Лекция 7

Средства журнализации и восстановления баз данных

План лекции

- Введение

- Индивидуальный откат транзакции
- Восстановление после мягкого сбоя
- Восстановление после жесткого сбоя

Введение

- Если посмотреть с практической точки зрения, то одним из основных требований к СУБД является надежность хранения баз данных.
- Требование надёжного хранения предполагает, в частности, возможность восстановления согласованного состояния базы данных после любого рода аппаратных и программных сбоев.
- Очевидно, что для выполнения восстановлений необходима некоторая дополнительная информация. В подавляющем большинстве современных реляционных СУБД такая избыточная дополнительная информация поддерживается в виде журнала изменений базы данных.

Введение

Общей целью журнализации изменений баз данных является обеспечение возможности восстановления согласованного состояния базы данных после любого сбоя. Общими принципами восстановления являются следующие:

- Результаты зафиксированных транзакций должны быть сохранены в восстановленном состоянии базы данных, т.е. должно поддерживаться свойство долговечности (durability) транзакций.
- Результаты незафиксированных транзакций должны отсутствовать в восстановленном состоянии базы данных, в противном случае состояние базы данных могло бы оказаться не целостным.

Возможны следующие ситуации, при которых требуется производить восстановление состояния базы данных:

- Индивидуальный откат транзакции. Для восстановления согласованного состояния БД при индивидуальном откате транзакции нужно устранить последствия операторов модификации базы данных, которые выполнялись в этой транзакции.
- Восстановление после внезапной потери содержимого оперативной памяти (мягкий сбой). Такая ситуация может возникнуть при аварийном выключении электрического питания, при возникновении неустранимого сбоя процессора и т.д. Ситуация характеризуется потерей той части базы данных, которая к моменту сбоя содержалась в буферах оперативной памяти СУБД.
- Восстановление после поломки основного внешнего носителя базы данных (жёсткий сбой). Основой восстановления является архивная копия и журнал изменений базы данных.

Журнал

Есть несколько способов ведения журнала:

- Поддержка отдельного локального журнала изменений базы данных для каждой транзакции. Эти локальные журналы используются для индивидуальных откатов транзакций и могут поддерживаться в оперативной памяти СУБД. Кроме того, поддерживается общий журнал изменений базы данных, используемый для восстановления состояния базы данных после мягких и жёстких сбоев.
- Поддержка только общего журнала изменений базы данных, который используется и для индивидуальных откатов, и для восстановления после мягкого и жёстких сбоев.

