Методы управления транзакциями. Сихронизационные блокировки, временные метки и версии

С.Д. Кузнецов. Базы данных. Тема 9

План (1)

- Введение
- Общее понятие транзакции и основные характеристики транзакций
  - Атомарность транзакций
  - Транзакции и целостность баз данных
  - Изолированность транзакций
  - Сериализация транзакций

План (2)

- Методы сериализации транзакций
  - Синхронизационные блокировки
  - Синхронизационные тупики, их распознавание и разрушение
  - Метод временных меток
  - Методы сериализации транзакций на основе поддержки версий объектов базы данных
- Заключение

Введение

- Поддержка механизма транзакций – показатель уровня развитости СУБД
- Корректное поддержание транзакций является основой обеспечения целостности баз данных
  - поэтому транзакции вполне уместны и в однопользовательских персональных СУБД
- Составляют базис изолированности пользователей в многопользовательских системах
- Эти два аспекта взаимосвязаны, что и будет показано в этой лекции

Общее понятие транзакции и основные характеристики транзакций (1)

- В современных СУБД поддерживается понятие транзакции, характеризуемое аббревиатурой ACID
  - Atomicy
  - Consistency
  - Isolation
  - Durability
- В соответствии с этим понятием под транзакцией разумеется последовательность операций над базой данных, обладающая следующими свойствами

Общее понятие транзакции и основные характеристики транзакций (2)

- *Атомарность (Atomicy)*
  - Результаты всех операций, успешно выполненных в пределах транзакции, должны быть
    - отражены в состоянии базы данных,
    - либо в состоянии базы данных не должно быть отражено действие ни одной операции
      - речь идет об операциях, изменяющих состояние базы данных
  - Свойство атомарности,
    - которое часто называют свойством “все или ничего”
  - позволяет относиться к транзакции, как к динамически образуемой составной операции над базой данных
    - в общем случае состав и порядок выполнения операций, выполняемых внутри транзакции, становится известным только на стадии выполнения

Общее понятие транзакции и основные характеристики транзакций (3)

- *Согласованность (Consistency)*
- В классическом смысле это свойство означает, что транзакция может быть успешно завершена с *фиксацией* результатов своих операций только в том случае, когда действия операций не нарушают *целостность* базы данных
  - т.е. удовлетворяют набору ограничений целостности, определенных для этой базы данных
- Это свойство расширяется тем, что во время выполнения транзакции разрешается устанавливать точки согласованности и явным образом проверять ограничения целостности.
- В контексте баз данных термины *согласованность* и *целостность* эквивалентны
  - Единственным критерием согласованности данных является их удовлетворение ограничениям целостности
    - т.е. база данных находится в согласованном состоянии тогда и только тогда, когда она находится в целостном состоянии

Общее понятие транзакции и основные характеристики транзакций (4)

- *Изоляция (Isolation)*
- Требуется, чтобы две одновременно (concurrently)
  - параллельно или квазипараллельно
- выполняемые транзакции никоим образом не действовали одна на другую
- Результаты выполнения операций транзакции *T1* не должны быть видны никакой другой транзакции *T2* до тех пор, пока транзакция *T1* не завершится успешным образом

Общее понятие транзакции и основные характеристики транзакций (5)

- *Долговечность (Durability)*
- После успешного завершения транзакции все изменения,
  - которые были внесены в состояние базы данных операциями этой транзакции,
- должны гарантированно сохраняться,
  - даже в случае сбоев аппаратуры или программного обеспечения
- Этому аспекту транзакционных систем посвящается следующая лекция
- Эти свойства особенно важны для многопользовательских систем

Общее понятие транзакции и основные характеристики транзакций (6) Атомарность транзакций (1)

- В этом смысле под транзакцией понимается неделимая с точки зрения воздействия на БД последовательность операторов манипулирования данными
  - (чтения, удаления, вставки, модификации),
- такая, что
  - либо результаты всех операторов, входящих в транзакцию, отображаются в состоянии базы данных,
  - либо воздействие всех этих операторов полностью отсутствует

Общее понятие транзакции и основные характеристики транзакций (7) Атомарность транзакций (2)

- Лозунгом транзакции является «Все или ничего»:
  - при завершении транзакции оператором COMMIT
    - (высокоуровневый аналог операции END TRANSACTION в интерфейсе RSS, см. лекцию 12)
  - результаты гарантированно фиксируются во внешней памяти
    - смысл термина *commit* состоит в запросе «фиксации» результатов транзакции
  - при завершении транзакции оператором ROLLBACK
    - высокоуровневый аналог операции RESTORE в интерфейсе RSS
  - результаты гарантированно отсутствуют во внешней памяти
    - смысл термина *rollback* состоит в запросе ликвидации результатов транзакции
- Каким образом в СУБД поддерживаются индивидуальные откаты транзакций, описывается в следующей

Общее понятие транзакции и основные характеристики транзакций (8) Транзакции и целостность баз данных (1)

- Понятие транзакции имеет непосредственную связь с понятием целостности базы данных
- Часто база данных может обладать такими ограничениями целостности, которые просто невозможно не нарушить, выполняя только один оператор изменения базы данных
- Например, в базе данных СЛУЖАЩИЕ-ОТДЕЛЫ естественным ограничением целостности является совпадение значения атрибута ОТД_РАЗМЕР в кортеже таблицы ОТДЕЛЫ, описывающей данный отдел
  - (например, отдел 625),
- с числом кортежей таблицы СЛУЖАЩИЕ, таких, что значение поля СЛУ_ОТД_НОМЕР равно 625
- Как в этом случае принять на работу в отдел 625 нового сотрудника?
- Независимо от того, какая операция будет выполнена первой,
  - вставка нового кортежа в таблице СОТРУДНИКИ
- или
  - модификация существующего кортежа в отношении ОТДЕЛЫ,
- после выполнения операции база данных окажется в нецелостном состоянии

Общее понятие транзакции и основные характеристики транзакций (9) Транзакции и целостность баз данных (2)

- Для поддержки подобных ограничений целостности допускается их нарушение внутри транзакции с тем условием, чтобы
  - к моменту завершения транзакции условия целостности были соблюдены
- В системах с развитыми средствами ограничения и контроля целостности каждая транзакция
  - начинается при целостном состоянии базы данных
  - и должна оставить это состояние целостными после своего завершения
- Несоблюдение этого условия приводит к тому, что вместо фиксации результатов транзакции происходит ее откат
  - т.е. вместо оператора COMMIT выполняется оператор ROLLBACK, и
- база данных остается в таком состоянии, в котором находилась к моменту начала транзакции,
  - т.е. в целостном состоянии

Общее понятие транзакции и основные характеристики транзакций (10) Транзакции и целостность баз данных (3)

- Более точно, различаются два вида ограничений целостности:
  - немедленно проверяемые и
  - откладываемые.
- К немедленно проверяемым ограничениям целостности относятся такие ограничения, проверку которых бессмысленно или даже невозможно откладывать.
- Примером ограничения, проверку которого откладывать бессмысленно, являются ограничения домена
  - например, возраст сотрудника не может превышать 150 лет
- Более сложным ограничением, проверку которого невозможно отложить, является следующее:
  - зарплата сотрудника не может быть увеличена за одну операцию более чем на 100000 рублей.
- Немедленно проверяемые ограничения целостности соответствуют уровню отдельных операторов языкового уровня СУБД
- При их нарушениях не производится откат транзакции, а лишь отвергается соответствующий оператор

Общее понятие транзакции и основные характеристики транзакций (11) Транзакции и целостность баз данных (4)

- Откладываемые ограничения целостности – это ограничения на базу данных, а не на какие-либо отдельные операции
- По умолчанию такие ограничения проверяются при конце транзакции, и их нарушение вызывает автоматическую замену оператора COMMIT на оператор ROLLBACK
- Однако в некоторых системах поддерживается специальный оператор насильственной проверки ограничений целостности внутри транзакции
- Если после выполнения такого оператора обнаруживается, что условия целостности не выполнены, пользователь может
  - сам выполнить оператор ROLLBACK с откатом транзакции до ее начала или до установленной ранее точки сохранения или
  - постараться устранить причины нецелостного состояния базы данных внутри транзакции
    - видимо, это осмысленно только при использовании интерактивного режима работы

Общее понятие транзакции и основные характеристики транзакций (12) Транзакции и целостность баз данных (5)

- Заметим, что концептуально в момент завершения транзакции проверяются все откладываемые ограничения целостности, определенные в этой базе данных
- Однако в реализации стремятся при выполнении транзакции динамически выделить те ограничения целостности, которые действительно могли бы быть нарушены
- Например, если при выполнении транзакции над базой данных СЛУЖАЩИЕ-ОТДЕЛЫ в ней не выполнялись операторы вставки или удаления кортежей из отношения СЛУЖАЩИЕ, то проверять упоминавшееся выше ограничение целостности не требуется
  - а для проверки подобных ограничений требуется достаточно большая работа

Общее понятие транзакции и основные характеристики транзакций (13) Транзакции и целостность баз данных (6)

- Описанный механизм поддержки целостности баз данных обеспечивает требуемое свойство транзакций:
- никакая транзакция не может быть зафиксирована, если ее действия нарушили целостность базы данных
- Однако в этом подходе имеются два серьезных дефекта
- Во-первых, если при выполнении транзакции не устанавливать точки сохранения и не проверять периодически соответствие текущего состояния базы данных
  - с точки зрения данной транзакции
- ограничениям целостности, то
  - долговременно выполняемая транзакция вполне вероятно может быть «откачена» системой при выполнении завершающего оператора COMMIT
- Это означает непроизводительный расход системных ресурсов и времени пользователей
- Во-вторых, чем длиннее транзакция, модифицирующая состояние базы данных, тем потенциально больше ограничений целостности придется проверять при ее завершении и тем накладнее становится оператор COMMIT

Общее понятие транзакции и основные характеристики транзакций (14) Транзакции и целостность баз данных (7)

- Простое и элегантное решение этой проблемы предлагают Дейт и Дарвен
- Авторы предлагают отказаться от откладываемых ограничений целостности базы данных, а вместо этого ввести составные операторы изменения базы данных
  - нечто наподобие блоков BEGIN … END, поддерживаемых в языках программирования
- После выполнения каждого такого блока
  - или отдельного оператора изменения базы данных, используемого без операторов начала и конца блока
- база данных должна находится в целостном состоянии
- Если составной оператор нарушает ограничение целостности, то он целиком отвергается, и вырабатывается соответствующий код ошибки
- Транзакция в этом случае не откатывается
- Ппри использовании такого подхода при выполнении оператора COMMIT
  - не требуется проверять ограничения целостности,
  - и каждая зафиксированная транзакция будет оставлять базу данных в целостном состоянии

Общее понятие транзакции и основные характеристики транзакций (15) Транзакции и целостность баз данных (8)

- Для реализации описанного подхода не требуются какие-либо новые механизмы, кроме
  - точек сохранения транзакции,
  - насильственной проверки ограничений целостности и
  - частичных откатов транзакций,
- а отмеченные ранее проблемы снимаются
- К сожалению, этот подход на практике пока не применяется

Общее понятие транзакции и основные характеристики транзакций (15) Изолированность транзакций (1)

- В многопользовательских системах с одной базой данных одновременно может работать несколько пользователей или прикладных программ
- Предельной задачей системы является обеспечение изолированности пользователей,
  - т.е. создание достоверной и надежной иллюзии того, что каждый из пользователей работает с базой данных в одиночку

Общее понятие транзакции и основные характеристики транзакций (16) Изолированность транзакций (2)

- В связи со свойством сохранения целостности базы данных транзакции являются подходящими единицами изолированности пользователей
- Действительно, если с каждым сеансом работы пользователя или приложений с базой данных ассоциируется транзакция, то
  - каждый пользователь начинает работу с согласованным состоянием базы данных,
    - т.е. с таким состоянием, в котором база данных могла бы находиться, даже если бы пользователь работал с ней в одиночку

