

Целостность базы данных

Целостность базы данных

- Восстановление
- Параллельный доступ
- Защита от несанкционированных действий

Восстановление

- **Восстановление в системе баз данных** - восстановление самой базы данных, т.е. возвращение базы данных в определенное состояние, которое считается правильным, после некоторого сбоя, в результате которого текущее состояние становится неправильным или, по крайней мере, достаточно неопределенным

Правильное состояние

- *Непротиворечивость* (отсутствие нарушений каких-либо известных ограничений целостности).
- *Допустимость* (соответствие логике предметной области)

Транзакции

- *Транзакция* — это логическая единица работы; она начинается с выполнения операции `BEGIN TRANSACTION` и заканчивается операцией `COMMIT` (выполнение всех действий транзакции) или `ROLLBACK` (откат всех действий транзакции).

Что должно быть в СУБД для работы с транзакциями

- Возможен неявно заданный оператор ROLLBACK.
- Неразрывность операций.
- Выполнение программы как последовательности транзакций.

Транзакции

- Оператор **COMMIT** (Зафиксировать) сигнализирует **об успешном окончании транзакции**. Он сообщает диспетчеру транзакций, что логическая единица работы успешно завершена, база данных вновь находится (или будет находиться после выполнения этого оператора) в непротиворечивом состоянии, а все обновления, выполненные данной логической единицей работы, теперь могут быть *зафиксированы, т.е. внесены в базу данных*.
- Оператор **ROLLBACK** (Выполнить откат) сигнализирует о **неудачном окончании транзакции**. Он сообщает диспетчеру транзакций, что произошла какая-то ошибка, база данных находится в противоречивом состоянии и следует осуществить *откат всех проведенных при выполнении этой транзакции обновлений, т.е. Они должны быть отменены*

ACID свойства транзакций

- **Atomicity** неразрывность
- **Correctness** правильность
- **Isolation** изолированность
- **Durability** устойчивостью

ACID свойства транзакций

- **Неразрывность.** Транзакции неразрывны (выполняются по принципу "все или ни чего").
- **Правильность.** Транзакции преобразуют базу данных из одного правильного состояния в другое; при этом правильность не обязательно должна обеспечиваться на всех промежуточных этапах.
- **Изолированность.** Транзакции изолированы одна от другой. Таким образом, даже если будет запущено множество транзакций, работающих параллельно, результаты любых операций обновления, выполняемых отдельной транзакцией, будут скрыты от всех остальных транзакций до тех пор, пока эта транзакция не будет зафиксирована. Иначе говоря, для любых отдельных транзакций А и В справедливо следующее утверждение: транзакция А сможет получить результаты выполненных транзакцией В обновлений только после фиксации транзакции В, а транзакция В сможет получить результаты выполненных транзакцией А обновлений только после фиксации транзакции А.
- **Устойчивость.** После того как транзакция зафиксирована, выполненные ею обновления сохраняются в базе данных на постоянной основе, даже если в дальнейшем произойдет аварийный останов системы

Транзакции MySQL

- START TRANSACTION
[*transaction_characteristic* [, *transaction_characteristic*] ...]
- *transaction_characteristic*: { WITH CONSISTENT SNAPSHOT | READ WRITE | READ ONLY }
- BEGIN [WORK]
- COMMIT [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
- ROLLBACK [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
- SET autocommit = {0 | 1}
- SAVEPOINT *identifier*
- ROLLBACK [WORK] TO [SAVEPOINT] *identifier*
- RELEASE SAVEPOINT *identifier*

Отказ

- Аварийный останов системы, который прерывает выполнение всех транзакций, действующих в системе во время его возникновения, называется также **глобальным отказом** или отказом системы
- отказ, который влияет только на одну транзакцию, называется локальным отказом (например, арифметическое переполнение)

Отказ

- Критическим моментом в отказе системы является *потеря содержимого основной (оперативной) памяти (в частности, буферов базы данных). Поскольку точное состояние любой выполнявшейся в момент отказа системы транзакции остается неизвестным, такая транзакция не может быть успешно завершена. Поэтому при перезапуске системы любая такая транзакция будет отменена (т.е. будет выполнен ее откат).*
- Зафиксированная транзакция не может быть отменена. Если окажется, что зафиксированная транзакция была ошибочной, потребуется выполнить другую транзакцию, отменяющую действия, выполненные первой транзакцией. Такая транзакция называется **компенсирующей**.

