**COMP 3005  
Assignment #5   
Solution**

**Instruction**

1. This is an individual assignment. Copying is not allowed.
2. There are two parts in this assignment. Put your answers to each part in a single word document.
3. Do Part 2 using Oracle virtual machine. Each question is an embedded SQL or PL/SQL program. You need to test each program carefully and submit the final version of the program and the execution results obtained using the script command of the form script –a Part2Q*n*, which records everything on screen in the file called Part2Q*n* for question *n* until you type keys Ctrl and d together.
4. Submit the single word document on culearn.

**Part 1 (20 Marks)**

Given the following inference rules:

(R0) X → XX

(R1) if Y ⊆X , then X → Y

(R2) X → Y ⊨XZ → YZ

(R3) X → Y, Y → Z ⊨ X → Z

Prove or disprove the following inference rules for functional/multi-valued dependencies.

1. X → Y and Z ⊆ Y and W ⊆ Z ⊨X → W

Z ⊆ Y then Y → Z (R1)

W ⊆ Z then Z → W (R1)

X → Y, Y → Z, Z → W then X→ W (R3)

2. X → Y, Z → Y ⊨X → Z

|  |  |  |
| --- | --- | --- |
| X | Y | Z |
| x1 | y1 | z1 |
| x1 | y1 | z2 |

Relation satisfies X → Y, Z → Y, but not X → Z since (x1, z1) and (x1, z2)

3. X → Y, XY → Z ⊨ X → Z

X → XX (R0)

X → Y (given) then XX → XY (R2)

X → XX, XX → XY, XY → Z then X → Z (R3)

4. X →→ Y ⊨X → Y

|  |  |  |
| --- | --- | --- |
| Z | X | Y |
| z1 | x1 | y1 |
| z1 | x2 | y2 |
| z2 | x1 | y1 |
| z2 | x2 | y2 |

Relation satisfies X →→ Y, but not X → Y since (x1, y1) and (x1, y2)

5. X → Y, Z → W ⊨ XZ → YW

X → Y then XZ → YZ (R2)

Z → W then YZ → YW (R2)

XZ → YZ, YZ → YW then XZ → YW

**Part 2 (60 Marks)**

This part is based on the NHL Fans Database that has three tables shown below.

**Fan**

|  |  |  |  |
| --- | --- | --- | --- |
| **P#** | | **T#** | **Levels** |
| P1 | T1 | | 1000 |
| P1 | T2 | | 2000 |
| P1 | T3 | | 3000 |
| P1 | T4 | | 4000 |
| P2 | T1 | | 2000 |
| P2 | T2 | | 3000 |
| P2 | T3 | | 4000 |
| P3 | T1 | | 3000 |
| P3 | T2 | | 4000 |
| P4 | T1 | | 4000 |

**Person**

|  |  |  |  |
| --- | --- | --- | --- |
| **P#** | **Name** | **Age** | **City** |
| P1 | Smith | 20 | Ottawa |
| P2 | Jones | 30 | Toronto |
| P3 | Blake | 25 | Calgary |
| P4 | Clark | 20 | Montreal |
| P5 | Adams | 30 | Vancouver |

**Team**

|  |  |  |
| --- | --- | --- |
| **T#** | **Name** | **City** |
| T1 | Maple Leafs | Toronto |
| T2 | Canucks | Vancouver |
| T3 | Canadiens | Montreal |
| T4 | Flames | Calgary |

1. Write an Embedded SQL program that uses dynamic SQL method 1 to create the database. You can hard code all create table statements in the program using the execute immediate statement.

#include <stdio.h>

int main(){

exec sql include sqlca;

exec sql begin declare section;

char sqlstmt1[1024];

char sqlstmt2[1024];

char sqlstmt3[1024];

char \*LOGIN= "oracle/oracle11g";

exec sql end declare section;

exec sql whenever sqlerror goto error;

exec sql whenever not found goto finish;

exec sql connect :LOGIN;

if (sqlca.sqlcode == 0)

printf("Connected to ORACLE\n");

else

printf("Connection Failed\n");

printf("Creating tables:\n");

strcpy(sqlstmt1, "create table Person(P# varchar(4) NOT NULL, Name varchar(20) NOT NULL, Age int NOT NULL, City varchar(10) NOT NULL, primary key (P#), check (Age>0))" );

strcpy(sqlstmt2, "create table Team(T# varchar(4) NOT NULL, Name varchar(20) NOT NULL, City varchar(10) NOT NULL, primary key (T#))" );

