South Valley Uni Data Base Project (Michael Emad)

Entities :

- Departments
- Professors
- Admin
- Students
- Courses
- Exams
- Question
- Evaluation

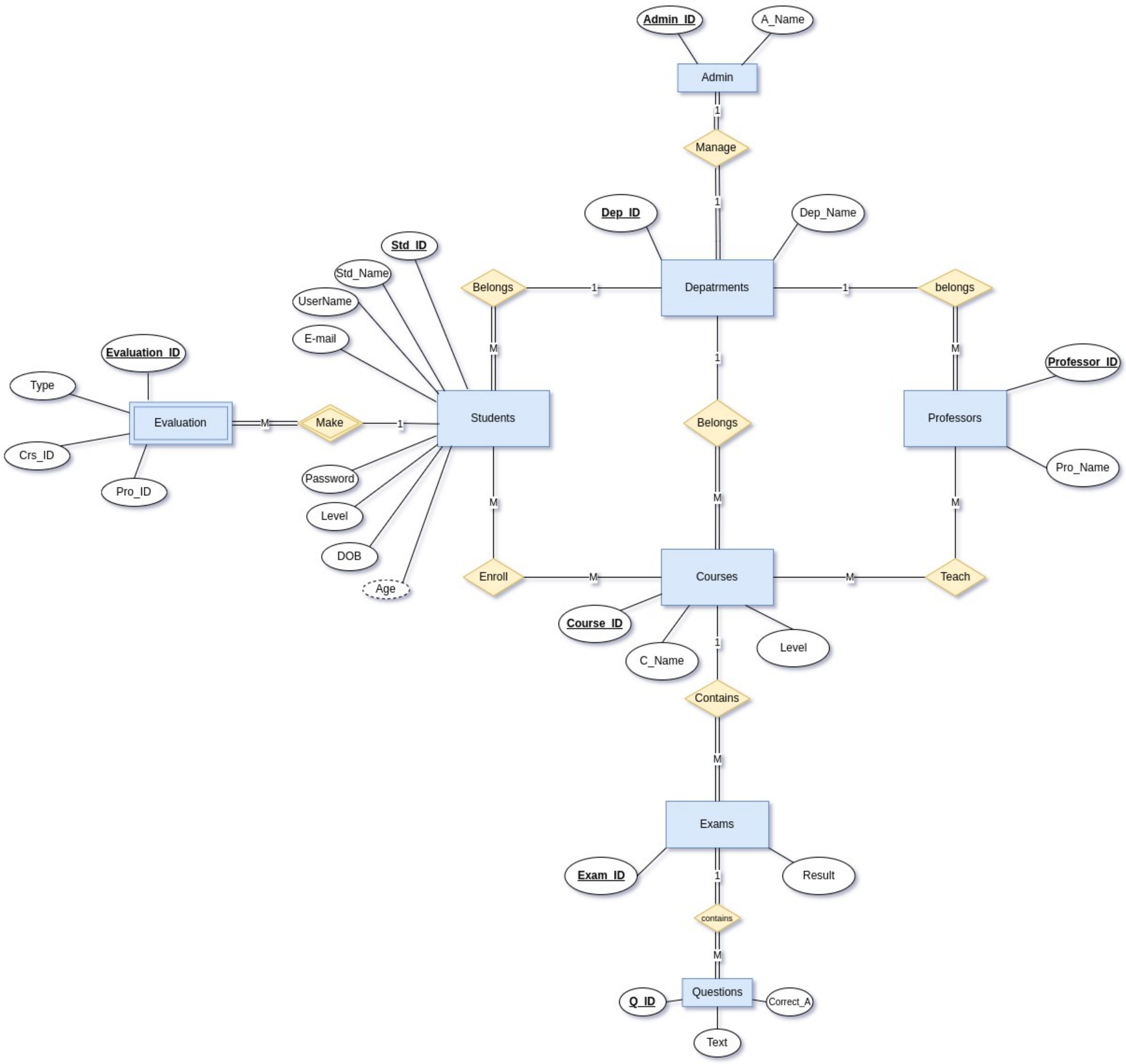
Business Rules & Relations :

- Admin & Departments → A dep can have one admin , admin lead one dep

- Professors & Departments → dep may have many prof , prof may belong to one dep

- students & Departments → students must belong to one dep , dep must have many students
- Course & Departments → course must belong to one dep , dep may have many courses
- Professors & Courses → prof may teach many courses , courses may taught by many prof
- Students & Courses → one student may enrol in many courses , courses may have many students
- Question & Courses → a question must belong to one courses , courses must have many question
- Exam & Courses → an exam must belong to one course , course may have many exams
- Exam & Question → exam must contain many questions, a question can appear in many exam

ERD :


Admin_ID
A_Name
Admin
1
Manage
1
Dep_ID
Dep_Name
Depatrments
Std_ID
Std_Name
UserName
E-mail
Belongs
1
1
belongs
M
M
Evaluation_ID
Type
Evaluation
M
Make
1
Students
1
Belongs
Professor_ID
Professors
Pro_Name
Crs_ID
Pro_ID
Password
Level
DOB
Age
M
M
M
Enroll
M
Courses
M
Teach
Course_ID
C_Name
Level
1
Contains
M
Exams
Exam_ID
Result
1
contains
M
Q_ID
Questions
Correct_A
Text

