**Транзакции, уровни изоляции, savepoint. Блокировки таблицы.**

**Ссылка на оригинал**

https://docs.oracle.com/en/database/oracle/oracle-database/19/cncpt/transactions.html#GUID-07517D95-D5F9-4851-B737-B076336CED13

### Введение в транзакции

**Транзакция** — это логическая атомарная единица некоторого процесса, содержащая одну или несколько команд SQL.

Транзакция группирует команды SQL таким образом, чтобы они либо все выполнялись и фиксировались, т.е. вносимые ими изменения применялись в базе данных, либо все откатывались, т.е. вносимые ими изменения отменялись в базе данных. Oracle Database присваивает каждой транзакции уникальный идентификатор, называемый идентификатором **транзакции (transaction ID)**.

Все транзакции Oracle подчиняются основным свойствам транзакций в базах данных, известными как **свойства ACID**. ACID – это аббревиатура для следующих понятий:

* Атомарность

Все команды транзакции выполняются или ни одна из них не выполняется. Частичных транзакций не существует. Например, если транзакция начинает обновлять 100 строк, но система выходит из строя после 20 обновлений, то база данных откатывает изменения в этих 20 строках.

* Согласованность

Транзакция переводит базу данных из одного согласованного состояния в другое согласованное состояние. Например, в банковской транзакции, которая дебетует сберегательный счет и кредитует текущий счет, сбой не должен приводить к тому, что база данных будет кредитовать только один счет, что приведет к несогласованности данных.

* Изоляция

Эффект транзакции не виден другим транзакциям до тех пор, пока она не будет зафиксирована. Например, один пользователь, обновляющий таблицу hr.employees, не видит незафиксированные изменения employees, внесенные одновременно другим пользователем. Таким образом, пользователям кажется, что транзакции выполняются последовательно.

* Долговечность

Изменения, внесенные зафиксированными транзакциями, являются постоянными. После завершения транзакции база данных с помощью механизмов восстановления гарантирует, что изменения из транзакции не будут потеряны.

Использование транзакций является одним из наиболее важных способов отличия системы управления базами данных от файловой системы.

#### **Пример транзакции: Дебет и кредит счета**

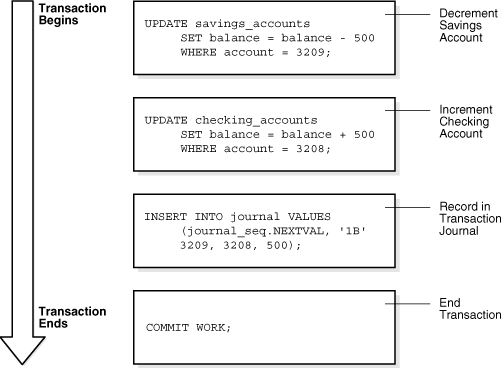
Чтобы проиллюстрировать концепцию транзакции, рассмотрим банковскую базу данных.

Когда клиент переводит деньги со сберегательного счета на текущий счет, транзакция должна состоять из трех отдельных команд:

* Уменьшите размер сберегательного счета
* Увеличьте текущий счет
* Записать транзакцию в журнал транзакций

База данных Oracle должна учитывать две ситуации. Если все три команды SQL поддерживают баланс счетов в правильном состоянии, то результаты транзакции могут быть зафиксированы в базе данных. Однако, если возникает некоторая проблема (например, недостаток средств, или неверный номер счета или аппаратный сбой) не позволяет завершить одну или две команды внутри транзакции, то база данных должна откатить всю транзакцию, чтобы баланс всех счетов был корректным.

На следующем рисунке показана банковская транзакция. Первая команда вычитает 500 долларов со сберегательного счета 3209. Вторая команда добавляет 500 долларов на расчетный счет 3208. Третья команда вставляет запись о переносе в таблицу журнала. Последняя команд фиксирует транзакцию.

  
Рисунок: Банковская операция

#### **Структура транзакции**

Транзакция базы данных состоит из одной или нескольких команд. Транзакция состоит из одного из следующих действий:

* Одна или несколько команд языка манипулирования данных (DML), которые вместе составляют атомарное изменение базы данных.
* Одна команда из языка определения данных (DDL).

