

Принципы построения и работы баз данных

Тема 08: Восстановление при сбоях

Часть 2

- Восстановление при сбоях Гл.17
- Управление параллельным доступом Гл.18
- Обработка транзакций Гл.19

Целостность и корректность данных

- Хотели бы вы, чтобы данные были правильными постоянно

EMP

Name	Age
White	52
Green	342
Gray	1
	1

Ограничения целостности

- Условия, которым должны удовлетворять данные
- Примеры:
 - x является ключом отношения R
 - выполняется функциональная зависимость $x \rightarrow y$ для всех кортежей из R
 - $\text{Domain}(x) = \{\text{Red}, \text{Blue}, \text{Green}\}$
 - никто из служащих не должен иметь зарплату, превышающую среднюю более чем в 2 раза

Определения:

- Согласованное состояние: состояние, в котором выполняются все ограничения целостности
- Согласованная БД: База данных в согласованном состоянии

Ограничения целостности (как мы их знаем)
могут не обеспечивать «полную корректность»

Пример 1 Условия на транзакции

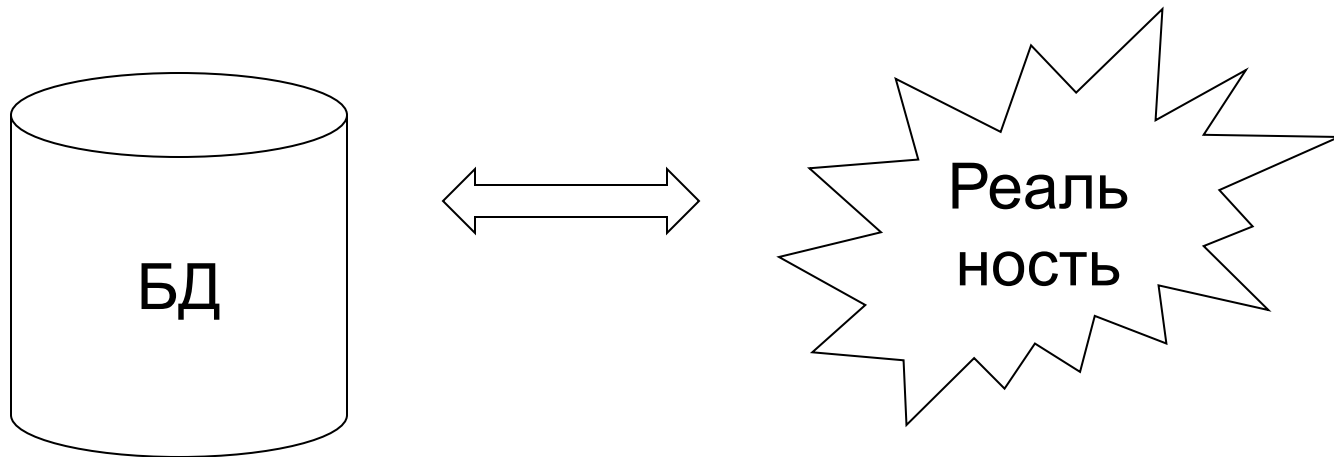
- Когда зарплата обновляется,
 новая зарплата > старая зарплата
- Когда удаляется запись счета в банке,
 остаток = 0

можно имитировать простым ограничением

Acct #		balanc e	delete d?
--------	------	-------------	--------------

Ограничения целостности (как мы их знаем)
могут не обеспечивать «полную корректность»

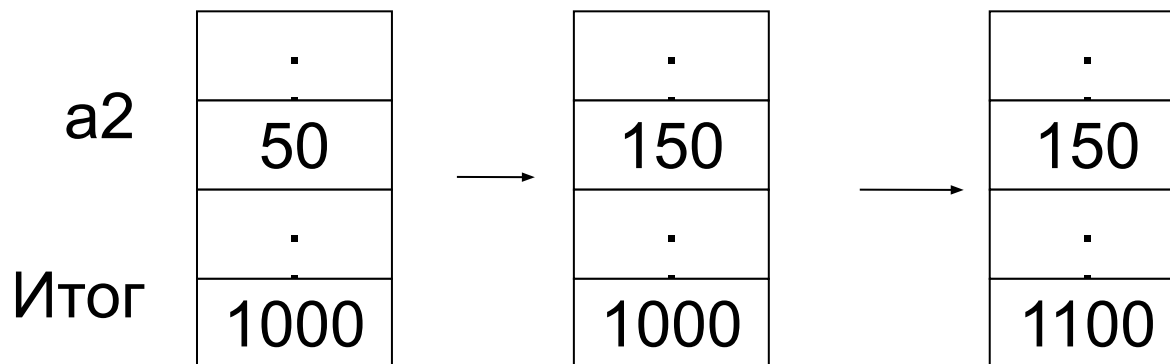
Пример 2 База данных должна
отражать реальный мир



