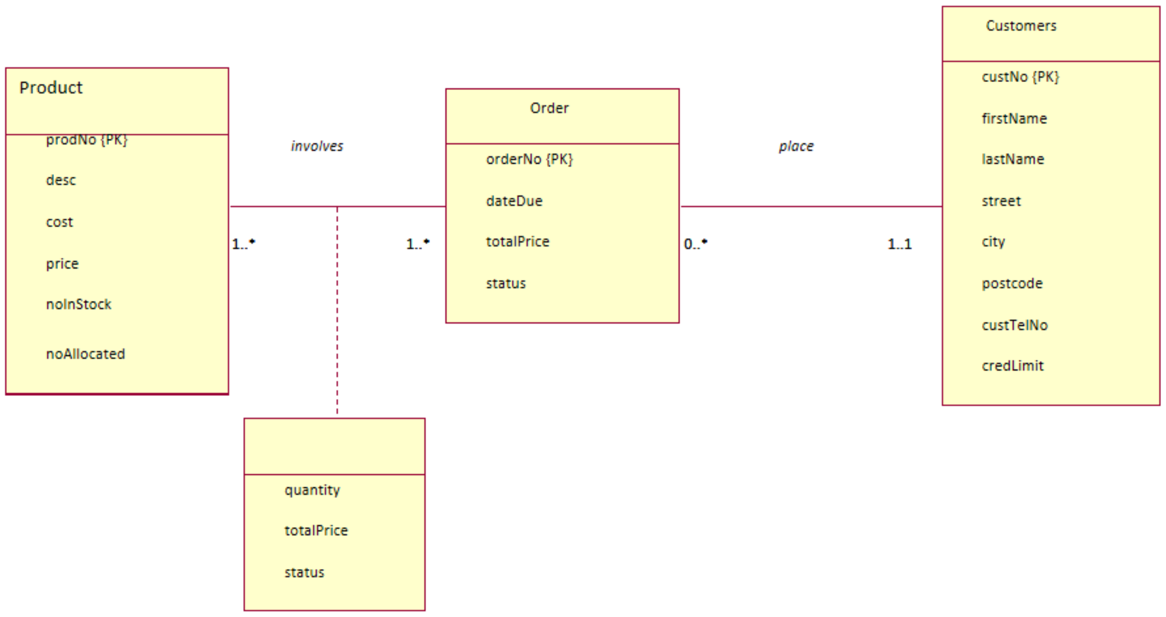
UNIVERSITI TEKNOLOGI MALAYSIA

SCSP5023 - Big Data Management
ERD Homework

Section 1

Ahmad Khaidir Amir bin Rodzman (A18CS0014)
Putera Muhammad Syabil Bin Sarianto (A18CS0235)
LI YEXIN(MCS211046)

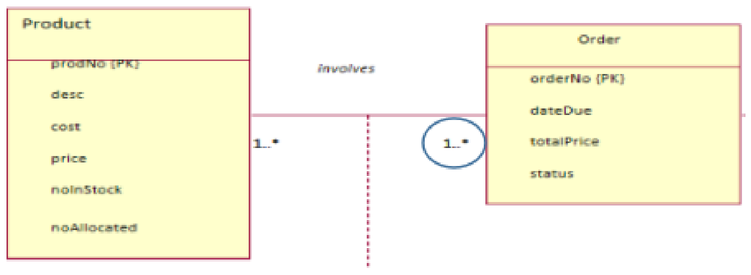
Practice 1: Derive Relations



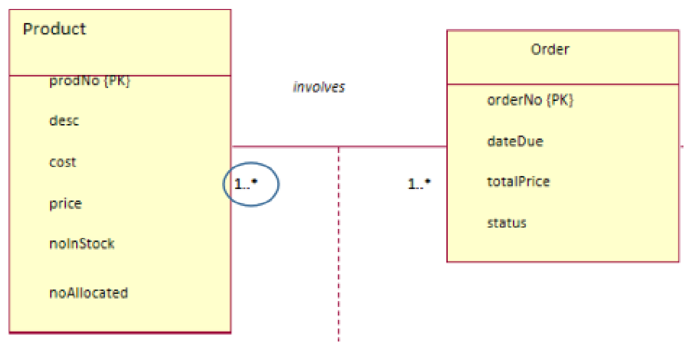
2. Determine the appropriate key and cardinality in a relationship.

Product (prodNo, desc, cost, price, noinStock, noAllocated) PK - prodNo FK - orderNo reference Order	Order (orderNo, dateDue,totalPrice,status) PK - orderNo FK - prodNo reference Product
Customers(custNo, firstName,lastName, street, city, postcode, custTelNo, credLimit) PK - custNo FK - orderNo reference Order AK - firstName, lastName, custTelNo	Involves(quantity, totalPrice, status) PK - orderNo reference Order

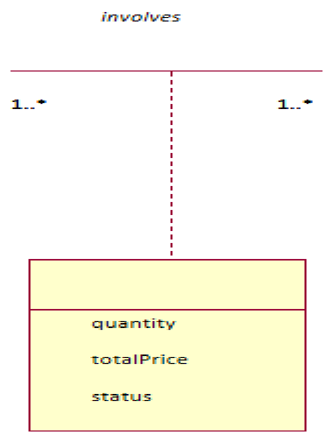
Cardinality in a relationship



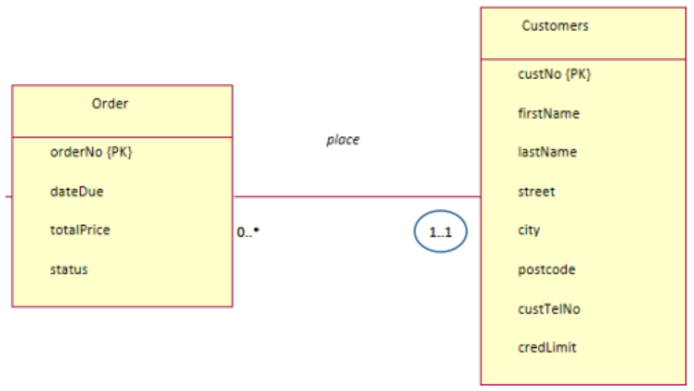
Each product involves 1 or many orders



Each order can have 1 or many products.The middle table contains two foreign keys, associate the primary key of Product and Order, respectively.



