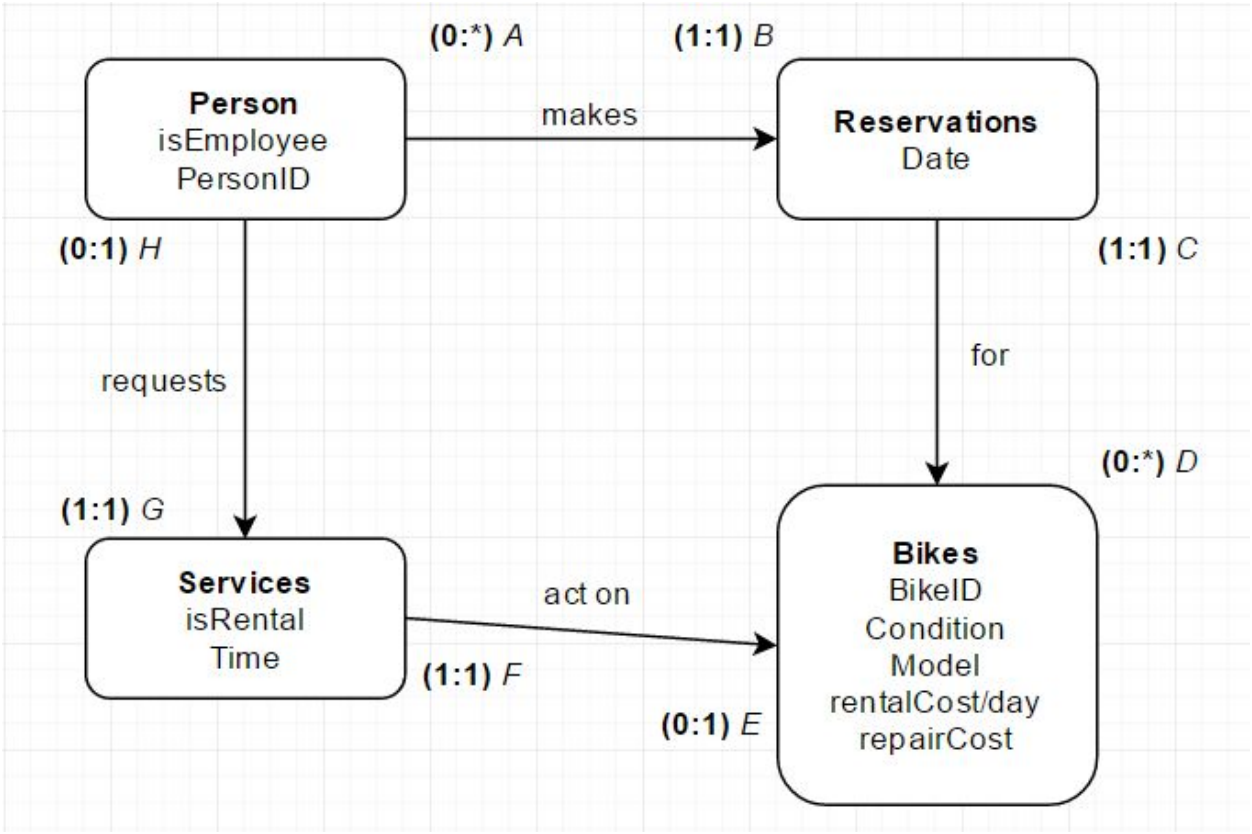
Sam Christensen
Lakshman Wadhwani



1.

ER Diagram

No foreign keys listed on diagram. They are listed in the schema provided below. Almost all access will be from Reservations and Services. Tables Bikes and Person act more like inventory to create reservations and services. I have included italicized letters next to my modalities so I would be able to justify my decisions below.

- **A.** A single Person may have many Reservations. The 0 means that each Person does not require a Reservation.
- **B.** A Reservation may only be assigned to a single Person.
- **C.** A single Reservation may only reserve a single bike.
- **D.** A Bike may have many Reservations (for different times), The 0 means that each Bike does not require a Reservation.
- **E.** A Bike may have only one service being performed on it. The 0 means that each Bike does not require a Service.
- **F.** A single Service may only act on a single Bike.
- **G.** A single Service may only be performed by a single Person.
- **H.** A single Person may only perform a single Service. The 0 means that each Person does not require a Service.

Sam Christensen
Lakshman Wadhwani

2.

