

**Student Grading Management**

**Sub-System**

DBI202 – Lecture: Sonnt5

Đào Vũ Hiệp

HE163693 – IA1604

Mục lục

I. Mô tả cơ sở dữ liệu

II. Mô hình ERD (Sơ đồ mối quan hệ thực thể)

III. Lược đồ quan hệ bắt buộc

IV. Câu lệnh tạo bảng

V. Truy vấn chứng minh tính hữu ích của cơ sở dữ liệu

VI. The trigger, store procedure, and the index (Empty)

I. Mô tả cơ sở dữ liệu

Cơ sở dữ liệu MarkManagerFPT là một cơ sở dữ liệu chứa các data về các môn học và điểm thành phần của sinh viên.

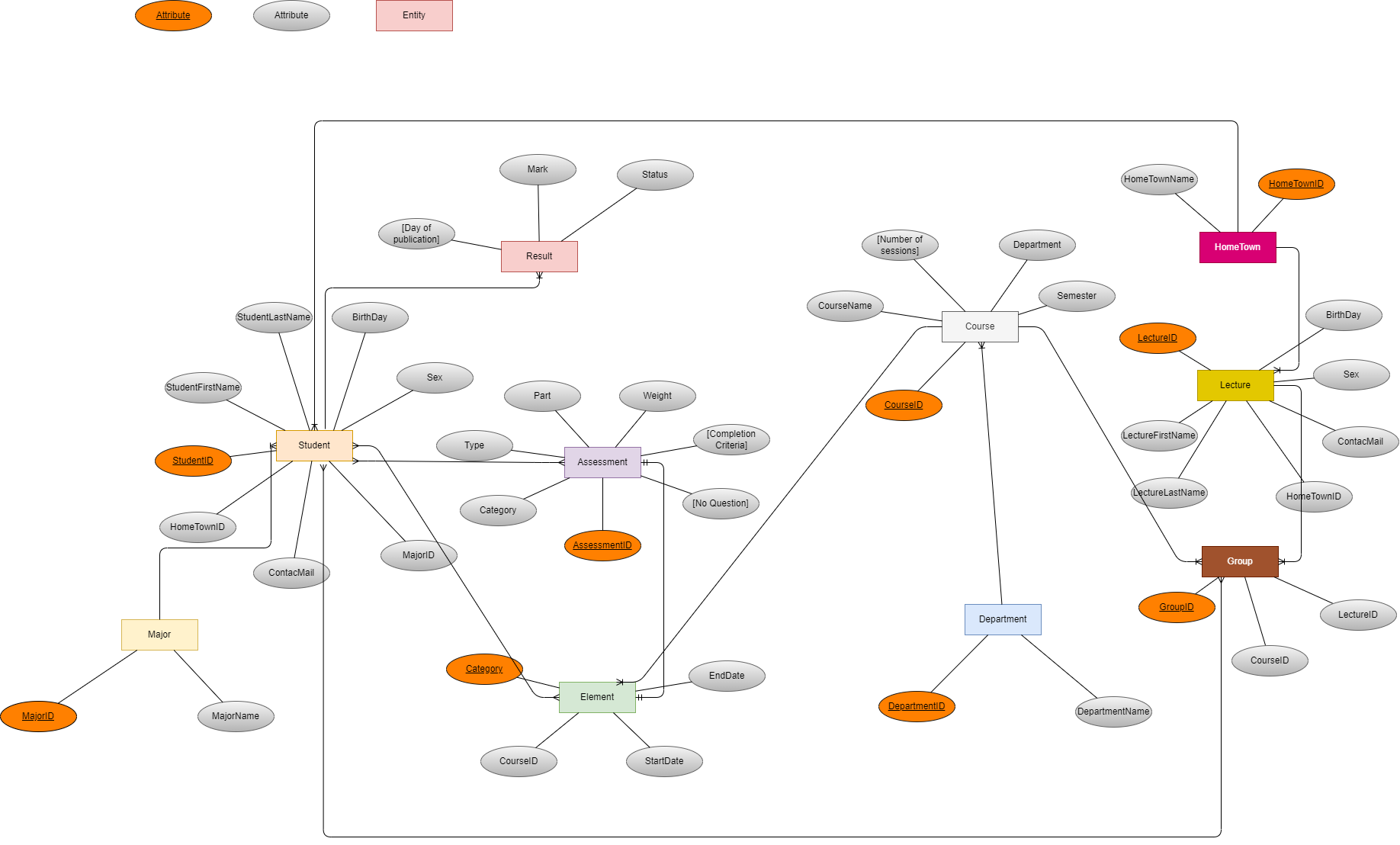
Trong đó có 3 phần chính:

1. danh sách sinh viên

2. danh sách thành phần các môn

3. điểm

II. Mô hình ERD



III. Lược đồ quan hệ

## Entity Student <-> Entity Major

Một Student chỉ theo học một Major(  
chuyên ngành) duy nhất trong hiện tại, nhưng 1 chuyên ngành có thể có nhiều Student theo học.

-> Xác Định Quan Hệ Giữa Entity Student Và Entity Major là quan hệ một nhiều ( 1-n )

## Entity Student <-> Entity Result

Một Student có thể có nhiều điểm khác nhau, nhưng một đầu điểm chỉ có thể cho đúng Student đó

* Xác Định Quan Hệ Giữa Entity Student Và Entity Result là quan hệ một nhiều ( 1-n )

## Entity Student <-> Entity HomeTown

Một Student chỉ có một HomeTown duy nhất, nhưng 1 HomeTown có thể có nhiều Student.

-> Xác Định Quan Hệ Giữa Entity Student Và Entity HomeTown là quan hệ một nhiều ( 1-n )

## Entity Group <-> Entity Lecturers

Một Group chỉ có thể được phụ trách bởi đúng 1 Lecturer và 1 Lecturer có thể phụ trách nhiều Group.

-> Xác Định Quan Hệ Giữa Entity Group Và Entity Lecturers là quan hệ một nhiều ( 1-n )

## Entity Assessment <-> Entity Element

Một Assessment chỉ có 1 category mà 1 Element chỉ tương ứng với 1 Asessment

* Xác định quan hệ 1-1

## Entity Element <-> Entity Course

## Một Course có thể có nhiều điểm thành phần (Element) , và 1 Element chỉ tương tương ứng với 1 Course

## Quan hệ 1 nhiều

## Entity Course <-> Entity Department

## Một Course thì thuộc về 1 Department, nhưng 1 Department có thể chứa nhiều Course

## Quan hệ 1 nhiều (1 – n)

## Entity Course <-> Entity Group

Một Course có thể được học bởi nhiều Group, nhưng 1 Group chỉ học 1 Course.

* Quan hệ 1 nhiều (1 – n)

## Entity Lecture <-> Entity HomeTown

Một Lecture chỉ có duy nhất 1 HomeTown, nhưng 1 HomeTown có thể áp dụng cho nhiều Lecture.

* Quan hệ 1 nhiều (1 – n)

**QUAN HỆ NHIỀU NHIỀU :**

**Entity Students <-> Entity Groups**

Một Student có thể đăng kí học nhiều Courses Và 1 Courses có thể có nhiều Student đăng kí học.

-> Xác Định Quan Hệ Giữa Entity Students Và Entity Groups là quan hệ nhiều nhiều ( n-n )

* Tạo bảng Student\_Group với khóa chính là khóa chính của bảng Student và bảng Group

Bảng này sẽ chứa thông tin Student nào đã enroll vào trong Group nào vào ngày nào.

