Управление транзакциями

Управление базами данных

Основные понятия

Транзакция – это неделимая последовательность действий, переводящая базу данных из одного непротиворечивого состояния в другое непротиворечивое состояние

Необходимость в достоверности и согласованности

- При наличии отказов оборудования и программ
- При эксплуатации БД в многопользовательской среде

От СУБД требуется

- Поддержка транзакций
- Поддержка параллельной обработки данных
- Восстановление БД

Основные понятия

Свойства транзакции:

- Атомарность (Atomicity)
- Согласованность (Consistency)
- Изолированность (Isolation)
- Устойчивость (Durability)

Основные понятия

```
SELECT * FROM Exemplars e JOIN Readers r ON e.reader_id = r.reader_id

BEGIN TRAN

UPDATE Exemplars SET reader_id = 2
WHERE inv = 2

SELECT * FROM Exemplars e JOIN Readers r ON e.reader_id = r.reader_id

ROLLBACK TRAN     /*COMMIT TRAN*/

SELECT * FROM Exemplars e JOIN Readers r ON e.reader_id = r.reader_id
```

	inv	isbn	reader_id	date_out	reader_id	last_name	first_name	second_name	work_phone	home_phone
1	2	123	1	NULL	1	Петров	Петр	Петрович	123123	321321

	inv	isbn	reader_id	date_out	reader_id	last_name	first_name	second_name	work_phone	home_phone
1	2	123	2	NULL	2	Иванов	Иван	Иванович	321123	123321

	inv	isbn	reader_id	date_out	reader_id	last_name	first_name	second_name	work_phone	home_phone
1	2	123	1	NULL	1	Петров	Петр	Петрович	123123	321321

Управление параллельностью

Управление параллельностью выполнения транзакций – это процесс организации выполнения различных операций с БД, гарантирующий исключение взаимного влияния этих операций

Управление параллельностью

Проблемы параллельной обработки данных:

- Проблема потерянного обновления. Lost Updates
- Зависимость от нефиксированных результатов (Проблема «грязного» чтения). Uncommitted Dependency (Dirty Read)
- Проблема несогласованной обработки (Проблема неповторяемого чтения). Inconsistent Analysis (Nonrepeatable Read)
- Проблема чтения фантомов. Phantom Reads

Управление параллельностью

Проблема потерянного обновления. Lost Updates

bal=100
T1 – снять 10
T2 – добавить 100
Ожидаемый результат: 100+100-10=190

t	bal	T1	T2	Комментарий
1	100		Begin	
2	100	Begin	Read(bal)	
3	100	Read(bal)	bal = bal + 100	
4	200	bal = bal -10	Write(bal)	
5	90	Write(bal)	Commit	Потеря обновления
6	90	Commit		

Решение проблемы: T1 не должна читать данные до того, как они будут зафиксированы

Управление параллельностью

Зависимость от нефиксированных результатов (Проблема «грязного» чтения)

bal=100
T1 – снять 10
T2 – попытка добавить 100, но без фиксации результата
Ожидаемый результат: 100-10=90

t	bal	T1	T2	Комментарий
1	100		Begin	
2	100		Read(bal)	
3	100		bal = bal + 100	
4	200	Begin	Write(bal)	
5	200	Read(bal)	Проверка правильности	T2 не должна быть выполнена
6	100	bal = bal -10	Rollback	Откат T2
7	190	Write(bal)		
8	190	Commit		

Решение проблемы: T1 не должна читать данные до того, как они будут зафиксированы

Управление параллельностью

Проблема несогласованной обработки (Проблема неповторяемого чтения)

bal1=100, bal2=50, bal3=25
T1 – снять 10 с bal1 и поместить в bal3
T2 – Вычислить сумму bal1, bal2, bal3
Ожидаемый результат: 175

База данных – в непротиворечивом состоянии, но sum - некорректная

t	bal1	bal2	bal3	T1	T2	sum	Комментарий
1	100	50	25		Begin		
2	100	50	25	Begin	sum = 0	0	
3	100	50	25	Read(bal1)	Read(bal1)	0	Чтение данных, которые будут изменены
4	100	50	25	bal1 = bal1 – 10	sum = sum + bal1	100	
5	90	50	25	Write(bal1)	Read(bal2)	100	
6	90	50	25	Read(bal3)	sum = sum + bal2	150	
7	90	50	25	bal3 = bal3 + 10		150	
8	90	50	35	Write(bal3)		150	
9	90	50	35	Commit	Read(bal3)	150	
10	90	50	35		sum = sum + bal3	185	

Управление параллельностью

Проблема чтения фантомов

T1 – удалить строку из таблицы
T2 – найти сумму по столбцу P

t	Строки таблицы	T1	T2	sum	Комментарий
1	1 | 100 2 | 50		Begin		
2	1 | 100 2 | 50	Begin	Select P where ID = 1, a1 = P		
3	2 | 50	Delete where ID = 1	Select P where ID = 2, a2 = P		Прочитанная строка удалена
4	2 | 50	Commit	sum = a1+a2	150	В сумме – значение из несуществующей строки
5	2 | 50		Commit		

Решение проблемы: T1 не должна удалять данные до того, как завершится T2

Управление параллельностью

Уровни изоляции транзакций в MS SQL Server:

- READ UNCOMMITED - незавершенное чтение. Гарантируется только физическая целостность БД. Изоляция отсутствует
- READ COMMITED – Решается проблема «грязного чтения», но остается неповторяемость чтения. Это уровень изоляции по умолчанию
- REPEATABLE READ – повторяемое чтение. Решается проблема неповторяемого чтения, но проблема фантомов остается. Снижается степень параллелизма
- SERIALIZABLE – сериализуемость. Полная изоляция транзакций, параллельность выполнения транзакций почти исключается

