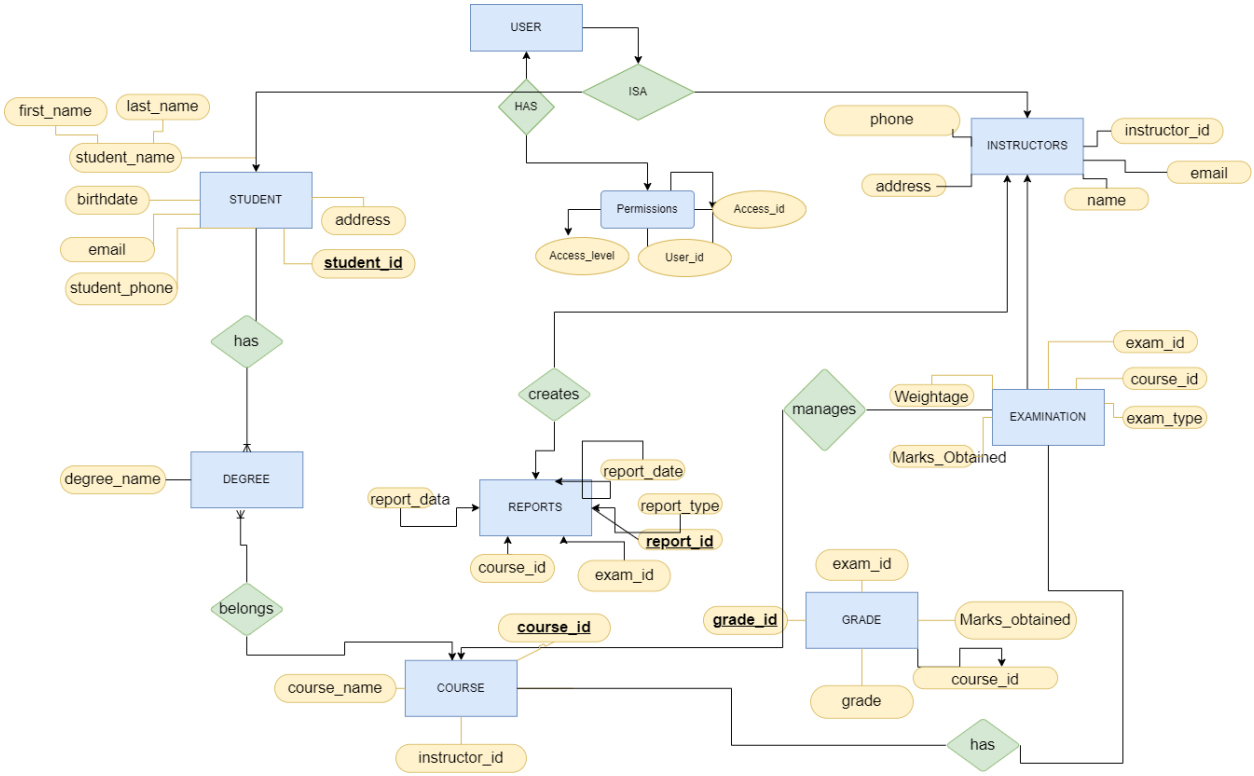
PROJECT NUMBER 14: GRADING MANAGEMENT SYSTEM

Bhavishya Garg 2020B3A71425P
Anushka Patil 2020B3A70767P

VIDEO LINK
https://drive.google.com/drive/folders/17kFWG4jVJQV4qJTXxtgO8x1c4JzLGZR9?usp=sharing



ENTITIES AND ATTRIBUTES

Entity 1

Name: User
Attributes: Access_ID,User_ID, Access_Level,
PK: Access_ID

Entity 2

Name: Student
Attributes: student_id, first_name, last_name, birth_date, email, phone, address, degree_name
PK: student_id

Entity 3
Name: Instructor
Attributes: instructor_id, name,gender,email,phone,address,
PK: instructor_id

