Обработка транзакций. Журнал транзакций. Транзакции и восстановление

Задание

1. Изучить материал лекции, законспектировать.
2. Фотоотчет прислать на электронную почту

С уважением, Хвастова Светлана Ивановна

!!! Если возникнут вопросы обращаться по телефону 0721389311. Электронная почта: xvsviv@rambler.ru

Определение

Транзакция – это группа инструкций SQL, исполняемых как единое целое. Невыполнение одной инструкции приводит к откату состояния данных на начало транзакции.

Реализация

- Транзакция состоит из некоторого набора (блока) команд. Если хотя одна инструкция была выполнена с ошибкой, то происходит откат транзакции, т. е. отменяется выполнение всех ее операторов.
- Все операции транзакции запоминаются в некотором журнале - журнале транзакций.
- Каждая отдельная команда *Transact-SQL* выполняется как самостоятельная транзакция.
- Завершенная транзакция фиксируется (оператор **commit**) в базе данных, которая переходит в новое согласованное состояние, а все обновления, выполненные в базе данных за время транзакции, фиксируются, т.е. становятся постоянными.
- Если обновление данных в базе было прервано каким-либо условием ошибки, то выполняется оператор **rollback** и любые изменения отменяются.

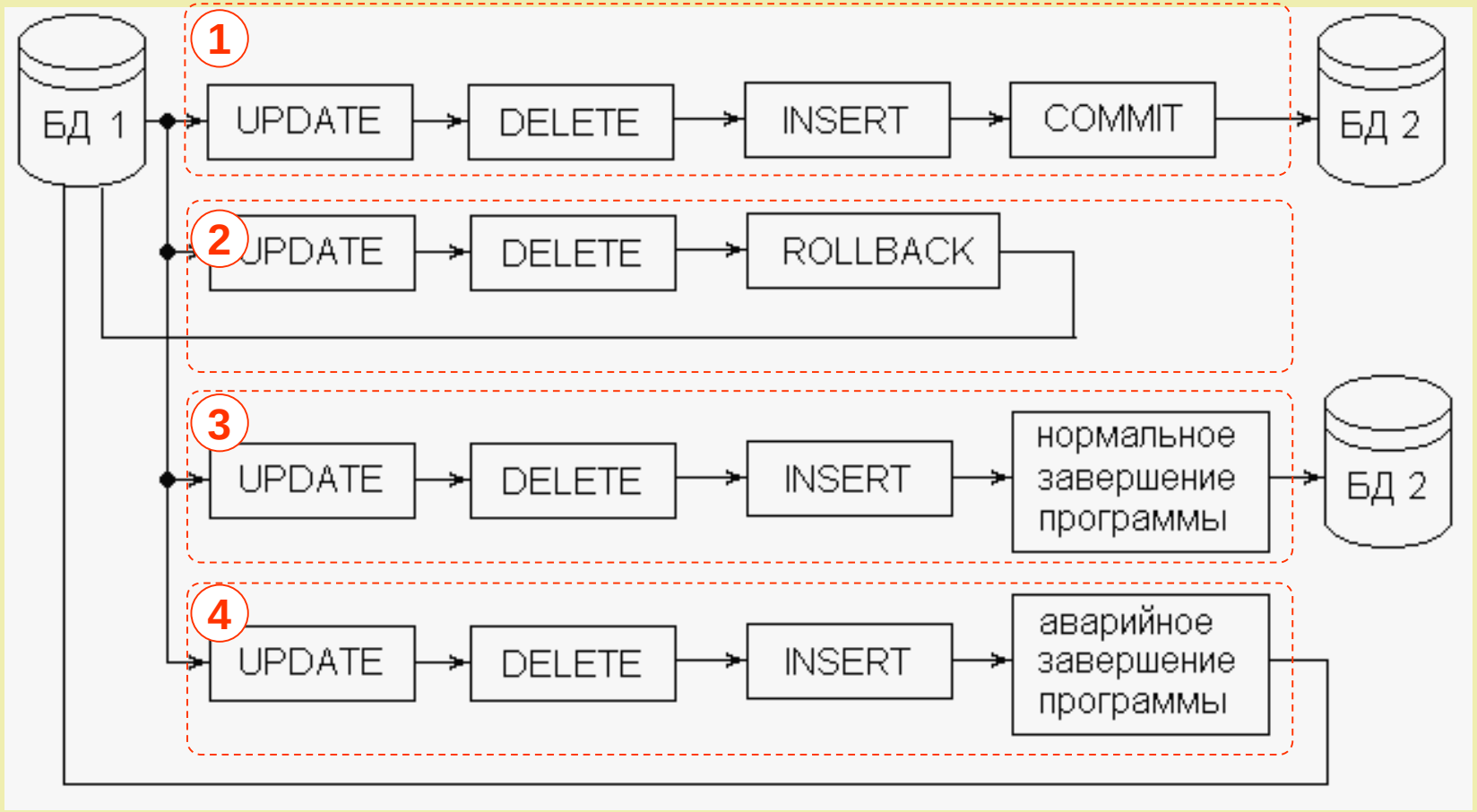
Свойства ACID

- Атомарность (Atomicity)
- Согласованность или непротиворечивость (Consistency)
- Изолированность (Isolation)
- Долговечность (Durability)

3.2. Типы транзакций

1. Явные транзакции
2. Автоматические транзакции
3. Неявные транзакции
   - alter table — изменение таблицы;
   - create — создание любого объекта базы данных;
   - delete — удаление строк данных из таблицы;
   - drop — удаление объектов базы данных;
   - fetch — извлечение столбца из курсора;
   - grant — разрешение доступа к объектам базы данных;
   - insert — вставка строк в таблицу;
   - open — открытие курсора;
   - revoke — неявное отклонение доступа к объектам базы данных;
   - select — выборка данных из одной или нескольких таблиц;
   - truncate table — усечение таблицы;
   - update — изменение содержимого таблицы.
4. Распределенные транзакции (Distributed transaction).
5. Вложенные транзакции (Nested transaction)

