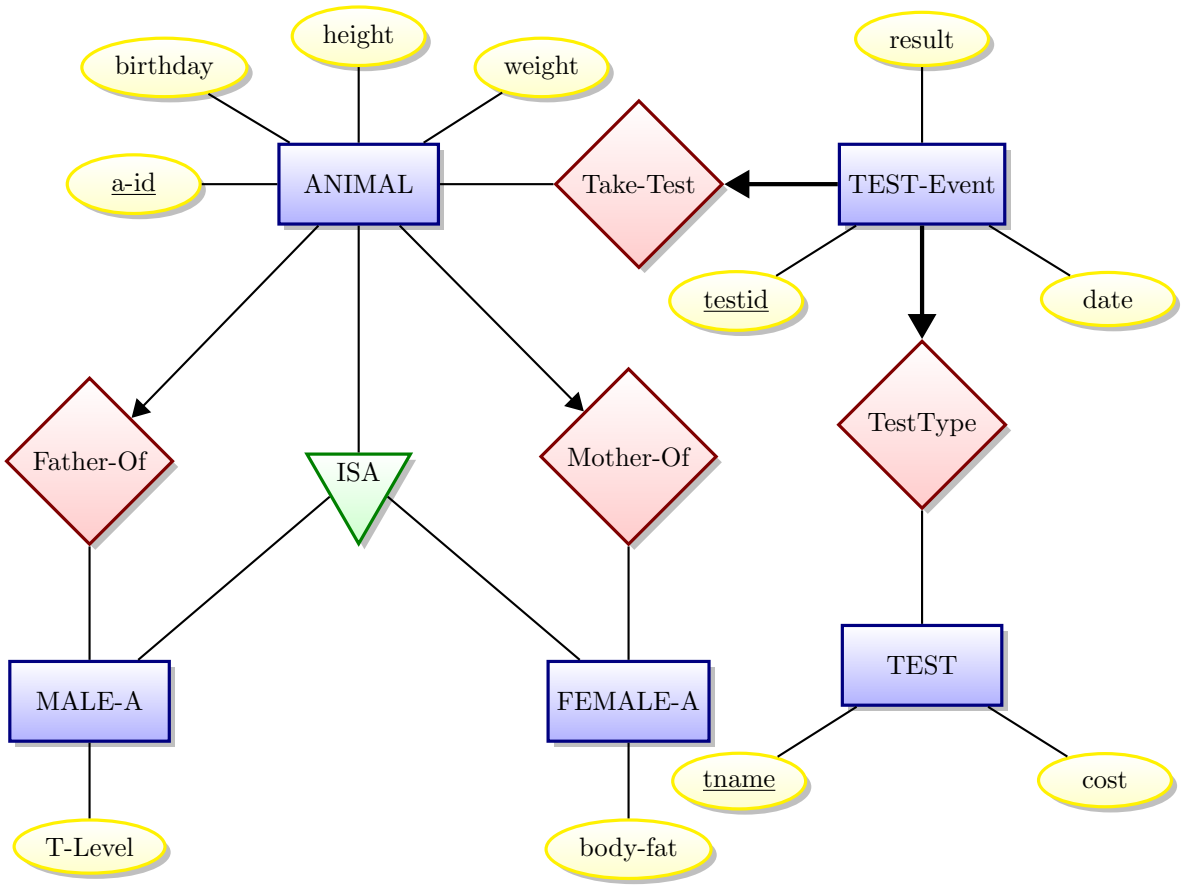
# HW1 Solution [1]

[1] Database Systems; Spring 2020

Instructor: Bin Yao

1 (a) A good design is as follows:




Note that a bold line indicates a total participation constraint.

(b) Tables (Schemas) and Keys

Animal (aid, mother-id, father-id, birthday, height, weight)
Female (aid, bodyfat)
Male (aid, t-level)
TestEvent (testid, result, date, aid, tname)
Test (tname, cost)

