Организация многопользовательского доступа к данным

"Кто хочет работать – ищет средства, кто не хочет – причины".

С.П. Королёв, советский ученый и конструктор в области космонавтики

Механизм транзакций

Транзакция – это упорядоченная последовательность операторов обработки данных, которая переводит базу данных из одного согласованного состояния в другое.

Транзакция обладает следующими свойствами:

- Логическая неделимость (**атомарность**, Atomicity) означает, что выполняются либо все операции (команды), входящие в транзакцию, либо ни одной.
- **Согласованность** (Consistency): транзакция начинается на согласованном множестве данных и после её завершения множество данных согласовано.
- **Изолированность** (Isolation), т.е. отсутствие влияния транзакций друг на друга.
- **Устойчивость** (Durability): результаты завершённой транзакции не могут быть потеряны. Возврат БД в предыдущее состояние может быть достигнут только путём запуска компенсирующей транзакции.

Транзакции, удовлетворяющие этим свойствам, называют ACID-транзакциями (по первым буквам названий свойств).

Команды управления транзакциями

Для управления транзакциями в системах, поддерживающих механизм транзакций и язык SQL, используются следующие операторы:

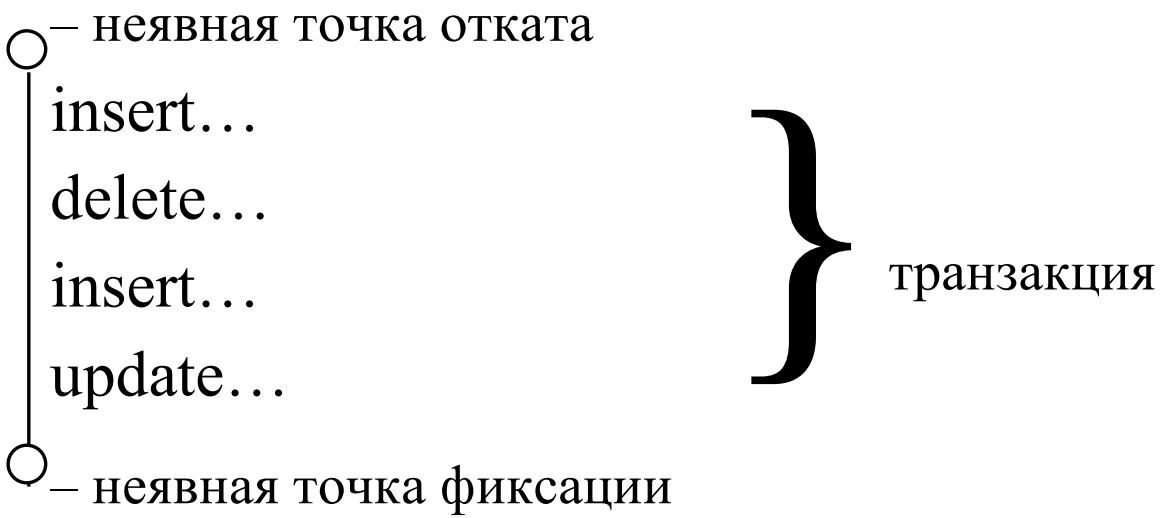
– **фиксация** транзакции (запоминание изменений):

COMMIT [WORK];

– **откат** транзакции (отмена изменений):

ROLLBACK [WORK];

– создание точки сохранения: **SAVEPOINT** <имя_точки_сохранения>;



Механизмы обеспечения работы транзакций

Сегмент отката (rollback segment, RBS) – это специальная область памяти на диске, в которую записывается информация обо всех текущих (незавершённых) изменениях. Данные в RBS хранятся до тех пор, пока транзакция, изменяющая эти данные, не будет завершена. Потом они могут быть перезаписаны данными более поздних транзакций.
Для сохранения сведений о транзакциях СУБД ведёт журнал транзакций.
Журнал транзакций – это часть БД, в которую поступают данные обо всех изменениях всех объектов БД. Журнал недоступен пользователям СУБД и поддерживается особо тщательно (иногда ведутся две копии журнала, хранимые на разных физических носителях). Форма записи в журнал изменений зависит от СУБД. Но обычно там фиксируется следующее:

- номер транзакции (номера присваиваются автоматически по возрастанию);
- состояние транзакции (завершена фиксацией или откатом, не завершена, находится в состоянии ожидания);
- точки сохранения (явные и неявные);
- команды, составляющие транзакцию, и проч.