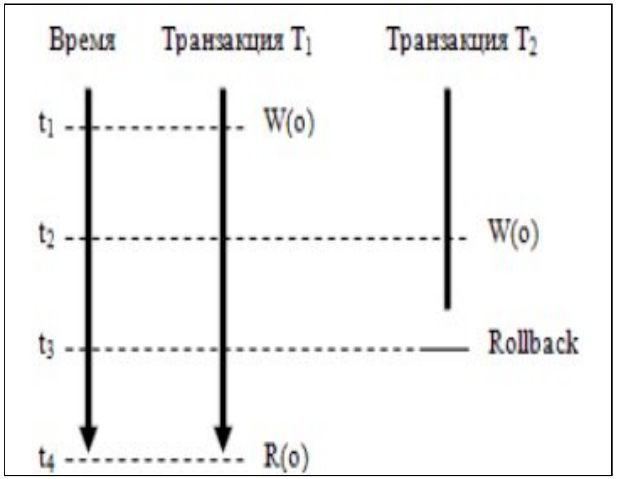
Общее понятие транзакции и основные характеристики транзакций (17) Изолированность транзакций (3)

- При соблюдении обязательного требования поддержки целостности базы данных возможно наличие нескольких уровней изолированности транзакций
- Впервые эти уровни изолированности транзакций были установлены и описаны участниками проекта System R

Общее понятие транзакции и основные характеристики транзакций (18) Изолированность транзакций (4)

Потерянные изменения (1)

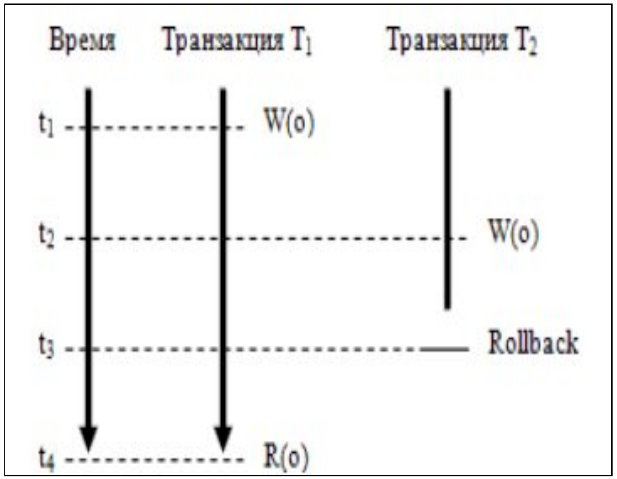
- В момент времени t1 транзакция T1 изменяет объект базы данных o (выполняет операцию W(o))
- До завершения транзакции T1 в момент времени t2 > t1 транзакция T2 также изменяет объект o
- В момент времени t3 > t2 транзакция T2 завершается оператором ROLLBACK (например, по причине нарушения ограничений целостности)



Общее понятие транзакции и основные характеристики транзакций (19) Изолированность транзакций (5)

Потерянные изменения (2)

- Тогда при повторном чтении объекта o
  - выполнении операции R(o)
- в момент времени t4 > t3 транзакция T1 не видит своих изменений этого объекта, произведенных ранее
  - в частности, из-за этого может не удастся фиксация этой транзакции, что, возможно, повлечет потерю изменений у еще одной транзакции и т.д.



Общее понятие транзакции и основные характеристики транзакций (19) Изолированность транзакций (5)

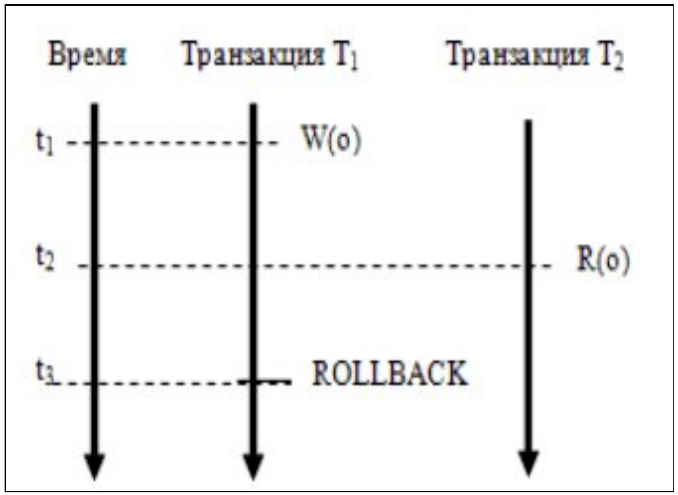
Потерянные изменения (3)

- Такая ситуация называется ситуацией *потерянных изменений*
- Естественно, она противоречит требованию изолированности пользователей
- Чтобы избежать такой ситуации в транзакции T1 требуется, чтобы до завершения транзакции T1 никакая другая транзакция не могла изменять никакой измененный транзакцией T1 объект o
  - в частности, достаточно заблокировать доступ по изменению к объекту o до завершения транзакции T1
- Отсутствие потерянных изменений является минимальным требованием к СУБД при обеспечении изолированности одновременно выполняемых транзакций

Общее понятие транзакции и основные характеристики транзакций (20) Изолированность транзакций (6)

Отсутствие чтения «грязных» данных (1)

- В момент времени t1 транзакция T1 изменяет объект базы данных o
  - выполняет операцию W(o)
- В момент времени t2 > t1 транзакция T2 читает объект o
  - выполняет операцию R(o)



- Поскольку транзакция T1 еще не завершена, транзакция T2 видит несогласованные «грязные» данные
- В частности, в момент времени t3 > t2 транзакция T1 может завершиться откатом
  - например, по причине нарушения ограничений целостности

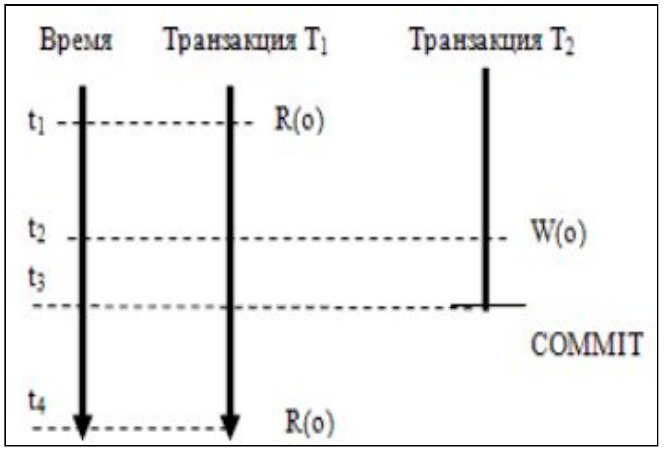
Общее понятие транзакции и основные характеристики транзакций (21) Изолированность транзакций (7)
Отсутствие чтения «грязных» данных (2)

- Эта ситуация тоже не соответствует требованию изолированности пользователей
  - каждый пользователь начинает свою транзакцию при согласованном состоянии базы данных и имеет право видеть только согласованные данные
- Чтобы избежать ситуации чтения "грязных" данных, до завершения транзакции T1, изменившей объект базы данных o, никакая другая транзакция не должна читать объект o
  - например, достаточно заблокировать доступ по чтению к объекту o до завершения изменившей его транзакции T1

Общее понятие транзакции и основные характеристики транзакций (22) Изолированность транзакций (8)

Отсутствие неповторяющихся чтений (1)

- В момент времени t1 транзакция T1 читает объект базы данных o
  - выполняет операцию R(o)
- До завершения транзакции T1 в момент времени t2 > t1 транзакция T2 изменяет объект o
  - выполняет операцию W(o)
- и успешно завершается оператором COMMIT
- В момент времени t3 > t2 транзакция T1 повторно читает объект o и видит его измененное состояние



Общее понятие транзакции и основные характеристики транзакций (23) Изолированность транзакций (9)
Отсутствие неповторяющихся чтений (2)

- Чтобы избежать *неповторяющихся чтений*, до завершения транзакции T1 никакая другая транзакция не должна изменять объект o
  - для этого достаточно заблокировать доступ по записи к объекту o до завершения транзакции T1
- Часто это является максимальным требованием к средствам обеспечения изолированности транзакций, хотя отсутствие неповторяющихся чтений еще не гарантирует реальной изолированности пользователей

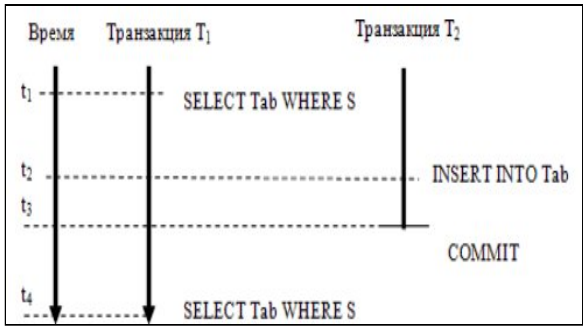
Общее понятие транзакции и основные характеристики транзакций (24) Изолированность транзакций (10)

- Существует возможность обеспечения разных уровней изолированности для разных транзакций, выполняющихся в одной системе баз данных
  - соответствующие операторы были предусмотрены уже в стандарте SQL:1992
- Как уже отмечалось, для корректного соблюдения ограничений целостности достаточен первый уровень
- Существует ряд приложений, которым хватает первого уровня изолированности
  - например, прикладные или системные статистические утилиты, для которых некорректность индивидуальных данных несущественна
- При этом удается существенно сократить накладные расходы СУБД и повысить общую эффективность.

Общее понятие транзакции и основные характеристики транзакций (25) Изолированность транзакций (11)

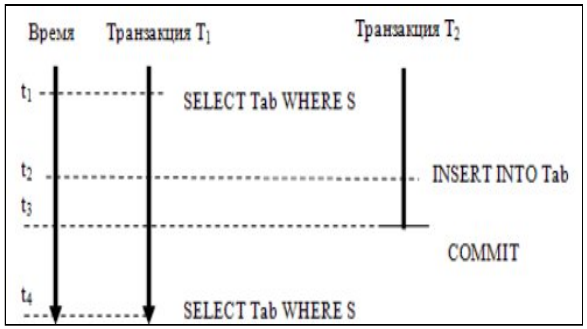
Проблема фантомов (1)

- К более тонким проблемам изолированности транзакций относится так называемая проблема *кортежей-«фантомов»*, приводящая к ситуациям, которые также противоречат изолированности пользователей



- В момент времени t1 транзакция T1 выполняет оператор выборки кортежей таблицы Tab с условием выборки S
  - т.е. выбирается часть кортежей таблицы Tab, удовлетворяющих условию S

Общее понятие транзакции и основные характеристики транзакций (26) Изолированность транзакций (12)
Проблема фантомов (2)



- До завершения транзакции T1 в момент времени t2 > t1 транзакция T2 вставляет в таблицу Tab новый кортеж r, удовлетворяющий условию S, и успешно завершается
- В момент времени t3 > t2 транзакция T1 повторно выполняет тот же оператор выборки, и в результате появляется кортеж, который отсутствовал при первом выполнении оператора.

Общее понятие транзакции и основные характеристики транзакций (27) Изолированность транзакций (13) Проблема фантомов (3)

- Конечно, такая ситуация противоречит идее изолированности транзакций и может возникнуть даже на третьем уровне изолированности транзакций
- Чтобы избежать появления кортежей-фантомов, требуется более высокий «логический» уровень изоляции транзакций
- Идеи требуемого механизма
  - предикатные синхронизационные блокировки
- появились также еще во время выполнения проекта System R, но в большинстве систем не реализованы.