Возможные схемы завершения транзакций



3.3. Проблемы одновременного доступа к данным

- Проблема потери результатов обновления
- Проблема незафиксированной зависимости (чтение "грязных" данных, неаккуратное считывание)
- Неповторяющиеся считывания
- Фантомные считывания (фиктивные элементы)
- Собственно несовместимый анализ

Пример

Рассмотрим две транзакции, ***A*** и ***B***, запускающиеся в соответствии с некоторым графиком. Пусть транзакции работают с некоторыми объектами базы данных, например со строками таблицы. Операцию чтение строки будем обозначать $P=P_0$, где P_0 – прочитанное значение. Операцию записи значения P_1 в строку будем обозначать $P_1 \rightarrow P$.

Проблема потери результатов обновления

Причина

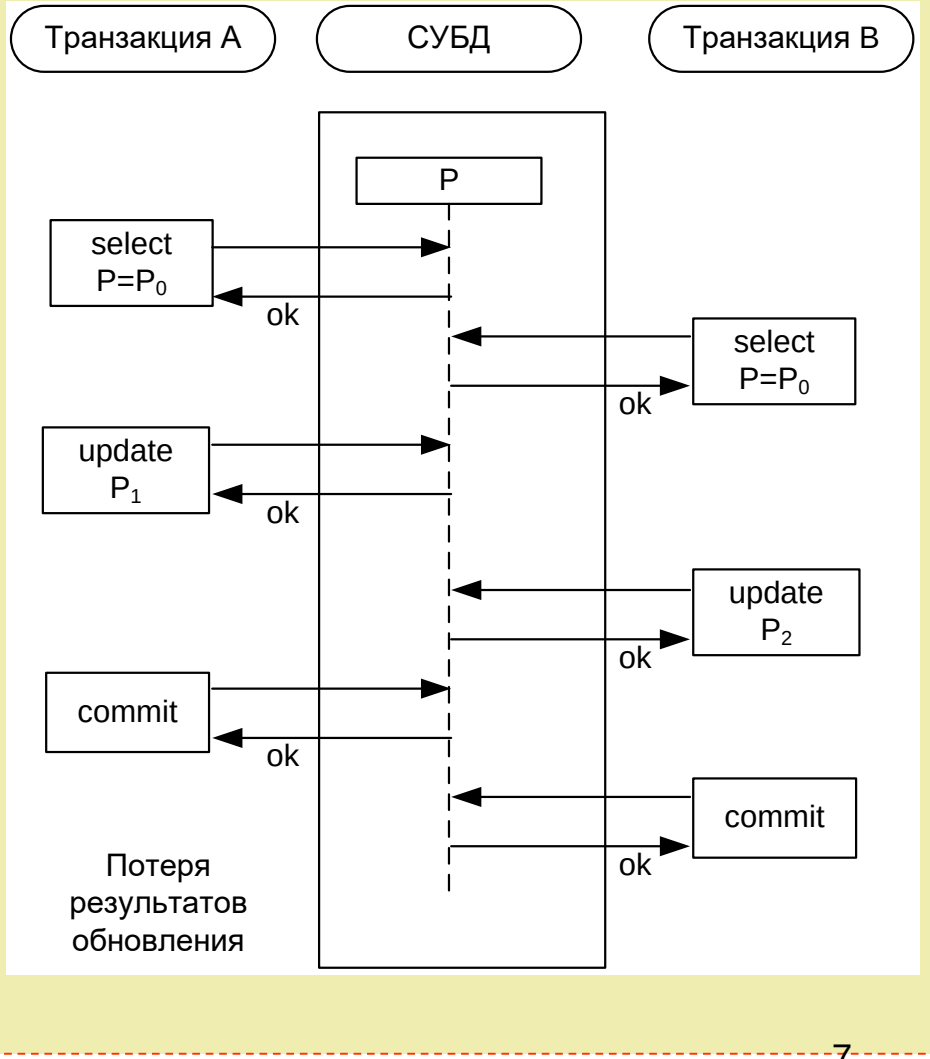
Использование неизолированных друг от друга транзакций. По этой причине несколько транзакций могут считывать и изменять одни и те же данные. В результате все изменения кроме последнего будут потеряны.

Ситуация

Две транзакции по очереди записывают некоторые данные в одну и ту же строку и фиксируют изменения.

Результат

После окончания обеих транзакций, строка ***P*** содержит значение P_2, занесенное более поздней транзакцией ***B***. Транзакция ***A*** ничего не знает о существовании транзакции ***B***, и естественно ожидает, что в строке ***P*** содержится значение P_1. Таким образом, транзакция ***A*** потеряла результаты своей работы.



Проблема незафиксированной зависимости (чтение "грязных" данных, неаккуратное считывание)

Причина

Одни и те же данные будут одновременно читаться и изменяться разными транзакциями. Полученная информация может оказаться противоречивой, т.к.:

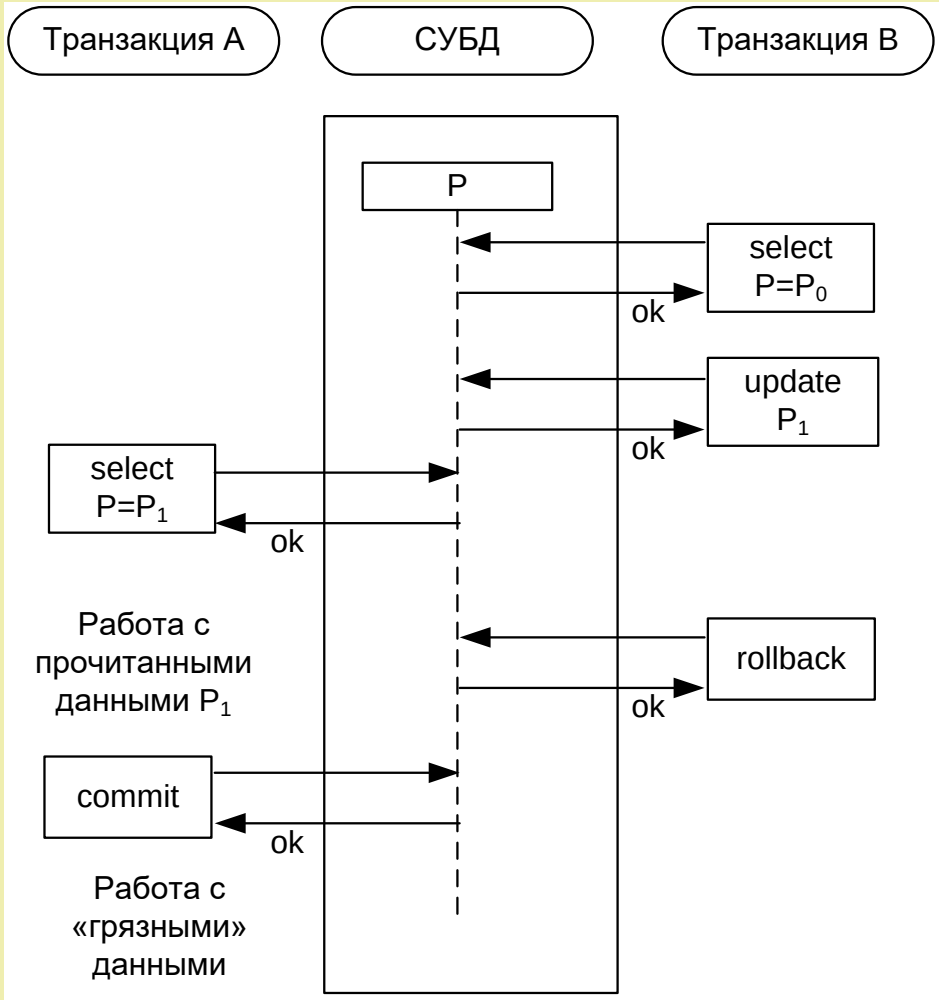
- транзакция еще не закончила все изменения данных (например, корректировка в связанных таблицах — часть данных уже изменилась, а часть нет);
- поскольку транзакция еще не закончена, то вообще неочевидно, что она будет зафиксирована. В случае ошибки произойдет ее откат. Получится так, что другая транзакция прочитала модифицированные данные, которые ни самом деле не изменились.

Ситуация

Транзакция ***B*** изменяет данные в строке. После этого транзакция ***A*** читает измененные данные и работает с ними. Транзакция ***B*** откатывается и восстанавливает старые данные.

Результат

Транзакция ***A*** в своей работе использовала данные, которых нет в базе данных. Более того, транзакция ***A*** использовала данные, которых нет, и ***не было*** в базе данных! Действительно, после отката транзакции ***B***, должна восстановиться ситуация, как если бы транзакция ***B*** ***вообще никогда не выполнялась***. Таким образом, результаты работы транзакции ***A*** некорректны, т.к. она работала с данными, отсутствовавшими в базе данных.



Неповторяющиеся считывания

Причина

Одна транзакция в процессе работы многократно считывает данные, а в это время вторая транзакция их изменяет. Первая транзакция, по крайней мере, дважды получит разные значения одних и тех же данных.

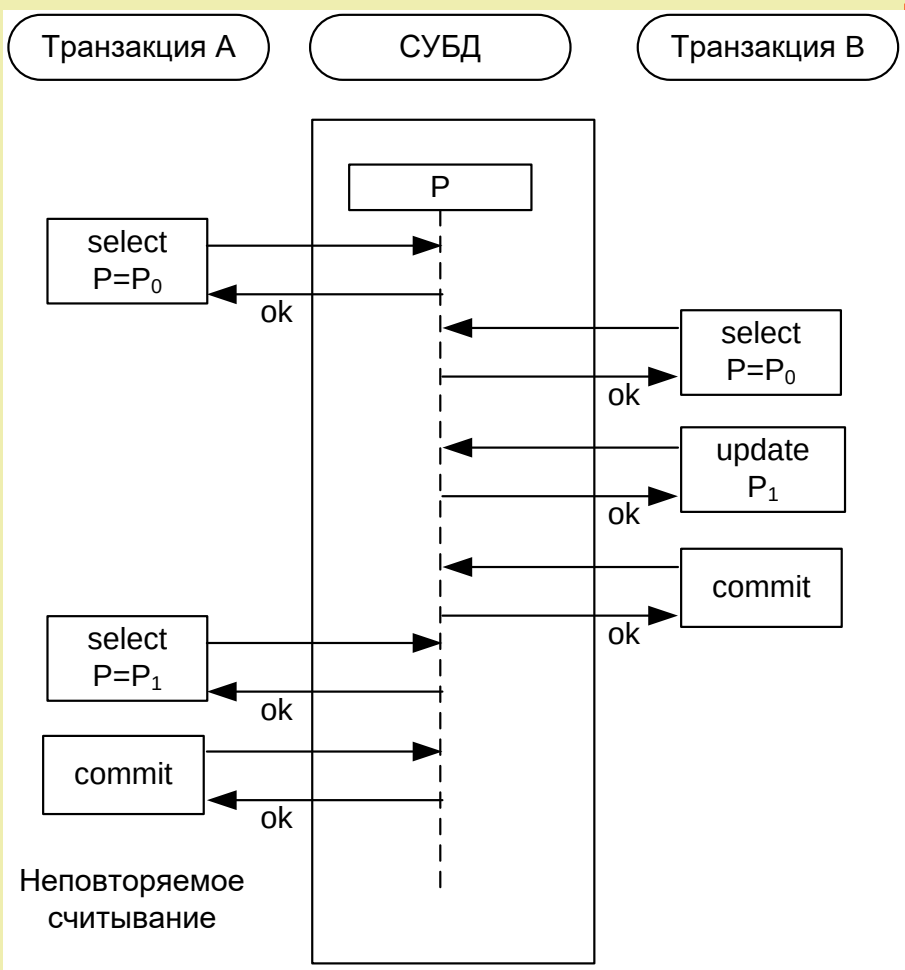
Ситуация

Транзакция ***A*** дважды читает одну и ту же строку. Между этими чтениями вклинивается транзакция ***B***, которая изменяет значения в строке.