## Entity Student <-> Entity Asessment

Một Student có thể có nhiều đầu điểm trong nhiều môn học, mà một Assessment thì có thể áp dụng cho nhiều Student

* Xác định quan hệ giữa Entity Student và Entity Assessment là quan hệ nhiều nhiều (n – n)
* Tạo bảng Reuslt với khóa chính là khóa chính của bảng Student và bảng Assessment

Bảng này chứa thông tin về điểm thành phần của học sinh

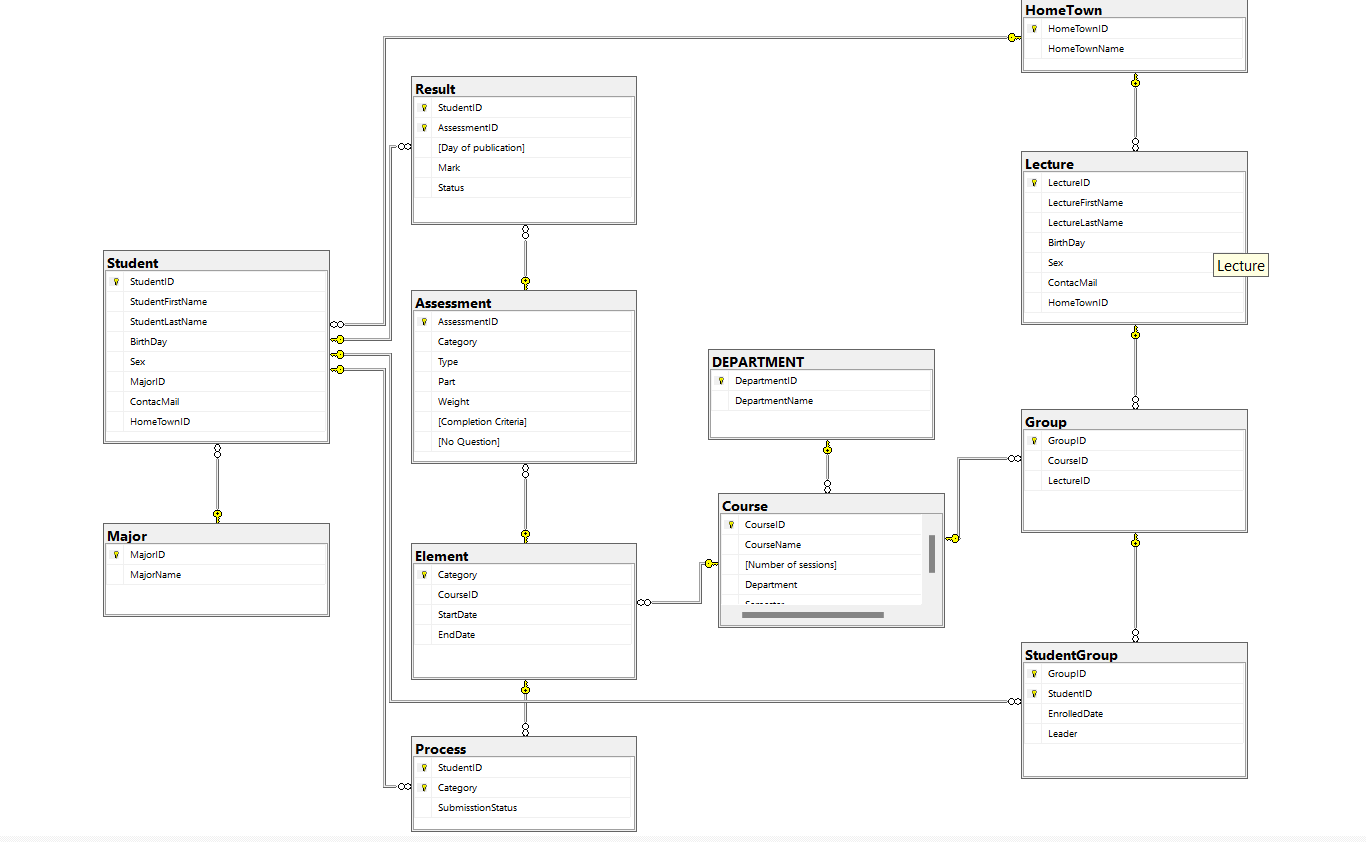
## Entity Student <-> Entity Element

Một student có thể có nhiều Element, và một Element có thể áp dụng với nhiều Student

