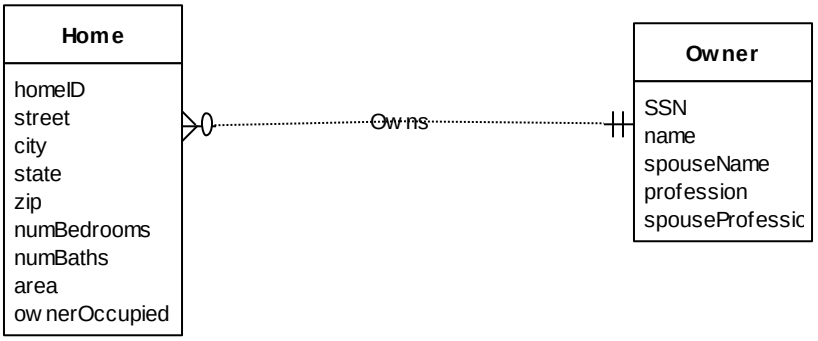
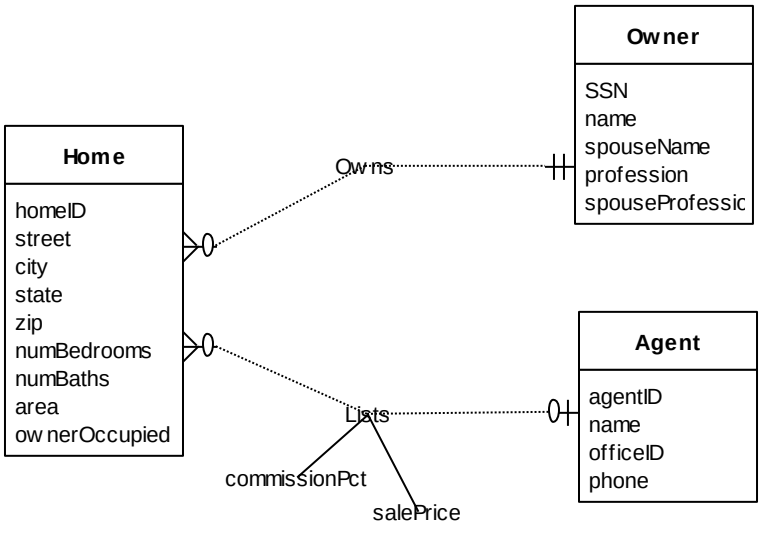
CIS/CSC384

HomeWork 5 Solution

1. Come up with an ER Diagram for the following narrative. The database should track homes and owners. A home has a unique home identifier, a street address, a city, a state, a zip, a number of bedrooms, a number of bathrooms and square feet. A home is either owner occupied or rented. An owner has a SSN, a name, an optional spouse name, a profession, and an optional spouse profession. An owner can possess zero or more homes. Each home has only one owner.

