Student Grading Management Subsystem

Giới thiệu

1. Thông tin cá nhân:

- **Họ tên**: Nguyễn Doanh Thịnh.
- **Mã số sinh viên:** HE161890.
- **Lớp**: IA1604.
- **Khoa**: Công nghệ thông tin.
- **Chuyên ngành**: An toàn thông tin.
- **Trường**: Đại học FPT cơ sở Hòa Lạc.
- **Email**: thinhndhe161890@fpt.edu.vn.
- **Số điện thoại liên lạc**: 0976622548.

2. Về bản báo cáo:

- **Tên**: Database lưu trữ dữ liệu hệ thống tính điểm cho sinh viên trường đại học FPT.
- **Nguyên do**: Bản báo cáo này được tạo ra theo yêu cầu của giảng viên thực hiện giảng dạy môn DBI202 của lớp IA1604 như một bài Progress Test có lấy điểm.
- **Tóm tắt nội dung**: đưa ra các phân tích về đề bài, hình ảnh, các thông tin có thể khai thác từ các hình ảnh, tạo dựng một database cơ bản cho 1 hệ thống tính điểm và test độ ứng dụng khi đưa vào thực tiễn.

Mục lục

11. Grade.
12. View.
13. Assess.

V. Module Diagram.

VI. Phân tích MD và dữ liệu sẽ được add vào database.

1. Students.
2. Groups.
3. Join.
4. Lecturers.
5. Courses.
6. Assignment.
7. Classes.
8. **Enroll.**
9. Categories.
10. Assessment System.
11. Grade.
12. View.
13. Assess.

VII. Tạo Database và insert dữ liệu.

1. Tạo database và các tables.
2. Insert dữ liệu.

IX. Index, View, Trigger và Stored Procedure.

1. Index.
2. View.
3. Trigger.
4. Stored Procedure.

X. Queries:

1. Query dùng ORDER BY.
2. Query dùng INNER JOIN.
3. Query dùng Aggregate function.
4. Query dùng GROUP BY và HAVING.
5. Query sử dụng sub-query như 1 quan hệ.
6. Query sử dụng sub-query ở mệnh đề WHERE:
7. Query sử dụng partial matching trong mệnh đề WHERE:
8. Query sử dụng self-join:
9. Query ngẫu nhiên 1:
10. Query ngẫu nhiên 2:

I. *Phân tích đề bài, các hình ảnh:*

1. Ảnh 1: Thống kê các bài test của 1 môn học.

Category	Type	Part	Weight	Completion Criteria	Duration	LO	Question Type	No Question	Knowledge and Skill	Grading Guide	Note
Progress Tests	quiz	2	10.0%	>0	20'		Multiple choices Marked by Computer or a suitable format	20	up to 04 covered chapters	by instructor using computer	Instruction and shedules for Progress tests must be presented in the Course Implementation Plan approved by director of the campus. Progress test must be taken right after the last lectures of required material. Instructor has resposibility to review the test for students after graded.
Assignment	on-going	1	20.0%	>0	at home		Design; Implementation; Presentation		Simple RDBS design and implementation using a DBMS	guided by instructor, prepare at home present in class	40% Design, 20% Implementation, 40% Presentation of the whole Project
Labs	on-going	5	15.0%	>0	in lab session		practical exercises		related to studied modules	Guided by instructor	may be continued at home.
Practical Exam	practical exam	1	25.0%	>0	85'		Preferable to be marked by Scripts		DB programing skills	by exam board and department	Practical Exam database is up load in CMS in advanced.
Final Exam	final exam	1	30.0%	5	60'		Multiple choices Marked by Computer	60	Knowledge and skills in the course, but with much focus on the items in Chapters 2 to 6, >= 70% new questions (for the current semester);	by exam board	

- Nhìn sơ qua, có thể thấy được ở ảnh này, có 1 bảng với nội dung là các loại bài test, hoặc đầu điểm mà một sinh viên phải đáp ứng của 1 môn học.
- Bảng bao gồm các cột :
  - Category: Loại bài test / đầu điểm.
  - Type: Kiểu bài test.
  - Part: Bao gồm bao nhiêu phần.
  - Weight: Trọng số của số điểm bài test đó ảnh hưởng lên điểm trung bình môn học.
  - Completion Criteria: Điều kiện pass môn.
  - Duration: Thời lượng của bài test.
  - Question Type: Loại câu hỏi xuất hiện.
  - No Question: Số lượng câu hỏi 1 bài.
  - Knowledge and Skill: Kiến thức và kĩ năng cần thiết để có thể làm được bài test.
  - Grading Guide: Hướng dẫn cho điểm.
  - Note: Chú thích của giáo viên.
- Nhận thấy rằng mỗi 1 môn đều có những loại bài test khác nhau (không phải 1 cơ cấu đánh giá điểm như trong hình được áp dụng cho tất cả các môn).
- Do vậy cần 1 entity là category lưu lại tất cả các loại bài test và các thông số cơ bản mà luôn áp dụng cho bài test đó.

2. Ảnh 2: Status của 1 sinh viên.

NO.	SUBJECT CODE	SUBJECT NAME	SEMESTER	GROUP	STARTDATE	ENDDATE	AVERAGE MARK	STATUS
1	SSL101c	Academic Skills for University Success	Spring2021					Not Passed
2	SSG103	Communication and In-Group Working Skills	Summer2021					Passed
3	NWC203c	Computer Networking	Summer2021					Passed
4	CEA201	Computer Organization and Architecture	Spring2021					Passed
5	MAD101	Discrete mathematics	Summer2021					Passed
6	JPD113	Elementary Japanese 1-A1.1	Fall2021					Passed
7	CSI104	Introduction to Computer Science	Spring2021					Passed
8	DBI202	Introduction to Databases	Fall2021					Not Passed

- Ở bảng này, sinh viên sẽ có thể xem được thông tin cơ bản của toàn bộ các môn học mà mình đã tham gia.
- Bảng bao gồm các cột:
  - No: Số hiệu / Số thứ tự.
  - Subject Code: Mã số của môn học.
  - Subject Name: Tên của môn học.
  - Semester: Kì học mà sinh viên đã tham gia môn học tương ứng.
  - Group: Nhóm sinh viên mà sinh viên này đã tham gia học cùng môn học tương ứng.
  - Start Date: Ngày bắt đầu của môn học tương ứng.
  - End Date: Ngày kết thúc của môn học tương ứng.
  - Average Mark: Điểm trung bình của sinh viên ở môn học tương ứng.
  - Status: Trạng thái của sinh viên ở môn học tương ứng (passed or not passed).
- Thấy rằng bảng này chứa rất nhiều thông tin và có nhiều attributes có thể làm thành khóa. Điều này không tốt cho 1 table.
- Do vậy một mình bảng này không thể chỉ được thể hiện trong 1 entity được, phải thông qua nhiều các entities nhỏ lẻ để tránh việc vi phạm vào 3rd-Normal Form Standard.

3. Ảnh 3: Bảng điểm của 1 môn học.

GRADE CATEGORY	GRADE ITEM	WEIGHT	VALUE	COMMENT
Quiz 2	Quiz 2	7.0 %	7.8	
	Total	7.0 %	7.8	
Quiz 1	Quiz 1	8.0 %	7.6	
	Total	8.0 %	7.6	
Activity	Activity	10.0 %	8.5	
	Total	10.0 %	8.5	
Group Assignment	Group Assignment	15.0 %	9	
	Total	15.0 %	9	
Group Project	Group Project	30.0 %	8.3	
	Total	30.0 %	8.3	
Final Exam	Final Exam	30.0 %	8.6	
	Total	30.0 %	8.6	
Final Exam Resit	Final Exam Resit	30.0 %		
	Total	30.0 %		
COURSE TOTAL	**AVERAGE**	**8.4**		
	STATUS	**PASSED**		

- Ở bảng này, thấy được các đầu điểm , điểm trung bình và trạng thái của 1 môn học mà 1 sinh viên sau khi học môn học đó đã đạt được.
- Bảng gồm 5 cột:
  - Grade Category: Các loại đầu điểm mà sinh viên phải có trong môn học này.
  - Grade Item: Từng phần của mỗi loại đầu điểm.
  - Weight: Trọng số của từng đầu điểm đối với môn học này.
  - Value: Điểm số sinh viên đạt được ứng với mỗi bài test.
  - Comment: Một số lời nhắc hoặc chú ý thì sẽ được note vào cột này.

- Phía cuối sau khi kết thúc bảng điểm, xuất ra điểm trung bình môn và trạng thái. 2 dòng này có thể dũng là 1 attribute của entity.
- Dự đoán: Để thu được bảng này, ta cần join khá nhiều bảng khác lại với nhau, vì riêng nó khó có thể làm 1 entity mà không vi phạm 3rd-Normal Form Standard.

II. ***Dự đoán các entities cần thiết:***

1. **Nhận thấy rằng các bảng đã được phân tích bên trên đều có liên quan tới một số các entities xác định:**

    - `Students`: Tất cả các bảng trên đều liên quan tới thực thể `Students`(bảng điểm là bảng điểm của 1 sinh viên, status cũng là của 1 sinh viên, phân loại đầu điểm cũng xem từ môn học mà sinh viên tham gia), mà 1 hệ thống tính điểm thì không chỉ có của riêng 1 sinh viên nào đó. Do vậy, 1 entity `Students` là chắc chắn phải có trong hệ thống này.
    - `Courses`: Một hệ thống tinh điểm thì không thể thiếu các `Courses`. Sinh viên sẽ tham gia các môn học này, và đầu điểm đánh giá sẽ được dựa theo môn học tương ứng.
    - `Lecturers`: Có các môn học thì chúng ta cũng sẽ có các `Lecturers` tham gia giảng dạy các môn học đó. Giảng viên sẽ trực tiếp cho điểm đánh giá sinh viên theo môn học tương ứng. Note: `Lecturers` sẽ có 1 attribute là `Report` phục vụ cho yêu cầu viết query phía sau của đề bài.
    - `Classes`: Sinh viên sẽ tham gia các lớp học, giảng viên sẽ giảng dạy các môn học cũng theo các lớp học tương ứng. Do vậy, chúng ta cũng có 1 entitiy là `Classes`.
    - `Groups`: Không thể để tất cả các sinh viên học cùng 1 lớp được. Họ sẽ được phân thành các nhóm và học các lớp theo ngày, giờ khác nhau. Các môn học ứng với từng nhóm cũng có thể được dạy bởi các giảng viên khác nhau. Do vậy, chúng ta cũng sẽ có 1 entity là `Groups`.