Транзакции имеют начало и конец.

##### **Начало транзакции**

Транзакция начинается в момент, когда встречается первая исполняемая команда SQL.

**Исполняемая команда SQL —** этокоманда SQL, во время которой происходит обращение к экземпляру, включая команды DML и DDL и команду SET TRANSACTION.

В начале транзакции Oracle Database закрепляет транзакцию за доступным сегментом отката для записи записей отката для новой транзакции. Идентификатор транзакции не выделяется до тех пор, пока не будут выделены сегмент отката и слоты таблицы транзакций, что происходит во время выполнения первой команды DML. Идентификатор транзакции уникален для транзакции и представляет номер сегмента отката, слот и порядковый номер транзакции.

В следующем примере выполняется команда UPDATE для начала транзакции и запрашивается в V$TRANSACTION подробная информация о транзакции:

SQL> UPDATE hr.employees SET salary=salary;

107 rows updated.

SQL> SELECT XID AS "txn id", XIDUSN AS "undo seg", XIDSLOT AS "slot",

2 XIDSQN AS "seq", STATUS AS "txn status"

3 FROM V$TRANSACTION;

txn id undo seg slot seq txn status

---------------- ---------- ---------- ---------- ----------------

0600060037000000 6 6 55 ACTIVE

##### **Завершение транзакции**

Транзакция может завершиться при различных обстоятельствах. Транзакция завершается при выполнении любого из следующих действий:

* Пользователь исполняет COMMIT или ROLLBACK *без* указания SAVEPOINT.

В случае выполнения COMMIT пользователь явно или неявно запрашивает, чтобы изменения в транзакции были зафиксированы. В таком случае изменения, внесенные транзакцией, являются внесенными в БД и видны другим пользователям только после фиксации транзакции.

* Пользователь запускает команду DDL, такую как CREATE, DROP, RENAME, или ALTER.

База данных исполняет неявную команду COMMIT до и после каждой команду DDL. Если текущая транзакция содержит команды DML, то Oracle Database сначала фиксирует транзакцию, а затем запускает и фиксирует команду DDL как новую транзакцию с одной командой.

* Пользователь штатно выходит из большинства утилит и инструментов Oracle Database, в результате чего текущая транзакция неявно фиксируется. Поведение по фиксации при отключении пользователя зависит от конкретного приложения и настраивается.

**Примечание:** Приложения всегда должны явно фиксировать или отменять транзакции до завершения программы.

* Клиентский процесс завершается ненормально, что приводит к неявному откату транзакции с использованием метаданных, хранящихся в таблице транзакций и сегменте отката.

После завершения одной транзакции следующий исполняемая SQL команда автоматически запускает следующую транзакцию. В следующем примере выполняется UPDATE для запуска транзакции, транзакция завершается командой ROLLBACK, а затем выполняется UPDATE для запуска новой транзакции (обратите внимание, что идентификаторы транзакций разные):

SQL> UPDATE hr.employees SET salary=salary;

107 rows updated.

SQL> SELECT XID, STATUS FROM V$TRANSACTION;

XID STATUS

---------------- ----------------

0800090033000000 ACTIVE

SQL> ROLLBACK;

Rollback complete.

SQL> SELECT XID FROM V$TRANSACTION;

no rows selected

SQL> UPDATE hr.employees SET last\_name=last\_name;

107 rows updated.

SQL> SELECT XID, STATUS FROM V$TRANSACTION;

XID STATUS

---------------- ----------------

0900050033000000 ACTIVE

#### **Атомарность на уровне команды**

Oracle Database поддерживает **атомарность на уровне команд**, это означает, что команда SQL является атомарной единицей процессов и либо полностью успешно завершаются, либо полностью откатываются. Успешно завершенная команда отличается от зафиксированной транзакции.

Если команда SQL вызывает ошибку во время выполнения, то она не является успешно завершенной, и поэтому все действия, выполняемые внутри команды – откатываются. Эта операция представляет собой **откат на уровне команды**. Этот откат имеет следующие характеристики:

* Команда SQL, которая не выполняется, приводит к потерям только части процесса, а именной той, которую он выполнил бы сам.