Транзакция ***A*** ничего не знает о существовании транзакции ***B***, и т.к. сама она не меняет значение в строке, то ожидает, что после повторного чтения значение будет тем же самым.

Результат

Транзакция ***A*** работает с данными, которые, с точки зрения транзакции ***A***, самопроизвольно изменяются.

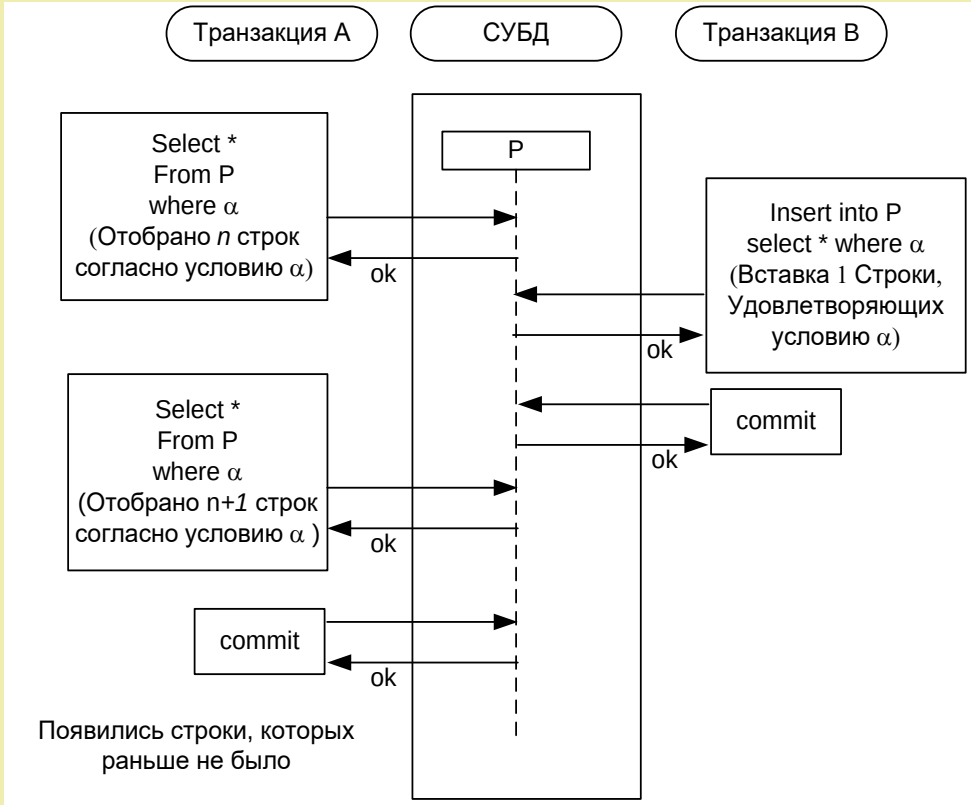


Фантомные считывания (фиктивные элементы)

Причина

Одна транзакция считывает данные из таблицы, а вторая в этот момент вставляет (или удаляет) строки. Результат выполнения первой транзакции может оказаться для клиента весьма неожиданным.

Эффект фиктивных элементов отличается от предыдущих транзакций тем, что здесь за один шаг выполняется достаточно много операций - чтение одновременно нескольких строк, удовлетворяющих некоторому условию.



Ситуация

Транзакция ***A*** дважды выполняет выборку строк с одним и тем же условием. Между выборками вклинивается транзакция ***B***, которая добавляет новую строку, удовлетворяющую условию отбора.

Транзакция ***A*** ничего не знает о существовании транзакции ***B***, и т.к. сама она не меняет ничего в базе данных, то ожидает, что после повторного отбора будут отобраны те же самые строки.

Результат

Транзакция ***A*** в двух одинаковых выборках строк получила разные результаты.

