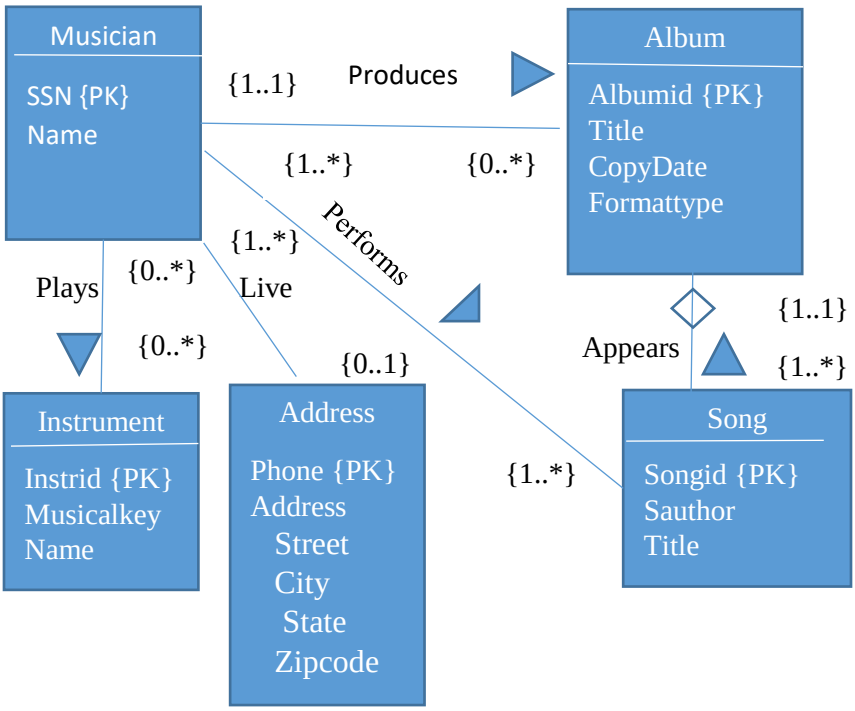
1) Notown Records has decided to store information about musicians who perform on its albums (as well as other company data) in a database. The company has wisely chosen to hire you as a database designer (at your usual consulting fee of $2500/day).

- Each musician that records at Notown has an SSN, a name, an address, and a phone number. Poorly paid musicians do not have cell phones, often share the same address, and no address has more than one phone.
- Each instrument used in songs recorded at Notown has a unique identification number, a name (e.g., guitar, synthesizer, flute) and a musical key (e.g., C, B-flat, E-flat).
- Each album recorded on the Notown label has a unique identification number, a title, a copyright date, a format (e.g., CD or MC), and an album identifier.
- Each song recorded at Notown has a title and an author.
- Each musician may play several instruments, and a given instrument may be played by several musicians.
- Each album has a number of songs on it, but no song may appear on more than one album.
- Each song is performed by one or more musicians, and a musician may perform a number of songs.
- Each album has exactly one musician who acts as its producer. A musician may produce several albums, of course.

Design a conceptual schema for Notown and draw an UML diagram for your schema. Be sure to indicate all key and cardinality constraints and any assumptions you make. Identify any constraints you are unable to capture in the ER diagram and briefly explain why you could not express them. Once you have created the diagram, create the necessary SQL CREATE TABLE commands necessary to support it.



SQL :

