CLASS: SE1814

NAME: Trần Nam Trường

MSSV: HE170291

**REPORT ASSIGNMENT**

**a.**

- When looking at the Assignment, I see that the required entity is "Student", then the "Subject" to contain the subjects.

- In addition, the fap also has a "Curriculum" to contain all the subjects of a student

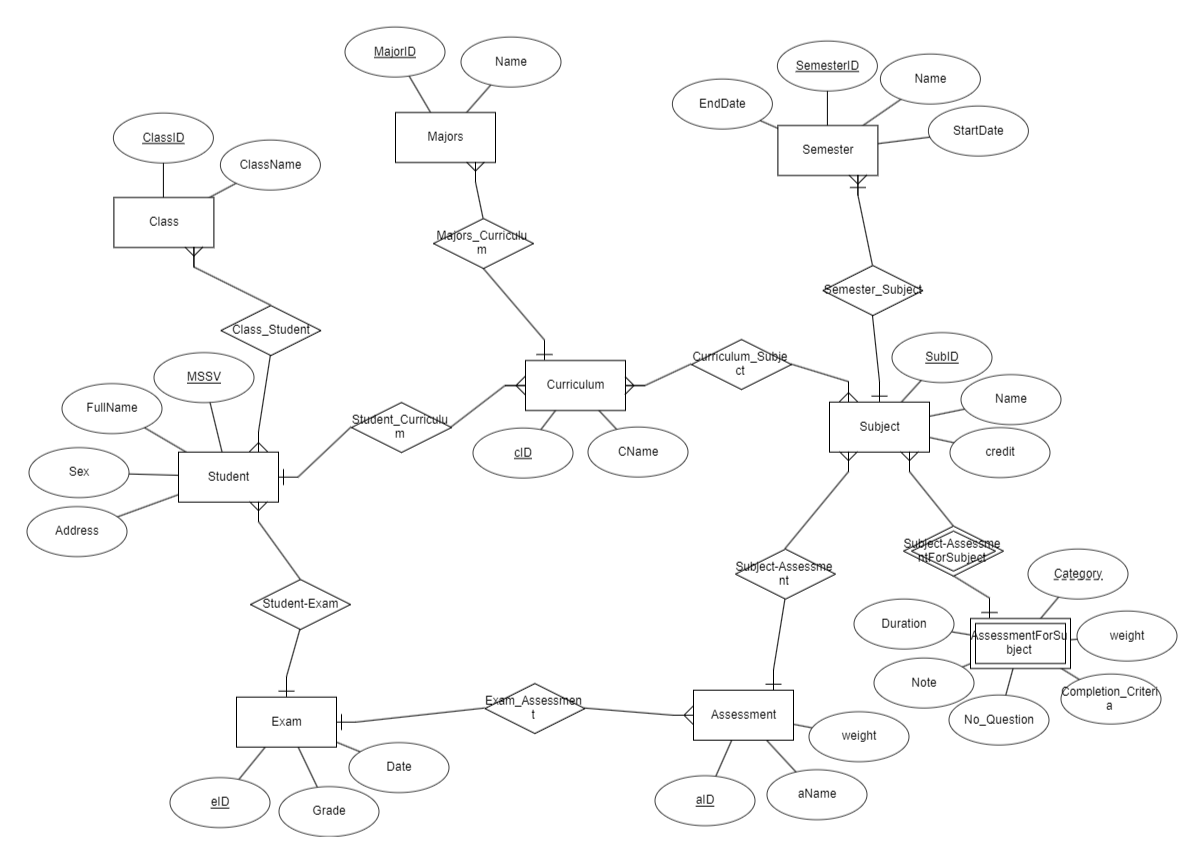
- Then comes “Semester”

- Because each subject will have an assessment requirement, we need an "Assessment" entity.

- To know the score of a certain subject of a student, we need an entity "Exam"

- Besides, we need to add 2 entities "Class" and "Majors"

**b.**



**Explanation**: ER Modeling diagram includes 8 entities and related attributes:

+ Student: MSSV, FullName, Sex, Address

+ Subject: SubID, Name, credit

+ Assessment: aID, aName, weight

+ Exam: eID, Date, Grade

+ Semester: SemesterID, Name, Date

+ Class: ClassID, ClassName

+ Curriculum: cID, cName

+ Majors: MajorID, Name

+ AssessmentForSubject: Category, weight, Completion\_Criteria, Duration, No\_Question, Note

- The relationship between Student and Class is N-N: 1 student studies in many class and 1 class will have many students.

