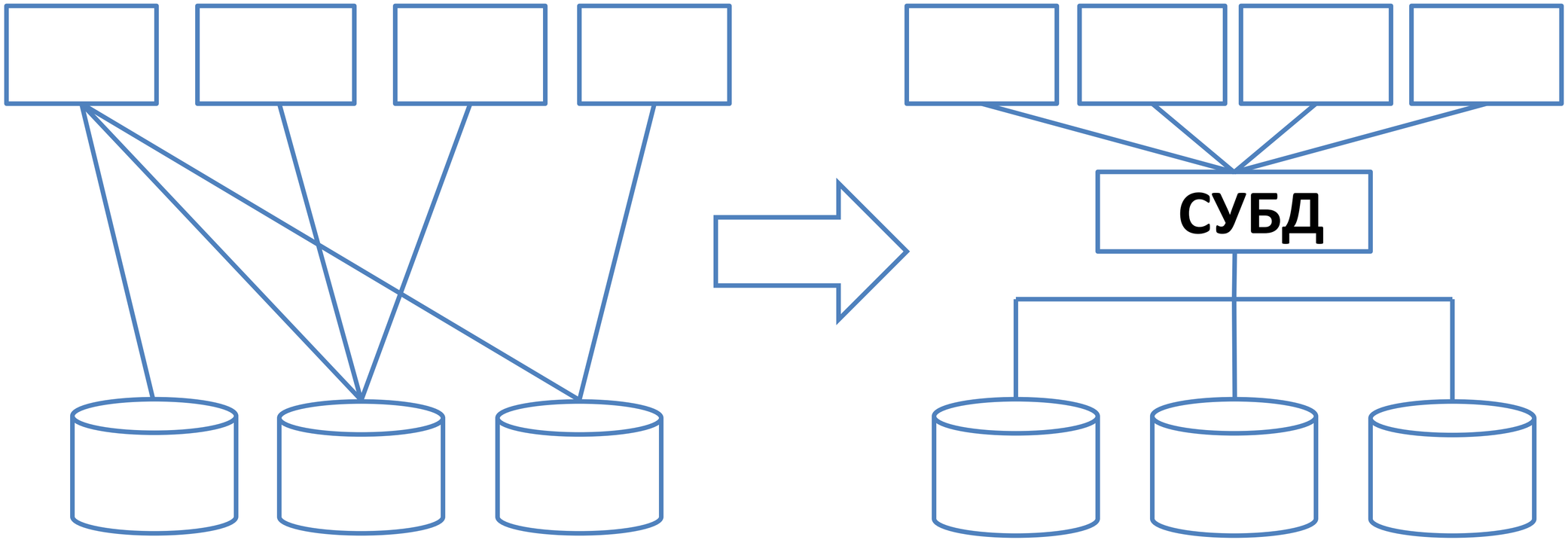
Основы технологий баз данных

Б. А. Новиков
Санкт-Петербургский университет, JetBrains Research

Предыдущие серии

краткое содержание

Централизация управления общими данными нескольких приложений



Демонстрационная база данных PostgreSQL

https://postgrespro.ru/education/demodb

Ключевые свойства моделей данных

- Методы идентификации объектов внутри и вне модели
- Поиск и связывание объектов
- Массовая или штучная обработка объектов данных

Теоретическая реляционная модель данных

- Домены, отношения, операции
- Функциональные зависимости и ключи
- Естественная идентификация по значениям атрибутов
- Идентификация определяется функциональными зависимостями
- Связи вычисляются динамически при выполнении операций соединения
- Ассоциативный доступ к данным по значениям (операция селекции)

Другие модели данных

- Сущность-связь
- Объектно-ориентированные БД
- Объектно-реляционные
- Сетевая и иерархическая
- Слабоструктурированные
- Тернарная
- Бинарная

Язык запросов SQL

- Декларативный язык
- Структуры данных и операторы описания данных: CREATE, DROP, ALTER
- Ограничения целостности: NOT NULL, CHECK, UNIQUE, PRIMARY KEY, FOREIGN KEY
- Операторы: SELECT, INSERT, UPDATE, DELETE
- Реализация операций реляционной алгебры в SQL и другие возможности SQL

Postgres PROFESSIONAL

Внешние ключи

- FOREIGN KEY — ограничение целостности
- Никак не влияет на семантику запросов на чтение данных
- Соединение (JOIN) никак не связано с понятием внешнего ключа, хотя часто ключом соединения является внешний ключ
- Ограничения целостности никак не влияет на эффективность операции соединения
- Миф о влиянии возник потому, что для внешних ключей часто создаются индексы

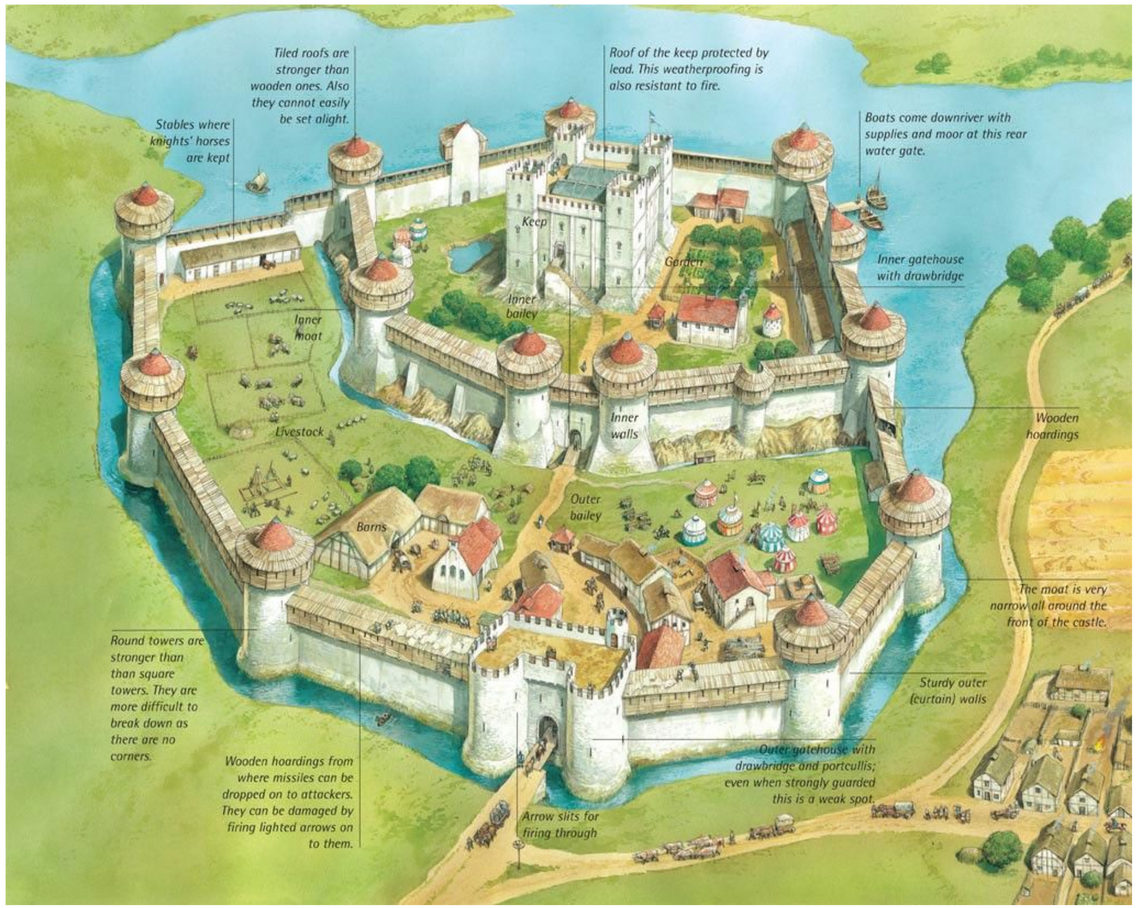
Дискуссия о дубликатах в CACM

SELECT ALL или SELECT DISTINCT?

+	−
Дубликаты могут быть необходимы для правильного моделирования реальности	Реляционная модель работает с множествами и не допускает дубликатов
Устранение дубликатов может приводить к снижению эффективности выполнения запросов	Введение дубликатов не дает новой информации
Отношения с дубликатами можно описать в теории мультимножеств	Введение дубликатов делает некорректными соотношения между операциями реляционной алгебры

Защита должна быть похожа на крепость

- Физическая защита оборудования
- Защита в вычислительной сети
- Операционная система
- Файловая система
- Сервер приложений и приложения
- База данных



Защита базы данных: итоги

- Средства безопасности должны быть многоуровневыми
- СУБД обеспечивают полноценную реализацию модели RBAC для объектов базы данных
- Применимость средств защиты на уровне базы данных ограничивается количеством пользователей, зарегистрированных на сервере базы данных
- Возможности доступа неограниченного количества пользователей реализуются на уровне приложений

Postgres PROFESSIONAL

Согласованность

Согласованность в базах данных

- Целостность, согласованность и безопасность (integrity, consistency, security)
- Конкурентное использование ресурсов корректными программами может привести к некорректным результатам

Согласованность

- Отказы при выполнении программ-клиентов, СУБД, операционной системы и оборудования могут привести к нарушениям согласованности

- Функции СУБД

1. гарантировать согласованность при конкурентном выполнении

2. восстановление согласованности после всех видов отказов

Транзакции

- Транзакция — это совокупность операций, выполняемых прикладной программой, которые переводят согласованное состояние базы данных в согласованное, если:

  1. отсутствуют помехи со стороны других приложений

  2. транзакция выполнена полностью

- Последовательное выполнение транзакций гарантирует сохранение согласованности, но несовместимо с высокой производительностью системы

- Система должна обеспечить сохранение согласованности при конкурентном выполнении транзакций, когда операции разных транзакций могут перемежаться

