Student Grading Management Sub-System

Final Report

Name: Lê Tuấn Đức

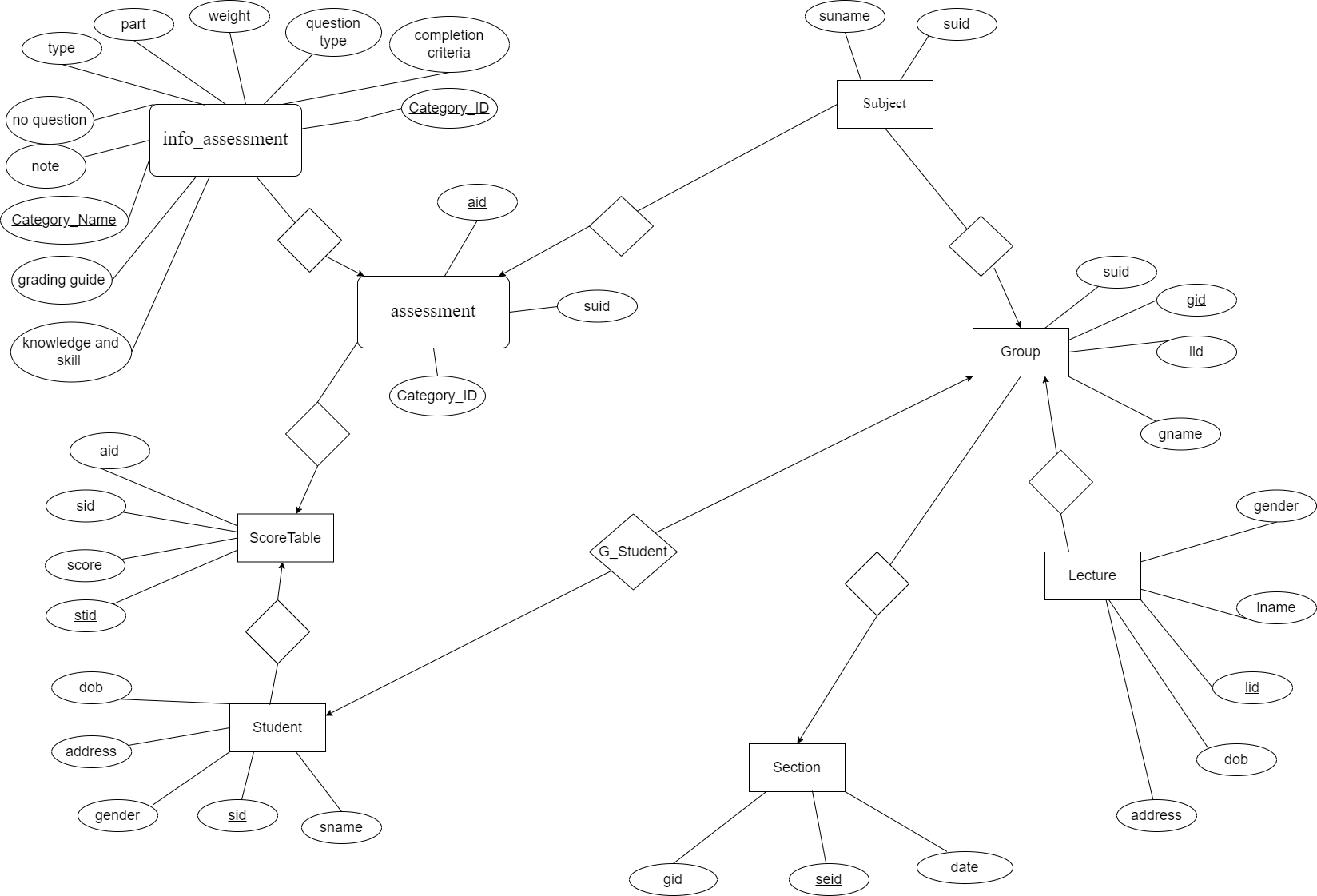
ID: HE163500

Class: SE1647

1. **A brief description of the database**
2. Each subject has many assessments but each assessment only for one subject so relationship between them are one-to-many. This help we know how many assessments on each subject.
3. Table Info\_Assessments give more info in each assessment
4. Each assessment has many scoretable because student can retake they assessment. Each scoretable have score for one assessment. So they relationship between them are one-to-many. This help we know student test how many time on each assessment and them score.
5. Each student has many scoretable but each scoretable only for one student so relationship between them are one-to-many. This help we know how many score each student has.
6. Each student belong to many group and each group have many student so relationship between them are many-to-many. Table G\_Student are created from 2 table, G\_student help we know in one group have how many student and one student belong to how many group.
7. Each group can learn only one subject but each subject can be learned by many group so they relationship between them are one-to-many.
8. Each group can be managed by one lecture but each lecture can manage many group so they relationship between them are one-to-many.
9. Each group can learn many section but each section can be learned by one group so they relationship between them are one-to-many.

