Midterm #1 Solution

Problem 1. (60 points) From the Las-Cruces Sun News, January 1, 3001: ...``*Best of the Best" -- a software company founded by NMSU graduates, specialized in the design and development of database applications -- is one of best software companies in the US. Its customers include several governmental agencies such as the ``Advance Home Land Security", the ``Future and Treasury", the ``Fast Transportation", the ``Basic and Advanced Education" department etc. and local businesses such as ``Cheap Home Improvement", ``Lot of Rebates Office", and ``Bank of People" of Las Cruces etc.*

``Best of the Best" has recently been contracted by the Veteran Medical Center (MC) of Las Cruces to design and develop a patient tracking system that the MC hopes to use for:

- Scheduling future appointments for patients
- Collecting fees
- Managing MC's resources (rooms, doctors, nurses, etc.)
- …

The two sides agree that such a huge undertaking should be done step-by-step. They agree that a database with information about patients, doctors, and nurses should be developed first. *Class03* -- a system analyst of ``Best of the Best" -- is instructed to get an initial design within the first two weeks of the contract. *Class03* visits the MC several times during the first week. His note about the current information flow in the MC says the following:

- The MC keeps information about each patient in a folder. Each folder is assigned a unique string, called *reference number*, of length 40 that helps to identify the patient and makes the sorting of the folders easier.

  This folder contains the following information: SSN, Name, Date of Birth, Gender, Address, Contact Phone Number, more than one Emergency Contact (each has Name and Phone number), and a list of visits (to the doctor office or to the MC's hospital) made by the patient.

  Each visit has the following information: date, a doctor, a nurse, reason, fee, and the amount the patient has paid. As a rule, the fee is always greater or equal the amount the patient has paid, which is greater than or equal 0.

  The folder of a patient is created at his/her first visit.

- The MC keeps information about each doctor in a folder. Each folder is also assigned a unique string, called *reference number*, of length 40 that helps to identify the doctor and makes the sorting of the folders easier.

  This folder contains the following information: SSN, Name, Date of Birth, Gender, Address, Contact Number, Specialty, the first working day of the doctor at MC, and the doctor's salary. MC pays its doctors between US$100,000 and US$400,000.

- The MC keeps information about each nurse in a folder. Each folder is also assigned a unique string, called *reference number*, of length 40 that helps to identify the nurse and makes the sorting of the folders easier.

  This folder contains the following information: SSN, Name, Date of Birth, Gender, Address, Contact Number, the first working day of the nurse at MC, and the nurse's salary. MC pays its nurses between US$30,000 and US$70,000.