- `Categories`: Một môn học được đánh giá qua nhiều bài test và nhiều đầu điểm khác nhau. Các giảng viên sẽ cho điểm đánh giá sinh viên học môn học tương ứng dựa vào các loại đầu điểm này. Do vậy, cần 1 entity là `Categories`

2. **Các entities đã dự đoán bên trên khả năng cao sẽ có một số relationships sau:**

- Đối với `Students` và `Groups`, các sinh viên sẽ tham gia vào các nhóm khác nhau khi dựa trên kì học, môn học và chuyên ngành, các nhóm cũng sẽ chứa nhiều sinh viên cùng các đặc điêm. Do vậy, ta đặt 2 entities này vào 1 quan hệ N—N.
- Đối với `Groups` và `Classes`, các nhóm sẽ được phân vào các lớp học với giảng viên và môn học tương ứng với kì học hiện tại của nhóm sinh viên đó. Một nhóm có thể tham gia nhiều lớp học, và 1 lớp cũng có thể có nhiều nhóm tham gia. Do vậy, ta sẽ đặt 2 entities này vào 1 quan hệ N—N.
- Đối với `Courses` và `Classes`, một lớp sẽ được phân 1 môn học và một môn học thì lại được phân vào nhiều lớp khác nhau. Do vậy, 2 entities này sẽ được đặt vào quan hệ `Courses` 1—N `Classes`.
- Đối với `Courses` và `Categories`, một môn học thì được đánh giá qua nhiều loại đầu điểm, và một loại đầu điểm thì cũng được dùng để đánh giá nhiều môn học. Do vậy, 2 entities này sẽ được đặt vào quan hệ `Courses` N—N `Categories`.
- Đối với `Lecturers` và `Classes`, tương tự như với `Courses`, một lớp thì cũng sẽ chỉ có 1 giảng viên tham gia giảng dạy chính (không tính dạy thay vì khi xét về cuối kì thì không xét đến giảng viên dạy thay) và một giảng viên cũng được phân vào nhiều lớp để giảng dạy. Do vậy, 2 entities này cũng sẽ được đặt vào quan hệ `Lecturers` 1—N `Classes`.
- Đối với `Lecturers` và `Courses`, một giảng viên có thể dạy được nhiều môn học và một môn học cũng có thể được dạy bởi nhiều giảng viên. Phụ thuộc vào kì học, 1 giảng viên có thể được phân công dạy 1 môn ở kì này, nhưng có thể không phải dạy môn đó ở kì khác. 2 entities này cũng sẽ được đặt vào 1 quan hệ N—N.
- Đối với `Lecturers` và `Students`, giảng viên sẽ giảng dạy các môn học theo các lớp, các sinh viên sẽ tham gia vào các lớp này ứng với các nhóm được phân

vào. 1 kì học, sinh viên học 5 môn có thể có tối đa 5 giảng viên giảng dạy, 1 giảng viên dạy nhiều lớp cũng gồm nhiều sinh viên theo học

- Đối với `Students` và `Courses`, học sinh khi tham gia vào các lớp theo nhóm thì cũng có quan hệ với môn học thông qua các lớp đó. Trong 1 kì, sinh viên được trường phân cho 5 lớp ứng với 5 môn, nhưng sinh viên cũng có thể đăng kí thêm hoặc đăng kí tạm hoãn. Một môn học cũng sẽ được dạy cho rất nhiều sinh viên.
- Đối với `Groups` và `Courses`, các nhóm được phân vào các lớp, và các lớp lại được giao cho các môn học riêng. Do vậy các nhóm cũng có quan hệ với môn học thông qua các lớp. trong 1 kì, các nhóm sẽ được phân cho học nhiều môn, và 1 môn cũng được đem đi dạy cho nhiều nhóm sinh viên.
- Đối với `Students` và `Classes`, sinh viên sẽ được phân vào các nhóm và các nhóm sẽ được phân vào học ở các lớp khác nhau ở mỗi kì, sinh viên sẽ có quan hệ với lớp thông qua các nhóm. Trong 1 kì, sinh viên có thể tham gia nhiều lớp, và 1 lớp cũng được phân cho nhiều nhóm có nhiều sinh viên.
- Đối với `Lecturers` và `Groups`, các giảng viên được giao cho vào các lớp học, các nhóm sinh viên học lại được phân vào học ở các lớp. Do vậy, giảng viên và các nhóm sinh viên cũng có quan hệ với các giảng viên thông qua các lớp. Trong 1 kì, 1 nhóm sinh viên được nhiều giảng viên dạy, 1 giảng viên cũng giảng dạy cho nhiều nhóm sinh viên.
- Đối với `Lecturers` và `Lecturers`, các bộ môn sẽ có những giảng viên làm trưởng môn môn của môn học đó, những giảng viên dạy môn này đều phải báo cáo tính hình giảng dạy, điểm trác của sinh viên cho trưởng môn. Do vậy, ca có quan hệ `Lecturers` 1—1 `Lecturers`

3. **Từ những dữ liệu trên, các attributes ứng với từng entities được dự đoán như sau:**

- `Students`: Các attributes của `Students` chính là các thông tin cơ bản của 1 sinh viên:
  - Mã số sinh viên được cấp bởi nhà trường cho 1 sinh viên.

    - Địa chỉ email mà nhà trường cấp cho sinh viên.
    - Tên gọi của sinh viên.
    - Họ và tên đệm của sinh viên.
    - Giới tính của sinh viên.
    - Ngày tháng năm sinh của sinh viên.
    - Những kì học mà sinh viên tham gia.
    - Những nhóm mà sinh viên tham gia ở mỗi kì.
    - Những môn mà sinh viên học mỗi kì.
    - Loại đầu điểm ở mỗi môn học.
    - Giảng viên thực hiện giảng dạy ở môn học tương ứng.
    - Điểm của sinh viên tại mỗi đầu điểm.

- `Groups`: Các attributes của `Groups` là các thông tin cơ bản của 1 nhóm:
    - Mã nhận diện của 1 nhóm.
    - Sinh viên đã tham gia nhóm đó.
    - Chuyên ngành chính của nhóm sinh viên đó.
    - Số sinh viên của 1 nhóm.
    - Kì học mà nhóm đó tham gia học tập.
    - Lớp học theo kì.
    - Môn học mà nhóm đó tham gia.
    - Điểm trung bình của môn học tương ứng.
    - Giảng viên đã giảng dạy môn học tương ứng

- `Classes`: Các attributes của `Classes` là các thông tin cơ bản của 1 lớp hoc:
    - Mã nhận diện 1 lớp học.
    - Các nhóm học sinh đã tham gia học lớp đó.

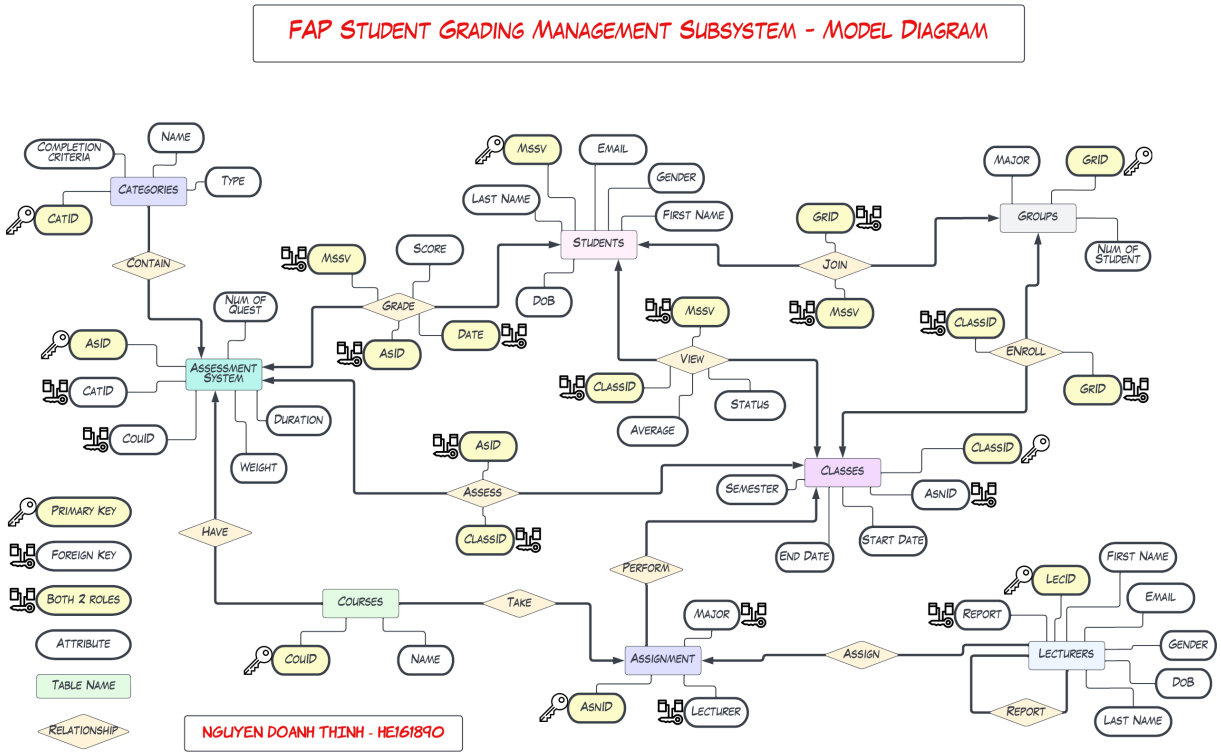
    - Môn học được giảng dạy ở lớp học đó.
    - Giảng viên tham gia giảng dạy ở lớp học đó.
    - Điểm trung bình của cả nhóm tham gia học môn học tương ứng.

- `Courses`: Các attributes của `Courses` là các thông tin cơ bản của 1 môn học:
    - Mã nhận diện 1 môn học.
    - Các loại bài test, các đầu điểm mà một môn học cần phải có.
    - Các giáo viên sẽ tham gia dạy môn học này.
    - Các lớp học mà được phân cho giảng dạy môn học này.
    - Kì học mà lớp học này được phân vào.

- `Lecturers`: Các attributes của `Lecturers` là các thông tin cơ bản của 1 giảng viên:
    - Mã số giảng viên được nhà trường cấp.
    - Địa chỉ email mà nhà trường cấp cho giảng viên.
    - Tên gọi của giảng viên.
    - Họ và tên đệm của giảng viên.
    - Giới tính của giảng viên.
    - Ngày tháng năm sinh của giảng viên.
    - Các lớp mà giảng viên tham gia giảng dạy.
    - Các môn học mà giảng viên tham gia giảng dạy.
    - Trưởng bộ môn mà giảng viên giảng dạy, giảng viên sẽ phải báo các tính hình giảng dạy cho trưởng bộ môn.
    - Kì học mà giảng viên phải tham gia giảng dạy.

- `Categories`: Các attributes của `Categories` là các thông tin cơ bản của 1 loại đầu điểm.
  - Mã nhận diện.
  - Tên của đầu điểm, hoặc bài test mà sinh viên cần thực hiện.
  - Loại bài test (on-going, at home, …).
  - Loại đầu điểm tương ứng cần mấy phần.
  - Trọng số mà đầu điểm này ảnh hưởng đến điểm trung bì 1 môn.
  - Điều kiện pass môn.
  - Thời lượng của bài test.
  - Loại câu hỏi xuất hiện.
  - Số lượng câu hỏi 1 bài.
  - Kiến thức và kĩ năng cần thiết để có thể làm được bài test.
  - Hướng dẫn cho điểm cho giảng viên.
  - Chú thích của giảng viên (nếu có).

III. Entity Relationship Diagram:



IV. ***Chuẩn hóa 3nd Normal Form đưa ra các tables:***

1. **Students**:

- Nhận thấy 1 học sinh có thể tham gia vào nhiều nhóm khác nhau, nhiều môn học khác nhau, nhiều kì khác nhau, nhiều lớp khác nhau và được nhiều giáo viên giảng dạy

  → Khi đưa vào bảng, sẽ hình thành các multiple values attributes.

  → Còn chưa đạt chuẩn 1st.

  → Phải tách hết những attributes này ra, đưa vào entities hoặc relationships khác.

- Sau khi tách bỏ hết, ta được bảng `Students` với các cột:
  - `Mssv`: Mã số sinh viên được cấp bởi nhà trường cho 1 sinh viên. Đây sẽ chính là Primary Key của bảng này.
  - Email: Địa chỉ email mà nhà trường cấp cho sinh viên.
  - First Name: Tên gọi của sinh viên.
  - Last Name: Họ và tên đệm của sinh viên.
  - Gender: Giới tính của sinh viên.
  - DoB: Ngày tháng năm sinh của sinh viên.
- Sau khi tách hết các multiple values attributes, bảng `Students` giờ chỉ còn 1 primary key là `Mssv`, không có non-key attributes nào phụ thuộc 1 phần vào key, không có non-key attributes nào phụ thuộc bắc cầu vào key.

  → Bảng `Students` đã đạt chuẩn 3rd Normal Form.

2. **Groups**:

- Nhận thấy 1 nhóm thì có nhiều sinh viên tham gia vào, 1 nhóm cũng tham gia nhiều kì học. Ở 1 kì học thì 1 nhóm cũng tham gia vào nhiều lớp học (5 lớp nếu theo sắp xếp của nhà trường), và nhiều lớp thì lại có nhiều môn và nhiều giảng viên.

  → Khi đưa vào bảng thì các attributes này sẽ có phụ thuộc bắc cầu với nhau.

  → Chưa đạt chuẩn 2nd.

  → Các attributes này cũng là các multiple values attributes.

  → Chea đạt cả chuẩn 1st.

  → Phải tách hết những attributes này ra, đưa vào entities hoặc relationships khác.
- Sau khi tách bỏ hết, bảng `Groups` sẽ có các cột sau:
  - `GrID`: Mã nhận diện của 1 nhóm. Mã này cũng là primary key của bảng.