Неудачное выполнения команды не приводит к потере каких-либо действий, совершенных в рамках других команд, которые предшествовали ей в текущей транзакции. Например, если выполнение второй команды UPDATE в “[Примере транзакции: дебет и кредит](https://docs.oracle.com/en/database/oracle/oracle-database/19/cncpt/transactions.html#GUID-A049FE81-8B67-4386-B599-9CDD7E6B6C59) счета” вызывает ошибку и откатывается, то работа, выполненная первой командой UPDATE, не откатывается. Первая команда UPDATE может быть зафиксирован или отменена явно пользователем.

* Эффект отката таков, как если бы команда никогда не выполнялся.

Любые побочные процессы, вызванные действиями атомарной команды, например запуск триггера, считаются частью атомарной команды. Либо все действия, выполняемые в рамках атомарной команды, завершаются успешно, либо ни одни из них не выполняются.

Примером ошибки, вызывающей откат на уровне команды, является попытка вставить дубликат **первичного ключа**. Отдельные команды SQL, участвующие во взаимоблокировке, которые возникает в случае конкуренции за одни и те же данные, также могут вызвать откат на уровне команды. Однако ошибки, обнаруженные во время синтаксического анализа команд SQL, такие как синтаксическая ошибка, еще не были запущены и поэтому не вызывают отката на уровне команд.

#### **Системный номер изменения (SCNS)**

**Системный номер изменения (SCN)** — является логической внутренней меткой времени, используемой базой данных Oracle.

Многоверсионная СУБД Oracle создаёт и поддерживает системный номер изменения **SCN** (**system change number**). Каждая завершённая транзакция увеличивает его. Поэтому можно считать **SCN** уникальным идентификатором последней завершённой транзакции.

Oracle Database увеличивает SCN в **системной глобальной области (SGA)**. Когда транзакция изменяет данные, база данных записывает новый SCN в сегмент отката данных связанных с транзакцией. Затем процесс записи журнала (LGWR) немедленно записывает транзакцию в **журнал (online redo log)**. Зафиксированная транзакция имеет уникальный SCN. Oracle Database также использует SCNS как часть своих **механизмов восстановления экземпляров** и восстановления **данных**.

### *Обзор контроля транзакций*

**Управление транзакциями** — это управление изменениями, вносимыми командами DML, и группировка команд DML в транзакции.

Как правило, разработчики приложений занимаются управлением транзакциями таким образом, чтобы работа выполнялась в логических единицах, а данные поддерживались согласованными.

Управление транзакциями включает в себя использование следующих команд:

* Команда COMMIT завершает текущую транзакцию и вносит все изменения, выполненные в транзакции. COMMIT также стирает все точки сохранения в транзакции и снимает установленные блокировки во время транзакции.
* Команда ROLLBACK отменяет действия, выполненные в текущей транзакции. Команда ROLLBACK TO SAVEPOINT отменяет изменения с момента последней точки сохранения (SAVEPOINT), но не завершает всю транзакцию.
* Команда SAVEPOINT определяет некоторую точку транзакции, к которой впоследствии можно выполнить откат.

Ниже продемонстрирована сессия транзакции.

|  |  |  |
| --- | --- | --- |
| T | Действия | Пояснения |
| t0 | COMMIT; | Эта команда завершает любую существующую транзакцию в сессии. |
| t1 | SET TRANSACTION NAME 'sal\_update'; | Эта команда начинает транзакцию и называет ее sal\_update. |
| t2 | UPDATE employees  SET salary = 7000  WHERE last\_name = 'Banda'; | Эта команда обновляет зарплату для Banda до 7000. |
| t3 | SAVEPOINT after\_banda\_sal; | Эта команда создает именованную точку сохранения after\_banda\_sal, позволяющую откатывать изменения в этой транзакции до этой точки. |
| t4 | UPDATE employees  SET salary = 12000  WHERE last\_name = 'Greene'; | Эта команда обновляет зарплату Грина до 12000. |
| t5 | SAVEPOINT after\_greene\_sal; | Эта команда создает именованную точку сохранения after\_greene\_sal, позволяющую откатывать изменения в этой транзакции до этой точки. |
| t6 | ROLLBACK TO SAVEPOINT  after\_banda\_sal; | Эта команда откатывает транзакцию до t3, отменяя обновление зарплаты Грина в t4. sal\_update Транзакция *не* завершилась. |
| t7 | UPDATE employees  SET salary = 11000  WHERE last\_name = 'Greene'; | Эта команда обновляет зарплату для Greene до 11000. |
| t8 | ROLLBACK; | Эта команда выполняет откат всех изменений в транзакции sal\_update, завершая транзакцию. |
| t9 | SET TRANSACTION NAME 'sal\_update2'; | Эта команда начинает новую транзакцию в сессии и называет ее sal\_update2. |
| t10 | UPDATE employees  SET salary = 7050  WHERE last\_name = 'Banda'; | Эта команда обновляет зарплату для Banda до 7050. |
| t11 | UPDATE employees  SET salary = 10950  WHERE last\_name = 'Greene'; | Эта команда обновляет зарплату для Грина до 10950. |
| t12 | COMMIT; | Эта команда фиксирует все изменения, внесенные в транзакцию sal\_update2, завершая транзакцию. Фиксация гарантирует сохранение изменений в файлах журнала. |