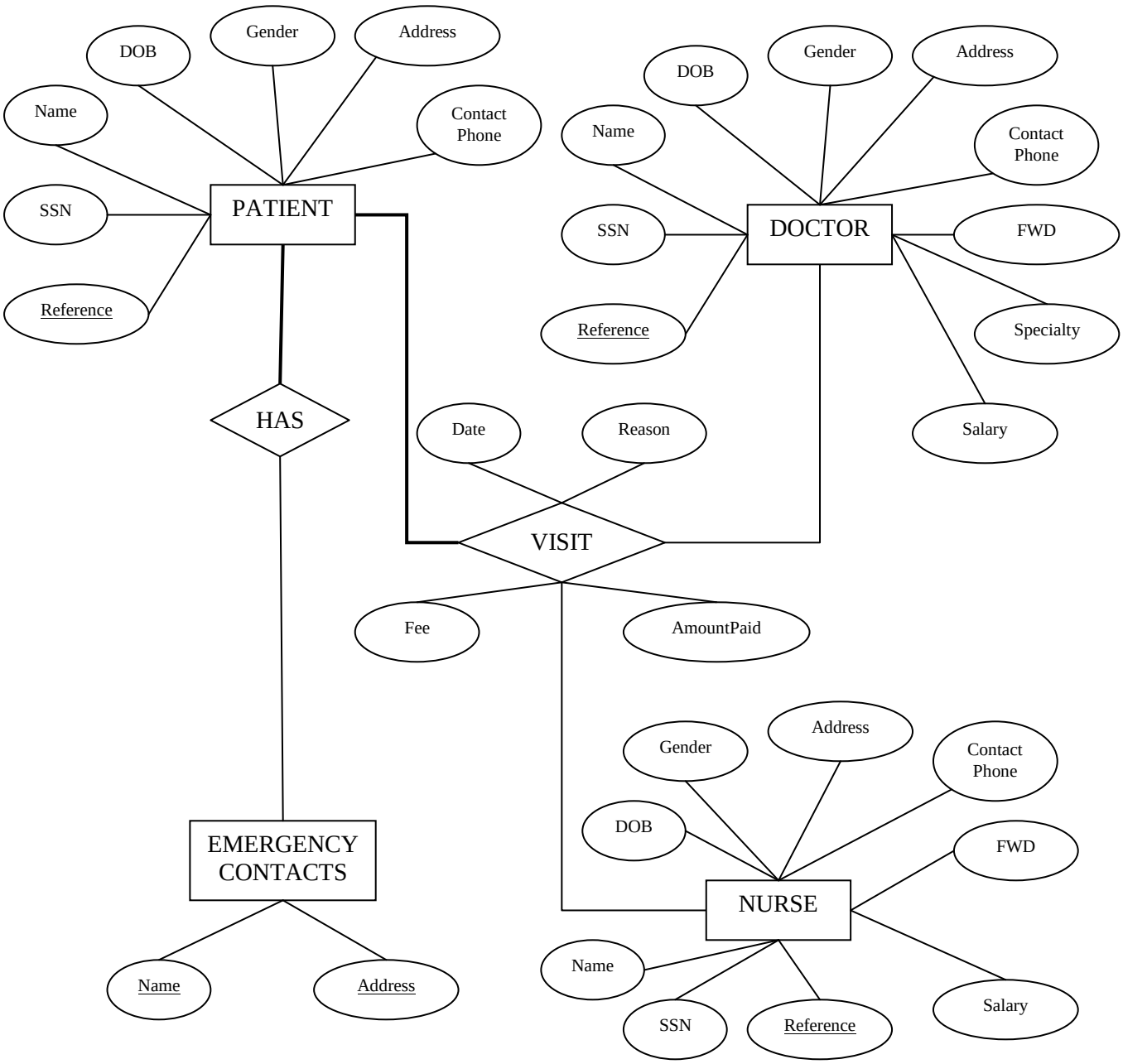
Help *Class03* in designing the database by completing the following:

1. Create an ER diagram for the information that he has collected. In your diagram, please spcify the key of entity types, the multiplicity of relationship types (one-to-many, many-to-many, etc.), and the constraints. Specify also the keys or constraints that cannot be represented in the ER diagram.

2. Translate your ER diagram into a set of relation schemas.
3. Provide the SQL (more precisely, the DDL) commands to create the relations. Be sure to include the (primary or foreign) key definition as well as the constraints.

Answer:

1. E-R Diagram *(40 points)*



Notes:

- FWD: First Working Day
- DOB: Date Of Birth

2. Relation Schema *(10 points)*

a. Relations

1. PATIENT(Reference: CHAR(40), SSN: CHAR(9), Name: CHAR(100), DOB: DATE, Gender: GENDERS, Address: CHAR(200), ContactPhone: CHAR(20))

   Key: {Reference}

2. DOCTOR(Reference: CHAR(40), SSN: CHAR(9), Name: CHAR(100), DOB: DATE, Gender: GENDERS, Address: CHAR(200), ContactPhone: CHAR(20), Specialty : CHAR(200), FWD: DATE, Salary: NUMBER(8,2))

   Key: {Reference}
   Constraint(s):
   $100{,}000 \leq \text{Salary} \leq 400{,}000$

3. NURSE(Reference: CHAR(40), SSN: CHAR(9), Name: CHAR(100), DOB: DATE, Gender: GENDERS, Address: CHAR(200), ContactPhone: CHAR(20), FWD: DATE, Salary: NUMBER(8,2))

   Key: {Reference}

   Constraint(s):
   $30{,}000 \leq \text{Salary} \leq 70{,}000$

4. VISIT(PatientRef: CHAR(40), DoctorRef: CHAR(40), NurseRef: CHAR(40), Date: DATE, Reason: CHAR(200), Fee: NUMBER(8,2), AmountPaid: NUMBER(8,2))

   Key: {PatientRef, DoctorRef, NurseRef, Date}[1]
   Foreign Keys:
   - PatientRef REFERENCES PATIENT(Reference)
   - DoctorRef REFERENCES DOCTOR(Reference)
   - NurseRef REFERENCES NURSE(Reference)