* Student can check student results at the end of semester by inner join table student, subject, group
* Each Subject code, student can check their detailed result by inner join table student, result, scoretable, assessment, subject.

1. ERD



1. Data

CREATE TABLE [Lecture](

[lid] [int] PRIMARY KEY NOT NULL,

[lname] [nvarchar](50) NOT NULL,

[gender] [bit] NOT NULL,

[dob] [datetime] NOT NULL,

[address] [nvarchar](50) NOT NULL

)

CREATE TABLE [Subject](

[suid] [int] PRIMARY KEY NOT NULL,

[suname] [nvarchar](50) NOT NULL

)

CREATE TABLE [Student](

[sid] [int] PRIMARY KEY NOT NULL,

[sname] [nvarchar](50) NOT NULL,

[gender] [bit] NOT NULL,

[address] [nvarchar](50) NOT NULL,

[dob] [datetime] NOT NULL

)

CREATE TABLE [Group](

[gid] [int] PRIMARY KEY NOT NULL,

[gname] [nvarchar](50) NOT NULL,

[lid] [int] NOT NULL,

[suid] [int] NOT NULL,

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

FOREIGN KEY (suid) REFERENCES [Subject] (suid)

)

CREATE TABLE [Section](

[seid] [int] PRIMARY KEY NOT NULL,

[date] [datetime] NOT NULL,

[gid] [int] NOT NULL,

FOREIGN KEY (gid) REFERENCES [Group] (gid)

)

CREATE TABLE [G\_Student](

[sid] [int] NOT NULL,

[gid] [int] NOT NULL,

PRIMARY KEY (sid, gid),

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

FOREIGN KEY (gid) REFERENCES [Group] (gid)

)

CREATE TABLE [Info\_Assessment](

[Category\_ID] [int] PRIMARY KEY NOT NULL,

[Category\_name] [nvarchar](50) NOT NULL,

[Type] [nvarchar](50) NOT NULL,

[Part] [int] NOT NULL,

[Weight] [float] NOT NULL,

[Completion criteria] [nvarchar](50) NOT NULL,

[Duration] [nvarchar](50) NOT NULL,

[Question type] [nvarchar](50) NOT NULL,

[No question] [int] NOT NULL,

[Knowledge and Skill] [nvarchar](150) NOT NULL,

[Grading guide] [nvarchar](150) NOT NULL,

[Note] [nvarchar](150) NOT NULL

)

CREATE TABLE [Assessment](

[aid] [int] PRIMARY KEY NOT NULL,

[suid] [int] NOT NULL,

[Category\_ID] [int] NOT NULL,

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

FOREIGN KEY (Category\_ID) REFERENCES [Info\_Assessment] (Category\_ID)

)

CREATE TABLE [ScoreTable](

[stid] [int] PRIMARY KEY NOT NULL,

[aid] [int] NOT NULL,

[sid] [int] NOT NULL,

[score] [float] NOT NULL,

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

FOREIGN KEY (aid) REFERENCES [Assessment] (aid)

)

* 20 Student
* 5 Group
* 5 Lecture
* 100 Section
* 5 Subject
* 500 scores ( 25 scores/ student)
* 25 assessment ( 5 assessment/ subject)

1. Query

-- CHECK pass or not pass

SELECT s.sid,s.sname AS [Student Name], su.suname AS [Subject Name],SUM(st.score \* ia.Weight) AS[AVG],

CASE

WHEN SUM(st.score \* ia.Weight) < 5 THEN 'NOTPASS'

ELSE 'PASS'

END AS [STATUS]

