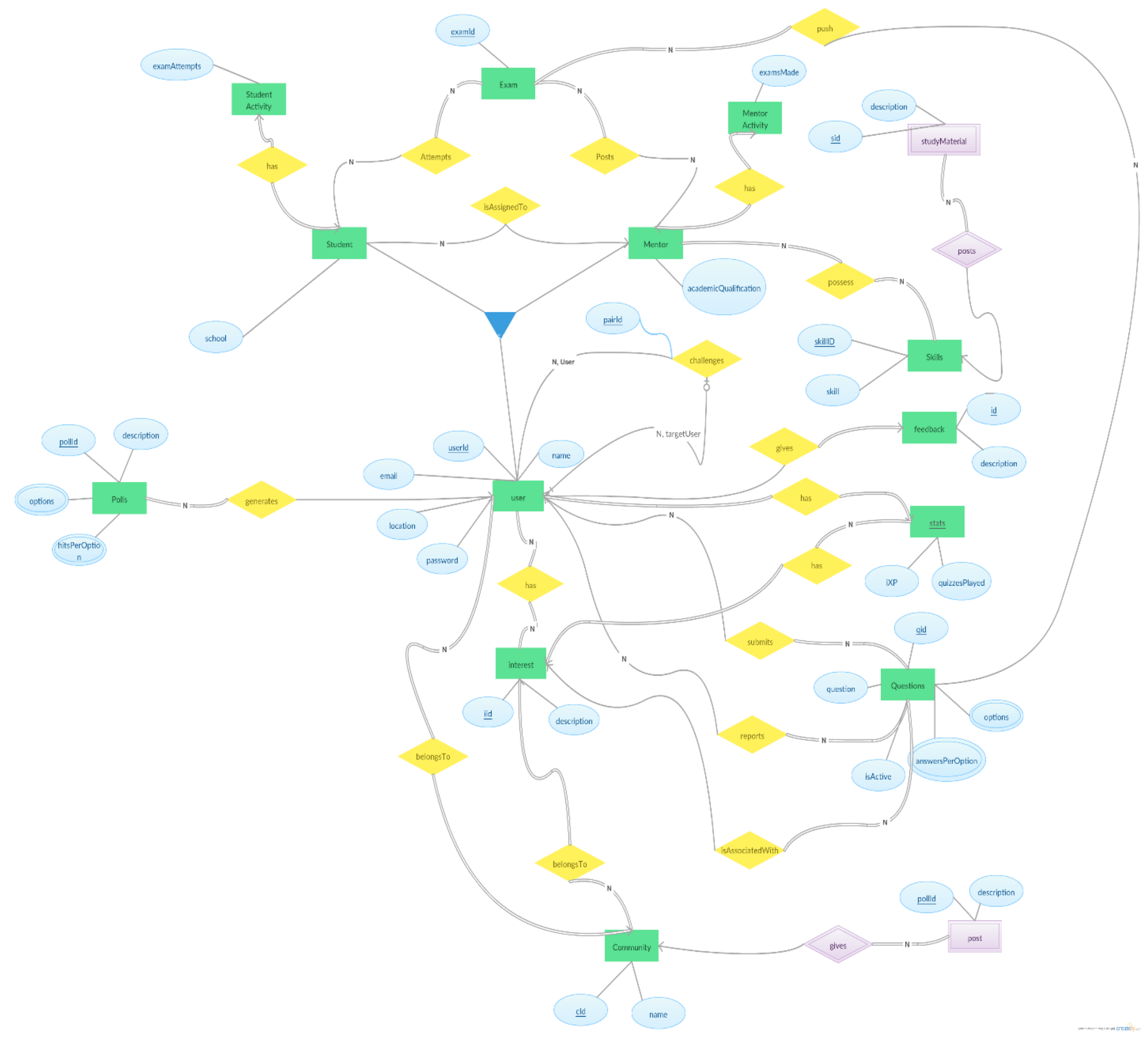
Quizon : Quiz, Challenge, Polls

RAGHAV MITTAL
SURAJ NATH SIDH
RAJDEEP MUKHERJEE

ER Diagram



Here is the Entity Relationship diagram of the Project

Project Description:

Our database consists of 3 main parts: Quizzes, Challenges and Polls.

Users are divided into 2 categories: Students and Mentors. Although each user has a unique userId, the same userId is also used in the Student and the Mentor tables. If the user is a student then his userId will be stored in the student table otherwise his userId will be stored in the mentors table.

Both students and mentors can play quizzes however exams are prepared by mentors and given by students only.

Both students and mentors can submit questions. If a question is made private then that question cannot be used in quizzes, i.e. it can only be used in exams. However if it is made public, it can be used for both quizzes and exams.

Mentors possess skills and can submit study material. Mentors can also make exams and students will give those exams.

Quizzes and Challenges:

Users can take part and play Quizzes. For each round of a quiz, 10 random questions based on a particular interest are pulled from the pool of questions stored in the Questions table. The Questions table consists of questions submitted by both the students and mentors.

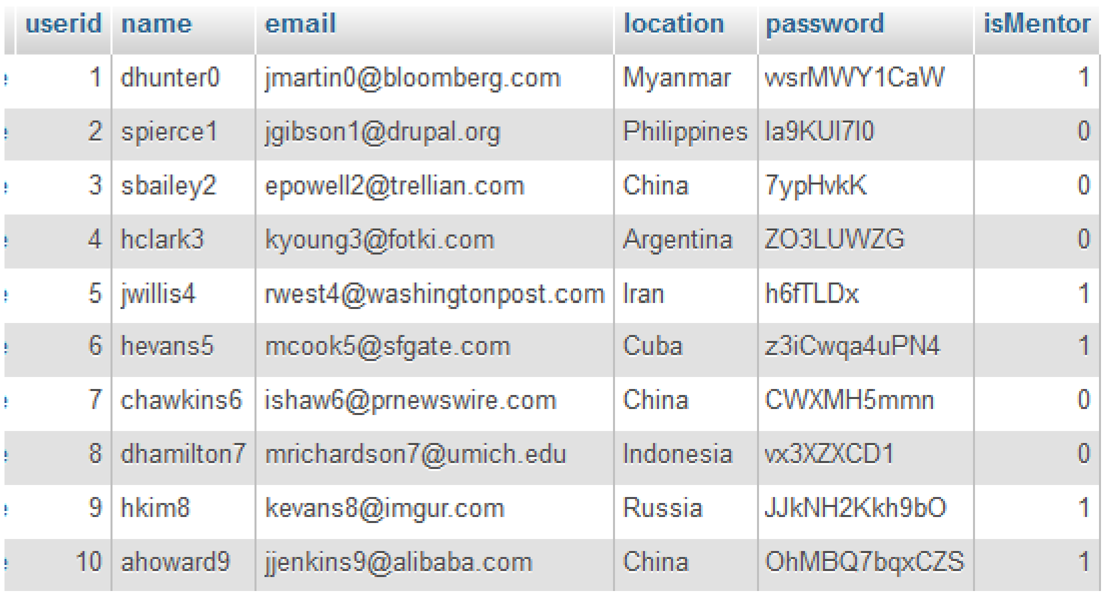
Users can also play against each other in challenges. In this case the ixp of the user who wins will increase.