* Quan hệ nhiều nhiều ( n – n)
* Tạo table process với khóa chính là khóa chính của Student và Element

Bảng này chứa tiến trình làm bài của sinh viên

IV. Câu lệnh tạo bảng



1. Table Assessment

USE [???]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Assessment] Script Date: 7/17/2022 12:29:44 AM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Assessment](

[AssessmentID] [int] IDENTITY(1,1) NOT NULL,

[Category] [varchar](40) NOT NULL,

[Type] [nvarchar](15) NOT NULL,

[Part] [int] NOT NULL,

[Weight] [float] NOT NULL,

[Completion Criteria] [int] NOT NULL,

[No Question] [int] NULL,

PRIMARY KEY CLUSTERED

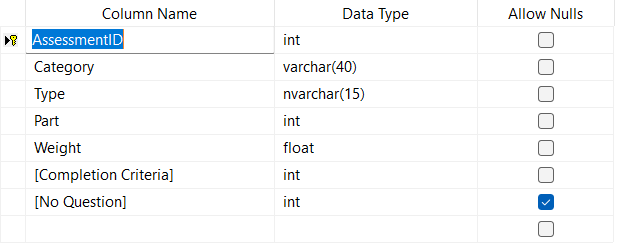
(

[AssessmentID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO



2. Table Course

/\*\*\*\*\*\* Object: Table [dbo].[Course] Script Date: 7/17/2022 12:29:44 AM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Course](

[CourseID] [varchar](15) NOT NULL,

[CourseName] [varchar](70) NOT NULL,

[Number of sessions] [int] NOT NULL,

[Department] [varchar](50) NOT NULL,

[Semester] [varchar](15) NOT NULL,

PRIMARY KEY CLUSTERED

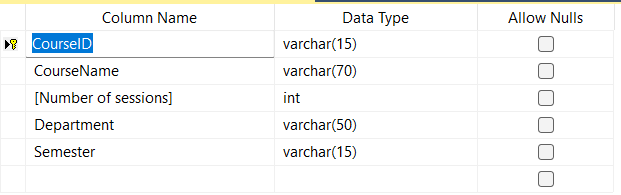
(

[CourseID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO



3. Table Department

/\*\*\*\*\*\* Object: Table [dbo].[DEPARTMENT] Script Date: 7/17/2022 12:29:44 AM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[DEPARTMENT](

[DepartmentID] [varchar](50) NOT NULL,

[DepartmentName] [varchar](70) NOT NULL,

PRIMARY KEY CLUSTERED

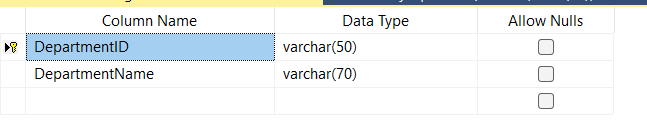
(

[DepartmentID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO



4. Table Element

/\*\*\*\*\*\* Object: Table [dbo].[Element] Script Date: 7/17/2022 12:29:44 AM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Element](

[Category] [varchar](40) NOT NULL,

[CourseID] [varchar](15) NOT NULL,

[StartDate] [date] NOT NULL,

[EndDate] [date] NOT NULL,

PRIMARY KEY CLUSTERED

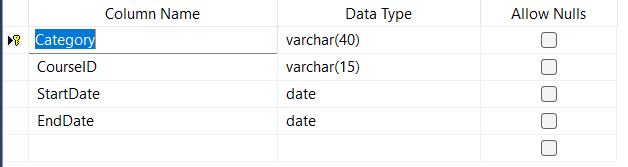
(

[Category] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO



5. Table Group

/\*\*\*\*\*\* Object: Table [dbo].[Group] Script Date: 7/17/2022 12:29:44 AM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Group](