Entity 4
Name: Degree
Attributes: degree_name
PK: degree_name

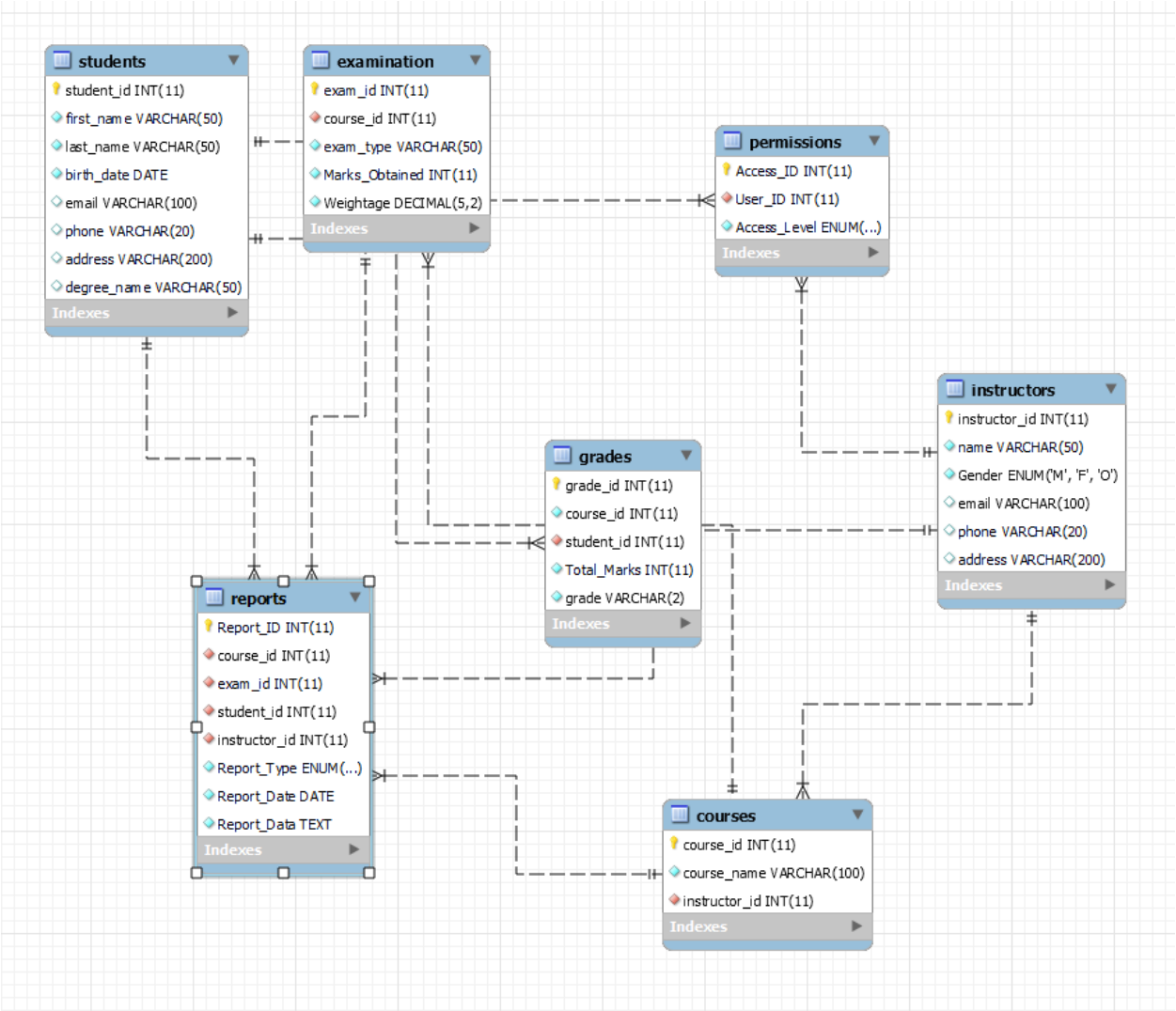
Entity 5
Name: Reports
Attributes: course_id, exam_id, student_id, instructor_id, Report_Type, Report_Date, Report_Data
PK:report_id

Entity 6
Name: Grade
Attributes: grade_id, course_id, student_id, Total_Marks, Grade
PK: grade_id

Entity 8
Name: Examination
Attributes: xam_id, course_id, exam_type, Marks_Obtained, Weightage
PK: exam_id

