NYU Tandon School of Engineering **Oct 21, 2017**
Computer Science and Engineering
CS6083, Fall 2017

Problem Set #2 Sample Solution

Problem 1

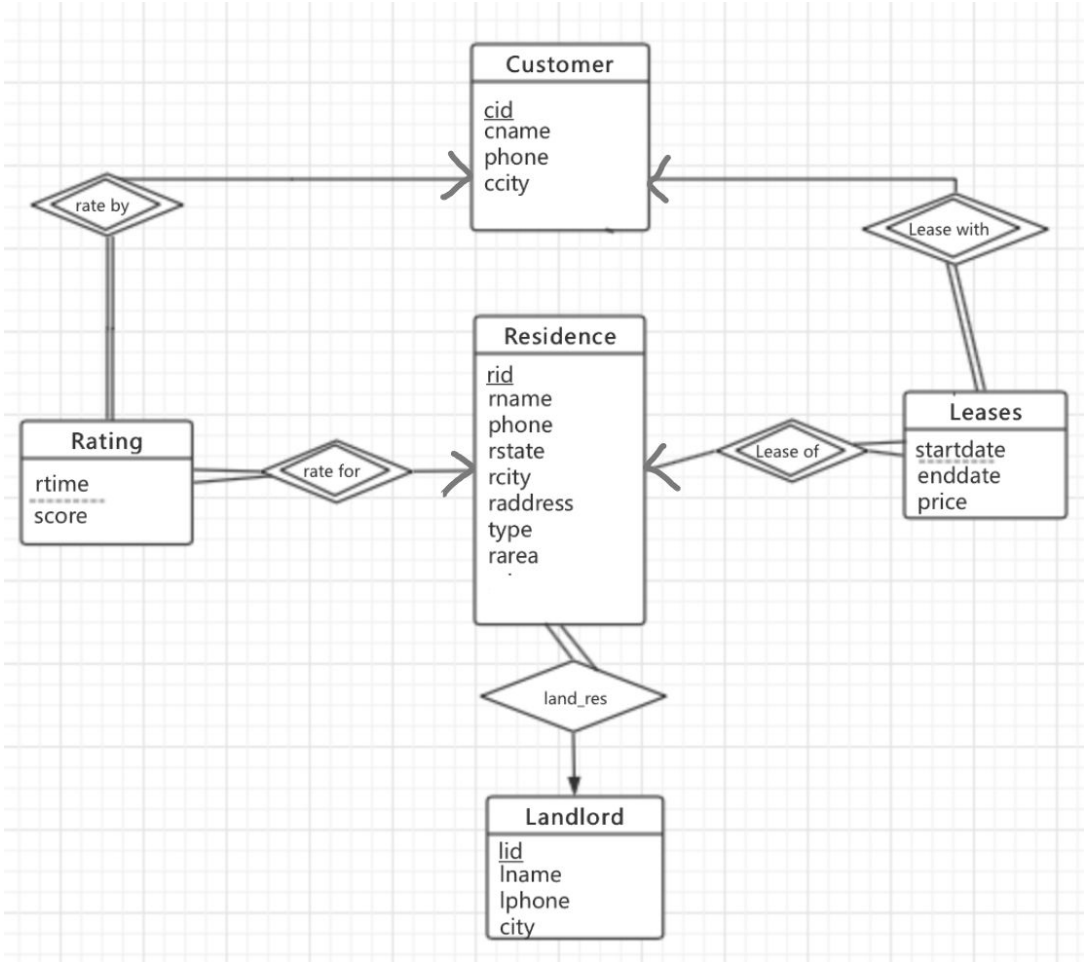
(a)

Yes, (rstate, rcity, raddr) can be used as the new primary key, because each tuple in Residence table could be identified by its detailed address.