Пример: конкурентное обновление

Read balance	950
-100	850
Write balance	850

Read balance	950
-200	750
Write balance	**750**

Приложения, в которых важны транзакционные свойства

- Короткие транзакции в банковских системах, системах резервирования и т. п. (On-line transaction processing — OLTP)
- Распределенные транзакции в системах электронной коммерции
- Координация повторяющихся деловых процедур (потоки работ)

Свойства транзакций: ACID

ACID = Atomicity, Consistency, Isolation, Durability

- **Атомарность**: транзакция либо выполняется полностью, либо не оставляет никаких изменений (фиксация или обрыв)
- **Согласованность**: сохранение согласованного состояния данных
- **Изолированность**: результаты незавершенной транзакции недоступны для других транзакций
- **Долговечность**: результаты успешно завершенной (зафиксированной) транзакции не могут быть потеряны

Транзакции, истории и расписания

- База данных: $x, y, z, \ldots$
- Операции: $r(x), w(x)$
- Транзакции: конечное частично упорядоченное множество операций $r_1(x)\ r_1(y)\ w_1(z)$
- История: (частично) упорядоченное множество операций нескольких транзакций $r_1(x)\ r_2(x)\ w_1(x)\ w_2(x)\ c_1\ c_2$
  - операции над одним элементом данных должны быть упорядочены
- Расписание: префикс истории

Фиксация и обрыв

- После успешного выполнения всех операций транзакции приложение должно выполнить фиксацию (commit)
- Транзакция может быть оборвана явной операцией приложения (abort) или по инициативе СУБД
- Фиксации и обрывы реализуют принцип атомарности
- Каждая транзакция заканчивается ровно одной из операций commit, abort

Проблемы, вызванные конкурентностью выполнения (аномалии)

- Потеря обновлений

$$r_1(x)\ r_2(x)\ w_1(x)\ w_2(x)$$

- Несогласованное чтение

$$r_1(x)\ r_2(x)\ r_2(y)\ w_2(x)\ w_2(y)\ r_1(y)$$

- Грязное чтение

$$r_1(x)\ w_1(x)\ r_2(x)\ w_2(x)\ a_1$$

Серийное расписание

- Расписание, в котором все операции любой транзакции либо предшествуют, либо следуют за операциями любой другой транзакции

- Серийное расписание всегда корректно

Эквивалентность и сериализуемость по видимому состоянию — VSR

- Эквивалентность ⇔ совпадение видимых и конечных состояний всех элементов данных
- Сериализуемость ⇔ эквивалентность серийному расписанию
- Предотвращает все аномалии
- Высокая вычислительная сложность проверки сериализуемости

Конфликты

- Пара операций из расписания, такая, что
  1. операции принадлежат разным транзакциям
  2. работают с одним элементом данных
  3. по крайней мере одна из двух — операция записи
- Конфликты присутствуют в любом нетривиальном расписании

Эквивалентность и сериализуемость по конфликтам — CSR

- Множество конфликтов расписания содержит пары, в которых первая операция предшествует второй (и пары находятся в конфликте)
- Расписания эквивалентны, если их множества конфликтов совпадают
- Расписание называется сериализуемым, если оно эквивалентно по конфликтам серийному
- Сериализуемые расписания корректны

Критерий сериализуемости

- Граф конфликтов (граф сериализуемости)
  - вершины соответствуют транзакциям
  - дуги проводятся для каждого конфликта в направлении конфликта
- Расписание сериализуемо по конфликтам тогда и только тогда, когда граф сериализуемости не содержит контуров

План доказательства

- Граф конфликтов серийного расписания не может иметь контуров, потому что транзакции упорядочены и все дуги направлены от начала к концу расписания

- Если граф не имеет контуров, эквивалентное серийное расписание можно построить с помощью топологической сортировки графа

Сериализуемость по коммутативности

- Любые операции чтения коммутируют
- Любые операции над разными элементами коммутируют
- Расписание сериализуемо по коммутативности, если его можно преобразовать в серийное перестановками соседних операций
- Сериализуемость по коммутативности эквивалентна сериализуемости по конфликтам

Диспетчер транзакций

- Модель СУБД: диспетчер (транзакций) и исполнитель запросов
- Требования к диспетчеру транзакций:
  - корректность
  - производительность
    - фактический уровень (псевдо)параллелизма
    - процент оборванных транзакций
- Пессимистические и оптимистические протоколы

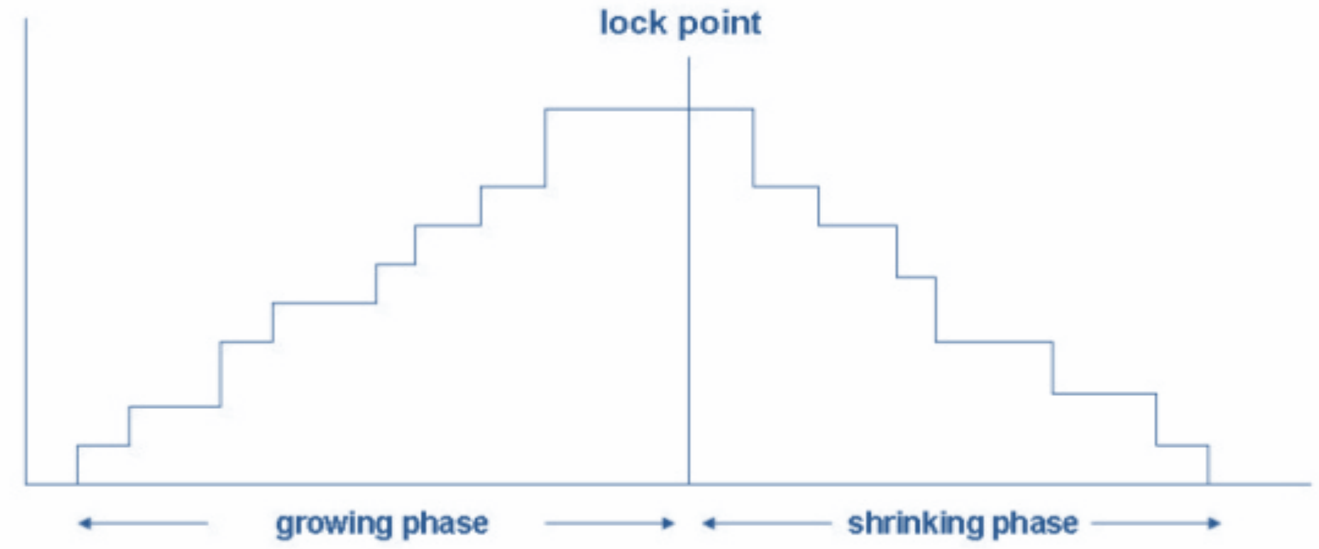
Использование блокировок

- Операции установки $lr(x)$, $lw(x)$ и снятия блокировки $ur(x)$, $uw(x)$
- Совместимость блокировок: блокировки одного элемента данных несовместимы, если они
  - устанавливаются разными транзакциями
  - по крайней мере одна из них — на запись
- Попытка установки несовместимой блокировки переводит транзакцию в состояние ожидания
- Проблемы, связанные с использованием блокировок:
  - корректность
  - тупики
  - производительность

Протокол блокирования 2PL

- Для каждой операции необходимо предварительно установить блокировку, все блокировки должны быть сняты до завершения транзакции

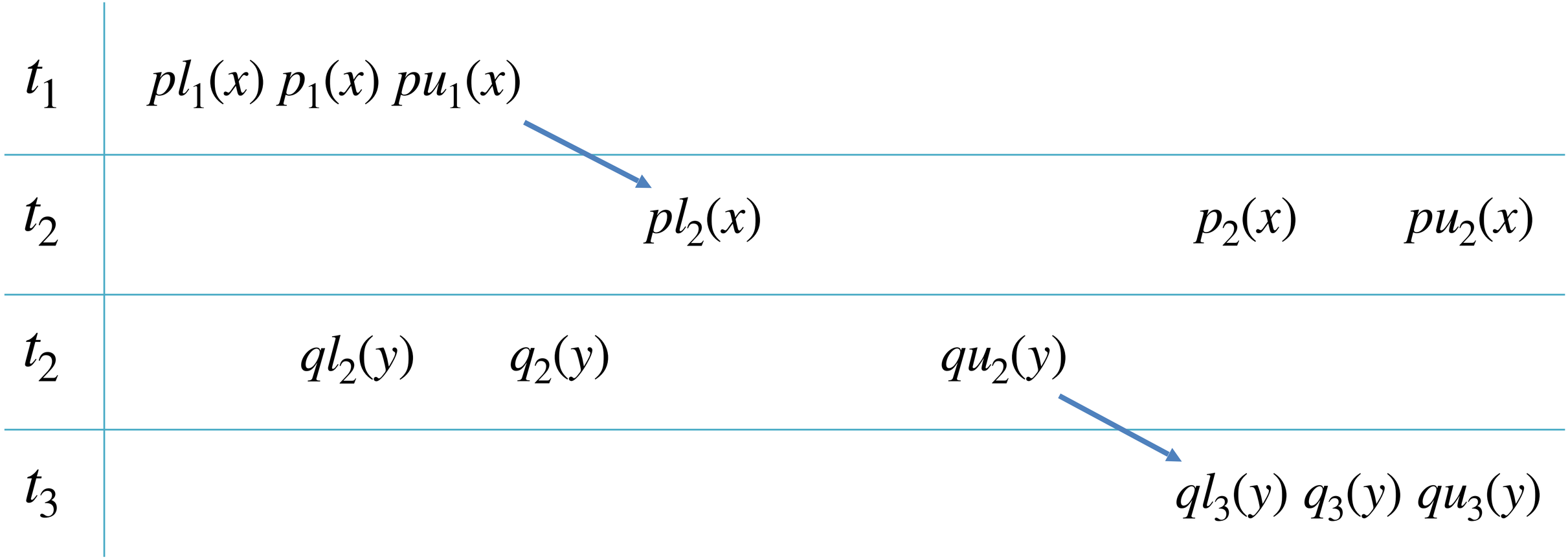
- Транзакция не может устанавливать новые блокировки после того, как она сняла какую-либо из блокировок