[GroupID] [varchar](15) NOT NULL,

[CourseID] [varchar](15) NOT NULL,

[LectureID] [varchar](25) NOT NULL,

PRIMARY KEY CLUSTERED

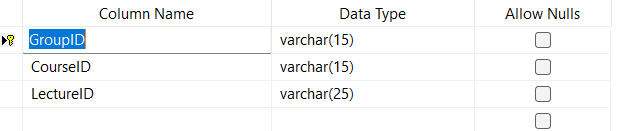
(

[GroupID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO



6. Table HomeTown

/\*\*\*\*\*\* Object: Table [dbo].[HomeTown] Script Date: 7/17/2022 12:29:44 AM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[HomeTown](

[HomeTownID] [int] IDENTITY(1,1) NOT NULL,

[HomeTownName] [nvarchar](30) NOT NULL,

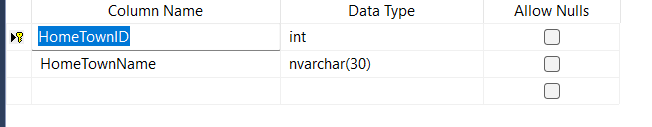
PRIMARY KEY CLUSTERED

(

[HomeTownID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

7. Table Lecture

/\*\*\*\*\*\* Object: Table [dbo].[Lecture] Script Date: 7/17/2022 12:29:44 AM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Lecture](

[LectureID] [varchar](25) NOT NULL,

[LectureFirstName] [nvarchar](30) NOT NULL,

[LectureLastName] [nvarchar](30) NOT NULL,

[BirthDay] [date] NULL,

[Sex] [bit] NOT NULL,

[ContacMail] [varchar](70) NOT NULL,

[HomeTownID] [int] NOT NULL,

PRIMARY KEY CLUSTERED

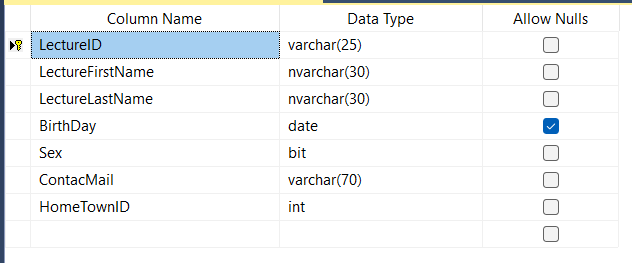
(

[LectureID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO



8. Table Major

/\*\*\*\*\*\* Object: Table [dbo].[Major] Script Date: 7/17/2022 12:29:44 AM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Major](

[MajorID] [varchar](10) NOT NULL,

[MajorName] [nvarchar](55) NOT NULL,

PRIMARY KEY CLUSTERED

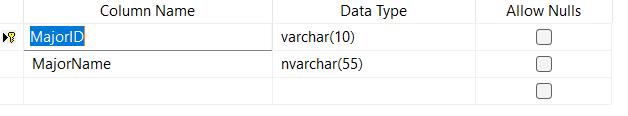
(

[MajorID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO



9. Table Process

/\*\*\*\*\*\* Object: Table [dbo].[Process] Script Date: 7/17/2022 12:29:44 AM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Process](

[StudentID] [varchar](25) NOT NULL,

[Category] [varchar](40) NOT NULL,

[SubmisstionStatus] [bit] NOT NULL,

CONSTRAINT [pk\_StudentID\_Category] PRIMARY KEY CLUSTERED

(

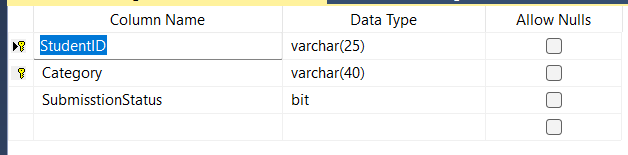
[StudentID] ASC,

[Category] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO



10. Table Result

/\*\*\*\*\*\* Object: Table [dbo].[Result] Script Date: 7/17/2022 12:29:44 AM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Result](