Проблемы организации параллельной работы

- Проблема потерянного обновления
- Проблема зависимости от незафиксированных результатов
- Проблема анализа несовместимости(несовместный анализ)

Проблема потерянного обновления



транзакция А теряет обновление в момент времени t4

Time	T ₁	T ₂	bal _x
t ₁		begin_transaction	100
t ₂	begin_transaction	read(bal _x)	100
t ₃	read(bal _x)	bal _x = bal _x + 100	100
t ₄	bal _x = bal _x - 10	write(bal _x)	200
t ₅	write(bal _x)	commit	90
t ₆	commit		90

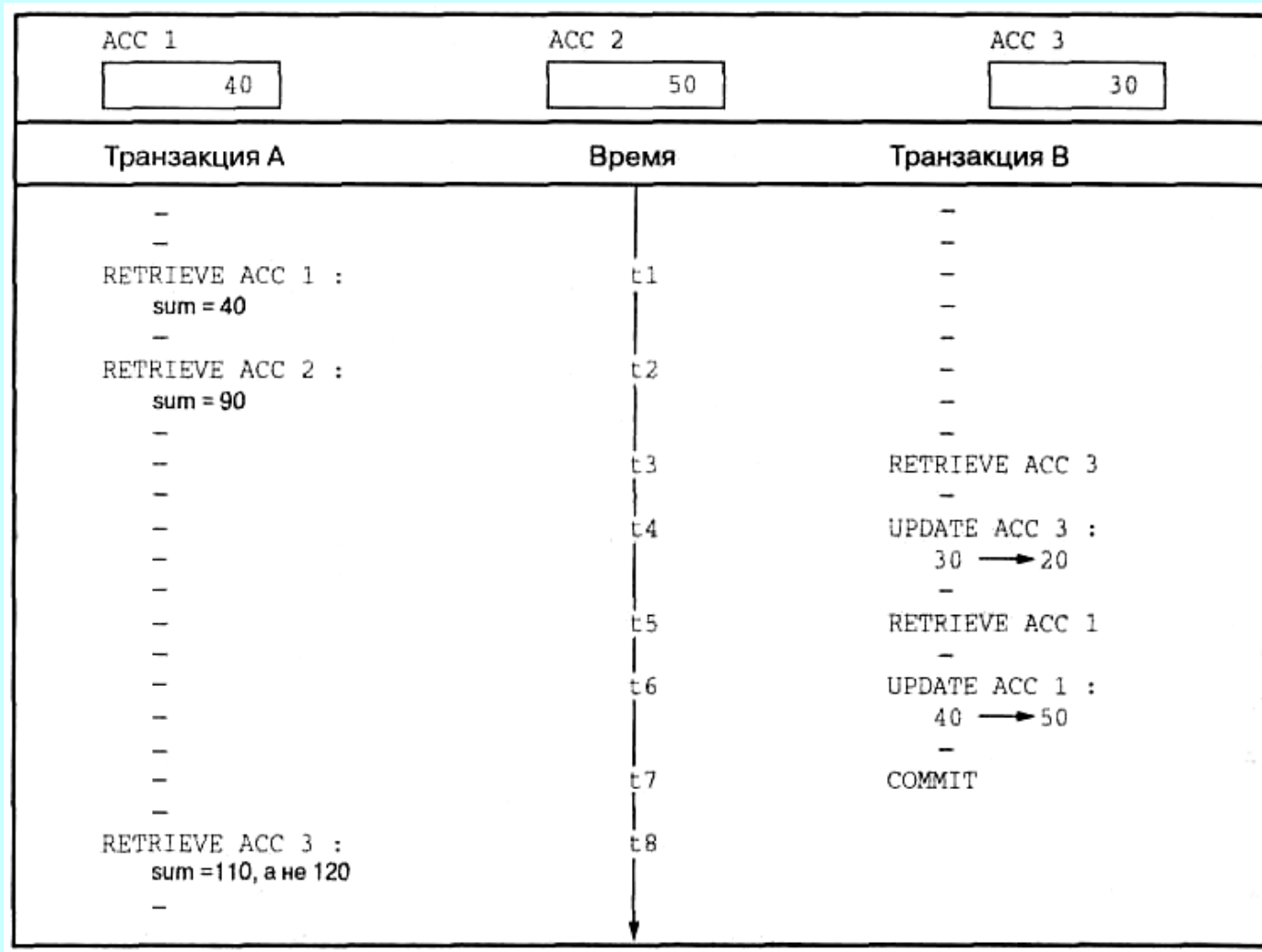
Проблема зависимости от незафиксированных результатов (грязное чтение)