Начало и завершение транзакции

Начало транзакции соответствует появлению первого исполняемого SQL-оператора. При этом в журнале появляется запись об этой транзакции.

По стандарту ANSI/ISO транзакция завершается при наступлении одного из следующих событий:

- Поступила команда commit (результаты транзакции фиксируются).
- Поступила команда rollback (результаты транзакции откатываются).
- Успешно завершена программа (exit, quit), в рамках которой выполнялась транзакция. В этом случае транзакция фиксируется автоматически.
- Программа, выполняющая транзакцию, завершена аварийно (abort). При этом транзакция автоматически откатывается.

Примечания:

Возможна работа в режиме AUTOCOMMIT, когда каждая команда воспринимается системой как транзакция. В этом режиме пользователи меньше задерживают друг друга, требуется меньше памяти для сегмента отката, зато результаты ошибочно выполненной операции нельзя отменить командой rollback.

В некоторых СУБД реализованы расширенные модели транзакций, в которых существуют дополнительные ситуации фиксации транзакций. Например, в СУБД Oracle команды DDL выполняются в режиме AUTOCOMMIT, т.е. не могут быть откачены.

Запись изменений на диск

Все изменения данных выполняются в оперативной памяти в буфере данных, затем фиксируются в журнале транзакций и в сегменте отката и периодически (при выполнении контрольной точки) переписываются на диск.

Процесс формирования **контрольной точки** (КТ) заключается в синхронизации данных, находящихся на диске (т.е. во вторичной памяти) с теми данными, которые находятся в ОП: все модифицированные данные из ОП переписываются во вторичную память. В разных системах процесс формирования контрольной точки запускается по-разному.

Например, в СУБД Oracle КТ формируется:

- при поступлении команды commit,
- при переполнении буфера данных,
- в момент заполнения очередного файла журнала транзакций,
- через три секунды со времени последней записи на диск.

Внесение изменений в журнал транзакций всегда носит опережающий характер по отношению к записи изменений в основную часть БД (протокол WAL – Write Ahead Log). Эта стратегия заключается в том, что запись об изменении любого объекта БД должна попасть во внешнюю память журнала транзакций раньше, чем изменённый объект попадёт во внешнюю память основной части БД.

Действия при завершении транзакции

При использовании протокола WAL изменённые данные почти сразу попадают в базу данных, ещё по поступления команды commit.
Поэтому фиксация транзакции чаще всего заключается в следующем:

1. Изменения, внесённые транзакцией, помечаются как постоянные.
2. Уничтожаются все точки сохранения для данной транзакции.
3. Если выполнение транзакций осуществляется с помощью блокировок, то освобождаются объекты, заблокированные транзакцией.
4. В журнале транзакций транзакция помечается как завершённая, уничтожаются системные записи о транзакции в оперативной памяти.

А при откате транзакции вместо п.1 обычно выполняется считывание из сегмента отката прежних значений данных и переписывание их обратно в БД (остальные пункты сохранятся без изменений). Поэтому откат транзакции практически всегда занимает больше времени, чем фиксация.

Взаимовлияние транзакций

Транзакции в многопользовательской БД должны быть изолированы друг от друга, т.е. в идеале каждая из них должна выполняться так, как будто выполняется только она одна. В реальности транзакции выполняются одновременно и могут влиять на результаты друг друга, если они обращаются к одному и тому же набору данных и хотя бы одна из транзакций изменяет данные.

Транзакции не влияют друг на друга, если:
1. они только читают данные;
2. они обращаются к разным данным.

В общем случае взаимовлияние транзакций может проявляться в виде:
1. потери изменений;
2. чернового чтения;
3. неповторяемого чтения;
4. фантомов.

