DBMS – PROJECT REPORT

Assumptions:

No duplication of username: We assume that *username* is always unique and hence uniquely identifies each user. The system will not allow to create a user with the username XYZ, if a different user with the same username already exists. As a practical instance, if a user has different roles (such as a student as well as a TA), the system will not allow the user to use "Create a User" functionality to add the other role.

Picking questions for a homework: During preparing exercises, we assume that, only the questions matching the difficulty level of the exercise are the candidate questions.

Problem Statement:

The problem statement is to design a Course Assessment Application which describes a system for managing assignments. The users of the system will be Course Instructors, Students and Teaching Assistants who will have different views of the system based on their roles.

As a student, one can do the following:

- Enroll in different courses using the system.
- Attempt Exercises which are posted by the Instructor(s) for the courses that the student is enrolled in.
- View his scores and past submissions for all the exercises he/she attempted.

As an instructor, the system offers the following features:

- Add self as an instructor for a course.
- Add exercises for the courses and edit them in anyway.
- Add or Remove questions and their respective options to different exercises.

As a Teaching Assistant for a course, one can:

- View all the exercises posted by the Instructor for that course.

Entities and Relationships:

Entities :

- Courses : A master catalog of the courses offered. The primary identifier of it is the Course ID that is assumed to be unique and hence uniquely identify each entry in this entity's instance.

- Course Offerings : An instance (Offering) of one of the courses listed in the catalog "Courses". This entity contains the various offerings of the courses. Hence, the combination of Course ID and Course token (which typically is "ABCFALL14" , for a course ABC offered in Fall 2014, for an instance) uniquely identifies every tuple in this entity set. All the properties of a course offer such as maximum enrollment, current enrollment, start and end dates etc are captured here as they are the properties of a course offering and not of a course (as it can have multiple offerings across semesters).

- Textbooks : A master catalog of all the textbooks which could/could not be used for any course.

- Chapters of Textbooks (Weak entity) : This is modelled as a *weak entity* as the chapter, in its essence has no existence unless it is tagged to a 'textbook' of which it is a part of.

- Sections of Chapters (Weak entity) : This is also modelled as a *weak entity* as a section is only valid when we relate it to a particular 'chapter' which becomes the identifying relation for 'sections'.

- Subsections of Sections (Weak entity) : This is a *weak entity* similar to the above. The only difference being that this is one hierarchy lower.

- Topics : A master catalog of all the topics.

- Questions : The set of all the questions.

- Options for Questions (Weak entity) : The collection of all the options for all the questions in the entity set 'Questions'. We modelled this as a *weak entity* as well. This is because an option is a property of a question and hence, we must include the corresponding question to unique identify any given option.

- Incorrect Options (ISA hierarchy, a child of 'Options') : The set of all the incorrect options of all the questions. This entity set is modelled as one of the children of the entity set Options.

- Correct Options (ISA hierarchy, a child of 'Options') : The set of all the correct options of all the questions. This entity set is modelled as one of the children of the entity set Options.

- Homework : This entity set captures all the homeworks that have been prepared by instructors of various courses.
- Submission : This entity set captures all the submissions to different available homeworks that students enrolled in a particular course have made.

- Users : 'Users' is an entity set that has the general information about all the users of the system.

- Students (ISA hierarchy, a child of 'Users') : Stores all the users that may use the application as a 'Student' role. It is modelled as a child of 'Users' which is clearly the superset of 'Students'.

- Teaching Assistants (ISA hierarchy, a child of 'Users') Stores all the users that may use the application as a 'Teaching Assistant' role. It is modelled as a child of 'Users' which is clearly the superset of 'Teaching Assistants'.

- Instructors (ISA hierarchy, a child of 'Users') : Stores all the users that may use the application as an 'Instructor' role. It is modelled as a child of 'Users' which is clearly the superset of 'Instructors'.

Relationships:

- Enrolls – Relationship between “Students” and “Course Offerings”
- Teaches – Relationship between “Instructors” and “Course Offerings”
- Is a TA for – Relationship between “Teaching Assistants” and “Course Offerings”
- Course_Topics – Relationship between “Course Offerings” and “Topics”
- Course_Textbooks – Relationship between “Course Offerings” and “Textbooks”
- is Offered – Relationship between “Courses” and “Course Offerings”
- Has1- Relationship between “Textbooks” and “Chapters”
- Has2- Relationship between “Chapters” and “Sections”
- Has3- Relationship between “Sections” and “Subsections”
- Contains – Relationship between “Topics” and “Questions”
- Has Answer – Relationship between “Questions” and “Options”
- HW_Questions – Relationship between “Homework” and “Questions”
- Has4- Relationship between “Course Offerings” and “Homework”
- is submitted- Relationship between “Homework” and “Submission”
- is submitted by – Relationship between “Submission” and “Students”

Users:
Students
Teaching Assistants
Instructors

Relational Schema:
Note: The DDL Statements have been directly picked up from Oracle using dbms_metadata.get_ddl() method. The formatting of the text is due to the same.