(b)

Residence (lid) references Landlord(lid)
Leases (cid) references Customer(cid)
Leases (rid) references Residence(rid)
Rating (cid) references Customer(cid)
Rating (rid) references Residence(rid)

(c)



(e)
(i) List the cids and cnames of all customers who have rented residences in Chicago.

```
SELECT DISTINCT Customer.cid, Customer.cname
FROM Customer, Residence, Leases
WHERE Customer.cid = Leases.cid AND Residence.rid = Leases.rid
AND Residence.rcity = 'Chicago'
```

(ii) List the cids of any customers living in Chicago who had a lease with a landlord living in Seattle.

```
SELECT DISTINCT Customer.cid
FROM Customer, Residence, Leases, Landlord
WHERE Customer.cid = Leases.cid AND Residence.rid = Leases.rid
AND Residence.lid = Landlord.lid AND Customer.ccity = 'Chicago'
AND Landlord.lcity = 'Seattle'
```

(iii) Output the cid and cname of any customers who have only rented residences in Chicago. (That is, they have rented at least one residence in Chicago, and have not rented anything anywhere else.)

```
SELECT DISTINCT Customer.cid, Customer.cname
FROM Customer, Residence, Leases
WHERE Customer.cid = Leases.cid AND Residence.rid = Leases.rid
AND Residence.rcity = 'Chicago'
AND Customer.cid NOT IN

(SELECT DISTINCT Customer.cid
FROM Customer, Residence, Leases
WHERE Customer.cid = Leases.cid
AND Residence.rid = Leases.rid
AND Residence.rcity != 'Chicago')
```

(iv) Output the residence type that is owned by the largest number of distinct landlords.

```
SELECT DISTINCT Residence.rtype
FROM Residence
GROUP BY Residence.rtype
HAVING COUNT(DISTINCT lid)
>= ALL
(SELECT COUNT(DISTINCT lid)
FROM Residence
GROUP BY Residence.rtype)
```

(v) Output the cid and cname of the customer(s) who spent the most money (sum of prices) for rentals with starting dates in 2016.

```
SELECT Customer.cid, Customer.cname
FROM Customer, Leases
WHERE Customer.cid = Leases.cid AND Year(startdate) = '2016'
GROUP BY Customer.cid, Customer.cname
HAVING SUM(price) >= ALL
(SELECT SUM(price)
FROM Customer, Leases
WHERE Customer.cid = Leases.cid
AND Year(startdate) = '2016'
GROUP BY Leases.cid)
```

(vi) For each lcity, output the lids of the landlords living in that city who made the most money overall.

```
SELECT a.lcity, a.lid, maxsum
FROM
(
SELECT DISTINCT Landlord.lcity, Landlord.lid, sum(price) AS sumprice
FROM Landlord, Residence, Leases
WHERE Landlord.lid = Residence.lid AND Leases.rid = Residence.rid
GROUP BY Landlord.lid, Landlord.lcity
)
AS a,
(
SELECT lcity,MAX(sumprice) AS maxsum
FROM
(
SELECT DISTINCT Landlord.lcity, Landlord.lid, sum(price) AS
sumprice
FROM Landlord, Residence, Leases
WHERE Landlord.lid = Residence.lid AND Leases.rid = Residence.rid
GROUP BY Landlord.lid, Landlord.lcity
)
AS b
GROUP BY lcity
)
AS c
WHERE c.lcity = a.lcity AND c.maxsum = a.sumprice
```

(vii) For each month in 2016, output the rcity whose houses had the best average rating, based only on ratings that were given during that month.

```
SELECT avg_month.month, rcity, max_month.max_avg
FROM(
SELECT MONTH(rtime) AS month, AVG(score) AS avg_value, rcity
FROM Rating, Residence
WHERE Rating.rid = Residence.rid AND YEAR(rtime) = '2016'
GROUP BY rcity, MONTH(rtime))
AS avg_month,
(SELECT month, MAX(avg_value) AS max_avg
FROM (
SELECT MONTH(rtime) AS month, AVG(score) AS avg_value,
rcity
```

```
FROM Rating,Residence
WHERE Rating.rid = Residence.rid AND YEAR(rtime) = '2016'
GROUP BY rcity, MONTH(rtime))
GROUP BY month)
AS max_month
WHERE avg_month.month = max_month.month AND avg_month.avg_value = max_month.max_avg
```

