

Введение в моделирование данных, базы данных и SQL

Лекции 10-11

Транзакции

Что такое транзакция?

- Транзакция —логическая **единица активности** в ИС, это последовательность операций над БД.
- неделимая (с тч.зр. воздействия на БД) **единица обработки** (*все или ничего*)
 - Если часть действий транзакции не выполнена, БД **возвращается в исходное состояние**, которое было до начала транзакции (**откат транзакции**)
- **Единица восстановления** данных после сбоев
 - восстанавливаясь, СУБД ликвидирует следы транзакций, не успевших завершиться в результате сбоя
- Обеспечивает **изолированную работу пользователей**, работающих с одной БД
 - в многопользовательских системах

Единица обработки

- Транзакция –это последовательность операций над БД
 - которые должны быть исполнены как целое (*все или ничего*)
 - переводящих БД из одного целостного состояния в другое целостное состояние;
 - в течение выполнения БД может временно быть в несогласованном состоянии

БД в целостном состоянии

Счет Тимы	1000
Счет Алекса	0

Начало транзакции

БД в целостном состоянии

Счет Тимы	500
Счет Алекса	500

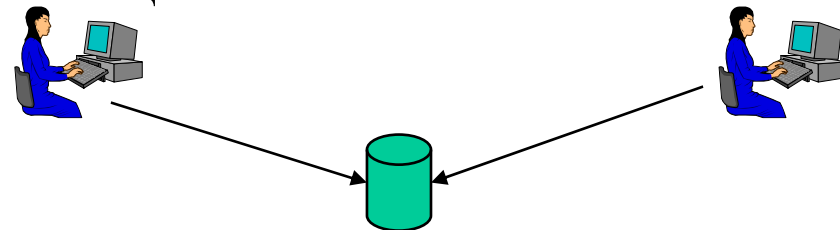
Выполнение транзакции

Конец транзакции

Перевести
500

Требования к согласованности БД

- **Восстановление согласованного состояния**
 - Системные **сбои** как аппаратные, так программные не должны сказываться на целостном состоянии БД
- **Управление параллельной работой**
 - Большинство СУБД **многопользовательские**
 - Параллельное выполнение многих различных транзакций разных пользователей должно быть организовано так, чтобы каждая транзакция **не мешала** (не интерферировала) с другими, чтобы это не приводило к некорректным результатам.
 - Параллельное выполнение транзакций должно быть таким, чтобы каждая транзакция как бы выполнялась изолировано



АСИД (ACID) свойства транзакций

(A) **Атомарность** (Atomicity). Транзакция выполняется как атомарная операция

- либо выполняется вся транзакция целиком, либо она целиком не выполняется.

(C) **Согласованность** (Consistency). Сохранение целостности БД.

- Транзакция переводит БД из одного согласованного (целостного) состояния в другое согласованное состояние. Внутри транзакции согласованность данных может нарушаться.

(I) **Изоляция** (Isolation/Independence). Выполнение транзакций изолировано друг от друга.

- Модификации БД, выполняемые в транзакции, не видны другим транзакциям, до ее завершения (решает проблему временной модификации).
- **Сериализуемость**: транзакции сериализуемы, если эффект их выполнения в режиме чередования элементарных операций (параллельно) эквивалентен эффекту их некоторого последовательного выполнения

(D) **Долговечность** (Durability). Если транзакция выполнена, то результаты ее работы должны сохраниться в БД, даже если в следующий момент произойдет сбой системы.

COMMIT и ROLLBACK

- Транзакция **начинается**
 - автоматически с момента присоединения пользователя к СУБД
 - автоматически после операторов **COMMIT** и **ROLLBACK**
- и **продолжается** до тех пор, пока
 - произошло **отсоединение** пользователя от СУБД
 - произошел **сбой** системы
 - подана **команда COMMIT** (зафиксировать транзакцию)
 - завершить текущую и автоматически начать новую
 - результаты работы сохраняются в базе данных
 - подана **команда ROLLBACK** (откатить транзакцию)
 - отменить текущую и автоматически начать новую
 - все изменения отменяются так *как будто их вообще не было*
- Операторы **COMMIT** и **ROLLBACK** обеспечивают управление транзакциями

Многопользовательская работа

- Свойства
 - (И) - изолированность транзакций
 - (С) – согласованность, сохранение целостности БД
- параллельная одновременная работа большого количества пользователей
- пользователи **не должны мешать** друг другу
- логической единицей работы – транзакция
 - транзакции как бы выполняются независимо от транзакций других пользователей
- все транзакции **выстраивать в единую очередь** и выполнять строго **по очереди**
- транзакции необходимо выполнять одновременно так, чтобы **результат был бы такой же**, как если бы транзакции выполнялись по очереди

Смеси транзакций

- Транзакция - последовательность элементарных атомарных операций
- внутри каждой транзакции последовательность элементарных операций строго определенная
- Элементарные операции различных транзакций могут выполняться в произвольной очередности
- *смесь транзакций* - чередующиеся друг с другом элементарные операции нескольких транзакций
 - изолированность пользователей - выбор подходящей смеси транзакций (графика запуска)
 - должна быть оптимальной в некотором смысле

$$T = \{T_1, T_2, T_3, \dots, T_n\}$$

$$Q = \{Q_1, Q_2, Q_3, \dots, Q_m\}$$

$$S = \{S_1, S_2, S_3, \dots, S_l\} \quad (T_1, Q_1, T_2, S_1, T_3, S_2, S_3, Q_2, \dots)$$

Проблемы параллельной работы

- три основные проблемы параллелизма:
 - Проблема *потери изменений* (записи "*грязных*" данных)
 - Проблема *чтения "грязных"* данных (незафиксированной зависимости , неаккуратное считывание)
 - Проблема **несовместимого анализа**.
 - *Неповторяемое считывание*
 - Фиктивные элементы (*фантомы*)
 - Собственно *несовместимый анализ*

- http://myy.haaga-helia.fi/~dbms/dbtechnet/download/SQL-Transactions_handbook_RU.pdf
- <https://habrahabr.ru/company/infopulse/blog/261097/> + <https://habrahabr.ru/company/infopulse/blog/261101/>
- <https://technet.microsoft.com/en-us/library/cc546518.aspx>

Потеря изменений

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
Потеря результата обновления		

- После окончания обеих транзакций, строка P содержит значение, занесенное более поздней транзакцией
- транзакция А потеряла свой результат, не знает о B .
 - ожидает, что в строке P содержится значение P_1

Потеря изменений

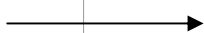

Запись $P \rightarrow 0$

t_0

Транзакция А	Время	Транзакция В
Чтение $P = P_0$	t_1	---
---	t_2	Чтение $P = P_0$
Запись $P \rightarrow P+11$	t_3	---
---	t_4	Запись $P \rightarrow P+22$
Фиксация транзакции	t_5	---
---	t_6	Фиксация транзакции

- До начала транзакций строка содержит 0, выполняются две параллельных транзакции, увеличивающие значение на 11 и 22 соответственно. Какое значение будет после t_6 ? Какое после t_5 до t_6 ?
- Каким будет значение, если не можем гарантировать такой порядок завершения транзакций? Если В откатить?