Polls:

User can generate polls which contains poll description and multiple poll options. After creation of poll it automatically get shared in all the community in which user belongs and other users can vote on poll and we will record hits per option of perticular poll which we use to disply percenteage per hit or plain no of hits per option on that poll.

Following are the schemas derived directly from the ER diagram along with the functional dependencies:

1. ***User { <u>uid</u>, name, email, location, password, isMentor}***



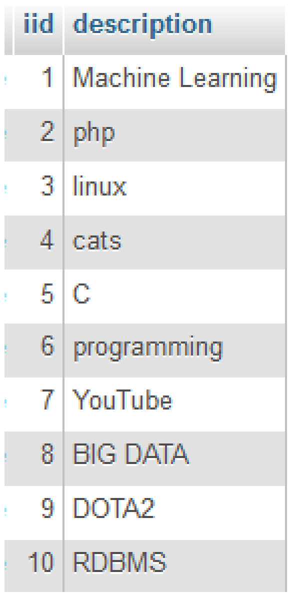
userid	name	email	location	password	isMentor
1	dhunter0	jmartin0@bloomberg.com	Myanmar	wsrMWY1CaW	1
2	spierce1	jgibson1@drupal.org	Philippines	la9KUI7I0	0
3	sbailey2	epowell2@trellian.com	China	7ypHvkK	0
4	hclark3	kyoung3@fotki.com	Argentina	ZO3LUWZG	0
5	jwillis4	rwest4@washingtonpost.com	Iran	h6fTLDx	1
6	hevans5	mcook5@sfgate.com	Cuba	z3iCwqa4uPN4	1
7	chawkins6	ishaw6@prnewswire.com	China	CWXMH5mmn	0
8	dhamilton7	mrichardson7@umich.edu	Indonesia	vx3XZXCD1	0
9	hkim8	kevans8@imgur.com	Russia	JJkNH2Kkh9bO	1
10	ahoward9	jjenkins9@alibaba.com	China	OhMBQ7bqxCZS	1

The functional dependencies are prevalent are:

{<u>uid</u>} -> {name, email, location, password, isMentor}

Since there are no more functional dependencies that are present here, we do not need to decompose this relation further into any more tables as these are already in BCNF.

2. ***Interest { <u>iid</u>, description}***

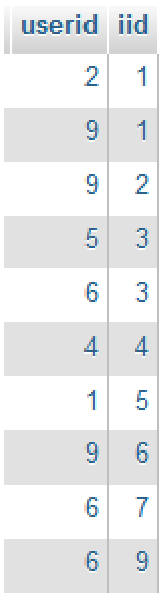


iid	description
1	Machine Learning
2	php
3	linux
4	cats
5	C
6	programming
7	YouTube
8	BIG DATA
9	DOTA2
10	RDBMS

{iid}-> {description}

The above functional dependencies suggest that the table is already in BCNF and hence we do not need to decompose this further.

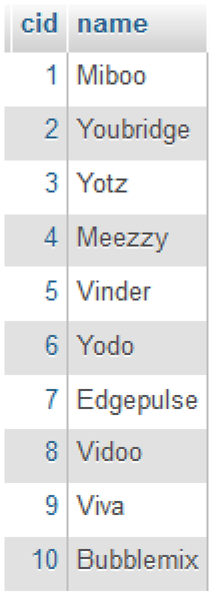
3. userInterest { uid, iid}



userid	iid
2	1
9	1
9	2
5	3
6	3
4	4
1	5
9	6
6	7
6	9

The above relation is only a mapping relation and there are no notable fds in this relation. Both the uid and iid values combine to form the key.

4. Community { cid, name}

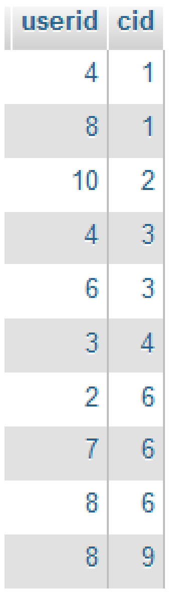


cid	name
1	Miboo
2	Youbridge
3	Yotz
4	Meezzy
5	Vinder
6	Yodo
7	Edgepulse
8	Vidoo
9	Viva
10	Bubblemix

{cid}_>{name}

The above fd suggest that the relation is already in BCNF

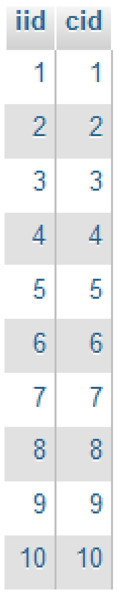
5. userCommunity { uid, cid}



userid	cid
4	1
8	1
10	2
4	3
6	3
3	4
2	6
7	6
8	6
8	9

No fd can be found here. Both uid and cid combined is the key to the relation.