1. Courses
RS :
Courses(cid : string, cname:string)

DDL Statement:

```
CREATE TABLE "ABOKE"."COURSES"
(    "CID" VARCHAR2(20),
     "CNAME" VARCHAR2(50),
      PRIMARY KEY ("CID")
)
```

2. Course Offerings
RS:
Course_Offerings(cid : string, ctoken:string, start_dt:date, end_dt: date, level:string, Maximum Enrollment: number, Current Enrollment: number)

DDL Statement:

```
CREATE TABLE "ABOKE"."COURSE_OFFERINGS"
```

```
(       "CID" VARCHAR2(20),
        "CTOKEN" VARCHAR2(50),
        "START_DT" DATE,
        "END_DT" DATE,
        "CLEVEL" VARCHAR2(20),
        "MAXIMUM_ENROLLMENT" NUMBER,
        "CURRENT_ENROLLMENT" NUMBER,
         PRIMARY KEY ("CID", "CTOKEN")
)
```

3. Topics
RS:
Topics(tid: string, tname:string)

DDL Statement:

```
CREATE TABLE "ABOKE"."TOPICS"
(       "TID" NUMBER,
        "TNAME" VARCHAR2(75),
         PRIMARY KEY ("TID")
)
```

4. TextBooks
RS:
TextBooks(isbn:string, author:string, title:string)
DDL Statement:

```
CREATE TABLE "ABOKE"."TEXTBOOKS"
(       "ISBN" VARCHAR2(50),
        "AUTHOR" VARCHAR2(50),
        "TITLE" VARCHAR2(75),
```

```
 PRIMARY KEY ("ISBN")

)
```

5. Chapters (Weak entity to Topics)
RS:
Chapters(isbn:string (Foreign key to Textbooks), chnumber:number (partial key), title:string)

DDL Statement:

```
CREATE TABLE "ABOKE"."CHAPTERS"

(    "ISBN" VARCHAR2(50),

     "CHNUMBER" NUMBER,

     "TITLE" VARCHAR2(50),

      PRIMARY KEY ("ISBN", "CHNUMBER")

     )
```

6. Sections (Weak entity to Chapters)
RS:
Sections(isbn:string (Foreign key to Textbooks), chnumber:number (partial key of 'Chapters'), secid:string (partial key), sec_title:string)

DDL Statement:

```
CREATE TABLE "ABOKE"."SECTIONS"

(    "ISBN" VARCHAR2(50),

     "CHNUMBER" NUMBER,

     "SECID" VARCHAR2(20),

     "SEC_TITLE" VARCHAR2(50),

      PRIMARY KEY ("ISBN", "CHNUMBER", "SECID")

     )
```

7. Subsections (Weak entity to Sections)
RS:

Subsections(isbn:string (Foreign key to Textbooks), chnumber:number (partial key of 'Chapters'), secid:string (partial key of 'Sections'), ssecid:string (partial key), subsec_title:string)

DDL Statement:

```
CREATE TABLE "ABOKE"."SUBSECTIONS"
(       "ISBN" VARCHAR2(50),
        "CHNUMBER" NUMBER,
        "SECID" VARCHAR2(20),
        "SSECID" VARCHAR2(20),
        "SUBSEC_TITLE" VARCHAR2(50),
         PRIMARY KEY ("ISBN", "CHNUMBER", "SECID", "SSECID")
        )
```

8. Course_Topics (Relationship between 'Course Offerings' and 'Topics')
Constraints : None (zero or more from both sides)
RS:

Course_Topics(cid:string (Foreign key to Course Offerings), ctoken:string (Foreign key to Course Offerings) isbn:string (Foreign key to Chapters), chnumber (Foreign key to Chapters))

DDL Statement:

```
CREATE TABLE "ABOKE"."COURSE_TOPICS"

(       "CID" VARCHAR2(20),

        "CTOKEN" VARCHAR2(50),

        "TID" NUMBER,

         PRIMARY KEY ("CID", "CTOKEN", "TID")

         FOREIGN KEY ("CID", "CTOKEN")

          REFERENCES "ABOKE"."COURSE_OFFERINGS" ("CID", "CTOKEN"),

         FOREIGN KEY ("TID")

          REFERENCES "ABOKE"."TOPICS" ("TID")

        )
```

9. Course_Textbooks (Relationship between 'Course Offerings' and 'Textbooks')
Constraints : Every course offering must have atleast one textbook (1 to many from Courses-> Textbooks)
RS:
Course_Textbooks(cid:string (Foreign key to Course Offerings), ctoken:string (Foreign key to Course Offerings) isbn: string (Foreign key to Textbooks))

DDL Statement:

```
CREATE TABLE "ABOKE"."COURSE_TEXTBOOKS"
(       "CID" VARCHAR2(20),
        "CTOKEN" VARCHAR2(50),
        "ISBN" VARCHAR2(50),
         PRIMARY KEY ("CID", "CTOKEN", "ISBN")
         FOREIGN KEY ("CID", "CTOKEN")
          REFERENCES "ABOKE"."COURSE_OFFERINGS" ("CID", "CTOKEN") ON
DELETE CASCADE,
         FOREIGN KEY ("ISBN")
          REFERENCES "ABOKE"."TEXTBOOKS" ("ISBN") ON DELETE CASCADE
        )
```

10. Course Teaching Assistants (Relationship between Course Offerings and Tas)
Constraints : A student can be a TA to exactly one course (self inflicted)

This relationship won't have any separate relation as it is a total participation (exactly one kind of a relationship from the TA side) constraint from the TA side.