«Грязное» чтение

Транзакция А	Врем	Транзакция В
---	Я t_1 	Чтение $P = P_0$
---	t_2 	Запись $P_1 \rightarrow P$
Чтение $P = P_1$ 	t_3	---
Работа с прочитанными P_1	t_4	---
данными ---	t_5 	Откат $P_0 \rightarrow P$
Фиксация транзакции	 t_6	транзакции ---
Незафиксированная зависимость		

- Транзакция А в своей работе использовала данные, которых нет, и *не было* в БД

Проблема несовместимого анализа

несколько различных вариантов:

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

Неповторяемое считывание

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	---
Неповторяющееся чтение		



- Транзакция А работает с данными, которые, ее с точки зрения самопроизвольно меняются.

Фиктивные элементы (фантомы)

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

- Транзакция А в двух одинаковых выборках строк получила разные результаты

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	---
Сумма \$250 по всем счетам неправильная - должно быть \$300		

- транзакция А подсчитала неверную общую сумму.

Конфликты между транзакциями

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

Способы разрешения конкуренции

1. **"Притормаживать"** некоторые из поступающих транзакций настолько, насколько это необходимо для обеспечения правильности смеси транзакций в каждый момент времени (т.е. обеспечить, чтобы конкурирующие транзакции выполнялись *в разное время*).
2. Предоставить конкурирующим транзакциям **"разные"** экземпляры **данных** (т.е. обеспечить, чтобы конкурирующие транзакции работали с *разными версиями данными*).

Блокировки

Два типа 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»).

Свойства блокировок

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

Совместимость блокировок

Блокировка транзакции А	Блокировка транзакция В	
	S-блокировка	X-блокировка
S-блокировка	Да	НЕТ (Конфликт R-W)
X-блокировка	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 (для чтения), то перед обновлением объекта S-блокировка должна быть заменена X-блокировкой.
3. Если блокировка **объекта** транзакцией В отвергается из-за того, 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.

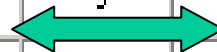
Проблема потери результатов обновления

Транзакция А	Время	Транзакция В
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...
Ожидание...		Ожидание...

- Обе транзакции ожидают друг друга и не могут продолжаться. Возникла ситуация **тупика**

Общий вид тупика (dead locks)

Транзакция А	Время	Транзакция В
Блокировка объекта P_1 - успешна	t_1	---
---	t_2	Блокировка объекта P_2 - успешна
Блокировка объекта P_2 конфликтует с блокировкой, наложенной транзакцией В	t_3	---
Ожидание...	t_4	Блокировка объекта P_1 конфликтует с блокировкой, наложенной транзакцией А
Ожидание...	t_5	Ожидание...
Ожидание...		Ожидание...





Проблема незафиксированной зависимости

Транзакция А	Время	Транзакция В
---	t_1	S-блокировка P - успешна
---	t_2	Чтение $P = P_0$
---	t_3	X-блокировка P - успешна
---	t_4	Запись $P_1 \rightarrow P$
S-блокировка P - отвергается	t_5	---
Ожидание...	t_6	Откат транзакции $P_0 \rightarrow P$ (Блокировка снимается)
S-блокировка P - успешна	t_7	---
Чтение $P = P_0$	t_8	---
Работа с прочитанными данными P_0	t_9	---
---	t_{10}	---
Фиксация транзакции	t_{11}	---
Все правильно		

- Транзакция А «**притормозилась**» до (отката) окончания транзакции В. Конфликт разрешен за счет увеличения времени работы транзакции А

Неповторяемое считывание

Транзакция А	Время	Транзакция В
S-блокировка P - успешна	t_1	---
Чтение $P = P_0$ ←	t_2	---
---	t_3	X-блокировка P - отвергается
---	t_4	Ожидание...
Повторное чтение $P = P_0$ ←	t_5	Ожидание...
Фиксация транзакции (Блокировка снимается) 	t_6	Ожидание...
---	t_7	X-блокировка P - успешна
---	t_8 ←	Запись $P_1 \rightarrow P$
---	t_9 	Фиксация транзакции (Блокировка снимается)
Все правильно		

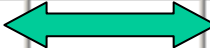
- Транзакция В «**притормозила**» до окончания транзакции А. Конфликт разрешен за счет увеличения времени работы транзакции В.

Фиктивные элементы (фантомы)

Транзакция А	Время	Транзакция В
S-блокировка строк, удовлетворяющих условию α . (Заблокировано n строк)	t_1	---
Выборка строк, удовлетворяющих условию α . (Отобрано n строк)	t_2	---
---	t_3	Вставка новой строки, удовлетворяющей условию α .
---	t_4	Фиксация транзакции
S-блокировка строк, удовлетворяющих условию α . (Заблокировано $n+1$ строка)	t_5	---
Выборка строк, удовлетворяющих условию α . (Отобрано $n+1$ строк)	t_6	---
Фиксация транзакции	t_7	---
Появились строки, которых раньше не было		

- Блокировка на уровне строк **не решила** проблему появления фиктивных элементов

Собственно несовместимый анализ

Транзакция А	Время	Транзакция В
S-блокировка счета P_1 - успешна	t_1	---
Чтение счета $P_1 = 100$ и суммирование. $SUM = 100$	t_2	---
---	t_3	X-блокировка счета P_3 - успешна
---	t_4	Снятие денег со счета P_3 . $P_3 : 100 \rightarrow 50$
---	t_5	X-блокировка счета P_1 - отвергается
---	t_6	Ожидание...
S-блокировка счета P_2 - успешна	t_7	Ожидание...
Чтение счета $P_2 = 100$ и суммирование. $SUM = 200$	t_8	Ожидание...
S-блокировка счета P_3 - отвергается	t_9	Ожидание...
Ожидание...	t_{10}	Ожидание...
Ожидание...		Ожидание...

