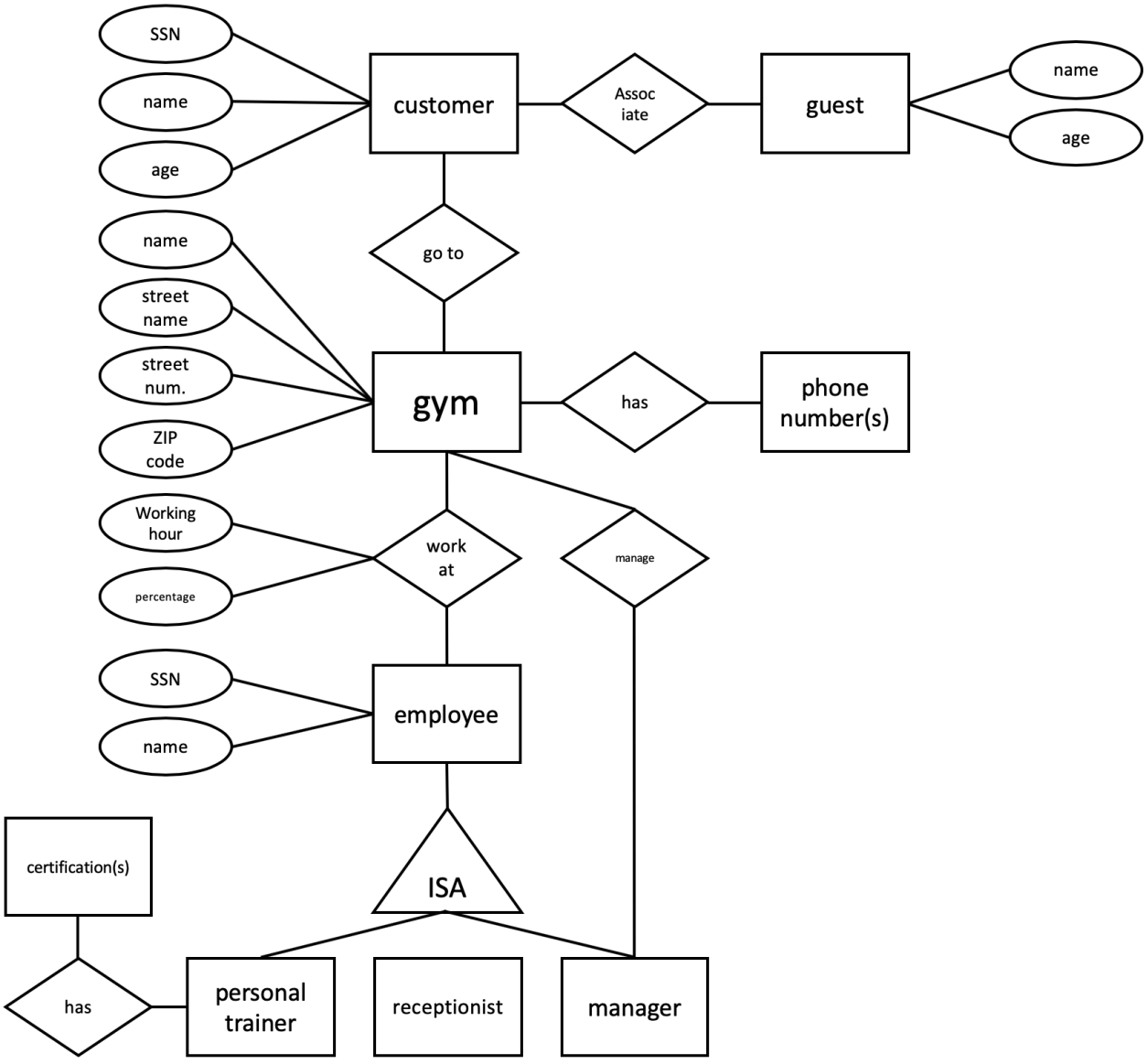
Web Application Assignment 1

Ruiyu Zhang || rz213
2019.02.05

Question 1

1) ER-Diagram



2) SQL codes that creates database

```
CREATE TABLE gym (
    gym_id BIGINT NOT NULL AUTO_INCREMENT,
    name CHAR(30) NOT NULL,
    street_name CHAR(30),
    street_num CHAR(10),
    zip_code CHAR(5),
    manager CHAR(11) NOT NULL,
    FOREIGN KEY (manager) REFERENCES employee (SSN),
    PRIMARY KEY (gym_id)
)

CREATE TABLE employee (
    emp_id BIGINT NOT NULL AUTO_INCREMENT,
    ssn CHAR(11) NOT NULL,
    name CHAR(30),
    specialization CHAR(20),
    PRIMARY KEY (emp_id)
)

CREATE TABLE customer (
    uid BIGINT NOT NULL AUTO_INCREMENT,
    ssn CHAR(11) NOT NULL,
    name CHAR(30),
    age INT,
    PRIMARY KEY (uid)
)

CREATE TABLE guest (
    uid BIGINT,
    age INT,
    name CHAR(30),
    FOREIGN KEY (customer) REFERENCES (uid)
)

CREATE TABLE phone (
    pho_id BIGINT NOT NULL AUTO_INCREMENT,
    phone_num BIGINT NOT NULL,
    gym_id BIGINT,
    FOREIGN KEY (gym_id) REFERENCES gym (gym_id)
)

CREATE TABLE go_to (
    uid BIGINT,
    gym_id BIGINT,
    PRIMARY KEY (uid, gym_id),
    FOREIGN KEY (gym_id) REFERENCES gym (gym_id),
```

```
    FOREIGN KEY (uid) REFERENCES customer (uid)
)

CREATE TABLE work_at (
    gym_id BIGINT NOT NULL,
    emp_id BIGINT NOT NULL,
    percentage REAL,
    working_hours CHAR(20),