An Order involves an Involves, sets a foreign key on either party, associates the primary key of the other party, and makes the foreign key unique.



Each Order can only have 1 customer.
A foreign key is established on the Order side to associate Customers' primary key.

2. Define the relation in SQL

To create table:

```
CREATE DATABASE shop;
use shop;

CREATE TABLE product(
prodno INT UNSIGNED NOT NULL AUTO_INCREMENT,
dateDue TIMESTAMP NOT NULL DEFAULT '',
cost INT UNSIGNED NOT NULL DEFAULT 0,
price DOUBLE UNSIGNED NOT NULL DEFAULT 0,
noinstock  INT UNSIGNED NOT NULL DEFAULT 0,
noAllocated  INT UNSIGNED NOT NULL DEFAULT 0,

orderNo INT UNSIGNED NOT NULL,
PRIMARY KEY(driver_id)
);

CREATE TABLE orders(
orderno INT UNSIGNED NOT NULL AUTO_INCREMENT,
dateDue TIMESTAMP NOT NULL DEFAULT '',
totalPrice DOUBLE UNSIGNED NOT NULL DEFAULT 0,
status VARCHAR(30) NOT NULL DEFAULT '',

prodno INT UNSIGNED NOT NULL,
custNo INT UNSIGNED NOT NULL,
PRIMARY KEY(orderno)
);

CREATE TABLE Customer(
        custNo INT UNSIGNED NOT NULL AUTO_INCREMENT,
        firstName VARCHAR(30) NOT NULL DEFAULT '',
        lastName VARCHAR(30) NOT NULL DEFAULT '',
        street VARCHAR(30) NOT NULL DEFAULT '',
        city VARCHAR(30) NOT NULL DEFAULT '',
        postcode VARCHAR(30) NOT NULL DEFAULT '',
        custTelNo VARCHAR(30) NOT NULL DEFAULT '',
        credLimit  INT UNSIGNED NOT NULL DEFAULT 0,
        orderNo INT UNSIGNED NOT NULL,

        PRIMARY KEY(custNo)
);

create table Involve(
        quantity INT UNSIGNED NOT NULL DEFAULT 0,
        totalPrice DOUBLE UNSIGNED NOT NULL DEFAULT 0,
        status VARCHAR(30) NOT NULL DEFAULT '',
   prodno INT UNSIGNED NOT NULL,
        orderNo INT UNSIGNED NOT NULL,
   FOREIGN KEY (prodno) REFERENCES product(prodno),
   FOREIGN KEY (orderNo) REFERENCES orders(orderNo)
/*This is Involve asso*/
);
```

Product Associate with Order

```
use shop;

alter table Product
ADD FOREIGN KEY (orderno) REFERENCES orders(orderno);

alter table orders
ADD FOREIGN KEY (prodno) REFERENCES product(prodno);
```

Customer Associate with Order

```
use shop;
alter table Customer
ADD FOREIGN KEY (orderno) REFERENCES orders(orderno);

alter table orders
ADD FOREIGN KEY (custNo) REFERENCES Customer(custNo);
```

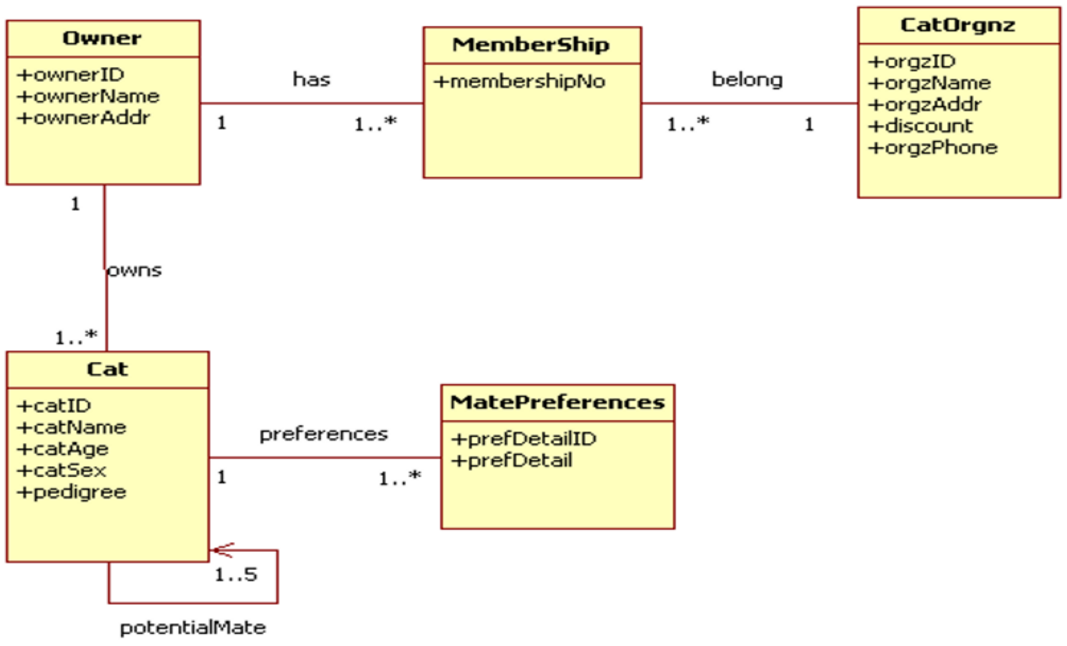
<u>Order Associate with Involve</u>

```
use shop;

alter table orders
ADD FOREIGN KEY (orderno) REFERENCES involve(orderno);


create table Involve(
            quantity INT UNSIGNED NOT NULL DEFAULT 0,
            totalPrice DOUBLE UNSIGNED NOT NULL DEFAULT 0,
            status VARCHAR(30) NOT NULL DEFAULT '',
    prodno INT UNSIGNED NOT NULL,
            orderNo INT UNSIGNED NOT NULL,
    FOREIGN KEY (prodno) REFERENCES product(prodno),
    FOREIGN KEY (orderNo) REFERENCES orders(orderNo)
/*This is Involve association to Product and Orders occur during Involve table  creation here*/
);
```