Relation TA will have a course id in it.

11. Users
RS:
Users(unityid:string, fname:string, lname:string, password:string)

DDL Statement:

```
CREATE TABLE "ABOKE"."USERS"
(       "UNITYID" VARCHAR2(20),
        "FNAME" VARCHAR2(25),
        "LNAME" VARCHAR2(25),
        "PASSWORD" VARCHAR2(20),
         PRIMARY KEY ("UNITYID")
        )
```

12. Tassistants (ISA Users)
RS:
Tassistants(unityid:string, cid:string, ctoken:string)

Note : cid,ctoken is added to hold 'Has 1' Constraints

DDL Statement:

```
CREATE TABLE "ABOKE"."TASSISTANTS"
(       "UNITYID" VARCHAR2(20),
        "CID" VARCHAR2(20) NOT NULL ENABLE,
        "CTOKEN" VARCHAR2(50) NOT NULL ENABLE,
         PRIMARY KEY ("UNITYID")
         FOREIGN KEY ("CID", "CTOKEN")
          REFERENCES "ABOKE"."COURSE_OFFERINGS" ("CID", "CTOKEN") ON
DELETE CASCADE,
         CONSTRAINT "FC_TAS" FOREIGN KEY ("UNITYID")
          REFERENCES "ABOKE"."USERS" ("UNITYID") ON DELETE CASCADE
        )
```

13. Students (ISA Users)
RS:
Students(unityid:string, ed_level varchar2(20));
DDL Statement:

```
CREATE TABLE "ABOKE"."STUDENTS"
(       "UNITYID" VARCHAR2(20),
        "ED_LEVEL" VARCHAR2(20),
         PRIMARY KEY ("UNITYID")
         CONSTRAINT "FC_STUDENTS" FOREIGN KEY ("UNITYID")
          REFERENCES "ABOKE"."USERS" ("UNITYID") ON DELETE CASCADE
        )
```

14. Instructors(ISA Users)
RS:
Instructors(unityid:string)

DDL Statement:

```
CREATE TABLE "ABOKE"."INSTRUCTORS"
```

```
(       "UNITYID" VARCHAR2(20),
         PRIMARY KEY ("UNITYID")
         CONSTRAINT "FC_INSTRUCTORS" FOREIGN KEY ("UNITYID")
          REFERENCES "ABOKE"."USERS" ("UNITYID") ON DELETE CASCADE
        )
```

15. Users_Roles

DDL Statement:

```
CREATE TABLE "ABOKE"."USERS_ROLES"
(       "UNITYID" VARCHAR2(20),
        "ROLE" VARCHAR2(20),
         PRIMARY KEY ("UNITYID", "ROLE")
         FOREIGN KEY ("UNITYID")
          REFERENCES "ABOKE"."USERS" ("UNITYID") ON DELETE CASCADE
        )
```

16. Course_Instructors (Relationship between 'Course_Offerings' and 'Instructors')

Constraints : (*We may put this)* Every course must atleast have one instructor (1 to many from Courses-> Instructors)

RS:

Course_Instructors (unityid:string (Foreign key to Instructors), cid:string (Foreign key to Course Offerings), ctoken:string (Foreign key to Course Offerings))

DDL Statement:

```
CREATE TABLE "ABOKE"."COURSE_INSTRUCTORS"
(       "UNITYID" VARCHAR2(20),
        "CID" VARCHAR2(20),
        "CTOKEN" VARCHAR2(50),
         PRIMARY KEY ("UNITYID", "CID", "CTOKEN")
         FOREIGN KEY ("UNITYID")
          REFERENCES "ABOKE"."INSTRUCTORS" ("UNITYID") ON DELETE CASCADE
ENABLE,
```

FOREIGN KEY ("CID", "CTOKEN")

REFERENCES "ABOKE"."COURSE_OFFERINGS" ("CID", "CTOKEN") ON DELETE CASCADE ENABLE

)

17. Course_Students (Relationship between 'Course_offerings' and 'Students')
Constraints : None

RS:
Course_Students (unityid:string (Foreign key to Students), cid:string (Foreign key to Course_offerings), ctoken:string (Foreign key to Course_offerings))
DDL Statement:

CREATE TABLE "ABOKE"."COURSE_STUDENTS"

("UNITYID" VARCHAR2(20),

"CID" VARCHAR2(20),

"CTOKEN" VARCHAR2(50),

PRIMARY KEY ("UNITYID", "CID", "CTOKEN")