6. interestCommunity { iid, cid}

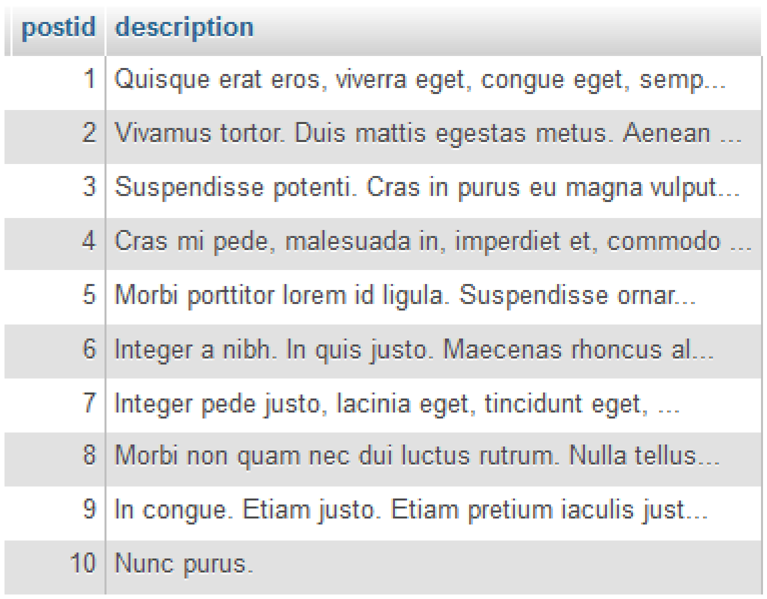


iid	cid
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

{iid}->{cid}

The above relation is already in BCNF

7. ***post { <u>postId</u>, description}***

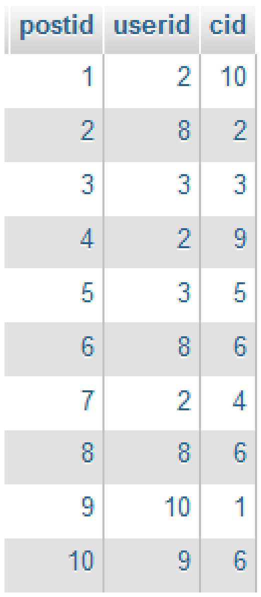


postid	description
1	Quisque erat eros, viverra eget, congue eget, semp...
2	Vivamus tortor. Duis mattis egestas metus. Aenean ...
3	Suspendisse potenti. Cras in purus eu magna vulput...
4	Cras mi pede, malesuada in, imperdiet et, commodo ...
5	Morbi porttitor lorem id ligula. Suspendisse ornar...
6	Integer a nibh. In quis justo. Maecenas rhoncus al...
7	Integer pede justo, lacinia eget, tincidunt eget, ...
8	Morbi non quam nec dui luctus rutrum. Nulla tellus...
9	In congue. Etiam justo. Etiam pretium iaculis just...
10	Nunc purus.

{<u>postId</u>}->{description}

The above relation is already In BCNF

8. ***compost { cid, <u>postId</u>, uid}***

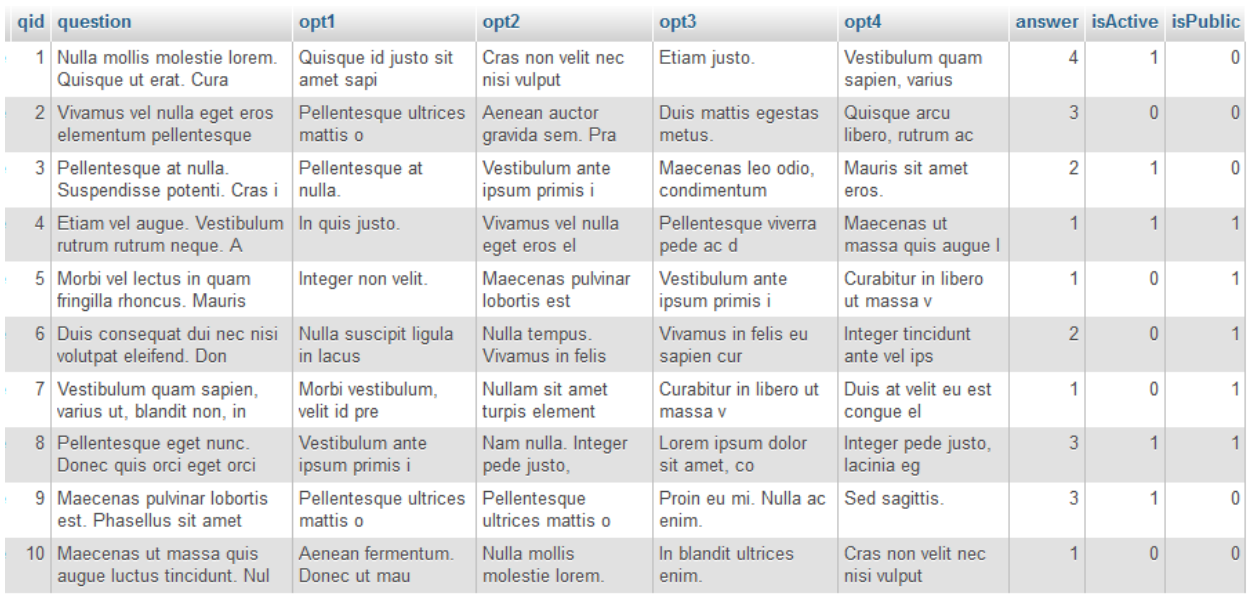


postid	userid	cid
1	2	10
2	8	2
3	3	3
4	2	9
5	3	5
6	8	6
7	2	4
8	8	6
9	10	1
10	9	6

{<u>postId</u>}->{cid, uid}

The above relation is already in BCNF

9. Questions{ <u>qid</u>, question, options(array), answers(array), isActive}



qid	question	opt1	opt2	opt3	opt4	answer	isActive	isPublic
1	Nulla mollis molestie lorem. Quisque ut erat. Cura	Quisque id justo sit amet sapi	Cras non velit nec nisi vulput	Etiam justo.	Vestibulum quam sapien, varius	4	1	0
2	Vivamus vel nulla eget eros elementum pellentesque	Pellentesque ultrices mattis o	Aenean auctor gravida sem. Pra	Duis mattis egestas metus.	Quisque arcu libero, rutrum ac	3	0	0
3	Pellentesque at nulla. Suspendisse potenti. Cras i	Pellentesque at nulla.	Vestibulum ante ipsum primis i	Maecenas leo odio, condimentum	Mauris sit amet eros.	2	1	0
4	Etiam vel augue. Vestibulum rutrum rutrum neque. A	In quis justo.	Vivamus vel nulla eget eros el	Pellentesque viverra pede ac d	Maecenas ut massa quis augue l	1	1	1
5	Morbi vel lectus in quam fringilla rhoncus. Mauris	Integer non velit.	Maecenas pulvinar lobortis est	Vestibulum ante ipsum primis i	Curabitur in libero ut massa v	1	0	1
6	Duis consequat dui nec nisi volutpat eleifend. Don	Nulla suscipit ligula in lacus	Nulla tempus. Vivamus in felis	Vivamus in felis eu sapien cur	Integer tincidunt ante vel ips	2	0	1
7	Vestibulum quam sapien, varius ut, blandit non, in	Morbi vestibulum, velit id pre	Nullam sit amet turpis element	Curabitur in libero ut massa v	Duis at velit eu est congue el	1	0	1
8	Pellentesque eget nunc. Donec quis orci eget orci	Vestibulum ante ipsum primis i	Nam nulla. Integer pede justo,	Lorem ipsum dolor sit amet, co	Integer pede justo, lacinia eg	3	1	1
9	Maecenas pulvinar lobortis est. Phasellus sit amet	Pellentesque ultrices mattis o	Pellentesque ultrices mattis o	Proin eu mi. Nulla ac enim.	Sed sagittis.	3	1	0
10	Maecenas ut massa quis augue luctus tincidunt. Nul	Aenean fermentum. Donec ut mau	Nulla mollis molestie lorem.	In blandit ultrices enim.	Cras non velit nec nisi vulput	1	0	0