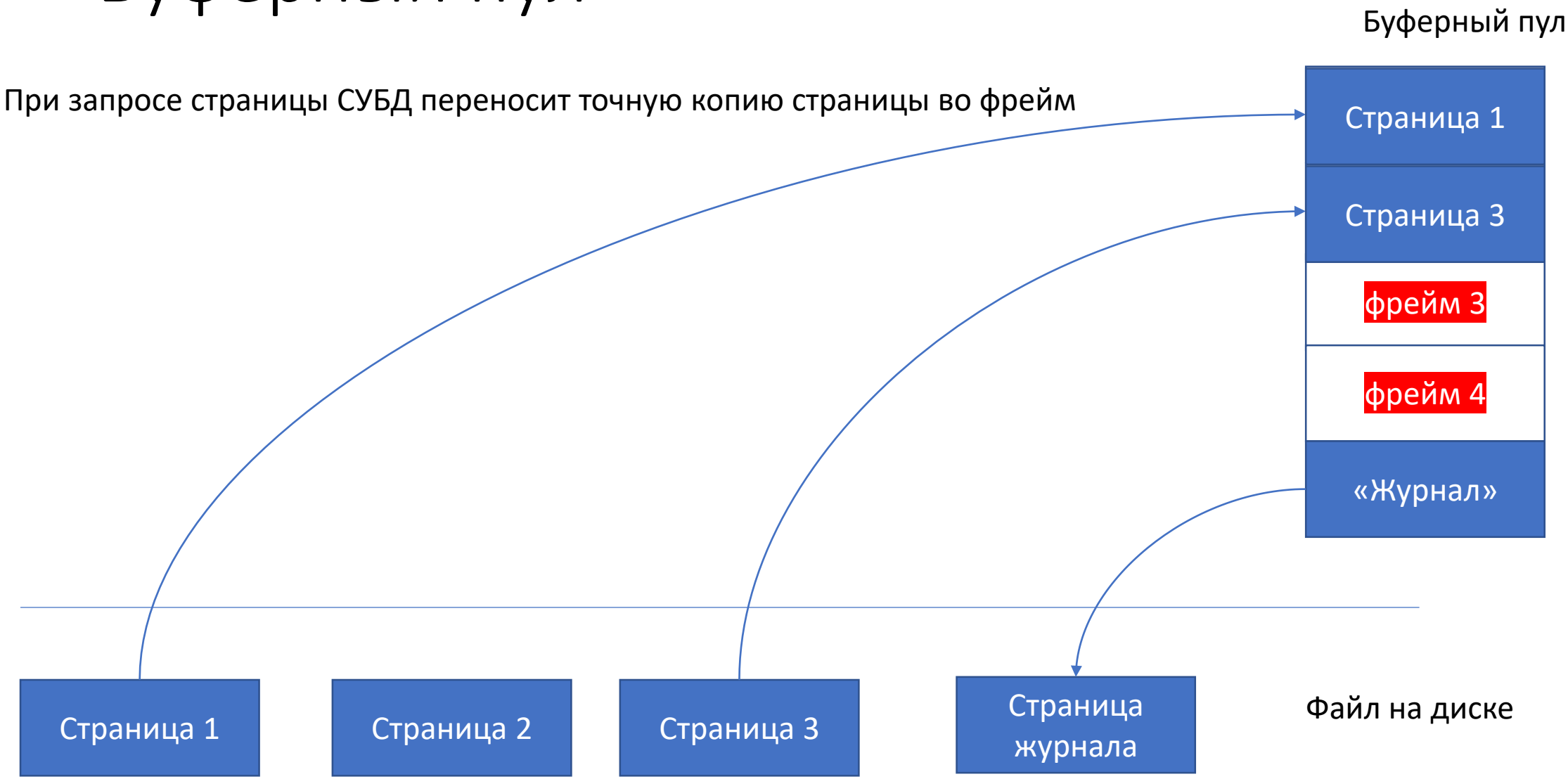
Управление буферным пулом базы данных

Если при выполнении некоторой операции в некоторой транзакции требуется доступ к некоторому блоку базы данных, и копия этого блока отсутствует в буферном пуле, СУБД должна:

- выделить какую-либо страницу буферного пула
- считать в нее с диска требуемый блок базы данных
- предоставить доступ к этой странице запросившей операции

В буферном пуле может не оказаться свободных страниц, и тогда СУБД в соответствии с некоторым критерием находит некоторую занятую страницу и освобождает её.