FOREIGN KEY ("UNITYID")

REFERENCES "ABOKE"."STUDENTS" ("UNITYID") ON DELETE CASCADE ENABLE,

FOREIGN KEY ("CID", "CTOKEN")

REFERENCES "ABOKE"."COURSE_OFFERINGS" ("CID", "CTOKEN") ON DELETE CASCADE ENABLE

)

18. Submission
RS:
Submission (TimeStamp: TIME, SubmissionID: INTEGER, Score FLOAT, Report VARCHAR(1000), UnityID VARCHAR(20), HWID INTEGER, Long_Report VARCHAR(3000))
DDL Statement:

CREATE TABLE "ABOKE"."SUBMISSION"

("TIMESTAMP" TIMESTAMP (0),

"SUBMISSIONID" NUMBER,

"SCORE" FLOAT(126),

"REPORT" VARCHAR2(1000),

"UNITYID" VARCHAR2(20) NOT NULL ENABLE,

"HWID" NUMBER NOT NULL ENABLE,

"LONG_REPORT" VARCHAR2(3000),

PRIMARY KEY ("SUBMISSIONID") ,
FOREIGN KEY ("UNITYID")

REFERENCES "ABOKE"."STUDENTS" ("UNITYID") ON DELETE CASCADE ENABLE,

FOREIGN KEY ("HWID")

REFERENCES "ABOKE"."HOMEWORK" ("HWID") ON DELETE CASCADE ENABLE

)

19. Homework
RS:
Homework(HWID:Number, StartDate:Date, EndDate:Date, NumOfRetries:Number, PointsForCorrectQues:Number, PointsForIncorrectQues:Number)
DDL Statement:

CREATE TABLE "ABOKE"."HOMEWORK"

("HWID" NUMBER,

"CID" VARCHAR2(20) NOT NULL ENABLE,

"CTOKEN" VARCHAR2(40) NOT NULL ENABLE,

"STARTDATE" DATE,

"ENDDATE" DATE,

"NUMOFRETRIES" NUMBER,

"SELECTIONMETHOD" VARCHAR2(50),

"DIFFICULTYRANGE" NUMBER,

"POINTSFORCORRECTQUES" NUMBER,

"POINTSFORINCORRECTQUES" NUMBER,

"NUMOFQUESTIONS" NUMBER,

"TID" NUMBER,

CONSTRAINT "DIFFICULTY_CHECK" CHECK (DifficultyRange BETWEEN 1 and 6) ENABLE,

PRIMARY KEY ("HWID")

FOREIGN KEY ("CID", "CTOKEN")

REFERENCES "ABOKE"."COURSE_OFFERINGS" ("CID", "CTOKEN") ON DELETE CASCADE ENABLE,

CONSTRAINT "FK_HW" FOREIGN KEY ("TID")

REFERENCES "ABOKE"."TOPICS" ("TID") ENABLE

)

20. HW_Questions (Relationship between Homework and Questions)
RS:
HWQues(HWID:Integer, QID:Integer)

DDL Statement:

CREATE TABLE "ABOKE"."HW_QUESTIONS"

("HWID" NUMBER,

"QID" NUMBER,

PRIMARY KEY ("HWID", "QID")

FOREIGN KEY ("HWID")

REFERENCES "ABOKE"."HOMEWORK" ("HWID") ENABLE,

FOREIGN KEY ("QID")

REFERENCES "ABOKE"."QUESTIONS" ("QID") ENABLE

)

21. Options (Weak entity to Questions)
RS:
Options(QID:Integer(Foreign Key to Questions), OID:Integer (partial key), OptionText:String, ShrtDesc:String)

DDL Statement:

```
CREATE TABLE "ABOKE"."OPTIONS"
(       "OID" NUMBER(*,0),
        "OPTIONTEXT" VARCHAR2(500),
        "SHRTDESC" VARCHAR2(500),
        "QID" NUMBER,
         PRIMARY KEY ("QID","OID")
         CONSTRAINT "FK_OPTIONS" FOREIGN KEY ("QID")
          REFERENCES "ABOKE"."QUESTIONS" ("QID") ON DELETE CASCADE
ENABLE
        )
```

22. Questions
RS:
Questions(QID:Integer, DiffLevel:Integer, Hint:String)

DDL Statement:

```
CREATE TABLE "ABOKE"."QUESTIONS"
(       "QID" NUMBER(*,0),
        "QUES" VARCHAR2(500),
        "DIFFLEVEL" NUMBER(*,0),
        "HINT" VARCHAR2(100),
        "TOPIC_ID" NUMBER,
         PRIMARY KEY ("QID")
         CONSTRAINT "FK_QUESTIONS" FOREIGN KEY ("TOPIC_ID")
          REFERENCES "ABOKE"."TOPICS" ("TID") ENABLE
        )
```

23. Incorrect Options (ISA Options)
RS:
Incorrect Options(QID:Integer(Foreign Key to Questions), OID:Integer(Foreign Key to

Options), IID:Integer)