The above relation violates 1NF because of the presence of multi valued attributes. Instead of decomposing the relation we can decompose the attribute.

Questions { <u>qid, </u>question, op1, op2, op3, op4, answer, isActive, isPublic}

isPublic is a new attribute here. isPublic defines whether the question can be in a quiz or it can be in an exam. If isPublic is 1 then the question can be in quiz and also in exam. However if isPublic is 0 then the question can only be in exams and not in quiz.

10. interestQuestion { <u>iid, qid</u>}

The above relation has no notable functional dependencies. Both iid and qi is the key.

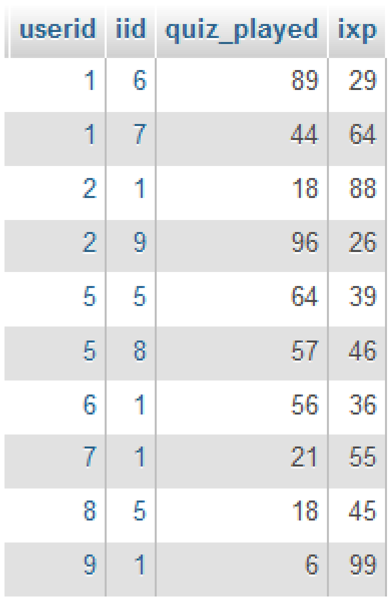
11. userReports { <u>qid, uid</u>}

The above relation has no notable functional dependencies. Both qid and uid combined is the key.

12. userSubmits { <u>qid, uid</u>}

The above relation has no notable functional dependencies. Both qid and uid combined is the key. Although the schema of userSubmits and userReports is the same, the users are different. In case of userSubmits the user is a mentor and in case of userReports, it can be any user.

13. userStats { <u>uid, iid</u>, ixp, quizzesPlayed}

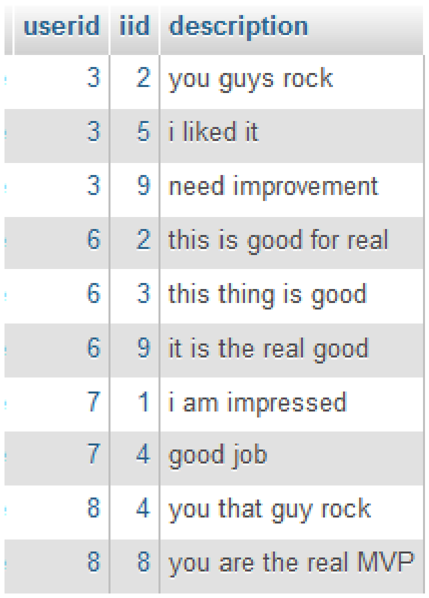


userid	iid	quiz_played	ixp
1	6	89	29
1	7	44	64
2	1	18	88
2	9	96	26
5	5	64	39
5	8	57	46
6	1	56	36
7	1	21	55
8	5	18	45
9	1	6	99

{uid, iid}->{ixp, quizzesPlayed}

This relation is already in BCNF and hence requires no decomposition.

14. ***Feedback { id, description, uid}***



userid	iid	description
3	2	you guys rock
3	5	i liked it
3	9	need improvement
6	2	this is good for real
6	3	this thing is good
6	9	it is the real good
7	1	i am impressed
7	4	good job
8	4	you that guy rock
8	8	you are the real MVP

{Id}->{description, uid}

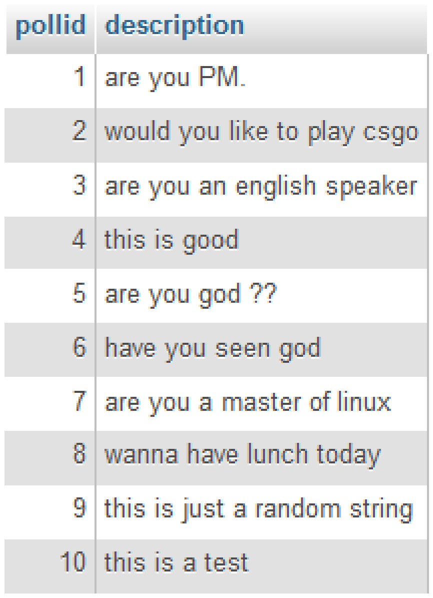
This relation is already in BCNF and hence requires no decomposition.

15. ***Polls { pollId, description, options(array), hitsPerOption(array)}***

This relation violates 1NF as the relation has multi valued attributes. Hence we are required to decompose the relation.