    - Major: Chuyên ngành chính của gần như toàn bộ sinh viên được xếp vào 1 nhóm.
    - Number of Students: Số sinh viên đã tham gia nhóm này.
- Sau khi tách hết các multiple values attributes, bảng `Groups` giờ chỉ còn 1 primary key là `GrID`, không có non-key attributes nào phụ thuộc 1 phần vào key, không có non-key attributes nào phụ thuộc bắc cầu vào key.

  → Bảng `Groups` đã đạt chuẩn 3rd Normal Form.

3. **Join**:

- Đây là 1 bảng mới được tạo ra như là hệ quả của việc tách multiple values attributes từ 2 bảng trên.
- Một số attributes đã tách ra từ 2 bảng trên sẽ được thêm vào bảng này để mô tả quan hệ `Students` N—N `Groups`.
- Gồm các cột:
    - Mssv: Mã số sinh viên.
    - GrID: Mã nhóm mà sinh viên tương ứng đã tham gia.
- 2 cột này cũng tạo thành primary key của bảng `Join`.
- Do bảng chỉ có 2 cột và 2 cột này là primary key nên bảng này cũng đạt chuẩn 3rd Normal From.

4. **Lecturers**:

- Nhận thấy một giảng viên có thể dạy nhiều môn học, một môn học lại được xếp vào nhiều lớp, một lớp lại được phân vào nhiều kì.

  → Khi đưa vào bảng thì các attributes này sẽ có phụ thuộc bắc cầu với nhau.

  → Chưa đạt chuẩn 2nd.

  → Các attributes này cũng là các multiple values attributes.

→ Chea đạt cả chuẩn 1st.

→ Phải tách hết những attributes này ra, đưa vào entities hoặc relationships khác.

- Sau khi tách bỏ hết, bảng `Lecturers` sẽ có các cột sau:
  - `LecID`: Mã số mà nhà trường cấp cho mỗi giảng viên. Mã số này là độc nhất của mỗi giảng viên, do vậy, nó cũng chính là primary key của bảng.
  - Email: Địa chỉ email mà nhà trường cấp cho giảng viên.
  - First Name: Tên gọi của giảng viên.
  - Last Name: Họ và tên đệm của giảng viên.
  - Gender: Giới tính của giảng viên.
  - DoB: Ngày tháng năm sinh của giảng viên.
  - Report: Mã số nhận diện của giảng viên trưởng bộ môn của giảng viên tương ứng.
- Sau khi tách hết các multiple values attributes, bảng `Lecturers` giờ chỉ còn 1 primary key là `LecID`, không có non-key attributes nào phụ thuộc 1 phần vào key, không có non-key attributes nào phụ thuộc bắc cầu vào key.

  → Bảng `Lecturers` đã đạt chuẩn 3rd Normal Form.

5. **Courses**:

- Nhận thấy 1 môn học thì cần rất nhiều đầu điểm để đánh giá. 1 môn thì cũng được giảng dạy bởi nhiều giảng viên và được phân vào nhiều lớp.

  → Khi đưa vào bảng, sẽ hình thành các multiple values attributes.

  → Còn chưa đạt chuẩn 1st.

  → Phải tách hết những attributes này ra, đưa vào entities hoặc relationships khác.
- Sau khi tách bỏ hết, bảng `Courses` sẽ có các cột sau:
  - `CouID`: Mã của môn học, mã này cũng là unique đối với mỗi môn.

    - Name: Tên của môn học tương ứng.
- Bỏ hết attributes liên quan đến đầu điểm hay giảng viên và class vì chúng ta sẽ thiết kế bảng riêng cho chúng. Như vậy, thì sau khi tách hết các multiple values attributes, bảng `Courses` giờ chỉ còn 1 primary key là `CouID`, không có non-key attributes nào phụ thuộc 1 phần vào key, không có non-key attributes nào phụ thuộc bắc cầu vào key.

  → Bảng `Courses` đã đạt chuẩn 3rd Normal Form.

6. **Assignment**:

- Bảng này lên kết `Courses` với `Lecturers` lại với nhau theo đúng quan hệ N—N
- 1 môn học thì do nhiều giảng viên dạy và 1 giảng viên cũng có thể được phân công dạy nhiều môn. Do vậy ta tạo 1 bảng phân công để thể hiện rõ những giảng viên nào được phân công dạy môn nào.
- Ở mỗi kì học, có thể giảng viên này được phân công môn này nhưng ở kì khác thì lại được phân công môn khác. Để linh hoạt trong việc tái sử dụng những phân công thì không để `Assignment` thành 1 bảng relationship được mà chuyển thành 1 entity.
- Các attributes của entity phân công này gồm có
    - Mã nhận diện 1 phân công, mã này sẽ được sử dụng ở các kì học và lớp học, thuận tiện cho việc xếp luôn cả giảng viên và môn học vào các lớp theo kì.
    - Môn học mà phân công này chỉ định.
    - Giảng viên được phân công dạy môn học tương ứng.
- Khi đưa vào bảng thì `Assignment` sẽ có các cột như sau:
    - AsnID: Mã nhận diện 1 phân công, cột này chính là primary key của bảng.
    - Major: Môn học mà phân công này chỉ định.
    - Lecturer: Giảng viên được phân công dạy môn học tương ứng.
- 2 cột `Major` và `Lecturer` là 1 candidate key của bảng `Assignment`.

- Bảng này cũng toàn các key nên bản thân nó cũng đã đạt chuẩn 3rd Normal Form.

7. **Classes**:

- Nhận thấy rằng 1 lớp thì sẽ có thể có nhiều nhóm được xếp cho tham gia vào và do vậy cũng kéo theo việc 1 lớp có nhiều điểm trung bình từ các nhóm sinh viên khác nhau.

  → Khi đưa vào bảng thì các attributes này sẽ có phụ thuộc bắc cầu với nhau.

  → Chưa đạt chuẩn 2nd.

  → Các attributes này cũng là các multiple values attributes.

  → Chea đạt cả chuẩn 1st.

  → Phải tách hết những attributes này ra, đưa vào entities hoặc relationships khác.

- Ngoài ra, như đã nói ở trên, chúng ta sẽ sử dụng 1 quan hệ `Classes` N—1 `Assignments`, để có thể lấy được môn học và giảng viên dạy môn học đó phân vào các lớp. 1 lớp thì chỉ có thể có 1 phân công được bổ nhiệm vào, và 1 phân công thì được giao vào nhiều lớp khác nhau.
- Sau khi tách bỏ hết, bảng `Classes` sẽ có các cột sau:
  - `ClassID`: Mã số nhận diện một lớp học. Mã này cũng là unique đối với mỗi lớp học. Đây cũng chính là primary key của bảng `Classes`.
  - AsnID: Mã phân công được giao cho lớp tương ứng.
  - Semester: Kì học mà lớp này có được phân cho các nhóm sinh viên.
  - Start Date: Ngày bắt đầu của kì học tương ứng.
  - End Date: Ngày kết thúc của kì học tương ứng.
- Sau khi tách hết các multiple values attributes, bảng `Classes` giờ chỉ còn 1 primary key là `ClassID`, không có non-key attributes nào phụ thuộc 1 phần vào key, không có non-key attributes nào phụ thuộc bắc cầu vào key.

  → Bảng `Classes` đã đạt chuẩn 3rd Normal Form.

8. **Enroll**:

- Bây giờ, để biết nhóm sinh viên nào được giao vào lớp nào tại kì học nào, chúng ta cần 1 bảng thể hiện quan hệ giữa 2 entities `Classes` N—N `Groups`.
- Các multiple values attributes mà ta tách ra từ 2 bảng trên cũng sẽ được thể hiện ở đây.
- Gồm các cột:
  - `ClassID`: Mã nhận diện lớp.
  - `GrID`: Mã nhận diện nhóm được phân vào học lớp tương ứng.
- 2 cột này cũng tạo thành primary key của bảng `Enroll`.
- Do bảng chỉ có 2 cột và 2 cột này là primary key nên bảng này cũng đạt chuẩn 3rd Normal From.

9. **Categories**:

- Sau khi tách các multiple values attributes liên quan đến các đầu điểm đánh giá môn học từ bảng `Courses` thì ta sẽ đặt những attributes đó vào bảng `Categories` này.
- Tuy nhiên, tùy vào từng môn học mà việc phân loại này có thể khác nhau. Ví dụ: 1 đầu điểm ở 1 môn học này có trọng số 0.05 nhưng ở một môn khác thì sẽ có trọng số 0.1, thời gian làm các bài test ứng với từng môn học cũng khác nhau tùy theo đặc trưng yêu cầu của môn học.

  → Khi đưa vào bảng, sẽ hình thành các multiple values attributes.

  → Còn chưa đạt chuẩn 1st.

  → Phải tách hết những attributes này ra, đưa vào entities hoặc relationships khác.
- Sau khi tách bỏ hết, ta được bảng `Categories` với các cột:

    - `CatID`: Mã nhận diện loại điểm. Đây chính là primary key của bảng này.
    - Name: Tên gọi của bài test hoặc loại đầu điểm.
    - Type: Hình thức kiểm tra, đánh giá.
    - Completion Criteria: Điều kiện pass bài test.
- Bảng `Categories` giờ chỉ còn 1 primary key là `ClassID`, không có non-key attributes nào phụ thuộc 1 phần vào key, không có non-key attributes nào phụ thuộc bắc cầu vào key.

  → Bảng `Categories` đã đạt chuẩn 3rd Normal Form.

10. **Assessment System:**

- Từng môn học sẽ có hệ thống đánh giá riêng phù hợp theo các tiêu chí của khung chương trình. Môn học có thể có các loại đầu điểm khác nhau, thời lượng làm bài khác nhau, số câu hỏi cũng khác nhau, trọng số cũng có thể khác nhau.
- `Assessment System` phản ánh quan hệ của 2 entities `Courses` N—N `Categories` nên nó sẽ được đặt vào 1 bảng thể hiện được quan hệ này.
- Tuy nhiên, ta nhận thấy rằng `Assessment System` sẽ còn liên quan nhiều đến phần chấm điểm cho sinh viên của giảng viên và việc xem điểm số của mình của sinh viên. Do vậy ta đặt `Assessment System` làm 1 entity hoàn chỉnh.
- Bảng này sẽ chứa các attributes của việc đánh giá 1 môn học dựa theo loại đầu điểm tương ứng. Do vậy, bảng này sẽ gồm các côt:
    - `AsID`: Mã nhận diện 1 hệ đánh giá điểm. Mã này là unique đối với mỗi hệ đánh giá và cũng chính là primary key phân định các attributes khác của bảng `Assessment System`.
    - CatID: Mã nhận diện loại điểm và bài test sẽ được sử dụng để đánh giá 1 môn học tương ứng.
    - CouID: Mã nhận diện môn học.

    - Number of Questions: Số lượng câu hỏi sẽ được sử dụng trong bài test tương ứng.
    - Duration: Thời gian sinh viên làm bài test tương ứng.
    - Weight: Trọng số của đầu điểm này ảnh hướng đến kết quả sau cùng của môn học. trọng số của 1 đầu điểm sẽ nằm trong khoảng từ 0 đến 1.
- Bảng này chỉ có 1 key là `AsnID`. Do vậy, không có non-key attributes nào phụ thuộc 1 phần vào key, không có non-key attributes nào phụ thuộc bắc cầu vào key.

  → Bảng `Categories` đã đạt chuẩn 3rd Normal Form.