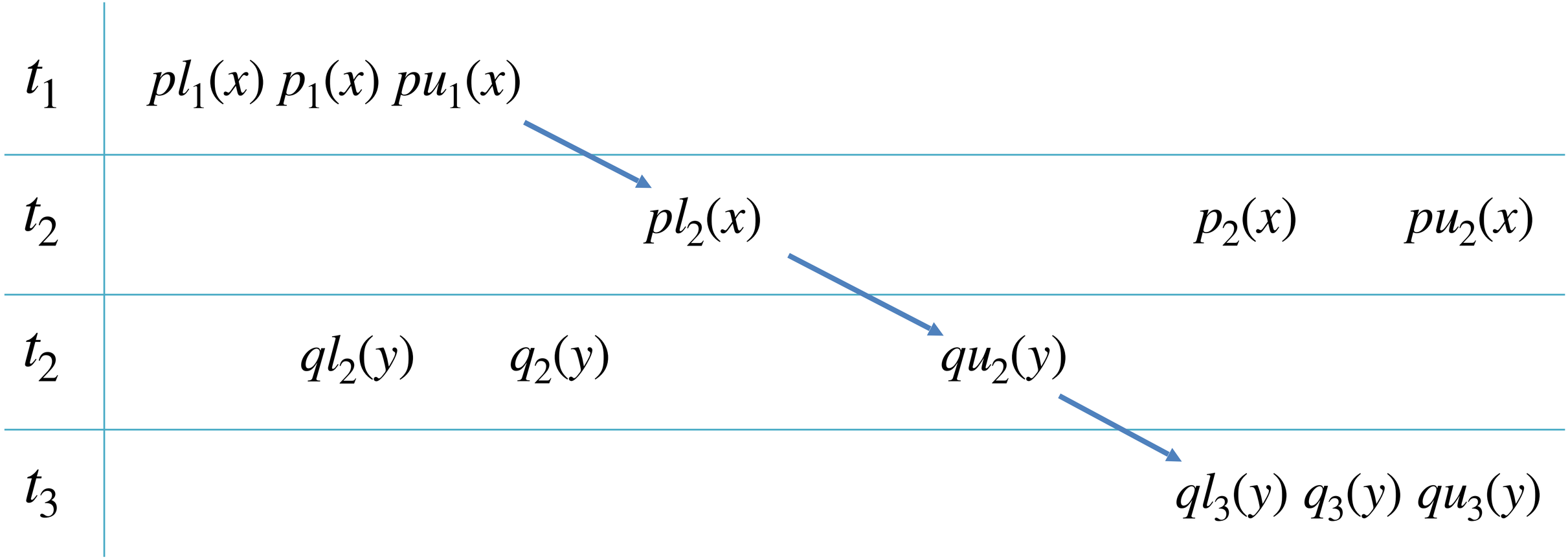
Gen(2PL) ⊆ CSR

t_1	$pl_1(x)$	$p_1(x)$	$pu_1(x)$					
t_2				$pl_2(x)$			$p_2(x)$	$pu_2(x)$
t_2		$ql_2(y)$	$q_2(y)$			$qu_2(y)$		
t_3							$ql_3(y)$ $q_3(y)$	$qu_3(y)$

Gen(2PL) ⊆ CSR


t_1
$pl_1(x)$ $p_1(x)$ $pu_1(x)$
t_2
$pl_2(x)$
$p_2(x)$
$pu_2(x)$
t_2
$ql_2(y)$
$q_2(y)$
$qu_2(y)$
t_3
$ql_3(y)$ $q_3(y)$ $qu_3(y)$

Gen(2PL) ⊆ CSR


t_1
$pl_1(x)$ $p_1(x)$ $pu_1(x)$
t_2
$pl_2(x)$
$p_2(x)$
$pu_2(x)$
t_2
$ql_2(y)$
$q_2(y)$
$qu_2(y)$
t_3
$ql_3(y)$ $q_3(y)$ $qu_3(y)$

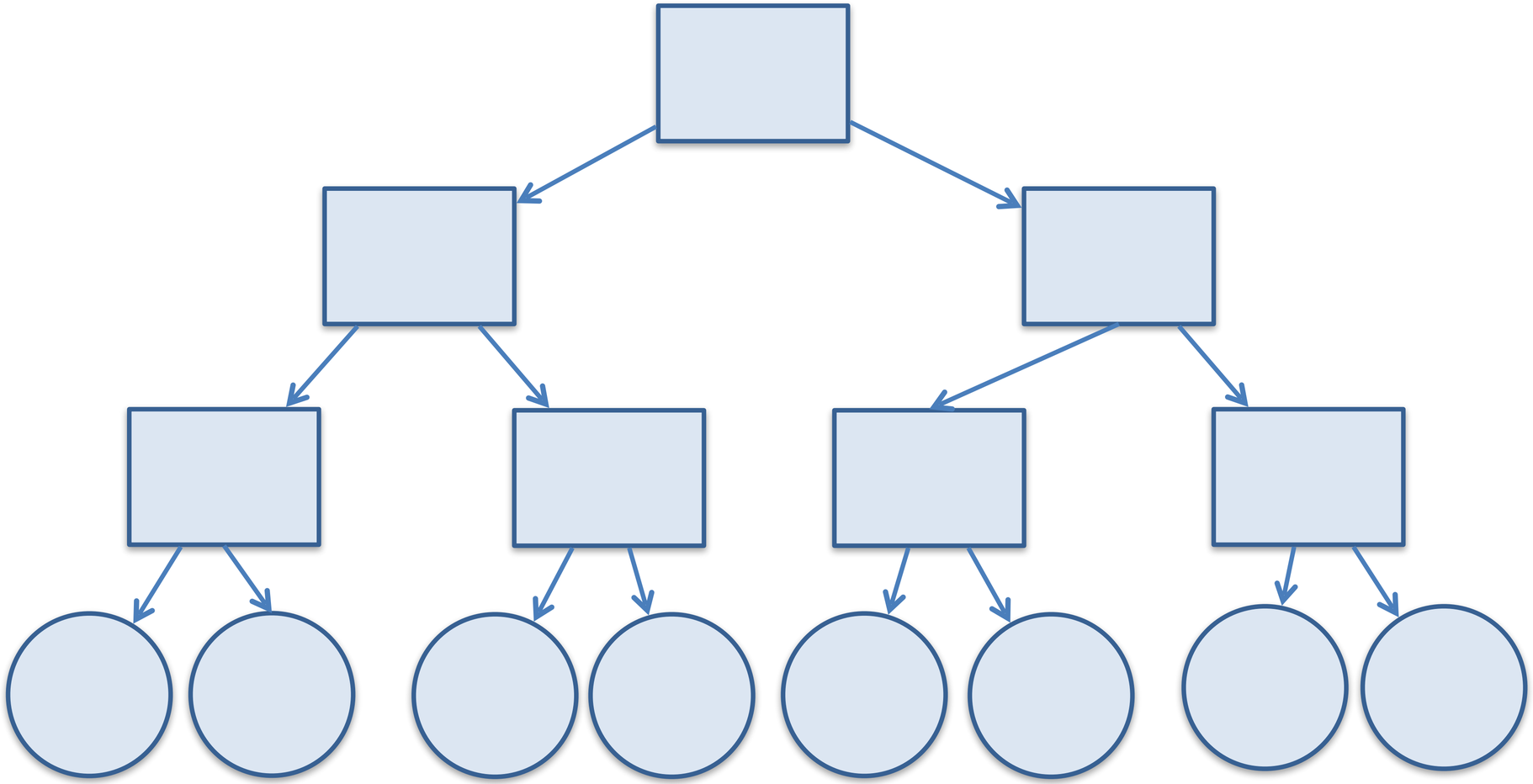
Тупики

- $w_1(x)\ w_2(y)\ r_1(y)\ r_2(x)$
- Транзакции, попавшие в тупик, должны быть оборваны
- Граф ожиданий
  - вершины — активные транзакции
  - дуги проводится из ожидающей транзакции в транзакцию, установившую несовместимые замки
- Тупик имеет место тогда и только тогда, когда в графе ожиданий имеется контур

