COURSE: COMP 5531 Files and Databases

ASSIGNMENT: Project Demo Report
LECTURE SECTION: NN
PROFESSOR: Dr. bipin c. DESAI
LAB INSTRUCTOR : yogesh YADAV
DUE: April 18, 2022

GROUP INFORMATION

Group Name: COMP 5531_group_3

Group Leader: fung sim, LEUNG, 40195538, f_eung@encs.concordia.ca
victor, NWATU, 40206992, v_nwatu@encs.concordia.ca
akshat, TYAGI, 40195161, ak_tyag@encs.concordia.ca
rim, EL OSTA, 40083858, r_elosta@encs.concordia.ca

Demo Report of COMP 5531 Group 3

By Akshat Tyagi, Sim Fung Leung, Victor Nwatu, Rim El Osta

Project Description

The admin starts by creating courses and adding instructors to teach them. Then, each instructor can add students and TAs for their course. Instructors can then assign students into groups. When students log into GCA, they will have to select a course to view from the courses that they are taking. This brings them to a discussion forum where they are able to view other students', TA's, or instructor's posts about the selected course. They can also make posts in the discussion forum, and such posts can be visible to group members, instructors, and TAs. Such discussions are mediated by the instructor. Students also have the ability to submit files related to marked entities within their groups in the selected course. A submission can be read-only or can be edited/deleted by other group members. Instructors and TAs are able to view those submitted files for grading. Students can send private messages to group members within their groups in their courses.

Assumptions with Rational Explanations

A person may have more than one role (e.g. a student in one course and a TA in another course). The person will need two different accounts, one for the studetn role and one for the TA role.

Design Decisions with Rational Explanations

All user accounts are already created with the respective roles assigned (admin, instructor, ta, and student) in the User_tbl.

Admin

The system administrator is responsible for creating courses and adding an instructor to the course. Each course has a course section, course name, course term (fall, summer, winter), course year, and course subject (COMP, ENCS…..). All these are added to a CourseSection_tbl. Course name and number must be unique, no two courses can have the same name and number. There must be an instructor for each course, and there can only be one instructor. An instructor could teach multiple courses. Instructors are added to an Instructor_tbl which takes user and course id. The admin can also add ta's and students to a particular course, just like the instructor has a Student_tbl and Ta_tbl. The user id and course id reference the User_tbl and CourseSection_tbl respectively. Each course could have multiple students and TAs. A student could also take multiple courses, a TA could also teach multiple courses. The admin can also edit courses, delete courses, change instructors, and delete TAs and students. If a course is deleted, all information related to the course such as instructor, TA, and students will also be deleted.

Instructor

For each course an instructor is responsible for, they could add and remove ta's. They can also add and remove students. The instructor is responsible for creating groups as well as assigning students to a group and selecting a group leader. Each group is connected to a course. A course could have multiple groups. Groups are added to a Group_tbl which has a course id that references CourseSection_tbl. Group names are unique. A student cannot be in two groups at a time. Once a student is added to a group, he is added to a GroupMember_tbl which has a group ID, date joined, and user ID. Group ID references group table, ensuring a student is not associated with multiple groups. If a student is to be added to another group, they must first be removed from the current group and then added to the new group. The instructor can also delete groups automatically removing all members from the group. The instructor creates marked entities (assignments, projects), which consist of a deadline, entity name, and file. Entities cannot be created if these details are not provided. Entities are added to a GroupMarked_tbl, and a course ID references the course table. A course could have multiple entities. The instructor could also view group submission to a marked entity. Each group can only submit one file, but submission could be updated up till the deadline for entities passes. A FinalSubmission_tbl stores the submission for each group and the instructor downloads and grades entities. The instructor also has access to group discussions and can view students' discussions for each group. Discussions are tied to entities, meaning for each marked entity, group members could have a discussion, sharing files and messages concerning the entity. The instructor has access to all this and could also drop suggestions or comments in the discussion. Discussions are unique to groups. For each marked entity, each group will have separate discussions. A DiscussionPagesPost_tbl is responsible for this. The instructor could also start a general discussion allowing all students regardless of groups to participate. A GeneralDiscussion_tbl is responsible for this. Both discussion tables have user_id and entity id, keeping track of students who send messages, as well as particular entity messages, is sent for. Once the entity deadline passes, the instructor can view submissions as well as a summary page with all the files shared and group members' actions in each group's discussion forum. A FileAuditHistory_tbl is responsible for this

TA

A TA can be associated with multiple courses. Ta_tbl keeps track of ta and the course he is in. The TA cannot remove or add students to a course. The Ta cannot create or edit groups, he also cannot upload marked entities. As the instructor, the TA can view entity submissions, along with as well as a summary page with all the files shared and group members' actions in each group's discussion forum. The TA could also start group discussions and view group discussions, adding comments and suggestions wherever needed.