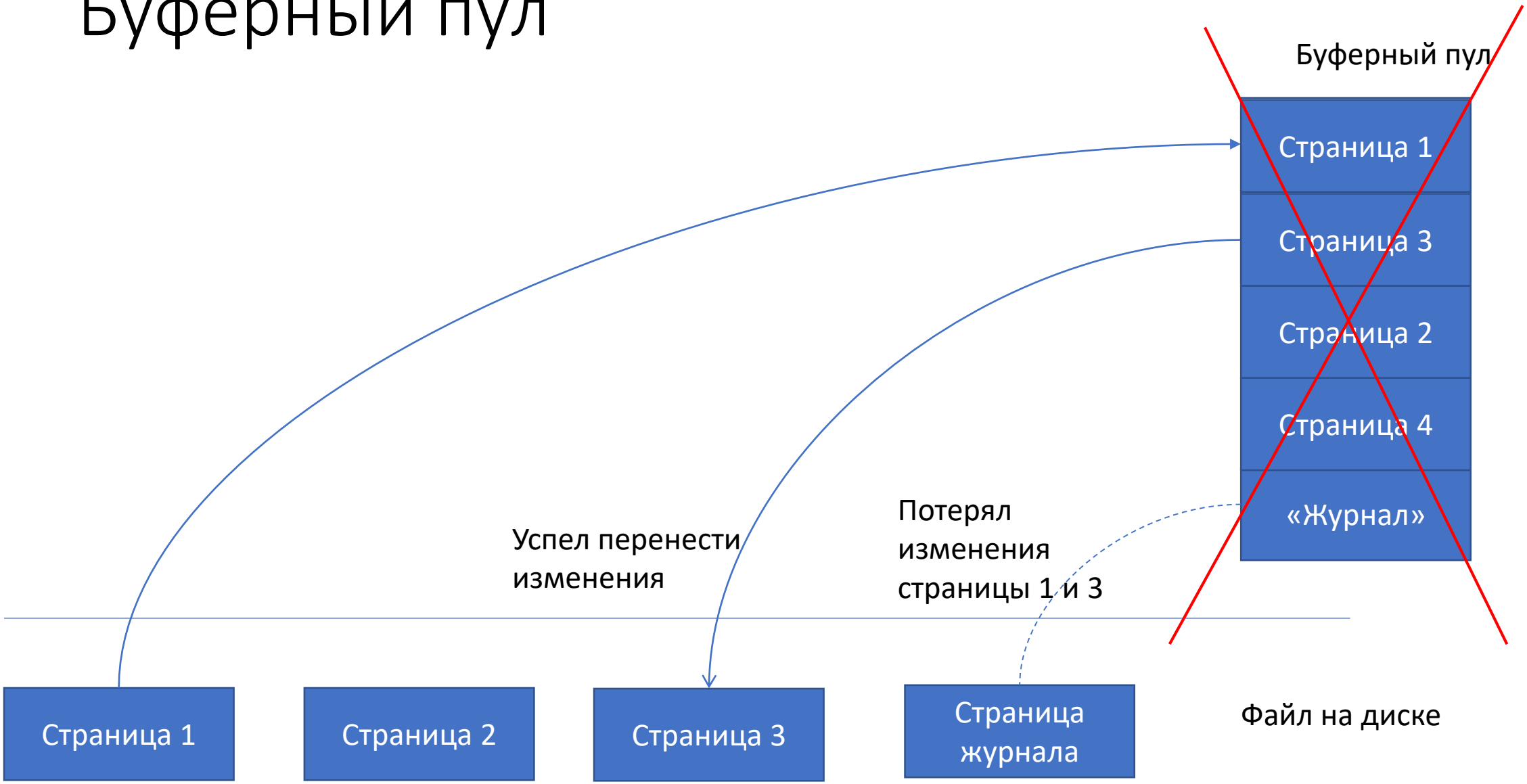
Буферный пул



Буферизация журнала

Используются два буфера размером в блок внешней памяти. Во время выполнения операций при журнализации изменений заполняется один буфер, и когда он полон, запускается обмен с внешней памятью, и начинается заполняться второй буфер.

Буферный пул



Протокол упреждающей записи в журнал

Основным принципом согласованной политики выталкивания буфера журнала и буферных страниц базы данных является то, что запись об изменении объекта базы данных должна оказаться во внешней памяти журнала раньше, чем изменённый объект окажется во внешней памяти базы данных.

Соответствующий протокол журнализации называется WAL (Write Ahead Log) и состоит в том, что если требуется вытолкнуть во внешнюю память буферную страницу, содержащую изменённый объект базы данных, то перед этим нужно гарантировать выталкивание во внешнюю память журнала буферной страницы журнала, содержащей запись об изменении этого объекта.

Индивидуальный откат транзакции

- Для обеспечения возможности индивидуального отката транзакции в журнале все записи от данной транзакции связываются в обратный список
- В начале списка находится хронологически последняя запись - это запись о последнем изменении базы данных произведённом данной транзакцией.
- В случае индивидуального отката транзакции, хронологически последние записи могут быть еще не вытолкнуты во внешнюю память журнала и находиться в буфере оперативной памяти.
- Для закончившихся транзакций, индивидуальные откаты уже невозможны.

Индивидуальный откат транзакции

Выполняется следующим образом:

- Выбирается очередная журнальная запись из списка записей данной транзакции.
- Выполняется противоположная по смыслу операция:
  - вместо операции INSERT выполняется соответствующая операция DELETE
  - вместо операции DELETE выполняется INSERT
  - вместо прямой операции UPDATE – обратная операция UPDATE, восстанавливающая предыдущее состояние объекта базы данных

Эти обратные операции называются undo, и они тоже журналируются.

