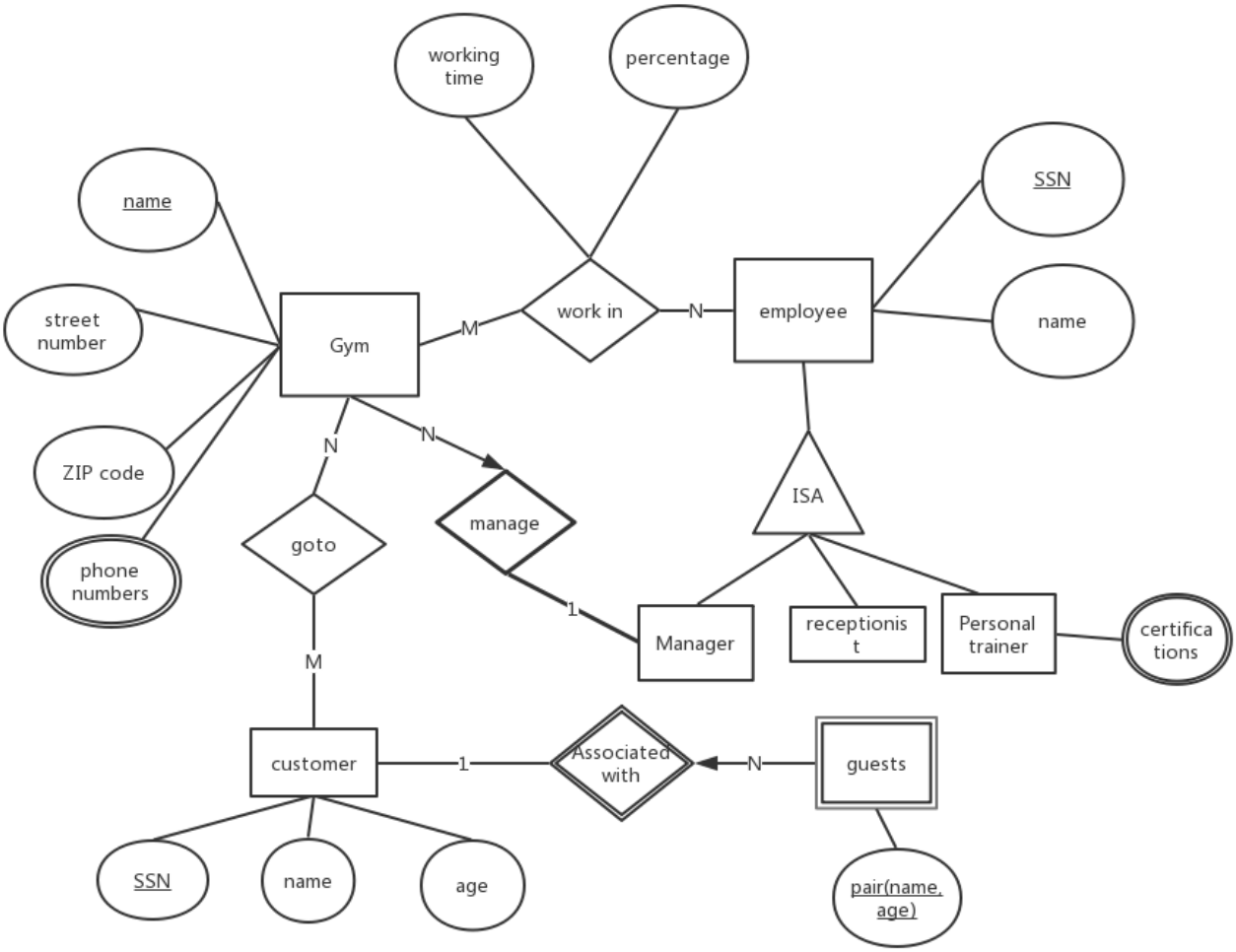
Homework 1 of ECE568 Web Application

Name: Chaoji Zuo RUID: 190003416 Date: 2.5

1 Table



1.2

```
CREATE TABLE Gym
(
    Name    CHAR(20) NOT NULL,
    StreetNumber    INTEGER,
    ZIPCode     CHAR(5),
    PhoneNumber CHAR(10),
    ManagerSSN     CHAR(20) NOT NULL,
    PRIMARY KEY (Name),
    FOREIGN KEY (ManagerSSN) REFERENCES Employee(SSN)
)
```

```
CREATE TABLE Customer
(
    SSN     CHAR(11) NOT NULL,
    Name    CHAR(20),
    Age     INTEGER,
    PRIMARY KEY (SSN)
)
```

```
CREATE TABLE Guest
(
    NameAge     CHAR(20),
    CustomerSSN     CHAR(20),
    PRIMARY KEY (NameAge,Customer),
    FOREIGN KEY (CustomerSSN) REFERENCES Customer(SSN)
)
```

```
CREATE TABLE Employee
(
    SSN     CHAR(11) NOT NULL,
    Name    CHAR(20),
    Specialization  CHAR(10),
    PRIMARY KEY (SSN)
)
```

```
CREATE TABLE PhoneNumber
(
    PhoneNumber1    CHAR(10),
    PhoneNumber2    CHAR(10),
    PhoneNumber3    CHAR(10),
    GymName     CHAR(20),
    PRIMARY KEY (GymName) REFERENCES Gym(Name)
)
```

```
CREATE TABLE Work_In
(
    GymName       CHAR(20),
    EmployeeSSN  CHAR(11),
    Percentage   REAL NOT NULL,
    WrokingTime   INTEGER,
    PRIMARY KEY (GymName, Employee, percetage),
    FOREIGN KEY (GymName) REFERENCES Gym(Name),
    FOREIGN KEY (EmployeeSSN) REFERENCES Employee(SSN),
)
```

```
CREATE TABLE Certification
(
    EmployeeSSN     CHAR(11),
    CertificationName CHAR(40),
    PRIMARY KEY (EmployeeSSN, CertificationName),
    FOREIGN KEY (EmployeeSSN) REFERENCES Employee (SSN)
)
```

```
CREATE TABLE Go_To
(
    CustomerSSN CHAR(11),
    GymName      CHAR(20),
    PRIMARY KEY (CustomerSSN, GymName),
    FOREIGN KEY (CustomerName) REFERENCES Customer (SSN),
    FOREIGN KEY (Gym) REFERENCES Gym (name)
)
```