Общее понятие транзакции и основные характеристики транзакций (28) Сериализация транзакций (1)

- Чтобы добиться изолированности транзакций, в СУБД должны использоваться какие-либо методы регулирования совместного выполнения транзакций
- Пусть в системе одновременно выполняется некоторое множество транзакций S = {T1, T2, …, T*n*}
- План (способ) выполнения набора транзакций S
  - в котором, вообще говоря, чередуются или реально параллельно выполняются операции разных транзакций
- называется сериальным, если результат совместного выполнения транзакций эквивалентен результату некоторого последовательного выполнения этих же транзакций (T*i*1, T*i*2, …, T*in*)

Общее понятие транзакции и основные характеристики транзакций (29) Сериализация транзакций (2)

- Сериализация транзакций – это механизм их выполнения по некоторому сериальному плану
- Обеспечение такого механизма является основной функцией компонента СУБД, ответственного за управление транзакциями
- Система, в которой поддерживается сериализация транзакций, обеспечивает реальную изолированность пользователей.
- Основная реализационная проблема состоит в выборе метода сериализации набора транзакций, который не слишком ограничивал бы чередование их операций или реальную параллельность
- Приходящим на ум тривиальным решением является действительно последовательное выполнение транзакций
- Но существуют ситуации, в которых можно выполнять операторы разных транзакций в любом порядке с сохранением свойства сериальности
- Примерами могут служить только читающие транзакции, а также транзакции, не конфликтующие по объектам базы данных

Общее понятие транзакции и основные характеристики транзакций (30) Сериализация транзакций (3)

- Между транзакциями T1 и T2 могут существовать следующие виды конфликтов:
  - W/W – транзакция T2 пытается изменять объект, измененный не закончившейся транзакцией T1
    - наличие такого конфликта может привести к возникновению ситуации потерянных изменений
  - R/W – транзакция T2 пытается изменять объект, прочитанный не закончившейся транзакцией T1
    - наличие такого конфликта может привести к возникновению ситуации неповторяющихся чтений
  - W/R – транзакция T2 пытается читать объект, измененный не закончившейся транзакцией T1
    - наличие такого конфликта может привести к возникновению ситуации «грязного» чтения
- Практические методы сериализации транзакций основываются на учете этих конфликтов

Методы сериализации транзакций (1)

- Существуют два базовых подхода к сериализации транзакций:
  - основанный на синхронизационных захватах объектов базы данных и
  - основанный на использовании временных меток.
- Суть обоих подходов состоит в обнаружении конфликтов транзакций и их устранении
- Ниже мы рассмотрим эти подходы сравнительно подробно
- Кроме того, кратко обсудим возможности использования версий объектов базы данных для ускорения выполнения «только читающих» транзакций,
  - т.е. транзакций, в которых не выполняются операции изменения базы данных

Методы сериализации транзакций (2)

- Предварительно заметим, что для каждого из подходов имеются две разновидности:
  - пессимистическая и
  - оптимистическая.
- При применении пессимистических методов,
  - ориентированных на ситуации, когда конфликты возникают часто,
- конфликты распознаются и разрешаются немедленно при их возникновении
- Оптимистические методы основываются на том, что результаты всех операций модификации базы данных сохраняются в рабочей памяти транзакций
  - Реальная модификация базы данных производится только на стадии фиксации транзакции
  - Тогда же проверяется, не возникают ли конфликты с другими транзакциями

Методы сериализации транзакций (3)

- Далее мы ограничимся рассмотрением более распространенных пессимистических разновидностей методов сериализации транзакций
- Пессимистические методы сравнительно просто трансформируются в свои оптимистические варианты

Методы сериализации транзакций (4)
Синхронизационные блокировки (1)

- Наиболее распространенным в централизованных СУБД
  - включающих системы, основанные на архитектуре «клиент-сервер»
- является подход, основанный на соблюдении двухфазного протокола синхронизационных захватов объектов баз данных
  - Two-Phase Locking Protocol, 2PL
- В общих чертах подход состоит в том, что перед выполнением любой операции в транзакции **T** над объектом базы данных ***o*** от имени транзакции **T** запрашивается синхронизационная блокировка объекта ***o*** в соответствующем режиме
  - в зависимости от вида операции

Методы сериализации транзакций (5)
Синхронизационные блокировки (2)

- Основными режимами синхронизационных блокировок являются следующие:
  - совместный режим – S (Shared)
    - означающий совместную (по чтению) блокировку объекта и требуемый для выполнения операции чтения объекта
  - монопольный режим – X (eXclusive)
    - означающий монопольную (по записи) блокировку объекта и требуемый для выполнения операций вставки, удаления и модификации объекта

Методы сериализации транзакций (6)
Синхронизационные блокировки (3)

- Блокировки одних и тех же объектов по чтению несколькими транзакциями совместимы,
  - т.е. нескольким транзакциям допускается одновременно читать один и тот же объект
- Блокировка объекта одной транзакцией по чтению не совместима с блокировкой другой транзакцией того же объекта по записи,
  - т.е. никакой транзакции нельзя
    - изменять объект, читаемый некоторой транзакцией
      - кроме самой этой транзакции, и
  - никакой транзакции нельзя
    - читать объект, изменяемый некоторой транзакцией
      - кроме самой этой транзакции

Методы сериализации транзакций (7)
Синхронизационные блокировки (4)

- Блокировки одного и того же объекта по записи разными транзакциями не совместимы
  - т.е. никакой транзакции нельзя изменять объект, изменяемый некоторой транзакцией
    - кроме самой этой транзакции

Методы сериализации транзакций (8)

Синхронизационные блокировки (5)

- Правила совместимости захватов одного объекта разными транзакциями приведены в таблице

	X	S
-	да	да
X	нет	нет
S	нет	да

- В первом столбце приведены возможные состояния объекта с точки зрения синхронизационных захватов
- При этом "-" соответствует состоянию объекта, для которого не установлен никакой захват
- Транзакция, запросившая синхронизационный захват объекта БД, уже захваченный другой транзакцией в несовместимом режиме, блокируется до тех пор, пока захват с этого объекта не будет снят

Методы сериализации транзакций (9)

Синхронизационные блокировки (6)

- Заметим, что слово «нет»
  - отсутствие совместимости блокировок

	X	S
-	да	да
X	нет	нет
S	нет	да

- в этой таблице соответствует описанным ранее возможным случаям конфликтов транзакций по доступу к объектам базы данных (W/W, R/W, W/R)
- Совместимость S-блокировок соответствует тому, что конфликт R/R не существует

Методы сериализации транзакций (10)

Синхронизационные блокировки (7)

- Для обеспечения сериализации транзакций
  - третьего уровня изолированности
- синхронизационные блокировки объектов, произведенные по инициативе транзакции, можно снимать только при ее завершении
- Это требование порождает двухфазный протокол синхронизационных захватов – 2PL
- В соответствии с этим протоколом выполнение транзакции разбивается на две фазы:
  - первая фаза транзакции (выполнение операций над базой данных) – накопление блокировок;
  - вторая фаза (фиксация или откат) – снятие блокировок

Методы сериализации транзакций (11)

Синхронизационные блокировки (6)

- Достаточно легко убедиться, что при соблюдении двухфазного протокола синхронизационных блокировок действительно обеспечивается сериализация транзакций на третьем уровне изолированности
- Также легко видеть, что для обеспечения отсутствия потерянных данных достаточно
  - блокировать в режиме X изменяемые объекты базы данных и
  - удерживать эти блокировки до конца транзакции,
- а для обеспечения отсутствия чтения «грязных» данных достаточно
  - блокировать в режиме X изменяемые объекты до конца транзакции и
  - блокировать в режиме S читаемые объекты на время выполнения операции чтения

Методы сериализации транзакций (12)

Синхронизационные блокировки (7)

- Основная проблема состоит в том, что следует считать объектом для синхронизационного захвата?
- В контексте реляционных баз данных возможны следующие альтернативы:
  - файл (сегмент в терминах System R) – физический (с точки зрения базы данных) объект, область хранения нескольких таблиц и, возможно, индексов
  - таблица – логический объект, соответствующий множеству кортежей данной таблицы
  - страница данных – физический объект, хранящий кортежи одной или нескольких таблиц, индексную или служебную информацию
  - кортеж – элементарный физический объект базы данных

Методы сериализации транзакций (13)

Синхронизационные блокировки (8)

- На самом деле, любая операция над объектом базы данных фактически воздействует и на объемлющие его объекты
- Например, операция над кортежем является и операцией
  - над страницей,
- в которой этот кортеж хранится, и
  - над соответствующей таблицей,
  - и над файлом, содержащим таблицу
- Поэтому действительно имеется выбор уровня объекта блокировки.

Методы сериализации транзакций (14)

Синхронизационные блокировки (9)

- Понятно, что для поддержки блокировок требуются системные ресурсы, и что чем крупнее объект синхронизационного захвата
  - неважно, какой природы этот объект – логический или физический,
- тем меньше синхронизационных блокировок будет поддерживаться в системе, и на это, соответственно, будут тратиться меньшие накладные расходы
- Более того, если устанавливать блокировки на уровне файлов или таблиц, то будет решена даже проблема фантомов

Методы сериализации транзакций (15)

Синхронизационные блокировки (10)

- Но при использовании для блокировок крупных объектов возрастает
  - вероятность конфликтов транзакций
- и, тем самым,
  - уменьшается допускаемая степень чередования их операций или реального параллельного выполнения
- Фактически, при укрупнении объекта синхронизационной блокировки мы умышленно огрубляем ситуацию и видим конфликты в тех ситуациях, в которых на самом деле конфликтов нет

Методы сериализации транзакций (16)

Синхронизационные блокировки (11)

- Разработчики многих систем начинали с использования страничных блокировок, полагая это некоторым компромиссом между стремлениями сократить накладные расходы и сохранить достаточно высокий уровень параллельности транзакций
- Но это не очень хороший выбор
- Использование страничных блокировок в двухфазном протоколе иногда вызывает очень неприятные синхронизационные проблемы, усложняющие организацию СУБД
  - эти проблемы связаны с тем, что страницы приходится блокировать на двух разных уровнях
    - уровне управления буферами страниц в основной памяти и
    - уровне выполнения логических операций
- В большинстве современных систем используются покортежные синхронизационные блокировки

Методы сериализации транзакций (17)

Синхронизационные блокировки (12)

- Но при этом возникает очередной вопрос
- Если единицей блокировки является кортеж, то какие синхронизационные блокировки потребуются при выполнении таких операций как уничтожение заполненной таблицы?
- Было бы довольно нелепо перед выполнением такой операции потребовать блокировки всех существующих кортежей таблицы
- Кроме того, это не предотвратило бы возможности параллельной вставки нового кортежа в уничтожаемое отношение в некоторой другой транзакции

Методы сериализации транзакций (18)
Синхронизационные блокировки (13)
Гранулированные синхронизационные блокировки (1)

- Подобные рассуждения привели к разработке механизма гранулированных синхронизационных блокировок
- При применении этого подхода синхронизационные блокировки могут запрашиваться по отношению к объектам разного уровня:
  - файлам, таблицам и кортежам
- требуемый уровень объекта определяется тем, какая операция выполняется
  - например, для выполнения операции уничтожения таблицы объектом синхронизационной блокировки должна быть вся таблица, а для выполнения операции удаления кортежа – этот кортеж
- Объект любого уровня может быть заблокирован в режиме S или X

Методы сериализации транзакций (19)
Синхронизационные блокировки (14)
Гранулированные синхронизационные блокировки (2)

- Для согласования блокировок разного уровня вводятся специальный протокол гранулированных блокировок и новые типы блокировок
- Коротко говоря, перед установкой блокировки на некоторый объект базы данных в режиме S или X соответствующий объект верхнего уровня должен быть заблокирован в режиме IS, IX или SIX
- Что же собой представляют эти режимы блокировок?