Пример:

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITED
```

Методы управления параллельностью

Подходы:

- *Пессимистический подход* – выполнение транзакции откладывается, если возможен конфликт между транзакциями
  - **Метод выявления взаимных блокировок**
  - **Метод временных отметок**
- *Оптимистический подход* – все транзакции выполняются асинхронно и после завершения транзакции принимается решение о корректности ее выполнения

Блокировка (lock) – это временно накладываемое ограничение на выполнение операции над ресурсом (ресурс – строка, столбец, таблица, БД)

Методы управления параллельностью

Основные типы блокировок:

- **S-блокировка** (Shared Lock) - разделяемая (нежесткая) блокировка. Запрещает изменение объекта, но чтение объекта разрешается любой транзакции. Снять блокировку может только та транзакция, которая ее установила.
- **X-блокировка** (E**x**clusive Lock) – монопольная (жесткая, эксклюзивная) блокировка. Позволяет изменять объект только той транзакции, которая наложила эту блокировку. Другие транзакции не могут ни читать, ни изменять объект.

Двухфазный протокол блокировки

Выполнение транзакции имеет две фазы:

- **Фаза нарастания** (расширения): транзакция устанавливает блокировки на необходимые ресурсы. Ресурсы не освобождаются, даже если они не будут использованы транзакцией
- **Фаза сжатия**: транзакция только снимает блокировки с ресурсов

Двухфазный протокол блокировки

Решение проблемы потерянного обновления

t	bal	T1	T2
1	100		Begin
2	100	Begin	X_Lock(bal)
3	100	X_Lock(bal)	Read(bal)
4	100	Wait	bal = bal + 100
5	200	Wait	Write(bal)
6	200	Wait	Commit Unlock(bal)
7	200	Read(bal)	
8	200	bal = bal -10	
9	190	Write(bal)	
10	190	Commit Unlock(bal)	

Двухфазный протокол блокировки

Решение проблемы зависимости от нефиксированных результатов

t	bal	T1	T2
1	100		Begin
2	100	Begin	X_Lock(bal)
3	100	X_Lock(bal)	Read(bal)
4	200	Wait	bal = bal + 100
5	200	Wait	Write(bal)
6	200	Wait	Проверка правильности
7	100	Wait	Rollback Unlock(bal)
8	100	Read(bal)	
9	100	bal = bal -10	