Транзакция А	Время	Транзакция В
-		-
-		-
-	t1	UPDATE t
-		-
RETRIEVE t	t2	-
-		-
-	t3	ROLLBACK
-		

транзакция А становится зависимой от незафиксированного обновления в момент времени t2

Time	T ₃	T ₄	bal _x
t ₁		begin_transaction	100
t ₂		read(bal _x)	100
t ₃		bal _x = bal _x + 100	100
t ₄	begin_transaction	write(bal _x)	200
t ₅	read(bal _x)	:	200
t ₆	bal _x = bal _x - 10	rollback	100
t ₇	write(bal _x)		190
t ₈	commit		190

Проблема анализа несовместимости



транзакция А
выполняет
анализ
несовместимости

Проблема анализа несовместимости

Time	T ₅	T ₆	bal _x	bal _y	bal _z	sum
t ₁		begin_transaction	100	50	25	
t ₂	begin_transaction	sum = 0	100	50	25	0
t ₃	read(bal _x)	read(bal _x)	100	50	25	0
t ₄	bal _x = bal _x - 10	sum = sum + bal _x	100	50	25	100
t ₅	write(bal _x)	read(bal _y)	90	50	25	100
t ₆	read(bal _z)	sum = sum + bal _y	90	50	25	150
t ₇	bal _z = bal _z + 10		90	50	25	150
t ₈	write(bal _z)		90	50	35	150
t ₉	commit	read(bal _z)	90	50	35	150
t ₁₀		sum = sum + bal _z	90	50	35	185
t ₁₁		commit	90	50	35	185

Блокировки

- Блокировка -Процедура, используемая для управления параллельным доступом к данным. Когда некоторая транзакция получает доступ к базе данных, механизм блокировки позволяет (с целью предотвращения получения некорректных результатов) запретить попытки получения доступа к этим же данным со стороны других транзакций.
- разделяемые блокировки (блокировки S — shared) (блокировки чтения)
- исключительные блокировки (блокировки X — exclusive) (блокировки записи)

	X	S	-
X	Н	Н	Д
S	Н	Д	Д
-	Д	Д	Д

Матрица совместимости
для блокировок типов X и S

Взаимная блокировка

Транзакция А	Время	Транзакция В
-		-
-		-
LOCK r1 EXCLUSIVE	t1	-
-		-
-	t2	LOCK r2 EXCLUSIVE
-		-
LOCK r2 EXCLUSIVE	t3	-
ожидание		-
ожидание	t4	LOCK r1 EXCLUSIVE
ожидание		ожидание
ожидание		ожидание

Если произошла взаимоблокировка, желательно, чтобы система обнаружила ее и разорвала. Для обнаружения взаимоблокировки необходимо определить наличие цикла в **графе ожидания (Wait-For Graph)**. Так называется граф, который, неформально выражаясь, показывает "кто кого ожидает"