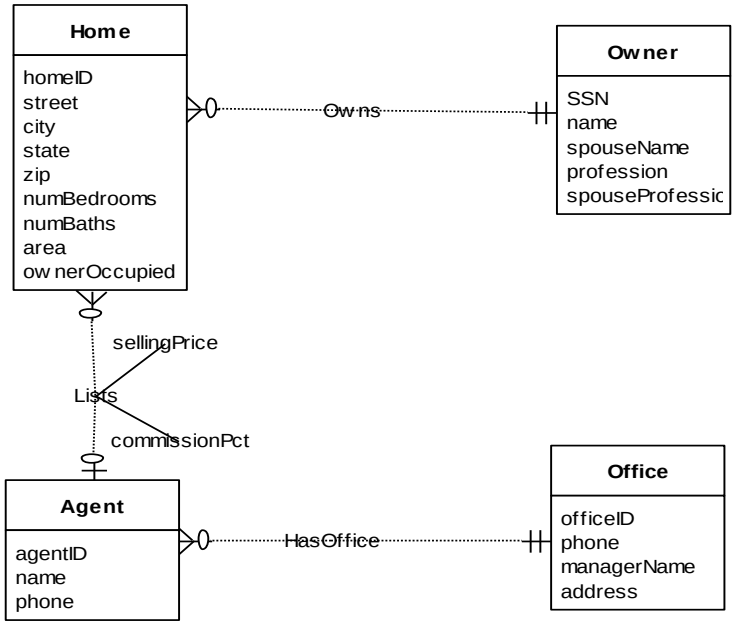


2. Refine the ER Diagram from (1) by adding an agent entity type. Agents represent owners in the sale of a home. An agent can list many homes, but only one agent can list a home. An agent has a unique agent identifier, a name, an office identifier, and a phone number. When an owner agrees to list a home with an agent, a commission (percentage of the sale price) and a selling price are determined.



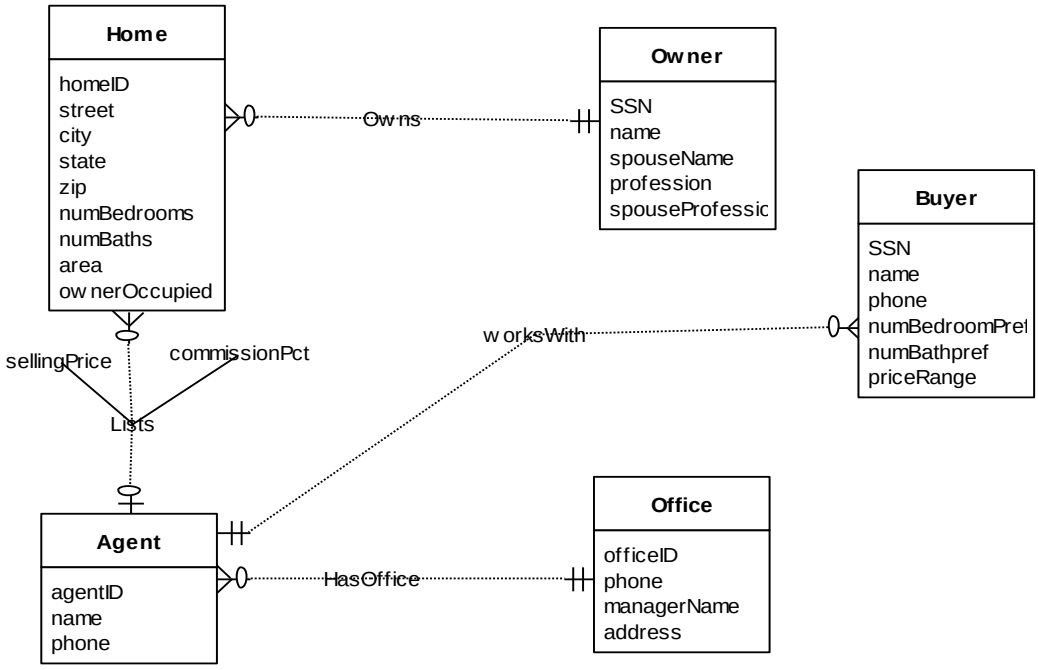
Note: commissionPct and salePrice can also be specified as attributes of Home. A home can have 0 or 1 agent.

3. Refine the ER Diagram from (2), transform the attribute office identifier, into an entity type. Data about an office include the phone number, the manager name, and the address.