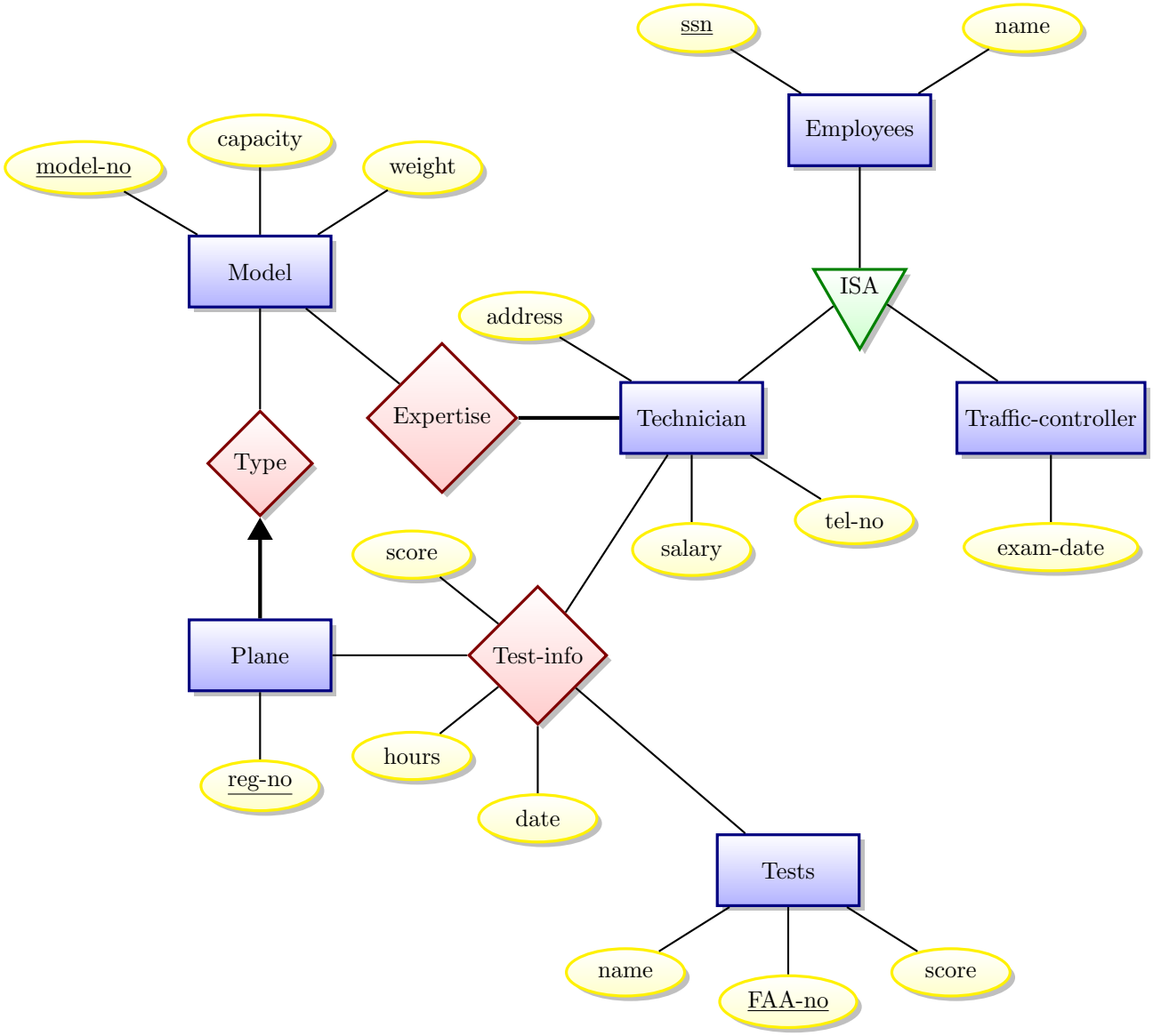
Note that cost is an "imagined", optional attribute; just to illustrate the fact that you may have other attributes of interest for a type of test.

- SQL statement:

```
CREATE TABLE Male (
aid INTEGER, T-level INTEGER,
Primary Key (aid),
Foreign Key (aid) References Animal );

CREATE TABLE Female (
aid INTEGER,  bodyfat REAL,
Primary Key(aid)
Foreign Key (aid) References Animal );

CREATE TABLE Animal (
aid INTEGER,  birthday DATE, height REAL, weight REAL,
mother-id INTEGER, father-id INTEGER,
```

```
Primary Key (aid),
Foreign Key (father-id) References (Male),
Foreign Key (mother-id) References (Female));

CREATE TABLE TestEvent (
testid INTEGER, aid INTEGER NOT NULL,  tname CHAR[40] NOT NULL,
date DATE, result REAL,
Primary Key (testid),
Foreign Key (aid) References (Animal),
Foreign Key (tname) References (Test));

CREATE TABLE Test(
tname CHAR[40], cost INTEGER,  Primary Key (tname));
```

2 (a) Here is an ER diagram for this database:




*

Note: The bold line indicates total participation.

