Средства журнализации и восстановления баз данных

С.Д. Кузнецов. Базы данных. Тема 10

План лекции (1)

- Введение
- Буферизация блоков базы данных в основной памяти и ее связь с журнализацией
  - Управление буферным пулом базы данных
  - Физическая синхронизация
  - Протокол упреждающей записи в журнал и его связь с буферизацией
- Индивидуальный откат транзакции

Введение (1)

- Одним из основных требований к развитым СУБД является надежность хранения баз данных
- Это требование 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 (2)

- Итак, общей целью журнализации изменений баз данных является обеспечение возможности восстановления согласованного состояния базы данных после любого сбоя
- Поскольку основой поддержания целостного состояния базы данных является механизм транзакций,
  - журнализация и восстановление тесно связаны с понятием транзакции
- 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
- Это и означает, что восстанавливается последнее по времени согласованное состояние базы данных

Введение (3)

- Возможны следующие ситуации, при которых требуется производить восстановление состояния базы данных:
  - Индивидуальный откат транзакции
    - Тривиальной ситуацией отката транзакции является ее явное завершение оператором ROLLBACK
    - Возможны также ситуации, когда откат транзакции инициируется системой
      - Примерами могут быть возникновение исключительной ситуации в прикладной программе
        - например, деление на ноль
        - или выбор транзакции в качестве жертвы при разрушении синхронизационного тупика
    - Для восстановления согласованного состояния базы данных при индивидуальном откате транзакции нужно устранить последствия операторов модификации базы данных, которые выполнялись в этой транзакции

Введение (4)

- Восстановление после внезапной потери содержимого оперативной памяти
  - мягкий сбой
- Такая ситуация может возникнуть при аварийном выключении электрического питания, при возникновении неустранимого сбоя процессора
  - например, срабатывании контроля основной памяти
- и т.д.
- Ситуация характеризуется потерей той части базы данных, которая к моменту сбоя содержалась в буферах оперативной памяти СУБД

Введение (5)

- Восстановление после поломки основного внешнего носителя базы данных
  - жесткий сбой
- Эта ситуация при достаточно высокой надежности современных устройств внешней памяти может возникать сравнительно редко, но, тем не менее, СУБД должна быть в состоянии восстановить базу данных даже и в этом случае
- Основой восстановления является архивная копия и журнал изменений базы данных.

- Во всех трех случаях основой восстановления является хранение избыточных данных
- Эти избыточные данные хранятся в журнале, содержащем последовательность записей об изменении базы данных

Введение (6)

- Восстановление после поломки основного внешнего носителя базы данных
  - жесткий сбой
- Эта ситуация при достаточно высокой надежности современных устройств внешней памяти может возникать сравнительно редко, но, тем не менее, СУБД должна быть в состоянии восстановить базу данных даже и в этом случае
- Основой восстановления является архивная копия и журнал изменений базы данных.

- Во всех трех случаях основой восстановления является хранение избыточных данных
- Эти избыточные данные хранятся в журнале, содержащем последовательность записей об изменении базы данных

Введение (7)

- Возможны два основных варианта ведения журнальной информации
- В первом варианте для каждой транзакции поддерживается отдельный локальный журнал изменений базы данных этой транзакцией
- Эти локальные журналы используются для индивидуальных откатов транзакций и могут поддерживаться в основной
  - правильнее сказать, в виртуальной
- памяти СУБД
- Кроме того, поддерживается общий журнал изменений базы данных,
  - используемый для восстановления состояния базы данных после мягких и жестких сбоев.

Введение (8)

- Данный подход позволяет быстро выполнять индивидуальные откаты транзакций,
  - но приводит к дублированию информации в локальных и общем журналах
- Поэтому чаще используется второй вариант – поддержка только общего журнала изменений базы данных,
  - который используется и при выполнении индивидуальных откатов
- Здесь мы рассматриваем именно этот вариант

Буферизация блоков базы данных в основной памяти (1)

- Журнализация операций изменения базы данных тесно связана не только с управлением транзакциями, но и
  - с буферизацией блоков базы данных в основной памяти
- По причинам объективно существующей разницы в скорости работы процессоров и основной памяти и устройств внешней памяти
  - эта разница в скорости существовала, существует, и будет существовать всегда
- буферизация блоков базы данных в основной памяти является единственным реальным способом достижения приемлемой эффективности СУБД
- Без поддержки буферизации базы данных СУБД работала бы со скоростью магнитных дисков,
  - т.е. на несколько порядков медленнее, чем если бы обработка данных происходила в основной памяти

Буферизация блоков базы данных в основной памяти (2)

- Если бы каждая запись об изменении базы данных, которая должна поступить в журнал при выполнении любой операции обновления базы данных, реально немедленно перемещалась бы во внешнюю память,
  - это привело бы к существенному замедлению работы системы
- Фактически, тогда каждая операция обновления базы данных выполнялась бы со скоростью магнитного диска
- Поэтому записи в журнал тоже буферизуются:
  - при нормальной работе буфер выталкивается во внешнюю память журнала только при полном заполнении записями

Буферизация блоков базы данных в основной памяти (3)

- Более точно, для буферизации записей журнала обычно используются два буфера
- После полного заполнения первый буфер выталкивается на магнитный диск, и пока совершается этот обмен,
  - журнальные записи размещаются во втором буфере
- К моменту конца обмена заполняется второй буфер,
  - он выталкивается во внешнюю память,
  - а журнальные записи снова размещаются в первом буфере и т.д.

Буферизация блоков базы данных в основной памяти (4)

- Здесь идет речь об использовании буферов
  - и базы данных, и журнала,
- располагающихся именно в физической основной памяти,
  - управляемой непосредственно СУБД, а не виртуальной памяти СУБД, управляемой операционной системой
- Использование буферов виртуальной памяти является практически бессмысленным делом, поскольку в этом случае операционная система, руководствуясь своими собственными стратегиями управления основной памяти, в любой момент может
  - удалить буферную страницу СУБД из основной памяти и
  - перенести ее копию во внешнюю память в область свопинга
- Тогда при следующей попытке записи СУБД в эту страницу возникнет прерывание, при обработке которого операционная система подкачает страницу в основную память,
  - выполнив совершенно не ожидаемый СУБД обмен с внешней памятью

Буферизация блоков базы данных в основной памяти (5)

- Нельзя надеяться на то, что операционная система настолько грамотно управляет основной памятью, что
  - нужные страницы виртуальной памяти СУБД в нужное время будут находиться в основной памяти
- Операционная система просто не обладает достаточной информацией, чтобы всегда принимать правильные решения
- Правильно управлять своей буферной памятью может только сама СУБД, «отбирающая» у операционной системы часть физической основной памяти для размещения в ней буферов базы данных и журнала

Буферизация блоков базы данных в основной памяти (6)
Управление буферным пулом базы данных (1)

- В развитых
  - вернее сказать, правильно организованных
- СУБД поддерживается собственная стратегия замещения страниц буферного пула
- Задача, которую решает СУБД, очень похожа на задачу, которую решает операционная система при управлении виртуальной памятью.
- В случае операционной системы, если некоторый процесс требует обеспечения доступа к странице виртуальной памяти, отсутствующей в основной памяти, и нет свободных страниц основной памяти,
  - в соответствии с некоторым критерием выбирается некоторая занятая страница основной памяти, освобождается
    - т.е. изымается из виртуальной памяти какого-то процесса и, может быть, копируется на диск
    - и подключается к виртуальной памяти запросившего процесса с предварительным считыванием с диска нужных данных

Буферизация блоков базы данных в основной памяти (7)
Управление буферным пулом базы данных (2)

- В случае СУБД, 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 и
    - предоставить доступ к этой странице запросившей операции
- Конечно, в буферном пуле может не оказаться свободных страниц, и тогда
  - СУБД в соответствии с некоторым критерием
    - находит некоторую занятую страницу,
    - освобождает ее
    - возможно, выталкивает во внешнюю память

Буферизация блоков базы данных в основной памяти (8)
Управление буферным пулом базы данных (3)

- Основная разница между этими случаями состоит в критерии выборки занятой страницы для «откачки»
- Не будем обсуждать здесь стратегии замещения страниц, используемые в операционных системах
- Заметим лишь, что почти всегда операционная система стремится заменить страницу, к которой
  - предположительно дольше всего не будет обращений,
- но, поскольку предвидение будущего невозможно,
  - оно аппроксимируется прошлым.