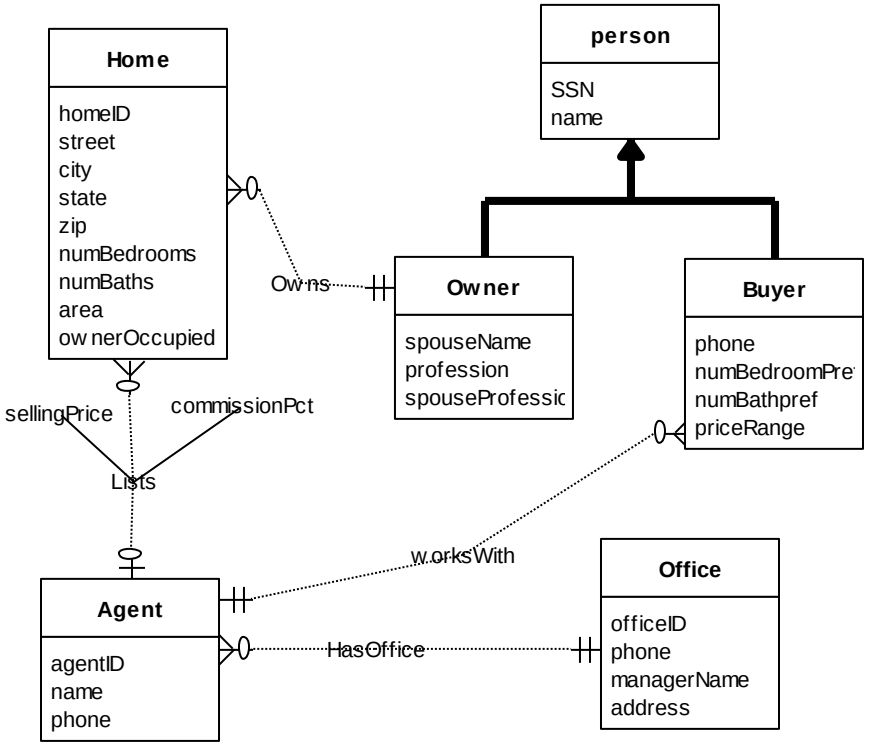
Note: I assume that an office can have multiple agents. The cardinality of Office must either be (1, 1) (as shown) or it can also be (0, 1).

4. In the ER Diagram from (3), add a buyer entity type. A buyer entity type has a SSN, a name, a phone, preferences for the number of bedrooms and bathrooms, and a price range. An agent can work with many buyers, but a buyer works with only one agent.



Note: A price range can also be represented as 2 attributes, the lower limit, and the upper limit.

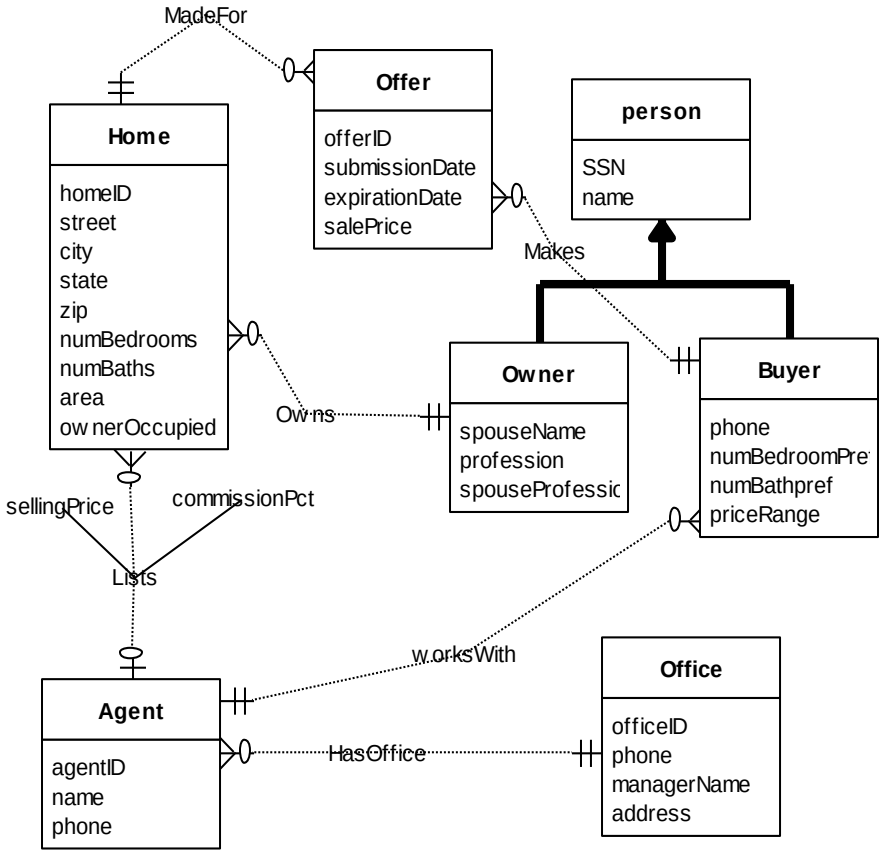
5. Refine the ER Diagram from (4) with a generalization hierarchy to depict similarities between buyers and owners.



