

# Транзакции и параллелизм



свойство (**И**) - *изолированность транзакций*

# Изолированность

- Современные СУБД являются **многопользовательскими системами**, т.е. допускают параллельную одновременную работу большого количества пользователей. При этом пользователи не должны мешать друг другу.
- СУБД должна быть организована так, чтобы у пользователя складывалось **впечатление**, что их транзакции выполняются **независимо от транзакций других пользователей**.

# Изолированность

- Простейший способ обеспечить такую иллюзию у пользователя - все поступающие транзакции выстраивать в **единую очередь** и выполнять строго по очереди, но это **не параллельная работа**
- Таким образом, транзакции необходимо выполнять **одновременно**, но так, чтобы результат был бы такой же, как если бы транзакции выполнялись по очереди.

# Условия



СУБД гарантирует, что, с точки зрения пользователя, будут выполнены **два условия**:

- Эта операция будет выполнена **целиком** или не выполнена вовсе (*атомарность* - все или ничего).
- Во время выполнения этой операции не выполняются никакие другие операции других транзакций (**строгая очередность элементарных операций**).

# Работа транзакций в смеси

- Элементарные операции различных транзакций могут выполняться в произвольной очередности. Например, если есть несколько транзакций, состоящих из последовательности операций элементарных:

$T = \{T_1, T_2, T_3, \dots, T_n\}$      $Q = \{Q_1, Q_2, Q_3, \dots, Q_m\}$

- то реальная последовательность, в которой СУБД выполняет эти транзакции может быть, например, такой:

$\{T_1, Q_1, T_2, T_3, Q_2, T_4, Q_3, Q_4, Q_5, T_5, T_6, \dots\}$

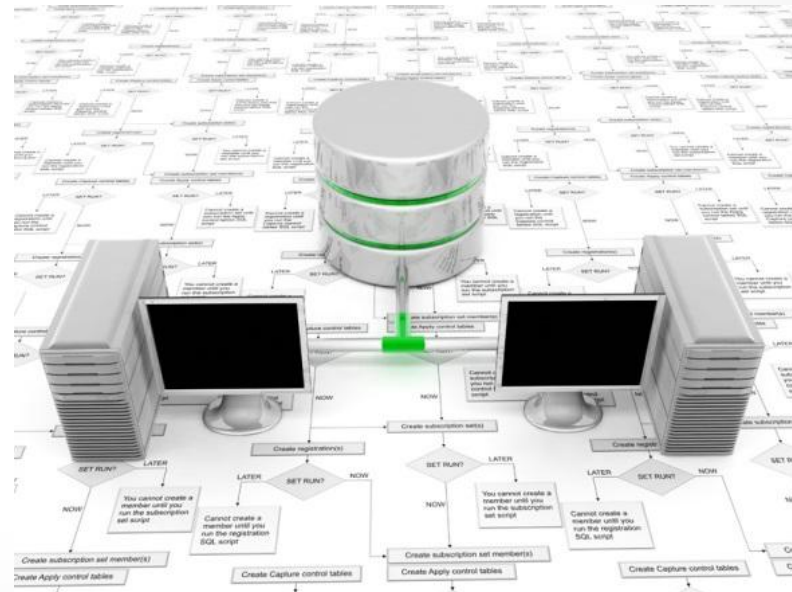
# Смесь транзакций

- Определение: Набор из нескольких транзакций, элементарные операции которых чередуются друг с другом, называется **смесью транзакций**.
- Определение: Последовательность, в которой выполняются элементарные операции заданного набора транзакций, называется **графиком запуска набора транзакций**.

# Требование к графику запуска транзакций

## запуска транзакций

- График запуска должен быть **ОПТИМАЛЬНЫМ**, то есть давать минимальное среднее время выполнения транзакций каждым пользователем.



## работы транзакций

### (параллелизма)

- 1) Проблема потери результатов обновления
- 2) Проблема незафиксированной зависимости (чтение "грязных" данных)
- 3) Проблема несовместимого анализа



# Обозначения:

- Рассмотрим две транзакции, А и В, запускающиеся в соответствии с некоторыми графиками.
- Пусть транзакции работают с некоторыми объектами базы данных, например со строками таблицы.
- Операцию **чтения строки** Р будем обозначать  **$P=P_0$** , где  $P_0$  - прочитанное значение.
- Операцию **записи** значения  $P_1$  в строку Р будем обозначать  **$P_1 \rightarrow P$** .