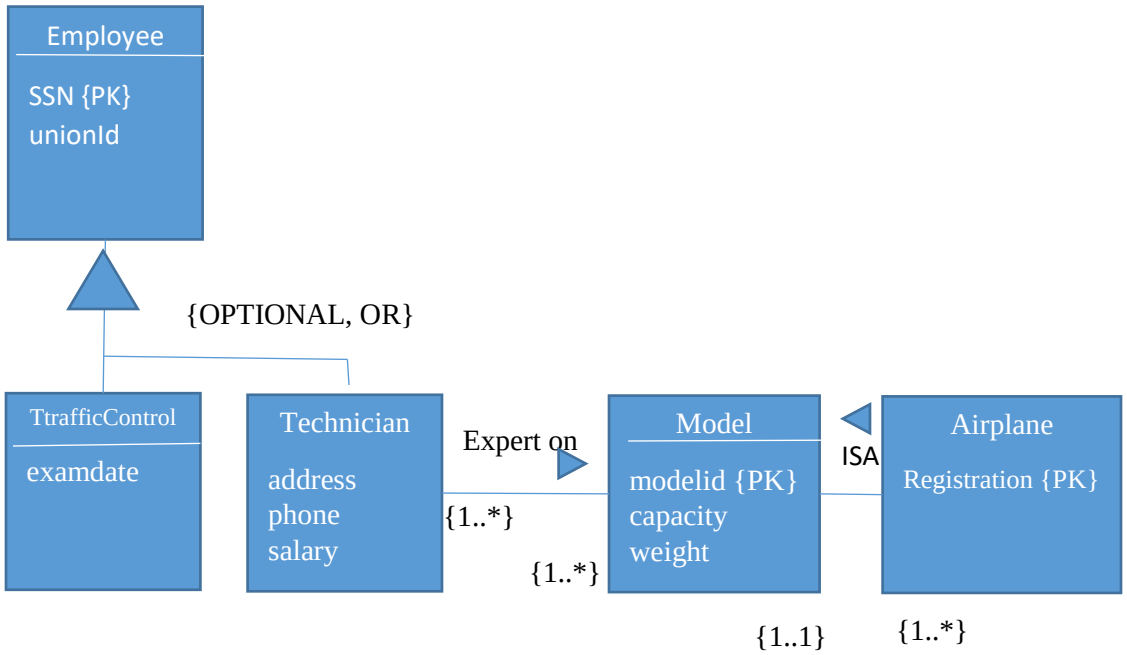
```
CREATE TABLE musician
          ( ssn INT PRIMARY KEY,
            name VARCHAR(64) NOT NULL,
             phone CHAR(11),
             CONSTRAINT musician_address_fk FOREIGN KEY  (phone) REFERENCES address(phone)
                    ON DELETE SET NULL,
                    ON UPDATE SET NULL
            );
CREATE TABLE address

              ( phone CHAR(11) PRIMARY KEY,
               Street VARCHAR(64) NOT NULL ,
                City VARCHAR(64) NOT NULL,
                State CHAR(2) NOT NULL,
              );


 CREATE TABLE instrument
          ( instrumentid INT PRIMARY KEY,
            name VARCHAR(64) NOT NULL,
            musicalkey VARCHAR(64)
            );
```

```
CREATE TABLE album
        ( albumid INT AUTO_INCREMENT PRIMARY KEY,
          releasedate DATE NOT NULL,
          formattype char(8) NOT NULL,
          producer INT NOT NULL,
          FOREIGN KEY (producer) REFERENCES musician(ssn)
            ON UPDATE CASCADE ON DELETE CASCADE
          );


CREATE TABLE song
        ( songid INT AUTO_INCREMENT PRIMARY KEY,
          title VARCHAR(128) NOT NULL,
          author INT NOT NULL,
          albumid INT NOT NULL,
          FOREIGN KEY (albumid) REFERENCES album(albumid)
            ON UPDATE RESTRICT ON DELETE RESTRICT,
          FOREIGN KEY (author) REFERENCES musician(ssn)
            ON UPDATE RESTRICT ON DELETE RESTRICT
          );
-- mapping tables to support the multiple artists on a song

CREATE TABLE performs
          (artist INT,
           song INT,
           PRIMARY KEY (artist,song),
           FOREIGN KEY (artist) REFERENCES musician(ssn)
             ON UPDATE RESTRICT ON DELETE RESTRICT,
           FOREIGN KEY (song) REFERENCES song(songid)
             ON UPDATE RESTRICT ON DELETE RESTRICT
                  );
-- mapping table to support the multiple instruments a musician can play

CREATE TABLE musiciantoinstrument
          ( instrumentid INT,
           artist INT,
           PRIMARY KEY (instrumentid,artist),
           FOREIGN KEY (instrumentid) REFERENCES instrument(instrumentid)
               ON UPDATE RESTRICT ON DELETE RESTRICT,
          FOREIGN KEY (artist) REFERENCES musician(ssn)
               ON UPDATE RESTRICT ON DELETE RESTRICT
          );
```

2) The Computer Science Department frequent fliers have been complaining to Dane County Airport officials about the poor organization at the airport. As a result, the officials decided that all information related to the airport should be organized using a DBMS, and you have been hired to design the database. Your first task is to organize the information about all the airplanes stationed and maintained at the airport. The relevant information is as follows:

- Every airplane has a registration number, and each airplane is of a specific model.
- The airport accommodates a number of airplane models, and each model is identified by a model number (e.g., DC-10) and has a capacity and a weight.
- A number of technicians work at the airport. You need to store the name, SSN, address, phone number, and salary of each technician.
- Each technician is an expert on one or more plane model(s), and his or her expertise may overlap with that of other technicians. This information about technicians must also be recorded.
- Traffic controllers must have an annual medical examination. For each traffic controller, you must store the date of the most recent exam.
- All airport employees (including technicians) belong to a union. You must store the union membership number of each employee. You can assume that each employee is uniquely identified by a social security number.

Design a conceptual schema for the airport and draw an UML diagram for your schema. Be sure to indicate all key and cardinality constraints and any assumptions you make. Once you have created the diagram, create the necessary SQL CREATE TABLE commands necessary to support it.



```
CREATE TABLE employee
        ( ssn INT PRIMARY KEY,
          unionid INT
          );

CREATE TABLE traffic_controller

        ( ssn INT PRIMARY KEY,
          physicalexam DATE NOT NULL,
          FOREIGN KEY (ssn) REFERENCES employee(ssn)
            ON UPDATE CASCADE ON DELETE CASCADE
          );

CREATE TABLE technician

        ( ssn INT PRIMARY KEY,
          address VARCHAR(128),
          salary DECIMAL(10,2),
          phone CHAR(10),
          FOREIGN KEY (ssn) REFERENCES employee(ssn)
            ON UPDATE CASCADE ON DELETE CASCADE
          );

 CREATE TABLE model
          ( modelid INT PRIMARY KEY,
          capacity DECIMAL(10,2) NOT NULL,
          weight DECIMAL(10,2) NOT NULL
          );

 CREATE TABLE airplane
        ( registration INT PRIMARY KEY,
          modeltype INT NOT NULL,
          FOREIGN KEY (modeltype) REFERENCES model(modelid)
            ON UPDATE CASCADE ON DELETE CASCADE
          );

CREATE TABLE expertise
        ( ssn INT,
          modelid INT,
```

```
PRIMARY KEY (ssn,modelid),
FOREIGN KEY (ssn) REFERENCES technician(ssn)
  ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (modelid) REFERENCES model(modelid)
  ON UPDATE CASCADE ON DELETE CASCADE
);
```