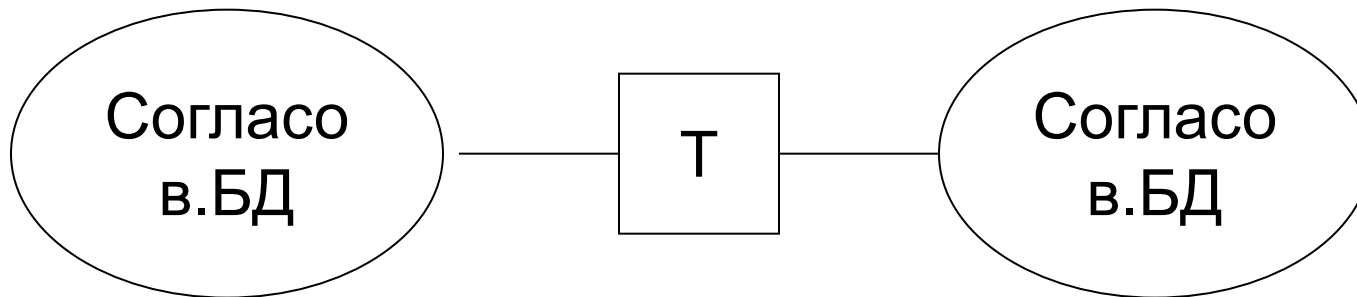
Наблюдение: БД не может быть согласована постоянно из-за необходимости изменений!

Пример: $a_1 + a_2 + \dots + a_n = \text{Итог}$ (ограничение)

Добавить \$100 к a_2 : $\left\{ \begin{array}{l} a_2 \leftarrow a_2 + 100 \\ \text{Итог} \leftarrow \text{Итог} + 100 \end{array} \right.$



Транзакция: совокупность действий, сохраняющая согласованность БД



Важное предположение

Если Т начинается в согласованном состоянии и выполняется изолированно (т.е. без конкуренции с другими действиями над БД), то после ее завершения БД остается в согласованном состоянии

Как могут быть нарушены ограничения целостности?

- Ошибка в алгоритме транзакции
- Ошибка в СУБД
- Сбой оборудования, например, сбой диска, в результате которого изменяется остаток на счете
- Общие данные

T1: увеличить на 10% зарплату программистам из списка

T2: изменить программистов из списка на системных аналитиков

Как можно предотвратить/исправить нарушения?

- Глава 17: только при сбоях системы
- Chapter 18: только из-за общих данных
- Chapter 19: при сбоях системы и общих данных

Не будем рассматривать:

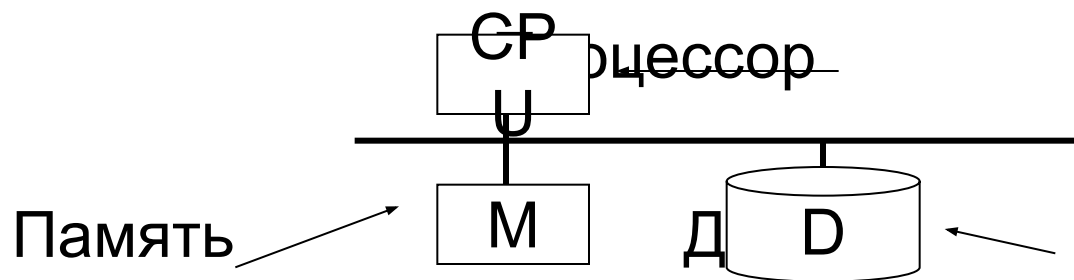
- Как писать правильные транзакции
- Как писать правильные СУБД
- Проверку и восстановление ограничений

Рассматриваемые здесь решения не знают об ограничениях

Восстановление при сбоях

- Первое, что нужно сделать – описать модель сбоя

События — ~~Желательные~~
~~Нежелательные~~ ~~Ожидаемые~~
 ~~Неожидаемые~~



Желательные события: то, что программа должна делать в соответствии с ее руководством.

Нежелательные ожидаемые события

- зависание или сбой системы,
- потеря памяти,
- остановка процессора, сброс

Нежелательные неожиданные события

- все остальное (необъяснимая потеря данных, землетрясение, нападение террористов)

Является ли эта модель приемлемой?

Подход: Добавить низко уровневые проверки + избыточность для увеличения вероятности, что система выживет при сбое

Например,

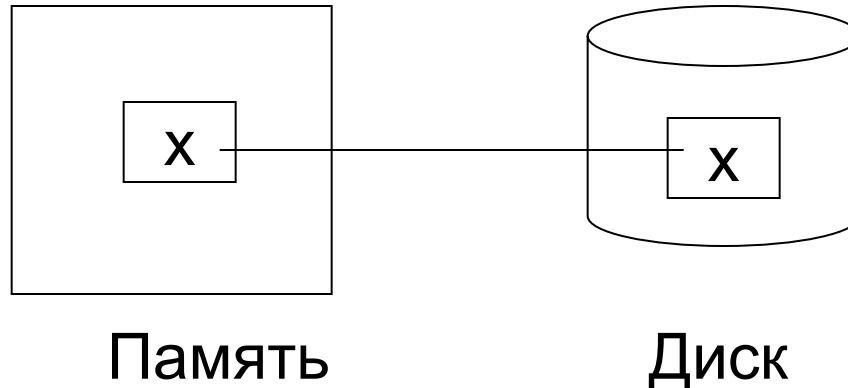
— поддержка точной копии диска или дискового массива

— Проверка четности при обращении к памяти

— Проверки процессора

Второй важный момент после выбора модели:

