**Student Grading Management Sub-System Report**

**HE161372**

**Trịnh Tiến Đạt**

**I, Database ‘s brief description**

Student management database is divided into 12 tables to store data about the relationship of students and schools (attendance, grades, personal information, .....)

And the following is a summary of these tables :

1.Student : 52 records (Student personal information)

2.Subject : 4 Records ( Subject Name)

3.Group : 1 Records (Group Name)

4.Lecture : 4 Records (Lecture information)

5.Semester: 1 Records (Semester describe)

6.Semester\_Subject : 4 Records (The subject in a season )

7.Subject\_Detail: 14 Records (Subject Detail)

8. Subject\_Mark : 14 Records (Grade weight in a subject)

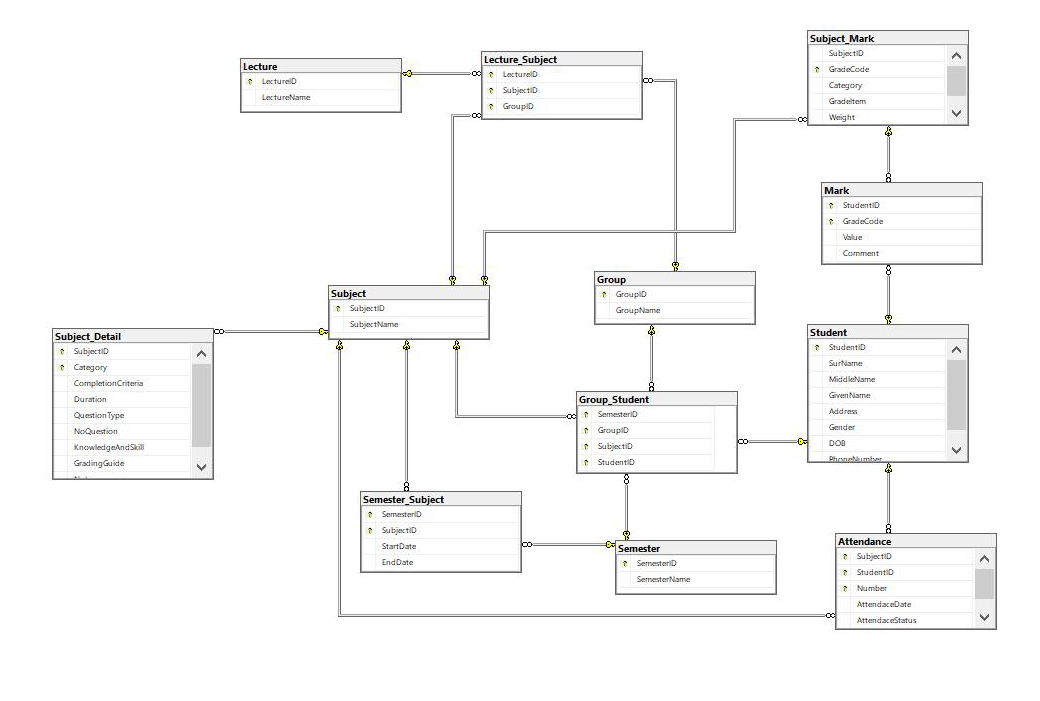
9. Mark : 762 Record (Student Mark)

10.Attendance: 3.480 Records (Attendace Report)

11. Group\_Student : 118 Records (Describe a student in which group)

12: Lecture \_Subject : 4 Records (Describe the subject that lecture teach)

II) **An ERD (Entity Relationship Diagram) that fully describes the database**



**III. The set of database statements used to create the tables used in database.**

CREATE TABLE Student (

StudentID VARCHAR (8) NOT NULL,

SurName NVARCHAR(15) NOT NULL,

MiddleName NVARCHAR(15) NOT NULL,

GivenName NVARCHAR(15) NOT NULL,

[Address] NVARCHAR(50) NOT NULL,

Gender BIT NOT NULL,

DOB DATE NOT NULL,

PhoneNumber CHAR (10) NOT NULL,

PRIMARY KEY(StudentID),

)

CREATE TABLE Lecture (

LectureID VARCHAR(10) NOT NULL,

LectureName NVARCHAR (30) NOT NULL,

PRIMARY KEY (LectureID),

)

CREATE TABLE Semester (

SemesterID VARCHAR(10) NOT NULL,

SemesterName VARCHAR (20) NOT NULL,

PRIMARY KEY (SemesterID),

)

CREATE TABLE [Subject](

SubjectID VARCHAR (10) NOT NULL,

SubjectName NVARCHAR (30) NOT NULL,

PRIMARY KEY (SubjectID),

)

CREATE TABLE Subject\_Mark (

SubjectID VARCHAR (10) NOT NULL,

GradeCode VARCHAR (15) NOT NULL,

Category VARCHAR (20) NOT NULL,

GradeItem VARCHAR(20) NOT NULL,

[Weight] INT ,

PRIMARY KEY (GradeCode),

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

)

