كلية الحاسبات و الذكاء الصناعي

Database Systems – 2

Team Members

Name	ID
أحمد محمد متولي بيومي	20210108
أحمد عصام حمدي	20210078
أحمد رؤوف علي	20210054
سلمى أشرف ابراهيم أمام	20210408
سلمى طه محمد السيد	20210414
أحمد عمر حسين محمد	20210085
أحمد سيد فراج	20210059

a) **Create a Manager User and grant them a role of privileges to create two users. Let User 1 create the Employee and the Department table. Let User 2 insert 5 rows of employees. [2 Marks]**

!-Connected as SYSDBA:

--Creating the manager user

```
create user manager identified by 123;
```

--Creating the manager role and granting the necessary privileges

```
create role mngRole;

grant create session, create user,create table, create procedure, unlimited tablespace to
mngRole with admin option;
```

--Granting the manager user the mngRole role

```
grant mngRole to manager;
```

!-Connected as MANAGER:

--Creating the necessary users and granting them the necessary privileges

```
create user user1 identified by 123;

grant create session,create table, unlimited tablespace to user1;

create user user2 identified by 123;

grant create session,unlimited tablespace to user2;

grant create procedure to user1,user2;
```

!-Connected as USER1:

--Creating the tables

```
create table department(
id int primary key,
name varchar(50)
);
create table employee(
id int primary key,
name varchar(50),
salary int,
depID int,
CONSTRAINT departmentID_fk foreign key (depID) references department(id)
);
```

--Inserting department information

```
insert into user1.department values(1,'HR');
insert into user1.department values(2,'IT');
insert into user1.department values(3,'Finance');
```

--Granting USER2 the necessary privileges

```
grant insert,delete,update,select on department to user2;
grant insert,delete,update,select on employee to user2;
```

!-Connected as USER2:

--Inserting 5 rows into employee table

```
insert into user1.employee values(1,'Ahmed',10000,1);
insert into user1.employee values(2,'Mohamed',8000,2);
insert into user1.employee values(3,'Moaz',5000,3);
insert into user1.employee values(4,'Salma',10,1);
insert into user1.employee values(5,'Gasser',6700,3);
```

b) Demonstrate generating a blocker-waiting situation using two transactions by user 1 and user 2. The Transaction is calling a function that raises the rate of salary by 10% for department 1. [2 Marks]

!-Connected as USER1:

--Creating the necessary function

```
CREATE OR REPLACE FUNCTION updatedSal(deptID  number)
return number as
begin
IF (deptID = 1)
THEN
update employee set salary = salary + salary *.1
where depID = deptID;
return SQLCODE;
end IF;
end updatedSal;
/
```

--Granting USER2 the privilege to execute the function

```
grant execute on updatedSal to user2;
```

!-Connected as USER1 and USER2:

--Executing the function

```
begin
dbms_output.put_line(updatedSal(1));
end;
/
```

/-This results in USER1 having a lock over the table (blocker), causing USER2 to wait (waiting) for USER1 to free the table either by committing and saving the data or by rolling back.

c) Identify the sessions in the situation using SID and serial# for both blocker and waiting sessions. [2 Marks]

```
select
blocker.sid as blockerSID,
blocker.serial# as blockerSerial,
waiter.sid as waiterSID,
waiter.serial# as waiter_serial
from
v$session blocker
join v$session waiter on blocker.sid = waiter.blocking_session
where waiter.blocking_session is not null;
```

d) Demonstrate a deadlock scenario and display the expected result. [2 Marks]

!-Connected as USER1 and USER2:

```
update user1.Employee set salary = salary + salary * 0.1 where user1.Employee.ID = 1;

update user1.Employee set salary = salary + salary * 0.1 where user1.Employee.ID = 2;
```

/-In this scenario, USER1 is trying to update the salary of Employee whose ID is 1, causing a lock over the resources, USER2 at the same time is trying to update the salary of Employee whose ID is 2, causing another lock, when USER1 attempts to update the salary of Employee whose ID is 2, it's forced to wait until USER2 frees it up, same goes for USER2 when trying to update the salary of Employee whose ID is 1, which causes a deadlock that is later automatically resolved by Oracle SQL Developer

e) Perform the following functions [2 Marks]

i. Create a function that calculates the average salary for any department

ii. Create a function that calculates the Total Salary in a Department.

iii. Create a function that calculates the maximum Salary

--Function that calculates the average Salary of any department

```
CREATE OR REPLACE FUNCTION avgSalary(deptID number)
return number as avgSal number;
begin
select avg(salary) into avgSal from user1.employee where depID = deptID;
return avgSal;
end avgSalary;
/
```

--Executing it

```
select distinct(avgSalary(2)) from employee;
```

--Function that calculates the total salary in a Department

```
CREATE OR REPLACE FUNCTION sumSalary(deptID number)
return number as sumSal number;
begin
select sum(salary) into sumSal from user1.employee where depID = deptID;
return sumSal;
end sumSalary;
/
```

--Executing it

```
select distinct(sumSalary(1)) from employee;
```

--**A function that calculates max salary**

```
CREATE OR REPLACE FUNCTION maxSalary
return number as maxSal number;
begin
select max(salary) into maxSal from user1.employee;
return maxSal;
end maxSalary;
/
```

--Executing it

```
select distinct(maxSalary) from employee;
```

--Granting all users execution of all functions

```
grant execute on avgSalary to user1;
grant execute on sumSalary to user1;
grant execute on maxSalary to user1;
grant execute on avgSalary to user2;
grant execute on sumSalary to user2;
grant execute on maxSalary to user2;
```