Буферизация блоков базы данных в основной памяти (9)
Управление буферным пулом базы данных (4)

- В частности, в одном из популярных алгоритмов замещения страниц LRU (Least Recently Used) принимается предположение, что
  - дольше всего в будущем не потребуется та страница, к которой дольше всего не обращались в прошлом
- В стратегии замещения страниц буферного пула СУБД тоже чаще всего используется некоторая разновидность алгоритма LRU
- Но СУБД располагает большей информацией о страницах буферного пула, чем операционная система о страницах основной памяти

Буферизация блоков базы данных в основной памяти (10)
Управление буферным пулом базы данных (5)

- Например, если в некоторой транзакции выполняется
  - сканирование некоторой таблицы без использования индекса, и
  - при выполнении операции NEXT был затребован доступ к некоторому блоку базы данных
    - с соответствующим перемещением копии этого блока в некоторую страницу буферного пула,
- то подсистема управления буферным пулом «знает», что
  - эта страница еще точно потребуется до тех пор, пока не будет прочитан последний кортеж сканируемой таблицы, располагающийся в данной странице
- Более того, СУБД «знает»,
  - какой блок базы данных потребуется после завершения просмотра кортежей данного блока, и
- может заранее переместить его копию в некоторую страницу буферного пула

Буферизация блоков базы данных в основной памяти (11)
Управление буферным пулом базы данных (6)

- Кроме того, некоторые блоки базы данных заведомо требуются чаще других блоков
- Например, при любом просмотре таблицы на основе некоторого индекса гарантированно потребуется
  - доступ к корневому блоку соответствующего B-дерева
- При вставке кортежа в любую таблицу или удалении из нее кортежа будет необходимо должным образом изменить все определенные для нее индексы, и для этого тоже гарантированно потребуется
  - доступ к корневым блокам всех соответствующих B-деревьев

Буферизация блоков базы данных в основной памяти (12)
Управление буферным пулом базы данных (7)

- Поэтому в стратегии замещения страниц буферного пула базы данных обычно используется алгоритм LRU с приоритетами страниц
  - высокоприоритетные страницы стареют,
    - т.е. становятся кандидатами на замещение,
  - медленнее, чем низкоприоритетные страницы
- В частности, страницы, содержащие копии корневых блоков индексов, являются настолько высокоприоритетными, что обычно никогда не замещаются
- Кроме того, поддерживается предварительное считывание в буферную память копий блоков, доступ к которым вскоре понадобится

Буферизация блоков базы данных в основной памяти (12)
Физическая синхронизация (1)

- Поскольку в СУБД может одновременно
  - «параллельно»
- выполняться несколько транзакций, вполне реальна ситуация, когда
  - в двух одновременно выполняемых операциях требуется доступ к одному и тому же блоку базы данных
    - т.е. к одной и той же буферной странице, содержащей копию этого блока
- Понятно, что в одновременном доступе для чтения содержимого блока ничего плохого нет, но
  - параллельное изменение блока может привести к непредсказуемым результатам
- Следует заметить, что, вообще говоря, координацию параллельного доступа к страницам буферного пула
  - не обеспечивает логическая синхронизация, используемая для сериализации транзакций

Буферизация блоков базы данных в основной памяти (13)
Физическая синхронизация (2)

- Например, предположим, что в двух параллельно выполняемых транзакциях одновременно выполняются операции модификации кортежей,
  - у одного из которых tid = (*n*, 1), а
  - у другого tid = (*n*, 2)
- Если в СУБД используются блокировки на уровне кортежей, то система
  - допустит параллельное выполнение этих двух операций,
  - и они будут одновременно изменять страницу,
    - содержащую копию блока базы данных с номером *n*
- При выполнении обеих операций может потребоваться перемещение кортежей внутри этого блока, и понятно, что в результате ничего хорошего, скорее всего, не получится

Буферизация блоков базы данных в основной памяти (14)
Физическая синхронизация (3)

- Аналогично, логическая синхронизация может легко допустить
  - параллельное выполнение нескольких операций,
    - требующих обновления одного и того же индекса
- Некоординированное параллельное обновление B-дерева с большой вероятностью приводит к разрушению его структуры

Буферизация блоков базы данных в основной памяти (15)
Физическая синхронизация (6)

- Поэтому при выполнении операций уровня RSS необходимо поддерживать
  - дополнительную «физическую» синхронизацию, в которой единицами блокировки служат
    - страницы буферного пула (или блоки) базы данных
- В пределах операции перед чтением из страницы буферного пула
  - блока базы данных
- требуется запросить у подсистемы управления буферным пулом
  - блокировку соответствующей страницы
    - блока
  - в режиме S,
- а перед записью в страницу (в блок) –
  - ее блокировку в режиме X
- Совместимость блокировок обычная

Буферизация блоков базы данных в основной памяти (16)
Физическая синхронизация (7)

- Но блокировки страниц буферного пула нужны не только для координации параллельного доступа к страницам при параллельном выполнении транзакций
- При выполнении операций уровня RSS могут возникать ошибки, обнаруживаемые в середине операции, уже после того, как одна или несколько страниц буферного пула
  - блоков базы данных
- было изменено
- Например, может выполняться операция вставки кортежа в некоторую таблицу,
  - нарушающая уникальность некоторого индекса, определенного над этой таблицей
- Нарушение уникальности этого индекса будет обнаружено при попытке вставить в него новый ключ, но до этого
  - новый кортеж уже мог быть размещен в блоке данных, и
  - некоторые индексы уже могли быть успешно обновлены

Буферизация блоков базы данных в основной памяти (17)
Физическая синхронизация (8)

- При обнаружении ошибки операции нужно ликвидировать все ее следы в базе данных и выдать соответствующий код ошибки на уровень RDS
- Проще всего сделать это, произведя обратные изменения всех страниц
  - блоков базы данных,
- которые были изменены при прямом выполнении операции. Но для этого требуется, чтобы все страницы
  - блоки базы данных,
- заблокированные при выполнении операции,
  - оставались заблокированными до конца этой операции

Буферизация блоков базы данных в основной памяти (18)
Физическая синхронизация (9)