- Обе транзакции ожидают друг друга и не могут продолжаться. Возникла ситуация **тупика**.

Итог применения протокола

- Проблема потери результатов обновления
 - возник *тупик*.
- Проблема незафиксированной зависимости (чтение "грязных" данных, неаккуратное считывание)
 - *разрешилась*.
- Неповторяемое считывание
 - *разрешилась*.
- Появление фиктивных элементов
 - *не разрешилась*.
- Проблема несовместимого анализа
 - возник *тупик*.

Преднамеренные блокировки

- Блокировка самой **базы** данных.
- Блокировка **файлов** базы данных.
- Блокировка **таблиц** базы данных.
- Блокировка **страниц** (единиц обмена с диском, обычно 2-16 Кб. На одной странице содержится несколько строк одной или нескольких таблиц).
- Блокировка отдельных **строк** таблиц.
- Блокировка отдельных **полей**.

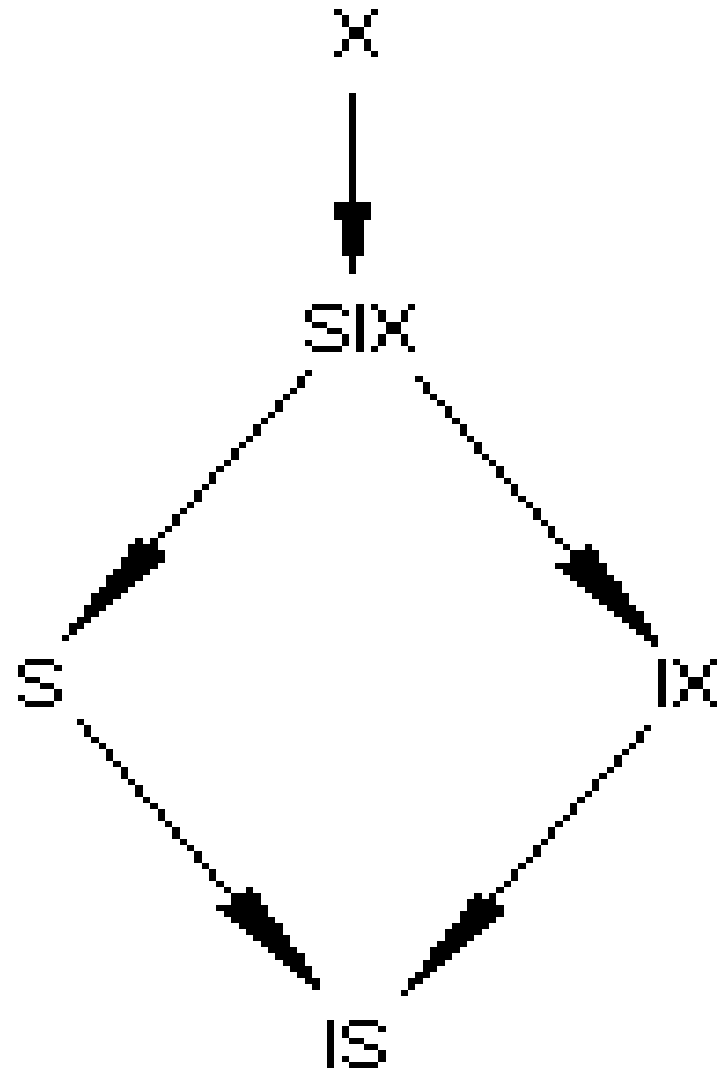
Дополнительные блокировки

- *Преднамеренная блокировка с возможностью **взаимного доступа** (IS-блокировка - Intent Shared lock).* Накладывается на некоторый **составной объект Т** и означает намерение заблокировать некоторый **входящий в Т объект** в режиме S-блокировки.
- *Преднамеренная блокировка **без взаимного доступа** (IX-блокировка – Intent eXclusive lock).* Накладывается на некоторый **составной объект Т** и означает намерение заблокировать некоторый **входящий в Т объект** в режиме X-блокировки.
- *Преднамеренная блокировка как **с возможностью взаимного доступа, так и без него** (SIX-блокировка - Shared Intent eXclusive lock).* Накладывается на некоторый **составной объект Т** и означает **разделяемую S-блокировку** всего этого объекта с намерением впоследствии заблокировать какие-либо **входящие в него объекты** в режиме X-блокировок.

Расширенная таблица совместимости блокировок

	Транзакция В пытается наложить на таблицу блокировку:				
Транзакция А наложила на таблицу блокировку:	IS	S	IX	SIX	X
IS	Да	Да	Да	Да	Нет
S	Да	Да	Нет	Нет	Нет
IX	Да	Нет	Да	Нет	Нет
SIX	Да	Нет	Нет	Нет	Нет
X	Нет	Нет	Нет	Нет	Нет

Диаграмма приоритета блокировок



Уровни изоляции транзакций в SQL

- ***READ UNCOMMITTED*** - уровень незавершенного считывания.
Нет потери изменений
- ***READ COMMITTED*** - уровень завершенного считывания.
Нет и "грязного" чтения
- ***REPEATABLE READ*** - уровень повторяемого считывания.
Нет еще и неповторяющегося чтения
- ***SERIALIZABLE*** - уровень способности к упорядочению. *Нет конфликтов.*

Уровни изоляции SQL

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

Оператор управления изоляциями

```
SET TRANSACTION ISOLATION LEVEL {  
    READ COMMITTED  
    | READ UNCOMMITTED  
    | REPEATABLE READ  
    | SERIALIZABLE }
```