10	90	Write(bal)	
11	90	Commit Unlock(bal)	

Двухфазный протокол блокировки

Решение проблемы несогласованной обработки данных

t	bal1	bal2	bal3	T1	T2	sum
1	100	50	25		Begin	
2	100	50	25	Begin	sum = 0	0
3	100	50	25	X_Lock(bal1)		0
4	100	50	25	Read(bal1)	S_Lock(bal1)	0
5	100	50	25	bal1 = bal1 – 10	Wait	0
6	90	50	25	Write(bal1)	Wait	0
7	90	50	25	X_Lock(bal3)	Wait	0
8	90	50	25	Read(bal3)	Wait	0
9	90	50	25	bal3 = bal3 + 10	Wait	0
10	90	50	35	Write(bal3)	Wait	0
11	90	50	35	Commit Unlock(bal1, bal3)	Wait	0
12	90	50	35		Read(bal1)	0
13	90	50	35		sum = sum + bal1	90
14	90	50	35		S_Lock(bal2)	90
15	90	50	35		Read(bal2)	90
16	90	50	35		sum = sum + bal2	140
17	90	50	35		S_Lock(bal3)	140
18	90	50	35		Read(bal3)	140

Двухфазный протокол блокировки

Решение проблемы чтения фантомов

t	Строки таблицы	T1	T2	sum
1	1 | 100 2 | 50		Begin	
2	1 | 100 2 | 50	Begin	S_Lock(таблица)	
3	1 | 100 2 | 50	X_Lock(таблица)	Select P where ID = 1, a1 = P	
4	1 | 100 2 | 50	Wait	Select P where ID = 2, a2 = P	
5	1 | 100 2 | 50	Wait	sum = a1+a2	150
6	1 | 100 2 | 50	Wait	Commit Unlock(таблица)	
7	2 | 50	Delete where ID = 1		
8	2 | 50	Commit Unlock(Таблица)		

Взаимные блокировки

Взаимная блокировка (тупиковая блокировка, «мертвая» блокировка, deadlock) имеет место тогда, когда две или более транзакции находятся в бесконечном ожидании освобождения ресурса, занимаемого каждой из них

T1: bal1=bal1-10, потом bal2=bal2-10
T2: bal2=bal2+100, потом bal1=bal1+100

t	bal1	bal2	T1	T2	Комментарий
1	100	100	Begin		
2	100	100	X_Lock(bal1)	Begin	
3	100	100	Read(bal1)	X_Lock(bal2)	
4	100	100	bal1=bal1-10	Read(bal2)	
5	90	100	Write(bal1)	bal2=bal2+100	
6	90	200	X_Lock(bal2)	Write(bal2)	
7	90	200	Wait	X_Lock(bal1)	T1 ожидает освобождения bal2
8	90	200	Wait	Wait	T2 ожидает освобождения bal1
9	90	200	Wait	Wait	

Взаимные блокировки

Двухфазный протокол блокировки не решает проблемы взаимной блокировки

Взаимные блокировки

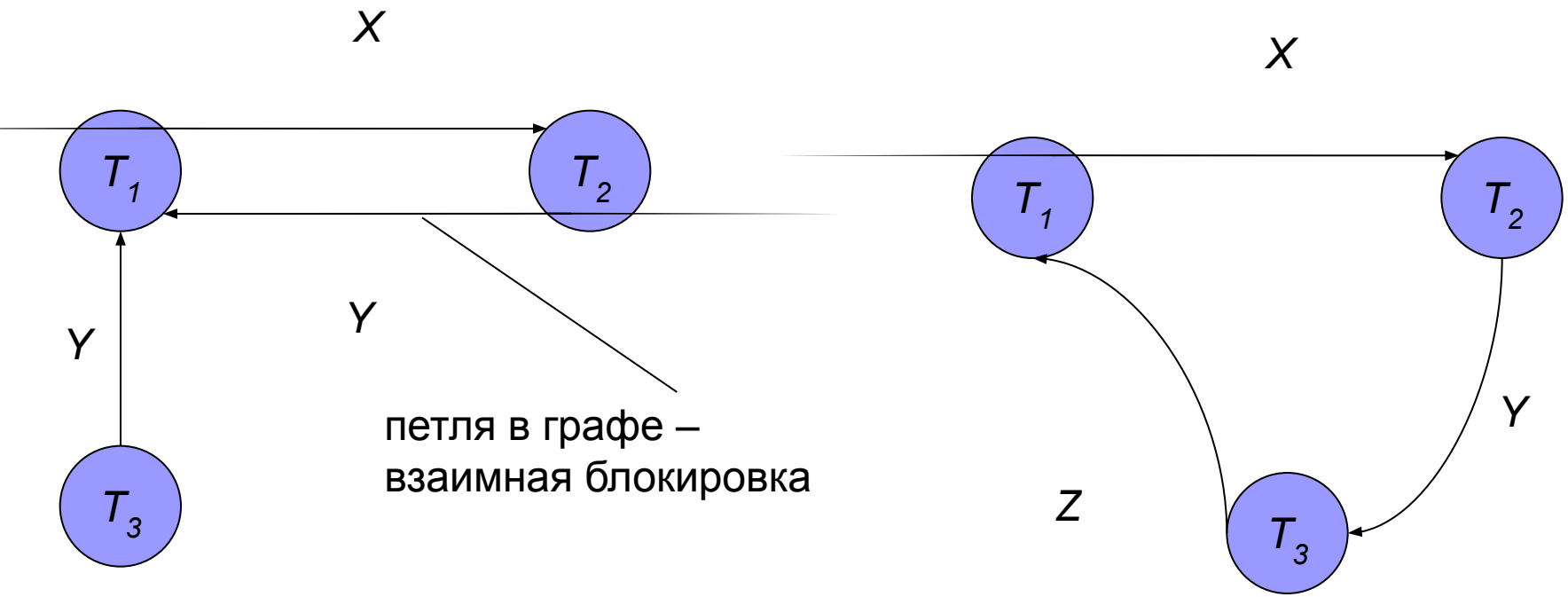
```
ALTER PROCEDURE [dbo].[P1]
AS
BEGIN
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
BEGIN TRAN