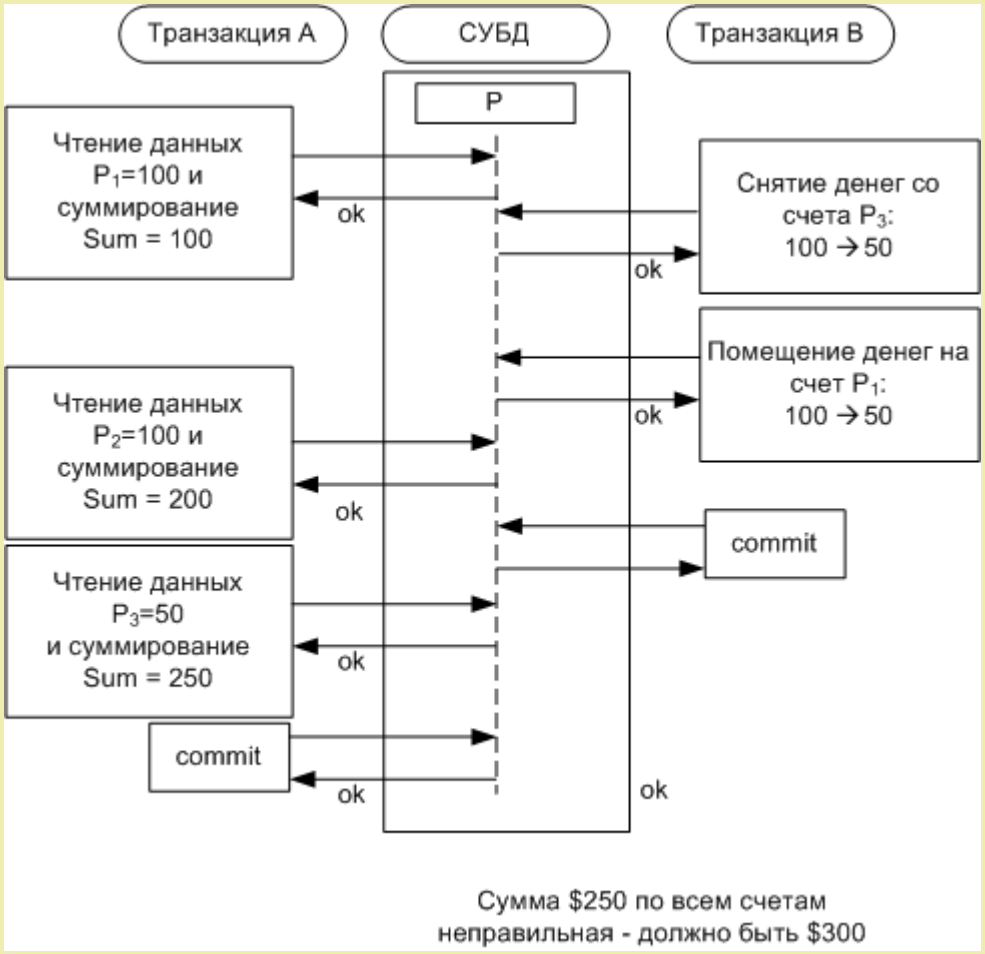
Собственно несовместимый анализ

Ситуация

Отличается от предыдущих примеров тем, что в смеси присутствуют две транзакции – одна длинная, другая короткая.

Длинная транзакция выполняет некоторый анализ по всей таблице, например, подсчитывает общую сумму денег на счетах клиентов банка для главного бухгалтера.

Пусть на всех счетах находятся одинаковые суммы, например, по $100. Короткая транзакция в этот момент выполняет перевод $50 с одного счета на другой так, что общая сумма по всем счетам не меняется.



Результат

Хотя транзакция ***B*** все сделала правильно - деньги переведены без потери, но в результате транзакция ***A*** подсчитала неверную общую сумму.

Т.к. транзакции по переводу денег идут обычно непрерывно, то в данной ситуации следует ожидать, что главный бухгалтер никогда не узнает, сколько же денег в банке.

Принципы решения проблем

1. Согласованные состояния

В процессе выполнения транзакции пользователю доступны только согласованные состояния БД, которые фиксируются оператором COMMIT;

2. Гарантированный результат

Параллельное выполнение двух транзакций приводит к одному и тому же результату
независимо от последовательности их реализации.

3. Сериализация транзакций

Сериализация (реализация серии транзакций, график их выполнения) транзакций гарантирует, что каждый пользователь программы работает с БД так, как будто не существует других пользователей (программ),
обращающихся к этой БД.

3.4. Блокировка транзакций

Блокировка (lock) – временно накладываемое ограничение на выполнение некоторых операций обработки данных

Идея

В случае, когда для выполнения некоторой транзакции необходимо, чтобы некоторый объект не менялся непредсказуемо
и без ведома этой транзакции, такой объект блокируется

Результат

Эффект блокировки состоит в том, чтобы заблокировать доступ к этому объекту со стороны других транзакций, а значит, предотвратить непредсказуемое изменение этого объекта.

Уровни блокировок (*протокол разработан ANSI*)

Уровень 0 – **запрет «загрязнения» данных (no trashing of data).**

Изменять данные может только одна транзакция. Если другой транзакции необходимо изменить эти же данные, она должна дождаться завершения первой транзакции

Уровень 1 – **запрет на «грязное» чтение (no dirty reads).**

Если транзакция начала изменение данных, то никакая другая транзакция не сможет прочитать эти данные до тех пор, пока первая транзакция не завершится