Потеря изменений

Представим, что одновременно начали выполняться две транзакции:

транзакция 1 – UPDATE *СОТРУДНИКИ*
SET *Оклад* = 39200
WHERE ***Номер*** = 1123;

транзакция 2 – UPDATE *СОТРУДНИКИ*
SET *Должность* = "старший экономист"
WHERE ***Номер*** = 1123;

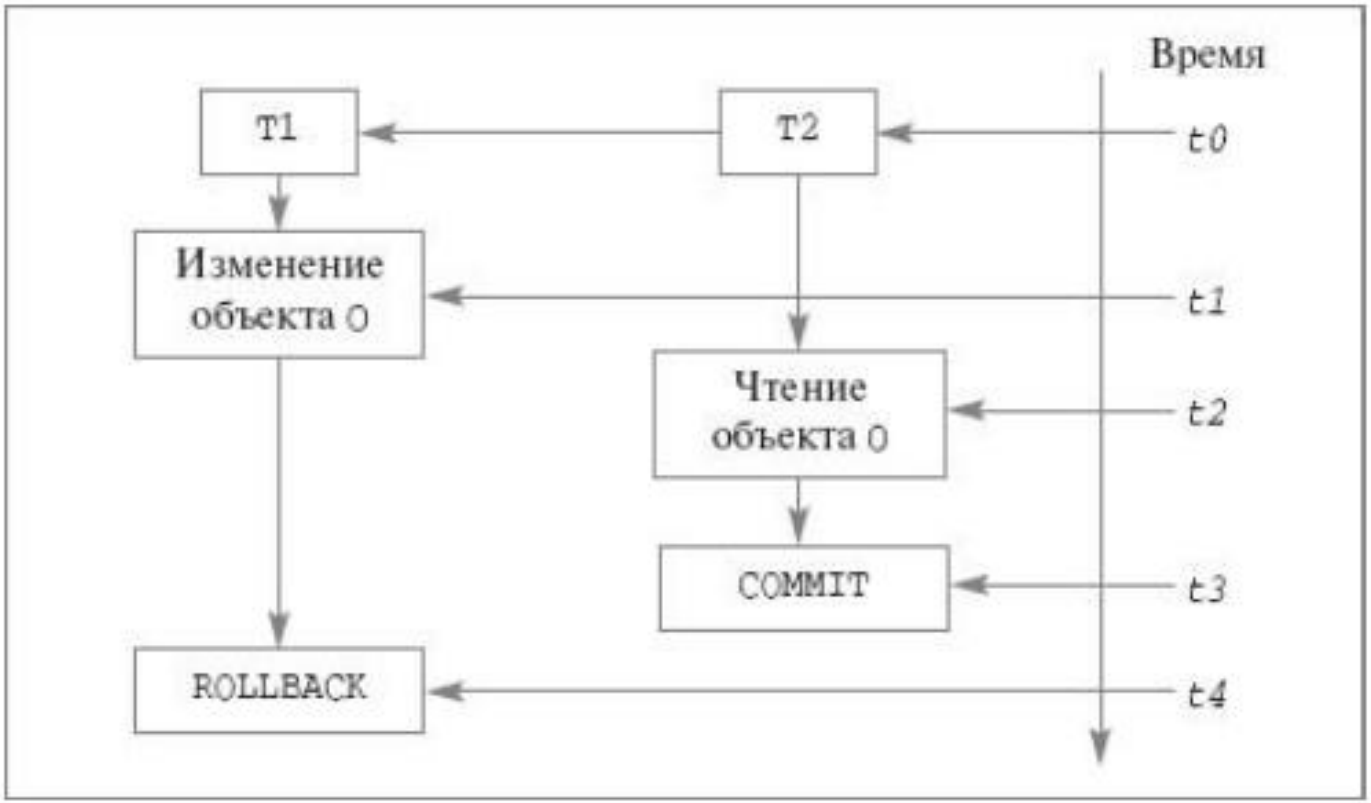
Отношение "Сотрудники"

Номер	ФИО	Должность	Оклад	
1123	Рудин В.П.	экономист	28300	
1123	Рудин В.П.	экономист	**39200**	Транзакция 1
1123	Рудин В.П.	**старший экономист**	28300	Транзакция 2

СУБД не допускает такого взаимовлияния транзакций, при котором возможна потеря изменений.

