Управление транзакциями в СУБД Oracle

Бежать быстро – это значит делать медленные шаги без перерывов между ними.

Б. Джонсон

Содержание

- Понятие транзакции
- Управление транзакциями
  - Фиксация транзакции
  - Откат транзакции
  - Точки сохранения транзакции
  - READ-ONLY транзакции
  - Вложенные транзакции
  - Автономные транзакции

Транзакция

- *Транзакция* – набор операций над базой данных, переводящих ее из одного согласованного состояния в другое.
- В ходе выполнения транзакции согласованность базы данных может временно нарушаться.
- Результатом транзакции может быть *фиксация (COMMIT)* или *откат (ROLLBACK)* **всех** входящих в нее операций.
- *АСИД-транзакция* – транзакция со свойствами:
  - *Атомарность* – выполняются все операторы транзакции или ни один
  - *Согласованность* – перевод базы данных из одного согласованного состояния в другое
  - *Изолированность* – параллельные транзакции не могут повлиять друг на друга
  - *Долговечность* – изменения, произведенные зафиксированной транзакцией, не могут быть потеряны ни при каких обстоятельствах

Начало и завершение транзакции

- Транзакция начинается *неявно* первым выполняемым оператором SQL.
- Транзакция завершается
  - *явно*
    - *фиксируется* оператором COMMIT
    - *откатывается* оператором ROLLBACK (без фразы TO SAVEPOINT)
  - *неявно*
    - *фиксируется* оператором модификации объекта схемы (CREATE/DROP/RENAME/ALTER) или в случае завершения сессии пользователем
    - *откатывается* в случае аварийного завершения по ошибке пользовательского процесса

Корпоративные системы баз данных © М.Л. Цымблер 4

Выполнение оператора vs фиксация транзакции

- Успешное выполнение оператора транзакции не гарантирует сохранение результатов его выполнения в случае отката всей транзакции.

```
SQL>
create table emp (id number primary key, name char(20), age number);
insert into emp values (1, 'Иванов', 40);
insert into emp values (2, 'Петрова', 30);
insert into emp values (3, 'Сидоров', 50);
rollback;
select * from emp;

ID    NAME        AGE
--------------------
нет строк
```

Корпоративные системы баз данных © М.Л. Цымблер 5

Откат оператора vs откат транзакции

- Неуспешное выполнение одного оператора транзакции не означает откат всей транзакции.

```
SQL>
create table emp (id number primary key, name char(20), age number);
insert into emp values (1, 'Иванов', 40);
insert into emp values (2, 'Петрова', 30);
insert into emp values (2, 'Сидоров', 50);
ORA-00001: Дублирующееся значение первичного ключа
commit;
select * from emp;

ID    NAME        AGE
--------------------
1     Иванов      40
2     Петрова     30
```

Корпоративные системы баз данных © М.Л. Цымблер 6

Фиксация транзакции

- Действия *до* фиксации:
  - Генерация записей отката в буферах сегмента отката (старые значения данных, измененные транзакцией).
  - Генерация записей журнала в буферах журнала; эти записи могут попасть на диск до фиксации транзакции (выполняется процессом LGWR).
  - Внесение изменений данных в буфера базы данных; эти изменения могут попасть на диск до фиксации транзакции (выполняется процессом DBWR).
- Действия *после* фиксации:
  - Отметка во внутренней таблице транзакций для соответствующего сегмента отката, что транзакция фиксирована; назначение транзакции и запись в эту таблицу уникального номера системных изменений (SCN).
  - Копирование записей журнала и SCN транзакции из буферов журнала онлайновый файл журнала (выполняется процессом LGWR).
  - Освобождение всех наложенных блокировок.
  - Пометка транзакции как завершенной.