<u>Product Associate with Involve</u>

```
use shop;

alter table product
ADD FOREIGN KEY (prodno) REFERENCES involve(prodno);

create table Involve(
            quantity INT UNSIGNED NOT NULL DEFAULT 0,
            totalPrice DOUBLE UNSIGNED NOT NULL DEFAULT 0,
            status VARCHAR(30) NOT NULL DEFAULT '',
    prodno INT UNSIGNED NOT NULL,
            orderNo INT UNSIGNED NOT NULL,
    FOREIGN KEY (prodno) REFERENCES product(prodno),
    FOREIGN KEY (orderNo) REFERENCES orders(orderNo)
/*This is Involve association to Product and Orders occur during Involve table  creation here*/
);
```

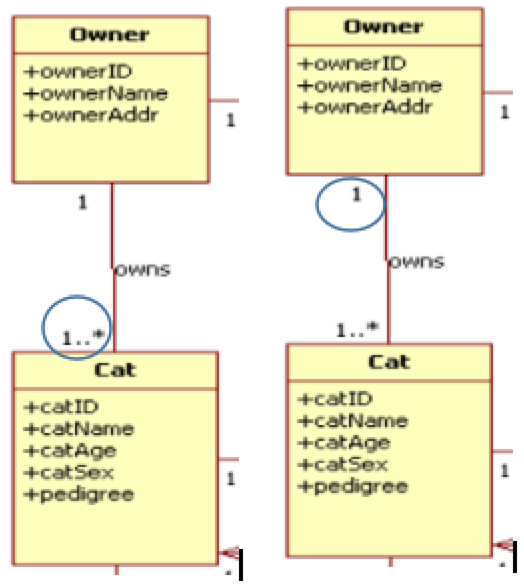
Practice 2: Derive Relations



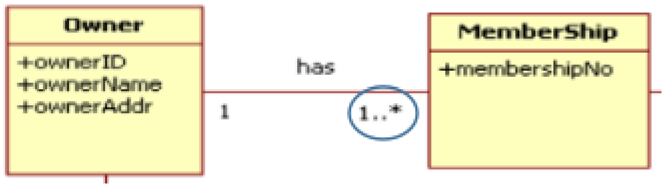
1. Determine the appropriate key and cardinality in a relationship.

Owner (ownerID, ownerName, ownerAddr) PK - ownerID FK - membershipNo reference MemberShip FK - catID reference Cat AK - ownerName, ownerAddr	MemberShip(membershipNo) PK - membershipNo FK - ownerID reference Owner FK - orgzID reference CatOrgnz
Cat(catID, catName,catAge, catSex,pedigree) PK - catID FK - ownerID reference Owner FK - prefDetailID reference MatePreferences AK - catName	potentialMate(catID, catName,catAge, catSex,pedigree) PK - catID FK - ownerID reference Owner FK - prefDetailID reference MatePreferences AK - catName
CatOrgnz(orgzID, orgzName,orgzAddr,discount,orgzPhone) PK - orgzID FK - membershipNo reference MemberShip AK - orgzName, orgzAddr, orgzPhone	MatePreferences(prefDetailID,prefDetail) PK - prefDetailID FK - catID reference Cat

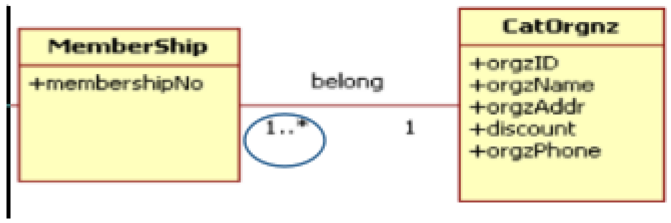
Cardinality in a relationship



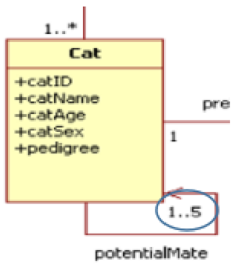
Each Owner can have one or many cats while a Cat can only have one owner.



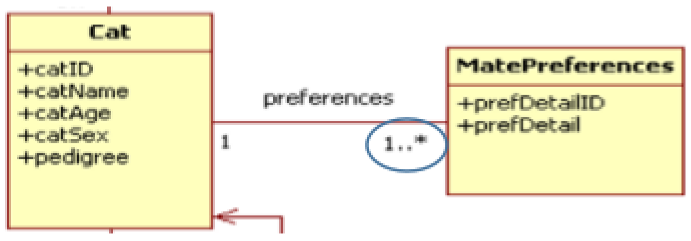
Each Owner can have one or many MemberShip while a MemberShip only can have one Owner.



A CatOrgnz belong to one or many MemberShip while a Membership belong to only one CatOrgnz.



Each Cat can have around one to five potentialMate.



Eact Cat can prefer one or many MatePreferences while each MatePreferences only preferred by a Cat.