Student

A student like the TA could be in multiple courses. Student_tbl keeps track of the students and all courses they are in. Once the instructor starts a general discussion, each student regardless of group can comment but sharing of files is restricted to only group discussions. Only text (comments) can be shared for group discussions. Every student must be associated with a group. For group discussion, each group member could write messages or share files with all group members. File_tbl is responsible for this. When a file is uploaded, file permission will be set. Meaning a file could be read-only and no other member can delete or edit the file. Unless the file is set to read-only, all other group members could delete or update the file. File_tbl has a file_permission field which is responsible for what can be done

on a file. Each group member could also send private messages to fellow group members. This allows students to write and read messages. Each message cannot be longer than 2000 characters as Most messages should be shorter than the limit. We used the varchar type instead of text for the messages because a text object is big (its fixed size is 65535 characters), which can take up a lot of storage space in the server. We added pagination to the inbox and outbox pages for better user experience and faster webpage loading in case a student has received or sent many messages. On or before the deadline, a final copy must be submitted for grading. Only the group leader can submit the final copy. FinalSubmission_tbl is responsible for this, it has a user_id which is compared with leader_user_id in the Group_tbl ensuring only the group leader submits the final copy. Once the deadline passes no other submission can be made for an entity. But before then, the group leader could delete or update the final submission.

Messaging

The feature consists of three pages: sendmessage, outbox, and inbox. This basic messenger allows students to write and read messages. Each message cannot be longer than 2000 characters as most messages should be shorter than the limit. We used the varchar type instead of text for the messages because a text object is big (its fixed size is 65535 characters), which can take up a lot of storage space in the server. We added pagination to the inbox and outbox pages for better user experience and faster webpage loading in case a student has received or sent many messages.

Limitations

Discussion Board

Students can view discussions from one course at a time rather than a collection of discussions from all of their courses combined. This was simpler to implement and more organized for students. Viewing discussions from a different course requires a student to go back to their main page and select that course to view.

Messaging

We chose a simple approach when implementing private messaging. A student can only privately send a message to one student at a time. Group messages are not supported due to the complexity of the database query. Our messaging app also does not support direct message replies, forward, and reaction functionalities.

Applications Supported

Administrators can add courses and instructors.
Instructors can add students and TAs to their courses.
Students can send and receive private messages to their group members.
Students can create posts in the discussion forum. Instructors, TAs and students can reply to the posts.
Students can submit files to group marked entities and edit file submissions.
Instructors and TAs can view file submissions.

E-R Diagrams and Relational Database Design

Database Tables

User Table

Attribute	Type	Note
user_id (A)	Char(8)	PK, Same as the student id
user_name (B)	Varchar(255)	
user_email (C)	Varchar(255)	
user_password (D)	Varchar(255)	Hashed
User_role (E)	Varchar(20)	Options: Student / Instructor / TA

Note: A person may have more than one account if they have more than one role. Each account can only have one role.

Functional dependencies:

A→BCDE

Also, CE→ABD

Referential constraints: user_id is used to link data in different tables.

Course Section Table

course_id (A)	Char(12)	PK
course_name (B)	Varchar(255)	
course_subject (C)	Char(4)	
course_number (D)	Varchar(255)	
course_section (E)	Varchar(255)	
course_term (F)	Varchar(255)	
course_year (G)	Char(4)	

Functional dependencies:

A→BCDEFG

Also, CDEFG→AB

Referential constraints:

course_id is used to link data in different tables.

Student Table

Role_id (A)	Int AI	PK & FK
user_id (B)	Char(8)	FK
course_id (C)	Char(12)	FK

Functional dependencies:

A→BC

Also, BC→A

Referential constraints: user id from the user table and the course id from the course table

Instructor Table

Role_id (A)	Int AI	PK & FK
user_id (B)	Char(8)	FK
course_id (C)	Char(12)	FK

Functional dependencies:

A→BC

Also, BC→A

Referential constraints: user id from the user table and the course id from the course table

TA Table

Role_id (A)	Int AI	PK & FK
user_id (B)	Char(8)	FK
course_id (C)	Char(12)	FK

Functional dependencies:
A→BC
Also, BC→A
Referential constraints: user id from the user table and the course id from the course table

Group Table (If the group has 4 students, there will be 4 records in the group table, each with a different user id.)

group_id (A)	Char(12)	PK
course_id (B)	Char(12)	FK
group_name (C)	Char(12)	
group_order (D)	Int	The group number within a course.
leader_user_id (E)	Char(8)	

Functional dependencies:
A→BCDE
Also, BD→ACE
Referential constraints: group_id is used to link data in different tables. course_id from the course table, leader_user_id refers to user_id in the user table.

