CSC 540 PROJECT 1
TEAM 9
PROJECT REPORT
2nd NOVEMBER 2014

Gradiance
Student Assignment Management system

Team:
Nikhil Nayak [nnayak2]
Vishal Mishra [vmishra]
Maneesh Boruthalupula [mboruth]
Nikhil Satish Pai [npai]

Contents

1. *Assumptions* 03
2. *Problem Statement* 05
3. *Entities and Relationships* 06
4. *Users* 09
5. *E-R diagram* 10
6. *Relational Schemas* 11
7. *Functional Dependencies and Normal Forms* 17
8. *SQL queries* 20
9. *Use case scenarios* 21
10. *References* *26*

ASSUMPTIONS

Administrator Level Assumptions:

- Insertions, Updations or Deletions to be done by the System Administrator:
  - ✓ Enrolling the professors in the system.
  - ✓ Enrolling existing students as Teaching Assistants for a given course.
  - ✓ Adding a list of active courses to the system
  - ✓ Updating the Textbook and other data related to the textbook namely
    - Chapters
    - The corresponding sections
    - The corresponding subsections
  - ✓ All the Questions associated with a particular set of topics.
    - The corresponding set of parameters in the case of parameterized questions.
  - ✓ A set of correct and incorrect options for both the question types
  - ✓ All the data will be updated as per the table structure requirement provided

System Level Assumptions:

- Any student can create an account in the system without the system administrator's intervention.
- Each Course has an Enrollment End Date beyond which the students will not be able to enroll into the course. This is different from the course End Date.
- Passwords for both the student and the Professors are not stored in an encrypted format
- Each user (professor or student) will have will be assigned a unique login id
- Once an exercise is added it can only be deleted by the system Administrator
- Professors cannot edit the following parameters of an exercise after they are assigned during the exercise creation:
  - Topics
  - Difficulty Range
- The Professor cannot add a fixed number of questions during exercise creation. The number of questions will be dynamically updated
- Randomization is done only on the selection of options in the homework questions and not on the order in which the questions are displayed to the student
- The random seed for generation of question options used is
  (Epoch Time - random seed of the exercise entered)
- Random seed default value is 20071969
- The set of topics for a course which are entered are case sensitive
- The correct points entered for a question will always be greater than zero
- There are no default scores assigned for both correct and incorrect answers, these values have to be entered by the professor during the creation of the exercise
- All the non-parameterized questions start with the character 'Q' followed by a numeric value
- All the parameterized questions start with the character 'P' followed by a numeric value
- There is no time gap between attempting successive homeworks
- If the professor changes the number of attempts then the change will not be reflected for students who have already attempted the exercise at least once
- If the professor enters an incorrect value for the score selection method for a particular exercise then the system chooses it as maximum score of all attempts
- Once an attempt is made on a homework then there is no way to quit it midway, the homework has to be finished and it will be considered as an attempt

- The TA must be registered as a student in system before he is assigned a TA role
- Only Graduate students can be made Teaching Assistants
- Students who will be added as TA's will not be already enrolled into courses where there are overlapping topics with the course for which he will be made a TA
- If a TA tries to enroll into a course where the set of course topics overlap with the set of topics of the course for which he is a TA, then he will not be enrolled into the course and a notification will be sent to the professors of both the courses
- Any adjustment in grades will have to be done by the system administrator and not by the professor

General:

- All dates are in 'DD-MM-YYYY' format
- The terms homework, exercise and assignment are used interchangeably in the system

PROBLEM STATEMENT

Issues:

The current issue faced by the university is that there is that all the assignments are managed manually. This increases the workload on the Professors and Teaching Assistants and is in turn a time consuming and repetitive process.

Vision:

To create an automated assignment management system where the time taken for the manual processing of assignments is reduced and in turn helps students master concepts by offering problem sets and giving advice in response to errors. Much of the repetitive work of grading and of creating and debugging assignments done by faculty and teaching assistants is greatly reduced allowing them to develop more challenging work for students and/or spend more time with students who need help [1].

Method:

We have decided to implement this using a Relational Database as the back end. The front end has been implemented as a console level Java application. The Solution can be implemented as a web based service by converting the front end to a hosted application on a server.

The addition of parameterized questions will automate the process of assigning and grading analytic questions which will improve the conceptual understanding of the students [1].