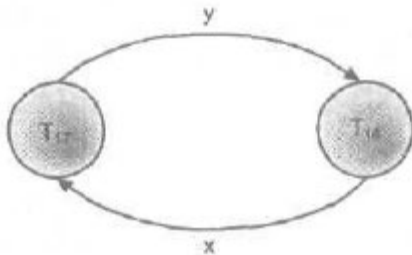
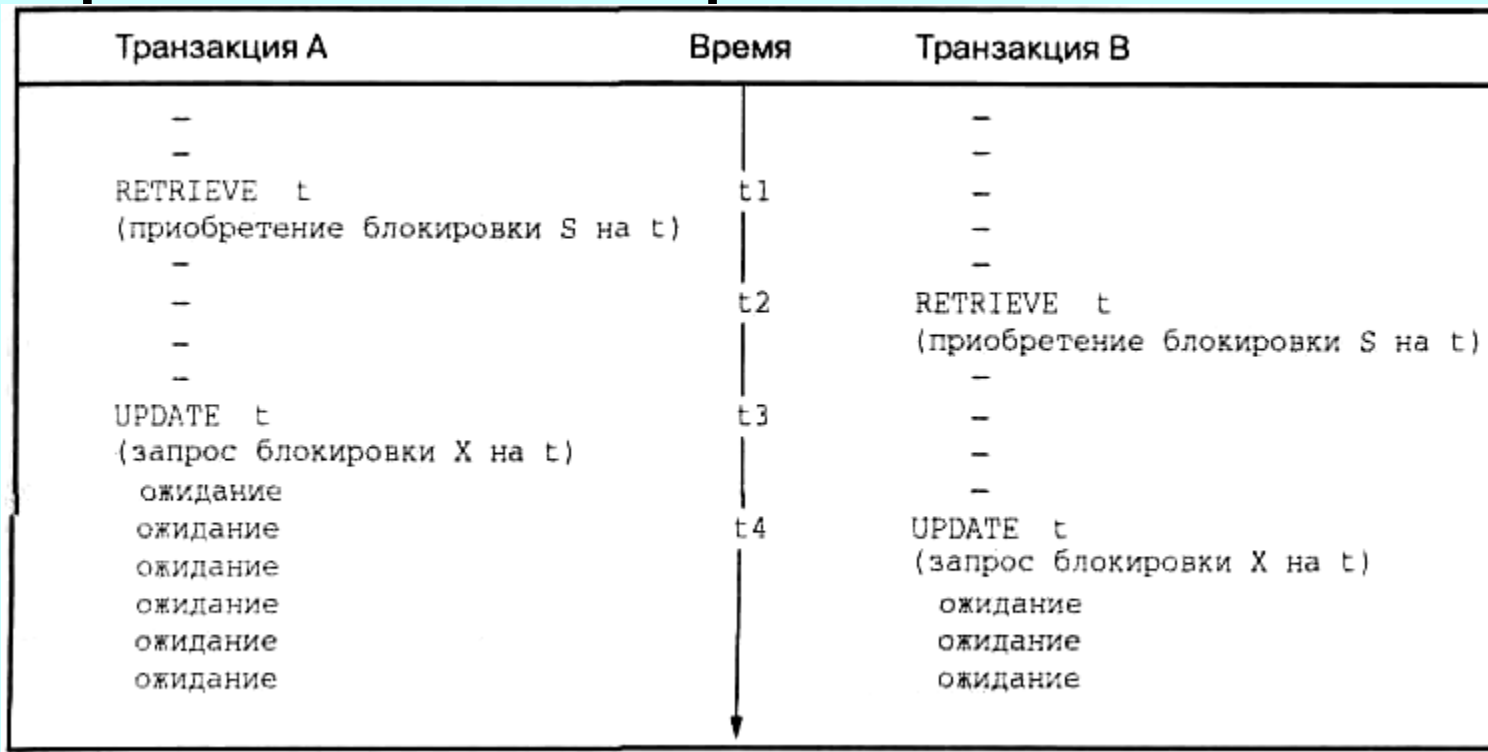