Poll { pollId, description}

pollOptions { pollId, options, hits}

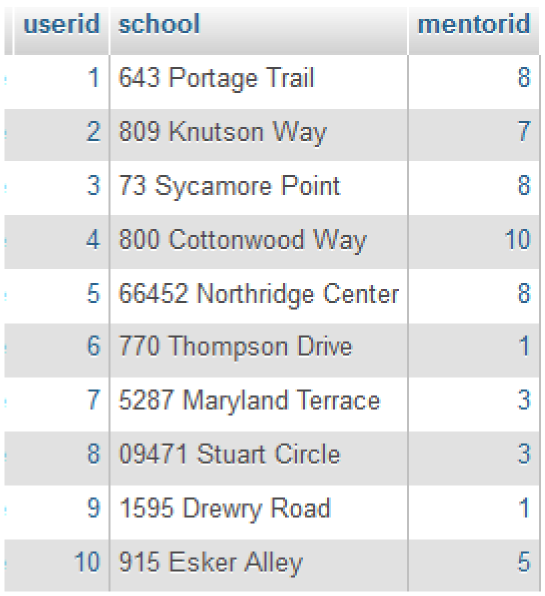


pollid	description
1	are you PM.
2	would you like to play csgo
3	are you an english speaker
4	this is good
5	are you god ??
6	have you seen god
7	are you a master of linux
8	wanna have lunch today
9	this is just a random string
10	this is a test

{pollId, options}->{hits}

Now the relation is already in BCNF

16. ***userStudent { uid, school, mentorId}***

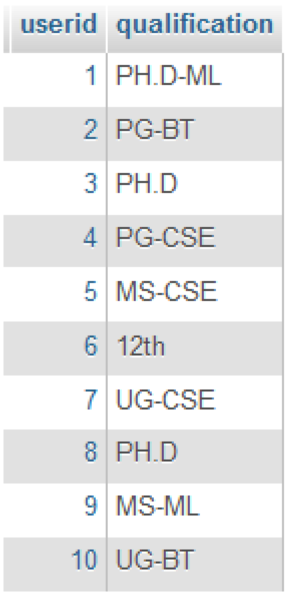


userid	school	mentorid
1	643 Portage Trail	8
2	809 Knutson Way	7
3	73 Sycamore Point	8
4	800 Cottonwood Way	10
5	66452 Northridge Center	8
6	770 Thompson Drive	1
7	5287 Maryland Terrace	3
8	09471 Stuart Circle	3
9	1595 Drewry Road	1
10	915 Esker Alley	5

{uid}->{ school, mentorId}

The above relation is already in BCNF and does not require any decomposition. This relation will store the user id of the students, mind it that, it is the same user id that is generated from the user table. If the user is a student according to that table then his/her user id is stored in this table, otherwise his/her user id is stored in the userMentor table.

17. ***userMentor { uid, academicQualification}***

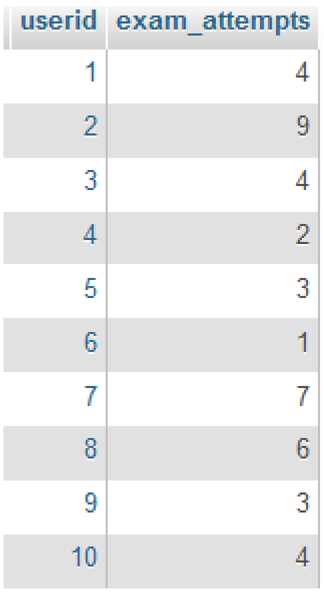


userid	qualification
1	PH.D-ML
2	PG-BT
3	PH.D
4	PG-CSE
5	MS-CSE
6	12th
7	UG-CSE
8	PH.D
9	MS-ML
10	UG-BT

{uid}->{academic}

The above relation is already in BCNF.

18. ***stuActivity { uid, examAttempts}***

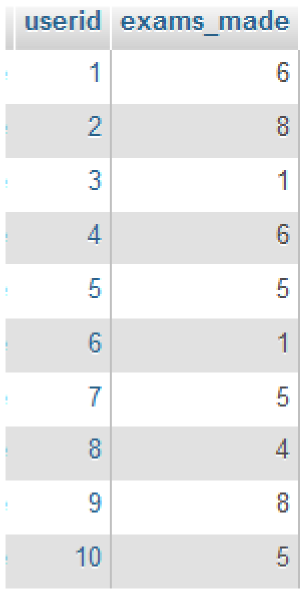


userid	exam_attempts
1	4
2	9
3	4
4	2
5	3
6	1
7	7
8	6
9	3
10	4

{<u>uid</u>}->{examAttempts}

The above relation is already in BCNF.

19. ***mentActivity { <u>uid</u>, examsMade}***

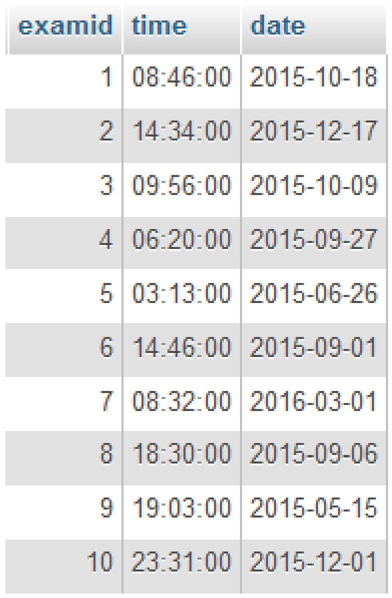


userid	exams_made
1	6
2	8
3	1
4	6
5	5
6	1
7	5
8	4
9	8
10	5

{<u>uid</u>}->{examsMade}

The above relation is already in BCNF

20. ***Exams { <u>examId</u>, timeStamp}***

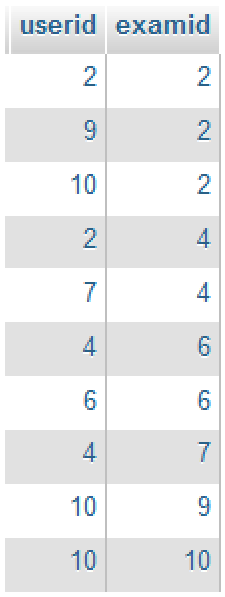


examid	time	date
1	08:46:00	2015-10-18
2	14:34:00	2015-12-17
3	09:56:00	2015-10-09
4	06:20:00	2015-09-27
5	03:13:00	2015-06-26
6	14:46:00	2015-09-01
7	08:32:00	2016-03-01
8	18:30:00	2015-09-06
9	19:03:00	2015-05-15
10	23:31:00	2015-12-01

{examId}->{timeStamp}

The above relation is already in BCNF

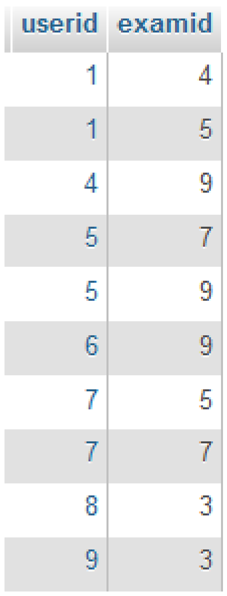
21. stuExams { uid, examId}



userid	examid
2	2
9	2
10	2
2	4
7	4
4	6
6	6
4	7
10	9
10	10

The above relation does not have any functional dependencies and hence no decomposition is required.