We have also included an additional feature such that the student will be able to provide an explanation of 100 characters of each of his answers if the professor decides to give him partial points based on his understanding of the concepts.

ENTITIES AND RELATIONSHIPS

The major entities in the project are:

- ***Student***
  Attributes*: Id, Student_name, Password, Student_Level, IsTA*
  This maps the students of the university; it contains their unique login and password along with their names and indicates whether they are Undergrads or graduate students.

- ***Professor***
  Attributes*: Id, Professor_name, Password*
  This contains the login details for the Professors of the University to access the system

- ***Course***
  Attributes: *TokenNumber,CourseId,CourseName ,CourseStartDate, EnrollmentEndDate,CourseEndDate, ProfessorId, CourseLevel,MaxEnrolled,StudentsEnrolled*
  Contains all the details about the course like the name, EndDate etc

- ***Textbook***
  Attributes: *ISBN , Title ,Author ,TokenNum*
  Contains a list of all the textbooks which are referenced by the course

- ***Chapters*** *– Weak Entity dependent on Textbook*
  Attributes: *ISBN , Chapter_no, ChapterTitle*
  Lists the chapters within each textbook

- ***Section*** *– Weak Entity dependent on Chapter*
  Attributes: *ISBN , Chapter_No,SectionTitle,*
  Lists the sections within each chapter of the textbook

- ***Sub section*** *– Weak Entity dependent on Sub Section*
  Attributes: *ISBN,Chapter_No,SectionTitle,SubSectionTitle*
  Lists the subsections within each section of the textbook

- ***Questions*** *– is a hierarchical entity with two sub classes, No Overlaps and Covering*
  Non-Parameterized Questions *- QID,Topic,Hint,QuestionLevel, Detailedexplaination,Text*
  Parameterized Questions *- PQId, PTopic, PHint, PLevel, PText, Detailedexplaination*
  List the questions which are used to make up the assignment

- ***Options*** *– Options for the non-parameterized questions*
  Attributes*: QID,OptionId,Optiontext, OptionExplaination,Option_Value*

  POptions *– Options for parameterized questions*
  Attributes: *PQID, OptionId, Optiontext, OptionExplaination, Option_Value, Param_Value*
  Options corresponding to parameterized and non-parameterized questions

- ***Exercise***
  Attributes: *Tokennum, ExerciseId, RandomSeed, NumOfRetries, ScoreSelection, PointsCorrect, PntInPenalty, Edifficultylow, Edifficultyhigh, ExerciseStartDate, ExerciseEndDate*
  Maps the exercise parameters like start date, end date, Number of retires etc.

Relations

- ***Enrollment***
  Between the student and the course
  Maps the students enrolled in a particular course

- ***TA***
  Between the students and the course
  Maps the students who have been added as teaching assistants in a particular course

- ***Teaches***
  Between Professor and Course
  Maps the relationship between a course and the professor who is teaching it

- ***Associated textbook***
  Between a Course and a textbook
  Specifies which textbooks are included for a given course

- ***Topic***
  Between a Course and a Chapter of textbook
  Specifies which Topics are included from the textbook

- ***Under_Chapter***
  Between a Chapter and Section
  Specifies which sections come under a given chapter

- ***Under_Section***
  Between a Section and a Subsection
  Specifies which sub-sections come under a given section

- ***Have_Opt***
  Between a non-parameterized question and it's correct and incorrect options

- ***Have_Popt***
  Between a parameterized question and it's correct and incorrect options

- ***Form***
  Between Exercise and Questions
  Combination of which are used to form an exercise

- ***Gen_Exercise***
  Between Exercise and a course
  Used to generate Exercise by specifying the required parameters

- **Customized**
  Between Exercise and a student
  Generates customised exercise for each student based on the already existing homework for a particular course

USERS

- ***Professor***

- ✓ Identified by a unique login ID which is used to access the features of the system
- ✓ Can only be added only by the system administrator
- ✓ Associated or can teach single or multiple courses
- ✓ Only user who has access to add or update an exercise with all its parameters like start date, end date.
- ✓ Only user who can insert or delete questions into a particular homework.
- ✓ Able to access any question of any valid topic apart from those for which they are teaching

- ***Student***

- ✓ Identified by a unique login ID which is used to access the features of the system
- ✓ Can create an account, and enroll into new courses if all the criteria are met
- ✓ Is able to view or attempt homework only after they are published, and has to attempt them based on restrictions specified in the homework.
- ✓ Does not have access to see information regarding other students nor any other reporting queries of the system.