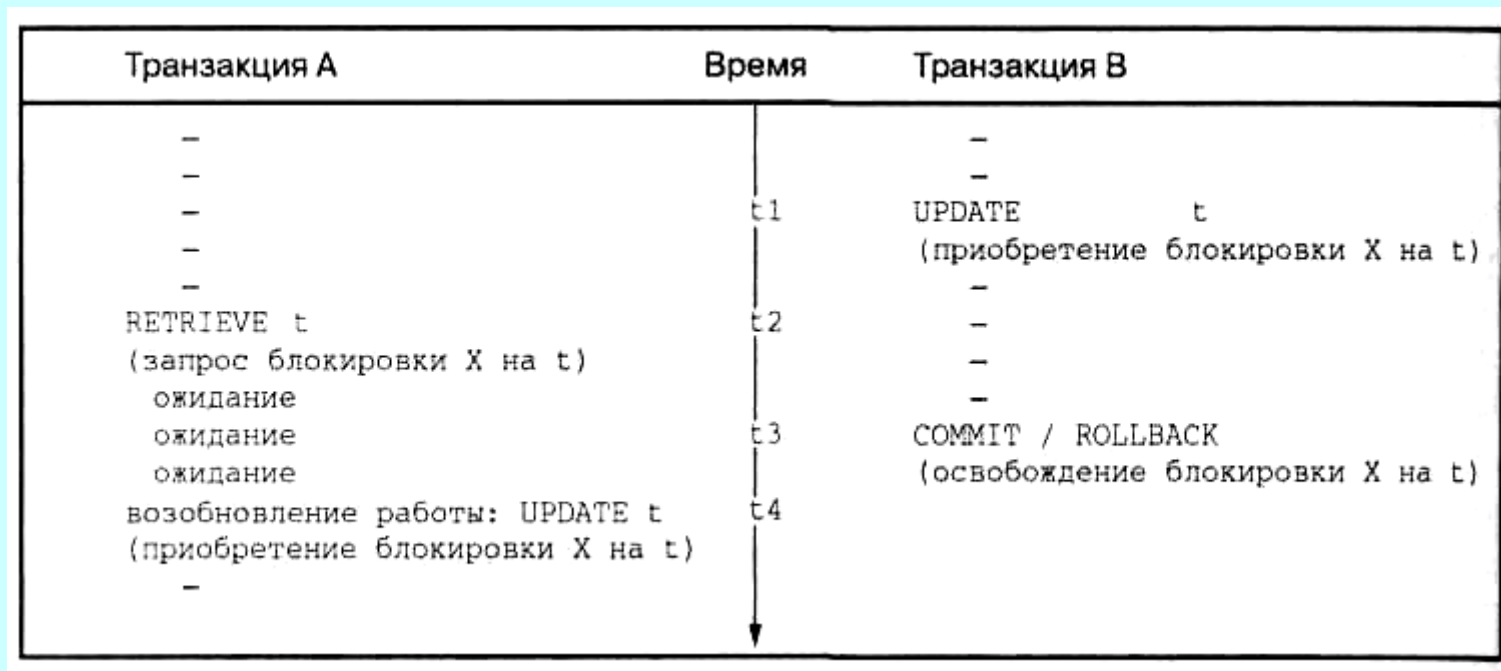
Рис. 19.4. Граф ожидания, который показывает наличие взаимоблокировки между двумя транзакциями