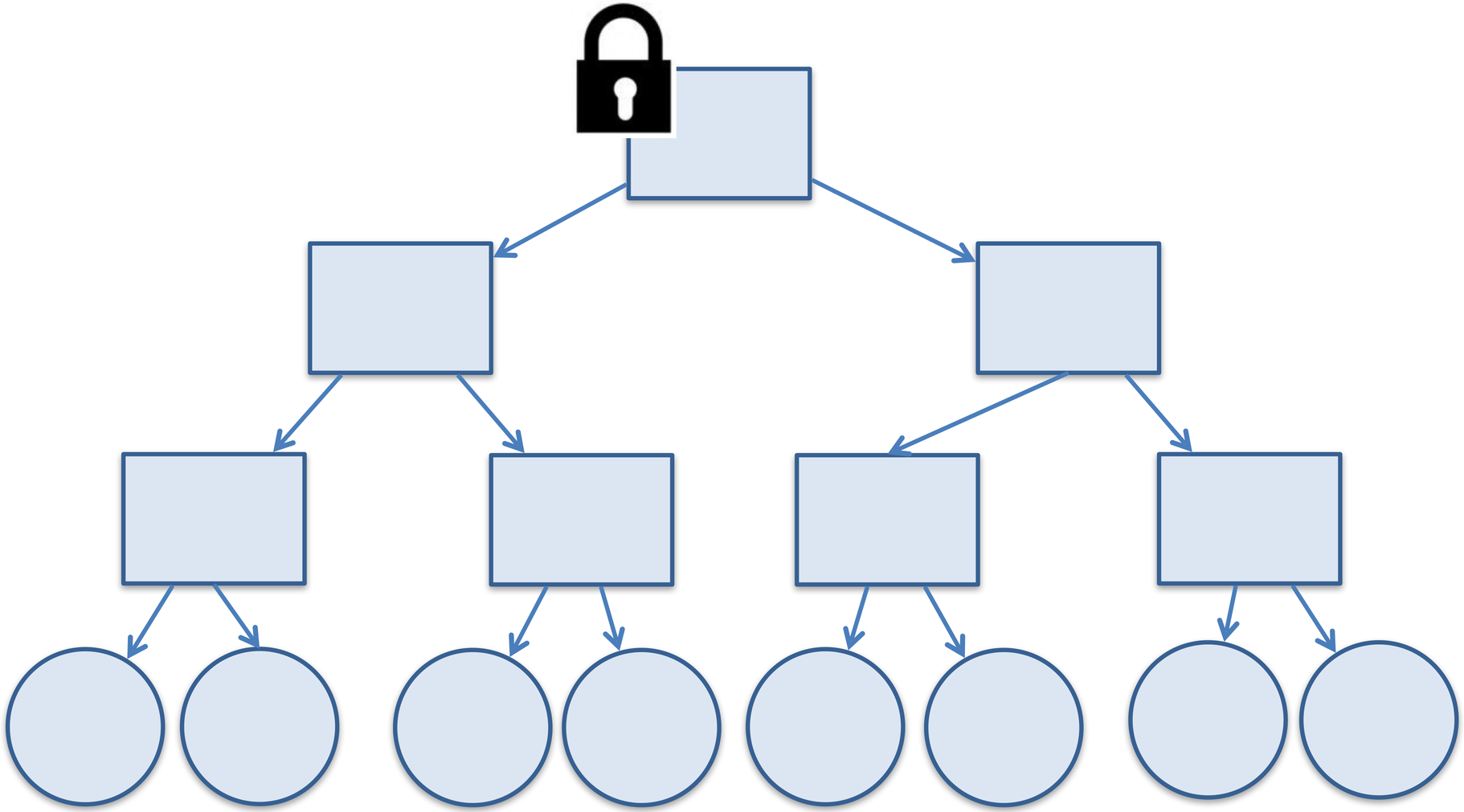
Протокол для деревьев WTL

- База данных структурирована как дерево
- Протокол:
  - Все блокировки — на запись
  - Для установки блокировки необходимо иметь установленную блокировку на родительскую вершину (кроме корня дерева)
  - Снятие блокировок возможно в любое время и не препятствует установке новых блокировок