Корпоративные системы баз данных © М.Л. Цымблер 7

Фиксация транзакции: пример 1

```
begin
   ...
   update accounts set balance=my_bal-debit
   where acctno= 7715;
   ...
   update accounts set balance=my_bal+credit
   where acctno=7720;
   commit work; -- Изменения видимы другим пользователям
   …
end;
```

Корпоративные системы баз данных © М.Л. Цымблер 8

Фиксация транзакции: пример 2

```
procedure transfer_London_Paris(from_acct in number, to_acct in number,
amount in number) is
begin
   update accounts@London
   set balance=balance-amount
   where acctno=from_acct;
   update accounts@Paris
   set balance= balance+amount
   where acctno=to_acct;
   commit work comment ‘Распределенная транзакция с узла London.’;
exception
   when others then
      rollback;
   …
end;
```

Корпоративные системы баз данных © М.Л. Цымблер 9

Откат транзакции

- Все изменения всех операторов транзакции отменяются с помощью соответствующих сегментов отката.
- Все точки сохранения транзакции теряются.
- Все наложенные транзакцией блокировки освобождаются.
- Транзакция помечается как завершенная.

Откат транзакции: пример

```
declare
   emp_id number;
   ...
begin
   select empno into emp_id from new_emp where ... ;
   ...
   insert into emp values (emp_id, ...);
   insert into tax values (emp_id, ...);
   insert into pay values (emp_id, ...);
   ...
exception
   when dup_val_on_index then
      rollback;
   ...
end;
```


Точки сохранения транзакции

- *Точка сохранения (savepoint)* – текстовая метка внутри транзакции.
- Точки сохранения используются для разбиения длинной транзакции на небольшие части.
- *Откат до точки сохранения* позволяет откатить не всю транзакцию, а только изменения после указанной точки до текущей точки транзакции.

Откат до точки сохранения

- Откатываются только операторы транзакции, выполненные после точки сохранения.
- Указанная точка сохранения остается, но все точки сохранения, установленные после указанной, теряются.
- Блокировки данных, наложенные отменяемыми операторами транзакции, освобождаются; остальные блокировки удерживаются.

Использование точек сохранения: пример 1

```
SQL>
savepoint update_cur_data;
    update emp set …;
    update dept set …;
    update job set …;
savepoint delete_old_data;
    delete from emp …;
    delete from dept …;
    delete from job …;
savepoint insert_new_data;
    insert into emp values (…);
    insert into dept values (…);
    insert into job values (…);
rollback to delete_old_data;
    delete from job …;
savepoint insert_new_data;
    insert into emp values (…);
    insert into dept values (…);
    insert into job values (…);
commit;
```


Использование точек сохранения: пример 2

```
procedure modify_emp(emp_id in number, …) is
    …
begin
    ...
savepoint rollback_point;
    update emp set ...
    where empno = emp_id;
    ...
savepoint rollback_point;
    insert into emp
    values (emp_id, ...);
exception
    when others then
        rollback to rollback_point;
end;
```

Использование точек сохранения: пример 3

```
declare
   name char(20);
   ans1 char(3);
   ans2 char(3);
   ans3 char(3);
   suffix number := 1;
begin
   ...
   loop -- или, например, 10 попыток обработки: for i in 1..10 loop
      begin
         savepoint start_transaction;
         delete from results where answer1 = 'NO';
         insert into results values (name, ans1, ans2, ans3); -- возможно исключение
dup_val_on_index
         commit;
         exit;
      exception
         when dup_val_on_index then
            rollback to start_transaction; -- отмена изменений
            suffix := suffix + 1; -- пытаемся решить проблему дублирования ключа
            name := name || to_char(suffix);
      end;
   end loop;
end;
```


READ-ONLY транзакции

- Транзакция может быть запущена в одном из двух режимов: READ-WRITE (по умолчанию) и READ-ONLY.
- В *режиме READ-WRITE* транзакция
  - может модифицировать объекты базы данных
  - видит изменения, вносимые в базу данных другими транзакциями, – после фиксации этих транзакций.
- В *режиме READ-ONLY* транзакция
  - не может модифицировать объекты базы данных
  - не видит изменений, вносимых в базу данных другими транзакциями.