Методы сериализации транзакций (20)
Синхронизационные блокировки (15)
Гранулированные синхронизационные блокировки (3)

- Блокировка в режиме IS
  - Intented for Shared lock
- некоторого составного объекта o базы данных означает намерение заблокировать некоторый объект o', входящий в o, в совместном режиме
  - режиме S
- Например, при намерении читать кортежи из таблицы Tab эта таблица должна быть заблокирована в режиме IS
  - а до этого в таком же режиме должен быть заблокирован файл, в котором располагается таблица Tab

Методы сериализации транзакций (21)
Синхронизационные блокировки (16)
Гранулированные синхронизационные блокировки (4)

- Блокировка в режиме IX
  - Intented for eXclusive lock
- некоторого составного объекта o базы данных означает намерение заблокировать некоторый объект o', входящий в o, в монопольном режиме
  - режиме X
- Например, для удаления кортежей из таблицы Tab эта таблица должна быть заблокирована в режиме IX
  - а до этого в таком же режиме должен быть заблокирован файл, в котором располагается таблица Tab

Методы сериализации транзакций (22)
Синхронизационные блокировки (17)
Гранулированные синхронизационные блокировки (5)

- Блокировка в режиме SIX
  - Shared, Intented for eXclusive lock
- некоторого составного объекта о базы данных означает совместную блокировку всего этого объекта с намерением впоследствии блокировать какие-либо входящие в него объекты в монопольном режиме
  - режиме X
- Например, если выполняется длинная операция просмотра таблицы Tab с возможностью удаления некоторых просматриваемых кортежей, то экономичнее всего заблокировать таблицу Tab в режиме SIX
  - а до этого заблокировать в режиме IS файл, в котором располагается таблица Tab

Методы сериализации транзакций (23)
Синхронизационные блокировки (18)
Гранулированные синхронизационные блокировки (6)

- Таблица совместимости блокировок S, X, IS, IX и SIX
- Поясним правила совместимости
- Для атомарных объектов разумны только блокировки в режимах S и X, для которых правила совместимости остаются такими же, как прежде
- Пусть теперь *o* – это некоторый составной объект

	X	S	IX	IS	SIX
-	да	да	да	да	да
X	нет	нет	нет	нет	нет
S	нет	да	нет	да	нет
IX	нет	нет	да	да	нет
IS	нет	да	да	да	да
SIX	нет	нет	нет	да	нет

Методы сериализации транзакций (24)
Синхронизационные блокировки (19)
Гранулированные синхронизационные блокировки (7)

	X	S	IX	IS	SIX
-	да	да	да	да	да
X	нет	нет	нет	нет	нет
S	нет	да	нет	да	нет
IX	нет	нет	да	да	нет
IS	нет	да	да	да	да
SIX	нет	нет	нет	да	нет

- Тогда блокировка объекта *o* в режиме X в транзакции T1 не совместима с блокировкой этого объекта в режимах X, S, IX, IS или SIX в транзакции T2
- Действительно, блокировка объекта *o* в режиме X в транзакции T1 направлена на то, чтобы изменять объект *o* целиком
- Несовместимость блокировки объекта *o* в режиме X в транзакции T1 с его блокировкой в режиме X или IX в транзакции T2 устраняет конфликты транзакций T1 и T2 вида W/W

Методы сериализации транзакций (25)
Синхронизационные блокировки (20)
Гранулированные синхронизационные блокировки (8)

- Несовместимость блокировки объекта *o* в режиме X в транзакции T1 с его блокировкой в режиме S или IS в транзакции T2 устраняет конфликты транзакций T1 и T2 вида W/R

	X	S	IX	IS	SIX
-	да	да	да	да	да
X	нет	нет	нет	нет	нет
S	нет	да	нет	да	нет
IX	нет	нет	да	да	нет
IS	нет	да	да	да	да
SIX	нет	нет	нет	да	нет

- Наконец, несовместимость блокировки объекта *o* в режиме X в транзакции T1 с его блокировкой в режиме SIX в транзакции T2 устраняет конфликты транзакций T1 и T2 вида W/R и W/W

Методы сериализации транзакций (26)
Синхронизационные блокировки (21)
Гранулированные синхронизационные блокировки (9)

	X	S	IX	IS	SIX
-	да	да	да	да	да
X	нет	нет	нет	нет	нет
S	нет	да	нет	да	нет
IX	нет	нет	да	да	нет
IS	нет	да	да	да	да
SIX	нет	нет	нет	да	нет

- Блокировка объекта *o* в режиме S в транзакции T1 совместима с блокировкой этого объекта в режимах S или IS в транзакции T2,
  - поскольку эти блокировки в транзакциях T1 и T2 направлены только на то, чтобы только читать некоторые объекты *o*', входящие в *o*

Методы сериализации транзакций (27)
Синхронизационные блокировки (22)
Гранулированные синхронизационные блокировки (10)

- Блокировка объекта *o* в режиме S в транзакции T1 не совместима с блокировкой этого объекта в режимах X, IX или SIX в транзакции T2,
  - поскольку любая из этих блокировок направлена на то, чтобы изменять в транзакции T2 объект o целиком или какой-либо объект *o*', входящий в o

	X	S	IX	IS	SIX
-	да	да	да	да	да
X	нет	нет	нет	нет	нет
S	нет	да	нет	да	нет
IX	нет	нет	да	да	нет
IS	нет	да	да	да	да
SIX	нет	нет	нет	да	нет

- Несовместимость блокировки объекта *o* в режиме S в транзакции T1 с блокировкой этого объекта в режимах X, IX или SIX в транзакции T2, тем самым, устраняет конфликты транзакций T1 и T2 вида R/W

Методы сериализации транзакций (28)
Синхронизационные блокировки (23)
Гранулированные синхронизационные блокировки (11)

	X	S	IX	IS	SIX
-	да	да	да	да	да
X	нет	нет	нет	нет	нет
S	нет	да	нет	да	нет
IX	нет	нет	да	да	нет
IS	нет	да	да	да	да
SIX	нет	нет	нет	да	нет

- Блокировка объекта *o* в режиме IX в транзакции T1 совместима с блокировкой этого же объекта в режимах IS или IX в транзакции T2
- Действительно, блокировка объекта *o* в режиме IX в транзакции T1 направлена на то, чтобы в этой транзакции изменять какой-либо объект *o*', входящий в *o*, а блокировка этого же объекта в режиме IS в транзакции T2 – на то, чтобы читать в транзакции T2 какой-либо объект *o*", входящий в *o*

Методы сериализации транзакций (29)
Синхронизационные блокировки (24)
Гранулированные синхронизационные блокировки (12)

- Если объекты *o*' и *o*'' – разные, то конфликт транзакций T1 и T2 не возникнет. Если *o*' = *o*'', то перед изменением этот объект будет заблокирован в транзакции T1 в режиме X, а перед чтением – в транзакции T2 в режиме S

	X	S	IX	IS	SIX
-	да	да	да	да	да
X	нет	нет	нет	нет	нет
S	нет	да	нет	да	нет
IX	нет	нет	да	да	нет
IS	нет	да	да	да	да
SIX	нет	нет	нет	да	нет

- Несовместимость этих блокировок позволит избежать конфликта транзакций T1 и T2 вида W/R, и для этого не требуется несовместимость блокировок IX и IS объекта *o*
- Аналогично обосновывается совместимость блокировок IX и IX

Методы сериализации транзакций (30)
Синхронизационные блокировки (25)
Гранулированные синхронизационные блокировки (13)

	X	S	IX	IS	SIX
-	да	да	да	да	да
X	нет	нет	нет	нет	нет
S	нет	да	нет	да	нет
IX	нет	нет	да	да	нет
IS	нет	да	да	да	да
SIX	нет	нет	нет	да	нет

- Блокировка IX не совместима с блокировкой S, поскольку иначе мог бы проявиться конфликт транзакций T1 и T2 вида W/R
- Блокировка IX не совместима с блокировкой X, поскольку иначе мог бы проявиться конфликт транзакций T1 и T2 вида W/W
- Наконец, блокировка IX не совместима с блокировкой SIX, поскольку иначе мог бы проявиться конфликт транзакций T1 и T2 вида W/R или W/W

Методы сериализации транзакций (31)
Синхронизационные блокировки (26)
Гранулированные синхронизационные блокировки (14)

	X	S	IX	IS	SIX
-	да	да	да	да	да
X	нет	нет	нет	нет	нет
S	нет	да	нет	да	нет
IX	нет	нет	да	да	нет
IS	нет	да	да	да	да
SIX	нет	нет	нет	да	нет

- Блокировка объекта *o* в режиме IS в транзакции T1 совместима с блокировкой этого же объекта в режимах S, IS, IX или SIX в транзакции T2
- Совместимость с блокировкой в режиме S или IS уже обосновывалась
- Покажем, что блокировка объекта *o* в режиме IS в транзакции T1 совместима с блокировкой того же объекта в режиме IX в транзакции T2
- Действительно, блокировка объекта *o* в режиме IS в транзакции T1 направлена на то, чтобы в этой транзакции
  - читать какой-либо объект *o'*, входящий в *o*,
- а блокировка этого же объекта в режиме IX в транзакции T2 – на то, чтобы в транзакции T2
  - изменять какой-либо объект *o''*, входящий в *o*

Методы сериализации транзакций (32)
Синхронизационные блокировки (27)
Гранулированные синхронизационные блокировки (15)

- Если объекты *o*' и *o*'' – разные, то конфликт транзакций не возникнет
- Если *o*' = *o*'', то перед чтением этот объект будет заблокирован в транзакции T1 в режиме S, а перед изменением – в транзакции T2 в режиме X
- Несовместимость этих блокировок позволит избежать конфликта транзакций T1 и T2 вида R/W, и для этого не требуется несовместимость блокировок IS и IX объекта *o*
- Аналогично можно показать совместимость блокировок IS и SIX
- Несовместимость блокировок IS и X очевидна, поскольку иначе мог бы проявиться конфликт транзакций T1 и T2 вида R/W

	X	S	IX	IS	SIX
-	да	да	да	да	да
X	нет	нет	нет	нет	нет
S	нет	да	нет	да	нет
IX	нет	нет	да	да	нет
IS	нет	да	да	да	да
SIX	нет	нет	нет	да	нет

Методы сериализации транзакций (33)
Синхронизационные блокировки (28)
Гранулированные синхронизационные блокировки (16)

- Блокировка объекта *o* в режиме SIX в транзакции T1 позволяет этой транзакции
  - читать любой объект *o'*, входящий в *o*, без его дополнительной блокировки и
  - изменять любой объект *o'*, входящий в *o*, с его предварительной блокировкой в режиме X

	X	S	IX	IS	SIX
-	да	да	да	да	да
X	нет	нет	нет	нет	нет
S	нет	да	нет	да	нет
IX	нет	нет	да	да	нет
IS	нет	да	да	да	да
SIX	нет	нет	нет	да	нет

- Эта блокировка совместима с блокировкой объекта *o* в режиме IS в транзакции T2
- Действительно, блокировка объекта *o* в режиме IS в транзакции T2 направлена на то, чтобы в транзакции T2 читать какой-либо объект *o'*, входящий в *o*
- Перед этим в транзакции T2 должна быть установлена блокировка объекта *o'* в режиме S

Методы сериализации транзакций (34)
Синхронизационные блокировки (29)
Гранулированные синхронизационные блокировки (17)

- К этому моменту у объекта *o'* может отсутствовать явная блокировка, установленная в транзакции T1, что, в соответствии с семантикой блокировки SIX, означает наличие неявной блокировки *o'* по чтению
- Очевидно, что в этом случае конфликт транзакций T1 и T2 не возникает
- К этому же моменту у объекта *o'* может иметься блокировка в режиме X, установленная в транзакции T1
- В этом случае запрос блокировки объекта *o'* в режиме S удовлетворен не будет, и конфликт транзакций T1 и T2 вида W/R будет предотвращен без потребности в несовместимости блокировок SIX и IS

	X	S	IX	IS	SIX
-	да	да	да	да	да
X	нет	нет	нет	нет	нет
S	нет	да	нет	да	нет
IX	нет	нет	да	да	нет
IS	нет	да	да	да	да
SIX	нет	нет	нет	да	нет