22. mentExams { uid, examId}



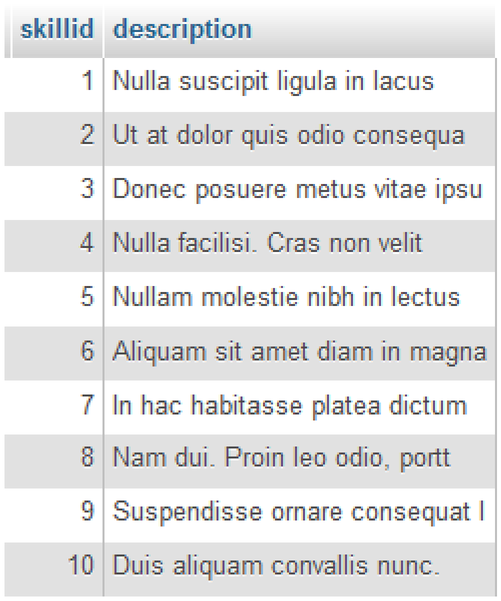
userid	examid
1	4
1	5
4	9
5	7
5	9
6	9
7	5
7	7
8	3
9	3

The above relation is also a mapping relation and hence does not have any functional dependencies.

23. examQuestions { <u>examId, qid</u>}

The above relation is also a mapping relation and hence does not have any functional dependencies.

24. Skills { <u>skillId</u>, skill}

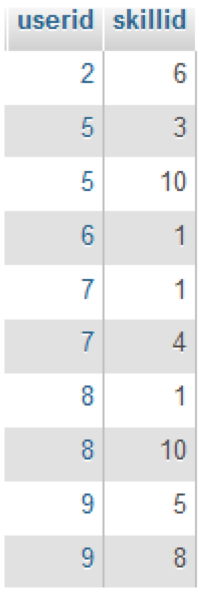


skillid	description
1	Nulla suscipit ligula in lacus
2	Ut at dolor quis odio consequa
3	Donec posuere metus vitae ipsu
4	Nulla facilisi. Cras non velit
5	Nullam molestie nibh in lectus
6	Aliquam sit amet diam in magna
7	In hac habitasse platea dictum
8	Nam dui. Proin leo odio, portt
9	Suspendisse ornare consequat l
10	Duis aliquam convallis nunc.

{<u>skillId</u>}->{skill}

The functional dependency above is the only functional dependency found and the table is already in BCNF and does not require any further.

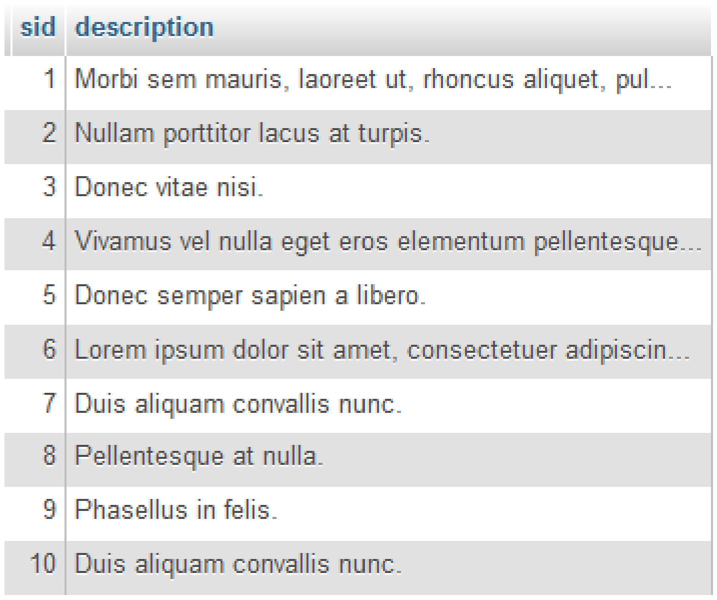
25. mentSkill { <u>uid, skillId</u>}



userid	skillid
2	6
5	3
5	10
6	1
7	1
7	4
8	1
8	10
9	5
9	8

This is another mapping relation and no functional dependency is found here.

26. StudyMaterial { sid, description}

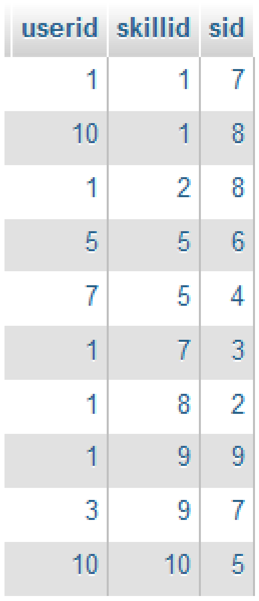


sid	description
1	Morbi sem mauris, laoreet ut, rhoncus aliquet, pul...
2	Nullam porttitor lacus at turpis.
3	Donec vitae nisi.
4	Vivamus vel nulla eget eros elementum pellentesque...
5	Donec semper sapien a libero.
6	Lorem ipsum dolor sit amet, consectetuer adipiscin...
7	Duis aliquam convallis nunc.
8	Pellentesque at nulla.
9	Phasellus in felis.
10	Duis aliquam convallis nunc.

{sid}->{description}

The above relation is already in BCNF

27. mentStudyMaterial { <u>sid</u>, uid, skillId}

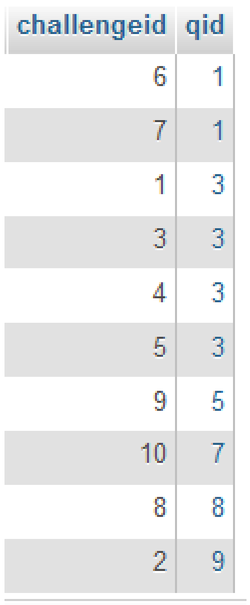


userid	skillid	sid
1	1	7
10	1	8
1	2	8
5	5	6
7	5	4
1	7	3
1	8	2
1	9	9
3	9	7
10	10	5

{<u>sid</u>}->{ uid, skillId}

The above relation is already in BCNF and hence does not require further normalization.