Пример транзакции READ-ONLY

```
declare
   daily_sales real;
   weekly_sales real;
   monthly_sales real;
begin
   ...
   commit; -- Завершение предыдущей транзакции
   set transaction read only;
   select sum(amount) into daily_sales from sales
   where sale_date = sysdate;
   select sum(amount) into weekly_sales from sales
   where sale_date > sysdate - 7;
   select sum(amount) into monthly_sales from sales
   where sale_date > sysdate - 30;
   commit; -- Завершение транзакции read-only
   set transaction read write; -- Начало транзакции read-write
   ...
end;
```

Вложенные транзакции

- СУБД Oracle поддерживает *Д-ослабленные транзакции* – АСИД-транзакции без свойства *Д*олговечности.
- Д-ослабленная транзакция используется как *вложенная* (в *главную* транзакцию):
  - вложенная транзакция разделяет все ресурсы главной транзакции (блокировки и др.)
  - если главная транзакция откатывается, то изменения, внесенные вложенной транзакцией, тоже откатываются
  - изменения, внесенные вложенной транзакцией, становятся видимыми другим транзакциям после фиксации главной транзакции

Пример: вложенные транзакции

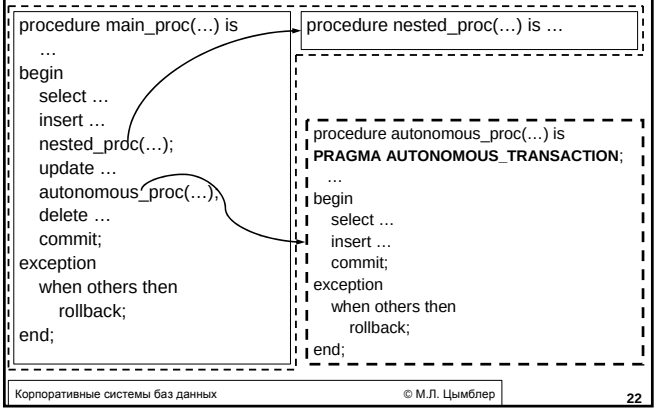
```
procedure main_proc(…) is
   …
begin
   select …
   insert …
   nested_proc(…);
   update …
   delete …
   commit;
exception
   when others then
      rollback;
end;
```

```
procedure nested_proc(…) is
   …
begin
   select …
   insert …
   commit;
exception
   when others then
      rollback;
end;
```


Автономные транзакции

- СУБД Oracle поддерживает *расширенные транзакции* – транзакции, представляющие собой иерархию АСИД-транзакций; в документации обозначаются термином *автономные транзакции*.
- Транзакция, помеченная как автономная
  - не разделяет ресурсы главной транзакции (блокировки и др.)
  - не зависит от главной транзакции (если главная транзакция откатывается, то изменения, внесенные автономной транзакцией, не откатываются)
  - делает видимыми внесенные изменения для других транзакций немедленно после своей фиксации

Автономная транзакция: пример 1



Корпоративные системы баз данных © М.Л. Цымблер 22

Автономная транзакция: пример 2

```
create table p( -- Детали
    p# number primary key, pname char(20), price number(4,2), weight number(4,2));
create table p_log (p# number, pname char(20)); -- Журнал деталей
create trigger parts_trig -- Автономный триггер
insert into p values (1, 'болт',  2.50, 0.05); commit;
insert into p values (2, 'гайка',  1.50, 0.03); rollback;
select * from parts order by p#;
P#       PNAME    PRICE    WEIGHT
-------------------------------
1        болт     2.50     0.05
select * from p_log order by p#;
P#       PNAME
--------------
1        болт
2        гайка
:new.pname);
```

Корпоративные системы баз данных © М.Л. Цымблер 23