**Software Engineering of Web Applications**

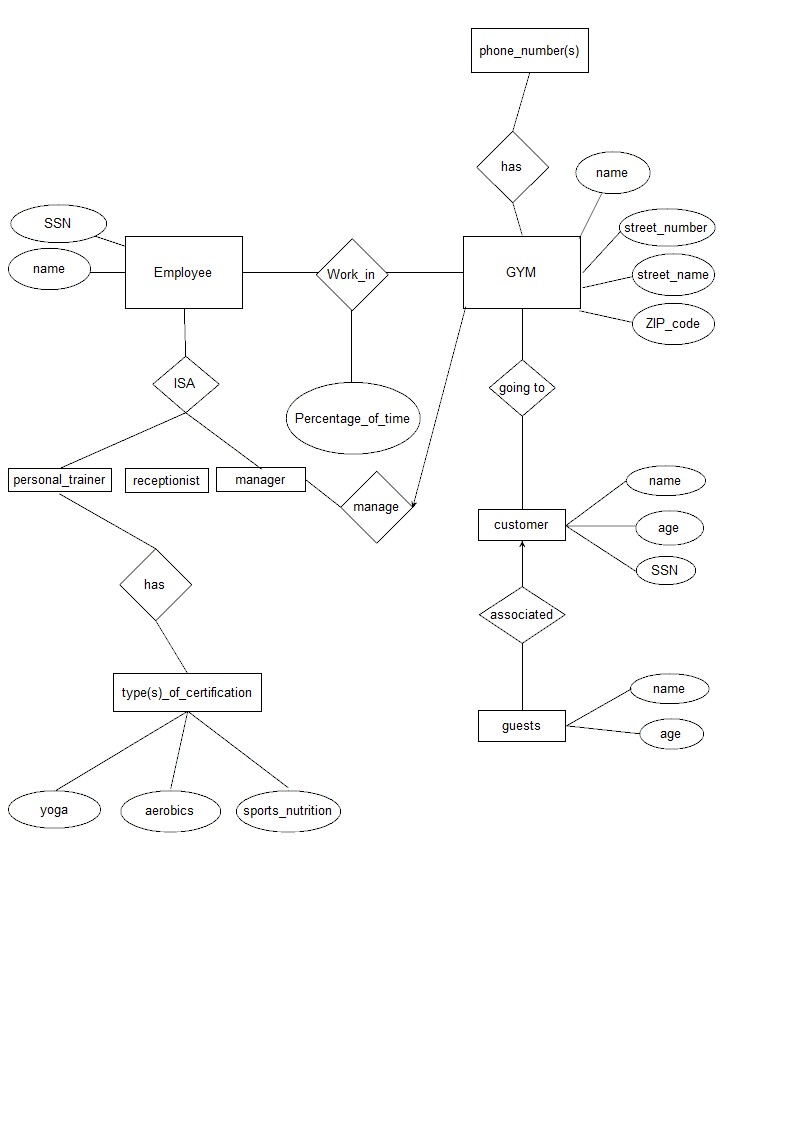
**Assignment 1**

**Ze Liu (174007753)**

#### Question 1

1. The ER diagram as follow:

Each key and participation constraints are signed and there is no overlap and covering constraints. Because each employee has **zero** or one specialization.



1. CREATE TABLE Employee

(

SSN CHAR(11),

ename CHAR(20),

PRIMARY KEY(SSN)

)

CREATE TABLE Work\_in

(

SSN CHAR(11),

gname CHAR(20),

Percentage of time CHAR(20),

PRIMARY KEY(SSN, gname),

FOREIGN KEY(SSN)

REFERENCE Employee,

FOREIGN KEY(gname)

REFERENCE GYM

)

CREATE TABLE GYM

(

gname CHAR(20),

street\_number: CHAR (20),

street\_name: CHAR (20),

ZIP\_code: INTEGER,

phone\_number: INTEGER,

PRIMARY KEY (gname)

)

CREATE TABLE customer

(

cssn: CHAR (20),

cname: CHAR (20),

cage: CHAR (20),

PRIMARY KEY (cssn)

)

CREATE TABLE Going\_to

(

gname: CHAR (20),

cssn: CHAR(20),

PRIMARY KEY (cssn , gname),

FOREIGN KEY (cssn)

REFERENCE customer,

FOREIGN KEY (gname)

REFERENCE GYM

)

CREATE TABLE personal\_trainer

(

ssn: CHAR(20),

name: CHAR(20),

type: CHAR (20),

PRIMARY KEY (ssn)

)

CREATE TABLE Manager

(

ssn: CHAR(20),

name: CHAR (20),

gname: CHAR (20),

street\_name: CHAR (20),

street\_number: INTEGER,

ZIP\_code: INTEGER,

FOREIGN KEY (gname)

REFERENCE GYM)

FOREIGN KEY (essn)

REFERENCE Employee

)

CREATE TABLE associated

(

cssn: CHAR (20),

cname: CHAR (20),

cage: INTEGER,

gname: CHAR (20),

gage: INTEGER,

PRIMARY KEY(cssn),

FOREIGN KEY(cssn) REFERENCE Customers,

UNIQUE (gname, gage)

