

ASSIGNMENT

DBI202 - Student Grading Management

Sub-System

**Student Name: Hoàng Thế Anh**

**Roll Number: HE160176**

**Class: SE1647**

1. **Description of the database**

- The entity Student has StudentID is primary key

- The entity Group has GroupID is primary key

- The entuty Group\_Student has StudentID and GroupID is primary key

- The entity Lecture has LectureID is primary key

- The entity Subject has SubjectID is primary key

- The entity Assessment has AssessmentID is primary key

- The entity Assessment\_Info has CategoryID is primary key

- The entity Subject\_Semester has SubjectSemesterID is primary key

- The entity Result\_Table has ResultTableID is primary key

- Each student can join many group and group has many student. So we have a table Group\_Student

+ The relationship between Lecture and Group is one-to-many

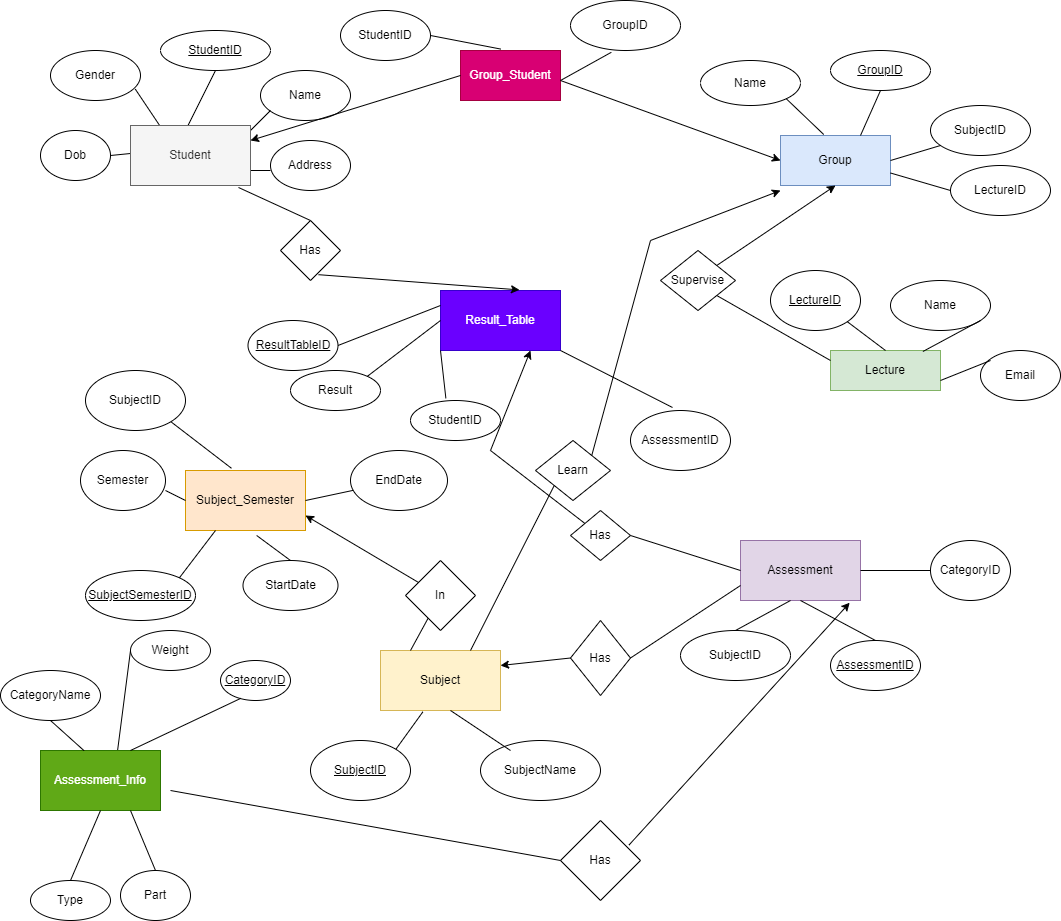
+ The relationship between Subject and Group is one-to-many