DDL Statement:

```
  CREATE TABLE "ABOKE"."INCORRECTOPTIONS"

   (      "QID" NUMBER(*,0),

          "OID" NUMBER(*,0),

          "IID" NUMBER(*,0),

           CONSTRAINT "PK_IO" PRIMARY KEY ("QID", "OID", "IID")

          CONSTRAINT "FK_IO_QUESTIONS" FOREIGN KEY ("QID")
 REFERENCES                "ABOKE"."QUESTIONS" ("QID") ON DELETE CASCADE
ENABLE,
          CONSTRAINT "FK_IO" FOREIGN KEY ("OID")

           REFERENCES "ABOKE"."OPTIONS" ("OID") ON DELETE CASCADE ENABLE

         )
```

24. Correct Options (ISA Options)
RS:
Correct Options(QID:Integer(Foreign Key to Questions), OID:Integer(Foreign Key to Options), CID:Integer, longdesc:VARCHAR2(500))

```
  CREATE TABLE "ABOKE"."CORRECTOPTIONS"

   (      "QID" NUMBER,

          "OID" NUMBER(*,0),

          "CID" NUMBER(*,0),

          "LONGDESC" VARCHAR2(500),

           CONSTRAINT "PK_CO" PRIMARY KEY ("QID", "OID", "CID")

          CONSTRAINT "FK_CO_QUESTIONS" FOREIGN KEY ("QID")
 REFERENCES                "ABOKE"."QUESTIONS" ("QID") ON DELETE CASCADE
ENABLE,
          CONSTRAINT "FK_CO" FOREIGN KEY ("OID")

           REFERENCES "ABOKE"."OPTIONS" ("OID") ON DELETE CASCADE ENABLE

         )
```

25. Topic_Chapters (Relationship between Topics and Chapters)
RS:

Course_Topics(TID:String (Foreign key to Topics), ISBN:String(Foreign Key to Chapters), chnumber:Number(Foreign Key to Chapters))

DDL Statement:

```
CREATE TABLE "ABOKE"."TOPIC_CHAPTERS"

(      "TID" VARCHAR2(20),

       "ISBN" VARCHAR2(50),

       "CHNUMBER" NUMBER,

        CONSTRAINT "PK_TOPIC_CHAPTERS" PRIMARY KEY ("TID", "ISBN", "CHNUMBER")

        CONSTRAINT "FK_TOPIC_CHAPTERS" FOREIGN KEY ("ISBN", "CHNUMBER")

         REFERENCES "ABOKE"."CHAPTERS" ("ISBN", "CHNUMBER") ENABLE

       )
```

Functional Dependencies and Normal Forms:

All the relations are in BCNF. This is due to the following:

- All the attributes in any given relation are wholly dependent on the primary key.
- There are no transitive dependencies.
- FDs of all the FD sets have 'primary/candidate key' on the left side (for all the relations).

Following are the entities and the set of Fds that are valid and hold over an entity set-

Users :

- Initially, the users schema was:

Users(unityid:string, fname:string, lname:string, password:string, role:string)

- However, this schema was not even in 1NF. This is because a user, identified by 'unityid' can have multiple roles which meant multivalued column 'role'. This clearly violates 1NF.
- To get rid of this, we decomposed 'users' into:

Users(unityid:string, fname:string, lname:string, password:string)
Users_roles(unityid:string, role:string)

After decomposition, both the relations achieve BCNF as the 'primary key' decides all the other attributes:
Users- <unityid> --> <all the attributes of 'Users'>
Users_roles - <unityid, role> --> <attributes of users_roles>

Normal form for Users and User_roles : BCNF

Courses :
Primary Key: cid

Functional Dependencies that exist on the relation: cid --> <cid,cname>
Normal form: BCNF

Course Offerings :
Primary Key: <cid, ctoken>
Functional Dependencies that exist on the relation: <cid, ctoken> --> <All other attributes>
Normal form: BCNF

Textbooks :
Primary Key: <isbn>
Functional Dependencies that exist on the relation: <isbn> --> <All other attributes>
Normal form: BCNF

Chapters of Textbooks (Weak entity) :
Primary Key:<isbn, chnumber>
Functional Dependencies that exist on the relation: <isbn, chnumber> --> <All other attributes>
Normal form: BCNF

Sections of Chapters (Weak entity) :
Primary Key: <isbn, chnumber, secid>
Functional Dependencies that exist on the relation: <isbn, chnumber, secid> --> <All other attributes>
Normal form: BCNF

Subsections of Sections (Weak entity) :
Primary Key: <isbn, chnumber, secid, ssecid>
Functional Dependencies that exist on the relation: <isbn, chnumber, secid, ssecid> --> <All other attributes>
Normal form: BCNF

Topics :
Primary Key: <tid>
Functional Dependencies that exist on the relation: <tid> --> <All other attributes>
Normal form: BCNF

Questions :
Primary Key: <qid>
Functional Dependencies that exist on the relation: <qid> --> <All other attributes>
Normal form: BCNF