— определяет режим выполнения *следующей* транзакции

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE  
    READ
```

```
GO
```

```
BEGIN TRANSACTION
```

```
SELECT * FROM publishers
```

```
...
```

```
COMMIT TRANSACTION
```

SQL Server

- поддерживает три вида определения транзакций:
 - автоматическое
 - **SET IMPLICIT_TRANSACTIONS OFF**
 - по умолчанию; каждая команда рассматривается как отдельная транзакция
 - подразумеваемое
 - **SET IMPLICIT_TRANSACTIONS ON**
 - автоматически начинает новую транзакцию, как только завершена предыдущая
 - явное
 - режим работает поверх двух других;
 - указал начало и конец транзакции;
 - создать транзакцию, включающую несколько команд.

Явные транзакции

- **BEGIN TRANSACTION** [имя_транзакции]
 - фиксируются первоначальные значения изменяемых данных и момент начала транзакции
- **COMMIT [TRANSACTION [имя_транзакции]]**
 - зафиксировать все изменения, сделанные в транзакции,
 - применяется к наиболее "глубокой" вложенной транзакции, даже если указано имя транзакции
- **SAVE TRANSACTION** [имя_точки_сохранения]
 - сохраняет состояние БД в текущей точке и присваивает сохраненному состоянию имя точки сохранения
- **ROLLBACK [TRANSACTION] [имя_транзакции | имя_точки_сохранения]**
 - отменяет все изменения, сделанные в БД после первого BEGIN TRANSACTION (допустимо только имя самой верхней транзакции) или после точки сохранения (откатить лишь часть транзакции).
- **@@TRANCOUNT** - количество активных транзакций
- **@@NESTLEVEL** - уровень вложенности транзакций.

Пример

- **BEGIN TRANSACT TRANSACTION point1**
 - в point1 сохраняется первоначальное состояние таблицы Товар
- **DELETE FROM Товар WHERE КодТовара=2
SAVE TRANSACTION point2**
 - в point2 сохраняется состояние таблицы Товар без товаров с кодом 2.
- **DELETE FROM Товар WHERE КодТовара=3
SAVE TRANSACTION point3**
 - в point3 сохраняется состояние таблицы Товар без товаров с кодами 2 и 3.
- **DELETE FROM Товар WHERE КодТовара<>1
ROLLBACK TRANSACTION point3**
 - отменяется последнее удаление, возврат в состояние без товаров 2 и 3.
- **SELECT * FROM Товар**
 - покажет таблицу Товар без товаров с кодами 2 и 3.
- **ROLLBACK TRANSACTION point1**
 - Происходит возврат в первоначальное состояние таблицы.

Транзакции и восстановление данных

Транзакция как единица восстановления БД

- Свойства
 - (Д) - долговечность транзакций, данные зафиксированных транзакций **должны сохраняться**, даже если произойдет сбой системы
 - (С) – согласованность, **сохранение целостности БД**
- При аппаратном/программном сбое, возникающем в течение выполнения транзакции, БД окажется в несогласованном состоянии
 - Мягкие сбои – внезапная остановка
 - Аварийное завершение работы программы, СУБД, компьютера
 - Жесткие сбои - потеря информации
 - Сбои внешних носителей
- БД восстанавливается в некоторое состояние на основе последнего целостного состояния
- СУБД гарантирует, что если транзакция выполняет некоторые модификации, и происходит сбой до нормального завершения транзакции, то эти модификации будут ликвидированы (будет выполнен **откат**).

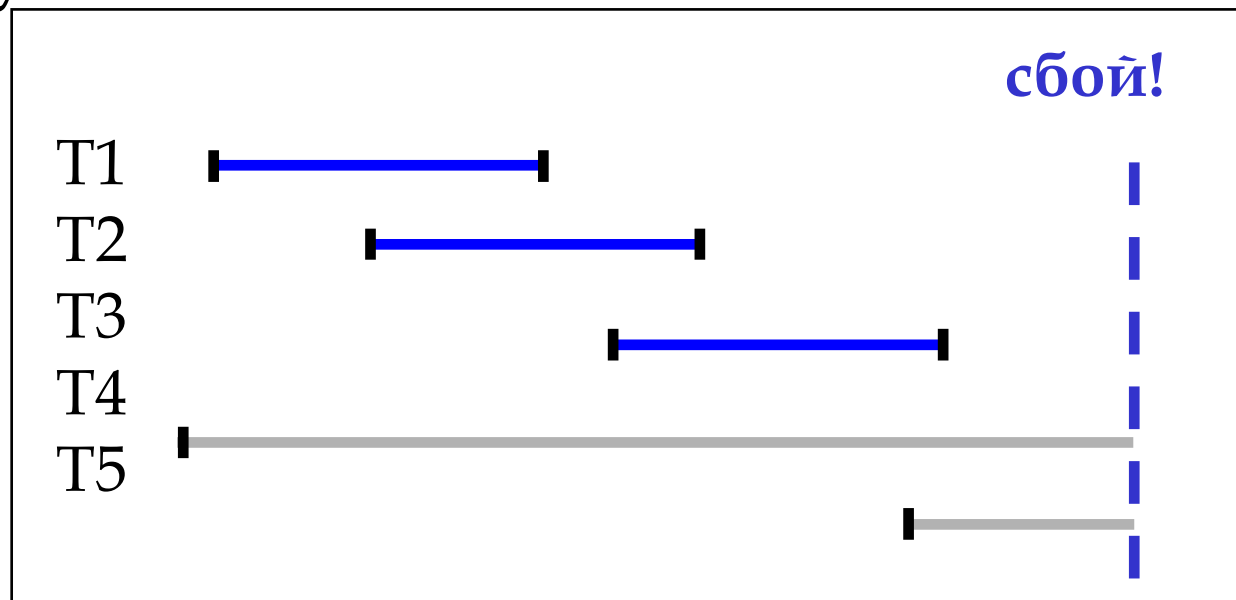
Виды восстановления данных

- Индивидуальный откат транзакции
 - ROLLBACK
 - СУБД
 - ошибка в работе транзакции (например, деление на нуль)
 - разрешение тупика (блокировки)
 - нужно устранить последствия операторов модификации
- Мягкий сбой системы
 - *аварийный отказ программного обеспечения (software)*
 - утрата данных в оперативной памяти
 - выполняющиеся в момент сбоя транзакции
 - *теряется содержимое всех буферов*
 - не повреждены данные на диске
- Жесткий сбой системы
 - *аварийный отказ аппаратуры (hardware)*
 - повреждение носителей внешней памяти

Восстановление БД

Общие принципы восстановления, в восстановленной БД:

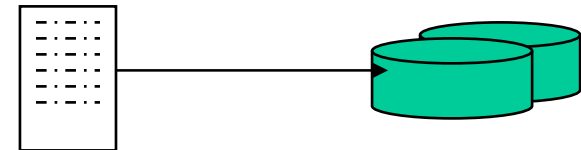
- ⑩ результаты **зафиксированных** транзакций должны быть **сохранены**
- ⑩ результаты **незафиксированных** транзакций должны **отсутствовать**.



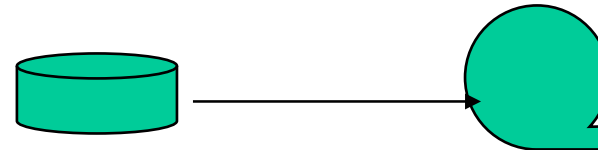
- ⑩ После перезапуска системы:
 - T1, T2 & T3 должны быть **сохранены**
 - T4 & T5 должны быть **ликвидированы** (их эффекты не должны быть обнаружены)

Варианты обеспечения восстановления

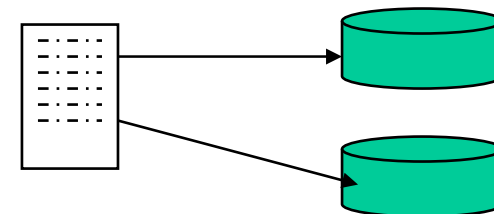
- Зеркалирование