Проблема потерянного обновления



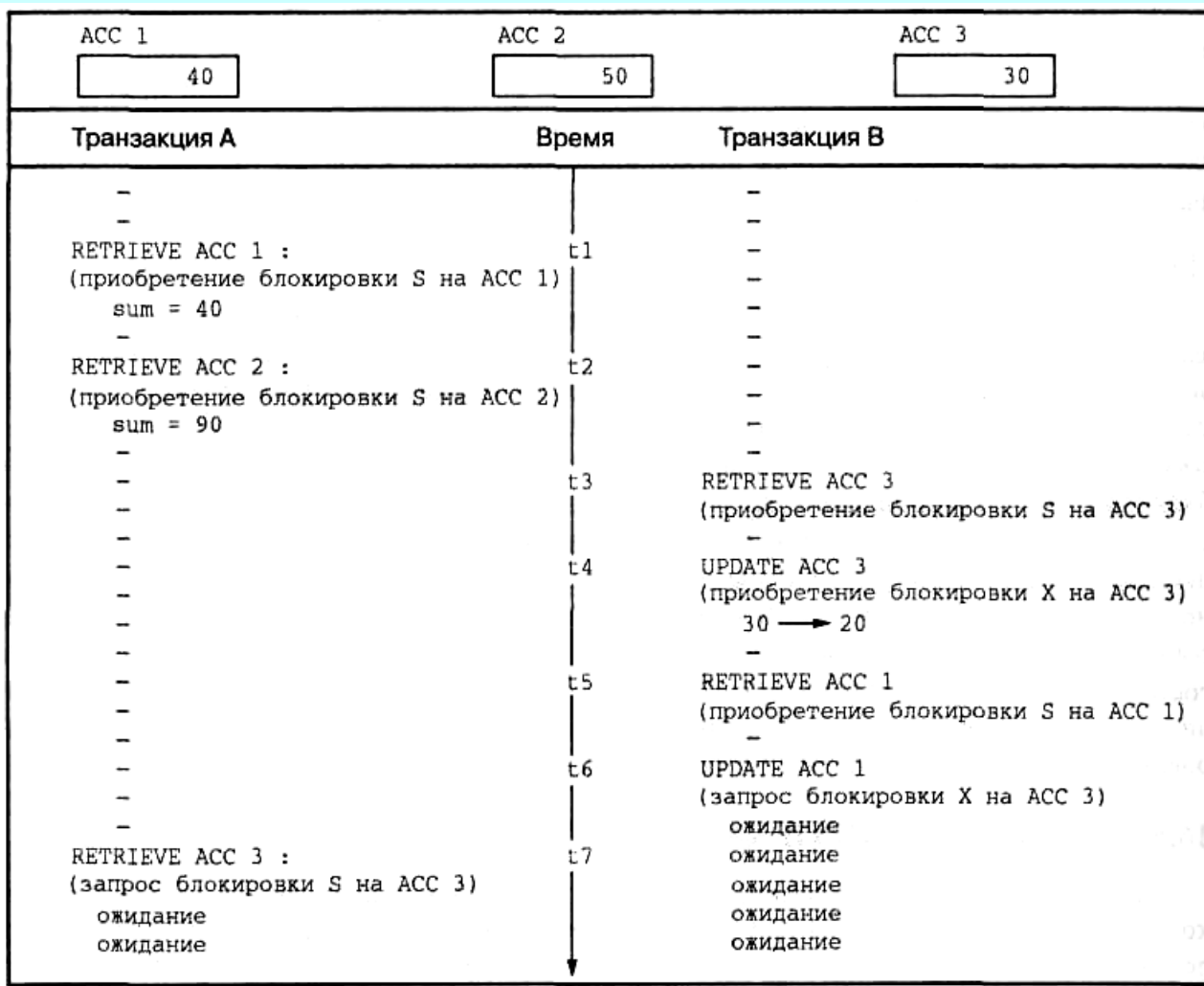
Потеря обновления не определяется, но в момент времени t4 возникает

Проблема зависимости от незафиксированных результатов



Пример того, что для транзакции А
исключена возможность
обновлять незафиксированное изменение в
момент времени t2

Проблема анализа несовместимости исключается, но в момент времени t_7 возникает взаимоблокировка



Выполнение операции UPDATE транзакцией В в момент времени t_6 не допускается, поскольку она представляет собой неявный запрос на блокировку X кортежа ACC 1, а такой запрос конфликтует с блокировкой S, которой уже владеет транзакция А, поэтому транзакция В переходит в состояние ожидания.

выполнение операции RETRIEVE транзакцией А в момент времени t_7 также не допускается, поскольку она представляет собой неявный запрос на блокировку S кортежа ACC 3, а такой запрос конфликтует с блокировкой X, которой уже владеет транзакция В, поэтому транзакция А также переходит в состояние ожидания.

Предотвращение взаимоблокировок

Каждая транзакция обозначается отметкой времени ее начала (которая должна быть уникальной).

Если транзакция А запрашивает блокировку на кортеже, который уже заблокирован транзакцией В, то выполняются описанные ниже действия в зависимости от применяемого варианта.

- "Ожидание-отмена" (Wait-Die). Если выполнение транзакции А началось раньше, чем В, А переходит в состояние ожидания; в противном случае происходит ее *отмена*. Это означает, что *осуществляется откат и перезапуск транзакции А*.
- "Отмена—ожидание" (Wound-Wait). Если выполнение транзакции А началось позже, чем В, она переходит в состояние ожидания; в противном случае, она *отменяет В*. Это означает, что *происходит откат и перезапуск транзакции В*.

Если транзакция должна быть перезапущена, ей после запуска присваивается ее первоначальная отметка времени.

Намеченные блокировки

- **протокол намеченной блокировки**, согласно которому ни одной транзакции не разрешается приобрести блокировку на кортеже перед тем, как будет вначале приобретена блокировка (а, возможно, лишь намечена такая блокировка, как описано в следующем абзаце) на переменную отношения, которая ее содержит.

