УО «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Лабораторная работа №3

РАСПРЕДЕЛЕННЫЕ ТРАНЗАКЦИИ В ORACLE

Выполнили студенты

3 курса 4 группы факультета ИТ:

Миневич К.В., Стальмахова Н.А.

Проверила: преподаватель-стажер Комарова Е.И.

Минск 2023

Цель работы: смоделировать распределенную транзакцию, которая блокируется и будет ожидать освобождение ресурса на удаленном сервере; смоделировать распределенную транзакцию, у которой нарушается ограничение целостности на стороне уделенного сервера.

Листинг кода:

*--user: system*CREATE PROFILE CUSTOMER\_PROFILE LIMIT PASSWORD\_LIFE\_TIME 180 SESSIONS\_PER\_USER 20 FAILED\_LOGIN\_ATTEMPTS 7 PASSWORD\_LOCK\_TIME 1 PASSWORD\_REUSE\_TIME 10 PASSWORD\_GRACE\_TIME DEFAULT CONNECT\_TIME 180 IDLE\_TIME 30;  
  
DROP PROFILE CUSTOMER\_PROFILE CASCADE;  
DROP USER CUSTOMER CASCADE;  
  
CREATE USER customer IDENTIFIED BY qwerty  
 DEFAULT TABLESPACE USERS  
 PROFILE CUSTOMER\_PROFILE  
 QUOTA UNLIMITED ON USERS;  
  
grant connect to customer;  
DROP ROLE CUSTOMER\_ROLE;  
  
CREATE ROLE CUSTOMER\_ROLE;  
Alter session set container = cdb$root;  
  
commit;  
  
Grant connect, resource to customer;  
grant customer\_role to customer;  
grant customer\_role to customer\_2;  
  
*-- DBLINK по схеме USER1 - USER2  
--1*CREATE DATABASE LINK anotherdb  
 CONNECT TO customer  
 IDENTIFIED BY qwerty  
 USING 'NAT-PC:1521/XE';  
  
DROP DATABASE LINK anotherdb;  
*--grant create database link to customer;*GRANT CREATE DATABASE LINK TO CUSTOMER;  
  
*-- user: CUSTOMER*create table MKV (  
 MKV\_id number(10) generated as identity(start with 1 increment by 1),  
 MKV\_name varchar2(30) not null,  
 MKV\_number number(3) not null,  
 CONSTRAINT MKV\_pk PRIMARY KEY (MKV\_id)  
);  
commit;  
insert into MKV (MKV\_name, MKV\_number) values ('Kristina', 20);  
  
select *\** from MKV;  
drop table MKV;  
  
*-- user: system*SELECT *\** FROM SNA@anotherdb; *-- подключение и просмотр информации в таблице SNA сервера удаленной базе данных  
  
  
--2*SELECT *\** FROM SNA@anotherdb;  
INSERT INTO SNA@anotherdb(MKV\_name, MKV\_number) values('NewRow', 20);  
UPDATE SNA@anotherdb SET MKV\_NUMBER = 13 WHERE MKV\_NAME = 'NewRow';  
DELETE SNA@anotherdb where MKV\_NAME = 'NewRow';  
commit;  
  
  
*--3*BEGIN  
 DELETE SNA@anotherdb;  
 UPDATE SNA@anotherdb set MKV\_NAME ='New' where MKV\_NUMBER=1;  
END;  
  
BEGIN  
 SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  
 UPDATE SNA@anotherdb set MKV\_NAME ='New' where MKV\_NUMBER=1;  
 SYS.DBMS\_LOCK.*SLEEP*(60);  
 COMMIT;  
END;

1. На обоих серверах создаем пользователей в рамках PDBORCL.

CREATE USER customer IDENTIFIED BY qwerty  
 DEFAULT TABLESPACE USERS  
 PROFILE CUSTOMER\_PROFILE  
 QUOTA UNLIMITED ON USERS;

1. Создаем две таблицы на разных серверах в схемах созданных пользователей.

*-- user: CUSTOMER*create table MKV (  
 MKV\_id number(10) generated as identity(start with 1 increment by 1),  
 MKV\_name varchar2(30) not null,  
 MKV\_number number(3) not null,  
 CONSTRAINT MKV\_pk PRIMARY KEY (MKV\_id)  
);  
commit;  
insert into MKV (MKV\_name, MKV\_number) values ('Kristina', 20);  
  
select *\** from MKV;  
drop table MKV;

1. Устанавливаем DBLINK типа USER1-USER1 между пользователями, расположенными на двух серверах.

CREATE DATABASE LINK anotherdb  
 CONNECT TO customer  
 IDENTIFIED BY qwerty  
 USING 'NAT-PC:1521/XE';  
  
DROP DATABASE LINK anotherdb;  
*--grant create database link to customer;*GRANT CREATE DATABASE LINK TO CUSTOMER;

1. Разрабатываем распределенную транзакцию, по условиям которой выполнение скрипта вторым сервером невозможно, пока задачу не закончит первый. При выполнении данной транзакции происходит ожидание, продемонстрированное на рисунке 2.

BEGIN  
 DELETE SNA@anotherdb;  
 UPDATE SNA@anotherdb set MKV\_NAME ='New' where MKV\_NUMBER=1;  
END;  
  
BEGIN  
 SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  
 UPDATE SNA@anotherdb set MKV\_NAME ='New' where MKV\_NUMBER=1;  
 SYS.DBMS\_LOCK.*SLEEP*(60);  
 COMMIT;  
END;

**Вывод:** в ходе лабораторной работы мы смоделировали распределенную транзакцию, которая блокируется и будет ожидать освобождение ресурса на удаленном сервере, смоделировали распределенную транзакцию, у которой нарушается ограничение целостности на стороне уделенного сервера. По итогу можно сказать, что распределенная транзакция требует стабильной работы и подключения обоих серверов и эффективна в случаях, когда несколько пользователей должны взаимодействовать с одним и тем же ресурсом.