- Тем самым, для подсистемы управления буферным пулом операции уровня RSS являются (почти) тем же, чем являются транзакции для подсистемы управления транзакциями
- Достаточным условием корректного выполнения операций является соблюдение двухфазного протокола синхронизационных блокировок над страницами буферного пула в пределах операций.
- Это условие не является необходимым
- Каждую операцию уровня RSS можно разбить на последовательность «микроопераций» и
  - потребовать соблюдения двухфазного протокола синхронизационных блокировок в пределах микроопераций

Буферизация блоков базы данных в основной памяти (19)
Физическая синхронизация (10)

- Например, операцию INSERT уровня RSS можно разбить на следующие микрооперации:
  - нахождение блока данных для вставки;
  - вставка кортежа в найденный блок;
  - обновление индекса 1;
  - ….
  - обновление индекса *n*,
- где *n* – число индексов, определенных для данной таблицы
- Общий принцип состоит в том, что в пределах одной микрооперации блокируются все блоки базы данных, которые обязаны быть изменены согласованным образом

Буферизация блоков базы данных в основной памяти (20)
Протокол упреждающей записи в журнал (1)

- Реальная ситуация является более сложной
- Имеются два вида буферов –
  - буфера журнала и
  - буферный пул страниц основной памяти, –
- которые содержат связанную информацию
- И те, и другие буфера могут выталкиваться во внешнюю память
- Основной причиной выталкивания буфера журнала является его полное заполнение журнальными записями
- Страницы буферного пула базы данных чаще всего выталкиваются во внешнюю память, когда
  - требуется переместить в основную память некоторый блок базы данных, а
  - свободных страниц в буферном пуле нет

Буферизация блоков базы данных в основной памяти (21)
Протокол упреждающей записи в журнал (2)

- Тогда срабатывает алгоритм замещения страниц, выбирается страница,
  - содержимое которой, вероятно, дольше всего не потребуется,
- и эта страница
  - если ее содержимое изменялось
- выталкивается в соответствующий блок внешней памяти базы данных
- Проблема состоит в выработке некоторой общей политики выталкивания, которая обеспечивала бы
  - возможность восстановления состояния базы данных после сбоев

Буферизация блоков базы данных в основной памяти (22)
Протокол упреждающей записи в журнал (3)

- Эта проблема не возникает при индивидуальных откатах транзакций, поскольку в этих случаях
  - содержимое основной памяти не утрачено, и
  - при восстановлении можно пользоваться содержимым
    - как буфера журнала,
    - так и буферных страниц базы данных
- Но если произошел мягкий сбой, и
  - содержимое буферов утрачено,
- то для проведения восстановления базы данных
  - необходимо иметь некоторое согласованное состояние журнала и базы данных во внешней памяти

Буферизация блоков базы данных в основной памяти (23)
Протокол упреждающей записи в журнал (4)

- 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 измененный объект окажется во внешней памяти базы данных
- Соответствующий протокол журнализации (и управления буферизацией) называется WAL (Write Ahead Log, «пиши сначала в журнал») и состоит в том, что
  - если требуется вытолкнуть во внешнюю память буферную страницу, содержащую измененный объект базы данных, то
    - перед этим нужно гарантировать выталкивание во внешнюю память журнала буферной страницы журнала, содержащей запись об изменении этого объекта

Буферизация блоков базы данных в основной памяти (24)
Протокол упреждающей записи в журнал (5)

- При следовании протоколу WAL,
  - если во внешней памяти базы данных находится некоторый объект базы данных, по отношению к которому выполнена операция модификации,
  - то во внешней памяти журнала обязательно находится запись, соответствующая этой операции
- Обратное неверно, т.е.
  - если во внешней памяти журнала содержится запись о некоторой операции изменения объекта базы данных,
  - то сам измененный объект может отсутствовать во внешней памяти базы данных

Буферизация блоков базы данных в основной памяти (25)
Протокол упреждающей записи в журнал (6)

- Дополнительное условие на выталкивание буферов накладывается тем требованием, что
  - каждая успешно завершенная транзакция должна быть реально зафиксирована во внешней памяти
- Какой бы сбой не произошел,
  - система должна быть в состоянии восстановить состояние базы данных, содержащее результаты всех транзакций, зафиксированных до момента сбоя
- Самым простым решением было бы
  - выталкивание буфера журнала, за которым следует
  - массовое выталкивание буферов страниц базы данных, изменявшихся данной транзакцией
- Довольно часто так и делают, но это вызывает существенные накладные расходы при выполнении операции фиксации транзакции

Буферизация блоков базы данных в основной памяти (26)
Протокол упреждающей записи в журнал (7)

- Оказывается, что минимальным требованием, гарантирующим
  - возможность восстановления последнего согласованного состояния базы данных,
- является
  - выталкивание при фиксации транзакции во внешнюю память журнала всех записей об изменении базы данных этой транзакцией
- При этом последней записью в журнал, производимой от имени данной транзакции, является
  - специальная запись о конце транзакции
- Рассмотрим теперь, как можно выполнять операции восстановления базы данных в различных ситуациях,
  - если в системе поддерживается общий для всех транзакций журнал с общей буферизацией записей,
    - поддерживаемый в соответствии с протоколом WAL

Индивидуальный откат транзакции (1)

- Для обеспечения возможности индивидуального отката транзакции по общему журналу все записи в журнале от данной транзакции связываются в обратный список
- В начале списка для незавершенных транзакций находится запись о последнем изменении базы данных, произведенном данной транзакцией
  - в этом случае хронологически последние записи могут быть еще не вытолкнуты во внешнюю память журнала и могут находиться в буфере основной памяти
- Для закончившихся транзакций
  - индивидуальные откаты которых уже невозможны
- началом списка является запись о конце транзакции, которая обязательно вытолкнута во внешнюю память журнала,
  - т.е. весь список находится во внешней памяти
- Концом списка всегда служит первая запись об изменении базы данных, произведенном данной транзакцией
- Обычно в каждой записи проставляется уникальный идентификатор транзакции,
  - чтобы можно было восстановить прямой список записей об изменениях базы данных данной транзакцией

Индивидуальный откат транзакции (2)

- Итак, индивидуальный откат транзакции
  - это возможно только для незавершенных транзакций
- выполняется следующим образом:
  - Выбирается очередная журнальная запись из списка данной транзакции.
  - Выполняется противоположная по смыслу операция:
    - вместо операции INSERT выполняется соответствующая операция DELETE,
    - вместо операции DELETE выполняется INSERT, и
    - вместо прямой операции UPDATE – обратная операция UPDATE, восстанавливающая предыдущее состояние объекта базы данных

Индивидуальный откат транзакции (3)

- Любая из этих обратных операций также журнализуется
- Собственно для индивидуального отката это не нужно,
  - но при выполнении индивидуального отката транзакции может произойти мягкий сбой, при восстановлении после которого потребуется откатить транзакции,
    - для которых не полностью выполнен индивидуальный откат
- При успешном завершении отката в журнал заносится запись о конце транзакции
  - С точки зрения журнала такая транзакция является зафиксированной

Восстановление после мягкого сбоя (1)

- К числу основных проблем восстановления после мягкого сбоя относится то, что одна логическая операция изменения базы данных может изменять несколько физических блоков базы данных,
  - например, блок данных и несколько блоков индексов
- Блоки базы данных буферизуются в оперативной памяти и выталкиваются независимо
- После мягкого сбоя набор блоков внешней памяти базы данных может оказаться несогласованным,
  - т.е. часть блоков внешней памяти соответствует объекту до изменения, часть – после изменения.