+ The relationship between Assessment\_Info and Assessment is one-to-many

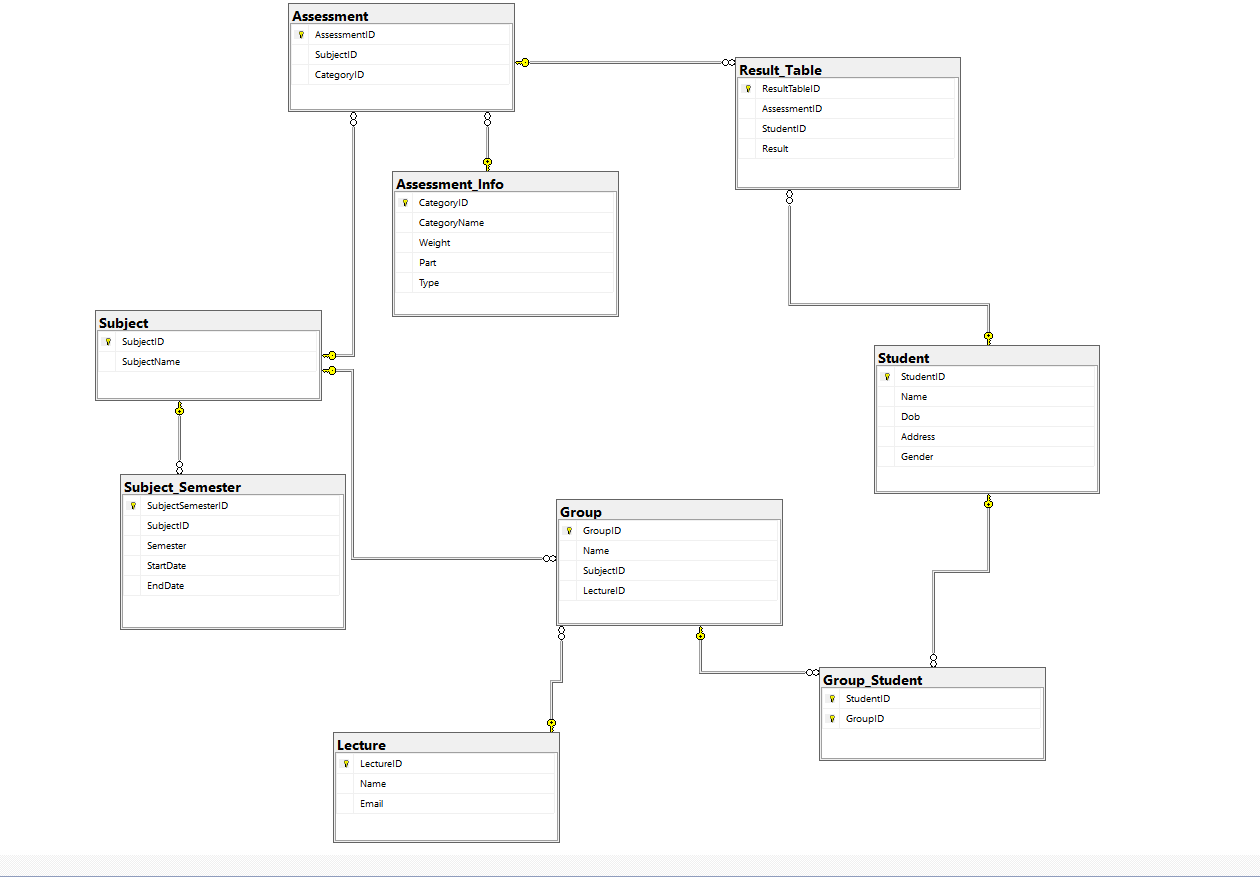
+ The relationship between Student and Result\_Table is one-to-many

+ The relationship between Assessment and Result\_Table is one-to-many

1. **An ERD ( Entity Relationship Diagram )**



1. **Relational**

****

1. **Create Table from ERD**

CREATE TABLE Student(

StudentID int PRIMARY KEY NOT NULL,

Name nvarchar(150) NOT NULL,

Dob date NOT NULL,

Address nvarchar(150) NOT NULL,

Gender bit NOT NULL

)

CREATE TABLE Subject(

SubjectID int PRIMARY KEY NOT NULL,

SubjectName nvarchar(150) NOT NULL

)

CREATE TABLE Lecture(

LectureID int PRIMARY KEY NOT NULL,

Name nvarchar(150) NOT NULL,

Email varchar(50) NOT NULL

)

CREATE TABLE [Group](

GroupID varchar(50) PRIMARY KEY NOT NULL,

Name nvarchar(150) NOT NULL,

SubjectID int NOT NULL

FOREIGN KEY (SubjectID) REFERENCES Subject(SubjectID),

LectureID int NOT NULL

FOREIGN KEY (LectureID) REFERENCES Lecture (LectureID),

)

CREATE TABLE Group\_Student(

StudentID int NOT NULL,

GroupID varchar(50) NOT NULL,

PRIMARY KEY(StudentID, GroupID),

FOREIGN KEY (StudentID) REFERENCES Student (StudentID),

FOREIGN KEY (GroupID) REFERENCES [Group] (GroupID),

)

CREATE TABLE Subject\_Semester(

SubjectSemesterID int PRIMARY KEY NOT NULL,

SubjectID int NOT NULL

FOREIGN KEY (SubjectID) REFERENCES [Subject] (SubjectID),

Semester nvarchar(150) NOT NULL,

StartDate date NOT NULL,

EndDate date NOT NULL

)

CREATE TABLE Assessment\_Info(

CategoryID int PRIMARY KEY NOT NULL,

CategoryName nvarchar(150) NOT NULL,

Weight float NOT NULL,

Part int NOT NULL,

Type nvarchar(50) NOT NULL

)