strcpy(sqlstmt3, "create table Fan (P# varchar(4) NOT NULL, T# varchar(4) NOT NULL, Levels int NOT NULL, primary key (P#,T#), foreign key (P#) references Person(P#) ON DELETE CASCADE, foreign key (T#) references Team(T#) ON DELETE CASCADE)");

exec sql execute immediate :sqlstmt1;

exec sql execute immediate :sqlstmt2;

exec sql execute immediate :sqlstmt3;

printf("Finished creating tables\n");

finish:

printf("Creation all done\n");

exit(0);

error:

printf("Error found\n");

printf("Table might exist\n");

exec sql rollback release;

exit(1);

}

1. Write an Embedded SQL program that uses dynamic SQL method 2 to insert all tuples into the database. It should prompt the user to choose a table and then prompt the user with the attribute names and accept values for each tuple.

#include <stdio.h>

int main() {

exec sql include sqlca;

exec sql begin declare section;

char pnum[5];

char pname[20];

int age;

char pcity[20];

char tnum[5];

char tname[10];

char tcity[10];

int levels;

char sqlstmt[1024];

char pick[2];

char \*LOGIN= "oracle/oracle11g";

exec sql end declare section;

exec sql whenever sqlerror goto error;

exec sql whenever not found goto finish;

exec sql connect :LOGIN;

if (sqlca.sqlcode == 0)

printf("Connected to ORACLE\n");

else

printf("Connection Failed\n");

printf("There are 3 tables in this db: \n");

printf("1. Person \n");

printf("2. Team \n");

printf("3. Fan \n");

printf("\n\n");

printf("Please enter number 1, 2 or 3 to select a table or enter q to quit. \n");

scanf("%s",&pick);

while(strcmp(pick, "q") != 0){

if(strcmp(pick, "1") == 0){

while(strcmp(pick, "q") != 0) {

printf("P#:\n");

scanf("%s", &pnum);

printf("Name:\n");

scanf("%s", &pname);

printf("Age:\n");

scanf("%d", &age);

printf("City:\n");

scanf("%s", &pcity);

strcpy(sqlstmt, "insert into Person values (:v1, :v2, :v3, :v4)");

exec sql prepare s from :sqlstmt;

exec sql execute s using :pnum, :pname, :age, :pcity;

printf("quit or continue?('q' to quit Person table else, '1' to continue.)\n");

scanf("%s",&pick);

}

}

if(strcmp(pick, "2") == 0){

while(strcmp(pick, "q") != 0) {

printf("T#:\n");

scanf("%s", &tnum);

printf("Name:\n");

scanf("%s", &tname);

printf("City:\n");

scanf("%s", &tcity);

strcpy(sqlstmt,"insert into Team values (:v1, :v2, :v3)");

exec sql prepare s from :sqlstmt;

exec sql execute s using :tnum, :tname, :tcity;

printf("quit or continue?('q' to quit Team table else, '2' to continue.)\n");

scanf("%s",&pick);

}

}

if(strcmp(pick, "3") == 0) {

while(strcmp(pick, "q") != 0) {

printf("P#:\n");

scanf("%s", &pnum);

printf("T#:\n");

scanf("%s", &tnum);

printf("Levels (In integer):\n");

scanf("%d", &levels);

strcpy(sqlstmt,"insert into Fan values (:v1, :v2, :v3)");

exec sql prepare s from :sqlstmt;

exec sql execute s using :pnum, :tnum, :levels;

printf("quit or continue?('q' to quit Fan table else, '3' to continue.)\n");

scanf("%s", &pick);

}

}

printf("There are 3 tables in db: \n");

printf("1. Person \n"); printf("2. Team \n"); printf("3. Fan \n");

printf("\n\n");

printf("Please enter a number 1, 2 or 3 to select a table or q to quit.\n");

scanf("%s",&pick);

exec sql execute immediate "commit";

}

finish:

printf("Insertion done!\n");

exit(0);

error:

printf("SQL error found, insert correct values or check duplicates. \n");

exec sql rollback release;

exit(1);

}

1. Write an Embedded SQL program that uses dynamic SQL method 3 to list all person rows, in person number order. Each person row should be immediately followed in the listing by all team rows for teams the person is fan of, in team number order. Persons who are not fan of any team should still be listed.