Методы сериализации транзакций (35)
Синхронизационные блокировки (30)
Гранулированные синхронизационные блокировки (18)

	X	S	IX	IS	SIX
-	да	да	да	да	да
X	нет	нет	нет	нет	нет
S	нет	да	нет	да	нет
IX	нет	нет	да	да	нет
IS	нет	да	да	да	да
SIX	нет	нет	нет	да	нет

- Блокировка объекта *o* в режиме SIX в транзакции T1 не совместима с блокировкой объекта *o* в режиме X в транзакции T2,
  - поскольку иначе мог бы проявиться конфликт транзакций T1 и T2 вида R/W
- Блокировка объекта *o* в режиме SIX в транзакции T1 не совместима с блокировкой объекта o в режиме S или IS в транзакции T2
  - поскольку иначе мог бы проявиться конфликт транзакций T1 и T2 вида W/R при доступе к некоторым объектам *o*', входящим в o
- Наконец, блокировка объекта *o* в режиме SIX в транзакции T1 не совместима с блокировкой объекта *o* в режиме IX или SIX в транзакции T2,
  - поскольку иначе мог бы проявиться конфликт транзакций T1 и T2 вида R/W при доступе к некоторым объектам *o*', входящим в *o*

Методы сериализации транзакций (36)
Синхронизационные блокировки (31)
Предикатные синхронизационные блокировки (1)

- Несмотря на привлекательность метода гранулированных синхронизационных захватов, следует отметить, что он не решает проблему фантомов
  - если, конечно, не ограничиться использованием блокировок таблиц в режимах S и X
- Давно известно, что для решения этой проблемы необходимо перейти от блокировок
  - индивидуальных («физических») объектов базы данных,
- к блокировке
  - условий (предикатов), которым удовлетворяют эти объекты
- Проблема фантомов не возникает при использовании для блокировок уровня таблиц именно потому, что таблица как логический объект
  - представляет собой неявное условие для входящих в него кортежей
- Блокировка таблицы – это простой и частный случай предикатной блокировки

Методы сериализации транзакций (37)
Синхронизационные блокировки (32)
Предикатные синхронизационные блокировки (2)

- Поскольку любая операция над реляционной базой данных задается некоторым условием
  - т.е. в ней указывается не конкретный набор объектов базы данных, над которыми нужно выполнить операцию, а условие, которому должны удовлетворять объекты этого набора,
- идеальным выбором было бы требовать синхронизационную блокировку в режиме S или X именно этого условия
- Но если посмотреть на общий вид условий, допускаемых, например, в языке SQL, то становится абсолютно непонятно,
  - как определить совместимость двух предикатных блокировок
- Ясно, что без этого использовать предикатные блокировки для сериализации транзакций невозможно, а в общей форме проблема неразрешима

Методы сериализации транзакций (38)
Синхронизационные блокировки (33)
Предикатные синхронизационные блокировки (3)

- Один из компромиссных подходов предлагался участниками проекта System R
- Подход основывался на том, что при открытии сканирования таблицы по индексу в RSS передается дополнительная информация
  - диапазон сканирования,
- которая ограничивает множество кортежей, среди которых не должны возникать фантомы.
- Опираясь на наличие этой информации, предлагалось ввести в систему блокировок System R элементы предикатных блокировок
- В System R блокировки сегментов (файлов), таблиц и кортежей технически трактовались единообразно, как блокировки идентификаторов кортежей (tid'ов)
  - При блокировке кортежа на самом деле блокировался его tid
  - При блокировке сегмента или таблицы на самом деле блокировался tid описателя соответствующего объекта во внутренних таблицах-каталогах сегментов или таблиц

Методы сериализации транзакций (39)
Синхронизационные блокировки (34)
Предикатные синхронизационные блокировки (4)

- Предлагалось расширить систему синхронизации, разрешив применять блокировки к паре
  - «идентификатор индекса, интервал значений ключа этого индекса»
- К такой паре можно было применять блокировки в любом из допустимых режимов, причем две такие блокировки считались совместимыми в том и только в том случае, если они были совместимы в соответствии с таблицей совместимости или указанные диапазоны значений ключей не пересекались.
- При наличии такой возможности, если открывается сканирование таблицы через индекс, то таблица блокируется в режиме IS, и в этом же режиме блокируется пара «идентификатор индекса, диапазон сканирования»
- При занесении (удалении) кортежа таблица блокируется в режиме IX, и в этом же режиме для каждого индекса, определенного на данной таблице отношении, блокируется пара «идентификатор индекса, значение ключа из затрагиваемого операцией кортежа»
- Это позволяет избежать конфликтов читающих транзакций с теми изменяющими транзакциями, которые затрагивают диапазоны сканирования читающих транзакций

Методы сериализации транзакций (40)
Синхронизационные блокировки (35)
Предикатные синхронизационные блокировки (5)

- При этом решается проблема фантомов, и параллельность транзакций ограничивается «по существу», т.е. только в тех случаях, когда их параллельное выполнение создает проблемы
- Однако описанное решение проблемы фантомов далеко от идеального
- Во-первых, по-прежнему при сканировании таблиц без использования индексов отсутствие фантомов можно гарантировать только при блокировке всего отношения в режиме S
- Во-вторых, даже при сканировании по индексу условие реальной выборки кортежа часто может быть гораздо строже простого указания диапазона сканирования, а это значит, что
  - блокировка этого диапазона будет слишком сильной, т.е. затронет более широкое множество кортежей, чем то, которое будет реальным результатом сканирования

Методы сериализации транзакций (41)
Синхронизационные блокировки (36)
Предикатные синхронизационные блокировки (6)

- Известно следующее более совершенное решение. Будем называть простым условием конъюнкцию простых предикатов сравнения, имеющих вид имя_поля { = > < } значение
- В типичных СУБД в интерфейсе подсистемы управления памятью допускаются только простые условия
- Подсистема языкового уровня производит компиляцию оператора SQL со сложным условием в последовательность обращений к подсистеме управления памятью, в каждом из которых содержатся только простые условия
- Более точно, простое условие явно указывается в операции открытия сканирования таблицы
  - напрямую или через индекс
    - в последнем случае оно конъюнктивно соединяется с условием, задаваемым диапазоном сканирования

Методы сериализации транзакций (42)
Синхронизационные блокировки (37)
Предикатные синхронизационные блокировки (7)

- Кроме того, при открытии сканирования всегда можно указать, для какой цели оно будет использоваться: для выборки кортежей, для их удаления или для их обновления
  - это известно компилятору SQL
- Кроме того, неявные условия задаются операциями вставки и удаления кортежей
  - конъюнктивное логическое выражение, состоящее из простых предикатов вида имя_поля = значение для всех полей таблицы,
- а также операциями обновления кортежей
  - конъюнктивное логическое выражение, состоящее из простых предикатов вида имя_поля = значение для всех обновляемых полей таблицы
- Поэтому в случае типовой организации SQL-ориентированной СУБД простые условия можно использовать как основу предикатных захватов

Методы сериализации транзакций (43)
Синхронизационные блокировки (38)
Предикатные синхронизационные блокировки (8)

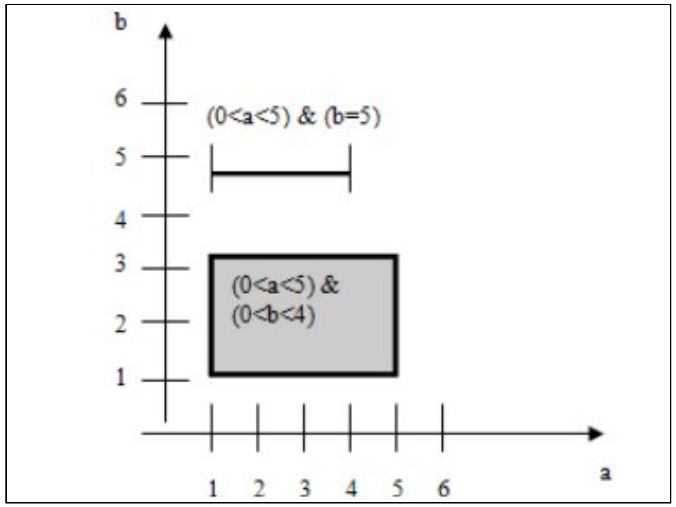
- Для простых условий совместимость предикатных блокировок легко определяется на основе следующей геометрической интерпретации
- Пусть Tab – таблица с полями a1, a2, ..., a*n*, a m1, m2, ..., m*n* – множества допустимых значений a1, a2, ..., a*n* соответственно
  - естественно, все эти множества – конечные
- Тогда можно сопоставить Tab конечное *n*-мерное пространство возможных значений кортежей Tab
- Легко видеть, что любое простое условие, представляющее собой конъюнкцию простых предикатов, «вырезает» в этом пространстве *k*-мерный прямоугольник ($k \le n$)

Методы сериализации транзакций (44)
Синхронизационные блокировки (39)
Предикатные синхронизационные блокировки (9)

- Достаточно очевидно следующее утверждение:
- Пусть имеются два простых условия scond1 и scond2
- Пусть транзакция T1 запрашивает блокировку scond1, а транзакция T2 – scond2 в режимах, которые
  - были бы несовместимы, если бы scond1 и scond2 являлись не условиями, а объектами базы данных (S-X, X-S, X-X)
- Эти блокировки совместимы в том и только в том случае, когда прямоугольники, соответствующие scond1 и scond2, не пересекаются
  - каждому *k*-мерному прямоугольнику в *n*-мерном пространстве возможных значений кортежей Tab соответствует некоторое подмножество возможных значений кортежей, и
    - отсутствие пересечения у двух прямоугольников гарантирует отсутствие конфликтов транзакций

Методы сериализации транзакций (45)
Синхронизационные блокировки (40)
Предикатные синхронизационные блокировки (10)

- В каких бы режимах не требовала транзакция T1 блокировки условия
  - (0 < a < 5) & (b = 5),
- а транзакция T2 – блокировки условия
  - (0 < a <6) & (0 < b <4),
- эти блокировки всегда будут совместимы

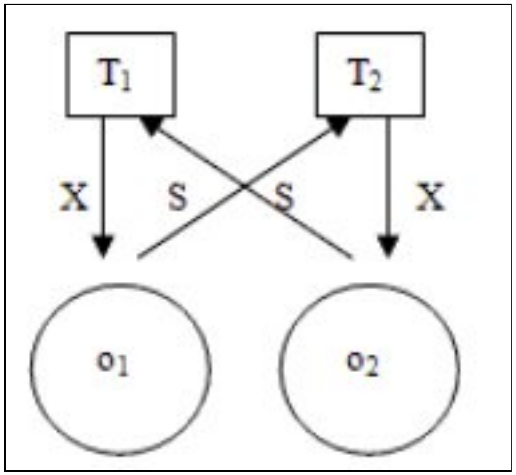


Методы сериализации транзакций (46)
Синхронизационные блокировки (41)
Предикатные синхронизационные блокировки (11)

- При поддержке такой системы блокировок простых условий можно обойтись без гранулированных блокировок
- В частности, чтобы гарантированно заблокировать таблицу целиком, достаточно заблокировать условие
  - $\&_{1 \le i \le n}$ (min(m*i*) < имя_поля*i* < max(m*i*))
- Чтобы заблокировать базу данных, достаточно заблокировать условие, являющееся конъюнкцией условий блокировки всех таблиц этой базы данных.
- Блокировки простых условий описываются таблицами, немногим отличающимися от таблиц традиционных синхронизаторов с гранулированными блокировками
- Поэтому введение в СУБД механизма предикатных блокировок не приводит к значительным усложнениям