(viii) Output the average duration of leases (enddate - startdate) for customers living in San Diego, for leases with start date in 2016. The unit of average time should be days.

```
SELECT AVG(datediff(enddate,startdate)) AS duration
FROM Leases, Customer
WHERE Leases.cid = Customer.cid AND Customer.ccity = 'San Diego'
AND Year(startdate) = '2016'
```

(f)

(iii)

CusInChicago ← $\Pi_{Customer.cid,\ cname}\sigma_{rcity = "Chicago"}(Customer \bowtie Leases \bowtie Residence)$

CusNotInChicago ← $\Pi_{Customer.cid,\ cname}\sigma_{rcity\ != "Chicago"}(Customer \bowtie Leases \bowtie Residence)$

Result = CusInChicago - CusNotInChicago

(iv)

temp ← $_{rtype}G_{count(lid)\ as\ lcount}(Residence)$

temp2 ← $G_{max(lcount)\ as\ maxlcount}(temp)$

$\Pi_{rtype}\sigma_{lcount = maxlcount}(temp \times\ temp2)$

(v)

SumRent ← $_{Leases.cid}G_{SUM(price)\ as\ totalRent}\sigma_{Year(startdate) = 2016}(Leases)$

MaxSum ← $G_{max(totalRent)\ as\ maxTRent}(SumRent)$

$\Pi_{Customer.cid,\ cname}\sigma_{totalRent = maxTRent}((SumRent\ \times\ MaxRent) \bowtie Customer)$

(vi)

temp ← $_{Landlord.lcity,\ Landlord.lid}G_{SUM(price)\ as\ totalIncome}\ (Leases \bowtie Residence \bowtie Landlord)$

temp2 ← $_{lcity}G_{max(totalIncome)\ as\ maxIncome}(temp)$

$\Pi_{temp.lcity,\ lid}\sigma_{totalIncome = maxIncome}(temp \bowtie temp2)$

(vii)

temp ← $_{MONTH(rtime)\ as\ month,\ rcity}G_{avg(score)\ as\ avgRate}\sigma_{Year(rtime) = 2016}(Residence \bowtie Rating)$

temp2 ← $_{month}G_{max(avgRate)\ as\ maxRate}(temp)$

$\Pi_{temp.month,\ rcity,\ avgRate}\sigma_{avgRate = maxRate}(temp \bowtie temp2)$

(viii)

$$G_{avg(datediff(enddate,\ startdate))\ as\ duration}\sigma_{YEAR(startdate)\ =\ 2016\ and\ ccity\ =\ "San\ Diego"}(Leases \bowtie Customer)$$

(g)

(i)

```
UPDATE Landlord
SET Landlord.lname = 'Amy Daniels'
WHERE Landlord.lname = 'Bob Daniels';
```

(ii)

Note: Customer cid is foreign keys of Rating and Leases, add ON DELETE CASCADE first before executing the query.

```
DELETE FROM Customer
WHERE cid Not IN (
SELECT cid
FROM Leases
WHERE timestampdiff (YEAR, Leases.startdate,
CURDATE ()) <= 3)
```

(iii)

```
DELETE FROM Rating
WHERE score = '1' AND rid IN (
SELECT rid
FROM Rating
GROUP BY rid
HAVING AVG(score) < 3)
```

Problem 2

(a)

Define a view that contains for each residence the lid, lname, lcity, rid, rname, and rcity.

```
CREATE VIEW LandlordofResidence
AS
(
SELECT Landlord.lid, lname, lcity, Residence.rid, rname, rcity
FROM Residence, Landlord
WHERE Residence.lid = Landlord.lid
)
```