Иерархия памяти



Операции

- Input (x): блок с элементом БД $x \rightarrow$ память
- Output (x): блок с элементом БД $x \rightarrow$ диск
- Read (x,t): если нужно выполнить input(x), копировать в переменную $t \leftarrow$ значение x из блока
- Write (x,t): если нужно выполнить input(x), значение x в блоке $\leftarrow t$

Ключевая проблема Незавершенная транзакция

Пример

Ограничение: $A=B$

T1: $A \leftarrow A \times 2$

$B \leftarrow B \times 2$

T1: Read (A,t); $t \leftarrow t \times 2$

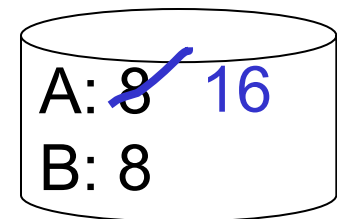
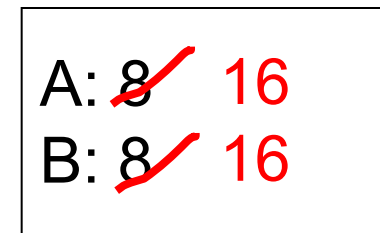
Write (A,t);

Read (B,t); $t \leftarrow t \times 2$

Write (B,t);

Output (A);

Output (B); - сбой
Диск



Необходима атомарность (неделимость) транзакции – выполнение либо всех действий транзакции либо никаких.

Файл протокола (журнал регистрации)

Протокол – последовательность записей о том что происходит с каждой транзакцией.

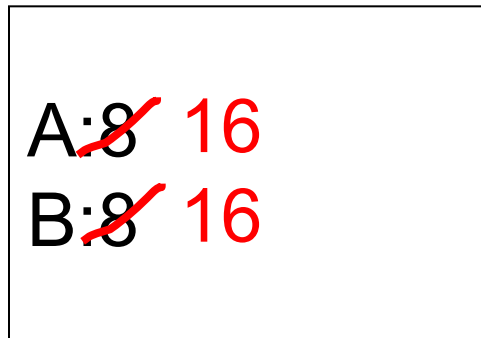
Эти последовательности для разных транзакций могут перекрываться.

При сбое системы происходит просмотр протокола, незавершенные по протоколу транзакции должны быть либо повторены, либо полностью отменены (каждое действие, отмеченное в протоколе).

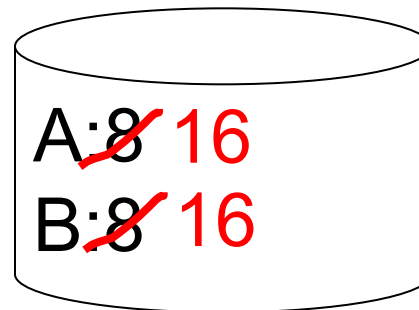
Типы записей: START, COMMIT, ABORT

Одно из решений - протокол-возврата (немедленная модификация)

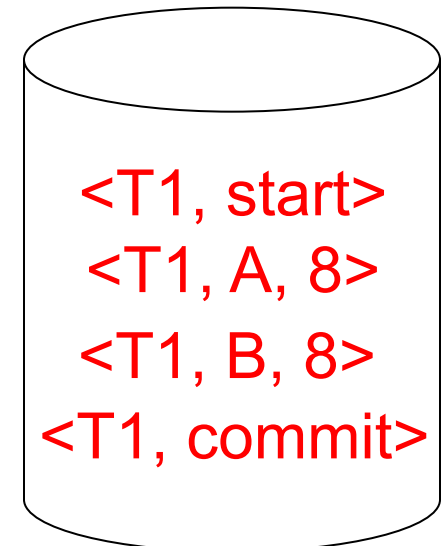
T1:Read (A,t); $t \leftarrow t \times 2$ A=B
Write (A,t);
Read (B,t); $t \leftarrow t \times 2$
Write (B,t);
Output (A);
Output (B);



память



ДИСК

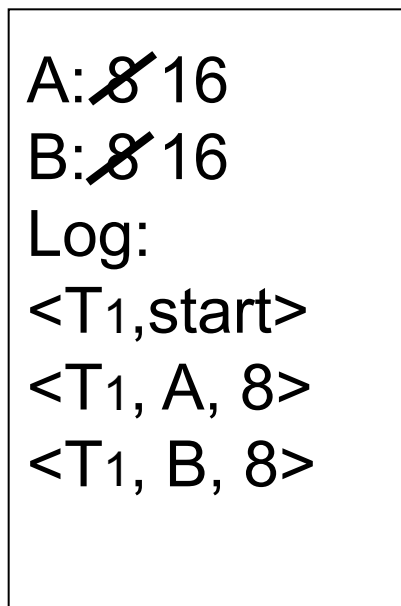


протокол

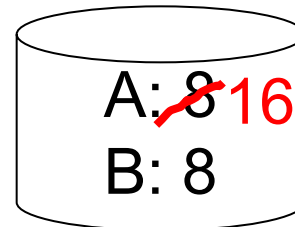
Одно “осложнение”

- Протокол сначала заносится в память, а потом уже на диск (для эффективной буферизации)
- Не записывается на диск при каждом действии