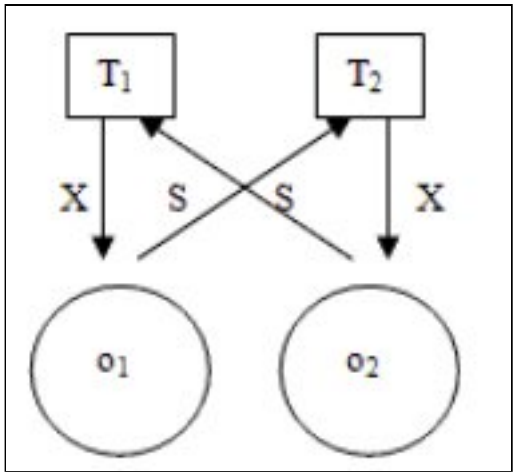
Методы сериализации транзакций (47)
Синхронизационные тупики, их распознавание и разрушение (1)

- Одним из наиболее чувствительных недостатков метода сериализации транзакций на основе синхронизационных блокировок является возможность возникновение тупиков (deadlocks) между транзакциями
- Синхронизационные тупики возможны при применении любого из рассмотренных выше вариантов механизмов блокировок
- На рисунке показан простой сценарий возникновения синхронизационного тупика между транзакциями T1 и T2



Методы сериализации транзакций (48)
Синхронизационные тупики, их распознавание и разрушение (2)

- Транзакции T1 и T2 устанавливают монопольные блокировки объектов *o*1 и *o*2 соответственно
- после этого T1 требуется совместная блокировка объекта *o*2, а T2 – совместная блокировка объекта *o*1;
- ни одно из этих требований блокировки не может быть удовлетворено, следовательно,
  - ни одна из транзакций не может продолжаться;
  - поэтому монопольные блокировки объектов никогда не будут сняты, а
  - требования совместных блокировок не будут удовлетворены
- Поскольку тупики возможны, и никакого естественного выхода из тупиковой ситуации не существует, то эти ситуации необходимо обнаруживать и искусственно устранять

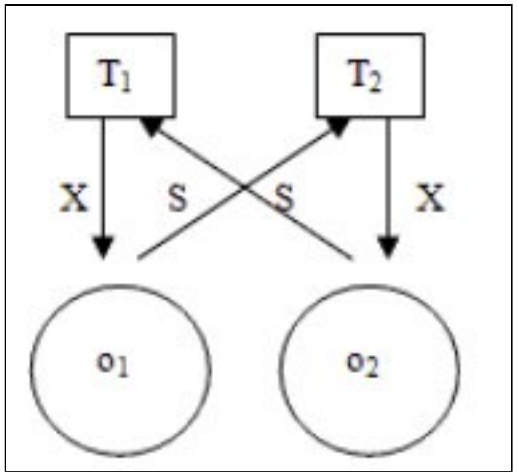


Методы сериализации транзакций (49)

Синхронизационные тупики, их распознавание и разрушение (3)

Обнаружение тупиковых ситуаций (1)

- Основой обнаружения тупиковых ситуаций является построение
  - или постоянное поддержание
- графа ожидания транзакций
- Граф ожидания транзакций – это ориентированный двудольный граф, в котором существует два типа вершин – вершины, соответствующие транзакциям
  - будем изображать их прямоугольниками
- и вершины, соответствующие объектам блокировок
  - будем изображать их окружностями
- В этом графе дуги соединяют только вершины-транзакции с вершинами-объектами.

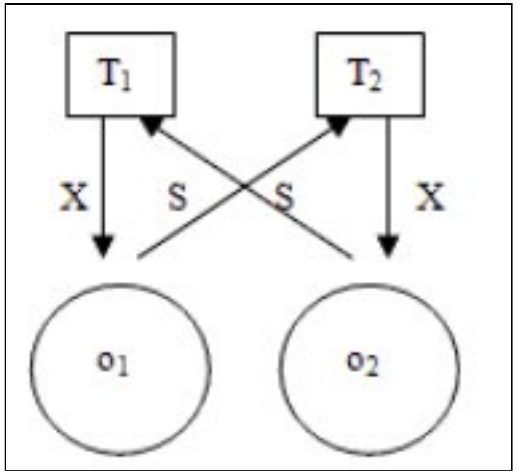


Методы сериализации транзакций (50)

Синхронизационные тупики, их распознавание и разрушение (4)

Обнаружение тупиковых ситуаций (2)

- Дуга из вершины-транзакции к вершине-объекту существует в том и только в том случае, если для этой транзакции имеется удовлетворенная блокировка данного объекта
- Дуга из вершины-объекта к вершине-транзакции существует тогда и только тогда, когда эта транзакция ожидает удовлетворения запроса блокировки данного объекта
- Легко показать, что в системе существует тупиковая ситуация в том и только в том случае, когда в графе ожидания транзакций имеется хотя бы один цикл.

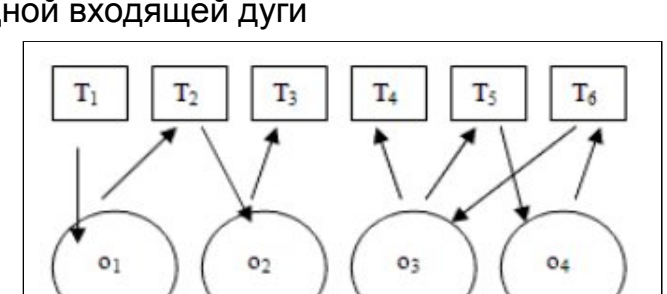


Методы сериализации транзакций (51)
Синхронизационные тупики, их распознавание и разрушение (5)
Обнаружение тупиковых ситуаций (3)

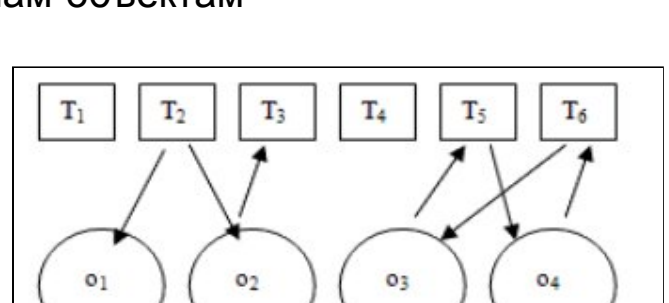
- Для распознавания тупиковых ситуаций периодически производится построение графа ожидания транзакций
  - как уже отмечалось, иногда граф ожидания поддерживается постоянно,
- и в этом графе ищутся циклы
- Традиционной техникой
  - для которой существует множество разновидностей
- нахождения циклов в ориентированном графе является редукция графа

Методы сериализации транзакций (52)
Синхронизационные тупики, их распознавание и разрушение (6)
Обнаружение тупиковых ситуаций (4)

- В целях упрощения примера предполагается, что все блокировки являются монопольными,
  - т.е. для каждой вершины-объекта имеется не более одной входящей дуги
- Прежде всего, из графа ожидания удаляются все дуги, исходящие из вершин-транзакций, в которые не входят дуги из вершин-объектов
  - Это основывается на том разумном предположении, что транзакции, не ожидающие удовлетворения запроса блокировок, могут успешно завершиться и освободить блокировки



- Кроме того, удаляются дуги, входящие в вершины-транзакции, из которых не исходят, ведущие к вершинам-объектам
  - транзакции, ожидающие удовлетворения блокировок, но не удерживающие заблокированные объекты, не могут быть причиной тупика



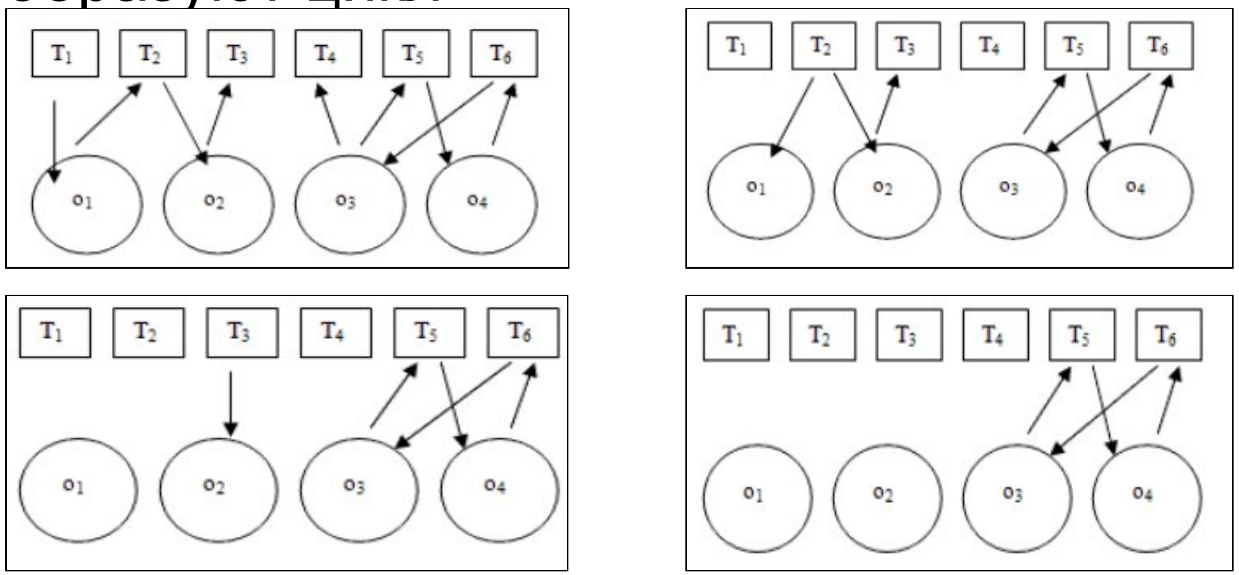
- Для тех вершин-объектов, для которых не осталось входящих дуг, но существуют исходящие, ориентация одной из исходящих дуг
  - выбираемой произвольным образом
- изменяется на противоположную это моделирует удовлетворение запроса блокировки

Методы сериализации транзакций (53)

Синхронизационные тупики, их распознавание и разрушение (7)

Обнаружение тупиковых ситуаций (5)

- После этого снова повторяются описанные действия
  - и так до тех пор, пока не прекратится удаление дуг
- Если в графе остались дуги, то они обязательно образуют цикл



Методы сериализации транзакций (54)
Синхронизационные тупики, их распознавание и разрушение (8)
Разрушение тупиков (1)

- Предположим теперь, что нам удалось найти цикл в графе ожидания транзакций
- Что делать теперь?
- Нужно каким-то образом обеспечить возможность продолжения работы хотя бы для части транзакций, попавших в тупик
- Разрушение тупика начинается с выбора в цикле транзакций так называемой транзакции-жертвы,
  - т.е. транзакции, которой решено пожертвовать, чтобы обеспечить возможность продолжения работы других транзакций

Методы сериализации транзакций (55)

Синхронизационные тупики, их распознавание и разрушение (9)

Разрушение тупиков (2)

- Выбрать «жертву» не так уж легко, поскольку для этого могут использоваться различные, зачастую противоречивые критерии
- С одной стороны, было бы разумно жертвовать наиболее «богатой» транзакцией, т.е. той транзакцией, которая удерживает наиболее число блокировок объектов
- В этом случае после принудительно завершения такой транзакции освободилось бы наибольшее число объектов, что с большой вероятностью привело бы к исчезновению тупиковой ситуации
- Но, с другой стороны, «богатая» транзакция, скорее всего, выполнялась дольше других транзакций
- На ее выполнение уже затрачено большое количество системных ресурсов и, вероятно, она скоро завершится самостоятельно
- Поэтому этот выбор может оказаться в системном отношении не самым удачным

Методы сериализации транзакций (56)
Синхронизационные тупики, их распознавание и разрушение (10)
Разрушение тупиков (3)

- Можно пожертвовать самой «молодой» транзакцией, которая существует в системе в течение наименьшего времени
- Такую транзакцию менее всего жалко, поскольку она еще не успела израсходовать много системных ресурсов
- Но, с другой стороны, такая транзакция не могла и накопить много блокировок, и поэтому ее насильственное завершение вряд ли поможет устранить тупиковую ситуацию
  - Так стоит ли ею жертвовать?
- Можно выбрать транзакцию-жертву случайным образом из всех транзакций, попавших в тупик
- Возможно, что в среднем этот подход привел бы к хорошим результатам
- Но, к сожалению, в нем не учитывается возможная приоритетность транзакций
  - Было бы не слишком хорошо, например, жертвовать транзакцией, запущенной от имени руководителя организации