(i) For each lcity, output the lname that owns the most residences.

```
SELECT temp2.lcity, lname
FROM
(SELECT lcity, lid, lname, COUNT(DISTINCT rid) as countrid
```

```
FROM LandlordofResidence
GROUP BY lid,lcity
)
AS temp1,
(SELECT a.lcity, MAX(countrid)as maxcount
FROM (SELECT lcity,lid,COUNT(DISTINCT rid) as countrid
FROM LandlordofResidence
GROUP BY lid,lcity) as a
GROUP BY lcity
)
AS temp2
WHERE temp1.lcity = temp2.lcity AND temp1.countrid = temp2.maxcount
```

(ii) Change the name of any residence with name “Little Cabin” to “Big Cabin”.

```
UPDATE LandlordofResidence
SET rname = 'Big Cabin'
WHERE rname = 'Small Cabin'
```

(b)

If you are using mysql,Then the trigger cannot be implemented because "A stored function or trigger cannot modify a table that is already being used (for reading or writing) by the statement that invoked the function or trigger." according to https://dev.mysql.com/doc/refman/5.7/en/stored-program-restrictions.html#stored-routines-function-restrictions

```
CREATE trigger hw2b
AFTER INSERT on Rating
FOR EACH ROW
IF
((SELECT AVG(score) as avg_score
FROM Rating r
WHERE r.rid = NEW.rid) <3
AND EXISTS (SELECT 1 From Rating r2 WHERE score = '1' AND
r2.rid = NEW.rid))
THEN
DELETE FROM Rating
WHERE score = '1' AND Rating.rid = NEW.rid;
END IF
```

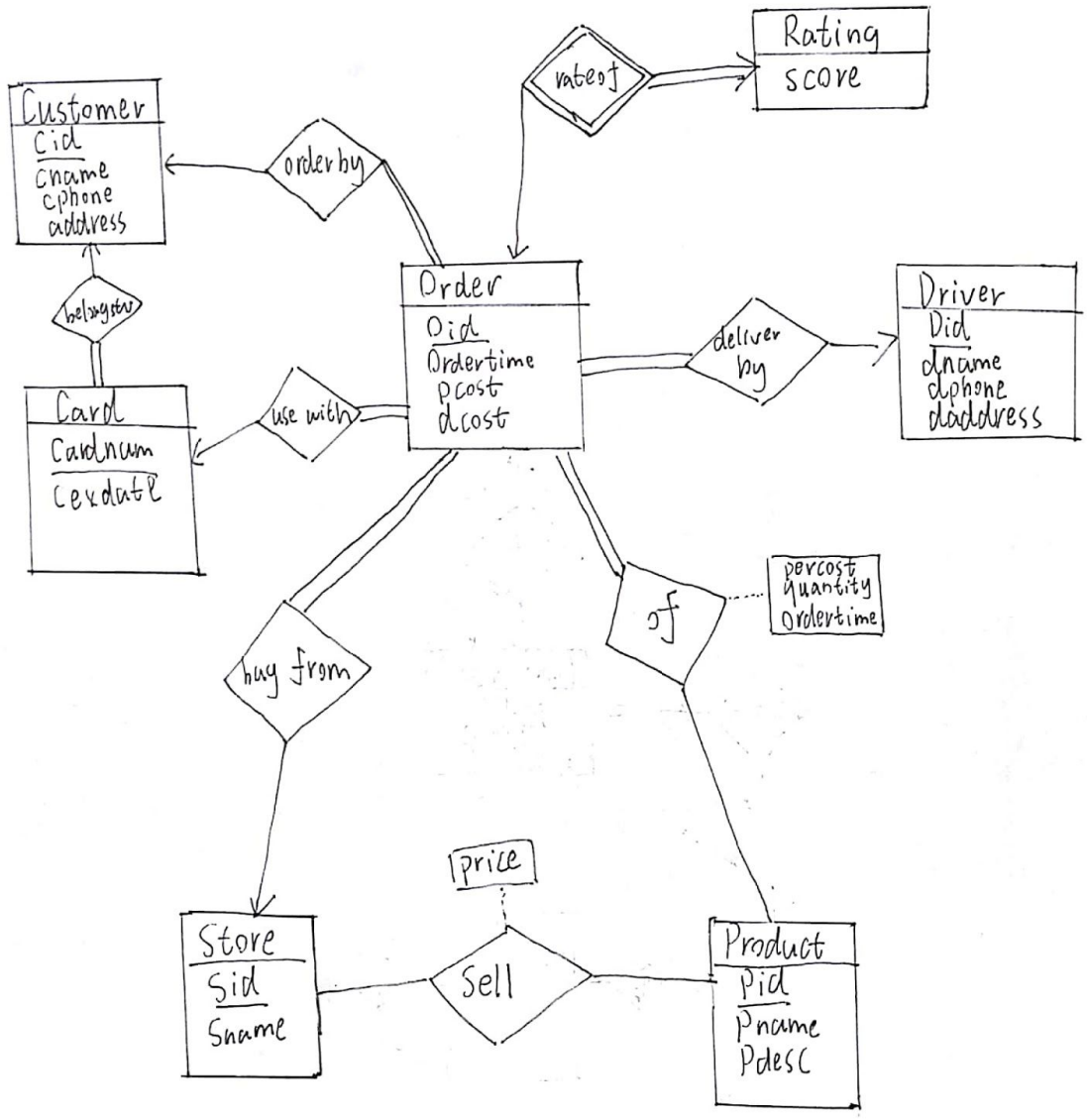
(c)

```
CREATE trigger hw2c
Before INSERT on Rating
FOR EACH ROW
IF
(
SELECT count (*)
FROM Rating
WHERE NEW.cid = cid AND score = '1' and timestampdiff(Week, rtime,
CURDATE())<=2
```

```
) >=5
THEN
ROLLBACK
END IF;
```