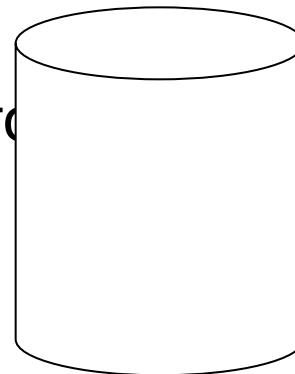
память



БД



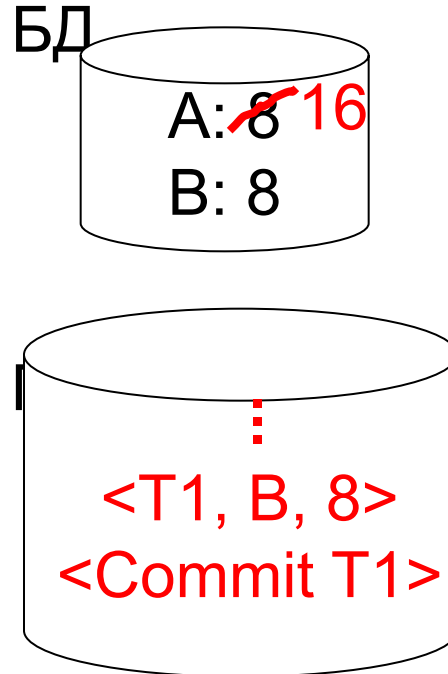
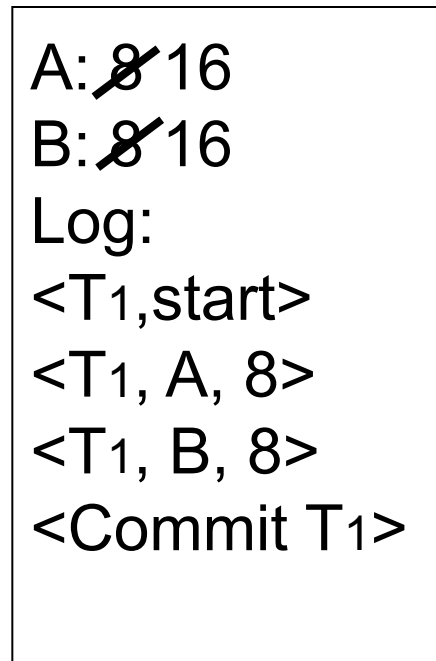
Прото



Плохое
сост.# 1

- Протокол сначала заносится в память, а потом уже на диск (для эффективной буферизации)
- Не записывается на диск при каждом действии

память



Плохое
сост.# 2

Правила протокола-возврата (Undo logging rules)

- (1) Для каждого действия создать запись в протоколе-возврата (содержащую старое значение)
- (2) Перед изменением x на диске, запись, содержащая старое значение x должна быть уже записана в файл протокола-возврата
- (3) Перед записью `<Commit T>` в файл протокола все операции записи действий транзакции на диск должны быть уже завершены

Правила восстановления: протокол-возврата

- Для каждой транзакции T_i , для которой в протоколе имеется запись $\langle \text{Start } T_i \rangle$:
 - Если для нее в протоколе имеется также запись $\langle \text{Commit}, T_i \rangle$ или $\langle \text{Abort}, T_i \rangle$, то ничего не делать
 - Иначе для всех записей $\langle T_i, X, v \rangle$ в протоколе: write (X, v)
output (X)
записать $\langle \text{Abort } T_i \rangle$ в протокол

Правила восстановления: протокол-возврата

- (1) Пусть S = множество транзакций с записями $\langle \text{Start } T_i \rangle$ в протоколе, но без записей $\langle \text{Commit } T_i \rangle$ (или $\langle \text{Abort } T_i \rangle$)
- (2) Для каждой записи $\langle T_i, X, v \rangle$ в протоколе в обратном порядке (более поздние обрабатываются раньше) выполнить:
 - если $T_i \in S$ то $\left\{ \begin{array}{l} \text{- write } (X, v) \\ \text{- output } (X) \end{array} \right.$
- (3) Для каждой транзакции $T_i \in S$ записать $\langle \text{Abort } T_i \rangle$ в протокол

Что если произойдет сбой во время восстановления?