CREATE TABLE Assessment(

AssessmentID int PRIMARY KEY NOT NULL,

SubjectID int NOT NULL,

CategoryID int NOT NULL

FOREIGN KEY (SubjectID) REFERENCES [Subject] (SubjectID),

FOREIGN KEY (CategoryID) REFERENCES Assessment\_Info (CategoryID)

)

CREATE TABLE Result\_Table(

ResultTableID int PRIMARY KEY NOT NULL,

AssessmentID int NOT NULL

FOREIGN KEY (AssessmentID) REFERENCES Assessment (AssessmentID),

StudentID int NOT NULL,

FOREIGN KEY (StudentID) REFERENCES Student (StudentID),

Result float NOT NULL,

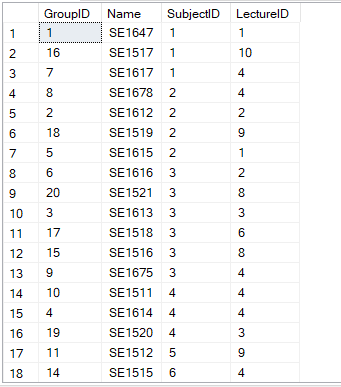
)

1. **Query**

--1. A query that uses ORDER BY to sort SubjectID ascending order--

SELECT \* FROM [Group]

ORDER by SubjectID ASC



--2. A query that uses INNER JOINS to check the Subject that Student learn about

SELECT s.Name, su.SubjectName

FROM Student s INNER JOIN Group\_Student gs ON s.StudentID = gs.StudentID

INNER JOIN [Group] g ON gs.GroupID = g.GroupID

INNER JOIN Subject su ON su.SubjectID = g.SubjectID

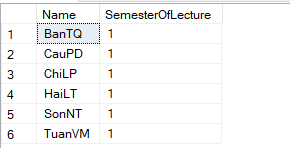


--3. A query that uses aggregate functions count the semester of lecture

SELECT l.Name, COUNT (ss.SubjectID) AS SemesterOfLecture

FROM Lecture l INNER JOIN Subject\_Semester ss ON ss.SubjectID = l.LectureID

GROUP BY l.Name



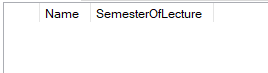
--4. A query that uses the GROUP BY and HAVING clauses to check lecture join more than 1 semester

SELECT l.Name, COUNT (ss.SubjectID) AS SemesterOfLecture

FROM Lecture l INNER JOIN Subject\_Semester ss ON ss.SubjectID = l.LectureID

GROUP BY l.Name

HAVING COUNT (ss.SubjectID) > 1



--5. A query that uses a sub-query as a relation to check 8 student have lowest avarage

SELECT TOP 8 tb1.[NameOfStudent], tb1.AVG

FROM

(SELECT s.StudentID ,s.Name AS [NameOfStudent], SUM (rt.Result \* ai.Weight) AS[AVG]

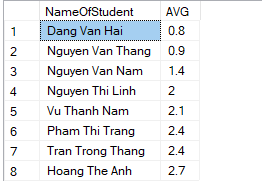
FROM Student s INNER JOIN Result\_Table rt ON s.StudentID = rt.ResultTableID

INNER JOIN Assessment a ON rt.AssessmentID = a.AssessmentID

INNER JOIN Assessment\_Info ai ON ai.CategoryID = a.CategoryID

GROUP BY s.StudentID, s.Name) tb1

ORDER BY tb1.AVG



--6. A query that uses a sub-query in the WHERE clause check the student have lowest average on MAD101

SELECT TOP 1 tb1.[Student Name], tb1.AVG, tb1.[Subject Name]

FROM

(SELECT s.StudentID, s.Name AS [Student Name], su.SubjectName AS [Subject Name],SUM(rt.Result \* ai.Weight) AS[AVG]

FROM Student s INNER JOIN Result\_Table rt ON s.StudentID = rt.StudentID

INNER JOIN Assessment a ON rt.AssessmentID = a.AssessmentID

INNER JOIN Subject su ON su.SubjectID = a.SubjectID

INNER JOIN Assessment\_Info ai ON ai.CategoryID = a.CategoryID

WHERE su.SubjectName = 'MAD101'

GROUP BY s.StudentID, s.Name,su.SubjectName) tb1

ORDER BY tb1.AVG

q6

--7. A query that uses partial matching in the WHERE clause to find the student have the last character is a

SELECT \*

FROM Student s

WHERE s.Name LIKE '%a'

q7

--8. A query that check average of student

SELECT s.StudentID,s.Name AS [Student Name], su.SubjectName AS [Subject Name],SUM(rt.Result \* ai.Weight) AS[AVG]

FROM Student s INNER JOIN Result\_Table rt ON s.StudentID = rt.StudentID

INNER JOIN Assessment a ON rt.AssessmentID = a.AssessmentID

INNER JOIN Subject su ON su.SubjectID = a.SubjectID

INNER JOIN Assessment\_Info ai ON ai.CategoryID = a.CategoryID

GROUP BY s.StudentID, s.Name,su.SubjectName

ORDER BY s.StudentID