Note: No Disjointness, Completeness constraints are assumed for person – owner/buyer generalization hierarchy. We never specified phone for owner, so the common attributes are only ssn and name.

6. Revise the ER Diagram from (5) by adding an offer entity type. A buyer makes an offer on a home for a specified sales price. The offer starts on the submission date, and expires on the specified date. A unique number identifies an offer. A buyer can submit multiple offers for the same home.

   Note: A buyer making multiple offers for the same home is automatically captured. What you must not do is use a m-n binary relationship to denote offer between home and buyer. If I choose the m-n binary relationship between home and buyer to represent offer, then we cannot have one buyer having multiple offers for the same home.



7. Come up with SQL CREATE TABLE statements for the final ER diagram in (6). You must check your set of SQL CREATE TABLE statements will execute against Oracle (otherwise, report what errors you get).

```
CREATE TABLE Office (officeID varchar(10) PRIMARY KEY,
                     phone varchar(15),
                     managerName varchar(25),
                     address varchar(50));

CREATE TABLE Agent (agentID varchar(10) PRIMARY KEY,
                    name varchar(15),
                    phone varchar(15),
                    officeID varchar(10) NOT NULL REFERENCES Office(OfficeID));

-- The HasOffice Relationship is represented using the Agent.OfficeID foreign key
-- It could be represented using a separate table also..
-- NOT NULL captures cardinality of Office in HasOffice is mandatory.
```

```
CREATE TABLE Person (SSN varchar(15) PRIMARY KEY, name varchar(25));

CREATE TABLE Owner (SSN varchar(15) PRIMARY KEY REFERENCES
Person(SSN),
                    spouseName varchar(25),
                    profession varchar(25),
                    spouseProfession varchar(25));

CREATE TABLE Buyer (SSN varchar(15) PRIMARY KEY REFERENCES
Person(SSN),
                    phone varchar(15),
                    numBedroomPref int,
                    numBathPref int,
                    priceRange varchar(25),
                    agentID varchar(10) NOT NULL REFERENCES Agent(agentID));

-- worksWith Relationship is represented using the Buyer.agentID foreign key
-- can be represented using a separate table also..
-- cardinality of Agent in worksWith is mandatory. represented using NOT NULL.

CREATE TABLE Home (homeID varchar(10) PRIMARY KEY,
                    street varchar(25),
                    city varchar(15),
                    state char(2),
                    zip varchar(10),
                    numBedrooms int,
                    numBaths int,
                    area float,
                    ownerOccupied int,
                    sellingPrice float,
                    commissionPct float,
                    ownerSSN varchar(15) NOT NULL REFERENCES Owner(SSN),
                    agentID varchar(10) REFERENCES Agent(agentID));

-- Owns relationship is represented using the OwnerSSN foreign key.
-- OwerSSN is NOT NULL ensuring cardinality is mandatory for Owner in Owns.
-- Lists relationship is represented using the agentID foreign key
-- See that cardinality for agent is optional in Lists. Therefore it can be NULL.
-- Both relationships can be represented using separate tables also.
```

```
CREATE TABLE Offer (OfferID varchar(10) PRIMARY KEY,
                    submissionDate DATE,
                    expirationDate DATE,
                    salePrice float,
                    buyerSSN varchar(15) NOT NULL REFERENCES Buyer(SSN),
                    homeID varchar(10) NOT NULL REFERENCES Home(homeID));

-- Makes relationship is represented using buyerSSN foreign key.
-- MadeFor relationship is represented using homeID foreign key.
-- Both relationships can be represented using separate tables also.
-- The cardinality for Buyer is mandatory in relationship Makes. Captured using NOT
--  NULL.
-- Similarly the cardinality for Home is mandatory in MadeFor relationship. Captured
-- using NOT NULL.
```

Grading: If it does not work against Oracle, and the student has not tested the working against Oracle, decrease half the points.

