Jay Ganguli, 1001575191

Sayantan Chattopadhyay, 1000818145

**CSC343 Assignment 1**

**Question 1:**

**Logical data independence** is where users are protected from changes in the logical structure of the data.

**Physical data independence** is where the logical structure of the database protected from changes the way data is physically stored.

These properties protect programs from changes in the way data is structured and stored. We can alter details about the way data is handled without having to alter the applications using that data.

For example, say we have a an entity Bicycles with a model id, name, price, and manufacturer. Now we want to also store information about the manufacturers, but we don’t want to touch the software that queries the current information. So instead we decide to create a new table Manufacturers with a unique id, name, and other info we want to check. Then we can simply set a foreign key constraint on manufacturers in the Bicycles table.

**Question 2:**

1. Foreign Keys:

**Enrolled[sid] ⊆ Students[sid]**

Each sid in ‘Enrolled’ should correspond to an existing sid in ‘Students’. If a student is removed from the ‘Student’ table, the rows in ‘Enrolled’ with a corresponding sid should be removed.

**Enrolled[cid] ⊆ Courses[cid]**

Each cid should always point to existing courses. If a course is removed from the ‘Courses’ table, each row in ‘Enrolled’ with a corresponding cid should be removed.

**Teaches[fid] ⊆ Faculty[fid]**

Similarly to the above, if a member is removed from the ‘Faculty’ table, the rows in ‘Teaches’ with a corresponding fid should be removed.

**Teaches[cid] ⊆ Courses[cid]**

If a course is removed from the ‘Courses’ table, the corresponding rows should be removed from the ‘Teaches’ relation as well.

**MeetsIn[cid] ⊆ Courses[cid]**

If a course is removed from the ‘Courses’ table, they wouldn’t be meeting anywhere, so the rows corresponding to that course should not be in the ‘MeetsIn’ relation.

**MeetsIn[rno] ⊆ Rooms[rno]**

Again, rno should always point to a real room in the ‘Rooms’ table.

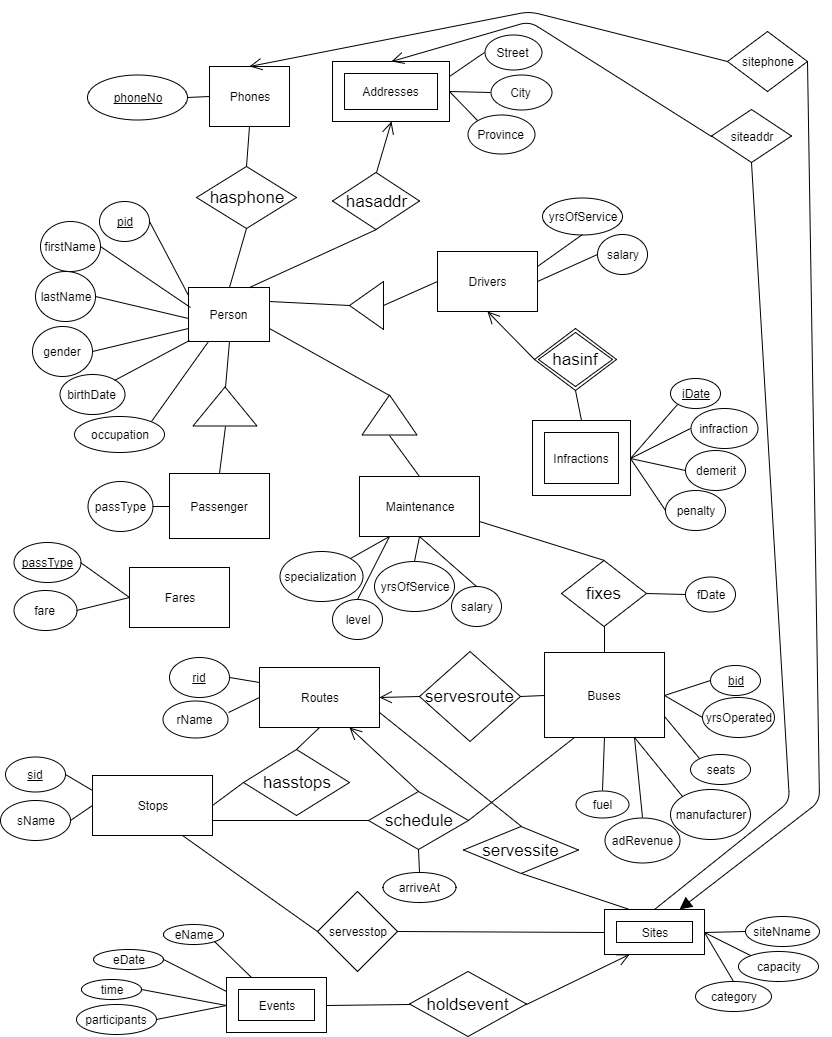
b) Plausible additional constraint:

Students[login] should be unique. In other words, there should be a one-to-one relationship between Students[login] and Students[sid]. Otherwise, multiple students having the same login could be disastrous.

(Part 3 on next page)

**Question 3:**

a)

****

b)

Person is an entity with attributes (pid: int, firstName: string, lastName: string, gender: string, birthdate: date, occupation: string) where pid is unique, and has a many-to-many relationship with Phones and a many-to-one relationship with Addresses. The attribute ‘gender’ must have a value of either “Male” or “Female”.

Phones is an entity with a unique attribute (phoneNo: string), and has a many-to-many relationship with Persons and a one-to-many relationship with Sites.

Addresses is a weak entity with attributes (street: string, city: string, province: string), and has a one-to-many relationship with Persons and a one-to-many relationship with Sites.

Passenger is a subclass entity with an attribute (passType: string) and inheriting attributes from Persons. passType is a foreign key from Fares.

Fares is an entity with attributes (passType: string, fare: real) where passType is unique.

Drivers is a subclass entity with attributes (yrsOfService: int, salary: real) and inheriting attributes from Persons. It has a one-to-many relationship with Infractions.

Infractions is a weak entity with attributes (iDate: date, infraction: string, demerit: int, penalty: real) and has a many-to-one relationship with Drivers. It’s key is the combination {Drivers.pid, Infractions.iDate} since a Driver can have at most one infraction per day.

Maintenance is a subclass entity with attributes (yrsOfService: int, salary: real, level: string, specialization: string) and inheriting attributes from Persons. It has a many-to-many relationship with Buses. Each instance if this relationship is associated an attribute (fDate: date).

Buses is an entity with attributes (bid: int, yrsOperated: int, seats: int, manufacturer: string, adRevenue: real, fuel: string) where bid is unique, and has a many-to-many relationship with Maintenance, a many-to-one relationship with Routes and a one-to-many relationship with Stops. The latter relationship (named “schedule”) also contains an attribute (arrivesAt: DATETIME) which represents a time from 6am to 11pm between May 1, 2017 and May 7, 2017.

Routes is an entity with attributes (rid: int, rName: string) where rid is unique, and has one-to-many relationship with Buses, a many-to-many relationship with Stops, a one-to-many relationship in “schedule” and an optional many-to-many relationship with Sites.

Stops is an entity with attributes (sid: int, sName: string) where sid is unique, and has a many-to-many relationship with Routes, a many-to-many relationship in “schedule” and a many-to-many relationship with Sites.

Sites is a weak entity with attributes (siteName: string, capacity: int, category: string) and has a one-to-one relationship with Phones, a many-to-one relationship with Addresses, a many-to-many relationship with Routes, a many-to-many relationship with Stops and a one-to-many relationship with Events.

Events is a weak entity with attributes (eName: string, eDate: date, eTime: TIME, participants: int) and has a many-to-one relationship with Sites.