DECLARE @i INT
DECLARE @id INT
SET @i=1

DELETE FROM T1

WHILE @i < 10000
BEGIN
        INSERT INTO T1 (data) VALUES(@i)
        SET @i = @i+1
END

WHILE (SELECT COUNT(*) FROM T2) > 0
BEGIN
        SET @id = (SELECT TOP 1 id FROM T2);
        DELETE T2 WHERE id = @id
END

COMMIT TRAN
END
```

```
ALTER PROCEDURE [dbo].[P2]
AS
BEGIN
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
BEGIN TRAN

DECLARE @i INT
DECLARE @id INT
SET @i=1

DELETE FROM T2

WHILE @i < 1000
BEGIN
        INSERT INTO T2 (data) VALUES(@i)
        SET @i = @i+1
END

WHILE (SELECT COUNT(*) FROM T1) > 0
BEGIN
        SET @id = (SELECT TOP 1 id FROM T1);
        DELETE T1 WHERE id = @id
END

COMMIT TRAN
END
```

Msg 1205, Level 13, State 56, Procedure P2, Line 27
Transaction (Process ID 55) was deadlocked on lock resources with another process and has been chosen as the deadlock victim. Rerun the transaction.

Взаимные блокировки

$T_i \xrightarrow{X} T_j$ транзакция T_i ожидает освобождения блокировки ресурса X, выставленной транзакцией T_j



Метод выявления взаимных блокировок

Выявление взаимных блокировок:

- СУБД строит граф ожиданий через определенные промежутки времени
- Если в графе ожидания имеются петли, то делается вывод о наличии взаимных блокировок. В петле выбирается транзакция жертва (по некоторому критерию стоимости) и происходит откат транзакции-жертвы. Петля разрывается и остальные транзакции продолжают выполнение.

Недостаток: построение графа ожиданий загружает СУБД

Метод предупреждения взаимных блокировок

Основная идея **метода предупреждения взаимных блокировок**: устанавливается порядок выполнения транзакций на основе временных отметок ресурсов. Если попытка транзакции использовать ресурс нарушает заданный порядок, то она откатывается и перезапускается снова. В результате взаимные блокировки не могут возникнуть.

Недостатки:

- Более частые откаты транзакций, чем в методе выявления взаимных блокировок
- В распределенных системах сложно генерировать временные метки с отношением полного порядка

Метод предупреждения взаимных блокировок

Алгоритм 1. «Ожидание-отмена»:

Допускается только ожидание старой транзакцией. Если более новая транзакция вынуждена ожидать освобождения блокировки, то она откатывается и запускается заново. В результате наиболее старая транзакция проходит без отката, а более новая транзакция со временем становится наиболее старой.

Алгоритм 2. «Отмена-ожидание»:

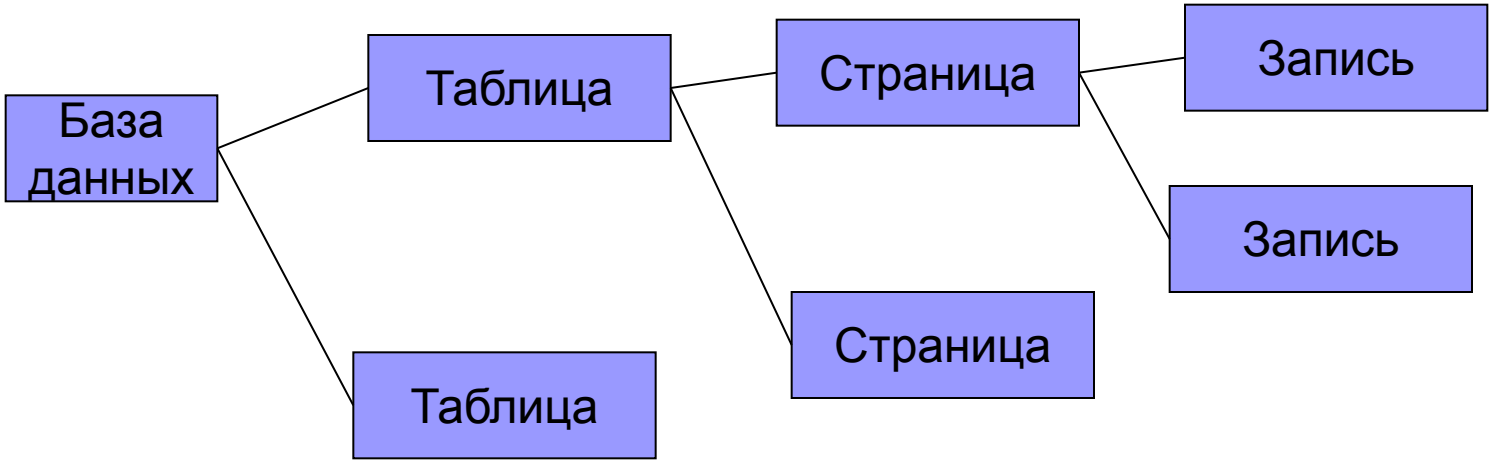
Только более новые транзакции могут ожидать завершения более старой транзакции. Если более старая транзакция вынуждена ожидать освобождения блокировки, то она откатывается.

Гранулированные захваты (уровни блокировок)

Уровни блокировок (выборочно):

- RID – блокировка отдельной строки (используется ID строки)
- Page – блокировка на уровне страницы
- Table – блокировка таблицы
- DB – блокировка базы данных

Чем крупнее элементы блокируемых данных, тем ниже уровень параллельности выполнения транзакций.

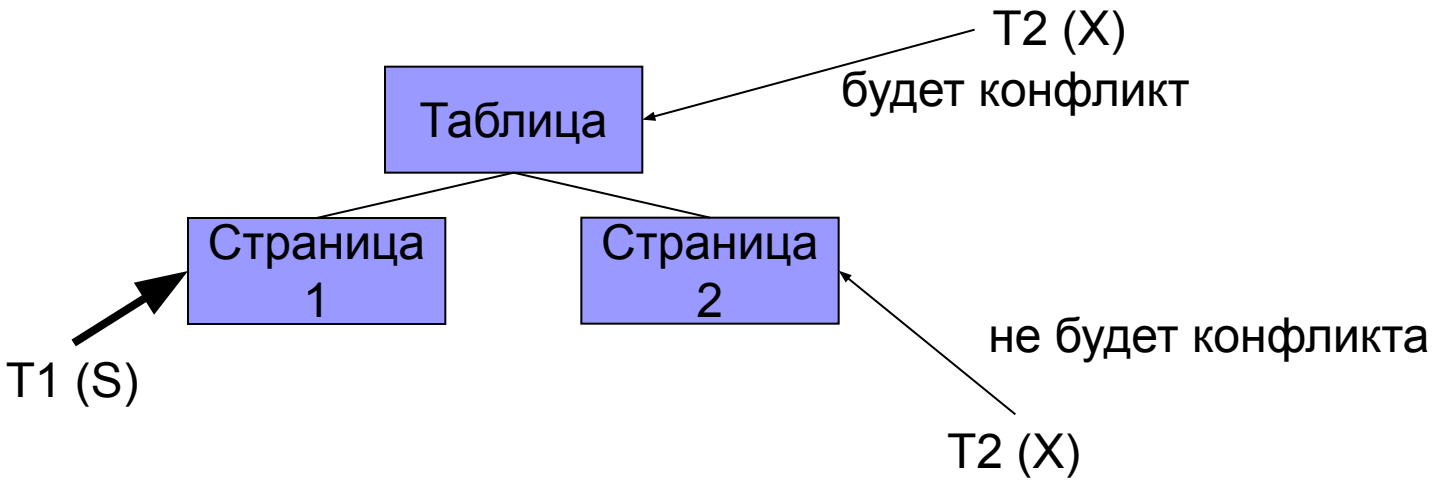


Если устанавливается блокировка страницы, то должны блокироваться все записи на странице (вниз по иерархии). При этом ~~другим~~ должны быть ограничены действия других транзакций с таблицей, для этого предназначены блокировки намерения (вверх по иерархии).

Гранулированные захваты (уровни блокировок)

Блокировки намерения: устанавливаются вверх по иерархии от блокируемого элемента для того, чтобы запретить другим транзакциям накладывать конфликтующие блокировки на верхнем уровне. Поэтому при выполнении другой транзакции оказывается достаточным проверить блокировку намерения на верхнем уровне без проверки блокировок вниз по иерархии

Назначение: повышение производительности параллельной обработки данных и уменьшение вероятности возникновения взаимных блокировок.



Гранулированные захваты (уровни блокировок)

S – разделяемая блокировка. Автоматически распространяет вниз по иерархии. Устанавливается для чтения данных

X – монопольная блокировка. Автоматически распространяется вниз по иерархии. Устанавливается для изменения данных

IS – блокировка намерения для разделяемой блокировки. Автоматически распространяется вверх по иерархии.

IX – блокировка намерения для монопольной блокировки. Автоматически распространяется вверх по иерархии