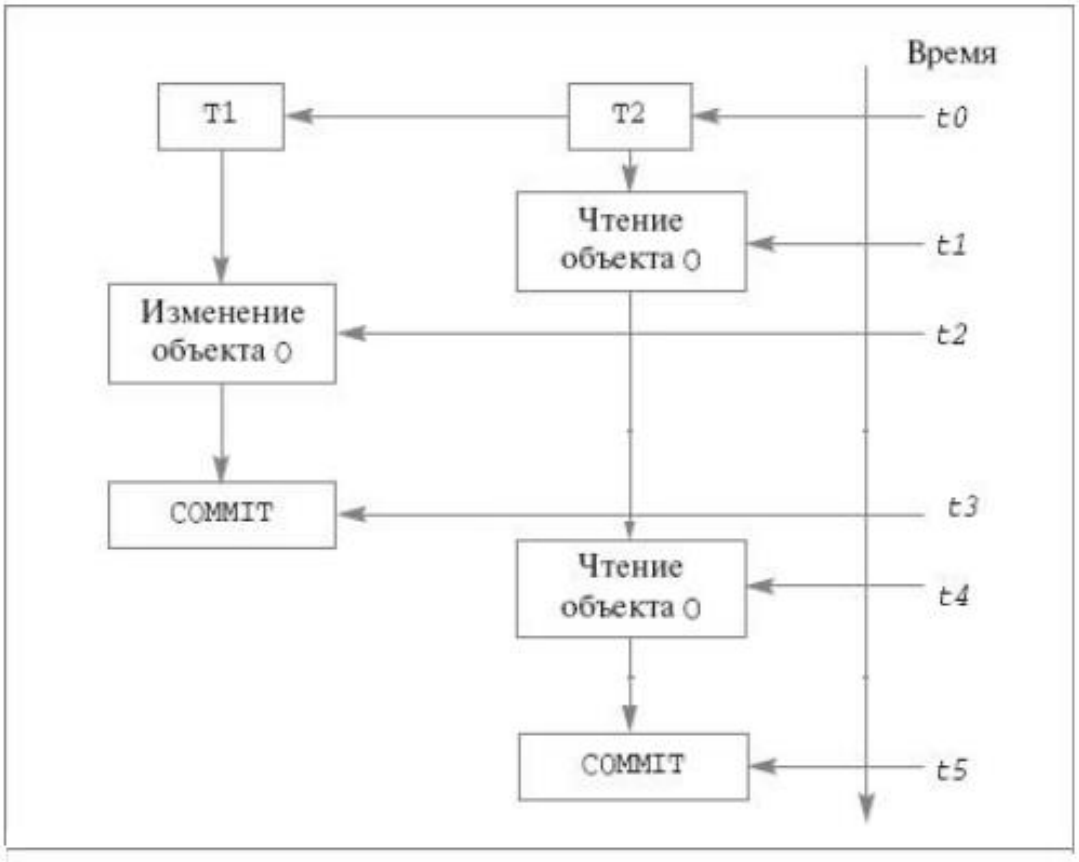
Черновое чтение

Ситуация **чернового чтения** возникает, когда транзакция считывает изменения, вносимые другой (незавершённой) транзакцией.



