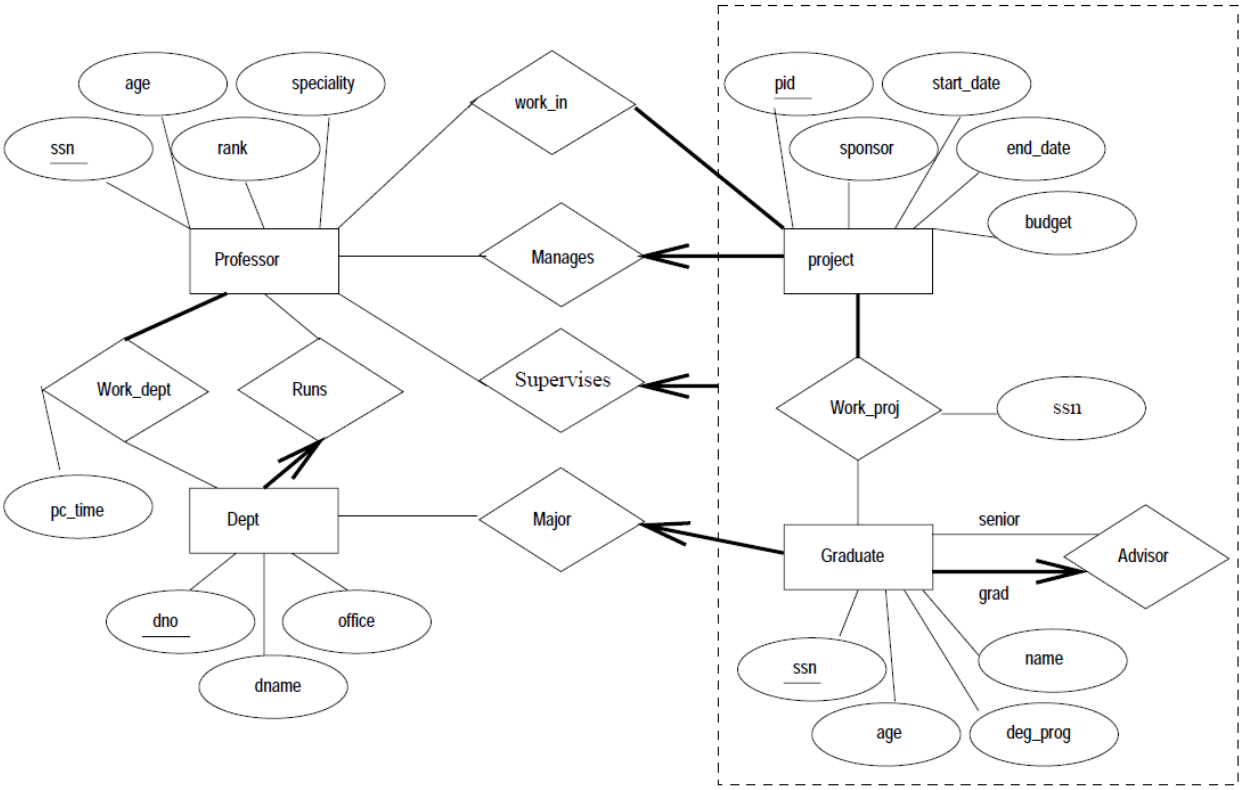
University ER Design Solution:



1. `CREATE TABLE` Professors (
   prof_ssn `CHAR(10)`,
   name `CHAR(64)`,
   age `INTEGER`,
   rank `INTEGER`,
   speciality `CHAR(64)`,
   `PRIMARY KEY` (prof_ssn))

2. `CREATE TABLE` Depts (
   dno `INTEGER`,
   dname `CHAR(64)`,
   office `CHAR(10)`,
   `PRIMARY KEY` (dno))

3. `CREATE TABLE` Runs (
   dno `INTEGER`,
   prof_ssn `CHAR(10)`,
   `PRIMARY KEY` (dno, prof_ssn),
   `FOREIGN KEY` (prof_ssn) `REFERENCES` Professors,
   `FOREIGN KEY` (dno) `REFERENCES` Depts)

4. CREATE TABLE Work_Dept (dno INTEGER,
prof_ssn CHAR(10),
pc_time INTEGER,
PRIMARY KEY (dno, prof_ssn),
FOREIGN KEY (prof_ssn) REFERENCES Professors,
FOREIGN KEY (dno) REFERENCES Depts)

Observe that we would need check constraints or assertions in SQL to enforce the rule that Professors work in at least one department.

5. CREATE TABLE Project (pid INTEGER,
sponsor CHAR(32),
start_date DATE,
end_date DATE,
budget FLOAT,
PRIMARY KEY (pid))

6. CREATE TABLE Graduates (grad_ssn CHAR(10),
age INTEGER,
name CHAR(64),
deg_prog CHAR(32),
major INTEGER,
PRIMARY KEY (grad_ssn),
FOREIGN KEY (major) REFERENCES Depts)

Note that the Major table is not necessary since each Graduate has only one major and so this can be an attribute in the Graduates table.

7. CREATE TABLE Advisor (senior_ssn CHAR(10),
grad_ssn CHAR(10),
PRIMARY KEY (senior_ssn, grad_ssn),
FOREIGN KEY (senior_ssn)
REFERENCES Graduates (grad_ssn),
FOREIGN KEY (grad_ssn) REFERENCES Graduates)

8. CREATE TABLE Manages (pid INTEGER,
prof_ssn CHAR(10),
PRIMARY KEY (pid, prof_ssn),
FOREIGN KEY (prof_ssn) REFERENCES Professors,
FOREIGN KEY (pid) REFERENCES Projects)

9. CREATE TABLE Work_In (pid INTEGER,
prof_ssn CHAR(10),
PRIMARY KEY (pid, prof_ssn),
FOREIGN KEY (prof_ssn) REFERENCES Professors,
FOREIGN KEY (pid) REFERENCES Projects)

Observe that we cannot enforce the participation constraint for Projects in the Work_In table without check constraints or assertions in SQL.

10. CREATE TABLE Supervises (prof_ssn CHAR(10),
grad_ssn CHAR(10),
pid INTEGER,
PRIMARY KEY (prof_ssn, grad_ssn, pid),
FOREIGN KEY (prof_ssn) REFERENCES Professors,
FOREIGN KEY (grad_ssn) REFERENCES Graduates,
FOREIGN KEY (pid) REFERENCES Projects)

Note that we do not need an explicit table for the Work_Proj relation since every time a Graduate works on a Project, he or she must have a Supervisor.