Options for Questions (Weak entity) :
Primary Key: <qid,oid>
Functional Dependencies that exist on the relation: <qid,oid> --> <All other attributes>
Normal form: BCNF

Incorrect Options (ISA hierarchy, a child of 'Options') :
Primary Key:
Functional Dependencies that exist on the relation: <PK> --> <All other attributes>
Normal form: BCNF

Correct Options (ISA hierarchy, a child of 'Options') :
Primary Key: <qid,oid,cid>
Functional Dependencies that exist on the relation: <qid,oid,cid> --> <All other attributes>
Normal form: BCNF

Homework :
Primary Key: <hwid>
Functional Dependencies that exist on the relation: <hwid> --> <All other attributes>
Normal form: BCNF

Submission :
Primary Key: <submissionid>
Functional Dependencies that exist on the relation: <submissionid> --> <All other attributes>
Normal form: BCNF

Teaching Assistants (ISA hierarchy, a child of 'Users'):
Primary Key: <unityid>
Functional Dependencies that exist on the relation: <unityid> --> <All other attributes>
Normal form: BCNF

Instructors (ISA hierarchy, a child of 'Users') :
Primary Key: <unityid>
Functional Dependencies that exist on the relation: <unityid> --> <All other attributes>
Normal form: BCNF

Students (ISA hierarchy, a child of 'Users') :
Primary Key: <unityid>
Functional Dependencies that exist on the relation: <unityid> --> <All other attributes>
Normal form: BCNF

SQL queries:

- Find students who did not take homework 1.

*SELECT CS.UnityID FROM Course_Students CS WHERE CS.CID = **course** AND CS.CToken = **ctoken** AND CS.UnityID NOT IN (SELECT S.UnityID FROM Submission S WHERE S.HWID = 1)*

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

```
SELECT UnityID, MAX(Score)
FROM (SELECT UnityID, Score FROM Submission
 WHERE TimeStamp IN
(SELECT MIN(TimeStamp)
     FROM Submission
     WHERE Submission.UnityID=UnityID))
GROUP BY UnityID
```

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

```
select temp.ids as ids, temp.hwid as hwid, temp.score as score
from
(
select sb.unityid as ids, sb.hwid as hwid, sb.score as score, rank() over (partition by
sb.unityid,sb.hwid order by sb.timestamp ) attempt_number from submission sb
where sb.hwid in (Select distinct hwid from Submission)
```

) temp
where attempt_number=1 and score= (Select max(s.score) from Submission s
where s.hwid=temp.hwid
group by s.hwid);

- For each student, show total score for each homework and average score across all homeworks.
  SELECT UnityID, HWID, MAX(Score) AS Score FROM Submission GROUP BY UnityID, HWID
  SELECT UnityID, AVG(Score) FROM Submission GROUP BY UnityID

- For each homework, show average number of attempts
  Select hwid,count(distinct submissionid)/count(distinct unityid) as avg_attempts from submission
  group by hwid;

Use case scenarios:

Appendix :
Insert statements to feed in the static data into the tables:

Users:
INSERT INTO USERS VALUES ('tregan','Tom','Regan','tregan');

INSERT INTO USERS VALUES ('jmick','Jenelle','Mick','jmick');

INSERT INTO USERS VALUES ('mfiser','Michal','Fiser','mfiser');

INSERT INTO USERS VALUES ('jander','Joseph','Anderson','jander');

INSERT INTO USERS VALUES ('jHarla','Jitendra','Harlalka','jHarla');

INSERT INTO USERS VALUES ('aneela','Aishwarya','Neelakantan','aneela';

INSERT INTO USERS VALUES ('mjones','Mary','Jones','mjones');

INSERT INTO USERS VALUES ('jmoyer','James','Moyer','jmoyer');

INSERT INTO USERS VALUES ('kogan','Kemafor','Ogan','kogan');

INSERT INTO USERS VALUES ('rchirkova','Rada','Chirkova','rchirkova');

INSERT INTO USERS VALUES ('chealey','Cristopher','Healey','chealey');

Students:
INSERT INTO STUDENTS VALUES ('tregan','Undergraduate');

INSERT INTO STUDENTS VALUES ('jmick','Graduate');

INSERT INTO STUDENTS VALUES ('mfiser','Undergraduate');

INSERT INTO STUDENTS VALUES ('jander','Undergraduate');

INSERT INTO STUDENTS VALUES ('jHarla','Graduate');

INSERT INTO STUDENTS VALUES ('aneela','Graduate');

INSERT INTO STUDENTS VALUES ('mjones','Graduate');

INSERT INTO STUDENTS VALUES ('jmoyer','Graduate');

Instructors:
INSERT INTO INSTRUCTORS VALUES ('kogan');

INSERT INTO INSTRUCTORS VALUES ('rchirkova');

INSERT INTO INSTRUCTORS VALUES ('chealey');

Textbooks:
INSERT INTO TEXTBOOKS VALUES('0072465638','Raghu Ramakrishnan and Johannes Gehrke','Database Management Systems (3rd edition)');