11. **Grade**:

- Sau khi đã hoàn thành bảng `Assessment System`, ta đã có đầy đủ mọi thứ cần thiết để cho điểm đánh giá cho sinh viên. Do vậy bảng `Assessment System` sẽ có 1 liên kết với bảng `Students`.
- Liên kết này phải thể hiện được 1 sinh viên được cho điểm theo nhiều môn và do vậy cũng có nhiều các hệ đánh giá các môn cho sinh viên đó. Ngoài ra, các hệ đánh giá này lại được áp dụng cho nhiều sinh viên.
- Do vậy, ta đặt liên kết này vảo 1 bảng tên là `Grade` để có thể thuận tiện sử dụng.
- Bảng này sẽ gồm các cột sau:
    - `Mssv`: Mã nhận diện 1 sinh viên được trường cấp cho.
    - `AsID`: Mã nhận diện hệ đánh giá điểm của sinh viên tương ứng.
    - Score: Điểm do giảng viên cho 1 sinh viên với 1 hệ đánh giá tương ứng.
    - `Date`: Ngày cho điểm.
- 3 cột `Mssv`, `AsID`, và `Date` cùng nhau tạo nên primary key cho bảng `Grade`.
- Ứng với 1 primary key gồm 3 attributes, chỉ có thể cho ra 1 attribute duy nhất. Do vậy, không có non-key attributes nào phụ thuộc 1 phần vào key, không có non-key attributes nào phụ thuộc bắc cầu vào key.

→ Bảng `Grade` đã đạt chuẩn 3rd Normal Form.

12. **View**:

- Như đã biết, 1 sinh viên thì tham gia nhiều các lớp và 1 lớp thì lại cũng có nhiều sinh viên tham gia. tuy nhiên các thông tin này chưa được thể hiện ra sau khi tạo ra các bảng trước do đã cắt đi những multiple values attributes để chuẩn hóa bảng.
- Do vậy, ta tạo thêm bảng `View` thể hiện được quan hệ giữa 2 entities `Students` N—N `Classes`. Ngoài ra, để tối ưu, ta sẽ thêm luôn điểm trung bình và trạng thái môn học dành cho việc nhập xuất dữ liệu sau này được thuận tiện hơn.
- Bảng `View` sẽ gồm các cột sau:
  - `Mssv`: Mã số sinh viên được cấp bởi nhà trường cho 1 sinh viên.
  - `ClassID`: Mã số nhận diện một lớp học.
  - Average: Điểm trung bình môn của môn học mà sinh viên đã tham gia học ở lớp học tương ứng.
  - Status: Trạng thái của môn học mà sinh viên đã tham gia học ở lớp học tương ứng.
- 2 cột `Mssv` và `ClassID` cùng với nhau tạo ra primary key cho bảng `View` này.
- Ứng với 1 primary key gồm 3 attributes, cho ra 2 attributes tách biệt, không phụ thuộc. Do vậy, không có non-key attributes nào phụ thuộc 1 phần vào key, không có non-key attributes nào phụ thuộc bắc cầu vào key.

  → Bảng `View` đã đạt chuẩn 3rd Normal Form.

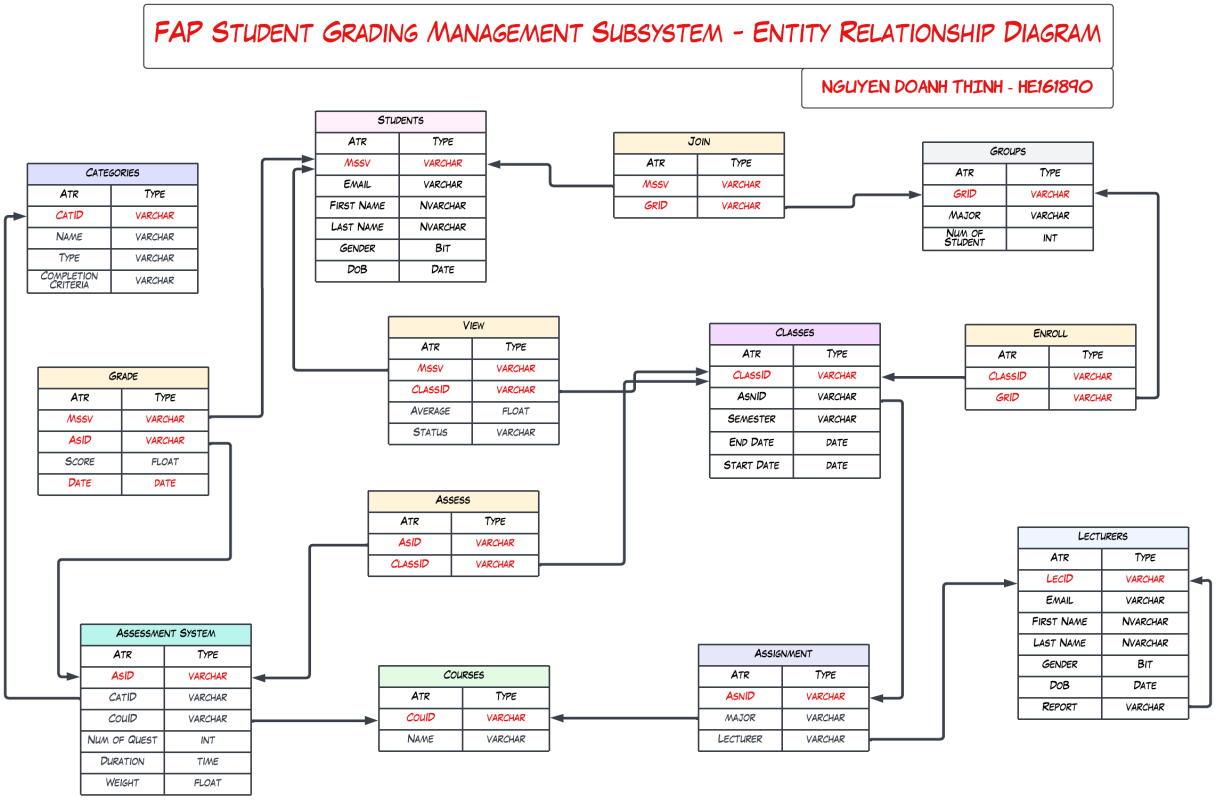
13. **Assess**:

- Ở mỗi lớp, môn học được đánh giá theo hệ thống đánh giá đặc trưng theo môn học đó.

- Các bảng trên xây dựng hệ thống thông tin và các liên kết để 1 sinh viên có thể xem được điểm và trạng thái môn của mình.
- Tuy nhiên, về phía các lớp và các giảng viên thì có phần khó khăn trong việc chấm điểm cũng như lấy ra thông tin các lớp học.
- Tận dụng quan hệ của môn học với hệ thống đánh giá điểm, có thể tạo ra bảng `Assess` để vừa thể hiện được quan hệ này, vừa thuận tiện cho việc nhập xuất dữ liệu sau này.
- Bảng này sẽ gồm các cột:
  - `AsID`: Mã nhận diện 1 hệ đánh giá điểm.
  - `ClassID`: Mã số nhận diện một lớp học.
- 2 cột `AsID` và `ClassID` cùng với nhau tạo ra primary key cho bảng `Assess` này.
- Bảng này gồm key của nó. Do vậy, không có non-key attributes nào phụ thuộc 1 phần vào key, không có non-key attributes nào phụ thuộc bắc cầu vào key.

  → Bảng `Assess` đã đạt chuẩn 3rd Normal Form.

V. ***Module Diagram:***



VI. ***Phân tích MD và dữ liệu sẽ được add vào database:***

1. **Students:**

Students	
Atr	Type
Mssv	varchar
Email	varchar
First Name	Nvarchar
Last Name	Nvarchar
Gender	Bit
DoB	Date

- Mssv:
  - Như đã nói, đây là primary key của bảng `Students`.
  - Dữ liệu này sẽ có dạng: MMKKXXXXXX
    - Ở vị trí MM là các kí tự biểu thị khoa mà sinh viên theo học.
    - Ở vị trí KK là các số biểu thị niên khóa mà sinh viên theo học.
    - Các vị trí còn lại sẽ là số hiệu của sinh viên, có thể là số thứ tự tuyển sinh.
    - Ví dụ: HE161890 - khoa công nghệ thông tin, niên khóa thứ 16, số thứ tự 1890 -
  - Kiểu dữ liệu: `VARCHAR`, độ dài là 50 kí tự. Kì 18 trường sẽ có khoảng hơn 18 nghìn sinh viên nên để có thể mở rộng nhiều thêm và áp dụng được cả vào các trường khác đông hơn, ta để độ dài 50.
- Email:
  - Dữ liệu này sẽ có dạng: FFFFFLLIIIII@SSSS.edu.vn
    - Ở vị trí FFFF là tên của sinh viên.
    - Ở vị trí LL là họ và tên đệm của sinh viên ở dạng viết tắt bằng các chữ cái bắt đầu của họ và tên đệm.
    - Ở vị trí IIIII là mã số sinh viên của sinh viên.

        - Ở vị trí SSSS là tên trường học (ở đây là FPT).
        - Ví dụ: tndhe161890@fpt.edu.vn - tên T, họ và tên đệm viết tắt là ND, mssv là HE161890, trường fpt -
    - Kiểu dữ liệu: `VARCHAR`, độ dài là 150 kí tự. Cũng như `Mssv`, khi mở rộng `Mssv` thì `Email` cũng sẽ được mở rộng ra.
- First Name: Kiểu dữ liệu là `NVARCHAR` để có thể viết dấu tiếng Việt, độ dài 50.
- Last Name: Kiểu dữ liệu là `NVARCHAR` để có thể viết dấu tiếng Việt, độ dài 150.
- Gender: Kiểu dữ liệu là `BIT`, 1 được coi là nam, 0 là nữ.
- Date of Birth: Kiểu dữ liệu là `DATE`.

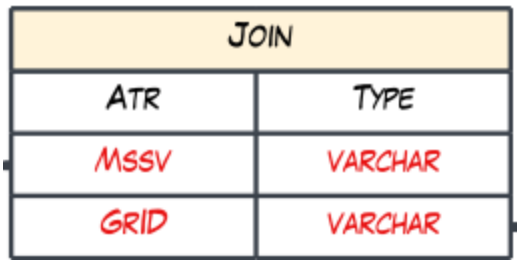
2. **Groups:**

GROUPS	
ATR	TYPE
GRID	VARCHAR
MAJOR	VARCHAR
NUM OF STUDENT	INT

- GrID:
    - Như đã nói, đây là primary key của bảng `Groups`.
    - Dữ liệu này sẽ có dạng: MMMKKXXXX
        - Ở vị trí MMM là các chữ cái đầu tiên viết tắt của tên chuyên ngành mà hầu hết sinh viên trong nhóm theo học.
        - Ở vị trí KK là các số thể hiện niên khóa của nhóm sinh viên.
        - Ở vị trí XXXX là các số thể hiện số thứ tự của nhóm trong danh sách.

        - Ví dụ: IA1604 - chuyên ngành IA (Information Assurance), niên khóa 16, stt 04 -
    - Kiểu dữ liệu là `VARCHAR`, độ dài 50.
- Major: Kiểu dữ liệu là `VARCHAR`, độ dài 50.
- Number of Students: Kiểu dữ liệu là `INT`.

3. **Join:**



- Mssv: Tương ứng với `Mssv` của `Students`.
- GrID: Tương ứng với `GrID` của `Groups`.

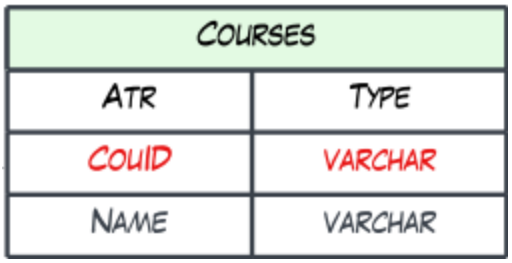
4. **Lecturers:**

Lecturers	
Atr	Type
LecID	VARCHAR
Email	VARCHAR
First Name	NVARCHAR
Last Name	NVARCHAR
Gender	Bit
DoB	Date
Report	VARCHAR

- LecID:
  - Như đã nói, đây là primary key của bảng `Lecturers`.
  - Dữ liệu này sẽ có dạng: FFFLLLL(TT)
    - Ở vị trí FF là các kí tự biểu thị tên của giảng viên.
    - Ở vị trí LLLL là các kí tự biểu thị các chữ cái đầu tiên viết tắt của họ tên giảng viên.
    - Vị trí còn lại là số hiệu của giảng viên (optional)
    - Ví dụ: snt5 - tên giảng viên là S, họ và tên đệm viết tắt bởi 2 chữ NT, số hiệu 5 (optional) -
  - Kiểu dữ liệu: `VARCHAR`, độ dài là 50 kí tự.
- Email:
  - Dữ liệu này sẽ có dạng: FFFFFLLTT@SSSS.edu.vn
    - Ở vị trí FF là các kí tự biểu thị tên của giảng viên.
    - Ở vị trí LLLL là các kí tự biểu thị các chữ cái đầu tiên viết tắt của họ tên giảng viên.
    - Ở vị trí TT là số hiệu của giảng viên.