- ***Teaching Assistant (TA)***

- ✓ A student can also be a TA, provided he is a graduate level student and not enrolled in any course with overlapping topics as the course for which he will be a TA
- ✓ Have access to more information related to the course, such as checking scores and seeing any homework once it is added by the professor
- ✓ Cannot add or modify any given homework of a particular topic
- ✓ Also has access to all the reporting queries same as that of the professor

Note:

A system administrator is required for updating the data into the tables and doing overall supervision and maintenance of the system

E-R DIAGRAM

Note:

The ER diagram is present in the zip file along with the submission – ER.pdf

RELATIONAL SCHEMAS

Note: The relational Schemas have been mapped to Oracle data types instead of the generic types as in the reference book

Student (*Id: VARCHAR2 (10)*, Sname: VARCHAR2 (25), Pswrd: VARCHAR2 (10),
Slevel: VARCHAR2 (2), IsTA: CHAR (1))
Id – The unique login id of the student (Primary Key)
Sname (Not Null) – Then name of the student
Pswrd (Not Null) – Password the student will use to login to the database
Slevel (Not Null) – indicate whether the student is a graduate or undergraduate
IsTA (Not Null) Default 'N' – Flag to indicate if the student is also a TA for some course

TAssistant (*Id: VARCHAR2(10), TokenNum : VARCHAR2(20)*, Sname: VARCHAR2(25),
Isactive: CHAR (1))
Id – Login Id of the student who has been assigned as a TA
TokenNum – Token number of the course for which he is a TA
Sname (Not Null) – Name of the student
Isactive (Not Null) Default 'Y'– Flag to indicate if the he is still a TA (check for course end Date)
Primary Key is (Id,TokenNum)
Foreign Keys:
(TokenNum) referencing Course(TokenNum)
(Id) referencing Student(Id)

Enrollment (*Id: VARCHAR2 (10), TokenNum: VARCHAR2 (20)*, Sname: VARCHAR2 (25))
Id – ID of the student enrolled
TokenNum – Token Number of the course for which he is enrolled
Primary Key (Id,TokenNum)
Foreign Keys :
(TokenNum) referencing Course(TokenNum)
(Id) referencing Student(Id)

Professor (*Id: VARCHAR2 (10)*, Pname: VARCHAR2 (25), Pswrd: VARCHAR2 (10))
Id – Login Id assigned to the professor(Primary Key)
Pname (Not Null) – Name of the professor
Pswrd (Not Null) – Password used to login to the account

Course (*TokenNum: VARCHAR2 (20)*, CourseId: VARCHAR2 (20),CourseName: VARCHAR2(25),
CStartDate DATE, EnrollEndDate DATE, CEndDate DATE, PId: VARCHAR2(10),
CourseLevel: VARCHAR2(10), MaxEnrolled: NUMBER(3), SEnrolled: NUMBER(3))
TokenNum – Token Number of a given course (Primary Key)
CourseId (Not Null) – CourseId assosicated with a particular course
CourseName (Not Null) – Name of the course
CStartDate (Not Null) – The day the course starts
EnrollEndDate (Not Null) - The day the course enrolment ends
CEndDate (Not Null) - The last day of the course
Pid – Id of the professor teaching the course
CourseLevel (Not Null) – Flag to Indicate whether the course is a graduate or undergraduate course
MaxEnrolled (Not Null) – Maximum number of students who can be enrolled in the course
Senrolled (Not Null)– Number of students currently enrolled in the course

Foreign Key:
Pid References Professor (Id)

TopicCourseMap (*Tokennum: VARCHAR2(20), Topic: VARCHAR2(50)*)
TokenNum – Course Token Number
Topic – List of topics associated with the given course
Primary Key (TokenNum, Topic)
Foreign Key(TokenNum) referencing Course(TokenNum)

TextBook (*ISBN: VARCHAR2(13), FTokenNum: VARCHAR2(20)*, Title: VARCHAR2(50),
Author: VARCHAR2(100))
ISBN – The ISBN number of the textbook
Title (Not Null) – The title of the textbook
Author (Not Null) – The author of the textbook
FtokenNum – Token Number of the course with which the text book is associated
Primary Key (FTokenNum,ISBN)
Foreign Key(FTokenNum) referencing Course(TokenNum)