SIX – разделяемая блокировка намерения для монопольной блокировки. Устанавливается явным образом на верхнем уровне иерархии при намерении читать все данные вниз по иерархии и изменять часть данных внизу иерархии. В один момент времени на ресурс может быть наложена только одна блокировка SIX.

Пример: на строку установлена X, на страницу таблицы установлена IX, а на таблицу установлена SIX. Другие транзакции могут установить на таблицу IS для чтения тех страниц, для которых не установлена IX, но установить IX или X они не могут.

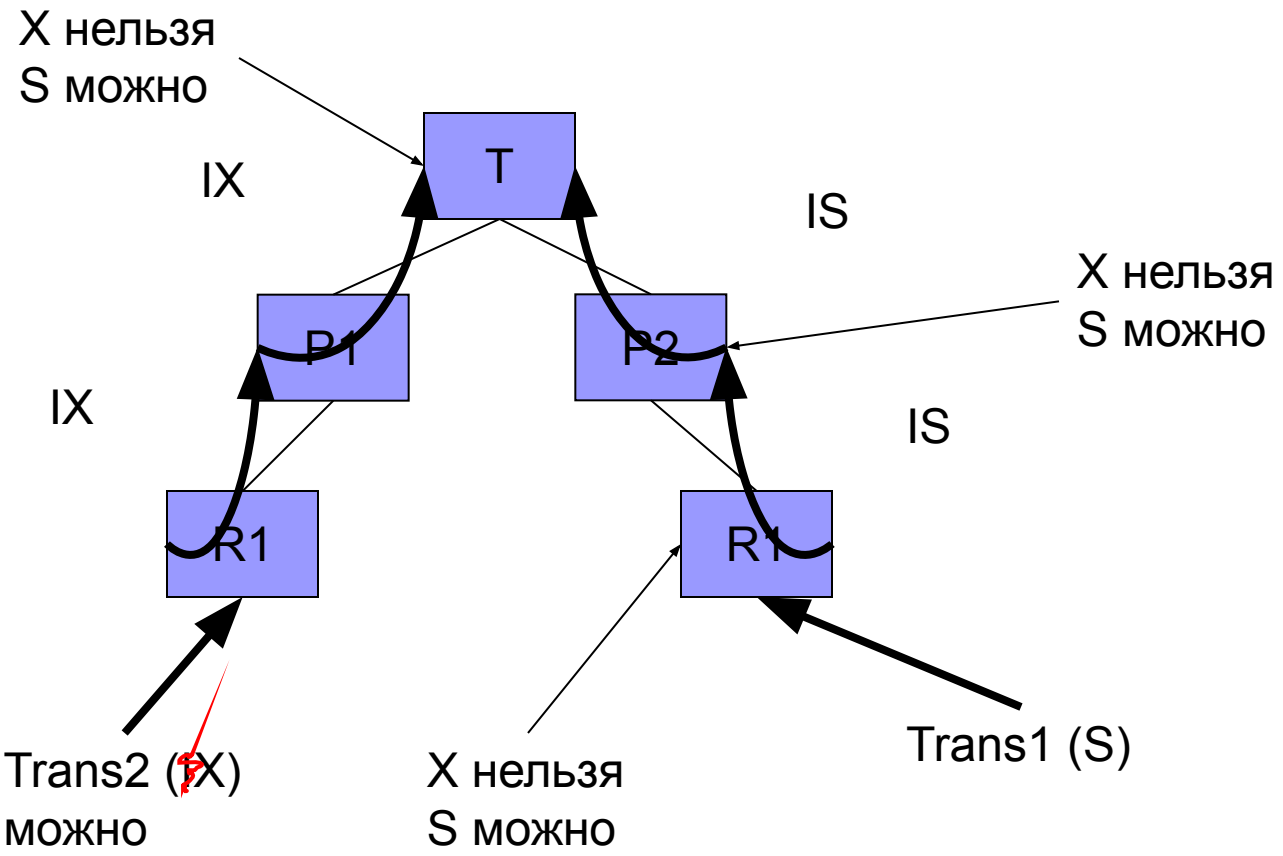
Гранулированные захваты (уровни блокировок)

Таблица совместимости блокировок

Запрашиваемая блокировка	Наложенная блокировка					
		IS	IX	S	SIX	X
	IS	+	+	+	+	-
	IX	+	+	-	-	-
	S	+	-	+	-	-
	SIX	+	-	-	-	-
	X	-	-	-	-	-

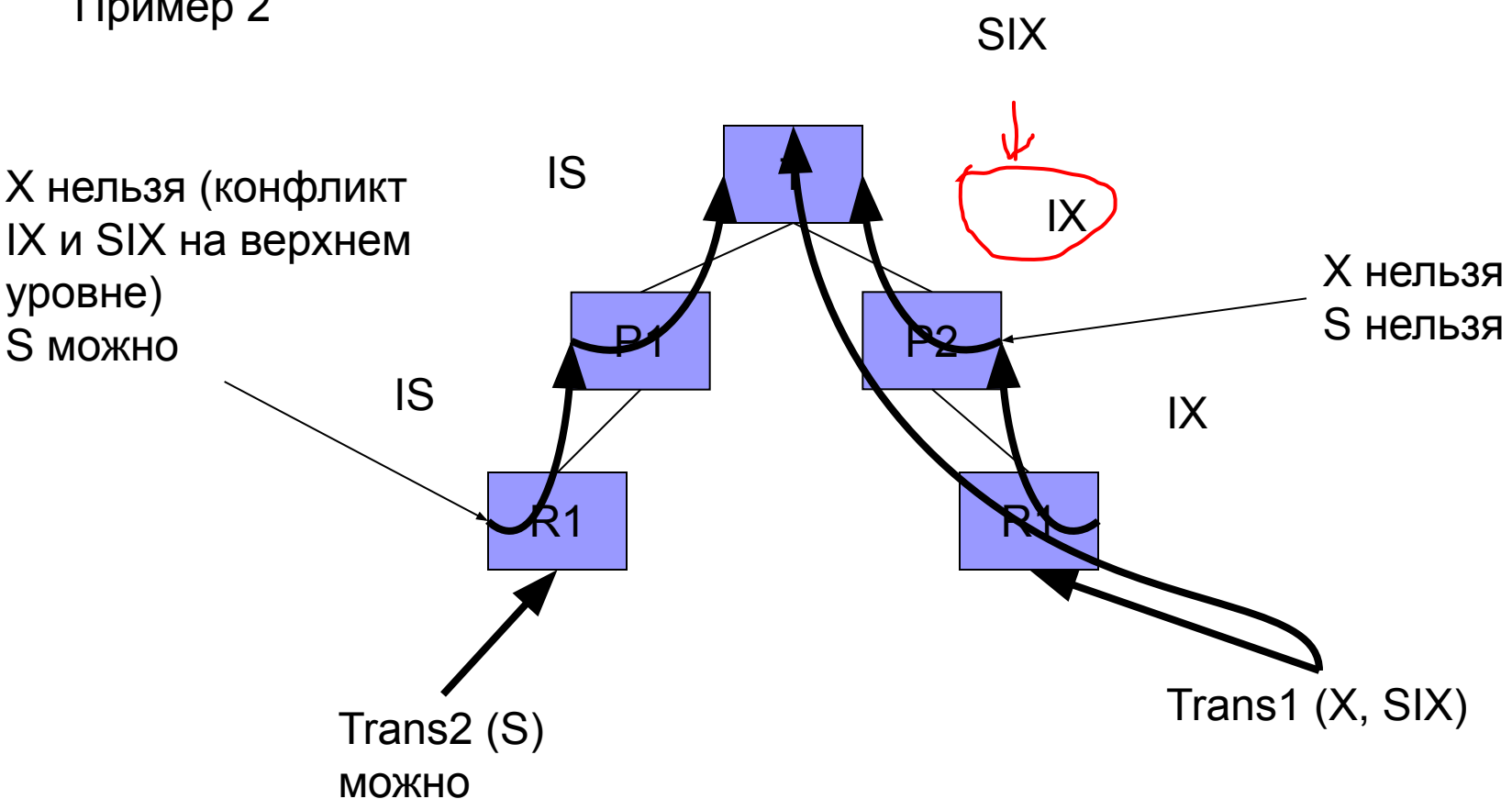
Гранулированные захваты (уровни блокировок)

Пример 1



Гранулированные захваты (уровни блокировок)

Пример 2



Управление транзакциями

- Явные транзакции:

BEGIN TRAN, COMMIT TRAN, ROLLBACK TRAN

- Автоматические (по умолчанию): каждая команда – отдельная транзакция

SET IMPLICIT_TRANSACTION OFF

- Неявные транзакции: до ROLLBACK TRAN или COMMIT TRAN. Началом транзакции является одна из команд: ALTER TABLE, CREATE, DELETE, DROP, INSERT, SELECT и некоторые другие

SET IMPLICIT_TRANSACTION ON

Вложенные транзакции

Метод временных отметок

Оптимистический подход управления параллельным выполнение транзакций

Восстановление баз данных

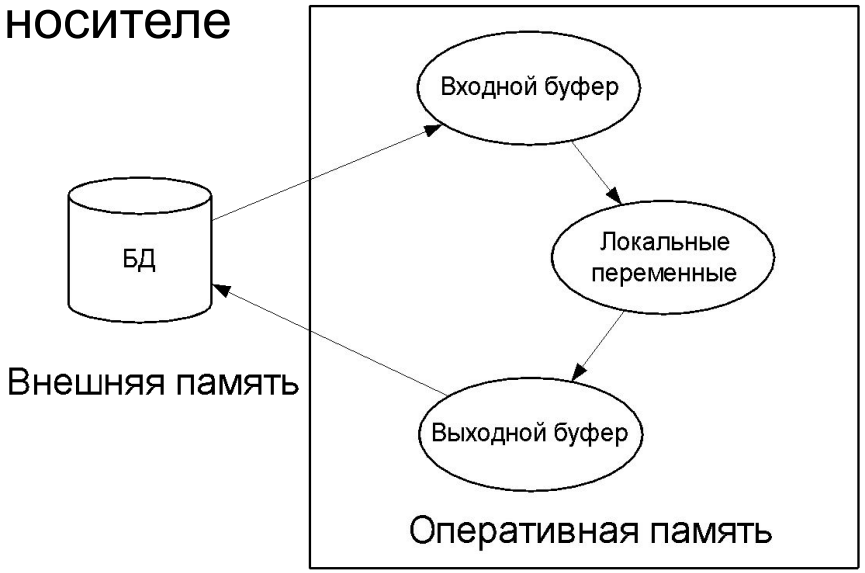
Восстановление базы данных – это процесс возвращения базы данных в корректное состояние, утраченное в результате отказа

Причины отказов:

- Аварийное прекращение работы системы (сбой аппаратуры)
- Ошибки прикладных программ
- Стихийные бедствия, небрежность пользователя, диверсия

Следствия отказов:

- Утрата содержимого оперативной памяти
- Утрата базы данных на внешнем носителе



Восстановление баз данных

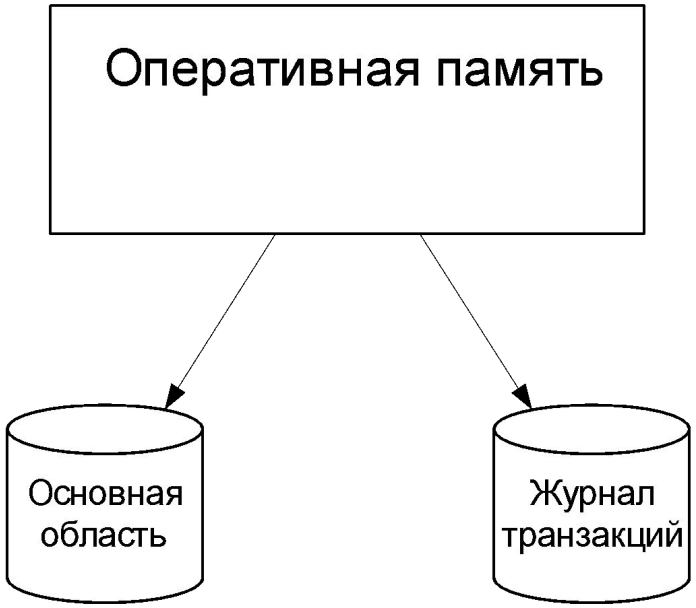
Задачи восстановления БД:

- Резервное копирование
- Ведение журнала транзакций и изменений БД
- Создание контрольных точек (перенос изменений данных во вторичную память)