 - Держать две копии БД и поддерживать их одновременно



- Резервирование (резервная копия БД)
 - Периодически сбрасывать полное состояние БД на в некоторую форму вспомогательной памяти



- Журнализация (Logging)
 - В журнале (log) сохраняется все операции транзакций, влияющие на состояние БД. Журнал сохраняется на диске так, что на него не влияют сбои, кроме сбоев диска.



Журнализация изменений

- различие в скорости работы с оперативной и с внешней памятью
 - **буферизация** страниц базы данных
 - измененные данные появляются во внешней (долговременной) памяти через некоторое время, а **не сразу** после внесения изменений
- для восстановления необходима некоторая дополнительная/избыточная информация
 - **журнал транзакций** - последовательность записей об изменении БД
 - содержит «старые» данные - уже измененные транзакциями
 - предназначен для выполнения операции отката при неуспешном выполнении транзакции или для восстановления данных после сбоя системы

Восстановление после прерывания транзакции

Катастрофические сбои

- Восстановить предыдущую копию БД на основе резервной
- Применить журнал транзакций к копии, чтобы реконструировать наиболее позднее целостное состояние, применяя операции завершенных транзакций до точки сбоя
- Нарастиваемый дамп + журнализация транзакций

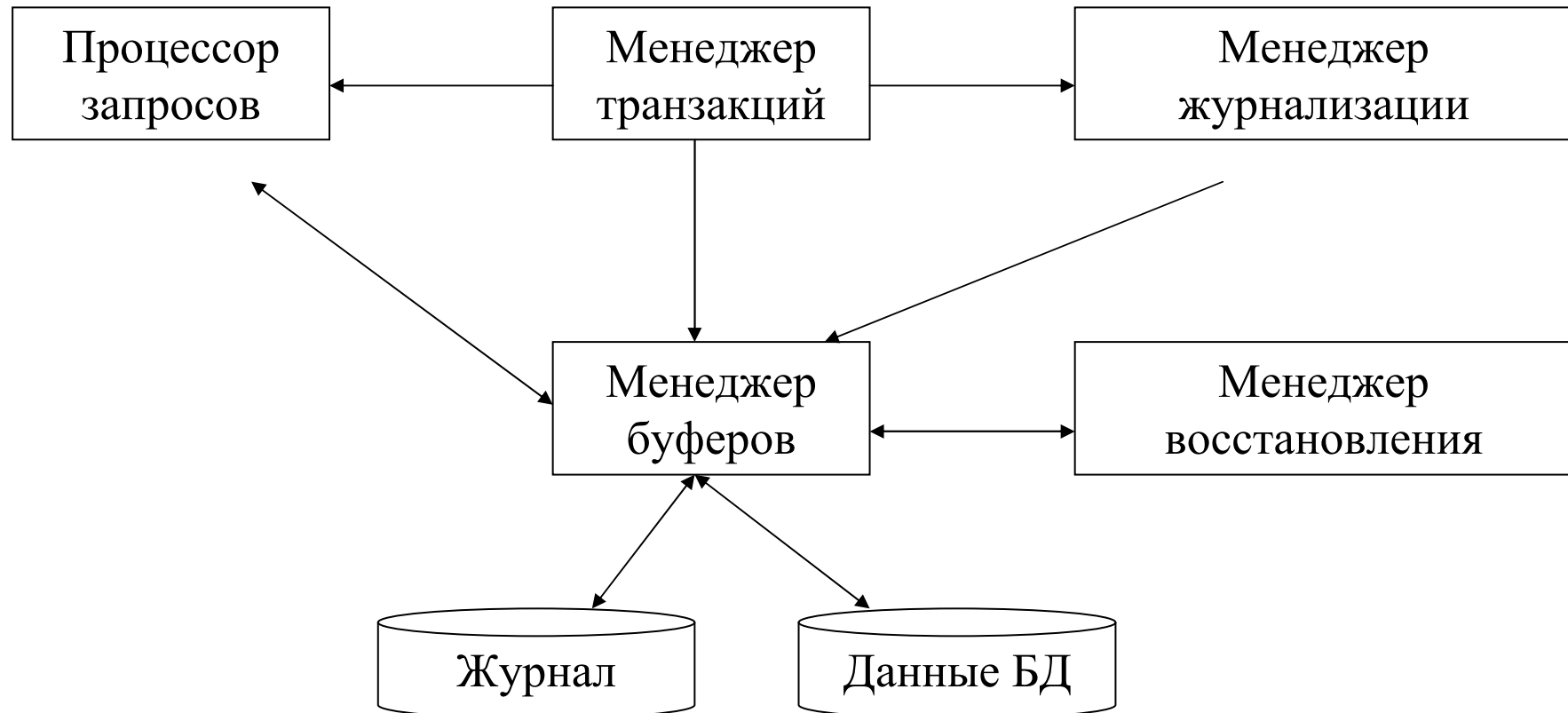
Некатастрофические сбои

- Откатить изменения, которые вызывают не согласованность данных, ликвидируя действие операций, возможно, повторяя легальные изменения, которые были потеряны
- Записи, сделанные в журнале, принимаются во внимание в ходе восстановления.
- Не нужно использовать полную резервную копию БД

Адресные пространства транзакций

- Дисковые блоки
 - элементы БД
- Оперативная память
 - управляемая менеджером буферов
- Локальное адресное пространство транзакции
 - программа – последовательность действий, хранимые процедуры, локальные, «глобальные» переменные

Взаимодействия менеджеров



- Структура взаимодействий менеджера транзакций с другими частями СУБД

Журнал транзакций

- локальные (транзакций) и общее (БД) журналы
- журнал так же буферизируются
 - при нормальной работе очередная страница выталкивается во внешнюю память при полном заполнении
- два вида буферов
 - буферы страниц БД
 - буферы журнала транзакций
- "грязные" (dirty) страницы БД
 - страницы БД, содержимое которых в буфере отличается от содержимого на диске
 - СУБД поддерживает список "грязных" страниц

Состояния транзакций

- Для целей восстановления необходимо фиксировать, когда транзакция начинается, завершается и фиксируется.
- **Begin_Transaction**: отмечает начало выполнения транзакции;
- **End_Transaction**: указывает, что операции чтения и записи завершены, и отмечает точку завершения выполнения транзакции (*из-за параллельной работы может быть произведен ее откат*);
- **Commit_Transaction**: сигнализирует об успешном завершении транзакции, о ее фиксации. *Все модификации, выполненные транзакцией, могут быть безопасно зафиксированы в БД и не будут ликвидированы*;
- **Rollback** (или **Abort**): сигнализирует о неуспешном завершении транзакции. *Все модификации, выполненные транзакцией, должны быть ликвидированы*;
- **Undo**: указывает, что действие некоторой операции транзакции должно быть ликвидировано; *аналогична **ROLLBACK**, но применяется к отдельной операции, а не к целой транзакции*;
- **Redo**: указывает, что некоторая операция транзакции должна быть повторена, *чтобы гарантировать успешное выполнение зафиксированных транзакций*

Записи журнала

Для каждой транзакции СУБД генерирует уникальный идентификатор (*transaction-id*).

- **[start_transaction, transaction-id]:** начало выполнения транзакции, представляемой *transaction-id*
- **[read_item, transaction-id, X]:** транзакции, идентифицируемая *transaction-id*, считывает значение элемента БД X. Опциональна в некоторых протоколах
- **[write_item, transaction-id, X, old_value, new_value]:** транзакция, представляемая *transaction-id*, изменяет значение элемента X с *old_value* на *new_value*
- **[commit, transaction-id]:** транзакция, представляемая *transaction-id*, успешно завершила операции, ее действия могут быть зафиксированы
- **[abort, transaction-id]:** транзакция, представляемая *transaction-id*, прервана