Chapter (*FISBN: VARCHAR2(13), ChptrNo: NUMBER(3)*, ChptrTitle: VARCHAR2(50))
FISBN – ISBN Number of the textbook
ChptrNo – Chapter Number in the textbook
ChptrTitle (Not Null) – Title no of the chapter in the textbook
ChptrTitle (Not Null) – Title text of the chapter in the textbook
Primary Key (FISBN, ChptrNo)

Section (*FISBN: VARCHAR2(13), FChptrNo: NUMBER(3), SectionTitle: VARCHAR2(50)*)
FISBN – Refers to the textbooks ISBN number
FchptrNo – The chapter number to which this section belongs to
SectionTitle – Name of the section
Primary Key(FISBN, FchptrNo, SectionTitle)

SubSection (FISBN: VARCHAR2(13), FChptrNo: NUMBER(3), FSectionTitle: VARCHAR2(50),
SubSectionTitle: VARCHAR2(50))
FISBN – ISBN number of the text book
FchptrNo – Chapter number
FsectionTitle – Title of the section within the chapter
SubSectionTitle – Title of the subsection
Primary Key(FISBN, FChptrNo, FsectionTitle, SubSectionTitle)

Question (*QId: VARCHAR2(4)*,Topic: VARCHAR2(50),Hint: VARCHAR2(100),
QLevel: NUMBER(1), DXPLN: VARCHAR2(50),Text: VARCHAR2(50))
QID – Question Id of the non-parameterized question(Primary Key)
Topic (Not Null) – The topic corresponding to the question
Hint - Hint to solve the question
Qlevel (Not Null) - Difficulty Level of the question
DXPLN (Not Null) - Detailed explaination of the question
Text (Not Null) – Question text

PQuestion (*PQId: VARCHAR2(4)*, PTopic: VARCHAR2(50), PHint : VARCHAR2(100),
PLevel: NUMBER(1), PDxpln: VARCHAR2(50), PText VARCHAR2(50))
PQid – Question Id of the parameterized question(Primary Key)
PTopic (Not Null) – The topic corresponding to the question
PHint - Hint to solve the question
Plevel (Not Null) - Difficulty Level of the question
PDxpln (Not Null) - Detailed explaination of the question
PText (Not Null) – Question text

Options (*QId: VARCHAR2(4),OptionId: NUMBER(10)*,Optiontext : VARCHAR2(25),
OptionExpln: VARCHAR2(25), Crctop: CHAR(1))
QId – Question Id to which the option corresponds
OptionId – Option Id
Optiontext (Not Null) – Option Text
OptionExpln (Not Null) – Explaination for each option
Crctop (Not Null) – value of the option whether it is correct or not
Primary Key(Qid, OptionId)
Foreign Key(QId) referencing Question(PQId)

POptions (*PId: VARCHAR2(4),POptionId: NUMBER(10) ,Pvalue : VARCHAR2(25)*,
POptiontext: VARCHAR2(25), POptionExpln: VARCHAR2(25), Pcorrect: CHAR(1))
PId – Question Id to which the option corresponds
POptionId – Option Id
POptiontext (Not Null) – Option Text
POptionExpln (Not Null) – Explaination for each option
Pcorrect (Not Null) – value of the option whether it is correct or not
Pvalue (Not Null) – Parameter Value
Primary Key(PId, PoptionId,Pvalue)
Foreign Key(PId) referencing PQuestion(QId)

Exercise (*Tokennum: VARCHAR2(20), Eid: VARCHAR2(4)* ,RandomSeed: NUMBER(10),
NumRetries: NUMBER(2), ScoreSelection: CHAR(3), PntCorrect : NUMBER(2),
PntInPenalty: NUMBER(2), Edifficultylow : NUMBER(1), Edifficultyhigh: NUMBER(1),
EStartDate : DATE, EEndDate : DATE)

Tokennum – Token number of the course
Eid - Exercise number
RandomSeed (Not Null) Defalut: 20071969– Seed value used to randomize questions
NumRetries (Not Null)– Number of retries allowed
ScoreSelection (Not Null)– Score selection for the homework
PntCorrect (Not Null)– The number of points for a correct answer
PntInPenalty (Not Null)– Points for incorrect answer
Edifficultylow (Not Null)– Lower range of the exercise difficulty
Edifficultyhigh (Not Null)- Lower range of the exercise difficulty
EStartDate (Not Null)- Start date of the exercise
EendDate (Not Null) – End date of the exercise
Primary key (TokenNum,Id)
Foreign Key(TokenNum) referencing Course(TokenNum)