8. For the following problem, come up with an ER diagram for the initial requirements, and then revise the ER diagram for the new requirements. Your solution must include an initial ER Diagram, and a revised ER diagram. [1 + 1 pts]

   The database is to assist physical plant personnel in managing assignments of keys to employees.

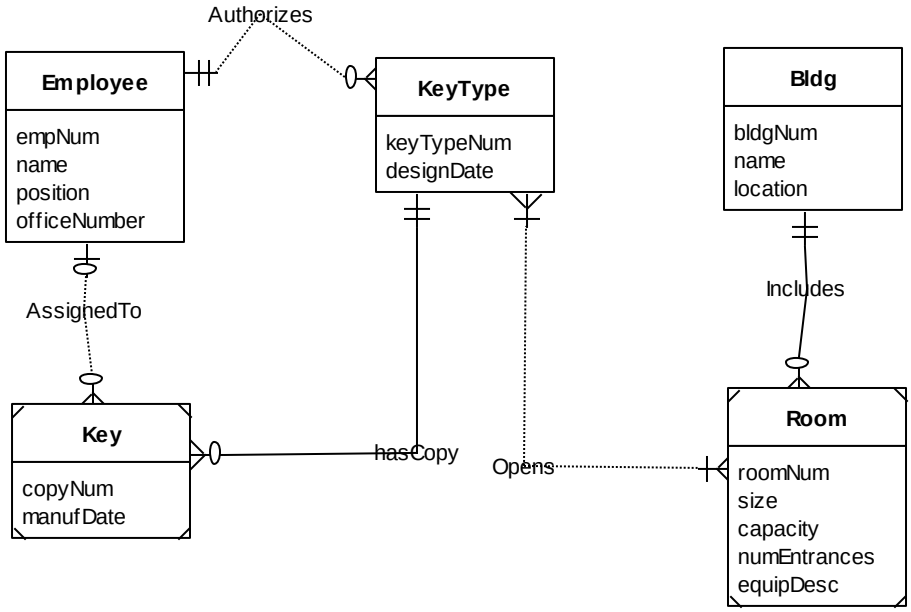
   - An employee has a unique emp number, a name, a position and an optional office number.
   - A building has a unique building number, a name, and a location within the campus.
   - A room has a room number, a size, a capacity, a number of entrances, and a description of equipment in the room. Because each room is located in exactly one building, the identification of a room depends on the identification of the building.
   - Key types(also known as master keys) are designed to open one or more rooms. A room may have one or more key types that open it. A key type has a unique key type number, a date designed, and the employee authorizing the key type. A key type must be authorized before it is created.
   - A copy of a key type is known as a key. Keys are assigned to employees. Each key is assigned to exactly one employee, an employee can hold multiple keys. The key type number plus a copy number uniquely identifies a key. The date the copy was made is recorded in the database.

   After reviewing the intial design, the supervisor decides to revise the requirements as follows. Show the ER diagram before the revisions, and after the revisions.

- The physical plant needs to know not only the current holder of a key, but the past key holders. For past key holders, the date range that a key was held must be recorded.
- The physical plant needs to know the current status of each key: in use by an employee, in storage or reported lost. If lost, the date reported lost should be stored.