CREATE TABLE Mark (

StudentID VARCHAR (8) NOT NULL,

GradeCode VARCHAR (15) NOT NULL,

[Value] FLOAT,

Comment VARCHAR(50),

PRIMARY KEY (StudentID,GradeCode),

FOREIGN KEY (GradeCode) REFERENCES [Subject\_Mark](GradeCode),

FOREIGN KEY (StudentID) REFERENCES [Student](StudentID),

)

CREATE TABLE Subject\_Detail(

SubjectID VARCHAR (10) NOT NULL,

Category VARCHAR (20) NOT NULL,

CompletionCriteria CHAR(3),

Duration VARCHAR (20),

QuestionType VARCHAR(30),

NoQuestion TINYINT,

KnowledgeAndSkill VARCHAR(40),

GradingGuide VARCHAR(30),

Note VARCHAR(50),

PRIMARY KEY(SubjectID,Category),

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

)

CREATE TABLE [Group](

GroupID VARCHAR (8) NOT NULL,

GroupName VARCHAR (25) NOT NULL,

PRIMARY KEY (GroupID),

)

CREATE TABLE Group\_Student(

SemesterID VARCHAR(10) NOT NULL,

GroupID VARCHAR (8) NOT NULL,

SubjectID VARCHAR (10) NOT NULL,

StudentID VARCHAR (8) NOT NULL,

PRIMARY KEY (StudentID,SemesterID,GroupID,SubjectID),

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

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

FOREIGN KEY (StudentID) REFERENCES [Student](StudentID),

FOREIGN KEY (SemesterID) REFERENCES [Semester](SemesterID),

)

CREATE TABLE Lecture\_Subject(

LectureID VARCHAR(10) NOT NULL,

SubjectID VARCHAR (10) NOT NULL,

GroupID VARCHAR (8) NOT NULL,

PRIMARY KEY (LectureID,SubjectID,GroupID),

FOREIGN KEY (LectureID) REFERENCES [Lecture](LectureID),

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

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

)

CREATE TABLE Attendance (

SubjectID VARCHAR (10) NOT NULL,

StudentID VARCHAR (8) NOT NULL,

Number INT,

AttendaceDate DATE NOT NULL,

[AttendaceStatus] BIT ,

PRIMARY KEY ( StudentID,SubjectID,Number),

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

FOREIGN KEY (StudentID) REFERENCES [Student](StudentID),

)

CREATE TABLE Semester\_Subject (

SemesterID VARCHAR(10) NOT NULL,

SubjectID VARCHAR (10) NOT NULL,

StartDate DATE NOT NULL,

EndDate DATE NOT NULL,

PRIMARY KEY(SemesterID,SubjectID),

FOREIGN KEY (SemesterID) REFERENCES [Semester](SemesterID),

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

)

**IV,10 queries that demonstrate the usefulness of the database**

1. **Query that uses ORDER BY**

SELECT \* FROM Student

ORDER BY DOB DESC

(USE TO SORT STUDENT BY AGE)

1. **Query that uses INNER JOINS**

SELECT CONCAT(s.SurName , ' ', s.MiddleName , ' ' , s.GivenName) AS [FullName] ,att.Number,att.AttendaceDate,att.AttendaceStatus

FROM Student s

INNER JOIN Attendance att ON att.StudentID = s.StudentID

WHERE s.StudentID = 'HE160929' AND att.SubjectID = 'CSD201'

(USE TO QUERY ALL THE ATTENDACE OF A STUDENT IN A SUBJECT)

**3. Query that uses aggregate functions**

SELECT CONCAT(s.SurName , ' ', s.MiddleName , ' ' , s.GivenName) AS [FullName] ,sub.SubjectName,COUNT(att.AttendaceStatus) AS [TOTAL]

FROM Student s

INNER JOIN Attendance att ON att.StudentID = s.StudentID

INNER JOIN [Subject] sub ON sub.SubjectID = att.SubjectID

WHERE att.AttendaceStatus = '1'

GROUP BY s.SurName,s.MiddleName,s.GivenName,s.StudentID,sub.SubjectName

HAVING s.StudentID = 'HE160929'

(USE TO QUERY TOTAL ATTENDANCE OF A STUDENT IN ALL SUBJECT)

**4. Query that uses the GROUP BY and HAVING**

SELECT CONCAT(s.SurName , ' ', s.MiddleName , ' ' , s.GivenName) AS [FullName],Sm.Category,m.[Value]

FROM Student s

LEFT JOIN Mark m ON m.StudentID = s.StudentID

LEFT JOIN Subject\_Mark sm ON sm.GradeCode = m.GradeCode

GROUP BY s.SurName,s.MiddleName,s.GivenName,sm.Category,m.[Value],s.StudentID,sm.SubjectID