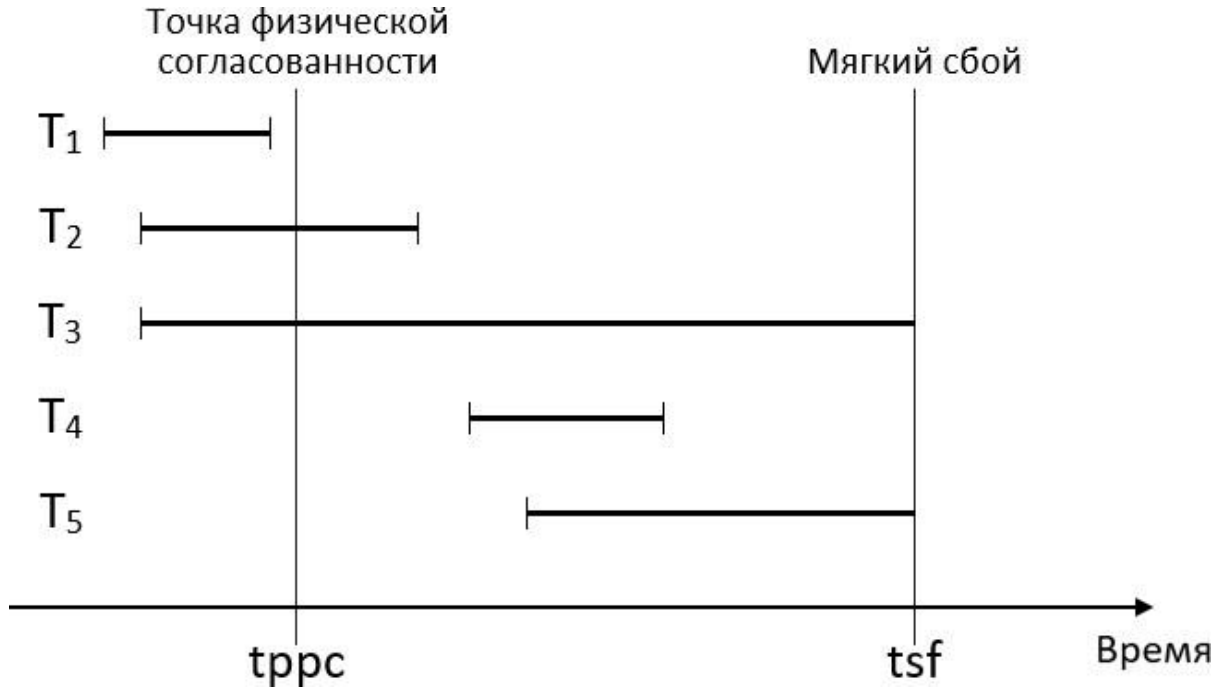
Если полный откат транзакции выполнен успешно, то в журнал заносится запись о конце транзакции, и с точки зрения журнала такая транзакция является зафиксированной.

Восстановление после мягкого сбоя

- К числу основных проблем восстановления после мягкого сбоя относится то, что одна логическая операция изменения БД может изменять несколько физических блоков базы данных.
- Блоки базы данных буферизуются в оперативной памяти и выталкиваются независимо. Если происходит мягкий сбой, то может оказаться, что набор блоков внешней памяти базы данных несогласован, т.е. часть блоков внешней памяти соответствует объекту до изменения, часть – после изменения.
- Состояние внешней памяти базы данных называется **физически согласованным**, если наборы страниц всех объектов согласованы, т.е. они соответствуют либо состоянию объекта либо до его изменения, либо после его изменения.

Точки физической согласованности

В журнале отмечаются моменты времени, в которые база данных находится в физически согласованном состоянии. Такие моменты времени называются точками физической согласованности базы данных (point of physical consistency, ppc).



Восстановление от ppc

Рассмотрим, какие действия нужно выполнить для каждой из транзакций:

- Для транзакции T_1 никаких действий производить не требуется.
- Для транзакции T_2 нужно повторно выполнить (redo) последовательность операций, которые выполнялись после установки точки физически согласованного состояния в момент tppc.
- Для транзакции T_3 нужно выполнить в обратном направлении (undo) ту часть операций, которую она успела выполнить до момента tppc.
- Для транзакции T_4, которая успела начаться после момента tppc и закончиться до момента мягкого сбоя tsf, нужно произвести полное повторное выполнение операций в прямом направлении.
- Для транзакции T_5, начавшейся после момента tppc и не успевшей завершиться к моменту мягкого сбоя tsf, никаких действий предпринимать не требуется.