Восстановление после мягкого сбоя (2)

- Например, в результате выполнения операции UPDATE соответствующий кортеж мог переместиться в другой блок
- В этом случае изменяются два блока:
  - в описатель кортежа в его исходном блоке записывается его новый tid,
  - а в новом блоке размещается сам модифицированный кортеж
- Очевидно, что если хотя бы один из этих блоков не попал во внешнюю память базы данных к моменту мягкого сбоя, то при восстановлении не удастся вернуть кортеж на его прежнее место

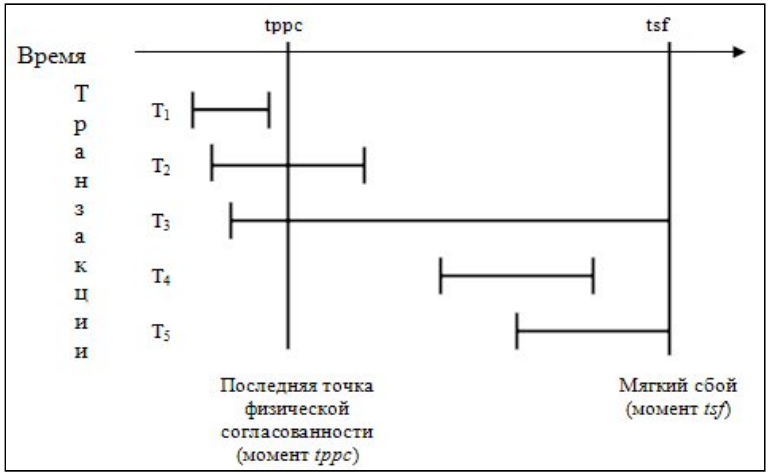
Восстановление после мягкого сбоя (3)

- Другими словами, к такому состоянию внешней памяти базы данных не применимы операции логического уровня
- Состояние внешней памяти базы данных называется *физически согласованным*, если наборы страниц всех объектов согласованы,
  - т.е. соответствуют состоянию любого объекта либо после его изменения, либо до изменения

Восстановление после мягкого сбоя (4)

Схема восстановления от точки физической согласованности (1)

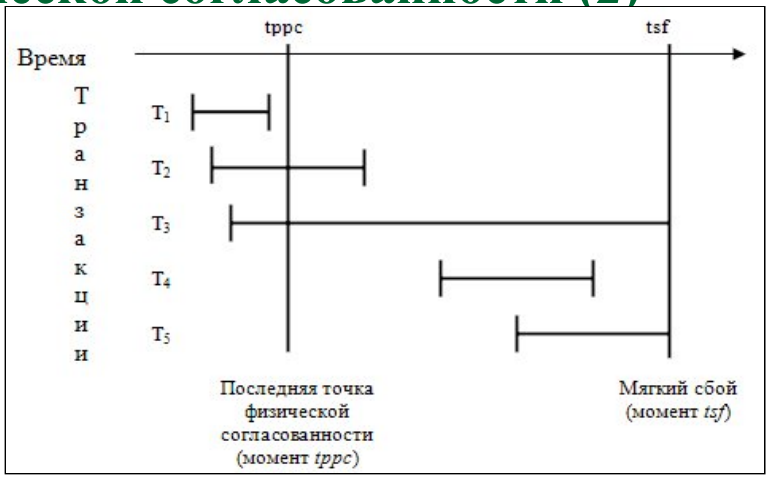
- Будем считать, что в журнале отмечаются точки физической согласованности базы данных – моменты времени, в которые во внешней памяти
  - содержатся согласованные результаты операций, завершившихся до соответствующего момента времени,
  - и отсутствуют результаты операций, которые не завершились,
  - а буфер журнала вытолкнут во внешнюю память
- Позже мы обсудим, как можно достичь физической согласованности
- Назовем такие точки ppc (point of physical consistency)
- Все возможные состояния транзакций к моменту мягкого сбоя показаны на рисунке



Восстановление после мягкого сбоя (5)

Схема восстановления от точки физической согласованности (2)

- Предположим, что каким-то образом удалось восстановить внешнюю память базы данных к состоянию на момент времени tppc
  - как это можно сделать, обсудим позже

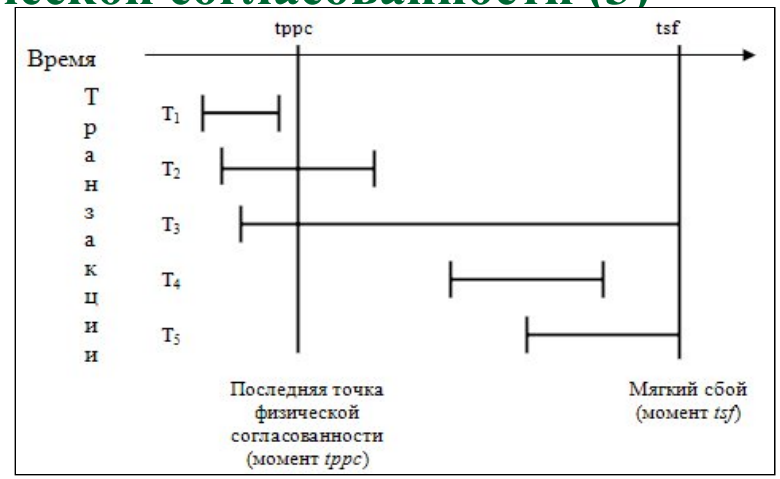


- Тогда восстановление последнего по времени логически целостного состояния базы данных производится следующим образом
  - Для транзакции T1 никаких действий производить не требуется
    - Она закончилась до момента *tppc*, и все ее результаты гарантированно отражены во внешней памяти базы данных

Восстановление после мягкого сбоя (6)

Схема восстановления от точки физической согласованности (3)

- Для транзакции Т2 нужно повторно выполнить (*redo*) последовательность операций,
  - которые выполнялись после установки точки физически согласованного состояния в момент *tppc*

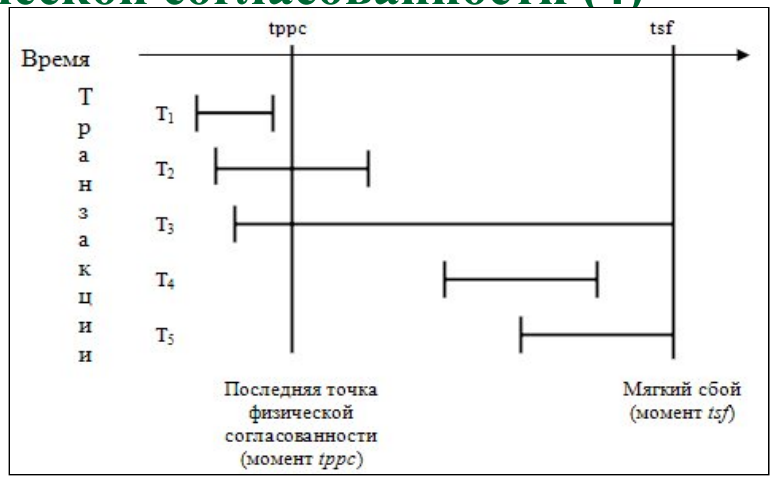


- Действительно, во внешней памяти полностью отсутствуют следы операций, которые выполнялись в транзакции Т2 после момента *tppc*
- Следовательно, повторное прямое
  - по смыслу и хронологии
- выполнение операций транзакции Т2 корректно и приведет к логически согласованному состоянию базы данных
  - Поскольку транзакция Т2 успешно завершилась до момента мягкого сбоя *tfs*, в журнале содержатся записи обо всех изменениях базы данных, произведенных этой транзакцией