    PRIMARY KEY (gym_id, emp_id),
    FOREIGN KEY (gym_id) REFERENCES gym (gym_id),
    FOREIGN KEY (emp_id) REFERENCES employ (emp_id)
)

CREATE TABLE certification (
    cer_id BIGINT NOT NULL AUTO INCREMENT,
    emp_id BIGINT NOT NULL,
    certification_title CHAR(30),
    FOREIGN KEY (emp_id) REFERENCES empolyee (emp_id)
)
```

Question 2

1)

```
SELECT s.sname FROM Suppliers s
WHERE NOT EXISTS ( -- true if 'all parts' included in 'all parts from supplier'
    SELECT p.pid FROM Parts p EXCEPT ( -- all kinds of parts
        SELECT c.pid FROM Catalog c -- all kinds of parts from current supplier
        WHERE c.sid = s.sid
    )
)
```

2)

```
SELECT DISTINCT c1.sid FROM Catalog c1
WHERE c1.cost > (
    SELECT AVG(c2.cost) FROM Catalog c2 -- average of all of current kind of
part
    WHERE c1.pid = c2.pid
)
```

3)

```
SELECT s.sname FROM Suppliers s, Catalog c1
WHERE c1.sid = s.sid AND c1.cost = (
    SELECT MAX(c2.cost) FROM Catalog c2 -- max of all of current kind of part
    WHERE c1.pid = c2.pid
)
```

4)

```
SELECT c.sid FROM Catalog c
WHERE NOT EXISTS (
    SELECT p.color FROM Parts p -- all colors (except red) of parts from cur
supplier
    WHERE p.pid = c.pid AND p.color <> "red"
)
```

5)

```
SELECT c.sid FROM Catalog c
WHERE EXISTS (
    SELECT p.color FROM Parts p
    WHERE p.pid = c.pid AND (p.color = "red" AND p.color = "green")
)
```