(b) The relational schema is as follows:

Model (model-no, capacity, weight)
PlaneType (reg-no, model-no)
Expertise(model-no, ssn)
Employee (ssn,name)
Technician (ssn, address, salary, tel-no)
TrafficController (ssn, exam-date)
Tests (FAA-no, name, score)
TestInfo (reg-no, ssn, FAA-no, hours, date, score)

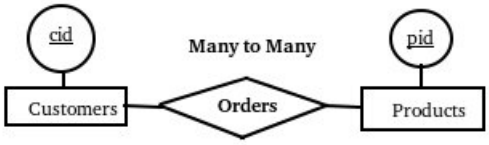
- The SQL statements for creating these tables are as follows (Note that the integrity constraints are added):

```
CREATE TABLE Model(
model-no INTEGER,
capacity INTEGER,
weight REAL,
PRIMARY KEY (model-no))

CREATE TABLE PlaneType(
reg-no INTEGER,
model-no INTEGER NOT NULL,
PRIMARY KEY (reg-no),
FOREIGN KEY (model-no) REFERENCES Model)

CREATE TABLE Expertise(
ssn  CHAR(20) NOT NULL,
model-no INTEGER,
PRIMARY KEY (ssn, model-no),
FOREIGN KEY (ssn) REFERENCES Technician,
FOREIGN KEY (model-no) REFERENCES Model)

CREATE TABLE Employees(
ssn  CHAR(20),
union-no INTEGER,
PRIMARY KEY (ssn))

CREATE TABLE Technician(
ssn  CHAR(20),
name CHAR(20),
address CHAR(20),
tel-no CHAR(20),
PRIMARY KEY (ssn),
FOREIGN KEY (ssn) REFERENCES Employees)

CREATE TABLE TrafficController(
ssn  CHAR(20),
exam-date DATE,
PRIMARY KEY (ssn),
FOREIGN KEY (ssn) REFERENCES Employees)

CREATE TABLE Tests(
FFA-no INTEGER,
name CHAR(20),
score INTEGER,
```

```
    PRIMARY KEY (FFA-no))

    CREATE TABLE TestInfo(
    reg-no INTEGER,
    ssn  CHAR(20),
    FFA-no INTEGER,
    hours INTEGER,
    date DATE,
    score INTEGER,
    PRIMARY KEY (FFA-no, ssn, reg-no),
    FOREIGN KEY (reg-no) REFERENCES PlaneType,
    FOREIGN KEY (ssn) REFERENCES Technicians,
    FOREIGN KEY (FFA-no) REFERENCES Tests)
```

3

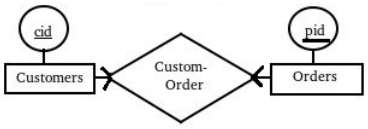
a (7 points) Draw a (simple) E-R diagram that results in a primary key/foreign key constraint to be created between tables. Show the SQL statements that create the tables including the foreign key and primary key indications.




```
CREATE TABLE Customers(
    cid CHAR(10),
    primary key(cid))
CREATE TABLE Products(
    pid CHAR(10),
    primary key(pid))
CREATE TABLE Orders(
    cid CHAR(10),
    pid CHAR(10),
    PRIMARY KEY (cid, pid),
    FOREIGN KEY (cid) REFERENCES Customers,
    FOREIGN KEY (pid) REFERENCES Products)
```


```
 CREATE TABLE Salespersons(
    sid CHAR(10),
    primary key (sid))
 CREATE TABLE Customers(
    cid CHAR(10),
```

```
  sid CHAR(10),
  PRIMARY KEY(cid),
  FOREIGN KEY(sid) REFERENCES Salespersons)
```




```
CREATE TABLE Customers(
  cid CHAR(10),
  primary key(cid))
CREATE TABLE Orders(
  cid CHAR(10),
  pid CHAR(10),
  PRIMARY KEY (pid),
  UNIQUE (cid),
  FOREIGN KEY(cid) REFERENCES Customers)
```

b (8 points) Consider a database of employees and departments, in which we need to record information about: 1) who works in what departments, and 2) the period a person works in a department.

Discuss the difference between the following two designs: 1) Add "From" and "To" as attributes of the works-in relationship set. 2) Model period as an entity set.

*The first approach* is only able to model the case where an employee may work for the same department only once, since given an employee A and a department X, there can be only one relationship instance in the works-in relationship set that connects A with X.

*The second approach* allows a tenary relationship set to connect Employee, Department, and Period entity sets. Hence, given an employee A and a department X, there could be multiple relationship instance from the works-in relationship set to connect A and X with different entities of Period, e.g., (A, X, p1), (A, X, p2) where p1 and p2 are two different entities from the Period entity set.

c (5 points) Continue from the example in b), how to enforce that an employee may work for at most one department, but a department may have more than one employees. Furthermore, each department must have at least one employee.

*To enforce the first constraint*, we will make the works-in relationship set a M-1 relationship set (M is on the Employee entitye set and 1 is on the Department entity set, i.e., an arrow coming out of the Employee entity set pointing towards the works-in relationship set).

*To enforce the second constraint*, we will add total participation constraint on the side with the Department entity set for the works-in relationship set, i.e., make the line connecting the Department entity set with the works-in relationship set a bold line.