Восстановление после мягкого сбоя (7)

Схема восстановления от точки физической согласованности (4)

- Для транзакции Т3 нужно выполнить в обратном направлении (undo) ту часть операций,
  - которую она успела выполнить до момента *tppc*
- Действительно, во внешней памяти базы данных полностью отсутствуют результаты операций Т3, которые были выполнены после момента *tppc*

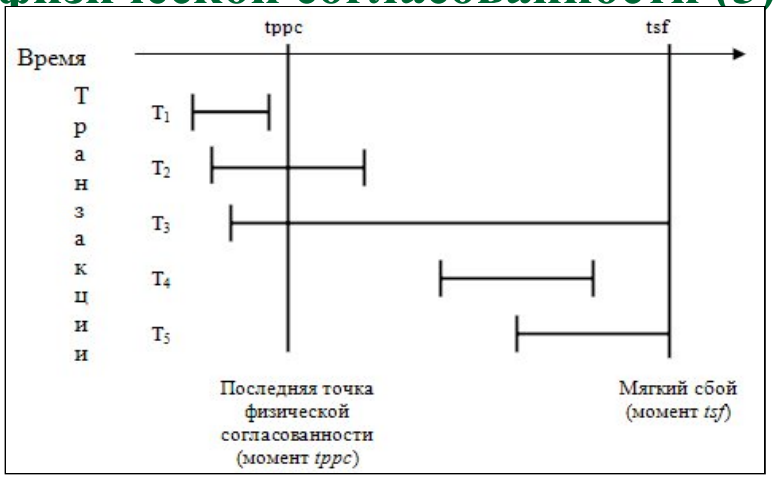


- С другой стороны, во внешней памяти гарантированно присутствуют результаты операций Т3, которые были выполнены до момента *tppc*
- Следовательно, обратное выполнение
  - по смыслу и хронологии
- операций Т3 корректно и приведет к согласованному состоянию базы данных
  - Поскольку транзакция Т3 не завершилась к моменту мягкого сбоя *tfs*, при восстановлении необходимо устранить все последствия ее выполнения

Восстановление после мягкого сбоя (8)

Схема восстановления от точки физической согласованности (5)

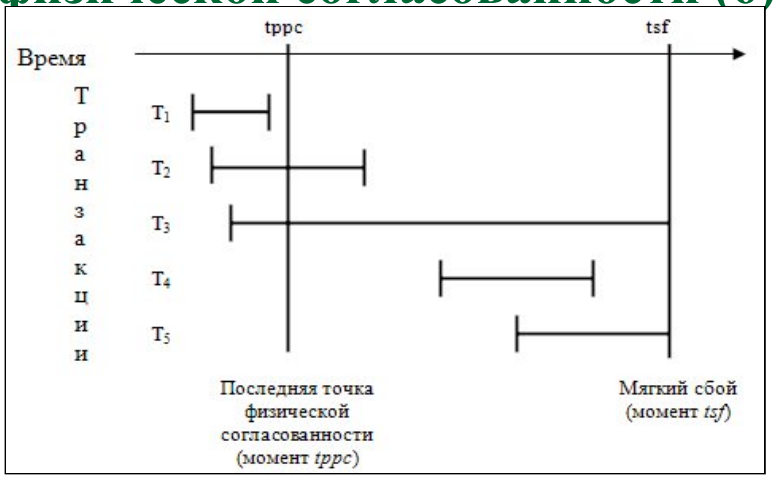
- Для транзакции T4, которая успела начаться после момента *tppc* и закончиться до момента мягкого сбоя *tfs*,



  - нужно произвести полное повторное выполнение операций в прямом направлении
  - Поскольку транзакция T4 успешно завершилась до момента мягкого сбоя *tfs*, в журнале содержатся записи обо всех изменениях базы данных, произведенных этой транзакцией

Восстановление после мягкого сбоя (9)

Схема восстановления от точки физической согласованности (6)



- Наконец, для транзакции T5, начавшейся после момента *tppc* и не успевшей завершиться к моменту мягкого сбоя *tfs*,
  - никаких действий предпринимать не требуется
  - Результаты операций этой транзакции полностью отсутствуют во внешней памяти базы данных

Восстановление после мягкого сбоя (10)

Восстановление физической согласованности базы данных (1)

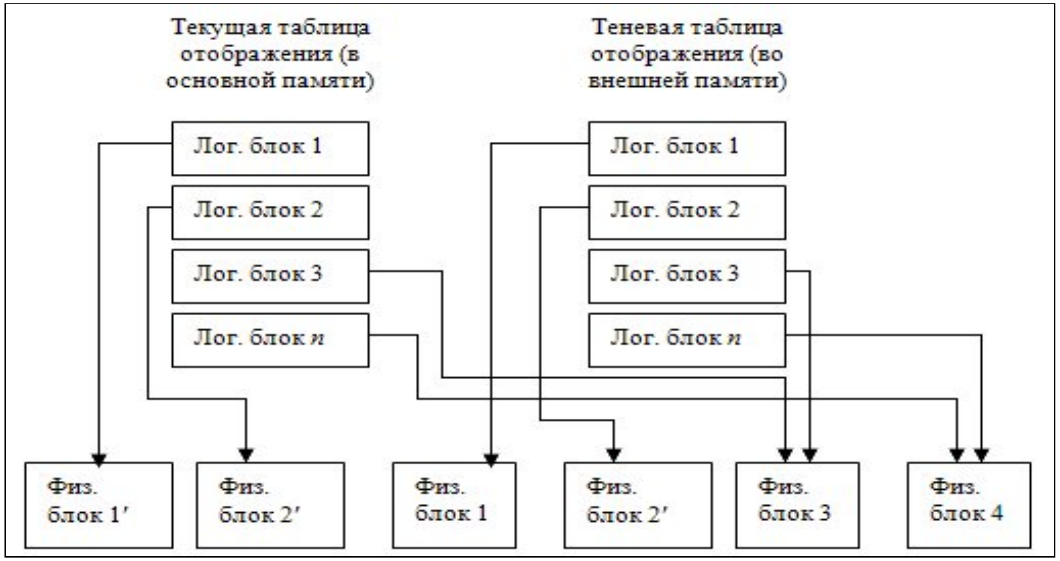
- Каким же образом можно обеспечить наличие точек физической согласованности базы данных,
  - т.е. как восстановить состояние базы данных в момент *tppc*?
- Для этого используются два основных подхода:
  - подход, основанный на использовании теневого механизма, и
  - подход, в котором применяется журнализация постраничных изменений базы данных

Восстановление после мягкого сбоя (11)

Восстановление физической согласованности базы данных (2)

Теневой механизм (1)

- Теневой механизм был изначально предложен для поддержки целостности файлов при аварийном отключении питания компьютера