2. **Define the relation in SQL**

<u>Create Table:</u>

```
CREATE DATABASE catshop;
use catshop;
create table Cat(
   catID INT UNSIGNED NOT NULL AUTO_INCREMENT,
         catName VARCHAR(30) NOT NULL DEFAULT '',
         catAge INT UNSIGNED NOT NULL DEFAULT 0,
         catSex VARCHAR(30) NOT NULL DEFAULT '',
         pedigree VARCHAR(30) NOT NULL DEFAULT '',

         ownerID INT UNSIGNED NOT NULL DEFAULT 0,
         prefDetailID INT UNSIGNED NOT NULL DEFAULT 0,


         PRIMARY KEY(catID)
);
```

```
create table Owner(
   ownerID INT UNSIGNED NOT NULL AUTO_INCREMENT,
         ownerName VARCHAR(30) NOT NULL DEFAULT '',
         ownerAddr VARCHAR(30) NOT NULL DEFAULT '',

         membership INT UNSIGNED NOT NULL DEFAULT 0,
         catID INT UNSIGNED NOT NULL DEFAULT 0,


         PRIMARY KEY(ownerID)
);
create table MemberShip(
   membershipNo INT UNSIGNED NOT NULL AUTO_INCREMENT,

         ownerID INT UNSIGNED NOT NULL DEFAULT 0,
         orgzID INT UNSIGNED NOT NULL DEFAULT 0,

         PRIMARY KEY(membershipNo)
);
create table CatOrgnz(
   orgzID INT UNSIGNED NOT NULL AUTO_INCREMENT,
         orgzName VARCHAR(30) NOT NULL DEFAULT '',
         orgzAddr VARCHAR(30) NOT NULL DEFAULT '',
         discount double,
         orgzPhone VARCHAR(30) NOT NULL DEFAULT '',

         membershipNo INT UNSIGNED NOT NULL DEFAULT 0,

         PRIMARY KEY(orgzID)
);

create table MatePreferences(
   prefDetailID INT UNSIGNED NOT NULL AUTO_INCREMENT,

         catID INT UNSIGNED NOT NULL DEFAULT 0,

         PRIMARY KEY(prefDetailID)
);
```

```
create table potentialMate(
   catID INT UNSIGNED NOT NULL DEFAULT 0,
         catName VARCHAR(30) NOT NULL DEFAULT '',
         catAge INT UNSIGNED NOT NULL DEFAULT 0,
         catSex VARCHAR(30) NOT NULL DEFAULT '',
         pedigree VARCHAR(30) NOT NULL DEFAULT '',

         ownerID INT UNSIGNED NOT NULL DEFAULT 0,
         prefDetailID INT UNSIGNED NOT NULL DEFAULT 0,

         PRIMARY KEY(catID)
);
```

<u>Membership associate with CatOrgnz</u>

```
use catshop;

ALTER TABLE membership
ADD FOREIGN KEY (orgzID) REFERENCES catorgnz(orgzID);

ALTER TABLE catorgnz
ADD FOREIGN KEY (membershipNo) REFERENCES membership(membershipNo);
```

<u>Membership associate with Owner</u>

```
use catshop;

ALTER TABLE membership
ADD FOREIGN KEY (ownerID) REFERENCES owner(ownerID);

ALTER TABLE owner
ADD FOREIGN KEY (membershipNo) REFERENCES membership(membershipNo);
```

Owner associate with Cat

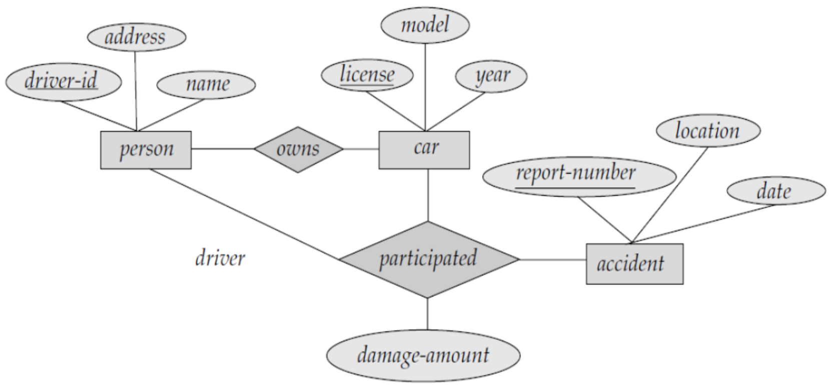
```
use catshop;

ALTER TABLE owner
ADD FOREIGN KEY (catID) REFERENCES cat(catID);

ALTER TABLE cat
ADD FOREIGN KEY (ownerID) REFERENCES owner(ownerID);
```