        - Ở vị trí SSSS là tên trường học (ở đây là FE - FPT Education -).
        - Ví dụ: snt5@fe.edu.vn - tên giảng viên là S, họ và tên đệm viết tắt bởi 2 chữ NT, số hiệu 5, trường FPT Education -
    - Kiểu dữ liệu: `VARCHAR`, độ dài là 150 kí tự.
- First Name: Kiểu dữ liệu là `NVARCHAR` để có thể viết dấu tiếng Việt, độ dài 50.
- Last Name: Kiểu dữ liệu là `NVARCHAR` để có thể viết dấu tiếng Việt, độ dài 150.
- Gender: Kiểu dữ liệu là `BIT`, 1 được coi là nam, 0 là nữ.
- Date of Birth: Kiểu dữ liệu là `DATE`.
- Report: Tương ứng với `LecID` của giảng viên làm trương bộ môn.

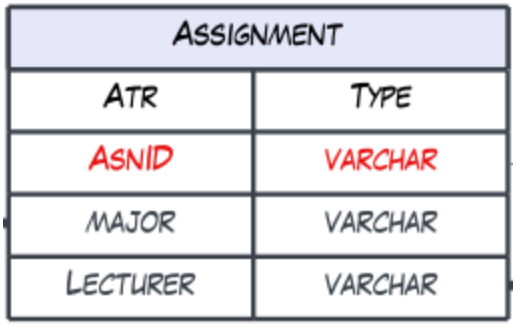
5. **Courses:**



- CouID:
    - Dữ liệu này có dạng: CCCDDD
        - Ở vị trí CCC là các kí tự biểu thị các chữ cái viết hoa khi viết tắt của tên môn học.
        - Ở vị trí DDD là các số biểu thị mã số môn đó.
        - Ví dụ: DBI202 - tên viết tắt của môn là DBI (Database Introduction), mã số là 202 -
    - Kiểu dữ liệu: `VARCHAR`, độ dài là 50 kí tự.

- Name: Kiểu dữ liệu là `VARCHAR`, độ dài 50.

6. **Assignment:**



ASSIGNMENT	
ATR	TYPE
ASNID	VARCHAR
MAJOR	VARCHAR
LECTURER	VARCHAR

- AsnID:
  - Dữ liệu này có dạng: LLLOOCCC
    - Ở vị trí LLL là các kí tự biểu thị tên của giảng viên.
    - Ở vị trí OO là các kí tự biểu thị mã số của giảng viên nếu bị trùng tên.
    - Ở vị trí CCC là các kí tự biểu thị các chữ cái viết hoa khi viết tắt của tên môn học.
    - Ví dụ: S5DBI - tên S, số hiệu 5, mã môn DBI -
  - Kiểu dữ liệu: `VARCHAR`, độ dài là 50 kí tự.
- Major: Kiểu dữ liệu: `VARCHAR`, độ dài là 50 kí tự.
- Lecturer: Tương ứng với `LecID` của giảng viên dược phân công dạy bộ môn.

7. **Classes:**

CLASSES	
ATR	TYPE
CLASSID	VARCHAR
ASNID	VARCHAR
SEMESTER	VARCHAR
END DATE	DATE
START DATE	DATE

- ClassID:
  - Dữ liệu này có dạng: SSSSAAAA
    - Ở vị trí SSSS là các kí tự viết tắt của kì học mà lớp này được giao.
    - Ở vị trí AAAA là `AsnID` của class tương ứng.
    - Ví dụ: SP22S5DBI - kì học là SP22 (Spring 2022), AsnID là S5DBI -
  - Kiểu dữ liệu: `VARCHAR`, độ dài là 50 kí tự.
- AsnID: Tương ứng với `AsnID` của bảng `Assignment`.
- Semester: Kiểu dữ liệu: `VARCHAR`, độ dài là 50 kí tự.
- End Date: Kiểu dữ liệu là `DATE`.
- Start Date: Kiểu dữ liệu là `DATE`.

8. **Enroll:**

ENROLL	
ATR	TYPE
CLASSID	VARCHAR
GRID	VARCHAR

- ClassID: Tương ứng với `ClassID` của bảng `Classes`.
- GrID: Tương ứng với `GrID` của bảng `Groups`.

9. **Categories:**

CATEGORIES	
ATR	TYPE
CATID	VARCHAR
NAME	VARCHAR
TYPE	VARCHAR
COMPLETION CRITERIA	VARCHAR

- CatID:
  - Dữ liệu này là dạng viết tắt của tên loại đầu điểm.
  - Kiểu dữ liệu: `VARCHAR`, độ dài là 50 kí tự.
  - Ví dụ: AS - Assignment -
- Type: Kiểu dữ liệu là `VARCHAR`, độ dài là 50 kí tự.
- Completion Criteria: Kiểu dữ liệu là `VARCHAR`, độ dài là 50 kí tự.

10. **Assessment System:**

Assessment System	
Atr	Type
AsID	VARCHAR
CatID	VARCHAR
CouID	VARCHAR
Num of Quest	INT
Duration	TIME
Weight	FLOAT

- AsID:
  - Dữ liệu này có dạng: CCCSSS
    - Ở vị trí CCC là biểu thị tên môn học ở dạng viết hoa các chữ cái đầu.
    - Ở vị trí SSS là biểu thị tên đầu điểm đánh giá / bài test ở dạng viết hoa các chữ cái đầu.
    - Ví dụ: DBIAS - môn học DBI202, đầu điểm là Assignment -
  - Kiểu dữ liệu: `VARCHAR`, độ dài là 50 kí tự.
- CatID: Tương ứng với `CatID` của bảng `Categories`.
- CouID: Tương ứng với `CouID` của bảng `Courses`.
- Number of Questions: Kiểu dữ liệu là `INT`.
- Duration: Kiểu dữ liệu là `VARCHAR`, độ dài là 50 kí tự.
- Weight: Kiểu dữ liệu là `FLOAT`.

11. **Grade:**

GRADE	
ATR	TYPE
MSSV	VARCHAR
AsID	VARCHAR
SCORE	FLOAT
DATE	DATE

- Mssv: Tương ứng với `Mssv` của bảng `Students`.
- AsID: Tương ứng với `AsID` của bảng `Assessment System`.
- Score: Kiểu dữ liệu là `FLOAT`.
- Date: Kiểu dữ liệu là `DATE`.

12. **View:**

VIEW	
ATR	TYPE
MSSV	VARCHAR
CLASSID	VARCHAR
AVERAGE	FLOAT
STATUS	VARCHAR

- Mssv: Tương ứng với `Mssv` của bảng `Students`.
- ClassID: Tương ứng với `ClassID` của bảng `Classes`.

- Average: Kiểu dữ liệu là `FLOAT`.
- Status: Kiểu dữ liệu là `VARCHAR`, độ dài là 50 kí tự.

13. **Assess:**

Assess	
Atr	Type
AsID	VARCHAR
ClassID	VARCHAR

- AsID: Tương ứng với `AsID` của bảng `Assessment System`.
- ClassID: Tương ứng với `ClassID` của bảng `Classes`.

VII: ***Tạo Database và insert dữ liệu:***

1. **Tạo Datdbase và các bảng:**

```
CREATE DATABASE [Student Grading Management Subsystem]

USE [Student Grading Management Subsystem]

CREATE TABLE Students (
  [Mssv] [varchar](50) NOT NULL PRIMARY KEY,
  [Email] [varchar](150) NOT NULL,
  [First Name] [nvarchar](50) NOT NULL,
  [Last Name] [nvarchar](150) NOT NULL,
  [Gender] [bit] NOT NULL,
  [Date of Birth] [date] NOT NULL,
)

CREATE TABLE Groups (
  [GrID] [varchar](50) NOT NULL PRIMARY KEY,
```

```
  [Major] [varchar](50) NOT NULL,
  [Number of Students] [int] NOT NULL,
)

CREATE TABLE [Join] (
  [Mssv] [varchar](50) NOT NULL,
  [GrID] [varchar](50) NOT NULL,

  CONSTRAINT [PK_Join] PRIMARY KEY (Mssv, GrID),
  CONSTRAINT [FK_Join_Groups] FOREIGN KEY([GrID]) REFERENCES [dbo].[Groups] ([GrID]),
  CONSTRAINT [FK_Join_Students] FOREIGN KEY([Mssv]) REFERENCES [dbo].[Students] ([Mss
v])
)

CREATE TABLE Lecturers (
  [LecID] [varchar](50) NOT NULL PRIMARY KEY,
  [Email] [varchar](150) NOT NULL,
  [First Name] [nvarchar](50) NOT NULL,
  [Last Name] [nvarchar](150) NOT NULL,
  [Gender] [bit] NOT NULL,
  [Date of Birth] [date] NOT NULL,
  [Report] [varchar](50) NULL,

  CONSTRAINT [FK_Lecturers_Lecturers] FOREIGN KEY([Report]) REFERENCES [dbo].[Lecturer
s] ([LecID]),
)

CREATE TABLE Courses (
  [CouID] [varchar](50) NOT NULL PRIMARY KEY,
  [Name] [varchar](50) NOT NULL,
)

CREATE TABLE Assignment (
  [AsnID] [varchar](50) NOT NULL PRIMARY KEY,
  [Major] [varchar](50) NOT NULL,
  [Lecturer] [varchar](50) NOT NULL,

  CONSTRAINT [FK_Assignment_Courses] FOREIGN KEY([Major]) REFERENCES [dbo].[Courses]
 ([CouID]),
  CONSTRAINT [FK_Assignment_Lecturers] FOREIGN KEY([Lecturer]) REFERENCES [dbo].[Lectu
rers] ([LecID]),
)

CREATE TABLE Classes (
  [ClassID] [varchar](50) NOT NULL PRIMARY KEY,
  [AsnID] [varchar](50) NOT NULL,
  [Semester] [varchar](50) NOT NULL,
  [Start Date] [date] NOT NULL,
  [End Date] [date] NOT NULL,

  CONSTRAINT [FK_Classes_Assignment] FOREIGN KEY([AsnID]) REFERENCES [dbo].[Assignmen
t] ([AsnID]),

)
```

```
CREATE TABLE Enroll (
  [ClassID] [varchar](50) NOT NULL,
  [GrID] [varchar](50) NOT NULL,

  CONSTRAINT [PK_Enroll] PRIMARY KEY (ClassID, GrID),
  CONSTRAINT [FK_Enroll_Classes] FOREIGN KEY([ClassID]) REFERENCES [dbo].[Classes] ([C
lassID]),
  CONSTRAINT [FK_Enroll_Groups] FOREIGN KEY([GrID]) REFERENCES [dbo].[Groups] ([GrID])
)

CREATE TABLE Categories (
  [CatID] [varchar](50) NOT NULL PRIMARY KEY ,
  [Name] [varchar](50) NOT NULL,
  [Type] [varchar](50) NOT NULL,
  [Completion Criteria] [varchar](50) NOT NULL,
)

CREATE TABLE [Assessment System] (
  [AsID] [varchar](50) NOT NULL PRIMARY KEY,
  [CatID] [varchar](50) NOT NULL,
  [CouID] [varchar](50) NOT NULL,
  [Number of Questions] [int] NOT NULL,
  [Duration] [varchar](50) NOT NULL,
  [Weight] [float] NOT NULL,

  CONSTRAINT [FK_Assessment System_Categories] FOREIGN KEY([CatID]) REFERENCES [dbo].
[Categories] ([CatID]),
  CONSTRAINT [FK_Assessment System_Courses] FOREIGN KEY([CouID]) REFERENCES [dbo].[Cou
rses] ([CouID]),

)

CREATE TABLE Grade (
  [Mssv] [varchar](50) NOT NULL,
  [AsID] [varchar](50) NOT NULL,
  [Score] [float] NOT NULL,
  [Date] [date] NOT NULL,

  CONSTRAINT [FK_Grade_Assessment System] FOREIGN KEY([AsID]) REFERENCES [dbo].[Assess
ment System] ([AsID]),
  CONSTRAINT [FK_Grade_Students] FOREIGN KEY([Mssv]) REFERENCES [dbo].[Students] ([Mss
v])

  CONSTRAINT [PK_Grade] PRIMARY KEY (Mssv, AsID, [Date]),
)

CREATE TABLE [View] (
  [Mssv] [varchar](50) NOT NULL,
  [ClassID] [varchar](50) NOT NULL,
  [Average] [float] NOT NULL,
  [Status] [varchar](50) NOT NULL,

  CONSTRAINT [PK_View] PRIMARY KEY (Mssv, ClassID),
```

```
  CONSTRAINT [FK_View_Classes] FOREIGN KEY([ClassID]) REFERENCES [dbo].[Classes] ([Cla
ssID]),
  CONSTRAINT [FK_View_Students] FOREIGN KEY([Mssv]) REFERENCES [dbo].[Students] ([Mss
v]),
)

CREATE TABLE Assess (
  [AsID] [varchar](50) NOT NULL,
  [ClassID] [varchar](50) NOT NULL,

  CONSTRAINT [PK_Assess] PRIMARY KEY (AsID, ClassID),
  CONSTRAINT [FK_Assess_Assessment System] FOREIGN KEY([AsID]) REFERENCES [dbo].[Asses
sment System] ([AsID]),
  CONSTRAINT [FK_Assess_Classes] FOREIGN KEY([ClassID]) REFERENCES [dbo].[Classes] ([C
lassID])
)
```