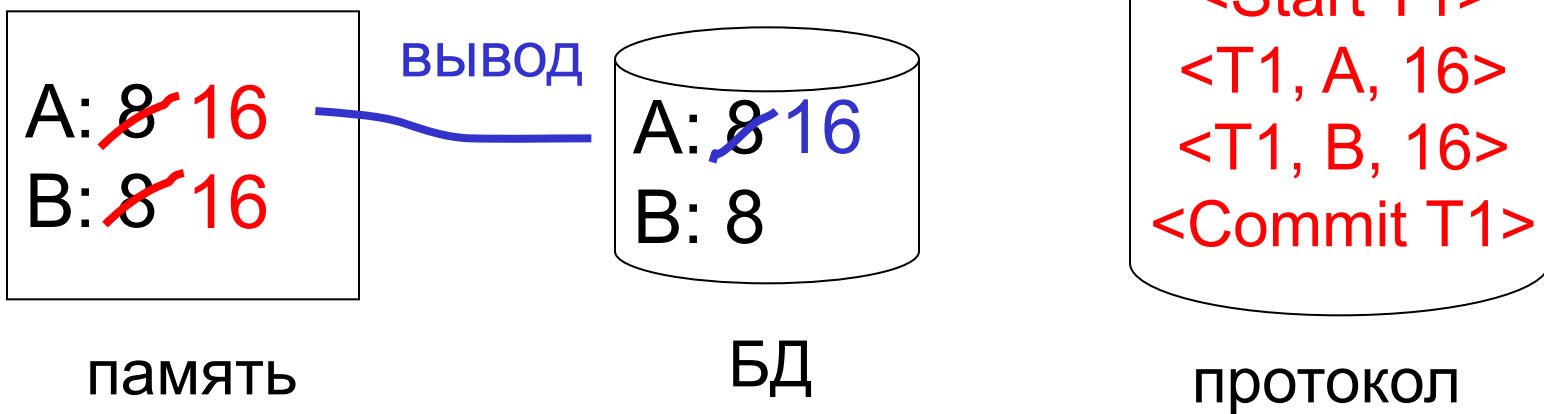
Нет проблем! Восстановления с использованием протокола возврата – идемпотентно (т.е. $X * X = X$)

Дальнейшее обсуждение:

- Протокол-восстановления(повтора)
- Протокол-возврата-повтора, почему оба?
- Действия в реальности
- Контрольные точки
- Сбои физических носителей информации

Протокол-повтора (отложенная модификация)

T₁: Read(A,t); $t \leftarrow t \times 2$; write (A,t);
Read(B,t); $t \leftarrow t \times 2$; write (B,t);
Output(A); Output(B)



Правила протокола-повтора (Redo logging rules)

- (1) Для каждого действия создать запись в протоколе-повтора (содержащую новое значение)
- (2) Перед модификацией X на диске (БД), все записи транзакции, модифицирующие X (включая commit) должны быть записаны на диск
- (3) Очистить буфер протокола (записать его содержимое на диск) при завершении (commit) транзакции

Правила восстановления: протокол-повтора

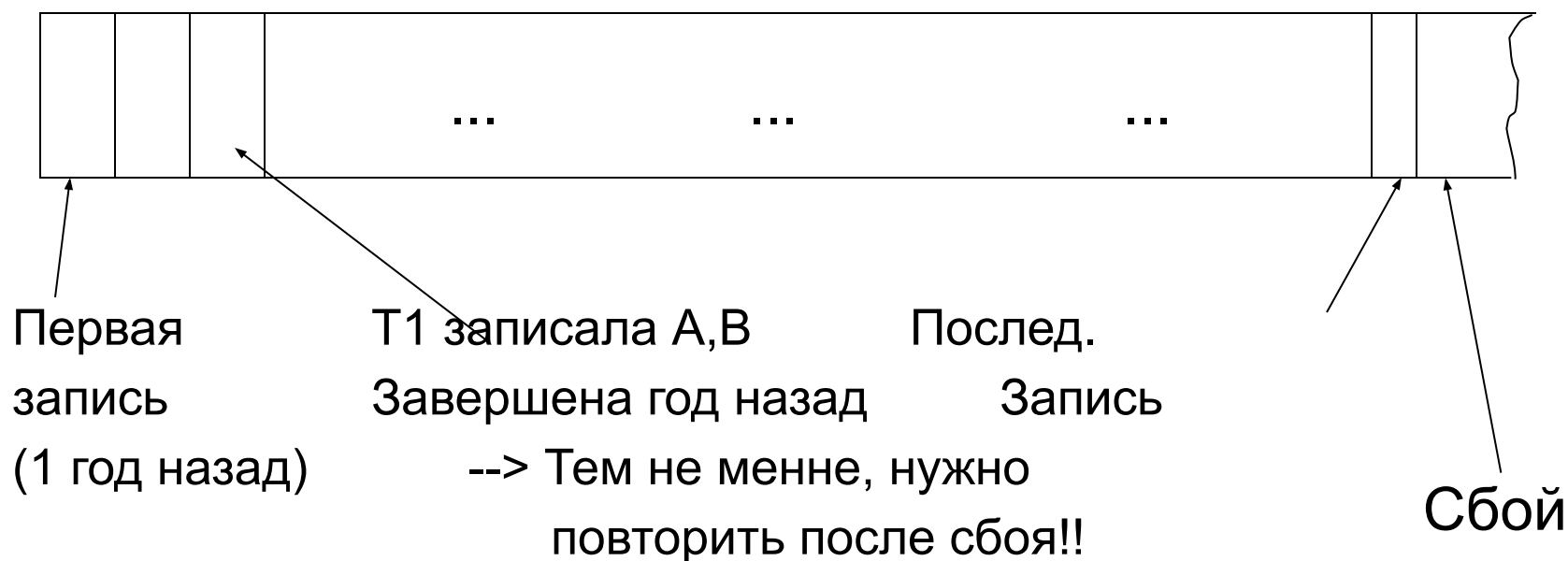
- Для каждой транзакции T_i , для которой в протоколе имеется запись $\langle \text{Commit } Y_i \rangle$:
 - Для всех записей этой транзакции $\langle T_i, X, v \rangle$:
 - Write(X, v)
 - Output(X)

Правила восстановления: протокол-повтора

- (1) Пусть S = множество транзакций, для которых в протоколе-повтора имеется запись $\langle \text{Commit } T_i \rangle$
- (2) Для каждой записи $\langle T_i, X, v \rangle$ в порядке от начала к концу протокола выполнить:
 - если $T_i \in S$ то $\left\{ \begin{array}{l} \text{Write}(X, v) \\ \text{Output}(X) \end{array} \right.$
- (3) Для каждой незавершенной транзакции ($T_i \notin S$) записать $\langle \text{Abort } T_i \rangle$ в протокол

Восстановление очень медленное!