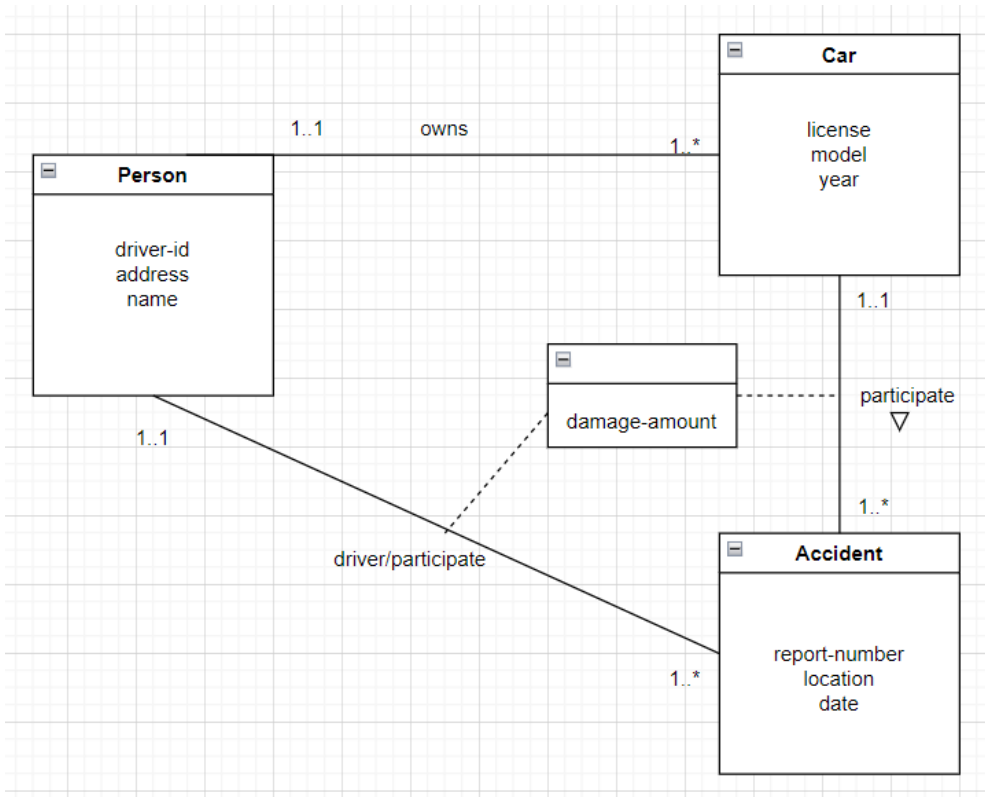
Cat assosciate with MatePreferences

```
use catshop;

ALTER TABLE cat
ADD FOREIGN KEY (prefDetailID) REFERENCES MatePreferences(prefDetailID);

ALTER TABLE MatePreferences
ADD FOREIGN KEY (catID) REFERENCES cat(catID);
```

Cat assosciate with PotentialMate

```
use catshop;

ALTER TABLE cat
ADD FOREIGN KEY (catID) REFERENCES potentialMate(catID);

ALTER TABLE potentialMate
ADD FOREIGN KEY (catID) REFERENCES cat(catID);
```

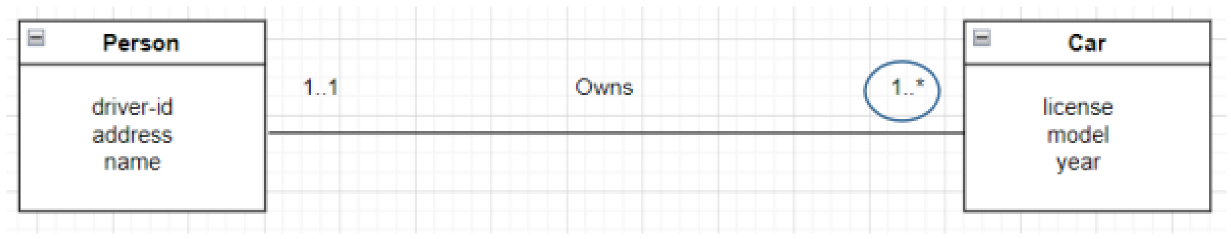




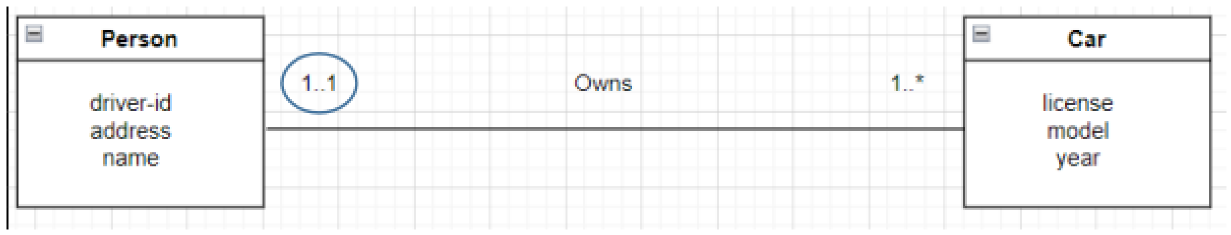
Determine the appropriate key and cardinality in a relationship.

Person (driver-id,address,name) PK - driver-id FK - license reference Car AK - address, name	Car (license , model, year) PK - license FK - driver-id reference Person
Accident(report-number, location,date) PK - report-number FK - license reference Car FK - driver-id reference Person AK - location	Participation (damage-amount) FK - license reference Car FK - driver-id reference Person FK - report-number reference Accident

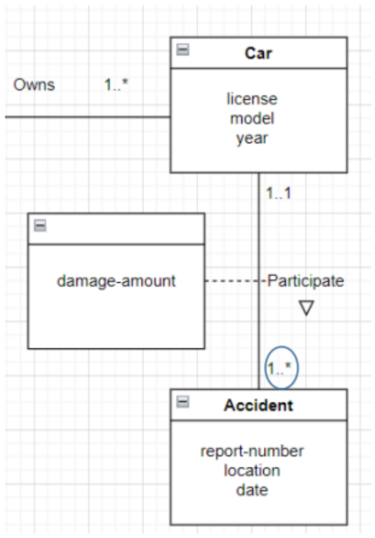
Cardinality in a relationship



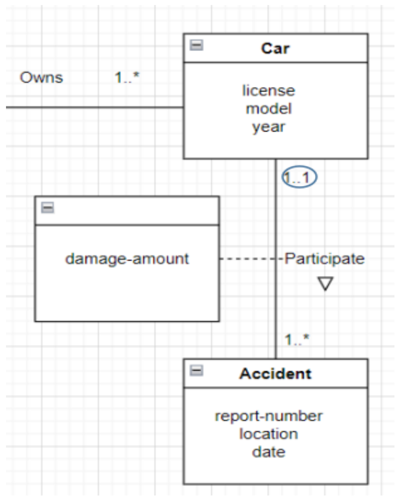
Each person can own one or multiple cars. 1..*