1. **Insert dữ liệu:**

```
INSERT Students(Mssv, Email, [First Name], [Last Name], Gender, [Date of Birth]) VALUES
 ('HE00027', 'anv@fpt.edu.vn', 'A', N'Nguyễn Thị', 0, CAST('02-01-2002' AS DATE));
INSERT Students(Mssv, Email, [First Name], [Last Name], Gender, [Date of Birth]) VALUES
 ('HE00028', 'bnv@fpt.edu.vn', 'B', N'Nguyễn Thị', 0, CAST('02-02-2002' AS DATE));
INSERT Students(Mssv, Email, [First Name], [Last Name], Gender, [Date of Birth]) VALUES
 ('HE00029', 'cnv@fpt.edu.vn', 'C', N'Nguyễn Thị', 0, CAST('02-03-2002' AS DATE));
INSERT Students(Mssv, Email, [First Name], [Last Name], Gender, [Date of Birth]) VALUES
 ('HE00030', 'dnv@fpt.edu.vn', 'D', N'Nguyễn Thị', 0, CAST('02-04-2002' AS DATE));
INSERT Students(Mssv, Email, [First Name], [Last Name], Gender, [Date of Birth]) VALUES
 ('HE00031', 'env@fpt.edu.vn', 'E', N'Nguyễn Thị', 0, CAST('02-05-2002' AS DATE));
INSERT Students(Mssv, Email, [First Name], [Last Name], Gender, [Date of Birth]) VALUES
 ('HE00032', 'fnv@fpt.edu.vn', 'F', N'Nguyễn Thị', 0, CAST('02-06-2002' AS DATE));
INSERT Students(Mssv, Email, [First Name], [Last Name], Gender, [Date of Birth]) VALUES
 ('HE00033', 'gnv@fpt.edu.vn', 'G', N'Nguyễn Thị', 0, CAST('02-07-2002' AS DATE));
INSERT Students(Mssv, Email, [First Name], [Last Name], Gender, [Date of Birth]) VALUES
 ('HE00039', 'nnv@fpt.edu.vn', 'N', N'Nguyễn Thị', 0, CAST('02-14-2002' AS DATE));

INSERT Groups(GrID, Major, [Number of Students]) VALUES ('IA1604', 'Information Assuranc
e', 15);
INSERT Groups(GrID, Major, [Number of Students]) VALUES ('AI1605', 'Artificial Intelligenc
e', 15);
INSERT Groups(GrID, Major, [Number of Students]) VALUES ('SE1606', 'Software Engineering',
20);

INSERT [Join](Mssv, GrID) VALUES ('HE00001', 'IA1604');
INSERT [Join](Mssv, GrID) VALUES ('HE00002', 'IA1604');
INSERT [Join](Mssv, GrID) VALUES ('HE00003', 'IA1604');
INSERT [Join](Mssv, GrID) VALUES ('HE00004', 'IA1604');
```

```
INSERT [Join](Mssv, GrID) VALUES ('HE00005', 'IA1604');
INSERT [Join](Mssv, GrID) VALUES ('HE00001', 'IA1604');
INSERT [Join](Mssv, GrID) VALUES ('HE00002', 'IA1604');
INSERT [Join](Mssv, GrID) VALUES ('HE00003', 'IA1604');
INSERT [Join](Mssv, GrID) VALUES ('HE00004', 'IA1604');
INSERT [Join](Mssv, GrID) VALUES ('HE00005', 'IA1604');
INSERT [Join](Mssv, GrID) VALUES ('HE00006', 'IA1604');
INSERT [Join](Mssv, GrID) VALUES ('HE00007', 'IA1604');
INSERT [Join](Mssv, GrID) VALUES ('HE00008', 'IA1604');
INSERT [Join](Mssv, GrID) VALUES ('HE00009', 'IA1604');
INSERT [Join](Mssv, GrID) VALUES ('HE00010', 'IA1604');
INSERT [Join](Mssv, GrID) VALUES ('HE00011', 'IA1604');
INSERT [Join](Mssv, GrID) VALUES ('HE00012', 'IA1604');

INSERT Lecturers(LecID, Email, [First Name], [Last Name], Gender, [Date of Birth], Report)
VALUES ('GVA', 'gva@fe.edu.vn', 'A', N'Giảng Viên', 1, CAST('01-01-1985' AS date), NULL);
INSERT Lecturers(LecID, Email, [First Name], [Last Name], Gender, [Date of Birth], Report)
VALUES ('GVB', 'gvb@fe.edu.vn', 'B', N'Giảng Viên', 0, CAST('01-02-1985' AS date), 'GVA');
INSERT Lecturers(LecID, Email, [First Name], [Last Name], Gender, [Date of Birth], Report)
VALUES ('GVC', 'gvc@fe.edu.vn', 'C', N'Giảng Viên', 1, CAST('01-03-1985' AS date), 'GVA');
INSERT Lecturers(LecID, Email, [First Name], [Last Name], Gender, [Date of Birth], Report)
VALUES ('GVG', 'gvg@fe.edu.vn', 'G', N'Giảng Viên', 1, CAST('01-07-1985' AS date), 'GVF');
INSERT Lecturers(LecID, Email, [First Name], [Last Name], Gender, [Date of Birth], Report)
VALUES ('GVH', 'gvh@fe.edu.vn', 'H', N'Giảng Viên', 0, CAST('01-08-1985' AS date), NULL);
INSERT Lecturers(LecID, Email, [First Name], [Last Name], Gender, [Date of Birth], Report)
VALUES ('GVI', 'gvi@fe.edu.vn', 'I', N'Giảng Viên', 1, CAST('01-09-1985' AS date), 'GVH');
INSERT Lecturers(LecID, Email, [First Name], [Last Name], Gender, [Date of Birth], Report)
VALUES ('GVJ', 'gvj@fe.edu.vn', 'J', N'Giảng Viên', 0, CAST('01-10-1985' AS date), 'GVH');
INSERT Lecturers(LecID, Email, [First Name], [Last Name], Gender, [Date of Birth], Report)
VALUES ('GVK', 'gvk@fe.edu.vn', 'K', N'Giảng Viên', 1, CAST('01-11-1985' AS date), 'GVH');

INSERT Assignment(AsnID, Major, Lecturer) VALUES ('ACSD', 'CSD201', 'GVA');
INSERT Assignment(AsnID, Major, Lecturer) VALUES ('BIAO', 'IAO202', 'GVB');
INSERT Assignment(AsnID, Major, Lecturer) VALUES ('CLAB', 'LAB211', 'GVC');
INSERT Assignment(AsnID, Major, Lecturer) VALUES ('DLAB', 'LAB211', 'GVD');
INSERT Assignment(AsnID, Major, Lecturer) VALUES ('FJPD', 'JPD113', 'GVF');
INSERT Assignment(AsnID, Major, Lecturer) VALUES ('GJPD', 'JPD113', 'GVG');
INSERT Assignment(AsnID, Major, Lecturer) VALUES ('IMAE', 'MAE101', 'GVI');
INSERT Assignment(AsnID, Major, Lecturer) VALUES ('JMAE', 'MAE101', 'GVJ');
INSERT Assignment(AsnID, Major, Lecturer) VALUES ('ISSL', 'SSL101c', 'GVI');
INSERT Assignment(AsnID, Major, Lecturer) VALUES ('JMAD', 'MAD101', 'GVJ');

INSERT Classes(ClassID, AsnID, Semester, [Start Date], [End Date]) VALUES ('SP22ACSD', 'AC
SD', 'SPRING 2022', CAST('01-01-2022' AS date), CAST('04-01-2022' AS date));
INSERT Classes(ClassID, AsnID, Semester, [Start Date], [End Date]) VALUES ('SP22CLAB', 'CL
AB', 'SPRING 2022', CAST('01-01-2022' AS date), CAST('04-01-2022' AS date));
INSERT Classes(ClassID, AsnID, Semester, [Start Date], [End Date]) VALUES ('SP22DLAB', 'DL
AB', 'SPRING 2022', CAST('01-01-2022' AS date), CAST('04-01-2022' AS date));
INSERT Classes(ClassID, AsnID, Semester, [Start Date], [End Date]) VALUES ('SP22FJPD', 'FJ
PD', 'SPRING 2022', CAST('01-01-2022' AS date), CAST('04-01-2022' AS date));
INSERT Classes(ClassID, AsnID, Semester, [Start Date], [End Date]) VALUES ('SP22GJPD', 'GJ
PD', 'SPRING 2022', CAST('01-01-2022' AS date), CAST('04-01-2022' AS date));
INSERT Classes(ClassID, AsnID, Semester, [Start Date], [End Date]) VALUES ('SP22JMAD', 'JM
AD', 'SPRING 2022', CAST('01-01-2022' AS date), CAST('04-01-2022' AS date));
```

```
INSERT Classes(ClassID, AsnID, Semester, [Start Date], [End Date]) VALUES ('SP22BNWC', 'BN
WC', 'SPRING 2022', CAST('01-01-2022' AS date), CAST('04-01-2022' AS date));
INSERT Classes(ClassID, AsnID, Semester, [Start Date], [End Date]) VALUES ('SP22HOSG', 'HO
SG', 'SPRING 2022', CAST('01-01-2022' AS date), CAST('04-01-2022' AS date));
INSERT Classes(ClassID, AsnID, Semester, [Start Date], [End Date]) VALUES ('SP22FSSG', 'FS
SG', 'SPRING 2022', CAST('01-01-2022' AS date), CAST('04-01-2022' AS date));
INSERT Classes(ClassID, AsnID, Semester, [Start Date], [End Date]) VALUES ('SP22GSSG', 'GS
SG', 'SPRING 2022', CAST('01-01-2022' AS date), CAST('04-01-2022' AS date));

INSERT Enroll(ClassID, GrID) VALUES ('SU21KCSI', 'IA1604')
INSERT Enroll(ClassID, GrID) VALUES ('SU21IMAE', 'IA1604')
INSERT Enroll(ClassID, GrID) VALUES ('SU21JCEA', 'IA1604')
INSERT Enroll(ClassID, GrID) VALUES ('FA21JMAD', 'IA1604')
INSERT Enroll(ClassID, GrID) VALUES ('SP22BCSD', 'IA1604')

INSERT Categories(CatID, [Name], [Type], [Completion Criteria]) VALUES ('AS', 'Assignment
 ', 'On-Going', '>0');
INSERT Categories(CatID, [Name], [Type], [Completion Criteria]) VALUES ('LAB', 'Lab', 'On-
Going', '>0');
INSERT Categories(CatID, [Name], [Type], [Completion Criteria]) VALUES ('LOC', 'Line of Co
de', 'On-Going', '>=750');
INSERT Categories(CatID, [Name], [Type], [Completion Criteria]) VALUES ('PT', 'Progress Te
st', 'On-Going', '>0');
INSERT Categories(CatID, [Name], [Type], [Completion Criteria]) VALUES ('PE', 'Practical E
xam', 'On-Going', '>0');
INSERT Categories(CatID, [Name], [Type], [Completion Criteria]) VALUES ('FE', 'Final Exa
m', 'Final Exam', '>4');
INSERT Categories(CatID, [Name], [Type], [Completion Criteria]) VALUES ('FER', 'Final Exam
Retake', 'Final Exam', '>4');

INSERT [Assessment System](AsID, CatID, CouID, [Number of Questions], Duration, [Weight])
 VALUES ('CEAAS1', 'AS1', 'CEA201', 20, '30 mins', 0.1);
INSERT [Assessment System](AsID, CatID, CouID, [Number of Questions], Duration, [Weight])
 VALUES ('CEAAS2', 'AS2', 'CEA201', 20, '30 mins', 0.1);
INSERT [Assessment System](AsID, CatID, CouID, [Number of Questions], Duration, [Weight])
 VALUES ('CEAPT1', 'PT1', 'CEA201', 30, '60 mins', 0.2);
INSERT [Assessment System](AsID, CatID, CouID, [Number of Questions], Duration, [Weight])
 VALUES ('CEAPT2', 'PT2', 'CEA201', 30, '60 mins', 0.2);
INSERT [Assessment System](AsID, CatID, CouID, [Number of Questions], Duration, [Weight])
 VALUES ('CEAFE', 'FE', 'CEA201', 50, '90 mins', 0.4);

INSERT Assess(AsID, ClassID) VALUES ('PROAS', 'FA21APRO')
INSERT Assess(AsID, ClassID) VALUES ('PROLAB1', 'FA21APRO')
INSERT Assess(AsID, ClassID) VALUES ('PROLAB2', 'FA21APRO')
INSERT Assess(AsID, ClassID) VALUES ('PROPT', 'FA21APRO')
INSERT Assess(AsID, ClassID) VALUES ('PROPE', 'FA21APRO')
INSERT Assess(AsID, ClassID) VALUES ('PROFE', 'FA21APRO')

INSERT Grade(Mssv, AsID, Score, [Date]) VALUES ('HE00001', 'JPDFE', 8.4, CAST('2021-08-01'
AS date))
INSERT Grade(Mssv, AsID, Score, [Date]) VALUES ('HE00001', 'JPDLAB1', 5, CAST('2021-08-01'
AS date))
INSERT Grade(Mssv, AsID, Score, [Date]) VALUES ('HE00001', 'JPDLAB2', 6, CAST('2021-08-01'
AS date))
```

```
INSERT Grade(Mssv, AsID, Score, [Date]) VALUES ('HE00001', 'JPDPT1', 8, CAST('2021-08-01'
 AS date))
INSERT Grade(Mssv, AsID, Score, [Date]) VALUES ('HE00001', 'JPDPT2', 9, CAST('2021-08-01'
 AS date))
INSERT Grade(Mssv, AsID, Score, [Date]) VALUES ('HE00001', 'MAEAS1', 10, CAST('2021-08-01'
AS date))
INSERT Grade(Mssv, AsID, Score, [Date]) VALUES ('HE00001', 'MAEAS2', 6.4, CAST('2021-08-0
1' AS date))
INSERT Grade(Mssv, AsID, Score, [Date]) VALUES ('HE00001', 'MAEFE', 2.5, CAST('2021-08-01'
AS date))

INSERT [dbo].[Courses] ([CouID], [Name]) VALUES (N'LAB211', N'OOP with Java Lab')
INSERT [dbo].[Courses] ([CouID], [Name]) VALUES (N'MAD101', N'Discrete mathematics')
INSERT [dbo].[Courses] ([CouID], [Name]) VALUES (N'MAE101', N'Mathematics for Engineerin
g')
INSERT [dbo].[Courses] ([CouID], [Name]) VALUES (N'NWC204', N'Computer Networking')
INSERT [dbo].[Courses] ([CouID], [Name]) VALUES (N'OSG202', N'Operating Systems')
INSERT [dbo].[Courses] ([CouID], [Name]) VALUES (N'PRF192', N'Programming Fundamentals')
INSERT [dbo].[Courses] ([CouID], [Name]) VALUES (N'PRO192', N'Object-Oriented Programmin
g')

INSERT [dbo].[View] ([Mssv], [ClassID], [Average], [Status]) VALUES (N'HE00006', N'FA21JMA
D', 7.415, N'PASSED')
INSERT [dbo].[View] ([Mssv], [ClassID], [Average], [Status]) VALUES (N'HE00006', N'SP22ADB
I', 5.965, N'PASSED')
INSERT [dbo].[View] ([Mssv], [ClassID], [Average], [Status]) VALUES (N'HE00006', N'SP22BCS
D', 6.195, N'PASSED')
INSERT [dbo].[View] ([Mssv], [ClassID], [Average], [Status]) VALUES (N'HE00006', N'SP22BIA
O', 6.245, N'PASSED')
INSERT [dbo].[View] ([Mssv], [ClassID], [Average], [Status]) VALUES (N'HE00006', N'SP22CLA
B', 450, N'PASSED')
INSERT [dbo].[View] ([Mssv], [ClassID], [Average], [Status]) VALUES (N'HE00006', N'SP22FJP
D', 5.735, N'PASSED')
INSERT [dbo].[View] ([Mssv], [ClassID], [Average], [Status]) VALUES (N'HE00006', N'SU21HPR
F', 7.555, N'PASSED')
INSERT [dbo].[View] ([Mssv], [ClassID], [Average], [Status]) VALUES (N'HE00006', N'SU21IMA
E', 7.0250000000000012, N'PASSED')
```

