными, выполняющаяся как единое целое (все или ничего) и переводящая базу данных из одного целостного состояния в другое целостное состояние

Демаркация транзакций

- Транзакция обычно начинается автоматически с момента присоединения пользователя к СУБД
- Транзакция завершается при следующих событиях:
  - Подана команда зафиксировать транзакцию
  - Подана команда откатить транзакцию
  - Произошло отсоединение пользователя от СУБД
    - Автоматическая фиксация транзакции
  - Произошел сбой системы
    - При последующем запуске СУБД выполняется накат или откат тех транзакций, результаты которых не были сохранены в БД

Управление транзакциями в SQL

- Операторы начала и завершения транзакции могут отличаться в разных СУБД
- Режим AUTO COMMIT - каждый запрос выполняется в отдельной транзакции, автоматически завершающейся подтверждением
- СУБД MySQL, механизм InnoDB:
  - `START TRANSACTION` - явное начало транзакции
  - `COMMIT` - подтверждение транзакции
  - `ROLLBACK` - откат транзакции
  - `SET TRANSACTION ISOLATION LEVEL` - установка уровня изоляции транзакции
  - `SET AUTOCOMMIT = 0` - отключение режима

Транзакции и параллелизм

Работа транзакций в смеси

- Транзакция рассматривается как последовательность элементарных атомарных операций
- Набор из нескольких транзакций, элементарные операции которых чередуются друг с другом, называется ***смесью транзакций***
- Последовательность, в которой выполняются элементарные операции заданного набора транзакций, называется ***графиком запуска*** набора транзакций

Работа транзакций в смеси - пример

- Смесь транзакций:

$$T = (T_1, T_2, T_3, \ldots, T_n)$$

$$Q = (Q_1, Q_2, Q_3, \ldots, Q_m)$$

$$S = (S_1, S_2, S_3, \ldots, S_l)$$

- Графики запуска:

$$(T_1, Q_1, T_2, S_1, T_3, S_2, S_3, Q_2, \ldots)$$

$$\{S_1, Q_1, T_1, T_2, T_3, S_2, Q_2, S_3, \ldots\}$$

> Какой из графиков «правильнее» (и оптимальнее)?

Проблемы параллельной работы транзакций

- Потеря результатов обновления
- Незафиксированная зависимость (чтение "грязных" данных, неаккуратное считывание)
- Неповторяемое считывание
- Фиктивные элементы (фантомы)
- Несовместимый анализ

Потеря результатов обновления

Транзакция A	Время	Транзакция B
Чтение $P = P_0$	t_1	---
---	t_2	Чтение $P = P_0$
Запись $P_1 \rightarrow P$	t_3	---
---	t_4	Запись $P_2 \rightarrow P$
Фиксация транзакции	t_5	---
---	t_6	Фиксация транзакции
Потеря результата обновления		

- Транзакция A потеряла результаты своей работы

«Грязное чтение»

Транзакция A	Время	Транзакция B
---	t_1	Чтение $P = P_0$
---	t_2	Запись $P_1 \rightarrow P$
Чтение $P = P_1$	t_3	---
Работа с прочитанными данными P_1	t_4	---
---	t_5	Откат транзакции $P_0 \rightarrow P$
Фиксация транзакции	t_6	---
Работа с "грязными" данными		

- Результаты работы транзакции A некорректны

Неповторяемое считывание

Транзакция A	Время	Транзакция B
Чтение $P = P_0$	t_1	---
---	t_2	Чтение $P = P_0$
---	t_3	Запись $P_1 \rightarrow P$
---	t_4	Фиксация транзакции
Повторное чтение $P = P_1$	t_5	---
Фиксация транзакции	t_6	---
Неповторяемое считывание		

- Транзакция A работает с данными, которые, с точки зрения транзакции A, самопроизвольно изменяются

Фантом

Транзакция A	Время	Транзакция B
Выборка строк, удовлетворяющих условию α. (Отобрано n строк)	t_1	---
---	t_2	Вставка новой строки, удовлетворяющей условию α.
---	t_3	Фиксация транзакции
Выборка строк, удовлетворяющих условию α. (Отобрано n+1 строк)	t_4	---
Фиксация транзакции	t_5	---
Появились строки, которых раньше не было		

- Транзакция A в двух одинаковых выборках строк получила разные результаты

Несовместимый анализ

Транзакция A	Время	Транзакция B
Чтение счета P_1 = 100 и суммирование. SUM = 100	t_1	---
---	t_2	Снятие денег со счета P_3. P_3 : 100 → 50
---	t_3	Помещение денег на счет P_1. P_1 : 100 → 150
---	t_4	Фиксация транзакции
Чтение счета P_2 = 100 и суммирование. SUM = 200	t_5	---
Чтение счета P_3 = 50 и суммирование. SUM = 250	t_6	---
Фиксация транзакции	t_7	---
Сумма $250 по всем счетам неправильная - должно быть $300		