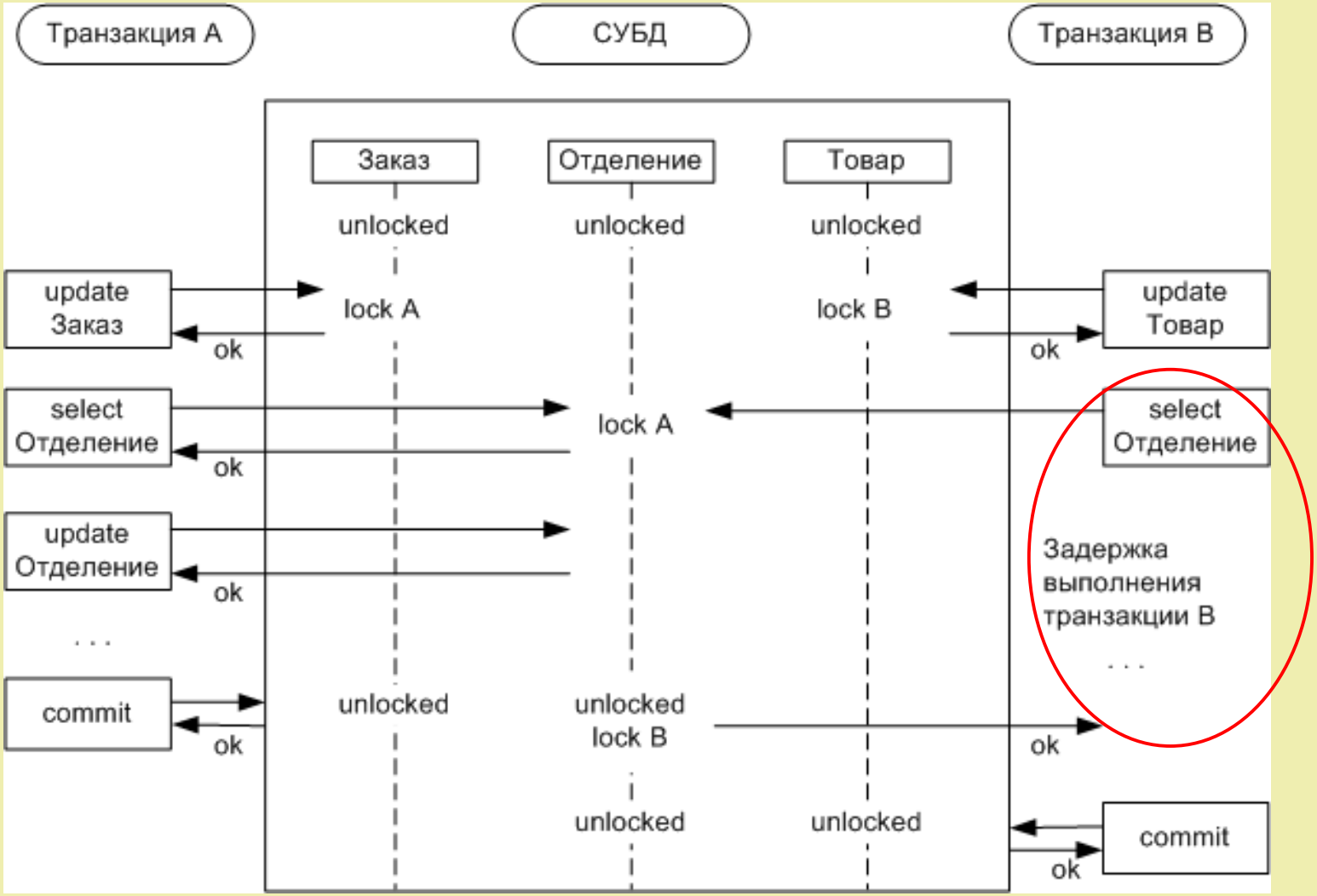
Уровень 2 – **запрет неповторяемого чтения (no nonrepeatable reads).**

Если транзакция считывает данные, то никакая другая транзакция не сможет их изменить. Таким образом, при повторном чтении данных они будут находиться в исходном состоянии.

Уровень 3 – **запрет фантомов (no phantom)**

Если транзакция обращается к данным, то никакая другая транзакция не сможет добавить/удалить строки, которые могут быть считаны при выполнении транзакции. Реализация этого уровня достигается блокированием диапазона ключей

Пример выполнения транзакций с использованием блокировок



Типы блокировок

S-блокировка (S-locks – Shared locks)

Блокировка с взаимным доступом (коллективная блокировка, блокировка на чтение).

Устанавливается при выполнении операции чтения данных.

Этот вид блокировки на одни и те же данные могут устанавливать несколько транзакций одновременно.

Она разрешает чтение заблокированных данных конкурирующим транзакциям, однако блокированные данные защищены от изменений;

X-блокировка (X-locks – eXclusive locks).

Блокировка без взаимного доступа (монопольная блокировка, блокировка на запись).

Если этот вид блокировки накладывается на ресурс, то никакая другая транзакция не сможет прочитать или изменить эти данные.

Матрица совместимости захватов

		Транзакция 2		
		unlocked	S	X
Транзакция 1	unlocked	да	да	да
	S	да	да	нет
	X	да	нет	нет

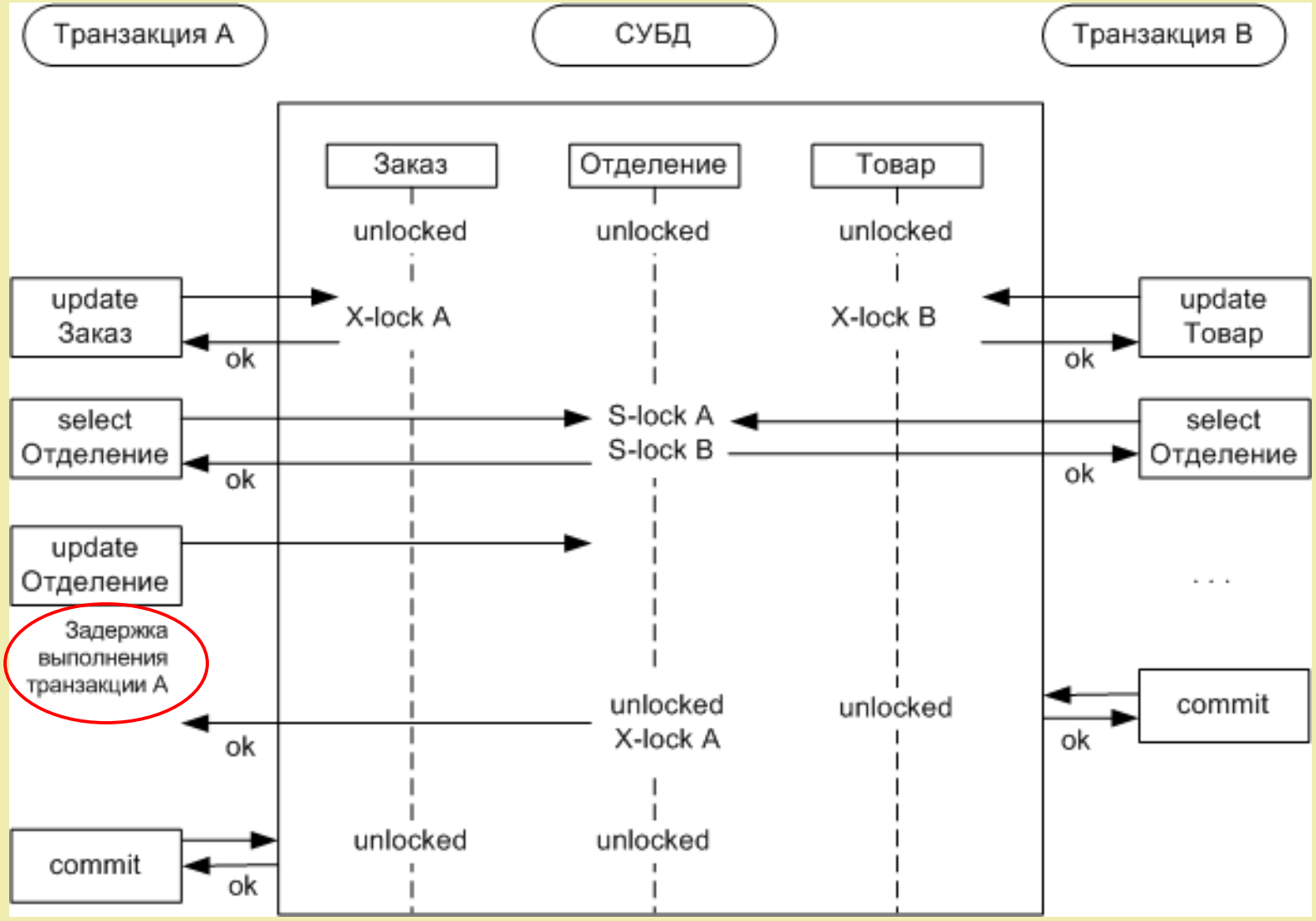
- захваты одних и тех же данных несколькими транзакциями по чтению совместимы;
- захват на чтение одной транзакции не совместим с захватом того же объекта другой транзакцией по записи;
- захваты одного и того же объекта по записи несовместимы.