Протокол-повтора:



Решение: Использование контрольных точек (простая версия)

Периодически, создавать контрольную точку:

- (1) Прервать прием новых транзакций
- (2) Подождать завершения всех текущих транзакций
- (3) Очистить буферы протокола (записать на диск)
- (4) Записать буферы БД на диск (не очищая буферы)
- (5) Записать информацию о контрольной точке (“checkpoint”) в протокол и на диск
- (6) Возобновить обработку транзакций

Пример: Что делать при восстановлении?

Протокол-повтора (на диске):

⋮	<T1,A,16>	⋮	<Commit T1>	⋮	Checkpoint	⋮	<T2,B,17>	⋮	<Commit T2>	⋮	<T3,C,21>	Сбой
---	-----------	---	-------------	---	------------	---	-----------	---	-------------	---	-----------	------

Основные недостатки:

- *Протокол-возврата:* не может восстановить копию текущего состояния БД
- *Протокол-повтора:* необходимо держать в памяти все модифицированные блоки до завершения транзакции

Решение: протокол-возврата-повтора!

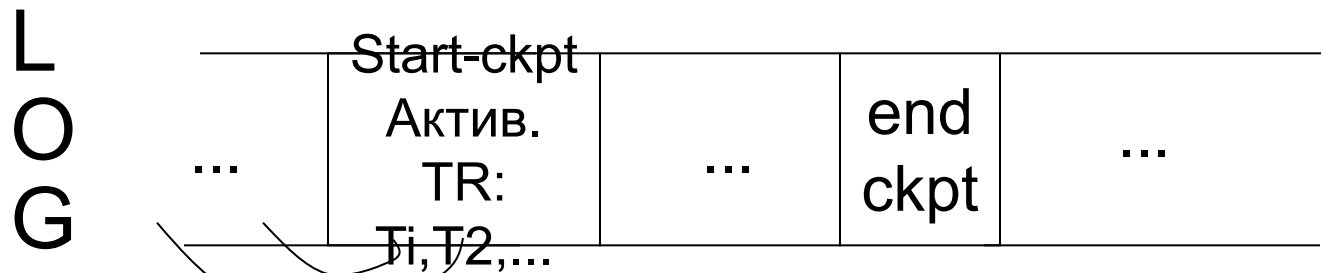
Запись протокола $\Rightarrow$

$\langle T, X, \text{новое значение } X, \text{старое значение } X \rangle$

Правила

Перед изменением на диске элемента X БД,
вызываемого транзакцией T, необходимо записать
на диск соответствующую запись протокола
<T, X, новое значение X, старое значение X >

«Не останавливающие» контрольные точки

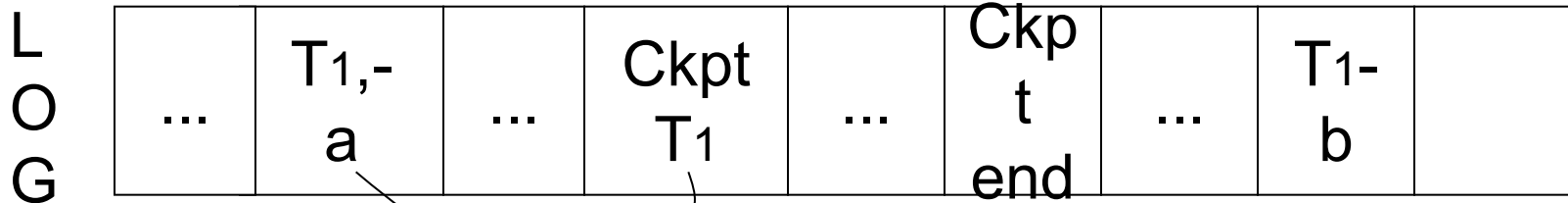


для
возврата

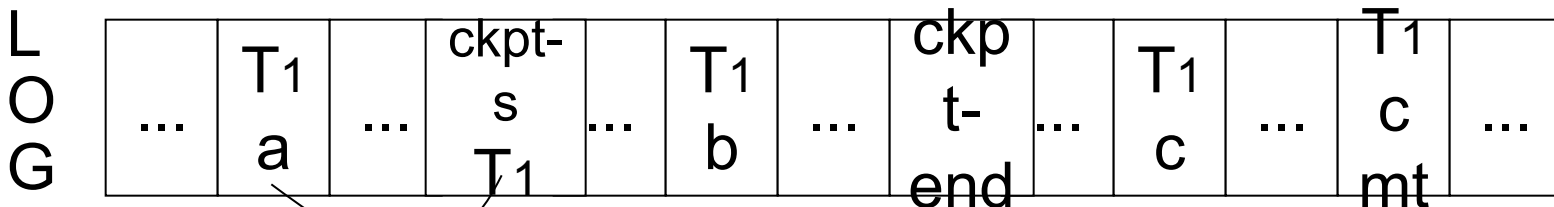
измененные
буферы памяти
записываются на диск

Примеры: что делать при восстановлении?