[StudentID] [varchar](25) NOT NULL,

[AssessmentID] [int] NOT NULL,

[Day of publication] [date] NOT NULL,

[Mark] [float] NULL,

[Status] [bit] NULL,

CONSTRAINT [pk\_OutputID\_Graded\_by\_AssessmentID] PRIMARY KEY CLUSTERED

(

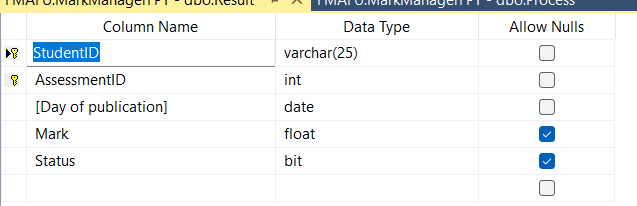
[StudentID] ASC,

[AssessmentID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO



11. Table Student

/\*\*\*\*\*\* Object: Table [dbo].[Student] Script Date: 7/17/2022 12:29:44 AM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Student](

[StudentID] [varchar](25) NOT NULL,

[StudentFirstName] [nvarchar](30) NOT NULL,

[StudentLastName] [nvarchar](30) NOT NULL,

[BirthDay] [date] NOT NULL,

[Sex] [bit] NOT NULL,

[MajorID] [varchar](10) NOT NULL,

[ContacMail] [varchar](70) NOT NULL,

[HomeTownID] [int] NOT NULL,

PRIMARY KEY CLUSTERED

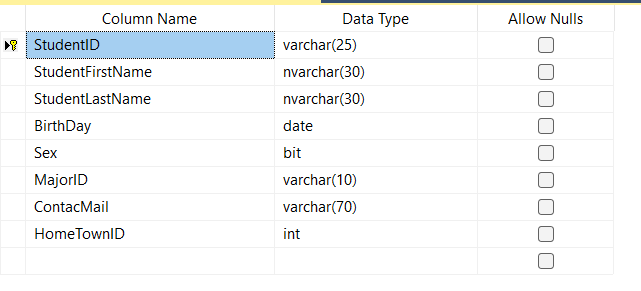
(

[StudentID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO



12. table StudentGroup

/\*\*\*\*\*\* Object: Table [dbo].[StudentGroup] Script Date: 7/17/2022 12:29:44 AM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[StudentGroup](

[GroupID] [varchar](15) NOT NULL,

[StudentID] [varchar](25) NOT NULL,

[EnrolledDate] [date] NOT NULL,

[Leader] [varchar](15) NULL,

CONSTRAINT [pk\_StudentID\_CourseID\_GroupID] PRIMARY KEY CLUSTERED

(

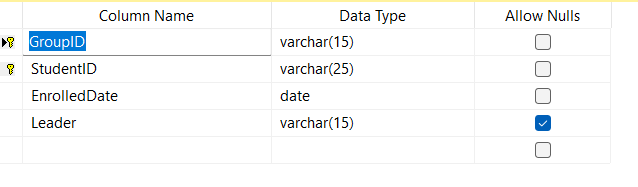
[GroupID] ASC,

[StudentID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO



Foregin key

ALTER TABLE [dbo].[Assessment] WITH CHECK ADD FOREIGN KEY([Category])

REFERENCES [dbo].[Element] ([Category])

GO

ALTER TABLE [dbo].[Course] WITH CHECK ADD FOREIGN KEY([Department])

REFERENCES [dbo].[DEPARTMENT] ([DepartmentID])

GO

ALTER TABLE [dbo].[Element] WITH CHECK ADD FOREIGN KEY([CourseID])

REFERENCES [dbo].[Course] ([CourseID])

GO

ALTER TABLE [dbo].[Group] WITH CHECK ADD FOREIGN KEY([CourseID])

REFERENCES [dbo].[Course] ([CourseID])

GO

ALTER TABLE [dbo].[Group] WITH CHECK ADD FOREIGN KEY([LectureID])

REFERENCES [dbo].[Lecture] ([LectureID])

GO

ALTER TABLE [dbo].[Lecture] WITH CHECK ADD FOREIGN KEY([HomeTownID])

REFERENCES [dbo].[HomeTown] ([HomeTownID])

GO

ALTER TABLE [dbo].[Process] WITH CHECK ADD FOREIGN KEY([Category])

REFERENCES [dbo].[Element] ([Category])