#include <stdio.h>

int main(){

exec sql include sqlca;

exec sql begin declare section;

char sqlstmt1[1024];

char sqlstmt2[1024];

char pnum[3];

char pname[10];

int age;

char pcity[10];

char tnum[3];

char tname[10];

char tcity[10];

int levels;

char \*LOGIN= "oracle/oracle11g";

exec sql end declare section;

exec sql whenever sqlerror goto error;

exec sql whenever not found do break;

exec sql connect :LOGIN;

if (sqlca.sqlcode == 0)

printf("Connected to ORACLE\n");

else

printf("Connection Failed\n");

strcpy(sqlstmt1, "select \* from Person order by P#");

exec sql prepare p from :sqlstmt1;

exec sql declare p\_cur cursor for p;

strcpy(sqlstmt2, "select \* from Team where exists (select \* from Fan where

Fan.P#=:pnum and Fan.T#=Team.T#) order by Team.T#");

exec sql prepare t from :sqlstmt2;

exec sql declare t\_cur cursor for t;

printf("+-----------------------------------------------------+\n");

printf("| P# | Name | Age | City |\n");

printf("+-----------------------------------------------------+\n");

while(1) {

exec sql fetch p\_cur into :pnum, :pname, :age, :pcity;

printf("| %s | %s | %d | %s |\n", pnum, pname, age, pcity);

exec sql open t\_cur using :pnum;

printf("+-----------------------------------------------------+\n");

printf("| +-----------------------------------------------+ |\n");

printf("| | T# | Name | City | |\n");

printf("| +-----------------------------------------------+ |\n");

while(1){

exec sql fetch t\_cur into :tnum, :tname, :tcity;

printf("| | %s | %s | %s | |\n", tnum, tname, tcity);

printf("| +-----------------------------------------------+ |\n");

}

printf("+-----------------------------------------------------+\n");

exec sql close t\_cur;

}

exec sql close p\_cur;

exit(0);

error:

printf("Error found in SQL. \n");

exec sql rollback release;

exit(1);

}

1. Write a PL/SQL program to redo part 3. Your program should contain a cursor for persons and a parameterized cursor for teams.

DECLARE

p\_tuple Person%rowtype;

t\_tuple Team%rowtype;

CURSOR p\_cursor IS select \* from Person ORDER BY P# ASC;

CURSOR t\_cursor(pnum Person.P#%type)

IS select T.\*

FROM Team T, Fan F

WHERE T.T# = F.T#

AND F.P# = pnum

ORDER BY T.T# ASC;

BEGIN

dbms\_output.put\_line(' Person ');

dbms\_output.put\_line('+----------+--------------------------------------------------+----------+--------------------------------------------------+');

dbms\_output.put\_line('| P# | Name | Age | City |');

dbms\_output.put\_line('+==========+==================================================+==========+==================================================+');

OPEN p\_cursor;

LOOP

fetch p\_cursor into p\_tuple;

exit when p\_cursor%NOTFOUND;

dbms\_output.put\_line('|' || lpad(to\_char(p\_tuple.P#),10) ||

'|' || rpad(p\_tuple.Name,50) ||

'|' || rpad(p\_tuple.Age,10) ||

'|' || rpad(p\_tuple.City,50) ||

'|');

dbms\_output.put\_line('+----------+--------------------------------------------------+----------+--------------------------------------------------+');

dbms\_output.put\_line('| |');

dbms\_output.put\_line('| Fans of |');

dbms\_output.put\_line('| +----------+--------------------+------------------------------+-----+--------------------+ |');

dbms\_output.put\_line('| | T# | Name | City |');

dbms\_output.put\_line('| +==========+====================+==============================+=====+====================+ |');

OPEN t\_cursor(p\_tuple.P#);

LOOP

FETCH t\_cursor into t\_tuple;

exit when t\_cursor%NOTFOUND;

dbms\_output.put\_line('| |' || lpad(to\_char(t\_tuple.T#),10) ||

'|' || rpad(t\_tuple.Name,20) ||

'|' || rpad(t\_tuple.City,10) ||

'|' || ' |' );

dbms\_output.put\_line('| +----------+--------------------+------------------------------+-----+--------------------+ |');

END LOOP;

CLOSE t\_cursor;

dbms\_output.put\_line('| |');

dbms\_output.put\_line('+----------+--------------------------------------------------+----------+--------------------------------------------------+');

END LOOP;

CLOSE p\_cursor;

END;