нет T1 commit



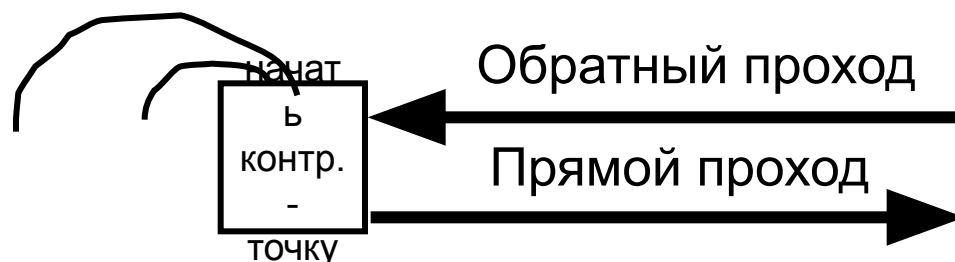
☐ отменить T1 (возвратить a,b)



☐ Повторить T1: (повторить b,c)

Процесс восстановления:

- Обратный проход (с конца протокола до места начала последней контрольной точки)
 - Построить множество S завершенных транзакций
 - Отменить действия транзакций, не вошедших в S
- Отменить активные транзакции контр.точки
 - Пройти по цепочке возврата для транзакций из (список активных транзакций контр.точки) - S
- Прямой проход (от начала последней контрольной точки до конца протокола)
 - Повторить действия транзакций из S



Действия в реальном мире

Например, выдача денег в банкомате

$$T_i = a_1 a_2 \dots a_j \dots a_n$$

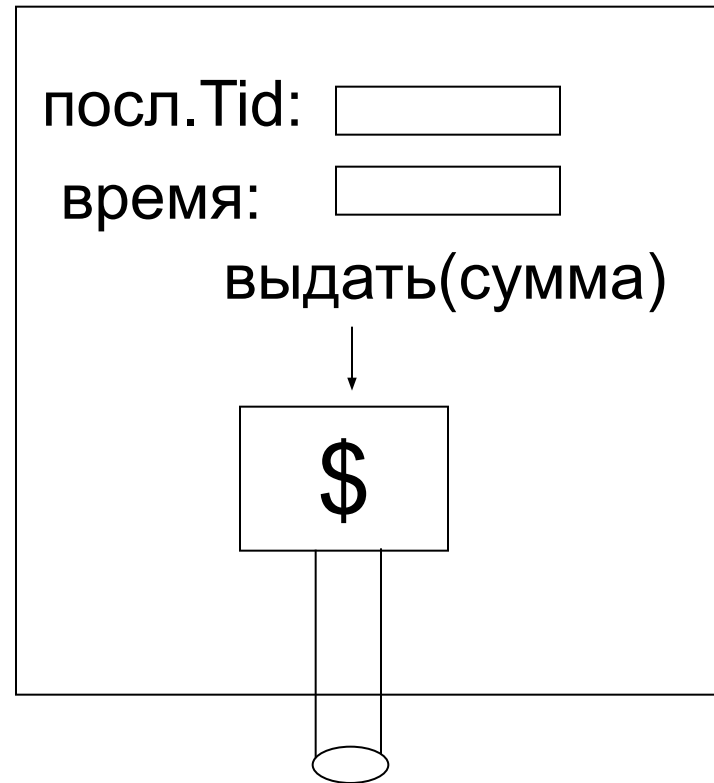
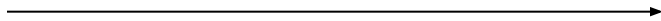


Решение

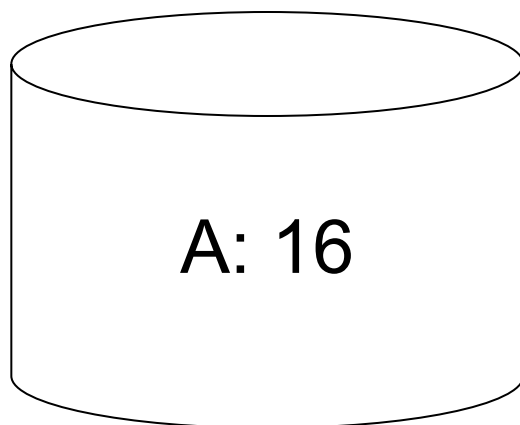
- (1) Выполнять реальные действия после завершения транзакции
- (2) Желательно, чтобы транзакция была идемпотентной

банкомат

Выдать \$\$
(сумма, Tid, время)