- Semester – Subject is 1-N: 1 subject is only studied in 1 semester and 1 semester has many subjects

- Subject – Assessment 1-N relationship: 1 subject has many Assessments and 1 Assessment is for 1 subject

- Assessment – Exam relationship is 1-N: 1 Assessment for 1 exam but 1 exam needs many Assessments

- Student – Exam relationship 1-N: 1 student has many points but points are only given to 1 student

- Curriculum – Majors relationship 1-N: 1 curriculum is for 1 major but 1 major can have many curriculum

- Curriculum – Subject N-N relationship: 1 curriculum has many subjects and a subject also exists in many curriculum

- Curriculum – Student relationship 1-N: 1 student has 1 curriculum, and the curriculum can be for many students

- Subject – AssessmentForSubject relationship 1-N: 1 subject has many Assessments and 1 Assessment is for 1 subject

**c.**

**Relational Mapping:**

Class( ClassID, ClassName)

Majors: MajorID, Name

Class\_Student(ClassID, MSSV)

Semester( SemesterID, Name, Date)

Student( MSSV, FullName, Sex, Address, cID

Curriculum: cID, cName, MajorID

Curriculum\_ Subject(cID, SubID)

Subject( SubID, Name, credit, SemesterID

Assessment( aID, aName, weight, SubID

Exam( eID, Date, Grade, MSSV, aID

AssessmentForSubject(Category, SubID, weight, Completion\_Criteria, Duration, No\_Question, Note)

**d.**

create table Class(

ClassID varchar(30) primary key,

ClassName nvarchar(200)

)

create table Majors(

MajorID varchar(30) primary key,

[Name] nvarchar(200)

)

create table Semester(

SemesterID int primary key,

SemesterName nvarchar(200),

StartDate date,

EndDate date

)

create table Curriculum(

cID int primary key,

cName nvarchar(200),

MajorID varchar(30) foreign key references Majors(MajorID)

)

create table Student(

MSSV varchar(50) primary key,

FullName Nvarchar(200),

Sex bit,

CurriculumID int foreign key references Curriculum(cID)

)

create table Class\_Student(

StudentID varchar(50) foreign key references Student(MSSV),

ClassID varchar(30) foreign key references Class(ClassID),

primary key (StudentID, ClassID)

)

Create table [Subject](

SubID varchar(50) primary key,

Name varchar(50),

credit int,

SemesterID int foreign key references Semester(SemesterID),

Subject\_Prequisite varchar(50) foreign key references [Subject](SubID)

)

create table AssessmentForSubject(

SubID varchar(50) foreign key references [Subject](SubID),

Category varchar(50),

[weight] varchar(10),

Completion\_Criteria varchar(10),

Duration varchar(10),

No\_Question varchar(10),

Note nvarchar(500),

primary key (SubID, Category)

)

create table Curriculum\_Details(

CurriculumID int foreign key references Curriculum(cID),

SubID varchar(50) foreign key references [Subject](SubID),

primary key(CurriculumID, SubID)

)

create table assessment(

aID int,

aname varchar(50),

SubID varchar(50) foreign key references [Subject](SubID),

[weight] float,

primary key(aID)

)

create table Exam(

eid int primary key,

MSSV varchar(50) foreign key references Student(MSSV),

aID int foreign key references assessment(aID),

[dateTime] date,

Grade int,

)

**e.**

1. A query that uses ORDER BY

-- get grade for each Student each subject (use ORDER BY)

Select s.MSSV, s.FullName, sub.SubID, Sum(Exam.Grade \* a.weight) as Grase from

(select e.\* from (select e.MSSV, e.aID, max(e.dateTime) as datetime from Exam e

group by e.MSSV, e.aID) Exam , Exam e

where Exam.MSSV = e.MSSV and Exam.aID = e.aID and Exam.datetime = e.dateTime) Exam

join Student s on Exam.MSSV = s.MSSV

join assessment a on Exam.aID = a.aID

join Subject sub on sub.SubID = a.SubID

group by s.MSSV, s.FullName, sub.SubID

order by s.MSSV

2. A query that uses INNER JOINS

-- get curriculum for all student (use INNER JOIN)

select s.MSSV, s.FullName, c.cID as CurriculumID, c.cName as CurriculumName

, sub.SubID, sub.Name, sub.SemesterID from Student s

inner join Curriculum c on s.CurriculumID = c.cID

join Curriculum\_Details cd on cd.CurriculumID = c.cID

join [Subject] sub on sub.SubID = cd.SubID

3. A query that uses aggregate functions

-- get GPA for each Student on all subject (use aggregate functions)