Резервное копирование: сохранение базы данных целиком, либо сведений об изменениях (инкрементный режим) на автономных носителях большого объема

Транзакции и восстановление

Основная идея



1) Журнал используется для определения состояния транзакций на момент сбоя
2) Журнал содержит подтверждение фиксации данных.
3) До любой попытки изменить данные, в журнал записывается это намерение, и только после этого данные изменяются.
Поэтому на момент сбоя можно выявить следующие ситуации, имевшие место на момент сбоя:

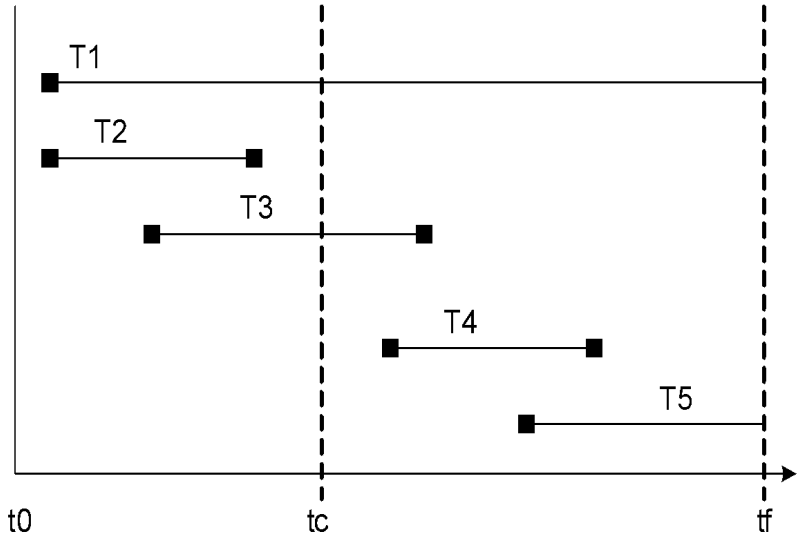
- изменений нет
- есть запись о намерении, но изменения не выполнены
- выполнено все

Журнал транзакций

Журнал содержит следующие записи о транзакциях:

- Идентификатор транзакции
- Тип записи (начало транзакции, завершение транзакции, обновление данных, удаление данных, вставка данных)
- Идентификатор элемента данных
- Копия элемента данных ***ДО*** операции
- Копия элемента данных ***ПОСЛЕ*** операции - может отсутствовать

Журнал содержит записи о контрольных точках. Контрольная точка – момент синхронизации оперативной памяти с журналом транзакций и основной областью (если фиксация). Все буферы СУБД принудительно записываются во вторичную память)



tc – контрольная точка (3-4 в час)
tf – момент сбоя
T1 – не завершена. Откат.
T2 – завершена и зафиксирована.
T3 – записана в журнал, но не полностью в основную область. Прогнать (повторно выполнить)
T4 – ~~возможно не все данные попали в журнал. Откат~~ прогнать (ж. есть фикс)
T5 – не завершена. Откат

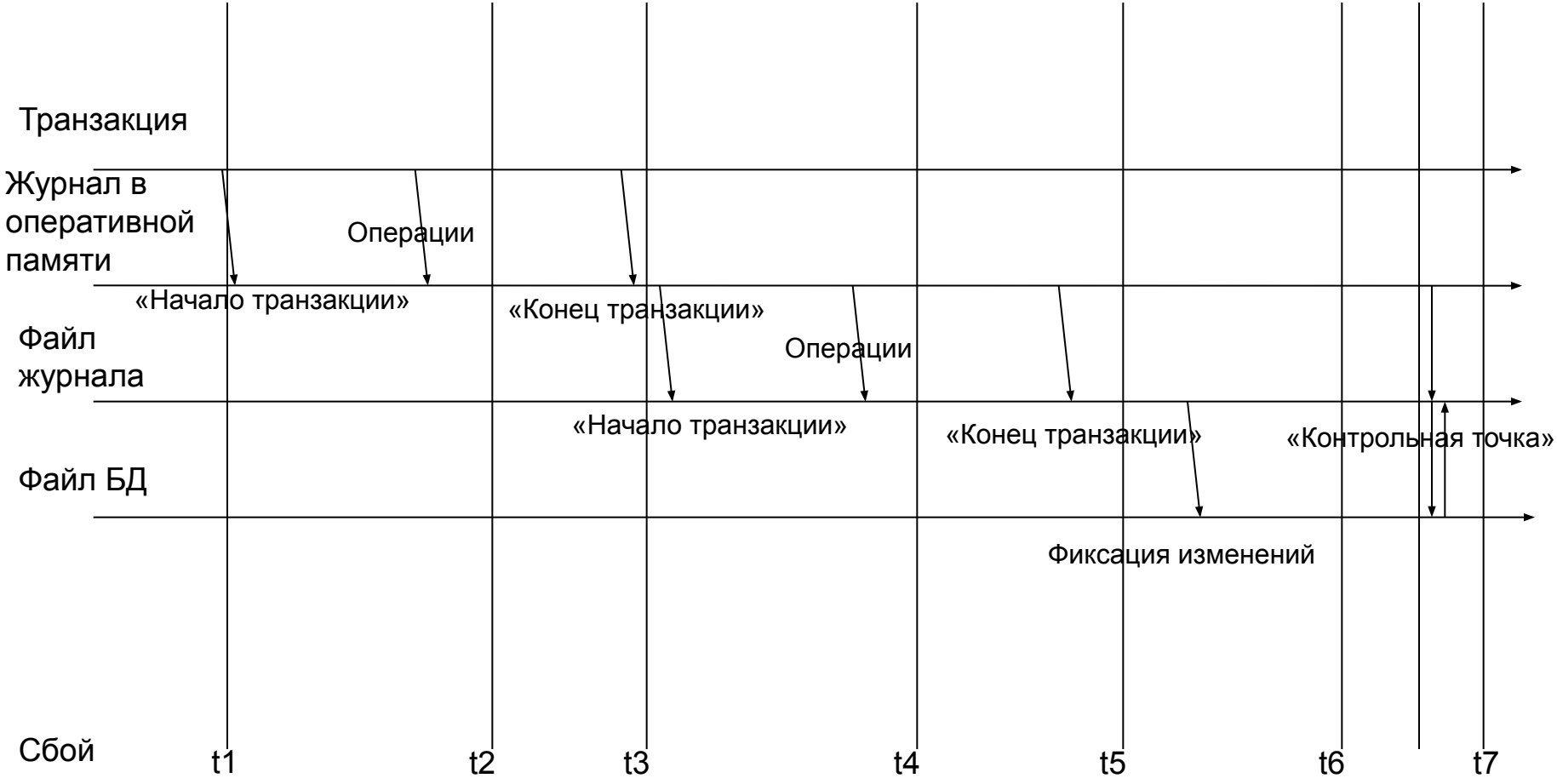
Метод восстановления с использованием отложенного обновления

При использовании метода восстановления с отложенным обновлением, нормальная работа СУБД выполняется следующим образом: информация об изменениях записывается сначала в журнал транзакций, обновления не заносятся в БД до тех пор, пока транзакция не выдаст команду фиксации результатов:

1) При запуске транзакции в журнал помещается запись «начало транзакции».
2) При выполнении любой операции записи, в журнал помещается информация об операции: тип операции, идентификатор изменяемого элемента, копия элемента **ПОСЛЕ** операции (копия элемента до операции не используется).