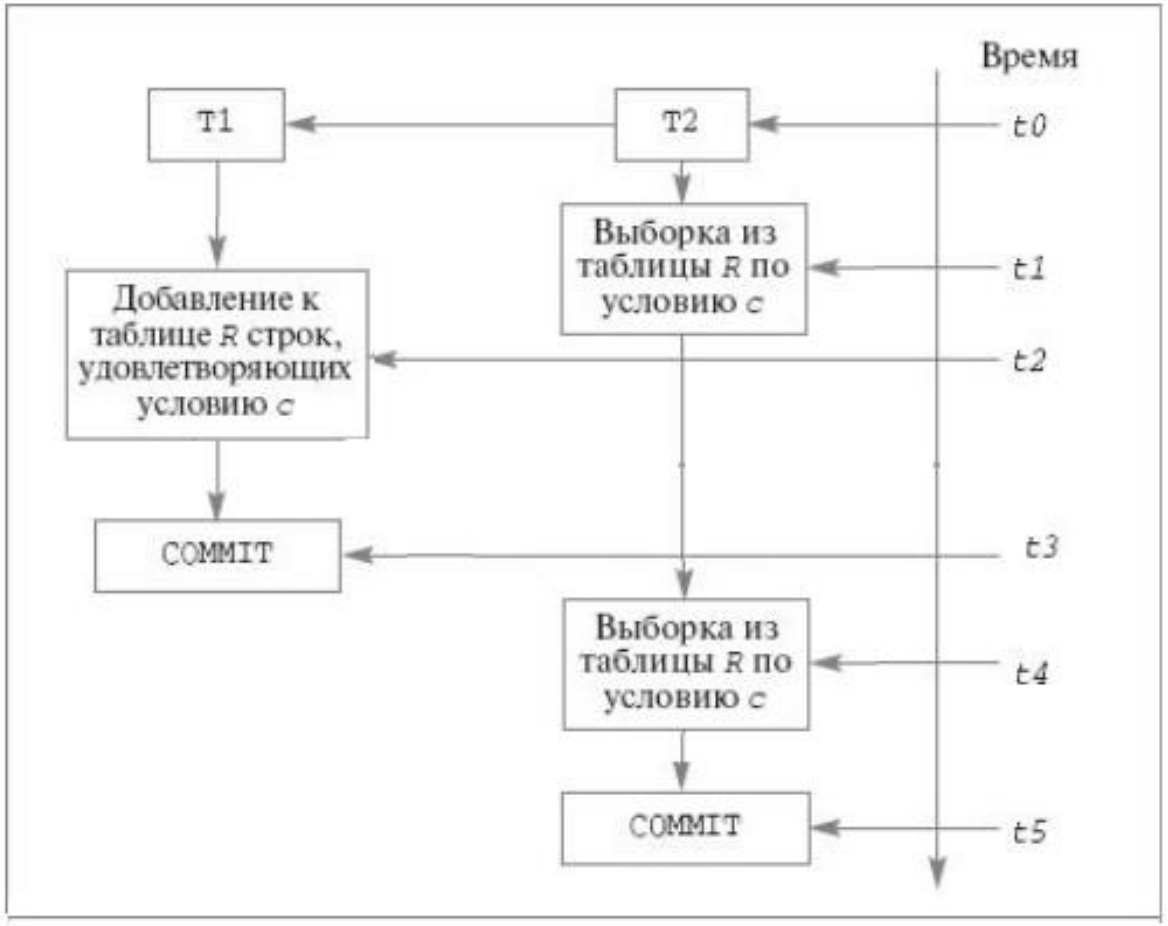
Неповторяемое чтение

Неповторяемое чтение является противоположностью повторяемого, т.е. транзакция "видит" изменения существующих данных, внесённые другими (завершёнными!) транзакциями.



Фантомы

Фантомы возникают, если транзакция "видит" новые данные, добавленные другими (завершёнными!) транзакциями.



Уровни изоляции транзакций

Уровень изоляции	Черновое чтение	Неповторяемое чтение	Фантомы
Read Uncommited – чтение незавершённых транзакций	да	да	да
Read Commited – чтение завершённых транзакций	нет	да	да
Repeatable Read – повторяемое чтение	нет	нет	да
Serializable – последовательное чтение	нет	нет	нет

Блокировки

Блокировка – это временное ограничение доступа к данным, участвующим в транзакции, со стороны других транзакций.