6)

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

Question 3

1)

```sql
SELECT m.MovieName
FROM Movies m, MovieSupplier ms, Suppliers s
WHERE ms.SupplierID = s.SupplierID AND ms.MovieID = m.MovieID
    AND (s.SupplierName = "Ben's Video" OR s.SupplierName = "Video Clubhouse")
```

2)

```sql
SELECT m.MovieName
FROM Movies m, Inventory i, Rentals r
WHERE i.MovieID = m.MovieID AND i.TapeID = r.TapeID
    AND r.Duration >= ALL(
        SELECT Duration FROM Rentals
    )
```

3)

```sql
SELECT s.SupplierName
FROM Suppliers s
WHERE NOT EXISTS (
    SELECT i.MovieID -- Movies that current Supplier doesnot supply
    FROM Inventory i
    EXCEPT (
        SELECT ms.MovieID -- Movies that current Suplier supplies
        FROM MovieSupplier ms
        WHERE ms.SupplierID = s.SupplierID
    )
)
```

4)

```sql
SELECT s.SupplierName, COUNT(DISTINCT ms.MovieID)
FROM Supplier s, MovieSupplier ms
WHERE s.SupplierID = ms.SupplierID
GROUP BY s.SupplierName
```

5)

```sql
SELECT m.MovieName
FROM Movies m, Orders o
WHERE m.MovieID = o.MovieID
GROUP BY m.MovieName
HAVING SUM(o.Copies) > 4
```

6)

```
SELECT c.FirstName, c.LastName
FROM Customers c, Rentals r, Movies m, Inventory i
WHERE c.CustID = r.CustID AND r.TapeID = i.TapeID
    AND i.MovieID = m.MovieID AND m.MovieName = "Kung Fu Panda"
UNION
SELECT c.FirstName, c.LastName
FROM Customers c, Rentals r, Inventory i, MovieSupplier ms, Suppliers s
WHERE c.CustID = r.CustID AND r.TapeID = i.TapeID
    AND i.MovieID = ms.MovieID AND ms.SupplierID = s.SupplierID
    AND s.SupplierName = "Palm Video"
```

7)

```
SELECT m.MovieName
FROM Movies m, Inventory i
WHERE m.MovieID = i.MovieID
GROUP BY m.MovieID
HAVING COUNT(i.TapeID) > 1
```

8)

```
SELECT DISTINCT c.FirstName, c.LastName
FROM Customer c, Rentals r
WHERE c.CustID = r.CustomerID AND r.Duration > 5
```

9)

```
SELECT DISTINCT s.SupplierName
FROM Suppliers s, MovieSupplier ms
WHERE s.SupplierID = ms.SupplierID AND ms.Price = (
    SELECT MIN(ms2.Price) -- Min Price for "C~2015"
    FROM MovieSupplier ms2, Movies m
    WHERE ms2.MovieID = m.MovieID AND m.MovieName = "Cinderella 2015"
)
```

10)

```
SELECT m.MovieName
FROM Movies m
WHERE m.MovieID NOT IN (
    SELECT DISTINCT i.MovieID
    FROM Inventory i
)
```

Question 4

1)

```
TRIGGER START
UPDATE Row = (NewTuple.purchaseID, NewTuple.price / 2) = (111, 1.5)
TRIGGER END
UPDATE Row = NewTuple = (111, 3)
```

- Conclusion: Target Row updated as (111, 3)

2)

```
UPDATE Row = NewTuple = (111, 3)
TRIGGER START
UPDATE Row = (NewTuple.purchaseID, NewTuple.price / 2) = (111, 1.5)
TRIGGER END
```

- Conclusion: Target Row updated as (111, 1.5)

3)

```
TRIGGER START
UPDATE Row = (NewTuple.purchaseID, NewTuple.price / 2) = (111, 1.5)
TRIGGER END
```

- Conclusion: Target Row updated as (111, 1.5)