Database design and programming

Yannis and the TAs want a web application that can manage well the process of collecting the solutions to the project's phases and grading them. This problem is about the database part of this application.

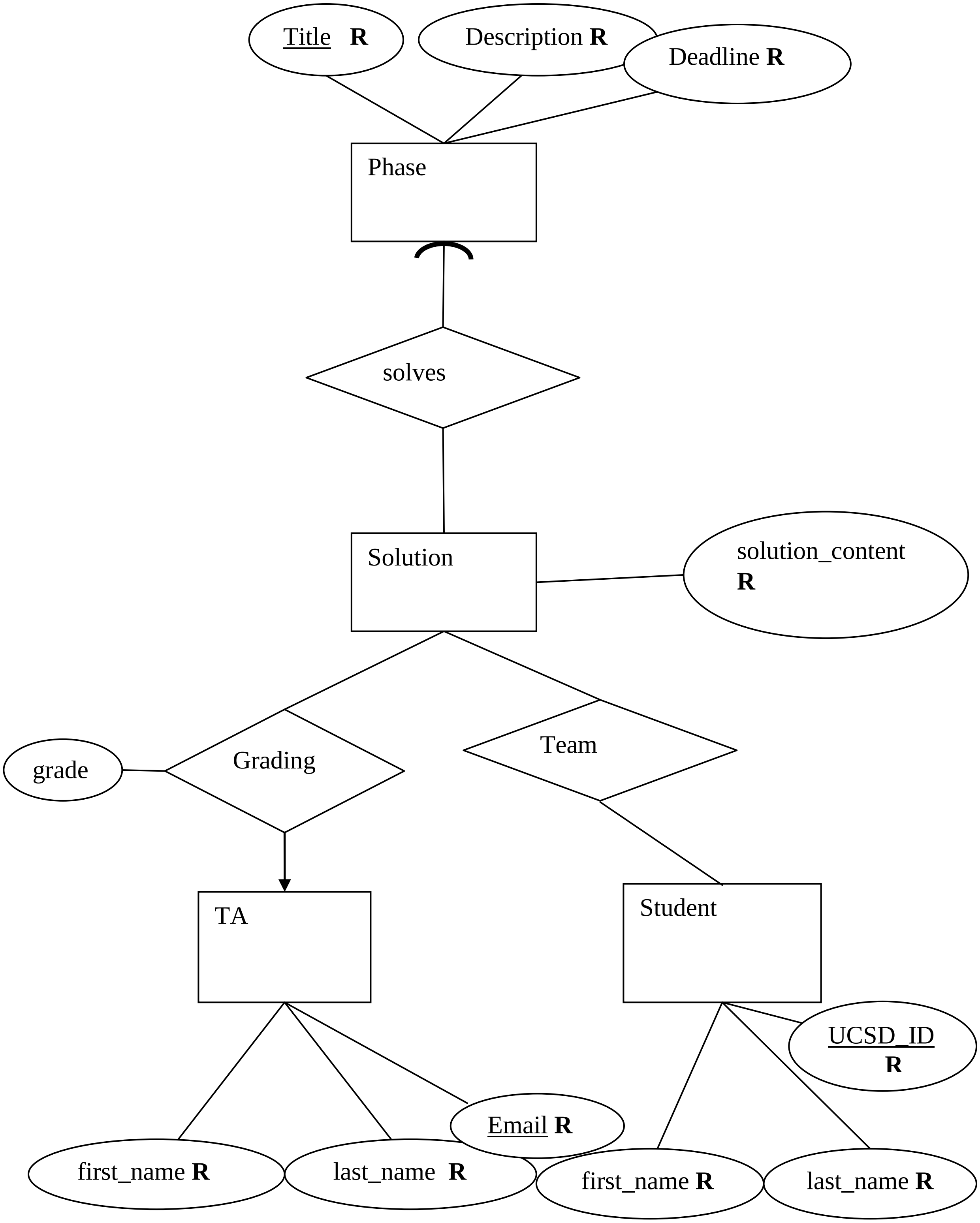
The database must capture the following information:

- Each phase of the project must have a unique title, a descriptive text and a deadline. We want the database to be able to manage any number of phases.
- We have student information that must include first name, last name, UCSD ID and email.
- Students team up to submit a solution to each Phase. The teams are not fixed. For example, Jim and Helen submitted together their solution to Phase 1. John and Mary also submitted together their solution to Phase 1. But then Helen dropped the class and Mary could not stand John. So, Jim and Mary submitted together Phases 2, 3, 4, 5 and 6 and John solved Phases 2-6 by himself. Do not restrict the number of members of the team. Could be 1, could be 2, could be more.
- We have TA information that must include first name, last name and a unique email.
- The assignment of solutions to TAs is as follows: Generally, a few hours after a solution has been submitted, a TA is assigned to it. This TA later assigns a grade to the solution. Notice, the TAs cannot assign different grades to each member of the team that submitted the solution. Rather, the team solution gets the grade.

Do not model aggregate measures, which can be derived from other data, into the E/R or the schema.

1. Create an E/R design for the above. Take a look at the queries before you decide on a design. Some E/R designs and schemas may make querying easier than others. Furthermore, the queries will make some requirements more clear.

   Here is one of the best possible solutions. It is probably the best when it comes to simplicity and minimizing the amount of effort needed to write the queries.


Title R
Description R
Deadline R
Phase
solves
Solution
solution_content R
Grading
grade
Team
TA
Student
UCSD_ID R
Email R
first_name R
last_name R
first_name R
last_name R

2. Create a relational schema, by providing the CREATE TABLE commands. Include UNIQUE constraints whenever applicable. Generally ignore CHECK and NOT NULL constraints but include any applicable NOT NULL constraints on foreign keys. Do not use people names or phase titles as primary keys and foreign keys of the respective tables.

```
CREATE TABLE     Phase(
id                   SERIAL PRIMARY KEY,
title                  TEXT UNIQUE,
descriptive_text    TEXT,
deadline               DATE
);

CREATE TABLE     Student(
id               SERIAL PRIMARY KEY,
first_name      TEXT,
last_name       TEXT,
UCSD_ID       TEXT UNIQUE
);

CREATE TABLE     Solution(
id                          SERIAL PRIMARY KEY,
solution_content           TEXT,
grade                        INTEGER,
phase,                      INTEGER REFERENCES
ta                           INTEGER REFERENCES    TA(id)
);

CREATE TABLE     Team(
solution                    INTEGER REFERENCES    Solution (solution_id) NOT NULL,
student                     INTEGER REFERENCES    Student(s_id) NOT NULL
UNIQUE (solution, student)
);

CREATE TABLE TA(
id             SERIAL PRIMARY KEY,
first_name      TEXT,
last_name       TEXT,
email            TEXT UNIQUE
);
```

3. Write a prepared query (i.e. a query with a “?”) where the parameter is a student ID and the query returns tuples, where each tuple consists of

    a. A phase title
    b. A student’s first name and last name
    c. The grade that the student got for his solution to that phase. If student x has not submitted a solution for phase y then there will be no tuple (y, x, grade)

Here is a solution. You can produce other solutions by use of JOIN

```
SELECT    p.title, s.first_name, s.last_name, l.grade
FROM      Phase p, Solution l, Team t, student s
WHERE     p.id = l.phase AND l.id = t.solution
          AND t.student = s.id AND s.id = ?
```

4. Write a query that lists each student’s name and his/her total project grade, where the total project grade is the sum of the individual grades that the student got for his/her solution to each phase. Assume all phases have equal weight in the total project grade.

```
SELECT     s.first_name, s.last_name, SUM(l.grade)
FROM       Student s, Team t, Solution l
WHERE      s.id = t.student AND t.solution = l.id
GROUPBY s.id
```