**MSCD 600 Database Architecture**

**Lab 2**

The purpose of this lab is to work with tables and learn more about the SQL\*Plus environment.

Part 1

To show that you have successfully executed each step, you can either cut and paste the output on your screen and paste it into this Word document, or you can capture screen shots and paste them into this document. For information on how you will be evaluated, refer to the Labs Rubric on the last page of this document below.

Login to the database with your username before starting. To login, refer to the [Course Resources\Student\_Instructions\_Static\_Database\_Xen\_Servers.pdf](Course%20Resources/Student_Instructions_Static_Database_Xen_Servers.pdf) which is also located in the Course Resources folder for easy access throughout this course.

**Note:** If you get part way through the lab and want to start over from scratch, run the following, or if you want to make sure these objects are dropped before you start then run the following:

drop table emp purge;

drop table dept purge;

drop sequence app\_seq;

drop table inv purge;

drop table users\_report\_table purge;

It’s okay if you see an error saying that the object doesn’t exist.

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

In SQL\*Plus, -- starts a text comment on a line.

In SQL\*Plus, REM starts a text comment on a line.

In SQL\*Plus, /\* Anything in here is a comment... \*/

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

# Step 1: Verify you’re logged in as you and create two tables:

show user;

create table dept(dept\_id number,

dept\_name varchar2(20));

create table emp(emp\_id number,

emp\_name varchar2(20),

dept\_id number);

desc dept;

desc emp;

# Step 2: Alter the tables to have primary keys and unique keys:

alter table dept add constraint dept\_pk primary key(dept\_id);

alter table emp add constraint emp\_pk primary key(emp\_id);

alter table dept add constraint dept\_uk1 unique(dept\_name);

alter table emp add constraint emp\_uk1 unique(emp\_name);

select table\_name, constraint\_name, constraint\_type from user\_constraints;

# Step 3: Alter emp to have a FK constraint:

alter table emp add constraint emp\_fk1

foreign key (dept\_id) references dept(dept\_id);

select table\_name, constraint\_name, constraint\_type from user\_constraints;

# Step 4: Insert some data into the tables (you should see some errors, why?):

insert into dept (dept\_id, dept\_name) values (1, 'HR');

insert into dept (dept\_id, dept\_name) values (1, 'HR');

insert into dept (dept\_id, dept\_name) values (2, 'IT');

insert into dept (dept\_id, dept\_name) values (3, 'FINANCE');

insert into dept (dept\_id, dept\_name) values (4, 'ACCOUNTING');

insert into dept (dept\_id, dept\_name) values (5, 'SHIPPING');

insert into dept (dept\_id, dept\_name) values (6, 'WAREHOUSE');

insert into dept (dept\_id, dept\_name) values (7, 'EDUCATION');

insert into dept (dept\_id, dept\_name) values (8, 'EDUCATIONINTHEUNVERSITY');

insert into emp (emp\_id, emp\_name, dept\_id) values (100, 'Orton', 1);

insert into emp (emp\_id, emp\_name, dept\_id) values (200, 'DJ', 2);

insert into emp (emp\_id, emp\_name, dept\_id) values (300, 'Amy', 3);

insert into emp (emp\_id, emp\_name, dept\_id) values (400, 'Harrison', 4);

insert into emp (emp\_id, emp\_name, dept\_id) values (500, 'Vineetha', 5);

insert into emp (emp\_id, emp\_name, dept\_id) values (600, 'Joshes', 3);

insert into emp (emp\_id, emp\_name, dept\_id) values (700, 'Elaine', 3);

insert into emp (emp\_id, emp\_name, dept\_id) values (700, 'Elaine', 3);

insert into emp (emp\_id, emp\_name, dept\_id) values (800, 'Darl', 7);

-- Not assigned a departpment yet

insert into emp (emp\_id, emp\_name) values (900, 'Bob');

insert into emp (emp\_id, emp\_name) values (1000, 'Glenn');

insert into emp (emp\_id, emp\_name) values (1100, 'Glenn');

select \* from dept;

select \* from emp;

commit;

# Step 5: Add a column to each table and show details:

alter table emp add(create\_dtt timestamp);

alter table dept add(create\_dtt timestamp);

desc emp;

desc dept;

update emp set create\_dtt = sysdate;

update dept set create\_dtt = sysdate;

column create\_dtt format a15

set pagesize 50

select \* from emp;

select dept\_name, trunc(create\_dtt) from dept;

commit;

# Step 6: Join the two tables to report on who is in what dept:

-- Commonly used style

select d.dept\_name, e.emp\_name

from dept d,

emp e

where d.dept\_id = e.dept\_id;

-- ANSI standard style

select d.dept\_name, e.emp\_name

from dept d join emp e on(d.dept\_id = e.dept\_id);

-- ANSI using natural join clause

select dept\_name, emp\_name

from dept natural join emp;

# Step 7: Drop the tables:

drop table dept;

