CS4350 Database Systems Spring 2024 Test#2

Name: Loc Nguyen

Q1. (25 points) Given the schema S= < { A,B,C,D,E,G,H }, F>, where F represents the following dependencies:

EH→AB
E→H
A→C
E→C
B→G
G→D
E→D
EB→G

i. Find a minimal cover for this schema.
Rules – Singleton RHS, no extraneous solution on LHS, no redundant functional dependencies

Solution:
E→ABH
A→C
B→G
G→D

Step 1:

E+	H+
EHABCGD	H

E alone can determine A, B so rewrite EH→A and EH→B

Step 2:

E+	B+
EHABCGD	BGD

E alone can determine G so rewrite EB →G to E→G

Step 3: Find extraneous solution on LHS
E+ (without $E \rightarrow A$)
EBHCDG (no A so keep $E \rightarrow A$)

E+ (without $E \rightarrow B$)
EAHCDG (no B so keep $E \rightarrow B$)

E+ (without $E \rightarrow H$)
EABCDG (no H so keep $E \rightarrow H$)

A+ (without $A \rightarrow C$)
A (no C so keep $A \rightarrow C$)

E+ (without $E \rightarrow C$)
EABCGDH (has C so remove $E \rightarrow C$)

B+ (without $B \rightarrow G$)
B (no G so keep $B \rightarrow G$)

G+ (without $G \rightarrow D$)
G (no D so keep $G \rightarrow D$)

E+ (without $E \rightarrow D$)
EABHCGD (has D so remove $E \rightarrow D$)

E+ (without $E \rightarrow G$)
EABHCGD (has G so remove $E \rightarrow G$)

ii. Find a key for this schema
Key for schema is whatever not represented on RHS but still in the set
Key = {E}

iii. Find a third normal form decomposition for this schema
R1 = {EABH}
R2 = {AC}
R3 = {BG]
R4 = {GD}
R5 = {E}

iv. Find a BCN form decomposition for this schema.

E→ABH doesn't satisfy BCNF so we use it to partition the table.
R1 = {EABH}
Now, we can consider the remaining set of values {C, D, E, G} and the relation G→D and key {E}
G→D doesn't satisfy BCNF so we use it to partition the table
R2 = {GD}
R3 = {EC} satisfies the dependency because none is left

Solution:
R1 = {EABH}
R2 = {GD}
R3 = {EC}

v. Determine whether the following decompositions are lossy or lossless.

R1= { E, D,H}   R2= { D, C, E, G,B, A }
R1&&R2 = {E, D} and R1 – R2 = {H}
R1&&R2 does determine H so compression is lossless

R1= {A, C}   R2= { B, D, E, G, H, C }
R1&&R2 = {C} and R1 – R2 = {A}
R1&&R2 = {C} and R2 – R1 = {B, D, E, G, H}
R1&&R2 does not determine R1 – R2 or R2 – R1 so compression is lossy