1. Транзакция, предназначенная для извлечения данных, прежде всего должна наложить S-блокировку на эти данные.

2. Транзакция, предназначенная для обновления данных, прежде всего должна наложить X-блокировку на эти данные.

3. Если запрашиваемая блокировка со стороны транзакции ***B*** отвергается из-за конфликта с некоторой другой блокировкой со стороны транзакции ***A***, то транзакция ***B*** переходит в состояние ожидания до тех пор, пока не будет снята блокировка, заданная транзакцией ***A***.

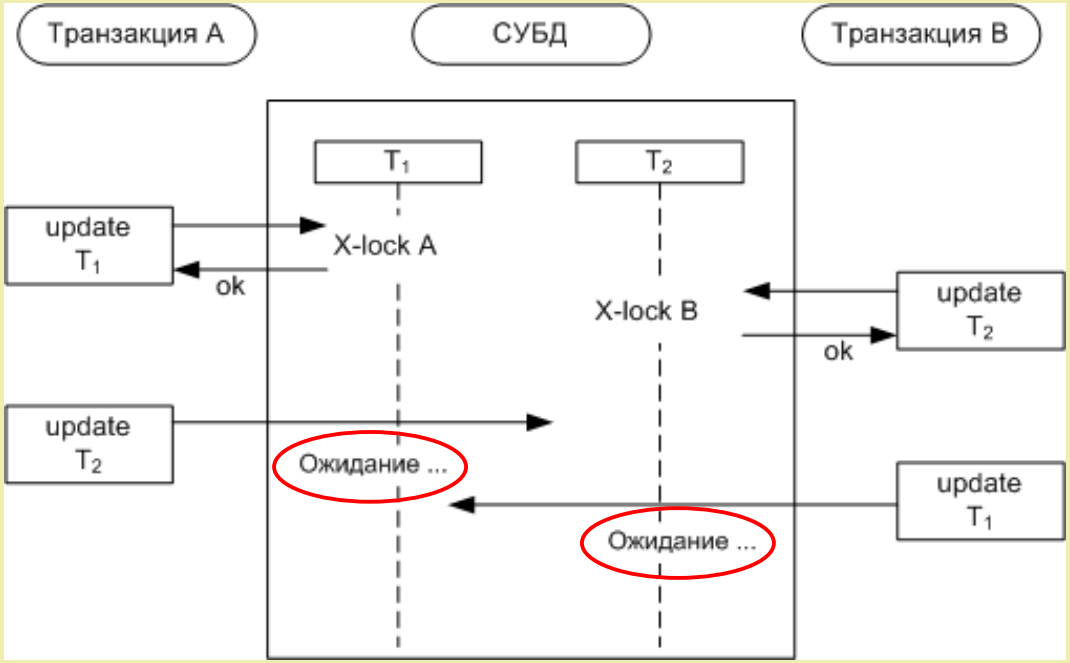
Пример выполнения транзакций с использованием S- и X-блокировок



3.5. Тупиковые ситуации, их распознавание и разрушение

Возникновение «мертвых блокировок» (deadlocks)

Пусть первая транзакция ***A*** блокирует данные T_1, а вторая транзакция ***B*** – данные T_2. Причем, для завершения первой транзакции ей необходим доступ к данным T_2, а для завершения второй – доступ к данным T_1.



3.6. Механизмы фиксаций (откатов) транзакций

Принципы восстановления данных

- Результаты зафиксированных транзакций должны быть сохранены в восстановленном состоянии базы данных;
- Результаты незафиксированных транзакций должны отсутствовать в восстановленном состоянии базы данных.

Ситуации, требующие восстановление состояния базы данных

- Индивидуальный откат транзакции
- Восстановление после внезапной потери содержимого оперативной памяти (мягкий сбой).
- Восстановление после поломки основного внешнего носителя базы данных (жесткий сбой).