   Constraints:
   - AmountPaid ≥ 0
   - Fee ≥ AmountPaid

- HAS(PatientRef: CHAR(40), Name: CHAR(100), Address: CHAR(200))

  Key: {PatientRef, Name, Address}

  Foreign Keys:
  - PatientRef REFERENCES PATIENT(Reference)
  - (Name, Address) REFERENCES EMERGENCY_CONTACTS(Name, Address)

- EMERGENCY_CONTACTS(Name: CHAR(100), Address: CHAR(200))

  Key: {Name, Address}[2]

Note that for better performance, we only need a single table to store emergency contacts for a patient rather than use the two tables HAS and EMERGENCY_CONTACTS. As a result, the table HAS is no longer needed. The only modification needed to be made here is just to add a field to the table EMERGENCY_CONTACTS to refer to the field Reference of the table PATIENT. However, for clarity, here I'm using a straightforward way to translate an E-R diagram into the corresponding relation schema.

b. Domains

GENDERS = {'M', 'F'}

[1] Assume that a patient cannot have more than one visit per day with the same doctor and the same nurse
[2] Assume that (Name, Address) are unique among emergency contacts

3. SQL DDL *(10 points)*

a. Table PATIENT

```
CREATE TABLE PATIENT (
        Reference CHAR(40),
        SSN CHAR(9),
        Name CHAR(100),
        DOB DATE,
        Gender GENDERS,
        Address CHAR(200),
        ContactPhone CHAR(20),
        PRIMARY KEY (Reference)
)
```

b. Table DOCTOR

```
CREATABLE DOCTOR(
        Reference CHAR(40),
        SSN CHAR(9),
        Name CHAR(100),
        DOB DATE,
        Gender GENDERS,
        Address CHAR(200),
        ContactPhone CHAR(20),
        Specialty  CHAR(200),
        FWD DATE,
        Salary NUMBER(8,2),
        PRIMARY KEY (Reference),
        CHECK (100,000 ≤ Salary AND Salary ≤ 400,000)
)
```

c. Table NURSE

```
CREATABLE NURSE(
        Reference CHAR(40),
        SSN CHAR(9),
        Name CHAR(100),
        DOB DATE,
        Gender GENDERS,
        Address CHAR(200),
        ContactPhone CHAR(20),
        FWD DATE,
        Salary NUMBER(8,2),
        PRIMARY KEY (Reference),
        CHECK (30,000 ≤ Salary AND Salary ≤ 70,000)
)
```

d. Table VISIT

```
CREATE TABLE VISIT(
        PatientRef CHAR(40),
        DoctorRef CHAR(40),
        NurseRef CHAR(40),
        Date DATE, Reason CHAR(200),
        Fee NUMBER(8,2),
        AmountPaid NUMBER(8,2)
        PRIMARY KEY (PatientRef, DoctorRef, NurseRef, Date),
        FOREIGN KEY PatientRef REFERENCES PATIENT(Reference)
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
        FOREIGN KEY DoctorRef REFERENCES DOCTOR(Reference)
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
        FOREIGN KEY NurseRef REFERENCES NURSE(Reference)
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
        CHECK (AmountPaid ≥ 0),
        CHECK (Fee ≥ AmountPaid)
        )
```

e. Table HAS

```
        CREATE TABLE HAS(
                PatientRef CHAR(40),
                Name CHAR(100),
                Address CHAR(200),
                PRIMARY KEY (PatientRef, Name, Address),
                FOREIGN KEY PatientRef REFERENCES PATIENT(Reference)
                        ON DELETE NO ACTION
                        ON UPDATE CASCADE,
                FOREIGN KEY (Name, Address) REFERENCES EMERGENCY_CONTACTS(Name,
                Address)
                        ON DELETE NO ACTION
                        ON UPDATE CASCADE
        )
```

f. Table EMERGENCY_CONTACTS

```
        CREATE TABLE EMERGENCY_CONTACTS(
                Name CHAR(100),
                Address CHAR(200),
                PRIMARY KEY (Name, Address)
        )
```

g. Domain GENDERS

```
        CREATE DOMAIN GENDERS CHAR(1)
                CHECK (VALUE IN ('M','F'))
```

Problem 2. (25 points) The university theater has a reservation system that allows users to

- look up information about a show (number of seats available, title, duration, cast, etc.),
- make reservation, and
- cancel a reservation.