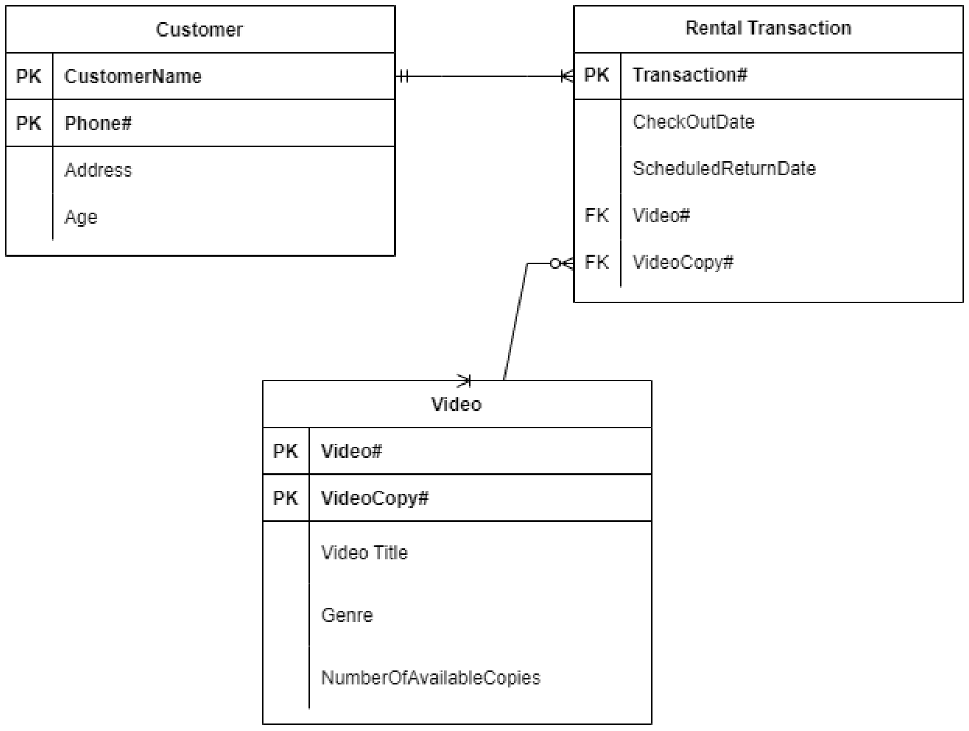
Q2. (20 points) Use the IDEFX notation to draw an ER of the following video rental shop.

- Video: Video#, VideoCopy#, Video Title, Rating, Genre, Number of available copies;
- Customer: Customer Name, Address, Age, Phone#;
- Rental Transaction: Transaction#, Check-out-Date, Scheduled Return Date, Video#, VideoCopy#

  The assumptions are as follows:
- Every transaction is associated with only one customer and <u>may include several video copies .</u>
- Customer name and Phone# are the key for Customer and each Customer has at least one Rental Transaction ;
- Video# is the key for the video attributes.
- VideoCopy# and Video# are the key for a video copy;
- Rental Transaction is the key for transaction attributes.

Please clearly state any additional assumptions
In addition to assumptions stated above, the following assumption is also made

1. Each video must belong can belong to 0 or many transactions but each transaction must belong to at least 1 video or many videos



Q3. (10 points) consider the application of Question#2. Write all possible dependencies and determine a key for all attributes.

- Video: Video#, VideoCopy#, Video Title, Rating, Genre, Number of available copies;
- Customer: Customer Name, Address, Age, Phone#;
- Rental Transaction: Transaction#, Check-out-Date, Scheduled Return Date, Video#, VideoCopy#

  The assumptions are as follows:
- Every transaction is associated with only one customer and <u>may include several video copies .</u>
- Customer name and Phone# are the key for Customer and each Customer has at least one Rental Transaction ;
- Video# is the key for the video attributes.
- VideoCopy# and Video# are the key for a video copy;
- Rental Transaction is the key for transaction attributes.

Solution:

As seen in the ERD, the following dependencies and key can be infer

1. Video#, VideoCopy#, VideoTitle, Rating, Genre, and NumberOfAvailableCopies from the VIDEO table has dependency with the Transaction#, CheckOutDate, ScheduledReturnDate of the TRANSACTION table
2. CustomerName, Address, Age, Phone# from the CUSTOMER table has dependency with Transaction#, CheckOutDate, ScheduledReturnDate, Video#, and VideoCopy# from the TRANSACTION table.
3. Key for the attributes from the CUSTOMER table are the CustomerName and Phone#
4. Key for the attributes from the TRANSACTION table is the Transaction# as primary key and Video# and VideoCopy# as foreign keys
5. Key for attributes from the VIDEO table are the Video# and VideoCopy#

Q4.(35 points) Given the following relational database schema:

• FLIGHT = (FlightN, FromCity, ToCity, Date, DepartureTime, ArrivalTime)
// FlightN is the key for this table which represents all possible direct flights. You may use <, >, !=, or = between any two dates or between any two times. Also, you may assume the attribute that ToCity and FromCity are in the same time zone. The listed flights are only direct flights between FromCity andToCity.

• TICKET = (TicketN, FlightN, Cost, Completed)
// Ticket# and Flight# together are the key, this means a single ticket may include several flights. Completed may assume the
values 'Yes' or NULL, Null means the flight hasn't been completed.