TopicExerciseMap (*Eid: VARCHAR2(20), Tokennum: VARCHAR2(20)*,Topic : VARCHAR2(50))
Eid – Exercise Id of the assignment
TokenNum – Token Number of the course
Topic – A set of Topics associated with the given Exercise
Primary Key(Eid, Tokennum,Topic)
Foreign Keys :
(TokenNum) referencing Course(TokenNum)

FormExercise(*EId : VARCHAR2(4), Tokennum : VARCHAR2(20), Id : VARCHAR2(4)*)
Eid – The Exercise Id of the assignment
TokenNum – Token Number of the course for which the assignment is loaded
Id – Question id
Primary key (Eid,TokenNum,Id)
Foreign Keys:
TokenNum referencing Course(TokenNum)

View_Scores(*Tokennum : VARCHAR2(20), Eid : VARCHAR2(4), Stud_Id: VARCHAR2(10), Attempt_No : NUMBER(2)*, Scores : NUMBER(5))
Tokennum – Token number of the course
EID – Exercise Id
STUD_ID – Student Id
ATTEMPT_NO – The attempt number
SCORES (Not Null) – total score obtained in the exercise
Primary Key(Tokennum, Eid, Stud_Id, Attempt_No)
Foreign Keys:
TokenNum referencing Course(TokenNum)
Stud_Id referencing Student(Id)

Submission (*Eid : VARCHAR2(4) , Tokennum : VARCHAR2(20), SId : VARCHAR2(10), Attempt : NUMBER(2)*, QId : VARCHAR2(4), Submitted : CHAR(1),
Question : VARCHAR2(50), Optiontext1 : VARCHAR2(25),
Optiontext2 : VARCHAR2(25), Optiontext3 : VARCHAR2(25),
Optiontext4 : VARCHAR2(25), Selected_option : VARCHAR2(25),
Expltn : VARCHAR2(100), Correctvalue : CHAR(1), ParamValue : VARCHAR2(25))

Eid – Exercise Id
Tokennum – Token Number of the course
QId – The question Id
SId – Student Id
Attempt – Attempt Number
Submitted (Not Null) Default 'N'– Flag to indicate if the homework has been submitted
Question (Not Null) – Text of the question
Optiontext1 (Not Null) – Text of option1
Optiontext2 (Not Null)– Text of option2
Optiontext3 (Not Null) – Text of option3
Optiontext4 (Not Null) – Text of option4
Selected_option (Not Null) – The option selected by the user
Expltn (Not Null) - Explaination provided by the user for his answer choice
Correctvalue (Not Null) – Correct answer choice
ParamValue – The parameter value used in case of a parameterized question
Primary Key(TokenNum,Eid,Sid,Attempt)
Foreign Keys:
TokenNum referencing Course(TokenNum)
Stud_Id referencing Student(Id)

ALERTS (Id : VARCHAR2(10), Msgread : CHAR(1),Hwrkno : CHAR(4) ,
Tokennum : VARCHAR2(20) , Msg: VARCHAR2(50))

Id – Login Id of the user for whom the alert is generated can bw student or professor
MsgRead – Flag to indicate if the message has been read
HwrkNo – Homework number for which the message is generated
Tokennum – Token number for which the message is generated
Msg – The message statement

Attempt (*Tokennum : VARCHAR2(20), Studid : VARCHAR2(10), Homework_No: NUMBER(2)*,
Attemptsno: NUMBER(2))

Tokennum – Token number of the course
Studid – Student Id
Homework_No – The exercise Id
AttemptsNo – Specifies the number of attempts remaining
Primary Key (Tokennum, Studid, Homework_no, Attemptsno)
Foreign Keys:
TokenNum referencing Course(TokenNum)
Studid referencing Student(Id)

Course_Inactive (*TokenNum : VARCHAR2(20)*, CourseId : VARCHAR2(20),
CourseName : VARCHAR2(25), CStartDate : DATE, EnrollEndDate : DATE,
CEndDate : DATE, PId : VARCHAR2(10), CourseLevel : VARCHAR2(10),
MaxEnrolled : NUMBER(3), SEnrolled : NUMBER(3))