Создание ppc

Два основных подхода:

- подход, основанный на использовании **теневого механизма**
- подход, в котором применяется **журнализация постраничных изменений базы данных**

Теневой механизм (для файлов)

- Изначально теневой механизм был предложен для поддержки целостности файлов при аварийном отключении питания компьютера.
- Файл представляется как набор блоков внешней памяти, и каждому логическому блоку соответствует некоторый реальный блок на физическом устройстве. Это отображение должно храниться в некоторой таблице. Когда работы с файлом не происходит, то данная таблица хранится в метаданных файла во внешней памяти и называется теневой таблицей. При открытии файла содержимое теневой таблицы копируется в оперативную память, и эта копия называется текущей таблицей.

Теневой механизм (для файлов)

- Когда работающая с файлом программа изменяет блок этого файла, во внешней памяти выделяется новый блок, в который записывается новое изменённое содержимое. При этом только текущая таблица отображения изменяет ссылку на новый блок, а теневая таблица во внешней памяти остаётся неизменной. При закрытии файла текущая таблица записывается на место теневой таблицы, и освобождаются ранее изменённые блоки.
- Если во время работы с открытым файлом происходит мягкий сбой, то для восстановления согласованного состояния файла, достаточно просто использовать теневую таблицу отображения.

Теневой механизм (для СУБД)

Периодически СУБД устанавливает точки физической согласованности базы данных для каждого файла, при этом:

1) Все логические операции завершаются, а новые не начинаются.
2) Выталкиваются все страницы буферного пула базы данных, содержимое которых отличается от содержимого соответствующих блоков внешней памяти.
3) Теневая таблица отображения файлов базы данных заменяется текущей таблицей отображения.

Атомарность записи теневой таблицы

- Есть проблема, состоящая в том, что в любой момент времени теневая таблица отображения должна быть корректной. Для этого необходимо обеспечить атомарность операции замены теневой таблицы отображения. Т.е. надо, чтобы либо она полностью заменилась, либо отсутствовала во внешней памяти.
- Для этого во внешней памяти поддерживаются две области хранения таблицы отображения файлов. Будем называть их областями *A* и *B*. Кроме того, в отдельном блоке внешней памяти хранится флаг *F*, показывающий, какая из этих областей в данный момент содержит действующую теневую таблицу отображения. Назовём соответствующие значения флага F_A и F_B.

Атомарность записи теневой таблицы

Считается, что операция записи одного блока на диск является атомарной:

- Если эта операция заканчивается успешно, это означает, что новая теневая таблица отображения хранится в области *B*.
- Если же запись текущей таблицы отображения в область *B* не удалась, или если не выполнилась операция записи блока с флагом *F*, то продолжает действовать старая теневая таблица отображения.

Журнализация постраничных изменений

Наряду с логической журнализацией операций изменения базы данных производится ещё и журнализация постраничных изменений.

Первый этап восстановления после мягкого сбоя состоит в постраничном откате недовыполненных логических операций. Подобно тому, как это делается с логическими записями по отношению к транзакциям, последней записью о постраничных изменениях от одной логической операции является запись о конце операции. Вообще, выполнение логических операций уровня RSS носит транзакционный характер.