Mapping :

- Admin: [# admin_ID , admin_name , dep_ID (**fk**)]

- Departments: [# dep_ID , dep_name , admin_ID (**fk**)]

- Professors : [# pro_ID , pro_name , dep_ID (**fk**)]

- Students : [# std_ID , std_name , username , email , password , level , dob , dep_ID (**fk**)]

- Course : [# course_ID , c_name , level , dep_ID (**fk**)]

- ProfessorsCourse : [# pro_course_ID , pro_ID (**fk**) , course_ID (**fk**) , academic_year]

- StudentsCourse : [# std_course_ID , std_ID (**fk**) , course_ID (**fk**) , academic_year]

- Evaluation : [# evaluation_ID , std_ID (**fk**) , type , course_ID , pro_ID , rating , comments]

- Exam : [# exam_ID , course_ID (**fk**) , std_ID , date , pro_ID , results]

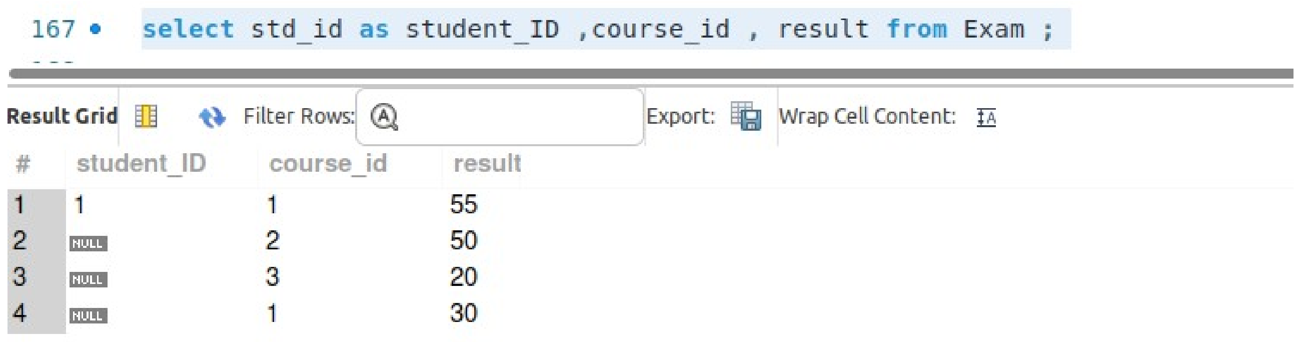
- Questions : [# question_ID , exam_ID (**fk**) , text , type , correct_answer]

Normalization :

- All tables in the third normal form (**3NF**), since there is no multi valued attributes or repeating group,there is no non-key attributes is partially dependent on the composite primary keys and no fake dependencies exist.

Queries :

- Write a query that enables the students to view their results per course



```
select std_id as student_ID ,course_id , result from Exam ;
```

#	student_ID	course_id	result
1	1	1	55
2	NULL	2	50
3	NULL	3	20
4	NULL	1	30

- Write a query that enables the head of department to see evaluation of each course and professor.

```
SELECT
    d.dep_id,
    d.dep_name,
    c.c_name,
    p.pro_name,
    ev.rating,
    ev.comments,
    ev.type
FROM
    Department d
JOIN
    Course c ON d.dep_id = c.dep_id
JOIN
    Evaluation ev ON c.course_id = ev.course_id
JOIN
    Professor p ON ev.pro_id = p.pro_id
JOIN
    Admin h ON d.head_id = h.admin_id
WHERE
    h.admin_id = 1;
```

- Write a query that enables you to get top 10 high scores per course .

```
    SELECT
    se.course_id,
    c.c_name,
    se.std_id,
    s.std_name,
    se.result
FROM
    (SELECT
        e.course_id,
        se.std_id,
        se.result,
        ROW_NUMBER() OVER (PARTITION BY e.course_id ORDER BY se.result DESC) as rank
     FROM
        StudentExam se
     JOIN
        Exam e ON se.exam_id = e.exam_id) se
JOIN
    Course c ON se.course_id = c.course_id
JOIN
    Student s ON se.std_id = s.std_id
WHERE
    se.rank <= 10;
```

- Write a query to get the highest evaluation professor from the set of professors teaching the
same course .

```
SELECT
    e.course_id,
    c.c_name,
    e.pro_id,
    p.pro_name,
    AVG(ev.rating) as average_rating
FROM
    Evaluation ev
JOIN
    Course c ON ev.course_id = c.course_id
JOIN
    Professor p ON ev.pro_id = p.pro_id
JOIN
    Exam e ON ev.course_id = e.course_id AND ev.pro_id = e.pro_id
WHERE
    ev.type = FALSE  -- FALSE indicates professor evaluation
GROUP BY
    e.course_id,
    c.c_name,
    e.pro_id,
    p.pro_name
ORDER BY
    average_rating DESC
LIMIT 1;
```