Problem 3

(a)



(b)

Driver (<u>did</u>, dname, daddress, dphone)

Customer (<u>cid</u>, cname, caddress, cphone)

Card (ccardnum, cid, cexdate)
Store (sid, sname)
Product (pid, pname, pdes)
Order (oid, cid, sid, did, pcost, dcost, cardnum)
Selling(sid,pid,price)
Orderdetail(oid, pid, percost, quantity, ordertime)
Rate (oid, score)

Foreign Key:
Card.cid references Customer.cid
Order.cid references Customer.cid
Order.sid references Store.sid
Order.did references Driver.did
Order.cardnum references Card.cardnum
Selling.sid references Store.sid
Selling.pid references Product.pid
Orderdetail.oid references Order.oid
Orderdetail.pid references product.pid
Rate.oid references Order.oid

(c)
(i)

```
SELECT driver.did,dname,SUM(order.dcost)
FROM driver LEFT OUTER JOIN (`order` NATURAL JOIN store)
WHERE sname = 'Tom's Store' and `order`.did = driver.did
GROUP BY driver.did
```

(ii)

```
SELECT driver.did, dname
FROM driver
WHERE driver.did NOT IN
(
SELECT did
FROM `order`,rate
WHERE `order`.oid = rate.oid
AND rate.score =5
)
```

(iii)

```
SELECT customer.cid, cname
FROM customer, `order`
WHERE customer.cid = `order`.cid
GROUP BY cid,sid
HAVING COUNT(oid)>=4
```

(iv)

```
SELECT customer.cid, cname, SUM(dcost+pcost) as sumcost
FROM customer, `order`
where customer.cid = `order`. cid AND YEAR(`order`. ordertime) = '2016'
GROUP BY customer.cid
HAVING sumcost>=ALL
(
SELECT SUM(dcost+pcost) as sumcost
FROM customer, `order`
where customer.cid = `order`.cid
AND YEAR(`order`.ordertime) = '2016'
GROUP BY customer.cid
)
```