3) При завершении транзакции в журнал запись «транзакция завершена»
4) Если транзакция завершена с командой фиксации результатов, то данные из оперативной памяти переносятся на диск. Если транзакция завершается с откатом транзакции, то изменения на диск не переносятся.

Через определенные промежутки времени формируются контрольные точки.

Метод восстановления с использованием отложенного обновления



t1, t2, t3, t4 - требуется откат

t5, t6 - требуется прогон

t7 – результаты фиксации гарантировано на диске

Метод восстановления с использованием отложенного обновления

Последовательность действий при восстановлении после сбоя:

1) Журнал просматривается в обратном направлении до последней контрольной точки.
2) Если для транзакции имеются обе записи «начало» «конец», то выполняется прогон транзакции. Т.е. изменения из записей об операциях переносятся в БД без повторных вычислений.
3) Если есть запись «начало», но нет записи «конец», то требуется откат и повторное выполнение. Откат, в данном случае, не требует никаких действий, т.к. данные в БД еще не попадали.
4) Если есть запись «начало транзакции» и «откат транзакции», то никакие действия не выполняются.

Метод восстановления с использованием немедленного обновления

Все изменения немедленно заносятся в БД без ожидания завершения транзакции. Для законченных транзакций нужен прогон, для не закончившихся – откат.

В журнал записывается следующая информация:

1) При запуске транзакции – запись «начало транзакции»
2) При выполнении операции – информация об операции, включая копию данных **ДО** выполнения и **ПОСЛЕ** выполнения.
3) Как только информация об операции помещается в журнал транзакций, изменения заносятся в буферы оперативной памяти.
4) Изменения в файлах происходят при разгрузке буферов.
5) При завершении транзакции в журнал помещается запись «конец транзакции»

Важно: строка об изменении данных помещается в журнал до физического изменения данных (протокол предварительной записи журнала).

При прогоне транзакции используются значения «после операции»

При откате транзакции в БД записываются значения из записи «до операции»