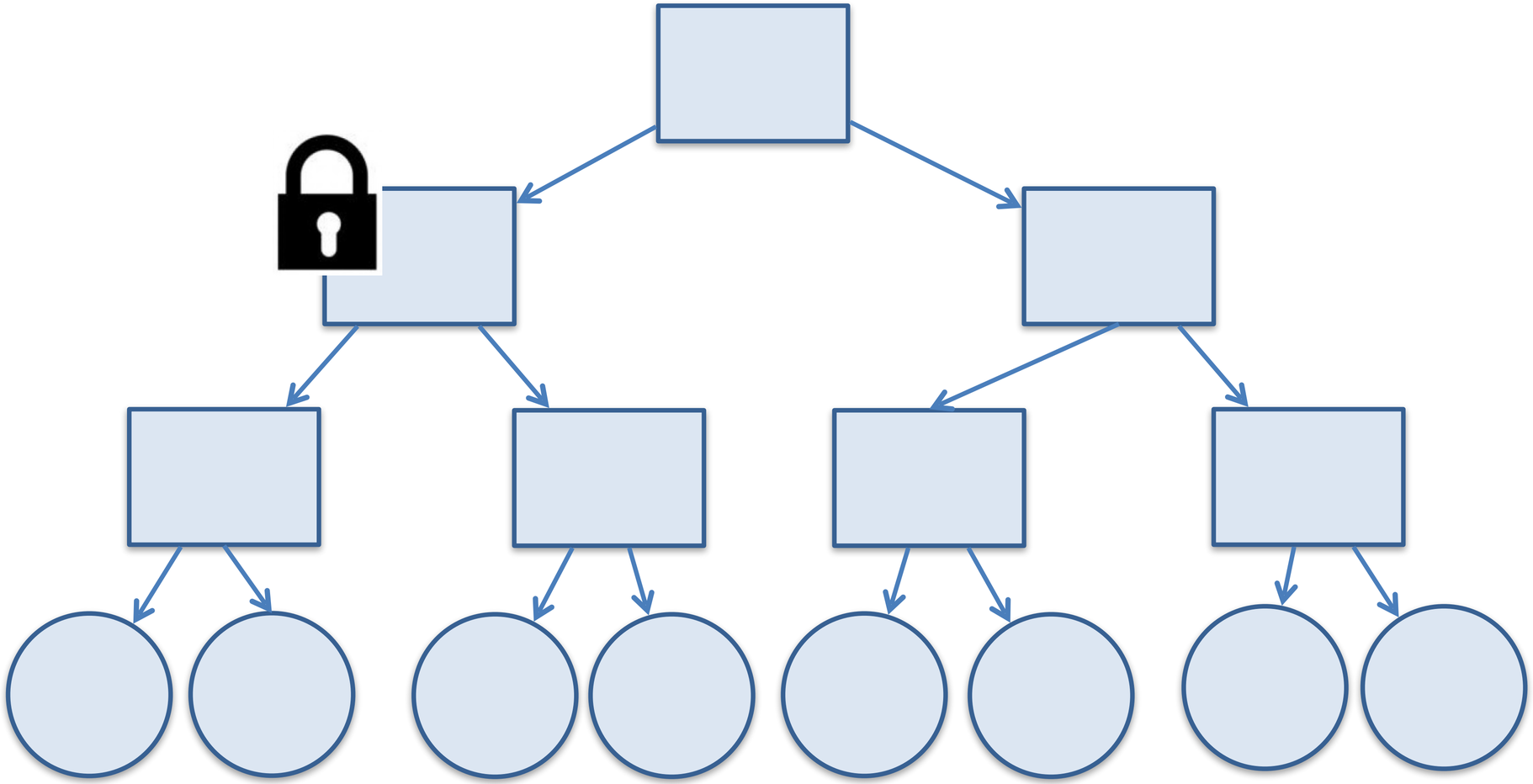
Протокол для деревьев WTL



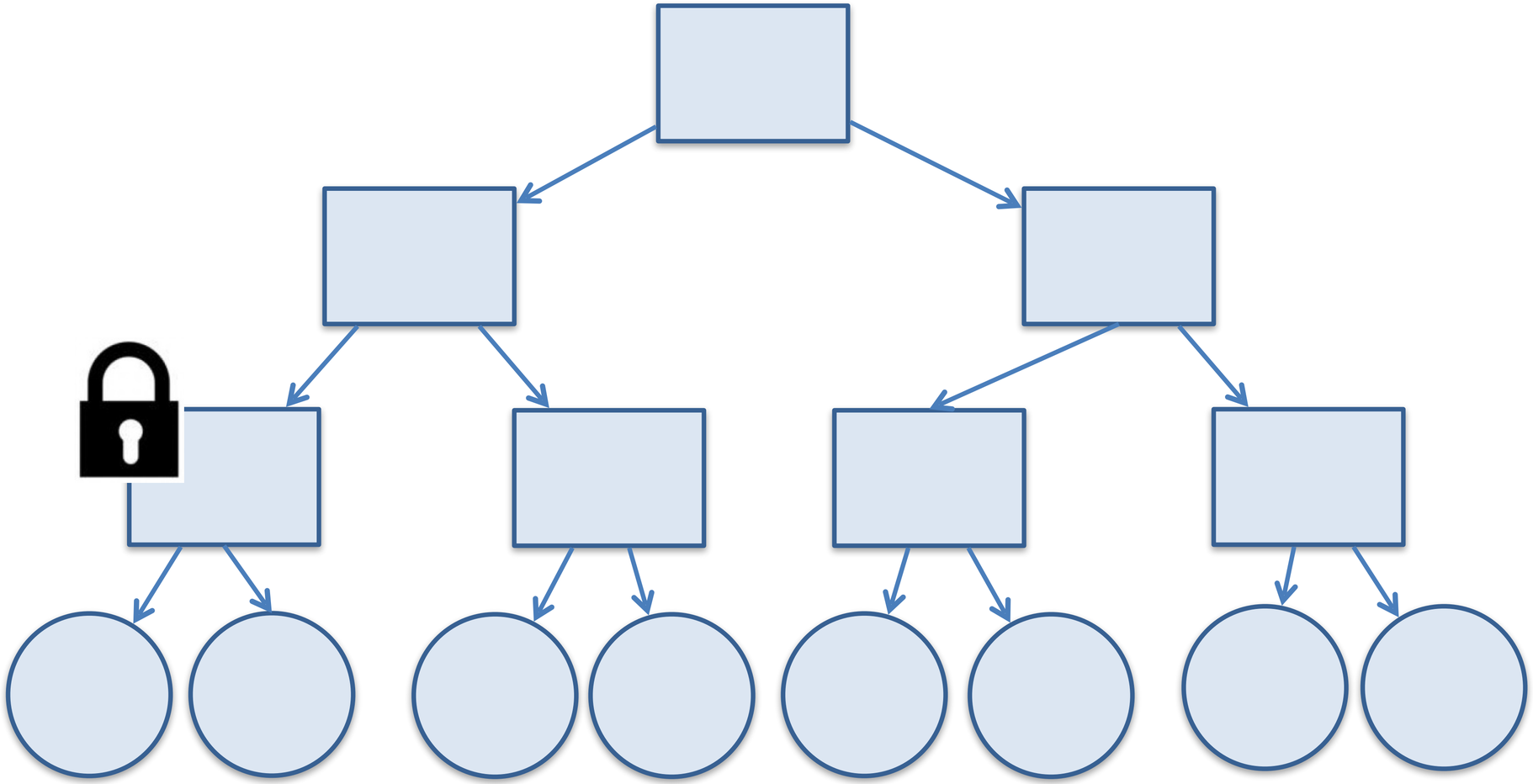
Протокол для деревьев WTL



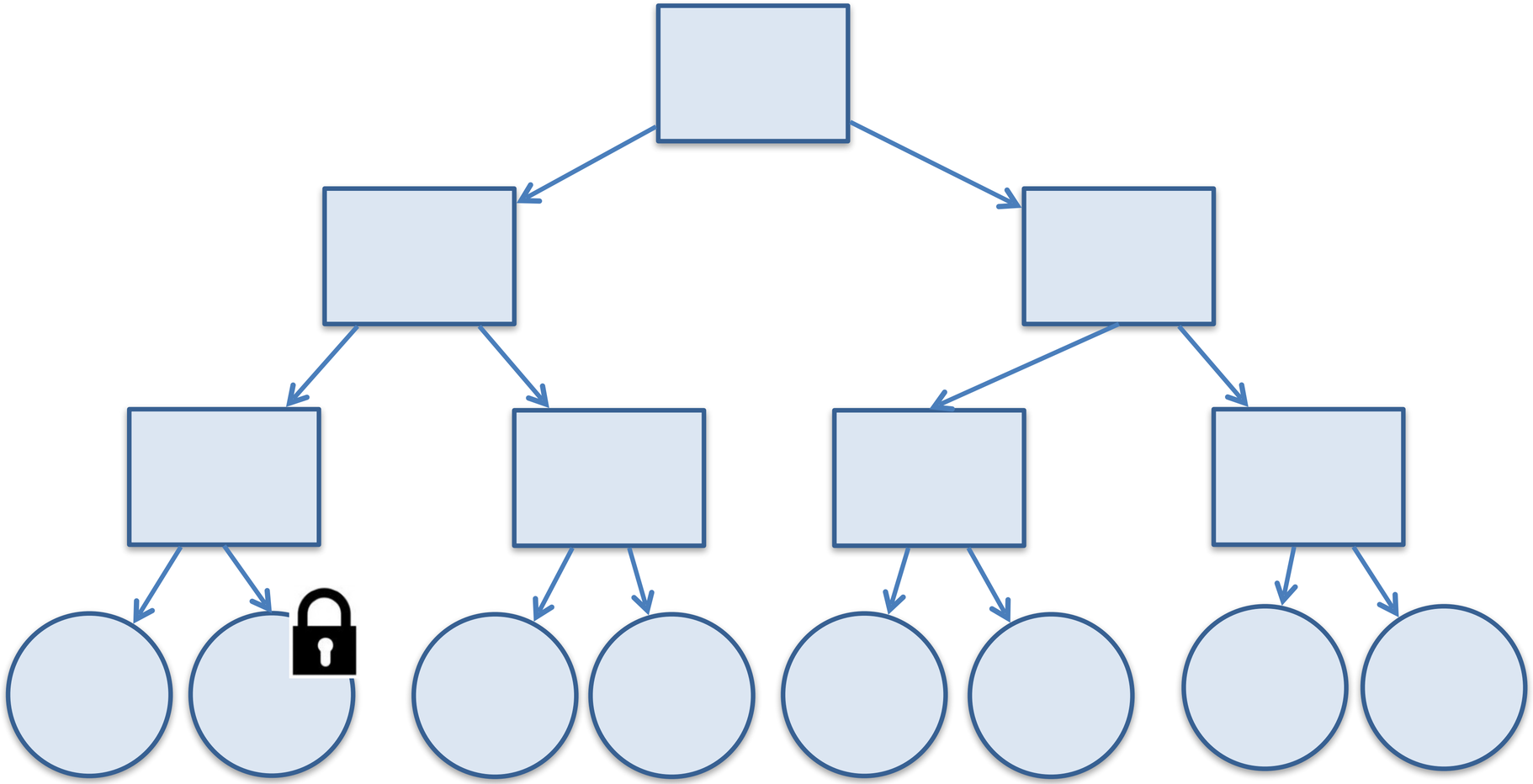
Протокол для деревьев WTL



Протокол для деревьев WTL



Протокол для деревьев WTL



Корректность WTL

- Граф сериализуемости ацикличен
- Пусть y — родитель x, и $w_1(x) < w_2(x)$, тогда $w_1(y) < w_2(y)$
- Транзакции сериализуются в том порядке, в котором они устанавливают блокировки на корень
- Протокол WTL свободен от тупиков
  - Последовательность установки блокировок определяется последовательностью их установки на корень

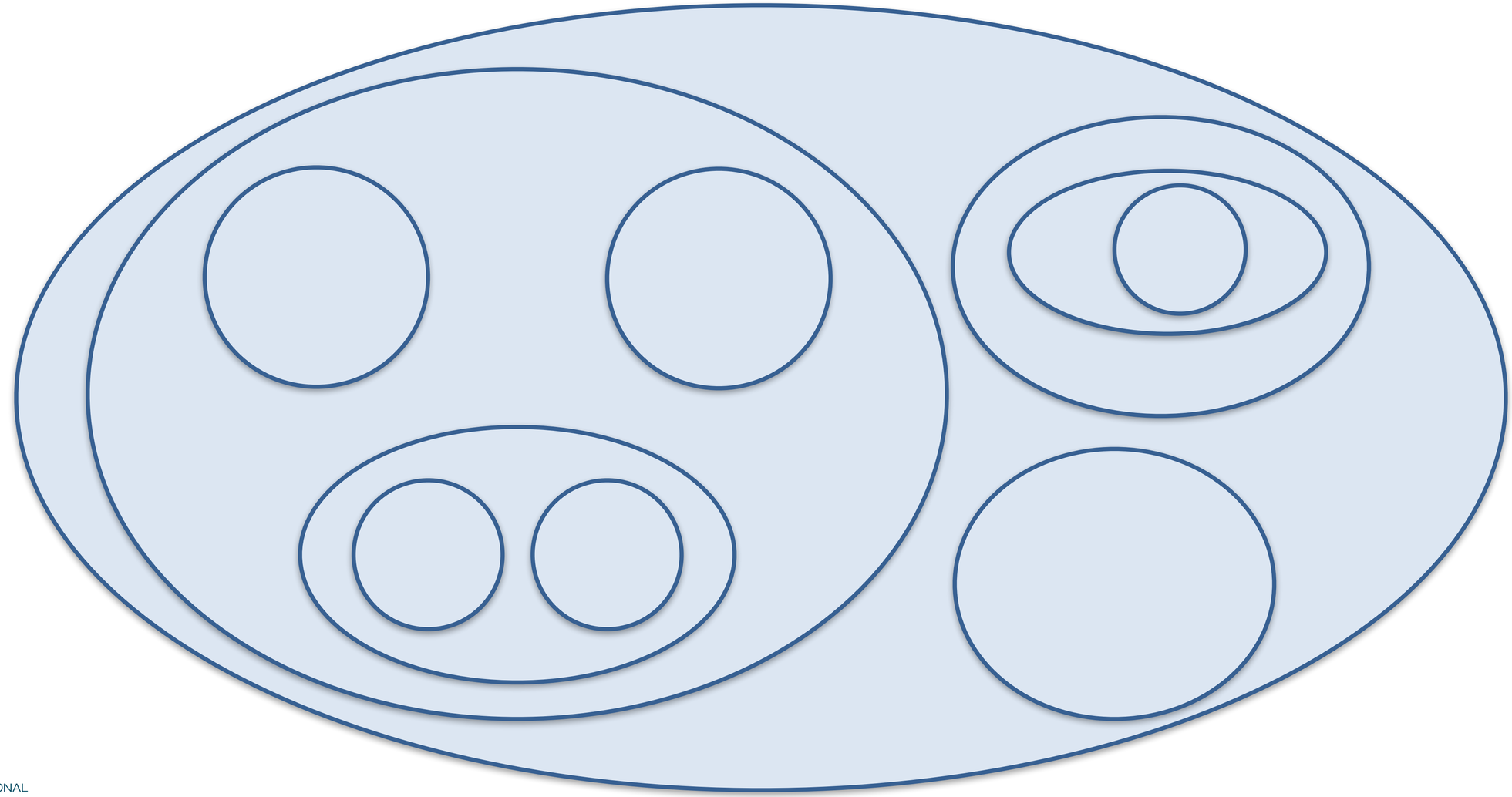
Мультигранулярные блокировки

- База данных представляется как дерево вложенных объектов
- Типы блокировок
  - S: совместное использование
  - X: монопольное использование
  - IS: совместное использование вложенных гранул
  - IX: монопольное использование вложенных гранул
  - SIX: совместное использование гранулы, монопольное для вложенных гранул