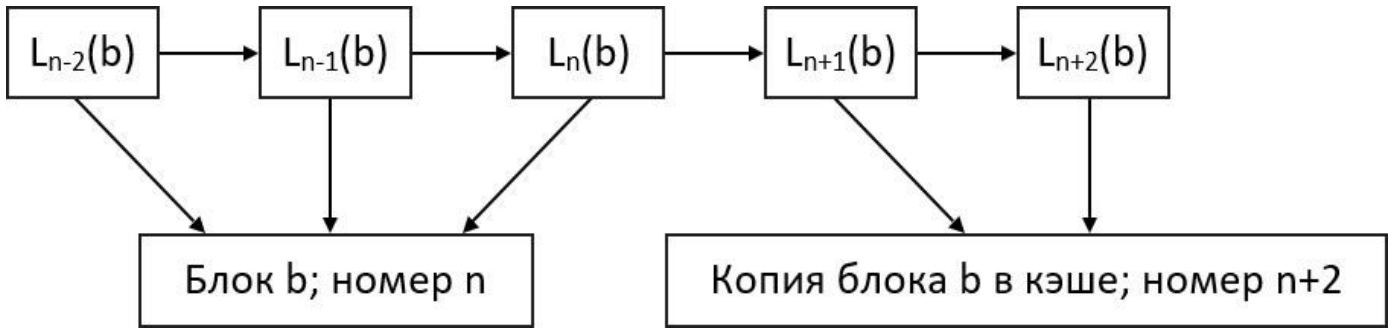
Журнализация постраничных изменений

Допустим, что база данных в момент мягкого сбоя находилась в физически согласованно состоянии, но часть изменённых блоков осталась в кэше и во внешнюю память не попала, а другая часть попала.

Чтобы их различать, в каждый блок внешней памяти и в каждую запись об изменении этого блока в журнале постраничных изменений записывается монотонно возрастающий номер записи. Чтобы понять, нужно ли применить данную запись о постраничном изменении соответствующего блока внешней памяти для восстановления состояния этого блока, требуется всего лишь сравнить номер, содержащийся в этом блоке, с номером, содержащимся в журнальной записи.

Журнализация постраничных изменений

Ниже изображена цепочка записей в журнале об изменении блока *b*. Изменения блока, произведённые операциями, которым соответствуют две хронологически последние журнальные записи $L_{n+1}(b)$ и $L_{n+2}(b)$, в его состоянии во внешней памяти не отражены, поэтому при восстановлении состояния блока требуется выполнить обратные операции изменения блока *b*, соответствующие журнальным записям $L_n(b)$,$L_{n-1}(b)$ и $L_{n-2}(b)$.



Журнал изменений

В ведении журнала постраничных изменений имеются два поднаправления. Можно сказать, что есть логический и физический журналы. В результате получаются два вида записей:

- В логическом журнале записи содержат идентификатор транзакции, старый кортеж до изменения и новый кортеж после изменения.
- В физическом журнале записи содержат предыдущее содержимое блока и несут иной смысл. Он содержит записи об изменении физических объектов – блоков внешней памяти.

Журнал изменений

- Логический журнал должен поддерживать как обратное выполнение журнализованных операций (undo), так и их повторное прямое выполнение (redo). От физического журнала требуется только поддержка обратного выполнения постраничных операций.
- Логический журнал обычно начинает заполняться заново только после выполнения операций резервного копирования базы данных или архивирования самого журнала. До этого времени он линейно растет. В отличие от этого, физический журнал существует сравнительно недолгое время - интервал времени между соседними точками физической согласованности БД.

Переполнение логического журнала

- Устанавливаются «желтая» и «красная» зоны.
  - Когда записи в журнал достигают «желтой» зоны, выдается предупреждение администратору базы данных и прекращается образование новых транзакций.
  - Если все существующие транзакции завершаются до достижения «красной» зоны, автоматически выполняется архивация базы данных или логического журнала.
  - Если какие-то транзакции не успевают завершиться до достижения «красной» зоны журнала, выполняется их аварийный откат, после чего производится архивация базы данных или журнала.

Восстановление БД после жёсткого сбоя

Самый простой способ восстановления основывается на использовании логического журнала и последней архивной копии базы данных.

В этом случае восстановление начинается с обратного копирования базы данных из архивной копии на исправный носитель. Затем для всех закончившихся транзакций выполняется redo, т.е. операции повторно выполняются в прямом смысле.