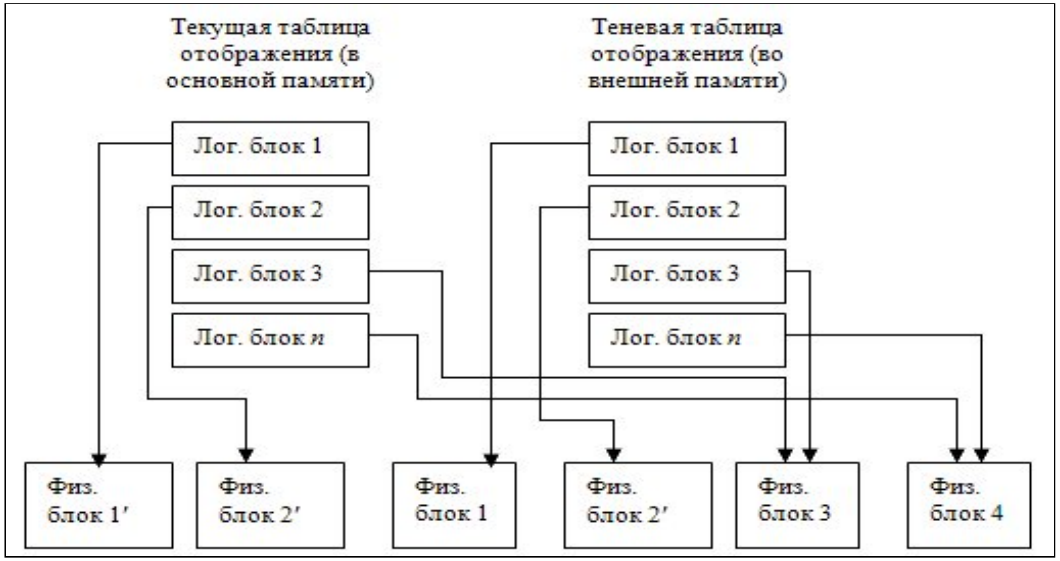
- Файл представляется как набор блоков внешней памяти,
  - для доступа к которым поддерживается таблица отображения
- При открытии файла таблица отображения номеров его логических блоков в адреса физических блоков внешней памяти считывается в оперативную память

Восстановление после мягкого сбоя (12)

Восстановление физической согласованности базы данных (3)

Теневой механизм (2)

- При модификации любого блока файла во внешней памяти выделяется новый блок
- При этом текущая таблица отображения
  - в основной памяти



- изменяется, а теневая остается неизменной
- Если во время работы с открытым файлом происходит сбой,
  - во внешней памяти автоматически сохраняется состояние файла до его открытия
- Для явного восстановления файла достаточно повторно считать в основную память теневую таблицу отображения

Восстановление после мягкого сбоя (13)

Восстановление физической согласованности базы данных (4)

Теневой механизм (3)

- В контексте базы данных теневой механизм используется следующим образом
- Периодически выполняются операции установки точки физической согласованности базы данных
- При выполнении этой операции
  - все логические операции завершаются,
  - выталкиваются все страницы буферного пула базы данных, содержимое которых отличается от содержимого соответствующих блоков внешней памяти
  - теневая таблица отображения файлов (сегментов) базы данных заменяется текущей таблицей отображения
    - правильнее сказать, текущая таблица отображения записывается на место теневой

Восстановление после мягкого сбоя (14)

Восстановление физической согласованности базы данных (5)

Теневой механизм (4)

- Здесь имеется некоторая проблема, состоящая в том, что
  - в любой момент времени теневая таблица отображения должна быть корректной,
    - т.е. соответствовать некоторому ранее зафиксированному физически целостному состоянию базы данных
- Для этого необходимо обеспечить атомарность операции замены теневой таблицы отображения
- В общем случае таблица отображения может занимать несколько блоков внешней памяти, и для записи текущей таблицы отображения на место теневой таблицы в этом случае потребуется несколько обменов с дисками
- Если в промежутке между этими обменами возникнет мягкий сбой, то будет
  - благополучно утрачена текущая таблица отображения и
  - безнадежно испорчена теневая таблица,
    - т.е. мы просто лишимся возможности восстанавливаться за счет использования последнего физически согласованного состояния базы данных

Восстановление после мягкого сбоя (15)

Восстановление физической согласованности базы данных (6)

Теневой механизм (5)

- Чтобы это не произошло, 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
    - назовем соответствующие значения флага F_A и F_B
- Тогда, если сохраненным во внешней памяти значением флага является F_A, то
  - текущая таблица отображения записывается в область B

Восстановление после мягкого сбоя (16)

Восстановление физической согласованности базы данных (7)

Теневой механизм (6)

- Если эта операция выполняется успешно, то в блок флага записывается значение F_B
- 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

Восстановление после мягкого сбоя (17)

Восстановление физической согласованности базы данных (8)

Теневой механизм (7)

- Восстановление хронологически последнего сохраненного физически согласованного состояния базы данных происходит мгновенно:
  - текущая таблица отображения заменяет теневой таблицей
    - при восстановлении просто считывается действующая теневая таблица отображения
- Все проблемы восстановления решаются, но за счет слишком большого перерасхода внешней памяти
  - В пределе может потребоваться вдвое больше внешней памяти, чем реально нужно для хранения базы данных

Восстановление после мягкого сбоя (18)

Восстановление физической согласованности базы данных (9)

Журнализация постраничных изменений (1)

- Возможен другой подход, при использовании которого наряду с логической журнализацией операций изменения базы данных производится
  - 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

Восстановление после мягкого сбоя (19)

Восстановление физической согласованности базы данных (10)

Журнализация постраничных изменений (2)

- В частности, как уже отмечалось раньше, при выполнении логической операции обновления базы данных, вообще говоря,
  - изменяется несколько блоков базы данных
- Для обеспечения возможности отката отдельной операции
  - а это может потребоваться, например, если обнаруживается нарушение свойства уникальности какого-либо индекса
- приходится до конца операции монопольно блокировать
  - все страницы буферного пула базы данных,
    - содержащие копии изменяемых этой операцией блоков базы данных

Восстановление после мягкого сбоя (20)

Восстановление физической согласованности базы данных (11)

Журнализация постраничных изменений (3)

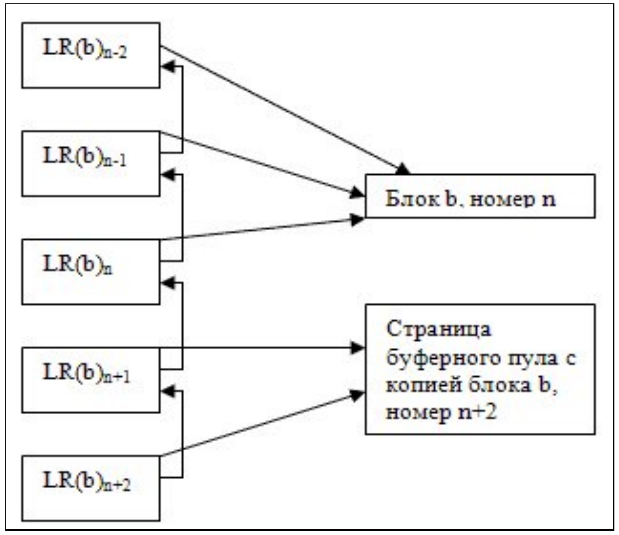
- Чтобы распознать, нуждается ли страница внешней памяти базы данных в восстановлении, при выталкивании любой страницы из буферного пула основной памяти в нее помещается
  - номер последней записи о постраничном изменении этой страницы
- Этот же номер запоминается в самой записи
- Чтобы понять, нужно 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
- Если в блоке содержится номер, меньший номера журнальной записи, то это означает, что буферная страница, в которой выполнялось соответствующее изменение,
  - не была к моменту мягкого сбоя вытолкнута во внешнюю память, и
  - применять данную запись для восстановления соответствующего блока внешней памяти не требуется