3.7. Журнализация и буферизация

Журнал транзакций

Особая часть базы данных, недоступная пользователям СУБД и поддерживаемая с особой тщательностью, в которую поступают записи обо всех изменениях основной части БД

Виды буферов

- буфер журнала;
- буфер страниц оперативной памяти

Основной принцип согласованной политики выталкивания буфера журнала и буферов страниц базы данных

Запись об изменении объекта базы данных должна попадать во внешнюю память журнала раньше, чем измененный объект оказывается во внешней памяти базы данных.

Протокол журнализации Write Ahead Log (WAL) – «пиши сначала в журнал»

Если требуется вытолкнуть во внешнюю память измененный объект базы данных, то перед этим нужно гарантировать выталкивание во внешнюю память журнала записи о его изменении.

3.8. Принципы восстановления базы данных

3.8.1. Индивидуальный откат транзакций

Возможен только для не закончившихся транзакций

Все записи в журнале от данной транзакции связываются в обратный список.

Технология восстановления:

1. Выбирается очередная запись из списка данной транзакции.

2. Выполняется противоположная по смыслу операция:

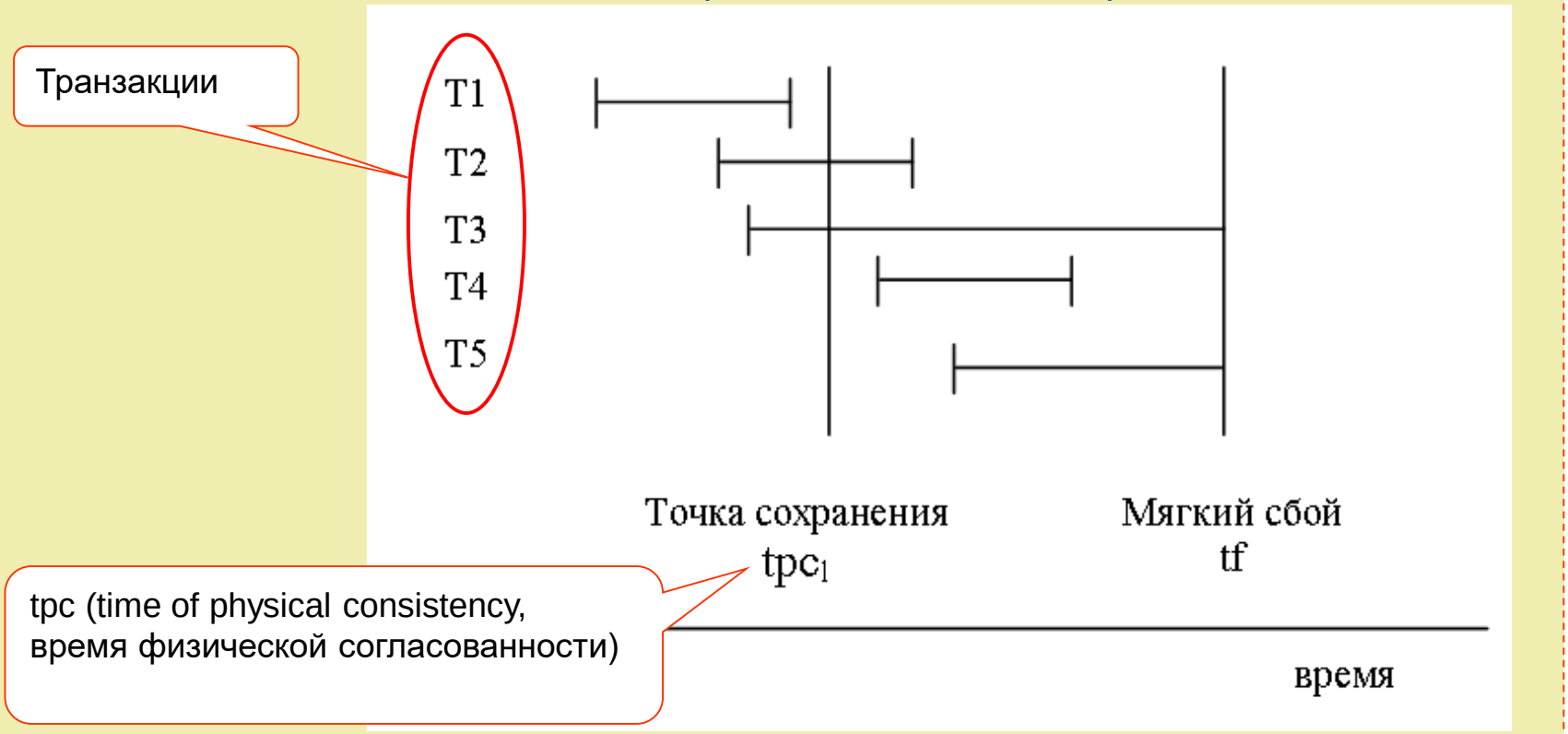
- вместо операции INSERT выполняется соответствующая операция DELETE,
- вместо операции DELETE выполняется INSERT,
- вместо прямой операции UPDATE обратная операция UPDATE, восстанавливающая предыдущее состояние объекта базы данных.

3.8.2. Восстановление после мягкого сбоя

Цель восстановления

Установить состояние внешней памяти основной части базы данных, которое возникло бы при фиксации во внешней памяти изменений всех незавершившихся транзакций и которое не содержало бы никаких следов незаконченных транзакций.

Возможные варианты состояния транзакций



3.8.3. Восстановление после жесткого сбоя

Основа восстановления - журнал транзакций и архивная копия базы данных

Технология восстановления:

1. Обратное копирование базы данных из архивной копии.
2. Для всех закончившихся транзакций выполняется redo, т.е. операции повторно выполняются в прямом смысле. Т.е. по журналу транзакций в прямом направлении выполняются все операции
3. Для транзакций, которые не закончились к моменту сбоя, выполняется откат.