-- What happened?

drop table emp;

drop table dept;

# Step 8: Recreate the tables:

create table dept(dept\_id number,

dept\_name varchar2(20) not null,

create\_dtt timestamp default sysdate);

create table emp(emp\_id number,

emp\_name varchar2(20) not null,

dept\_id number,

create\_dtt timestamp default sysdate

);

desc dept;

desc emp;

# Step 9: Alter the tables to have primary keys:

alter table dept add constraint dept\_pk primary key(dept\_id);

alter table emp add constraint emp\_pk primary key(emp\_id);

select table\_name, constraint\_name, constraint\_type from user\_constraints;

-- What’s the BIN stuff?

purge recyclebin;

select table\_name, constraint\_name, constraint\_type from user\_constraints;

# Step 10: Alter emp to have a FK constraint:

alter table emp add constraint emp\_fk1

foreign key (dept\_id) references dept(dept\_id);

select table\_name, constraint\_name, constraint\_type from user\_constraints;

# Step 11: Create a sequence:

create sequence app\_seq;

# Step 12: Insert some data into the tables using the sequence:

insert into dept (dept\_id, dept\_name) values (app\_seq.nextval, 'HR');

insert into dept (dept\_id, dept\_name) values (app\_seq.nextval, 'IT');

insert into dept (dept\_id, dept\_name) values (app\_seq.nextval, 'FINANCE');

insert into dept (dept\_id, dept\_name) values (app\_seq.nextval, 'ACCOUNTING');

insert into dept (dept\_id, dept\_name) values (app\_seq.nextval, 'SHIPPING');

insert into dept (dept\_id, dept\_name) values (app\_seq.nextval, 'WAREHOUSE');

insert into dept (dept\_id, dept\_name) values (app\_seq.nextval, 'EDUCATION');

insert into emp (emp\_id, emp\_name, dept\_id) values (app\_seq.nextval, 'Orton', 1);

insert into emp (emp\_id, emp\_name, dept\_id) values (app\_seq.nextval, 'DJ', 2);

insert into emp (emp\_id, emp\_name, dept\_id) values (app\_seq.nextval, 'Amy', 3);

insert into emp (emp\_id, emp\_name, dept\_id) values (app\_seq.nextval, 'Harrison', 4);

insert into emp (emp\_id, emp\_name, dept\_id) values (app\_seq.nextval, 'Vineetha', 5);

insert into emp (emp\_id, emp\_name, dept\_id) values (app\_seq.nextval, 'Joshes', 3);

insert into emp (emp\_id, emp\_name, dept\_id) values (app\_seq.nextval, 'Elaine', 3);

insert into emp (emp\_id, emp\_name, dept\_id) values (app\_seq.nextval, 'Darl', 7);

select dept\_id, dept\_name, trunc(create\_dtt) from dept;

select emp\_id, emp\_name, trunc(create\_dtt) from emp;

select dept\_id, emp\_name from emp where dept\_id = 3;

select emp\_id, emp\_name from emp where emp\_id < 15;

# Step 13: Select the current value of the sequence:

select app\_seq.currval from dual;

select app\_seq.nextval from dual;

select app\_seq.currval from dual;

# Step 14: Create a view:

create or replace view emp\_dept\_view

as

select e.emp\_name, d.dept\_name

from emp e, dept d

where e.dept\_id = d.dept\_id;

desc emp\_dept\_view;

select \* from emp\_dept\_view;

# Step 15: Create a b-tree index:

create index emp\_name\_idx on emp(emp\_name);

# Step 16: Set your SQL prompt:

SET SQLPROMPT '&\_USER.@&\_CONNECT\_IDENTIFIER.> '

# Step 17: Create a table with a virtual (derived) column:

create table inv(

inv\_id number primary key

,inv\_count number

,inv\_status generated always as(

case when inv\_count <= 100 then 'GETTING LOW!'

when inv\_count > 100 then 'OKAY, RELAX'

end)

);

set lines 80

desc inv;

# Step 18: Insert some data into inv:

insert into inv (inv\_id, inv\_count) values(1, 200);

insert into inv (inv\_id, inv\_count) values(2, 50);

insert into inv (inv\_id, inv\_count) values(3, 300);

insert into inv (inv\_id, inv\_count) values(4, 1);

insert into inv (inv\_id, inv\_count) values(5, 5000);

insert into inv (inv\_id, inv\_count) values(6, 250);

select \* from inv;

# Step 19: Create a table using a select statement:

create table users\_report\_table as select username from all\_users;

select count(\*) from users\_report\_table;

select \* from users\_report\_table where rownum < 10;

# Step 20: Create a data dictionary comments on a table:

comment on table inv is 'This is the inventory table.';

column table\_name format A15

column comments format A30

column column\_name format A15

select table\_name, comments from user\_tab\_comments;

comment on column inv.inv\_id is 'This is the primary key.';

comment on column inv.inv\_count is 'This is the inventory count.';

comment on column inv.inv\_status is 'This is a derived column.';

select table\_name, column\_name, comments from user\_col\_comments;