Select tbl.MSSV, tbl.FullName, SUM(tbl.Grade \* tbl.credit) / (select sum(s.credit) from Subject s) AS GPA

from

(Select s.MSSV, s.FullName, sub.SubID, Sum(Exam.Grade \* a.weight) as Grade, sub.credit from

(select e.\* from (select e.MSSV, e.aID, max(e.dateTime) as datetime from Exam e

group by e.MSSV, e.aID) Exam , Exam e

where Exam.MSSV = e.MSSV and Exam.aID = e.aID and Exam.datetime = e.dateTime)Exam

join Student s on Exam.MSSV = s.MSSV

join assessment a on Exam.aID = a.aID

join Subject sub on sub.SubID = a.SubID

group by s.MSSV, s.FullName, sub.SubID, sub.credit)tbl

group by tbl.MSSV, tbl.FullName

order by tbl.MSSV

4. A query that uses the GROUP BY and HAVING clauses

-- get grade of Student for subject 'PRF' have grade >= 8

select s.MSSV, s.FullName, a.SubID as [Subject], sum(e.Grade \* a.weight) as Grade from Student s

join Exam e on s.MSSV = e.MSSV

join assessment a on a.aID = e.aID

where a.SubID = 'PRF'

group by s.MSSV, s.FullName, a.SubID

having sum(e.Grade \* a.weight) >= 8

5. A query that uses a sub-query as a relation

--get avg grade for class in 1 subject

select tbl.ClassID, avg(tbl.Grase) from

(Select cs.ClassID, s.MSSV, s.FullName, sub.SubID, Sum(Exam.Grade \* a.weight) as Grase from

(select e.\* from (select e.MSSV, e.aID, max(e.dateTime) as datetime from Exam e

group by e.MSSV, e.aID) Exam , Exam e

where Exam.MSSV = e.MSSV and Exam.aID = e.aID and Exam.datetime = e.dateTime) Exam

join Student s on Exam.MSSV = s.MSSV

join assessment a on Exam.aID = a.aID

join Subject sub on sub.SubID = a.SubID

join Class\_Student cs on s.MSSV = cs.StudentID

where cs.ClassID = 'SE001' and sub.SubID = 'PRF'

group by cs.ClassID, s.MSSV, s.FullName, sub.SubID)tbl

group by tbl.ClassID

6. A query that uses a sub-query in the WHERE clause

-- get subjects in major

select c.MajorID, cd.CurriculumID, s.SubID, s.Name from Curriculum c

join Curriculum\_Details cd

on c.cID = cd.CurriculumID

join Subject s on s.SubID = cd.SubID

where c.MajorID = (select m.MajorID from Majors m where m.MajorID like 'HE')

7. A query that uses partial matching in the WHERE clause

--get fullname student contain 'Van'

select \* from Student s

where s.FullName like '%Van%'

8. A query that uses a self-JOIN

-- get subject have Subject Prerequisite

select s2.SubID,s2.Name, s2.SubjectPrequisite, s1.Name from Subject s1

join Subject s2 on s1.SubID = s2.SubjectPrequisite

9. A query that uses Procedure

--create proceder to count number of student have fullname contain 'Van'

create proc p1

@mssv varchar(30), @numberOfStudent int output

as

begin

set @numberOfStudent = (select count(s.MSSV) from Student s

where s.FullName like '%'+@mssv+'%')

end

declare @t int

exec p1'Van', @t output

print @t

10. A query that uses Trigger

--create trigger insert Student

create trigger insertStudent

on Student

after insert

as

begin

select \* from inserted s

join Curriculum c on s.CurriculumID = c.cID

end

insert into Student(MSSV, FullName, Sex, CurriculumID)

values ('SI003', 'Nguyen Hoang Long', 1, 2)