GO

ALTER TABLE [dbo].[Process] WITH CHECK ADD FOREIGN KEY([StudentID])

REFERENCES [dbo].[Student] ([StudentID])

GO

ALTER TABLE [dbo].[Result] WITH CHECK ADD FOREIGN KEY([AssessmentID])

REFERENCES [dbo].[Assessment] ([AssessmentID])

GO

ALTER TABLE [dbo].[Result] WITH CHECK ADD FOREIGN KEY([StudentID])

REFERENCES [dbo].[Student] ([StudentID])

GO

ALTER TABLE [dbo].[Student] WITH CHECK ADD FOREIGN KEY([HomeTownID])

REFERENCES [dbo].[HomeTown] ([HomeTownID])

GO

ALTER TABLE [dbo].[Student] WITH CHECK ADD FOREIGN KEY([MajorID])

REFERENCES [dbo].[Major] ([MajorID])

GO

ALTER TABLE [dbo].[StudentGroup] WITH CHECK ADD FOREIGN KEY([GroupID])

REFERENCES [dbo].[Group] ([GroupID])

GO

ALTER TABLE [dbo].[StudentGroup] WITH CHECK ADD FOREIGN KEY([StudentID])

REFERENCES [dbo].[Student] ([StudentID])

GO

ALTER TABLE [dbo].[Assessment] WITH CHECK ADD CHECK (([Weight]<=(1) AND [Weight]>(0)))

GO

V. Truy vấn chứng minh tính hữu ích của cở dữ liệu

1. Query dùng ORDER BY:

select \* from Student Order by StudentFirstName, StudentLastName

Query này đưa ra danh sách sinh viên được sắp xếp theo chiều tang dần của bảng chữ cái. Vì thế mà khi tìm kiếm bằng mắt thường, chúng ta có thể nhanh chóng tìm đến khu vực có chứa tên của sinh viên cần tìm.

2. Query dùng Inner Join

select st.StudentID,st.StudentFirstName,st.StudentLastName,g.CourseID,c.Department,c.Semester, g.GroupID

from

Student st inner join StudentGroup sg on st.StudentID = sg.StudentID

inner join [Group] g on sg.GroupID = g.GroupID

inner join Course c on g.CourseID = c.CourseID

Query này cho ra một bảng gồm 6 cột StudentID, StudentFirstName, StudentLastName, CourseID, Department, Semester và Group

Mỗi kì học mới, sinh viên cần được sắp xếp vào các Group tương ứng với các Course cần học trong Semester đó. Bảng này sẽ đưa ra danh sách Student nào cần học Course nào và được sắp xếp vào Group tương ứng để dễ dàng khi gửi thông báo nhận lớp.

3. Query dùng Aggregate functions

select r.StudentID,sum(r.Mark \* a.Weight) as Average, e.CourseID

from

Result r inner join Assessment a

on r.AssessmentID = a.AssessmentID

inner join Element e

on a.Category = e.Category

group by r.StudentID, e.CourseID

Query này cho ra một bảng gồm 3 cột : StudentID, Average và CourseID.

Bảng kết quả chứa thông tin về tổng điểm hiện đạt được (Không tính điểm chưa có) của những Course đã và đang học của từng sinh viên.

Giúp sinh viên dễ dàng tra cứu điểm của bản thân và biết cần bao nhiêu điểm để qua môn.

4. Query dùng Group by và Having

select r.StudentID,sum(r.Mark \* a.Weight) as Average, e.CourseID

from

Result r inner join Assessment a on r.AssessmentID = a.AssessmentID

inner join Element e on

a.Category = e.Category

group by r.StudentID, e.CourseID

having r.StudentID = 'HE163693'

Query này cho ra một bảng gồm 3 cột : StudentID, Average và CourseID.

Bảng kết quả chứa thông tin về tổng điểm hiện đạt được (Không tính điểm chưa có) của những Course đã và đang học của sinh viên có mã ở mục having.