# Step 21: How many users are connected to the database:

select count(\*), machine from v$session group by machine;

# Step 22: Report on your objects:

col object\_name form a20

col object\_type form a15

col table\_name form a20

col tablespace\_name form a20

col index\_name form a20

set linesize 132

select object\_name, object\_type from user\_objects;

select table\_name, tablespace\_name from user\_tables;

select index\_name, tablespace\_name from user\_indexes;

# Step 23: Report on the sizes of objects:

set linesize 80

col segment\_name form a20 heading "Segment Name"

col segment\_type form a20 heading "Segment Type"

col bytes form 999,999,999 heading "Bytes Consumed"

select segment\_name, segment\_type, bytes from user\_segments;

Part 2

Before you begin, startup SQL Data Modeler from Citrix (or if you have a local copy use that).

# Step 1: Create a DEPT table

\* Right click on Logical Model in the left pane and select “Show”

\* Click on the “New Entity” icon on the toolbar

\* Draw a square in the Relational diagram area

\* Name the table DEPT (Note: Table names should always be capitalized and they should never be plural)

\* Click on “Attributes”

\* Click on the green plus symbol to add a new attribute

\* Name the attribute DEPT\_ID

\* Make its datatype logical

\* Make its type numeric

\* Select the Primary UID box to make this the primary key

\* Click on the green plus symbol to add a new attribute

\* Name the attribute DEPT\_NAME

\* Make its datatype logical

\* Make its type varchar

\* Click OK

# Step 2: Create an EMP table

\* Click on the “New Entity” icon on the toolbar

\* Draw a square in the Relational diagram area

\* Name the table EMP

\* Click on “Attributes”

\* Click on the green plus symbol to add a new attribute

\* Name the attribute EMP\_ID

\* Make its datatype logical

\* Make its type numeric

\* Select the Primary UID box to make this the primary key

\* Click on the green plus symbol to add a new attribute

\* Name the attribute EMP\_NAME

\* Make its datatype logical

\* Make its type varchar

\* Click OK

# Step 3: Create the one-to-many relationship between the two tables

\* Click on the green arrow icon in the toolbar called “New 1:N Relation”

\* Click on the DEPT table (the “one” side of the relationship)

\* Then, click on the EMP table (the “may” side of the relationship)

\* A dialog box will appear and you may select OK

# Step 4: Create the Relational Model so that you can generate DDL

\* Click on the “Engineer to Relational Model” icon on the toolbar (looks like two greater than signs together)

\* When a dialog box appears click on “Engineer”

\* Notice that you now see a Relational Model window

\* Also notice that the relational model shows the primary key of the DEPT table (the “one” side) as now also being a foreign key value in the EMP table (the “many” side). This is depicted by the letter F next to the newly added attribute.

# Step 5: Generate DDL statement

\* Click on the “Generate DDL” icon on the toolbar (looks like a small datastore object on the top right)

\* Click on Generate

\* Accept defaults and click OK

# Step 6: Save your model and print output below

\* Click the Save button on the DDL File Editor dialog box

\* Save the file to the directory called C:\Users\<your\_name>

\* Copy and paste the DDL output into your lab document for submission

l

**MSCD 600 Database Architecture**

**Labs Rubric**

|  |  |
| --- | --- |
| **Grade** | **Rubric: Labs** |
| A  90-100% | -The lab work is technically sound  -The lab work addresses the criteria and exceeds the minimum requirements for the lab assignment.  -Met submission deadline.  - The deliverable format meets all requirements.  -Each item is clearly delineated and easy to follow.  -The submitted work is free from errors. |
| B  80-89% | -The lab work is technically sound  -The lab work addresses the criteria and meets the minimum requirements for the lab assignment.  -Met submission deadline.  - The deliverable format meets all requirements.  -Most items are clearly delineated and easy to follow.  -The submitted work has one to two errors. |
| C  70-79% | -The lab work is technically sound  -The lab work addresses the criteria and meets the minimum requirements for the lab assignment.  -Met submission deadline.  - The deliverable format meets most requirements.  -Most items are clearly delineated and easy to follow.  -The submitted work has more than two errors. |
| D  60-69% | -The lab work is not technically sound  -The lab work addresses the criteria and meets some of the minimum requirements for the lab assignment.  -Met submission deadline.  - The deliverable format meets some requirements.  -Few items are clearly delineated and easy to follow.  -The submitted work has more than three errors. |
| F  Less than 60% | -The lab work is not technically sound  -The lab work does not meet the base criteria nor the minimum requirements for the lab assignment.  -Did not meet submission deadline.  -The deliverable format meets some requirements.  -Few items are clearly delineated and easy to follow.  -The submitted work has more than four errors. |