Entity 9
Name : Courses
Attributes: course_id,course_name, instructor_id
PK : course_id

Relational Schema



RELATIONS

Checking for Normal Forms

Table 1: Student

Attributes: student_id,first_name,last_name,birth_date,email,phone,address
student_id → first_name,last_name,birth_date,email,phone,address

Candidate Key: {student_id}
Non-prime attributes: {first_name,last_name,birth_date,email,phone,address }

Checking for 2NF

The candidates keys are { student_id}
The set of key attributes are: { student_id }
For each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes
Checking FD: student_id → first_name,last_name,birth_date,email,phone,address

Checking for 3NF

The candidates keys are { student_id},
The set of key attributes are: {student_id }
for each FD, check whether the LHS is super key or the RHS are all key attributes
checking functional dependency
student_id → first_name,last_name,birth_date,email,phone,address

Table 2: Grade

Attributes: grade_id,exam_id,student_id, Marks_Obtained,Grade
grade_id → exam_id,student_id, Marks_Obtained,Grade

Checking for 2NF

The candidates keys are { grade_id}
The set of key attributes are: { grade_id }
For each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes
Checking FD: grade_id → exam_id,student_id, Marks_Obtained,Grade

Checking for 3NF

The candidates keys are { grade_id},
The set of key attributes are: {grade_id }
for each FD, check whether the LHS is super key or the RHS are all key attributes
checking functional dependency
grade_id → exam_id,student_id, Marks_Obtained,Grade

Table 3: Instructor

Attributes: instructor_id, name,gender,email,phone,address,
instructor_id → name,gender,email,phone,address,

Checking for 2NF

The candidates keys are {instructor_id }
The set of key attributes are: {instructor_id}
For each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes
Checking FD: instructor_id → name,gender,email,phone,address,

Checking for 3NF

The candidates keys are {instructor_id }
The set of key attributes are: {instructor_id}
for each FD, check whether the LHS is super key or the RHS are all key attributes
checking functional dependency
instructor_id → name,gender,email,phone,address

Table 4: Permissions

Attributes: Access_id,user_id,Access_Level
Access_id → user_id, Access_Level

Checking for 2NF

The candidates keys are {Access_id }
The set of key attributes are: {Access_id}
For each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes
Checking FD: Access_id → user_id, Access_Level

Checking for 3NF

The candidates keys are {Access_id}
The set of key attributes are: {Access_id}
for each FD, check whether the LHS is super key or the RHS are all key attributes
checking functional dependency
Access_id → user_id, Access_Level

Table 5: Course

Attributes: course_id,course_name, instructor_id
Candidate key: {course_id}
Course_id → course_name, instructor_id

Checking for 2NF

The candidates keys are { course_id }
The set of key attributes are: {course_id}
For each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes
Checking FD: Course_id → course_name, instructor_id

Checking for 3NF

The candidates keys are { course_id}
The set of key attributes are: { course_id}
for each FD, check whether the LHS is super key or the RHS are all key attributes
checking functional dependency
Course_id → course_name, instructor_id

Table 6: Reports

Attributes: Report_ID,course_id, exam_id,student_id ,instructor_id ,Report_Type ,Report_Date,
Report_Data

Report_ID→ course_id, exam_id,student_id ,instructor_id,Report_Type ,Report_Date, Report_Data
Candidate Key :{Report_ID}
Non-prime attributes: {course_id, exam_id,student_id ,instructor_id}

Checking for 2NF

The candidates keys are {Report_ID}

The set of key attributes are:{course_id, exam_id,student_id ,instructor_id}
For each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes
Checking FD: Report_ID→ course_id, exam_id,student_id ,instructor_id

Checking for 3NF

The candidates keys are {Report_ID}
The set of key attributes are:{course_id, exam_id,student_id ,instructor_id}
For each FD, check whether the LHS is super key or the RHS are all key attributes checking functional dependency
Report_ID→ course_id, exam_id,student_id ,instructor_id

Table 7: Examinations

Attributes: exam_id, course_id, exam_type, Marks_Obtained
exam_id → exam_type, Marks_Obtained

Candidate Key: {exam_id}
Non-prime attributes: {exam_type, Marks_Obtained, course_id}