INSERT INTO TEXTBOOKS VALUES('0471605212','Alan L. Tharp','File Organization and Processing');

Chapters:
INSERT INTO CHAPTERS VALUES('0072465638',1,'Introduction to database design');

INSERT INTO CHAPTERS VALUES('0072465638',2,'SQL: Queries, Constraints, Triggers');

INSERT INTO CHAPTERS VALUES('0072465638',3,'Storing data: Disks and Files');

INSERT INTO CHAPTERS VALUES('0471605212',1,'Primary File Organiztions');

INSERT INTO CHAPTERS VALUES('0471605212',2,'Tree Structures');

Sections:
INSERT INTO SECTIONS VALUES('0072465638',1,1,'Database design and ER diagram');

INSERT INTO SECTIONS VALUES('0072465638',1,2,'Additional Features of ER Model');

INSERT INTO SECTIONS VALUES('0072465638',2,1,'Union, Intersect and Except');

INSERT INTO SECTIONS VALUES('0072465638',2,2,'Aggregate Operators');

INSERT INTO SECTIONS VALUES('0072465638',3,1,'The Memory Hierarchy');

INSERT INTO SECTIONS VALUES('0072465638',3,2,'Redundant Arrays of Independent disks');

INSERT INTO SECTIONS VALUES('0471605212',1,1,'Sequential file organizations');

INSERT INTO SECTIONS VALUES('0471605212',1,2,'Direct file organization');

INSERT INTO SECTIONS VALUES('0471605212',2,1,'Binary tree structures');

INSERT INTO SECTIONS VALUES('0471605212',2,2,'Hashing techniques');

Subsections:
INSERT INTO SUBSECTIONS VALUES('0072465638',1,2,1,'Key Constraints');

INSERT INTO SUBSECTIONS VALUES('0072465638',1,2,2,'Participant constraints');

INSERT INTO SUBSECTIONS VALUES('0072465638',2,2,1,'Group by and Having clause');

INSERT INTO SUBSECTIONS VALUES('0072465638',3,1,1,'Magnetic disks');

INSERT INTO SUBSECTIONS VALUES('0072465638',3,2,1,'Data striping');

INSERT INTO SUBSECTIONS VALUES('0072465638',3,2,2,'Redundancy');

INSERT INTO SUBSECTIONS VALUES('0471605212',1,1,1,'Binary search');

INSERT INTO SUBSECTIONS VALUES('0471605212',1,1,2,'Interpolation search');

INSERT INTO SUBSECTIONS VALUES('0471605212',1,2,1,'Hashing functions');

INSERT INTO SUBSECTIONS VALUES('0471605212',2,1,1,'AVL Trees');

INSERT INTO SUBSECTIONS VALUES('0471605212',2,2,1,'Extendible hashing');

Courses:
INSERT INTO COURSES VALUES('CSC440','Database Systems');

INSERT INTO COURSES VALUES('CSC540','Database Systems');

INSERT INTO COURSES VALUES('CSC541','Advanced Data Structures');

Course_Offerings:

```
INSERT INTO COURSE_OFFERINGS VALUES
('CSC440','CSC440FALL14','27-August-2014','12-December-2014','Undergraduate',5,3);

INSERT INTO COURSE_OFFERINGS VALUES
('CSC540','CSC540FALL14','25-August-2014','10-December-2014','Graduate',5,3);

INSERT INTO COURSE_OFFERINGS VALUES
('CSC541','CSC541FALL14','25-August-2014','6-December-2014','Graduate',5,3);
```

Course Instructors:

```
INSERT INTO COURSE_INSTRUCTORS VALUES('rchirkova','CSC440','CSC440FALL14');

INSERT INTO COURSE_INSTRUCTORS VALUES('kogan','CSC540','CSC540FALL14');

INSERT INTO COURSE_INSTRUCTORS VALUES('chealey','CSC541','CSC541FALL14');
```

Course Students:

```
INSERT INTO COURSE_STUDENTS VALUES('tregan','CSC440','CSC440FALL14');

INSERT INTO COURSE_STUDENTS VALUES('mfiser','CSC440','CSC440FALL14');

INSERT INTO COURSE_STUDENTS VALUES('jander','CSC440','CSC440FALL14');

INSERT INTO COURSE_STUDENTS VALUES('aneela','CSC540','CSC540FALL14');

INSERT INTO COURSE_STUDENTS VALUES('mjones','CSC540','CSC540FALL14');

INSERT INTO COURSE_STUDENTS VALUES('jmick','CSC540','CSC540FALL14');

INSERT INTO COURSE_STUDENTS VALUES('aneela','CSC541','CSC541FALL14');

INSERT INTO COURSE_STUDENTS VALUES('mjones','CSC541','CSC541FALL14');

INSERT INTO COURSE_STUDENTS VALUES('jmick','CSC541','CSC541FALL14');
```

Tassistants:

```
INSERT INTO TASSISTANTS VALUES ('aneela','CSC440','CSC440FALL14');

INSERT INTO TASSISTANTS VALUES ('jmick','CSC440','CSC440FALL14');

INSERT INTO TASSISTANTS VALUES ('jHarla','CSC540','CSC540FALL14');
```