Schema

#comments should provide clarification on attributes

```
CREATE TABLE Person (
        PersonID                int PRIMARY KEY,
        isEmployee              BIT DEFAULT 0
);
#isEmployee, 0 = Customer / 1 = Employee

CREATE TABLE Reservations (
        BikeID                  int FOREIGN KEY REFERENCES Bikes(BikeID),
        PersonID                int FOREIGN KEY REFERENCES Person(PersonID),
        Date                    varchar(255)
);

CREATE TABLE Bikes (
        BikeID                  int PRIMARY KEY,
        Condition               varchar(255),
        Model                   varchar(255),
        rentalCost/day          float,
        repairCost              float
);

CREATE TABLE Services (
        PersonID                int FOREIGN KEY REFERENCES Person(PersonID),
        BikeID                  int FOREIGN KEY REFERENCES Bikes(BikeID),
        isRental                BIT DEFAULT 1,
        Time                    TIME
);
#isRental, 0 = Repair / 1 = Rental
#Time, if isRental = 0 then timeBeingRepairedUntil
#Time, if isRental = 1 then timeRentedOutUntil
```

List all functional dependencies

Person			
PersonID	determines	isEmployee	#PersonID is the candidate key
Reservations			
BikeID and Date	determines	PersonID	#Only 1 person can reserve a bike for a certain time #BikeID and Date are the candidate key
Bikes			
BikeID	determines	Condition	#BikeID is the candidate key
BikeID	determines	Model	
BikeID	determines	rentalCost/day	
BikeID	determines	repairCost	
Services			
BikeID	determines	isRental	#Each bike can only be tied to a single service
BikeID	determines	Time	#Each bike can only be serviced for some time #BikeID is the candidate key

Sam Christensen
Lakshman Wadhwani

Check for BCNF or 3NF

Person

Because PersonID is the candidate key and is also on the L.H.S of the determines statement this relation is BCNF.

Reservations

Because BikeID and Date make up the candidate key and is also on the L.H.S of the determines statement this relation is BCNF.

Bikes

Because BikeID is the candidate key and is also on the L.H.S of the determines statement this relation is BCNF.

Services

Because BikeID is the candidate key and is also on the L.H.S of the determines statement this relation is BCNF.

Update/Delete/Insert anomalies

While working with my ER diagrams I found some anomalies that are worth mentioning. Bikes and Person attributes should not be updated or deleted while their is a Reservation or Service that includes the same Bike and Person. Both relations Bikes and Persons act as a big pool of data to fill Service and Reservation requests. Almost all queries on our database will be focussing on tables Reservations and Services. I decided I could leave in these anomalies for that reason. I will just have to check before I modify or delete a Bike or Person attribute. We are free to delete Reservations and Services as we see fit.

As far as I know there are not any Insert anomalies. This is because relations Reservations and Services can only be performed with a Bike and Person that already exist. If I add a new Person or Bike I can be sure there is no Service or Reservation including those new records. Of course If you were to add a Reservation for a Bike that doesn't exist you would not be able to find that Bike, but this is the same as saying we should not just put nonsense in for our attributes. A check should be done to confirm this in reality because people sometimes mess up and enter incorrect information.

3.

Testing plan

What situations will we need to cover?

In creating a testing plan there are a few situations I intend to cover. I firstly need to test that adding both a Person and Bike with new PersonID and BikeID attributes does not cause any problems. My database is built in such a way populating both Person and Bike should be the first thing we do. I will be assuming that this is how the database gets filled. That being said, if I need to modify or delete old records I should always make sure that both PersonID and BikeID are not being used in a Service or Reservation relation. This means that we can not change Bike or Person attributes while the Bike is being repaired, rented or reserved.

So overall:

1. Test that we can Add Persons and Bikes to our model.
2. Be sure to check that we are not altering or deleting an in use Person or Bike.
3. Always populate Bike and Person first.
4. Limit access to Bike and Person, most information can be found in Reservations or Services.

Sam Christensen
Lakshman Wadhwani

What specific errors could occur due to concurrent access of 2 users?

I could foresee lots of conflicts if two users try to access the database concurrently. The first that comes to mind is the situation in which the same BikeID or PersonID is assigned to two entities. This really messes up our plan because both BikeID and PersonID are essential to creating every candidate key. Transactional conflicts could occur with two or more concurrent users accessing the same services (rental/repair) at the same time. If 2 register Persons both try to reserve the same bike for the same day and time there will also be issues setting up the reservation records.

Any additional constraints?

Repair costs are not computed on labor hours, but rather a flat rate fee for a specific repair job. If we wanted to go more depth for repairs we could create a Repair table similar to our Reservation table. Time inputs will follow a 12hr am/pm format. One constraint we need to be aware of in the future is that although it is easy to see all the Bikes being repaired and all the Bikes being renter in the Services table it is somewhat hard to find which Bikes (with individual BikeID values) are available in store. The work around for this involves us comparing the Reservation and Services tables to the Bike catalog that is the Bike table. If we take away any BikeID (and the rest of the record) that appears in Services or Reservation tables from the Bike table we are left with the available Bikes left in store. Each one of these Bikes should have an individual BikeID.