Различают следующие типы блокировок:

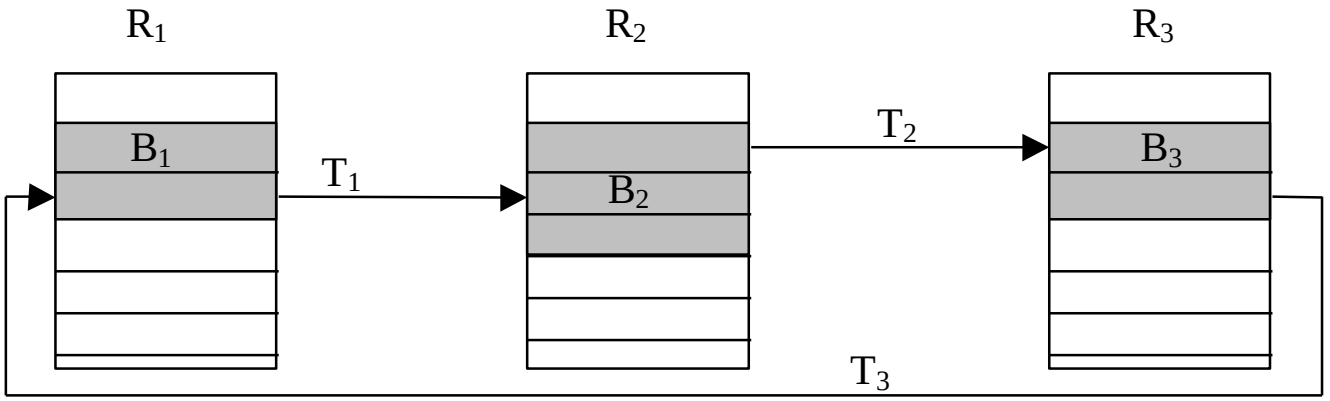
- по степени доступности данных: разделяемые и исключающие;
- по множеству блокируемых данных: строчные, страничные, табличные;
- по способу установки: автоматические и явные.

Явная блокировка, накладываемая командой LOCK TABLE языка SQL.
LOCK TABLE <имя таблицы> IN <тип блокировки> [NOWAIT | WAIT];

Явную блокировку также можно наложить с помощью ключевых слов for update, например:

```
SELECT *
    FROM <имя_таблицы>
    WHERE <условие>
    for update;
```

Тупиковые ситуации (deadlocks)



Стратегии разрешения проблемы взаимной блокировки:

- Транзакция запрашивает сразу все требуемые блокировки.
- СУБД отслеживает возникающие тупики и отменяет одну из транзакций с последующим рестартом через случайный промежуток времени. Этот метод требует дополнительных накладных расходов.
- Вводится **таймаут (**time-out) – максимальное время, в течение которого транзакция может находиться в состоянии ожидания. Если транзакция находится в состоянии ожидания дольше таймаута, считается, что она находится в состоянии тупика, и СУБД инициирует её откат с последующим рестартом через случайный промежуток времени.

Временн***ы***е отметки

Временн***а***я отметка – это уникальный идентификатор, который СУБД создаёт для обозначения относительного момента запуска транзакции. Каждая транзакция ***Ti*** имеет временн***у***ю отметку ***ti***, и каждый элемент данных в БД (запись или блок) имеет две отметки:

tread(x) – временн***а***я отметка транзакции, которая последней считала элемент ***x***,

twrite(x) – временн***а***я отметка транзакции, которая последней записала элемент ***x***.

При выполнении транзакции ***Ti*** система сравнивает отметку ***ti*** и отметки ***tread(x)*** и ***twrite(x)*** элемента ***x*** для обнаружения конфликтов:

для читающей транзакции ***Ti***:

ti < twrite(x).

для пишущей транзакции:

ti < tread(x),

ti < twrite(x).

Во всех случаях обнаружения конфликта система перезапускает текущую транзакцию ***Ti*** с более поздней временн***о***й отметкой.

Многовариантность

Алгоритм многовариантности позволяет обеспечивать согласованность данных при чтении, не блокируя эти данные.
Согласованность данных для операции чтения заключается в том, что все значения данных должны относиться к тому моменту, когда начиналась эта операция. Для этого можно предварительно запретить другим транзакциям изменять эти данные до окончания операции чтения, но это снижает степень параллельности работы системы.
При использовании алгоритма многовариантности каждый блок данных хранит номер последней транзакции, которая модифицировала данные, хранящиеся в этом блоке (SCN – system change number). И каждая транзакция имеет свой SCN. При чтении данных СУБД сравнивает номер транзакции и номер считываемого блока данных:
• если блок данных не модифицировался с момента начала чтения, то данные считываются из этого блока;
• если данные успели измениться, то система обратится к сегменту отката и считает оттуда значения данных, относящиеся к моменту начала чтения.