#### **Имена транзакций**

Имя транзакции — является необязательным пользовательским тегом, который служит именем для действий, выполняемой транзакцией. Для указания имени транзакции следует воспользоваться командой SET TRANSACTION ... NAME, которая, если используется, то должна быть указана первой командой транзакции.

В примере выше первая транзакция была названа sal\_update, а вторая sal\_update2.

Использование имён транзакции обеспечивают следующие преимущества:

* Легче отслеживать транзакции.
* Вы можете просматривать имена транзакций вместе с идентификаторами транзакций в приложениях. Например, администратор базы данных может просматривать имена транзакций в Oracle Enterprise Manager (Enterprise Manager) при мониторинге активности системы.
* База данных записывает имена транзакций в соответствующую запись журнала, поэтому вы можете использовать LogMiner для поиска конкретной транзакции в журнале.
* Имена транзакций можно использовать для поиска конкретной транзакции в представлениях словаря данных, таких как V$TRANSACTION.

#### **Активные транзакции**

**Активная транзакция** — это транзакция, которая началась, но еще не зафиксирована или не откатана.

В таблице выше первой командой для изменения данных в транзакции sal\_update является обновление заработной платы Banda. С момента успешного выполнения этого обновления до завершения транзакции командой ROLLBACK транзакция sal\_update активна.

Изменения данных, внесенные транзакцией, являются временными до тех пор, пока транзакция не будет зафиксирована или откатана. Перед завершением транзакции состояние данных выглядит так, как показано в следующей таблице.

Состояние данных до завершения транзакции

|  |  |
| --- | --- |
| Состояние | Описание |
| Oracle Database сгенерировала информацию об откате в SGA | Данные об откате содержат старые значения данных, измененные командами SQL транзакции |
| Oracle Database сгенерировала запись в журнале в SGA | Запись журнала содержит изменение блока данных и изменение блока отмены |
| Были внесены изменения в буферы базы данных в SGA | Изменения данных для зафиксированной транзакции, хранящиеся в буферах базы данных SGA, не обязательно записываются немедленно в файлы данных [средством записи базы данных (DBWR)](https://docs.oracle.com/en/database/oracle/oracle-database/19/cncpt/Chunk520365104.html#GUID-C27AAA54-E60B-49BC-AB04-7B3848EBAFD6). Запись на диск может происходить до или после фиксации |
| Строки, затронутые изменением данных, блокируются | Другие пользователи не могут изменять данные в затронутых строках и не могут видеть незафиксированные изменения |

#### **Точки сохранения**

**Точка сохранения** — это объявленный пользователем промежуточный маркер в контексте транзакции.

Внутри маркер точки сохранения преобразуется в SCN. Точки сохранения разделяют длинную транзакцию на более мелкие части.

Если вы используете точки сохранения в длинной транзакции, у вас есть возможность позже откатить действия, выполненные до текущего действия в транзакции, но выполненных после объявленной точки сохранения в транзакции. Таким образом, если вы допустили ошибку, вам не нужно повторно выполнять каждое действие. В таблице выше создана точку сохранения after\_banda\_sal, чтобы обновление зарплаты Грина можно было откатить к этой точке сохранения.