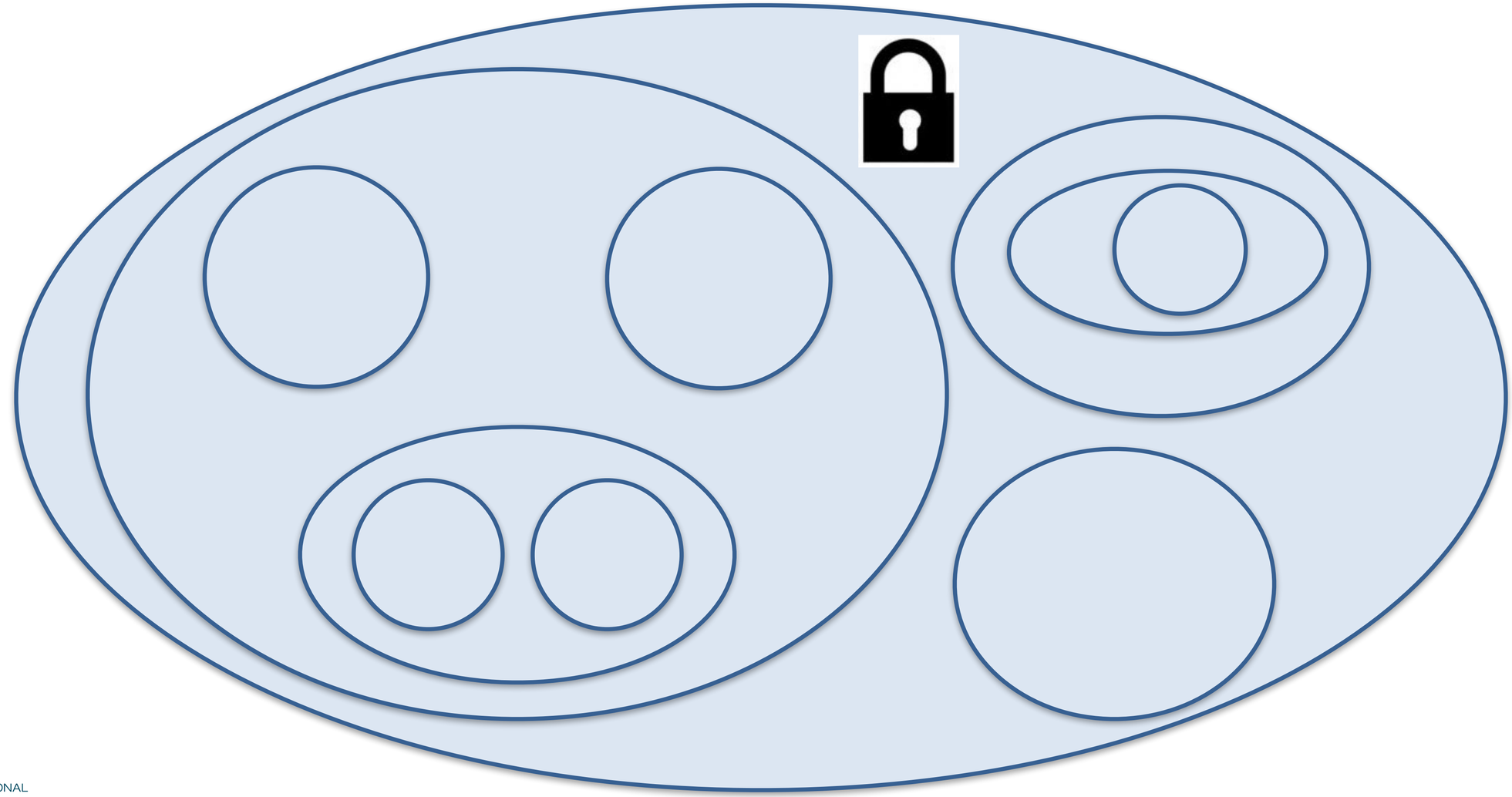
Мультигранулярный 2PL

- Для установки S/X необходимо иметь IS/IX на охватывающую гранулу
- Правила совместимости блокировок основаны на коммутативности
- Обычные правила 2PL для всех блокировок