2 Parts

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))
```

2-(2)

```
SELECT DISTINCT C.sid
FROM Catalog C
WHERE C.cost > (SELECT AVG(C1.cost)
                FROM Catalog C1
                WHERE C1.pid=C.pid)
```

2-(3)

```
SELECT P.pid S.sid
FROM Parts P, Suppliers S, Catalog C
WHERE P.pid=C.pid
```

```
    AND C.sid=S.sid
    AND C.cost= (SELECT C1.cost
                 FROM Catalog C1
                 WHERE C1.pid=C.pid)
```

2-(4)

```
SELECT C.sid
FROM Catalog C
WHERE NOT EXIST (SELECT *
                 FROM Parts P
                 WHERE P.pid=C.pid
                    AND P.color <> "red")
```

2-(5)

```
SELECT C.sid
FROM Catalog C
WHERE EXIST (SELECT P.cid
             FROM Parts P
             WHERE (P.color="red" OR P.color="green")
                AND P.pid=C.cid)
```

2-(6)

```
SELECT S.sname, MAX(C.cost)
FROM Suppliers S, Catalog C, Parts p
WHERE S.sid=C.sid
    AND P.pid=C.pid
    AND P.color IN ("red","green")
```

3 Movies

3-1

```
SELECT M.MovieName
FROM Movies M, MovieSuppliers MS, Suppliers S
WHERE M.MovieID = MS.MoiveID
    AND MS.SupplierID = S.SupplierID
    AND (S.SupplierName="Ben's Video"
        OR S.SupplierName="Video Clubhouse")
```

3-2

```
SELECT M.MovieName
FROM Movies M, Rentals R, Inventory I
WHERE M.MovieID = I.MovieID
    AND I.TapeID = R.TapeID
    AND R.Duration = (SELECT MAX(Duration)
                      FROM Rentals)
```

3-3

```
SELECT S.SupplierName
FROM Suppliers S
WHERE S.SupplierID NOT IN
    (SELECT MS.SupplierID
     FROM MoviesSupplier MS, Inventory I
     WHERE NOT EXISTS
        (SELECT *
         FROM MoviesSupplier MS1, Inventory I1
         WHERE MS1.MovieID = I1.MovieID
         AND I2.MovieID = I.MovieID
         AND MS2.SupplierID = MS.SupplierID
        )
    )
```

3-4

```
SELECT S.SupplierName, COUNT(DISTINC MS.MovieID)
FROM Suppliers S, MovieSuppliers MS
WHERE S.SupplierID = MS.SupplierID
GROUP BY S.SupplierName
```

3-5

```
SELECT M.MovieName
FROM Movies M, Orders O
WHERE M.MovieID = O.MovieID
GROUP BY M.MovieName
HAVING SUM(O.copies) > 4
```

3-6

```
SELECT C.LastName, 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 = "Kung Fu Panda"
UNION
SELECT C.LastName, C.FirstName
FROM Customers C, Rentals R, Inventory I, MovieSupplier MS, Suppliers S
WHERE C.CustID = R.CustomerID
    AND R.TapeID = I.TapeID
    AND I.MovieID = MS.MovieID
    AND MS.SupplierID = S.SupplierID
    AND S.SupplierName = "Palm Video"
```

3-7

```
SELECT M.MovieName
FROM Movies M, Inventory I1, Inventory I2
WHERE I1.TapeID <> I2.TapeID
    AND I1.MovieID = I2.MovieID
    AND I1.MovieID = M.MovieID
```

3-8

```
SELECT C.LastName, C.FirstName
FROM Customers C, Rentals R
WHERE C.CustID = R.CustomerID
     AND R.Duration>=5
```

3-9

```
SELECT S.SupplierName
FROM Suppliers S, MovieSupplier MS, Movies M
WHERE S.SupplierID = MS.SupplierID
    AND M.MovieName = "Cinderella 2015"
    AND M.MovieID = MS.MovieID
    AND MS.Price <= ALL(SELECT Price
                        FROM MovieSupplier MS, Movies M
                        WHERE MS.MovieID = M.MovieID
                            AND M.MovieName = "Cinderella 2015")
```

3-10

```
SELECT M.MovieName
FROM Movies M
WHERE M.MovieID NOT IN
```

```
      (SELECT I.MovieID
       FROM Inventory I)
```

4

4-a)

At first, the new price of row 111 would be 3 and the old price is 4, so it satisfies the requirment of triggle, then the trigger would be triggled and the updated price would be 3/2. But this all happened before the update process, so the price would be rewirte and since the purchaseID haven't change, the price wolud became 3. So the price of purchaseID 111 would be 3 at the end.

4-b)

The trigger would be triggled after the update process, so the price of 111 would be 1.5 at the end.

4-c)

INSTEAD OF means the original update process wolud be replace, so the price first became 3 and then trigger the trigger. So the price became 1.5 at last.