```
PROC Credit_labmark
(sno NUMBER, cno CHAR,
credit NUMBER)
old_mark  NUMBER;
new_mark  NUMBER;

SELECT labmark INTO
old_mark FROM enrol
WHERE studno = sno and
courseno = cno FOR
UPDATE OF labmark;

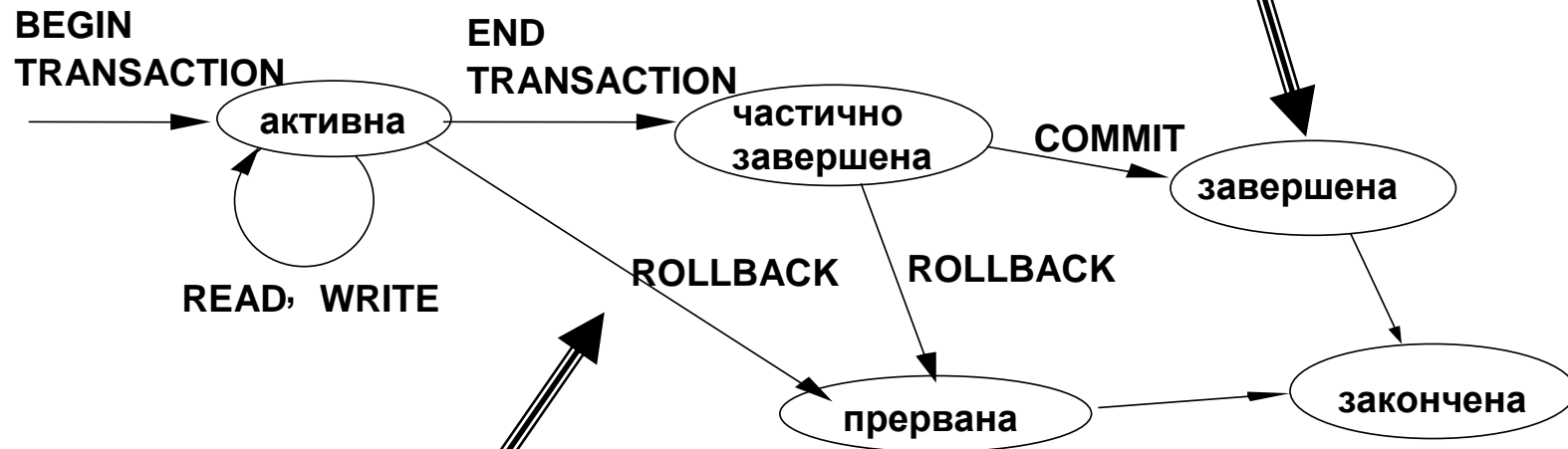
new_mark := old_mark
+ credit;

UPDATE enrol SET
labmark = new_mark
WHERE studno = sno and
courseno = cno ;

COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
END credit_labmark;
```

Выполнение транзакции

Транзакция достигает *точки фиксации*, когда все операции над БД завершены и записи о них занесены в журнал. Тогда в него записывается [commit, transaction-id].



Если происходит сбой, просматривается журнал и откатываются операции транзакций, занесенные журнал,

[start_transaction, transaction-id]

[write_item, transaction-id, X, old_value, new_value]

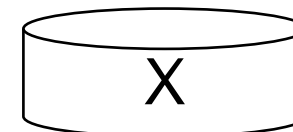
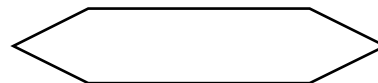
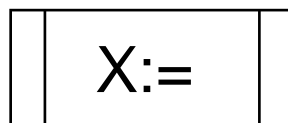
но для которых в журнал не записано: **[commit, transaction-id]**

Правила выталкивания буферов

- два требования:
 - *Максимальная скорость выполнения транзакций*
 - выталкивать страницы как можно реже
 - работа с данными *только в оперативной памяти*, и запись измененных страниц на диск *только в момент завершения* работы всей системы или нехватки буферов.
 - *Гарантия восстановления последнего согласованного состояния БД при сбое любого типа*
 - восстановление завершенных(зафиксированных) транзакций
 - откат незавершенных транзакций бесследно удалить
 - что-то необходимо выталкивать на диск
- две причины выталкивания страниц
 - недостаток оперативной памяти
 - возможность сбоев

Операции Read и Write транзакций

- операции read и write над элементами БД, выполняемыми как часть транзакции
- **read_item(X):**
 - Считывает элемент БД X в одноименную переменную X .
 1. Найти адрес блока диска, содержащего элемент X
 2. Скопировать этот блок диска в буфер в оперативной памяти
 3. Скопировать элемент X из буфера в переменную X
- **write_item(X):**
 - Записать значение переменную X в элемент БД X
 1. Найти адрес блока диска, содержащего элемент X
 2. Скопировать этот блок диска в буфер в оперативной памяти
 3. Скопировать элемент X из переменной X в его положение в буфере, сохранить измененный блок на диске (обновить БД на диске)



Политика выталкивания буферов журнала и страниц БД

- запись в журнале об изменении объекта БД должна попадать во внешнюю память раньше, чем измененный объект оказывается во внешней памяти
- Write Ahead Log (WAL) - "пиши сначала в журнал"
 - протокол журнализации (и управления буферизацией)
 - если требуется вытолкнуть во внешнюю память измененный объект БД, то перед этим нужно гарантировать выталкивание во внешнюю память журнала записи о его изменении
 - если во внешней памяти базы данных содержится объект, к которому применена некоторая команда модификации, то во внешней памяти журнала транзакций содержится запись об этой операции
 - обратное неверно

Вариации протокола WAL

Отложенные изменения:

- Никаких изменений БД, пока транзакция не достигнет точки фиксации

1. Модификации записываются в журнал
2. Транзакция завершена
3. Вытолкнуть журнал на диск
4. Модифицировать БД

При сбое!
REDO БД по записям журнала
UNDO не нужен, поскольку БД не менялась

Немедленные изменения:

- БД модифицируется операторами до завершения транзакции

1. Модификация X записывается в журнал
2. Модификация X в записывается БД (и соотв. записи журнала)
3. Модификация Y записывается в журнал
4. Завершение транзакции
5. Вытолкнуть журнал на диск
6. Модификация Y в БД

Сбой!
UNDO X

Сбой!
REDO Y

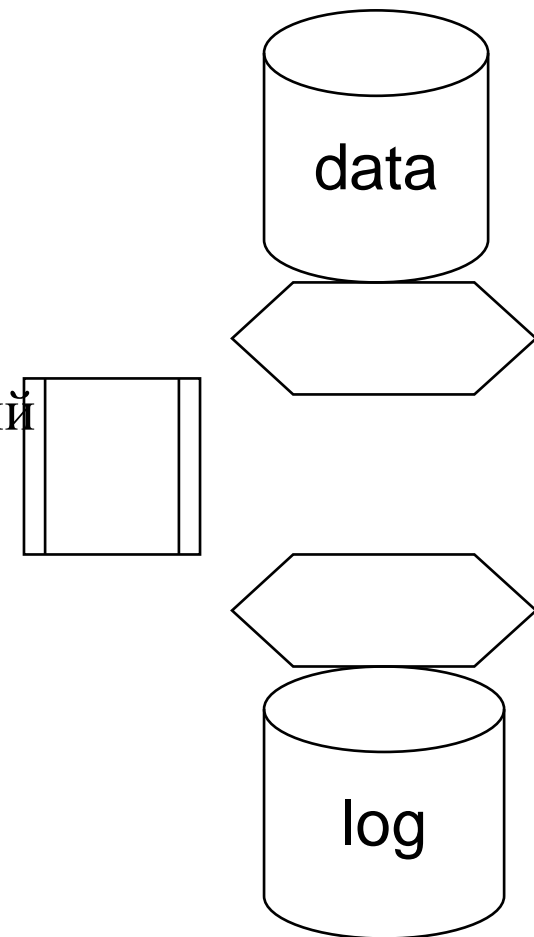
- Undo в обратном порядке
- Redo в прямом порядке
- Используем записи write_item

Другие условия на выталкивание буферов

- успешно завершившаяся транзакция должна быть реально зафиксирована во внешней памяти
 - восстановить состояние базы БД, содержащей результаты всех зафиксированных к моменту сбоя транзакций
 - при фиксации транзакции выталкивание буфера журнала, за которым следует массовое выталкивание буферов страниц БД (существенные накладные расходы)
- ограниченность объемов буферов БД и журнала транзакций
 - при наступлении определенного события
 - количество "грязных" или свободных страниц превысило определенный предел и т.п.
 - система проходит *контрольную точку*
 - выталкивание во внешнюю память содержимого буферов БД
 - в журнал вносится *запись контрольной точки* (список всех осуществляемых в данный момент транзакций)

Контрольные точки