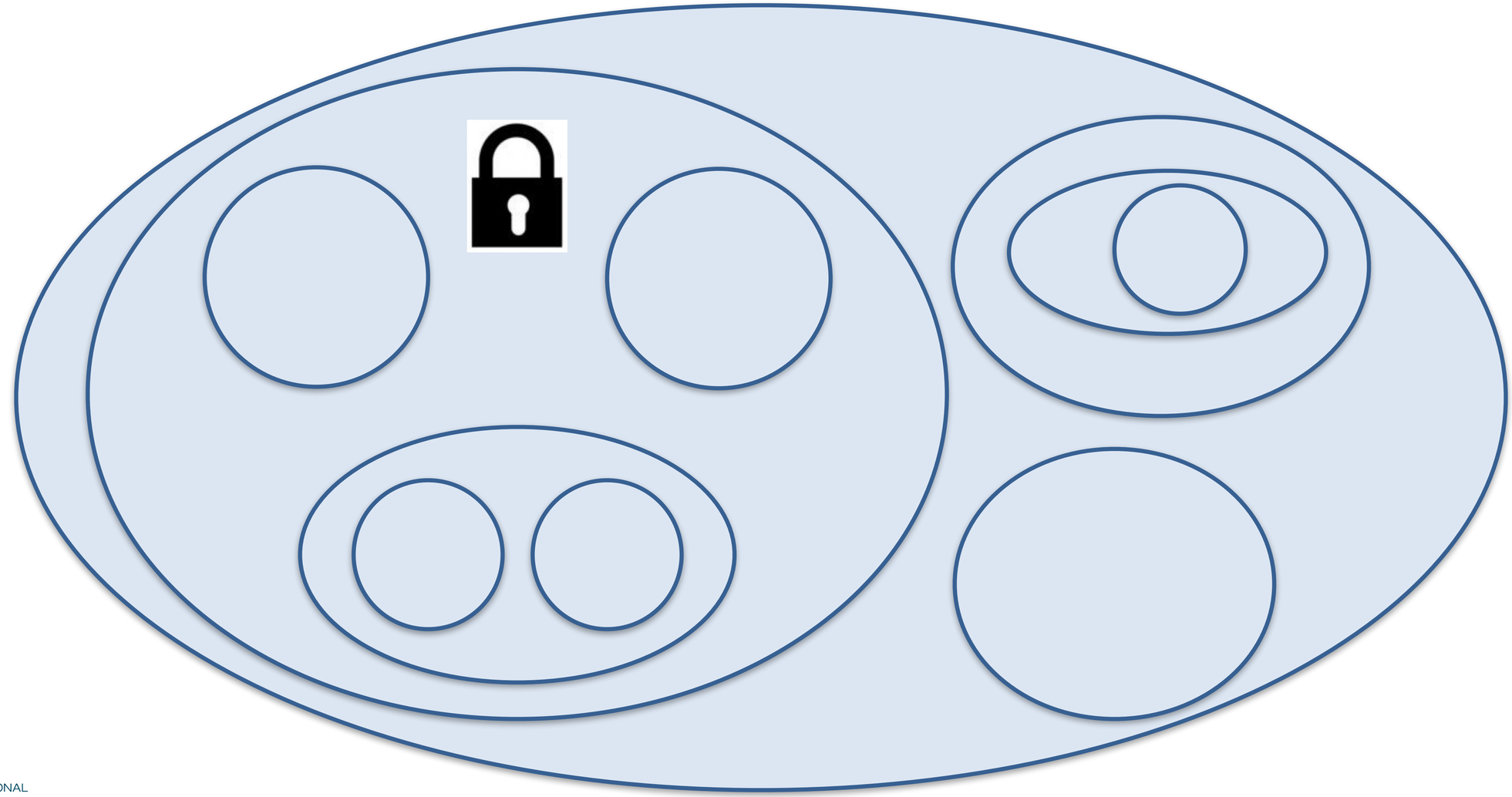
Мультигранулярный 2PL



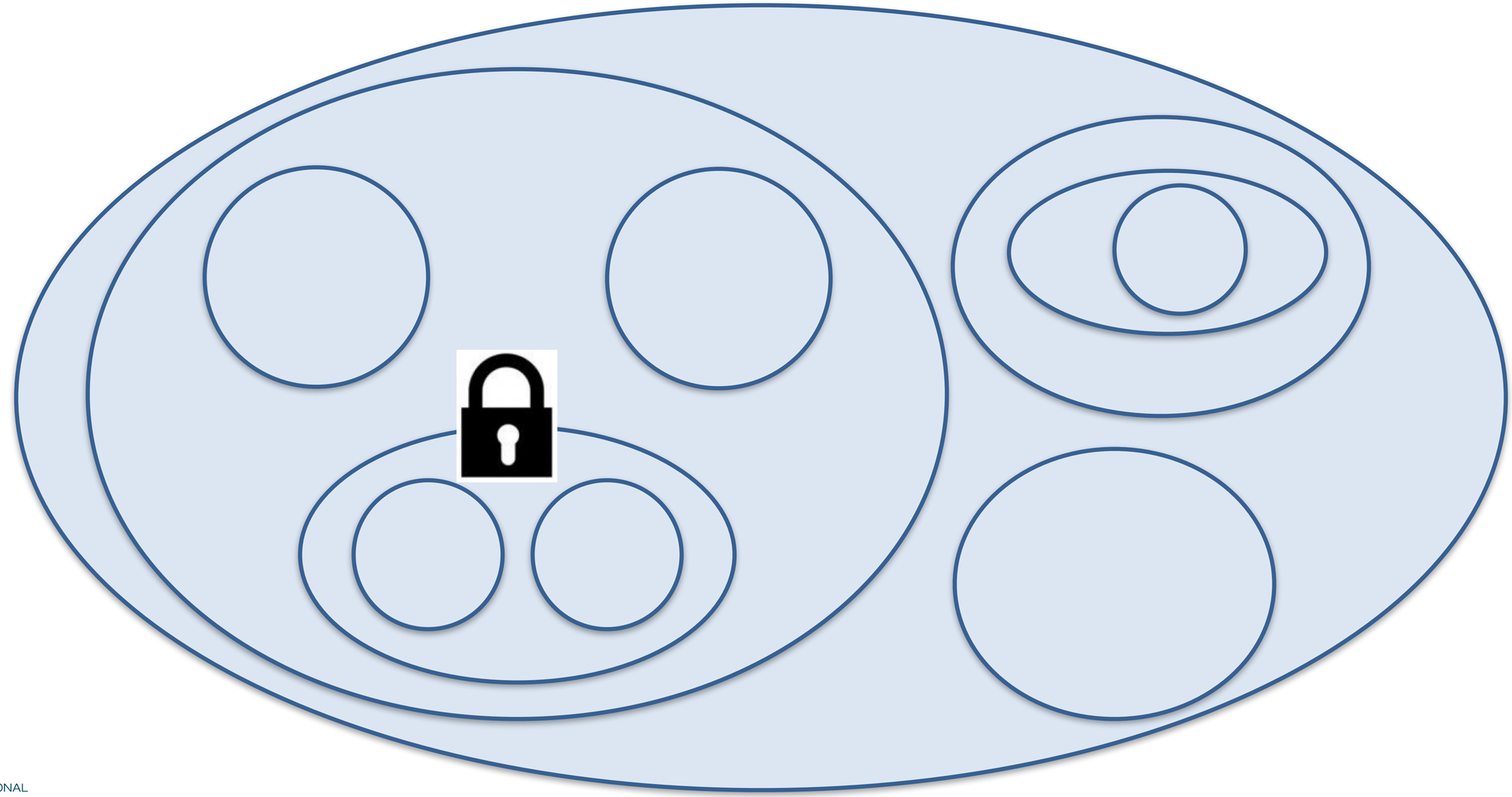
Мультигранулярный 2PL



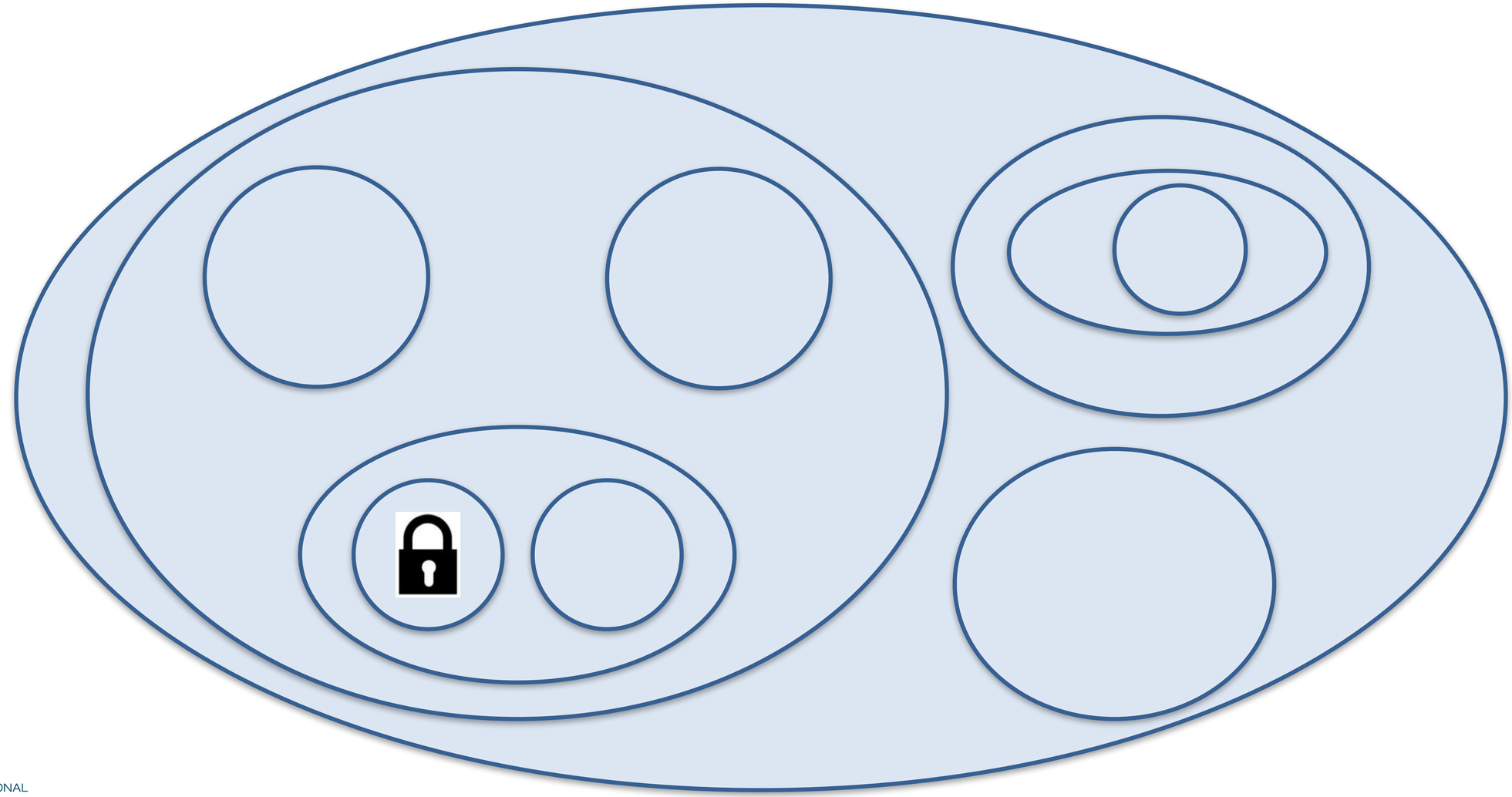
Мультигранулярный 2PL



Мультигранулярный 2PL



Мультигранулярный 2PL



Уровни изоляции SQL

- Read Uncommitted
- Read Committed
- Repeatable Read
- Serializable

- Snapshot Isolation — не определяется в стандарте, но используется в промышленных СУБД, иногда вместо сериализуемости

Критерии для оценки транзакционных протоколов

- Степень конкурентности: максимальное количество конкурентно выполняемых транзакций
- Частота обрывов: доля транзакций, которые обрываются протоколом в целях обеспечения корректности
- Пропускная способность системы
- Способ измерения: на эталонных тестах (TPC-C и т. п.)

Проектирование транзакций в приложении

- Атомарность: вывод результатов только после фиксации транзакции
- Блокировки не должны удерживаться в течение длительного времени
- Недопустимо взаимодействие с пользователем, пока транзакция не завершена
- Ослабленные критерии согласованности при вычислении агрегированных значений

Многоверсионные протоколы

- Основаны на временном хранении нескольких версий элементов данных
- Версии прозрачны для приложения
- Обеспечивают нормальное выполнение «опоздавших» операций чтения
- «Только читающие» транзакции никогда не обрываются

Протокол Snapshot Isolation (SI)

- Практика опередила теорию
- Для каждой транзакции определяются:
  - Интервал времени $I(t) = [\text{sts}(t), \text{cts}(t)]$
  - Write set $W(t)$

- Ограничение протокола:

$$I(t_1) \cap I(t_2) = \varnothing \vee W(t_1) \cap W(t_2) = \varnothing$$

- При обнаружении ограничения транзакция обрывается
- Чтение: по состоянию на момент начала транзакции

Реализация SI на основе блокировок

- Блокировки устанавливаются только на запись
- При невозможности установить блокировку транзакция обрывается
- Чтение зафиксированных (committed) состояний

Аномалии, связанные с SI

- Несогласованная запись

$$r_1(x)\ r_2(x)\ r_1(y)\ r_2(y)\ w_1(x)\ w_2(y)\ c_1\ c_2$$

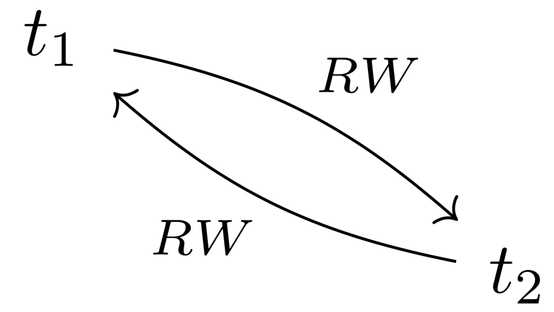
- Аномалия только читающей транзакции

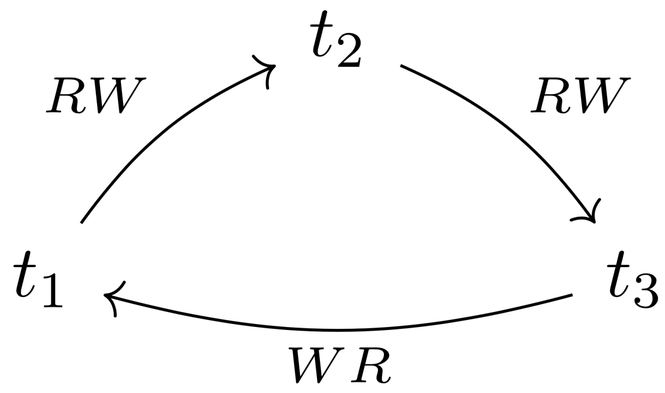
$$r_1(x)\ r_2(x)\ r_2(y)\ w_2(x)\ c_2\ r_3(y)\ w_3(z)\ c_3\ r_1(z)\ c_1$$

Serializable SI (SSI)

- Зависимости RW, WW, RW между транзакциями
- WW исключается протоколом SI
- Граф сериализуемости на основе зависимостей

- Запрещенные подграфы:





Оптимистический протокол управления транзакциями

- Фазы выполнения транзакции
  - чтение
  - проверка
  - запись
- Проверка назад: проверяются конфликты с зафиксированными транзакциями
- Проверка вперед: проверяются конфликты с активными транзакциями

Восстановление после отказов

Типы отказов

- Отказы приложений/транзакции: транзакции обрываются
- Отказы системы: функционирование возобновляется после рестарта и восстановления согласованности БД
- Разрушение носителя: рестарт, восстановление с резервной копии, восстановление согласованности