TokenNum – Token Number of a given course (Primary Key)
CourseId (Not Null) – CourseId assosicated with a particular course
CourseName (Not Null) – Name of the course
CStartDate (Not Null) – The day the course starts
EnrollEndDate (Not Null) - The day the course enrolment ends
CEndDate (Not Null) - The last day of the course
Pid (Not Null) – Id of the professor teaching the course
CourseLevel (Not Null)– Flag to Indicate whether the course is a graduate or undergraduate course
MaxEnrolled (Not Null) – Maximum number of students who can be enrolled in the course
Senrolled (Not Null)– Number of students currently enrolled in the course

Fscore (Tokennum : VARCHAR2(20), Eid : VARCHAR2(4),Stud_Id : VARCHAR2(10),
Scores : NUMBER(5))

Tokennum – Token Number of the course
Eid – Exercise Id of the coures
Stud_Id – Student Id
Scores (Not Null) – score of the homework
Primary Key(Tokennum , Eid ,Stud_Id)

FUNCTIONAL DEPENDENCIES AND NORMAL FORM

Table: Student(Id, Sname, Pswrd, SLevel, IsTA)
FDs : Id → Sname, Pswrd, SLevel, IsTA
The table is in BCNF

TAssistant(Id,TokenNum,Sname,Isactive)
FDs : Id,TokenNum → Sname, Isactive
The table is in BCNF

Enrollment(Id, TokenNum, Sname)
FDs : Id,TokenNum → Sname
The table is in BCNF

Professor(Id, Pname, Pswrd)
FDs : Id → Pname, Pswrd
The table is in BCNF

Course(TokenNum,CourseId,CourseName,CStartDate,EnrollEndDate,CEndDate,PId,CourseLevel,Max Enrolled,SEnrolled)
FDs: TokenNum → CourseId, CourseName, CStartDate, EnrollEndDate, CEndDate, PId, CourseLevel, MaxEnrolled, SEnrolled
The table is in BCNF

For each course there can be multiple topics, this was violating 1NF, hence we have split the mapping between course and topics into another table
TopicCourseMap(Tokennum,Topic)
There are no FD's
The table is in BCNF

TextBook(ISBN, FTokenNum, Title,Author)
ISBN, FTokenNum → Title,Author
ISBN → Title,Author
The table is in 1NF

Chapter(FISBN,ChptrNo,ChptrTitle)
FDs: FISBN,ChptrNo → ChptrTitle
The table is in BCNF

Section(FISBN,FChptrNo,SectionTitle)
There are no FD's
The table is in BCNF

SubSection(FISBN, FChptrNo,FSectionTitle,SubSectionTitle)
There are no FD's
The table is in BCNF

Question(QID,Topic,Hint,QLevel,DXPLN,Text)
FDs: QID → Topic,Hint,QLevel,DXPLN,Text
The table is in BCNF

PQuestion(PQId,PTopic,PHint,PLevel,PText,DXPLN)
FDs: PQId → PTopic,PHint,PLevel,PText,DXPLN
The table is in BCNF

Options(QID,OptionId,Optiontext,OptionExpln,CRCTOP)
FDs: QID,OptionId → Optiontext,OptionExpln,CRCTOP
The table is in BCNF

POptions(PID,POptionId,POptiontext, POptionExpln,PCORRECT,PVALUE,DXPLN)
FDs : PID,PoptionId → POptiontext, POptionExpln,PCORRECT,PVALUE,DXPLN
The table is in BCNF

Exercise(Tokennum,EId,RandomSeed,NumRetries,ScoreSelection,PntCorrect,PntInPenalty,
Edifficultylow,Edifficultyhigh,EStartDate,EEndDate)
FDs : Tokennum,Eid → RandomSeed,NumRetries,ScoreSelection,PntCorrect,PntInPenalty,
Edifficultylow,Edifficultyhigh,EStartDate,EEndDate
The table is in BCNF

For each exercise there can be multiple topics, this was violating 1NF, hence we have split the mapping between exercise and topics into another table
TopicExerciseMap(EId,Tokennum,Topic)
There are no FD's
The table is in BCNF

For each exercise there can be multiple questions, this was violating 1NF, hence we have split the mapping between exercise and questions into another table
FormExercise(EId,Tokennum,Id)
There are no FD's
The table is in BCNF