Намеченные блокировки

- **Намеченная разделяемая блокировка (IS).** В транзакции t намечается установка блокировок S на отдельных кортежах переменной отношения R для того, чтобы гарантировать постоянство этих кортежей в процессе их обработки.
- **Намеченная исключительная блокировка (IX).** То же, что и **IS**, наряду с тем, что t может обновлять отдельные кортежи в R и поэтому устанавливать блокировки X на этих кортежах.
- **Разделяемая блокировка (S).** Транзакция t может допускать параллельное применение других транзакций чтения, но не параллельное применение других транзакций обновления в переменной отношения R (сама транзакция t не обновляет ни одного из кортежей BR).
- **Разделяемая намеченная исключительная блокировка (SIX).** Представляет собой сочетание S и IX ; это означает, что t допускает присутствие параллельно выполняемых транзакций чтения, но не параллельно выполняемых транзакций обновления в R , но наряду с этим в транзакции t могут обновляться отдельные кортежи в R и поэтому должны устанавливаться блокировки X на эти кортежи.
- **Исключительная блокировка (X).** Транзакция t вообще не допускает выполнения какого-либо параллельного с ней доступа к R (в самой транзакции T отдельные кортежи R могут обновляться или не обновляться).

Матрица совместимости, расширенная с учетом намеченных блокировок

	X	SIX	IX	S	IS	-
X	Н	Н	Н	Н	Н	Д
SIX	Н	Н	Н	Н	Д	Д
IX	Н	Н	Д	Н	Д	Д
S	Н	Н	Н	Д	Д	Д
IS	Н	Д	Д	Д	Д	Д
-	Д	Д	Д	Д	Д	Д

Фантомы (фантомные строки)

- Прежде всего, предположим, что в транзакции А вычисляется средний остаток на всех счетах, принадлежащих клиенту Джо. Допустим, что в настоящее время имеются три таких счета и на каждом из них остаток составляет 100 долл. Поэтому транзакция А просматривает три счета, в ходе своего выполнения приобретает на них разделяемые блокировки и получает результат (100 долл.).
- А теперь предположим, что выполняется параллельная транзакция в, в результате которой в базу данных вводится еще один счет клиента Джо, с остатком 200 долл. Для определенности допустим, что новый счет вводится после того, как в транзакции А было вычислено среднее значение, равное 100 долл. Предположим также, что сразу же после введения нового счета в транзакции в происходит фиксация (и освобождение исключительной блокировки нового счета, которой она владела).
- Затем допустим, что было принято решение снова воспользоваться транзакцией А для просмотра счетов клиента Джо, подсчета их количества и суммирования их остатков, а затем деления суммы остатков на количество счетов (возможно, потому что пользователь захотел определить, действительно ли полученное раньше среднее значение равно сумме, деленной на количество). На данный момент обнаруживаются четыре счета вместо трех, а полученный результат равен 125 долл. вместо 100 долл.!

Фантомные строки

< Result Grid Filter Rows: Edit:

	id_st	surname	name	patronym	id_gr
▶	1	Иванов	Иван	Иванович	1
	2	Петров	Петр	Петрович	1
	3	Сидоров	Сидор	Сидорпович	2
	4	Кузнецов	Кузьма	Кузьмич	2
	5	Иванова	Анна	Ивановна	3

student 1 x

< Result Grid Filter

	id_gr	number_gr
▶	1	4831
	2	4832K
	3	5811
*	NULL	NULL

Фантомные строки

```
19
20 • start transaction ;
21 • select count(id_st) from student join
22   st_group on student.id_gr=st_group.id_gr
23   where number_gr='4831'
24   group by st_group.id_gr;
```

Result Grid	Filter Rows:	Export:
count(id_st)		
2		

- start transaction ;
- INSERT INTO `student` (`id_st`,`surname`,`name`,`patronym`,`id_gr`) VALUES (6,'test','test name','test',1);
- commit;

```
25
26 • select count(id_st) from student join
27   st_group on student.id_gr=st_group.id_gr
28   where number_gr='4831'
29   group by st_group.id_gr;
30 • commit ;
```