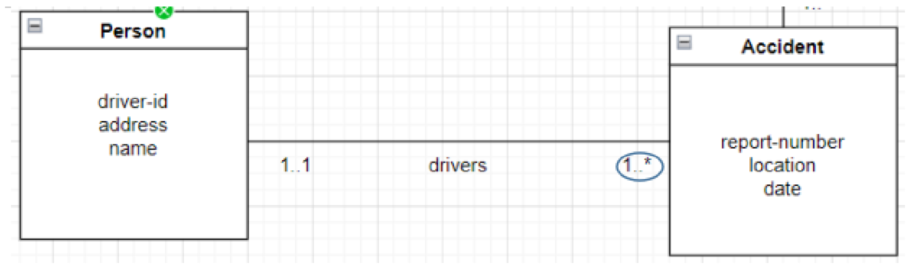
Each car can only be owned by one person. 1..1



Each car can participate in multiple accidents, 1..*



Each accidents can only participate by one car, so it is 1..1
An accident can cause many kinds of damage amount, including person and car, but a damage amount only corresponds to a single accident.
So damage amount and accident are many-to-one.



Each Person can drive on one or many Accident in lifetime while each Accident can only include a driver's Person.

2. Define the relation in SQL

To create table:

```
create database  accident_report;
use accident_report;
create table Person(
        driver_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
        address VARCHAR(30) NOT NULL DEFAULT '',
        name VARCHAR(30) NOT NULL DEFAULT '',

        license VARCHAR(30) NOT NULL DEFAULT '',
        report_number INT UNSIGNED NOT NULL,
        PRIMARY KEY(driver_id)
);

create table Cars(
        license VARCHAR(30) NOT NULL DEFAULT '',
        model VARCHAR(30) NOT NULL DEFAULT '',
        year int NOT NULL DEFAULT 0,

   driver_id  INT UNSIGNED NOT NULL,
   report_number INT UNSIGNED NOT NULL,

        PRIMARY KEY(license)
);

create table Accident(
        report_number INT UNSIGNED NOT NULL AUTO_INCREMENT,
        location VARCHAR(30) NOT NULL DEFAULT '',
        date TIMESTAMP NOT NULL,


        driver_id  INT UNSIGNED NOT NULL,
        license VARCHAR(30) NOT NULL DEFAULT '',

        PRIMARY KEY(report_number)
);
```

```
create table Participation(
        driver_id INT UNSIGNED NOT NULL,
        license VARCHAR(30) NOT NULL DEFAULT '',
        report_number INT UNSIGNED NOT NULL,
        damage_amount double NOT NULL

);
```

Person Associate with Car

```
use accident_report;
alter table Person
ADD FOREIGN KEY (license) REFERENCES Cars(license);

alter table Cars
ADD FOREIGN KEY (driver_id) REFERENCES Person(driver_id);
```

Car Associate with Accident

```
use accident_report;

alter table Cars
ADD FOREIGN KEY (report_number) REFERENCES accident(report_number);

alter table accident
ADD FOREIGN KEY (license) REFERENCES cars(license);
```

Person Associate with Accident

```
use accident_report;

alter table person
ADD FOREIGN KEY (report_number) REFERENCES accident(report_number);

alter table accident
ADD FOREIGN KEY (driver_id) REFERENCES person(driver_id);
```

Participation Associate with Person

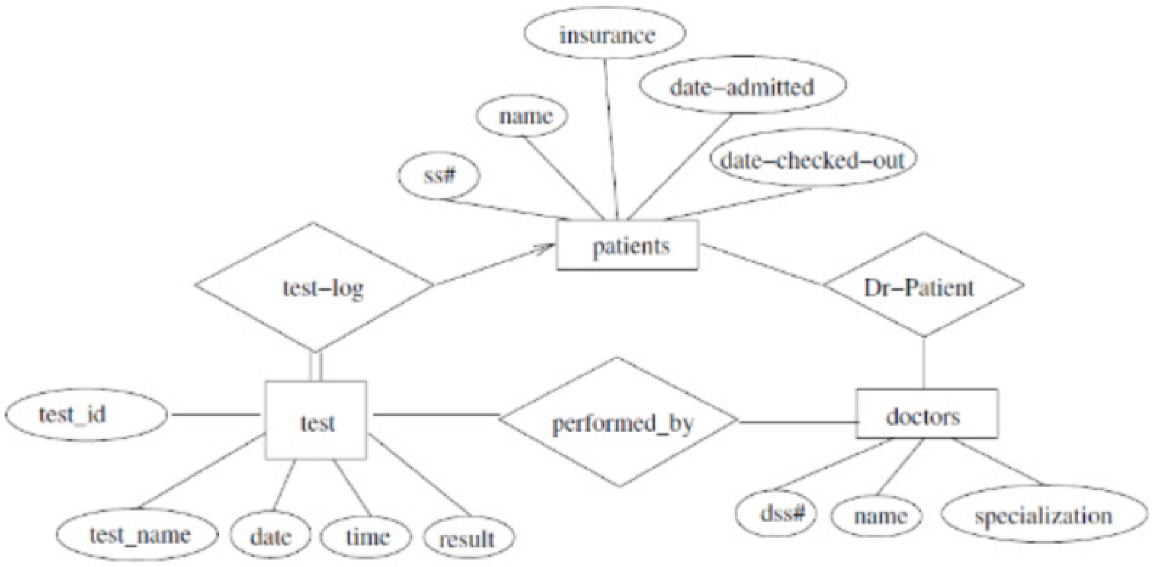
```
alter table participation
ADD FOREIGN KEY (driver_id) REFERENCES person(driver_id);

alter table person
ADD FOREIGN KEY (driver_id) REFERENCES participation(driver_id);
```