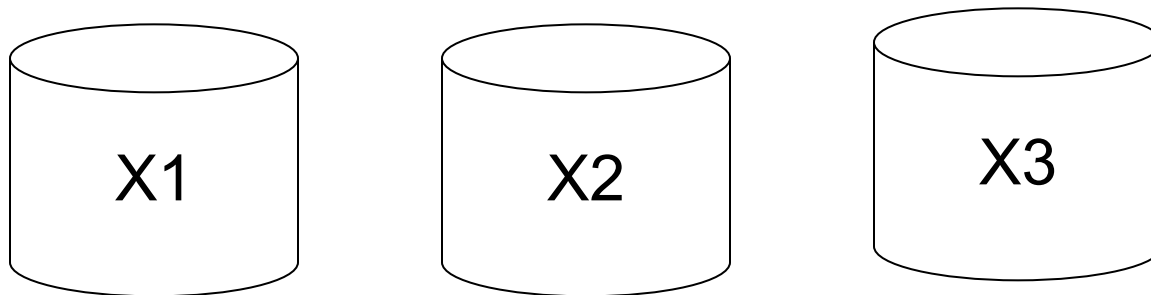
Сбои внешних носителей



Решение: Иметь несколько копий данных

Пример 1 Тройная избыточность

- Поддерживать 3 копии на различных дисках
- Output(X) --> 3 операции вывода
- Input(X) --> 3 операции ввода + голосование



Пример #2 Избыточная запись, единственное чтение

- Поддерживать N копий на различных дисках
 - $\text{Output}(X) \rightarrow N$ операций вывода
 - $\text{Input}(X) \rightarrow$ чтение 1 копии -
если шибок нет - принять
- иначе читать копию
- ☐ Предполагается, что можно определить, являются ли прочитанные данные ошибочными

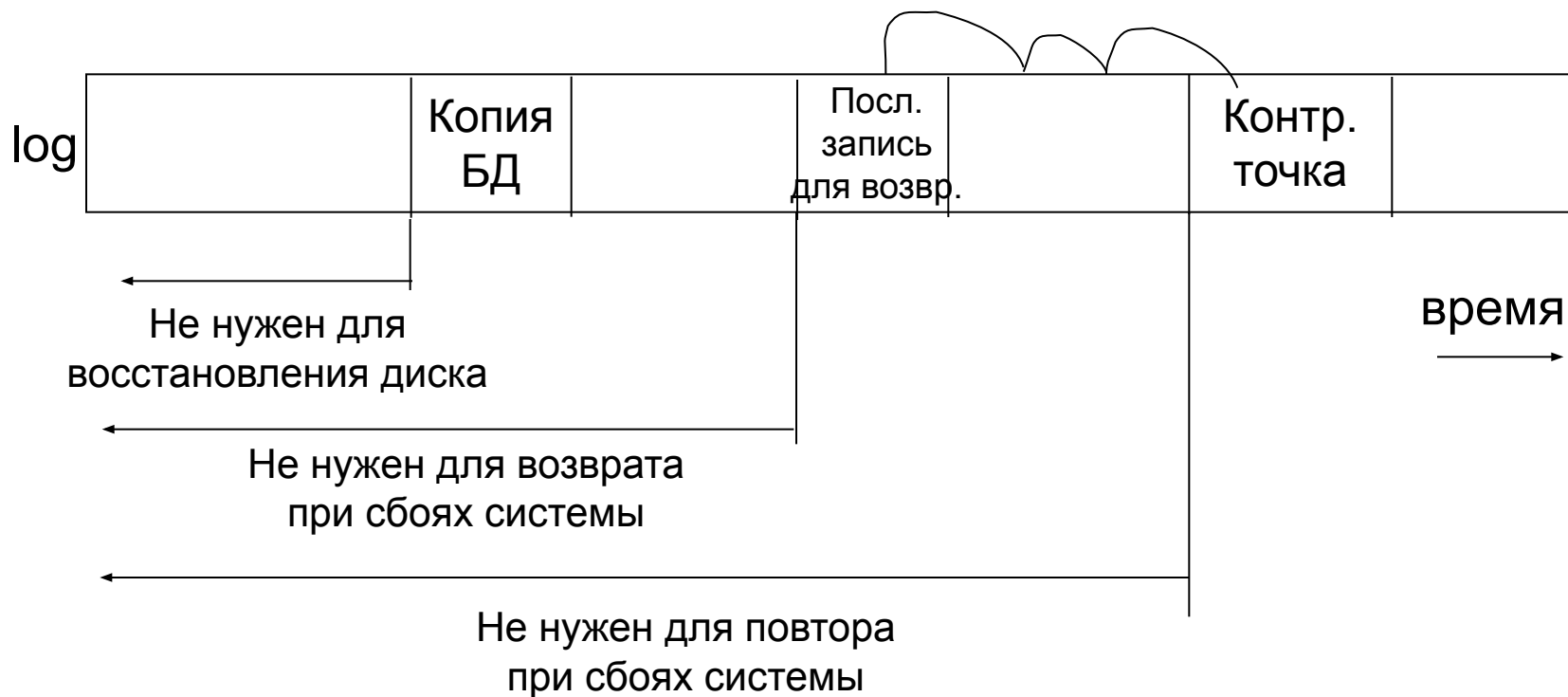
Пример #3:

Резервная копия БД + протокол



- если текущая БД потеряна,
 - восстановить резервную копию
 - восстановить текущее состояние, используя протокол

Когда протокол (или его часть) станет ненужным?



Итог

- Согласованность данных
- Один источник проблем - сбои:
 - Использование протоколов
 - Избыточность
- Другой источник проблем – совместное использование данных:
 - > следующая тема