FROM Student s INNER JOIN ScoreTable st ON s.sid = st.sid

INNER JOIN Assessment a ON st.aid = a.aid

INNER JOIN Subject su ON su.suid = a.suid

INNER JOIN Info\_Assessment ia ON ia.Category\_ID = a.Category\_ID

GROUP BY s.sid, s.sname,su.suname

1. .A query that uses ORDER BY to sort section by group id

SELECT \* FROM Section

ORDER BY gid ASC

1. .A query that uses INNER JOINS to check which teachers Student studied

SELECT s.sname, L.lname

FROM Student s INNER JOIN G\_Student gs ON s.sid = gs.sid

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

INNER JOIN Lecture l ON l.lid = g.lid

1. A query that uses aggregate functions to count how many section lecture teached

SELECT l.lname, COUNT (se.seid) AS NumberSectionTeached

FROM Lecture l INNER JOIN [Group] g ON g.gid = l.lid

INNER JOIN Section se ON se.gid = g.lid

GROUP BY l.lname

1. A query that uses the GROUP BY and HAVING clauses to check lecture teaches more than 10 section

SELECT l.lname, COUNT (se.seid) AS NumberSectionTeached

FROM Lecture l INNER JOIN [Group] g ON g.gid = l.lid

INNER JOIN Section se ON se.gid = g.lid

GROUP BY l.lname

HAVING COUNT (se.seid) > 10

1. A query that uses a sub-query as a relation to check 10 student have Lowest avg

SELECT TOP 10 tb1.[Student Name], tb1.AVG, tb1.[Subject Name]

FROM

(SELECT s.sid,s.sname AS [Student Name], su.suname AS [Subject Name],SUM(st.score \* ia.Weight) AS[AVG]

FROM Student s INNER JOIN ScoreTable st ON s.sid = st.sid

INNER JOIN Assessment a ON st.aid = a.aid

INNER JOIN Subject su ON su.suid = a.suid

INNER JOIN Info\_Assessment ia ON ia.Category\_ID = a.Category\_ID

GROUP BY s.sid, s.sname,su.suname) tb1

ORDER BY tb1.AVG

1. A query that uses a sub-query in the WHERE clause check 5 student have Lowest avg on subject dbi

SELECT TOP 5 tb1.[Student Name], tb1.AVG, tb1.[Subject Name]

FROM

(SELECT s.sid,s.sname AS [Student Name], su.suname AS [Subject Name],SUM(st.score \* ia.Weight) AS[AVG]

FROM Student s INNER JOIN ScoreTable st ON s.sid = st.sid

INNER JOIN Assessment a ON st.aid = a.aid

INNER JOIN Subject su ON su.suid = a.suid

INNER JOIN Info\_Assessment ia ON ia.Category\_ID = a.Category\_ID

WHERE su.suname = 'DBI'

GROUP BY s.sid, s.sname,su.suname) tb1

ORDER BY tb1.AVG

1. A query that uses partial matching in the WHERE clause to find student have name start with letters M

SELECT \*

FROM Student s

WHERE s.sname LIKE 'M%'

1. Check avg of student

SELECT s.sid,s.sname AS [Student Name], su.suname AS [Subject Name],SUM(st.score \* ia.Weight) AS[AVG]

FROM Student s INNER JOIN ScoreTable st ON s.sid = st.sid

INNER JOIN Assessment a ON st.aid = a.aid

INNER JOIN Subject su ON su.suid = a.suid

INNER JOIN Info\_Assessment ia ON ia.Category\_ID = a.Category\_ID

GROUP BY s.sid, s.sname,su.suname

ORDER BY s.sid

1. Check name student not enroll any group

SELECT s.sid,s.sname

FROM Student s LEFT JOIN G\_Student r ON s.sid = r.sid

LEFT JOIN [Group] g ON g.gid= r.gid

GROUP BY s.sid,s.sname

HAVING COUNT(g.gid) = 0

1. Check name student enroll all group

SELECT tb2.sname

FROM (SELECT COUNT (g.gid) AS [Total Group]

FROM [Group] g) tb1 INNER JOIN (SELECT s.sname, COUNT(g.gid) AS [Number Group]

FROM Student s LEFT JOIN G\_Student r ON s.sid = r.sid

LEFT JOIN [Group] g ON g.gid= r.gid

GROUP BY s.sid,s.sname) tb2

ON tb1.[Total Group] = tb2.[Number Group]

1. Trigger, Store Procedure, Index
2. Store Procedure : Calculate avg score of student

CREATE PROC CalculateAVG

AS

BEGIN

SELECT s.sid,s.sname AS [Student Name], su.suname AS [Subject Name],SUM(st.score \* ia.Weight) AS[AVG]

FROM Student s INNER JOIN ScoreTable st ON s.sid = st.sid

INNER JOIN Assessment a ON st.aid = a.aid

INNER JOIN Subject su ON su.suid = a.suid

INNER JOIN Info\_Assessment ia ON ia.Category\_ID = a.Category\_ID

GROUP BY s.sid, s.sname,su.suname

ORDER BY s.sid

END

1. Index

CREATE INDEX index\_score ON [ScoreTable](stid)