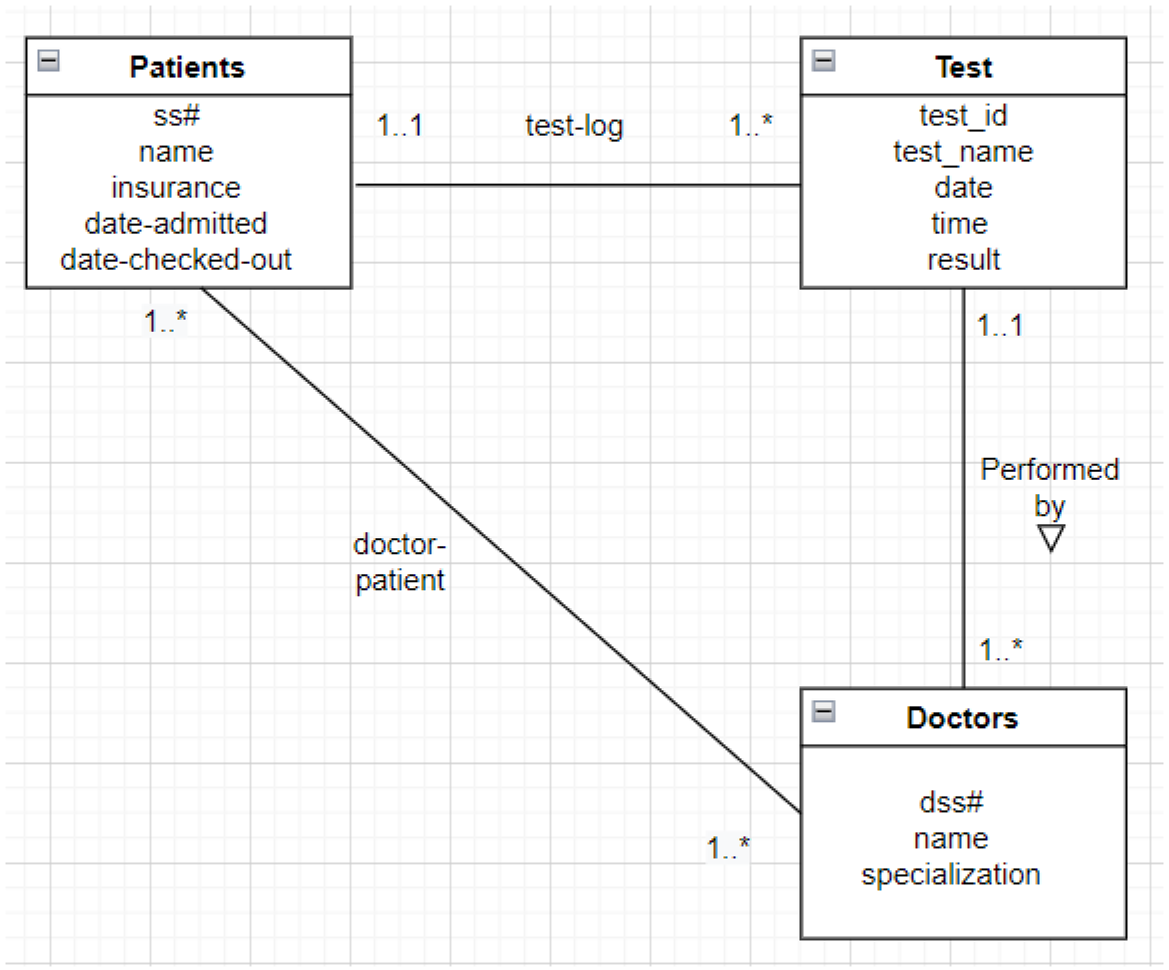
Participation Associate with Accident

```
alter table participation
ADD FOREIGN KEY (report_number) REFERENCES accident(report_number);

alter table accident
ADD FOREIGN KEY (report_number) REFERENCES participation(report_number);
```

Participation Associate with cars

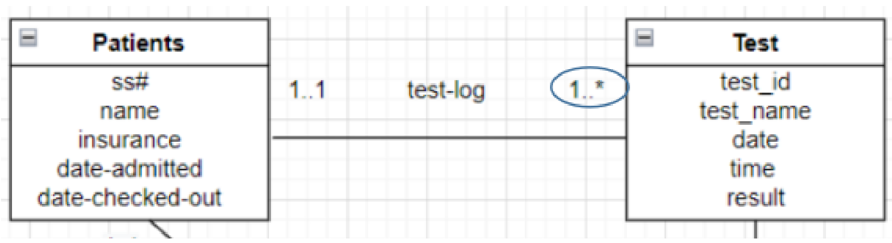
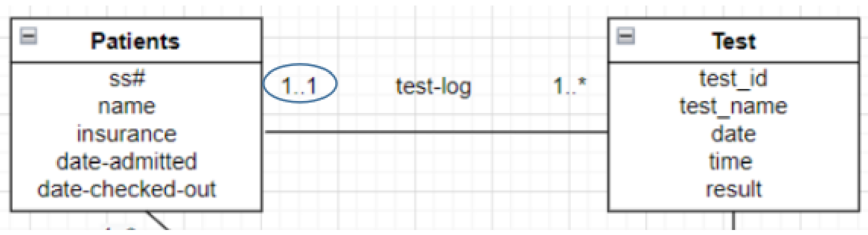
```
alter table participation
ADD FOREIGN KEY (license) REFERENCES cars(license);

alter table cars
ADD FOREIGN KEY (license) REFERENCES participation(license);
```