Конфликты доступа к данным

- Транзакции называются конкурирующими, если они пересекаются по времени и обращаются к одним и тем же данным

А \ Б	**Чтение**	**Запись**
Чтение	* ОК *	Неповторяемое считывание
Запись	«Грязное» чтение	Потеря обновления

Сериализация транзакций

- График запуска называется последовательным, если элементарные операции транзакций *не чередуются* друг с другом
- График запуска называется верным (сериализуемым), если он эквивалентен какому-либо последовательному графику
  - При их выполнении будет получен один и тот же результат, независимо от начального состояния базы данных
- Задача обеспечения изолированной работы пользователей — поиск оптимального сериализуемого графика запуска транзакций
  - Например, суммарное время выполнения всех транзакций в наборе

Способы разрешения конкуренции между транзакциями

- Транзакции не мешают друг другу, если они обращаются к разным данным или выполняются в разное время
- 1. "Притормаживать" некоторые из поступающих транзакций настолько, насколько это необходимо для обеспечения правильности смеси транзакций в каждый момент времени
  - > блокировки различных видов, метод временных меток
- 2. Предоставить конкурирующим транзакциям "разные" экземпляры данных
  - > многоверсионная БД, использование данных из журнала транзакций

Блокировки

- Если для выполнения транзакции необходимо, чтобы некоторый объект не изменялся без ведома этой транзакции, то этот объект должен быть заблокирован
- Базовая модель — два типа блокировок:
  - Монопольные блокировки (X-блокировки) - блокировки без взаимного доступа (блокировка записи)
  - Разделяемые блокировки (S-блокировки) - блокировки с взаимным доступом (блокировка чтения)
- Преднамеренные блокировки (IS-, IX-, SIX-блокировки)

Матрица совместимости блокировок

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

Протокол доступа к данным с блокировками

- 1. Прежде чем прочитать объект, транзакция должна наложить на этот объект S-блокировку.
- 2. Прежде чем обновить объект, транзакция должна наложить на этот объект X-блокировку. Если транзакция уже заблокировала объект S-блокировкой (для чтения), то перед обновлением объекта S-блокировка должна быть заменена X-блокировкой.
- 3. Если блокировка объекта транзакцией B отвергается оттого, что объект уже заблокирован транзакцией A, то транзакция B переходит в состояние ожидания. Транзакция B будет находиться в состоянии ожидания до тех пор, пока транзакция A не снимет блокировку объекта.
- 4. X-блокировки, наложенные транзакцией A, сохраняются до конца транзакции A.

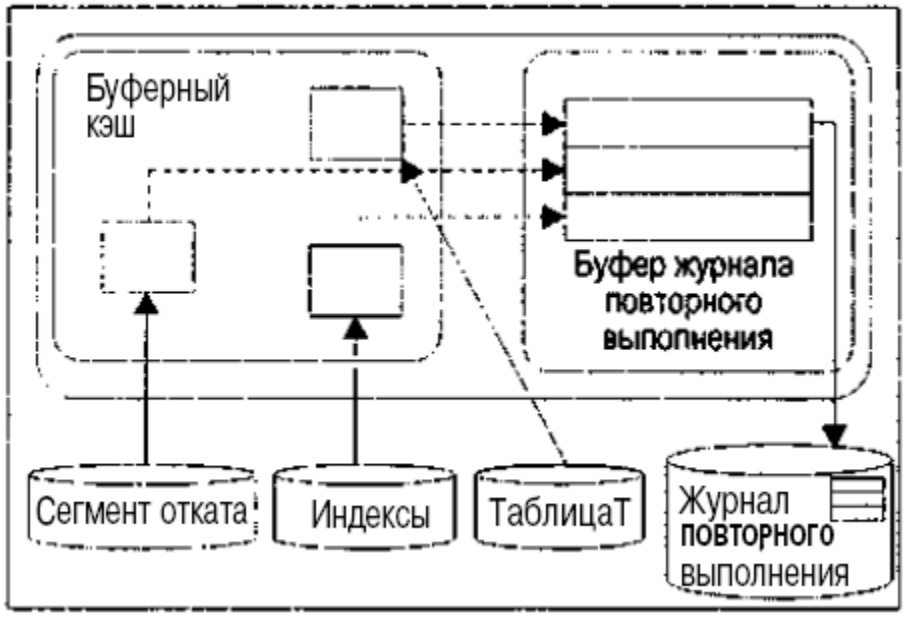
Потеря результатов обновления

Транзакция A	Время	Транзакция B
S-блокировка P - успешна	t_1	---
Чтение $P = P_0$	t_2	---
---	t_3	S-блокировка P - успешна
---	t_4	Чтение $P = P_0$
X-блокировка P - отвергается	t_5	---
Ожидание…	t_6	X-блокировка P - отвергается
Ожидание…	t_7	Ожидание…