Checking for 2NF

The candidates keys are {exam_id}
The set of key attributes are: { exam_id}
For each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes
Checking FD: exam_id --> Marks_Obtained,exam_type, course_id

Checking for 3NF

The candidates keys are {exam_id},
The set of key attributes are: {exam_id}
For each FD, check whether the LHS is super key or the RHS are all key attributes checking functional dependency
exam_id --> Marks_Obtained,exam_type, course_id

SQL Queries

- Insert a new student into the Student table

```
use grading;
-- Insert a new student into the student table
INSERT INTO Students (student_id, first_name, last_name, birth_date, email, phone, address, degree_name)
VALUES (012, 'Jake', 'Peralta', '1992-11-21', 'jakeperalta@example.com', '123-456-7890', '123 Main St', 'Computer Science');

SELECT * FROM Students WHERE student_id = 012;
```

student_id	first_name	last_name	birth_date	email	phone	address	degree_name
12	Jake	Peralta	1992-11-21	jakeperalta@example.com	123-456-7890	123 Main St	Computer Science

- Insert new instructor into the Instructor table

```
-- Insert new teacher into the Instructor table
INSERT INTO Instructors (instructor_id, name, Gender, email, phone, address)
VALUES (12, 'Anushka Patil', 'F', 'anushkapatil@gmail.com', '9175916459', '123 Main St, Anytown, USA');

SELECT * FROM Instructors WHERE instructor_id = 12
```

instructor_id	name	Gender	email	phone	address
12	Anushka Patil	F	anushkapatil@gmail.com	9175916459	123 Main St, Anytown, USA

- Update student's information in the Student table

```
-- Update the students information in the Students table
UPDATE Students SET first_name = 'Bhavishya', last_name = 'Garg', email = 'bhavishyagarg@gmail.com', phone = '555-555-5555', address = '456 Main St', degree_name = 'Data Scie
WHERE student_id = 1;

SELECT * FROM Students WHERE student_id = 1;
```

student_id	first_name	last_name	birth_date	email	phone	address	degree_name
1	Bhavishya	Garg	1995-05-10	bhavishyagarg@gmail.com	555-555-5555	456 Main St	Data Science

- Retrieve all information about a particular student

```
-- Retrive all information about a particular student
SELECT * from Students where student_id = 02;
```

student_id	first_name	last_name	birth_date	email	phone	address	degree_name
2	Jane	Smith	1996-08-15	janesmith@example.com	555-555-5555	456 Oak Ave, Anytown, USA	Marketing

- Retrieve the number of students who scored above a certain grade for a specific course for a specific class.

```
-- Retrieve the number of students who scored above a certain grade for a specific course for a specific class.
SELECT COUNT(*)
FROM Grades
JOIN Courses ON Courses.course_id = Courses.course_id
WHERE Courses.course_name = 'Database Systems'
AND Grades.course_id = 1
AND Grades.Total_Marks > 70;
```

COUNT(*)
2

- Retrieve the lowest grade for a specific course for a specific class.

```
-- Retrieve the lowest grade for a specific course for a specific class.
SELECT MIN(Total_Marks) AS lowest_grade
FROM examination
JOIN grades ON Examination.course_id = Grades.course_id
WHERE Examination.course_id = 1
AND examination.exam_id = 1;
```

lowest_grade
70

- Retrieve a student's progress

```
-- Retrive a student's progress
SELECT Courses.course_name, Grades.Total_marks, Grades.grade
FROM Grades
JOIN Courses ON Grades.course_id = Courses.course_id
WHERE Grades.student_id = 1;
```

	course_name	Total_marks	grade
▶	Introduction to Programming	85	A
	Database Systems	90	A

- Update a student's grade in a course

```
-- Update a student's grade in a course.
UPDATE Grades SET grade = 'A'
WHERE student_id = 1;

SELECT grade FROM Grades WHERE student_id = 1;
```

	grade
▶	A

- Delete an instructor from the Instructor table

```
-- Delete Instructor from Instructor table
DELETE FROM Instructor WHERE instructor_id = 2;
```

It was a pleasure to work on this project, thank you for all your guidance throughout!