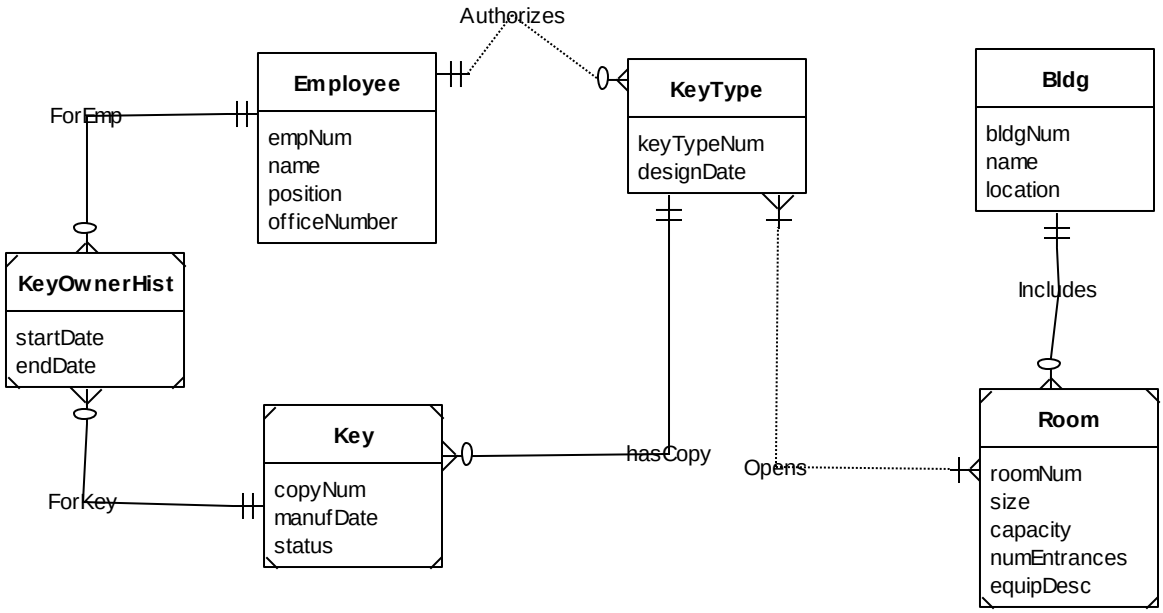
Basic ER Diagram:

Notes: officeNumber is optional for Employee
A keyType must be authorized before it is designed is captured in the ER, when we say that the cardinality of Employee in the Authorizes relationship is Mandatory.
I am going to assume that a key is assigned to 0 or 1 employee. (though the problem says a key is assigned to exactly one employee)



Revised ER Diagram:

Notes: Status of a key is specified using an attribute for the key entity type.