Восстановление оборванных транзакций: откат

- Обрывы транзакций могут быть вызваны ошибками при выполнении приложений или невозможностью выполнить транзакцию сериализуемым образом

- Операции обрыва a можно заменить на выполнение отката для всех операций записи w^{-1}, включаемых в расписание в обратном порядке с последующей фиксацией

Восстановимость

- Восстановимость расписаний: транзакции должны фиксироваться до того, как их результаты используются другими транзакциями

- Пример невосстановимого расписания:

$$w_1(x)\ w_2(x)\ c_2\ a_1$$

- S2L (двухфазный протокол с удержанием блокировок на запись до конца транзакции) гарантирует восстановимость

Восстановление после системных отказов

- Необходим рестарт сервера БД
- Для того, чтобы восстановление было возможным, необходимо вести журнал обновлений
- При нормальной работе БД все изменения записываются в журнал
- При рестарте журнал используется для повторения операций и для отката незавершенных транзакций

Ведение журнала

- Журнал ведется последовательно
- Опережающая запись в журнал (WAL, write-ahead log)
- Регистрируются операции записи, начала и конца транзакций
- Каждая запись может (должна) содержать данные отката (undo) и «наката» (redo)
- При фиксации запись журнала обязательно выталкивается на диск

Что нужно сделать при восстановлении

- Определить, какие транзакции были зафиксированы до отказа и какие были активными (не были завершены)
- Если необходимо, повторить изменения зафиксированных транзакций
- Оборвать незавершенные до отказа транзакции (выполнить откат)

Запись обновлений базы данных

	Force	**No force**
No steal	Только при фиксации undo и redo не нужны	Только после фиксации только данные redo
Steal	Обязательно до фиксации только undo	Полностью асинхронно требуются undo и redo

Алгоритм восстановления при рестарте

- Фаза просмотра: найти все зафиксированные и активные транзакции (прямой просмотр журнала)
- Фаза наката (redo): при прямом просмотре выполнить все операции зафиксированных транзакций
- Фаза отката (undo): обратный просмотр журнала, откат всех операций незавершенных транзакций

Завершение восстановления

- После завершения фазы отката необходимо
  - записать на диск все измененные блоки БД
  - после записи БД можно очистить журнал
- Возобновить нормальную работу системы

Корректность алгоритма восстановления

- Восстанавливает согласованное состояние (такое, как после всех зафиксированных транзакций)

- Допускает многократные рестарты

Алгоритм восстановления Redo History

- Анализ: обнаружение транзакций, подлежащих обрыву
- Redo: повторное выполнение всех операций, записанных в журнал
- Undo: откат незафиксированных транзакций
- Первые две фазы можно совместить

Контрольные точки

- Для сокращения времени восстановления записываются контрольные точки
- При записи контрольной точки все измененные блоки записываются на диск и делается запись о контрольной точке в журнале
- При рестарте фазу наката можно начинать с последней контрольной точки

Контрольные точки малого веса

- Запись журнала содержит информацию об активных транзакциях и список грязных страниц
- Запись грязных страниц выполняется асинхронно
- Все страницы, включенные в контрольную точку, должны быть записаны до следующей контрольной точки
- Восстановление можно начинать с предпоследней контрольной точки

Характеристики защищенности

- Доступность
- Задержка резервирования
- Выживаемость

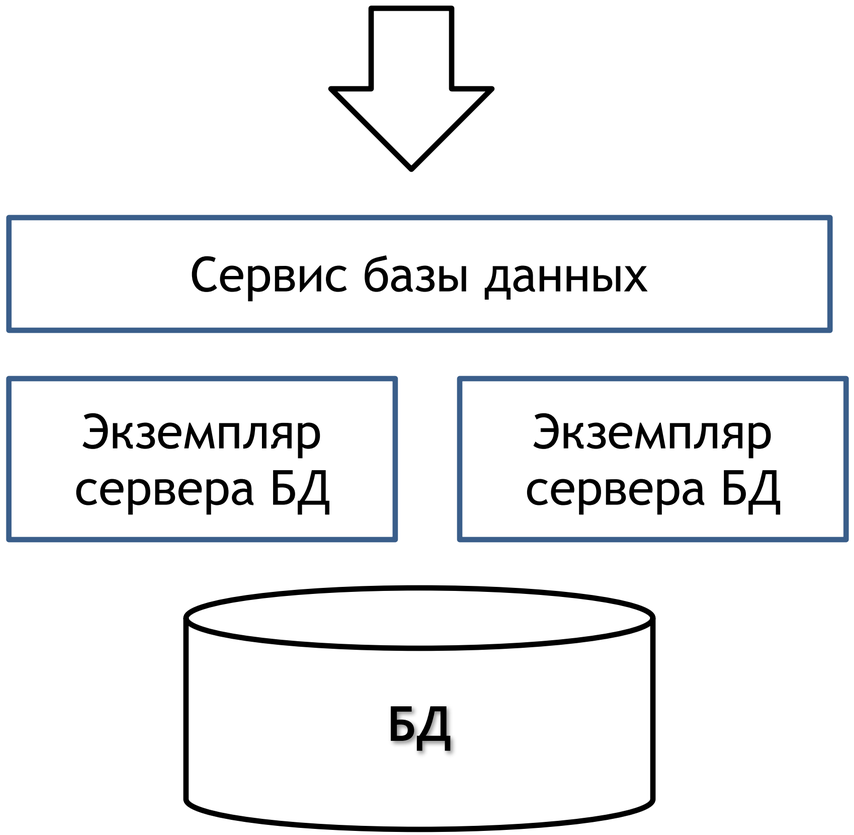
Защита от разрушения носителя

- Создание резервных копий
  - экспорт данных
  - offline backup
  - копирование средствами операционной системы
  - online backup
- Специальное оборудование (зеркальные дисковые системы)
- Серверы в горячем резерве (stand-by)

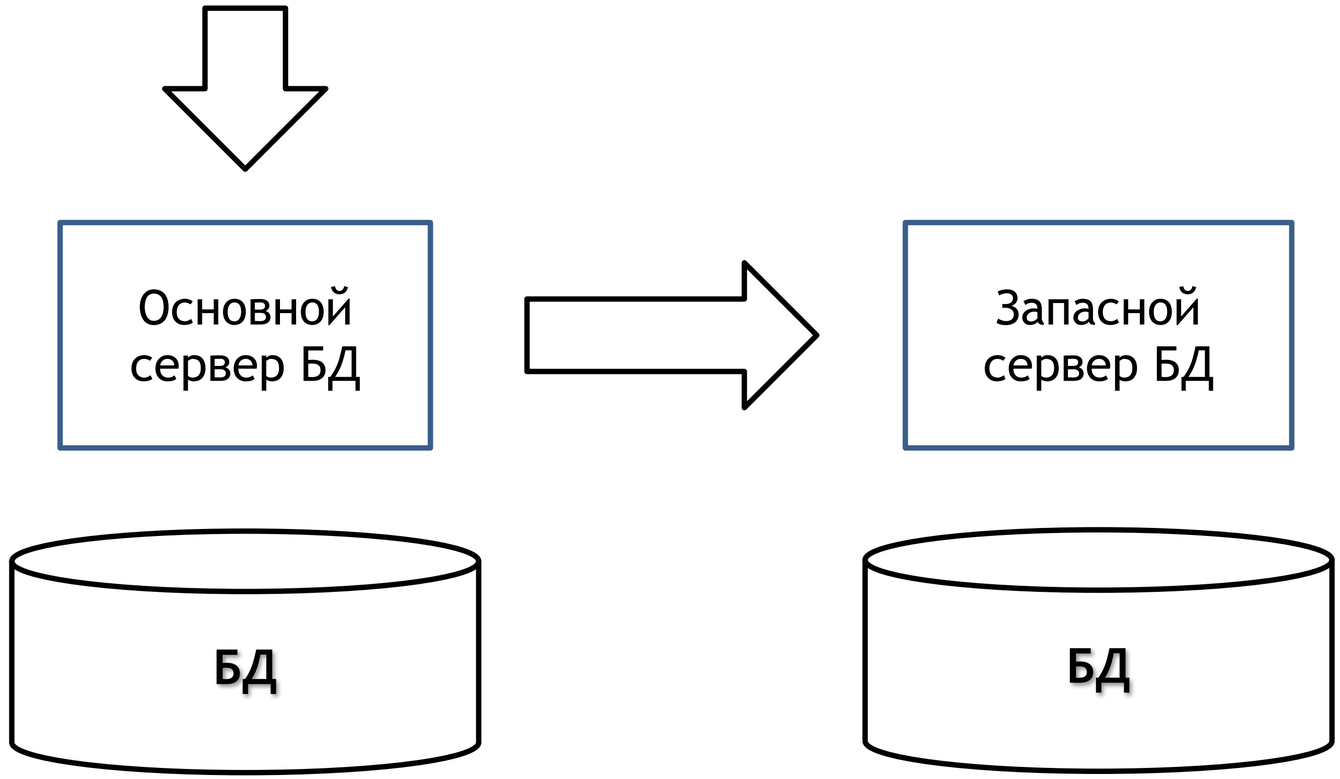
Копии в горячем резерве

- Основной и дополнительные серверы
- Распространение изменений
  - синхронное внесение изменений
  - повторение транзакций
  - пересылка и повторение журнала
  - накат по журналу с задержкой

Кластер с зеркальными дисками



Сервер в горячем резерве



Согласованность и надежность: итоги

- Согласованность необходима для сохранения корректности данных
- Простые протоколы ограничивают пропускную способность системы
- Многоверсионные протоколы SI существенно превосходят протоколы двухфазового блокирования
- Средства защиты от отказов позволяют обеспечить любую степень надежности
- Создание и эксплуатация высоконадежных систем оказываются дорогостоящими