##### **Откат к точке сохранения**

Откат к точке сохранения в незафиксированной транзакции означает отмену любых изменений, внесенных после указанной точки сохранения, но это не означает откат самой транзакции.

Когда транзакция откатывается к точке сохранения, как при запуске ROLLBACK TO SAVEPOINT after\_banda\_sal в [таблице выше](https://docs.oracle.com/en/database/oracle/oracle-database/19/cncpt/transactions.html#GUID-5BB15405-8A03-47DE-8A20-63E1B83E1361__CHDCIJGA), происходит следующее:

1. Oracle Database откатывает только те команды, которые выполняются после точки сохранения.

В разобранном примере ROLLBACK TO SAVEPOINT производится откат UPDATE для Greene, но без отката UPDATE для Banda.

1. Oracle Database сохраняет точку сохранения, указанную в команде ROLLBACK TO SAVEPOINT, но все последующие точки сохранения теряются.

В разобранном примере ROLLBACK TO SAVEPOINT приводит к потери точки сохранения after\_greene\_sal.

1. Oracle Database освобождает все блокировки таблиц и строк, вызванные после указанной точки сохранения, но сохраняет все блокировки данных, полученные до точки сохранения.

Транзакция остается активной и может быть продолжена.

##### **Транзакции, поставленные в очередь**

В зависимости от сценария транзакции, ожидающие ранее заблокированных ресурсов, могут быть заблокированы после отката к точке сохранения.

Когда транзакция блокируется другой транзакцией, она ставит в очередь саму блокирующую транзакцию, так что вся блокирующая транзакция должна быть зафиксирована или откатана для продолжения заблокированной транзакции.

В сценарии, показанном в следующей таблице, сессия 1 выполняет откат к точке сохранения, созданной до выполнения команды DML. Однако сессия 2 по-прежнему заблокирован, поскольку он ожидает завершения транзакции сессии 1.

***Таблица Откат к точке сохранения***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вре  мя | Сессия 1 | Сессия 2 | Сессия 3 | Описание |
| T0 | UPDATE employees  SET salary=7000  WHERE last\_name= 'Banda'; |  |  | Сессия 1 запускает транзакцию. Сессия устанавливает эксклюзивную блокировку строки Banda (TX) и субэксклюзивную блокировку таблицы (SX) в таблице. |
| T1 | SAVEPOINT after\_banda\_sal; |  |  | Сессия 1 создает точку сохранения с именем after\_banda\_sal. |
| T2 | UPDATE employees  SET salary=12000  WHERE last\_name= 'Greene'; |  |  | Сессия 1 блокирует строку Greene. |
| T3 |  | UPDATE employees SET salary=14000 WHERE last\_name= 'Greene'; |  | Сессия 2 пытается обновить строку Greene, но не может получить блокировку, поскольку сеанс 1 имеет блокировку в этой строке. В сессии 2 транзакция не началась. |
| T4 | ROLLBACK TO SAVEPOINT after\_banda\_sal; |  |  | Сессия 1 откатывает обновление для зарплаты Greene, которое освобождает блокировку строки Greene. Блокировка таблицы, полученная при t0, не снимается.  На данный момент сессия 2 по-*прежнему* блокируется сессией 1, поскольку сессия 2 ставит в очередь *транзакцию сессии 1*, которая еще не завершена. |
| T5 |  |  | UPDATE employees  SET salary= 11000  WHERE last\_name= 'Greene'; | В Greene данный момент строка разблокирована, поэтому сессия 3 получает блокировку для обновления строки Greene. Эта команда запускает транзакцию в сессии 3. |
| T6 | COMMIT; |  |  | Сессия 1 фиксируется, завершая свою транзакцию. Сессия 2 теперь ставится в очередь для обновления строки Greene за транзакцией в сессии 3. |

#### **Откат транзакций**

Откат незафиксированной транзакции отменяет любые изменения данных, выполненные командами SQL в транзакции.