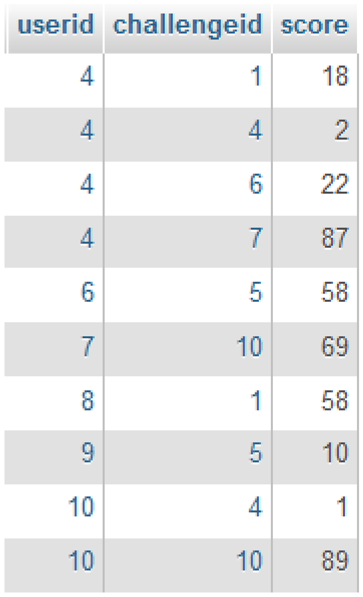
28. Challenges { challengeId, qid}



challengeid	qid
6	1
7	1
1	3
3	3
4	3
5	3
9	5
10	7
8	8
2	9

The above relation is a mapping relation and hence there are no functional dependencies found here.

29. userChallenges { uid, challengeId, score}



userid	challengeid	score
4	1	18
4	4	2
4	6	22
4	7	87
6	5	58
7	10	69
8	1	58
9	5	10
10	4	1
10	10	89

{uid, challengeId}->{score}

This relation is already in BCNF and does not require any further normalization.

Finally, we landed with the following set of relational schemas:

1. **user {userid(key), name, email, location, Password, isStudent, isMentor }**
2. **interestes {iid(key), description }**
3. **userInterests {userid(key), iid(key)}**
4. **Community {cid(key), name}**
5. **Posts {postid(key), description}**
6. **questions {Qid(key), questions, op1, op2, op3, op4, Answer, isActive, isPublic }**
7. **user_com {uid(key), cid(key)}**
8. **com_post {postid(key), uid, cid}**
9. **Q_int_user {Qid(key), uid, iid}**
10. **stats {uid(key), iid(key), QuizzesPlayes, iXP}**
11. **feedback {uid(key), iid(key), Description}**
12. **student { uid(key), school, mentorId }**
13. **Mentor {uid(key), A_Qalificaton }**
14. **student_activity {userid(key), exam_Attempts }**
15. **Mentor_activity {userid(key), exams_made}**
16. **Exam {examid(key), timestamp}**
17. **Exam_Attempts {userid(key), examid(key)}**
18. **Exam_posts {userid(key), examid(key)}**
19. **Question_push {examid(key), qid(key)}**
20. **Skill {skillid(key), skill}**
21. **Mentor_skill {userid(key), Skillid(key)}**
22. **Study_Matarial {sid(key), Description}**
23. **Skill_Studymatarial {uid(key),skillid(key), sid(key)}**
24. **poll {pollid(key), description}**
25. **poll_options {pollid(key), options(key), hits}**