HAVING s.StudentID = 'HE161275' AND sm.SubjectID LIKE 'DBI202'

(USE TO QUERY ALL RESULT OF A SUBJECT THAT A STUDENT FINISHED)

**5.Query that uses a sub-query**

SELECT s.SubjectID,s.SubjectName,sd.Category,sd.CompletionCriteria,sd.Duration,sd.GradingGuide,sd.KnowledgeAndSkill,sd.NoQuestion,sd.QuestionType,sd.Note

FROM [Subject] s

INNER JOIN Subject\_Detail sd ON s.SubjectID = sd.SubjectID

(USE TO QUERY SUBJECT DETAIL)

**6.Query that use GROUP BY and ORDER BY**

SELECT gs.SubjectID,sub.SubjectName,s.StudentID,CONCAT(s.SurName , ' ', s.MiddleName , ' ' , s.GivenName) AS [FullName] FROM Student s

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

INNER JOIN [Subject] sub ON sub.SubjectID = gs.SubjectID

GROUP BY s.StudentID,s.SurName,s.MiddleName,s.GivenName,gs.StudentID,gs.SubjectID,sub.SubjectName

ORDER BY gs.SubjectID DESC

(USE TO SHOW THE STUDENT LIST)

**7.Query that use WHERE**

SELECT \* FROM Student

WHERE Gender = '0'

(USE TO SHOW FEMALE STUDENT)

**8. Query that use GROUP BY**

SELECT CONCAT(s.SurName , ' ', s.MiddleName , ' ' , s.GivenName) AS [ FullName],sub.SubjectName,SUM(m.[value] \* sm.[weight])/100 AS [Total]

FROM Student s

INNER JOIN Mark m ON m.StudentID = s.StudentID

INNER JOIN Subject\_Mark sm ON sm.GradeCode = m.GradeCode

INNER JOIN [Subject] sub ON sub.SubjectID = sm.SubjectID

GROUP BY s.SurName,s.MiddleName,s.GivenName,sub.SubjectName,s.StudentID,sub.SubjectID

HAVING sub.SubjectID LIKE 'DBI202'

ORDER BY [Total] DESC

(USE TO QUERY ALL A SUBJECT GRADE OF WHOLE CLASS IN DESCENDING ORDER)

**9. Query that use GROUP BY and aggregate functions**

SELECT SUM(m.[value] \* sm.[weight])/100 AS [TOTAL] FROM Student s

INNER JOIN Mark m ON m.StudentID = s.StudentID

INNER JOIN Subject\_Mark sm ON sm.GradeCode = m.GradeCode

INNER JOIN [Subject] sub ON sub.SubjectID = sm.SubjectID

GROUP BY sub.SubjectName,s.StudentID,sub.SubjectID

HAVING s.StudentID = 'HE160153' AND sub.SubjectID LIKE 'DBI202'

(USE TO SHOW THE STUDENT TOTAL SCORE)

**10. Query that use GROUP BY and aggregate functions**

SELECT SUM(sm.[Weight] \* m.[Value])/100 AS [FE] FROM Student s

LEFT JOIN Mark m ON m.StudentID = s.StudentID

LEFT JOIN Subject\_Mark sm ON sm.GradeCode = m.GradeCode

GROUP BY m.[Value],s.StudentID,sm.SubjectID,sm.[Weight],sm.GradeCode

HAVING s.StudentID LIKE '@Subject\_ID' AND sm.SubjectID LIKE '@Subject\_ID' AND sm.GradeCode LIKE '%FE';

(USE TO CHECK FINAL EXAM MARK FOR A STUDENT)

**V.The trigger, store procedure, and the index**

**Trigger**

CREATE OR ALTER TRIGGER UTG\_InsertMark ON Mark

FOR INSERT

AS

DECLARE @Grade INT ;

SELECT @Grade =[Value] FROM Mark WHERE GradeCode NOT LIKE 'LAB211\_EX'

IF (@Grade > 10)

BEGIN

PRINT 'INVALID VALUE GRADE,PLEASE TRY AGAIN'

ROLLBACK TRAN

END

**Stored Procedure**

CREATE OR ALTER PROC USP\_Show\_Lecture\_List

AS

SELECT sem.SemesterName,sub.SubjectID,sub.SubjectName,ss.StartDate,ss.EndDate,ls.LectureID,l.LectureName FROM Semester sem

INNER JOIN Semester\_Subject ss ON sem.SemesterID = ss.SemesterID

INNER JOIN [Subject] sub ON sub.SubjectID = ss.SubjectID

INNER JOIN Lecture\_Subject ls ON ls.SubjectID = sub.SubjectID

INNER JOIN Lecture l ON l.LectureID = ls.LectureID

**Indexed**

CREATE UNIQUE INDEX Index\_Student\_Name

ON Student (StudentID);