View_Scores(Tokennum,Eid,Stud_Id,Attempt_No,Scores)
Tokennum,Eid,Stud_Id, Attempt_No → Scores
The table is in BCNF

Submission(EId ,Tokennum, SId, QId, Attempt, Submitted, Question, Optiontext1, Optiontext2,
Optiontext3, Optiontext4, Selected_option, Expltn , Correctvalue , ParamValue)
FD's : EId ,TOKENNUM, Sid → QId, Attempt, Submitted, Question, Optiontext1, Optiontext2,
Optiontext3, Optiontext4, Selected_option, Expltn , Correctvalue ,
ParamValue
The table is in BCNF

ALERTS (Id,Msgread, Hwrkno ,Tokennum ,Msg)
This table has no primary key defined as it is used only for the purpose of maintaining a log of the alert messages
The table is in 1NF

Attempt(Tokennum,Studid,Homework_No,Attemptsno)
Tokennum,Studid,Homework_No → Attemptsno
The table is in BCNF

Course_Inactive(TokenNum,CourseId,CourseName,CStartDate,EnrollEndDate,CEndDate,PId,CourseLevel,MaxEnrolled,SEnrolled)
TokenNum → CourseId, CourseName, CStartDate, EnrollEndDate, CEndDate, CourseLevel,
MaxEnrolled, SEnrolled, PId
The table is in BCNF

Fscore(Tokennum, Eid, Stud_Id, Scores)
Tokennum, Eid, Stud_Id → Scores
The table is in BCNF

QUERIES

Note: The character string 'coursetoken' refers to a particular course token for which the reports are being generated

1. Find students who did not take homework 1

```
SELECT SNAME , ID
FROM ENROLLMENT
WHERE TOKENNUM = 'coursetoken' AND
ID NOT IN
 (SELECT DISTINCT STUD_ID FROM VIEW_SCORES WHERE TOKENNUM = 'coursetoken'
   AND EID="1")
```

2. Find students who scored the maximum score on the first attempt for homework 1

```
SELECT E.SNAME, E.ID
FROM ENROLLMENT E
WHERE E.ID IN
(SELECT STUD_ID FROM VIEW_SCORES WHERE EID = "1" AND ATTEMPT_NO = "1" AND
TOKENNUM = 'coursetoken' AND
SCORES = (SELECT MAX(SCORES) FROM VIEW_SCORES WHERE EID = "1" AND ATTEMPT_NO = "1"
AND TOKENNUM =  'coursetoken'))
```

3. Find students who scored the maximum score on the first attempt for each homework

```
SELECT E.SNAME, F1.EID, F1.STUD_ID, F1.SCORES, F2.SCR
FROM FSCORE F1 JOIN
 (SELECT AVG(SCORES) AS SCR, STUD_ID
    FROM FSCORE WHERE TOKENNUM = 'coursetoken'  GROUP BY STUD_ID) F2
   ON F2.STUD_ID = F1.STUD_ID AND TOKENNUM = 'coursetoken'
```

4. For each student, show total score for each homework and average score across all homeworks

```
SELECT E.SNAME, F1.EID, F1.STUD_ID, F1.SCORES, F2.SCR
FROM FSCORE F1 JOIN
(SELECT AVG(SCORES) AS SCR,STUD_ID
   FROM FSCORE
    WHERE TOKENNUM = 'coursetoken' GROUP BY STUD_ID) F2 ON
 F2.STUD_ID = F1.STUD_ID AND TOKENNUM = 'coursetoken'  JOIN
ENROLLMENT E ON E.ID = F1.STUD_ID
```

5. For each homework, show average number of attempts

```
SELECT AVG(TEMP.CNT), TEMP.EID
FROM
(SELECT MAX(ATTEMPT_NO) AS CNT,EID,STUD_ID FROM VIEW_SCORES
   WHERE TOKENNUM = ?   GROUP BY EID, STUD_ID) TEMP GROUP BY TEMP.EID
```

USE CASE SCENARIOS
UML DIAGRAM

Note:

The UML diagram is present in the zip file along with the submission – UML.pdf

SELECTED USE CASES

Actor: ***Student***

1. Students can register in the system without needing permission of the system administrator

Triggers:

- The student wants to register to use the gradiance offline system

Preconditions:

- The user ID which the student wants to use must not be already existing in the system

Post-conditions:

- The student will be registered in the system with a unique Id

Normal Flow:

- The student access the system and chooses the option to register
- The student enters all the details and if all conditions are met a message is displayed to the user whether the registration is successful or not

2. Students can enroll in subjects without needing a permission of database administrator. Triggers:

- The student wants to enroll in a course

Preconditions:

- The user must have a unique Id to login

Post-conditions:

- The student will be enrolled in the course if all the prerequisites are met

Normal Flow:

- The student logins into the system
- The student is displayed a list of open courses to which he can enroll and if all conditions are met a message is displayed to the user whether the enrollment is successful or not

3. A homework is not made visible to the students before the publish date and after deadline:

Triggers:

- The student wants to view or attempt the homework

Preconditions:

- The user must have a unique id to login to the system
- The user must be already enrolled in the course for which he wants to attempt or view the homework
- Current date must be between homework start date and homework deadline

Post-conditions:

- The student will able to view or attempt the homework if all preconditions are met

Normal Flow:

- The student access the system and selects the course for which he wants to attempt the homework
- A list of active homeworks with the number of attempts remaining are displayed to him

4. Student gets an alert if he does not attempt a particular homework even once before one day before the end date

Triggers:

- The student wants to view notifications

Preconditions:

- The student must have a unique id to login to the system
- The student must be already enrolled in the course
- Current date must be one day less than the homework deadline
- The number of attempts of the homework must be zero

Post-conditions:

- The student will able to view the notification in his alerts tab

Normal Flow:

- The student access the system and selects the course
- In the alerts tab a message will be displayed

<u>Actor:</u> Teaching Assistant - TA

5. TA can see the of details homework of the class

Triggers:

- The TA wants to view the assignment details of the course for which he is a TA

Preconditions:

- The student must be registered as a TA for the course
- The course must already have at least 1 homework created by the professor

Post-conditions:

- The TA will able to view all the homework details

Normal Flow:

- The TA logs into the system and selects the course for which he wants to
- A list of all homeworks are displayed to him

6. TA wants to view the reports for a particular homework.

Triggers:

- The TA wants to view the reports for a particular homework

Preconditions:

- The homework must have been already added by the professor

Post-conditions:

- The TA will be able to see all the reports

Normal Flow:

- The TA logs into the system and through his TA menu views the reports for a particular homework

7. TA is not allowed to enroll for courses that have similar topic.

Triggers:

- The TA wants to enroll as a student to a course where the course topics overlap with the course for which he is a TA

Preconditions:

- The course to which the TA wants to enroll to must have be open for enrollment

Post-conditions:

- The TA will not be enrolled and a message is displayed
- A notice is sent to the professors of both the cases

Normal Flow:

- The TA logs into the system and through his student menu tries to enroll into the course

Actor: ***Professor***

8. The professor wants to create an exercise and add questions from the question bank

Triggers:

- The Professor wants to assign an assignment to a course

Preconditions:

- The professor must have a valid login Id
- The professor must be the teacher assigned the course

Post-conditions:

- The homework will be updated in the system

Normal Flow:

- The professor logs into the system and selects the course which he wants to add the homework
- The professor chooses the add homework option to add a new homework with the parameters like Start date, points per correct answer, randomization seed etc.
- The professor updates the questions to the homework by searching through all questions for the set of topics under the exercise

9. The professor wants to view reports for the homeworks

Triggers:

- The Professor wants to view the reports of a particular homework

Preconditions:

- The professor must have already created the homework
- The homework must be for a course which he teaches

Post-conditions:

- All the relevant details of the homework will be displayed

Normal Flow:

- The professor logs into the system and selects the course for which he wants to view the homeworks
- All the reports will be displayed through the Reports menu

10. Professor gets an alert when a TA tries to enroll for a course with similar topics

Triggers:

- The Professor wants to view his alerts

Preconditions:

- A TA must have tried to enroll into a course which the professor is teaching with topics overlapping with the course for which the student is a TA
  OR
- A TA must have tried to enroll into a course with overlapping topics along with the one for which he is a TA

Post-conditions:

- The professor will be able to view the alerts

Normal Flow:

- The professor logs into the system and checks the alerts option for a particular course
- A list of all unread messages are displayed with the details of the TA

REFERENCES

[1] - http://www.gradiance.com/idea.html