После отката транзакции последствия действий, выполненных в транзакции, больше не существуют. При откате всей транзакции без ссылки на какие-либо точки сохранения Oracle Database выполняет следующие действия:

* Отменяет все изменения, внесенные всеми командами SQL в транзакции, используя соответствующие сегменты отмены

Запись таблицы транзакций для каждой активной транзакции содержит указатель на все данные, изменения в которых должны быть отменены (в обратном порядке применения). База данных считывает данные из сегмента отката, отменяет команду и затем делает пометку в записи отмены как примененную. Таким образом, если транзакция вставляет строку, то откат удаляет ее. Если транзакция обновляет строку, то откат отменяет обновление. Если транзакция удаляет строку, то откат повторно устанавливает ее. В таблице выше ROLLBACK изменения заработной платы Greene и Banda изменяются на противоположные.

* Освобождает все блокировки данных, хранящихся в транзакции
* Удаляет все точки сохранения в транзакции

В таблице выше во время команды ROLLBACK выполняется удаление точки сохранения after\_banda\_sal. Точка сохранения after\_greene\_sal была удалена командой ROLLBACK TO SAVEPOINT.

* Завершение транзакции

В таблице выше база данных остается в том же состоянии, в котором она была после выполнения начальной команды COMMIT.

Продолжительность отката зависит от объема измененных данных.

#### **Фиксация транзакций**

Фиксация завершает текущую транзакцию и делает постоянными все изменения, выполненные в транзакции.

В таблице выше вторая транзакция начинается с sal\_update2 и явного выполнения команды COMMIT и заканчивается им. Изменения, возникшие в результате двух команд UPDATE, теперь являются постоянными.

При фиксации транзакции выполняются следующие действия:

* База данных генерирует SCN для COMMIT.

Внутренняя таблица транзакций для связанных **записей табличного пространства отмены**, зафиксированных транзакцией. Присваивается и записывается в таблицу транзакций соответствующий уникальный SCN транзакции.

* Процесс **записи журнала (LGWR)** записывает оставшиеся записи журнала повтора в буферах журнала повтора в онлайн-журнал повтора и записывает SCN транзакции в онлайн-журнал повтора. *Это атомарное событие представляет собой фиксацию транзакции.*
* Oracle Database освобождает блокировки строк и таблиц.

Пользователи, которые были поставлены в очередь в ожидании блокировок, удерживаемых незафиксированной транзакцией, могут продолжить свою работу.

* Oracle Database удаляет точки сохранения.

В таблице 10-1в транзакции не было точек сохранения sal\_update, поэтому точки сохранения не удалялись.

* Oracle Database выполняет **очистку фиксации**.

Если измененные блоки, содержащие данные из зафиксированной транзакции, все еще находятся в SGA, и если никакая другая сессия не изменяет их, то база данных удаляет информацию о транзакции, связанной с блокировкой (запись ITL), из блоков.

В идеале COMMIT очищает блоки, так что последующие SELECT не должны выполнять эту задачу. Если для конкретной строки не существует записи ITL, она не блокируется. Если запись ITL существует для определенной строки, то она, возможно, заблокирована, поэтому сеанс должен проверить заголовок сегмента отмены, чтобы определить, зафиксирована ли эта заинтересованная транзакция. Если связанная транзакция зафиксирована, то сеанс очищает блок, который генерирует повтор. Однако, если COMMITT ITL был очищен ранее, то проверка и очистка не нужны.

**Примечание:**

Поскольку очистка блока генерирует повтор, запрос может генерировать повтор и, таким образом, вызывать запись блоков во время следующей **контрольной** точки.

* Oracle Database отмечает транзакцию завершенной.

После фиксации транзакции пользователи могут просматривать изменения.

Как правило, фиксация — это быстрая операция, независимо от размера транзакции. Скорость фиксации не увеличивается с размером данных, измененных в транзакции. Самая длинная часть фиксации — это физический дисковый ввод-вывод, выполняемый LGWR. Однако время, затрачиваемое LGWR, уменьшается, поскольку он постепенно записывает содержимое буфера журнала повтора в фоновом режиме.

По умолчанию LGWR синхронно записывает повтор в онлайн-журнал повторов, а транзакции ждут буферизованного повтора на диске, прежде чем возвращать фиксацию пользователю. Однако для снижения задержки фиксации транзакций разработчики приложений могут указать, что повтор записывается асинхронно, чтобы транзакции не ждали, пока повтор будет на диске, и могли немедленно вернуться из COMMIT вызова.