Восстановление после мягкого сбоя (21)

Восстановление физической согласованности базы данных (12)

Журнализация постраничных изменений (4)

- Пять записей об изменении блока b с номерами *n*-2, *n*-1, *n*, *n*+1, *n*+2
- В блоке b содержится номер *n*

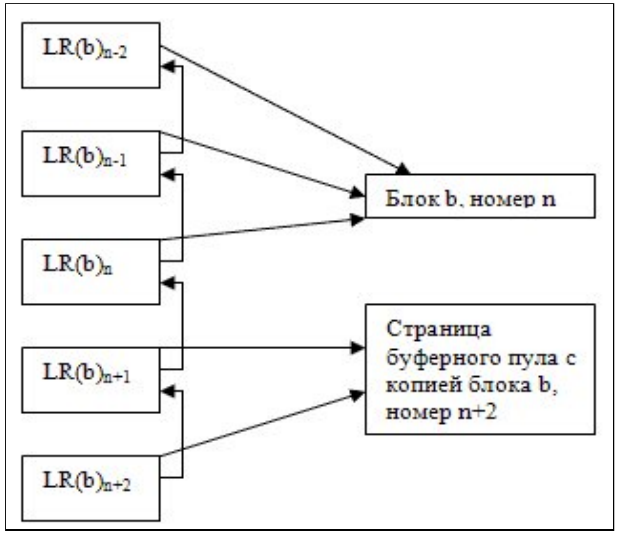


- Это означает, что в состоянии блока отражены результаты операций изменения блока, соответствующих журнальным записям $LR(b)_n$, $LR(b)_{n-1}$ и $LR(b)_{n-2}$

Восстановление после мягкого сбоя (22)

Восстановление физической согласованности базы данных (13)

Журнализация постраничных изменений (5)



- Изменения блока, произведенные операциями, которым соответствуют две хронологически последние журнальные записи $LR(b)_{n+1}$ и $LR(b)_{n+2}$, в его состоянии во внешней памяти не отражены,
  - поскольку не было выполнено выталкивание во внешнюю память страницы буферного пула, содержащей копию блока b
- Поэтому при восстановлении состояния блока требуется выполнить обратные операции изменения блока b,
  - соответствующие журнальным записям $LR(b)_n$, $LR(b)_{n-1}$ и $LR(b)_{n-2}$

Восстановление после мягкого сбоя (23)

Восстановление физической согласованности базы данных (14)

Журнализация постраничных изменений (6)

- В этом подходе имеются два поднаправления
- В первом поднаправлении поддерживается общий журнал логических и страничных операций
- Естественно, наличие двух видов записей,
  - интерпретируемых абсолютно по-разному,
- усложняет структуру журнала
- Кроме того, записи о постраничных изменениях, актуальность которых носит локальный характер, существенно
  - и не очень осмысленно
- увеличивают журнал

Восстановление после мягкого сбоя (24)

Восстановление физической согласованности базы данных (15)

Журнализация постраничных изменений (7)

- Поэтому распространено поддержание отдельного
  - короткого
- журнала постраничных изменений
- Такой журнал обычно называют *физическим* журналом,
  - поскольку он содержит записи об изменении физических объектов – блоков внешней памяти
- В отличие от этого, журнал логических операций принято называть *логическим* журналом,
  - поскольку в нем содержатся записи об операциях над логическими объектами – кортежами

Восстановление после мягкого сбоя (25)

Восстановление физической согласованности базы данных (16)

Журнализация постраничных изменений (8)

- Как уже отмечалось, логический и физический журналы имеют разную природу
- Во-первых, логический журнал должен поддерживать
  - как обратное выполнение журнализованных операций (*undo*),
  - так и их повторное прямое выполнение (*redo*)
- В отличие от этого, от физического журнала требуется только поддержка обратного выполнения постраничных операций

Восстановление после мягкого сбоя (26)

Восстановление физической согласованности базы данных (17)

Журнализация постраничных изменений (9)

- Во-вторых, логический журнал обычно начинает заполняться заново только после выполнения операций резервного копирования базы данных или архивирования самого журнала
  - До этого времени он линейно растет
- Понятно, что в любом случае для размещения журнала выделяется внешняя память ограниченного размера
- Предельный размер журнала определяется администратором базы данных и должен согласовываться с размером интервала времени, через которое производится резервное копирование базы данных

Восстановление после мягкого сбоя (27)

Восстановление физической согласованности базы данных (18)

Журнализация постраничных изменений (10)

- Потенциальное переполнение логического журнала регулируется следующим образом
  - На пути к достижению максимально возможного размера 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
- Естественно, размер «желтой» и «красной» зон логического журнала должен устанавливаться администратором базы данных с учетом максимально допустимого числа одновременно существующих транзакций и их возможной протяженности

Восстановление после мягкого сбоя (28)

Восстановление физической согласованности базы данных (19)

Журнализация постраничных изменений (11)

- В отличие от этого, физический журнал существует сравнительно недолгое время
  - интервал времени между соседними операциями установки точки физической согласованности базы данных
- и, как правило, занимает существенно меньшее дисковое пространство, чем логический журнал
- При выполнении операции установки точки физической согласованности выполняются следующие действия:
  - прекращают инициироваться новые логические операции
  - после завершения всех выполняемых логических операций происходит выталкивание во внешнюю память всех модифицированных страниц буферного пула

Восстановление после мягкого сбоя (29)

Восстановление физической согласованности базы данных (20)

Журнализация постраничных изменений (12)

- формируется и выталкивается во внешнюю память логического журнала специальная запись о точке физически согласованного состояния
- в случае успешного предыдущего действия
  - разрешается инициация новых логических операций,
  - и физический журнал пишется заново

- Предпоследняя операция является атомарной
  - Это опять же запись одного блока на диск
- Если она успешно выполняется,
  - то при следующем восстановлении после мягкого сбоя будет использоваться новая точка физически согласованного состояния,
  - иначе ситуация воспринимается как мягкий сбой с восстановлением логически согласованного состояния базы данных от предыдущей точки физически согласованного состояния
    - с оповещением об этом администратора базы данных

Восстановление базы данных после жесткого сбоя (1)

- Понятно, что для восстановления последнего согласованного состояния базы данных после жесткого сбоя журнала изменений базы данных явно недостаточно
- Самый простой способ восстановления основывается на использовании
  - логического журнала и
  - архивной копии базы данных
- Восстановление начинается с обратного копирования
  - на исправный носитель
- базы данных из архивной копии

Восстановление базы данных после жесткого сбоя (2)

- Затем для всех закончившихся транзакций выполняется *redo*,
  - т.е. операции повторно выполняются в прямом смысле
- Более точно, происходит следующее:
  - по журналу в прямом направлении выполняются все операции
  - для транзакций, которые не закончились к моменту сбоя, выполняется откат
- Очевидно, что после этого будет получено
  - хронологически последнее до момента жесткого сбоя логически согласованное состояние базы данных

Восстановление базы данных после жесткого сбоя (3)