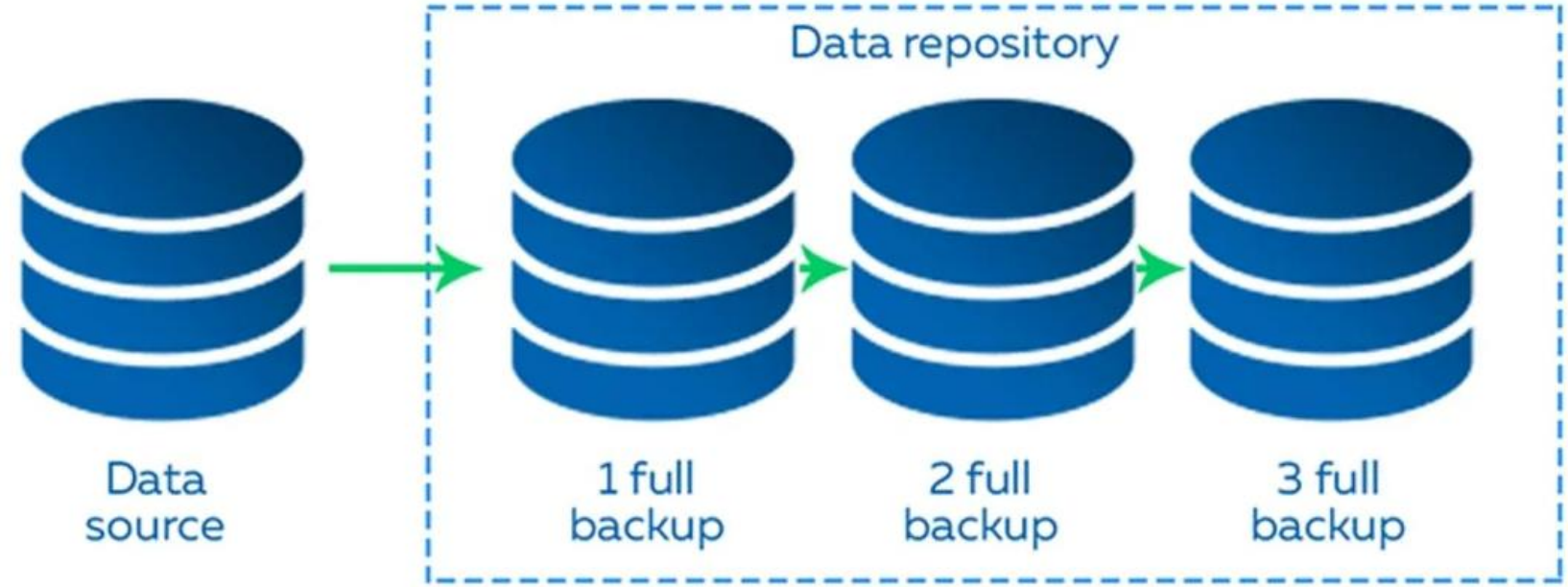
Как выполнять redo

Самый прямолинейный способ - это проигрывать все транзакции в рабочем режиме системы, т.е. для каждой новой транзакции в журнале образовывать новую транзакцию в СУБД и в ней выполнять нужные операции со всей необходимой синхронизацией, т.е. точно так же как в рабочем режиме.

Однако, если придерживаться некоторой дисциплины выполнения логических операций над базой данных, то при восстановлении базы данных после жёсткого сбоя можно просто последовательно повторно выполнять операции по журналу, не обращая внимание на то, в каких транзакциях они выполнялись до жёсткого сбоя.

Архивные копии (БД и журнала)

- В случае утраты журнала – лучше, чем ничего
- Традиционное архивирование базы данных происходит в offline, когда база данных не используется (например, в ночное время).



Архивные копии (БД и журнала)

- В случае утраты журнала – лучше, чем ничего
- Традиционное архивирование базы данных происходит в offline, когда база данных не используется (например, в ночное время).

Для создания архивной копии журнала нужно:

- запретить создание новых транзакций и дождаться завершения всех транзакций
- вытолкнуть все страницы буферного пула во внешнюю память
- скопировать журнал

Тогда для полного восстановления базы данных после жёсткого сбоя достаточно иметь архивную копию базы данных, последовательность архивных копий журнала и последний логический журнал.