26. **`poll_generation {pollid(key), userid}`**

27. **`challenges_score {userid(key), challengeid(key), score}`**

28. **`challenges {challengeid(key), qid(key)}`**

SQL CODE:

```
-- create database use that databse before running this
-- --------------------------------------------------------
-- Table structure for table `user`
CREATE TABLE `user` (
  `userid` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(30) NOT NULL,
  `email` varchar(40) NOT NULL,
  `location` varchar(40) NOT NULL,
  `password` varchar(20) NOT NULL,
  `isMentor` BOOLEAN NOT NULL,
  PRIMARY KEY (`userid`)
);
-----------------------------------------------------------
-- Table structure for table `interests`
CREATE TABLE `interests` (
  `iid` int(11) NOT NULL AUTO_INCREMENT,
  `description` varchar(50) NOT NULL,
  PRIMARY KEY (`iid`)
);
-- --------------------------------------------------------
-- Table structure for table `userInterests`
CREATE TABLE `userInterests` (
  `userid` int(11) NOT NULL,
  `iid` int(11) NOT NULL,
  PRIMARY KEY (`userid`,`iid`),
  FOREIGN KEY (`userid`) REFERENCES `user` (`userid`) on delete cascade on
update no action,
  FOREIGN KEY (`iid`) REFERENCES `interests` (`iid`) on delete no action on
update cascade
);
```

```
----------------------------------------------------
-- Table structure for table `community`
CREATE TABLE `community` (
  `cid` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(30) NOT NULL,
  PRIMARY KEY (`cid`),
  FOREIGN KEY (`cid`) REFERENCES `interests` (`iid`) on delete cascade on
update no action
);
-- --------------------------------------------------------
-- Table structure for table `posts`
CREATE TABLE `posts` (
  `postid` int(11) NOT NULL AUTO_INCREMENT,
  `description` text NOT NULL,
  PRIMARY KEY (`postid`)
);
-- --------------------------------------------------------
-- Table structure for table `questions`
CREATE TABLE `questions` (
  `qid` int(11) NOT NULL AUTO_INCREMENT,
  `question` varchar(50) NOT NULL,
  `opt1` varchar(30) NOT NULL,
  `opt2` varchar(30) NOT NULL,
  `opt3` varchar(30) NOT NULL,
  `opt4` varchar(30) NOT NULL,
  `answer` INT(4) NOT NULL,
  `isActive` BOOLEAN NOT NULL DEFAULT TRUE,
  `isPublic` BOOLEAN NOT NULL DEFAULT TRUE,
  PRIMARY KEY (`qid`)
);
-- --------------------------------------------------------
-- Table structure for table `com_post`
CREATE TABLE `com_post` (
  `postid` int(11) NOT NULL,
  `userid` int(11) NOT NULL,
  `cid` int(11) NOT NULL,
  PRIMARY KEY (`postid`),
  FOREIGN KEY (`userid`) REFERENCES `user` (`userid`) on delete no action on
update no action,
  FOREIGN KEY (`cid`) REFERENCES `community` (`cid`) on delete no action on
update no action,
  FOREIGN KEY (`postid`) REFERENCES `posts` (`postid`) on delete cascade on
update no action
);
-- --------------------------------------------------------
-- Table structure for table `q_int_user`
CREATE TABLE `q_int_user` (
  `qid` int(11) NOT NULL,
  `userid` int(11) NOT NULL,
  `iid` int(11) NOT NULL,
  PRIMARY KEY (`qid`),
  FOREIGN KEY (`iid`) REFERENCES `interests` (`iid`) on delete no action on
update no action,
  FOREIGN KEY (`qid`) REFERENCES `questions` (`qid`) on delete cascade on
update no action,
  FOREIGN KEY (`userid`) REFERENCES `user` (`userid`) on delete cascade on
update no action
);
-- --------------------------------------------------------
-- Table structure for table `user_com`
CREATE TABLE `user_com` (
  `userid` int(11) NOT NULL,
  `cid` int(11) NOT NULL,
  PRIMARY KEY (`userid`, `cid`),
  FOREIGN KEY (`userid`) REFERENCES `user` (`userid`) on delete cascade on
update no action,
  FOREIGN KEY (`cid`) REFERENCES `community` (`cid`) on delete cascade on
update no action
);
```

```
- --------------------------------------------------------
-- Table structure for table `stats`
create table `stats`(
    `userid` int NOT NULL,
    `iid` int NOT NULL,
    `quiz_played` int NOT NULL,
    `ixp` int NOT NULL,
    PRIMARY KEY (`userid`,`iid`),
      foreign key (`userid`) references `user` (`userid`) on delete cascade
on update no action,
      foreign key (`iid`) references `interests` (`iid`) on delete cascade on
update no action
);
-- --------------------------------------------------------
-- Table structure for table `feedback`
create table `feedback`(
      `userid` int NOT NULL,
      `iid` int NOT NULL,
      `description` text not null,
      primary key (`userid`,`iid`),
      foreign key (`userid`) references `user` (`userid`) on delete cascade
on update no action,
      foreign key (`iid`) references `interests` (`iid`) on delete cascade on
update no action
);
- --------------------------------------------------------
-- Table structure for table `mentors`
create table `mentors`(
      `userid` int not null,
      `qualification` text,
      primary key (`userid`),
 foreign key (`userid`) references `user` (`userid`) on delete cascade on
update no action
);
```

```
-- ---------------------------------------------------------
-- Table structure for table `student`
create table `student`(
      `userid` int not null,
      `school` varchar (40),
      `mentorid` int not null,
      primary key (`userid`),
      foreign key (`userid`) references `user` (`userid`) on delete cascade
on update no action,
      foreign key (`mentorid`) references `mentors` (`userid`) on delete no
action on update cascade
);
---------------------------------------------------------
-- Table structure for table `stats`
create table `student_activity` (
`userid` int not null,
`exam_attempts` int not null ,
primary key (`userid`),
foreign key (`userid`) references `student`(`userid`) on delete cascade on
update no action
);
-- ---------------------------------------------------------
-- Table structure for table `mentor_activity`
create table `mentor_activity` (
`userid` int not null,
`exams_made` int not null,
primary key (`userid`),
foreign key (`userid`) references `mentors` (`userid`) on delete cascade on
update no action
);



-- ---------------------------------------------------------
-- Table structure for table `exam`
create table `exam` (
`examid` int not null auto_increment,
`time` time,
`date` date,
primary key (`examid`)
);
-- --------------------------------------------------------
-- Table structure for table `exam_posts`
create table `exam_posts` (
      `userid` int not null,
      `examid` int not null,
      primary key (`userid`, `examid`),
      foreign key (`userid`) references `mentors` (`userid`) on delete
cascade on update no action
);
-- --------------------------------------------------------
-- Table structure for table `exam_attempts`
create table `exam_attempts` (
      `userid` int not null,
      `examid` int not null,
      primary key (`userid`, `examid`),
      foreign key (`userid`) references `student` (`userid`) on delete
cascade on update no action,
    foreign key (`examid`) references `exam` (`examid`) on delete no action
on update no action
);
-- --------------------------------------------------------
-- Table structure for table `question_push`
create table `question_push` (
      `examid` int not null,
      `qid` int not null,
      primary key (`qid`, `examid`)
);


- --------------------------------------------------------
```

```
-- Table structure for table `skill`
create table `skill` (
      `skillid` int not null auto_increment,
`description` varchar(30),
primary key (`skillid`)
);
- -------------------------------------------------------
-- Table structure for table `mentor_skill`
create table `mentor_skill` (
      `userid` int not null,
      `skillid` int not null,
      primary key(`userid`, `skillid`),
      foreign key (`userid`) references `mentors` (`userid`) on delete
cascade on update no action
);
-- -------------------------------------------------------
-- Table structure for table `study_material`
create table `study_material` (
      `sid` int not null auto_increment,
      `description` text not null,
      primary key(`sid`)
);
- -------------------------------------------------------
-- Table structure for table `skill_studymaterial`
create table `skill_studymaterial`(
      `userid` int not null,
      `skillid` int not null,
      `sid` int not null,
      primary key (`userid`, `skillid`, `sid`),
      foreign key (`userid`) references `mentors`(`userid`) on delete cascade
on update no action,
    foreign key (`skillid`) references `skill`(`skillid`) on delete no action
on update no action,
    foreign key (`sid`) references `study_material`(`sid`) on delete no
action on update no action);
-- -------------------------------------------------------
-- Table structure for table `poll`
create table `poll` (
      `pollid` int not null auto_increment,
      `description` text not null,
      primary key (`pollid`)
);
-- --------------------------------------------------------
-- Table structure for table `poll_options`
create table `poll_options` (
      `pollid` int not null,
      `options` VARCHAR(100) NOT NULL,
      `hits` int default 0,
      primary key(`pollid`, `options`),
      foreign key(`pollid`) references `poll` (`pollid`) on delete cascade on
update cascade
);
-- --------------------------------------------------------
-- Table structure for table `spoll_generation`
create table `poll_generation` (
      `pollid` int not null,
      `userid` int not null,
      primary key (`pollid`),
      foreign key (`pollid`) references `poll` (`pollid`) on delete cascade
on update no action,
      foreign key (`userid`) references `student` (`userid`) on delete
cascade on update no action
);
-- --------------------------------------------------------
-- Table structure for table `challenges`
create table `challenges`(
      `challengeid` int not null auto_increment,
      `qid` int not null,
      primary key (`challengeid`,`qid`),
      foreign key(`qid`) references `questions` (`qid`) on delete cascade on
update no action);
-- --------------------------------------------------------
```

```sql
-- Table structure for table `challenges_score`

create table `challenges_score` (

      `userid` int not null,

      `challengeid` int not null,

      `score` int default 0,

      primary key (`userid`,`challengeid`),

      foreign key(`userid`) references `student`(`userid`) on delete cascade
on update no action,

      foreign key(`challengeid`) references `challenges`(`challengeid`) on
delete cascade on update no action

);
```