Group Member Table (If the group has 4 students, there will be 4 records in the group table, each with a different user id.)

groupMember_id (A)	Int AI	PK
group_id (B)	Char(12)	PK & FK.
user_id (C)	Char(8)	PK & FK. If the group has 4 students, there will be 4 records in the group table, each with a different user id.
dateJoined (D)	Datetime	
dateLeft (E)	Datetime	After a student left the group, they can only browse the post on or before the date they left.

Functional dependencies:
A→BCDE
Also, BC→ADE

Referential constraints: group_id from the group table, and user_id from the user table

Group Marked Entity (GME) Table

GME_id (A)	Char(12)	PK
course_id (B)	Char(12)	FK
file_name (C)	Varchar(255)	
entity_name (D)	Char(20)	E.g. Assignment 1
file_type (E)	Varchar(255)	Can be assignment or project
deadline (F)	Date	A group member will not get the score of the GME if dateLeft is earlier than the deadline OR dateJoined is later than the deadline.
start_date (G)	Datetime	

Functional dependencies:
A→BCDEFG
Also, BCD→AEFG

Referential constraints: GME_id is used to link data in different tables, course_id from the course table

Discussion Pages/Post Table

post_id (A)	Char(20)	PK
post_text (B)	Varchar(2000)	
GME_id (C)	Char(12)	FK
user_id (D)	Char(8)	FK, the user id of the person that wrote the post.
post_date (E)	Date	
post_time (F)	Time	

Functional dependencies:
A→BCDEF
A user might create two identical posts within the same second under abnormal circumstances, so no other candidate keys.
Referential constraints: GME_id from the Group Marked Entity table, and user_id from the user table.

Discussion Reply Table

reply_id (A)	Char(20)	PK
post_id (B)	Char(20)	FK
reply_text (C)	Varchar(2000)	
user_id (D)	Char(8)	FK, the user id of the person that wrote the reply.
reply_date (E)	Date	
reply_time (F)	Time	

Functional dependencies:
A→BCDEF
No other candidate keys in this table because it is possible that a user creates two identical replies within the same second under abnormal circumstances.
Referential constraints: post_id from the Discussion Pages/Post Table, and user_id from the user table

File Table

file_id (A)	Char(12)	PK
user_id (B)	Char(8)	FK, the person who first uploads the file
GME_id (C)	Char(12)	FK
file_content (D)	tinyblob	
file_date (E)	Date	
file_time (F)	Time	
file_visibility_type (G)	Varchar(255)	Option: private / public E.g., records would be private, public
file_permission (H)	Varchar(255)	A set of instruction

Functional dependencies:
A→BCDEFGH

No other candidate keys in this table because it is possible that a user uploads two files within the same second under abnormal circumstances.

Referential constraints: user_id from the user table, and GME_id from the GME table

File Audit History Table

history_id (A)	Char(20)	PK
file_id (B)	Char(12)	FK
user_id (C)	Char(8)	FK
file_action (D)	Varchar(255)	Options: view only, download, update (replace), delete
history_date (E)	Date	
history_time (F)	Time	

Functional dependencies:
A→BCDEF
A user may create two file histories within the same second under abnormal circumstances, so no other candidate keys.
Referential constraints: file_id from the file table, or user_id from the user table

Final Submission Table

submission_id (A)	Char(20)	PK
group_id (B)	Char(12)	FK
GME_id (C)	Char(8)	FK
user_id (D)	Varchar(255)	FK
submission_date (E)	Datetime	