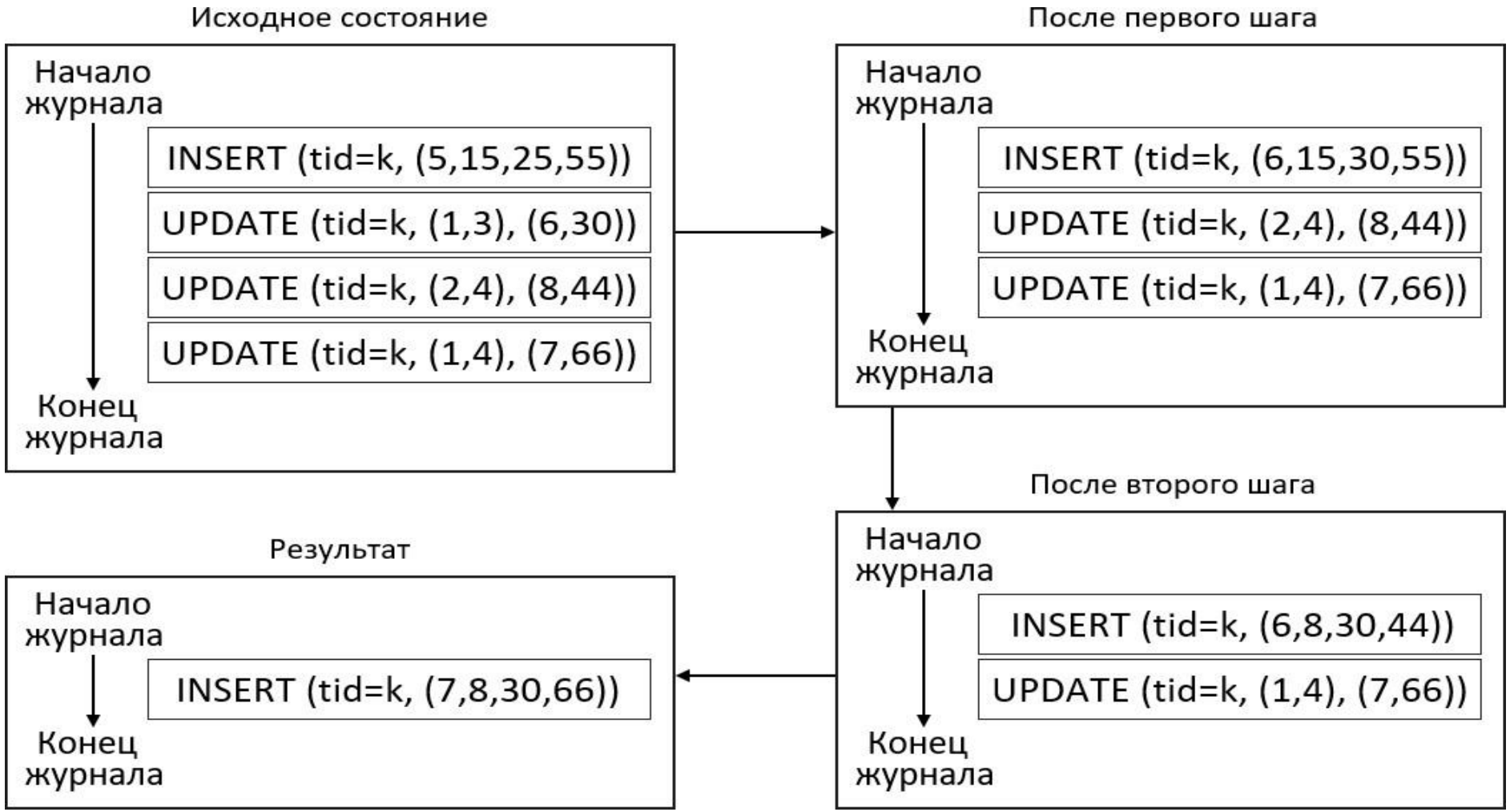
Оптимизация

Может показаться, что восстановление базы данных на основе таких архивных источников будет занимать недопустимо большое время, однако здесь возможна значительная оптимизация.

Эта оптимизация состоит в том, что архивированный логический журнал можно существенно сжать:

- Для этого для каждого объекта базы данных нужно найти последовательность журнальных записей, относящихся к этому объекту, в хронологическом порядке
- И заменить их одной записью, соответствующей операции над объектом, результат которой эквивалентен результату последовательного выполнения журнализованных операций из построенной последовательности.

Оптимизация



Восстановление БД после жёсткого сбоя

Таким образом, для восстановления базы данных после жёсткого сбоя можно воспользоваться:

- исходной архивной копией
- одним сжатым архивным журналом
- последним логическим журналом