**Student Grading Management Sub-System**

Name: LÊ HOÀNG THÁI

StudentID: HE163249

Class:SE1647

DBI202 – SLOT 3

LECTURER: Ngô Tùng Sơn

I.DESCRIPTION OF THE DATABASE

How many course student study?

one student can join many groups

a course can have many students

Calculate AVG score of each course

II. An ERD (Entity Relationship Diagram)

Diagram

Description automatically generated

III.The relational schema derived from the ERD that is at least in 3NDiagram

Description automatically generated

IV.The set of database statements used to create the tables used in your database. You do NOT need to include all the data and insert statements

CREATE TABLE [Student](

StudentID VARCHAR(10) PRIMARY KEY,

StudentName NVARCHAR(30) NOT NULL,

gender BIT,

DoB DATE NOT NULL,

Address VARCHAR(30) NOT NULL,

)

CREATE TABLE [Course] (

CourseID INT PRIMARY KEY,

CourseName NVARCHAR(30) NOT NULL,

)

CREATE TABLE [Lecturer] (

LecturerID INT PRIMARY KEY,

LecturerName NVARCHAR (30) NOT NULL,

)

CREATE TABLE [Group] (

GroupID VARCHAR(10) PRIMARY KEY,

GroupName NVARCHAR(30) NOT NULL,

)

CREATE TABLE [Group\_student](

StudentID VARCHAR(10) NOT NULL FOREIGN KEY REFERENCES [Student](StudentID),

GroupID VARCHAR(10) NOT NULL FOREIGN KEY REFERENCES [Group](GroupID)

)

CREATE TABLE [Group\_lecturer](

LectureID INT FOREIGN KEY REFERENCES [Lecturer](LecturerID),

GroupID VARCHAR(10)NOT NULL FOREIGN KEY REFERENCES[Group](GroupID)

)

CREATE TABLE [Group\_course](

CourseID INT FOREIGN KEY REFERENCES [Course](CourseID),

GroupID VARCHAR(10)NOT NULL FOREIGN KEY REFERENCES[Group](GroupID)

)

CREATE TABLE [Lecturer\_Course](

LectureID INT FOREIGN KEY REFERENCES [Lecturer](LecturerID),

CourseID INT FOREIGN KEY REFERENCES [Course](CourseID)

)

CREATE TABLE Assessment(

AssessmentID INT PRIMARY KEY,

AssessmentName VARCHAR(150),

WEIGHT FLOAT,

CourseID INT FOREIGN KEY REFERENCES [Course](CourseID)

)

CREATE TABLE RESULT(

RESULTID INT PRIMARY KEY,

StudentID VARCHAR(10) NOT NULL FOREIGN KEY REFERENCES [Student](StudentID),

AssessmentID INT FOREIGN KEY REFERENCES [Assessment](AssessmentID),

CourseID INT FOREIGN KEY REFERENCES [Course](CourseID),

Score FLOAT

)

V. Queries that demonstrate the usefulness of the database. Also state why and when each query would be used. The following must be demonstrated by at least one of your queries:

1.A query that uses ORDER BY

---Sort student list by Date of birth

SELECT \* FROM Student s

ORDER BY DoB

Table

Description automatically generated with medium confidence

2.Aggregate function

---Student can check AVG of each course

SELECT r.StudentID,a.CourseID, SUM(r.Score \* a.WEIGHT) AS [AVG]

FROM Assessment a INNER JOIN RESULT r ON a.AssessmentID = r.AssessmentID

WHERE r.StudentID = 'HE160'

GROUP BY r.StudentID,a.CourseID

ORDER BY [AVG]

Table

Description automatically generated

3.INNER JOIN

---Student can check groupname,lecturername,coursename

SELECT s.StudentID,s.StudentName,g.GroupName,l.LecturerName,c.CourseName FROM Student s INNER JOIN [Group\_student] gs ON s.StudentID = gs.StudentID

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

INNER JOIN [Group\_lecturer] gl ON g.GroupID = gl.GroupID

INNER JOIN [Lecturer] l ON gl.LectureID = l.LecturerID

INNER JOIN [Lecturer\_Course] ls ON l.LecturerID = ls.LectureID

INNER JOIN [Course] c ON ls.CourseID = c.CourseID

WHERE s.StudentID = 'HE160';

Table

Description automatically generated

4. self-join

---matches student that are from the same address

SELECT s1.StudentID AS [StudentID1] ,s2.StudentID AS[StudentID2],s1.Address

FROM Student s1,Student s2

WHERE s1.StudentID <> s2.StudentID

AND s1.Address = s2.Address

ORDER BY s1.Address;

Table

Description automatically generated

5.Left Join

--- Check student if student join group or not

SELECT s.StudentID,s.StudentName,gs.GroupID

FROM Student s LEFT JOIN Group\_student gs ON s.StudentID = gs.StudentID

WHERE gs.GroupID IS NULL

Table

Description automatically generated

6.Trigger

--- LIMIT LECTURER < 4

GO

CREATE TRIGGER TRIGER1

ON Lecturer

AFTER INSERT

AS

DECLARE @COUNT NVARCHAR(10)

SET @COUNT = (SELECT COUNT (LecturerID) FROM Lecturer)

IF (@COUNT > 4)

BEGIN

PRINT 'LECTURER LIMITED IS 4'

ROLLBACK

END

INSERT INTO Lecturer(LecturerID,LecturerName)

VALUES (5,'CAO BA KI')

Text

Description automatically generated

7.PROCEDURE

--STORE PROCEDURE : Infomation of each student

CREATE PROCEDURE INFO\_STUDENTS AS

SELECT s.StudentID,s.StudentName,g.GroupName,l.LecturerName,c.CourseName

FROM Student s INNER JOIN [Group\_student] gs ON s.StudentID = gs.StudentID

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

INNER JOIN [Group\_lecturer] gl ON g.GroupID = gl.GroupID

INNER JOIN [Lecturer] l ON gl.LectureID = l.LecturerID

INNER JOIN [Lecturer\_Course] ls ON l.LecturerID = ls.LectureID

INNER JOIN [Course] c ON ls.CourseID = c.CourseID

ORDER BY s.StudentName

EXEC INFO\_STUDENTS

Table

Description automatically generated