)

#### Question 2

1. SELECT S.sname

FROM Suppliers S

WHERE NOT EXISTS

((SELECT P.pid

FROM Parts P

EXCEPT

(SELECT C.pid

FROM Catalog C

WHERE C.sid=s.sid))

1. SELECT C.sid,

FROM Catalog C

WHERE C.cost>(SELECT AVG(C2.cost)

FROM CatalogC2

WHERE C.pid=C2.pid)

1. SELECT S.sname

FROM Suppliers S, Catalog C

WHERE C.sid=S.sid and C.cost=(SELECT MAX(C2.cost)

FROM CatalogC2

WHERE C.pid=C2.pid)

(4) SELECT C.sid

FROM Parts P, Catalog C

WHERE P.pid=C.pid and P.color=‘red’

(5) SELECT C.sid

FROM Parts P,Catalog C

WHERE P.pid=C.pid and P.color=‘red’

UNION

SELECT C.sid

FROM Parts P,Catalog C

WHERE P.pid=C.pid and P.color=‘green’

(6) SELECT S.sname, C.cost

FROM Suppliers S, Catalog C

WHERE S.sid EXISTS (SELECT C.sid

FROM Catalog C, Parts P

WHERE P.pid=C.pid and P.color=‘red’

INTERSECT

SELECT C.sid

FROM Catalog C, Parts P

WHERE P.pid=C.pid and P.color=‘green’)

and C.cost =(SELECT MAX(C2.cost)

FROM Catalog C2

WHERE S.pid=C2.pid)

#### Question 3:

(1)

SELECT M.MovieName

FROM Movies M, MovieSupplier MS, Suppliers S

WHERE S.SupplierName = ‘Video Clubhouse’

and M.MovieID=MS.MovieID

and S.SupplierID=MS.SupplierID

UNION

SELECT M.MovieName

FROM Movies M, MovieSupplier MS, Suppliers S

WHERE S.SupplierName = ‘Ben's Video’

and M.MovieID=MS.MovieID

and S.SupplierID=MS.SupplierID

(2)

SELECT M.MovieName

FROM Rentals R,Movies M, Inventory I

WHERE M.MovieID = I.MovieID

and I.TapeID = R.TapeID

and R.Duration>=ANY(SELECT Duration

FROM Rentals)

(3)

SELECT S.SupplierName

FROM Suppliers S

WHERE NOT EXISTS (SELECT I.MovieID

FROM Inventory I

EXCEPT (SELECT MS.MovieID

FROM MovieSupplier MS

WHERE MS.supplierID = S. supplierID))

(4)

SELECT S.SupplierName, COUNT(DISTINCT MS.MovieID)

FROM Suppliers S, MovieSupplier MS, Movies M

WHERE S.SupplierID = MS.SupplierID

and MS.MovieID = M.MovieID

GROUP BY S.SupplierName

(5)

SELECT M.MovieName

FROM Movies M, Orders O

WHERE O.MovieID= M.MovieID

GROUP BY M.MovieName

HAVING SUM(Copies)> 4

(6)

SELECT C.FirstName

FROM Customers C, Rentals R, Inventory I, Movies M

WHERE C.CustID=R.CustomerID

and R.TapeID=I.TapeID

and I.MovieID=M.MovieID

and M.MovieName LIKE’% Kung Fu Panda %’

UNION

SELECT C.FirstName

FROM Customers C, Rentals R, Inventory I, Movies M, MovieSupplier MS, Suppliers S

WHERE C.CustID=R.CustID

and R.TapeID =I.TapeID

and I.MovieID=M.MovieID

and M.MovieID=MS.MovieID

and MS.SupplierID=S.SupplierID

and S.SupplierName AS ‘ Palm Video’

(7)

SELECT M.MovieName

FROM Movies M

WHERE M.MovieID EXISTS (SELECT I.MovieID

FROM Inventory I

WHERE COUNT(I.MovieID)>1)

(8)

SELECT DISTINCT C.FirstName

FROM Customers C

WHERE C.CustID EXISTS (SELECT R.CustomerID

FROM Rentals R

WHERE R.Duration >= 5)

(9)

SELECT DISTINCT Suppliers.suppliersname

FROM MovieSuppliers MS

WHERE MS.Price=(SELECT MIN(MS2.Price)

FROM MovieSuppliers2 MS2, Movies M

WHERE MS2.MovieID=M. MovieID

and M.MovieName AS ‘Cinderella 2015’

(10)

SELECT M.MovieName

FROM Movies M

WHERE M.MovieID NOT EXISTS (SELECT I.MovieID

FROM Inventory I

WHERE NULL)

#### Question 4

* We first complete the trigger, then we found the OldTuple.price(4) is bigger than NewTuple.price(3), also bigger than 1. So we update it. And the result is (111,1.5).
* We first choose to update it and then trigger. The result is the same (111,1.5).
* We first judge the trigger condition, if it is right then trigger. The result is (111,1.5).