Недостаток: возможность возникновения ошибки при чтении данных, если старые значения данных в сегменте отката будут перезаписаны (в Oracle это ошибка too_old_snapshot).

Сводка блокировок таблиц для Oracle

Предложение SQL	Режим блокировки таблицы	Разрешенные блокировки				
		RS	RX	S	SRX	X
SELECT...FROM таблица	нет	да	да	да	да	да
INSERT INTO таблица...	RX	да	да	нет	нет	нет
UPDATE таблица...	RX	да*	да*	нет	нет	нет
DELETE FROM таблица...	RX	да*	да*	нет	нет	нет
SELECT...FROM таблица FOR UPDATE OF...	RS	да*	да*	да*	да*	нет
LOCK TABLE таблица IN ROW SHARE MODE	RS	да	да	да	да	нет
LOCK TABLE таблица IN ROW EXCLUSIVE MODE	RX	да	да	нет	нет	нет
LOCK TABLE таблица IN SHARE MODE	S	да	нет	да	нет	нет
LOCK TABLE таблица IN SHARE ROW EXCLUSIVE MODE	SRX	да	нет	нет	нет	нет
LOCK TABLE таблица IN EXCLUSIVE MODE	X	нет	нет	нет	нет	нет

RS: row share SRX: share row exclusive RX: row exclusive X: exclusive S: share

* - если другая транзакция не удерживает конфликтующих блокировок; в противном случае возникает ожидание.

Умалчиваемые блокировки

Команды DML включают следующие виды предложений:

SELECT ... FOR UPDATE OF ...;
INSERT ... SELECT ...;
UPDATE ...;
DELETE ...;

Они имеют следующие характеристики в смысле блокировок:

* Транзакция, содержащая предложение DML, получает монопольные блокировки строк по строкам, модифицируемым этим предложением. Поэтому другие транзакции, желающие обновить или удалить эти же строки, должны ожидать подтверждения или отката первой транзакции.

* Транзакция, содержащая предложение DML, не нуждается в получении блокировки по строкам, выбираемым подзапросом или неявным запросом, таким как запрос в фразе WHERE. Подзапросу или неявному запросу в предложении DML гарантируется согласованность данных на момент начала запроса, и этот запрос не видит эффектов от предложения DML, частью которого он является.

Умалчиваемые блокировки

Команды DML имеют следующие характеристики в смысле блокировок:

* Запрос в транзакции видит изменения, сделанные предыдущими предложениями DML в той же самой транзакции, но не видит никаких неподтвержденных изменений от других активных транзакций.

* Помимо необходимых монопольных блокировок строк, транзакция, содержащая предложение DML, получает по меньшей мере монопольную для строк блокировку по таблице, в которой содержатся эти строки. Если эта транзакция уже имеет блокировку S, SRX или X по данной таблице, то монопольная для строк блокировка таблицы не запрашивается; если же эта транзакция уже имеет разделяемую для строк блокировку по таблице (RS), то ORACLE автоматически конвертирует ее в блокировку RX.

Продолжительность блокировок данных

Все блокировки данных, получаемые транзакцией, включая все блокировки строк и таблиц, освобождаются при подтверждении или откате этой транзакции.

Блокировки данных, полученные после точки сохранения, освобождаются, если транзакция откатывается к этой точке сохранения.

Однако, лишь те транзакции, которые не ожидают ранее заблокированных ресурсов, могут получить блокировки для вновь освободившихся ресурсов.

Ждущие транзакции продолжат ожидать, пока вся транзакция полностью не будет подтверждена или отменена.