IX. ***Index, View, Trigger và Stored Procedure***

1. **Index:**

```
CREATE INDEX Stu_Name ON Students([Last Name], [First Name])

CREATE INDEX Lec_Name ON Lecturers([Last Name], [First Name])

SELECT * FROM Students WHERE [Last Name] = N'Nguyễn Văn' AND [First Name] = 'A'
```

- Chúng ta hầu như không thể nhớ được hết mã số sinh viên hoặc mã số của giảng viên. Do vậy, hầu hết các thao tác tìm kiếm, sắp xếp đều sẽ tìm theo tên là chủ yếu.
- Tuy nhiên, khi tìm theo tên, dữ liệu của chúng ta lại gồm 2 phần là `Last Name` và `First Name`, có nghĩa là mỗi khi đi tìm 1 sinh viên hoặc 1 giảng viên bằng cách `WHERE` 2 cột này lại với giá trị cần tìm, database sẽ phải lục hết các record, mỗi record lại so sánh hết các chuỗi với tên nhập vào

  → Rất tốn thời gian và performance.
- Do vậy, tạo hẳn 1 index cho 2 cột này ở 2 bảng `Students` và `Lecturers` thì khi nhập tên vào tìm kiếm, database sẽ chỉ cần đi dùng index để tìm ra luôn record sử dụng thuật toán B-Tree.

2. **View:**

```
CREATE VIEW View_Assess_AssessSystem AS
SELECT Classes.ClassID, Enroll.GrID, AsnID, Semester, [Start Date], [End Date], Asses
s.AsID, [AS].CouID, [AS].CatID, [Number of Questions], Duration, [Weight]
FROM
  Classes INNER JOIN Enroll ON Classes.ClassID = Enroll.ClassID
      INNER JOIN Assess ON Classes.ClassID = Assess.ClassID
      INNER JOIN [Assessment System] AS [AS] ON [AS].AsID = Assess.AsID

SELECT * FROM View_Assess_AssessSystem
```

- Có thể thấy ở Module Diagram, có 4 bảng rất quan trọng và có thể coi là trung tâm liên kết rát nhiều bảng lại với nhau, đó là bảng `Classes`, `Enroll`, `Assess` và

`Assessment System`.

- Từ `Classes` có thể dùng phép `JOIN` với bảng `Assignment` để lấy được ID của giảng viên và môn học, `JOIN` thêm với 2 bảng `Lecturers` và `Courses` để lấy được tên và các dữ liệu chi tiết hơn.
- Từ bảng `Enroll` ta lấy được ID của nhóm sinh viên, `JOIN` với bảng `Join`, `Students`, và `Groups` để lấy ra được các thông tin chi tiết về các nhóm sinh viên và các sinh viên.
- Từ bảng `Assessment System`, lấy được các thông tin về việc đánh giá môn học, đầu điểm và cả bảng `Grade` là bảng dùng để chấm điểm.
- Do các bảng này rất hay cần `JOIN` chung để lấy được các thông tin cần thiết, nên ta tạo luôn 1 bảng `View` cho `Classess JOIN Enroll JOIN Assess JOIN Assessment System`.

3. **Trigger:**

```
CREATE TRIGGER View_Average ON [View]
AFTER INSERT, UPDATE
AS
  DECLARE @AVG FLOAT;
  DECLARE @CLASS VARCHAR(50);
  DECLARE @STU VARCHAR(50);
  DECLARE @AVERAGE FLOAT;
  DECLARE @STA VARCHAR(50);
  SELECT @STU = Mssv, @CLASS = ClassID, @AVG = Average, @STA = [Status] FROM inserte
d;

  SELECT @AVERAGE = SUM(Score * [Weight])
  FROM
    [Join] AS J INNER JOIN Enroll ON J.GrID = Enroll.GrID
          INNER JOIN [dbo].View_Assess_AssessSystem AS VAA ON VAA.ClassID = Enroll.Cl
assID AND VAA.GrID = J.GrID
          INNER JOIN Grade ON J.Mssv = Grade.Mssv AND Grade.AsID = VAA.AsID
  WHERE J.Mssv = @STU AND VAA.ClassID = @CLASS
  GROUP BY J.Mssv, VAA.ClassID, VAA.CouID, VAA.GrID
  IF @AVG <> @AVERAGE
  BEGIN
    PRINT 'Average score added to [View] was not corresponding to the average scored
 counted from [Grade]'
    ROLLBACK TRAN
  END
```

```
  ELSE IF (NOT @STA = 'PASSED') AND (NOT @STA = 'NOT PASSED')
  BEGIN
    PRINT 'Status must be passed or not passed'
    ROLLBACK TRAN
  END
  ELSE IF (@AVG <= 4 AND @STA = 'PASSED') OR (@AVG > 4 AND @STA = 'NOT PASSED')
  BEGIN
    PRINT 'Incorrect Status'
    ROLLBACK TRAN
  END

  UPDATE [View] SET Average = 8.645, [Status] = 'ASDFK' WHERE Mssv = 'HE00001' AND Cl
assID = 'FA21APRO'

  SELECT * FROM [View]
```

- Ở bảng `View` có 1 cột là `Average`. Cột này được tính dựa trên cột `Score` của bảng `Grade`.
- Để có thể chắc chắn rằng người quản lý sẽ update hoặc add dữ liệu đúng vào cột `Average` này, ta đặt 1 trigger vào cột `Average`.
- Bất cứ khi nào người quản lý database add 1 dữ liệu mà không trùng khớp với dữ liệu tính được ở bảng `Grade`, một thông báo sẽ được xuất hiện và transaction đó sẽ không được thực hiện commit vào database.

```
  CREATE TRIGGER AssSys_Weight ON [Assessment System]
  AFTER INSERT, UPDATE
  AS
    DECLARE @ID VARCHAR(50);
    DECLARE @AsID VARCHAR(50);
    DECLARE @WEIGHT FLOAT;

    SELECT @ID = AsID, @WEIGHT = [Weight] FROM inserted
    SELECT @AsID = AsID  FROM [Assessment System]

    IF @WEIGHT <= 0 OR @WEIGHT > 1
    BEGIN
      PRINT 'The weight of an Assessment System can be neither less than or equal to
 0 nor greater than 1'
      ROLLBACK TRAN
    END

    UPDATE [Assessment System] SET Weight = 0 WHERE AsID =  'CEAAS1'
```

- Ở bảng `Assessment System`, cột dữ liệu ở cột `Weight` bắt buộc phải nằm trong nửa đoạn (0,1]
- Do vậy, để tránh việc người quản lý add hoặc update nhầm dữ liệu vào cột `Weight`, ta cũng đặt 1 trigger vào nó.
- Mỗi khi người quản lý add 1 số nhỏ hơn hoặc bằng 0 hoặc 1 số lớn hơn 1 thì sẽ có 1 thông bóa hiện lên và transaction đó cũng sẽ không được commit lên hệ thống.