Result Grid	Filter Rows:	Export:
count(id_st)		
3		

Уровни изоляции транзакций

- READ UNCOMMITTED
- READ COMMITTED
- REPEATABLE READ
- SERIALIZABLE

Уровень изоляции	Грязное чтение	Неповторяемое чтение	Фантомы
READ UNCOMMITTED	Y	Y	Y
READ COMMITTED	N	Y	Y
REPEATABLE READ	N	N	Y
SERIALIZABLE	N	N	N





Уровни изоляции транзакций MySQL (установка)

- **SET** [GLOBAL | SESSION] TRANSACTION *transaction_characteristic* [, *transaction_characteristic*] ...
- *transaction_characteristic*: { ISOLATION LEVEL *level* | *access_mode* }
- *level*: { **REPEATABLE READ** | READ COMMITTED | READ UNCOMMITTED | SERIALIZABLE }
- *access_mode*: { READ WRITE | READ ONLY }

Как устанавливать уровни изоляции транзакции

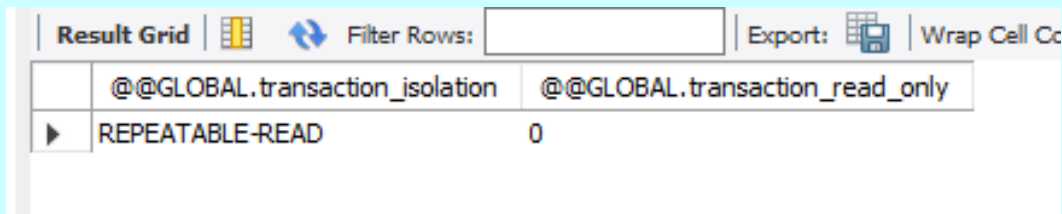
Синтаксис	На что влияет
SET GLOBAL TRANSACTION <i>transaction_characteristic</i>	Глобальная переменная
SET SESSION TRANSACTION <i>transaction_characteristic</i>	Переменная сессии
SET TRANSACTION <i>transaction_characteristic</i>	Только следующая транзакция

```
12
13 • SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED ;
14 • SELECT @@SESSION.transaction_isolation, @@SESSION.transaction_read_only;
```

  Filter Rows: <input type="text"/>		Export: 	Wrap Cell Content: 
@@SESSION.transaction_isolation	@@SESSION.transaction_read_only		
▶ READ-COMMITTED	0		

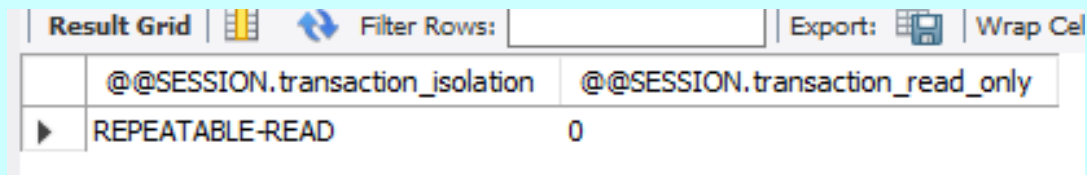
Уровни изоляции транзакций MySQL (получение)

- **SELECT @@GLOBAL.transaction_isolation,
@@GLOBAL.transaction_read_only;**
- **SELECT @@SESSION.transaction_isolation,
@@SESSION.transaction_read_only;**



The screenshot shows a MySQL query result window with a toolbar at the top containing 'Result Grid', a grid icon, a 'Filter Rows:' input field, an 'Export:' button with a grid icon, and a 'Wrap Cell Cc' option. The table below has two columns: '@@GLOBAL.transaction_isolation' and '@@GLOBAL.transaction_read_only'. The first row of data shows 'REPEATABLE-READ' and '0'.

@@GLOBAL.transaction_isolation	@@GLOBAL.transaction_read_only
REPEATABLE-READ	0



The screenshot shows a MySQL query result window with a toolbar at the top containing 'Result Grid', a grid icon, a 'Filter Rows:' input field, an 'Export:' button with a grid icon, and a 'Wrap Cell Cc' option. The table below has two columns: '@@SESSION.transaction_isolation' and '@@SESSION.transaction_read_only'. The first row of data shows 'REPEATABLE-READ' and '0'.

@@SESSION.transaction_isolation	@@SESSION.transaction_read_only
REPEATABLE-READ	0