Методы сериализации транзакций (57)
Синхронизационные тупики, их распознавание и разрушение (11)
Разрушение тупиков (4)

- Поэтому обычно при выборе транзакции-жертвы используется многофакторная оценка ее стоимости, в которую с разными весами входят время выполнения, число накопленных блокировок, приоритет и т.д.
  - В качестве «жертвы» выбирает транзакция, для которой эта оценка выдает наиболее подходящий результат
- После выбора транзакции-жертвы выполняется откат этой транзакции, который может носить полный или частичный (до некоторой точки сохранения) характер
- При этом, естественно, освобождаются блокировки, и может быть продолжено выполнение других транзакций

Методы сериализации транзакций (58)

Синхронизационные тупики, их распознавание и разрушение (12)

Разрушение тупиков (5)

- Естественно, такое насильственное устранение тупиковых ситуаций является нарушением принципа изолированности пользователей, которого невозможно избежать
- Заметим, что в централизованных системах стоимость построения графа ожидания сравнительно невелика, но она становится слишком большой в распределенных СУБД, в которых транзакции могут выполняться в разных узлах сети
- Поэтому в таких системах обычно используются другие методы сериализации транзакций

Методы сериализации транзакций (59)
Метод временных меток (1)

- Альтернативный метод сериализации транзакций, хорошо работающий в условиях редкого возникновения конфликтов транзакций и не требующий построения графа ожидания транзакций, основан на использовании *временных меток*
- Основная идея метода временн*ых* меток (Timestamp Ordering, TO), у которого существует множество разновидностей, состоит в следующем:
  - если транзакция T1 началась раньше транзакции T2, то система обеспечивает такой сериальный план, как если бы транзакция T1 была целиком выполнена до начала T2.
- Для этого каждой транзакции T предписывается временн*ая* метка t(T), соответствующая времени начала выполнения транзакции T
- При выполнении операции над объектом о транзакция T помечает его своими идентификатором, временн*ой* меткой и типом операции (чтение или изменение)

Методы сериализации транзакций (60)
Метод временных меток (2)

- Перед выполнением операции над объектом o транзакция T2 выполняет следующие действия:
  - Проверяет, помечен ли объект o какой-либо транзакцией T1
    - Если не помечен, то помечает этот объект своей временной меткой и типом операции и выполняет операцию
      - Конец действий
  - Иначе транзакция T2 проверяет, не завершилась ли транзакция T1, пометившая этот объект
    - Если транзакция T1 закончилась, то T2 помечает объект o и выполняет свою операцию
      - Конец действий.
  - Если транзакция T1 не завершилась, то T2 проверяет конфликтность операций
    - Если операции неконфликтны, то при объекте o
      - запоминается идентификатор транзакции T2,
      - остается или проставляется временная метка с меньшим значением, и
      - транзакция T2 выполняет свою операцию

Методы сериализации транзакций (61)
Метод временных меток (3)

- Если операции транзакций T2 и T1 конфликтуют, то если t(T1) > t(T2) (т.е. транзакция T1 является более «молодой», чем T2),
  - то производится откат T1 и всех других транзакций, идентификаторы которых сохранены при объекте o, и
  - T2 выполняет свою операцию
- Если же t(T1) < t(T2) (T1 «старше» T2),
  - то производится откат T2;
  - T2 получает новую временную метку и начинается заново

Методы сериализации транзакций (62)
Метод временных меток (4)

- К недостаткам метода ТО относятся потенциально более частые откаты транзакций, чем в случае использования синхронизационных захватов
- Это связано с тем, что конфликтность транзакций определяется более грубо
- Кроме того, в распределенных системах не очень просто вырабатывать глобальные временн*ы*е метки с отношением полного порядка
  - это отдельная большая наука
- Но в распределенных системах эти недостатки окупаются тем, что не нужно распознавать тупики, а построение графа ожидания в распределенных системах стоит очень дорого

Методы сериализации транзакций (63)
Версионные методы (1)

- Основная идея версионных алгоритмов сериализации транзакций состоит в том, что в базе данных допускается существование нескольких «версий» одного и того же объекта
- Эти алгоритмы, главным образом, направлены на преодоление конфликтов транзакций категорий R/W и W/R, позволяя выполнять операции чтения над
  - некоторой предыдущей версией объекта базы данных
- В результате операции чтения выполняются без задержек и тупиков, свойственных механизмам синхронизационных блокировок, а также без некоторых откатов, возможных при применении метода временн*ы*х меток
- Алгоритмы управления транзакциями, основанные на поддержке версий, достаточно широко распространены в области SQL-ориентированных СУБД
- В частности, подобные алгоритмы используются в СУБД Oracle и PostgreSQL

Методы сериализации транзакций (64)
Версионные методы (2)
Версионный вариант алгоритма временных меток (1)

- Одним из наиболее старых и простых версионных алгоритмов является *версионный вариант алгоритма временных меток (Multiversion Timestamp Ordering, MVTO)*
- Как и в простом методе временных меток, описанном в предыдущем подразделе, в алгоритме MVTO порядок выполнения операций одновременно выполняемых транзакций задается порядком временных меток, которые получают транзакции во время старта
- Временные метки также используются для идентификации версий данных при чтении и модификации – каждая версия получает временную метку той транзакции, которая ее записала
- Алгоритм MVTO не только следит за порядком выполнения операций транзакций, но также отвечает за трансформацию операций над объектами базы данных в операции над версиями этих объектов,
  - т.е. каждая операция над объектом базы данных *o* преобразуется в соответствующую операцию над некоторой версией объекта o

Методы сериализации транзакций (65)
Версионные методы (3)
Версионный вариант алгоритма временных меток (2)

- При описании алгоритма будем использовать следующие обозначения
  - Как и раньше, временную метку, полученную транзакцией Ti в начале ее работы, будем обозначать как t(Ti)
  - Операция чтения объекта базы данных o, выполняемая в транзакции Ti, будет обозначаться как Ri(o)
  - Для обозначения того, что транзакция Ti читает версию объекта базы данных o, созданную транзакцией Tk, будем использовать запись Ri(ok)
  - Для обозначения того, что транзакция Ti записывает версию элемента данных o, будем использовать запись Wi(oi)
- Алгоритм MVTO работает следующим образом

Методы сериализации транзакций (66)
Версионные методы (4)
Версионный вариант алгоритма временных меток (3)

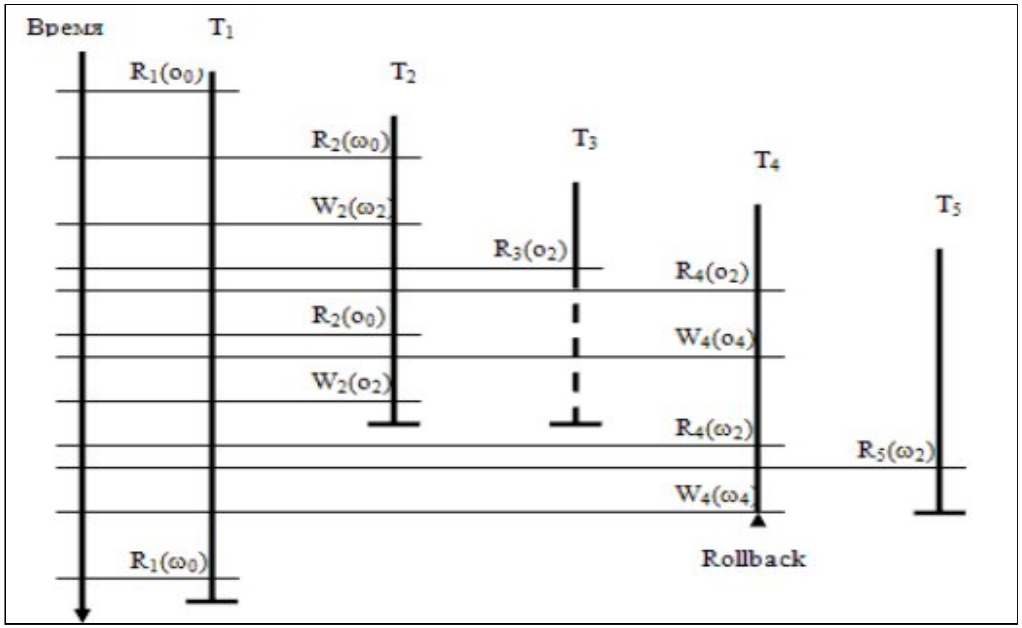
- Любая операция Ri(*o*) преобразуется в операцию Ri(*o*k), где *o*k – это версия объекта *o*, помеченная наибольшей временной меткой t(Tk), такой что t(Tk)≤ t(Ti)
  - Другими словами, транзакции Ti для чтения дается версия объекта *o*, созданная транзакцией Tk, которая не моложе Ti, но старше любой другой транзакции Tn, создававшей свою версию объекта *o*
- При обработке операции Wi(*o*) выполняются следующие действия:
  - если к этому времени некоторой незафиксированной транзакцией Tn уже выполнена некоторая операция Rn(*o*k), такая что t(Tk) ≤ t(Ti) < t(Tn), то
    - операция Wi(*o*) не выполняется, а
    - транзакция Ti откатывается;
  - в противном случае Wi(*o*) преобразуется в Wi(*o*i),
    - т.е. образуется еще одна версия объекта *o*

Методы сериализации транзакций (67)
Версионные методы (5)
Версионный вариант алгоритма временных меток (4)

- При откате любой транзакции уничтожаются все созданные ею версии объектов базы данных и откатываются все транзакции, прочитавшие хотя бы одну из этих версий
  - Тем самым, откаты транзакций могут быть «каскадными».
- Выполнение операции фиксации транзакции Ti (COMMIT) откладывается до того момента, когда завершатся все транзакции, записавшие версии данных, прочитанные Ti
- Без соблюдения этого требования не соблюдалось бы свойство долговечности (durability) транзакций, поскольку при откате некоторых транзакций потребовалось бы откатывать и ранее зафиксированные транзакции

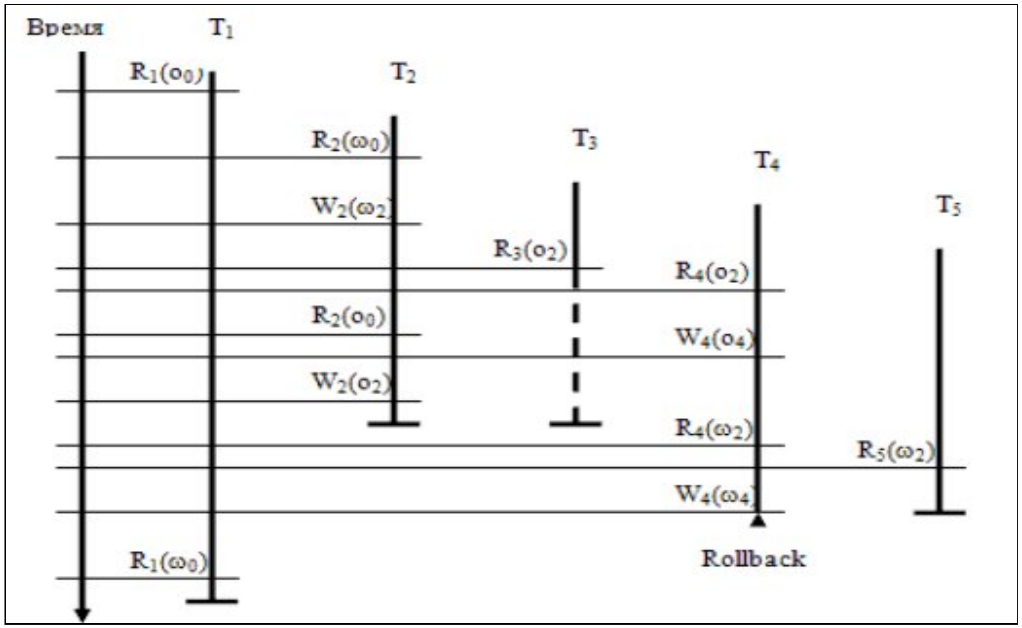
Методы сериализации транзакций (68)
Версионные методы (6)
Версионный вариант алгоритма временных меток (5)