- **Тупик**
  - Решение - откат одной из транзакций (транзакции-жертвы) так, чтобы другие транзакции продолжили свою работу

Выделение версий данных

- Для генерации разных версий данных используется журнал транзакций
- Журнал транзакций предназначен для выполнения операции отката при неуспешном выполнении транзакции или для восстановления данных после сбоя системы



← СУБД Oracle

Выделение версий данных

- Для каждой транзакции (или запроса) запоминается текущий системный номер (SCN - System Current Number).
- При записи страниц данных на диск фиксируется SCN транзакции, производящей эту запись.
- Транзакции, только читающие данные, не блокируют ничего в базе данных.
- Если транзакция A читает страницу данных, то SCN транзакции A сравнивается с SCN читаемой страницы данных.
- Если SCN страницы данных меньше или равен SCN транзакции A, то транзакция A читает эту страницу.
- Если SCN страницы данных больше SCN транзакции A, то значит некоторая транзакция B, начавшаяся позже транзакции A, успела изменить данные страницы. В этом случае транзакция A просматривает журнал транзакций назад в поиске первой записи об изменении нужной страницы данных с SCN меньшим, чем SCN транзакции A. Найдя такую запись, транзакция A использует старый вариант данных страницы.

Несовместимый анализ

Транзакция A	Время	Транзакция B
Проверка SCN счета P_1 - SCN транзакции больше SCN счета. Чтение счета $P_1 = 100$ без наложения блокировки и суммирование. $SUM = 100$	t_1	---
---	t_2	X-блокировка счета P_3 - успешна
---	t_3	Снятие денег со счета P_3. $P_3 : 100 \rightarrow 50$
---	t_4	X-блокировка счета P_1 - успешна
---	t_5	Помещение денег на счет P_1. $P_1 : 100 \rightarrow 150$
---	t_6	Фиксация транзакции (Снятие блокировок)

Несовместимый анализ (продолж.)

---	t_6	Фиксация транзакции (Снятие блокировок)
Проверка SCN счета P_2 - SCN транзакции больше SCN счета. Чтение счета $P_2 = 100$ без наложения блокировка и суммирование. $SUM = 200$	t_7	---
Проверка SCN счета P_3 - SCN транзакции **МЕНЬШЕ** SCN счета. Чтение старого варианта счета $P_3 = 100$ и суммирование. $SUM = 300$	t_8	---
Фиксация транзакции	t_9	---
Сумма на счетах посчитана правильно.		

Сравнение блокировок и выделения версий

- Блокировки
  - В базовой модели решают не все проблемы параллельной работы транзакций
  - Возможны тупики
- Механизм выделения версий
  - Реализован в СУБД Oracle, MySQL («движок» Falcon)
  - Дополняет механизм блокировок
  - Позволяет избежать тупиков

Реализация изолированности транзакций средствами SQL

- Уровень изоляции транзакции
  - В SQL нет понятия «блокировок»
  - Определяет требования к изолированности транзакций
  - Производитель СУБД может реализовать эти требования любыми способами
- READ UNCOMMITTED - уровень незавершенного считывания
- READ COMMITTED - уровень завершенного считывания
- REPEATABLE READ - уровень повторяемого считывания.
- SERIALIZABLE - уровень способности к упорядочению

Нарушения способности к упорядочению

Уровень изоляции	Неаккуратное считывание	Неповторяемое считывание	Фантомы
READ UNCOMMITTED	Да	Да	Да
READ COMMITTED	Нет	Да	Да
REPEATABLE READ	Нет	Нет	Да
SERIALIZABLE	Нет	Нет	Нет

- Потеря результатов обновления стандартом SQL не допускается

Установка уровня изоляции

- ```
  SET TRANSACTION {ISOLATION LEVEL
  {READ UNCOMMITTED | READ COMMITTED
       | REPEATABLE READ | SERIALIZABLE}
  | {READ ONLY | READ WRITE}}
  ```
- Уровень изоляции транзакции одновременно влияет на скорость выполнения и на целостность данных
- Не все проблемы параллелизма актуальны для конкретного приложения => необходимо управлять уровнем изоляции транзакции
- Большинству транзакций не требуется делать выборки одних и тех же данных более 1 раза => уровень изоляции READ COMMITTED (принят по умолчанию во многих СУБД)
  ```