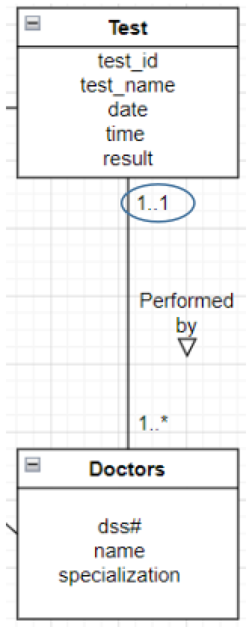

insurance
name
date–admitted
ss#
date–checked–out
patients
test–log
Dr–Patient
test_id
test
performed_by
doctors
test_name
date
time
result
dss#
name
specialization


Patients
ss#
name
insurance
date-admitted
date-checked-out
1..1
test-log
1..*
Test
test_id
test_name
date
time
result
1..*
1..1
Performed by
doctor-patient
1..*
1..*
Doctors
dss#
name
specialization

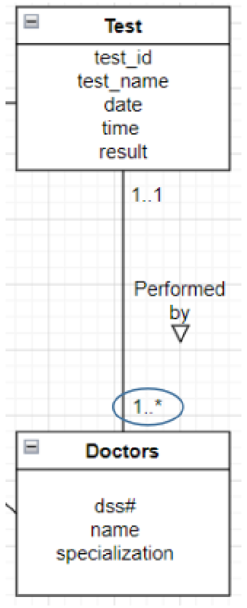
1. Determine the appropriate key and cardinality in a relationship.

Test (test_id, test_name, date, time, result) PK - test_id FK - ss# reference Patients FK - dss# reference Doctors AK - test_name	Doctors (dss# , name, specialization) PK - dss# FK - ss# reference Patients FK - test-id reference Test AK - name
Patients (ss#,name,insurance,date-admitted,date-checked-out) PK - ss# FK - dss# reference Doctors FK - test-id reference Test AK - name	

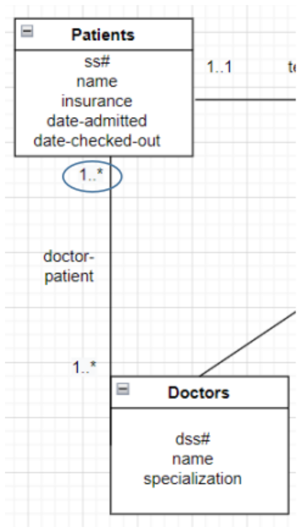
 Patients ss# name insurance date-admitted date-checked-out 1..1 test-log 1..* Test test_id test_name date time result Each patient can have one or multiple test which is 1..*
 Patients ss# name insurance date-admitted date-checked-out 1..1 test-log 1..* Test test_id test_name date time result Each test only can be take by one patient which is 1..1



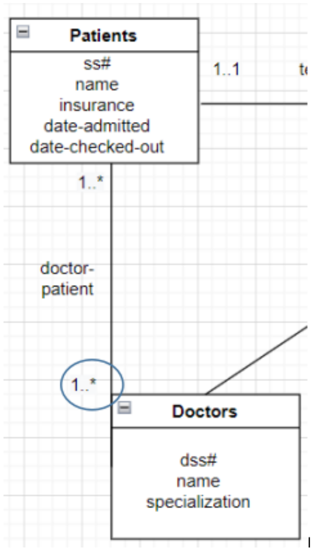
Each test only can be performed by one patient which is 1..1



Each doctor can perform one or many tests, so it is 1..*



Each doctor can treat one or many patients, so it is 1..*



Each patient can be treated by one or many doctors, so it is 1..*

2. Define the relation in SQL

To create table:

```
use hospital;   /*This is database name*/
create table Test(
   test_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
   test_name VARCHAR(30) NOT NULL DEFAULT '',
         dateTIMESTAMP NOT NULL DEFAULT '',
         time VARCHAR(30) NOT NULL DEFAULT '',
         result VARCHAR(30) NOT NULL DEFAULT '',

         dss INT UNSIGNED NOT NULL,
         ss INT UNSIGNED NOT NULL,

         PRIMARY KEY(test_id)
);
create table Doctors(
   dss INT UNSIGNED NOT NULL AUTO_INCREMENT,
   name VARCHAR(30) NOT NULL DEFAULT '',
         specialization VARCHAR(30) NOT NULL DEFAULT '',

         ss INT UNSIGNED NOT NULL,
    test_id INT UNSIGNED NOT NULL,

         PRIMARY KEY(dss)
);
create table Patients(
   ss INT UNSIGNED NOT NULL AUTO_INCREMENT,
   name VARCHAR(30) NOT NULL DEFAULT '',
         insurance VARCHAR(30) NOT NULL DEFAULT '',
   date_admitted VARCHAR(30) NOT NULL DEFAULT '',
         time_check_out VARCHAR(30) NOT NULL DEFAULT '',

   test_id INT UNSIGNED NOT NULL,
   dss INT UNSIGNED NOT NULL,

         PRIMARY KEY(ss)
);
```

Test associate patients :

```
USE hospital;

ALTER TABLE Test
ADD FOREIGN KEY (ss) REFERENCES Patients(ss);

ALTER TABLE Patients
ADD FOREIGN KEY (test_id) REFERENCES Test(test_id)
```

Patient associate doctors:

```
USE hospital;

ALTER TABLE Doctors
ADD FOREIGN KEY (ss) REFERENCES Patients(ss);

ALTER TABLE Patients
ANN FOREIGN KEY (dss) REFERENCES Doctors(dss);
```

Test associate doctors:

```
USE hospital;

ALTER TABLE test
ADD FOREIGN KEY (dss) REFERENCES Doctors(dss);

ALTER TABLE Doctors
ADD FOREIGN KEY (test_id) REFERENCES test(test_id);
```

REFERENCE

https://www.skytowner.com/explore/referencing_column_and_referenced_column_are_incompat ible_in_mysql