# Проблема потери результатов

## обновления

Две транзакции по очереди записывают некоторые данные в одну и ту же строку и фиксируют изменения.

Транзакция А	Время	Транзакция В
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
<b>Потеря результата обновления</b>		

# Результат

- После окончания обеих транзакций, строка содержит значение , занесенное более поздней транзакцией В.
- Транзакция А ничего не знает о существовании транзакции В, и естественно ожидает, что в строке содержится значение .
- Таким образом, транзакция А **потеряла результаты своей работы.**

# Проблема незафиксированной зависимости

(чтение "грязных" данных или неаккуратное считывание)

- Транзакция В изменяет данные в строке
- После этого транзакция А читает измененные данные и работает с ними
- Транзакция В откатывается и восстанавливает старые данные

# Проблема незафиксированной зависимости

Транзакция А	Время	Транзакция В
---	$t_1$	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	$t_6$	---
<b>Работа с "грязными" данными</b>		

# Результат

- Транзакция А в своей работе использовала данные, которых **нет и не было** в базе данных.
- После отката транзакции В, должна восстановиться ситуация, как если бы транзакция В вообще *никогда не выполнялась*.
- Таким образом, результаты работы транзакции А некорректны, т.к. она работала с данными, **отсутствовавшими в базе данных**.

# Проблема

## несовместимого анализа

Проблема несовместимого анализа включает следующие варианты:

- ▣ **Неповторяемое считывание**
- ▣ **Фиктивные элементы (фантомы)**
- ▣ **Собственно несовместимый анализ**

# Неповторяемое считывание

Транзакция А дважды читает одну и ту же строку. Между этими чтениями вклинивается транзакция В, которая изменяет значения в строке.

Транзакция А	Время	Транзакция В
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	$t_6$	---
<b>Неповторяемое считывание</b>		



# Результат

Транзакция А работает с данными, которые, с точки зрения транзакции А, самопроизвольно изменяются.

# Фиктивные элементы (фантомы)

Транзакция А дважды выполняет выборку строк с одним и тем же условием. Между выборками вклинивается транзакция В, которая добавляет новую строку, удовлетворяющую условию отбора.

# Фиктивные элементы

Транзакция А	Время	Транзакция В
Выборка строк, удовлетворяющих условию $\alpha$ . (Отобрано $n$ строк)	$t_1$	---
---	$t_2$	Вставка новой строки, удовлетворяющей условию $\alpha$ .
---	$t_3$	Фиксация транзакции
Выборка строк, удовлетворяющих условию $\alpha$ . (Отобрано $n+1$ строк)	$t_4$	---
Фиксация транзакции	$t_5$	---
Появились строки, которых раньше не было		

Результат. Транзакция А в двух одинаковых выборках строк получила разные результаты.

# Собственно

## несовместимый анализ

- в смеси присутствуют две транзакции - одна длинная, другая короткая.
- Длинная транзакция выполняет некоторый анализ по всей таблице, например, подсчитывает общую сумму денег на счетах клиентов банка для главного бухгалтера. Пусть на всех счетах находятся одинаковые суммы, например, по \$100. Короткая транзакция в этот момент выполняет перевод \$50 с одного счета на другой так, что общая сумма по всем счетам не меняется.

# Собственно несовместимый анализ

Транзакция А	Время	Транзакция В
Чтение счета $P_1 = 100$ и суммирование. $SUM = 100$	$t_1$	---
---	$t_2$	Снятие денег со счета $P_3$ . $P_3 : 100 \rightarrow 50$
---	$t_3$	Помещение денег на счет $P_1$ . $P_1 : 100 \rightarrow 150$
---	$t_4$	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	$t_7$	---
<b>Сумма S250 по всем счетам неправильная - должно быть S300</b>		