• PASSENGER = (Name , TicketN)
// Name and TicketN together are the key for this table
This table includes every passenger and the ticket
numbers they purchased. Assume the name of the passenger is unique.

Write an efficient SQL statement for each of the following queries:

a. List each FromCity, ToCity , and the number of direct flights between the two cities.
   SELECT DISTINCT FromCity, ToCity, COUNT (*)
   FROM FLIGHT

   Explanation: Query the distinct departure and arrival and count the number of flights between them

b. List each FromCity, ToCity , and the number flights between the two cities with only one stop.
   SELECT DISTINCT FromCity, ToCity, COUNT (T.TicketN)
   FROM FLIGHT F
   JOIN TICKET T ON F.FlightN = T.FlightN
   WHERE COUNT(T.FlightN) = 2

   Explanation: Query the distinct departure and arrival and count the number of tickets that are only associated with 2 flights in the TICKET table (which equal to 1 stop)

c. List every ToCity for which no ticket is purchased.
   SELECT F.ToCity

```
FROM FLIGHT F
JOIN TICKET T ON F.FlightN = T.FlightN
WHERE COUNT(TicketN) = 0
```

d. List the name of every customer who purchased tickets but did not complete any flight.

```
SELECT P.Name
FROM PASSENGER P
JOIN TICKET T ON P.TicketN = T.TicketN
WHERE T.Completed = NULL
```

e. List the name of every customer who purchased tickets to all ToCity .. i.e. to every ToCity.

```
SELECT P.Name
FROM PASSENGER P
JOIN TICKET T ON P.TicketN = T.TicketN
JOIN FLIGHT F on T.FlightN = F.FlightN
WHERE T.FlightN = F.FlightN AND COUNT(T.FlightN) = COUNT
DISTINCT(F.FlightN)
```

f. List the name of every customer and the number of flights purchased by the customer , number of flights purchased and completed by the customer, and number of flights purchased but did not completed by the customer.

```
SELECT P.Name, COUNT(T.TicketN), COUNT(T.Completed = ‘Yes’),
COUNT(T.Completed = NULL)
FROM PASSENGER P
JOIN TICKET T ON P.TicketN = T.TicketN
```

g. List every FromCity and ToCity with lowest cost flight purchased , highest cost flight purchased , average cost flight purchased, and the number of flights purchased for these two cities

```
SELECT DISTINCT F. FromCity, F. ToCity, MIN(T.Cost), MAX(T.Cost),
AVG(T.Cost), COUNT (*)
FROM Flight F
JOIN Ticket T ON F.FlightN = T.FlightN
```

Q5. (10 points) **GIVEN THE FOLOWING TABLES:**

EMPLOYEE=(EMPLOYEE-ID, SALARY)
PROJECT=(PROJECT-ID, EMPLOYEE-ID)

WRITEE A STATEMENT -LEVEL OR ROW-LEVEL AFTER INSERT TRIGGER THAT INSERTS THE EMPLOYEE-ID IN THE TABLE EMPLOYEE WITH NULL SALARY AND INSERTING A NEW PROJECT-ID AND EMPLOYEE-ID IN THE TABLE PROJECT

Assumption:
Insert every new employee onto the same project containing the same projectID of 25

Solution:
Statement Level

```
CREATE TRIGGER NewEmployeeCheck
AFTER INSERT ON EMPLOYEE
REFERENCING NEW TABLE AS NewEmployee
FOR EACH STATEMENT
WHEN (NewEmployee.Salary = NULL)
BEGIN
        INSERT INTO PROJECT (ProjectID, EmployeeID)
        VALUES (25, NewEmployee.EmployeeID)
END;
```

Row Level

```
CREATE TRIGGER NewEmployeeCheck
AFTER INSERT ON EMPLOYEE
REFERENCING NEW ROW AS NewEmployee
FOR EACH ROW
WHEN (NewEmployee.Salary = NULL)
        INSERT INTO PROJECT (ProjectID, EmployeeID)
        VALUES (25, NewEmployee.EmployeeID)
```