4. **Stored Procedure:**

```
CREATE PROCEDURE Count_Student
  @GR VARCHAR(50),
  @COUNT INT OUT
AS
BEGIN
  SELECT @COUNT = COUNT(Mssv) FROM [Join] WHERE GrID = @GR
END

DECLARE @NumOfStu INT = 0;
EXEC Count_Student @GR = 'IA1604', @COUNT = @NumOfStu OUT
PRINT @NumOfStu;
```

- Khi add hoặc update dữ liệu vào bảng Groups, trong cột Number of Students, dữ liệu cột này phải trùng khớp với dữ liệu tính toán từ bảng Join.
- Khi đó, tạo sẵn 1 procedure để tính toán số lượng học sinh này thì rất tiện lợi.

X. ***Queries:***

1. **Query dùng ORDER BY:**

```
SELECT * FROM Students ORDER BY [First Name], [Last Name]
```

- Query này sẽ cho chúng ta xem được toàn bộ sinh viên đã đăng kí vào trường.

2. **Query dùng INNER JOIN:**

```
SELECT Students.Mssv, [Last Name], [First Name], Classes.ClassID, AsnID, Semester, [S
tart Date], [End Date]
FROM
  Students INNER JOIN [Join] ON Students.Mssv = [Join].Mssv
       INNER JOIN Enroll ON Enroll.GrID = [Join].GrID
       INNER JOIN Classes ON Enroll.ClassID = Classes.ClassID
```

- Query này sẽ cho ra 1 bảng gồm 8 cột `Mssv`, `Last Name`, `First Name`, `ClassID`, `AsnID`, `Semester`, `Start Date`, `End Date` chính là sinh viên và lớp học tham gia ở các kì.
- Đầu mỗi kì học, sinh viên FPT đều nhận được email từ nhà trường báo về các lớp học có trong kì này. Sử dụng bảng output của query này, việc gửi mail có thể trở nên dễ dàng hơn.

3. **Query dùng Aggregate function:**

```
SELECT Lecturers.LecID, [Last Name], [First Name], Semester, COUNT(Assignment.AsnID)
 AS Assigned
FROM
  Lecturers INNER JOIN Assignment ON Lecturers.LecID = Assignment.Lecturer
```

```
        INNER JOIN Classes ON Assignment.AsnID = Classes.AsnID
GROUP BY Lecturers.LecID, [Last Name], [First Name], Semester
```

- Query này sẽ cho ra 1 bảng gồm 5 cột, `LecID`, `Last Name`, `First Name`, `Semester` và `Assigned` chính là ID là họ tên của giảng viên và lịch phân công của họ.
- Sử dụng bảng ouput của query này, ta có thể xem được lịch phân công theo kì của mỗi giảng viên, từ đó viết mail cho giảng viên để thông báo lịch trình.
- Ngoài ra, cũng có thể quản lý được toàn bộ giảng viên có lịch giảng dạy trong 1 kì học.

4. **Query dùng GROUP BY và HAVING:**

```
SELECT LecID, [Last Name], [First Name], Semester, Enroll.GrID, Enroll.ClassID, COUNT
(Mssv) AS [Num of Students]
FROM
  Lecturers INNER JOIN Assignment ON Lecturers.LecID = Assignment.Lecturer
        INNER JOIN Classes ON Classes.AsnID = Assignment.AsnID
        INNER JOIN Enroll ON Classes.ClassID = Enroll.ClassID
        INNER JOIN [Join] ON Enroll.GrID = [Join].GrID
GROUP BY LecID, [Last Name], [First Name], Semester, Enroll.GrID, Enroll.ClassID
HAVING LecID = 'GVA'
```

- Query này sẽ cho ra 1 bảng gồm 7 cột `LecID`, `Last Name`, `First Name`, `Semester`, `GrID`, `ClassID`, `Num of Students` chính là các thông tin về kì học, nhóm sinh viên, lớp học và số sinh viên theo từng kì mà giảng viên có ID GVA tham gia giảng dạy.
- Thay GVA bằng các giảng viên khác, chúng ta có thể xem được các thông tin kể trên.
- Có thể ứng dụng trong việc xem xét tăng lương hoặc phong trưởng bộ môn dựa vào kinh nghiệm (nhiều kì giảng dạy).

5. **Query sử dụng sub-query như 1 quan hệ:**

```
SELECT LEC.LecID, LEC.[Last Name], LEC.[First Name], Major, Semester, CouID, [AS].Cat
ID
FROM
  (SELECT LecID,[Last Name], [First Name] FROM Lecturers) AS LEC
    INNER JOIN
      Assignment
      ON Assignment.Lecturer = LecID
    INNER JOIN
      (SELECT ClassID, AsnID, Semester FROM Classes) AS CLASS
      ON Assignment.AsnID = CLASS.AsnID
    INNER JOIN
      Assess
      ON Assess.ClassID = CLASS.ClassID
    INNER JOIN
      (SELECT AsID, CouID, CatID FROM [Assessment System]) AS [AS]
      ON Assess.AsID = [AS].AsID
    INNER JOIN
      Categories
      ON [AS].CatID = Categories.CatID
ORDER BY LEC.LecID, Major, Semester, CatID
```

- Query này sẽ cho ra 1 bảng gồm 7 cột `LecID`, `Last Name`, `First Name`, `Major`, `Semester`, `CouID`, `CatID` chính là những hệ thống đánh giá theo môn học tương ứng và kì học tương ứng.
- Sử dụng bảng output ở query này, ta có thể giám sát tiến độ đánh giá các môn học của từng giảng viên.

6. **Query sử dụng sub-query ở mệnh đề WHERE:**

```
SELECT J.Mssv, J.GrID, Enroll.ClassID, CONCAT([Last Name], ' ', [First Name]) AS Lect
ure, CouID, CatID, Score, [Date]
FROM
  [Join] AS J INNER JOIN Enroll ON J.GrID = Enroll.GrID
        INNER JOIN View_Assess_AssessSystem AS VAA ON Enroll.ClassID = VAA.ClassID AN
D Enroll.GrID = VAA.GrID
        INNER JOIN Assignment ON Assignment.AsnID = VAA.AsnID
```

```
        INNER JOIN Lecturers ON Assignment.Lecturer = Lecturers.LecID
        INNER JOIN Grade ON Grade.Mssv = J.Mssv AND Grade.AsID = VAA.AsID
WHERE J.Mssv = (SELECT Mssv FROM Students WHERE [First Name] = 'A' AND [Last Name] =
 N'Nguyễn Văn')
```

- Query này sẽ cho ra 1 bảng gồm 8 cột `Mssv`, `GrID`, `ClassID`, `Lecture`, `CouID`, `CatID`, `Score`, `Date` chính là những điểm số mà các giảng viên đã cho ứng với sinh viên Nguyễn Văn A theo từng kì và từng môn, từng đầu điểm.
- Thay tên của sinh viên khác, ta sẽ có thể thu được các điểm số mà sinh viên đã nhận được qua từng kì học.

7. **Query sử dụng partial matching trong mệnh đề WHERE:**

```
SELECT J.Mssv, CONCAT([Last Name], ' ', [First Name]) AS [Student Name], ClassID, Ave
rage, [Status]
FROM
  Students INNER JOIN [Join] AS J ON Students.Mssv = J.Mssv
       INNER JOIN [View] AS V ON J.Mssv = V.Mssv
WHERE ClassID LIKE 'FA21%'
```

- Query này sẽ cho ra 1 bảng gồm 5 cột `Mssv`, `Student Name`, `ClassID`, `Average`, `Status` chính là điểm trung bình môn học và trạng thái của sinh viên trong kì học Fall 2021.
- Sử dụng bảng này, ta có thể lấy được trạng thái và điểm trung bình của sinh viên với kì học cần thiết tương ứng.

8. **Query sử dụng self-join:**

```
SELECT Lecturers.LecID, Lecturers.Email, CONCAT(Lecturers.[Last Name], ' ', Lecturer
s.[First Name]) AS Lecture, CONCAT(LEADER.[Last Name], ' ', LEADER.[First Name]) AS
```

```
 [Leader]
FROM Lecturers LEFT JOIN Lecturers AS [LEADER] ON Lecturers.Report = LEADER.LecID
```

- Query này sẽ cho ra 1 bảng gồm 4 cột `LecID`, `Email`, `Lecture`, `Leader` chính là giảng viên và người leader mà giảng viên đó phải tuân theo.
- Sử dụng bảng này, người quản lý sẽ bàn giao công việc cho các leader, người leader có thể thấy được nhân sự dưới trướng của mình và giao việc xuống nữa.

9. **Query ngẫu nhiên 1:**

```
SELECT Lecturers.LecID, Lecturers.Email, CONCAT(Lecturers.[Last Name], ' ', Lecturer
s.[First Name]) AS Lecture, Report, Major, Courses.[Name], Semester, Students.Mssv, C
ONCAT(Students.[Last Name], ' ', Students.[First Name]) AS [Student Name], Students.E
mail, Students.Gender
FROM
  Lecturers INNER JOIN Assignment ON Lecturers.LecID = Assignment.Lecturer
        INNER JOIN Courses ON Courses.CouID = Assignment.Major
        INNER JOIN Classes ON Classes.AsnID = Assignment.AsnID
        INNER JOIN Enroll ON Classes.ClassID = Enroll.ClassID
        INNER JOIN [Join] ON [Join].GrID = Enroll.GrID
        INNER JOIN Students ON Students.Mssv = [Join].Mssv
ORDER BY Lecturers.LecID, Semester, Students.Mssv, Major
```

- Query này sẽ cho ra 1 bảng gồm 11 cột `LecID`, `Email`, `Lecture`, `Report`, `Major`, `Name`, `Semester`, `Mssv`, `Student Name`, `Email`, `Gender` chính là danh sách các giảng viên và sinh viên tham gia học giảng viên đó với kì học và môn học tương ứng.
- Sử dụng bảng này, có thể chèn thêm các mệnh đề `WHERE` để lấy được danh sách sinh viên theo học 1 giảng viên nào đó tại 1 kì nào đó.

10. **Query ngẫu nhiên 2:**

```
SELECT Lecturers.LecID, Lecturers.Email AS [Lecturer Email], CONCAT(Lecturers.[Last N
ame], ' ', Lecturers.[First Name]) AS Lecture, J.GrID, CouID, J.Mssv, CONCAT(Student
s.[Last Name], ' ', Students.[First Name]) AS [Student Name], CatID, Score, [Date]
FROM
  Lecturers INNER JOIN Assignment ON Lecturers.LecID = Assignment.Lecturer
        INNER JOIN View_Assess_AssessSystem AS VAA ON VAA.AsnID = Assignment.AsnID
        INNER JOIN Enroll ON Enroll.ClassID = VAA.ClassID
        INNER JOIN [Join] AS J ON J.GrID = Enroll.GrID
        INNER JOIN Students ON Students.Mssv = J.Mssv
        INNER JOIN Grade ON J.Mssv = Grade.Mssv AND VAA.AsID = Grade.AsID
```

- Query này sẽ cho ra 1 bảng gồm 10 cột `LecID`, `Lecturer Email`, `Lecture`, `GrID`, `CouID`, `Mssv`, `Student Name`, `CatID`, `Score`, `Date` chính là danh sách điểm của sinh viên và giảng viên đã cho điểm đó.
- Sử dụng bảng này, ta tính được điểm trung bình, từ đó biết được sinh viên pass môn hay chưa.

Lời kết:

Qua các phân tích sơ lược, phân tích chi tiết, tạo database và query ứng dụng, ta có thể thấy được rằng database này vẫn còn sơ sài và áp dụng được vào rất ít hệ thống.

Cảm ơn đã theo dõi đến đây.

Ba Vì, ngày 17 tháng 07 năm 2022

Nguyễn Doanh Thịnh