- Запись [**checkpoint**] периодически записывается в журнал, когда СУБД записывает на диск результаты всех WRITE операций завершённых транзакций
- Все транзакции, чьи [**commit, transaction-id**] записи в журнале, не требуют redo их WRITE операций при сбое
- До прохождения контрольной точки журнал выталкивается на диск, до завершения транзакций
- Действия составляющие прохождение контрольной точки
 - временно приостановить выполнение транзакций
 - вытолкнуть "грязные" страницы на диск
 - занести [**checkpoint**] запись в журнал и вытолкнуть журнал на диск
 - восстановить выполнение транзакций



Характеристики журнала

- все записи в журнале от некоторой транзакции связываются в обратный список
- Начало списка
 - не закончившихся транзакций - запись о последнем изменении БД
 - закончившихся транзакций - запись о конце транзакции
 - обязательно вытолкнута во внешнюю память журнала
- Конец списка
 - всегда первая запись об изменении БД
- В каждой записи имеется
 - уникальный системный номер транзакции
 - чтобы восстановить прямой список записей об изменениях БД данной транзакцией

Минимальное требование восстановления

- *выталкивание при фиксации транзакции во внешнюю память журнала всех записей об изменении базы данных этой транзакцией*
- *последней записью в журнал, производимой от имени данной транзакции, является специальная запись о конце этой транзакции*

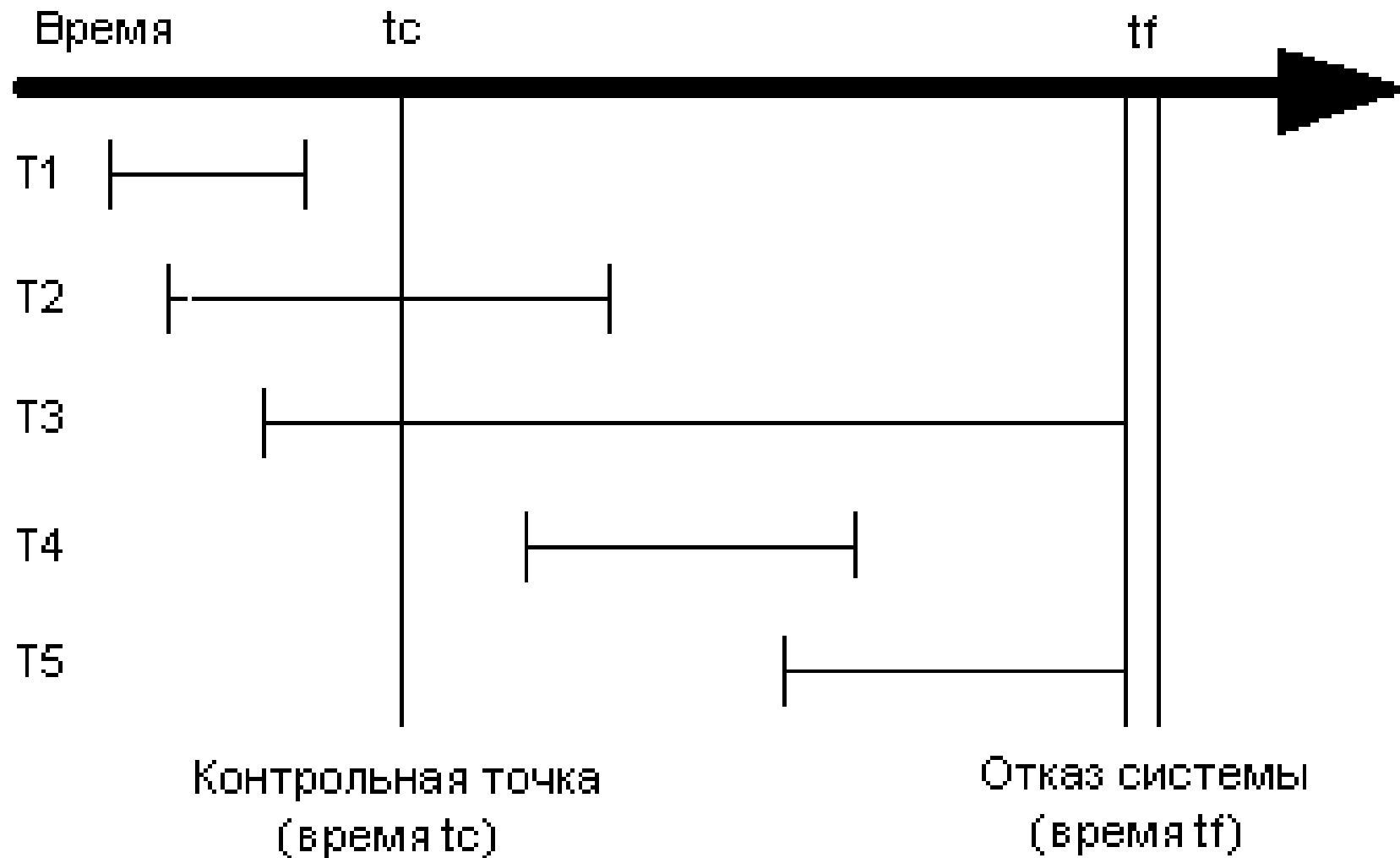
Индивидуальный откат транзакции

- В журнале транзакций просматривается список записей данной транзакции
 - от последнего изменения к первому изменению
- Выбирается очередная запись
- Выполняется противоположная по смыслу операция
 - INSERT – DELETE
 - DELETE – INSERT
 - UPDATE - обратная операция UPDATE
- Любая из этих обратных операций также журналируются
 - при индивидуальном откате может произойти сбой, потребуется откатить индивидуальный откат
- При успешном завершении отката в журнал заносится запись о конце транзакции.

Восстановление после мягкого сбоя

- не все "грязные" страницы БД были вытолкнуты во внешнюю память
- гарантированно были вытолкнуты "грязные" страницы в момент прохождения последней контрольной точки
- часть процедуры перезагрузки системы
- 5 вариантов состояния транзакций по отношению к моментам последней контрольной точки и сбоя
 - ***T1*** - транзакция успешно завершена до наступления контрольной точки.
 - ***T2*** - транзакция начата до наступления контрольной точки и успешно завершена после контрольной точки, но до наступления сбоя.
 - ***T3*** - транзакция начата до наступления контрольной точки и не завершена в результате сбоя.
 - ***T4*** - транзакция начата после наступления контрольной точки и успешно завершена до сбоя системы.
 - ***T5*** - транзакция начата после наступления контрольной точки и не завершена в результате сбоя.

Состояния транзакций



Действия при восстановлении - 1

- T1 и T5 никаких действий предпринимать не нужно
 - T1 - все данные сохранены во внешней памяти
 - T5 - никаких ее следов во внешней памяти ни журнала, ни БД нет
- T2 и T4 необходимо частично или полностью повторить
 - Записи журнала транзакций вытолкнуты во внешнюю память
 - T2 - измененные страницы БД частично вытолкнуты во внешнюю память. Повторить операции после контрольной точки.
 - T4 - изменения БД полностью отсутствуют во внешней памяти. Повторить целиком.

Действия при восстановлении –2

- ТЗ - частично откатить
 - до контрольной точки
 - во внешней памяти имеются измененные страницы БД, обновлены
 - все записи журнала до контрольной точки во внешней памяти
 - после контрольной точки.
 - следов изменений страниц БД нет
 - записи журнала отсутствуют во внешней памяти журнала

Восстановление после жесткого сбоя

- база данных на диске испорчена физически
- Восстановление по журналу транзакций и резервной копии БД
- Резервная копия БД
 - создается периодически
 - с учетом скорости наполнения журнала транзакций
- Восстановление
 - копирование резервной копии БД
 - просмотр журнала для выявления всех успешно завершенных до сбоя транзакций.
 - по журналу транзакций в прямом направлении повторяются все успешно законченные транзакции
 - нет необходимости отката транзакций, прерванных в результате сбоя - отсутствуют в резервной копии

Bce