Sinh viên chỉ cần nhập mã của mình vào là sẽ thấy điểm hiện tại của bản thân

5. Query dùng sub-query như một quan hệ

select LEC.LectureID,

concat(

LEC.LectureFirstName,' ',LEC.LectureLastName) as FullName,STU.StudentID

,ASSES.Category ,r.Mark

from

(Select LectureID, LectureFirstName,LectureLastName from Lecture) as LEC

inner join [Group] g

on g.LectureID = LEC.LectureID

inner join StudentGroup sg

on sg.GroupID = g.GroupID

inner join (select StudentID, StudentFirstName, StudentLastName from Student) as STU

on STU.StudentID = sg.StudentID

inner join Result r

on STU.StudentID = r.StudentID

inner join (select AssessmentID,Category from Assessment) ASSES

on ASSES.AssessmentID = r.AssessmentID

order by LEC.LectureID

Query này cho ra một bảng gồm 5 cột: LectureID FullName, StudentID, Category và Mark

Bảng chứa thông tin về điểm thành phần từng môn của sinh viên được đánh bởi giáo viên dạy môn đó

Do vậy mà bảng này giúp ta giám sát được tiến trình kiểm tra và đánh giá điểm của giáo viên.

6. Query dùng sub-query ở mệnh đề where

select LEC.LectureID, concat(LEC.LectureFirstName,' ',LEC.LectureLastName) as FullName,STU.StudentID

,ASSES.Category ,r.Mark

from

(Select LectureID, LectureFirstName,LectureLastName from Lecture) as LEC

inner join [Group] g

on g.LectureID = LEC.LectureID

inner join StudentGroup sg

on sg.GroupID = g.GroupID

inner join (select StudentID, StudentFirstName, StudentLastName from Student) as STU

on STU.StudentID = sg.StudentID

inner join Result r

on STU.StudentID = r.StudentID

inner join (select AssessmentID,Category from Assessment) ASSES

on ASSES.AssessmentID = r.AssessmentID

Where LEC.LectureID = (select LectureID from Lecture where LectureFirstName = 'Nguyen Tan' and LectureLastName = 'An')

Query này cho ra một bảng gồm 5 cột: LectureID FullName, StudentID, Category và Mark

Bảng chứa thông tin về điểm thành phần từng môn của sinh viên được đánh bởi giáo viên mà ta nhập tên vào.

Do vậy mà bảng này giúp ta giám sát được tiến trình kiểm tra và đánh giá điểm của giáo viên đó

7. Truy vấn sử dụng đối sánh từng phần trong mệnh đề WHERE

select \* from Student

where MajorID like 'S%'

Query này cho ra một bảng gồm 8 cột: StudentID, StudentFirstName, StudentLastName, BirthDay, Sex, MajorID, ContacMail, HomeTownID

Bảng chứa thông tin về Student học ngành Soft Engineer

**8. Query sử dụng self-join:**

select \* from StudentGroup

except

select b.GroupID,b.StudentID,b.EnrolledDate,b.Leader

from StudentGroup a left join StudentGroup b on b.Leader = a.StudentID

Query này sẽ cho ra 4 cột: GroupID, StudentID, EnrolledDate, Leader

Bảng chứa thông tin của leader tương ứng với mỗi Group.

Sử dụng bảng này, giáo viên có thể biết được leader của mỗi group mà mình dạy mà giao việc cho group thông qua leader.

Và Student có thể biết leader của group mình là ai để liên hệ

**9. Query sử dụng stored proceduce**

GO

CREATE PROC Count\_Std\_in\_Group

@Group VARCHAR(55)

AS

BEGIN

SELECT COUNT(GroupID) AS [TOTAL OF STUDENT ENROLLED IN THIS GROUP] FROM StudentGroup WHERE GroupID = @Group

END

GO

EXEC Count\_Std\_in\_Group N'IA1604\_CSD201'

EXEC Count\_Std\_in\_Group N'IA1604\_DBI202'

EXEC Count\_Std\_in\_Group N'IA1604\_IAO202'

Đếm số student enroll vào bất kì Group nào cần tìm