All inquires, reservations, and cancellations are made through the theater's website. Discuss the following:

- What would you consider as two appropriate transactions in this system? Justify your answer.
- Illustrate the desirable properties of a transaction using your transactions for this system. It will be enough if you state the properties and discuss the problems that might arise if certain property is not satisfied.

Answer:

a) Give two transactions (5 points)

Two possible transactions of the system can be:

(1) **Making a reservation.** This transaction may require the following steps (operations):

i. The user is asked to enter some required information, such as Name (or SSN), Address, and Credit card number;
ii. the system will check if there's a seat available;
iii. if any, the user will be charged for the reservation;
iv. then a ticket number will be assigned to the user by the system; and,
v. to indicate the seat is already booked by someone, the system will remove it from the list of seats available.

(2) **Canceling a reservation.** This transaction may require the following steps:

i. The user is asked to enter the ticket number just assigned to the user;
ii. the system will then check if this information is correct;
iii. if it is correct, the system will mark the seat as available by putting it back to the list of seats available; and,
iv. the money is given back to the user.

b) Illustrate the desirable properties of transactions (20 points)

We will discuss the desirable properties of the transactions in terms of atomicity, consistency, durability and isolation. In general, both transactions should have all of these properties as illustrated below.

- **Atomicity:** Both transactions should be atomic; otherwise, unexpected things may happen. For example, if the system crashes down just after step (iii) of transaction (1), the user may be charged for the reservation but he or she doesn't get the ticket number. For transaction (2), similar things may happen. If after the user's ticket is placed back in the list of seat available (step (iii)), the system suddenly crashes down due to some reason, the user cannot get his/her money back.

- **Consistency:** Assume that the system doesn't check what a valid ticket number is. When making a reservation, one may be assigned a ticket number that is invalid. As a result, he or she cannot attend the concert, although the reservation is already made. The inconsistency of transaction (2) may happen if the system doesn't check the date of the cancellation. It may be after the concert is shown.

- **Durability:** Clearly, these transactions should be durable. Otherwise, after recovery from an unexpected crash, all the seats booked may be lost for the transactions. Similar things could happen to the second transaction. For example, if a cancellation is made, the user gets back his/her money and then the system accidentally shutdowns. After recovery, all the canceling

information is lost. Consequently, the user got his/her money back while the seat is still assigned to the user.

- **Isolation:** For transaction (1), if the isolation of reservations is not guaranteed, there might be the case in which a ticket will be assigned to two users. Moreover, isolation between the two transactions above should be pointed here. For example, consider the situation in which a user is canceling a reservation and another one is trying to book a ticket. As soon as the ticket of the first user is already placed in the list of seats available, the system may assign this ticket to the second user. Afterwards, there's some reason that causes the system to rollback the cancellation of the first user. The transaction of the second user continues to be committed. So, both users are assigned the same ticket.

Problem 3. (20 points) Suppose that we have two relations *Couples* and 2Sisters with the schema *Couples(Husband,Wife)* and *2Sisters(Older,Younger)*, respectively. In other words, the relation *Couples* consists of tuples of the from *(h,w)* where *h* is the husband and *w* is the wife and the relation *2Sisters* consists of tuples of the form *(o,y)* where *o* and *y* are two sisters, *o* is older than y. Answer the following:

- Can the first attribute in the relation *Couples*(i.e., the husband) be a candidate key of the relation? Justify your answer.
- Can the first attribute in the relation *2Sisters* (i.e., the older sister) be a candidate key of the relation? Justify your answer.
- Give a candidate key for the relation *Couples*.
- Give a SQL query that gets all tuples of the form *(h,y)* where *y* is a younger sister-in-law of *h*.

Answer:

a) (5 points) No. For example, a man divorced his wife and then gets married with another woman. So, he has two records in the database, one for the first wife, and the other for the second wife.

b) (5 points) No. Let's see the following example. A has two younger sisters: B and C. So the table 2Sisters contains two records: (A,B) and (A,C). That's why the Older cannot be a candidate key.

c) (5 points) (Husband,Wife) can be a candidate key if we assume that a man has only one wife and vice versa. Additionally, a person (either Husband or Wife) has a unique identity.

d) (5 points) SQL query:

```
SELECT Couples.Husband AS[3] h, 2Sisters.Younger AS y
FROM Couples, 2Sisters
WHERE Couples.Wife = 2Sisters.Older
```

[3] AS is used for aliasing