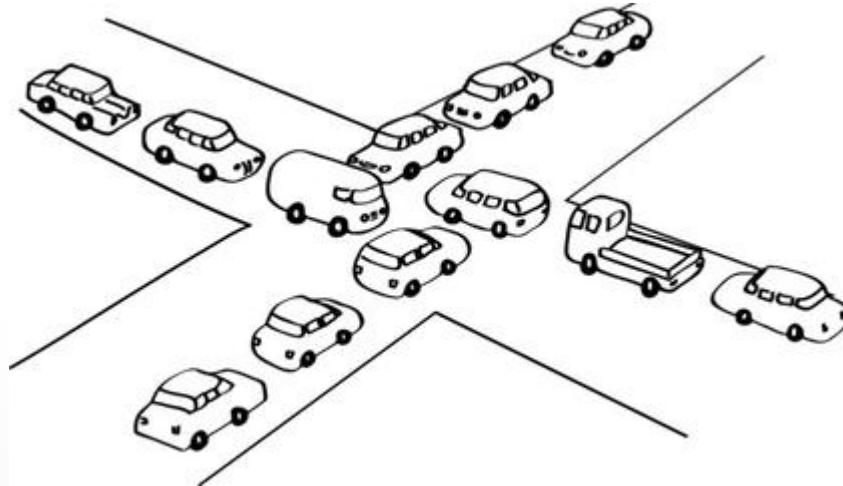
Результат. Хотя транзакция В все сделала правильно - деньги переведены без потери, но в результате транзакция А подсчитала неверную общую сумму.

# Конфликты между транзакциями

- при работе в смеси нарушается свойство (И) транзакций - изолированность. Транзакции реально мешают друг другу получать правильные результаты.
- Однако не всякие транзакции мешают друг другу.
- Транзакции не мешают друг другу, если они обращаются к разным данным или выполняются в разное время.

# Конфликты между транзакциями

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



# Виды конфликтов

1. **W-W (Запись - Запись)**. Первая транзакция изменила объект и не закончилась. Вторая транзакция пытается изменить этот объект. Результат - потеря обновления.
2. **R-W (Чтение - Запись)**. Первая транзакция прочитала объект и не закончилась. Вторая транзакция пытается изменить этот объект. Результат - несовместимый анализ (неповторяемое считывание).
3. **W-R (Запись - Чтение)**. Первая транзакция изменила объект и не закончилась. Вторая транзакция пытается прочитать этот объект. Результат - чтение "грязных" данных.

Конфликты типа **R-R (Чтение - Чтение)** отсутствуют, т.к. данные при чтении не изменяются.



# Способы разрешения конкуренции транзакций

Притормаживание  
некоторых  
транзакций



Блокировки



Предоставление  
транзакциям разных  
версий данных



Журнал  
транзакций



# Блокировки



- Основная идея блокировок заключается в том, что если для выполнения некоторой транзакции необходимо, чтобы объект не изменялся, то этот объект должен быть **заблокирован**, т.е. доступ к нему со стороны других транзакций ограничивается на время выполнения транзакции, вызвавшей блокировку.

# Типы блокировок

- **Монопольные блокировки (X-блокировки, X-locks** - exclusive locks) - блокировки без взаимного доступа (блокировка **записи**).
- **Разделяемые блокировки (S-блокировки, S-locks** - Shared locks) - блокировки с взаимным доступом (блокировка **чтения**).

# Блокировки

- Если транзакция А блокирует объект при помощи **X-блокировки**, то всякий доступ к этому объекту со стороны других транзакций **отвергается**.
- Если транзакция А блокирует объект при помощи **S-блокировки**, то
  - a. запросы со стороны других транзакций на X-блокировку этого объекта будут отвергнуты,
  - b. запросы со стороны других транзакций на S-блокировку этого объекта будут приняты.

# Матрица совместимости блокировок

	Транзакция В пытается наложить блокировку:	
Транзакция А 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

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, то перед обновлением объекта S-блокировка должна быть заменена X-блокировкой.
3. Если блокировка объекта транзакцией В отвергается оттого, что объект уже заблокирован транзакцией А, то транзакция В переходит в **состояние ожидания**. Транзакция В будет находиться в состоянии ожидания до тех пор, пока транзакция А не снимет блокировку объекта.
4. X-блокировки, наложенные транзакцией А, сохраняются до конца транзакции А.