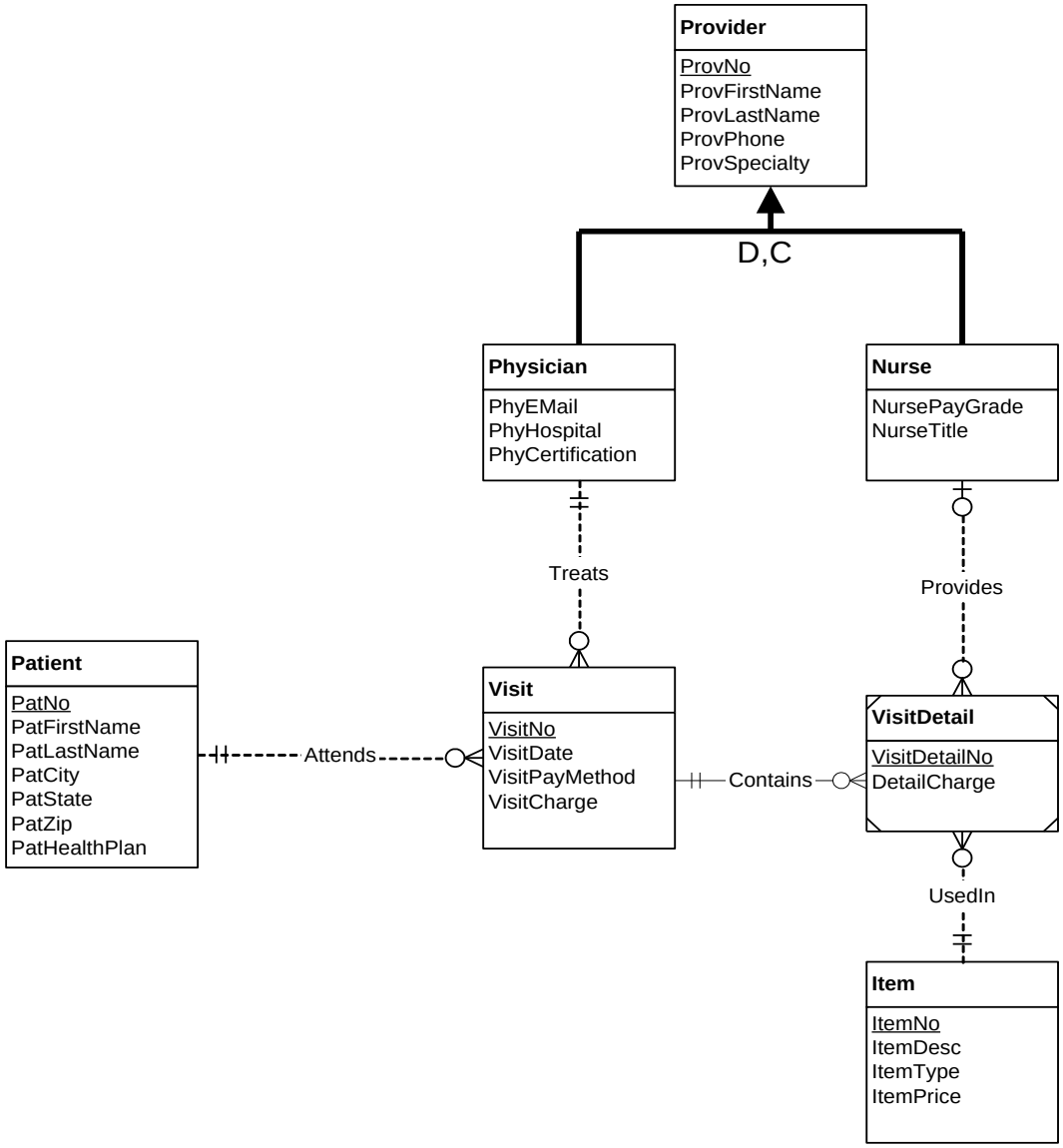
9. Translate the above revised ER diagram into SQL create table statements. You MUST check the SQL against Oracle, and report whether it succeeded/there were errors. [2 pts]

Note: If the student has not tested against Oracle, then decrease 1 point. If the student has tested and report errors that he/she encountered, that is fine.. Note that I cannot have a column called size in SQL, so I change it to roomSize (for table Room).

```
CREATE TABLE Employee (empNum varchar(10) PRIMARY KEY,
                       name varchar(25),
                       position varchar(15),
                       officeNum varchar(15));

CREATE TABLE Bldg (bldgNum varchar(10) PRIMARY KEY,
                   name varchar(25),
                   location varchar(25));

CREATE TABLE Room (roomNum varchar(10),
                   bldgNum varchar(10) REFERENCES Bldg(bldgNum),
                   roomSize varchar(10),
                   capacity int,
                   numEntrances int,
                   equipDesc varchar(100),
                   PRIMARY KEY (roomNum, bldgNum));

-- Note that bldgNum is part of PRIMARY KEY for Room. This makes sure that
-- bldgNum cannot be NULL, i.e., there is 1 bldg for every room.
```

```
CREATE TABLE KeyType (keyTypeNum varchar(10) PRIMARY KEY,
                      designDate DATE);

CREATE TABLE Opens (keyTypeNum varchar(10) REFERENCES KeyType(keyTypeNum),
                    roomNum varchar(10),
                    bldgNum varchar(10),
                    empNum varchar(10) NOT NULL REFERENCES Employee(empNum),
                    PRIMARY KEY (roomNum, bldgNum, keyTypeNum),
                    FOREIGN KEY (roomNum, bldgNum) REFERENCES
                          Room(roomNum, bldgNum));

-- Note that this is a m-n relationship. Therefore we MUST specify it as a separate table.
-- Also see that the foreign key to room must include the complete key of room,
-- which is (roomNum, bldgNum)
-- Also see that the employee authorizing key is specified using empNum which is NOT NULL
-- Therefore every key MUST be authorized by an employee.
-- Note that a KeyType opens at least 1 room is not captured in SQL
-- Note that a Room has at least 1 KeyType is not captured in SQL.
-- These 2 are captured in ER using minimum cardinality of mandatory.

CREATE TABLE Key (copyNum varchar(10),
                  keyTypeNum varchar(10) REFERENCES KeyType(keyTypeNum),
                  manufDate DATE,
                  status varchar(15),
                  PRIMARY KEY (copyNum, keyTypeNum));

-- Note that keyTypeNum is part of PRIMARY KEY for Key.

CREATE TABLE KeyOwnerHist (copyNum varchar(10),
                           keyTypeNum varchar(10),
                           empNum varchar(10) REFERENCES Employee(empNum),
                           startDate DATE,
                           endDate DATE,
                           PRIMARY KEY (copyNum, keyTypeNum, empNum),
                           FOREIGN KEY (copyNum, keyTypeNum) REFERENCES
                                 Key(copyNum, keyTypeNum));

-- Note that our ER diagram and our SQL specifies that an emp cannot have a key 2 different
-- intervals. i.e., an employee having a key now, then giving it up, and then getting it back again
--  is not allowed. If this is to be allowed, we need additional attributes for KeyOwnerHist
```