INSERT INTO TASSISTANTS VALUES ('jmoyer','CSC541','CSC541FALL14');

Course_Textbooks:
INSERT INTO COURSE_TEXTBOOKS VALUES
('CSC440','CSC440FALL14','0072465638');

INSERT INTO COURSE_TEXTBOOKS VALUES
('CSC540','CSC540FALL14','0072465638');

INSERT INTO COURSE_TEXTBOOKS VALUES
('CSC541','CSC541FALL14','0471605212');

Course Topics:
insert into course_topics values('CSC440','CSC440FALL14',1);

insert into course_topics values('CSC440','CSC440FALL14',2);

insert into course_topics values('CSC540','CSC540FALL14',1);

insert into course_topics values('CSC540','CSC540FALL14',3);

insert into course_topics values('CSC540','CSC540FALL14',4);

insert into course_topics values('CSC540','CSC540FALL14',5);

insert into course_topics values('CSC541','CSC541FALL14',4);

insert into course_topics values('CSC541','CSC541FALL14',5);

Topics:
INSERT INTO TOPICS VALUES(topic_id_seq.nextval,'Introduction to database design');

INSERT INTO TOPICS VALUES(topic_id_seq.nextval,'SQL: Queries, Constraints, Triggers');

INSERT INTO TOPICS VALUES(topic_id_seq.nextval,'Storing data: Disks and Files');

INSERT INTO TOPICS VALUES(topic_id_seq.nextval,'Primary File Organiztions');

INSERT INTO TOPICS VALUES(topic_id_seq.nextval,'Tree Structures');

Questions:
insert into questions values (1,'Question 1?',2,'Hint text Q1',1);

```
insert into questions values (2,'Question 2?',3,'Hint text Q2',1);

insert into questions values (3,'Consider a disk with a 512 bytes,2000, 50, 5, 10msec
                                What is the capacity of a track in bytes?',2,'Hint text Q3',1);

insert into questions values (4,'Consider a disk with a 256 bytes,1000, 100, 10, 20msec
                                What is the capacity of a track in bytes?',2,'Hint text Q3',1);
```

Options:

```
insert into options values (1,'Correct ans 1,','""",1);

insert into options values (2,'Correct ans 2,','""",1);

insert into options values (3,'Incorrect ans 3,','short explanation 3',1);

insert into options values (4,'Incorrect ans 4,','short explanation 4',1);

insert into options values (5,'Incorrect ans 5,','short explanation 5',1);

insert into options values (6,'Incorrect ans 6,','short explanation 6',1);

insert into options values (23,'Correct ans 1v2,','""",4);

insert into options values (24,'Correct ans 2v2,','""",4);

insert into options values (25,'Correct ans 3v2,','""",4);

insert into options values (26,'Incorrect ans 4v2,','short explanation 4',4);

insert into options values (27,'Incorrect ans 5v2,','short explanation 5',4);

insert into options values (28,'Incorrect ans 6v2,','short explanation 6',4);

insert into options values (29,'Incorrect ans 7v2,','short explanation 7',4);

insert into options values (30,'Incorrect ans 8v2,','short explanation 8',4);
```

Incorrect Options:

```
insert into incorrectoptions values(1,3,1);

insert into incorrectoptions values(1,4,2);

insert into incorrectoptions values(1,5,3);

insert into incorrectoptions values(1,6,4);
```

```
insert into incorrectoptions values(2,10,1);

insert into incorrectoptions values(2,11,2);

insert into incorrectoptions values(2,12,3);

insert into incorrectoptions values(2,13,4);

insert into incorrectoptions values(2,14,5);

insert into incorrectoptions values(3,18,1);

insert into incorrectoptions values(3,19,2);

insert into incorrectoptions values(3,20,3);

insert into incorrectoptions values(3,21,4);

insert into incorrectoptions values(3,22,5);

insert into incorrectoptions values(4,26,1);

insert into incorrectoptions values(4,27,2);

insert into incorrectoptions values(4,28,3);

insert into incorrectoptions values(4,29,4);

insert into incorrectoptions values(4,30,5);
```

Correct Options:

```
insert into correctoptions values(1,1,1,'detailed explanation Q1');

insert into correctoptions values(1,2,2,'detailed explanation Q1');

insert into correctoptions values(2,7,1,'detailed explanation Q2');

insert into correctoptions values(2,8,2,'detailed explanation Q2');

insert into correctoptions values(2,9,3,'detailed explanation Q2');

insert into correctoptions values(3,15,1,'detailed explanation Q3');

insert into correctoptions values(3,16,2,'detailed explanation Q3');

insert into correctoptions values(3,17,3,'detailed explanation Q3');
```

```
insert into correctoptions values(4,23,1,'detailed explanation Q3');

insert into correctoptions values(4,24,2,'detailed explanation Q3');

insert into correctoptions values(4,25,3,'detailed explanation Q3');
```