- При некоторой дисциплине выполнения логических операций над базой данных при восстановлении базы данных после жесткого сбоя можно просто последовательно повторно выполнять операции в соответствии с журнальными записями,
  - не обращая внимание на то, в каких транзакциях они выполнялись до жесткого сбоя
- В частности, если сериализация транзакций основывается на блокировках объектов, то эта дисциплина заключается в том, что при выполнении операции в штатном режиме нужно
  - сначала дождаться удовлетворения блокировки изменяемого объекта,
  - затем поместить запись в буфер логического журнала,
  - и только после этого реально выполнять операцию

Восстановление базы данных после жесткого сбоя (4)

- На самом деле, поскольку жесткий сбой не сопровождается утратой буферов оперативной памяти,
  - можно восстановить базу данных до такого уровня,
    - чтобы можно было продолжить даже выполнение незавершенных транзакций
- Но обычно это не делается, потому что восстановление после жесткого сбоя – это достаточно длительный процесс

Восстановление базы данных после жесткого сбоя (5)

- Хотя к ведению журнала предъявляются особые требования по части надежности, в принципе возможна и его утрата
- Тогда единственным способом восстановления базы данных является возврат к архивной копии
- Конечно, в этом случае не удастся получить последнее согласованное состояние базы данных,
  - но это лучше, чем ничего

Восстановление базы данных после жесткого сбоя (6)

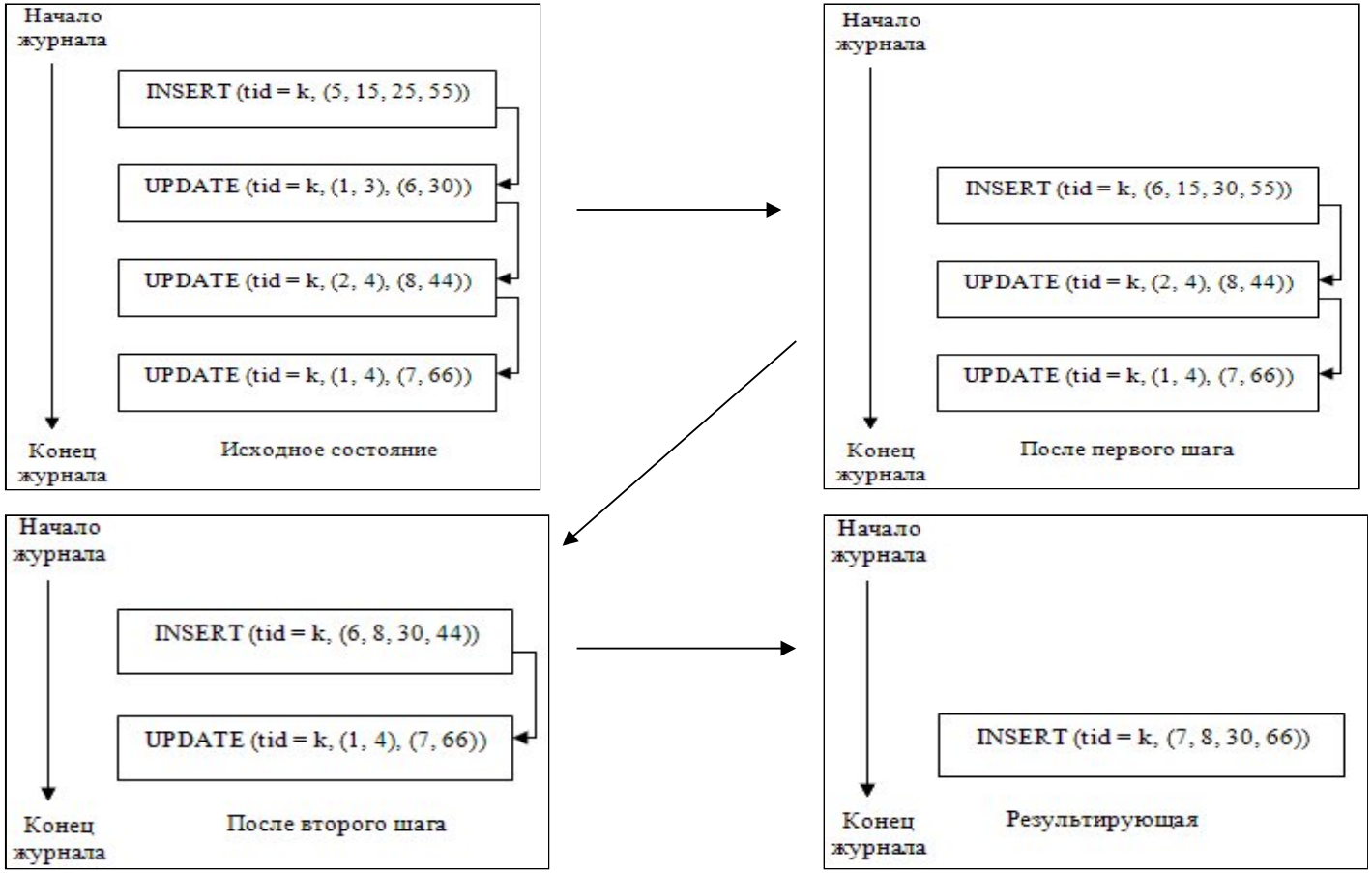
- Последний вопрос, который здесь следует обсудить, касается производства архивных копий базы данных и/или журнала
- Самый простой способ состоит в архивировании базы данных по явному указанию администратора или при переполнении журнала
- Но можно выполнять архивацию базы данных реже, чем переполняется журнал
- Вместо базы данных можно архивировать сам журнал
- В пределе для полного восстановления базы данных после жесткого сбоя достаточно иметь
  - исходную архивную копию базы данных,
  - последовательность архивных копий журналов
  - и последний логический журнал
- Может показаться, что восстановление базы данных на основе таких архивных источников будет занимать недопустимо большое время,
  - однако здесь возможна значительная оптимизация

Восстановление базы данных после жесткого сбоя (7)

- Во-первых, архивированный логический журнал можно сжимать
- 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
- На следующем слайде показан процесс сжатия последовательности журнальных записей, соответствующих последовательности операций над кортежем, у которого
  - tid = *k*
  - и имеются четыре целочисленных поля
- Если хронологически последней в последовательности является запись, соответствующая операции DELETE, то после сжатия этой последовательности она станет пусто

Восстановление базы данных после жесткого сбоя (8)

Процесс сжатия последовательности журнальных записей



Восстановление базы данных после жесткого сбоя (9)

- Во-вторых, точно таким же образом можно совместно сжать два хронологически последовательных полных или сжатых журнала
- Таким образом, для восстановления после жесткого сбоя можно воспользоваться
  - исходной архивной копией,
  - одним сжатым архивным журналом
  - и последним логическим журналом
- Снова могут возникнуть сомнения относительно сложности и продолжительности процесса сжатия журнала
- Но здесь следует заметить, что эта работа может выполняться на отдельном компьютере в режиме offline
- Кроме того, если имеется
  - архивная копия базы данных,
  - сжатый архивный журнал
  - и набор еще не сжатых архивных журналов,
- то этого уже достаточно для восстановления, так что сроки завершения процесса полного сжатия не являются критическими

Заключение

- В этой лекции рассматривались основные принципы и алгоритмы подсистем СУБД, предназначенных для
  - управления буферами основной памяти,
  - журнализации и
  - восстановления базы данных после различных сбоев
- Изложение велось без технических деталей, таких как возможные структуры данных журналов