Блокировки DDL (блокировки словаря)

Блокировка DDL защищает определение объекта схемы (например, таблицы), пока этот объект подвергается воздействию продолжающейся операции DDL в транзакции (вспомните, что предложение DDL неявно подтверждает предыдущую транзакцию).
Например, пользователь создает процедуру. От имени транзакции пользователя, состоящей из единственного предложения (CREATE), ORACLE автоматически получает блокировки DDL для всех объектов, упоминаемых в определении процедуры.
Блокировки DDL предотвращают изменение (ALTER) или удаление (DROP) объектов, на которые ссылается процедура, до тех пор, пока не завершится компиляция этой процедуры.
Блокировка словаря автоматически запрашивается ORACLE от имени любой транзакции DDL, требующей такой блокировки. Пользователи не могут явно запрашивать блокировок DDL. Блокируются лишь индивидуальные объекты схем, к которым имеются обращения; словарь данных в целом никогда не блокируется.
Блокировки DDL распадаются на три категории: монопольные блокировки DDL, разделяемые блокировки DDL и прерываемые блокировки разбора.

Монопольные блокировки DDL

Некоторые операции DDL требуют монопольных блокировок DDL для ресурса, чтобы предотвратить деструктивное взаимодействие с другими операциями DDL, которые могут обращаться к тому же самому объекту или модифицировать его. Например, операция DROP TABLE, удаляющая таблицу, не может быть разрешена, пока операция ALTER TABLE добавляет столбец в эту таблицу, и наоборот.

Помимо блокировок DDL, операции DDL получают также блокировки DML (блокировки данных) по модифицируемому объекту.

Большинство операций DDL требуют монопольных блокировок DDL по модифицируемому объекту (за исключением тех операций, которые перечислены в разделе «Разделяемые блокировки DDL»).

Во время запроса на монопольную блокировку DDL, если для данного объекта другой операцией уже получена другая блокировка DDL, запрос ожидает, пока более старая блокировка DDL не будет снята.

Разделяемые блокировки DDL

Некоторые операции DDL требуют разделяемых блокировок DDL для ресурса, чтобы предотвратить деструктивное взаимодействие с конфликтующими операциями DDL, но разрешить одновременный доступ для аналогичных операций DDL.
Например, когда выполняется предложение CREATE PROCEDURE, содержащая транзакция получает разделяемые блокировки DDL для всех таблиц, встречающихся в процедуре. Другие транзакции могут одновременно создавать процедуры, обращающиеся к тем же самым таблицам, и поэтому могут получить разделяемые блокировки DDL по этим же таблицам, но ни одна транзакция не сможет получить монопольную блокировку DDL ни по одной из этих таблиц, т.е. никакая транзакция не сможет изменить или удалить эти таблицы.
Как следствие, транзакции, удерживающей разделяемую блокировку DDL, гарантируется, что определение объекта, к которому она обращается, останется неизменным на все время транзакции.
Разделяемая блокировка DDL запрашивается по соответствующим объектам для следующих команд: AUDIT, NOAUDIT, GRANT, REVOKE, COMMENT, CREATE [OR REPLACE] VIEW / PROCEDURE / PACKAGE / PACKAGE BODY / FUNCTION / TRIGGER, CREATE SYNONYM, а также CREATE TABLE (если не включен параметр CLUSTER).

Прерываемые блокировки разбора

Предложение SQL (или программная единица PL/SQL), находящееся в разделяемом пуле, удерживает блокировку разбора для каждого объекта, к которому обращается это предложение SQL.
Блокировки разбора запрашиваются с тем, чтобы ассоциированная разделяемая область SQL могла быть помечена как недействительная в случае изменения или удаления соответствующего объекта.
Блокировка разбора не запрещает никаких операций DDL, и может быть прервана, чтобы снять конфликт с операциями DDL; отсюда ее название.
Блокировка разбора запрашивается на фазе синтаксического разбора предложения SQL, и удерживается столько, сколько разделяемая область SQL остается в разделяемом пуле.

Продолжительность блокировок DDL
Продолжительность блокировки DDL зависит от ее типа. Монопольные и разделяемые блокировки DDL удерживаются на время выполнения предложения DDL и последующего неявного завершения транзакции.
Блокировка разбора сохраняется столько, сколько ассоциированное предложение SQL остается в разделяемом пуле.