- Преимущества алгоритма MVTO лучше всего иллюстрируются поведением транзакций T1 и T2
- При использовании блокировок между ними возник бы синхронизационный тупик, а при использовании обычного метода временных меток одна из транзакций подверглась бы откату
- Однако при применении версий такие неприятности не возникают из-за того, что первая транзакция читает «старые» версии объектов *o* и *ω*



Методы сериализации транзакций (70)
Версионные методы (7)
Версионный вариант алгоритма временных меток (6)

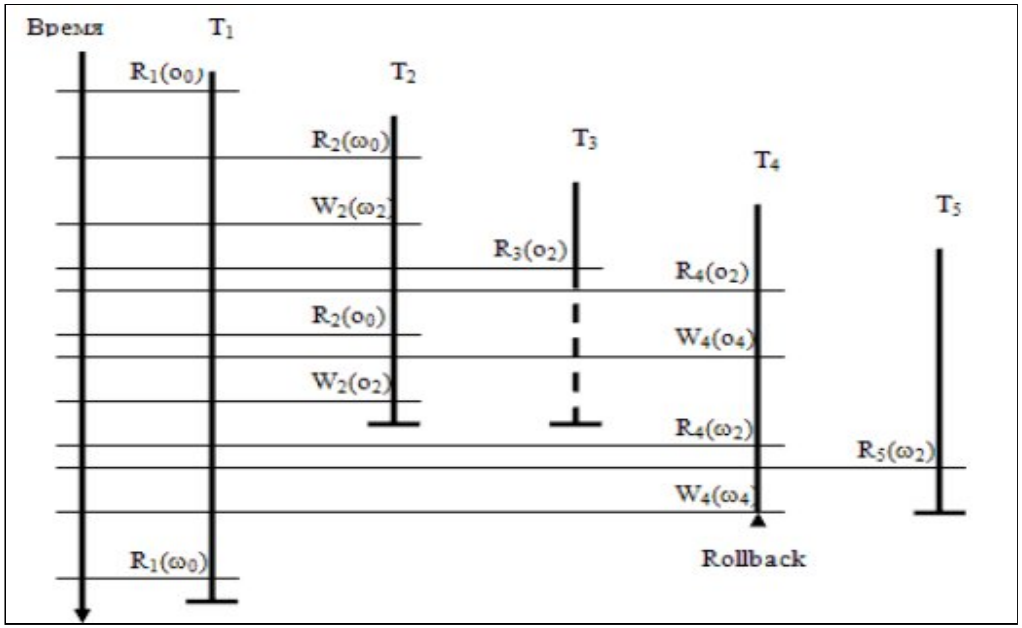
- Транзакция Т3 ожидает фиксации транзакции Т2 перед своим собственным завершением
  - это показано пунктирной линией



- Это происходит потому, что транзакция Т3 прочитала версию *o2* объекта *o*, образованную еще не зафиксированной транзакцией

Методы сериализации транзакций (71)
Версионные методы (8)
Версионный вариант алгоритма временных меток (7)

- Транзакция T4 пытается создать версию $\omega 4$ объекта ω после того, как еще не зафиксированная транзакция T5
  - начавшаяся позже
- уже прочитала более раннюю версию $\omega 4$
- Поэтому транзакция T5 не сможет «увидеть» изменения объекта ω, произведенные транзакцией T4. Следовательно, сериализация транзакций в порядке получения ими временных меток становится невозможной, и приходится произвести откат транзакции T4



Методы сериализации транзакций (72)
Версионные методы (9)
Версионный вариант алгоритма временных меток (8)

- Итак, основными преимуществами алгоритма MVTO является
  - отсутствие задержек и откатов при выполнении операций чтения,
- а основным недостатком –
  - возможность возникновение каскадных откатов транзакций при выполнении операций записи
- Кроме того, в базе данных может накапливаться произвольное число версий одного и того же объекта, и
  - определение того, какие версии больше не требуются, является серьезной технической проблемой

Методы сериализации транзакций (73)

Версионные методы (10)

Версионный вариант двухфазного протокола синхронизационных блокировок (1)

- При описании *двухверсионного варианта протокола 2PL*
  - *Two-Version Two-Phase Locking Protocol, 2V2PL*
- будем называть
  - *текущими* версиями объектов базы данных версии, созданные зафиксированными транзакциями с наиболее поздним временем фиксации
  - *незафиксированными* версиями – версии, созданные еще незавершившимися транзакциями
- При следовании протоколу 2V2PL в каждый момент времени существует не более одной незафиксированной версии каждого объекта базы данных

Методы сериализации транзакций (74)
Версионные методы (11)
Версионный вариант двухфазного протокола синхронизационных блокировок (2)

- Операции любой транзакции Ti над объектом базы данных o обрабатываются следующим образом:
  - операция Ri(o) немедленно выполняется над текущей версией объекта o;
  - операция Wi(o), приводящая к созданию новой версии объекта o, выполняется только после завершения
    - фиксации или отката
  - транзакции, создавшей незафиксированную версию объекта o;
  - выполнение операции COMMIT откладывается до тех пор, пока не завершатся все транзакции Tk, прочитавшие текущие версии объектов базы данных, которые должны замениться незафиксированными версиями этих объектов, созданными транзакцией Ti

Методы сериализации транзакций (75)
Версионные методы (12)
Версионный вариант двухфазного протокола синхронизационных блокировок (3)

- Для реализации такого поведения используются три типа блокировок:
  - RL (Read Lock) – в этом режиме блокируется любой объект базы данных *o* перед выполнением операции чтения его текущей версии
  - удержание этой блокировки до конца транзакции гарантирует, что при повторном чтении объекта *o* будет прочитана та же версия этого объекта

	RL(o)	WL(o)	CL(o)
RL(o)	да	да	нет
WL(o)	да	нет	нет
CL(o)	нет	нет	нет

Методы сериализации транзакций (76)
Версионные методы (13)
Версионный вариант двухфазного протокола синхронизационных блокировок (4)

- WL (Write Lock) – в этом режиме блокируется любой объект базы данных *o* перед выполнением операции, приводящей к созданию новой (незафиксированной) версии этого объекта

	RL(o)	WL(o)	CL(o)
RL(o)	да	да	нет
WL(o)	да	нет	нет
CL(o)	нет	нет	нет

- удержание этой блокировки до конца транзакции гарантирует, что в любой момент времени будет существовать не более одной незафиксированной версии любого объекта базы данных

Методы сериализации транзакций (76)
Версионные методы (13)
Версионный вариант двухфазного протокола синхронизационных блокировок (4)

- CL (Commit Lock) – блокировка устанавливается во время выполнения операции COMMIT транзакции и затрагивает любой объект базы данных, новую версию которого создала данная транзакция

	RL(o)	WL(o)	CL(o)
RL(o)	да	да	нет
WL(o)	да	нет	нет
CL(o)	нет	нет	нет

- удовлетворение этой блокировки для данной транзакции гарантирует, что завершились все транзакции, читавшие текущие версии объектов, новые версии которых были созданы при выполнении данной транзакции, и, следовательно, их можно заменить

Методы сериализации транзакций (77)
Версионные методы (14)
Версионный вариант двухфазного протокола синхронизационных блокировок (5)

- Как видно, операция чтения может блокироваться только на время фиксации транзакции, заменяющей текущую версию требуемого объекта базы данных
- Для выполнения операции записи требуется долговременная монопольная блокировка соответствующего объекта базы данных, которая, однако, в этом случае совместима с блокировкой этого же объекта по чтению
  - поскольку в действительности блокируются разные версии этого объекта
- И, конечно, как и во всех схемах сериализации транзакций на основе блокировок, здесь возможны синхронизационные тупики

Методы сериализации транзакций (78)

Версионные методы (15) Версионно-блокировочный протокол сериализации транзакций для поддержки только читающих транзакций (1)

- Гибридный протокол, поддерживающий эффективное выполнение транзакций, не изменяющих состояние базы данных
  - *Multiversion Protocol for Read-Only Transactions, ROMV*
- При применении этого протокола при образовании каждой транзакции явно указывается ее тип –
  - только читающая (read-only) или
  - изменяющая (update) транзакция
- В только читающих транзакциях допускается использование только операций чтения объектов базы данных, а в изменяющих транзакциях – операций и чтения, и записи.

Методы сериализации транзакций (70)

Версионные методы (16) Версионно-блокировочный протокол сериализации транзакций для поддержки только читающих транзакций (2)

- Изменяющие транзакции выполняются в соответствии с обычным протоколом 2PL,
  - т.е. перед выполнением операции чтения или записи объекта базы данных о этот объект должен быть заблокирован в режиме S или X соответственно, и
  - блокировки объектов удерживаются до конца изменяющей транзакции
- Каждая операции записи объекта о создает его новую версию, которая при завершении транзакции помечается временной меткой, соответствующей моменту фиксации этой транзакции

Методы сериализации транзакций (71)

Версионные методы (17) Версионно-блокировочный протокол сериализации транзакций для поддержки только читающих транзакций (3)

- Каждая только читающая транзакция при своем образовании получает соответствующую временную метку
- При выполнении операции чтения объекта базы данных о транзакция получает доступ к версии объекта *o*, образованной изменяющей транзакцией, которая
  - хронологически последней зафиксировалась к моменту образования данной читающей транзакции

Методы сериализации транзакций (72)

Версионные методы (18) Версионно-блокировочный протокол сериализации транзакций для поддержки только читающих транзакций (4)

- Основным плюсом протокола ROMV по сравнению с ранее описанным протоколом 2V2PL является принципиальное отсутствие синхронизационных задержек при выполнении операций чтения только читающих транзакций
- Если сравнивать ROMV с MVTO, то он выигрывает в принципиальном отсутствии откатов только читающих транзакций
- Конечно, при работе изменяющих транзакций возможно возникновение синхронизационных тупиков и откатов, и здесь требуется использовать обычные методы распознавания и разрушения тупиков

Методы сериализации транзакций (73)

Версионные методы (19) Версионно-блокировочный протокол сериализации транзакций для поддержки только читающих транзакций (5)

- При использовании протокола ROMV в базе данных может возникать произвольное число версий объектов
- Требуется создание специального сборщика мусора, который должен удалять ненужные версии данных
- Простейший сборщик мусора удаляет все неиспользуемые версии, значения временных меток которых меньше значения временной метки старейшей активной только читающей транзакции

Заключение (1)

- В этой лекции описаны основные принципы управления транзакциями в системах управления базами данных, различные методы, алгоритмы и протоколы, способствующие достижению целей управления транзакциями
- Следует заметить, что существует достаточно развитая теория управления транзакциями с собственными средствами формализации постановки задач и доказательства корректности алгоритмов
- Для обеспечения более простого понимания сути материала в него не включены все эти формализмы

Заключение (2)

- В лекции описаны два основных подхода к сериализации транзакций – на основе синхронизационных блокировок и временн*ы*х меток
- У каждого из этих подходов имеются свои достоинства и недостатки, но на практике существенно больше распространен метод синхронизационных блокировок
- В заключение лекции были рассмотрены расширения этих подходов с применением версий объектов базы данных
- Соответствующие алгоритмы и протоколы позволяют уменьшить число потенциальных конфликтов транзакций, но для их поддержки требуются дополнительные расходы внешней памяти и усложнение общей архитектуры СУБД