10. Consider the ER Diagram for the patient visiting hospitals from Homework 2. Start from the ER diagram in the posted solution, which is given below. Translate the ER Diagram into SQL CREATE TABLE statements. You MUST check the SQL against Oracle, and report whether it succeeded/there were errors. [2 pts]




```
CREATE TABLE Provider (provNum varchar(10) PRIMARY KEY,
                provFName varchar(10),
                provLName varchar(10),
                provPHone varchar(15),
                provSpecialty varchar(15));

CREATE TABLE Physician (phyNum varchar(10) PRIMARY KEY
                                REFERENCES Provider(provNum),
                PhyEmail varchar(20),
                PhyHospital varchar(20),
                PhyCert varchar(10));

CREATE TABLE Nurse (nurseNum varchar(10) PRIMARY KEY
                                REFERENCES Provider(provNum),
```

```
                nursePayGrade varchar(10),
                nurseTitle varchar(10));

CREATE TABLE Patient (patNum varchar(10) PRIMARY KEY,
                  patFName varchar(10),
                  patLName varchar(10),
                  patCity varchar(10),
                  patState char(2),
                  patZip varchar(10),
                  patHealthPlan varchar(10));

CREATE TABLE Visit (visitNum varchar(10) PRIMARY KEY,
                visitDate DATE,
                visitPayMethod varchar(10),
                vistiCharge int,
                patNum varchar(10) NOT NULL REFERENCES Patient(patNum),
                phyNum varchar(10) NOT NULL REFERENCES Physician(phyNum));

-- Note that the NOT NULL captures that a visit must include 1 patient, and 1 Physician.
-- i.e. the mandatory cardinality for Patient and Physician.

CREATE TABLE Item (itemNum varchar(10) PRIMARY KEY,
               itemDesc varchar(25),
               itemType varchar(10),
               itemPrice float);

CREATE TABLE VisitDetail (visitDetailNum varchar(10),
                     visitNum varchar(10) REFERENCES Visit(visitNum),
                     detailCharge varchar(100),
                     nurseNum varchar(10) REFERENCES Nurse(nurseNum),
                     itemNum varchar(10) NOT NULL REFERENCES Item(itemNum),
                     PRIMARY KEY (visitDetailNum, visitNum));

-- Note that the primary key for VisitDetail must include visitNum (because Contains
-- is the identifying relationship).
-- Note that nurseNum can be NULL, because the cardinality is optional.
-- itemNum is NOT NULL, because the cardinality is mandatory.
```