

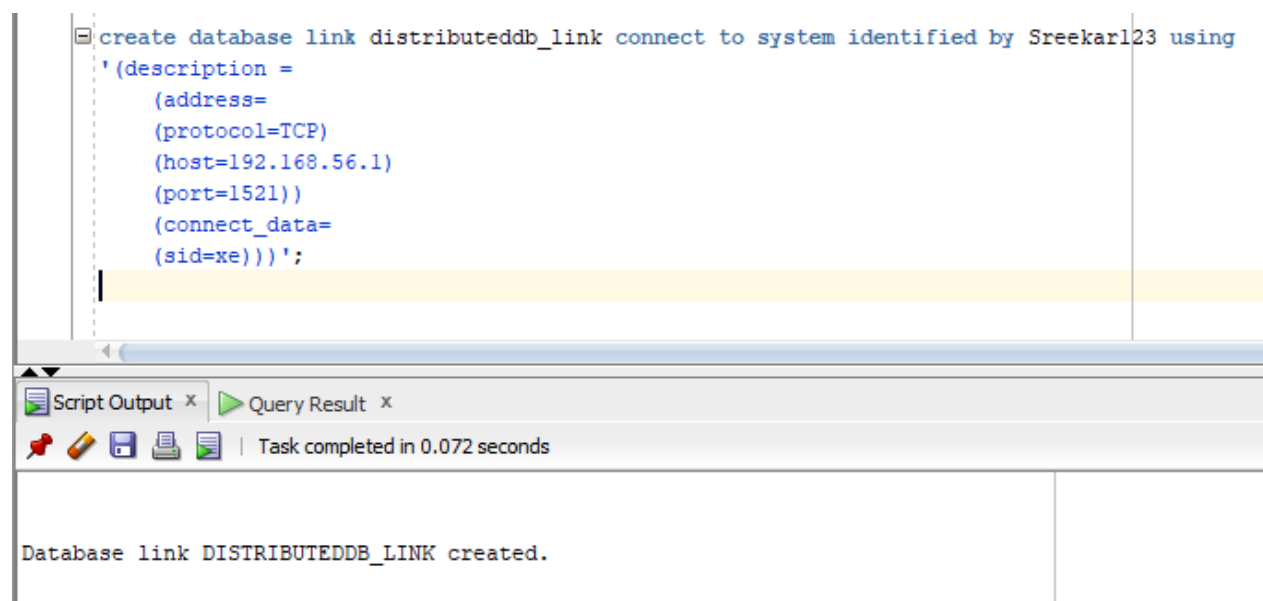
# Distributed And Parallel Database Systems

## Project Part – 3

### Group – 12

#### Site -1

Creating the Global database link “distributeddb\_link” to connect with my password and I have given IP Address to for network sync and port address as 1521 in Site 1.



```
create database link distributeddb_link connect to system identified by Sreekar123 using
'(description =
  (address=
    (protocol=TCP)
    (host=192.168.56.1)
    (port=1521))
  (connect_data=
    (sid=x)) )';
```

Script Output x Query Result x

Task completed in 0.072 seconds

Database link DISTRIBUTEDDB\_LINK created.

In the above screenshot, I have created a Globally distributed Database design system using the link without replication such that others could use my global database.

I am the owner of Site 1 as well;

I Fragmented the required tables into the respective sites to fetch the desired outputs.

Site 1)

1. List employee names for those with a salary greater than 30000;

```
create table empl as (select * from employees@distributeddb link where employees.salary@distributeddb_link>30000);
```

Script Output x Query Result x  
Task completed in 0.021 seconds

Table EMP1 created.

```
select name as employees, salary from empl;
```

Script Output x Query Result x  
All Rows Fetched: 3 in 0.004 seconds

	EMPLOYEES	SALARY
1	Harry	38000
2	Gary	34000
3	Kim	35000

For Question 1, I have fragmented the global employees table in such a way that all the employees with salaries greater than 30,000 are present in site1 and saved in table empl.

I have accessed their names using a select statement and displayed the employees names as employees and salary.

Hence, Employees Harry, Gary and Kim has salary greater than 30000.

2. List library users who only reserved fiction books.

```
create table fiction_users as ( SELECT u_id, u_name FROM users@distributeddb link WHERE u_id IN (
    SELECT DISTINCT u_id
    FROM book_transactions@distributeddb link bt
    JOIN books@distributeddb link b ON bt.b_id = b.b_id
    WHERE b.genre = 'Fiction')
AND u_id NOT IN (
    SELECT DISTINCT u_id
    FROM book_transactions@distributeddb link bt
    JOIN books@distributeddb link b ON bt.b_id = b.b_id
    WHERE b.genre != 'Fiction'
));
```

Script Output x  
Task completed in 0.042 seconds

Table FICTION\_USERS created.

```
select * from fiction_users;
```

U_ID	U_NAME
1	114 Jacob

I have fragmented the global database in such a way that list of users who took books of genre Fiction from the library into a table “Fiction\_Users”.Have accessed the u\_id and u\_name using select command.

I used subqueries with joins for this operation. The first subject gives out the list u\_id and u\_name of users who took books of genre fiction. The second subquery gives out the list of u\_id of users who took books of genre other than Fiction.

As there are some users who took books of genre Fiction and Mystery, to avoid that I have written query in such a way to give the user details who took Books of fiction genre only.

3. Show library users who have returned books late just one time.

```
create table late_submission as (SELECT u.u_id,u.u_name,count(*) as late_submission
FROM users@distributeddb link u
JOIN book_transactions@distributeddb link bt ON u.u_id = bt.u_id
WHERE bt.returned_date > bt.expected_return_date
GROUP BY u.u_id, u.u_name
HAVING COUNT(*) = 1);
```

Script Output x Query Result x

Task completed in 0.385 seconds

Table LATE\_SUBMISSION created.

>>Query Run In:Query Result

```
select * from late_submission;
```

U_ID	U_NAME	LATE_SUBMISSION
1	113 Lilly	1
2	114 Jacob	1
3	116 Manu	1

I have fragmented the data of users who have returned books lately for just one time into table late\_submission in the global database which is my site 1. And so, I displayed those details using select command. I used groupby and having function in the query.

Selected the list of users by joining the tables users and book transactions on user id and checked the condition where book returned data is more than expected returned date and grouped that data on the user id and username using group by and to find the user who was late exactly 1 time is found using the having. The result is then viewed in the query result tab by writing a select query