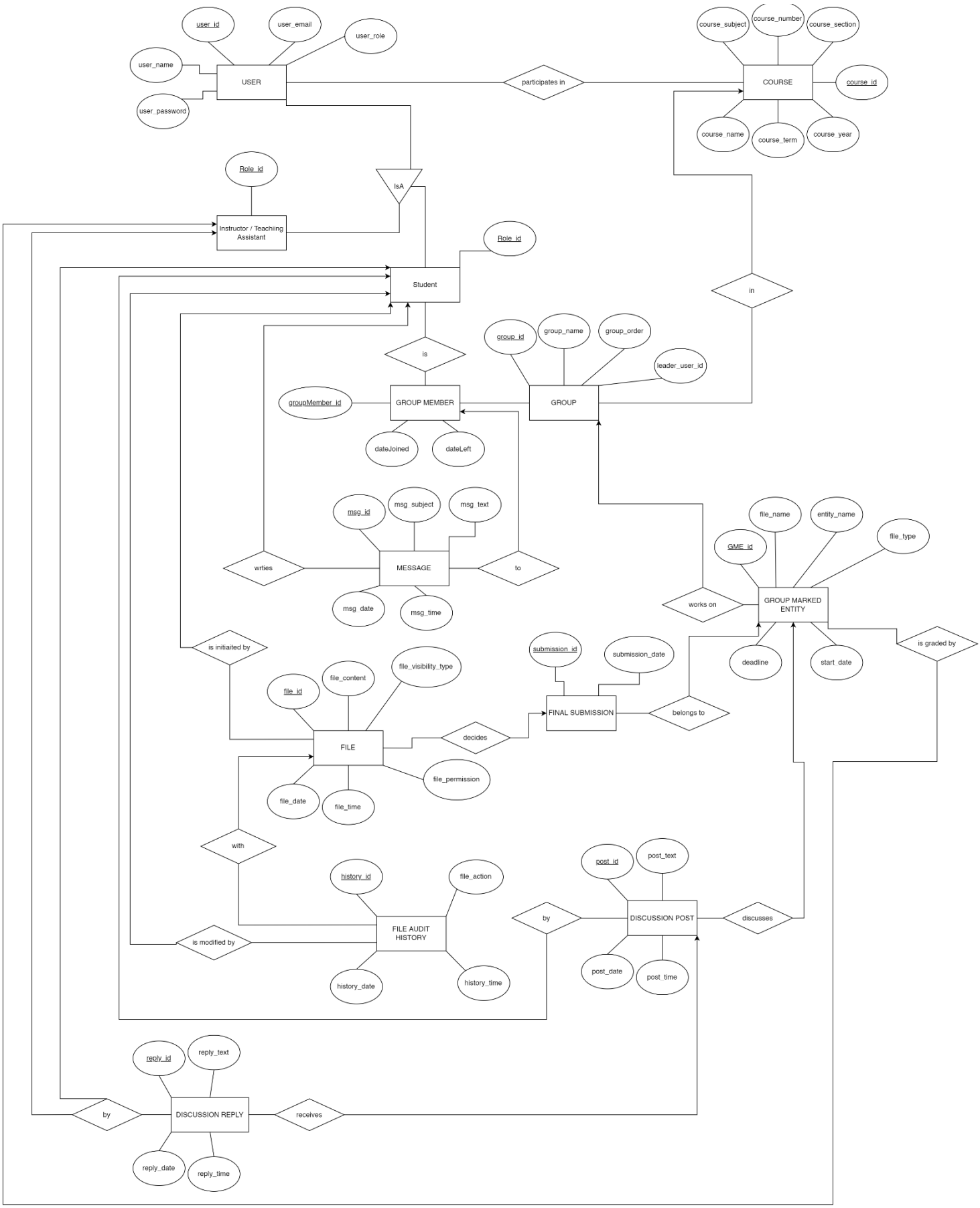
Functional dependencies:
A→BCDE
Also, BCD→AE (if files from multiple members within a group can be included in final submissions)
Or, BC→ADE (if only one member within a group can make a final submission.)
Referential constraints: group_id from the group table, GME_id from the GME table, user_id from the user table

PrivateMessage Table

A student can only send messages to another student who is in the same group in any course.

msg_id (A)	Char(20)	PK
msg_subject (B)	Varchar(200)	
msg_text (C)	Varchar(2000)	
from_user (D)	Char(8)	FK, the user id of the sender
to_user (E)	Char(8)	FK, the user id of the recipient
msg_date (F)	Date	
msg_time (G)	Time	

Functional dependencies:
A→BCDEFG
No other candidate keys in this table because it is possible that a user sends two identical messages within the same second under abnormal circumstances.
Referential constraints: Both from_user and to_user refer to the user_id attribute from the user table.


user_id
user_email
user_role
user_name
USER
user_password
participates in
course_subject
course_number
course_section
COURSE
course_id
course_name
course_term
course_year
Role_id
IsA
Instructor / Teachiing Assistant
Role_id
Student
in
group_id
group_name
group_order
leader_user_id
is
groupMember_id
GROUP MEMBER
GROUP
dateJoined
dateLeft
msg_id
msg_subject
msg_text
wrties
MESSAGE
to
msg_date
msg_time
file_name
entity_name
GME_id
file_type
works on
GROUP MARKED ENTITY
is graded by
deadline
start_date
is initiaited by
submission_id
submission_date
file_id
file_content
file_visibility_type
FINAL SUBMISSION
belongs to
FILE
decides
file_permission
file_date
file_time
with
history_id
file_action
FILE AUDIT HISTORY
is modified by
history_date
history_time
post_id
post_text
by
DISCUSSION POST
discusses
post_date
post_time
reply_id
reply_text
by
DISCUSSION REPLY
receives
reply_date
reply_time